UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA
ESCUELA TÉCNICA SUPERIOR DE
INGENIEROS DE TELECOMUNICACIÓN

EUROPEAN DOCTORATE

Real-Time and Low-Cost Super-Resolution Algorithms onto Hybrid Video Encoders

Author: D. GUSTAVO IVÁN MARRERO CALLICÓ
Director: DR. D. ANTONIO NÚÑEZ ORDÓÑEZ
Co-Director: DR. D. RAFAEL PESET LLOPIS

Las Palmas de Gran Canaria, July 2003

UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA
DOCTORADO EN INGENIERÍA DE TELECOMUNICACIÓN
DOCTORADO EUROPEO

DEPARTAMENTO DE INGENIERÍA ELECTRÓNICA Y AUTOMÁTICA
PROGRAMA DE INGENIERÍA ELECTRÓNICA AVANZADA

Mejora de la Calidad en Imagen Estática y Vídeo basada en Súper-Resolución con Prestaciones de Tiempo Real y Bajo Coste mediante Codificación Híbrida

TESIS DOCTORAL PRESENTADA POR:
GUSTAVO IVÁN MARRERO CALLICÓ

DIRIGIDA POR:
DR. D. ANTONIO NÚÑEZ ORDÓÑEZ

CO-DIRIGIDA POR:
DR. D. RAFAEL PESET LLOPIS

El Director, El Co-Director El Doctorando,

Las Palmas de Gran Canaria, Julio de 2003

A mi esposa Emi y a mi hijo Julio.
Ellos han sido la fuente a la que he ido a beber
cuando he sentido que mis fuerzas flaqueaban.

A Emi con todo mi amor

Si pudiera describir la belleza de tus ojos
Y enumerar con números nuevos todas tus gracias,
El futuro diría: - Este poeta miente;
Tales rasgos celestiales nunca fueron de rostros terrenales.

William Shakespeare (1564-1616)
Escritor y poeta inglés.

A mi hijo Julio con todo mi corazón

Es tu risa la espada
más victoriosa.
Vencedor de las flores
y las alondras.
Rival del sol,
porvenir de mis huesos
y de mi amor.
...
Desperté de ser niño.
Nunca despiertes.
Triste llevo la boca.
Ríete siempre.
Siempre en la cuna,
defendiendo la risa
pluma por pluma.

Miguel Hernández (1910-1942)
"Nanas de la cebolla". Poeta español.

A mis directores Antonio y Rafa con mucho cariño,

«Cantemos a los grandes hombres
- que no se envanecen-
por su trabajo seguido;
y su trabajo conseguido,
ancho y hondo proseguido,
¡los años lo acrecen!
...
Allá estaban grandes hombres
para vigilarnos;
nos trataban a baqueta
- podéis creerlo, a baqueta-
maltratados a baqueta.
Decían amarnos.
...
Juntos cantemos los Grandes
son nuestros mayores;
pues nos dieron buen juicio
- nos mostraron qué es juicio -
pues de Dios viene el Juicio;
¡vale más que honores!
...
Y de ellos lo aprendimos,
no sé como se usa,
mostrando, con su trabajo,
que hay que acabar el trabajo
- bien o mal, cualquier trabajo -
sin ninguna excusa.
...
Y de ellos lo aprendimos,
casi sin saberlo.
Pero al transcurrir los años
- sólo al transcurrir los años -
sin ayuda, al pasar los años,
vino el comprenderlo.
...
Cantemos a los grandes hombres
- que no se envanecen-
por su trabajo seguido;
y su trabajo conseguido,
ancho y hondo proseguido,
¡los años lo acrecen!»

Rudyard Kipling (1865-1936)
"Stalky & Cia". Premio Nóbel de Literatura en 1907.

El viento azotaba la bandera del templo, y dos monjes disputaban sobre la cuestión. Uno de ellos decía que la bandera se movía, el otro que se movía el viento. Argumentaban sin cesar. Eno el Patriarca dijo: «No es que el viento se mueva; no es que la bandera se mueva; es que vuestras honorables mentes se mueven».

Doctrina Sutra.

No nos hace falta valor para emprender ciertas cosas porque sean difíciles, sino que son difíciles porque nos falta valor para emprenderlas

Lucio Anneo Séneca (4 a.C.- 65 d.C.)
Filósofo, orador, escritor y político hispano-romano.

Summary

The most exciting phrase to hear in science, the one that heralds new discoveries, is not 'Eureka!' (I found it!) but 'That's funny ...'

Isaac Asimov (1920 - 1992)

In the current telecommunication world it is clear the great importance that the transmission, the processing and the multimedia storage data are getting. In this line, this Ph.D thesis has focused in the enhancement of the image quality, taking advantage of the information available in other images related among them, as it is the case of a video sequence or the successive acquisition of several photographical pictures. In this way a significant improvement in the resulting image quality is achieved, over the sensor resolution used to acquire the images. The mathematical basics of these kind of algorithms is a generalization of the sampling Nyquist criteria. This generalization establishes that it is possible to reconstruct a signal from several sampling sets in presence of aliasing, if the sampling periods prove to be different for every sampling set. In the scientific literature these techniques are known as super-resolution techniques.

When dealing with real image sequences, it is not possible to assure that the shifts among the low-resolution images will provide a sufficient number of samples that allow a perfect reconstruction of the high-resolution image. For this reason, the final developed algorithm has been incorporated with the possibility of performing an interpolation of the missing data when the lack of new data precludes to improve the image quality. Therefore, the developed algorithm exhibits a clear improvement on its robustness. In this case, the lower limit in the quality will be the interpolation quality, that at the sane time is the quality usually delivered by most of the available commercial systems in the video market to carry out a size increase.

Although there are currently several algorithms capable of obtaining super-resolution improvements, all of them present one or more of the following drawbacks:

- Are usually iterative algorithms, unable to work in real time with the present available technology.
- The high memory requirements drastically put up the costs, sometimes precluding a viable implementation.

- With the aim of making the problem simpler, the key factor of the motion estimation is separated from the rest of the algorithm, assuming a perfect knowledge of the movement prior to start the process. This fact reflects neither a complete nor a realistic solution to the established problem, although for sure, it will contribute with some other very interesting information.

In this sense, a complete algorithm that offers real time and low cost performance has been pursued. Therefore, the design has been carried out by restricting it to the resources commonly presented in a hybrid video encoder, sometimes performing slight modifications. By this way, not only the low cost objective has been reached, but also the possibility of compressing the super-resolution output image has been added. The final implementation has supposed a typical co-design case, where the more computing intensive processes have been taken to hardware and the more control related processes have been taken to software.

Besides of the most immediate super-resolution application as it is the increase of the image resolution, also some applications in the field of colour reconstruction from the raw data provided by sensors that do not sample all the pixels in all the colour planes, have been investigated. This situation is quite common in cameras that makes use of the Bayesian pattern. Other application addressed is to perform a digital zoom over regions of interest using super-resolution, thus avoiding the use of mechanical parts. Finally, a mixed application that combines at the same time the colour reconstruction and the digital zoom is presented. The achieved results show a clear improvement in the super-resolved images obtained by the use and adaptation of generic hybrid video encoders.

Resumen

En resumen, la luz es la forma más refinada de la materia

Louis de De Broglie (1892-1987)
Físico francés y premio Nóbel de física en 1929.

En el presente mundo de las telecomunicaciones es patente la importancia que están cobrando la transmisión, el procesamiento y el almacenamiento de datos multimedia. Dentro de esta línea, esta Tesis Doctoral se ha centrado en la mejora de la calidad de imágenes, aprovechando la información contenida en otras imágenes relacionadas entre si, como puede ser el caso de una secuencia de vídeo o la toma sucesiva de varias fotografías. De esta forma se consigue un aumento significativo de la calidad de la imagen resultante por encima de la resolución del sensor con que dichas imágenes fueron muestreadas. El principio matemático que sustenta este tipo de algoritmos es una generalización del teorema de muestro de *Nyquist*. Esta generalización establece que es posible reconstruir una señal a partir de varias series de muestras en presencia de *aliasing* siempre y cuando se pueda asegurar que los periodos de muestreo sean diferentes para cada serie de muestras. En la literatura científica este tipo de técnicas son conocidas como técnicas de súper-resolución.

Cuando se trata de secuencias de imágenes reales, no es posible asegurar que los desplazamientos existentes entre las imágenes de baja resolución ofrezcan un conjunto suficiente de muestras que permitan la perfecta reconstrucción de la imagen de alta resolución. Por este motivo se ha dotado al algoritmo final desarrollado con la posibilidad de realizar una interpolación de los datos ausentes cuando la falta de nuevos datos impida aumentar la calidad de la imagen resultante. De esta forma el algoritmo desarrollado ve incrementada notablemente su robustez, estableciendo como límite inferior a la calidad obtenida la calidad de interpolación, que es por otra parte la calidad que ofrecen normalmente la mayoría de los sistemas usados para aumentar el tamaño de las imágenes.

Aunque existen actualmente diferentes algoritmos capaces de obtener mejoras de súper-resolución, todos ellos presentan una o más de las siguientes desventajas:

- Son algoritmos normalmente iterativos, incapaces de trabajar en tiempo real con la tecnología disponible actualmente e incluso dentro de algunos años.
- Los altos requerimientos de memoria encarecen enormemente su coste, y en ocasiones impiden una implementación viable.

- Con la intención de simplificar, separan el factor clave de la estimación del movimiento del resto del algoritmo, asumiendo un perfecto conocimiento del movimiento antes de iniciar el proceso. Esto no refleja una solución completa ni realista al problema planteado, aunque sin duda aporte otro tipo de información de gran interés.

En este sentido se ha perseguido un algoritmo completo que ofrezca prestaciones de tiempo real y de bajo coste. Para ello, se ha realizado el diseño restringiéndonos a los recursos comúnmente ofrecidos por un codificador de vídeo híbrido, efectuando en algunos casos algunas modificaciones mínimas. De esta forma, no sólo se ha logrado el objetivo de bajo coste, sino que además se ha incorporado la posibilidad de comprimir al mismo tiempo la imagen de súper-resolución resultante. La implementación final ha supuesto un caso típico de codiseño, donde se han llevado a *hardware* aquellos procesos con un procesamiento intensivo de datos, y que por lo tanto comprometen de forma importante su funcionamiento en tiempo real, y se ha llevado a *software* todo lo correspondiente a la toma de decisiones y al control e intercambio de datos entre los coprocesadores *hardware*.

Además de la aplicación más inmediata de súper-resolución como es el aumento de la resolución de una imagen o de una secuencia de imágenes, también se han investigado nuevas aplicaciones en el campo de la reconstrucción del color a partir de los datos en crudo proporcionados por sensores que no muestrean todos los píxeles de todos los planos de color. Esta situación es muy común en cámaras que utilizan el llamado patrón Bayesiano. Otra aplicación ha sido la de realizar zoom digital sobre zonas de interés mediante súper-resolución, evitando así el uso de elementos mecánicos. Finalmente se ha estudiado una aplicación mixta que combina a un mismo tiempo la reconstrucción del color y el zoom digital. Los resultados alcanzados demuestran claras mejoras en la calidad de las imágenes de súper-resolución mediante el uso y adaptación de codificadores híbridos de video disponibles.

Agradecimientos

Cosa dulce es un amigo verdadero; bucea en el fondo de nuestro corazón inquiriendo nuestras necesidades. Nos ahorra el tener que descubrirlas por nosotros mismos.

Jean de la Fontaine (1621-1695).
Escritor francés.

Ningún trabajo se realiza en solitario, y este no es una excepción. Son muchas las personas que han contribuido directa o indirectamente a que esta tesis haya por fin visto la luz, posiblemente más de las que citaré en este escueto apartado. Siempre que se realiza una relación de personas se corre el riesgo de dejar fuera algún nombre. Ya de antemano pido perdón por si cometo la torpeza de dejar de nombrar a alguien que se sienta implicado en este trabajo. Sin embargo, existen algunos nombres claves que bajo ninguna circunstancia podrían dejar de aparecer.

Cuando en una ocasión le preguntaron a ese gran escritor y divulgador científico que fue Isaac Asimov cuál era el mayor científico de todos los tiempos respondió que eso no era difícil de contestar: Sir Isaac Newton, y añadió que lo difícil hubiese sido preguntarle cuál era el segundo mayor científico, en cuyo caso se lo ocurrían múltiples respuestas: Albert Einstein, Marx Plank, Niels Bohr, Pascal, etc. En mi caso sucede algo parecido: si tengo que decir quién es la persona a la que le estoy más agradecido por la elaboración de esta tesis doctoral, no duraría ni un momento en decir que a mi adorada esposa Emi. Ella ha sido la verdadera artífice de este trabajo. Sin su ayuda (y enorme comprensión) no hubiese podido realizar la estancia en los laboratorios de Philips en Eindhoven sobre la que se fundamenta el grueso del trabajo de esta tesis. Creo que si hubiese sido ella la que me hubiese pedido marcharse a más de tres mil kilómetros de distancia a trabajar durante un año, dejándome solo, con un bebe de un año y con un trabajo de gran responsabilidad y estrés, no hubiese dudado ni por un instante en darle mi respuesta: No. Sin embargo, y para mi enorme sorpresa, ella no sólo me dio su beneplácito, sino que encima me animó, me apoyó en todo momento y me ayudó para que la empresa pudiese llegar a buen puerto. Creo que nunca le podré estar lo bastante agradecido.

El segundo lugar en mi lista de agradecimientos lo ocupan por igual dos nombres claves en mi corta vida como investigador: Antonio Núñez Ordóñez y Rafael Peset Llopis. A Rafael Peset tengo que agradecerle las interminables horas encerrados en el laboratorio que me dedico como supervisor en Philips, su enorme paciencia conmigo para conseguir entender el funcionamiento de las miles de líneas de código que forman el codificador *Picasso*, sus valiosísimas ideas y contribuciones en el desarrollo del algoritmo, su confianza en que el trabajo podría realizarse (muchas veces en contra de mis propias opiniones) y sobre todo y antes que nada, su amistad. Sin él, mi estancia en Los Países Bajos no sólo hubiese resultado estéril, sino además un auténtico infierno. A través de su ejemplo pude aprender lo que constituyen las bases de las tareas de investigación: el seguir una metodología estructurada, el diseño adecuado y cuidadoso de los experimentos y el análisis concienzudo de los resultados. A Antonio Núñez tengo tanto que agradecerle que no sé realmente por donde empezar. En primer lugar, deseo agradecerle su confianza en mí. Su confianza en que podría llegar a ser un profesor de esta universidad, su confianza en proponerme para ir a Eindhoven a investigar, y su confianza en que esta tesis vería algún día la luz. En segundo lugar desearía agradecerle su ayuda y apoyo incondicional en todo momento, incluso en momentos en los que yo mismo dudaba que las cosas fuesen a salir bien (es obvio que no soy una persona demasiado optimista). Y en tercer lugar quería agradecerle también su valiosa amistad. Creo que he tenido muchísima suerte por haber tenido el privilegio de poder trabajar con ambos. Contraigo con ambos una deuda de gratitud que difícilmente podré pagar, pero que sin duda lo intentare el resto de mi vida. Muchas gracias a los dos de todo corazón.

Como ya apuntaba al principio, una tesis Doctoral es el resultado de un trabajo en equipo y en ésta, han sido dos equipos humanos los que han colaborado de forma decisiva: el formado por el Instituto Universitario de Microelectrónica Aplicada (IUMA) y el formado por los laboratorios de Philips Research en Eindhoven (Nat.Lab). Dentro del primer equipo, tengo que destacar la ayuda de Pedro Pérez Carballo, que ha actuado como guía y amigo desde que empecé esta increíble aventura. Asimismo, no puedo dejar de agradecer las innumerables ayudas prestadas por ese gran amigo que es Félix Tobajas. De no ser por él y por su incesable apoyo, es seguro que esta tesis no hubiese visto nunca la luz. Sería también injusto no mencionar a José Ramón Sendra y su inestimable apoyo logístico. No estaría donde estoy de no ser por ellos. Dentro de este grupo de buenos amigos creo que tampoco podría agradecer todos los favores que me han hecho Margarita Marrero y Alfonso Medina. Este es otro caso donde mi deuda de gratitud excede con mucho el breve espacio dedicado a este menester. Gracias también a Valentín de Armas, a Roberto Esper-Chaín, a José López, a Roberto Sarmiento y a Juan Antonio Montiel por los muchos ánimos que me han prestado durante la escritura de la tesis y por su pródiga amistad.

Mención aparte merece el agradecimiento a Sebastián López (Chano para los amigos), en quien he encontrado no sólo un valioso e inteligente compañero de investigación sino también un gran amigo y una excelente persona. A el le debo muchas horas de discusión sobre innumerables temas multimedia, un gran numero de horas dedicadas a la minuciosa corrección de este documento y un tan ingente número de favores que no sé si sólo en esta corta vida tendré tiempo de poder devolverle aunque tan sólo sea una pequeña parte.

No puedo dejar de agradecer el valioso apoyo técnico de Enrique Mostesdeoca y Agustín Quintana, gracias a cuyo trabajo las cosas van saliendo en este Instituto. Y por supuesto a Elvira Martín, gran amiga que no sólo me escucha y me da sabios consejos, sino que también me resuelve la inmensa mayoría de mis problemas.

Tampoco puedo olvidar a los amigos y compañeros de fatigas, como Javier García, Javier del Pino y Juan Cerezo. Muchos ánimos de mi parte y muchas gracias por los ánimos recibidos. Gracias también a Antonio Hernández por sus siempre sabios y bien ponderados consejos. Él no lo sabe, pero para mí siempre ha constituido, junto con Antonio Núñez, el ideal de docencia al que me gustaría parecerme.

Y hablando de compañeros de fatigas, tampoco puedo dejar de agradecerles a Tomás Bautista y a Nieves Hernández lo mucho que me aguantaron y ayudaron en Eindhoven. Fue un tiempo duro, pero gracias a ustedes más soportable.

Otro grupo al que quisiera dar las gracias es al formado por Aurelio Vega, Manolo Chaves y Pepe Cabrera. Espero poder seguir contando siempre con su amistad.

Dentro del segundo equipo, liderado por Rafael Peset en Los Países Bajos, también quería agradecer a Ramanathan Saturaman su inagotable paciencia y doctos consejos, así como todo el tiempo que me dedicó a realizar un análisis crítico y detallado de los resultados. A Richard Kleihorse por las indicaciones a seguir en mis primeros pasos sobre ese para mí desconocido mundo de la súper-resolución. Y sobre todo, quería agradecer a Marc op de Beack, padre intelectual del primer algoritmo iterativo de súper-resolución sobre el que se basa esta tesis, su infinita paciencia conmigo, las muchas horas dedicadas a este trabajo, y su gran valor como persona. De no ser por sus claras y amenas explicaciones todavía estaría perdido en ese intrincado mundo del procesamiento de imágenes.

En mi larga estancia en Eindhoven, no sólo me traje el grueso del trabajo de la tesis, también me traje otra cosa mucho más importante: grandes y muy buenos amigos: Leandro, Andréu, Carles, Pili, Piluca, Ana, Noemí, Lourdes, David, Estephanie, Lee Cum, Eeleon, ShewYen, Twan, Ghias,... Ellos son realmente lo mejor que me pasó en Eindhoven.

Como leí hace mucho tiempo, si uno de los objetivos de la tesis doctoral es el de hacer y reforzar lazos de amistad, creo que este objetivo se ha visto sobradamente cumplido, y es sin duda alguna el logro más valioso que tiene para mi la conclusión de este trabajo. Muchas gracias a todos.

Index

Chaos is but unperceived order.

Sir Fred Hoyle (1915-2001)
Astronomer

Index of Figures

The engineer is the key figure in the material progress of the world. It is his engineering that makes a reality of the potential value of science by translating scientific knowledge into tools, resources, energy and labor to bring them into the service of man ... To make contributions of this kind the engineer requires the imagination to visualize the needs of society and to appreciate what is possible as well as the technological and broad social age understanding to bring his vision to reality.

Sir Eric Ashby (1918-2003)
Naturalist and filmmaker

Index of Tables

Research is what I'm doing when I don't know what I'm doing.

Wernher Von Braun (1912-1977)
Rocket engineer

Acronyms

There are three principal means of acquiring knowledge: observation of nature, reflection, and experimentation. Observation collects facts; reflection combines them; experimentation verifies the result of that combination.

Denis Diderot (1713 - 1784)

2D-DCT	:	*2 Dimensional Discrete Cosine Transform*
2D-IDCT	:	*2 Dimensional Inverse Discrete Cosine Transform*
3-DRS	:	*3 – Dimensional Recursive Search*
ALU	:	*Arithmetic Logic Unit*
AMBA	:	*Advanced Microcontroller Bus Architecture*
APM	:	*Advanced Prediction Mode*
ARM	:	*Advanced RISC Machines*
ASIC	:	*Application Specific Integrated Circuits*
ASIP	:	*Application Specific Integrated Processor*
ASISP	:	*Application Specific Instruction-Set Processor*
ASSP	:	*Application Specific Signal Processor*
AVGA	:	*Adapted Video Gate Array*
BCU	:	*Bus Control Unit*
bpf	:	*bits per frame*
bps	:	*bits per second*
CAM	:	*Content Addressable Memory*
CAT	:	*Computer Aided Tomography*
CCD	:	*Charge-Coupled Device*
C-HEAP	:	CPU-*controlled Heterogeneous Embedded Architectures for signal Processing*
CIF	:	*Common Intermediate Format*
CISC	:	*Complex Instruction Set Computer*
CMOS	:	*Complementary Metal Oxide Semiconductor*
CPU	:	*Central Processing Unit*
DCT	:	*Discrete Cosine Transform*
DEC	:	*Decoder unit*

DISP : *Displacement unit*
DSP : *Digital Signal Processor*
EM : *Expectation Maximization*
ENC : *Encoder Unit*
ESPRIT : *Estimation of Signal Parameters via Rotational Invariance Techniques*
FIR : *Finite Impulse Response*
FPGA : *Field Programmable Gate Array*
fps : *frames per second*
GPR : *Ground Penetrating Radar*
HAVGA : *Half Adapted Video Gate Array*
HDTV : *High Definition Television*
HMRF : *Huber-Markov Random Field*
HR : *High Resolution*
IBP : *Iterative Back-Projection*
IBR : *Image Bases Rendering*
IC : *Integrated Circuit*
IDCT : *Inverse Discrete Cosine Transform*
iid : *independent and identically distributed*
IIR : *Infinite Impulse Response*
IMC : *Inverse Motion Compensator unit*
IP : *Intellectual Property*
JPEG : *Joint Pictures Expert Group*
Kbytes : *Kilo bytes (1024 bytes)*
LR : *Low Resolution*
LSI : *Linear Space Invariant*
MAP : *Maximum A-posteriori Probability*
MB : *Macro-Block*
Mbytes : *Mega bytes (1024·1024 bytes)*
MC : *Motion Compensator*
MCM : *Multi Chip Module*
ME : *Motion Estimator*
MIMO : *Multi Input Multi Output*
MLE : *Maximum Likelihood Estimator*
MMSE : *Minimum Mean Square Error*
MP@ML : *Main Profile at Main Level*
MPEG : *Motion Pictures Expert Group*
MRF : *Markov Random Field*

MRI : *Magnetic Resonance Images*
NSP : *Native Signal Processor*
OTF : *Optical Transfer Function*
PC : *Personal Computer*
PCB : *Printed Circuit Board*
POCS : *Projection Onto Convex Sets*
PSF : *Point Spread Function*
PSNR : *Peak Signal to Noise Ratio*
Q : *Quantifier*
QAVGA : *Quarter Adapted Video Gate Array*
QCIF : *Quarter CIF*
RAM : *Random Access Memory*
RASSP : *Rapid Prototyping of Application Specific Signal Processor*
RISC : *Reduced Instruction Set Computer*
RLE : *Run Length Encoder*
ROI : *Region of Interest*
ROM : *Read Only Memory*
RTOS : *Real Time Operating System*
SAD : *Sum of Absolute Differences*
SAR : *Synthetic Aperture Radar*
SBC : *Single Board Computer*
SCC : *Spectral Correlation Coefficient*
SIMO : *Single Input Multiple Output*
SNR : *Signal to Noise Ratio*
SOC : *System On Chip*
SP@ML : *Single Profile and Main Level*
SPARC : *Scalable Processor ArChitecture*
SR : *Super-Resolution*
SRA : *Super Resolution Algorithm*
UMV : *Unrestricted Motion Vectors*
VDSP : *Video Digital Signal Processor*
VGA : *Video Gate Array*
VHDL : *VHSIC Hardware Description Language*
VHR : *Very High Resolution*
VHSIC : *Very High Speed Integrated Circuits*
VIS : *Visual Instruction Set*
VLE : *Variable Length Encoder*

VLIW : *Very Long Instruction Word*
VLSI : *Very Large Scale Integration*
VOD : *Video On Demand*
VP : *Vision Processor*
ZZ : *Zig-Zag encoder unit*

Chapter 1

Make everything as simple as possible, but not simpler.
Albert Einstein (1879-1955)

Introduction

1.1 Introduction

It is clear the great importance that the audiovisual means are getting day by day. Since the first public projection of previously recorded moving images in 1885, the acquisition, storage, transmission and reproduction of moving images and in general the movie factory, has become in one of the most important nowadays businesses. The application fields of the audiovisual technology are expected to be unlimited: video-telephony, video-conference, storage of digital video, digital television, Video On Demand (VOD), mobile multimedia, High Definition Television (HDTV), video cameras, multimedia systems, etc. Most of these applications can benefit from the inclusion of mechanisms to improve the quality of images and video. This Ph.D. thesis covers the theoretical and practical aspects for the real-time, low-cost and low-power implementation of one of such mechanisms called super-resolution, making use of the resources provided by a generic video encoder.

Super-resolution is the process of generating images with a higher resolution than the resolution of the sensor used to acquire the images. Due to the fact that the majority of the images contain abrupt edges, we can not consider that they are strictly band-limited. Therefore, the sample process will unfailingly involve certain amount of aliasing that, in general, will be manifested as distortions in the spatial domain, more concretely as the loss of many details in the image and as a spectrum overlap in the frequency domain. Super-

resolution implies a conversion from a lower density sample grid to a higher density one [Tek95] provided that some amount of aliasing exists in the low-resolution images.

If we want to increase the resolution of an image, the most immediate way would be to use higher resolution sensors, i.e. sensors with a higher amount of photosensors. The problem is that, for increasing the density (number of sensors by μm^2) it is necessary to decrease the photosensor size and in consequence, the active pixel area, where the light integration is performed. As less amount of light (photons) reaches the photosensor, this last will be much more sensitive to the noise generated by the random fluctuations in the motion of charge carriers (shot noise). It has been estimated [KIA93] that the minimum size of photodetectors is approximately of $50\mu m^2$, limit that has been already reached by the Charged-Coupled Devices (CCD) technology. One solution to this problem is to perform the resolution increase by creating algorithms intended to this purpose, as it is the case of the super-resolution algorithms. Moreover, the use of this kind of strategies allows obtaining images with a resolution equivalent to the use of higher quality sensors, and that nonetheless have been acquired using lower quality sensors at lower costs.

The problem that these algorithms have traditionally posed is that their high complexity implies high computational load. Therefore, their real time implementation it is only possible when using high-speed and massive-parallel computing hardware platforms. In this case the savings in the sensor will be concealed by the higher costs of these kind of systems.

On the other hand, the steady increasing density in the transistors integration provided by the electronic industry over monolithic circuits, has open the research to new design paradigms that allow a more efficient use of this higher densities. These paradigms include the design of large application specific integrated circuits synthesized with modern design tools, as could be: Hardware/Software (HW/SW) codesign, that starts from the breakdown of the problem in simpler tasks that will be executed by specific circuits and tasks that will be executed by programs running on dedicated processors; or the design based on programs executed by specific instruction set processors. These large approaches are simplified by the terms ASIC (Application Specific Integrated Circuits), HW/SW codesign, System on Chip (SOC) and Application Specific Integrated Processor (ASIP) or Application Specific Instruction-Set Processor (ASISP).

By means of these approaches larger embedded systems can be developed [Ern98], [Ern97], [GVN+94], [LSV+96], [GZ97], than using the conventional technology developed

until now through the integration of the components on Printed Circuit Boards (PCB) level or in Multi Chip Modules (MCM). Nevertheless, the approach that supposes the most direct translation from the PCB integration to a chip is the previously mentioned SOC [Fos99].

The SOC technology it is only economically feasible for a high volume circuit production, what means that it must be oriented to large consumer markets. In other cases it must be arranged in such a way that their architectures can support several applications. The modules that compose the architecture should be reusable in other products and applications, either as synthesizable Intellectual Property (IP) models or as a certain re-configurability capability. This strategy seeks to create multipurpose architectural platforms and the integrated systems development that follows this strategy is denominated platform based design.

This thesis pursues the implementation of super-resolution algorithms without having to recourse to specific hardware, using (reusing) the previously existing resources available in a SOC platform for hybrid video encoding. This video encoder platform has been developed by Philips Research and has been named Picasso [PKL+99]. From this point of view, super-resolution is an added value to the video encoding platform, where it can be used to increase the resolution of the acquired images, to improve the quality of the decoded images, to reconstruct the colour of images where not all the colour pixels have been sampled, or as a way to perform digital zoom without the use of mechanical parts and with a higher quality than the one obtained by interpolation.

1.2 Objective of the thesis

The objective of this thesis is the analysis, development and implementation of different kind of super-resolution algorithms, trying to achieve real time and low-cost performances. In this way the following tasks have been performed:

- To break the iterative nature of the algorithms existing in the literature, as it is impossible that, with the available resources and with the high computation load that implies every iteration step, the real time objectives could be met.
- To remove the restrictions those are traditionally imposed to the input images as: smooth movement and small shifts compared with the image size, perfectly known shifts, absence of noise and/or blur, etc. The algorithms must be robust enough to face any kind of input image sequence, as it is the case when real image sequences are acquired.

- To restrict the components used to only those ones available in the Picasso hybrid video encoding platform. If the available resources evidenced to be insufficient or inadequate, the objective will be to study the possibility of adding new coprocessors or modify the existing ones, keeping always the compatibility with the rest of the applications supported by Picasso.

These general objectives can be furthermore detailed in the following way:

1. Analysis and adaptation of the iterative super-resolution algorithms to the resources provided by the video encoder. Once adapted to the Picasso platform, all the modifications and new generated versions must be supported by the same platform.
2. Creation of new non-iterative super-resolution algorithms. The iterative behavioural rupture is the first important step through the final objective of this thesis, and it is probably one of the main contributions.
3. Adaptation of the non-iterative super-resolution algorithms. Until now, the algorithms always generate only one super-resolution image per every N input images, what implies that a sequence of size M frames is reduced to an output size of M/N frames. It is necessary to face important changes in the algorithms to switch from a static super-resolution scheme (generation of a single image) to a dynamic super-resolution scheme (generation of video sequences).
4. Independence of the algorithm with respect to the input image. As it is usually very difficult to know a priori the type of images that are going to be grabbed by the acquisition system, and in consequence that are going to be treated to increase their quality, the algorithm must be robust enough to dynamically adapt to the particularities of the input images, seeking to increase the resolution at the output whenever it is possible.
5. Imaging applications for super-resolution. Not only increasing the resolution of images (and therefore their sizes) is a super-resolution application, but also some other useful applications like image zooming without the use of mechanical parts, or colour reconstruction when not all the colours samples are present in the sensor (normally the case), are also possible and taken into account in this research work.
6. Finally, the algorithm must be modified in order to avoid using more memory than the physical amount available on the chip. In this manner we try to avoid the inclusion of external memory, with the ensuing increase in the data access delays, in the power dissipation and in the final circuit costs.

1.3 Summary of results

This Ph.D. thesis has been focused at the improvement of the image resolution by making use of the information contained in some other correlated images. Examples of such cases can be found in a video sequence, in the successive taken of several photographs or the acquisition of several scanned images, where the sensor or the image can be displaced in every sweep. In this way a significant improvement in the image quality, above the sensor resolution used to take the images, is achieved. The mathematical principle that sustains this kind of algorithms is a generalization of the Nyquist criterion. This generalization establishes that it is possible to reconstruct a signal from several samples in presence of aliasing if it is possible to assure that the sampling periods are different for every sample set. In the scientific literature these kinds of techniques are known as super-resolution techniques.

When facing real image sequences, it is not possible to assure that the shifts among the low resolution images will provide a sufficient set of samples that allows the perfect reconstruction of the high resolution image. For this reason, the final algorithm set (version 3) includes the possibility of interpolating the missing data when the lack of new information precludes the quality increase of the resulting image. By means of this, the developed algorithm highly increases its robustness, while setting the interpolation quality as the lower boundary that the algorithm can achieve. As most of the commercial image systems usually employ interpolation to increase the size and/or resolution of the pictures, the lower quality boundary of the super-resolution algorithms coincide with the quality that these systems typically deliver.

Although currently different algorithms capable of obtaining super-resolution improvements exist, all of them exhibit one or more of the following drawbacks:

- They are usually iterative algorithms, unable to work in real time neither with the currently available technology, nor probably with the technology available in the next coming years.
- The high memory requirements largely raise their costs, and sometimes preclude them from a viable implementation.
- With the aim of simplification, many times the key factors of registration (computing of the motion) and restoration (data integration) are separated and isolated, usually assuming a perfect knowledge of the motion before starting the process. Although such studies are very valuable from a theoretical point of view, they do not suppose a realistic solution to the faced problem.

- For simplicity reasons, several restrictions are imposed to the type of images to be processed, restricting their application only to a limited set. Once again, this does not suppose a realistic solution to the wide range of image types that can be found in the real world.

After studding several super-resolution algorithms (chapter 2) we have opted for implementing the algorithm outlined in [BK99] by Philips researchers, adapting it to the platform of a hybrid video encoder Picasso. The algorithms proposed in [BK99] are of iterative nature, and versions v1.0, v1.1, v1.2 and v1.3 of the super-resolution algorithms developed in this thesis are based on them, although adapted to be executed on Picasso. The referred versions belong to the category of iterative algorithms for static super-resolution or for still image, resulting in a single quality-improved image. The resulting average luminance qualities, measured as the peak signal to noise ratio, for the used test sequences were of 23.19 dB for version v1.2 and of 28.24 dB for version v1.3, which are the most representative versions for iterative static super-resolution.

Since the iterative algorithms are unable to reach real time performances with the available resources, a new type of non-iterative algorithm has been developed. This algorithm is based on the interpolation of the images into a higher resolution grid, but leaving empty the pixels not covered by the low-resolution image. These empty pixels (holes) are later filled with the successive motion-compensated input images data and combined using the concept of weighted contributions created for that purpose. Additionally, the combination of weighted images using the contributions concept solves the problems that appear in the image borders. This first algorithm set is valid for static super-resolution, i.e. for the generation of a single high-resolution image through the combination of several low-resolution images. Versions v2.0 and v2.1 belong to this category of non-iterative algorithms for static super-resolution, and the average measured qualities for both versions are 30.4701 dB and 34.6331 dB respectively.

The next step was to undertake the problem of dynamic super-resolution, by modifying version v2.1 and adapting it to a continuous input image flow in order to obtain an output stream of the same size than the input stream. This has leaded us to version v3.0 and v3.1. Nevertheless these versions present the problem of requiring large memory amounts, what has been solved by feeding-back the super-resolution data obtained for the previous image. Therefore, we have reduced the memory requirements in a factor of 50 when comparing with versions v2.0 and v2.1, and in a factor of 35 when comparing with versions v1.2 and v1.3. The reuse of the data from the previous image supposes a dilution of the value

of the pixels, although experimentally the quality loss with respect to the previous versions is less than 2 dB. The average luminance quality for all the test sequence was of 28.6415 dB.

All the algorithms presented are implemented on the platform Picasso, modifying some components but reusing most of them without performing any modification. By means of this work, the capability of performing super-resolution improvements has been added at a very low cost, while keeping the pre-existing codification capabilities.

The final implementation has supposed a typical codesign situation, where the most compute intensive tasks have been placed in hardware, as they seriously compromise the real time functioning, and the more related with control and data interchange among the hardware coprocessors tasks have been placed in software.

We have, in summary, qualitatively and quantitatively demonstrated the quality improvement for still image and video using super-resolution algorithms implemented over hybrid video encoders with real time performance, low power dissipation and at very low costs.

1.4 Thesis organization

This Thesis is organized in seven chapters that can be divided in the following four thematic blocks: a first introduction block composed by chapters 1, 2, 3 and 4; a second block that reflects the super-resolution algorithms description and how to map them in the video encoder platform composed by chapter 5; a third block where some super-resolution applications are evaluated together with the qualitative and quantitative results of the experiments carried out, composed by chapter 6; and a fourth and last block with the conclusions and further research work, composed by chapter 7. Hereafter every chapter is commented in more detail.

Chapter. 1 Introduction. In this first chapter a general overview about the contents of the thesis, the targeted objectives, and the motivation that sustains this research, are given. Furthermore, a first summary of results are presented, that globally gather the thesis milestones.

Chapter 2. State-of-the-Art. In this chapter is presented the current state of the art for the scientific objectives of the thesis, focusing on the innovations in super-resolution

algorithms, on video encoder architectures based on SOC platforms composed by a processor and heterogeneous coprocessors and on the algorithm and architectural transformation methods for the mutual function mapping. A general classification in terms of dynamic and static super-resolution algorithms is established, being both types covered in this thesis.

Chapter 3. Super-Resolution Techniques. This chapter exposes the theoretical fundaments of the super-resolution algorithms that form the basis for the subsequent modifications. At the same time, an initial iterative algorithm in pseudo code that only makes use of the resources found in a generic hybrid video encoder is proposed. The super-resolution quality that can be achieved is limited by several factors also studied in this chapter. Finally, the experimental setup is described.

Chapter 4. The Video Encoder Platform. This last introduction chapter describes the architecture Picasso for hybrid video encoding, developed by the Embedded System Architectures on Silicon (ESAS) group included in the Information and Software Technologies (IST) sector of the Philips Research Laboratories in Eindhoven (Nat. Lab.), in The Netherlands [PKL+99]. This architecture is based in the CPU-controlled Heterogeneous Embedded Architectures for signal Processing (C-HEAP) architecture [Lip97]. The super-resolution algorithms are implemented using the resources available in this architecture.

Chapter 5. Mapping of Super-Resolution onto a Video Encoder. The methodology followed in the elaboration of the experiments to obtain the quality metrics of the resulting images and to assess the overall process is exposed in this chapter. The first iterative algorithms for static super-resolution are shown together with the preliminary results and some explanations about the super-resolved image quality decrease that takes place under certain test conditions. With the aim of achieving real-time performances, a new type of non-iterative super-resolution algorithm is proposed. In this sense, the modifications that drive us towards this new algorithm are explained. Making use of the concepts of weighted contributions and feedback of the previous image, a new non-iterative version of the algorithm for dynamic super-resolution is created, achieving, thanks to the memory contention, results that allow its on-chip implementation, with real-time performance and low-cost restrictions. Jointly with the algorithm description and the charts for the memory use, also the metrics for the image quality in every case are shown.

Chapter 6. Results. Super-resolution can find several applications into the image processing chain. In this chapter, three main applications are addressed: resolution enhancement with electronic zoom, colour reconstruction and colour reconstruction with

electronic zoom. The first application exhibits a small quality increase due to the lack of aliasing in the images after the standard colour reconstruction algorithms. The second application shows good results, but the large increase in the computing load preclude its use in the sense that other interpolation algorithms can deliver similar qualities at much lower computational loads, as it is the case of SmartGreen3 from Philips. The third application modifies the pre-processing in order to keep some aliasing and combines the best of the previous applications. In this last case, the results clearly overcome other interpolating algorithms.

Chapter 7. Conclusions and Further Research. This final chapter provides the conclusions of this research work and outlines some still open lines to be further studied and developed.

Chapter 2

All truly wise thoughts have been thought already thousands of times; but to make them truly ours, we must think them over again honestly, till they take root in our personal experience.

Johann Wolfgang von Goethe (1749-1832)

State-of-the-Art

2.1 Introduction

The possibility of reconstructing a super-resolved image from a set of images was initially proposed by Huang and Tsay in 1984 [HT84], although the general sampling theorems previously formulated by Yen in 1956 [Yen56] and Papoulis in 1977 [Pap77] showed exactly the same concept (from a theoretical point of view). From the Huang and Tsay proposal until the present days, several research groups have developed different algorithms for this task of reconstruction, obtained from different strategies or analysis of the problem.

The classical theory of image restoration from blurred images and with noise has caught the attention of many researches over the last three decades. In the scientific literature, several algorithms have been proposed for this classical problem and to the problems related with it, contributing to the construction of a unified theory that comprises many of the existing restorations methods [LB91]. In the image restoration theory, mainly three different approaches exist that are widely used in order to obtain reliable restoration algorithms: Maximum Likelihood Estimators (MLE) [GW87], [Jai89], [Pra91], [LB91], Maximum A-Posteriori (MAP) probability [GW87], [Jai89], [Pra91], [LB91], [ZRP01] and the Projection Onto Convex Sets (POCS) [You78].

The great advances experimented by computer technology in the last years has led onto a renewed and growing interest in the theory of image restoration. The main approaches

are based on non-traditional treatment of the classical restoration problem, oriented to new restoration problems of second generation, and the use of algorithms that are more complex and exhibit a higher computational cost. Based on the resulting image, these new second-generation algorithms can be classified in: problems of an image restoration [Hua83], [HK84], [HT84], [Sor93], [Zer92], [Kat90], restoration of an image sequence [KDE+89], [KKE+91], [KL91], [PTS93], and reconstruction of an imaged improved with super-resolution [PST94], [EF95], [SS95], [PST95], [SS96], [EF97a], [PRK98]. This thesis is positioned on the last mentioned approach, both for the reconstruction of static image as for the reconstruction of images sequences with super-resolution improvements [MPN+02], [MPN+03a], [MPN+03b].

An alternative classification [ZSHS02] based on the processing approach can be made, where the work on super-resolution can be divided into two main categories: reconstruction-based methods [EF97a], [IP93] and learning-based methods [FP99], [BK00], [CZ01], [BK02]. The theoretical foundations for reconstruction methods are non-uniform sampling theorems, while learning-based methods employ generative models that are learned from samples. The goal of the former is to reconstruct the original (super-sampled) signal while that of the latter is to create the signal based on learned generative models. In contrast with reconstruction methods, learning-based super-resolution methods assume that corresponding low-resolution and high-resolution training image pairs are available. The majority of super-resolution algorithms belong to the signal reconstruction paradigm that formulates the problem as a signal reconstruction problem from multiple samples. Among this category are frequency-based methods, Bayesian methods, Back-Projection (BP) methods, Projection Onto Convex Set (POCS) methods, and hybrid methods. From this second classification, this thesis is positioned in the reconstruction-based methods, as we seek to reconstruct the original image without making any assumption about the generative models and assuming that only the low-resolution images are available.

2.2 Image reconstruction using super-resolution

The reconstruction problem using super-resolution can be defined as the objective of reconstructing an image or video sequence with a higher quality or resolution from a finite set of lower resolution images taken from the same scene [Ela96], [EF97b], as shown in Figure 1. This set of low-resolution images must be obtained under different capturing conditions of the image, from different spatial positions and/or from different cameras. This reconstruction problem is an aspect of the most general problem of sensor fusion.

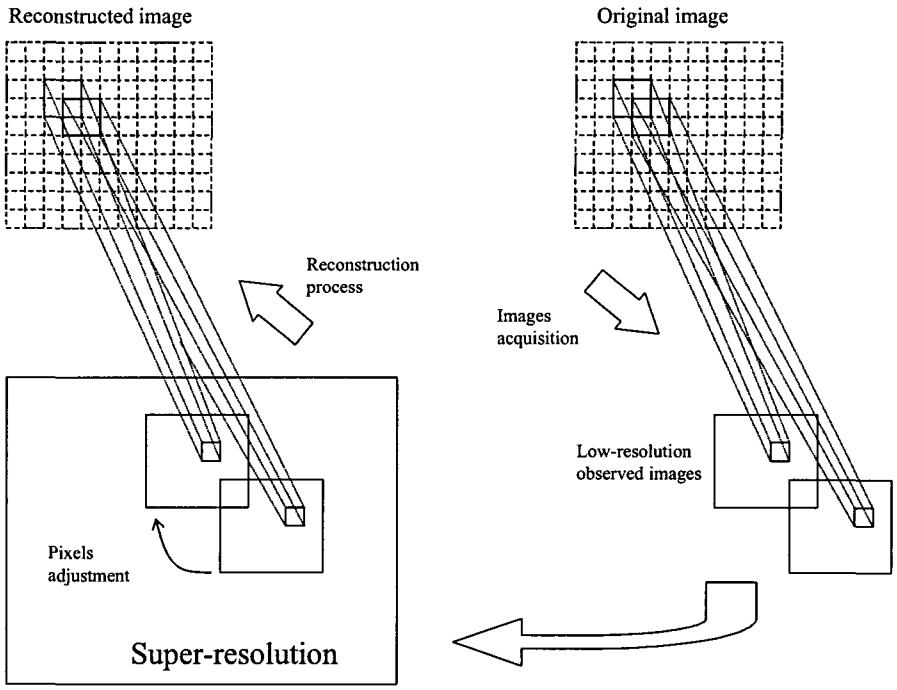


Figure 1. Model of the reconstruction process using super-resolution.

Among the multiple applications of reconstruction methods using super-resolution, the following can be highlighted:

(i) Remote acquisition [CKK+84], [Ghi84], [KIA+93], [KW93], where several images from the same scene are obtained and a higher resolution image is seeked.

(ii) Video frame improvement [BBZ96], [HKK97a], [HKK97b], [AD97], [Egg00]. Due to the fact that analogical video frames are of low quality, they are not normally suitable for performing directly a printed copy as in the digital photography case. It is possible to increase the quality of the image using several consecutive images combined in a higher resolution image by using a super-resolution algorithm (SRA).

(iii) Surveillance systems [Sch02], where super-resolution can be used to increase the quality in video surveillance systems, making it possible to use such recorded sequences as forensic digital video, and even to be admitted as evidence in the courts of law. Super-resolution can also improve night vision systems when images have been acquired with infrared sensors [SSS00] and the face recognition process for security purposes [GBA+02].

(iv) Text extraction process from image sequences [CZ00] is highly improved if the Regions Of Interest (ROI) containing the text are first super-resolved.

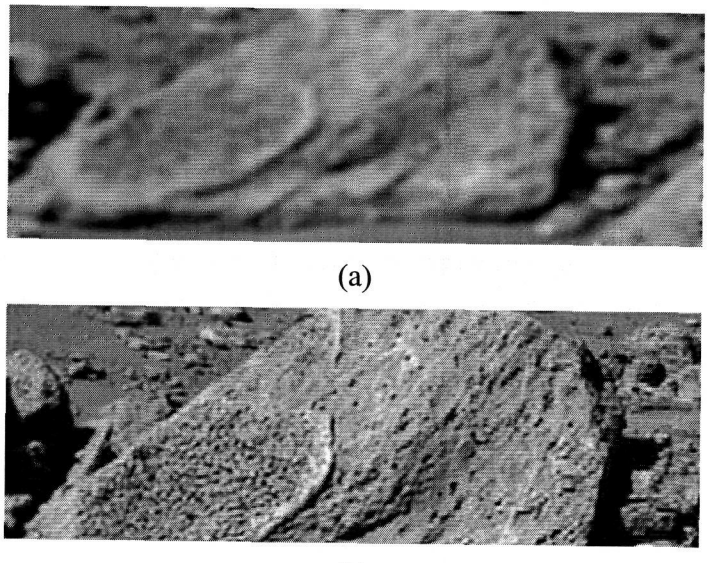
(v) Medical image acquisition [GLM+94]. Many medical equipments as the Computer Aided Tomography (CAT), the Magnetic Resonance Images (MRI) or the echography or ultra-sound images, allow the acquisition of several images, which can be combined, thus obtaining a higher resolution image.

(vi) Improvement of images from compressed video [KG98], [CS98], [ES00], [MF02]. In [APM02] the image high frequency information recovery, lost in the compression process, is addressed. These missing data is incorporated from transform-domain quantization information obtained from the compressed video bit stream.

(vii) Improvement of radar images [Can98], [CP98], [CLL02]. In this case super-resolution will allow a more clear observation of details sometimes critical for air or maritime security [PLF+01] or for land observations [WL00], [YYR+01], [LTZ+01], [TLA+01].

(viii) Quality improvement of images obtained from the outer space. An example of this is the results exposed in [CKK+94] of images taken by the Viking satellite or the resolution increase of the images transmitted from the Mars Pathfinder (Figure 2).

(ix) Image Based Rendering (IBR) of 3D objects [FJP02] use cameras to obtain rich models directly from the real-world data. Super-resolution is used to produce high-resolution scene texture from an omnidirectional image sequence [NYY00], [NYY02].



(a)

(b)

Figure 2. Low-resolution images transmitted from the *Mars Pathfinder* (a) and the results obtained by NASA after the super-resolution combination of 10 frames (images obtained from the NASA website [NASA99]).

Super-resolution techniques are not restricted only to the area of image enhancement. These techniques can also be used in other application fields. In [TFN+00] a spatial-domain path-diversity system using a multibean adaptive antenna to reduce frequency-selective fading and to realize path-diversity is proposed. The authors take four samples per symbol, thus increasing the accuracy in the estimation of the delays. In [Gin00] a locator with super-scanned phase array is used to locate objects by emitting probing pulses and sensing the reflected signals with super-resolution techniques. This technique ensured a substantial gain in energy and information as compared to traditional radar systems. In [TNB02] a system to perform a dynamic spatial-temporal measurement conducted at 5.2 GHz in a corridor is presented. The measurement was performed with a moving transmitting antenna and a fixed positioned receiving linear array. For the post-processing phase a 2-D Unitary ESPRIT super-resolution algorithm was used to extract the direction-of-arrival and the time-delay-of-arrival of the multi-path components. Super-resolution has also been used in [SAM+01] for improving the resolution of Ground Penetrating Radars (GPR), in [SLJ+00] to better resolve if the radar echoes contain single or multi-targets under certain conditions (SNR is above 10 dB), and in [XFH+01] to increase the precision in the estimation of the direction of arrival of a radar signal, thus avoiding the use of larger antenna or operating at higher frequencies.

The problem of an specific image reconstruction from a set of lower quality images with some relative movement among them is known as the static super-resolution problem. On the other side, it is the dynamic super-resolution problem, where the objective is to obtain a higher quality sequence from another lower resolution sequence, seeking that both sequences have the same length. These two problems also can be denominated as the super-resolution problem for static images and the super-resolution problem for video, respectively [Cha01].

2.2.1 Image reconstruction by static super-resolution

The idea of the super-resolution reconstruction was originally proposed by Huang and Tsay [HT84] in 1985. They faced the problem from the frequency domain to demonstrate the possibility of reconstructing an image with improved resolution from several low-resolution undersampled images without noise and from the same scene, based on the spatial aliasing effect. They assume a purely translational model and solve the dual problem of registration and restoration (the registration implies estimating the relative shifts among the observations and the restoration implies the estimation of samples on a uniform grid with a higher sampling rate). The restoration stage is actually an interpolation problem dealing with non-

uniform sampling. Other results suggest a simple generalization of this idea to include noise and blur in the low-resolution observations. Thus, in [KBV90], [BKV93], [Kim94] a recursive algorithm in the frequency domain for the reconstruction of a super-resolution image from noisy and blur samples based on a weighted recursive least square minimum theory is proposed. This method is later refined by Kim and Su, who considered the case of different blurs for every low-resolution observation, using the Tikhonov regularization to determine the solution of an inconsistent set of lineal equations [KW93]. Also [NMG01] solve the Tikhonov-regularized super-resolution problem by the conjugate gradient method using efficient block circular preconditioners. The authors extend to undetermined systems the derivation of the generalized cross-validation method for automatic calculation or regularization parameters.

Ur and Gross [UG92] propose an alternative in the spatial domain, based on the generalized multi-channel sample theorems of Papoulis [Pap77], Yen [Yen56] and Brown [Bro81] to perform a non-uniform interpolation of a spatially shifted low-resolution images set. This step is followed by another one of deblurring. A precise knowledge of the relative shifts among the input images is assumed. Other registration-interpolation method for super-resolution starting from observations with sub-pixel shifts is described in [VH99]. Srinivas and Srinath [SS90] propose a super-resolution reconstruction algorithm based in the Minimum Mean Square Error (MMSE) approach for the multiple images problem and for the interpolation of the restored images in one super-resolved image. All these restoration methods based on the use of super-resolution are limited to global and uniform translations shifts among the different samples of the same scene, to a Linear Space Invariant (LSI) blur movement and to homogeneous additive noise.

Peleg [PKS87] and Irani [IP91] suggest a different approach to the super-resolution reconstruction problem, based on the Iterative Back-Projection (IBP) method, adopted from the Computer Aided Tomography (CAT). This method starts with an initial prediction of the output image, projects the temporal result over the low-resolution samples (through simulation) and updates the temporal prediction according to the simulation error. This method is not limited, like the previous ones, to specific characteristics of the movement and allows smooth and arbitrary movement flows. In this case, the registration process uses the procedure described in [KPB88], followed by an iterative super-resolution algorithm that minimizes the error between the observed low-resolution images set and the obtained through the simulation of the low-resolution images from the high-resolution images. As the registration is performed independently from the restoration, the precision of the method mainly depends on the accuracy of the estimated shifts. The sub-pixel registration method

described in [KPB88] considers two images as two functions related between them through horizontal and vertical shifts and an angle of rotation. A serial Taylor expansion is performed for the obtained image in terms of movement parameters and it seeks to minimize an error function by the computation of their derivates with respect to their movement parameters. Nevertheless, the convergence of the proposed algorithm only has been tested for the case of correlated geometric warps among the taken images [IP93]. A recent application of this approach is the combination of the information coming from different sensors, as it is the case of colour RGB sensors using the Bayer pattern [ZP02].

Tom and Katsaggelos [TK95] have taken into account the interdependence of the registration, interpolation and restoration stages, formulating the problem as a Maximum Likelihood Estimation (MLE) that is solved with an algorithm for Expectation Maximization (EM). The problem is considered in a multi-channel environment, where the equation that describes the creation of the low-resolution images contains shift, blur and noisy variables. The structure of the matrices implied in the goal function allows an efficiently computation in the frequency domain. The MLE problem solves the computation of the sub-pixel shifts, of the variations of noise for every image and of the high-resolution image. In [KIA+93] a non-uniform sampling theorem proposed in [CPL85] to transform non-uniform distributed samples, acquired by multiple cameras in a sole uniform sample grid is used. However, if the cameras have the same aperture severe limitations both in the disposition and in the configuration of the scene will arise. Using multiple cameras with different apertures solves this difficulty. In [CB00] is described a super-resolution method using image warping. The warp characteristic of the real lenses is approximated by coupling the degradation model of the acquisition system of the images with the integration stage of the new sampling [Fan86].

Another approach to the super-resolution reconstruction problem is the one proposed by Schultz and Stevenson [SS95], [SS96]. Their approach uses the Maximum A-posteriori Probability (MAP) estimator, previously used in the Huber-Markov Random Field (HMRF). It is supposed that the distortion of the taken images is simply due to the average and that the additive noise of the samples is an Independent and Identically Distributed (IID) Gaussian vector. This choice yields to a non-quadratic algorithm complexity problem, with the subsequent increase in the complexity of the resulting minimization problem.

[WDM99] proposes a blind restoration algorithm of multi-channel high-resolution images, using several Finite Impulse Response (FIR) filters. This process consist of two stages, the first one where a blind deconvolution from Multiple Inputs and Multiple Outputs (MIMO) using FIR filters is performed and a second one that consists of a blind separation of

the combined poly-phase components. Due to the sub-sample process, every low-resolution image is a lineal combination of the high-resolution poly-phase components of the high-resolution input image, weighted with the poly-phase components of the impulse response of the individual channels. In consequence, the problem is presented as the blind two-dimensional deconvolution of a MIMO system guided by the poly-phase components of band-limited signal. Since the blind MIMO deconvolution based in second order statistical parameters contains some coherent interdependency, the poly-phase components need to be separated after the deconvolution.

Stark and Oskoui first suggested a set of theoretical estimations of high-resolution images [SO89] using the formulation of Projection Onto Convex Sets (POCS). The method was extended by Tekalp [TOS92] to include the observed noise. Moreover, they observed that POCS formulation could also be used as a new method for the restoration of images with space variable blur. It was shown that both the problem of reconstruction of high-resolution images and the space variable restoration could be reduced to the problem of solve a set of simultaneous lineal equations. Calle and Montanvert face the problem of increasing the resolution of an image as an inverse problem of image reduction [CM98]. The high-resolution image must belong to the image set that best fits the reduced estimation. The projection of an image onto this set provides one of the possible enlarged images and it is called induction. Therefore, the super-resolution problem is tackled by elaborating a regularized model that restores data lost during the enlargement process. Other theoretical approaches related to the POCS concept are exposed in [SO89], [TO92], [PST94], [PST95]. The main result obtained is the possibility of defining convex sets that represent strong restrictions in the sought image. The procedure of reconstruction based on POCS obtains the same benefits that the IBP method previously described: smooth arbitrary movement, lineal distortion variable in space and no-homogeneous additive noise. In fact, POCS can be better than IBP, as the non-linear restrictions can be easily combined with the reconstruction process. Nevertheless, the practical application of the projections that make use of the POCS method is computationally intensive, limiting their application scope.

Schultz and Stevenson describe in [SS94] a method to increase the resolution of only one observed image using super-resolution or interpolation. They propose a non-linear expansion method of the image that preserves the discontinuity, where the MAP estimation techniques optimize a convex function. Although they take into account both images with and without noise, they exclude any kind of blur in their model. In [HBA97] an environment to jointly estimate the registration parameters and the high-resolution image is presented. The registration parameters, horizontal and vertical shifts in this case, are iteratively updated

jointly with the high-resolution image through a cyclic optimization process. In [HBB+98] a two stages process to estimate the register parameters followed by the high-resolution image reconstruction from the knowledge of the optical system and the used matrix of sensors is exposed. The high-resolution estimated image is obtained through the minimization of a regularized cost function based in the observational model. Also it is shown as with the proper choose of the fit parameters, the algorithm exhibits great robustness in presence of noise. To minimize the cost function both the procedures of descendent gradient and the conjugated descendent gradient are used. In [BK00] Baker and Kanade propose an algorithm that learns based on previous recognitions for some specific types of scenarios and they apply the algorithm to face and text recognition. They also show that for large magnification factors, the super-resolution reconstruction restrictions do not allow obtaining more useful information as the enlargement increases.

Elad and Feuer [EF95], propose a unified methodology that combines the three main estimation tools in the image restoration: MLE, MAP estimator and the theoretical approximation using POCS. The proposed restoration method is quite general, but it assumes an explicit knowledge of the blur and imposes restrictions of smooth movement. They also propose a hybrid algorithm that combines the benefits of MLE simplicity and the ability of POCS to incorporate non-ellipsoidal solutions. This hybrid algorithm solves a convex minimization problem with restrictions, combining all the knowledge that *a priori* is obtained from the result required in the restoration process. Cheeseman et al applies in [CKK+94], [SCMM00] Bayesian estimation with a Gaussian model previous to the problem of integrating multiple images of satellites observed by the Viking spacecraft. In addition, some extensions of this method including tree-dimensional images are presented.

Most of the super-resolution algorithms proposed in the literature are confined to two-dimensional applications. In [SBZ+96] a three-dimensional version where the high-resolution albedo of a Lambertian surface is estimated with the knowledge of the high-resolution height and vice versa is proposed. The surfaces reconstruction problem has been formulated as that of expectation maximisation and has been tackled in a probabilistic framework using a Markov Random Field (MRF) model. The idea has been extended to the inverse problem of simultaneous reconstruction of albedo and height in [SBZ95] using the extension of Papoulis' generalized sampling theorem to N-dimensional cases.

Within the optical community, the resolution is described in terms of Optical Transfer Function (OTF). This has lead to a slightly different super-resolution definition. In [Hun95] super-resolution is defined as the image processing that allows recovering information from

beyond the spatial frequency bandwidth of the optical system used to capture the image. The physical size of the image remains unchanged. This can be seen as an equivalent method to the extrapolation in the frequency domain [Jai98]. The Gerchberg algorithm is one of the first super-resolution algorithms [Ger74]. Here constrains that exist in the object are imposed for the image in the spatial domain. The modified image is transformed to the Fourier domain after which constrains are imposed in the Fourier domain on the Fourier data. These constrains typically arise from the knowledge of the Fourier transform below the diffraction limits. The modified Fourier transform is then inversely transformed to the spatial domain. Walsh and Delaney present a modification to the Gerchberg algorithm that directly computes the components of the spatial frequencies above the diffraction limits [WN94]. Shepp and Vardi use an iterative technique based on a ML estimation of the Poisson statistics in the emission of positrons for Positrons Emission Tomography (PET). Hunt and Seminelli propose a similar algorithm where the Poisson statistics are assumed and a MAP estimation is iteratively reconstructed. The performance of such super-resolution algorithms has been studied in [SHN93]

In [Raj01], [RC01a], [RC01b] the authors expose an interesting application of the super-resolution techniques. The defocus blur due to different objects depth in an image with real aperture is used as a natural clue in the super-resolution process. The concept of depth from defocus in [CR99] has been incorporated in this scheme to recover the unknown space-varying defocus blur. As the depth is related with the relative blur in two or more observations of the same scene, it is possible to recover a high-density depth map. The author proposes a method for simultaneous super-resolution MAP estimations of both the image and the depth fields.

Mosaicing and super resolution are two ways to combine information from multiple frames in video sequences. Mosaicing displays the information of multiple frames in a single panoramic image. Super-resolution uses regions that appear in multiple frames to improve resolution and reduce noise. In [ZP00] a way for constructing a high-resolution mosaic from a video sequence in an efficient way is presented. Simple combination of the two methods is problematic since the alignment used in mosaicing may not be accurate enough for super resolution. Another issue is the efficiency of the super-resolution algorithm, which requires heavy computations, especially when applied to large images such as panoramic mosaics. Video sequences of a scene can be compactly represented in a single image using panoramic mosaicing by projecting the images to a common manifold [Sze96], [PH97], [RPF97], [RPFR98], [PRRZ00]. The overlap between the input images can be used to increase the resolution of the mosaic and reduce noise, by applying the super resolution described in

[ZP98]. However, the proposed super-resolution algorithms solve a very large optimization problem, and thus are computationally costly. They also require very accurate alignment over the entire image. In [SW01] a similar approach is presented but filtering the images prior the motion estimation to avoid aliasing and based in a global movement model.

More recently, Kursun and Favorov [KF02] propose a biological inspired method, based on a cortical model for perception, which takes advantage of the local dependences among pixels inherent in natural images. The work is based in the detection of regularities in the images that can be used to predict the interpolated pixels more accurately. The system is implemented in the form of a neural network that is modelled after the cerebral cortical network. One of the main problems of this method is how to train the sixteen neuronal cells to properly identify regularities. Moreover, it is not clear if the neuronal network will reach the same hits percentage in the regularities detection when the input images largely differ from the ones used to train the neuronal network.

In the previously commented results, the super-resolution problem has been defined as the objective to create a particular image of higher quality from a finite set of lower-quality images. All these results are based on an estimation of the motion (registering) among the initial images. Nevertheless, there exist several circumstances where the estimation of the motion is very difficult to be performed (for instance, in case of brusque movements and changes of the scene). In these cases, the obtained super-resolution reconstructed image is of low quality when classical block-matching motion-estimation and has few practical utility. On the other hand, most of the proposed methods lack feasible implementations, leaving aside the more suitable process architectures and the required performances in terms of speed, precision or costs.

2.2.2 Image reconstruction by means of dynamic super-resolution

An important and direct generalization of the specific image restoration application is the restoration of continuous image sequences [KDE+89], [KKE+91], [KL91], [PTS93]. The "continuous" term is referred to the basic assumption that the image sequence contains a filmed scene, with small motion vectors compared with size of the images. A standard video camera is supposed to be the main signal source. The theoretical interest of this problem resides in the fact that the required solution somehow implies a generalization of the known results of the still image restoration theory. Moreover, from a practical point of view, the

dynamic super-resolution reconstruction allows to remove the additive noise in a continuous image sequence, which is an important pre-processing procedure in several applications such as video encoding and computer vision [Lee80], [KSS+85], [ML85]. Deblurring this kind of images is not only an important technique to improve the visual data before showing them to a human observer or to an automatic system but also is a key phase in pre-processing. In spite of its importance, this problem of restoration of an image sequence has not been so widely considered as image restoration. This can be due in part to the memory requirements and the computation load required when even the simplest restoration algorithm is applied to such images. The existing restoration methods for continuous image sequences [KDE+89], [KKE+91], [KL91], [PTS93] are only those whose simplicity properly reduce the computational requirements, however providing some compromising solution acceptable in the results. The simplicity is typically accomplished by making assumptions in the models, such as immobility, homogeneity, causality in the image surface and locality in the deblurring filter.

As it was previously commented, most of the super-resolution algorithms applied to video are really extensions of their counterparts for still images. Irani and Peleg minimizes the average quadratic error between the observed and the simulated images using a retro-projection method similar to the one used in the computer aided tomography [IP93]. This method is the same as the one used by them in the still image super-resolution from observed shifts. Nevertheless, here the key factor is the accurate computation of the image movement. After computing the movement for different regions of the image, such regions are improved by the fusion of several successive frames that converge in the same region. Possible upgrades include an increase in the resolution, the filling of regions with occlusions and the reconstruction of transparent objects. Previously, in [KPB88] the same difference was minimized, but the minimization method was relatively simple: every pixel was examined and its value was at the same time increased in one unity, leave unchanged or decreased in one unity in such a way that a predefined cost function would decrease. In [BBZ96] a cost function is optimized that, besides of the quadratic differences between the observed low-resolution images and the simulated images, it contains continuity restrictions of second order of the reconstructed image. Likewise, the simulated low-resolution image takes into account the blur due to the movement, the optical blur and the average of the signal in every CCD cell due to the spatial sampling.

Schultz and Stevenson [SS96] use the modified algorithm for the hierarchical blocks adjustment to estimate the sub-pixel shift vectors and next solve the problem of estimate the high-resolution frame given a low-resolution sequence, as formulated using the MAP

estimation. This leads on to an optimization problem with constrains with a unique minimum. This method is similar to theirs own static image expansion method described in [SS94]. In [PST97] a complete method for video acquisition with arbitrary sampling grid and non-zero aperture time is proposed, by using an algorithm based on this model, using the POCS theory to reconstruct high-resolution video from low-resolution image sequences. However, the performance of the POCS based super-resolution algorithm will be eventually limited by the effectiveness of the motion estimation. Of course, this fact can be applied to any super-resolution algorithm based on the motion estimation. In [EST97] the technique introduced in [PST97] is extended to scenes with multiple movements objects, introducing the concepts of validation maps and segmentation maps. The validation maps were introduced to allow robust reconstruction in presence of errors in the estimation of the movement. The segmentation maps allow objects based processing. Moreover, the proposed method is capable of handling occlusions. The super-resolution algorithm for video improvement proposed by Shah and Zakhor [SZ99] also takes into account the fact that the motion estimation used in the reconstruction process can be inaccurate. With this aim, their algorithm searches a set of candidate motion estimations instead of only one motion vector for every pixel, and then a dense motion map with sub-pixel precision using both the chrominance and the luminance values is computed. The high-resolution frame estimation is subsequently generated by means of a method based on the Landweber algorithm. In [HKK97] a smooth convex function with multiple inputs is defined in order to obtain a high-resolution video sequence globally optimal. Baker and Kanade propose an algorithm for the simultaneous estimation of super-resolution video and optical-flow [Sin92a], [BFB94], taking as inputs conventional video sequences. This algorithm is especially useful when super-resolving sequences of human faces.

2.2.3 Motion estimation

An important factor in the recursive process of super-resolution reconstruction is the scene movement, which must be estimated from the captured images. There exist many approaches and methods for the motion estimation task between two given images [LK81], [Ter86], [Chi92], [CKM+93]. A group of these methods widely analyzed is the differential frame, initially proposed by Horn and Schunck [HS81]. In their approach, the motion estimation task is converted in a lineal estimation problem, where the movement is defined through motion vectors for every pixel, called optical flow [BK99b]. The main drawback of the Horn and Schunck approach is that their algorithm completely forgets the previous motion estimation results available in the image sequence, besides being an iterative method not

suitable for working with real time constrains. The method detaches the problem of plane changes in the scene and assumes that the movement flow is smooth in the time.

Chin and Willsky [CKM+93], [Chi92] propose a generalization of the Horn and Schunck algorithm, with the aim of overcome the inconvenient of not using the previous estimation results. Their algorithm is a very complex approach to the well-known Kalman Filter [CC90], applied to the optical flow recursive estimation task. Other authors had tried to make use of the temporal smoothness of the optical flow, as it is presented in [Sin92b], [FL95].

The motion estimation plays a crucial role in any super-resolution application [SSO99], [Bov00]. In fact, it is not possible to do any super-resolution reconstruction without appropriate motion estimation among images. This problem is the base of our interest in the motion estimation field. Another important motivation to concentrate our attention in the motion estimation is the fact that for the different approaches of the motion estimation, the obtained representation model is closely related and is very similar to the model obtained for the dynamic super-resolution reconstruction problem, which means that the solutions proposed to solve one of the problems can equally be valid for the other one.

Although the previously exposed method for the motion estimation are very accurate, and moreover the super-resolution algorithms are very sensible to the precision achieved by the motion estimation, in this Thesis an alternative method has been adopted widely used in the video coding systems: the block matching method [BK96], [HB98], [Ol98], [Bot00]. The lower precision of this method is compensated with its high-speed computation time, reaching real-time performance in the majority of the cases. The main drawback of the block matching is that it delivers a motion vector that is shared by the whole block of pixels, in contrast to other methods that are capable of estimating the individual movement of every pixel. On the other hand, the block matching methods are less influenced by the noise, increasing the convergence of the solutions. Additionally, the motion estimation resolution has been incremented to sub-pixel values, so increasing the sensibility of the system to small shifts. In [Tri01] the author takes an empirical approach, finding optimal sub-pixel interpolation filters by direct numerical interpolation over a large set of training examples generated by sub-sampling larger images at different translations. In addition, there exist block matching motion estimators in the DCT domain [KC98] very suitable for being used in video codecs, but their high complexity precludes them from the targeted performances of low-cost and real-time.

2.3 Image and video processing architectures

Once an algorithm has been developed and its functioning has been verified at the behavioural level for image processing, especially if it is targeted to achieve real-time performance, it will be necessary to implement it over some hardware/software platform. It is difficult for the image processing algorithms to offer real time performance in a full software implementation, as they usually include very intensive computing tasks. For instance, in [DBP+00] to obtain a 512×512 image from a sequence of ten noisy 256×256 noisy images at least 20 seconds are needed on a Sun Ultra 60 at 360 MHz, which is clearly far from real-time processing, for which less that 33 milliseconds are available to process every incoming frame at 30 frames per second. The usual way to solve this problem is through the partitioning of the algorithm, splitting up the intensive computing tasks, that will be mapped in hardware coprocessors and the intensive control tasks, which will be mapped in some more flexible components like programmable processors. By this way, we reach a trade-off solution, where the final system will increase its performance while keeping an acceptable degree of flexibility. In the following sections, different possibilities will be shown to select the image processing functional blocks.

2.3.1 Standard processors adapted to support video

The time employed to execute video processing applications in the conventional processors largely overcomes the reasonable limits. That is why new approaches to the construction of multimedia systems have started to grow.

The architecture designers have observed that the size of the data types provided by most of the processors is excessive to store the structure and the dynamic range of the multimedia information that it is going to be used. When an eight-bit data is stored in a 32 or 64 length word some space is wasted. Nevertheless, a 32-bits length word can simultaneously stored four 8-bits data. In the 64-bits processors, this store capability can be increased to eight 8-bits data or four 16-bits data. Therefore, a lower amount of storage units can store more information. If these data aggregations are treated as valid data formats and the data-path is modified in order to allow the concurrent execution of every different data field, then the execution time of the multimedia applications can be considerably speed up. These new operations employ the same Single Instruction Multiple Data (SIMD) techniques as used by some multiprocessors, where the same instruction is executed on different data. The achieved

acceleration allows the encoding and decoding of video in applications of low bit rate and can also achieve compressed video decoding for medium resolution in real time.

Based on that ideas and objectives, Hewlett-Packard designed the PA-7100LC, which incorporates a small multimedia instruction set denominated MAX-1 [Lee95]. After this, Sun Microsystems introduced these kind of extensions specially developed for multimedia applications in their UltraSPARC processors. This instruction set extension was denominated Visual Instruction Set or VIS [VIS], [KMT+95], [TONH96].

Some time later Hewlett-Packard developed the MAX-2 [Lee96], which consisted of the MAX-1 with some incorporated new instruction for data alignment and greater sub-words parallelism. Likewise, Intel incorporated the MMX instruction set in their Pentium processors [PW96]. These extensions were very similar to the previous ones, additionally incorporating instructions for the parallel multiplication of sub-words.

Other foundries that have used similar techniques are Silicon Graphics, that introduced MDMX [MIP97], and Compaq with its MVI instruction set for the Alpha processor 21264 specially thought to speed up the MPEG-2 algorithm.

In this sense the author developed a SPARC v.8 in VHDL [Mar95] which was extended with VIS based instructions of the UltraSPARC processors [BMC+96]. Furthermore, this work allowed establishing a methodology for fast prototyping of High performance RISC processing cores oriented to multimedia using VHDL [BMC+97].

Practically all the new standard processors are being developed with some real-time video processing capabilities. As the 600 MHz clock frequency has been widely exceeded, many standard processors are able to process low-resolution video. These processors with extensions in their instruction sets are sometimes call Native Signal Processors (NSP) [LBSL97], [Lap95].

2.3.2 DSPs and Application Specific Signal Processors (ASSPs)

In many circumstances and due to the application characteristics, the nature of the tasks that are going to co-exist and be interconnected is diverse. For a great amount of tasks it could be of interest that the processor, even having a general purpose instruction set,

comprises certain kind of specialized structures that eases the execution of specific processing methods. In this sense it is said that the processor is oriented to a specific application domain [BCF+97]. Typical examples of specific domains are the ones oriented to signal processing, both for audio and video data streams. This supposes a further step in the way of the specialised architectures with respect to the category exposed in section 2.3.1 based in extensions to the instruction set.

In this kind of processors, the architecture can incorporate certain structures that allow speeding up the execution of specific operations needed in such domain related applications. This choice has leaded to, among others, the Digital Signal Processors or DSPs. As an example, the DSPs usually include a multiplier to speed up the multiplications and, in some cases, specific hardware to support the consecutive data indexing. The consecutive data arrangement is an organization typically used to store tables of data (that normally come from signals that continuously evolve in the time). In such way, the execution of these sub-tasks is carried out in a shorter time, and additionally, in some particular cases, it is not necessary to include specific external circuitry to perform them, therefore avoiding an increase in the system costs, achieving a better global performance.

Inside the IUMA research group the TMS320 family from Texas Instruments [DSP] has been widely used, starting with the design of a standard-cell version of the TMS320C10 in 1992 [BNCS92]. Concretely, the author developed a parallel algorithm for the MPEG encoding loop capable of dynamically distributing the computation load among the four VLIW processors available in the TMS320C80 [GMC+98].

An important problem in this is the decision about the processor word length. There exist some tools [fro] and methods [SK95], [SVR+97], [CRS+98] to obtain and refine fixed point implementations from a high level analysis in MATLAB or C++ in floating point.

A wide documentation about DSP cores available for the integration of systems on a chip can be consulted in [Bie95]. It is highly suggested to consult guides of DSP core sellers, as for instance could be [ISD]. Examples of incorporation of modified DSP cores for Application Specific Signal Processor (ASSP) architectures can be found in [SM95], [Mad95]. The program RASSP from ARPA keeps information of their results in [RAS].

2.4 Video specific processors

The computational load of multimedia processing is dominated by the video processing tasks [Pir98], [Ack93], [Ack94], [Ack95], [Nun95]. These tasks require performing complex operations over great amounts of data at high sample rates. Real time requirements appear in order to accomplish with the requirements of the Human Vision System (HVS). Moreover, it is necessary to manage different data type flows (more frequently denominated streams, and stream processors to the processors that cope with them). The viability of many multimedia applications depends on compression schemes that ease the transmission and storage of multimedia data. Among the existing compression standards the following can be highlighted: H.261 [H261], H.263 [H263], H263+ [H263+], H.263++ [H263++] and H.264 [H264] of ITU-T, which cover communications applications such as video-phone; MPEG-1 of ISO[MPEG1], employed in the storage and CD-ROM reproduction; and the more generic MPEG-2 standard [H262], oriented to applications such as TV broadcasting or video under request. The MPEG-4 standard of ISO [MPEG4], uses a more efficient encoding approach, offering greater functionality like integration of synthetic and natural sources, use of independent video objects and user interaction based (or guided) on the contents.

The scientific literature about video specific processor architectures is very wide. In this brief summary we will follow the classification of [Fer98], [SPM98] for the most relevant VLIW issues.

As the multimedia algorithms are considerably sophisticated, the commercial success of the applications relies on their efficient execution on standard high-performance processor or on their VLSI implementation. The present standard processors (RISCs, super-scalars, etc.) cannot execute generic multimedia applications without performing previous adaptations. These processors are neither adapted to the signal processing nor to the stream processing. Moreover are too expensive and the power consumption is too high for portable multimedia applications, the field that expects a high growth for the next years. At the same time, although the standard DSPs are oriented and optimised for the speech an audio processing, they cannot reach the high performance demanded by the video applications. In consequence, new architectures specifically adapted to video processing, derived from standard processors and DSPs are being developed, always targeting low-cost restrictions. Currently there exists a convergence process among them. The first ones used to be oriented to low-binary rate applications or to real-time decoding. The second ones used to challenge the real-time MPEG-

2 encoding, as well as the higher levels and profiles of the standards. The monolithic integration of a MP@ML MPEG-2 decoder dates from 1994 [Tho94]. Nowadays many companies as Thomson, ST Microelectronics, Intel, LSI Logic, C-Cube, IIT, Amphion or IBM comprise these decoders, where the main market is the user set-top boxes of equipments connected to TVs. There are also available MP@ML MPEG-2 encoders on a chip, as the ones from DV^X5110 and DV^X6210 [Ccu97], but the problem of designing integrated encoders on a single chip for higher levels and profiles is still a challenge much more complex than decoding. Afterwards, we will review the evolution of these kinds of architectures in the last decade.

The specific architectures can be classified into fixed and programmable. The fixed ones used to be composed by a control unit and several specific units dedicated to different parts of the algorithm adopting an ASIP or ASSP architecture type. The programmable specific ones, denominated VDSP or VP, are composed by one or several programmable data-paths with less specific structures programmable, configurable or parameterizable. They offer higher flexibility and usability as the standards evolve. The standards evolve in terms of image resolution, output binary rate, syntax of the binary steam, variation margins of different parameters, interpretation algorithms, error resilience and concealment, temporal and spatial SNR scalability, etc., but many computing structures are commons. The progressive transparency in the visibility of the computing structures at the instruction format level has lead to a programmable specific architecture variant denominated Very Long Instruction Word or VLIW.

a) Fixed ASIP architectures

The Video RISC Processor (VRP) CL4000 [Bur93] is a scalable ASIP. With two VRPs it is possible to code MPEG-1 in SIF format. With 10 VRPs MPEG-2 MP@ML is coded [BK96]. A sole VRP2 CL4100 can code MPEG-1 in real-time.

The VideoFlow [Lee94] can code with two chips H.261 in CIF (Common Intermediate Format) or MPEG in SIF. The Enc-C and Enc-M of NTT [Kon96, Ike96] can code MPEG-2 SP@ML. All these architectures are ASIPs.

b) Programmable V-DSP architectures

Among the V-DSP specific heterogeneous the following architectures can be found the Vision Controller (VC), the Video Processor (VP) [Bai92] and the VCP [IIT93] from IIT,

the AVP1300E for H.261 or the AVP1400E and the AVP4310E [Ack93] for higher levels, both from Lucent Technologies, the VDSP [Aon92] and VDSP2 [Ara94], [Toy94], [Aki94] from Matsushita, for MPEG or the chip-set VISC from LSI Logic [LSI96] for MPEG-2.

Among the V-DSP homogeneous programmable specific architectures, the VSP3 [Ino93], [Eno93] from NEC can be found, that can code H.261 in CIF, or the HiPAR-USP [RK96] that can code MP@ML MPEG-2 without motion estimation. These architectures are homogeneous; the coprocessors work on different parts of the entire image.

c) Programmable VLIW architectures

The architectural trend of tight coupling the coprocessors or functional units with the same control unit has lead to the mono-processor architectures of Long Instruction Word (LIW) or Very Long Instruction Word (VLIW) [NC89], [FDF98]. These processors are very suitable for multimedia applications, specifically for image and graphics processing, and, in a lower amount, for video [DWW+96], [MT97].

The three most significant VLIW architectures are probably the VelociTI architecture from Texas Instrument [Ses98], the Mpact2 architecture from Chromatic Research [Pur98] and the Trimedia architecture from Philips Semiconductors [RS98]. Trimedia has been specially thought for the MPEG-2 real-time decoding and probably offers a better performance.

d) Programmable architectures for SOC platforms

The increasing importance in the design based on SOC platforms leads to the pre-eminence of an heterogeneous architectural scheme that comes from the combination of the previous b) and c) categories. The C-HEAP Philips architecture [Lip97] and Picasso [PKL+99] belong to this last category. The author has published some research about the design and reuse methodology for IP soft-cores with built-in performance metrics [MCE+02], [MCM+02a], including a quantitative approach to analyze and bound the synthesis-to-layout performance-spread of soft-IP cores [MCM+02b].

The present research work is centred in the study of iterative super-resolution algorithms and their modification to achieve real-time performance. These algorithms have been developed in such way that their execution is performed using the resources provided by the hybrid video encoder of the Picasso architectural platform, therefore assuring a low-cost implementation. The problem of mutual and adapted transformation between algorithm and

architecture is usually known as the mapping problem. The contributions to this problem from a base algorithm and an established architecture are a central part of this thesis.

2.5 Conclusions

The richness of the signals we are dealing –image sequences– offers a new possibility: to reconstruct the images of the sequence with an improved quality. This idea is based on the fact that the recorded data are taken for every image from a slightly different position (due both to the movement of the camera and to the movement of the objects inside the scene), in such a way that they can be combined to create a higher resolution output image. While in the static super-resolution approach a new image was created from a given image set, in the dynamic super-resolution approach we create an image sequence with the same length as the lower quality image sequence from where the process starts. Therefore, the application is a combination of two ideas: the restoration of a single image from an image sequence (static super-resolution) and the restoration of an image sequence (dynamic super-resolution). Each one of these problems is treated in chapter 5 of this thesis. The combination of both applications using almost the same algorithm is a novel contribution of this work. In fact, as it has been exposed, there are several known methods to address the super-resolution reconstruction, but none of them gives a solution to the problem of progressive real-time and low-cost reconstruction for video sequences. The existing methods for the restoration of continuous image sequences are still far from real-time performance for real video sequences. The reasons are that they use complex iterative methods such as POCS, or they need a perfectly prior knowledge of the movement and the blur, or they perform a set of presumptions as could be the smoothness of the movement, no occlusions, or global movement. The application faced in this thesis takes a low-resolution input sequence with no restrictions at all and tries to reconstruct its quality using super-resolution techniques, accomplishing real-time and low-cost performances. This application will not be possible without the computational resources existing nowadays. The basic challenge is to obtain new algorithms and to optimize them for their implementation on low-cost computation architectures. Therefore, it is necessary to address the solution of the problem by means of the use of powerful computational video cores oriented to the consumer markets, likewise to introduce new flexible computational cores that can support these new algorithms such super-resolution.

Chapter 3

It is a good morning exercise for a research scientist to discard a pet hypothesis every day before breakfast. It keeps him young.

Konrad Lorenz (1903 - 1989)
Nobel Prize in Physiology or Medicine 1973

Super-Resolution Techniques

3.1 Introduction

In this chapter, we will show the used approach to super-resolution. This work is initially based in the iterative super-resolution algorithm proposed by Marc op de Beeck and Richard Kleihorst [Bee97], [BK99] at the Philips Research Laboratories in Eindhoven. This algorithm neither uses the resources available in a real hybrid video encoder (although it is suggested a way to do it) nor is capable to work in real time, but it has supposed the started point to be implemented in the Picasso encoder. Based in this implementation, we been able to study its properties and perform the appropriate modifications to obtain a final implementation capable to process video sequences in real time. Although the iterative algorithms are not very suitable to work in real time (in general), they have the advantage of robustness in presence of noise and provide very good results even when the shifts among the images are inaccurately knew. As the number of frames to be combined can be modified, the non-iterative algorithms suppose an interesting approach to increase the quality of still pictures. Instead of taking only one picture (normal photograph function) we could take

several pictures (may be configurable) assuring some movement among the images and combine then in a higher resolution one.

3.2 Theoretical basics

We will start exposing an analytic approach to the super-resolution process, using the following terminology:

- x, y: low resolution coordinates.
- $\hat{x}, \hat{y}$: high resolution coordinates.
- p: number of low resolution images to be combined.
- $\Delta\delta_l(x,y)_{(fr2ref)}$: Horizontal shifts of pixel x,y of frame 'l' with respect to the reference.
- $\Delta\delta_l(x,y)_{(ref2fr)}$: Horizontal shift of pixel x,y of the reference with respect to the frame 'l'.
- $\Delta\lambda_l(x,y)_{(fr2ref)}$: Vertical shift of pixel x,y of frame 'l' with respect to the reference.
- $\Delta\lambda_l(x,y)_{(ref2fr)}$: Vertical shift of pixel x,y of the reference with respect to the frame 'l'.
- We will use the superscript $^{(n)}$ / $n \in \mathbb{N}$ to indicate n^{th} iteration data.

Calling $f(x,y,t)$ to the low resolution input image, all the input sub-system effects (lenses filtering, chromatic irregularities, sample distortions, information loss due to format conversions, system blur, etc.) will be included in $h(x,y)$. So, assuming linear effects in lens, sensors, and colour processing, the input to the algorithm will be the two dimensional convolution expressed in (1).

$$g(x,y,t) = f(x,y,t) ** h(x,y) \tag{1}$$

Calling $S(\hat{x},\hat{y},t)$ to the image obtained after applying the SR algorithm, and $SR(\hat{x},\hat{y})$ to the SR algorithm itself, they are related as indicated in (2).

$$S(\hat{x},\hat{y},t) = g(x,y,t) ** SR(\hat{x},\hat{y}) \tag{2}$$

These relationships are summarized in Figure 3-(a) concerning to the real system and are simplified in Figure 3-(b).

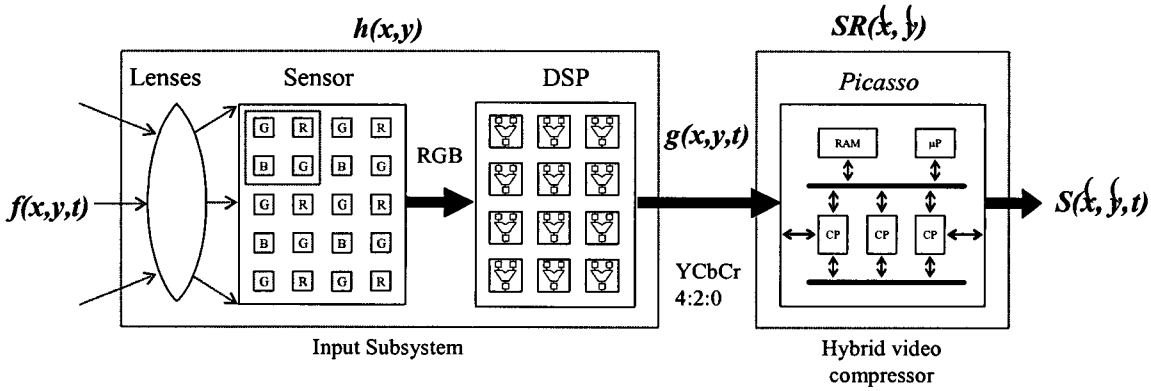
The algorithm starts supposing that a number '*p*' of low resolution images of size N×M is available as *g(x,y,ti)*, where '*ti*' denotes the sample time of the image. As the algorithm only refers the last '*p*' images, from now on the index '*l*' will be used, defined as *l=i mod p*, to refer the images inside the algorithm. For clearness, the memory image $g'_l(x,y)$ will be used to store the input image that the algorithm is going to use, and it is linked to *g(x,y,ti)* through (3) as can be seen in Figure 4. If we call $\overline{g}'_l(x,y)$ to the average input image, as given in (4), the average error for the first iteration is obtained by computing the differences between this average image and every input image, as show in (5).

$$g'_l(x,y) = g(x,y,t_i) \quad / \quad l = i \bmod p \tag{3}$$

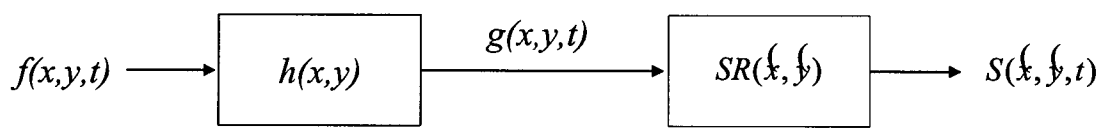
$$\overline{g}'(x,y) = \frac{1}{p}\sum_{l=0}^{p-1}\left(\frac{1}{N\cdot M}\cdot\sum_{i=0}^{N-1}\sum_{j=0}^{M-1} g'_l(i,j)\right) \quad , \quad \forall\, x,y \tag{4}$$

$$e_l(x,y)^{(1)} = g'_l(x,y) - \overline{g}'(x,y) \quad , \quad l = 0..(p-1) \tag{5}$$

This error must be transformed to high-resolution coordinates through, by example, a nearest neighbour replication interpolator (6). As the missing pixels are going to be recovered by the SR algorithm, it is not worth to use a higher quality interpolator, which will be slower and more expensive.



(a)



(b)

Figure 3. Scheme of the real system (a) and the simplified model (b).

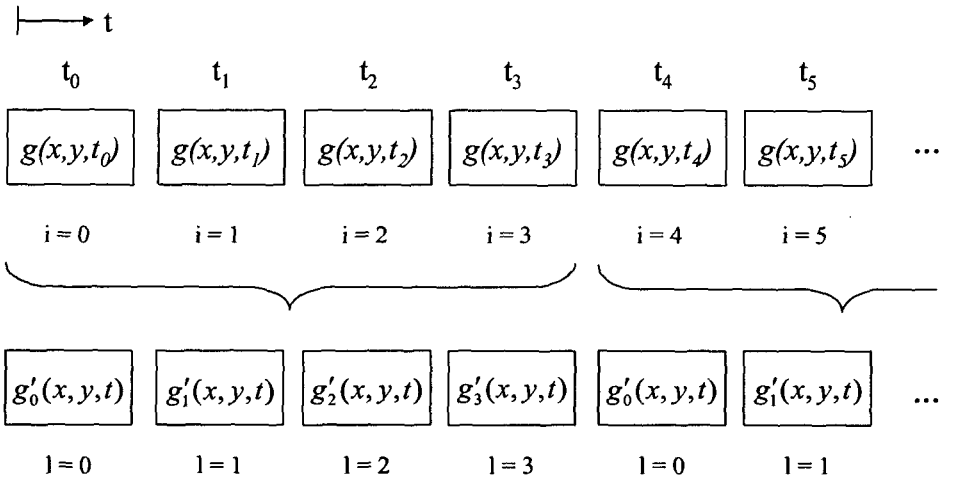


Figure 4. Nomenclature and numeration used for refer the input images.

$$e_l(\hat{x},\hat{y})^{(1)} = \textbf{\textit{upsample}}(e_l(x,y)^{(1)}) \quad , \quad l = 0..(p-1) \tag{6}$$

Once in high resolution the error must be adjusted to the reference frame, shifting the image $\Delta\delta_l(x,y)^{(1)}_{(fr2ref)}$ and $\Delta\lambda_l(x,y)^{(1)}_{(fr2ref)}$ amounts in the horizontal and vertical coordinates respectively. In principle, these displacements are applied to every pixel individually, but this will depend upon the motion estimation technique employed. When all the errors have been adjusted to the reference, they are averaged and this average will be taken as the first update of the SR image, as shown in (7).

$$S_0(\hat{x},\hat{y})^{(1)} = \frac{1}{p}\cdot\sum_{l=0}^{p-1} e_l\left(\hat{x} + \Delta\delta_l(x,y)^{(1)}_{(fr2ref)}, \hat{y} + \Delta\lambda_l(x,y)^{(1)}_{(fr2ref)}\right)^{(1)} \tag{7}$$

Equation (7) reflects the result of the first iteration. For instance, $S_0(\hat{x},\hat{y})^{(1)}$ is the first version of the SR image, corresponding to $t=t_0$, and it will be upgraded in every iteration. The nth iteration starts from that image and begins obtaining by decimation a low resolution version followed by the computation of the displacements between every one of these inputs images and this decimated image and vice versa, i.e. between the decimated image and the input images. In that manner the displacements of the nth iteration will be available: $\Delta\delta_l(x,y)^{(n)}_{(fr2ref)}$, $\Delta\lambda_l(x,y)^{(n)}_{(fr2ref)}$, $\Delta\delta_l(x,y)^{(n)}_{(ref2ref)}$ and $\Delta\lambda_l(x,y)^{(n)}_{(ref2fr)}$. The low-resolution version of the image obtained in high resolution is given by (8).

$$S_0(x,y)^{(n)} = \textbf{\textit{downsample}}\left(S_0(\hat{x},\hat{y})^{(n-1)}\right) \tag{8}$$

The next step is to compensate the motion of the high resolution image towards the input frames using the displacements $\Delta\delta_l(x,y)^{(n)}_{(ref2ref)}$ and $\Delta\lambda_l(x,y)^{(n)}_{(ref2fr)}$, converting then to low resolution and getting the error respecting to every input image, as shown in (9).

$$e_l(x,y)^{(n)} = g'_l(x,y) - S_0\left(x+\Delta\delta_l(x,y)^{(n)}_{(ref2fr)},\, y+\Delta\lambda_l(x,y)^{(n)}_{ref2fr}\right)^{(n)} \qquad l = 0\,..\,(p-1) \tag{9}$$

This low-resolution error must be taken again to high resolution through interpolation and compensate it motion again towards the reference. The average of these 'p' errors constitutes the nth incremental update of the high-resolution image, as shown in (10).

$$S_0(\hat{x},\hat{y})^{(n)} = S_0(\hat{x},\hat{y})^{(n-1)} + \frac{1}{p}\cdot\sum_{l=0}^{p-1} e_l\left(\hat{x}+\Delta\delta_l(x,y)^{(n)}_{(fr2ref)},\hat{y}+\Delta\lambda_l(x,y)^{(n)}_{(fr2ref)}\right)^{(n)} \tag{10}$$

The convergence is reached when the changes in the average error are negligible, i.e. when the variance of the average error is below of a certain threshold determined in an empirical way.

Once the SR image is obtained for time *t0* with the first '*p*' images, the process must be repeated with the next '*p*' images to obtain the next SR image using a previously established number of iterations or iterating up to convergence. Acquire a SR image implies the use of '*p*' low resolution images, and so, at the instant t_i, the SR image *k=integer(i/p)* will be generated. In such case, equation (10) must be generalized in (11).

$$S_k(\hat{x},\hat{y})^{(n)} = S_k(\hat{x},\hat{y})^{(n-1)} + \frac{1}{p}\cdot\sum_{l=0}^{p-1} e_l\left(\hat{x}+\Delta\delta_l(x,y)^{(n)}_{(fr2ref)},\hat{y}+\Delta\lambda_l(x,y)^{(n)}_{(fr2ref)}\right)^{(n)} \tag{11}$$

This last and more general equation (11) reflects the SR image at instant *'k'* as a combination of *'p'* low resolution images after *'n'* iterations.

3.2.1 Static versus dynamic super resolution

The problem of reconstructing an image from a set of low-resolution images with some relative movement among them is known as the static super-resolution problem. In contraposition, it is the dynamic super-resolution problem, where it is tried to obtain a higher quality sequence from another sequence of low-resolution images, seeking equal length in both sequences. These two problems can also be denominated static-image and video super-resolution problems, respectively [Cha01]. These two ideas can be graphically summarized in Figure 5 for static (a) and dynamic (b) super-resolution.

Static super-resolution is mainly intended to photographic applications, although, if the system is fast enough, it can be used for video applications, repeating the same process for several overlapped or not overlapped images sets. This work starts in static super-resolution but always with the aim of migrating to dynamic applications.

3.3 Initial algorithm

As depicted in section 3.2 the initial algorithm was an iterative one intended for static super-resolution. The first super-resolution algorithm follows as close as possible the initial version, based on the principle that the only available data are several sets of the sub-sample image sequences. In the first iteration, the sample positions of the image sets are undefined. Moreover, due to the existing aliasing in the low-resolution image sets, it is foreseeable to commit errors in the motion estimation among images, as the block correlation techniques do not work properly in presence of aliasing. Setting out from noisy data sequences, the method is able to iterate until obtain an interpolated higher resolution result that optimally fulfil all and every low resolution input image.

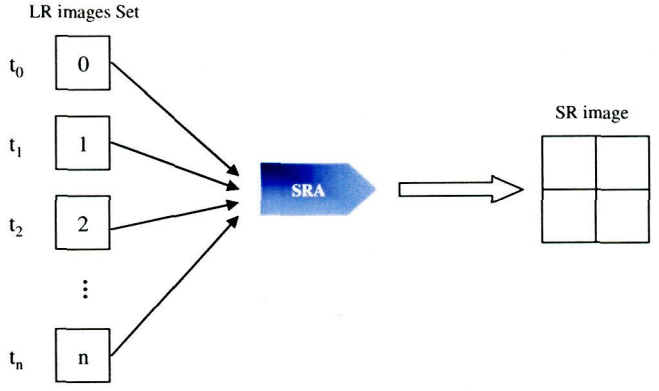
The iterative process is composed by the following steps:

1. Determine the relative shifts of the image set with respect to a reference image. Due to aliasing, these results may not be accurate and must be refined in the following steps.
2. Perform a nearest neighbour interpolation of the aliased images separately, creating a new image set over a high-resolution grid. Large deviations are to be expected from the exact interpolation curve. This procedure assures that the first iteration result will be of low resolution, but largely free of noise. High-resolution information will be

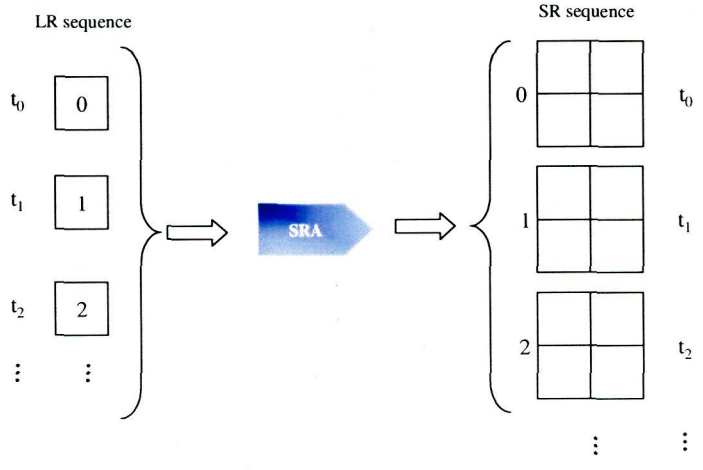
added in further iterations.

3. Sum all the interpolated images. This summation assures further noise reduction, while the first iteration result deviates from the original data set in almost all sampling positions.
4. Realign the low-resolution images with the newly created interpolation. Since this high-resolution image contains much less aliasing, standard techniques can be applied to generate sub-pixels displacement vectors.
5. Determine by sub-sampling the difference between the high-resolution estimation and different low-resolution image set. Sum all the differences and feedback them as an update to the high-resolution approximation.
6. Iterate steps 4 and 5 until convergence.

If redundant data sets are available, this iterative scheme is very well suited for noise reduction techniques as statistical information can easily be incorporated into the feedback loop. In the case that insufficient data was available, the iteration process can be stopped



(a)



(b)

Figure 5. Static (a) and dynamic (b) super-resolution schemes.

before full convergence, yielding a low pass version, which is optimally compatible with the given data sets.

3.3.1 Pseudo-code of the basic iterative algorithm

The next step has been to write a description of the algorithm in pseudo-code, with the objective of making its implementation easier. This first version is shown in Figure 6. This version uses three high-resolution memories (with HR prefix) and a variable amount of low-resolution memories (with LR prefix) equal to the number of stored input images for the algorithm to start, plus one additional memory for storage of the intermediate low-resolution data (LR_B).

Performing a conceptual analysis of the algorithm, we can see that the process starts with the calculus of differences between every input image (only luminance) and one image with an average value of its high-resolution equivalent. Every error image is shifted to be adjusted to the reference image and the average error is calculated. This shifted and averaged error constitutes the first update of the super-resolution image that has been previously initialized to a zero value. The next iterations basically follow the same process, but starting always from the last super-resolution image obtained, and thereby, the movement has to be calculated again, i.e. first the displacements between the input and the high-resolution (decimated to low-resolution) images are calculated and vice versa. Later on, the super-resolution image is shifted again to the input image. This shifted image is decimated to low-resolution to calculate the error with respect to the input image. The error is taken again to high-resolution and it is shifted to the reference. This procedure must be repeated, inside every iteration, for every input image. The resulting errors are averaged and this average constitutes the update of the super-resolution image. The quality of the super-resolution image will increase with the iterations, although this improvement will be lower as the super-resolution image will fit all the available input images.

3.4 Limitations

The super-resolution process has in fact several limitations. In our approach, we have found the following described main limitations concerning to aliasing, the motion estimation and the image borders. Nevertheless, in [LS01], [BK00] it can be found some fundamental limits and how to address them from a theoretical point of view.

```
Create HR image HR_B, HR_B_Shift, HR_B_Average
Create LR image LR_B
Create series of LR images LR_I[#images]

Read an aligned non-interlaced series to LR_I[]
Pixel align LR_I[] with LR_I[0], remember sub-pix shift

Fill HR_B with zero
Fill LR_B with average

First Iteration
    LR_B = LR_I[] - LR_B
    Upsample LR_B to HR_B_Shift
    Back shift HR_B_Shift and average in HR_B_Average
    HR_B = HR_B + HR_B_Average
Next Iterations
    Shift HR_B to HR_B_Shift
    Downsample HR_B_Shift to LR_B
    LR_B = LR_I[] - LR_B
    Upsample LR_B to HR_B_Shift
    Back shift HR_B_Shift and average in HR_B_Average
    HR_B = HR_B + HR_B_Average
```

Figure 6. Pseudo-code of the first version of the iterative algorithm.

3.4.1 Aliasing

In opposition to usual image operations as could be compression, super-resolution requires some amount of aliasing in the low-resolution images. This requirement can be seen from the following point of view: super-resolution tries to recover mainly the high-frequency information, but, if such information has been removed by a low-pass filter, then, there is no high-resolution information to recover. Of course, from a mathematical point of view, it must be expressed in a more formal way.

In Figure 7 it can be graphically seen the conditions that the magnitude of the Fourier transform of the image must accomplish to avoid aliasing (a) and to have aliasing (b). Calling $\tilde{G}(u,v)$ to the Fourier transform of the image as a function of the spatial frequencies 'u' and 'v', if the band-with in the horizontal direction is lower than the inverse of twice the horizontal sampled period X and the band-with in the vertical direction is lower than the

inverse of twice the vertical sampled period Y, then there will be not aliasing in the sampled image. In this case, the original image $g(x,y)$ can be reconstructed as establish the Whitaker-Kotelnikov-Shannon sampling expansion (2-D sampling reconstruction theorem), reflected in equation (12), where the two-dimensional sinc(x,y) function is given by (13).

$$g(x,y) = \sum_m \sum_n g(mX, nY) \cdot \mathrm{sinc}(x - mX, y - nY) \tag{12}$$

$$\mathrm{sinc}(x,y) = \frac{\sin(\pi x) \cdot \sin(\pi y)}{(\pi x) \cdot (\pi y)} \tag{13}$$

Nowadays, most of the acquisition systems incorporate some kind of anti-aliasing filters, such as lenses with Optical Low-Pass (OLP) filter, micro-lenses with OLP filters or any kind of additional low-pass filters. The micro-lenses not only avoid the aliasing but also they increase the fill-factor of the sensors, at the cost of increasing the image blur. If the low-pass filter can be separated from the rest of the system, keeping the aliasing, then there will be no problem for super-resolution, but if the sensor itself incorporates built-in micro-lenses, then the problem becomes much more complex. Although a raw sensor without micro-lenses should be cheaper (in comparison with another sensor with micro-lenses) the problem is that such a technological process is now well established in the manufacture process and removing it is not so easy. Moreover, only the super-resolution process requires aliasing, in contraposition of several other applications that need aliasing to be removed.

The solution could be using an acquisition system without micro-lenses and a configurable external low-pass filter that will be active for all applications except for super-resolution. When used in super-resolution mode, it would be able to deliver images with a higher resolution than the one the sensor is able to deliver. Nevertheless, even in the case that the anti-aliasing lenses could not be removed, from a strategic point of view, it is advisable to have a system capable of removing aliasing (and so increasing the resolution) for any imaging application that can arise in the close future.

3.4.2 Motion estimation

One key aspect in the super-resolution process is the correct computation of the movement among images [Bov00]. In subsequent sections, it will be shown how sensitive the super-resolution results are to an accurate knowledge of the motion.

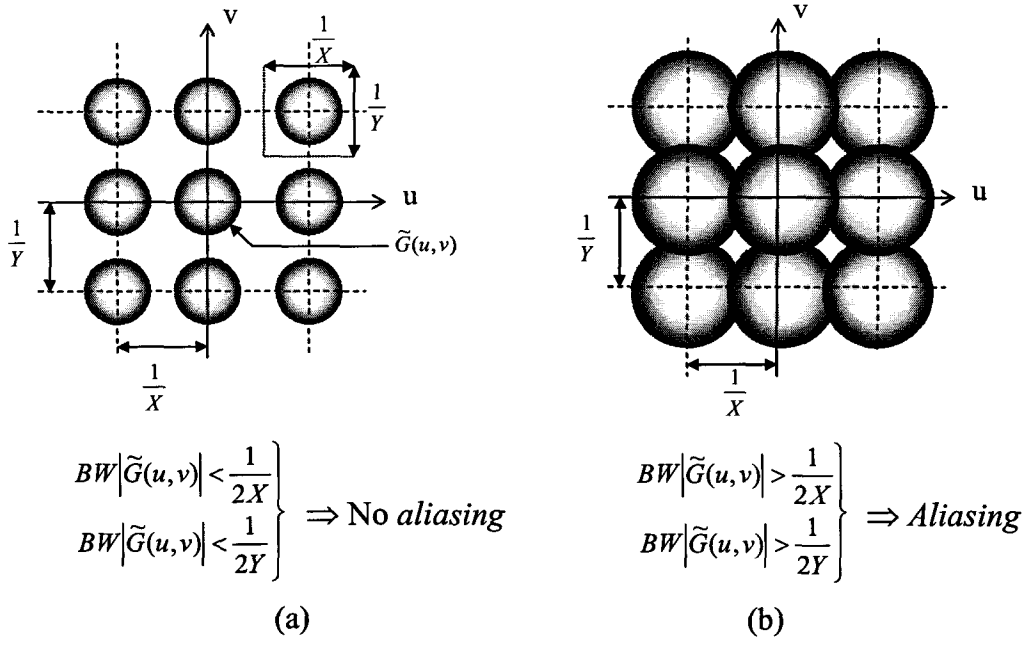


Figure 7. Spectral conditions of no aliasing (a) and of aliasing (b).

Several ways to compute the motion have been proposed in the specialized literature. One approach is the differential framework, initially proposed by Horn and Schunck [HS81]. In their approach, it is shown that the motion estimation task becomes a linear estimation problem, where the motion is defined by vectors of motion per each pixel, called optical flow. The main drawback in the Horn and Schunck approach is that their algorithm totally disregards prior estimation results on the image sequence. Assuming that the motion flow is temporally smooth, one can benefit from using previous results, both from the computation load and from the estimation quality points of view. Moreover the method is iterative and hardly suitable to work in real time.

Chin and Willsky, in [CKM+93], [Chi92], have proposed a generalization of the Horn and Schunck algorithm to overcome the above drawback. Their algorithm is a very complex approximation of the well known Kalman Filter [CC90], applied to the recursive optical flow estimation task. Other attempts to use the temporal smoothness of the optical flow are shown in [Sin92b], [FL95].

Although the above mentioned methods are very accurate, and in addition the super-resolution algorithms are very sensitive to the precision of the motion estimation, in this work we have opted for an alternative method widely used in video compression: the block matching method [BK96], [HB95], [Ol98], [Bot00] among others widely used in most hybrid

video encoders. The lowest precision that this method offers is compensated by its fast execution, reaching without major problems real-time performance for the majority of the cases. The main drawback of the block matching is that it obtains a motion vector that is shared by all the pixels in the macro-block, in contrast with other methods capable of compute the individual movement of every pixel. In contrast, the block matching methods are less sensitive to the environment noise, increasing the convergence of the solutions. Additionally, the resolution of the motion estimator has been incremented to sub-pixel values, thus increasing the sensitivity of the system. There also exist block-matching motion estimators in the DCT domain [KC98] very suitable to be used in video coders, but their high complexity move them away from low-cost and real-time performance.

3.4.3 Borders

Image borders are often a problem in most of the image processing applications. In super-resolution, borders are especially complicated because when getting new information from several images, it easier to get it from the central body of the image, but borders are usually different from image to image. For instance, if the scene 'cameraman' is recorded, taking four shifted pictures in consecutive instants of time (see Figure 8), consequently having a high time redundancy, when trying to merge the overall information in a new higher resolution image, we realize that we have four times information about the body center of the image, but few less information about borders. In this example, the right corner of the curve roof of the building (similar to a dome) in the background on the right side of the image is only present in the fourth frame, what makes impossible to improve the resolution of that detail of the image. Moreover, depending on the super-resolution algorithm, due to the non-redundant information on borders, it is possible to loose quality in such zones.

At the same time, it is important to keep in mind that a video compressor architecture is going to be used, which follows an specific strategy about borders. For example, when the motion estimation is allowed to be performed out of the borders, a virtual border replication is made, delivering motion vectors which are suitable for compression, but not for super-resolution applications. The same occurs with the motion compensation, which replicates the last borderline if the motion vector points out of the image.

3.5 Experimental setup

With the objective of being able to assess the quality of the obtained images continuously, an experimental test system has been developed based on the generation of low-resolution images to feed the super-resolution algorithms. These low-resolution images are obtained from high-resolution test images either by the application of controlled shifts followed by a sub-sampling stage or by merely sub-sampling a video sequence in order to incorporate aliasing. In a first stage, the objective was to perfectly control as much parameters as possible and to evaluate the impact of each one in the super-resolution process (aliasing, motion vectors accuracy and precision, type of movement in the scene, etc.). In this way, the shifts are intended to simulate the movement that the acquisition system will notice under real conditions, and the sub-sampling tries to emulate the image acquisition using sensors and lenses without anti-aliasing filters. The induced shifts can be done with ½ or ¼ pixel precision, depending on the precision that the motion estimation can reach. In that sense, the

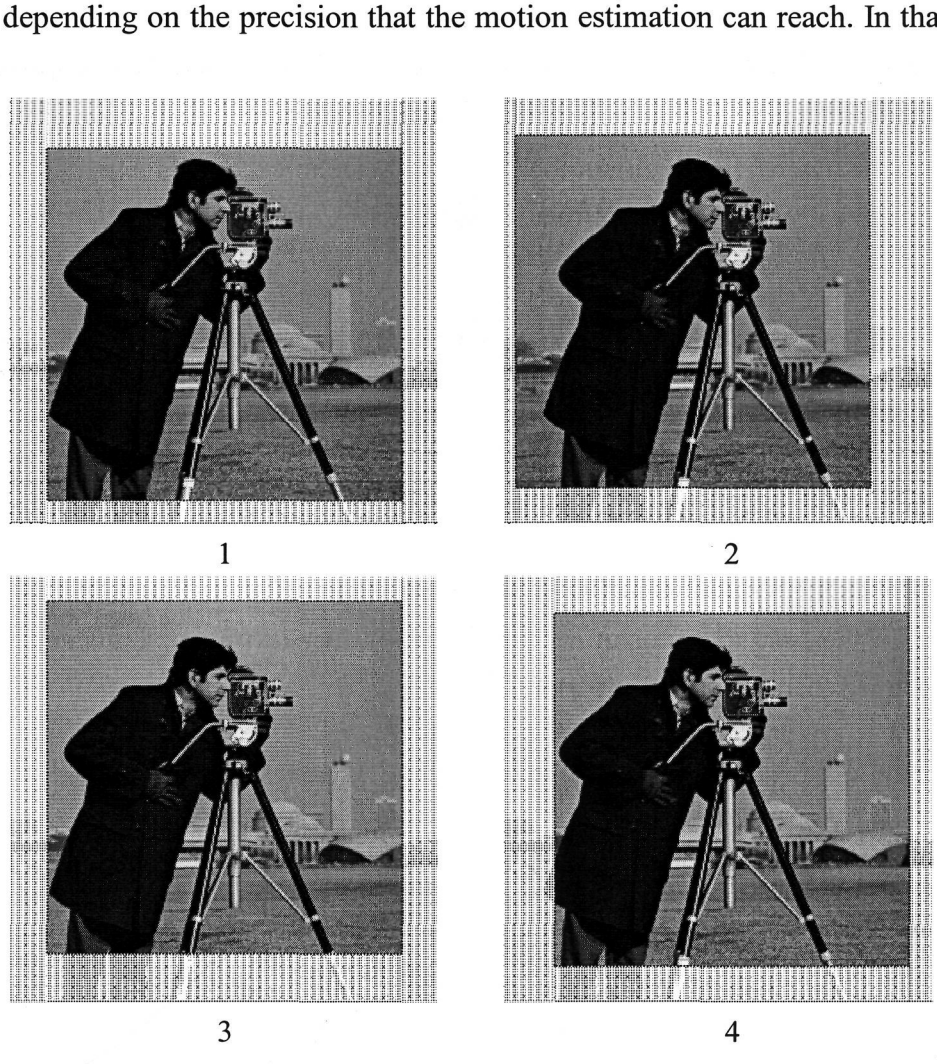


Figure 8. Four frames of the scene 'cameraman' sampled from different positions.

former versions used ½ pixel shifts and the last ones ¼ pixel shifts.

Another important issue has been to obtain an artificial motion model. In such way, the following possibilities have been enabled:

a. *Manually established shifts by the experimenter.* This is a way to meticulously analyse the correct behavioural of the motion estimation and to control how shifts affect to the obtained image quality.

b. *Random shifts.* These kinds of shifts aim to determine the behavioural of the system to cope with unknown shifts, uncorrelated among them and erratic. This is not the kind of shifts normally found in real acquisition systems, but it reflects an extreme case of system working.

c. *Bounded random shifts with zero average.* This method pretends to model the kind of movement that a domestic video camera will generate when it is manually hold by an operator. Normally, a human been will try to keep the video camera centred over the target regardless of the involuntary movement produced trying to hold the camera without using a tripod or another fastening technique. This attempt to keep the objective centred will generate relative shifts with zero average.

3.5.1 Artificial motion and aliasing

For the generation of the test image sequences with motion and aliasing we can follow any of the schemes proposed in the Figure 9. The scheme of Figure 9 (a) places firstly the shifter and secondly the decimator, while the scheme of Figure 9 (b) makes the opposite. In our case, the first scheme has been chosen because it easies the generation of sub-pixel shifts. In that sense, in the first scheme shifts *dx, dy* are introduced in pixels units and fractions of pixels are obtained thanks to the decimator.

Initially the scheme of Figure 9 (a) has been followed, with K=2 and manually generated shifts to cover the four pixels position, in order to facilitate as much as possible the super-resolution image reconstruction, as all the input data are present. This also means that the motion vectors precision will be of half-pixel; with K=4 a quarter-pixel precision will be achieved. The generated shift values (in pixels) are shown in Table 1, referred as shown in Figure 10. This vector set, which covers the four pixel positions in a base cell, will be called canonical vectors.

The shift of a pixel in the high-resolution image implies the shift of half-pixel in the low resolution image after decimation, which was the initial precision of the motion estimator.

dx	dy	Positions related to a base cell
0	0	Upper-left pixel
1	0	Upper right pixel
0	1	Lower left pixel
1	1	Lower right pixel

Table 1. Shifts generated at pixel level to cover all pixels in a base cell of size 2×2 pixels or canonical vectors.

The decimation process initially consisted in the alternative selection of one pixel out of every four pixels block, generating a low-resolution output image containing spatial aliasing. This process was repeated for every pixel of the base cell thus generating four low-resolution output images for every high-resolution input image what were used to reconstruct the super-resolution image in the iterative stage. For the non-iterative algorithms just the (0,0) pixel is picked in order to generate aliasing in the low-resolution sequence.

The iterative algorithms initially designed uses the more detailed scheme shown in Figure 11 for the test images generation. Every high-resolution image of size M×N was shifted as depicted by the canonical vector set and every shifted image was decimated to obtain a set of four low-resolution images with aliasing and quarter size than the original (half the size in each direction, i.e. M÷2×N÷2). These low resolution images set is introduced in the super-resolution algorithm that must obtain a high-resolution output image of double size than the input images in the horizontal and vertical directions, i.e. M×N.

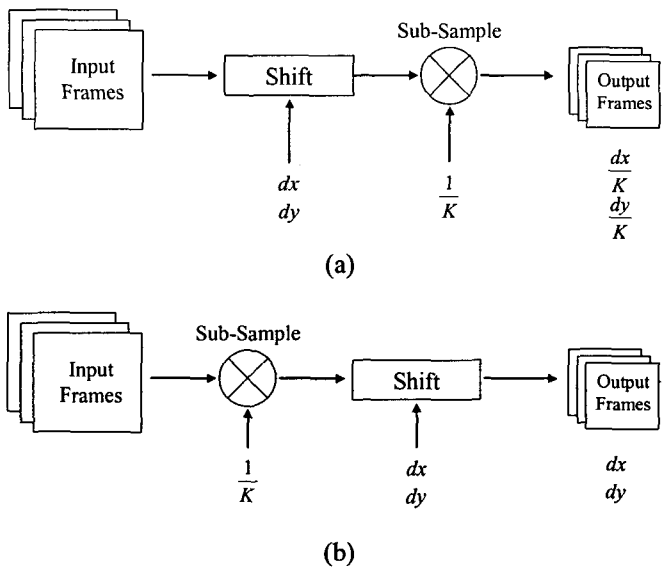


Figure 9. Two schemes for the generation of test images sequences: (a) placing the shifter first and (b) placing the decimator first.

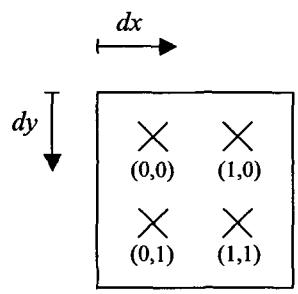


Figure 10. Base cell of size 2×2 pixels.

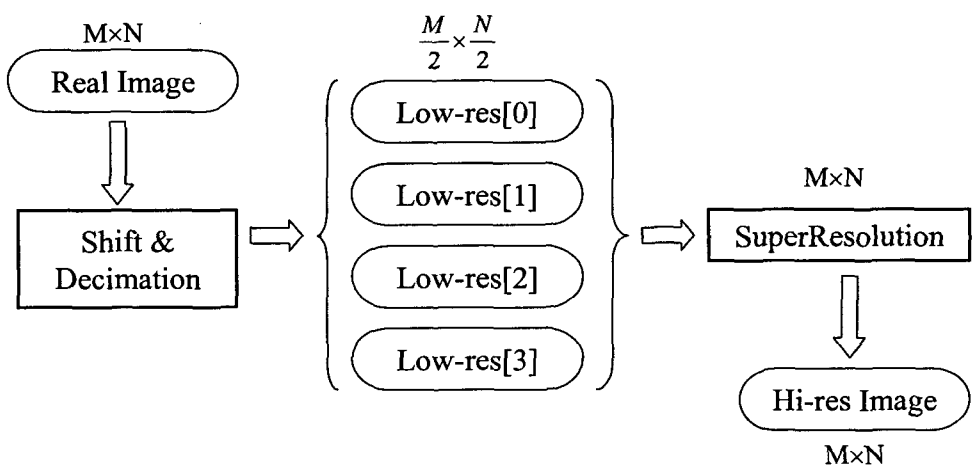


Figure 11. Scheme followed for the generation of test images.

3.5.2 Borders handling in the experimental setup

The method above mentioned for the generation of test images exhibits a serious problem in the image borders, which causes an important drop in the quality metrics. Depending on the generated vector, is almost impossible to avoid a loss of data in the image borders when shifting the reference image. A shift using the (-2,5) vector of high resolution pixels produces a data loss in the upper and right borders, as it can be seen in Figure 12, where this effect has been overstated for clearness. When shifting left and down the Picasso self-portrait, the only solution to avoid discontinuities is to replicate the image borders. These replicated data are going to hinder the reconstruction process of the borders of the original image.

The first solution is not to take into account the borders when computing the quality metrics of the images. The size of the borders excluded from the merit figure computation is of one macro-block, as the generated motion vectors are bounded to a lower value.

Nevertheless, another solution for the borders problem is to alter the process for the test images generation. Obviously, the borders replication solution does not model at all the real image acquisition process. As a matter of fact, if a real image acquisition system is moving, this will acquire new information for the real world that, at this level, is continuous. So, what we have done is to start from bigger images, shift them as previously seem and then crop them to keep the central part. With this method we avoid the undesired border effect. In Figure 13 is shown the modified diagram for the test images generation. In contrast to the previous system, which lumps together all the process in a single program, the new system is entirely modular and is formed by five modules for the images transformation. Theses are: `make_sequence`, `move`, `crop`, `subsample` and `downsample` and are shown as

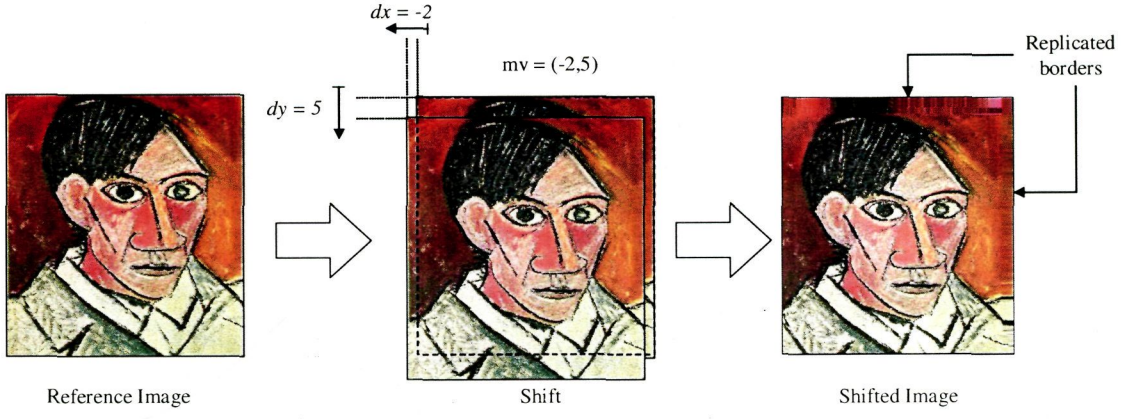


Figure 12. Borders replication when the image is shifted out of the boundaries.

shadowed rectangles. The `make_sequence` module selects the frames from the input sequence that are going to be incorporated in the test sequence. It is necessary to include such a module as some images are obtained using a Philips Vesta video camera and no tool was available to cope with raw images in the YCbCr format that allows the selection of the appropriate frames to carry out the experiments. Moreover, when recording images in a big format, sometimes the system suffers from saturation (specially the bus) giving as a result 'frame_skips' that must be removed later with this utility.

The `move` module shifts every input frame, using the vectors provided in the text file `shifts` or generating itself the shifts randomly (with or without zero mean). At the same time, it generates a log file with the shifts finally applied.

As previously mentioned, it is desirable to use an image format big enough to be cropped later, thus avoiding the border effect. In Table 2 are shown the most image sizes and its macro-block equivalence. Among these, the Video Gate Array (VGA) format has been adopted, as it is big enough for our purposes and it is the biggest format that the camera can deliver.

	SQCIF	QCIF	CIF	VGA	4CIF	16CIF
Pixels	128×96 (=12288)	176×144 (=25344)	352×288 (=101376)	640×480 (=307200)	704×576 (=405504)	1408×1152 (=1622016)
Macro Blocks	8×6 (=48)	11×9 (=99)	22×18 (=396)	40×30 (=1200)	44×36 (=1584)	88×72 (=6336)

Table 2. The most frequent image sizes and its macro-block equivalence.

Nevertheless, the VGA format has an important drawback: as it can be divided by two, maintaining an integer amount of macro-blocks (640÷2=320 pixels, i.e. 20 macro-blocks and 480÷2=204 pixels, i.e. 15 macro-blocks) , it can not be divided by four, maintaining an integer number of macro-blocks (640÷4=160 pixels, i.e. 10 macro-blocks and 480÷4=102 pixels, i.e. 7.5 macro-blocks).

This problem has been solved defining a new image format named Adapted VGA (AVGA) of size (576×448). This new format can be divided by two and four, maintaining an integer number of macro-blocks, as shown in Table 3. So, the followed procedure consists in

first to shift the VGA image, no matter what can occur in the image borders, and subsequently to crop the borders to obtain an AVGA size. This new image is not affected by the border effect and besides it can be divided by two and four. The requirement of being able to divide the image by four is imposed by the necessity of obtain images with shifts of ¼ pixel in low resolution. I.e., shifting the image an integer pixel amount in VGA will result in a quarter-pixel shift in Quarter Adapted VGA (QAVGA.).

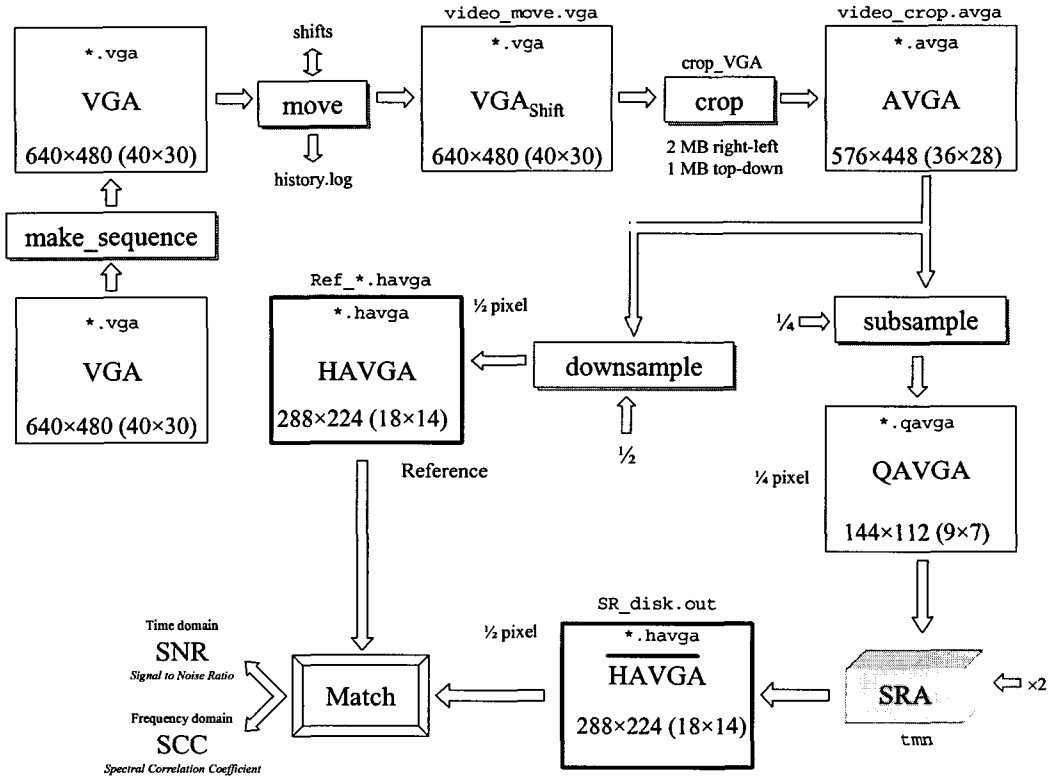


Figure 13. Modified diagram for the test images generation.

	VGA	AVGA	HAVGA	QAVGA
Pixels	640×480 (=307200)	576×448 (=258048)	288×224 (=64512)	144×112 (=16128)
Macro-blocks	40×30 (=1200)	36×28 (=1008)	18×14 (=252)	9×7 (=63)

Table 3. New image sizes based in the VGA format.

The module in charge of cropping the image borders, adapting its size to AVGA is `crop`. This module removes two macro-blocks from the left and right borders of the image (passing from 40 to 36 macro-blocks wide) and one macro-block from the upper and bottom borders (passing from 30 to 28 macro-blocks height).

From the shifted and size-adapted video sequence is performed a decimation of factor 4 with a sample frequency below the Nyquist frequency in order to produce a new video sequence with aliasing and QAVGA size. This sequence is the input to the super-resolution algorithm and it is generated from the `subsample` module.

Also from the AVGA sequence is performed a decimation of factor two, but this time using a low-pass filter to accomplish the Nyquist sample principle and thus avoiding image aliasing. This new video sequence of Half AVGA (HAVGA) size is the reference sequence because the super-resolution algorithm generates images of double size respect to the input size –i.e. from QAVGA it will generate HAVGA– that cannot be compared against the original AVGA sequence. The decimation process of factor two to obtain the reference sequence is performed by the `downsample` module.

The super-resolution algorithm delivers as a result a reconstructed sequence of size HAVGA, that must be compared with the reference sequence in order to obtain different quality parameters that allow us to evaluate the algorithm performance.

3.5.3 Quality metrics

We need metrics that allow us to measure the functioning of the algorithm in terms of the higher or lower similitude between the resulting image and the original image, here referred as the real image. This is depicted in Figure 14 at the pixel level. To better illustrate the measurement procedure we will start with the simple case of artificial motion and aliasing. The real image or reference is shifted and decimated as it has been previously shown, producing four low resolution images when the canonical vector set is applied. For instance, when the shift vector (0,0) is applied, and subsequently decimated, the corresponding low resolution image contains all the pixels represented in the Figure 14 as diamonds. The (1,0) vector gives as a result a low resolution image formed by the triangle-shaped pixels, the (0,1) vector gives as a result the low resolution image formed by the hexagonal-shaped pixels and finally, the vector (1,1) gives the low resolution image formed by the cross-shaped pixels.

As it can be seen in Figure 14 (a) the super-resolution algorithm provides a high resolution image formed by pixels $x_{1,2,3,4}$ of the basic cell, that ideally and in the best case they must be the same as the ones in the basic cell of the real image. Figure 14 (b) shows the location process of the pixels from a basic cell in an image using the described vector set. The quality of the super-resolution process will be given by the similitude degree between the super-resolution image and the real or reference image.

For the quality analysis, both in the space and in the frequency domain, we have opted for using the mathematical package MATLAB. In that sense, a set of functions have been developed to obtain different figures of merit in an automatic way once the super-resolved images have been processed. These two figures of merit are the peak signal to noise ratio and the spectral cross-correlation coefficient.

In order to avoid the dependences of the metric figure with the image variations,

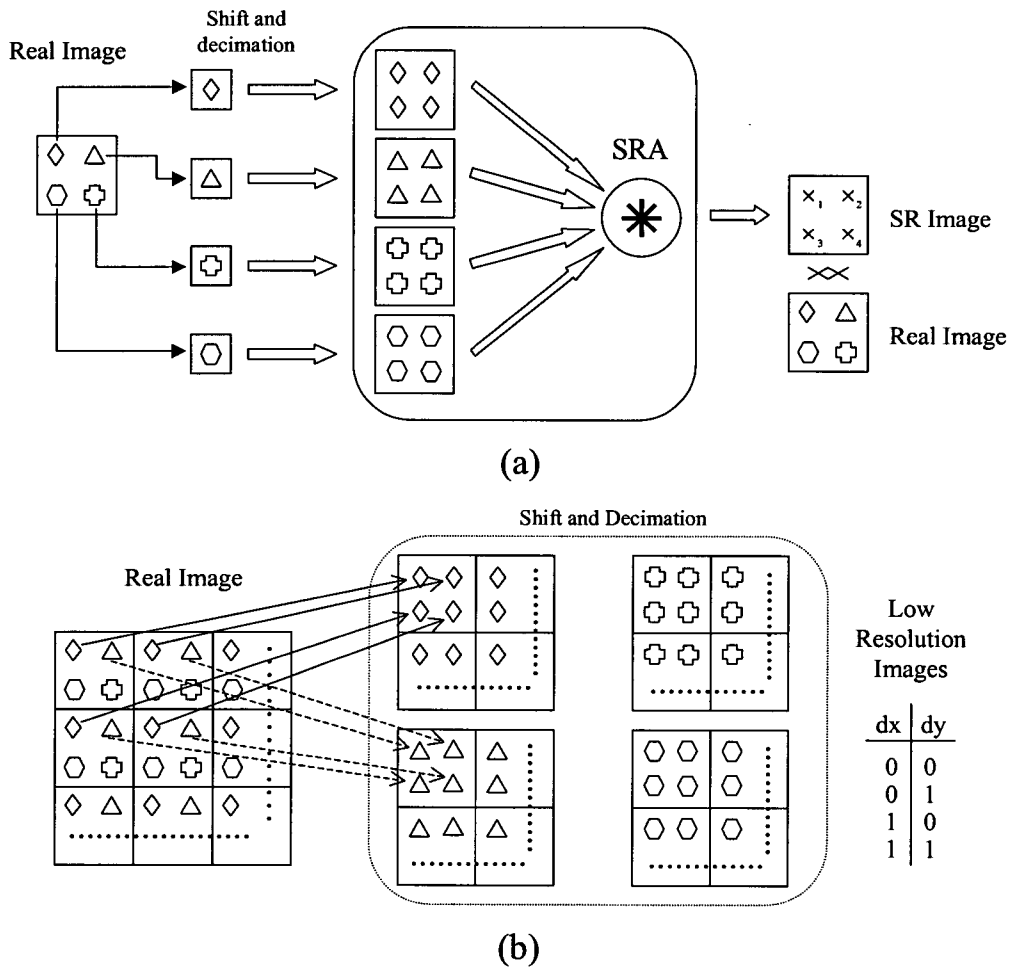


Figure 14. Scheme followed for the generation of test images at the pixel level.

instead of using the signal to noise ratio, in image processing applications is more typical to use the Peak Signal to Noise Ratio (PSNR) for comparisons between images. The PSNR is given in equation (14), assuming images of size M×N, where *R(i,j)* is the reference image and *S(i,j)* is the super-resolution image.

$$PSNR = 10 \cdot \log_{10} \left(\frac{255^2}{\frac{1}{M \cdot N} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} (R(i,j) - S(i,j))^2} \right) \tag{14}$$

The 255 value comes from the maximum value that a pixel can reach when it is stored in an eight-bit word, i.e. (2^8-1). The signal to noise ratio is one of the preferred metrics when measuring the quality of a signal and that is why it has been the metric adopted for images in the spatial domain.

When a big image is obtained from a smaller one, as we are doing in the super-resolution process, it can be objected that the same objective could be reached by interpolation. Among the different existing interpolation methods [PKT83], [Jai89], [PRA91] we have opted for the nearest neighbour interpolation, as it is the easiest an faster way in real time applications, and the bilinear interpolation because with relative additional complexity a substantial improvement in the interpolated image is achieved. During all the work the super-resolution results will be always compared against the interpolated images obtained with both methods. A quality above the interpolation level means a super-resolution improvement.

The next metric figure used has been the spectral cross-correlation coefficient, which will be applied in the frequency domain instead of in the time domain. Thereby, as the Fourier transform is a complex magnitude, there will be a cross-correlation for the magnitude and another one for the phase. The first step is to obtain the two-dimensional Fourier transform of both images: $\tilde{R}(u,v)$ and $\tilde{S}(u,v)$, as indicated in equation (15).

$$\begin{aligned} \tilde{R}(u,v) &= \mathrm{F}\{R(x,y)\} \\ \tilde{S}(u,v) &= \mathrm{F}\{S(x,y)\} \end{aligned} \tag{15}$$

Then, the spectral cross-correlation coefficient for the magnitude (*sccm*) will be given as expressed in equation (16). Previous to the calculus of the *sccm* it is necessary to compute the mean value of the magnitude of both images as can be seen in (17) and (18).

$$sccm = \frac{\sum_{u=0}^{M-1}\sum_{v=0}^{N-1}\left(\left(\left|\tilde{R}(u,v)\right| - \overline{\left|\tilde{R}\right|}\right)\cdot\left(\left|\tilde{S}(u,v)\right| - \overline{\left|\tilde{S}\right|}\right)\right)}{\sqrt{\sum_{u=0}^{M-1}\sum_{v=0}^{N-1}\left(\left|\tilde{R}(u,v)\right| - \overline{\left|\tilde{R}\right|}\right)^2 \cdot \sum_{u=0}^{M-1}\sum_{v=0}^{N-1}\left(\left|\tilde{S}(u,v)\right| - \overline{\left|\tilde{S}\right|}\right)^2}} \tag{16}$$

$$\overline{\left|\tilde{S}\right|} = \frac{1}{M \cdot N}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1}\left|\tilde{S}(u,v)\right| \tag{17}$$

$$\overline{\left|\tilde{R}\right|} = \frac{1}{M \cdot N}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1}\left|\tilde{R}(u,v)\right| \tag{18}$$

The same formula is used for the spectral cross-correlation coefficient in phase (*sccp*), using equations (19), (20) and (21).

$$sccp = \frac{\sum_{u=0}^{M-1}\sum_{v=0}^{N-1}\left(\left(\angle\tilde{R}(u,v) - \overline{\angle\tilde{R}}\right)\cdot\left(\angle\tilde{S}(u,v) - \overline{\angle\tilde{S}}\right)\right)}{\sqrt{\sum_{u=0}^{M-1}\sum_{v=0}^{N-1}\left(\angle\tilde{R}(u,v) - \overline{\angle\tilde{R}}\right)^2 \cdot \sum_{u=0}^{M-1}\sum_{v=0}^{N-1}\left(\angle\tilde{S}(u,v) - \overline{\angle\tilde{S}}\right)^2}} \tag{19}$$

$$\overline{\angle\tilde{R}} = \frac{1}{M \cdot N}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1}\left|\tilde{R}(u,v)\right| \tag{20}$$

$$\overline{\angle\tilde{S}} = \frac{1}{M \cdot N}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1}\left|\tilde{S}(u,v)\right| \tag{21}$$

Values close to unity in both cases (magnitude and phase) will denote a high similitude of the spectrums. Although close-to-one values are desirable, for visual human perception are more important higher values in the magnitude than in the phase. Moreover, the magnitude Fourier transform is insensitive to image shifts which is also in correspondence with human perception and enables more reliable measures. The shifts between the image reference and the super-resolution image will have more impact in the phase.

3.5.4 Picture repetition

The signal to noise ratio has the important drawback of being image dependent, i.e. its value depends on the image entropy. Images with lower entropy will exhibit higher PSNR and vice versa. This fact can be a problem when assessing the super-resolution algorithm because in a video sequence every frame is different, what precludes a reliable PSNR comparison among frames. As too many factors have influence on the image quality, it is necessary to apply the usual method of divide and conquer. Keeping the same image all across the video sequence, although with different shifts and aliasing degrees will enable the comparison of the PSNR among frames.

3.6 Conclusions

In this chapter, the basis of the super-resolution theory for iterative algorithms has been established. This first algorithm constitutes a starting point for our super-resolution approaches as far as it can be adapted to be executed onto a hybrid video encoder.

From the study of the super-resolution basis and taking into account some practical aspects, three main limitations have been found: i) the amount of aliasing in the input image drastically limits the maximum quality that the algorithm can deliver. If most of the high frequencies are suppressed by means of any low-pass anti-aliasing filter, it will be impossible to incorporate that missing information in the super-resolved image; ii) the precision of the motion estimation has also important effects on the final image quality. As we are limited to use the classical block-matching motion-estimator used for image compression, some adjusts must be made in the estimated motion vectors to better match the real motion of the scene; iii) as it is usual in many areas of image processing, borders suppose a discontinuity in the image process. In this case, it is more difficult to find pixel related information from other pictures in the time if the pixel is at a border instead of in the central region of the image.

Finally, the experimental setup is described, starting with an artificial procedure to generate motion and aliasing. If those two crucial parameters known a priori (which is almost impossible for real sequences), then it will be easier to understand their exact influence on the results. The use of replicated images isolates the quality metrics among frames from the specific characteristics of specific images, enabling a reliable comparison of the image qualities. Finally, some figures of merit are proposed, among the more usual found in the

specialised scientific literature: the peak signal to noise ratio and the spectral correlation in magnitude and phase.

Chapter 4

Each problem that I solved became a rule which served afterwards to solve other problems.

René Descartes (1596-1650), "Discours de la Méthode"

The Video Encoder Platform

4.1 Introduction

Philips is an important player in the market of Personal Computers (PC) cameras. The increasing demand for performance for PC cameras (VGA at 30 frames per second), and the apparition of new camera applications (such as wireless and detachable cameras), requires the use of video encoding (also called compression) inside the camera. Furthermore, the existence of several different encoding standards (such as JPEG, H.261, H.263, H.264, MPEG1, MPEG2 and MPEG4) requires a multi-standard approach.

The main objective of the Picasso project [PKL+99], where the super-resolution algorithm is intended to be executed on, is to apply research results in order to provide Philips with a state of the art video encoder Integrated Circuits (IC) for consumer (low cost and low power) camera applications.

Research results in the field of embedded compression allow a DCT domain video encoder [KVL99], leading to a considerable reduction of the image memory. Due to this reduction, it will be possible to place the image memory on-chip. This results in a significant

reduction in both system cost (less components), and in power dissipation (less memory bandwidth and only on-chip connections to the image memory).

The flexibility will be achieved by applying the results of the CPU-controlled Heterogeneous Embedded Architectures for signal Processing (C-HEAP) research project [LN97]. C-HEAP enables the system architect to make hardware and software trade-offs. It provides functionality to partition the design into hardware and software at different levels of abstraction, and to determine the impact of this partitioning on the performance. By mapping the compute intensive and inflexible parts of video encoding onto hardware, while keeping the flexible and encoding-standard-related parts in software, we address the multi-standard aspect.

Super-resolution is made on top of the Picasso platform, and it is intended to be an added-value to the compression features (for instance, performing digital zooming without mechanical-optical parts). A big effort has been made mapping every super-resolution algorithm in the existing Picasso architecture. Every time a new operation has been necessary to be incorporated, it has been carefully done to minimize its cost in the overall system, especially in terms of delays, memory requirements, minimize changes in the original architecture, impact in the compression working, etc.

4.2 Applications

There are two more potential application domains for the Picasso project:

- Hand-held multimedia terminals are expected to incorporate videophone functionality in the near future. Super-resolution can improve the quality of the decoded image with no band-wide penalty.
- Surveillance cameras could be based on a number of detachable cameras connected to an existing low bandwidth network. Each camera records a video sequence onto its local storage medium. The security operator then connects to the required camera, and downloads its recorded video sequence. Super-resolution could be of interest in this area, improving the quality of the regions of interest.

4.3 Video Encoding

Several video encoding standards have been (and are being) defined, like MPEG1, MPEG2, MPEG4, H.261, H.263, H.264, MPEG-4 part 10, MPEG-4-AVC and JPEG. All these encoding standards are based on the discrete cosine transform (the JPEG2000 standard is based on a different transform, and is not targeted by the Picasso project), and all but the JPEG standard, are based on motion compensated prediction. Therefore, all these standards can be implemented on a hybrid encoder.

The Picasso design is a hybrid encoder that can be programmed for several encoding standards, namely: MPEG4 SP@L3, JPEG and H.263.

4.3.1 Hybrid Encoder

A hybrid encoder combines compression in the spatial domain with compression in the temporal domain. Spatial domain compression is achieved by means of a decorrelating transform (a two-dimensional discrete cosine transform), and temporal domain compression by using motion compensated prediction.

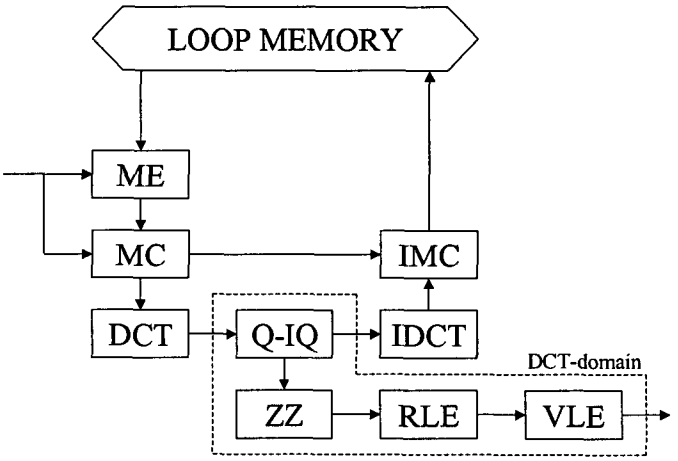


Figure 15. A typical hybrid encoder scheme.

Figure 15 shows an overview of a conventional hybrid encoder. It contains the following blocks: Motion Estimator (ME), Motion Compensator (MC), Inverse Motion Compensator (IMC), Discrete Cosine Transform (DCT), Inverse Discrete Cosine Transform

(IDCT), Quantizer (Q), Inverse Quantizer (IQ), Zig-Zag transform (ZZ), Run length Encoder (RLE), Variable Length Encoder (VLE), and a loop memory. The ME, MC and IMC blocks are responsible, together with the loop memory, for the temporal domain compression.

The ME block determines the best match of the current luminance macro block (16 by 16 pixels) in the reconstructed image stored in the loop memory. This is done by evaluating several candidate vectors in order to determine the vector with the smallest cost function (e.g. the sum of absolute difference, or the sum of squared difference). The difference between the current macro block and the best match in the reconstructed image is computed (MC), transformed (DCT), quantized (Q), coded (ZZ, RLE, and VLE), and transmitted to the decoder. The transmitted data is inverse quantized (IQ), inverse transformed (IDCT), reconstructed (IMC), and stored in the loop memory to encode the next picture.

The ME block is the most compute intensive part. In the case of the Picasso design, 23 candidate vectors are evaluated. This means that the loop memory is accessed 23 times, followed by a single execution of the remaining loop. From a power point of view it is therefore essential to concentrate on the power dissipation of the ME part.

A disadvantage of a conventional hybrid encoder is that the size of the loop memory is large, usually resulting in a two-chip solution: the encoder IC and the loop memory [WWV+97]. In the Picasso design this disadvantage is addressed by compressing the loop memory in order to embed it on the encoder IC, leading to a single-chip solution. The size of the loop memory has to be large enough to store the largest image to be encoded with motion compensation.

In case of intra coding, only the DCT, Q, ZZ, RLE, and VLE blocks are used. No other blocks are used, and no image is stored in the loop memory.

4.4 Video compression algorithms

The main disadvantage of a hybrid encoder in terms of implementation is that it requires a two-chip solution: one encoder chip for the core, and a memory chip. This is due to the large size of the loop memory (3.5 Mbit for a VGA stream). However, recent research

results [KVL99] have shown that it is possible to compress the picture stored in the loop memory by a factor of four. This has the following advantages:

- The resulting memory size allows a single-chip solution containing an embedded memory for the loop memory.
- The memory bandwidth is reduced considerably.

This technique is called embedded compression [KVL99], and is based on two principles:

- The loop memory is in the DCT domain. This implies that a larger part of the encoder is in the DCT domain.
- The DCT coefficients are compressed using a scalable coding approach in order to fit in the reduced loop memory.

4.4.1 DCT domain memory

Since the loop memory is organized in 8×8 blocks, reading a 8×8 block from memory at a displaced position becomes more complex. This is graphically explained in Figure 16.

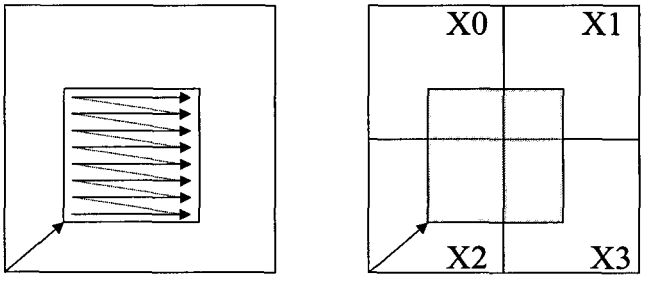


Figure 16. Reading an 8×8 block from pixel and 8×8 block oriented memories.

A conventional encoder has a pixel oriented loop memory. Reading a displaced block from memory is carried out by doing the read actions at displaced addresses. However, for a 8×8 block oriented memory four 8×8 blocks (X_0 to X_3) have to be read from memory. The displaced 8×8 block has to be computed from these four 8×8 blocks. Reading the luminance part of a macro block (16 by 16 pixels) from memory (four 8×8 blocks) requires nine 8×8 blocks to be read, as shown in Figure 17.

4.4.2 Scalable coding

A conventional encoder set-up is based on storing the reconstructed image in the spatial domain. This requires exactly 8 bits per pixel. In the DCT domain a coefficient consists of 12 bits. Therefore, reducing the memory size by a factor of four in comparison with a conventional encoder requires a compression of a factor of six for coefficient data. This leads to an (average) size of 2 bits per coefficient.

The objective of the scalable coding is to compress the 8×8 blocks before writing them into the loop memory. Due to the fixed size of the loop memory, two cases can arise:

- The first case is that the coefficients fit in the loop memory. In this case the compression is lossless, and no information is lost.
- The second case is that the coefficients do not fit in the loop memory. The encoded coefficients have to be reduced even further in order to adapt them to the available memory size. This implies a lossy compression, resulting in a loss of information.

From these two cases the following requirements for the scalable coding can be extracted:

- **High compression ratio**: The higher the compression ratio of the scalable coding is, the better, since the encoded coefficients are more likely to fit into the loop memory. This will lead to a more lossy compression.
- **Good scalability**: Due to the fixed size of the loop memory, it is very likely that the encoded coefficients do not fit in the loop memory. The size of the encoded coefficients is then further reduced by truncating the information to be stored. In

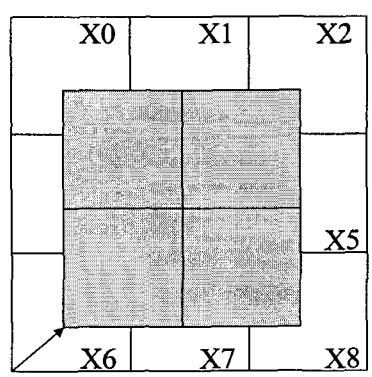


Figure 17. Reading a macro block from 8×8 block oriented memories.

this case it is important that the least important information of the encoded DCT coefficients is truncated.

- **Ease of implementation**: Due to the fact that the scalable coding is going to be implemented in hardware, it is very important to look at the ease of implementation. Unfortunately, lossless coding techniques with the highest compression ratios are often very hard to implement on silicon.

These constraints have resulted in a bitplane-based zonal coding of the DCT coefficients. The code converts the 8×8 block coefficients into a single bit string. A high compression ratio is obtained by reducing the information to be coded. In order to obtain scalability, the bitplanes are encoded from the highest to the lowest bitplanes. In case of truncation, only information of the lowest bitplanes is lost. Ease of implementation is achieved by not applying difficult to implement techniques, such as variable length encoding. Figure 18 shows how the scalable coding works.

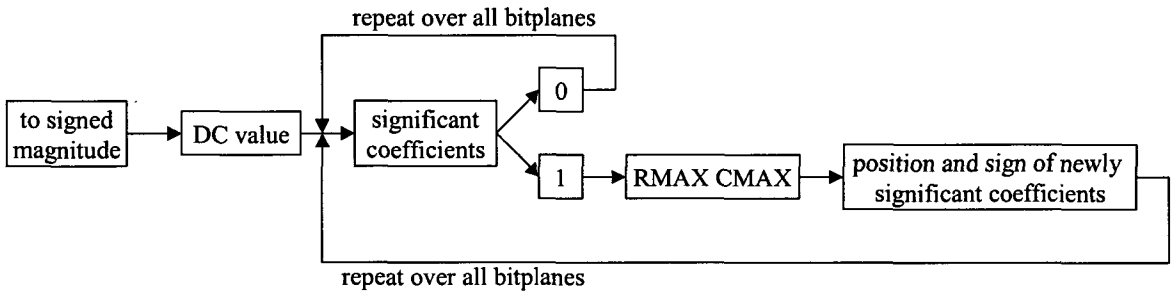


Figure 18. Scalable coding scheme.

The proposed encoding scheme provides a lossless compression resulting in on average 4 bits per coefficient. This is a reduction of 2x and 3x in comparison with uncompressed pixel (8 bits) and coefficient data (12 bits), respectively. An additional reduction of '2×' is obtained by truncating the bit-string, resulting in an average of 2 bits per coefficient. Note that part of the lower bitplanes is deleted when truncating a bit-string. Since most noise is contained in the lower bitplanes, truncating a string means that noise is removed.

The decoding scheme is simply the inverse of the coding scheme. The only complication arises when decoding a truncated bit-string. Several strategies can be followed when decoding a truncated string: adding zeros or adding random bits for the missing end of the known coefficients. Recall that the effect of truncation is that the noisy lower bitplanes are

removed. Under the assumption that this noise is randomly distributed it is clear that adding random bits at the end of the bit-string is the way to go. However, since the hardware to generate random bits for the missing ends of the known coefficients is much more complex than that to add zeros, we will follow the zero approach.

4.4.3 Motion estimation

The motion estimator is responsible for determining the best match of the current macro block in the picture stored in the loop memory. Instead of applying the brute-force approach of evaluating all possible vectors, a simpler approach is followed. This approach is based on a modified 3-Dimensional Recursive Search (3DRS) algorithm [Oli98].

The 3DRS algorithm as used in the I.McIC ([HBH+93], [WWV+97], [HB95]) is based on a set of 5 candidate vectors, carefully chosen in the neighbourhood of the current macro block, as shown in Figure 19. Let (x,y) be the coordinates of the current macro block. The following motion vectors are used as candidate vectors:

1. Motion vector of the macro block at (x-1,y-1).
2. Motion vector of the macro block at (x+1,y-1).
3. Motion vector of the macro block at (x-1,y) with a random update in the range (-1,1).
4. Motion vector of the macro block at (x-1,y) with a random update in the range (-6,6).
5. Motion vector of the macro block at (x+1, y+2).

The first four candidates are from the current image, whereas the last candidate is from the previous image.

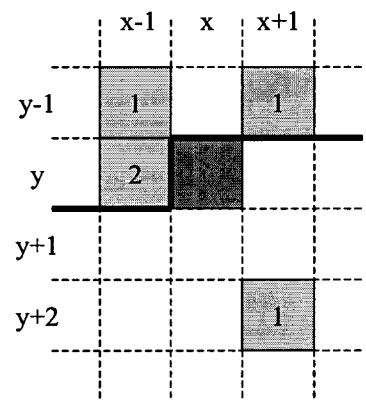


Figure 19. Candidate vectors of 3DRS.

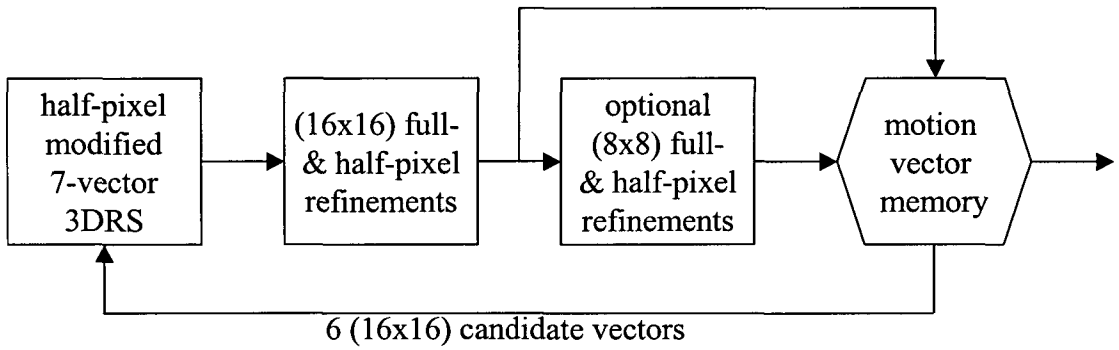


Figure 20. Organization of the motion estimator.

Figure 20 shows the resulting motion estimation set-up. Note that the refinement of the four (8×8) vectors is for the Advanced Prediction Motion (APM) option only.

4.5 Video encoding architecture

In this section, the architecture of the Picasso design will be described. Section 4.5.1 gives some background information about C-HEAP. Section 4.5.2 describes the resulting Picasso architecture in terms of hardware and software tasks.

4.5.1 C-HEAP

The following sections describe the C-HEAP architecture, communication protocol, and design flow, as used in the Picasso project.

4.5.1.1 C-HEAP architecture

C-HEAP advocates a multi-processor architecture where the processors run the tasks of the application independently and in parallel. The tasks synchronize on data- and buffer-space availability and may use on-chip memory to implement communication buffers to reduce the bandwidth requirements to off-chip memory. The individual processors should be chosen in such a way that a good trade-off between flexibility and efficiency is made. Functions that are well known and will not change, e.g. because they are standardized, can be implemented in a very efficient (but less flexible) way. Other functions require flexibility, e.g.

because uncertainties still exist or because the chip should adapt to the context in which it will operate. In fact, a gradual scale of very flexible cores (CPUs) to very rigid cores (dedicated hardware) exists. In order to facilitate a good trade-off between flexibility and efficiency, C-HEAP supports the application of all these cores. Figure 21 shows an example of such architecture. It consists of an ARM, a TriMedia, a R.E.A.L., and some dedicated hardware units. Due to the high level of parallelism in this architecture, it is very well suited for application with high computational demands.

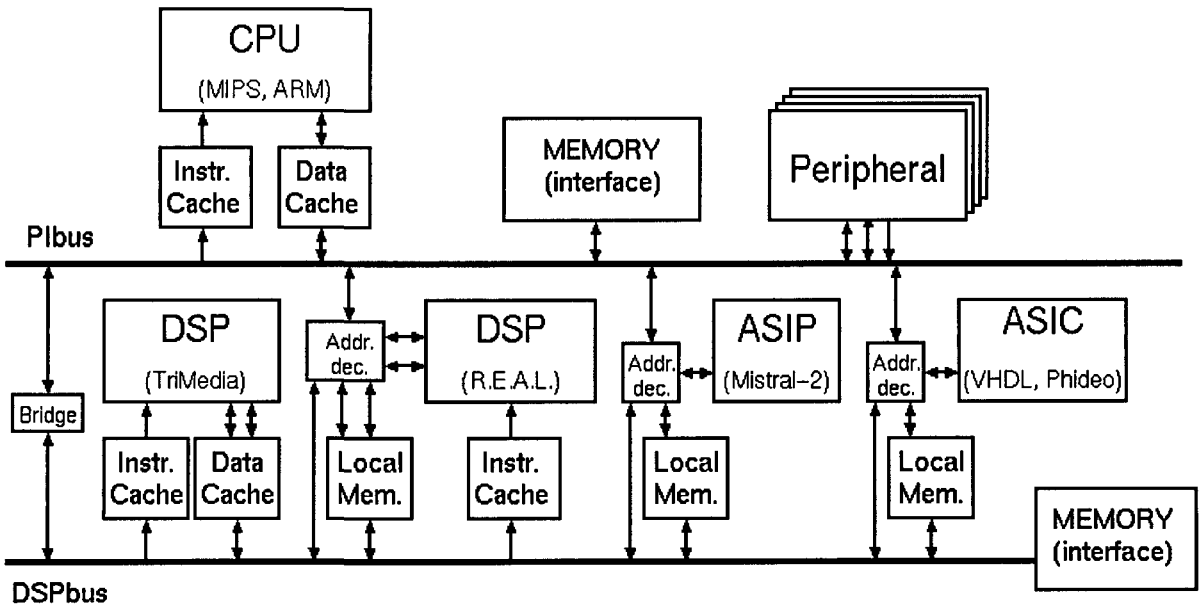


Figure 21. An example of an architecture used by C-HEAP.

4.5.1.2 C-HEAP protocol

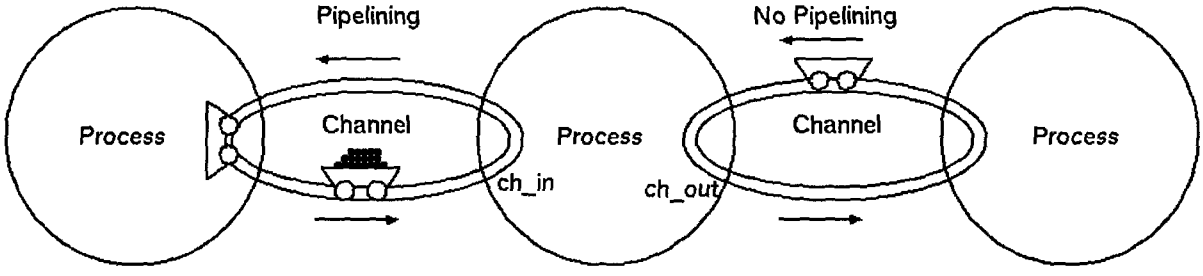
The C-HEAP protocol is intended for signal processing applications where infinite streams of data have to be processed. It is based on a model that is often referred to as process networks. In this model, the overall function is decomposed in a number of parallel processes communicating via point-to-point channels with First-In-First-Out (FIFO) behaviour. No data can be lost on these channels. When a process wants to read from a channel, and there is no data available, the process will block. Within C-HEAP, bounded FIFOs are assumed since we are addressing efficient implementation of the function and not only the specification. This means that a process will also block when it wants to write to a channel if the associated FIFO is full. Synchronization between processes is derived from the status of the FIFOs: a blocked

reading process is resumed when the writing process writes on the channel. Likewise, a blocked writing process is resumed when the reading process reads from the channel. This synchronization takes place on a per token basis. While a token is the unit of synchronization, the amount of data associated with a token can vary. It can be very large (e.g. video frames) or even zero (empty tokens). In the later case, the channel is only used for synchronization and not for communication. In order to implement the protocol efficiently on shared memory architectures, we have chosen to clearly separate synchronization from communication where, in this context, communication is the set of activities required to transport the data from one process to another. For instance, in a message passing implementation, synchronization and communication are combined in the primitives *message_send* and *message_receive*. In an implementation based on shared memory, no physical transport or copying of data is required, so only synchronization primitives are needed. Another requirement is that no dynamic memory allocation is required during operation. This means that during setup, all (communication) memory is allocated once, and reused during operation. This has to do with the fact that we are aiming at low cost, high performance solutions where we cannot afford the cost- and performance penalty of dynamic memory allocation for multi-processors.

The synchronization primitives are used to notify other processes of the input/output activity of the calling process. Concisely, a process has to perform the following synchronization actions to get input and to produce output. To get input data from an input channel, a process has to synchronize with the producing process on the input channel. This is done using the primitive *CHP_get_data* on the input channel. This primitive will return a reference (pointer) to the oldest not yet consumed token send by the producer. Note that this primitive will block when the FIFO on the channel is empty. When *CHP_get_data* "falls through", the reference to the data can be used to process the data. When the data is completely used, the process has to notify that the producing process can again use the data space (buffer). This is done by the primitive *CHP_put_space* on the input channel. Note that *CHP_put_space* does not block. On an output channel similar primitives exist: *CHP_get_space* to acquire a reference to a buffer that can be filled and *CHP_put_data* to notify the consumer at the other side of the channel that data has been produced. The primitive *CHP_get_space* blocks when the FIFO is full, the primitive *CHP_put_data* will not block.

Figure 22 illustrates the use of these four primitives. Here the buffers are visualized as train wagons and channels as railroads. A process first has to acquire a full wagon at its input channel and an empty wagon at its output channel. After the processing, an empty wagon is

pushed on the input channel and a full one on the output channel. The number of wagons initially put on the railroads determines how strongly processes are coupled. Only one wagon means that the processes have to be executed alternately, more than one wagon allows pipelining (parallel execution) of the processes.




```
Data *f, *e;
while (1) {
   f = CHP_get_data(ch_in);
   e = CHP_get_space(ch_out);
   process(f,e);
   CHP_put_space(ch_in);
   CHP_put_data(ch_out);
}
```

Figure 22. The four C-HEAP synchronization primitives.

4.5.2 Picasso implementation

One of the most important implementation issues is the trade-off between hardware and software. Doing everything in software it results in a maximum of flexibility, but at the expense of a high power dissipation and low performance. On the other hand, implementing everything in hardware leads to a low power and high performance architecture, but with almost no flexibility. In other words, a good balance between hardware and software leads to an optimal architecture in terms of power dissipation, performance, and flexibility.

4.5.3 Hardware-software partitioning

The hardware-software partitioning was done in several steps:

- The first step taken was to determine the parts of the encoding algorithm suited to be implemented in hardware. The combination of high performance requirements and no (or

few) flexibility constraints characterize these parts. This resulted in the (I)DCT, (I)MC, (I)Q, DISP3, ZZ, RLE, DEC, and ENC blocks.

The ME block was divided into two parts: vector selection and vector evaluation. Vector evaluation is also a candidate to be implemented in hardware, whereas the vector selection requires flexibility and is more likely to be done in software.

Besides of these, two additional hardware blocks were identified: the border generation, and the stream processor. The border generation block is responsible for generating an artificial border around the image by simply copying the edge pixels. This block is needed to support two options of the H.263 standard: the Unrestricted Motion Vectors (UMV) and the Advanced Prediction Mode (APM). The stream processor is responsible for concatenating the output bits of the VLE, since this functionality is very nasty to implement in software.

- Then, any flexibility (e.g. encoding standard dependent parameters) in these hardware blocks was moved to software. A good example is vector clipping. Since the maximum size of the motion vectors is encoding standard dependent, the maximum vector size is determined by software and passed to hardware as a parameter. The vectors are then clipped in hardware, based on this parameter.
- The next step was to group these blocks by combining neighbouring blocks into larger coprocessors. This has lead to three coprocessors:
  1. The motion estimator (DEC, DISP, and ME vector evaluation).
  2. The texture processor (DEC, ENC, DISP, (I)MC, (I)DCT, (I)Q, ZZ, RLE, and border generation).
  3. The stream processor.
- Then the C-description was divided into processes, one process per coprocessor, and one process for the software running on the ARM.
- The resulting model was extensively simulated in order to see if the performance requirements are met. These simulations showed that implementing the VLE of the coefficients in software (which is good from a flexibility point of view, since each encoding standard has different VLE tables) resulted in a poor performance (VGA at less than 30 frames per second). Therefore, the following approach was chosen. The VLE of the coefficients has to be done in hardware, and has been incorporated in the texture processor, resulting in meeting the performance constraint of 30 frames per second. The set-up of the hardware VLE has to be as flexible as possible (downloading of VLE tables in a content addressable memory [KWS+99]). However, as a fall back scenario, the hardware VLE of the coefficients can be switched off, to be able to do it in software. In this way, it is possible to support a completely different VLE, at the expense of lower

performance.

This has lead to the partitioning shown in Figure 23. In this figure the hardware blocks are represented as rectangles, and the software tasks as circles.

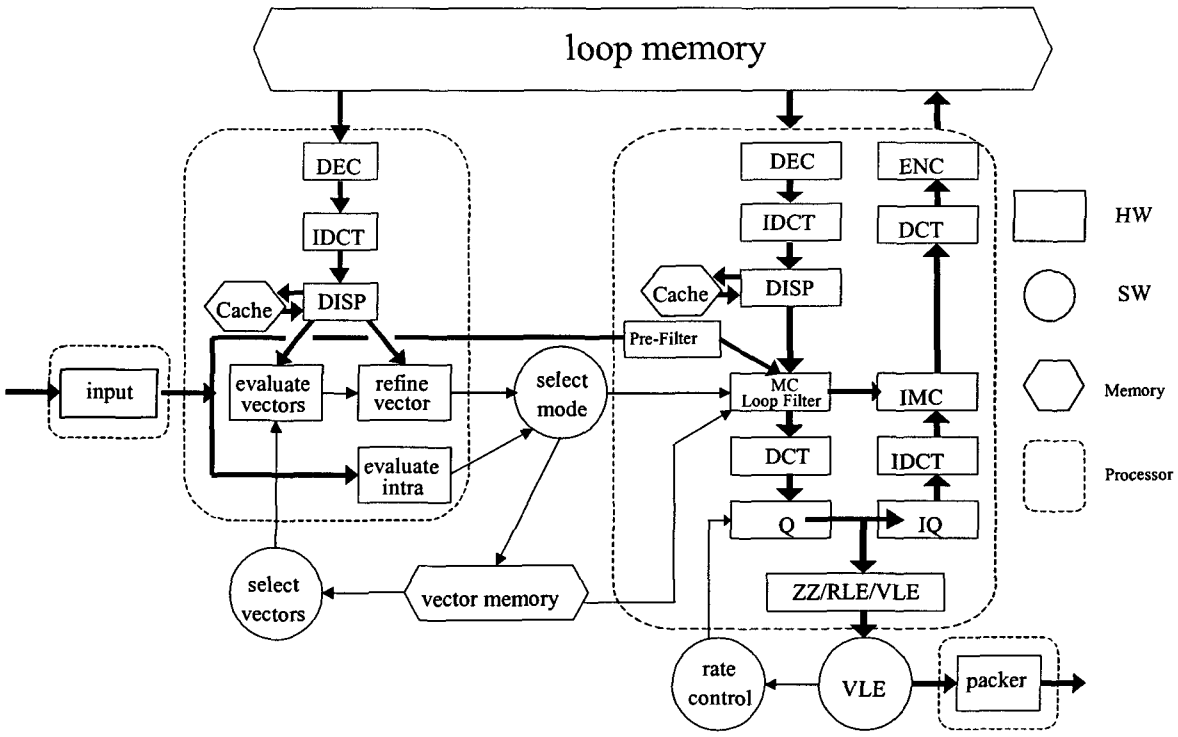


Figure 23. Result of hardware-software partitioning.

Figure 24 shows the final architecture. Besides the previously mentioned coprocessors, a pixel processor has been added. An ARM (Advance RISC Machine) has been used as microprocessor. Due to the large bandwidth to the loop memory, it was necessary to use two memories. One memory is used for the microprocessor (code, data, etc.), while the other is dedicated for pixel data exclusively. A double bus architecture has been used. Both busses are based on the PI bus protocol, and two Bus Control Units (BCU) are needed to control them. The upper bus (will be called the PI-bus) is used for the communication from and to the ARM (program data, variables, motion vectors, and handshake with coprocessors). The ARM is defined as the default master on that bus. The lower bus (will be called the DSP-bus) is used for the communication from and to the image memory (input buffer, loop memory, output buffer). Due to the large bandwidth requirements of the motion estimator, that coprocessor is defined as the default master on that bus. Furthermore, a bridge has been added between both busses in order to enable the ARM to access the image data.

The connection to the outside world of the memories is made through the buses. A single clock is used for the complete architecture. The following sections describe the architecture in more detail.

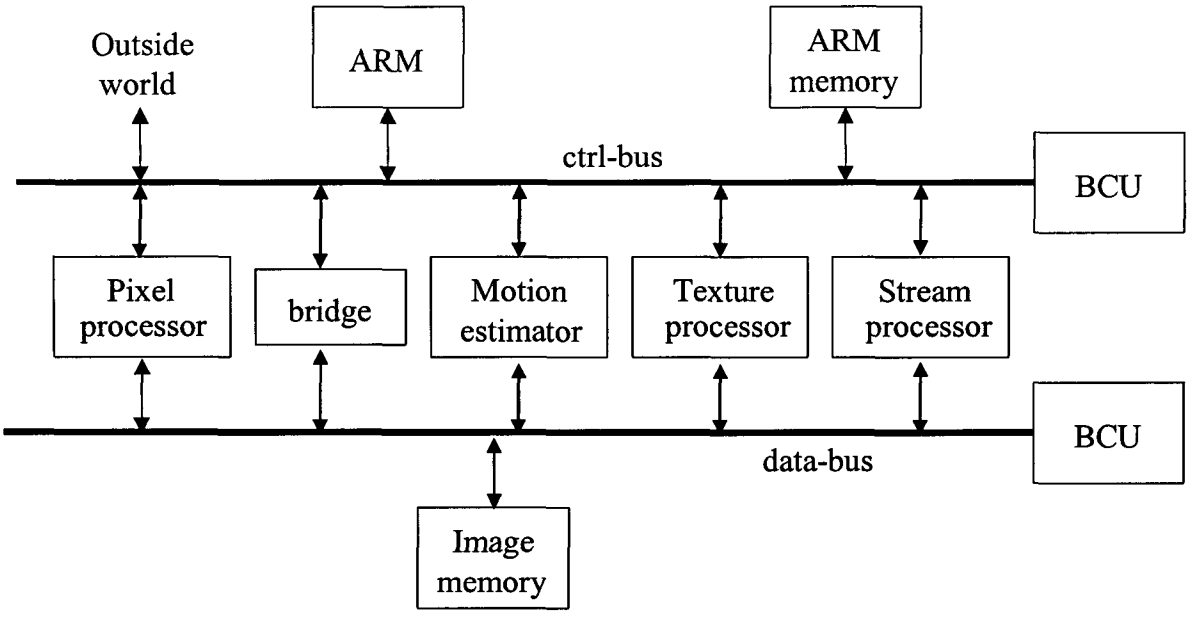


Figure 24. The resulting hardware architecture.

4.5.4 Pixel processor

The pixel processor is responsible for the communication with the front-end (the image sensor part), by doing the line to stripe conversion. Lines of pixels are read and written into a small buffer memory. After reading 16 lines, a pointer is passed to software indicating where the stripe can be found. This pointer is used to start processing the macro blocks in that stripe.

While processing the current stripe, the next stripe can be read. This requires a buffer size equal to 4 stripes: 2 for input data and 2 for output data. Note that since the input data is progressive, and since the Picasso encoder stops during line and field blanking, no additional buffer size is needed at the input part.

4.5.5 Motion estimator

The motion estimator receives a set of candidate vectors from the software. Each candidate vector consists of the vector coordinates, a penalty value, and the size of a random update. These vectors are evaluated, and the best vector in terms of lower sum of absolute differences (SAD) is selected and refined (full- half- and quarter-pixel).

The motion estimator is also responsible for generating a metrics representing the cost of coding of a macro block in intra mode.

The output of the motion estimator consists of the motion vector(s), and the intra metrics. This information is used in software to determine the encoding approach for the current macro block (skipped, intra coded, inter coded with zero vector, inter coded with vector, or inter coded with four 8×8 vectors). The main benefits of the 3DRS implementations are the following:

1. ***Busload:*** Using a refinement memory reduces the DSP-bus load considerably, resulting in more headroom for further improvements and modifications on the encoder algorithm.
2. ***Performance:*** Since less scalable decode actions are carried out, it is possible to design a slower scalable decoder.
3. ***Low power:*** As stated before, the number of scalable decode actions is reduced considerable by using a refinement memory inside the motion estimator.

4.5.6 Texture processor

The texture processor is responsible for using the results of the motion estimator (encoding mode and vectors) to encode and decode the macro blocks, and to store the decoded macro block in the loop memory.

The output of the texture processor consists of a list of VLE codes for the DCT coefficients of the current macro block (if the VLE of the coefficients is done in software, then the output consists of a list of RLE values). A content addressable memory is used to transform the RLE values into VLE values [KWS+99].

Since the texture processor needs (for the APM) the motion vectors of macro blocks (x-1,y), (x,y), (x+1,y), and (x, y-1) in order to encode macro block (x,y), the motion estimator is always two macro blocks ahead of the texture processor.

On the other hand, since the software needs the results of the texture processor in order to generate the VLE headers, to encode the motion vectors, and (eventually) to do the VLE of the coefficients in software, the texture processor must be one macro block ahead of the software.

The texture processor requires also some additional functionality to enable the UMV and APM options, since both options allow vectors pointing out of the image. The texture processor generates a border of non-existing pixels by simply copying the edge pixels.

4.5.7 Stream processor

The stream processor is responsible for collecting the VLE codes generated by the texture processor and software, to concatenate them, and to transfer them on the output of the Picasso design (storage or connection to PC). Due to the parallelism of the texture processor and the software task, multiplexing is needed in order to interleave the VLE codes of the headers and motion vectors with the VLE codes of the coefficients.

4.5.8 Software

The previous sections describe the hardware coprocessors of the Picasso design. However, several tasks are to be done in software.

As was explained in Section 4.2, Picasso addresses two scenarios: one based on full compression functionality up to CIF 30 frames per second (small scenario), and the other for VGA 30 frames per second (large scenario).

The following tasks are still done on the ARM:

- Initialization.
- Candidate vector generation for motion estimation.

- VLE of headers and motions vectors (and optionally also of coefficients).
- Bit rate control (quantizer value control).
- All standard dependent actions (e.g. motion vector clipping).
- Handshaking with hardware blocks.

The most important task of the ARM is to do the handshaking with the hardware coprocessors. The most important aspect of this handshaking is that it should be designed in such a way that as many as possible hardware and software is running in parallel. Figure 25 shows how this is achieved for the case of a SQCIF video stream (8 by 6 macro blocks per picture).

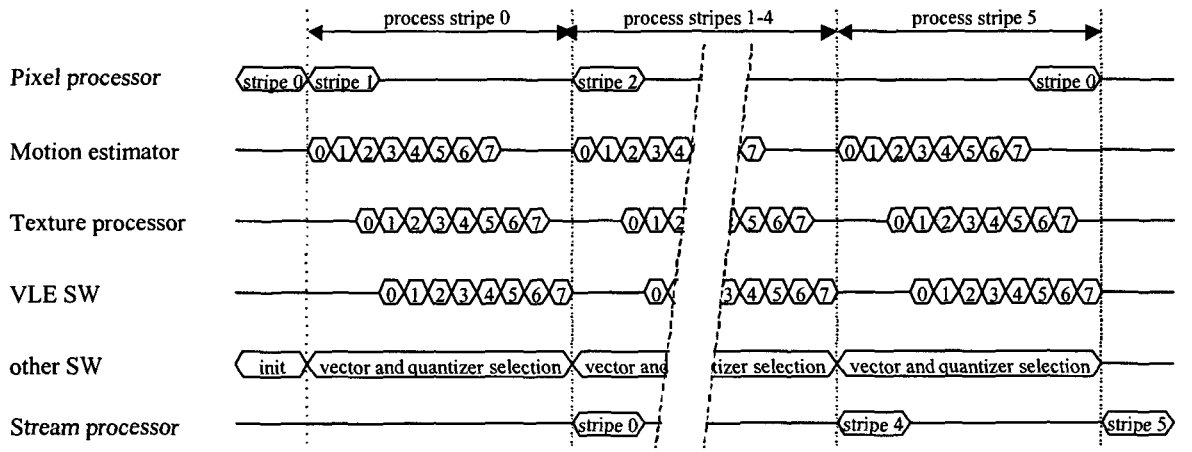


Figure 25. Pipelining of hardware and software.

The pixel processor is always one stripe in advance. The motion estimator, the texture processor, and the software VLE are done in parallel. As stated before, the motion estimator is two macro blocks ahead of the texture processor, which on its turn is one macro block ahead of the software VLE. As can be seen in the figure, the pipeline of these three blocks is filled and emptied at the beginning and at the end of each stripe, respectively. This allows to do bit rate control computations before starting a new stripe. Furthermore, the stream processor collects the VLE codes of several macro blocks in a buffer, and writes them to the output each time this buffer is full, or at the end of a macro block stripe. In the case of this example (SQCIF), the end of the macro block stripe is reached before this buffer is full.

This example shows that for some period the motion estimator, the texture processor, and the software VLE are running in parallel. During this period the pipeline is full, and the hardware and software blocks are used very efficiently. Outside this period the computations

are done less efficiently, since all three blocks are not running in parallel. However, for larger image sizes (e.g. VGA) the total number of macro-blocks per macro-block stripe increases, resulting in an increase of this efficient period. Since the length of the inefficient period stays the same, the overall efficiency increases.

4.5.9 Embedded memories

The Picasso design contains two important embedded memories, the ARM memory, and the image memory. The following two sections describe these memories in more detail.

4.5.9.1 CPU memory

The ARM memory is connected to the ARM through the ctrl-bus (an AMBA-bus), and is used to store all ARM related data. The following data is stored in this memory:

- Operating system and program of ARM.
- Variables of ARM, including the motion vectors.
- Communication data for C-HEAP communication with coprocessors.

The actual size of this memory depends on several application choices, such as if the code going to be stored in flash, embedded or external ROM (Read Only Memory). For standard applications the memory size is estimated to be around 500 Kbits.

4.5.9.2 Image memory

The image memory is connected through the data-bus to all coprocessors, and through a bridge indirectly to the ctrl-bus. The following data are stored in this memory:

- Input buffer consisting of four stripes. Since the memory size to store these stripes is rather large (4·16·3/2·640·8 = 480 Kbits for VGA and 4·16·3/2·352·8 = 264 Kbits for CIF).
- Loop memory containing the previous and current reconstructed images. The size of a single compressed (2 bits per pixel) reconstructed image is 1008 Kbits and 360 Kbits for VGA and CIF, respectively. Two images have to be stored: the previous for motion estimation, and the current image being encoded. However,

since the images are encoded in a progressive way, it is possible to merge both images, as shown in Figure 26. As soon as a part of the previous reconstructed image is not needed any more, it is overwritten by the current image. This requires an overlap of 2 stripes. However, in order to support the UMV option an overlap of 3 stripes is required. As stated before, this memory is organized in two parts, a pointer part and a data part. This results in a data memory of 40·(30+3)·384·2 = 990 Kbits and 22·(18+3)·384·2 = 346.5 Kbits for VGA and CIF, respectively, and a pointer memory of 40·(30+3)·6·8=61.875 Kbits for VGA and 22·(18+3)·6·8=21.656 Kbits for CIF. Both parts are organized in 8×8 blocks. This is done to average out size differences of 8×8 blocks (chrominance blocks are smaller than luminance blocks).

- Note that no output buffer is part of the Picasso design. An output buffer is normally used to accommodate changes in bit-rate. However, the size of this buffer is related to the application, more concrete, to the bandwidth of the channel used to transmit the compressed video stream. Therefore, this buffer will be part of the platform in which the Picasso design will be embedded.

4.6 Picasso performance

The complete architecture has been modelled in the C-language, and simulated in TSS (Tool for System Simulation). The hardware blocks have been modelled as a UNIX process, whereas the software part is running inside the context of TSS on a (near) cycle-true Instruction Set Simulator (ISS). The delay of the hardware blocks is assumed to be zero. However, since these hardware blocks are connected to a ctrl-bus, and the delay of this bus is

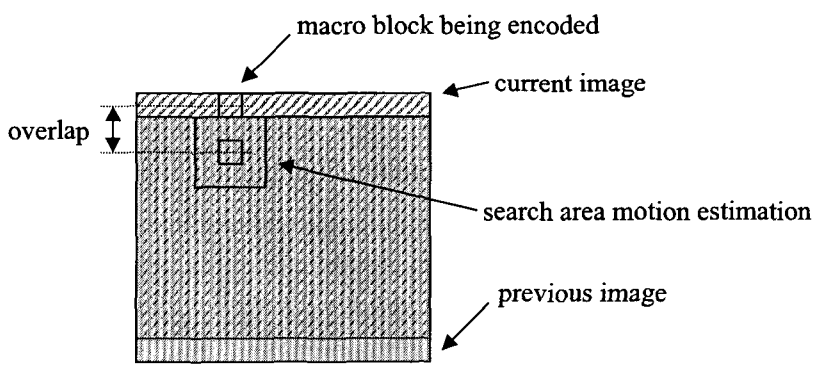


Figure 26. Merging the previous and current reconstructed images.

modelled correctly, some timing is taken into account for these blocks. This is not a serious modelling limitation, since pipelining can increase the speed of these blocks. Fast embedded memories (zero wait cycles) have been assumed for these simulations. The simulations have been done based on the DCT/spatial domain architecture, using lossless embedded compression. The latter is clearly a worst-case situation, since lossy embedded compression will reduce the memory bandwidth by an additional factor of two.

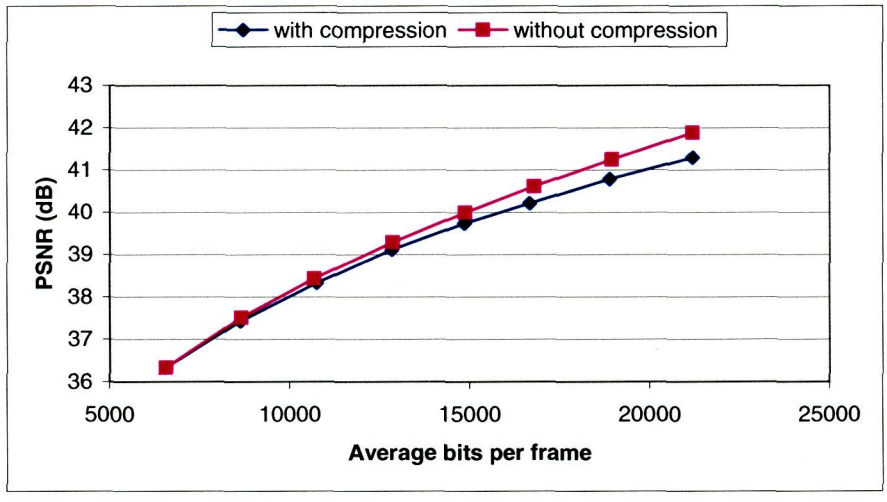


Figure 27. Rate distortion curve for 'SUZIE' video sequence (QCIF size) with and without embedded compression.

The use of embedded compression has enabled a single-chip and low-cost solution. The performance of embedded compression has been evaluated for 'SUZIE' and `CARPHONE' video sequences (CIF @ 10 frames per second) and are shown in Figure 27 and Figure 28 respectively. The rate distortion curves for these sequences indicate a marginal degradation in PSNR values even though there seems to be no perceptual degradation. Further, the embedded compression factor is 2.7 and the encoder compression factor range from 15 to 50.

Table 4 provides the details of the low-end CPU. The CPU utilization factor estimates are derived from processing CIF video sequences at 30 frames per second (fps). Table 5 provides the area and power dissipation estimates for the various components of the H.263 encoder implementation in 0.18µm CMOS technology. The worst-case estimation indicates a H263 implementation with 100mW of power dissipation and 15 mm^2 of area. These indicative numbers are some of the best reported in literature (see Figure 29), thus powering future mobile multimedia systems with low-cost and low-power video encoding. These numbers are obtained from estimations done at various levels of implementation abstraction.

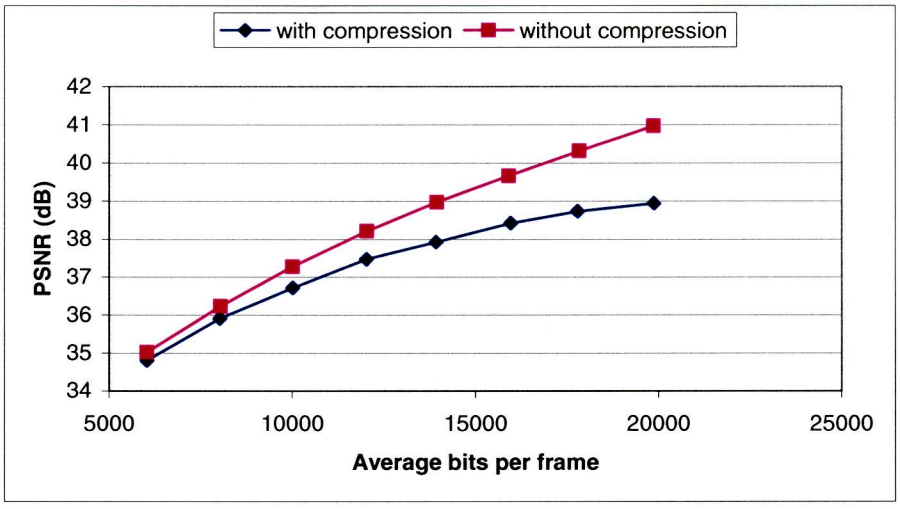


Figure 28. Rate Distortion Curve for 'CARPHONE' Video Sequence with and without embedded compression

CPU	ARM7 TDMI-S
Clock Frequency	110 MHz
Area	0.62 mm^2
Power	0.39mW/MHz
Utilization Factor	54 %

Table 4. CPU Features.

Instance	Without embedded compression		With embedded compression	
	Area (mm^2)	Power (mW)	Area (mm^2)	Power (mW)
ARM7 TDMI-S	0.62	3.93	0.62	5.15
Memory (CPU+Loop)	13.39	2.95	7.64	2.65
Motion estimator	1.62	6.03	2.74	8.70
Texture codec	4.24	4.30	5.39	6.21
Packer co-processor	0.50	0.08	0.50	0.08
Shells & busses	3.07	5.50	2.77	7.55
Total	265	299	382	262

Table 5. Area and Power Estimates for Encoder.

4.7 Conclusions

The architecture of the Picasso design has been overviewed. This architecture is based on a careful selection of the encoder domain, and on a careful hardware-software partitioning in order to obtain low power, low cost, and high performance in combination with flexibility.

Low power and low cost has been achieved by applying embedded compression, and by making a careful choice between the DCT and the spatial domains. Furthermore, a refinement memory is used in order to reduce the power dissipation even further.

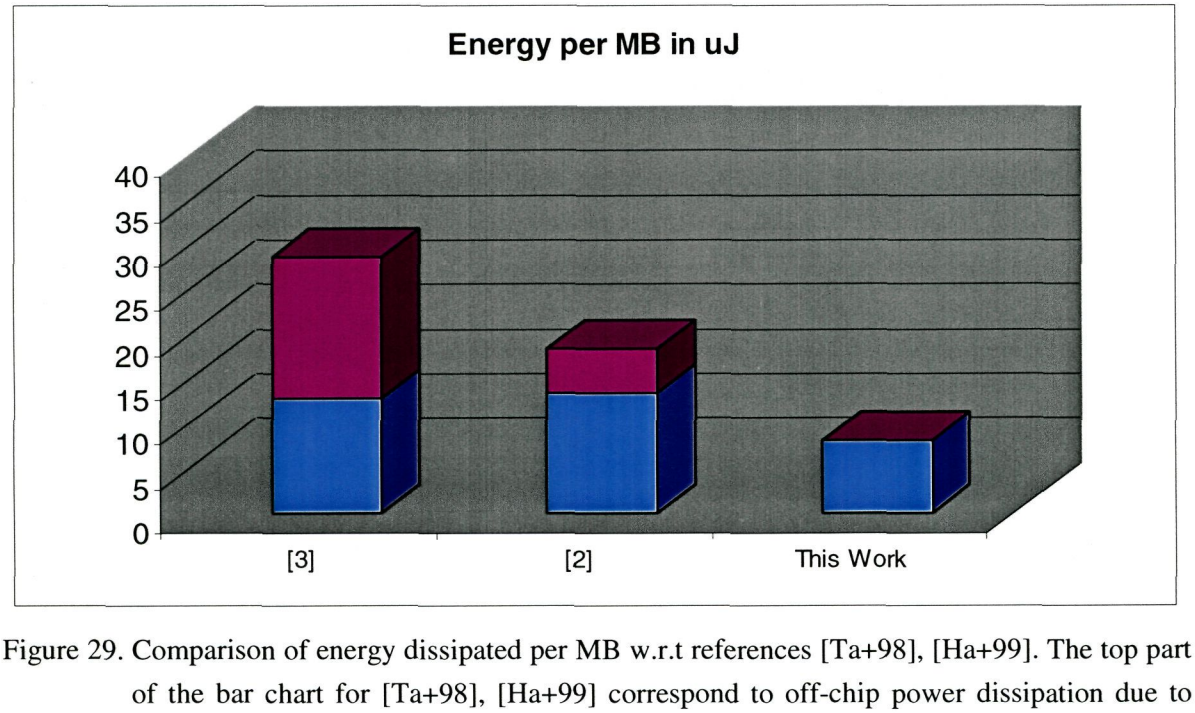


Figure 29. Comparison of energy dissipated per MB w.r.t references [Ta+98], [Ha+99]. The top part of the bar chart for [Ta+98], [Ha+99] correspond to off-chip power dissipation due to external loop memory. Further, all implementations are normalized to the same technology (0.18u CMOS). Furthermore, the 60-mW power dissipation reported in [Ta+98] excludes the power due to off-chip loop memory.

High performance has been obtained by implementing the compute intensive parts in dedicated hardware, using a 0.18μm CMOS library. A high degree of flexibility for multi standard purposes is offered by implementing the control and encoding-standard-related parts in software. The communication between hardware and software is based on the C-HEAP protocol.

Chapter 5

Every sentence that I utter must be understood not as an affirmation, but as a question.

Niels Bohr (1885 - 1962)
Physics Nobel Prize in 1922

Mapping of the Super-Resolution Algorithm onto a Video Encoder

5.1 Introduction

In order to meet low-cost features, a big effort has been made adapting the super-resolution algorithm to the existing video encoding platform. As only limited hardware is needed to achieve super-resolution improvements, we can see super-resolution as an added value feature to video encoding at low cost (except engineering design costs and slight modifications of the platform).

5.2 Iterative super resolution

The first approach to super-resolution is made through the adaptation of the algorithm described in section 3.3 to be executed on the hybrid video encoder.

5.2.1 Algorithm description. Version v1.0

The initially proposed iterative algorithm shown in Figure 6 can be written again, but restricting the resources to those ones that can be found in the *"Picasso"* hybrid video-encoding platform. This new algorithm is exposed in Figure 30. This first version, labeled as v1.1, supposes the first real implementation of the super-resolution algorithm. The algorithm is structured in three parts: initialization, first iteration, and loop for the remaining iterations. The algorithm finishes when it reaches the number of iterations previously established, or when it converges, i.e. the error variance is lower than a 0.5. The initialization sets off from a value call *'scale'* (two in our case) that will be the size increase in the horizontal and vertical directions. From the 'scale' value is established the *'nr_frames'* value (number of incoming frames) as $scale^2$ (4 in our case), since this will be the area increase. Next, the memories that are going to be used are declared and some values are initialized. Concretely:

- HR_B: high-resolution accumulative memory (*buffer*) where the result of the super-resolution process will be stored.
- HR_A: high-resolution memory (*average*) where the computation of the high-resolution average value is carried out.
- HR_S: high-resolution memory (*shift*) to store the shifted results.
- HR_T: high-resolution temporal memory.
- LR_B: low-resolution version of HR_B, is obtained by decimation.
- LR_I[]: low-resolution memory vector to store the incoming images. Its indexes go from 0 to nr_frames-1.
- MV_fr2ref[] and MV_ref2fr[]: memory vectors intended to store the estimated motion vectors. These memory vectors are in low-resolution.

One of the main problems to solve was the motion estimation, needed to compensate the movement in the consecutive incremental improvements. This is a key factor in the super-resolution algorithms and its influence on the resulting image quality is decisive, as it will be seen later. In the literature about this topic [EF97], [HKK97a] is usual to make a separation between this problem and the super-resolution one, usually limiting to suppose the displacements perfectly known when applying the algorithm. Even in the case that the movement estimation is included, the authors commonly use pixel level procedures, very accurate, but very expensive in computational terms, and of course, most of them with few or

none possibilities of been implemented in real time with the nowadays available technology, and even with the estimated technology available in the following years.

The objective of this work is to find a comprehensive and viable solution with real time restrictions. In this sense, it is necessary to incorporate the motion estimation inside the super-resolution algorithm. The chosen way, in order to minimize the implementation costs, has been to reuse the architecture for the Picasso hybrid video encoder, as it can also offer a solution to the real time super-resolution problem, if an appropriate mapping of the algorithm into the architecture can be set.

Under the light of the available resources and the real time and low-cost desired performances, the best way to perform the motion estimation is using the 'motion estimator' included in the magnitude video compression of Picasso. This supposes the review of the following matters:

- The motion estimation must be carried out, not at a pixel level, as would be desirable from an algorithm point of view, but at a block level, i.e. in groups of 8×8 pixels, as it is imposed by the video compression magnitude.
- The influence of the estimation method (full-search, modified three-dimensional recursive search 3DRS, etc.) and its parameters (search area, motion vectors length, accuracy, etc.) in the motion vectors quality, and especially in the quality of the final image, must be carefully analyzed.
- The optimal size of the block for the posed problem the must be established. A small size would approach us to the ideal pixel shift, but it will be strongly affected by the noise, whereas a big size block would offer, in general, a better global motion vector, but of poor quality when trying to determine small local movements.
- The influence of aliasing presence in the motion estimation must be studied, as the motion estimation is intended to compress video sequences without aliasing.

The motion estimator of the video compressor used in Picasso initially had half-pixel accuracy, so, the first modification has been to introduce an additional refine stage in the vectors computation to enable a quarter-pixel accuracy. Nevertheless, this feature has been let as a configurable parameter, with the aim of enable an experimental comparison of how the vector accuracy affects to the final image quality.

```
* Block-align super-resolution algorithm v1.0

Set the value of the improvement: scale
nr_frames = scale*scale
If LR_I[] is a MxN matrix, then LR_B is the same size: LR_B[M][N]
HR_B[scale*M][scale*N], HR_S[scale*M][scale*N], HR_A[scale*M][scale*N]
Read a set of aliased Low-Resolution images in LR_I[#nr_frames]
Set the value of the maximum number of iterations: nr_iterations
The size of motion vector matrixes depends on the block size of the Motion Estimation

MV_fr2ref[0] = 0
FOR fr = 1 .. nr_frames-1
    MV_fr2ref[fr] = Calc_Motion_Estimation (LR_I[fr], LR_I[0])
    MV_fr2ref[fr] = 2 .* MV_fr2ref[fr]
END FOR
HR_A = 0
HR_B = 0
// First Iteration
FOR fr = 0 .. nr_frames-1
    LR_B = 128
    LR_B = LR_I[fr] – LR_B
    HR_S = Upsample (LR_B, scale)
    HR_T = Motion_Compensation (HR_S, MV_fr2ref[fr])
    HR_S = HR_T
    HR_A = HR_A + HR_S
END FOR
HR_A = HR_A ./ nr_frames
HR_B = HR_B + HR_A
Contrast_Clip (HR_B, 0, 255)
// Next Iterations
FOR it = 1 .. nr_iterations
    LR_B = Downsample(HR_B, scale)
      FOR fr = 0 .. nr_frames-1
        MV_fr2ref[fr] = Calc_Motion_Estimation (LR_I[fr], LR_B)
        MV_ref2fr[fr] = Calc_Motion_Estimation (LR_B, LR_I[fr])
        MV_fr2ref[fr] = scale * MV_fr2ref[fr]
        MV_ref2fr[fr] = scale * MV_ref2fr[fr]
      END FOR
      HR_A = 0
      FOR fr = 0 .. nr_frames-1
        HR_S = Motion_Compensation (HR_B, MV_ref2fr[fr])
        LR_B = Downsample(HR_S, scale)
        LR_B = LR_I[fr] – LR_B
        HR_S = Upsample (LR_B, scale)
        HR_T = Motion_Compensation (HR_S, MV_fr2ref[fr])
        HR_S = HR_T
        HR_A = HR_A + HR_S
      END FOR
      HR_A = HR_A ./ nr_frames
      variance = Variance(HR_A)
      HR_B = HR_B + HR_A
  Contrast_Clip (HR_B, 0, 255)
  IF (variance < 0.5) break;
END FOR
```

Figure 30. Pseudo-code of the first iterative algorithm using the resources of a hybrid video encoder.

When performing the motion estimation in low resolution, the resulting motion vectors are also low-resolution, and thereby, to perform the motion compensation in high resolution the vectors must me conveniently scaled, multiplying them by the used scale factor.

In principle, the super-resolution algorithm is intended for short image sequences (`nr_frames`), where each image supposes a shifted version of the previous one. We will call to this kind of sequences "global shift images", because the entire image is affected by the same motion vector, which is unique and global for every image of the sequence. As the motion estimator is prepared to calculate the motion vector for every luminance macro-block (four 8×8 blocks, i.e. one motion vector every 16×16 pixels) we will calculate from that vector set the global vector. For this purpose, we can follow three approaches:

1. Calculate the average of all the motion vectors, in the horizontal and vertical directions, (*mv.x* and *mv.y*) and assign this value to the global vector (*MV.x* and *MV.y*). This operation is shown in ((22) for an M×N luminance block

$$MV.x = \frac{1}{M \cdot N} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} mv(i,j).x$$
$$MV.y = \frac{1}{M \cdot N} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} mv(i,j).y \qquad (22)$$

2. Find the most frequent vector and assign this vector to the global one.
3. Find the most frequent vector component in the horizontal and vertical directions and assign these two components to the global vector.

The tests performed in that direction show a great convergence among the tree options, seldom showing small deviations, normally below one quarter of pixel. For instance, the vectors set calculated following the three procedures and the error for each case are presented in Table 6. The selected motion estimation has been the modified three-dimensional recursive search (3DRS), with ¼-pixel precision, and the test sequence was composed of 12 frames, artificially shifted, of the *KANTOOR* sequence, where the frame zero was used as the reference.

Instead of computing directly the average image $\bar{g}'(x,y)$ as expressed in (4), it is faster and the accurate is the same, to start from the average value among all the possible pixel values. As every pixel is stored as an eight-bit number, and in the luminance case its

vales are always positive [0..255], the average value is 128. Although starting from the real average value convergence is reached faster, the algorithm however remains very far from real time operation, thereby it makes no sense to increase the computational load of the algorithm in the initial stage what, in any case, will save no more than one iteration. Although the computational load of that extra iteration will be certainty bigger than the one of computing the average value, its effect is negligible compared to the aim of reach the desired performances and thus the algorithm results in a more simplified style.

Real vector		Average vector				Most frequent vector				Most frequent component			
		Values		Error		Values		Error		Values		Error	
2	-2	2	-2	0	0	2	-2	0	0	2	-2	0	0
3	-1	3	0	0	-1	3	-1	0	0	3	-1	0	0
3	-3	3	-3	0	0	3	-3	0	0	3	-3	0	0
2	0	2	0	0	0	2	0	0	0	2	0	0	0
0	-3	0	-3	0	0	0	-3	0	0	0	-3	0	0
0	-1	0	-1	0	0	0	-1	0	0	0	-1	0	0
-2	-1	-2	0	0	-1	-2	-1	0	0	-2	-1	0	0
1	0	1	0	0	0	1	0	0	0	1	0	0	0
0	-2	0	-2	0	0	0	-2	0	0	0	-2	0	0
2	0	2	0	0	0	2	0	0	0	2	0	0	0
0	2	0	2	0	0	0	2	0	0	0	2	0	0

Table 6. Errors in the estimation of the global vector for the *KANTOOR* sequence composed of 12 frames. Frame zero is the reference.

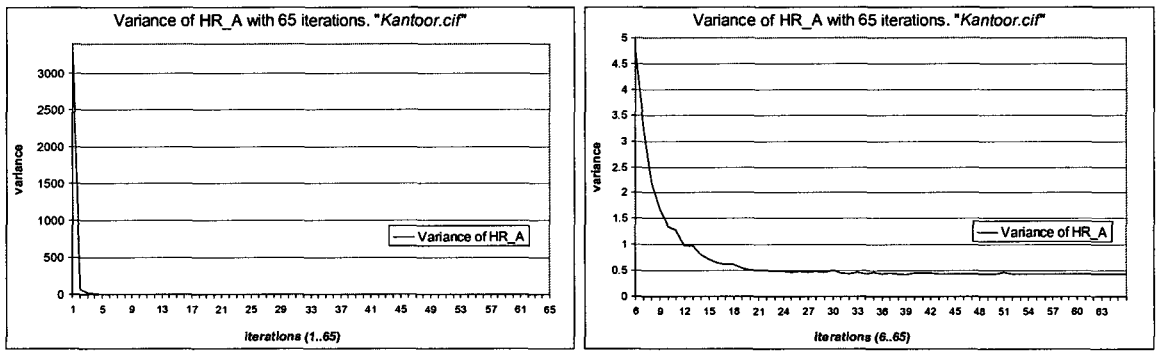
Another important problem is the fact that the Picasso architecture is not intended to simultaneously support two different memory sizes. In this case we have opted for the solution of assuming that the sensor only delivers the Region Of Interest (ROI) to improve, which is equivalent to use only high-resolution images.

This solution based in the use of only high-resolution images supposes the additional advantage of avoiding the interpolation and decimation operations for increase and reduce the size of the memories, avoiding the errors produced by these operations. Likewise, as far as the motion estimation is also performed in high resolution, the scaled vector is again unnecessary.

In Figure 31 (a) it can be seen the variance value of HR_A for the test image *KANTOOR* of original size QCIF as the iteration number increases until 65. In Figure 31(b) it can be seen this same value from iteration number six, with the intention of highlight the variance trend towards the value 0.439. For that reason, it is empirically selected the value 0.5 for the variance as threshold to stop the iterative process. In this case, the cut value was reached at iteration 21.

5.2.2 Overflows and code redistribution

After computing the average value for the first super-resolution proposal, we must take into account that no memory intended to store an image can include values that do not fit in eight bits, before storing the result of an arithmetic operation, a clip of the image values to 8 bits must be performed. This is reflected in the algorithm as '**Contrast_clip**'. In fact, this is one of the first problems founded when trying to map image-processing applications reusing an architecture conceived for image compression: the arithmetical problems.



(a) (b)

Figure 31. Variance behaviour of HR_A during 65 iterations (a) and detailed view that shows the variance from iteration number 6 (b).

It is a basic principle of the binary arithmetic that the addition of two N-bits numbers produces an N+1 bits number. This so basic principle has supposed a very important problem source in the algorithm develop. The Picasso architecture reads images in macro-blocks stripes inside the coprocessors. In that moment, every pixel is represented with an eight-bit number. Once in the coprocessor, the image values are processed in a sixteen-bit word-wide architecture (block level processing), but the result must be stored again in an eight-bit image

memory, which has been schematized in Figure 32. For image compression, this is not a major problem, but when reusing the same architecture for arithmetic image processing, we face the limitation that any intermediate result must be stored in 8 bits. Because of this problem, we have adopted the following solutions:

- Try to perform all the arithmetic operations at block-level.
- Rearrange the arithmetic operations in such a way that, when storing the intermediate results, these are limited, as far as possible, to eight bits.

Related with the arithmetic operations rearrange matters, the straightforward change is to distribute the division implied for the average value computation in divisions inside the main loop of Figure 30. For that purpose, two choices are presented, as shown in Figure 33:

- The first one is to divide every image by the total (4 in our case) before the addition, but this implies an important dynamic range reduction that negative impacts on the image quality.

- The second one is to distribute the division between the subtraction operation and the accumulator that supports the overall error summation. In that way, the dynamic range clip is split between the memory that stores the subtraction and the memory that stores the accumulated summation.

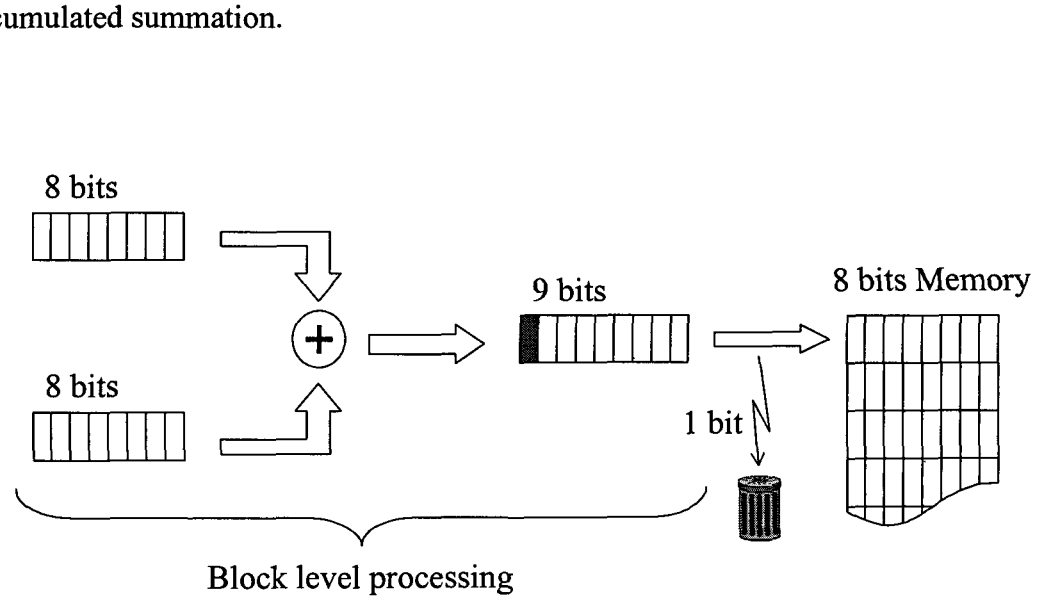


Figure 32. Effects of precision loose when performing arithmetic operations on 8-bit images.

This second option offers the best results and hence it has been the selected one. Although with this arrangement a satisfactory solution is achieved, for simulation we have opted for increase the accumulator memory size to 16 bits, with the intention of being able quantify overflows in a more accurate way.

5.2.2.1 Optimizations to the iterative super-resolution algorithm version v1.0 for being mapped in the original Picasso hybrid coder. Version v1.1

Under all that premises, the second version of the iterative super-resolution algorithm has been developed. It is shown in Figure 35 and labeled as v1.1. The algorithm has been substantially simplified due to removing the operations of resizing between high and low resolution. HR_A memory appears in boldface because it is a special nine-bits memory. The labels at the right side between brackets are the actions issued to the coprocessors and reflect the operation set performed by the coprocessor at the block-level, i.e. with an internal 16 bits precision.

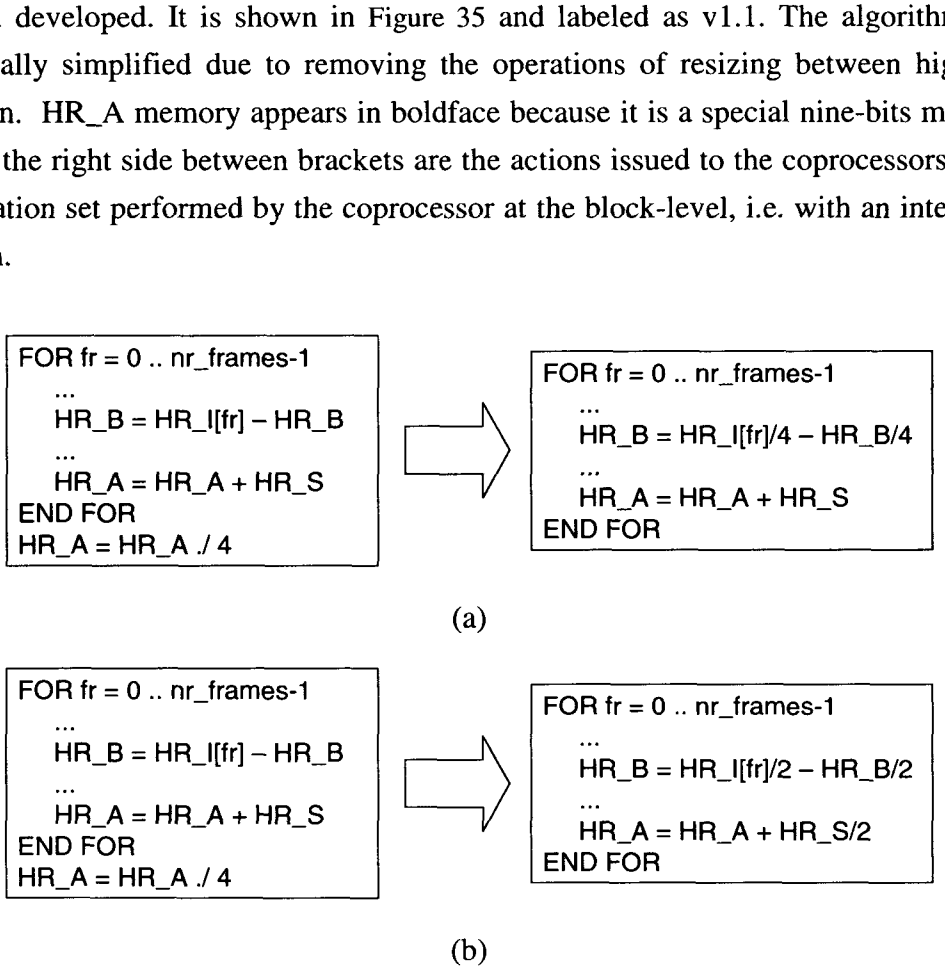
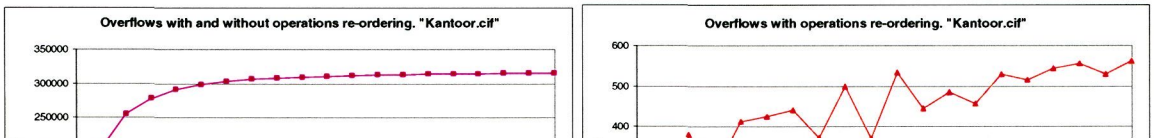


Figure 33. Code distribution strategies for the average value computation. (a) Grouping the division in a single operation. (b) Distributing the division.

The arithmetic operations redistribution drastically reduces the number of overflows produced during the code execution. Figure 34 (a) shows the number of overflows produced in version v1.0 without operations redistribution and in version v1.1, Figure 34 (b), with code redistribution. It must be taken into account that the maximum number of overflows per iteration will be the size of the high resolution image (352×288) multiplied by the three components (luminance, red chrominance and blue chrominance) and by the number of operations that may produce overflow, three in our case (SR_ACT1 o SR_ACT3, SR_ACT2 and SR_UPDATE), that is, 352×288×3×3=912384. It can be appreciated that without operation redistribution the number of overflows reaches 298196 in the 6^{th} iteration, i.e. a 32.68% of the arithmetical operations produces overflows. With redistribution, for the same iteration number, the overflows are equal to 424 that represent the 0.046% of the performed operations. In Table 7 the exact number of overflow produced in both versions and the percentage that it represents over the total arithmetic operations performed are shown.



(a) (b)

Figure 34. Arithmetic overflows produced during 20 iterations with and without reordering operations (a) and detail that shows only the overflows with arithmetic operations reordering (b).

The overflows produce image saturation. If the overflow is produced by an operation which its result is higher than 255, this will be clipped to 255 (overflow) and if the result is lower than zero, it will be clipped to zero (underflow). If the overflow number is high, a significant degradation of the image quality will be appreciated. Nevertheless, a small overflow number is acceptable as it only supposes a small quantification effect, imperceptible in most of the cases. However, although this effect will be inappreciable, it will be reflected as a drastically drop in the signal to noise ratio and in the spectral correlation coefficient.

Iteration	Version v1.0		Version 1.1	
	total	%	total	%
1	50688	5.56	0	0.000
2	202960	22.25	0	0.000
3	255064	27.96	380	0.042
4	277290	30.39	289	0.032
5	290483	31.84	413	0.045
6	298196	32.68	424	0.046
7	303131	33.22	440	0.048
8	306110	33.55	372	0.041
9	308189	33.78	499	0.055
10	309610	33.93	370	0.041
11	310888	34.07	533	0.058
12	311791	34.17	444	0.049
13	312464	34.25	484	0.053
14	313181	34.33	456	0.050
15	313600	34.37	529	0.058
16	314040	34.42	515	0.056
17	314402	34.46	543	0.060
18	314683	34.49	556	0.061
19	315017	34.53	530	0.058
20	315118	34.54	561	0.061

Table 7. Arithmetic overflows produced in versions v1.0 and v1.1 and the percentage over the total arithmetic operations performed at block level.

5.2.3 Transformations of the iterative algorithm with reference to the average image. Version v1.2

With the aim of decreasing the operations amount a new version of the super-resolution algorithm, labelled as v1.2, has been developed. We use again two different image sizes (and so two different memory sizes) simultaneously. The new version of the algorithm, shown in Figure 36, moreover reflects some additional modifications for being properly mapped in the new architecture.

We have decided to homogenize the iterative loop, removing the first and irregular iteration. We have opted for using as the first super-resolution image proposal the average of all the input images, therefore decreasing the average noise in the resultant image. The SR_AVERAGE action reads four low-resolution input images, taking them to high resolution thought closed neighbour interpolation and obtain the average, that will be taking as the first high-resolution image proposal.

```
* Iterative super-resolution algorithm v1.1

Set the value of the improvement: scale
nr_frames = scale*scale, M'= scale*M, N'= scale*N
HR_B[M'][N'], HR_S[M'][N'], HR_T[M'][N'] all of 8 bits. HR_A[M'][N'] is 9 bits,
Read a set of aliased Low-Resolution images in LR_I[#nr_frames][M][N]
Set the value of the maximum number of iterations: nr_iterations
The size of motion vector matrixes depends on the block size of the Motion Estimation

MV_fr2ref[0] = 0

FOR fr = 1 .. nr_frames-1
        HR_I[fr] = Upsample (LR_I[fr], scale)                          [SR_STORE]
        MV_fr2ref[fr] = Calc_Motion_Estimation (HR_I[fr], HR_I[0])
END FOR

    HR_A = 0
    HR_B = 0                                                           [SR_INIT_A_B]

// First Iteration
FOR fr = 0 .. nr_frames-1
        HR_S = 128                                                     [SR_ACT1]
        HR_S = HR_I[fr]/2 – HR_S/2
        HR_T = Motion_Compensation (HR_S, MV_fr2ref[fr])
        HR_S = HR_T                                                    [SR_ACT2]
        HR_A = HR_A + HR_S/2
END FOR

HR_B = HR_B + (HR_A)                                                   [SR_UPDATE]

// Next Iterations
FOR it = 1 .. nr_iterations
        FOR fr = 0 .. nr_frames-1
            MV_fr2ref[fr] = Calc_Motion_Estimation (HR_I[fr], HR_B)
            MV_ref2fr[fr] = Calc_Motion_Estimation (HR_B, HR_I[fr])
        END FOR

        HR_A = 0                                                       [SR_INIT_A]

        FOR fr = 0 .. nr_frames-1
            HR_S = Motion_Compensation (HR_B, MV_ref2fr[fr])           [SR_ACT3]
            HR_S = HR_I[fr]/2 – HR_S/2
            HR_T = Motion_Compensation (HR_S, MV_fr2ref[fr])
            HR_S = HR_T                                                [SR_ACT2]
            HR_A = HR_A + HR_S/2
        END FOR
        HR_B = HR_B + (HR_A)                                           [SR_UPDATE]
    variance = variance(HR_A)                                          [SR_STAT]
If (variance < 0.5) break
END FOR
```

Figure 35. Pseudo-code of the modified iterative algorithm v1.1.

As in version v1.1, the temporal high-resolution memory HR_T has been introduced in order to avoid data overlapping, at the same time that the image HR_S is ready to be shifted, this one would be overwritten with new data. At the implementation stage, this memory will not be necessary, due to the overlap existing in the Picasso architecture between the present memory and the reconstructed one of two macro-blocks rows, this data hazard problem will be automatically solved by the hybrid coder structure.

The LR_B memory will always contain a low-resolution version of the super-resolution image, obtained by decimation (down-sample) and its use is restricted to the computation of the low-resolution motion vectors. It must be taking into account that the motion estimation is one of the most computational intensive tasks, and therefore performing it in low-resolution implies an important time and power consumption save. Another way to reduce the algorithm requirements is obtained by removing the second motion estimation.

Until now, we always have performed two motion estimations per iteration and input frame: one of the super-resolution images with respect to the input frame and other of the input frame with respect to the super-resolution image. Nevertheless, due to the used motion estimator, with vectors in Cartesian form, it supposes a good approximation to consider that the second motion vectors can be derived from the first ones through a simple inversion of the direction. Although from a conceptual point of view this is right, in the practice the same inverted motion vectors are not obtained if both motion estimation are performed and compared, because the motion estimation process is really a simple block matching, i.e. a comparison of the accumulated absolute difference summations (SAD, Sum of Absolute Differences) obtained when comparing every 8×8 block with other 8×8 blocks of the other image inside a certain search area. In consequence it is possible that, as the search areas are different it can exist another 8×8 block in the second image with a lower SAD, thus altering the homologue motion vector, although normally slightly. In the algorithm proposed in the version v1.2, this is reflected as an instruction that it is really a function (that it is why it is not enclosed between brackets) named INVERT_MV().

5.2.3.1 Memory requirements for version v1.2

In Figure 37 the block diagram of the algorithm is shown. The functional blocks have been shadowed to distinguish them from memories. All the used image memories are eight bits per component, except HR_A that is nine bits, and for that reasons it has been

distinguished using double line in the borders. The functional blocks motion estimation and motion compensation are already included in the hybrid video encoder. The remaining blocks have to be added over some of the existing co-processor and we finally opted for the compressor block.

The diagram block of Figure 37 also supposes an important aid in the evaluation of the necessary resources for a sequential execution of the algorithm over the pixel flow in a macro-

```
* Iterative super-resolution algorithm v1.2

Set the value of the improvement: scale
nr_frames = scale*scale, M'= scale*M, N'= scale*N
HR_B[M'][N'], HR_S[M'][N'] , HR_T[M'][N'] all of 8 bits. HR_A[M'][N'] is 9 bits
LR_B[M][N] for the motion estimation
Read a set of aliased Low-Resolution images in LR_I[#nr_frames][M][N]
Set the value of the maximum number of iterations: nr_iterations

// Starting with the Average Image
IF(frame_no==3)
        HR_B = Upsample(LR_I[0]+LR_I[1]+LR_I[2])+LR_I[3])        [SR_AVERAGE]
        HR_B = HR_B/4
END IF
// Iterations
FOR it = 1 .. nr_iterations
   LR_B = Downsample(HR_B)                                       [SR_DOWNSAMPLE]
   FOR fr = 0 .. nr_frames-1
          MV_fr2ref[fr] = Calc_Motion_Estimation (LR_I[fr], LR_B)
          Select_global_motion_vector()
          MV_ref2fr[fr] = - MV_fr2ref[fr]                        INVERT_MV()
          MV_fr2ref[fr] = 2 .* MV_fr2ref[fr]
          MV_ref2fr[fr] = 2 .* MV_ref2fr[fr]
      END FOR
      HR_A = 0                                                   [SR_INIT_A]
      FOR fr = 0 .. nr_frames-1
          HR_S = Motion_Compensation (HR_B, MV_ref2fr[fr])       [SR_MOT_COMP1]
          HR_S = Upsample(LR_I[fr])/2 – HR_S/2                   [SR_UPSAMPLE]
          HR_T = Motion_Compensation (HR_S, MV_fr2ref[fr])       [SR_MOT_COMP2]
          SR_S = HR_T
          HR_A = HR_A + HR_S/2                                   [SR_ADD]
      END FOR
      HR_B = HR_B + HR_A                                         [SR_UPDATE]
      variance = variance(HR_A)                                  [SR_STAT]
      If (variance < 0.5) break
END FOR
```

Figure 36. Pseudo-code of the iterative algorithm v1.2, modified using two memory sizes simultaneously

block format. Table 8 summarizes the memory requirements that an implementation of that version would demand as a function of the number of input macro-blocks. These data can be obtained from Table 2 and Table 3. The number of column macro-blocks has been labeled as *mb_x*, and the number of row macro-block has been labeled as *mb_y*. For instance, the HR_A memory would have a number of macro-block *(2· mb_x) ·(2· mb_y)*, because as it is a high-resolution image, its size is double in both directions. As every macro-block has 16×16-luminance pixels and 8×8-chrominance pixels and furthermore there exists two chrominance components, the blue and the red one, this will suppose that the overall pixel number is *(2·mb_x·2·mb_y·16·16)* for the luminance and *(2·mb_x·2·mb_y·8·8·2)* for the chrominance components. Nevertheless, it must be taken into account that the HR_A memory is 9 bits wide, so, to obtain the total number of bits we have to multiply for the 9 bits of each pixel. For the remaining memories 8 bits per pixel must multiply every memory pixel size.

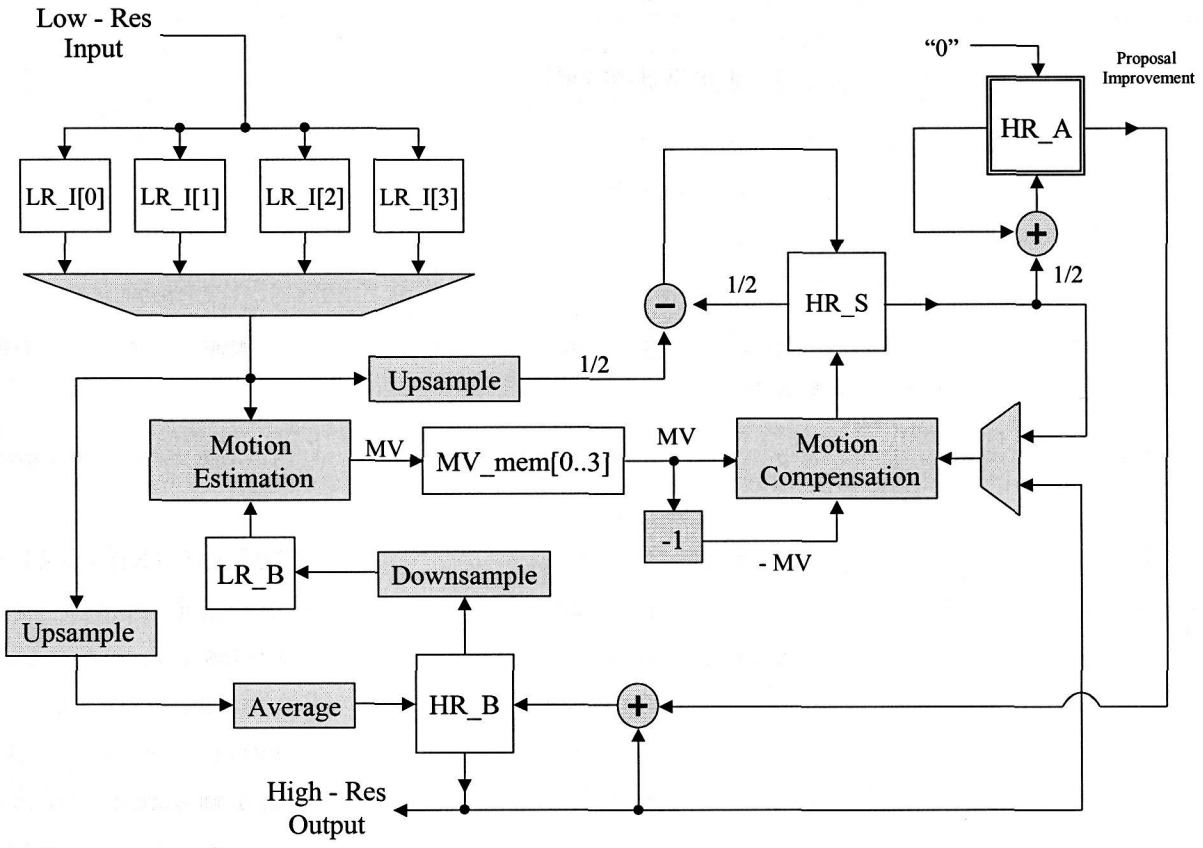


Figure 37. Block diagram of the data-flow for the super-resolution algorithm v1.2

Label	Memory		
	Luminance (bits)	Chrominance (bits)	Total (bits)
HR_A	*(2·mb_x·2·mb_y·16·16·9)*	*(2·mb_x·2·mb_y·8·8·2·9)*	*13,824·mb_x·mb_y*
HR_B	*(2·mb_x·2·mb_y·16·16·8)*	*(2·mb_x·2·mb_y·8·8·2·8)*	12,288·*mb_x·mb_y*
HR_S	*(2·mb_x·2·mb_y·16·16·8)*	*(2·mb_x·2·mb_y·8·8·2·8)*	12,288·*mb_x·mb_y*
3 Stripes HR	*(2·3·2·mb_y·16·16·8)*	*(2·3·2·mb_y·8·8·2·8)*	36,864·*mb_y*
LR_B	*(mb_x·mb_y·16·16·8)*	*(mb_x·mb_y·8·8·2·8)*	*3,072·mb_x·mb_y*
LR_I[0]	*(mb_x·mb_y·16·16·8)*	*(mb_x·mb_y·8·8·2·8)*	*3,072·mb_x·mb_y*
LR_I[1]	*(mb_x·mb_y·16·16·8)*	*(mb_x·mb_y·8·8·2·8)*	*3,072·mb_x·mb_y*
LR_I[2]	*(mb_x·mb_y·16·16·8)*	*(mb_x·mb_y·8·8·2·8)*	*3,072·mb_x·mb_y*
LR_I[3]	*(mb_x·mb_y·16·16·8)*	*(mb_x·mb_y·8·8·2·8)*	*3,072·mb_x·mb_y*
MV_mem[0]	(mb_x·mb_y·8)	0	8· *mb_x·mb_y*
MV_mem[1]	(mb_x·mb_y·8)	0	8· *mb_x·mb_y*
MV_mem[2]	(mb_x·mb_y·8)	0	8· *mb_x·mb_y*
MV_mem[3]	(mb_x·mb_y·8)	0	8· *mb_x·mb_y*
Total (bits)	mb_y· (35,872·mb_x + 24,576)	mb_y· (17,920· mb_x + 12288)	mb_y · (53,792 mb_x + 36,864)

Table 8. Summary of the memory used in version v1.2 of the super-resolution algorithm in function of the number of macro-blocks.

Table 9 shows the memory requirements of the algorithm for the memory sizes previously commented in *Kbytes* and *Mbytes,* while in Figure 38 these data are shown in a graphical way, expressed in *Kbytes*. In the chart can be appreciated the considerable increase in the memory requirements, due to the non-lineal behaviour of equations in Table 8. The memory HR_T has not been included in the list for the previously exposed reason, and instead of it, we have included the size if three additional macro-block slices, included in the Picasso architecture to avoid the data overlap when performing the motion compensation.

Nevertheless, it must be taken into account that the size in the abscissas axe is the input image size in low-resolution and that the output is double size.

Size	mb_x	mb_y	Memory (Kbytes)	Memory (Mbytes)
SQCIF	8	6	342.19	0.33
QAVGA	9	7	445.18	0.43
QCIF	11	9	690.57	0.67
HAVGA	18	14	1,717.73	1.68
CIF	22	18	2,681.30	2.62
AVGA	36	28	6,744.94	6.59
VGA	40	30	8,014.69	7.83
4CIF	44	36	10,563.19	10.32
16CIF	88	72	41,928.75	40.95

Table 9. Memory used in version v1.2 of the super-resolution algorithm for different image sizes.

5.2.4 Experimental set-up adjusts to enable reliable measures of version v1.2

Even though version v1.2 of the algorithm offers very good perceptual quality image, it presents an important problem related to the signal to noise computation. Concretely, the method of obtaining the first super-resolution image proposal as the average among the input images produces a shift of this first image with respect to the reference, as it is depicted in Figure 39. The sub-sample process picks up different image pixels from the real image to generate new low-resolution images with aliasing. The problem is that when the average of these low-resolution images is carried out, the pixels are shifted ½ pixel (in the example) with respect to the real or reference image. This shift is kept through the overall reconstruction process producing a drastically drop of the signal to noise ratio, because the reference image and the super-resolution image are on different grids.

The adopted solution, in order to enable the measures between the super-resolution image and the reference, has been to assure that the generated vector-set has zero mean. In that sense, the resulting super-resolution image is adjusted to position (0,0) that matches with the pixel position for the reference image. Moreover, this solution offers the advantage of modelling the movement produced during manual recording without subjection or supporting

system of a video sequence using a domestic video camera. Normally, the human being will try to keep the video-camera centred over the target, in spite of the involuntary movement produced while trying to hold the camera by hand. This attempt to keep the target centred will produce relative shifts with a close-to-zero mean, if the interval between frames is not too short. To obtain that kind of simulated motion vectors, it has been developed a program that generates series of random vectors with zero-mean. An example of that kind of series is the one shown in Table 10, where a sequence of 40 motion vectors has been generated, giving as a result of its application 10 super-resolution output images.

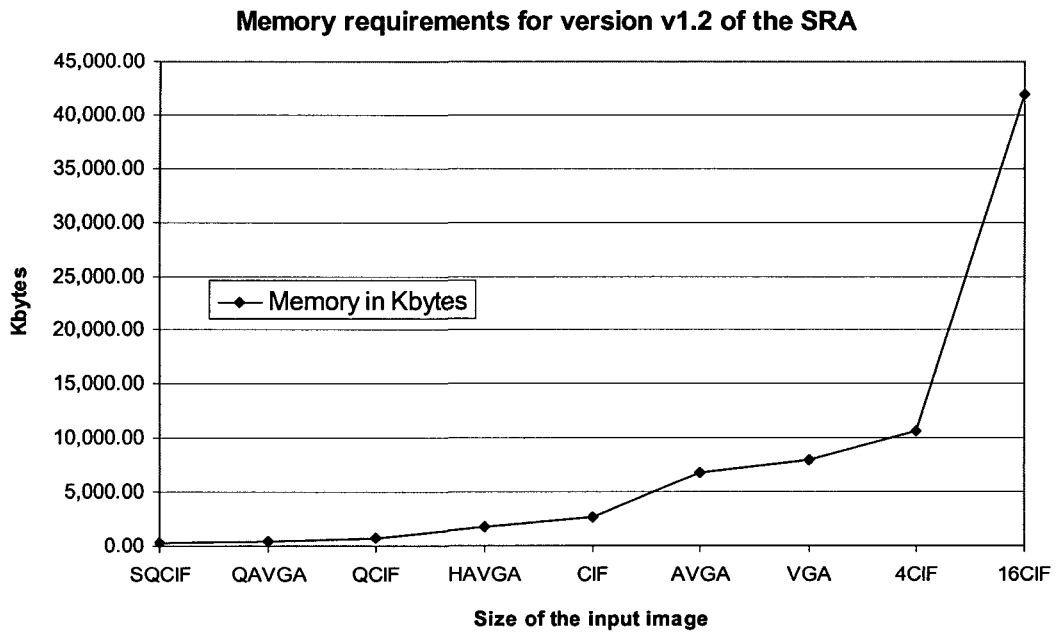


Figure 38. Memory used by the super-resolution algorithm v1.2 for the most common memory sizes.

Figure 40(a) shows the reference image of CIF format (342×288) labelled as KANTOOR. It is of great interest that there exists a foreground without too many details and a background plenty of details that will be loose in the sub-sample process for being recovered thanks to the super-resolution process. The first experiment has consisted in the generation of a sequence of random moving vectors with distances of two pixels in the same reference image repeated ten times, so that after decimation these distances will be reduced to a pixel. From each shifted reference image, four QCIF size (176×144) low-resolution images are obtained by decimation. For the first-image case the sequence (b), (c), (d) and (e) of Figure 40 has been generated. These images have been down sampled below the Nyquist frequency, and because of that, they will contain some amount of aliasing manifested in the time domain as

an information loss, especially in the edges (high frequency zones). For example, in the low-resolution images (b) and (c) the lower inner rectangle of the computer monitor has disappear and in all of them, the details of the objects on the table are unrecognisable. Also, it is appreciated that the quality of the straight lines has been degraded verifying a strong pixel-blocking effect in almost all the edges of the objects that appears in the scene.

In the first column of Table 10 the frame number generated in the super-resolution process is shown. The second column shows the vectors set that, applied to the reference frame, has been used for the output image reconstruction. The third column shows the low-resolution vectors, i.e. dividing by two theirs '*x*' and '*y*' components, while the fourth column shows the reduction of the vectors to its canonical form previously defined. The meaning of this reduction will be explained later on, with the PSNR results analysis.

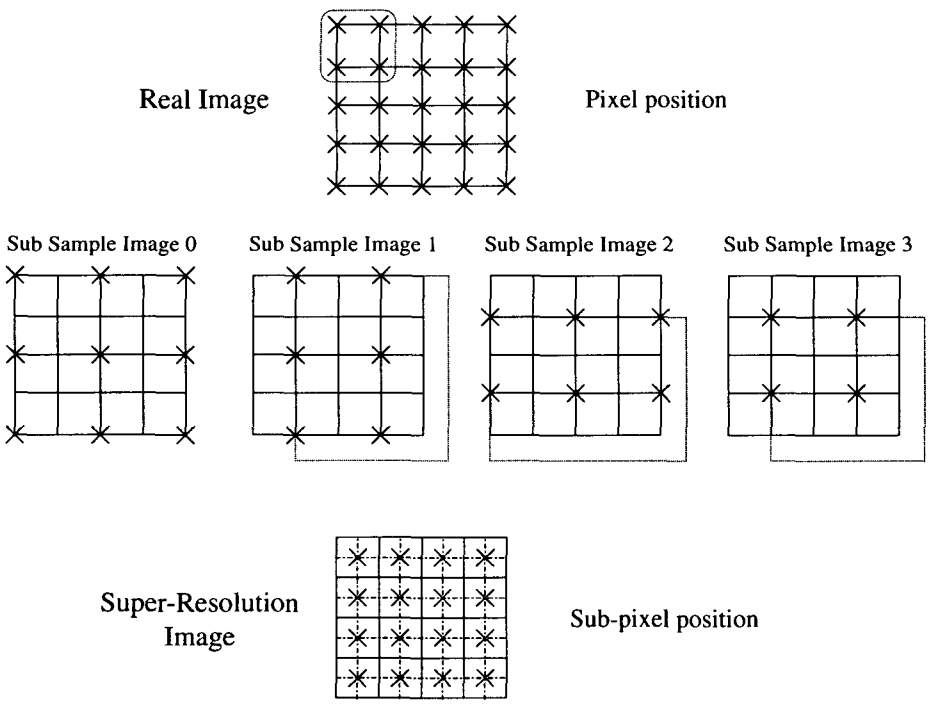


Figure 39. Sub-pixel shift effect when using as first proposal for super-resolution the average of the input images.

(a)

(b) (c) (d) (e)

Figure 40. Used images for the iterative algorithm. The image (a) of CIF size is the reference. From it are obtained by decimation the images (b)-(e) of low-resolution and QCIF size, which constitute the first input set to the super-resolution algorithm.

5.2.4.1 Quality analysis in the spatial domain

In Figure 41, Figure 42 and Figure 43 the result of executing the version v1.2 of the super-resolution algorithm on the input data previously commented (40 low-resolution images) after performing only two iterations of the algorithm are shown. The outputs are ten super-resolution images of double size that the input images. Likewise, next to the images their related error-images in the spatial domain are shown. This error is obtained as the subtraction between the reference image (before sub-sampling) and every super-resolution output image. The zero level has been shift to grey in order to accommodate positive and negative differences, which means that the great amount of grey colour is in the image, the better the super-resolution image will be, in the sense that both, the reference and the super-resolution images, will be more similar.

At a first glance it can be appreciated an important degradation of the output image (h) corresponding to the frame number seven. The signal to noise ratio is one of the preferred metrics when measuring the quality of a signal and that is why it has been the metric adopted

for images in the spatial domain. As a reference, the interpolated images using each one of the previously mentioned methods are shown in Figure 44.

Frame	High-resolution vectors		Low-resolution vectors		Canonical vectors reduction		*Frame*	High-resolution vectors		Low-resolution vectors		Canonical vectors reduction	
0	0	-2	0	-1	0	1	5	-2	-4	-1	-2	1	0
	-2	0	-1	0	1	0		0	2	0	1	0	1
	2	0	1	0	1	0		4	0	2	0	0	0
	0	2	0	1	0	1		-2	2	-1	1	1	1
1	-2	0	-1	0	1	0	6	2	0	1	0	1	0
	0	2	0	1	0	1		0	-4	0	-2	0	0
	-2	-4	-1	-2	1	0		-2	2	-1	1	1	1
	4	2	2	1	0	1		0	2	0	1	0	1
2	-2	0	-1	0	1	0	7	0	2	0	1	0	1
	-2	-2	-1	-1	1	1		-2	-2	-1	-1	1	1
	0	-2	0	-1	0	1		2	2	1	1	1	1
	4	4	2	2	0	0		0	-2	0	-1	0	1
3	2	2	1	1	1	1	8	2	2	1	1	1	1
	2	0	1	0	1	0		-2	-2	-1	-1	1	1
	0	-2	0	-1	0	1		0	0	0	0	0	0
	-4	0	-2	0	0	0		0	0	0	0	0	0
4	0	0	0	0	0	0	9	2	-2	1	-1	1	1
	2	2	1	1	1	1		2	2	1	1	1	1
	0	-2	0	-1	0	1		-2	0	-1	0	1	0
	-2	0	-1	0	1	0		-2	0	-1	0	1	0

Table 10. Moving vectors randomly generated with zero mean, with two pixels distances for high-resolution, of one pixel to low-resolution and its reduction to canonical vectors.

In Figure 45 the PSNR of every image of the output sequence together wit the PSNR of the interpolated images are presented. We will call 'interpolated level' to the PSNR of the image obtained through bilinear interpolation. It is very helpful in the sense that it establishes the level above that super-resolution it is really useful. If the quality of the output image was below the interpolation level, it means that with interpolation we would reach better quality than with super-resolution, and this last method is not worthwhile. Anyway, obtaining PSNR below the interpolation level uses to be symptomatic of any kind of errors, either in the reconstruction or in the result analysis processes. The difference between the super-resolution PSNR and the bilinear interpolated PSNR shows the quality super-resolution gain versus the interpolation techniques.

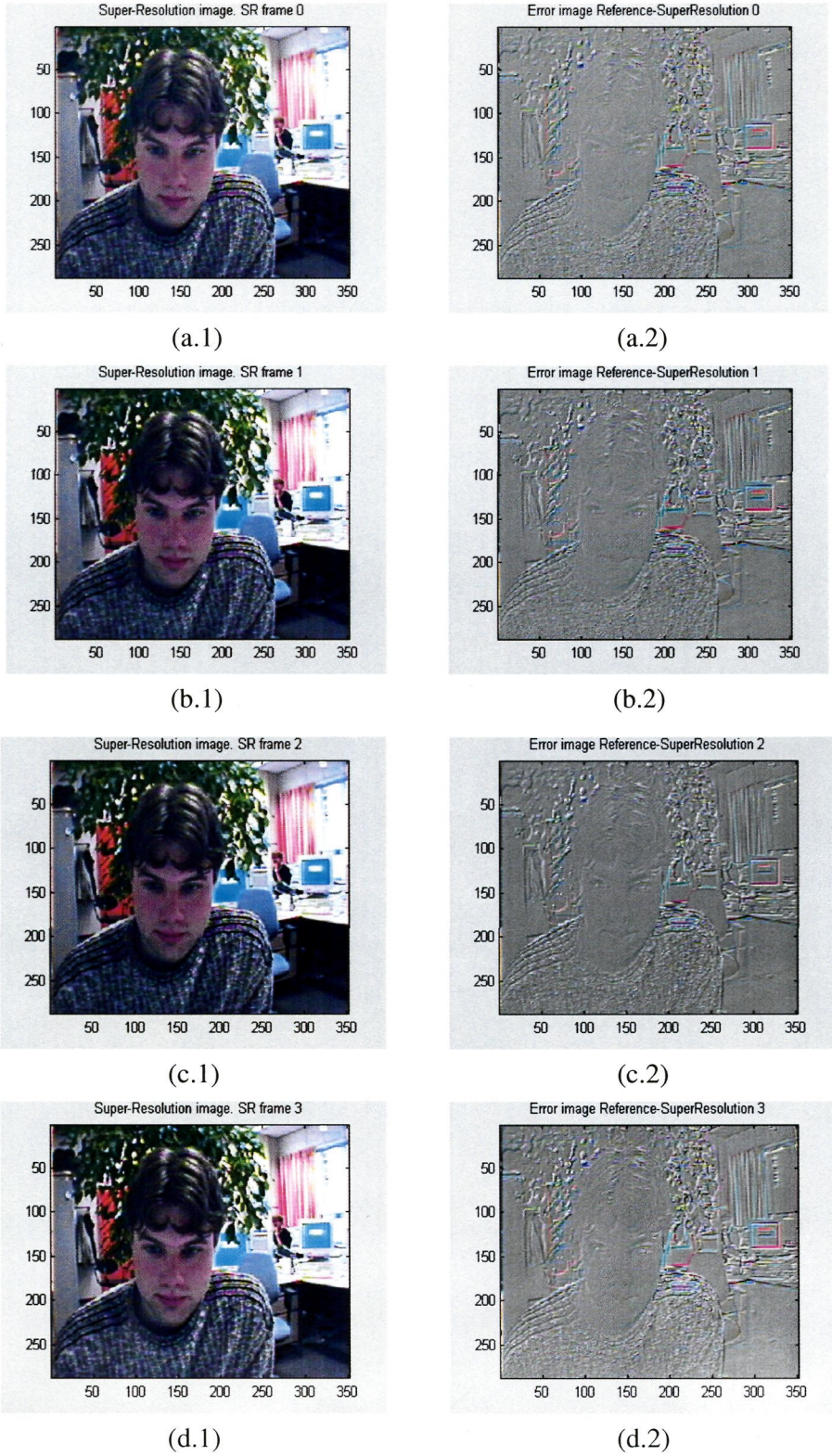
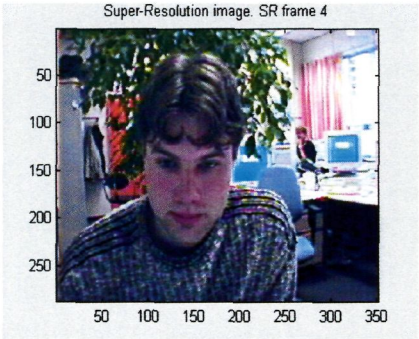
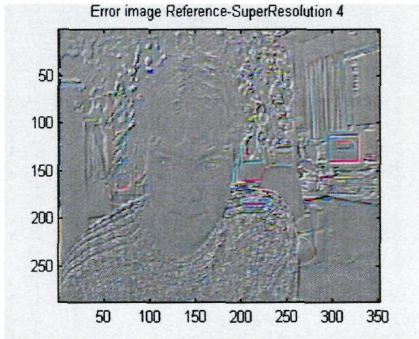


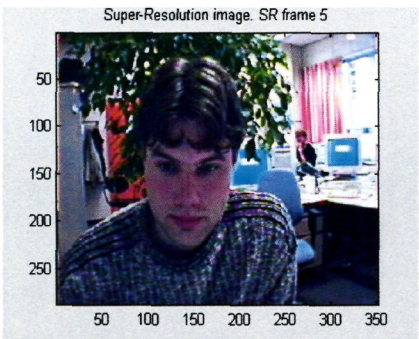
Figure 41. CIF images obtained with the super-resolution algorithm version v1.2. The (a)-(d) sequence correspond to frames 0 to 3 respectively.



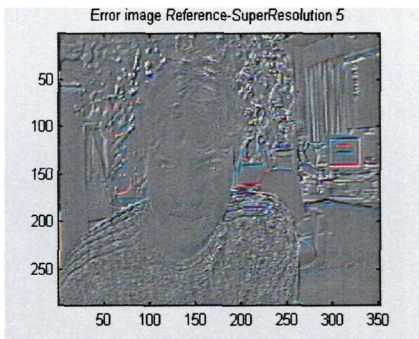
(e.1)



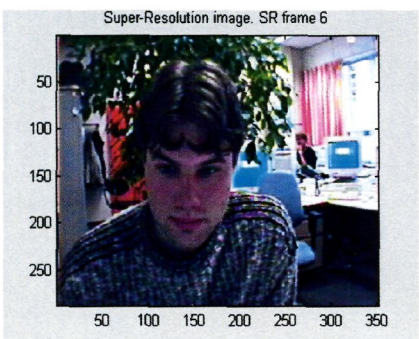
(e.2)



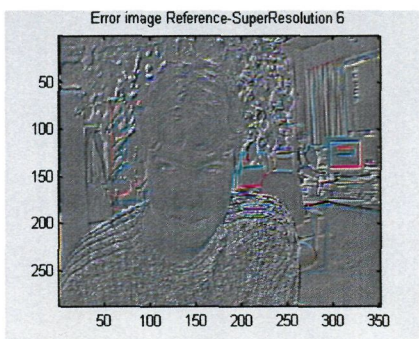
(f.1)



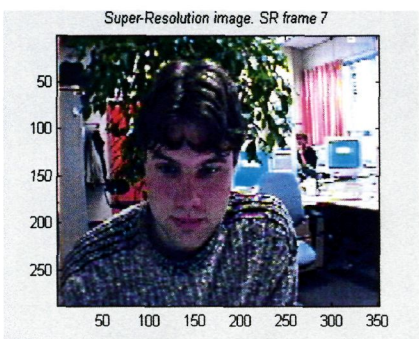
(f.2)



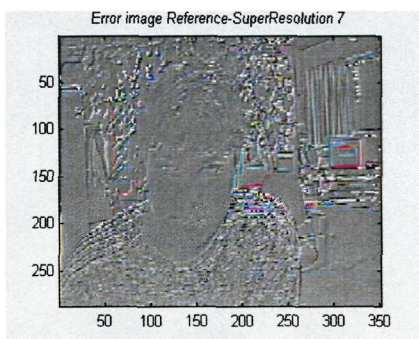
(g.1)



(g.2)

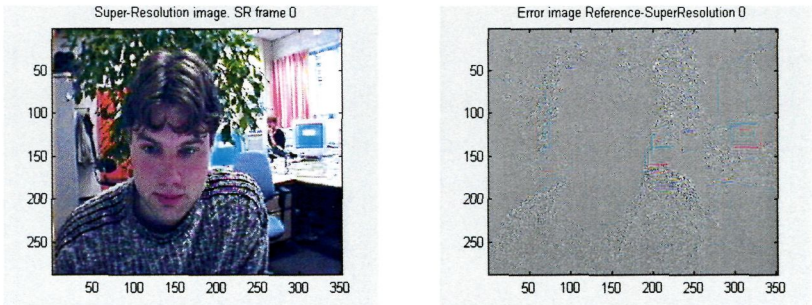


(h.1)

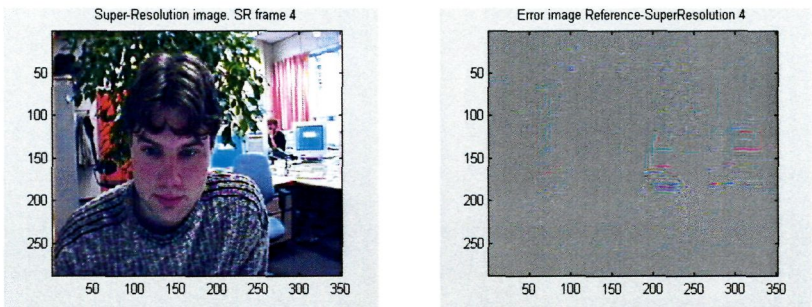


(h.2)

Figure 42. CIF images obtained with the super-resolution algorithm version v1.2. The (e)-(h) sequences correspond to frames 4 to 7 respectively.

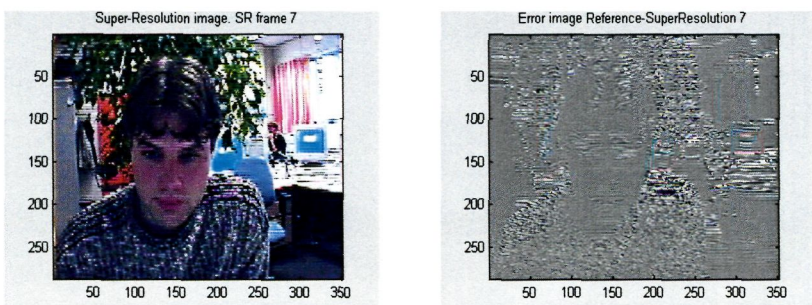


(a.1) (a.2)

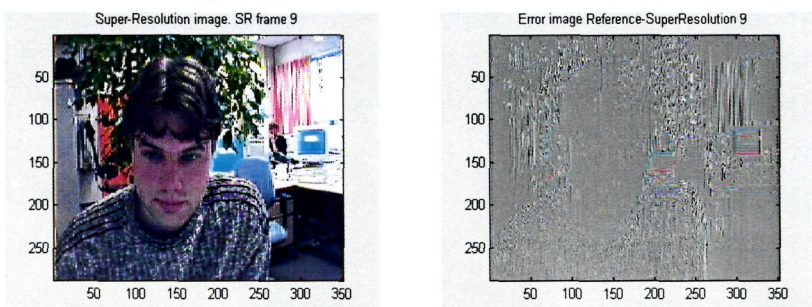


(b.1) (b.2)

Figure 56. Super-resolution images (1) and them associated errors (2) for frames 0 (a) and 4 (b) after 80 iterations per frame.



(c.1) (c.2)



(d.1) (d.2)

Figure 57. Super-resolution images (1) and them associated errors (2) for frames 7 (a) and 9 (b) after 80 iterations per frame.

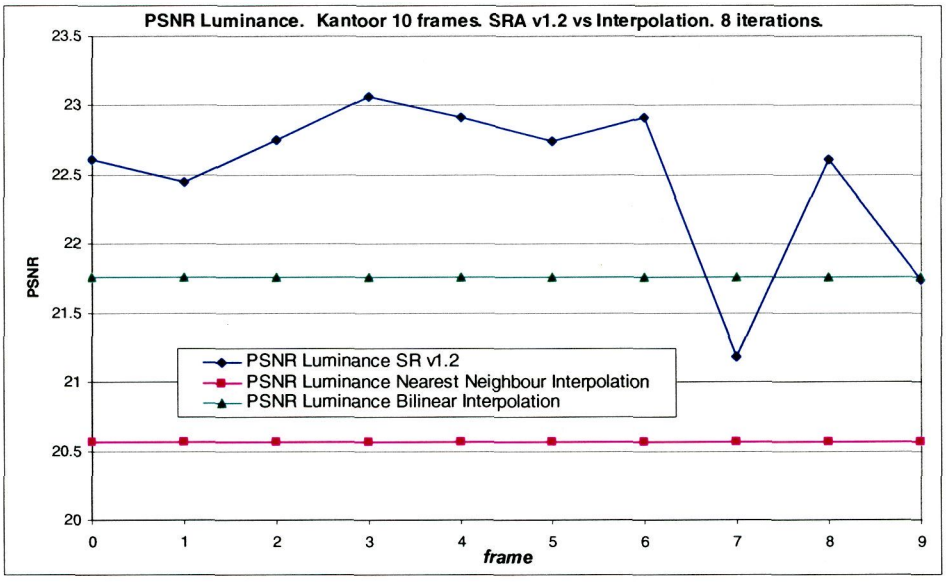


Figure 45. PSNR of the super-resolution output sequence (version v1.2) versus the interpolated images using nearest neighbour replication and bilinear interpolation.

5.2.4.2 Quality analysis in the spatial frequency domain

The first metric used to quantify the similitude degree between the reference image and the one generated through the super-resolution process is the peak signal to noise ratio, figure obtained in the signal spatial domain. However, the likeness degree in the frequency domain (spatial frequency) also adds valuable information. So, with the objective of measure the likeness degree between the image spectrums, we compute the bidimensional correlation coefficient between the reference and the super-resolution images. Due to the fact that the Fourier bidimensional transforms are complex values, this will give a magnitude and a phase correlation. In Figure 45 the values of the bidimensional Spectral Correlation Coefficient (SCC) of the magnitude and the phase for the referred output sequence are shown.

As it is seen in the chart, the magnitude spectral correlation is substantially greater than the phase one, mainly because the phase gather more information about the relative shifts among the images, and as the super-resolution image is a composition of several shifted images, the final resulting image is slightly shift with respect to the reference. For the same reason, the magnitude spectral correlation coefficient of the bilinear interpolation is greater than the correlation of the nearest neighbour interpolation. Nevertheless, it is just the opposite

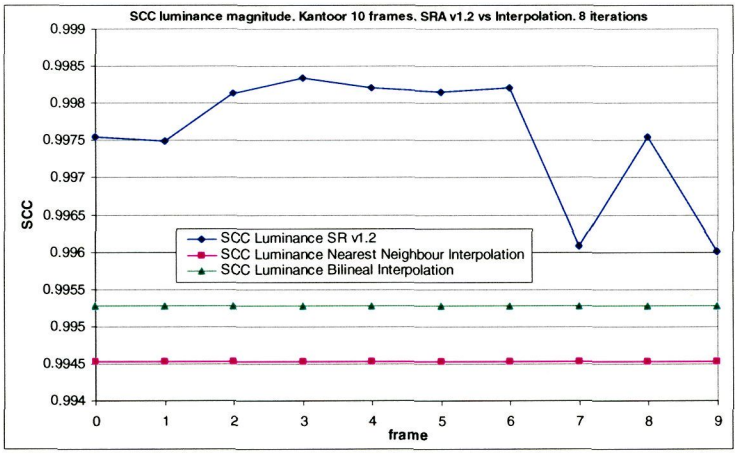
situation with the phase spectral correlation. Filling the intermediate values of the image obtained by bilinear interpolation with average values is equivalent to fill them with shifted versions of the image, which is reflected in a lower phase spectral correlation. In spite of this, the magnitude spectral correlation, much more sensitive for the human eye as it is more related to luminosity variations, reaps very high correlation figures, in all the cases above 0.996, reaching 0.998 occasionally.

Another aspect that is worldwide to be pointed out is the similar behavioural of the metric figures employed: notice that the first two frames (0 and 1) exhibit similar qualities in the three metrics (PSNR, phase SCC and magnitude SCC), always below to the subsequent five frames (2, 3, 4, 5 and 6). Frame 8 shows a quality similar to the initial frames and in frames 7 and 9 the qualities significantly fall.

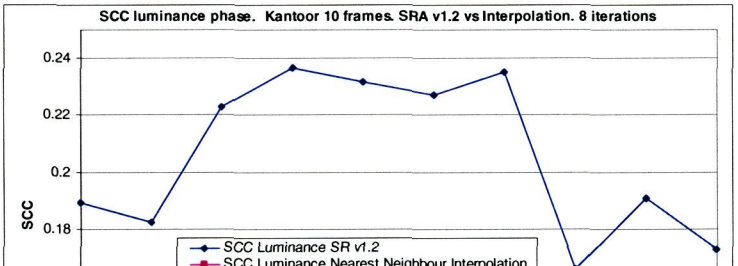
Although the information contributed by the bi-dimensional Fourier transforms in magnitude and phase about the quality of the resulting images is difficult to visually evaluate, it is much richer in the observation of the error images between the reference image spectrum and the spectrums of the interpolated and super-resolution images. In Figure 47 the bi-dimensional Fourier transforms in magnitude (a) and phase (b) of the reference image are shown.

These images are the ones that serve as reference to show the error in magnitude and phase in the interpolated and super-resolution images. Figure 48 shows the magnitude and the magnitude error of the interpolated images and Figure 49 shows the phase and phase error of the same interpolated images.

In the case of Figure 48 it can be appreciated as in both cases the error is lower in the low frequencies zone, where it can be seen a high homogeneous central zone of low error. In the case of the phases represented in Figure 49 is clear that the homogeneous zone is of low extension, which is coherent with the low correlation exhibit. As it is usual in the image Fourier transforms, the phase contributes with low visual information, and although normally they are not taken into account, we have decided to kept them for two main raisons: firstly for completeness, and secondly because evaluating the error we can obtain additional information about the similitude degree between the image phases, which is crucial in this kind of applications.

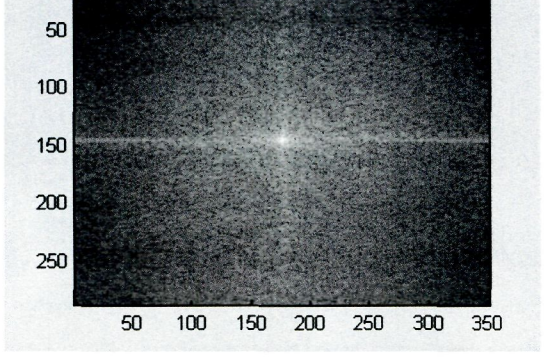


(a)

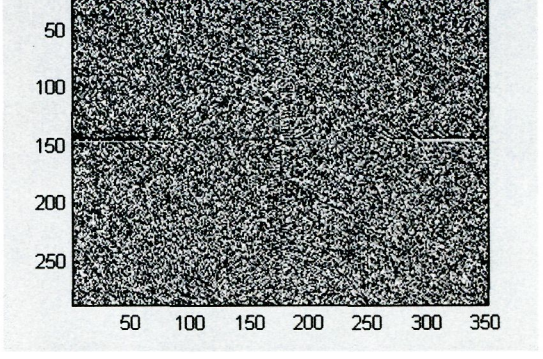


(b)

Figure 46. Spectral correlation coefficients in magnitude (a) and phase (b).



(a)



(b)

Figure 47. Bidimensional Fourier transforms in magnitude (a) and phase (b) of the reference image.

Figure 50 shows the bi-dimensional phase Fourier transforms and its associated error for same frames of the super-resolution sequence. Specifically, frames 0, 2, 7 and 9 are shown because they have the most abrupt variations both in the PSNR and in the spectral correlations. It is shown that in the low frequency zones the frame number 2 has an error lower than the rest of frames, which is coherent with all the obtained metrics. It is quite peculiar to see that frame 7 has less error in the horizontal frequencies zone, while frame 9 has exhibits a similar behaviour but in the vertical frequencies zone. The phase analysis of these same images, shown in Figure 51 offers the same behaviour that the magnitude. This phenomenon has supposed an important clue for the considerations reached in the following sections about the quality variations of the super-resolution images.

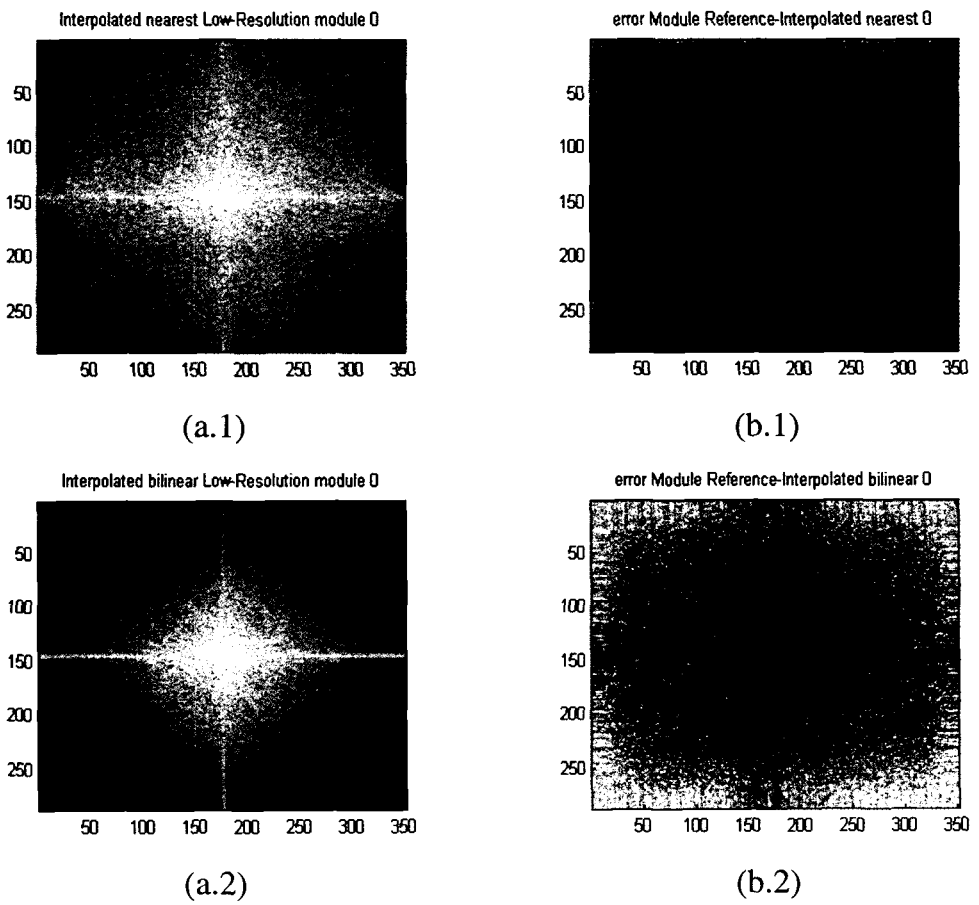
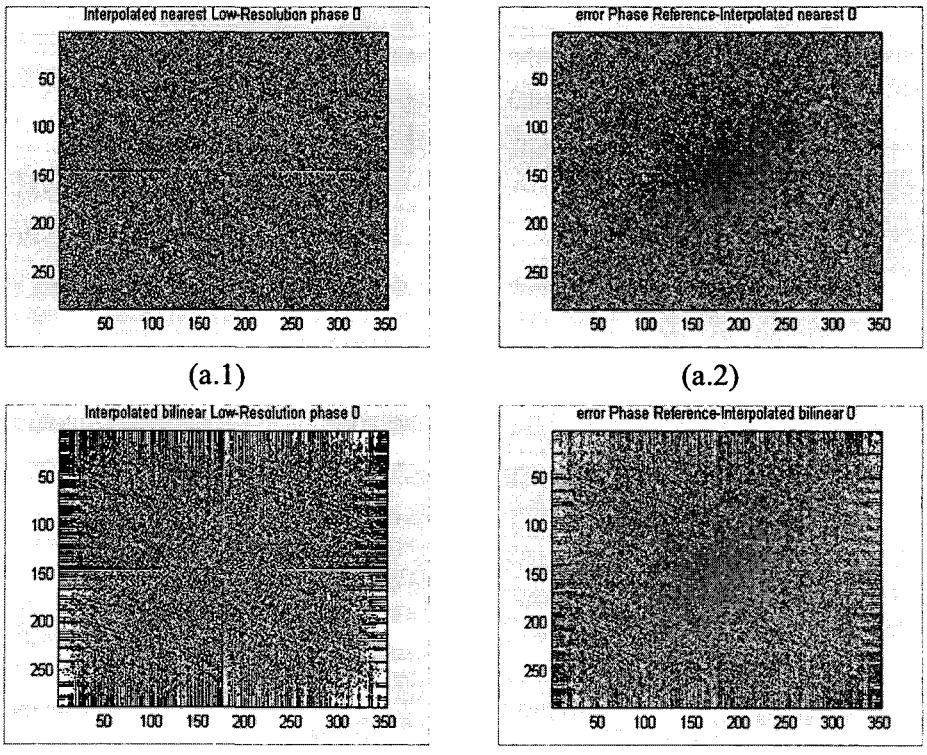


Figure 48. Bi-dimensional Fourier transforms in magnitude (a) and their associated error (b) for the interpolated images.

5.2.4.3 Borders effect

As previously shown, the frame number seven experiments a drastically drop of its quality, falling even down the interpolation level. Another important fact is that, being always the same reference picture, the qualities are all very diverse, against the expected. As the input frame is always the same, we expected a similar output quality, but as it is not the case, this fact indicates that the motion of the frame has an important effect on the final image quality. As previously established, the border effect could drastically drop the PSNR. A simple way to remove this problem of the measure procedure consists of redefining the computation of the PSNR, excluding the image borders in a certain number of pixels *'q'*. Initially this clip factor *'q'* is made equal to 16 pixels, i.e. one macro-block. The new peak signal to noise ratio, call from now on PSNR' or PSNR without borders, is given by equation (23).

$$PSNR' = 10 \cdot \log_{10} \left(\frac{255^2}{\frac{1}{(M-2\cdot q)\cdot(N-2\cdot q)} \sum_{i=q}^{M-1-q} \sum_{j=q}^{N-1-q} \left(R(i,j) - SR(i,j)\right)^2} \right) \tag{23}$$



(a.1) (a.2)

(b.1) (b.2)

Figure 49. Bi-dimensional Fourier transforms in phase (a) and theirs associated error (b) for the interpolated images.

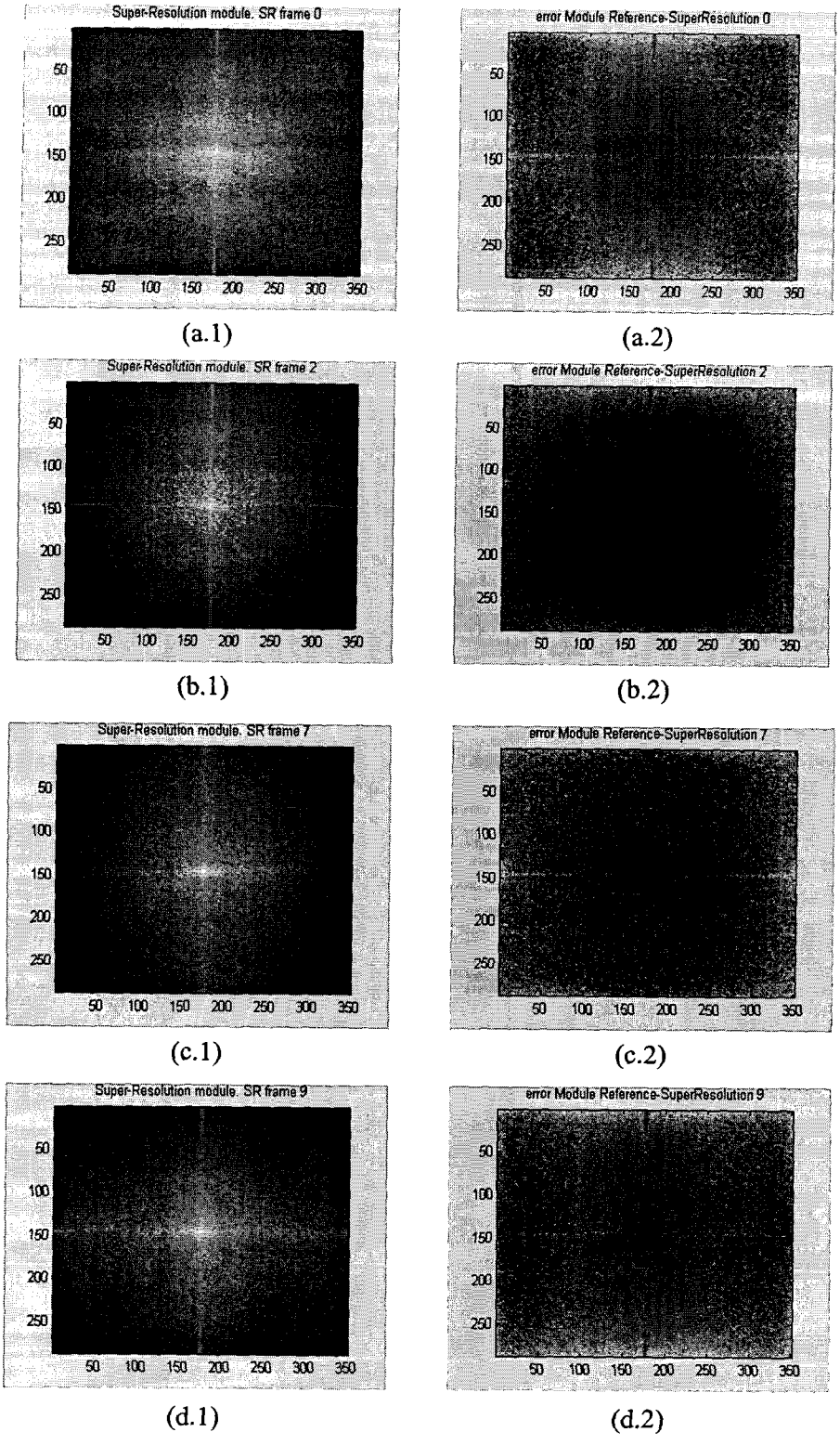


Figure 50. Bi-dimensional Fourier transforms in magnitude and theirs associated errors for the super-resolution images 0 (a), 2 (b), 7 (c) and 9 (d).

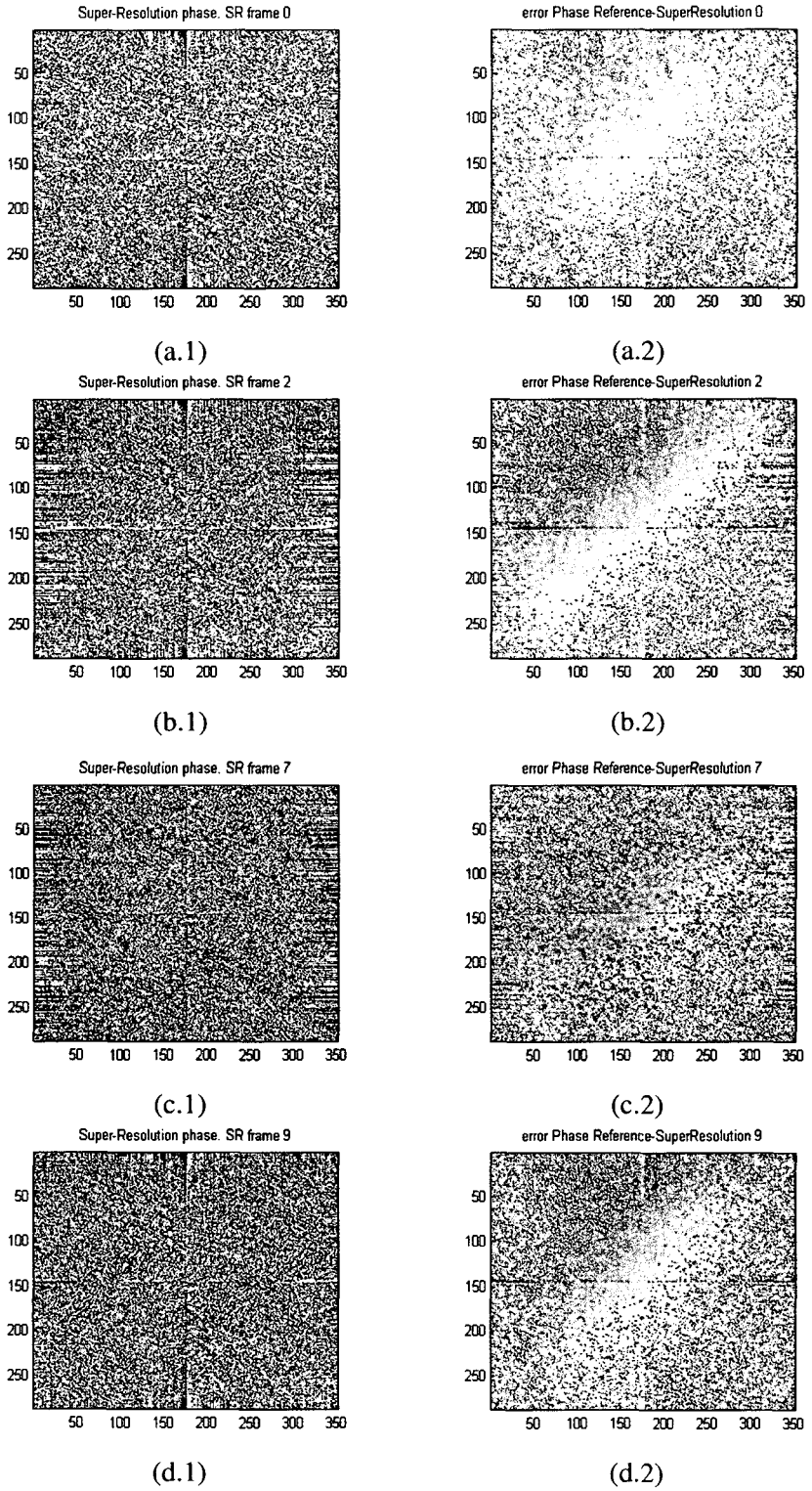


Figure 51. Bi-dimensional Fourier transforms in phase and theirs associated errors for the super-resolution images 0 (a), 2 (b), 7 (c) and 9 (d).

Figure 52 shows the PSNR' for the same sequence, where it can be seen a lower variation of the signal to noise ratio. Nonetheless, we can also appreciate how several frames share the same PSNR', as it is the case of frames 0, 1, and frames 2, 3, 4, 5 and 6. The explanation of this phenomenon will result in one of the main physic limitations in the images quality improvements using super-resolution techniques.

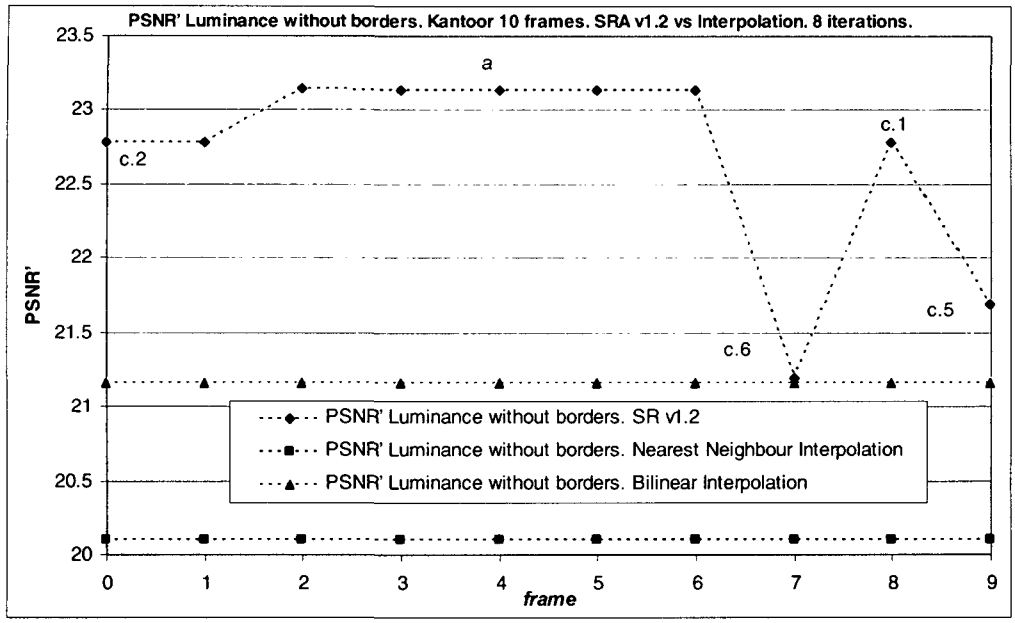


Figure 52. PSNR' of the output super-resolution sequence jointly with the PSNR' of the interpolated images using nearest neighbour and bilinear interpolation.

5.2.5 Sampling issues: an image classification

Obviously, the quality differences must rely on the shift differences among the input images, as the remainder factors remains unalterable. If we analyse in detail the sub-sample process, it can be observed, as shown in Figure 53 that motion vectors completely different in appearance can generate equivalent samples in the main body of the image, i.e. excluding the borders effect.

In this chart the same picture is shifted using the motion vectors (0,1) and (2,-3) and they are subsequently sub-sampled. The sub-sampling consists in select the pixels (represented as crosses) enclosed in a circle. It has been signalled with dot lines what would

suppose the new borders of the sub-sampled image. As it is noticed, the selected pixels (except for the borders) are exactly the same, and therefore the second shift will not add new information with respect to the first one. Even if we always start from four displacements, apparently different, it is obvious that if there exist equivalent shifts among them, the image quality will result degraded compared with another displacement set where every one will contribute with new information that enables the quality improvement of the details of the original image.

Since the used sub-sampling system resides in taken alternatively pixels from the original image, beginning from the generated shift, it is clear that any shift can be reduced to the base cell described in Figure 10, or in the same way, every motion vector can be reduced to its canonical vector. If two displacement vectors can be reduced to the same canonical vector, then, the supplied information is the same. The canonical vector $(\tilde{x},\tilde{y})$, of the displacement vectors with *(X,Y)* coordinates is obtained as indicated in equation (24).

$$\begin{aligned} \tilde{x} &= |X \bmod 2| \\ \tilde{y} &= |Y \bmod 2| \end{aligned} \tag{24}$$

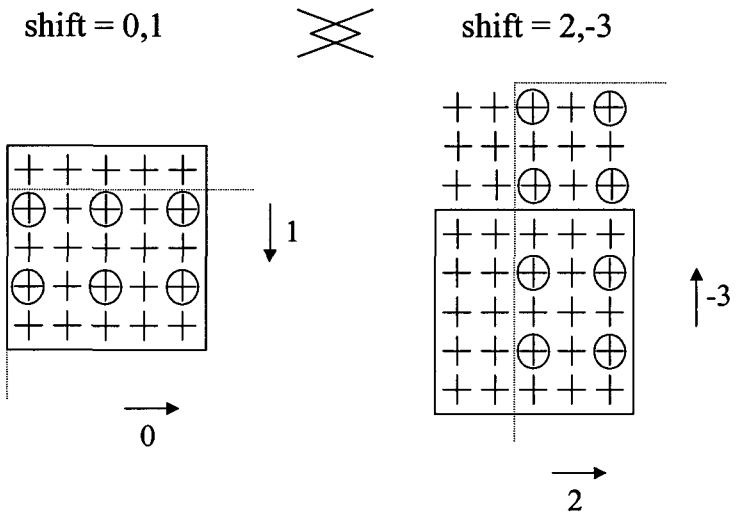


Figure 53. Effects of the equivalent sampling using equivalent displacement vectors.

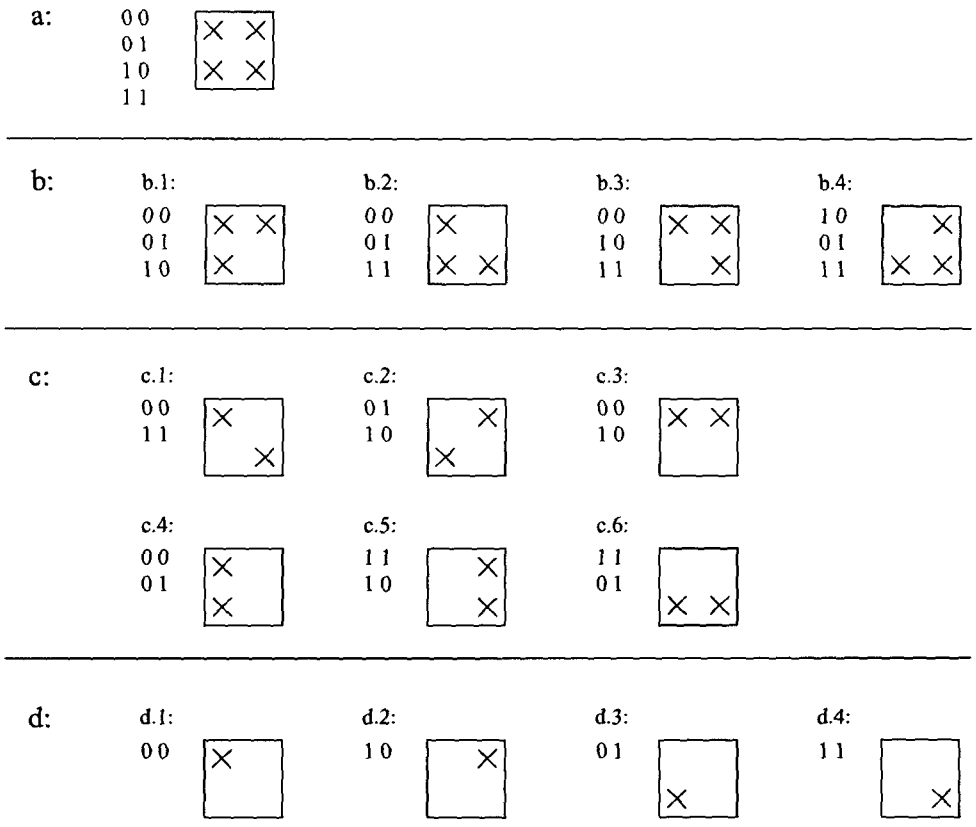


Figure 54. Classification of the super-resolution images as a function of canonical vectors from which they were reconstructed.

In Table 10 the generated displacement vectors are shown, where it is also included the reduction to canonical vectors. It can be realized that frames 0 and 1 have the same canonical vectors [(0,0),(1,1)] and therefore the same PSNR without borders: 22.78 dB. The same happens with frames 2, 3, 4, 5 and 6, which share the same canonical vectors [(0,0),(0,1),(1,0),(1,1)], which fill all the positions in the base cell, therefore giving the maximum PSNR', i.e. 23.13 dB.

Depending upon the number and position of the pixels of the base cell present after the sub-sampling process, or what it is the same, depending on the canonical vectors from what the image was reconstructed, we establish a classification shown in Figure 54. The first-level classification, labelled with a lowercase letter, attends to the number of different pixels that were sampled from the original image, and the second-level classification attends to the position of these pixels in the base cell. For instance, a type 'a' image is composed by the four original pixels, and so, it always will have the greater PSNR'. In the opposite case, images of type 'd' have been reconstructed using solely a pixel from the original image, and

its quality, of course, will be obviously lower. Retaking again the problem of frame 7 and according to this classification, it will be a 'c.6' image, i.e., it has been generated using only two pixels of the original image, but the two lower pixels, therefore loosing all the horizontal information of same rows, giving a lower quality than type 'c.1' images (frame 8) or 'c.2' images (frames 0 and 1). Although these later images have also been generated using only two pixels from the original image, the pixels correspond to alternate positions in the borders of the base cell, which allows recovering more information from the original image. The case for frame 9, of type 'c.5' is similar to the frame 7 case, but in that case pixels are in vertical position, producing a new drop of the PSNR'. In Figure 52 every frame has been marked with the corresponding label to the established classification.

5.2.6 Quality behavioural with respect to the number of iterations of version v1.2

The PSNR reached by the frame number 7 is practically the same as the bilinear interpolation level because of the reduced number of iterations. If the number of iterations were raised up, the algorithm would try to match the result to the available set of samples, and, as some samples are not available, these data will progressively miss the alignment. This fact is qualitative reflected as distorted lines or points in the super-resolution image, and quantitative manifested as a drastically drop of the PSNR.

In Figure 55 the evolution of the PSNR for the same set of displacement vectors during 80 iterations per output image is presented. Until now only the super-resolution image PSNR after 8 iterations has been shown, but in this chart the PSNR reached in every iteration is represented. The frame number is placed in boldface near every evolution curve. The obtained results perfectly fit the expected ones, in the sense that the images of type 'a' tend to increase their quality with the number of iterations, while the remainder images will decrease their quality with the iterations because of the previously established principle of adjust to the missing data. Also was expected a lower PSNR for the images of type 'c.5' and 'c.6' compared to the images of type 'c.1' and 'c.2' because of irregular distribution of the sampling process.

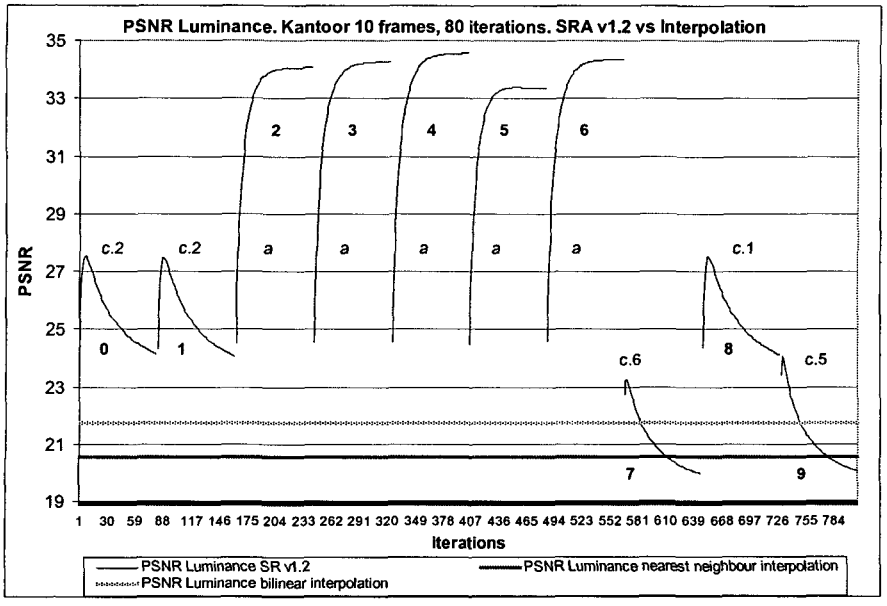


Figure 55. PSNR evolution during 80 iterations per frame of the SRA versus the interpolated images.

In Figure 56 and Figure 57 the super-resolution images after 80 iterations per frame are displayed. As we did previously, only the frames of interest will be displayed, i.e. the frames number 0 (type 'c.2'), 4 (type 'a') 7 (type 'c.6') and 9 (type 'c.5').

It must be noticed, especially in the error images, that the error is lower in the images of type 'a', because the entire original samples from the original high-resolution image are present in the reconstruction process. Even in absence of half of the original samples, as it is the case of the frame number 1 of type 'c', the reconstruction process offers good results if, at least one sample per line is available. However, if both samples belong to the same line, the quality of the image will be seriously decreased, appearing, as it can be appreciated in Figure 57 horizontal and vertical lines depending on the missing data.

It is interesting to point out that the quality of the images could be better if the iterative process would be interrupted in the first stages, as it can be appreciated in the Figure 55. Except for images of type 'a', the rest of them reach a maximum PSNR in the first iterations and then, start to drop down.

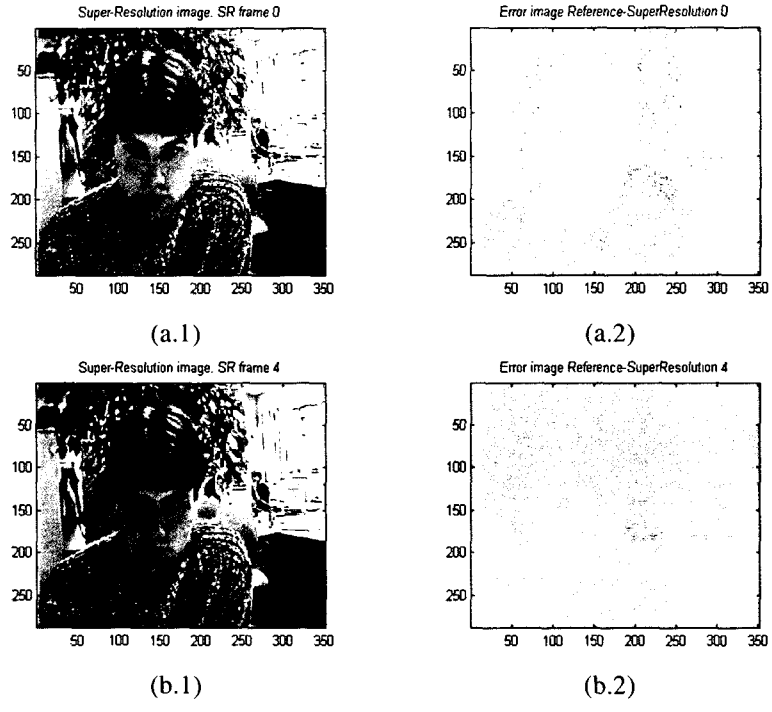


Figure 56. Super-resolution images (1) and them associated errors (2) for frames 0 (a) and 4 (b) after 80 iterations per frame.

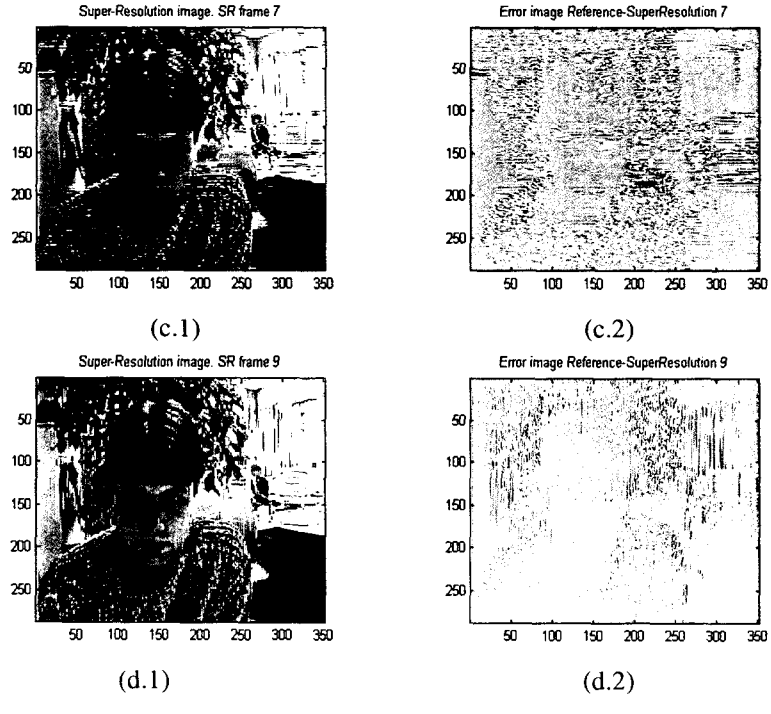


Figure 57. Super-resolution images (1) and them associated errors (2) for frames 7 (a) and 9 (b) after 80 iterations per frame.

The PSNR decrease is produced because in the first iterations the missing high-resolution values are filled with average values, what act as an interpolation, rising the PSNR, but while the number of iterations is increased, and with the aim of minimizing the error among between the super-resolution image and the set of available input images, the missing samples gradually loose weight in the super-resolution process, resulting in a bad alignment of the samples. In Figure 58 and Figure 59 are shown the magnitude and phase as well as the associated error with respect to the reference.

In Figure 58 it is verified that in the case of images of type 'a' (Figure 58 b.2), the spectral error in magnitude is very low for almost all the frequency range, especially for the low frequencies. For images of type 'c.1' and 'c.2' (Figure 58 a.2) the error is also very low, although it does not penetrate so much in the high frequencies as in the previous situation. As it was expected, the magnitudes of the images of type 'c.5' and 'c.6' are worst than the previous cases, and the minimum error is clearly oriented in the horizontal (Figure 58 c.2) and vertical (Figure 58 d.2) directions, depending on the absent samples. The same conclusions are applicable to the phases of Figure 59, although with a lower similitude degree.

5.2.6.1 Connotations in the real sampling process

The first simulations carried out using the iterative super-resolution algorithm previously described did not exhibit the problem of quality loose with the iteration number increase, because always were used image sets where all the original samples were presented. This problem arises when we started to use random shifts, in an attempt to model the real acquisition imaging system and thanks we are using an experimental setup where all the parameters are perfectly under control. Now the question is if that problem could be possible in a real system or if it is merely an experimental artefact.

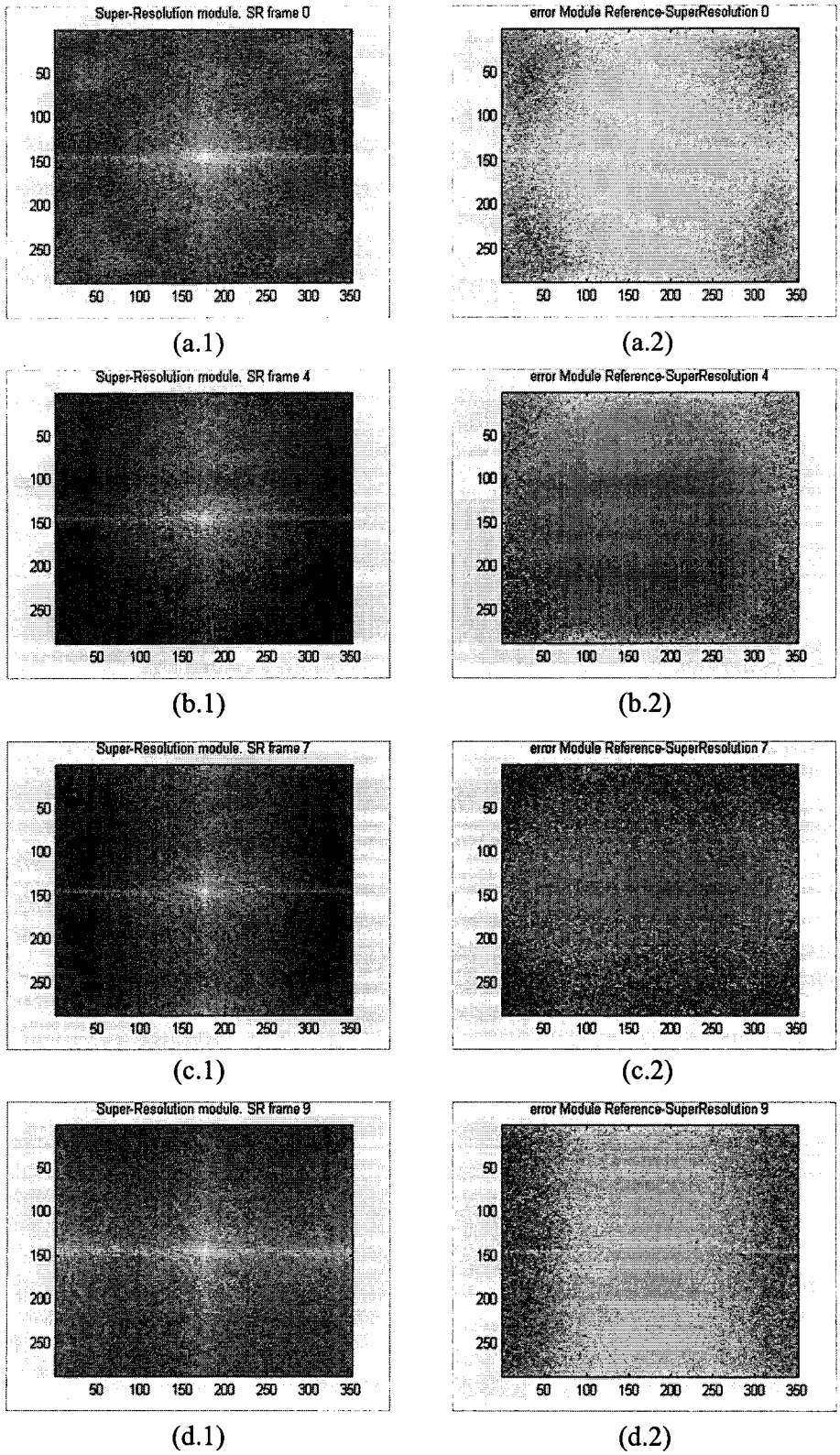


Figure 58. Magnitudes of the bi-dimensional Fourier transform (1) and the associated errors (2) for the frames 0 (a), 4 (b), 7 (c) and 9 (d) after 80 iterations per frame.

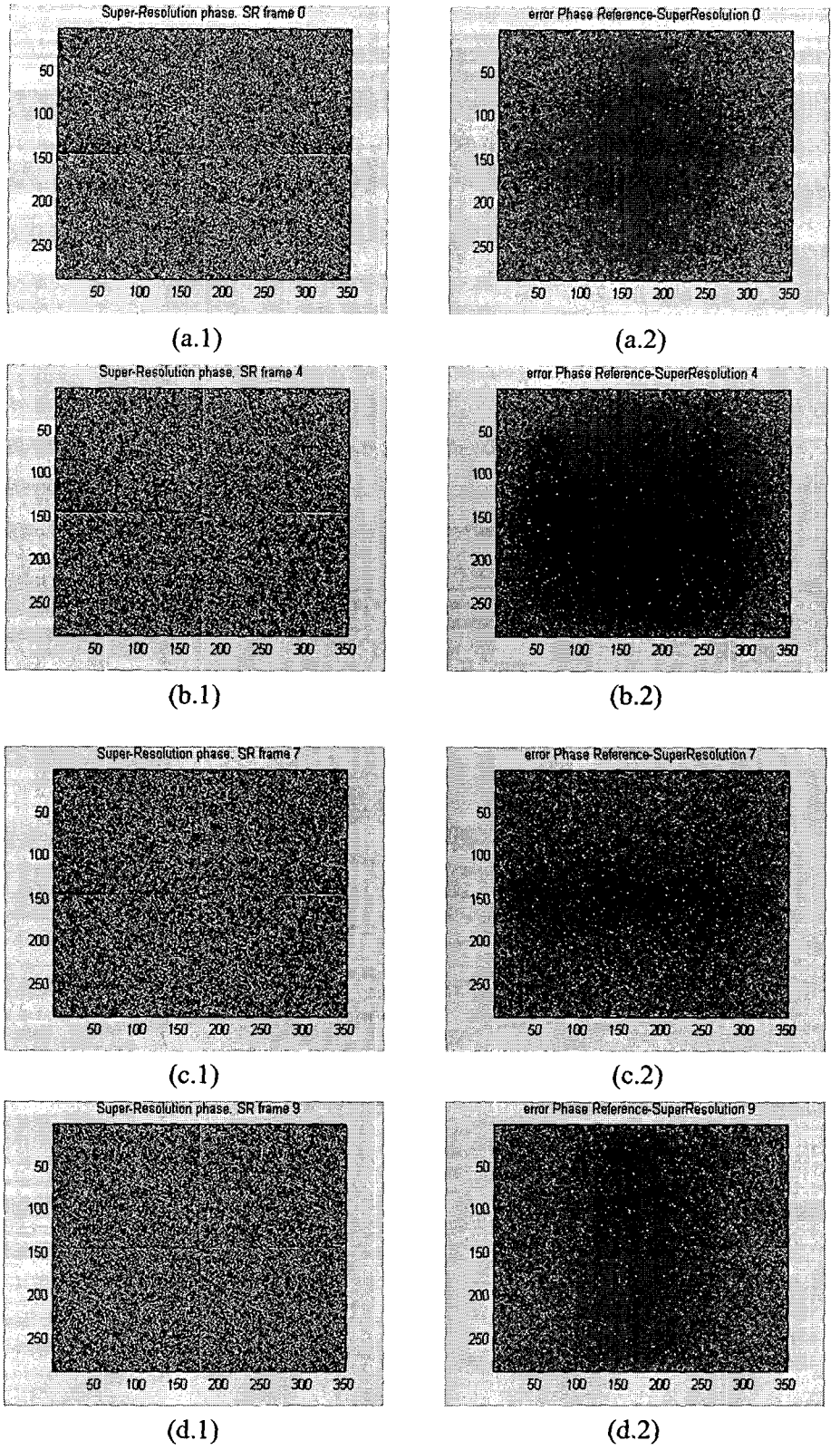


Figure 59. Phases of the bi-dimensional Fourier transform (1) and the associated errors (2) for the frames 0 (a), 4 (b), 7 (c) and 9 (d) after 80 iterations.

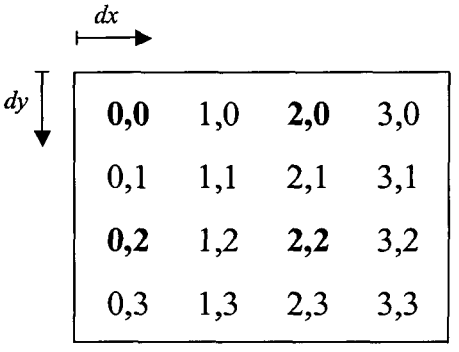


Figure 60. Base-cell using a motion estimator of ¼ pixel accuracy in low-resolution (unities in ½ pixel).

Reflecting on the sampling system, we must take into account that it is impossible to assure that four images, consecutively sampled in the time, will contain all the necessary information to increase significantly the quality of the output image. It is also impossible to assure that the sampling process will not deliver the same data (slow movement or high frame rate) either that the sampled image will not have redundant information (equivalent shifts). If that were the case, we have seen the impact in the final image quality, but statistically, it is of low probability that, among a random set of random shifts, repetitively images that will not supply new information to increase the resolution of the final image were obtained, provided that some confidence about the random distribution of the sensor movement is guarantee. If the acquisition system were limited by some subjection structure over physic guides that will not allow all the movements (tripods, subjection trolleys, etc.) the freedom degrees would be limited with the consequent repercussion in the final quality. Nevertheless, only the improvements of the image due to camera movements will be limited, but not the improvements due to movements inside the scene.

Another way to minimize the influence of this over-sampled effect is to increase the number of frames to be combined or increasing the accuracy of the motion estimator, in order to increase the number of pixels in the base cell. The drawback of increasing the number of frames is that it will also increase the memory requirements. In the other hand, the base-cell studied until now is based in the use of a motion estimator of half pixel in low-resolution, what will represent one pixel accuracy in high-resolution. If we increase the resolution of the motion estimator to quarter pixel in low-resolution then, in high resolution, we will have a discrimination capability of half pixel, what will modify the base-cell as shown in Figure 60.

In this case, the positions are in unities of half pixel. For instance, the vector (2,2) means a shift of 2 half-pixels (i.e. 1 pixel) in both directions. We have highlighted in boldface

the pixel positions, as a sampling of those positions will suppose an important quality increase of the final super-resolution image. That is because in fact the sub-pixel positions contribute with information that is share by the surrounding pixels in pixel positions. With this new base-cell the probability of equivalent sampling (same canonical vectors) is reduced from 25% to 6.25% in four consecutive frames.

Nevertheless, when assessing a real super-resolution system, which does not have the possibility of measure the real shift vectors, it must be taken into account a certain reduction factor in the expected quality due to possible equivalent sampling.

5.2.6.2 Super-resolution improvements in the chrominance

Until now, we have only coped with luminance improvements in the super-resolution images. In fact, very few super-resolution systems take into account the chrominance values. Since the low sensibility of the human eye to the chrominance and the computation increase that their inclusion will imply, typically the super-resolution systems improve the quality of the luminance but use a simple interpolated version of the chrominances. We have opted for including the chrominance in the super-resolution process, taking into account that, due to the sample process YCbCr 4:2:0, the two chrominances have half the size of the luminance in every direction, as every chrominance pixel influences on four luminance pixels. This kind of format is widely used in the image compression systems [BK96] because the format itself saves 50% of space in the memory storing. Although the chrominances have been included in the super-resolution process, the motion estimation still works with the luminance and so the motion vectors will be better fitted to the luminance than to the chrominances.

In Figure 61 the PSNR of the chrominances together with the luminance for the super-resolution sequence shown in Figure 42, Figure 43 and Figure 44 is shown. As it is clearly seen, the chrominances have higher PSNR than the luminance because they have much lower entropy, what produces lower average errors between the reference chrominances and the super-resolution references, increasing considerably the PSNR. This lower entropy of the chrominances is shown in Figure 63.

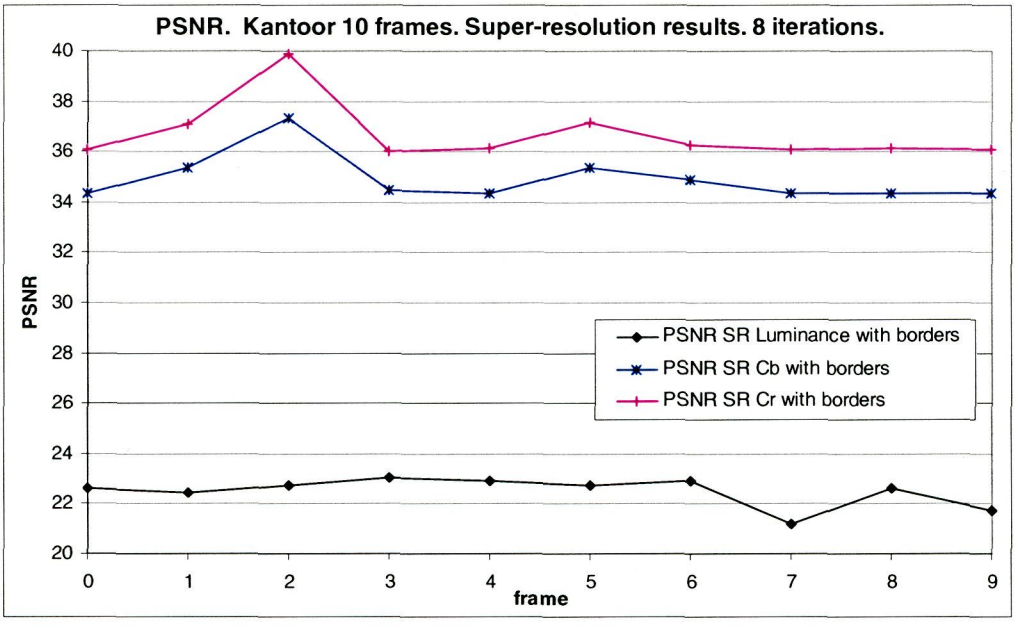
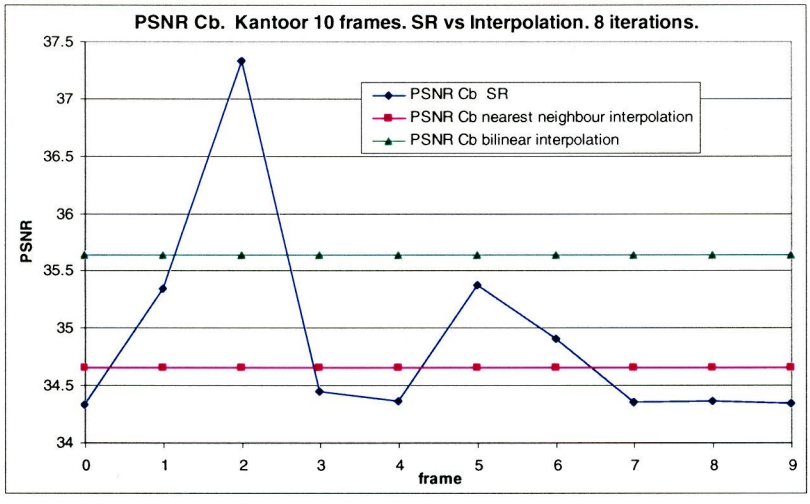
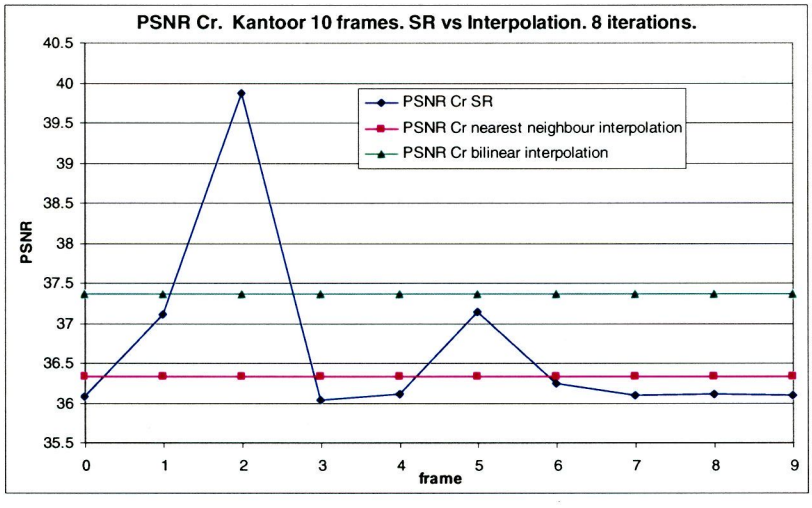


Figure 61. PSNR of the chrominances blue and red (Cb and Cr) of the super-resolution sequence compared with the PSNR of the luminance (see Figure 45).

Nevertheless, although the PSNR of the chrominances is much higher than the luminance one, it is outstanding that it is usually below the interpolation levels, as it is shown in Figure 62. That it is because we are applying image compression principles in image processing applications. When the motion estimation is performed on a video compressor, only the luminance motion vectors are calculated, and at the time of performing the motion compensation, the same motion vectors (conveniently scaled by a factor of two) are applied both to the luminance and the chrominances, in an attempt of saving computing time and under the assumption that the three images are very similar. Even in the case that the vectors of the chrominances largely differ from the luminance ones, this will have no impact in the final quality, because together with the motion vector are sent the image errors. Indeed it will negatively affect to the compression ratio, because with better vectors it could be possible to obtain images with lower entropy that will be more efficiently compressed by the discrete cosine transform (DCT) and the quantizer. But, since the chrominance images have low entropy in themselves, this effect is hardly manifested. Nevertheless, a glance to the chrominance and luminance images separately, as shown in Figure 63, manifest that, at effects of restoration and taking into account how sensible the SRA are to the motion vectors, the application of a set of motion vectors that are not specific for those images, barely improve the quality over the interpolation levels. The only solution will consist in performing one additional motion-estimation for each of the two chrominance images, what will largely increase the time required for computation, jeopardizing seriously it execution in real time.



(a)

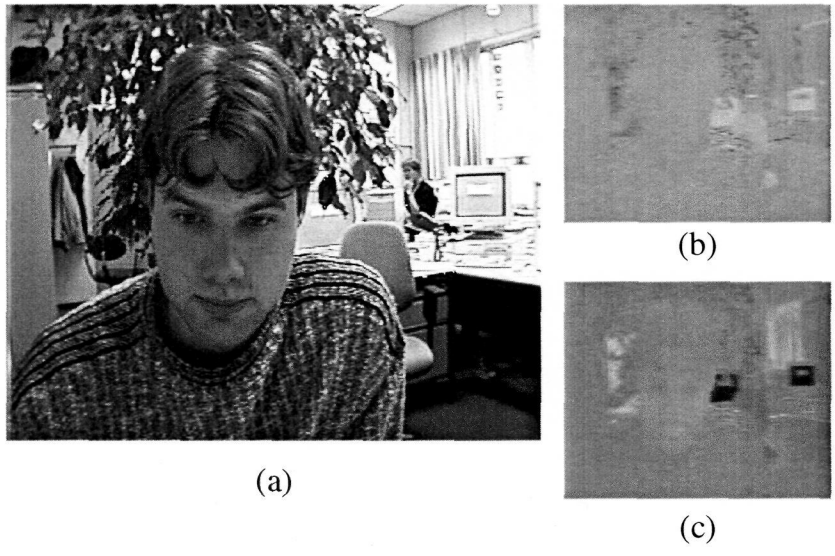


(b)

Figure 62. PSNR of the chrominances blue (a) and red (b) compared with the PSNR of the interpolated chrominances.

At the spectral level, the conclusions are very similar: the spectral correlation coefficients of the chrominances are higher than the coefficient of the luminance, and the correlation coefficients in phase seldom overcome the interpolation levels, although the correlation coefficient in magnitude is often above the interpolation levels, as it can be seen in Figure 64.

From these results, we can conclude that performing two additional motion estimations over the chrominances was not possible, as it would jeopardize the restrictions to work in real time. It is not worthwhile to apply the super-resolution algorithm over the chrominances, as a similar result can be obtained from the interpolation of the chrominances.



(a) (b) (c)

Figure 63. Reference image in format YCbCr 4:2:0 and CIF size, decomposed in luminance (a), blue chrominance (b) and red chrominance (c).

5.2.7 Iterative algorithm with reference to the first image. Version v1.3

The solution adopted in version v1.2 of performing the different quality metrics measures has the drawback that the vector set used for the super-resolution image generation must has zero mean. If the vector generation process is really random, this condition is impossible to achieve, especially if the number of frames to be combined is not a fixed number. This is why a new version of the algorithm has been developed, where the first interpolated frame is used as the first proposal of the super-resolution image, instead of the average image. In this way it is assured that the pixels of the final image will occupy the same positions as the reference pixels, enabling the direct computation of the peak signal to noise ratio, provided that the first image generated by sub-sampling uses the displacement vector (0,0). From the point of view of static images reconstruction this fact does not imposes any limitation to the process, but rather in the opposite, it supposes an improve in the model of static images acquisition. When we take a picture, usually the first image serves as the reference one in the improvement process. This is the meaning of the (0,0) vector: that the first taken image will be used as the reference and the next images will be used to increase the quality of the first image, aligning them to the reference. From the point of view of video reconstruction, it only supposes to make easier the measurement process. If the first vector

was not forced to be (0,0), then it would be necessary to shift the resulting image prior to compute the PSNR, either to the first image or to the average image if the previous strategy was followed. The new version (and the last iterative one) has been labelled as v1.3 and is shown in Figure 67.

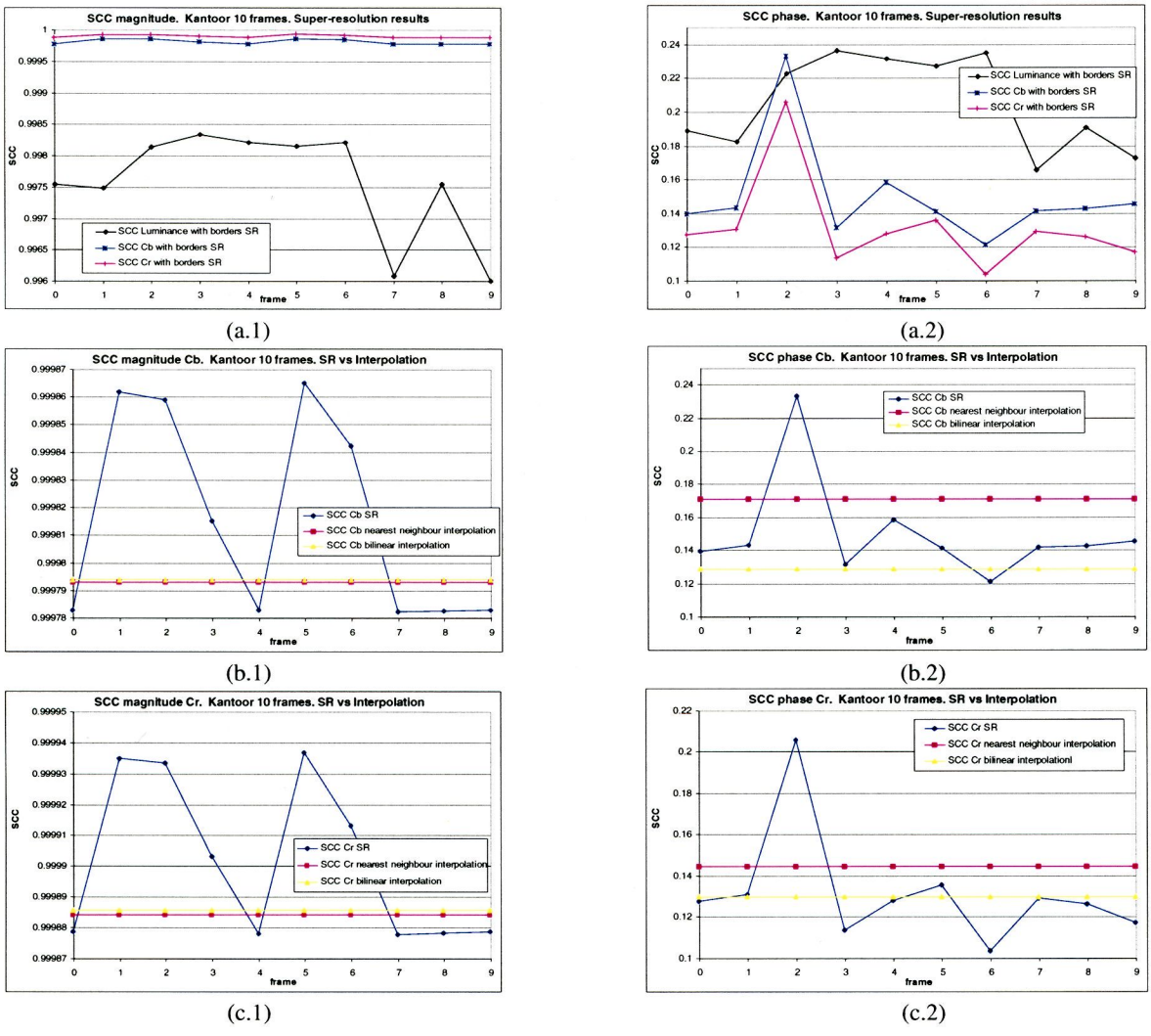


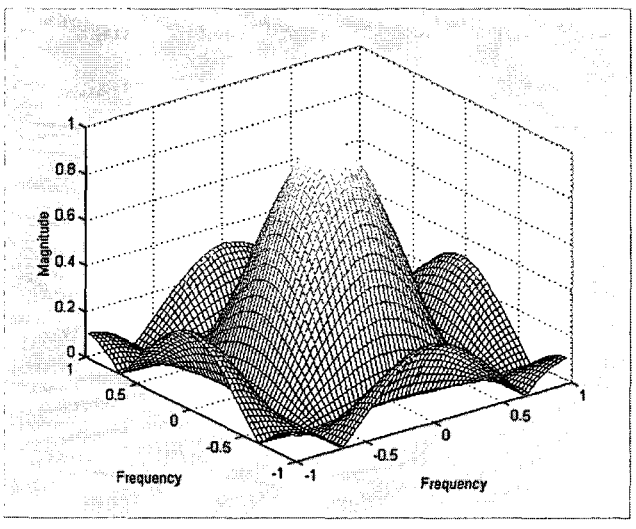
Figure 64. SCC in magnitude (1) and phase (2) of the chrominances. In (a) is compared the chrominances with the luminance, in (b) is shown the blue chrominance versus its interpolation levels and in (c) is shown the red chrominance versus its interpolation levels. Notice that the scale of the ordinate axis for the magnitude has been widely enlarged. All these figures are for a number of iterations equal to 8.

This version adds on some new aspects with respect to the previous ones. As it has been previously mentioned, in this case the first super-resolution image is obtained through a nearest neighbour interpolation from the first low-resolution image of the input sequence. As in previous versions, this image is stored in the HR_B memory, which will contain all the

time the super-resolution image. If it is not the first iteration, LR_B will be obtained through the decimation of HR_B, in the opposite case (it was the first iteration) LR_B is the first input low-resolution image. The next step inside the iterative loop is the computation of the motion vectors. If we are dealing with the first frame, it is not necessary to compute the first motion vectors, as they all will be the zero vector. For the remaining cases, the motions vectors are calculated between the input images and the low-resolution version of the super-resolution image, i.e. LR_B.

The novelty is that prior to perform the motion estimation, the image LR_B is filtered with a low-pass filter of order three, using the normalized transfer function shown in Figure 65. This filtering, previous to the motion estimation, seeks to reduce the initial amount of aliasing, therefore improving the quality of the motion vectors. The next section will deeply address this important issue in the proposed super-resolution algorithm.

$$m = \begin{pmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{pmatrix}$$



$$\begin{aligned} O(x,y) = {} & m(0,0)\, I(x-1,y-1) + m(0,1)\, I(x,y-1) + m(0,2)\, I(x+1,y-1) + \\ & + m(1,0)\, I(x-1,y) + m(1,1)\, I(x,y) + m(1,2)\, I(x+1,y) + \\ & + m(2,0)\, I(x-1,y+1) + m(2,1)\, I(x,y+1) + m(2,2)\, I(x+1,y+1) \end{aligned}$$

Figure 65. Normalized impulse response of the low-pass filter of order three used for filtering the images. I(x,y) is the input image and O(x,y) is the output-filtered image.

5.2.7.1 Pixel filtering to increase motion vectors accuracy

The motion estimator is intended to work with images that, in principle, do not contain aliasing. The presence of aliasing can substantially alter the results of the expected motion

vectors. When performing the motion estimation, we have two choices: performing it in low-resolution or in high-resolution. Doing it in high resolution has the advantage of using images with have been largely aliasing removed, but it has the drawback of requiring much more computation. Performing the motion-estimation in low-resolution requires less computation but have the drawback of using sequences with aliasing.

As our super-resolution algorithm is intended to work in real time, we first start the motion estimation in low-resolution, thus reducing computing and memory requirements. In order to cope with the input aliasing, more accurate motion vectors can be obtained removing aliasing previous the motion estimation. A large aliasing removing is achieved by simply filtering the input images before performing the motion estimation.

To evaluate the influence of aliasing on the motion-vectors accuracy we have performed the simple experiment of filtering the low-resolution images prior the motion estimation and compare the obtained motion vectors with those obtained from the motion estimation of the non-filtered low-resolution images. It has been used two copies of the first frame of the test sequence 'KRANT' of size 144×112 pixels, with global sub-pixel shifts artificially introduced to better control the results of the motion estimator.

The second frame has been shifted using a motion vector of value (-1,2), in units of quarter pixels. As all the objects of the scene are in the same depth plane, the correct result would consist in that same vector for all the macro-blocks of the frame. For the unfiltered images, we can obtain the average motion vectors absolute error, computed as the average of the absolute difference between the real and the computed vectors in both image directions. If the image is of size M×N pixels, then we will have one motion vector for every macro-block of size 16×16 pixels, i.e. MBX×MBY macro-blocks, where MBX and MBY are obtained dividing M and N by sixteen. Equation (25) shows the absolute error in the horizontal direction and (26) shows the absolute error in the vertical direction.

$$error.x = \sum_{i=0}^{MBX-1} \sum_{j=0}^{MBY-1} \left| mv.x_{real}(i,j) - mv.x_{computed}(i,j) \right| \qquad (25)$$

Iterative super-resolution algorithm v1.3

```
Set the value of the improvement: scale
nr_frames = scale*scale, M'= scale*M, N'= scale*N
HR_B[M'][N'], HR_S[M'][N'], HR_T[M'][N'] all of 8 bits. HR_A[M'][N'] is 9 bits
LR_B[M][N] for the motion estimation
Read a set of aliased Low-Resolution images in LR_I[#nr_frames][M][N]
Set the value of the maximum number of iterations: nr_iterations

// Starting with the First Image

        LR_B = LR_I[0]                                          |
        HR_B = Upsample(LR_I[0])                                |   [SR_INIT_B]

// Iterations
FOR it = 0 .. nr_iterations-1
            IF (it ≠ 0)   LR_B = Downsample(HR_B)               [SR_DOWNSAMPLE]
            FOR fr = 0 .. nr_frames-1
                IF (fr==0)
                    MV_fr2ref[fr] = 0
                ELSE
                    IF (it ≠ 0)   LR_B = Low_Pass_Filter(LR_B)  [SR_FILTER]
                    MV_fr2ref[fr] = Calc_Motion_Estimation (LR_I[fr], LR_B)
                END IF
                Select_global_motion_vector()
                MV_ref2fr[fr] = -MV_fr2ref[fr]                  INVERT_MV()
                MV_fr2ref[fr] = 2 .* MV_fr2ref[fr]
                MV_ref2fr[fr] = 2 .* MV_ref2fr[fr]
            END FOR
            HR_A = 0                                            [SR_INIT_A]
            FOR fr = 0 .. nr_frames-1
                HR_S = Motion_Compensation (HR_B, MV_ref2fr[fr])    [SR_MOT_COMP1]
                HR_S = Upsample(LR_I[fr])/2 – HR_S/2                [SR_UPSAMPLE]
                HR_T = Motion_Compensation (HR_S, MV_fr2ref[fr])    [SR_MOT_COMP2]
                HR_A = HR_A + HR_T/2                                [SR_ADD]
            END FOR
            HR_B = HR_B + HR_A                                  [SR_UPDATE]
            variance = variance(HR_A)                           [SR_STAT]
            If (variance < 0.5)  break
END FOR
```

Figure 66. Pseudo-code of the iterative algorithm v1.3, modified to use the first low-resolution image as the reference.

$$error.y = \sum_{i=0}^{MBX-1} \sum_{j=0}^{MBY-1} \left| mv.y_{real}(i,j) - mv.y_{computed}(i,j) \right| \qquad (26)$$

The average error is given as the medium value in each direction, as indicated in (27).

$$error = \frac{error.x + error.y}{2} \tag{27}$$

In Table 11 are summarized the resulting error for this test. It is clear that errors are lower when the images are filtered prior performing the motion estimation. Taking into account that we are using 63 macro-blocks, the average motion vector error per macro-block is of 1.13 quarters pixels for the unfiltered images and 0.8015 quarters pixels for the filtered images.

MV Errors	Unfiltered images	Filtered Images
Horizontal	65	38
Vertical	78	63
Average	71.5	50.5

Table 11. Motion vectors errors with and without pixel filtering of the low-resolution input images.

It is necessary to point out that the images are filtered only for performing the motion estimation, but for the remaining part of the super-resolution process the images are left unfiltered. In Figure 67 can be seen the huge influence of an accurate estimation of the displacements between images in the final super-resolution quality. It is clear that pre-filtering images before performing the motion estimation has a direct and important impact on the final super-resolution sequence. In this case, pre-filtering images supposes an average increase of 1.33 dB on the sequence quality.

5.2.7.2 Block diagram and memory requirements for version v1.3

The data-flow block-diagram of this version of the super-resolution algorithm is shown in Figure 68. Comparing it with version v1.2, it can be appreciated that the average operation has been removed, and have appear the operation of filtering and some modifications in the data-flow and the memories.

The memory requirements are the same as in the previous version v1.2, and so the results exhibit in Table 8 and Table 9 as well as in Figure 38 are also applicable here. The memory HR_A has been highlighted by using double line to point out that it is nine bit and the operators have been shadowed. The remaining memories have been drawn using single lines.

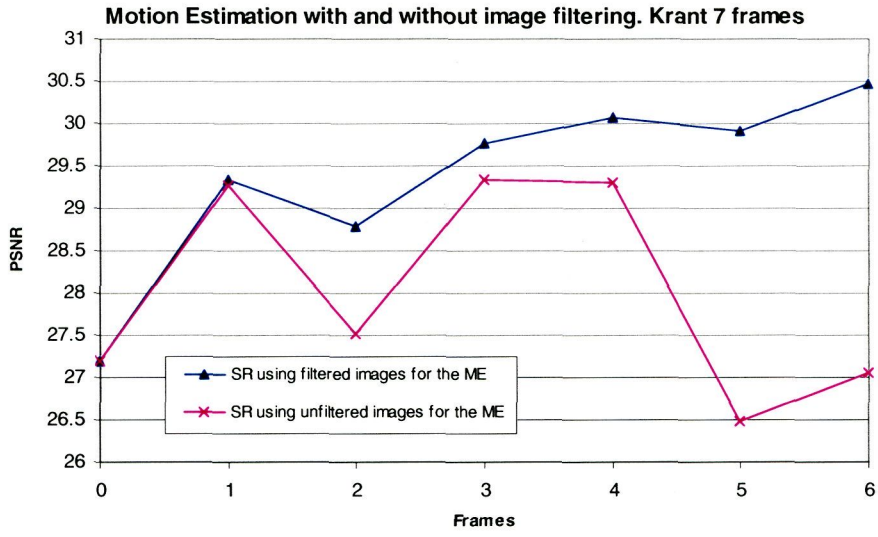


Figure 67. Influence of pre-filtering the input images before the motion estimation in the quality of the super-resolution images using 8 iterations.

In Table 13 the global shift vector applied to the first input frame in order to obtain the test video sequence are shown. The first column shows the order number of the output frame, the second and third columns show the shift vectors of high-resolution and low-resolution respectively, in units of ½ pixel. The last column shows the reduction to canonical vectors. Based on the canonical vectors, we can perform the same image classification shown in Figure 54. This classification is shown in Table 12 and supposes itself an anticipation of the quality that it will be obtained at the end of the super-resolution process. Due to the fact that the zero vector is always present, types b.4, c.2, c.5, c.6, d.2, d.3 and d.4 are not possible in this version of the algorithm.

frame	Type of output image	*frame*	Type of output image
0	c.1	5	a
1	c.1	6	a
2	a	7	c.3
3	a	8	c.1
4	a	9	c.4

Table 12. Types of output images as a function of the samples present in the base cell of 4×4 pixels.

frame	*High-resolution vectors*		*Low-resolution vectors*		*Reduction to canonical vectors*		*frame*	*High-resolution vectors*		*Low-resolution vectors*		*Reduction to canonical vectors*	
0	0	0	0	0	0	0	5	0	0	0	0	0	0
	-2	2	-1	1	1	1		2	6	1	3	1	1
	2	2	1	1	1	1		6	4	3	2	1	0
	0	4	0	2	0	0		0	6	0	3	0	1
1	0	0	0	0	0	0	6	0	0	0	0	0	0
	2	2	1	1	1	1		-2	-4	-1	-2	1	0
	0	-4	0	-2	0	0		-4	2	-2	1	0	1
	6	2	3	1	1	1		-2	2	-1	1	1	1
2	0	0	0	0	0	0	7	0	0	0	0	0	0
	0	-2	0	-1	0	1		-2	-4	-1	-2	1	0
	2	-2	1	-1	1	1		2	0	1	0	1	0
	6	4	3	2	1	0		0	-4	0	-2	0	0
3	0	0	0	0	0	0	8	0	0	0	0	0	0
	0	-2	0	-1	0	1		-4	-4	-2	-2	0	0
	-2	-4	-1	-2	1	0		-2	-2	-1	-1	1	1
	-6	-2	-3	-1	1	1		-2	-2	-1	-1	1	1
4	0	0	0	0	0	0	9	0	0	0	0	0	0
	2	2	1	1	1	1		0	4	0	2	0	0
	0	-2	0	-1	0	1		-4	2	-2	1	0	1
	-2	0	-1	0	1	0		-4	2	-2	1	0	1

Table 13. Shift vectors randomly generated in distances of two pixels for high-resolution, of one pixel for low-resolution and it reduction to canonical vectors.

5.2.7.3 Simulation results and quality and behavioural analysis of version v1.3 of the iterative algorithm

In Figure 69 is shown the PSNR obtained after the super-resolution process of the first frame of the KANTOOR sequence. This sequence has CIF (352x288) format, and has been sub-sampled and shifted the amounts reflected in Table 13. As in the previous case, also the PSNR of the interpolated images have been included in order to settle the interpolation levels.

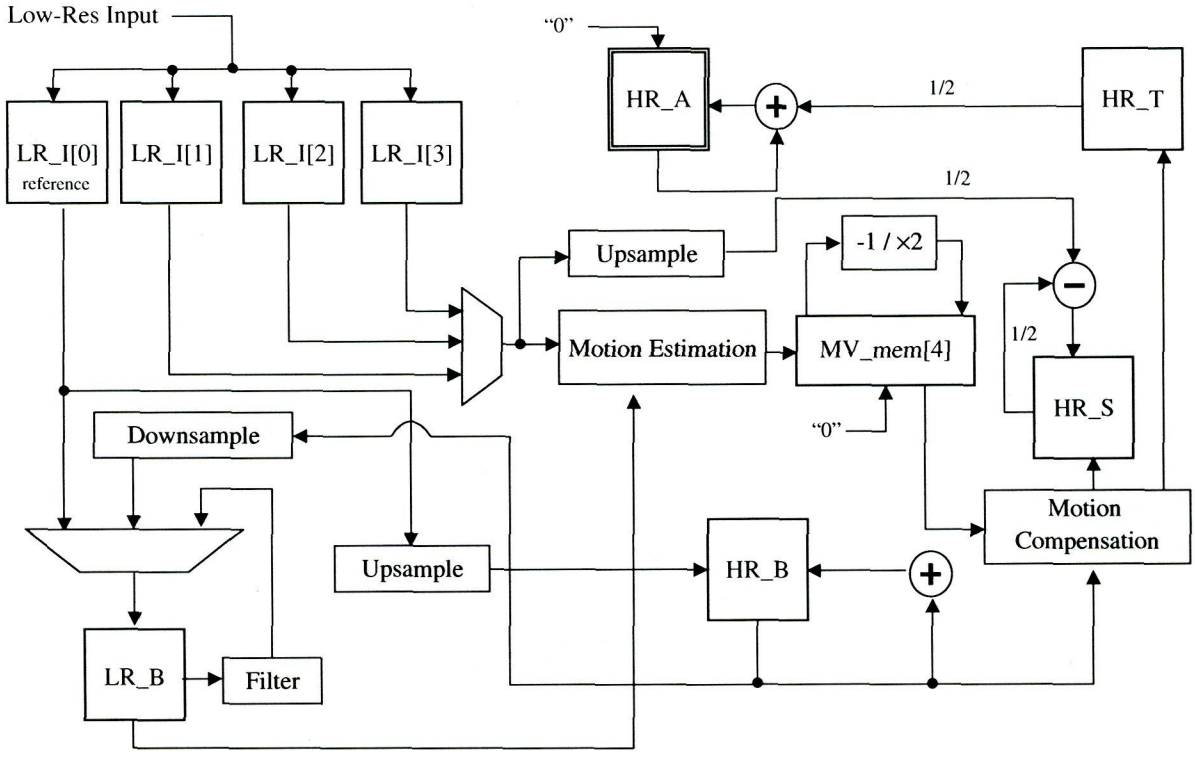


Figure 68. Data-flow block-diagram of version v1.3 of the super-resolution algorithm.

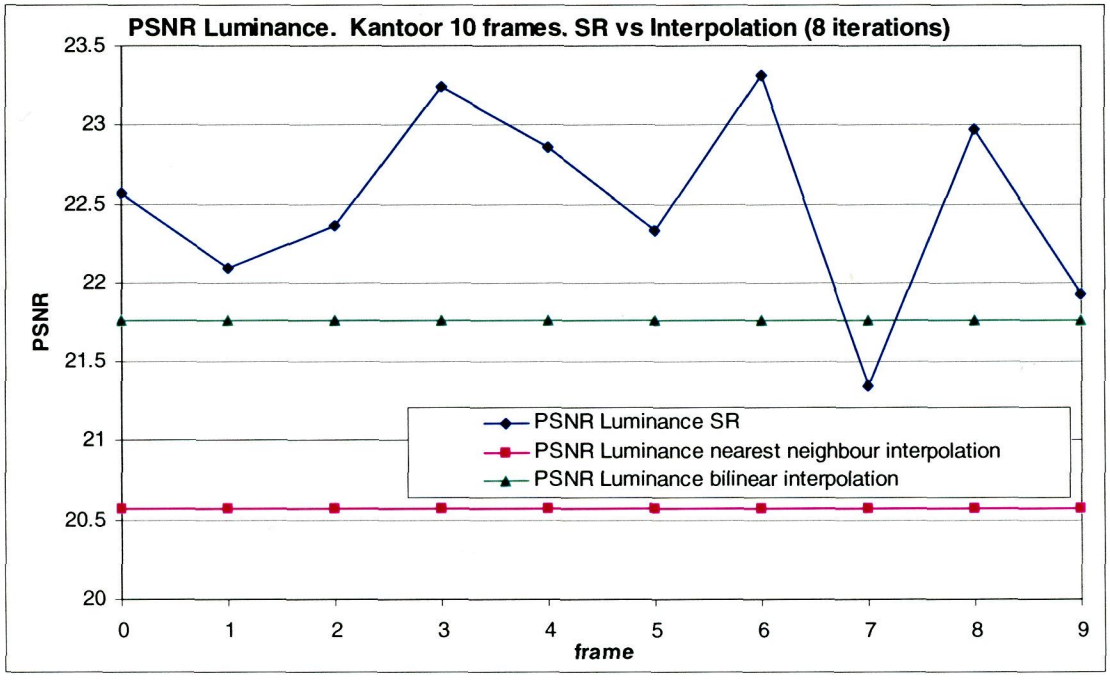


Figure 69. PSNR of the output sequence using version v1.3 of the SRA versus the interpolated images.

Similarly to the previous case, great variations in the quality of the image can be appreciated, basically due to well-known border effect. To subtract this effect from the PSNR, in Figure 70 is shown the PSNR of the same sequence or PSNR', removing a border of 16 pixels all around the images.

In the chart can be appreciated again a distribution of the PSNR that strongly depends on the type of sampling. In order to reduce the freedom degrees, making easy at the same time the analysis of the results, we have kept unalterable the seed for the generation of the random numbers which are used for the displacements. That is why the kind of output images is almost the same that the ones obtained in the previous version of the super-resolution algorithm. The differences rely on the aim of assuring zero median. In that sense, the last displacement vector of each set of four vectors has been adjusted in consequence. This adjust seeks to establish a common base (or at least more similar) to compare the algorithms, but obviously this does not affect in the behavioural of the real system.

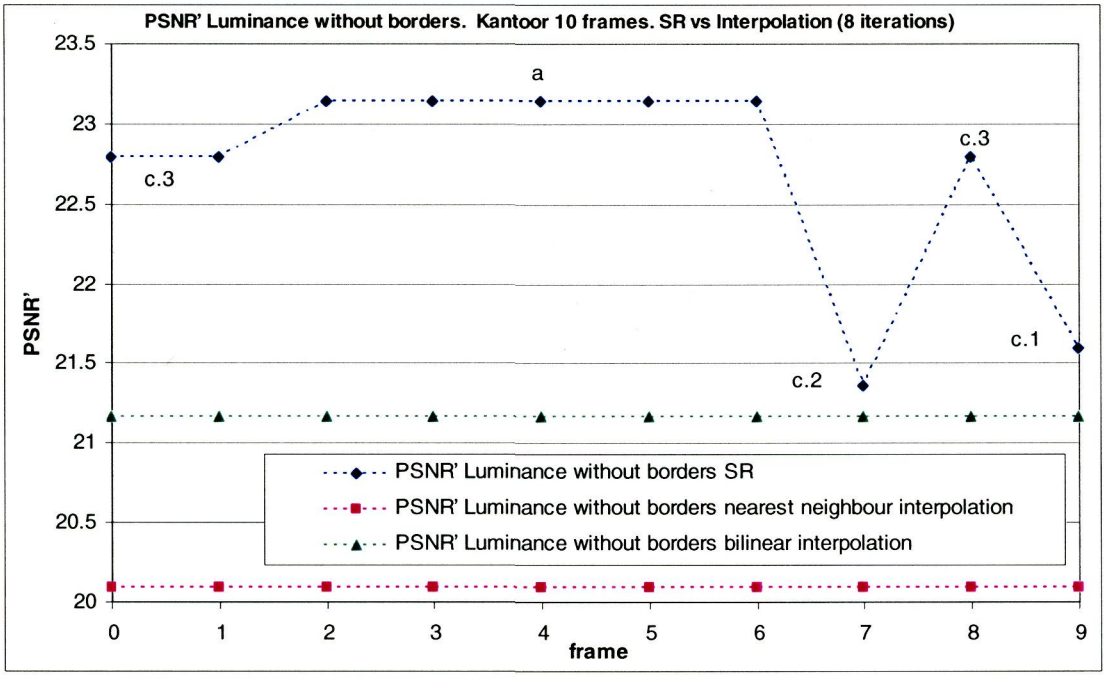
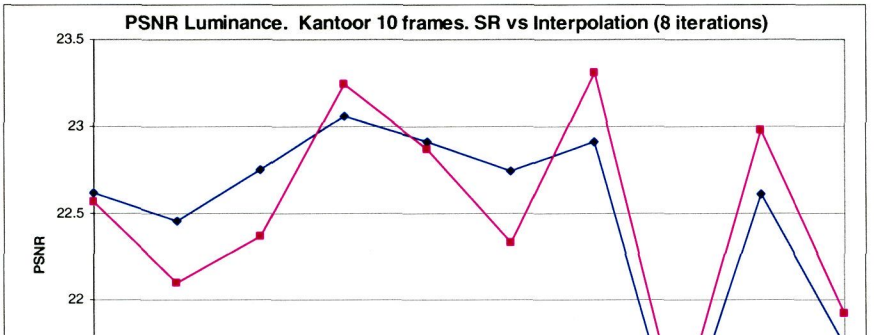


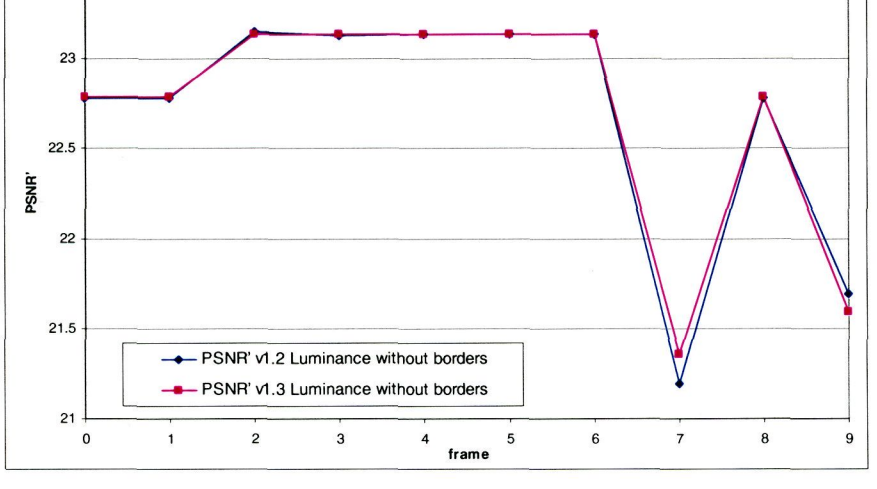
Figure 70. PSNR' of the output sequence using version v1.3 of the SRA removing a 16 pixels border all around the image.

The results obtained for this algorithm are very similar in quality and behavioural in both versions of the iterative algorithm. In Figure 71 is shown a comparison of the PSNR between version v1.2 and version v1.3 using the full image (a) and without borders (b), where can be verified that in the last case the obtained qualities are almost the same. However, when

the full image is taken into consideration, it seems that the version that uses the frame zero as a reference punctually obtains better results (frames 3, 6 and 8) although in the rest of the cases the qualities are slightly lower. In any case, this chart provides indications that the improvements obtained through variations in the super-resolution algorithm seems to reach a limit. In the next section, transformations of the algorithm toward non-iterative schemes and new problems that arise in when mapping the new algorithms in the Picasso architecture are analyzed.



(a)



(b)

Figure 71. PSNR comparison between versions v1.2 and v1.3 of the SRA, using the full image (a) and removing the borders (b).

5.2.8 Motion estimation search strategy

The quality of the images obtained by the super-resolution algorithms strongly depends on the precision of the motion vectors. The motion estimation is the most expensive function in terms of computing and time processing. With the aim of keeping the application under conditions of low-cost and real-time restrictions, we have used the 'motion-estimator' provided by the Picasso architecture. In chapter 3 a brief description of the Picasso architecture was developed and the two types of search strategies implemented were described: the full-search and the three dimensional recursive search (3DRS). As the first strategy has the advantage of search completeness and the drawback of high computational cost, the second strategy has the advantage of rapidness and the drawback of evaluate only some specific vectors, although the results appears to be accurate enough [LKL+01].

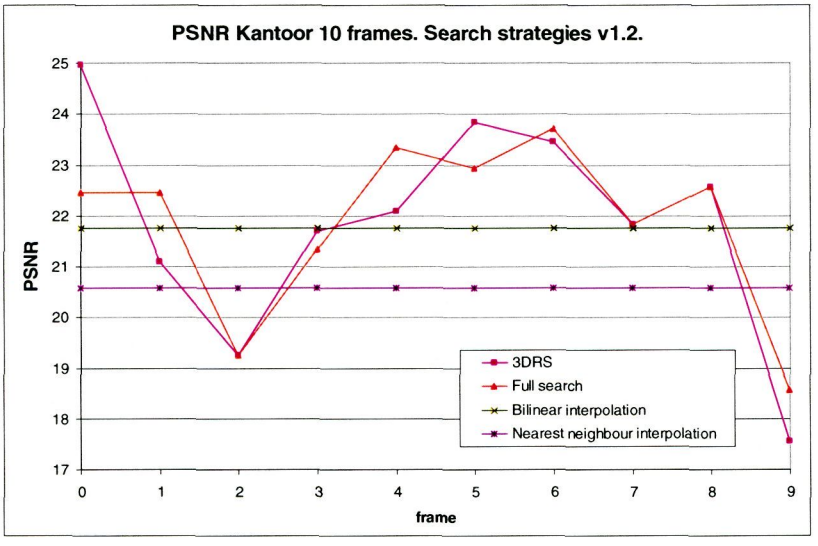
The search strategy selected seriously compromises the quality of the resulting images. In Figure 72 is shown the PSNR of the output sequence KANTOOR, using version v1.2 (a) and version v1.3 (b) for both motion vectors search strategies.

A very interesting aspect to point out in this results is that clearly version v1.3 exhibit better results than version v1.2, being all the PSNR figures above the interpolation level (except for frame number 2). That is because of the low-pass filter prior performing the motion estimation. In fact, removing the image aliasing before the motion estimation supposes an important improvement in the quality of the motion vectors and this improvement directly bounce back on quality of the final image. This difference is more patent when removing the image borders, as shown in Figure 73. In this case, all the output frames of version v1.3 present qualities above the interpolation level.

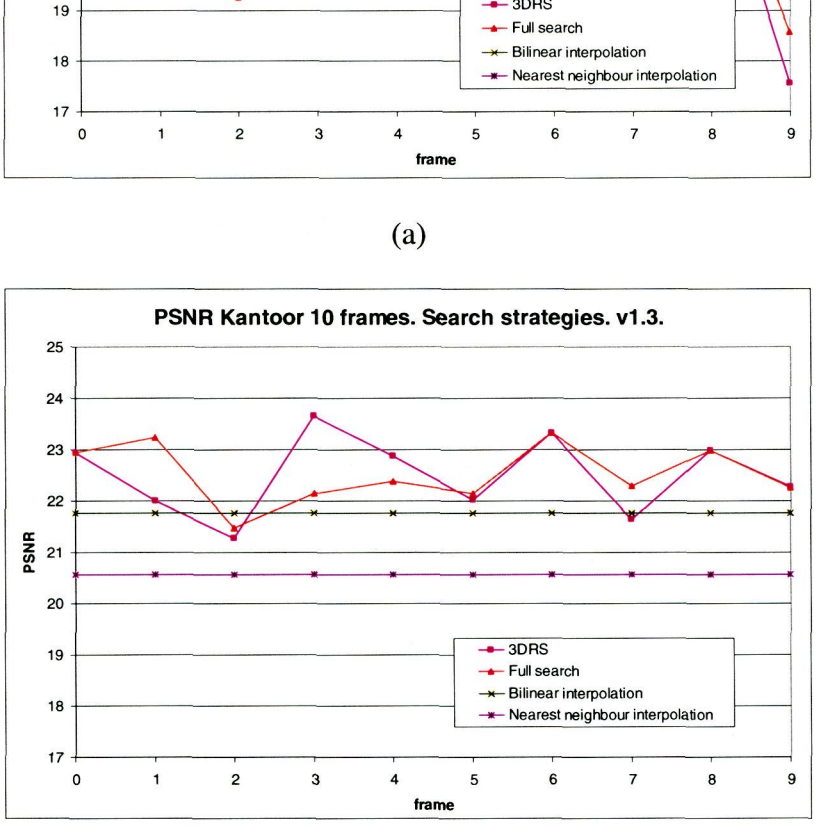
Moreover, version v1.3 exhibits a higher stability on the image quality, what will suppose a better view of the final sequence. A video sequence with abrupt jumps of the quality is more annoying to watch than another sequence of lower quality but more stable.

Even when the analysis of the search strategy has driven us to clearly realise about the superiority of the version v1.3 over version v1.2, we can still not conclude anything about the better search strategy for the motion estimation, as it is detached from the observation of the qualities obtained in Figure 73, where the border effect has already been removed. It can be verified that the full-search, that 'a priori' is the best candidate to obtain the motion vectors, does not prevail over the other strategies. That is because in fact, the search strategy does not reflect the real movement of the image, but it rather tries to minimize a block-level cost

function, in our case the average absolute error, and so, the quality of the image will vary depending on the similitude between the real movement and the vector which minimizes the cost, i.e. the motion vector.

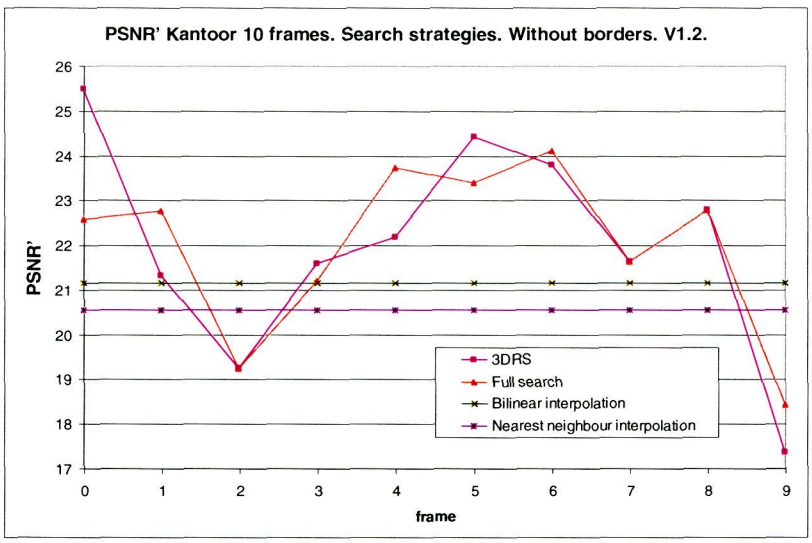


(a)

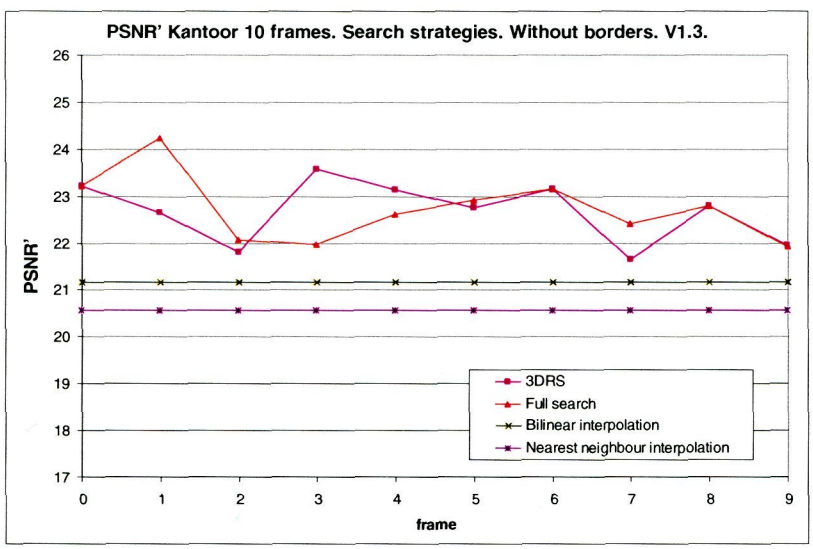


(b)

Figure 72. PSNR of versions v1.2 (a) and v1.3 (b) of the SRA using different strategies for searching the motion vectors using 8 iterations.



(a)



(b)

Figure 73. PSNR' of versions v1.2 (a) and v1.3 (b) of the SRA using different strategies for searching the motion vectors with the image borders removed and using 8 iterations.

5.3 Non-iterative super resolution

Although the iterative versions offer very good image quality mapped onto a hybrid video encoder, it is patent the necessity of create a new type of algorithm that, using the same resources, can operate in a single step, i.e. a non-iterative algorithm. The main idea is based in the following considerations:

- Every new image adds new information that must be combined in a new high-resolution grid in the proper way.
- It is impossible to know 'a priori' (for the super-resolution algorithm scope) the position of the new data and if they are going or not to contribute with new information
- In the case of having new information from several images it will be desirable to have a quantitative knowledge of how much every new pixel will contribute to the resolution of the super-resolved image. As a new pixel could fit in the super-resolved image in a half-pixel position, its contribution to the final image will not be the same that those of another pixel that fits in a full-pixel position. The former pixel will have half the contribution of the last.

5.3.1 Algorithm description

Based on the previous considerations, the following algorithm has been developed:

1. Initially, the first low-resolution image is taken and its pixels are placed in a high-resolution grid, leaving the uncovered pixels to a zero value. From now on, this process will be denominated 'up-sample holes'. As the size increase is a factor of two in both directions, in Figure 74 can be seen the location and relationship among the pixels of high and low resolution.
2. Latter on, the contributions of the pixels are generated. The contributions are the weights assigned to each pixel to denote the amount of information provided to that pixel position. As we are initially combining four low-resolution images in a grid 2-by-2 times bigger, an initial contribution of 4, for half pixel precision in low-resolution will be enough. If the resolution of the motion estimator was increased or the motion-estimation was performed in high-resolution, higher values would be necessary. These contributions are expressed over the high-resolution grid. Thereby, the contributions of

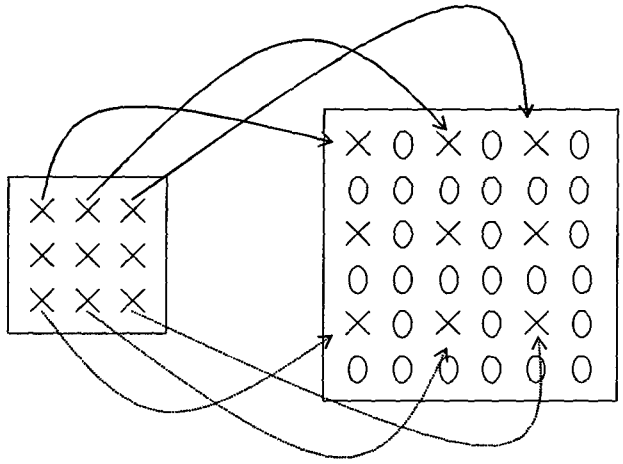


Figure 74. Mapping of the low-resolution pixels in the high-resolution grid, leaving holes in the missing pixels.

the image in Figure 74 are shown in Figure 75, pointing out that the pixels supplied have maximum contribution and the rest have zero value.

4	0	4	0	4	0
0	0	0	0	0	0
4	0	4	0	4	0
0	0	0	0	0	0
4	0	4	0	4	0
0	0	0	0	0	0

Figure 75. Contributions of the initial image.

3. Now, the relative displacements between the next input image and the first image, that will be the reference, are estimated. These displacements are stored in memory, as they will be lately used.
4. Steps 1 and 2 are applied to the new input image, i.e. it is adapted to the high-resolution grid, leaving the missing pixels to zero and generating the initial contributions.
5. In this step, both the new image over the high-resolution grid and its associated contributions are motion compensated toward the reference image. In the compensated contributions will be reflected the real contributions of the new pixels to the high-resolution reference image.
6. Now is performed the lineal summation of the initial image and contributions with the

compensated image and contributions. This summation assures further noise reduction in the resulting image.

7. Steps 3 to 6 are applied to the next incoming images.
8. Once finished step 7, we will have a high-resolution image with the summation of all the compensated pixels and a memory with the summation of all the compensated contributions. Then the high-resolution image is adjusted depending on the contributions, as it is indicated in equation (28).

$$SR(i,j) = 4 \cdot \frac{\sum_{fr=1}^{4} Motion\_Compesate(Upsample\_Holes(LR\_I[fr]))}{\sum_{fr=1}^{4} Motion\_Compesate(contributions[fr])} \tag{28}$$

9. After the adjustment, it is possible that some pixel positions remain empty, i.e., from the input image set those certain positions do not have new information. This case will be denoted with a zero, both in the high-resolution image position and in the contribution and the only solution is to interpolate the zeroes with the surrounding information. However, we cannot conclude that a zero in the image implies that the value must be interpolated, because a zero is a possible and valid value in an image. A pixel will be interpolated only if its final contribution is zero.

In Figure 76 is shown this algorithm in pseudo-code, using the memories and the resources of the Picasso video hybrid encoder.

Note that, with respect to the iterative versions of the previous section, not only the iterative feature has been removed, but also one of the motion estimations has been removed. As the motion estimation is performed in low-resolution and the motion compensation in high-resolution, it is necessary to multiply by two all the motion vectors to properly scale them. Initially, the number of frames has been set to four, following the philosophy of the iterative algorithms.

For a better understanding of the algorithm, we will show how it works step by step, using the macro-block (0,2) of frame 0 in luminance of the test sequence KRANT. Moreover, with the aim of cover all the possibilities, a set of motion vectors that leaves holes in the super-resolution image has been selected, although it is necessary to point out that it is not the usual case. Figure 77 shows initialization stage of the super-resolution image and the

contributions memory. In the left side, left for block diagrams, memories are shown as boxes with double-line borders to easily discern them from the operators, shown in single-line. The first low-resolution image LR_I[0] will be the movement reference and will be store in the memory HR_A. At the same time, the contributions will be stored in HR_C.

```
nr_frames = scale*scale, M'= scale*M, N'= scale*N
HR_B, HR_S, HR_T, HR_T2, HR_C and HR_S2 all of 8 bits and size [M'][N'].
HR_A is 10 bits and size [M'][N'].
LR_I[M][N] for the motion estimation
Read a set of aliased Low-Resolution  images in LR_I[#nr_frames][M][N]
MV_ref2fr[0] = 0,0

FOR fr = 1 .. nr_frames-1
   MV_ref2fr[fr] = Motion_Estimation( LR_I[fr], LR_I[0])
   MV_ref2fr[fr] = 2 .* MV_ref2fr[fr]
END FOR

HR_A  = Upsample_Holes(LR_I[0])
HR_C  = Create_image_contributions()

FOR fr = 1 .. nr_frames-1
   HR_S  = Upsample_Holes(LR_I[fr])
   HR_S2 = Create_image_contributions()

   HR_T  = Motion_Compensation(HR_S, MV_ref2fr[fr])
   HR_T2= Motion_Compensation(HR_S2, MV_ref2fr[fr])
   HR_A  = HR_A + HR_T
   HR_C  = HR_C + HR_T2
END FOR

HR_A = 4*HR_A/HR_C

IF  (HR_C(i)==0)  THEN
   HR_B.lum = Interpolate(HR_A.lum)
ELSE
   HR_B.lum = HR_A.lum
END IF
HR_B.chrom = HR_A.chrom

Clip(HR_B, 0, 255)
```

Figure 76. Kernel pseudo-code of the non-iterative versions of the super-resolution algorithm.

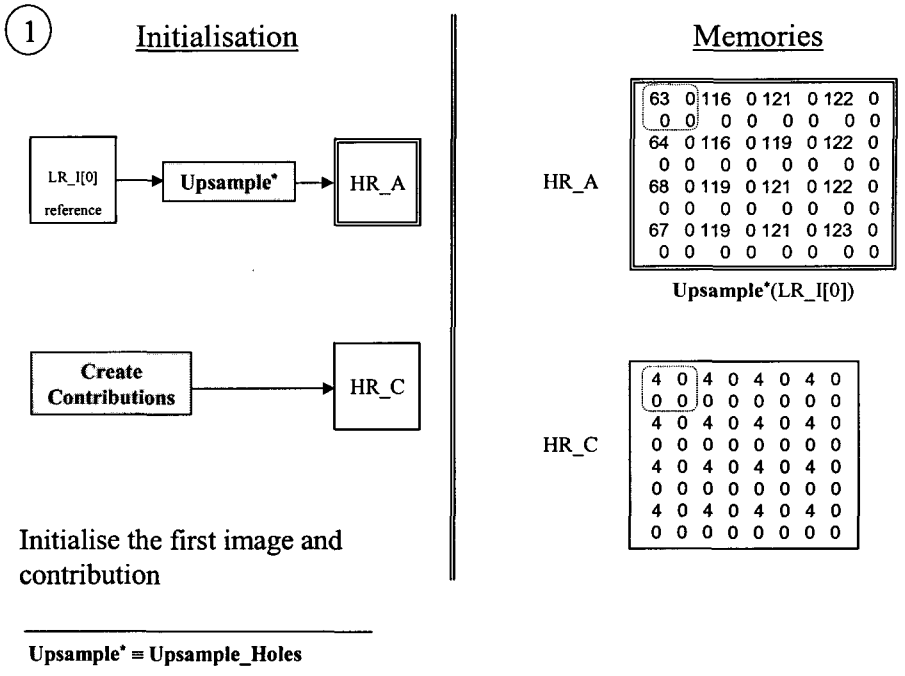


Figure 77. Initialization of the super-resolution image and the contributions.

Once the memories have been initialized, we go into the loop to treat the three remaining memories, starting with the motion estimation, as it is shown in Figure 78, where also the motion vectors used in high and low resolution are shown. These vectors are stored in the memory MV_ref2fr[0..3] for been subsequently been used.

The next step (Figure 79) is to initialize the remaining memories with the input images taking to high-resolution and their corresponding contributions. As the real interest relays on the summation of all the images, in fact the used memories (HR_S for images and HR_S2 for contributions) are reused for every new input image. In Figure 79 are also shown the values that HR_S will take for every input image, indexed as [1], [2], [3]. Notice that however the initial contributions are the same for all the four images.

Once the high-resolution images have been created, we must compensate their movement with respect to the reference image, and the same must be done with the contributions, but with a different objective: to determine the contribution of each pixel to the super-resolution image. Results are stored in HR_T and HR_T2 (Figure 80).

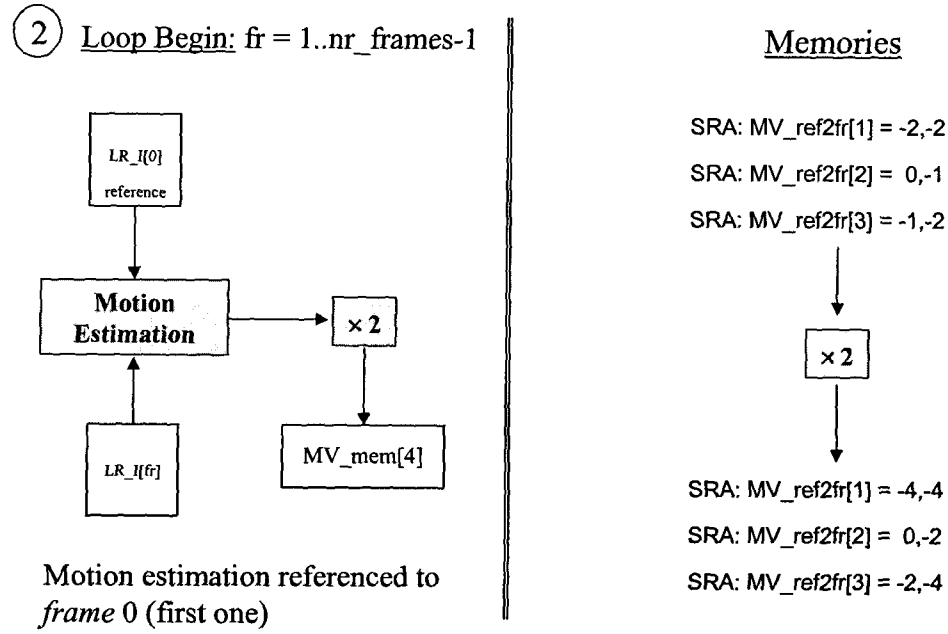


Figure 78. Motion estimation of the remaining images respect to the first one.

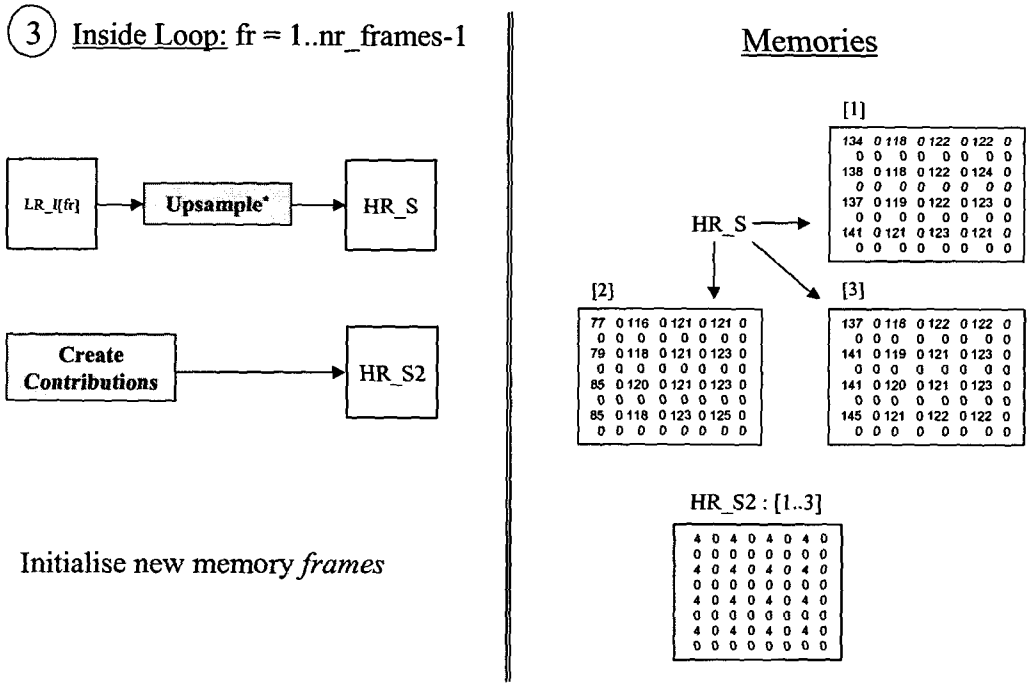


Figure 79. Initialization of the high-resolution images and their contributions inside the main processing loop.

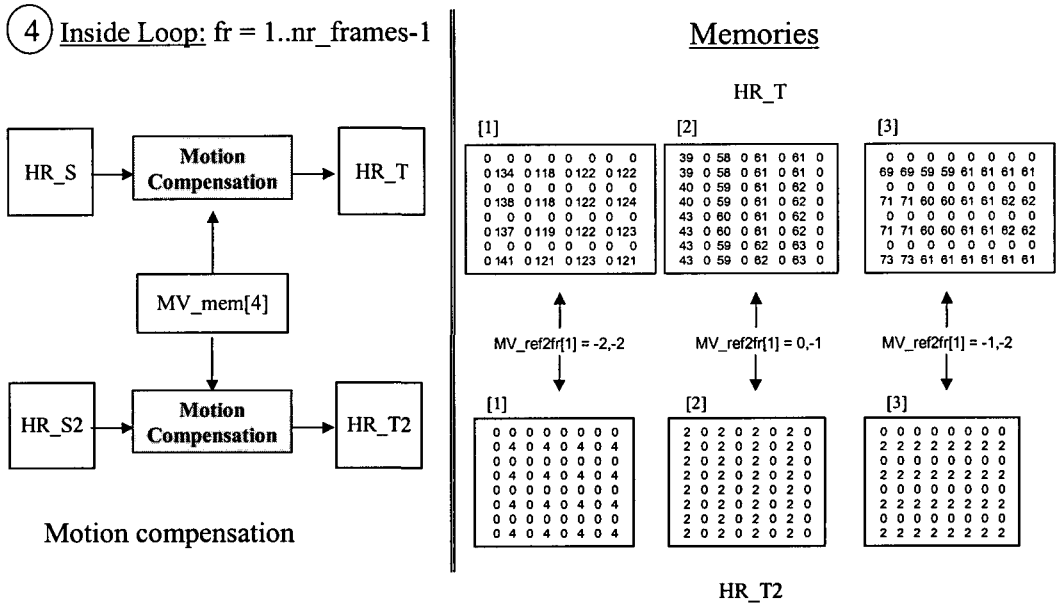


Figure 80. Motion compensation of the input images in high-resolution and their contributions inside the main processing loop.

Once the images and their contributions have been compensated, the summation of all of them is performed. From the image point of view, it supposes the superposition of the new data and the average of those that are repeated. From the point of view of the contribution, it supposes to add up the weight of each pixel over the final image. In Figure 81 is shown how the images are accumulated in HR_A and the contributions in HR_C. In the last step of the loop, in HR_A and HR_C will be accumulated the final values, where it is interesting to highlight the zero value in the right-upper pixel of the base cell. This means that the input images have not been capable of supply new information that fits in those positions.

Once the loop is over, it is necessary to adjust the accumulated values relying on the weights of every pixel. That is carried out applying equation (28) to the accumulative memory HR_A using HR_C as contributions, as depicted in Figure 82. If the summation of the contributions is zero, it can not be divided by its value, and the zero value of the image will be kept.

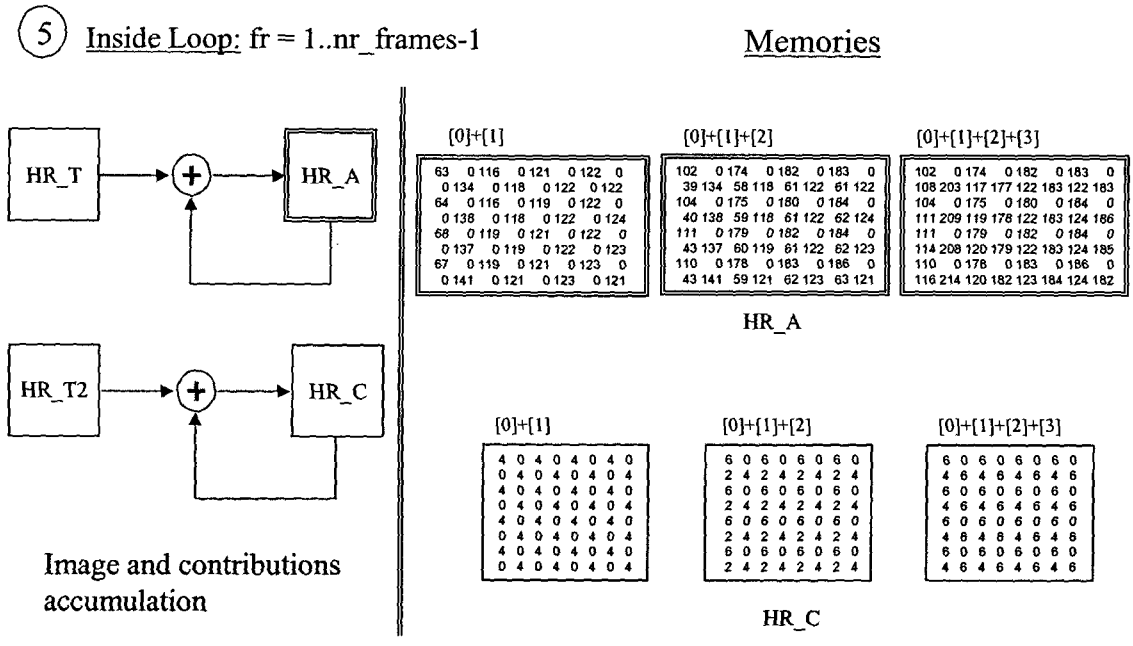


Figure 81. Summation of the input compensated images in high-resolution and their contributions inside the main processing loop.

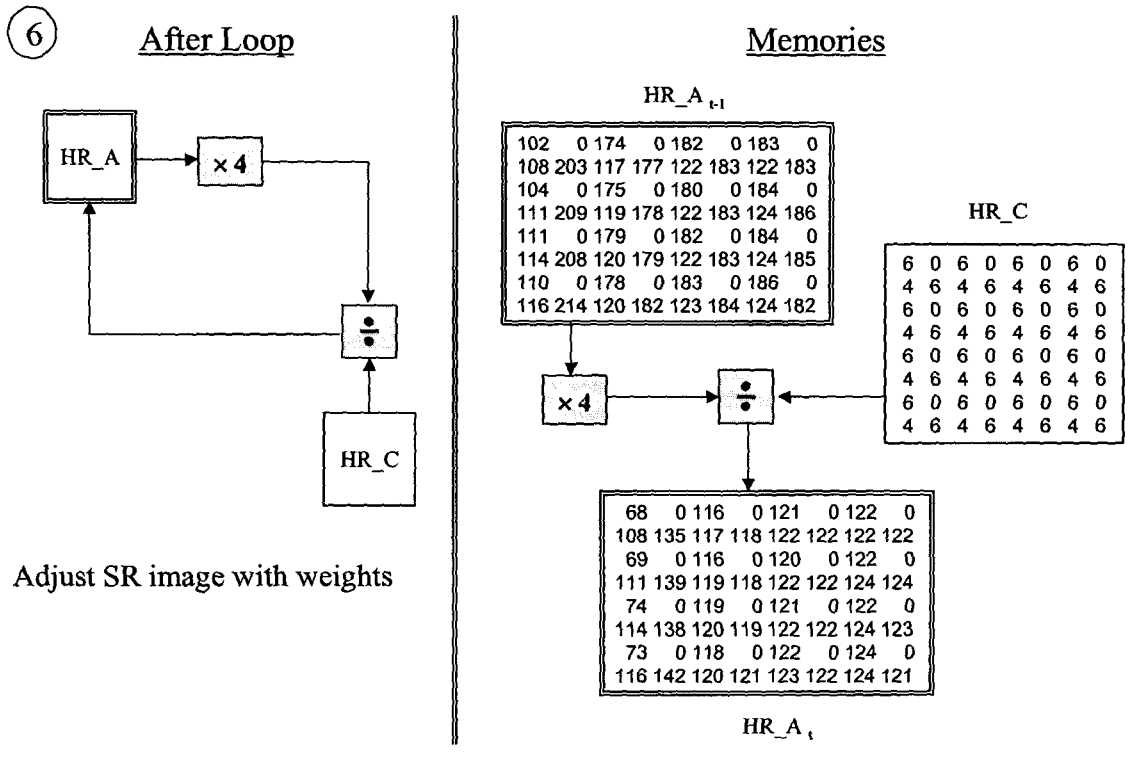


Figure 82. Adjustment of the final accumulative image as a function of its contributions.

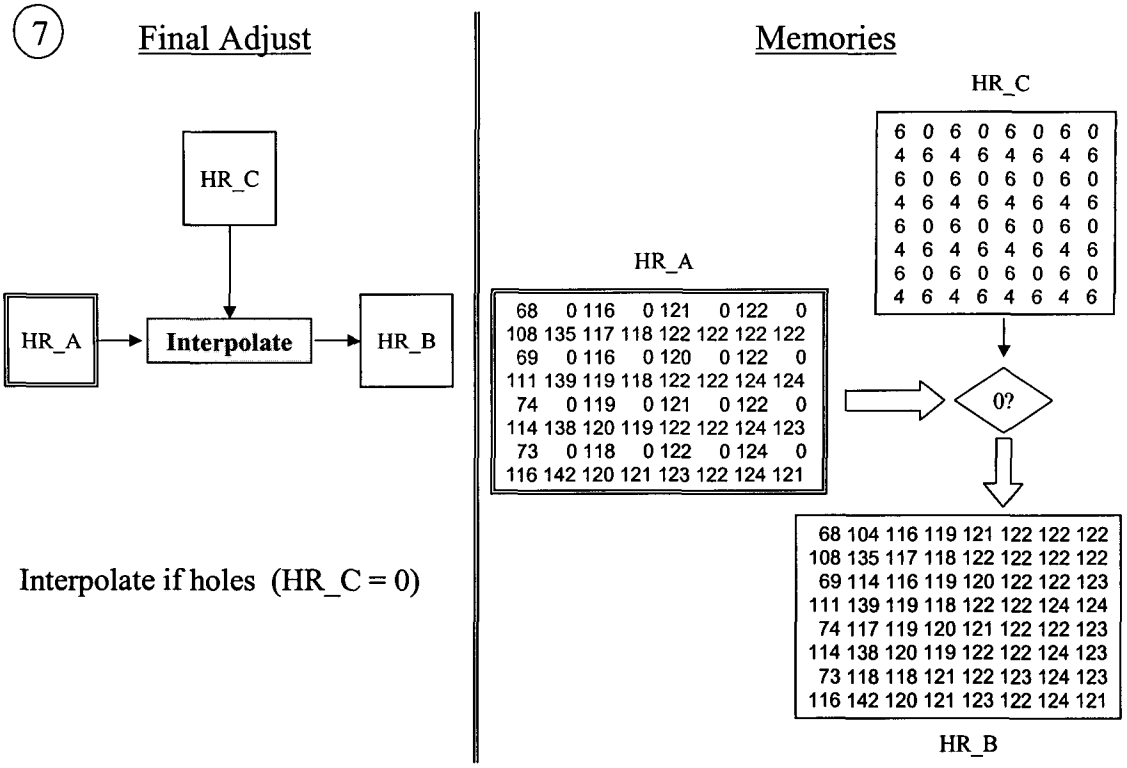


Figure 83. Interpolation of the zeroes of the image with zero contribution.

The values that remains at zero (not information available) will be interpolated in the last step of the algorithm (Figure 83), consulting the values of the contributions to avoid interpolating the inner zeroes of the image, i.e. a pixel can have an allowable value of zero and not for this reason the pixel has to be interpolated. Similarly to the iterative algorithms, the super-resolution image is store in HR_B. This principle of using HR_B to store the final super-resolution image will be keep through all the versions of the super-resolution algorithm for clearness.

5.3.1.1 Modifications in the motion estimator to handle non-iterative algorithms

The Picasso architecture is intended to compress/decompress images, using different commercial standards, but not to image processing applications.

The motion compensator implemented in Picasso is designed to avoid visual distortions in the resulting images when decompressing them, and in that sense, when an image is shifted out of the physic boundaries, it fills the empty zone by replicating the borders. As the motion vectors use to be small compared with the image size and to the less attention of the human eye to the borders compared with the centre of the images, this effect is negligible. But, when we want to obtain super-resolution improvements, the artificial introduction of non-existing data produces important drops of the quality on the borders. As it is obvious, the proper work in this case will be the motion compensator to fill the empty values with zeroes, so that the algorithm will have an opportunity to fill the holes with valid values coming from other images. The algorithm does not expect the spontaneous generation of "new" data that have not been taken into account neither in the previous stages of the algorithm nor in the contributions.

Therefore, the motion compensator must be modified in such a way that it replicates the borders when working in compression/decompression mode and in return, it injects zeroes when working in super-resolution mode.

5.3.1.2 Adjusts to apply the SRA in chrominance

Due to the different sampling scheme used for the luminance and the chrominances, it is necessary to perform some modifications in the proposed algorithm to obtain super-resolution improvements also in the chrominance values.

First of all, the way to take the samples to the high-resolution grid or up-sample, can not be the same that with the luminance. This fact is reflected in Figure 84, where can be seen how every chrominance pixel affects to four luminance pixels, and therefore, when taken to the chrominance high-resolution grid, every pixel must be replicated four times, in order to keep the chromatic coherence, as it is shown in Figure 85 (a).

For the same reason, the initial contributions can not be the same as the luminance. As there is no zero-filling, all the chrominance pixels must initially have the same contributed weight. In Figure 85 (b) are shown the initial contributions for the luminance and the chrominances.

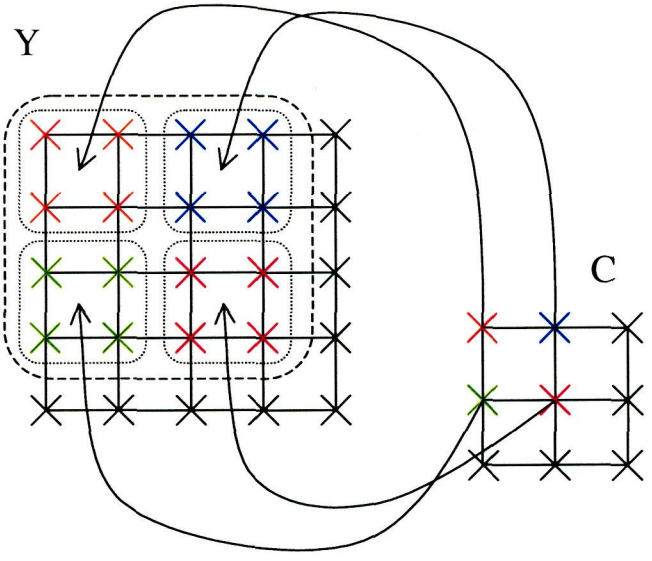
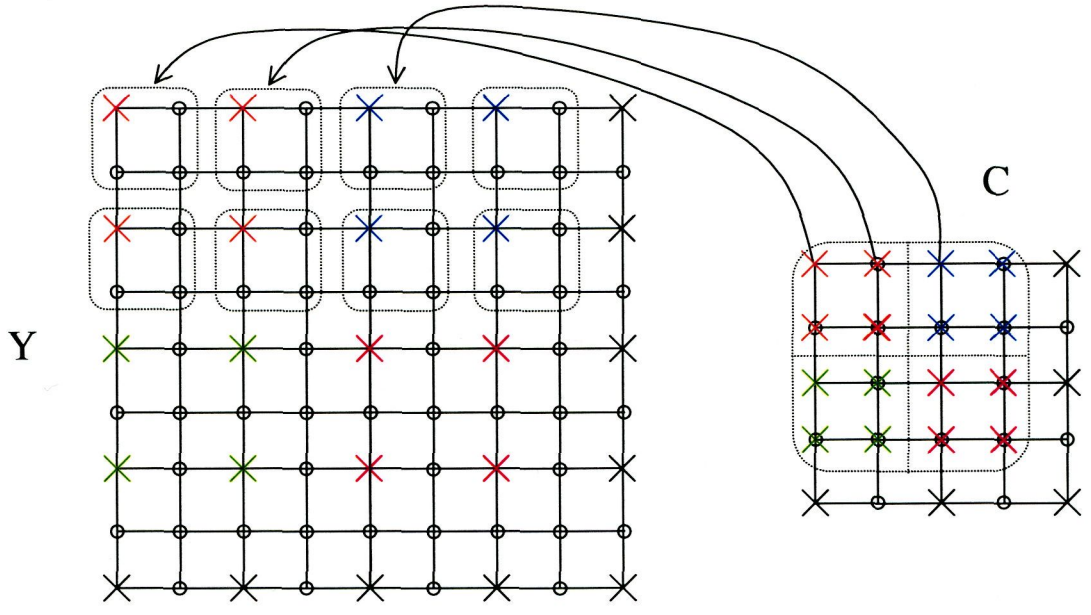


Figure 84. Relationship between the chrominance and the luminance in the sampling format YCbCr 4:2:0.



(a)

Luminance contributions:	4	0
	0	0

Chrominance contributions:	4	4
	4	4

(b)

Figure 85. Mapping of the chrominance C to the high-resolution grid by means of replicating the pixels and its relationship with the luminance Y (a). Initial contributions for the luminance and chrominance images (b).

As the chrominance matrixes contributions are homogenous, it is clear that the motion compensation will not affect them, and so, apparently, this step could be save. But, we can not forget that the motion compensation injects zeroes in the borders, both for the luminance and for the chrominances. That is why keeping the same compensation structure that in the luminance will allow us to correct the border effect in the chrominances.

5.3.2 Motion estimation. Models & parameters.

In this section, some aspects concerning to one of the most critical parts in the super-resolution algorithms will be reviewed: the motion estimator. We will put special emphasis in how the different models affect to the image quality and the parameters that can be adjusted to improve the super-resolution process.

5.3.2.1 Motion estimation models

Two different models have been used:

- The local motion model can be used when some objects are moving in the image. Since the motion estimation is based on macro blocks (16 by 16 pixels), this is the minimum granularity in the motion field.
- The global motion model can be used for a static scene recorded with a moving camera. In other words, all the objects have the same motion.

The motion estimation is performed in several steps:

1. A motion estimation is carried out for each macro block, resulting in a motion vector for each macro block.
2. The average motion vector is computed from the previous motion vectors. This average vector can be used for global motion compensation.
3. A control step is introduced to evaluate the motion estimation metrics, and to choose between local or global motion compensation. This control step also evaluates the possibility of a context change, since no motion compensation can be used in the case of a context change. This decision is taken for each macro-block.

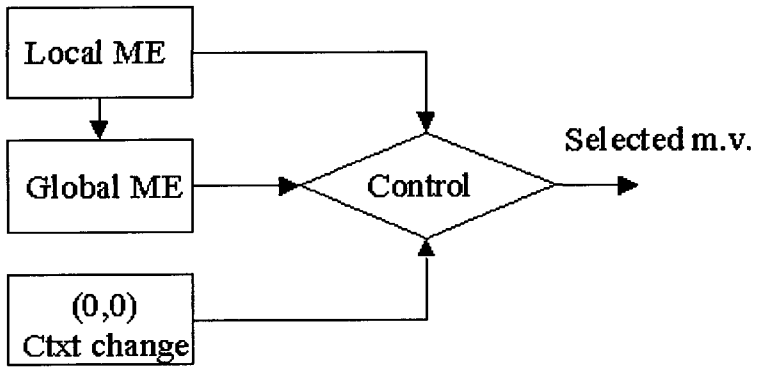


Figure 86. Motion estimator control block.

The evaluated metrics are:

- SAD_LOCAL: Sum of Absolute Differences between the current macro block and the previous macro block, compensated with the local motion vector. This number provides a degree of match between the previous and the current macro block when using the local motion vector.
- SAD_GLOBAL: Sum of Absolute Differences between the current macro block and the previous macro block, compensated with the global motion vector. This number provides a degree of match between the previous and the current macro block when using the global motion vector.
- SAD_INTRA: Sum of Absolute Differences for every macro block between the average value of the pixels and each pixel in the block. It provides a degree of the changes in the block, and in consequence, gives information about the spatial frequencies in the block.

In addition to these metrics, which are obtained from the motion estimator coprocessor, another metric is calculated in software. This is called mv_sad and it is the absolute difference between the local motion vector and global motion vector per macro block. This metric provides a degree of how different the local and global motion vectors are. The larger this metrics is, the more the local vectors are promoted.

Since the global motion vector can only be calculated after completing the local motion estimation, it is not possible to pipeline the motion estimator and compressor. This has the following two consequences:

1. It is necessary to store both the previous and current images in the loop memory loop. Since in the original architecture the motion estimator and the compressor are pipelined, only one image was stored in the loop memory.
2. The time to execute the entire processing is increased because those processors do not run in parallel.

5.3.2.2 HR versus LR motion estimation

An important parameter for the motion estimation is the image resolution. As we are using images in low and high resolution, the motion estimation can be done in both resolutions. This feature can introduce some advantages and disadvantages, as described in this section.

- High resolution images have less aliasing, and hence result in a better motion estimation.
- High resolution images have 4 times more pixels than low resolution images. This requires 4 times more motion estimations, and hence result in 4 times more motion vectors. Recall that the motion estimation is the most compute intensive part of the algorithm.
- Carrying out the motion estimation on high resolution images, results in using the same resolution for both the motion estimator and the compressor. This simplifies the design considerably, since all hardware blocks are working in the same resolution.
- Doing the motion compensation in high resolution based on low resolution motion vectors requires multiplying all motion vectors by a factor of 2. Since all the motion vectors are factors of two, we loose precision in the motion estimation.

Taking into account all this factors, we performed several experiments using motion estimation in low as well in high resolution. The results are shown in the following sections. This is clearly a trade-off between simplicity, compute power and precision.

5.3.2.3 Quarter pixel and half pixel motion estimation

The precision of the motion estimator is very important for super resolution. The Picasso architecture supports half pixel motion vectors, as required by the H.263 encoding standard. However, this architecture has also been extended to quarter pixel motion vectors, by adding another refinement step to the 3DRS algorithm.

This characteristic was also tested in some experiments. The obtained results are shown in the next sections.

5.3.3 Algorithms for still images

In this section of non-iterative SRA for still images two main version will be addressed. The first one, labelled as version v2.0 is intended to cope with a fixed number of low-resolution images, while the second one, labelled as version v2.1 is a generalization of the previous version to combine a variable number of low-resolution images. It will be demonstrated that, provided sufficient and independent amount of data, the final quality will increase with the number of low-resolution images combined.

5.3.3.1 Non-iterative basic SRA for still image (v2.0)

All the ideas exposed until now are gather in the version v2.0 of the SRA, which is shown in Figure 87.

In this version, all the memories are 8-bit wide, except HR_A which is 10 bits, as it is an accumulative memory. The larger value that a pixel can get is 2^8-1=255, so, the larger accumulated value after sum four images is 255·4=1020, value that needs 10 bits to be stored (2^{10}-1=1023). The contributions memory does not need to be 10 bits as the contribution values are small. The maximum value after accumulate four contributions of size 4 is 16, which loosely fits in 8 bits.

We have superscripted with an asterisk the 'motion compensation' operation to point out its different operation with respect to the one used in image compression, in the sense that it inject zeroes in the borders instead of replicating them.

5.3.3.1.1 Block diagram and memory requirements for version v2.0

In Figure 88 is shown the block-diagram of the data-flow for version v2.0 of the super-resolution algorithm shown in Figure 87.

```
* Non iterative super-resolution algorithm  v2.0

Set the value of the improvement: scale
nr_frames = scale*scale, M'= scale*M, N'= scale*N
HR_B, HR_S, HR_T, HR_T2, HR_Cont and HR_S2 all of 8 bits and size [M'][N']
HR_A is 10 bits and size [M'][N']
LR_I[M][N] for the motion estimation
Read a set of aliased Low-Resolution  images in LR_I[#nr_frames][M][N]

// First Image is the reference
MV_ref2fr[0] = 0,0

FOR fr = 1 .. 3
MV_ref2fr[fr] = Calc_Motion_Estimation ( LR_I[fr], LR_I[0])
IF Global_Motion THEN Select_global_motion_vector()
MV_ref2fr[fr] = 2 .* MV_ref2fr[fr]
END FOR

HR_A.lum    = Upsample_Holes(LR_I[0].lum)                    [SR_INIT_A]
HR_A.chrom = Upsample_Neighbours(LR_I[0].chrom)

HR_Cont     = Create_image_contributions()                   [SR_INIT_CONT]

FOR fr = 1 .. 3
    HR_S.lum    = Upsample_Holes(LR_I[fr].lum)               [SR_UPSAMPLE]
    HR_S.chrom = Upsample_ Neighbours(LR_I[fr].chrom)

HR_S2 = Create_image_contributions()

HR_T    = Motion_Compensation*(HR_S, MV_ref2fr[fr])          [SR_MOT_COMP]
HR_T2   = Motion_Compensation*(HR_S2, MV_ref2fr[fr])         [SR_MOT_COMP_CONT]
HR_A    = HR_A + HR_T                                        [SR_ADD]
HR_Cont = HR_Cont + HR_T2                                    [SR_ADD_CONT]
END FOR

HR_A = 4*HR_A/HR_Cont                                        [SR_ADJUST_A]
 IF (Hole)   HR_B = Interpolate(HR_A)                        [SR_UPDATE]
 Else        HR_B = HR_A

Clip(HR_B, 0, 255)

High Resolution result image in HR_B
```

Figure 87. Pseudo-code of the non-iterative algorithm version v2.0.

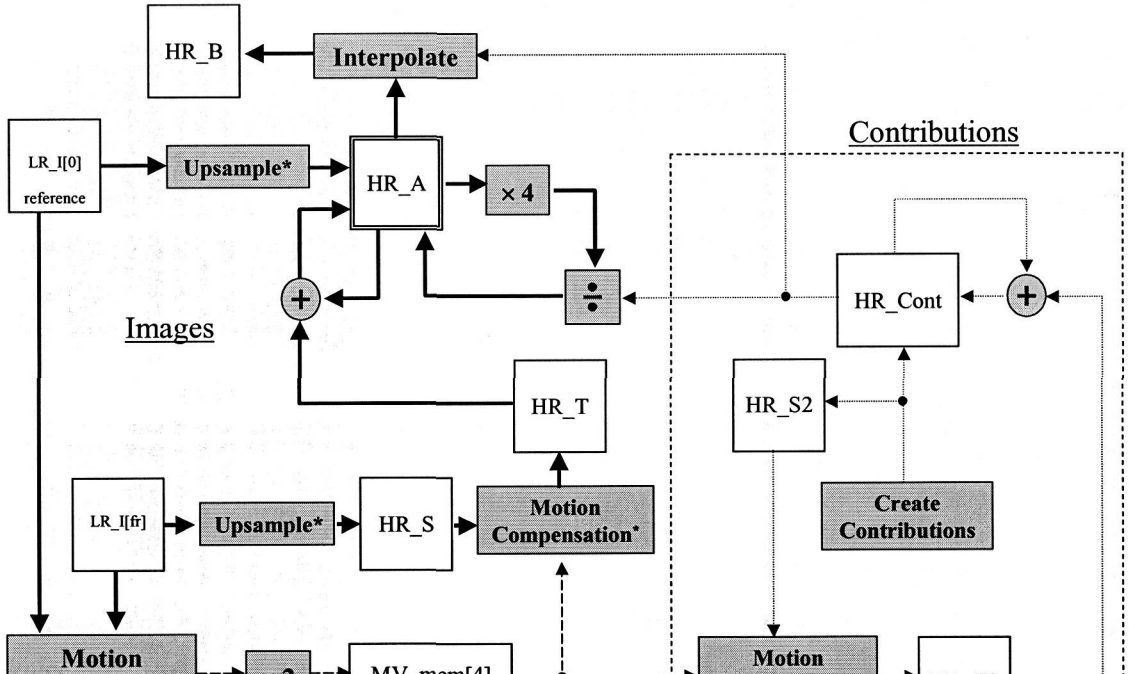


Figure 88. Block-diagram of the super-resolution algorithm version v2.0.

The block-diagram has been divided, on one side, in the zone that performs the image processing, that makes use of memories HR_A, HR_T, HR_S and LR_I[0..3], besides storing the motion vectors in MV_mem[0..3], and in the other side, in a zone for the contributions processing, that makes use of memories HR_S2, HR_T2 and HR_Cont. To highlight the relations among them, we have denoted with solid lines the images flow, with dotted lines the contributions flow and in dashed lines the motion vectors flow. Moreover, the functions of '*upsample*' and '*motion compensation*' have been superscripted with an asterisk to remark their different behavioural when used in super-resolution.

In Table 6 is shown a summary of the memory requirements for version v2.0 of the SRA, as described in Figure 88. It must be reminded that all memories are 8 bit except HR_A which is 10 bit.

Memories HR_T and HR_T2 are not included in this measures as they are used in this algorithm version to avoid the data overlap when performing the motion compensation, but this problem will not exist in the hardware version over Picasso, as there exist a buffer of

three macro-block slices that avoid that problem. In Table 15 are shown the total memory requirements for several memory sizes.

Denomination	Memory		
	Luminance (bits)	Chrominance (bits)	Total (bits)
HR_A	*(2·mb_x·2·mb_y·16·16·10)*	*(2·mb_x·2·mb_y·8·8·2·10)*	*15,360·mb_x·mb_y*
HR_B	*(2·mb_x·2·mb_y·16·16·8)*	*(2·mb_x·2·mb_y·8·8·2·8)*	12,288·*mb_x·mb_y*
HR_S	*(2·mb_x·2·mb_y·16·16·8)*	*(2·mb_x·2·mb_y·8·8·2·8)*	12,288·*mb_x·mb_y*
HR_S2	*(2·mb_x·2·mb_y·16·16·8)*	*(2·mb_x·2·mb_y·8·8·2·8)*	12,288·*mb_x·mb_y*
HR_Cont	*(2·mb_x·2·mb_y·16·16·8)*	*(2·mb_x·2·mb_y·8·8·2·8)*	12,288·*mb_x·mb_y*
3 Stripes HR	(2·3·2·mb_y·16·16·8)	(2·3·2·mb_y·8·8·2·8)	36,864·mb_y
LR_I[0]	*(mb_x·mb_y·16·16·8)*	*(mb_x·mb_y·8·8·2·8)*	*3,072·mb_x·mb_y*
LR_I[1]	*(mb_x·mb_y·16·16·8)*	*(mb_x·mb_y·8·8·2·8)*	*3,072·mb_x·mb_y*
LR_I[2]	*(mb_x·mb_y·16·16·8)*	*(mb_x·mb_y·8·8·2·8)*	*3,072·mb_x·mb_y*
LR_I[3]	*(mb_x·mb_y·16·16·8)*	*(mb_x·mb_y·8·8·2·8)*	*3,072·mb_x·mb_y*
MV_mem[0]	(mb_x·mb_y·8)	0	8· *mb_x·mb_y*
MV_mem[1]	(mb_x·mb_y·8)	0	8· *mb_x·mb_y*
MV_mem[2]	(mb_x·mb_y·8)	0	8· *mb_x·mb_y*
MV_mem[3]	(mb_x·mb_y·8)	0	8· *mb_x·mb_y*
Total (bits)	mb_y · (51,232· mb_x + 24,576)	mb_y · (25,600· mb_x + 12,288)	mb_y · (76.832·mb_x+ 36,864)

Table 14. Summary of the memory requirements of version v2.0 of the SRA.

Size	mb_x	mb_y	Memory (Kbytes)	Memory (Mbytes)
SQCIF	8	6	477.19	0.47
QAVGA	9	7	622.37	0.61
QCIF	11	9	969.01	0.95
HAVGA	18	14	2,426.48	2.37
CIF	22	18	3,795.05	3.71
AVGA	36	28	9,579.94	9.36
VGA	40	30	11,389.69	11.12
4CIF	44	36	15,018.19	14.67
16CIF	88	72	59,748.75	58.35

Table 15. Memories used by version v2.0 of the super-resolution algorithm for several image sizes.

In Figure 89 are shown the data of Table 15 expressed in Kbytes. The memory requirements are greater that in the case of the iterative algorithms because we are reserving a whole memory for the contributions. Further on we will see that the contributions adjust can be done at the macro-block level, what will save two high-resolution memories.

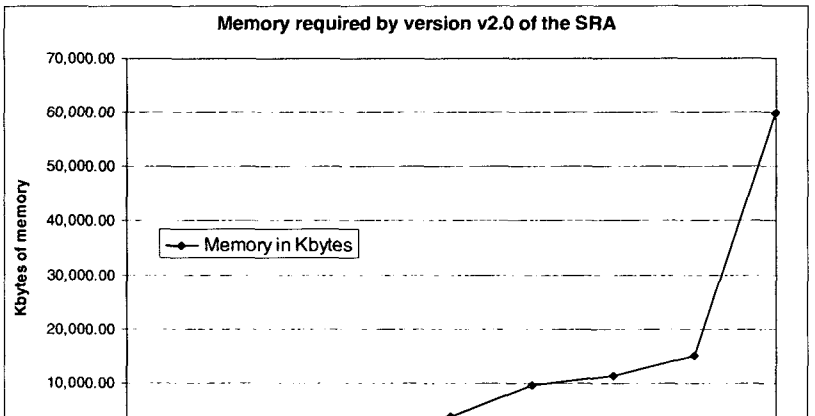


Figure 89. Memory used by the super-resolution algorithm version v2.0 for the most frequent memory sizes.

5.3.3.1.2 Simulation results and quality analysis of version v2.0 using ½ pixel precision

With the intention of performing a comparison with the previous iterative versions, the first simulations have been carried out using the same test images (Kantoor, CIF format (352x288) sub-sampled to QCIF (176x144) to feed the SRA that will submit an image of CIF size again), the same starting frame (frame number zero) and the same displacements. Therefore, any change in the resulting image quality will be only due to algorithm modifications and in any case due to changes in the input data. The vector set is shown in Table 13 and therefore their types are indicated in Table 12. The results for this test sequence are shown if Figure 90.

Comparing these results with the ones obtained for versions v1.2 and v1.3 (Figure 91), we can observe that the PSNR of the whole image (with borders) is significantly higher.

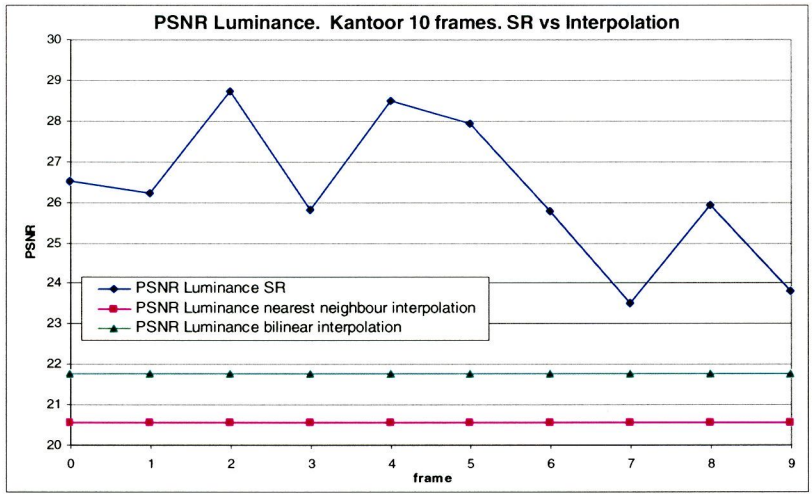


Figure 90. PSNR of the output sequence using version v2.0 of the SRA versus the interpolated images using nearest neighbour and bilinear interpolation.

If additionally the borders from the PSNR computation are subtracted, the chart of Figure 92 is obtained. In this second case, the results for images of type 'a' are really amazing. The PSNR is bounded to 99.9 dB, which would the value reached when both signals were identical, as it is happening in this case. This astonishing precision is because in the reconstruction process we are in fact following the inverse process to the test image generation. It is not so weird a so accurate result. As a matter of fact, the proposed super-resolution algorithm is inspired on this philosophy, which has been necessary to assess using

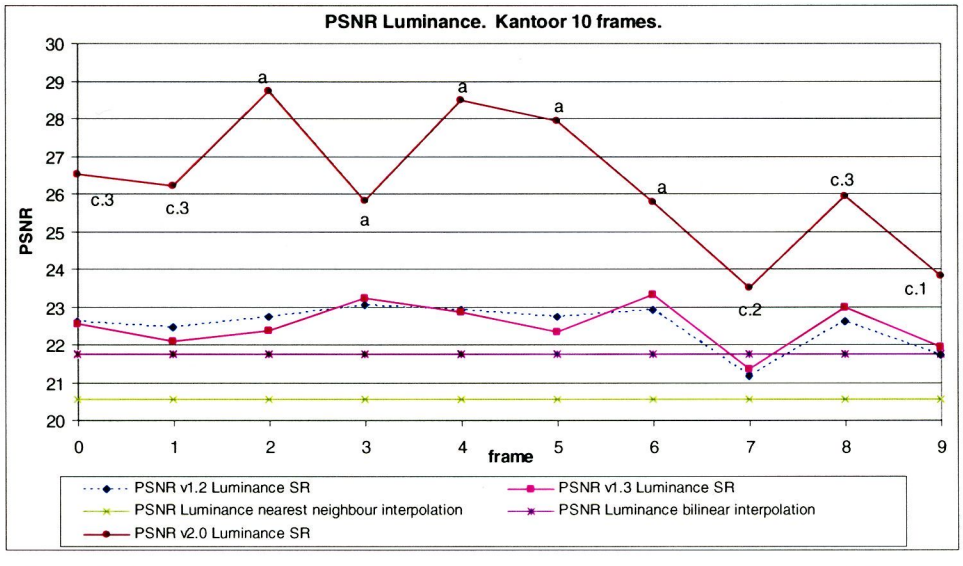


Figure 91. PSNR comparison using the KANTOOR sequence for version v2.0 of the SRA versus the interpolated images and the iterative algorithms of versions v1.2 and v1.3.

real images and implementing it with the available resources.

As we did for the previous algorithm versions, instead of showing all the image sequences, we will only show one of each significant type: the frame 0 of type 'c.3', the frame 4 of type 'a', the frame 7 of type 'c.2' and the frame 9 of type 'c.1'.

The obtained super-resolution frames are shown in Figure 93. At the left-hand side, labelled with extension '1', are shown the images of CIF size, and at the right-hand side, labelled with extension '2', are shown the associated errors when compared with the original images. In the same way as in the previous cases, images of type 'a' (Figure 93 b) exhibit the higher quality and consequently less image error, which is nearly zero in the central part of the image. The divergences are due to the chrominances, whose quality does not reach the luminance quality because of the previously commented problems about divergences of the motion vectors between the used motion vectors (the luminance ones) and the real chrominance motion vectors. Images of type 'c.1' (Figure 93 d) and 'c.2' (Figure 93 c) exhibit an error predominance in the vertical and horizontal directions respectively. This fact is also manifested in their bi-dimensional Fourier transforms, both in magnitude (Figure 94) and in phase (Figure 95), where the horizontal error spectral components are almost zero in magnitude and phase for images of type 'c.2' (Figure 94 c.2 and Figure 95 c.2) and almost zero in the vertical direction (Figure 94 d.2 and Figure 95 d.2) for images of type 'c.1'.

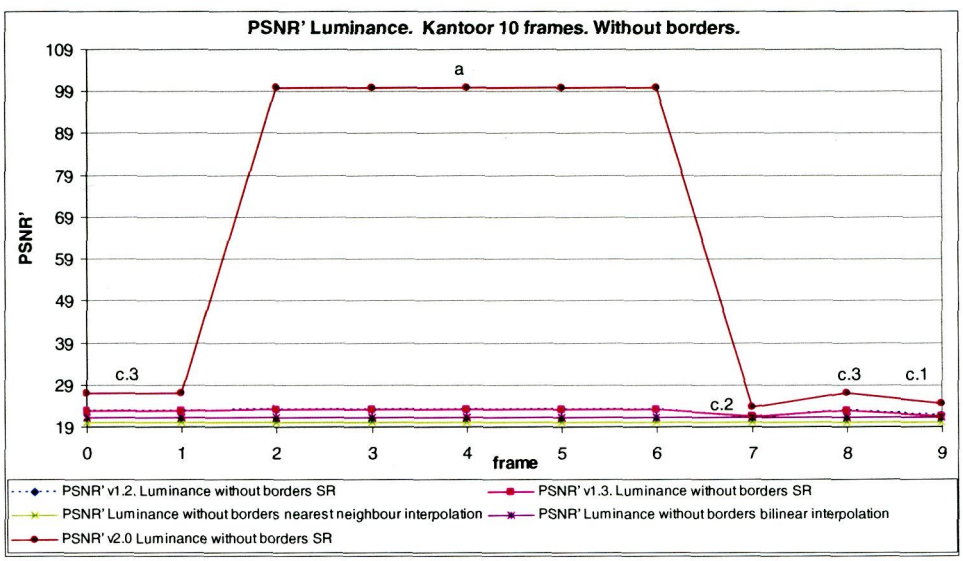


Figure 92. Comparison of the PSNR' without borders of the KANTOOR sequence for version v2.0 of the SRA versus the interpolated images and the iterative algorithms versions v1.2 and v1.3 .

For images of type 'a', it can be appreciated the very low error exhibit in the spatial domain and in a very wide zone of low and medium frequencies in the frequency domain. This low error in the high frequencies is a clear indicative of information recovery in steep areas as can being the edges of the objects contained in the image or the small details in the background.

Although the chrominances use to have higher PSNR than the luminance due to their lower entropy, as it really happen when the borders are taken into account (Figure 96 a), it is not the same case when the image borders are removed (Figure 96 b). Since the process is better tuned for luminance than for chrominances, when all the input samples are available (images of type 'a') the PSNR of the luminance without borders is higher than the PSNR of the chrominances.

Nevertheless, it is very interesting to perform a comparative between the PSNR values of the chrominance (for instance the red chrominance that exhibits higher average PSNR) achieved by this algorithm version and the iterative versions. If we represent the PSNR of the full images (Figure 97 a) the image quality seems to be almost the same, but if the borders are removed from the PSNR computation (Figure 97 b), then an important improvement in the quality of the chrominances using this new algorithm version can be appreciated, obtaining an average improve of 1.25 decibels against the bilinear interpolation, of 1.82 decibels against version v1.2 and of 1.09 decibels against version v1.3.

The spectral correlations in magnitude and phase, with and without borders (Figure 98), show the expected behavioural, coherent with the PSNR variations. It is acceptable to emphasize the high phase spectral correlation obtained, much higher than the one reached by the iterative algorithms, even in the case of including the image borders in the computation. Only in the frame number 9 the spectral correlation drops below the interpolation level, what it is recovered after removing the borders.

It is also interesting to highlight a higher independence with the image borders in the new algorithm. This is confirmed through the higher similitude of the curves with and without borders. The reason is that this algorithm tries to recover information from the image borders, interpolating the missing data, but never replicating them, thanks to the modifications introduced in the motion compensator.

As it was expected, the image borders elimination produces spectral correlation of unity value, what evidences the exact equivalence between the images of type 'a' without borders obtained and the reference images, without borders for enabling the comparison.

5.3.3.1.3 Simulation results and quality analysis of version v2.0 using quarter pixel precision

All the results shown until now have been obtained using motion estimation in low-resolution with half pixel precision. However, it is clearly desirable to increase the precision of the motion estimator to quarter pixel, and consequently the motion estimator has been modified in this sense. Also the experimental setup has been modified, as has been described in section 3.5 (experimental setup). We have opted for using again this same algorithm (version v2.0) but using a higher precision in the estimation of the displacements. For this new test we will use a new sequence call KRANT[1] of VGA size, which has a significant spatial distribution of the objects in the scene, what leads to a frequency spectrum with clearly oriented components, much less homogeneous that in the case of KA[2]NTOOR.

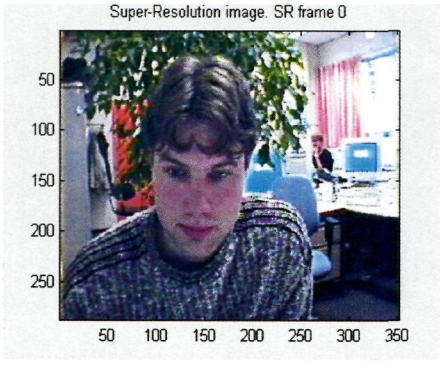
When changing the motion estimation accuracy, the base-cell also changes, and we have to use the base-cell shown in Figure 60. However, while in the case of the motion estimation with half pixel accuracy, it is possible to perform a super-resolution images classification based on the position of the available samples, in this new case the total number of possibilities is too high to establish an affordable classification. Concretely, the total number of combinations, taking the samples four by four, and being N the total number of pixels in the base-cell, is the one expressed in (29), that for N=4 (base-cell of half pixel) results in 15 (these are the classifications made) but for N=16 (base-cell of quarter pixel) results the much higher figure of 2516 combinations.

$$\sum_{k=1}^{4} \frac{N!}{k! \cdot (N-k)!} \tag{29}$$

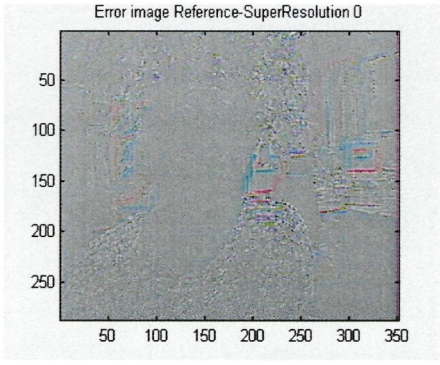
Given the huge amount of different samplings possibilities, we desist from making an exhaustive classification, but it is still valid the general principle that the sampled selected by the shifts will drastically affect the quality of the super-resolved images, reaching the maximum quality when the shifts include the four pixel positions, that correspond to the shift vectors (0,0), (0,2), (2,0) and (2,2) expressed in unities of very high resolution (VHR) pixels.

[1] KRANT means newspaper in Dutch.

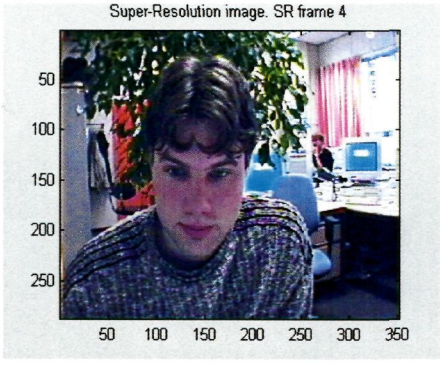
[2] KANTOOR means office in Dutch.



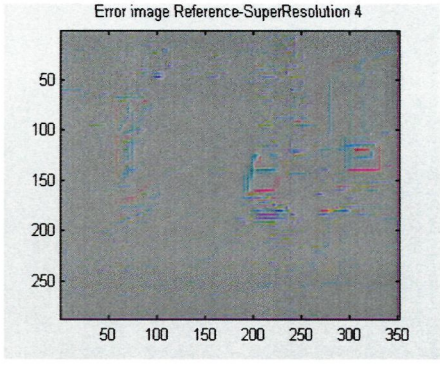
(a.1)



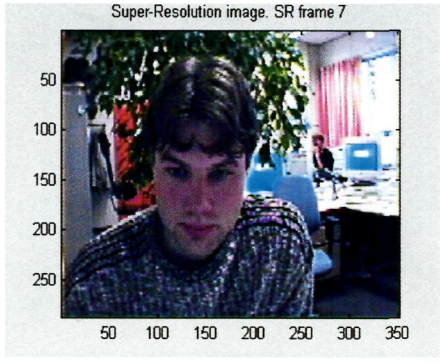
(a.2)



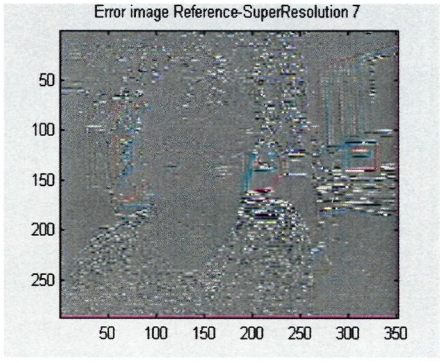
(b.1)



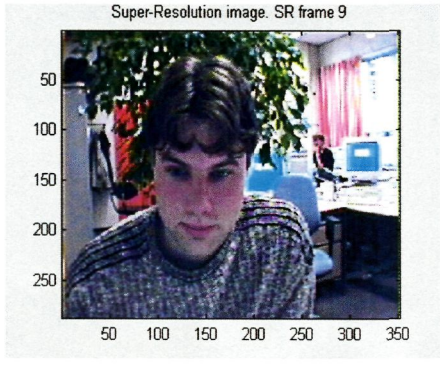
(b.2)



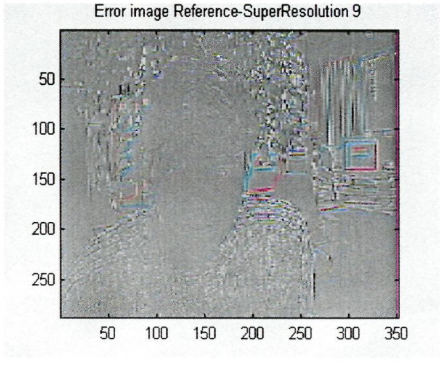
(c.1)



(c.2)



(d.1)



(d.2)

Figure 93. Super-resolved images (1) with their associated errors (2), of frame 0 (a), of frame 4 (b), of frame 7 (c) and of frame 9 (d).

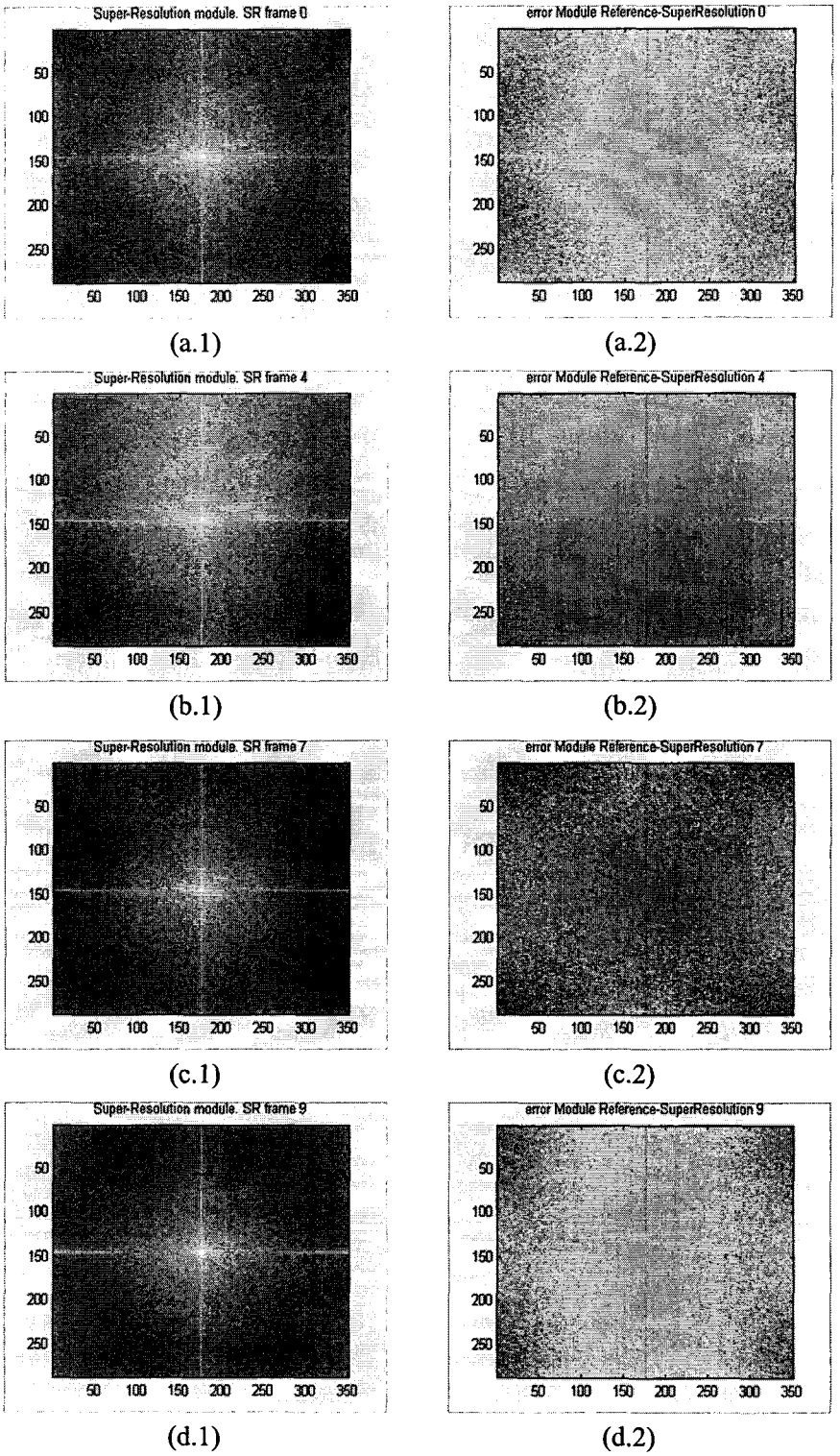


Figure 94. Bi-dimensional Fourier transforms in magnitude of the super-resolved images (1) with their associated errors (2), of frame 0 (a), of frame 4 (b), of frame 7 (c) and of frame 9 (d).

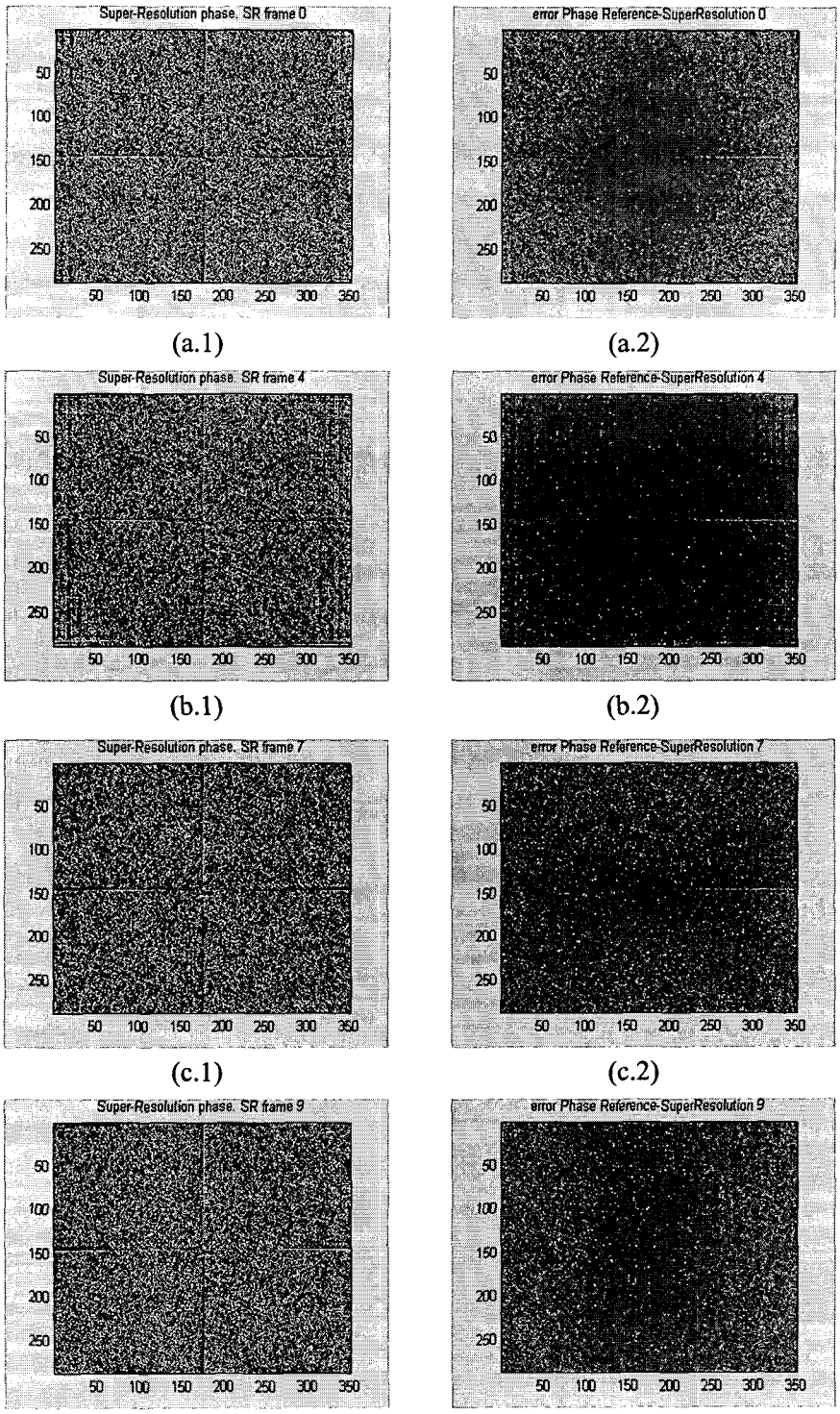
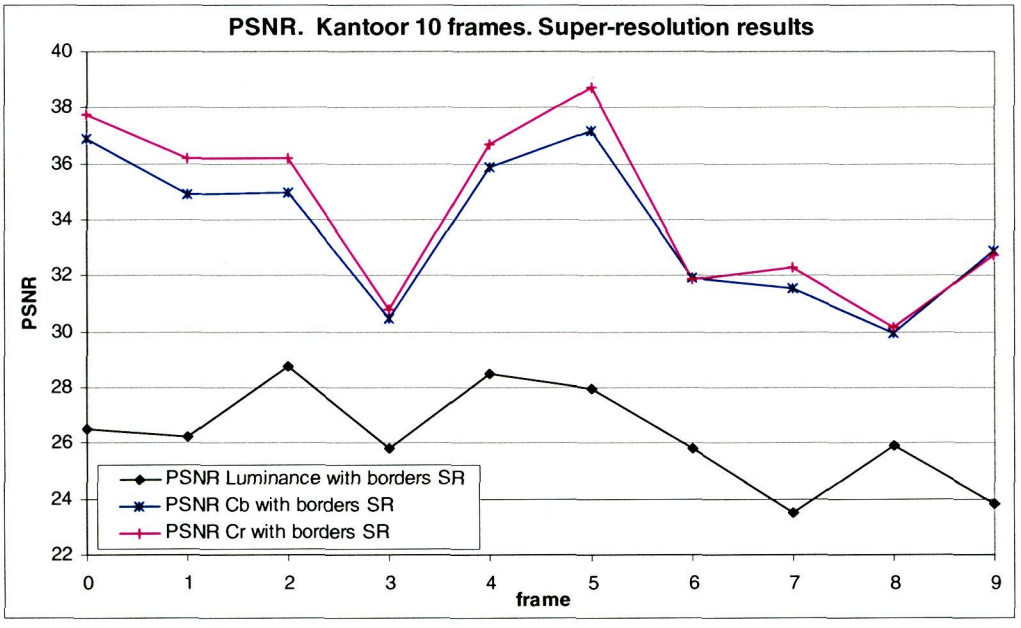
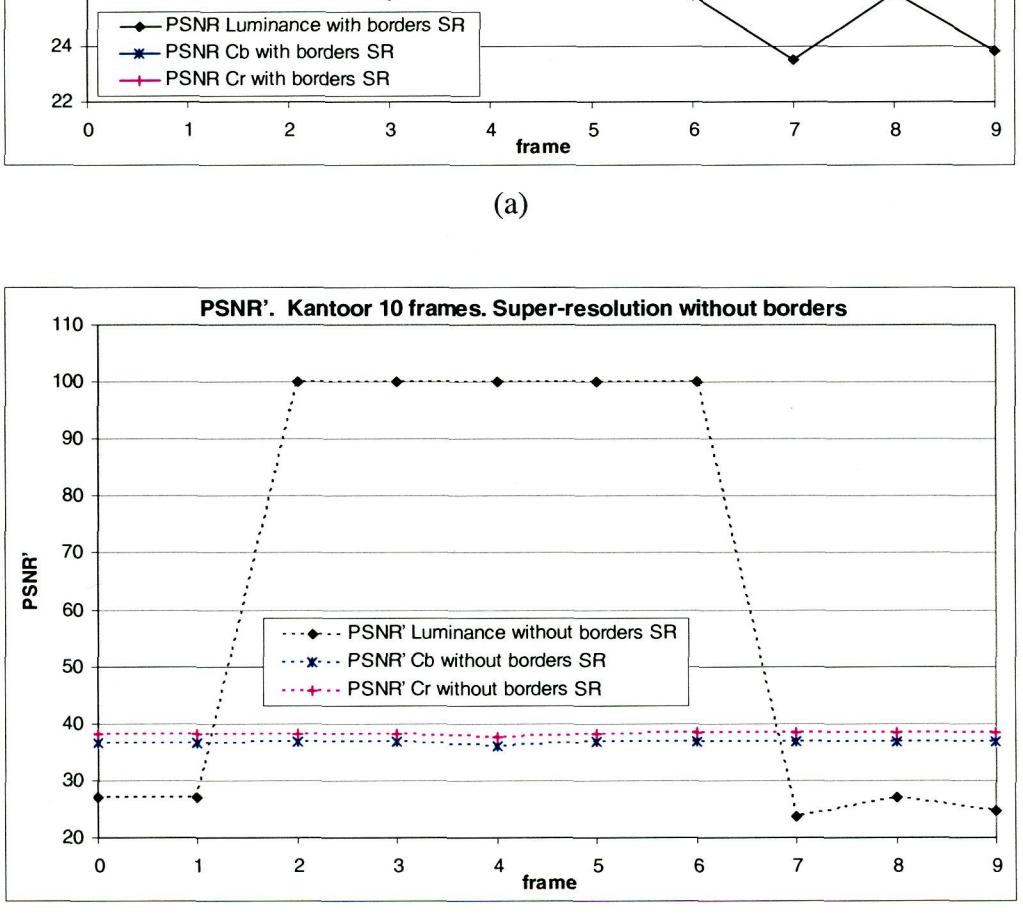


Figure 95. Bi-dimensional Fourier transforms in phase of the super-resolved images (1) with their associated errors (2), of frame 0 (a), of frame 4 (b), of frame 7 (c) and of frame 9 (d).

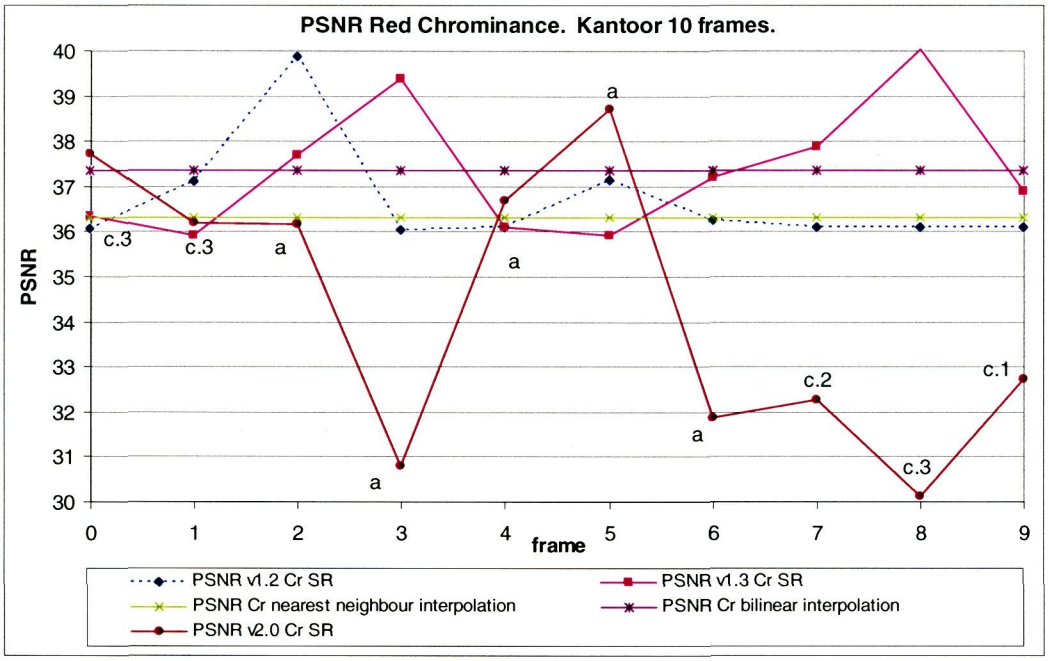


(a)

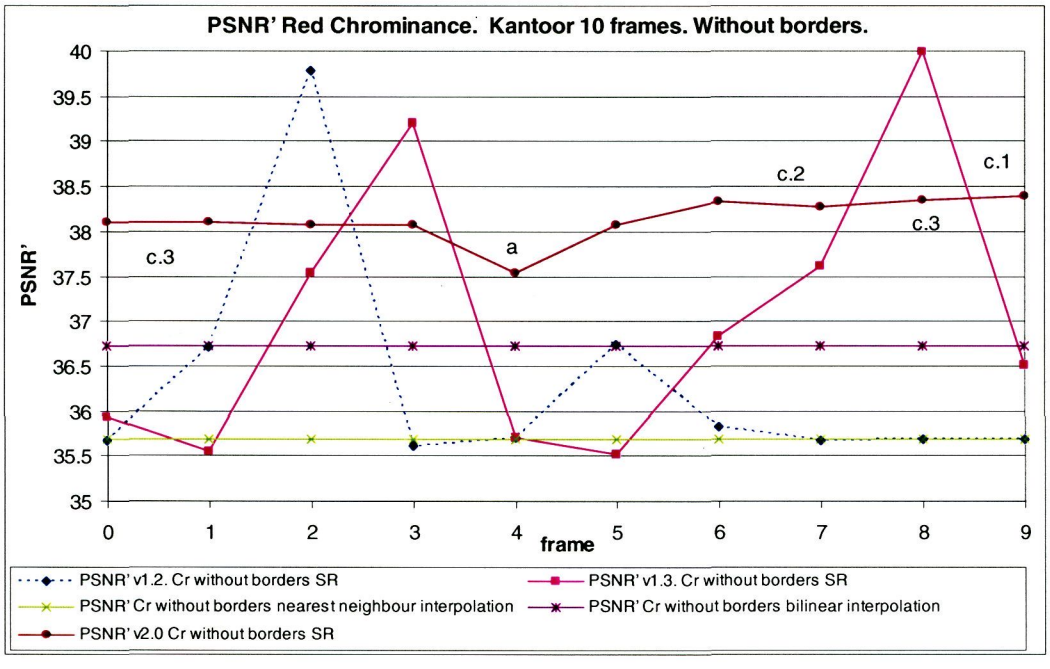


(b)

Figure 96. PSNR' for version v2.0 of the luminance and chrominances using the full images (a) and without borders (b).



(a)



(b)

Figure 97. PSNR' of version v1.2, v1.3 and v2.0 and the interpolated images of the red chrominance using the full images (a) and without borders (b).

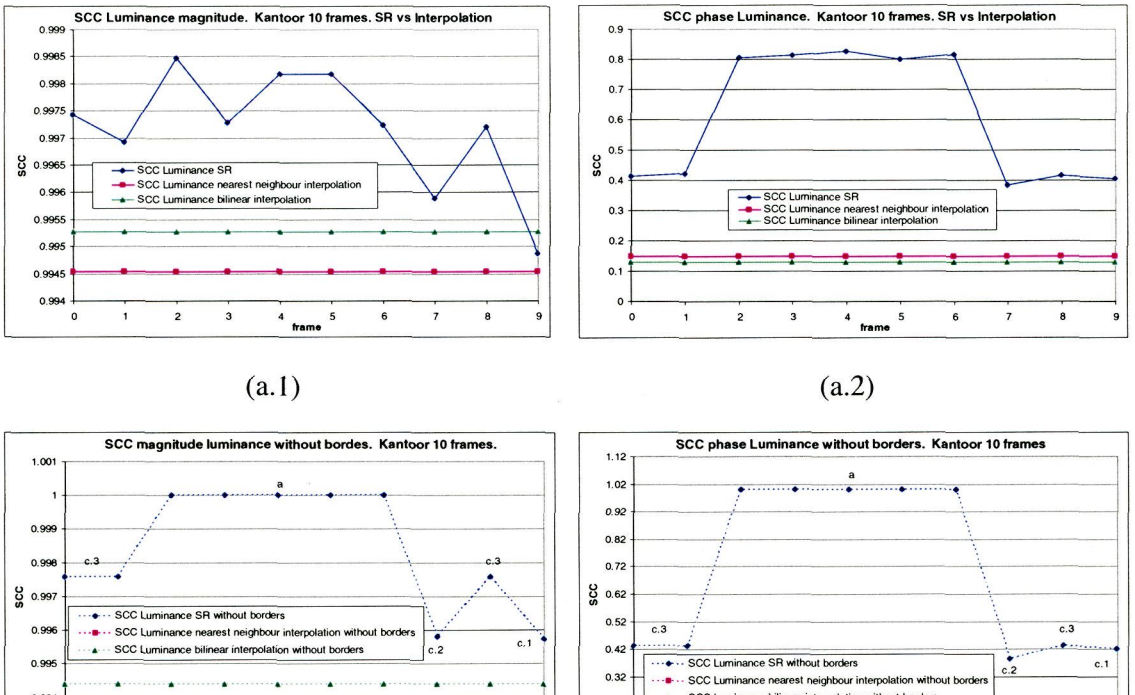


Figure 98. Spectral correlations in magnitude (1) and phase (2) of the luminance using the full

In Figure 99 can be seen the relationships among the changes in the scale and the unities across the super-resolution and test image generation process. In the bottom of the chart has been shown an example of shift with value (2,2) in very high resolution pixels, name given to the image of big format, as could be VGA or CIF, to lately crop it, avoiding the undesirable border effect. Starting from that image, cropped and shifted, it is decimated by a factor of ½ and filtered to obtain the High-Resolution (HR) image that will serve as the reference in order to obtain the quality measures. At the same time, the image is sub-sample by a factor of ¼ to generate aliasing, what will suppose the Low-Resolution (LR) input sequence to the SRA. Once inside the developed system, the used unities are of quarter pixel, that can be equally obtained multiplying by four the shifts in pixel unities. When going from low-resolution to high resolution, both the images and the motion vectors are multiplied by two, and can be verified that the shift generated in Very-High Resolution (VHR) of (2,2) pixels correspond to a shift in the high-resolution image of (1,1) pixels. Although the VHR image is really shifted (2,2) pixels, it is obvious that the relation between the HR and VHR images is a scale factor of two. Therefore, we can see that shifts in unities of half pixel of high-resolution. Thus, two

half-pixels are equivalent to one pixel, and can be seen in advance that the (2,2) shift in VHR is equivalent to (1,1) in high resolution.

For the test sequence KRANT, the first frame is selected, and are applied the displacement vectors (expressed in very high resolution pixels) shown in Table 16, generated 17 output frames. All the vectors are randomly generated, except the first set of four vectors, that it is artificially generated in the integer pixel positions of high-resolution to check the maximum quality that the algorithm can achieve.

In Figure 100 is shown the reference frame (a.1) together with its bi-dimensional Fourier transform in magnitude (a.2) and the low-resolution input sequence (b.1-e.1) together with their correspondent bi-dimensional Fourier transform in magnitude (b.2-e.2).

As the image presents a clear orientation of the objects contained on it, its bi-dimensional Fourier transform in magnitude will present very directional high-energy spectral components. This feature helps to clearly identify the great amount of aliasing in the input sequence. Since the aliasing is located in the pass-band of signal, it cannot be removed by conventional filtering techniques.

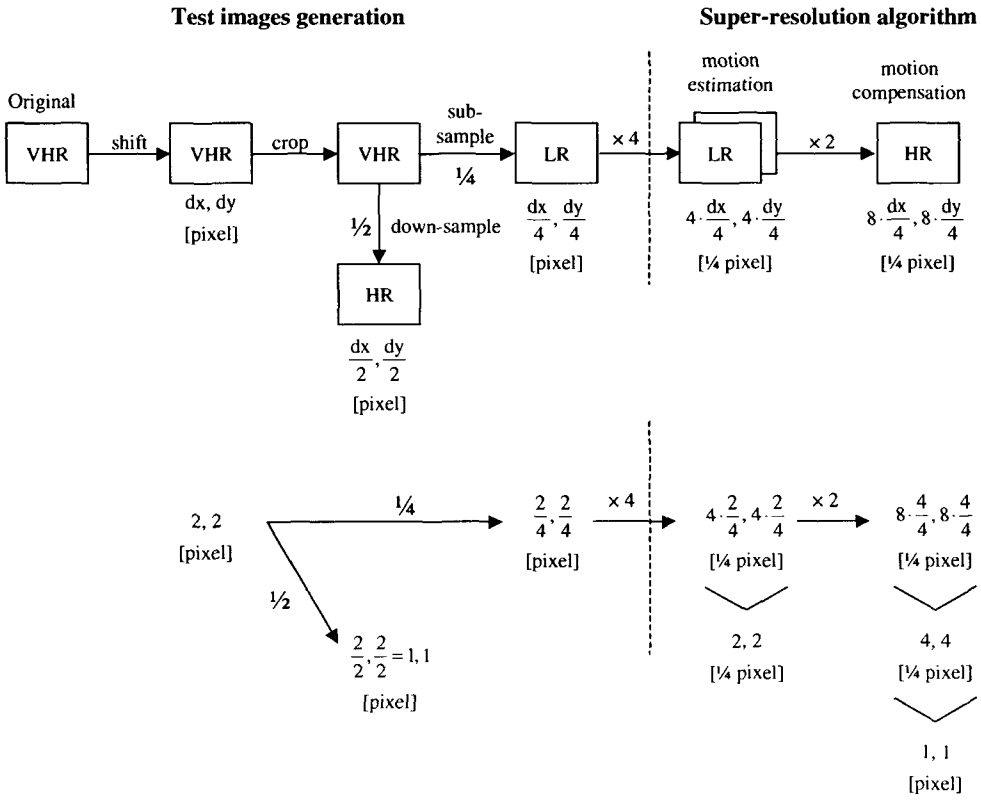


Figure 99. Unities and scaling relationship among the shifts during the test and super-resolution image generation process.

Also for reference are shown the interpolated images (a), in magnitude (b) and phase (c), both for nearest neighbour interpolation (Figure 101) and for bilinear interpolation (Figure 102), together with their associated errors (2).

After the execution of the algorithm over 17 frames of the image shown in Figure 100 (a.1), sub-sample in a factor of quarter after being shifted the amounts and directions given by the displacement vectors of Table 16, the PSNRs shown in Figure 103 (luminance with

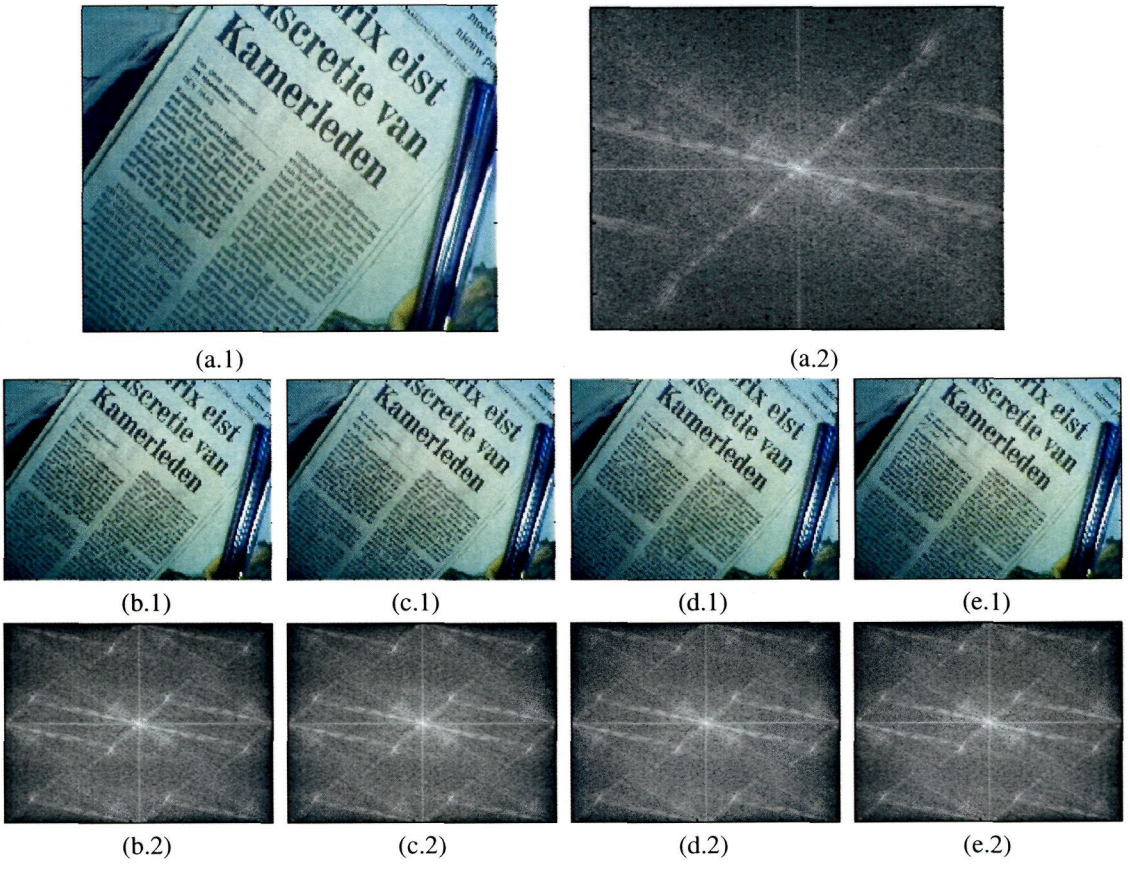


Figure 100. Frame zero of the test sequence KRANT (a.1) together with its bi-dimensional Fourier transform in magnitude (a.2) and the low-resolution input sequence (b.1-e.1) together with their correspondent bi-dimensional Fourier transform in magnitude (b.2-e.2).

borders) have been obtained. The frame zero exhibits a PSNR of 99.9 decibels, as it was foreseeable as it has all the input samples in integer pixel positions. The average PSNR (without taking into account the output frame zero) is 30.47 dB, that overcomes in 7.28 dB the average obtained for version v1.2 and in 2.23 dB the average obtained for version v1.3.

frame	Very high resolution vectors		High resolution vectors		Low resolution vectors	
0	0	0	0	0	0	0
	2	0	1	0	0.5	0
	0	2	0	1	0	0.5
	2	2	1	1	0.5	0.5
1	0	0	0	0	0	0
	-1	-3	-0.5	-1.5	-0.25	-0.75
	-1	-3	-0.5	-1.5	-0.25	-0.75
	-1	-2	-0.5	-1	-0.25	-0.5
2	0	0	0	0	0	0
	3	2	1.5	1	0.75	0.5
	3	1	1.5	0.5	0.75	0.25
	2	0	1	0	0.5	0
3	0	0	0	0	0	0
	3	-3	1.5	-1.5	0.75	-0.75
	2	-1	1	-0.5	0.5	-0.25
	1	0	0.5	0	0.25	0
4	0	0	0	0	0	0
	3	2	1.5	1	0.75	0.5
	1	3	0.5	1.5	0.25	0.75
	2	2	1	1	0.5	0.5
5	0	0	0	0	0	0
	-2	2	-1	1	-0.5	0.5
	-2	3	-1	1.5	-0.5	0.75
	-1	1	-0.5	0.5	-0.25	0.25
6	0	0	0	0	0	0
	-2	-3	-1	-1.5	-0.5	-0.75
	0	-1	0	-0.5	0	-0.25
	-1	-2	-0.5	-1	-0.25	-0.5
7	0	0	0	0	0	0
	2	2	1	1	0.5	0.5
	0	1	0	0.5	0	0.25
	1	2	0.5	1	0.25	0.5
8	0	0	0	0	0	0
	0	3	0	1.5	0	0.75
	-3	1	-1.5	0.5	-0.75	0.25
	0	1	0	0.5	0	0.25
9	0	0	0	0	0	0
	3	-1	1.5	-0.5	0.75	-0.25
	2	-3	1	-1.5	0.5	-0.75
	0	-1	0	-0.5	0	-0.25
10	0	0	0	0	0	0
	2	-3	1	-1.5	0.5	-0.75
	2	-1	1	-0.5	0.5	-0.25
	2	-2	1	-1	0.5	-0.5
11	0	0	0	0	0	0
	-1	2	-0.5	1	-0.25	0.5
	0	2	0	1	0	0.5
	0	1	0	0.5	0	0.25
12	0	0	0	0	0	0
	1	2	0.5	1	0.25	0.5
	3	3	1.5	1.5	0.75	0.75
	2	1	1	0.5	0.5	0.25
13	0	0	0	0	0	0
	0	1	0	0.5	0	0.25
	0	0	0	0	0	0
	-3	0	-1.5	0	-0.75	0
14	0	0	0	0	0	0
	-2	-3	-1	-1.5	-0.5	-0.75
	0	-1	0	-0.5	0	-0.25
	-3	-1	-1.5	-0.5	-0.75	-0.25
15	0	0	0	0	0	0
	-1	0	-0.5	0	-0.25	0
	-1	-1	-0.5	-0.5	-0.25	-0.25
	-1	-1	-0.5	-0.5	-0.25	-0.25
16	0	0	0	0	0	0
	-2	-3	-1	-1.5	-0.5	-0.75
	-2	-2	-1	-1	-0.5	-0.5
	0	0	0	0	0	0

Table 16. Displacement vectors randomly generated expressed in very high-resolution pixels, in high-resolution pixels and their reduction to canonical vectors in the base cell of ¼ pixel.

Showing in a same chart the PSNR of the sequence with and without borders (Figure 103) it can be appreciated that the qualities are very similar (the average error is 0.35 dB). As the only difference with the experiments of the previous section relies on the difference in the motion estimation accuracy, we can conclude that the precision increase in the motion estimator makes this version of the algorithm less sensitive to the undesirable border effects.

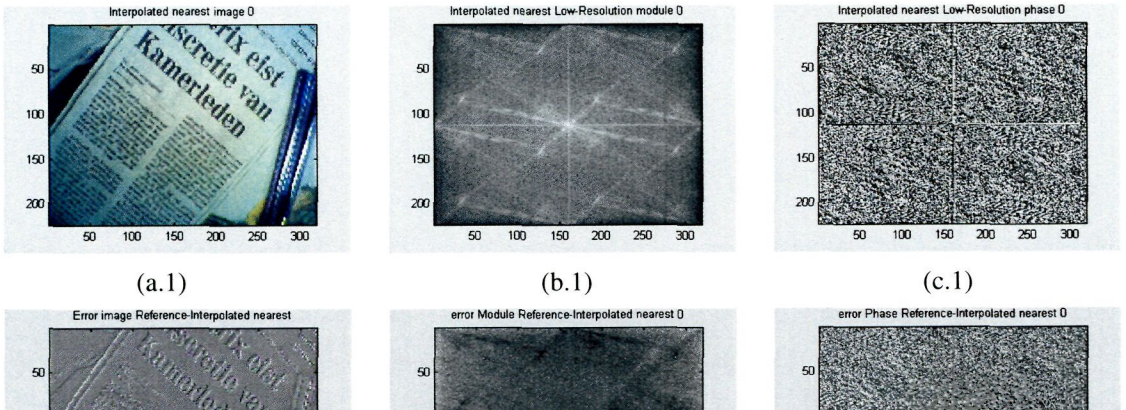


Figure 101. Nearest neighbour interpolated image, in the spatial domain (a), in the frequency domain in magnitude (b) and in phase (c), together with their associated errors.

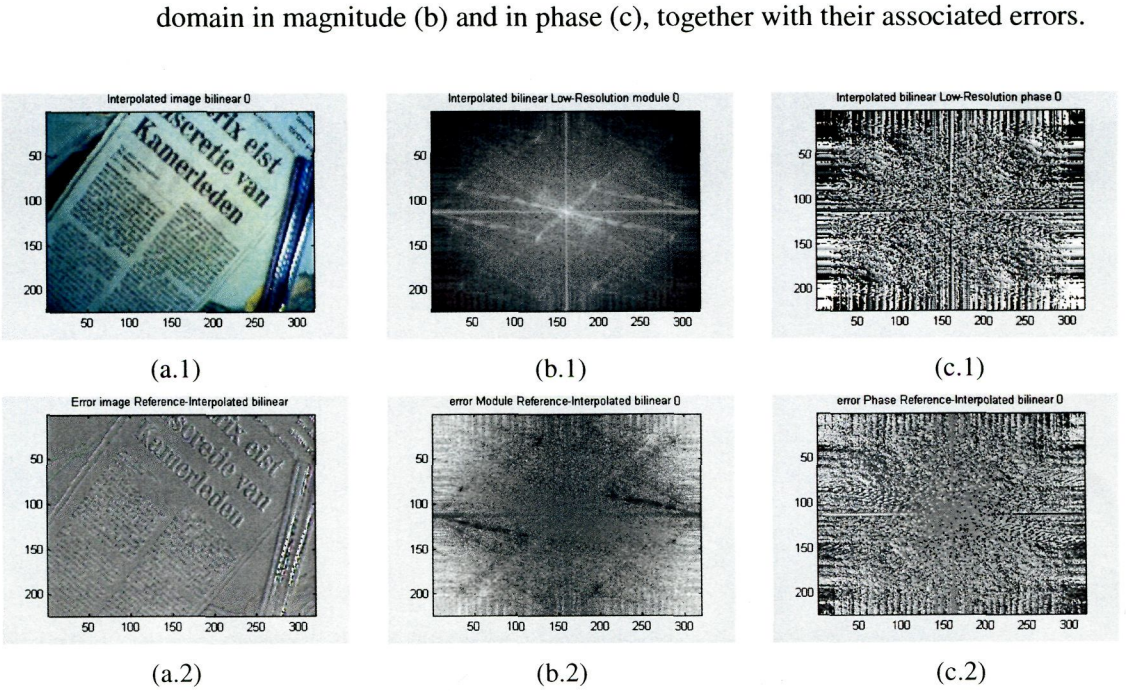


Figure 102. Bilinear interpolated image, in the spatial domain (a), in the frequency domain in magnitude (b) and in phase (c), together with their associated errors

Instead of showing all the frames, we will pay special attention to the process corners, i.e. to the best and the worst cases. The frame with higher PSNR is the frame 10 and the worst is the frame 15. In the Figure 104, is shown the frame 10 in the spatial domain (a), in the frequency domain in magnitude (b) and in phase (c), together with their associated errors (2).

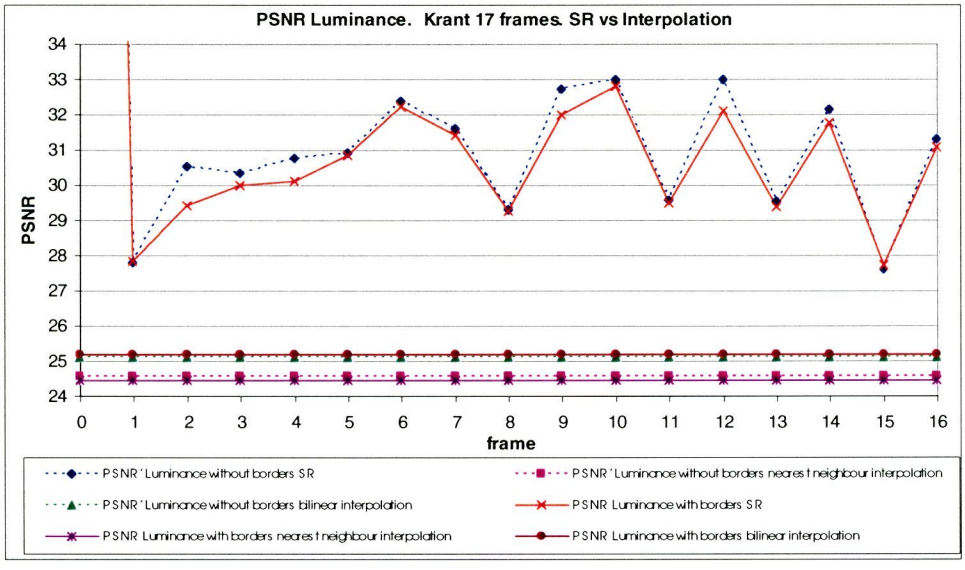
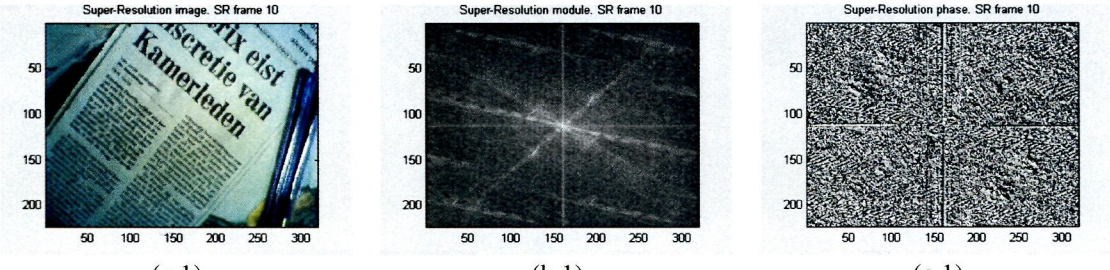
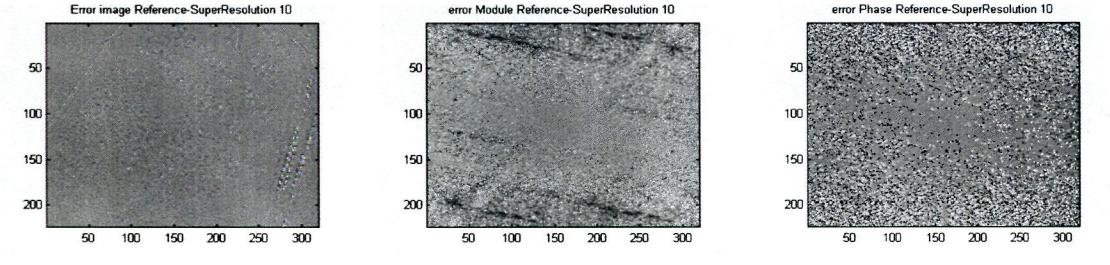


Figure 103. PSNR of the luminance for the Krant sequence with and without borders for the version v2.0 of the SRA with ¼ pixel precision.



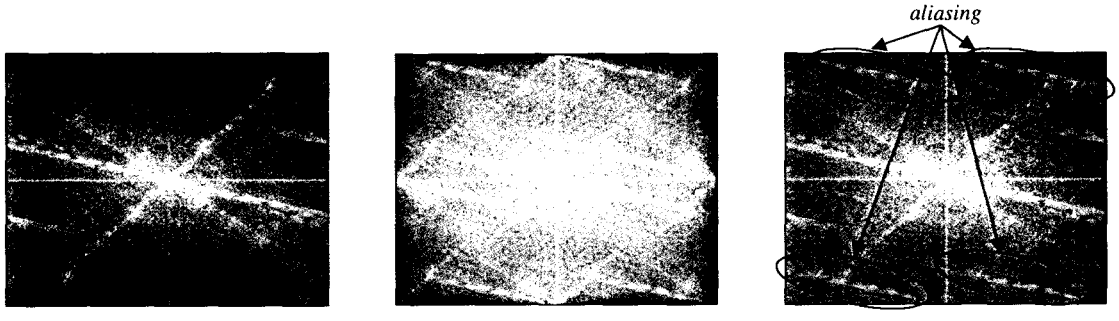
(a.1) (b.1) (c.1)



(a.2) (b.2) (c.2)

Figure 104. Super-resolved frame 10 in the spatial domain (a), in the frequency domain in magnitude (b) and in phase (c) together with their associated errors (2).

Representing together the bi-dimensional Fourier transforms in magnitude shown in Figure 100 (a.2), Figure 100 (b.2) and Figure 104 (b.1) in Figure 105, it is clear how the aliasing amount has considerably decreased in the spectrum of the super-resolution image (c) compared with the aliasing present in the input images (b) and compared both with the original spectrum (a). Nevertheless, already some aliasing remainders are still present, especially in the more energetic diagonal bands in the high frequency zone, as it is denoted in the figure.



(a) (b) (c)

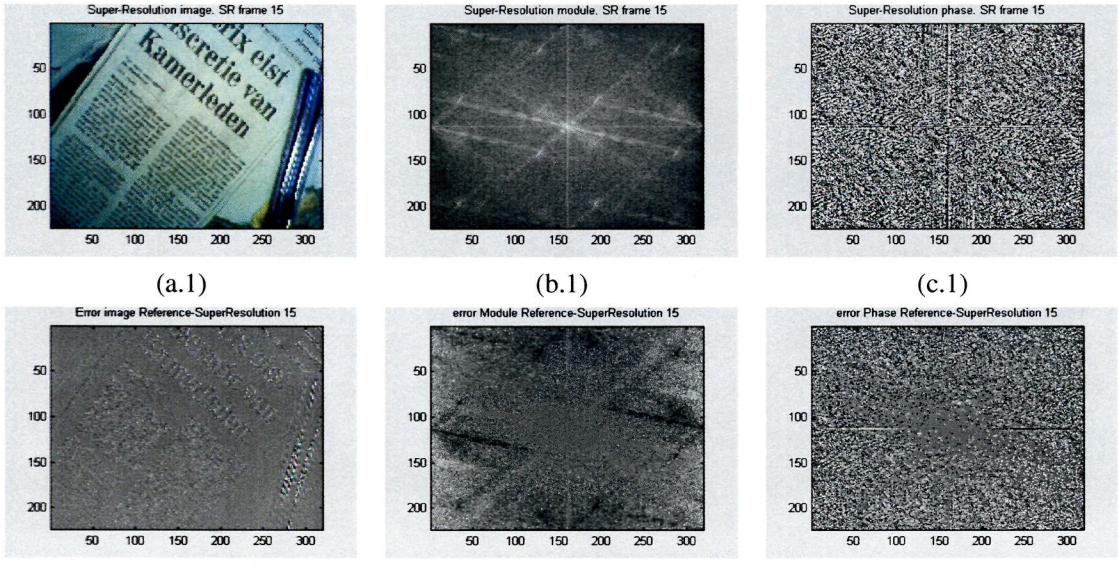
Figure 105. Bi-dimensional Fourier transforms in magnitude of the luminance of original image (a), of the first input image (b) and of the output super-resolved image number 10 (c).

The aliasing is also noticeable in the spatial domain as a quality loose due to the data absence. This is clearly seen in the small details, as the letters, and in the continuity loss at the edges of the pens in the right hand side of the picture. The super-resolution process enables the incorporation of new information coming from other close images, considerable increasing the quality of the resulting image. For instance, in Figure 106 is shown a detailed enlargement of the upper-right corner of the first input image of the KRANT sequence (a) and the same zone improved by means of the super-resolution techniques (b). Notice that before applying the algorithm it was almost impossible to read the word *'nieuw'* (new in Dutch) that appears in the upper-right corner. The edges of the pens and the letters of the newspaper headlines have clearly been improved. The resolution increase is made clear as a decrease in the apparent average size of the pixels.

Paying attention to the worst quality image (frame 15 depicted in Figure 107), we can appreciate that the aliasing amount is higher, although some aliasing has been removed compared with the amount presented in the input images.

(a) (b)

Figure 106. Enlarged detail of the first frame of the sequence KRANT, before (a) and after applying the super-resolution algorithm v2.0 (b).



(a.1) (b.1) (c.1)

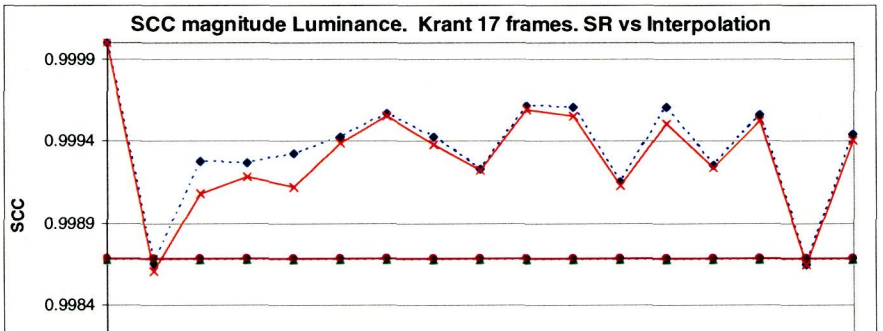
(a.2) (b.2) (c.2)

Figure 107. Super-resolved frame 15 in the spatial domain (a), in the frequency domain in magnitude (b) and in phase (c), together with the associated errors (2).

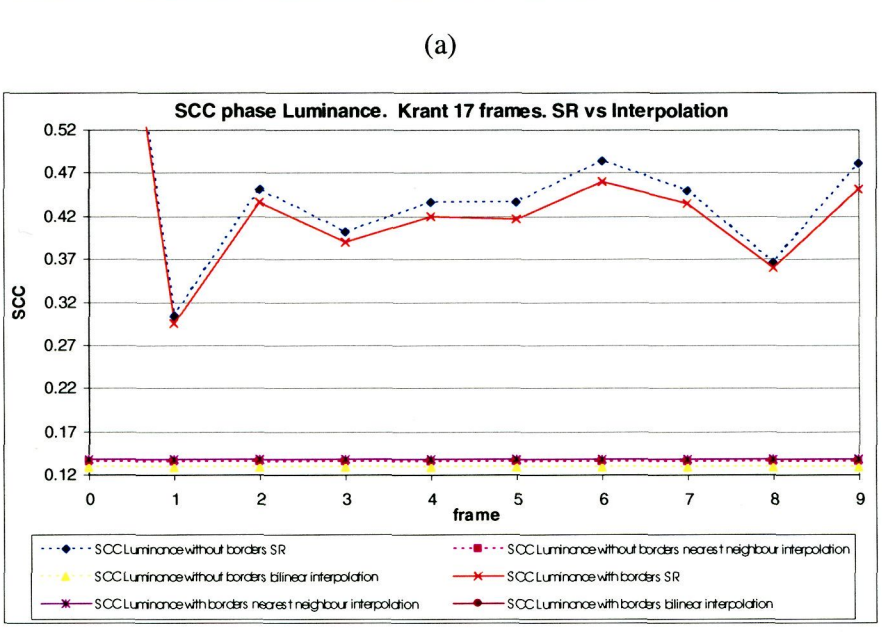
The spectral correlations also exhibit a lower difference between the images with and without borders (Figure 108). The average absolute error in magnitude between both of them is $5.8648 \cdot 10^{-5}$ and in phase is $1.568 \cdot 10^{-3}$. However, although the average error in magnitude is lower, the magnitude PSNR values are closer to the interpolation levels than the phase PSNR values, reaching the interpolation levels in frames 1 and 15, which never occur with the phases. The higher correlation and lower error between images with and without borders in magnitude suggest a better performance of the algorithm in magnitude, what supposes a very

good point inasmuch as the human sight is more sensitive to the magnitude than to the phase of the images. The higher distance of the phase correlation to the interpolation levels is due to the improvement in the motion estimator accuracy that enables a better shift adjust among the input images.

The PSNR and the spectral correlation in magnitude of the chrominances are higher than the luminance (Figure 109), because of the lower entropy of the first ones, but on the other hand, the phase spectral correlation of the luminance is higher than the chrominance



(a)



(b)

Figure 108. Spectral correlation coefficients in magnitude (a) and in phase (b) of the KRANT sequence of 17 frames.

ones. This is logic if we take into account that the motion vectors have been computed only for the luminance, and so, it is reasonable that its phase will result better adjusted to the original phase than the chrominance phases, for which have been used the luminance motion vectors. Also can be appreciated a high similitude between the curves with and without borders, although the merit figures without borders exhibit a slightly higher value than their counterparts with borders.

5.3.3.2 Non-iterative incremental super-resolution algorithm (v2.1)

A detailed analyse of the pseudo-code of the version v2.0 of the super-resolution algorithm in Figure 87, reveals that in fact there is no impediment to increase the number of low-resolution images to be combined. It will be only necessary to modify three subjects:

1. Increase the number of input images in the algorithm loops.
2. Increase the number of bits of the accumulative memory HR_A to properly summate the pixel values of several images.
3. Increase the number of bits of the contributions memory, although due to the lower value of the contributions it is foreseeable that the memory word width will be much lower than in the case of the memories intended to store the images.

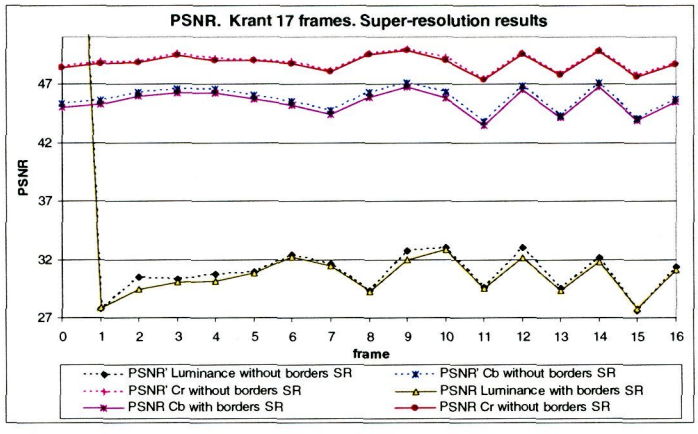
It seems reasonable to suppose that, if the image displacements are randomly distributed, then a higher number of frames combined will result in a higher possibility of disposing new information available, what will rebound in a higher super-resolution image. This assumption has been verified by executing the algorithm over several sequences.

In order to properly size the memory HR_A we have follow the next steps:

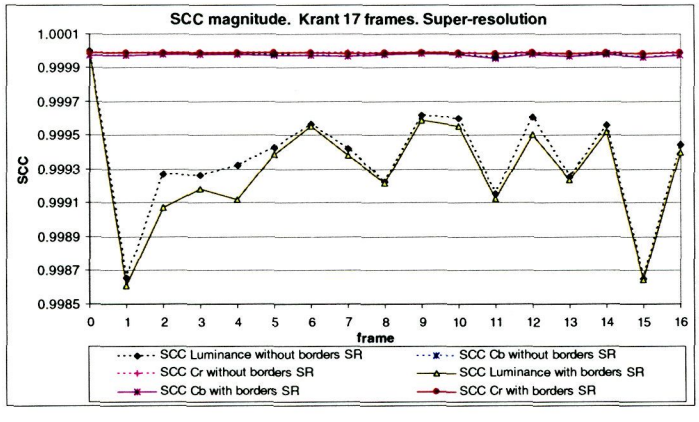
1. The maximum number of frame to accumulate is settled to 12. Then, in the worst case, we must be able to store the number $(2^8-1)\cdot 12 = 3060$, as the input images are 8 bit wide.
2. To store the number 3060 are needed $\log_2(3060) = 11.57$ bits, that are upper-rounded to 12 bits.
3. When performing the rounding in the number of bits, we have a fast increase in the number of images to be combined in HR_A. The larger number that can be stored in 12 bits is $2^{12}-1 = 4095$.
4. In 4095 fits $4095 \div 255 = 16.05$ images, what means that in the worst case we can store

16 input images of 8 bits wide each one.

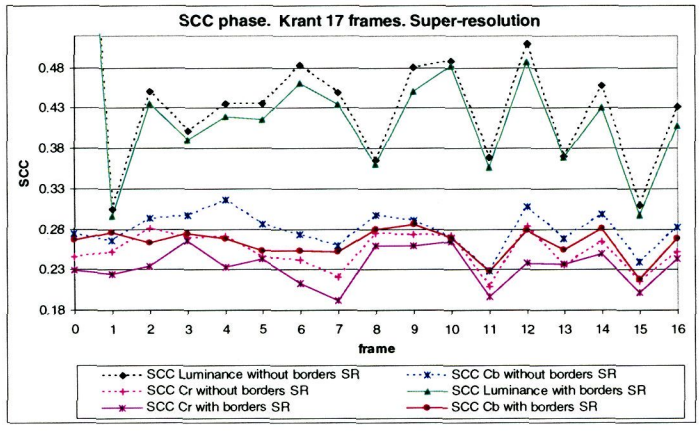
The pseudo-code of the version v2.1 of the super-resolution algorithm is shown in Figure 110, where HR_A is a 12 bits memory and the remainder memories are 8 bits wide.



(a)



(b)



(c)

Figure 109. PSNR (a), spectral correlation coefficients in magnitude (b) and phase (c) of the Krant sequence with borders (solid lines) and without borders (dot lines).

5.3.3.2.1 Simulation results and quality analysis of version v2.1 using ¼ pixel precision

To verify that an increase in the number of combined frames corresponds with a quality increase in the resulting image we have designed the following experiment: A set of 12 displacement vectors have been generated (Table 17), being the first vector the zero vector and the remainder eleven random vectors. The first displacement vector is (0,0) to assure than the resulting image (always adjusted to the first frame of the sequence) will be with zero displacement with respect to the reference, enabling reliable quality measurements.

From this vectors set, the three first vectors are used to compose a super-resolved image based on three low-resolution images starting from the frame 0 of the KRANT sequence. After that, the first four vectors are applied again to the frame 0 of KRANT to generate a new super-resolved image form four input images, and so on until a super-resolved image with 12 low-resolution frames has been generated. In total, have been generated 3+4+5+6+7+8+9+10+11+12=75 low-resolution frames to be used as inputs to the super-resolution algorithm version v2.1. In Figure 111 is shown the followed scheme for the generation of the test incremental sequence. These input 75 frames will generate 10 output frames, whose qualities are shown in Figure 112.

As it was expected, as the number of input frames to combine increases, the PSNR increases until reach 33.44 dB in the frame number 9, which come from the combination of 12 input low-resolution frames. Empirically, can be verified that after combining 6 input frames (output frame number 3), i.e. when the 36.55 dB is reached, the human eye hardly perceive enhancements in the quality of the image. In addition, it can be seen that the largest increment in the quality (greater PSNR slope) takes place in the four first output frames. All of this leads us to the conclusion that a system can be limited to combine 5 or 6 input frames, depending on the available resources and the desired output quality.

It must be taken into account that the maximum quality for the iterative super-resolution algorithms is of 34.5635 dB for the luminance after 73 iterations and for images of type 'a', whereas in this last version this value is reached after combining 6 input frames (34.5698 dB) and it is overcame in more than 3 dB combining some more additional frames.

*** Non iterative super-resolution algorithm v2.1**

Set the value of the improvement: scale
nr_*frames* = scale*scale, M'= scale*M, N'= scale*N
HR_B, HR_S, HR_T, HR_T2, HR_Cont and HR_S2 all of 8 bits and size [M'][N'].
HR_A is 12 bits and size [M'][N'].
LR_I[M][N] for the motion estimation
Store the first *frame* in LR_I_0[M][N] and the remainders in LR_I[M][N]

// First Image is the reference

```
HR_A.lum    = Upsample_Holes(LR_I_0.lum)                    [SR_INIT_A]
HR_A.chrom = Upsample_Neighbours(LR_I_0.chrom)

HR_Cont     = Create_image_contributions                    [SR_INIT_CONT]

FOR fr = 1 .. nr_frames-1
    MV_ref2fr = Calc_Motion_Estimation ( LR_I, LR_I_0)
    IF Global_Motion THEN Select_global_motion_vector()
    MV_ref2fr = 2 .* MV_ref2fr

    HR_S.lum    = Upsample_Holes(LR_I.lum)                  [SR_UPSAMPLE]
    HR_S.chrom = Upsample_ Neighbours(LR_I.chrom)

    HR_S2 = Create_image_contributions                      [SR_INIT_CONT]

    HR_T  = Motion_Compensation(HR_S, MV_ref2fr)            [SR_MOT_COMP]
    HR_T2 = Motion_Compensation(HR_S2, MV_ref2fr)           [SR_MOT_COMP_CONT]

    HR_A    = HR_A + HR_T                                   [SR_ADD]
    HR_Cont = HR_Cont + HR_T2                               [SR_ADD_CONT]
END FOR

HR_A = 4*HR_A /HR_Cont                                      [SR_ADJUST_A]

If (HR_Cont(i,j)==0) THEN HR_B = Interpolate(HR_A(i,j))     [SR_UPDATE]
ELSE  HR_B = HR_A

Clip(HR_B, 0, 255)
```

High Resolution result image in HR_B

Figure 110. Pseudo-code of the non-iterative algorithm version v2.1.

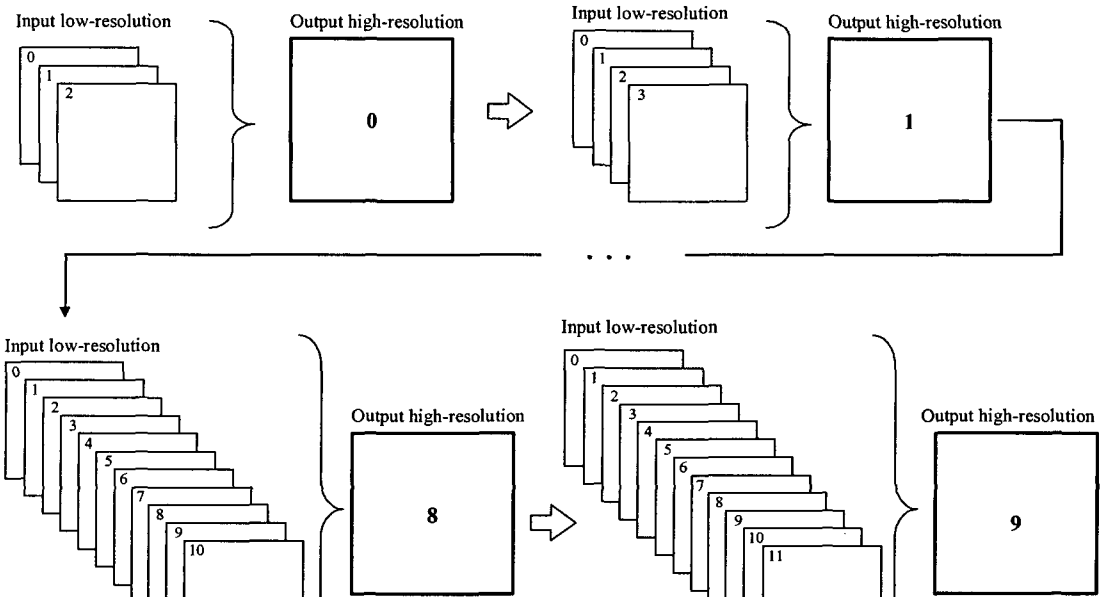


Figure 111. Scheme followed in the generation of the incremental test sequence.

Input frame	Vectors in VHR	Vectors in HR	Input frame	Vectors in VHR	Vectors in HR
0	0 , 0	0 , 0	6	0 , 1	0 , 0.5
1	2 , 2	1 , 1	7	-2 , 1	-1 , 0.5
2	3 , 1	1.5 , 0.5	8	1 , 0	0.5 , 0
3	3 , 3	1.5 , 1.5	9	0 , 2	0 , 1
4	2 , 0	1 , 0	10	2 , 0	1 , 0
5	0 , 3	0 , 1.5	11	0 , -2	0 , -1

Table 17. Displacement vectors pseudo-randomly generated to incrementally reconstruct the super-resolution image.

Regarding to the bilinear interpolation level, we are all the time above it: the minimum distance is 2.51 dB and the maximum distance is 11.37 dB.

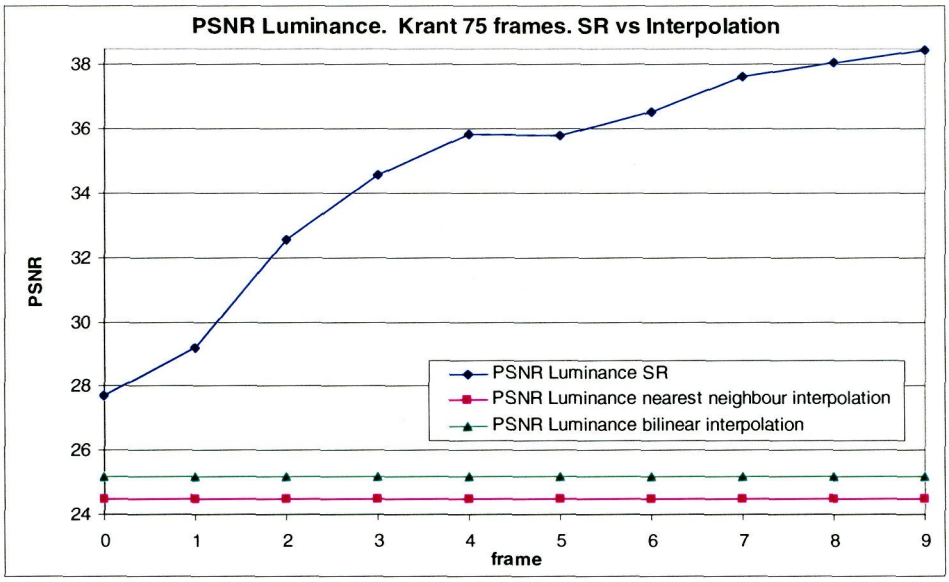


Figure 112. PSNR of the KRANT sequence with 10 incremental output frames.

In Figure 113 can be appreciated a slight increment in the PSNR, produced when the image borders are removed. Nevertheless, it must be taken into account that a reduction in the number of pixels of the image produces a small increase in the PSNR. In this case, the average increase of 1.029 dB is due to this fact, and not because differences in the borders. This hypothesis is verified through an ocular inspection of the error image (Figure 114) of frame 9. The PSNR of the image without borders of this frame is 1.32 dB higher than the image with borders, but in the image borders, no discontinuity can be observed. Therefore, we

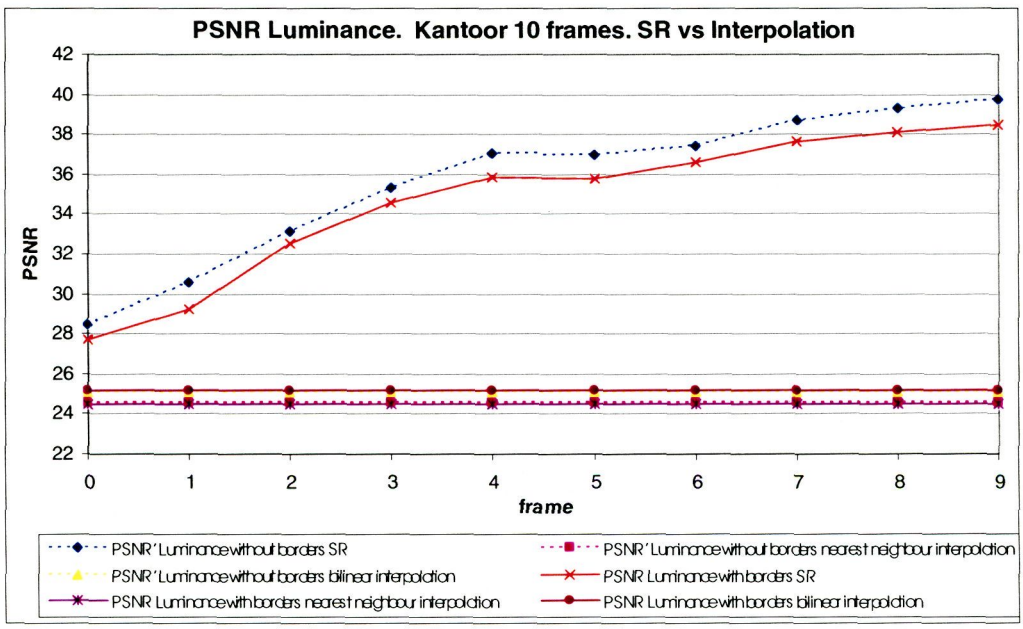
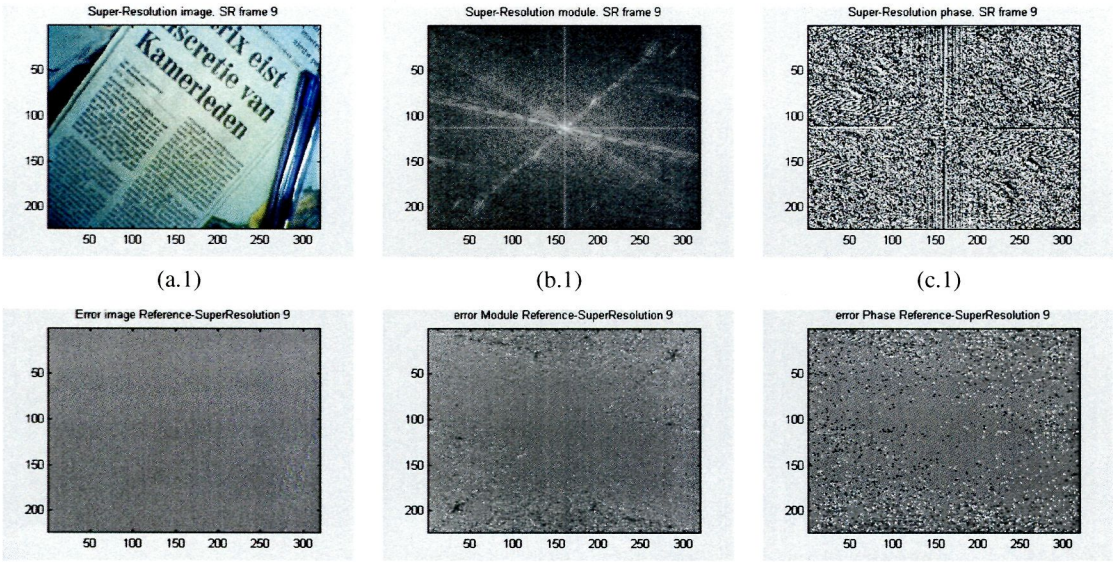


Figure 113. PSNR of the KRANT sequence with and without borders.

can conclude that the PSNR increase of these images is due to the reduction in the number of pixels. This effect has been taken into account all over this work.

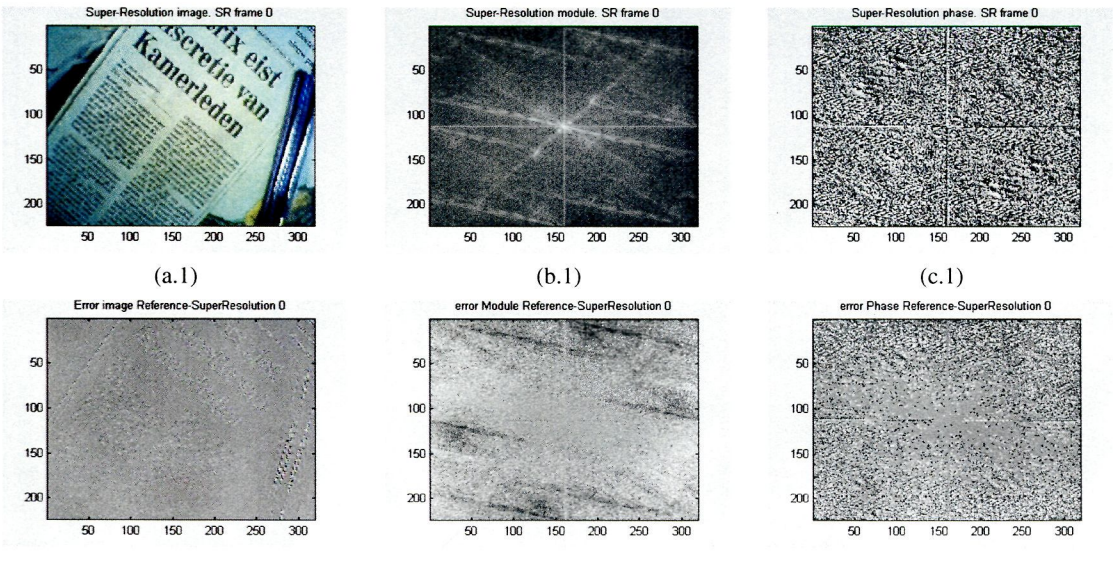
In Figure 114 can be appreciated that the image error in the space domain (a.2) is highly uniform, what means a very low error with respect to the reference image. The bi-dimensional Fourier transform in magnitude shows an almost complete removal of the aliasing, exhibiting a minimal image error in the high-power spectral bands (b.2) that it is approximately the same trend that follows the minimal spectral error in phase (c.2).



(a.1) (b.1) (c.1)

(a.2) (b.2) (c.2)

Figure 114. Super-resolved frame 9 in the spatial domain (a), in the frequency domain in magnitude (b) and in phase (c), together with their associated errors (2).



(a.1) (b.1) (c.1)

(a.2) (b.2) (c.2)

Figure 115. Super-resolved frame 0 in the spatial domain (a), in the frequency domain in magnitude (b) and in phase (c), together with their associated errors (2).

For completeness, it is also shown the worst image of the sequence (frame 0) coming from combining three low-resolution frames (Figure 115). It is clear the higher amount of aliasing and consequently higher errors in all the cases.

The chrominances PSNR are always above the interpolation levels, although they do not follow the same luminance evolution. In Figure 116 is shown the PSNR of the red chrominance, with higher energy that the blue one, which maximum and minimum differences with respect to the bilinear interpolation level are of 0.982 dB and 2.27 dB respectively.

In Figure 116 also can be appreciated the relative independence of the image quality with respect to the inclusion of the image borders. As it happen with the luminance, the small increment is justify through the decrease in the number of pixels.

In Figure 117 can be seen much higher values of the chrominance PSNR due to their lower entropy. The red chrominance has an average value of 49.6631 dB, versus the average value of 46.9091 of the blue chrominance and the average value of 34.6331 of the luminance.

In Figure 119 can be seen how the correlations with borders (solid lines) and without borders (dotted lines) are very similar (the average correlation coefficient is magnitude with

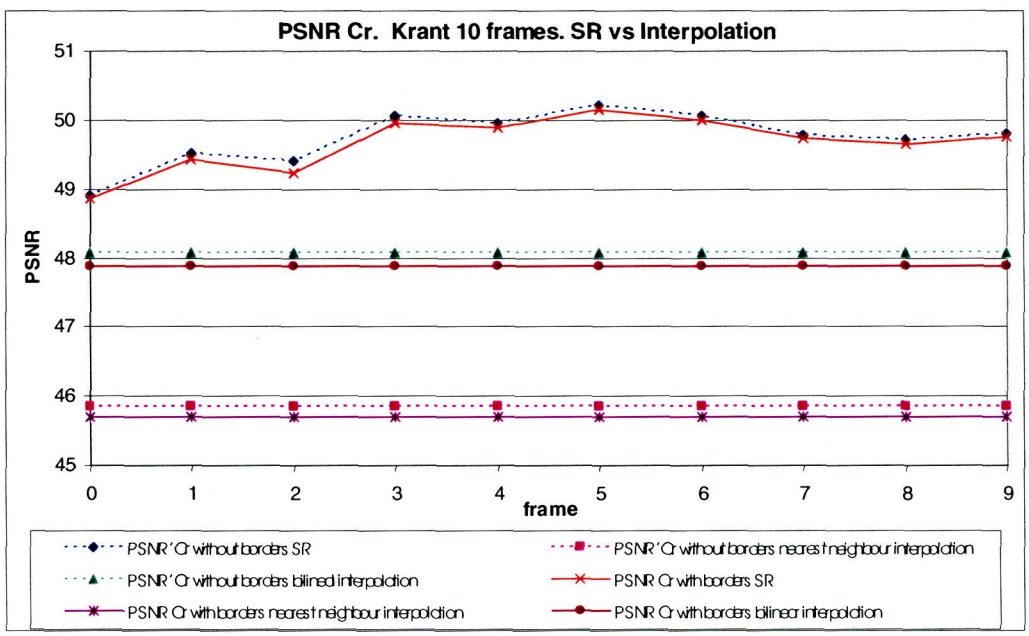


Figure 116. Red chrominance of the PSNR of the super-resolved and the interpolated images for the KRANT sequence with 10 incremental output frames with and without borders.

borders is 0.999600396, whereas without borders is 0.999669774 and in phase with borders is 0.624228535 and without borders 0.637115433), and like was established, the differences can be attributed to the different sizes of the images. As in the previous version, from where this version comes from, the magnitude correlations (a) are substantially higher than then the luminance correlations, due to the lower entropy of the formers. The spectral correlation in phase of the luminance (b) is higher than the chrominance ones, for having a better adjusted movement. It can also be confirmed that the chrominance metrics does not follow the luminance trends, and that the luminance follows the same increasing trend, both in module and in phase.

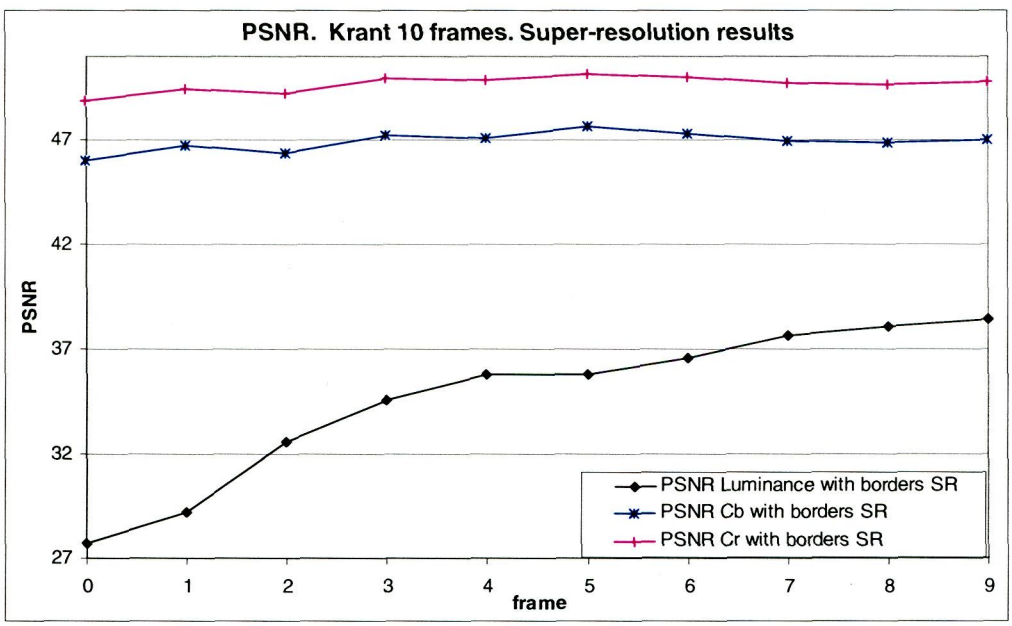


Figure 117. PSNR of the luminance and chrominances for the Krant sequence with 10 incremental output frames with borders.

(a) (b)

Figure 118. Enlarged detail of the original image (a) and the super-resolution image (b) for the KRANT sequence of 10 frames.

Table 18 gathers a summary of the average values of the different metrics along the whole output sequence. It can be seen how the differences between the images with and without borders have decreased, and how the PSNR and the magnitude correlation of the luminance are lower than the chrominance ones, although in phase this trend is inverted, being higher the luminance ones than the chrominance ones.

PSNR		
	With borders	Without borders
Luminance	34.63 dB	35.66 dB
Blue chrominance	46.90 dB	47.24 dB
Red chrominance	49.66 dB	49.73 dB
SCC magnitude		
	With borders	Without borders
Luminance	0.99960	0.99966
Blue chrominance	0.99998	0.99998
Red chrominance	0.99999	0.99999
SCC phase		
	With borders	Without borders
Luminance	0.62422	0.63711
Blue chrominance	0.29255	0.31720
Red chrominance	0.26746	0.28298

Table 18. Average values of the PSNR, the SCC in magnitude and phase of the KRANT sequence of 10 output frames with and without borders.

Figure 118 shows in detail the upper-right corner of the image Krant, before (a) and after the super-resolution process (b). Is clear the quality improvement of the image, where is possible to read the letters over the newspaper headlines.

This kind of system is very well suitable for taking digital pictures (digital photography) as they are still images. When the user wants to capture an image, he or she presses the actuator that takes the first reference image, and automatically and continually the system takes 'n' images more ('n' will depend on the desired final quality and the available resources) to combine all of them in a higher resolution image. An image taken with a real 2 mega-pixel sensor will be processed to result as an image taken with a virtual 8 mega-pixel

sensor. The images must contain some movement among them, as can be the case of hand-held cameras.

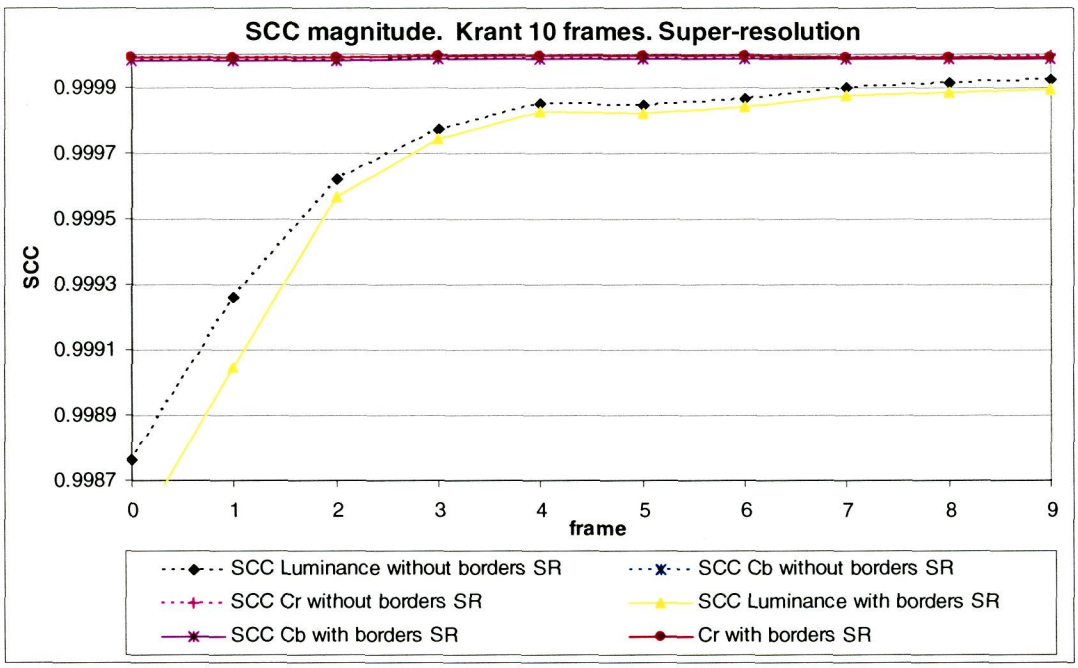
5.3.3.2.2 Block diagram and memory requirements for version v2.1

In Figure 120 is shown the block-diagram of the data-flow for version v2-1 of the SRA shown in Figure 110. As in the previous version v2.0, the block-diagram has been divided, by one side in the zone occupied in the image processing, that makes use of memories HR_A, HR_T, HR_S, LR_I_0 and LR_I, besides of storing the motion vectors in MV_ref2fr, and by the other side, in the zone occupied in contributions processing, that makes use of memories HR_S2, HR_T2 and HR_Cont. In order to clarify the relations among them, there have been drown in solid lines the image flow, in dotted-lines the contribution flow and in dashed-lines the motion vector flow. Moreover, the functions '*upsample*' and '*motion compensation*' have been marked with and asterisk to point out their different performing when they are the in super-resolution mode. With respect to version v2.0 two low-resolution memories have disappeared and now it is only necessary to store one motion vector, aside from the HR_A wide increasing to 12 bits.

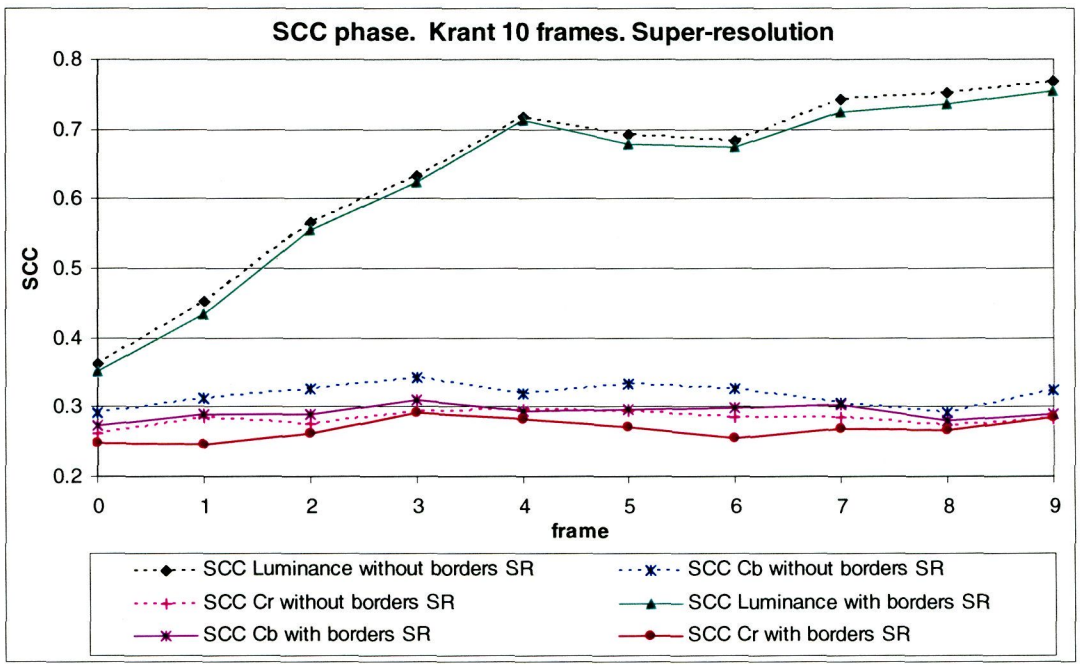
In Table 19 it is shown a summary with the memory requirements in version v2.1 of the SRA, as expressed if Figure 120.

Memories HR_T and HR_T2 are not included in these figures as they are used in this algorithm version to avoid the data overlap when performing the motion compensation, but this problem will not exist in the hardware version over Picasso, because there exist a 3 macro-block stripes buffer that avoids that problem. These 3 stripes have been included in the memory requirement.

In Table 20 the total memory requirements for the most common image sizes are shown. In Figure 121 the data of Table 19 in Kbytes together with the data of Table 15 for the version v2.0 are graphically shown. The memory requirements for version v2.1 are slightly lower than the ones for version v2.0 because two of the low-resolution memories are not needed and because the same memory for the motion vectors is always reused. In any case, the memory reduction is very slight and not significant against the total memory amounts required.



(a)



(b)

Figure 119. Spectral correlations in magnitude (a) in phase (b) of the luminance and the chrominances for the KRANT sequence with 10 incremental output frames.

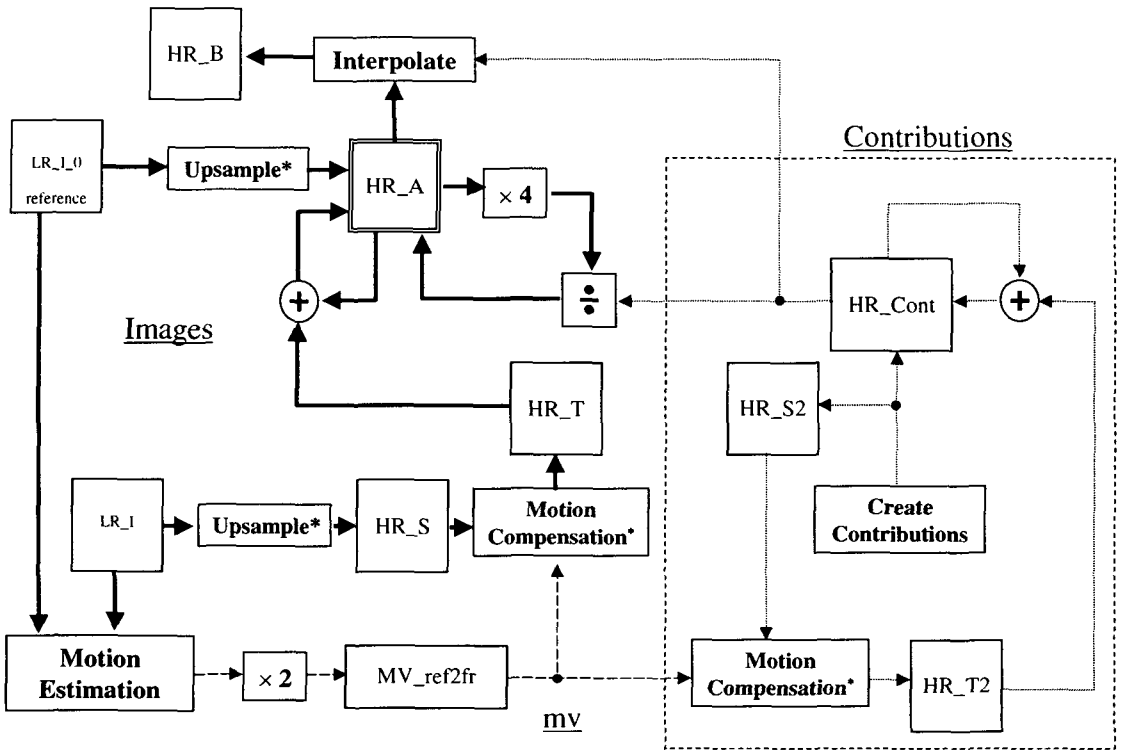


Figure 120. Block-diagram of the super-resolution algorithm version v2.1.

Label	Memory		
	Luminance (bits)	Chrominance (bits)	Total (bits)
HR_A	(2·mb_x·2·mb_y·16·16·12)	(2·mb_x·2·mb_y·8·8·2·12)	18,432·mb_x·mb_y
HR_B	(2·mb_x·2·mb_y·16·16·8)	(2·mb_x·2·mb_y·8·8·2·8)	12,288·mb_x·mb_y
HR_S	(2·mb_x·2·mb_y·16·16·8)	(2·mb_x·2·mb_y·8·8·2·8)	12,288·mb_x·mb_y
HR_S2	(2·mb_x·2·mb_y·16·16·8)	(2·mb_x·2·mb_y·8·8·2·8)	12,288·mb_x·mb_y
HR_Cont	(2·mb_x·2·mb_y·16·16·8)	(2·mb_x·2·mb_y·8·8·2·8)	12,288·mb_x·mb_y
3 Stripes HR	(2·3·2·mb_y·16·16·8)	(2·3·2·mb_y·8·8·2·8)	36,864·mb_y
LR_I[0]	(mb_x·mb_y·16·16·8)	(mb_x·mb_y·8·8·2·8)	3,072·mb_x·mb_y
LR_I[1]	(mb_x·mb_y·16·16·8)	(mb_x·mb_y·8·8·2·8)	3,072·mb_x·mb_y
MV_mem[0]	(mb_x·mb_y·8)	0	8· mb_x·mb_y
Total (bits)	mb_y · (49,160· mb_x + 24,576)	mb_y · (24,576· mb_x + 12,288)	mb_y · (73.736·mb_x+ 36,864)

Table 19. Summary of the memory used by version v2.1 of the super-resolution algorithm.

Size	mb_x	mb_y	Memory (Kbytes)	Memory (Mbytes)
SQCIF	8	6	459.05	0.45
QAVGA	9	7	598.56	0.58
QCIF	11	9	931.60	0.91
HAVGA	18	14	2,331.25	2.28
CIF	22	18	3,645.39	3.56
AVGA	36	28	9,198.98	8.98
VGA	40	30	10,936.17	10.68
4CIF	44	36	14,419.55	14.08
16CIF	88	72	57,354.19	56.01

Table 20. Memory amount used by version v2.1 of the super-resolution algorithm for the most common image sizes used.

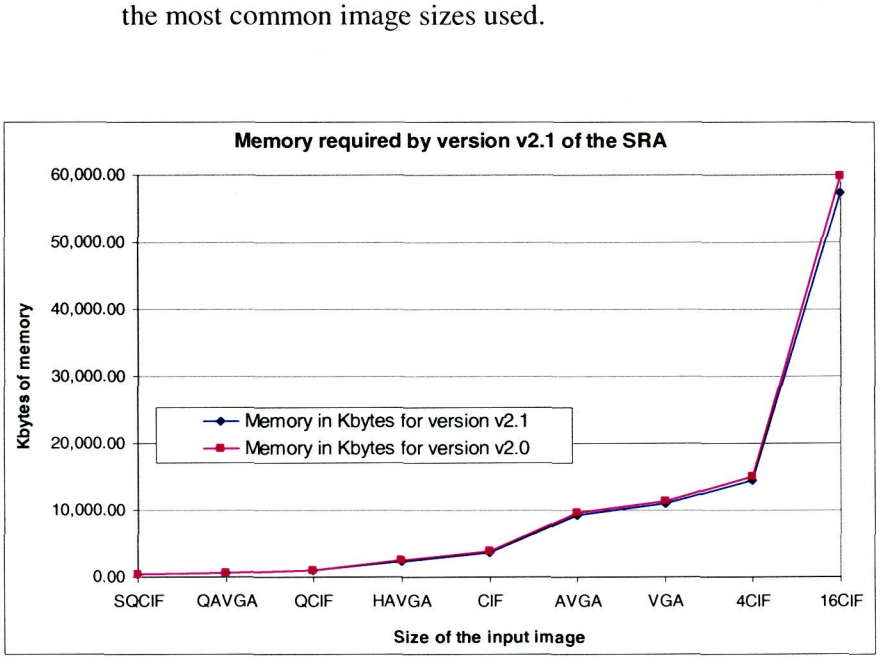


Figure 121. Memory used by the super-resolution algorithms v2.1 and version v2.0 for the most common image sizes.

5.3.4 Algorithms for video sequences

In this section the last pair of super-resolution algorithms is presented. They come from the modification of the previous versions for being able to work in a video framework. This means that they accept as an input a video sequence and as an output a same length video sequence. These last versions address the following aspects:

- The video movement must be compensated to the last input frame, seeking to follow the scene movement.
- We must find a mechanism that allows incorporating the information available in the new frames as they are being read.
- Likewise, it must be searched a mechanism that allows the algorithm to recover in the presence of context changes and/or fast movements in the scene that produce occlusions and objects out of focus.
- The memory amount must be further reduced, with the aim of fitting it in the on-chip memory and therefore reduce cost and consumption.

In Figure 122 (a) the strategy followed until now by generating the first image using the motion vector (0,0) and then adjust the remainder frames to the first frame or reference is shown. However, in Figure 122 (b) the strategy followed in the algorithm for video, where the movement is adjusted frame by frame, is shown.

5.3.4.1 Basic video algorithm description (v3.0)

In this part, the first algorithms for video sequences are developed. The main features of this version are:

- The motion estimation is performed in low-resolution.
- The image is reconstructed based on the contributions information, what is especially useful in the borders reconstruction.
- A mechanism has been incorporated to make decisions based on the available information, seeking to obtain always the higher possible quality.
- The super-resolved image is stored with zeroes (holes) where such information is not available, and the holes are interpolated in the output stage.

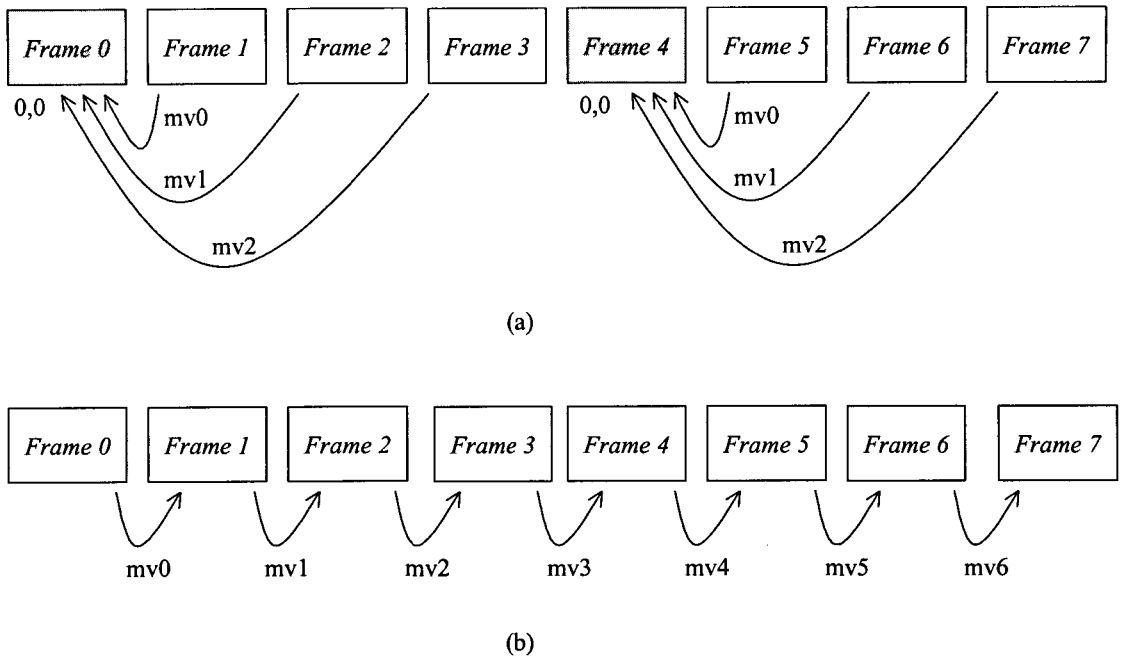


Figure 122. Motion estimation strategies for still-image (a) and for video (b).

5.3.4.1.1 Robustness increase of the super-resolution algorithms

The super-resolution algorithms assume that the frames of the sequence are correlated among them, with small movements among images that enable to incorporate new information to the high-resolution image. Nevertheless, when applying these algorithms to real sequences, it must be taken into account the following limitations:

- An object inside the image can move very fast, producing the occlusion of other objects of the scene. That will preclude the addition of new information about those later objects. It is also impossible to incorporate new information to the fast object, as very large motion vectors will produce undesirable artefacts when trying to compensate its movement at the block level.
- The fast movements of the camera produce changes among not correlated scenes. This is known as 'context changes' or 'scene changes', and in this conditions no information can be inferred from the previous image.

The only solution in these cases, in presence of information lack, is to interpolate those elements that can not been improved through super-resolution techniques. In this sense, we will make use of the information provided by the motion estimator to the compression

software to help it the task of deciding the type of compression for every macro-block. The available decision items are the following:

- **Motion vectors**. A very large motion vector, compared with the image size, is indication of large local image displacements.
- **SAD inter**. It is the summation of the absolute differences between the present image and the previous one, after the motion compensation. A value per macro-block is given, depending on the motion vector. Very high values of this parameter suggest that the macro-block has low correlation with the corresponding macro-block in the previous image. Depending on the used motion vectors we can perform a sub-classification in:
  - **Global SAD inter.** It is the SAD obtained when the global motion vector is used.
  - **Local SAD inter.** It is the SAD obtained when local motion vectors are used.
- **SAD intra.** It is a measure of the variations inside the own image, being the SAD computed for each macro-block as the absolute differences between every macro-block pixel and the macro-block average value. Very high values of this parameter suggest zones rich in details and, therefore, in high frequencies. A high SAD intra value of a macro-block assures a more reliable motion vector for that macro-block.
- **SAD SR**. It is the summation of the absolute differences between the previous super-resolved image and the present one, but only in the low-resolution pixel positions i.e. only in the upper-left pixel of every four pixels in the basic 2-by-2 cell. It is a measure between super-resolved images.

Figure 123 schematize the various inputs that help the system to make a decision about the action to be taken for each macro-block:

- Apply super-resolution using local vectors.
- Apply super-resolution using the global vector.
- Interpolate the present macro-block.

From the motion vectors, a new decision parameter is incorporated, and it is called 'mv_sad', obtained as the absolute differences of every motion vector with the global motion vector. If the 'mv-sad' is low, that means that the image is dominated by global movement, while if it is high, that will mean that the image is dominated by local movements.

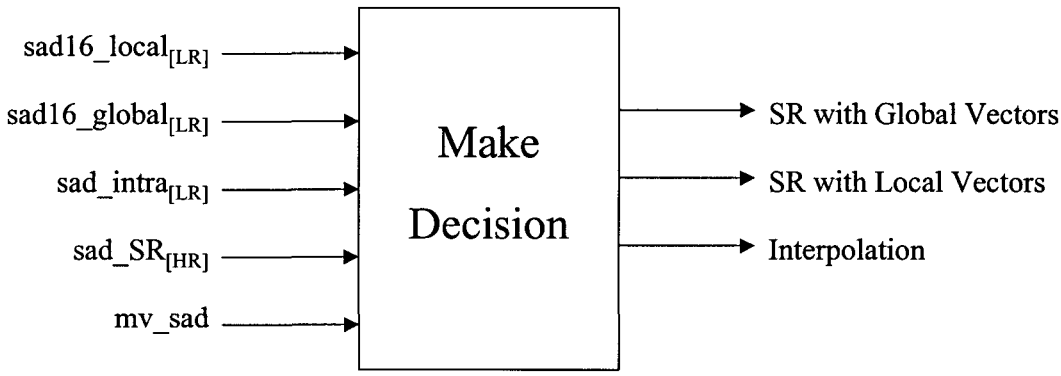


Figure 123. Input-output scheme for making decisions about the super-resolution mode.

All these parameters are obtained from low-resolution images, what speeds-up their computation. Performing a study over several sequences it has been determined a set of thresholds that presents good quality results for the used test sequences. The algorithm to make decisions is the one shown in Figure 124.

The algorithm starts by obtaining the values of the 'sad_local' (sad16_local) and the 'sad_intra', together with the local motion vectors. All of them are outputs from the motion estimator. From that local motion vectors, the global motion vector can be computed, either as the average of all of them, as the selection of the most frequent pair or as the selection of the horizontal and vertical components separately. The variable 'local_penalty' is initialized to a threshold value `LOCAL_PENALTY_THRESHOLD`. This variable seeks to favour or penalize the global movement against the local movement or vice versa. Once the global vector is kwon, it can be computed the 'mv_sad' as the summation of differences between every local vector and the global vector. Moreover, by passing the global vector to the motion estimator, we can get the SAD associated to the global vector as the 'sad_global'.

If the 'sad_mv' is above the threshold `MV_SAD_THRESHOLD`, then the global motion must be favoured by incrementing the value of the variable 'local_penalty'. In the opposite case we will decrement it to favour the local movement. Later on, the 'sad_local' is updated with the value of the 'local_penalty' to be compared with the 'sad_global'. If the 'sad_global' is lower than the modified 'local_sad' then we will chose the global motion vector, else we will choose the local motion vector. Once we have discriminated between local and global movement, we have to discriminate between applying super-resolution with the chosen motion vector or perform an interpolation. For that purpose, some thresholds have been established in the following manner: If it is the first frame of a scene, we will always

interpolate. If it is not the first frame of a sequence, we have to find out if we are in a context change situation. In that sense, we apply the empiric equation (30).

$$context\_change = 32 \cdot \frac{sad\_intra - sad16\_local}{sad\_intra + sad16\_local} \tag{30}$$

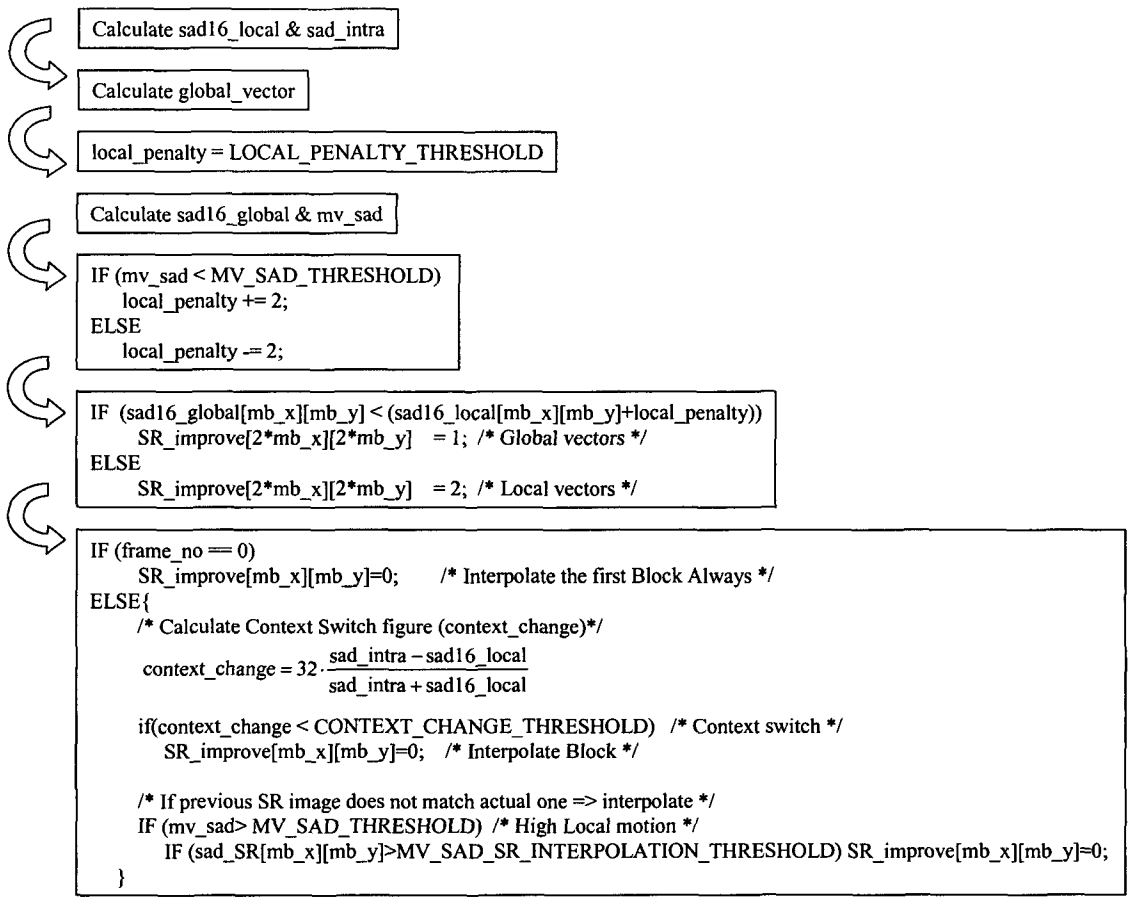


Figure 124. Scheme followed for making decisions about the super-resolution mode in the super-resolution algorithm.

Equation (30), that reflects the scale differences between the 'sad_intra' and the 'sad_inter', has demonstrated to be a quite stable estimator to determine the context change, as will be later seen. If the value given by the equation (30) is below the threshold named CONTEXT_CHANGE_THRESHOLD, then we will interpolate. Finally, we estimate if there exist much difference between the present and the previous image, in the sense that interpolation will result in a better choice that super-resolution. We will interpolate if there exist a great

deal of local movement in the macro-block ('mv_sad' greater than the threshold MV_SAD_THRESHOLD) or if besides the present super-resolution image exhibits large differences compared with the input image ('sad_SR' greater than the threshold MV_SAD_SR_INTERPOLATION_THRESHOLD).

These thresholds have been determined in an empirical way from the detailed study of fourteen image sequences, and offer good quality in all the tested conditions. This methodology increases even more the algorithm robustness, assuring that in no case we will drop bellow the interpolation level.

In order to determine the thresholds we have performed a study of the values of the parameters used as evaluators in different conditions. The most drastic case is the context change, and in that sense it has been prepared a synthetic video sequence based on the KRANT sequence. The first frame is taken and it is applied eight random vectors, generating eight shifted frames. Next, we jump to the frame number 15, quite different, and it is again applied eight new random shift vectors (Table 21). The result is a new sequence of 16 frames with random shifts and an abrupt context change in the frame number 8 (starting the count from zero). In Figure 125 are shown the frames 0 and 15 of the KRANT sequence.

Original frame	Sequence number	Shift vector	Original frame	Sequence number	Shift vector
0	0	3 -3	15	8	3 -3
0	1	1 -2	15	9	-2 -2
0	2	0 1	15	10	0 0
0	3	-1 -2	15	11	3 2
0	4	3 -3	15	12	1 -1
0	5	-2 0	15	13	0 -2
0	6	-2 3	15	14	1 -2
0	7	1 0	15	15	0 2

Table 21. Shift vectors applied to frames 0 and 15 of KRANT.

If the values of equation (30) are graphically represented for this sequence, it is obtained the graph of Figure 126. The frame where the context change takes place is clearly identified with values below 5. Consequently, this will be the value assigned to the threshold CONTEXT_CHANGE_THRESHOLD.

Notice however that, not only the frame where the context change takes place is below the threshold. There exist some other few macro-blocks that are also below the threshold and therefore will be interpolated.

To determine the value of the threshold `MV_SAD_THRESHOLD`, we have to represent the values of 'mv_sad', what it is depicted in Figure 127. In this case, also a value of 5 is obtained for the threshold `MV_SAD_THRESHOLD`.

(a) (b)

Figure 125. Frame 0 (a) and frame 15 (b) of the KRANT sequence.

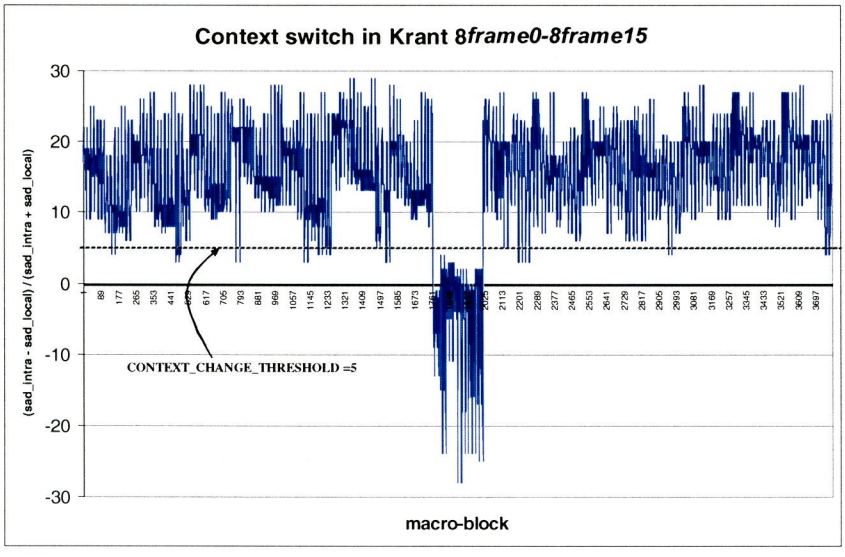


Figure 126. Determination of the threshold `CONTEXT_CHANGE_THRESHOLD`.

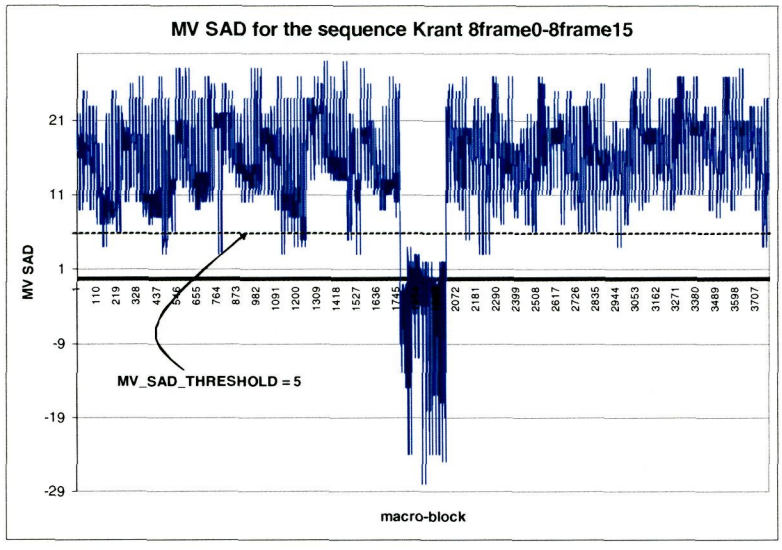


Figure 127. Determination of the threshold MV_SAD_THRESHOLD.

MV_SAD_SR_INTERPOLATION_THRESHOLD is the third threshold to be determined and in that sense it is necessary to observe the behavioural of the 'sad_SR', which is reflected in Figure 128. In this case, the greater values are produced, as it was expected, in the context change. A threshold value of 3000 assures at least a right interpolation in a context change situation, although some macro-block with large errors also crosses that threshold.

It must be taken into account that the 'sad_SR' is only computed for the super-resolution pixels, and therefore the remaining pixels are zero. This characteristic must be

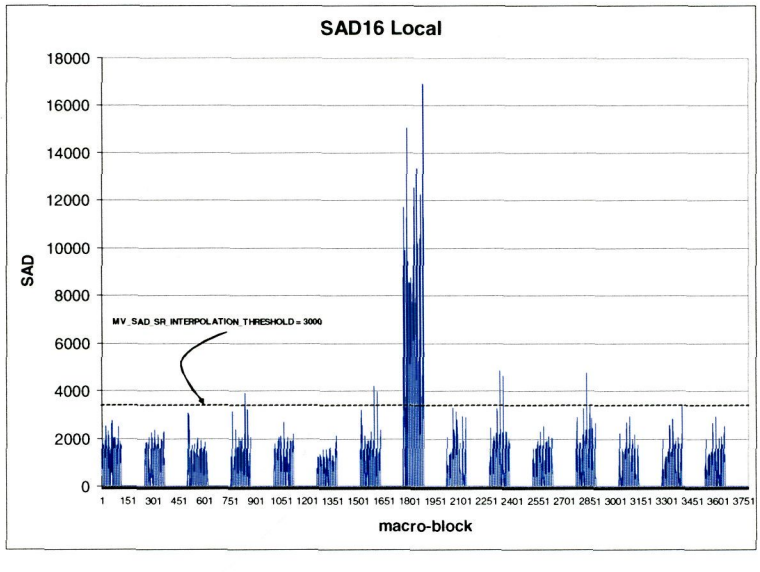


Figure 128. Determination of the threshold MV_SAD_SR_INTERPOLATION_THRESHOLD.

taken into account in order to avoid the comparison in the macro-blocks where the 'sad_SR' is zero.

The value of the threshold LOCAL_PENALTY_THRESHOLD has been settled to the same value as its increment, to duplicate or remove its weight as desired. Finally, Table 22 gathers the value of the thresholds determined for the set of sequences used for test.

Threshold	Value
MV_SAD_SR_INTERPOLATION_THRESHOLD	3000
MV_SAD_THRESHOLD	5
CONTEXT_CHANGE_THRESHOLD	5

Table 22. Thresholds used in version v3.0 of the super-resolution algorithm.

5.3.4.1.2 Pseudo-code of the algorithm version v3.0

In order to decrease the memory used by the algorithm we have opted for using a feedback scheme, where the last super-resolution image is reused to increase the resolution of the present image. In the whole scheme proposed, the new low-resolution image is interpolated to high-resolution, leaving holes in the place of the pixels added to increase the size. The pixels from the low-resolution image are considered valid and are all kept. The algorithm tries to fill the generated holes with the information of the previous super-resolution image, applying a similitude criterion per macro-blocks between the previous low-resolution image and the new obtained image. If the criterion is not satisfied, the holes will be interpolated with the surrounding information.

In Figure 131 and Figure 132, version v3.0 of the super-resolution algorithm is shown. Although the final super-resolved image is still stored in HR_B, what have been preserved throughout all the algorithm versions, in this case it has been necessary to define a new memory HR_B2, where will be stored the super-resolution image with holes that will be fed-back. Although it is necessary to fill the holes prior to display the image, it is necessary to keep such holes inside the algorithm loop (feeding them back), because in that way it is possible to fill the missing information without mixing it with spoiled values coming from the interpolation of the neighbour pixels. The contributions memory HR_Cont is mapped in this

version onto a high-resolution memory for convenience, but it will really store values that can be obtained at the block level for every macro-block. Therefore, neither is necessary to keep their values between images nor to have the size of a full image. Just keeping a small temporal memory of a macro-block size (64×6×8 in bits) will be enough.

The initialising phase only interpolates the first input image, as far as in the beginning there is no additional information to add. However, in the encoder remains stored all the generated holes i.e. the holes are interpolated only to be displayed. The memory HR_W stores the same values as HR_Cont, but only for temporal purposes. The relation between HR_W and HR_Cont is the same that the one between HR_B and HR_B2, but for the contributions.

To increase the convergence in the motion estimator, the first step is to perform a filtering of the low-resolution input images, which will be used to compute the motion vectors. The previous low-resolution image, always filtered, is stored in LR_P. Although in the part of the algorithm shown in Figure 131 appears a function ***Evaluate*** that compute the summation of absolute differences, in fact these values are directly provided by the motion estimator (that is why it was written in cursive) and not constitute a new function to be implemented in the encoder. The tasks that must be executed in software are the selection of the global vector and the 'mv_sad' computation per macro-block. Concretely, the 'mv_sad' per macro-block is computed as the summation of the absolute value of the differences of their components in both directions, as reflected in equation (31).

$$\begin{aligned} mv\_sad[mb\_x][mb\_y] = {} & |mv\_local.x[mb\_x][mb\_y] - mv\_global.x| + \\ & |mv\_local.y[mb\_x][mb\_y] - mv\_global.y| \end{aligned} \tag{31}$$

To compute the 'sad_SR' it necessary to previously compute and motion compensate HR_B towards the previous image, as HR_B contains the previous super-resolved image without holes.

The 'sad-SR' is computed as the summation of the absolute values of the differences between the present and the previous super-resolution image in the low-resolution pixel positions for every macro-block.

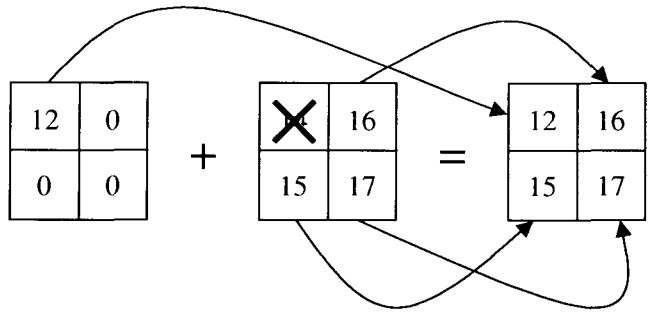


Figure 129. Addition performed as a pixel replacement.

In temporal memories HR_T and HR_T2 are stored the compensated images that are used to avoid overwriting data when performing the motion compensation. These two memories will not be necessary in the final hardware version as the integrated loop memory management avoids memory overlaps. It is necessary to remind that the motion compensator has been modified for super-resolution, injecting zeroes in the borders instead of replicating them when the motion vector points out of the image boundaries. That is why we have called it **Motion_Compensation°**.

The instructions SR_ADD_CONT and SR_ADD perform the summation of the contributions and the previous compensated pixels with the present ones. Nonetheless, as the

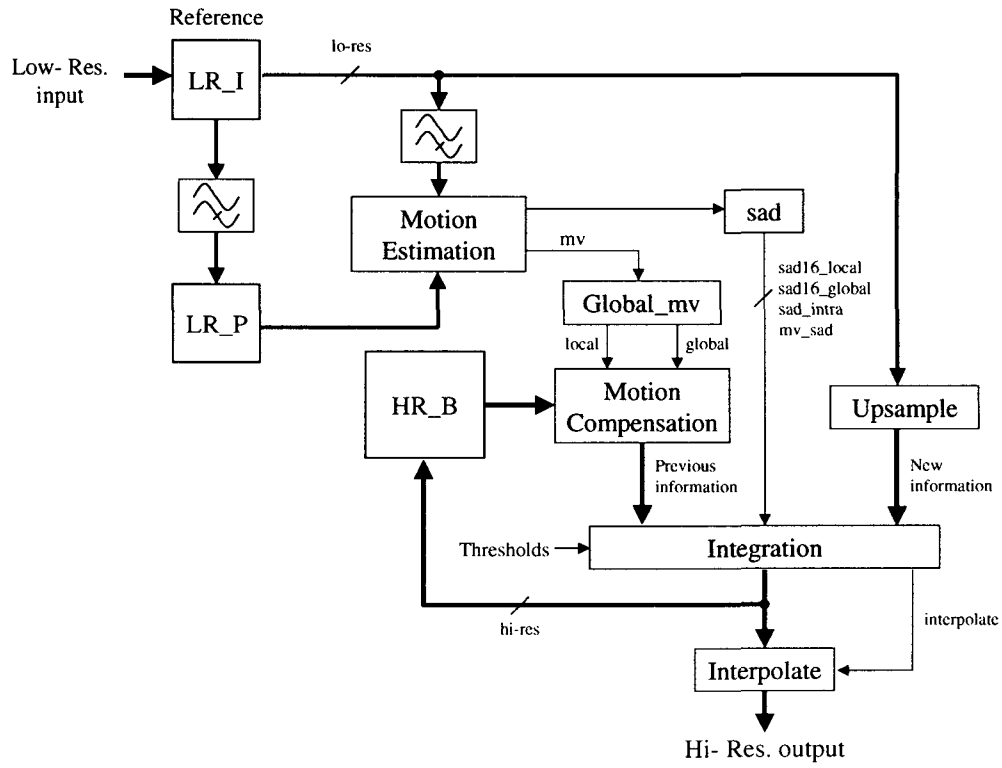


Figure 130. Block-diagram of the data-flow of the super-resolution algorithm v3.0.

present data are composed by three quarters of zeroes and it is only important to keep the new pixel, it is possible to replace the additions with a simple data replacement as shown in Figure 129. Therefore, we avoid the arithmetic addition operation. The variable 'SR_improve' informs if super-resolution improvements are going to be performed in the macro-block or not. If SR is not used, the macro-block zeroes will be interpolated.

However, as several luminance pixels share the same chrominance value, it is not possible to apply to the chrominances the same strategy and their values have to be firstly added and secondly averaged with the instruction SR_ADJUST. The instruction SR_UPDATE only affects to the memory that stores the image that is going to be displayed, i.e. HR_B. It interpolates the holes that could not be filled by super-resolution. Obviously, there cannot be holes in the chrominances, and consequently the instruction let the whole chrominance images to pass to the output. The last step of the algorithm is to store the filtered input low-resolution image in LR_P to be used in the next algorithm stage.

5.3.4.1.3 Block diagram and memory requirements for version v3.0

The block diagram of the data-flow for this version of the SRA is shown in Figure 130. In this figure all the temporal memories have been removed, only leaving those that will be present in a real implementation, i.e.: LR_I, LR_P and HR_B.

The path followed by the image data is highlighted with dark lines, the operators are shadowed and the memories squared with no shadow and simple lines. When implementing the algorithm it must be taken into account the addition of three macro-block stripes to avoid the data overlap when performing the motion compensation.

In Table 23 is shown a summary with the memory requirements of version v3.0 of the SRA expressed in low-resolution macro-blocks and excluding all the temporal memories. Notice that the high-resolution memory used is in fact the one denominated HR_B2, i.e., keeping the holes, and it is the moment of exporting the image out when the holes are interpolated, leaving to output sub-system the task of storing the final image delivered by Picasso.

*** Non iterative video super-resolution algorithm v3**

Initialization

```
Initial contribution: M = 4 . Each new image is the reference. (shadow lines are for data calculations)
LR_I = Read a new spatial-aliased low-resolution image
init: fr=0
IF (fr==0) THEN  // Initialize. HR_B2 and HR_Cont are for feedback purposes
  LR_P = filter(LR_I)                                                          [SR_INIT_P]
  Clip(LR_P, 0, 255)
  HR_A.lum    = Upsample_Holes(LR_I.lum)
  HR_A.chrom = Upsample_Neighbours(LR_I.chrom)                                 [SR_INIT_A_B]
  HR_B2 = HR_A
  HR_Cont = Create_image_contributions
  HR_W = HR_Cont                                                               [SR_INIT_CONT]
                  lum = [M 0; 0 0] ; chrom = [M M; M M]
  IF (HR_W(i)==0) THEN  HR_B.lum = Interpolate(HR_B2.lum)
  ELSE  HR_B.lum = HR_B2.lum                                                   [SR_UPDATE]
  HR_B.chrom = HR_B2.chrom
  Clip(HR_B, 0, 255) Clip(HR_B2, 0, 255)
  fr = 1;
ELSE
  LR_T = filter(LR_I)                                                          [SR_FILTER]
  Clip(LR_T, 0, 255)
```

Control

```
  MV_local[mb_x][mb_y] = Motion_Estimation (LR_P, LR_T)

  sad16_local[mb_x][mb_y] = Evaluate(LR_P, LR_T, MV_local[mb_x][mb_y])
  sad_intra[mb_x][mb_y] = Evaluate(LR_P, LR_T, MV_local[mb_x][mb_y])

  MV_global = Select_global_motion_vector(MV_local[mb_x][mb_y])
  sad16_global[mb_x][mby]=Evaluate(LR_P, LR_T, MV_global)

  MV_sad = Calc_MV_sad(MV_local, MV_global)   MV_sad = Σ|MV_local[mb_x][mb_y] - MV_global|
  local_penalty = LOCAL_PENALTY_THRESHOLD

  IF (MV_sad < MV_SAD_THRESHOLD)
local_penalty += 2
  ELSE
local_penalty -= 2

  IF (sad16_global[mb_x][mb_y] < sad16_local[mb_x][mb_y]+ local_penalty)
SR_improve[2·mb_x][ 2·mb_y] = 1                  // Global vectors
MV[mb_x][mb_y] = MV_global
  ELSE
SR_improve[2·mb_x][ 2·mb_y] = 2                  // Local vectors
MV[mb_x][mb_y] = MV_local[mb_x][mb_y]
```

Figure 131. First part of the pseudo-code of the version v3.0 of the SRA.

```
HR_T  = Motion_Compensation°(HR_B2, MV)                          [SR_MOT_COMP]
HR_T2 = Motion_Compensation° (HR_Cont, MV)                       [SR_MOT_COMP_CONT]

HR_S.lum    = Upsample_Holes(LR_I.lum)
HR_S.chrom = Upsample_ Neighbours(LR_I.chrom)                    [SR_UPSAMPLE]

HR_S2 = Create_image_contributions  lum=(M 0; 0 0); chrom=(M M; M M)   [SR_INIT_CONT]

HR_D = Motion_Compensation°(HR_B, MV)                            [SR_MOT_COMP_B]
sad_SR = Calc_sad_SR(HR_S, HR_D)                                 [SR_CALC_SAD]

Control
IF (frame_no == 0)
    SR_improve[mb_x][mb_y]=0              // Interpolate the first image always
ELSE
    context_change = 32 · (sad_intra − sad16_local) / (sad_intra + sad16_local)
    IF(context_change < CONTEXT_CHANGE_THRESHOLD)                // Context switch
        SR_improve[mb_x][mb_y]=0                                 // Interpolate block
    IF (mv_sad > MV_SAD_THRESHOLD)                               // High local motion
        IF(sad_SR[mb_x][mb_y]>MV_SAD_SR_INTERPOLATION_THRESHOLD) // High differences
            SR_improve[mb_x][mb_y]=0                             // Interpolate block
END IF

Lum  IF (even & even)          HR_W = HR_S2   (new contribution)
     ELSE IF(SR_improve)       HR_W = HR_T2   (previous contribution)
     ELSE                      HR_W = 0       (interpolation)
Chr  HR_W = HR_S2 + HR_T2                                        [SR_ADD_CONT]
                                                                 (HR_W = HR_S2 + HR_T2)
     IF (HR_W(i) ==0) THEN     HR_Cont = 0
     ELSE                      HR_Cont = M    (feedback)

Lum  IF (even & even)          HR_A = HR_S    (new pixel)
     ELSE IF (SR_improve)      HR_A = HR_T    (previous pixel)
          ELSE                 HR_A = 0       (interpolation)
Chr  IF (SR_improve)   HR_A = HR_S + HR_T     (superresolution)
     ELSE              HR_A = HR_S            (interpolation)     [SR_ADD]
                                                                 (HR_A = HR_S + HR_T)

IF (HR_W(i)==0)  HR_B2 = 0
ELSE             HR_B2 = M * HR_A/HR_W   (feedback)              [SR_ADJUST]

IF (HR_W(i)==0) THEN   HR_B.lum = Interpolate(HR_B2.lum)
ELSE  HR_B.lum = HR_B2.lum
HR_B.chrom = HR_B2.chrom                                         [SR_UPDATE]
Clip(HR_B, 0, 255)  Clip(HR_B2, 0, 255)

LR_P = filter(LR_I)
Clip(LR_P, 0, 255)                                               [SR_INIT_P]
 fr = fr +1;
END IF
```

Figure 132. Second part of the pseudo-code of the version v3.0 of the SRA.

Name	Memory		
	Luminance (bits)	Chrominance (bits)	Total (bits)
HR_B	(2·mb_x·2·mb_y·16·16·8)	(2·mb_x·2·mb_y·8·8·2·8)	12.288·mb_x·mb_y
3 Stripes HR	(2·3·2·mb_y·16·16·8)	(2·3·2·mb_y·8·8·2·8)	36.864·mb_y
LR_I	(mb_x·mb_y·16·16·8)	(mb_x·mb_y·8·8·2·8)	3.072·mb_x·mb_y
LR_P	(mb_x·mb_y·16·16·8)	(mb_x·mb_y·8·8·2·8)	3.072·mb_x·mb_y
MV_mem	(mb_x·mb_y·8)	0	8· mb_x·mb_y
Total (bits)	mb_y· (12.296·mb_x+24.576)	mb_y· (6.144·mb_x+ 12.288)	mb_y· (18.440·mb_x+36.864)

Table 23. Summary of the memory used by version v3.0 of the SRA.

In Table 24 are shown the total memory requirements for different image sizes.

Size	mb_x	mb_y	Memory (Kbytes)	Memory (Mbytes)
SQCIF	8	6	36.01	0.04
QAVGA	9	7	40.51	0.04
QCIF	11	9	49.51	0.05
HAVGA	18	14	81.03	0.08
CIF	22	18	99.05	0.10
AVGA	36	28	162.12	0.16
VGA	40	30	180.15	0.18
4CIF	44	36	198.19	0.19
16CIF	88	72	396.77	0.39

Table 24. Memory used by version v3.0 of the super-resolution algorithm for different image sizes.

In Figure 133, the data of Table 24 in Kbytes are shown. The main condition is the memory requirements to be very low, and in this case we can really talk about a realizable

low-cost super-resolution algorithm over a hybrid video-encoder. The low-cost characteristics are determined, on one side, by the possibility of implementing the algorithm with minimum modifications and as an added value over the existing platform, and on the other side, by the possibility of doing it without adding external memory (out-chip memory).

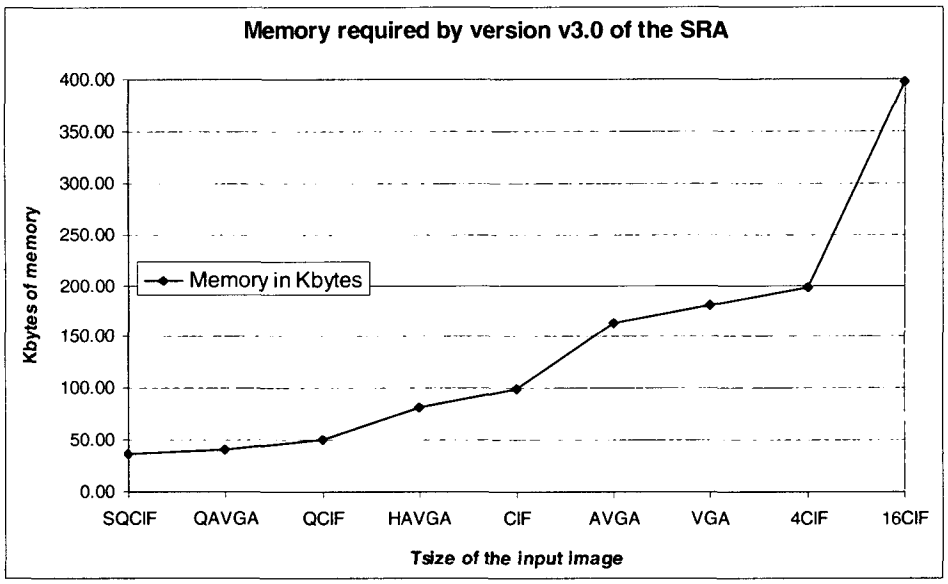


Figure 133. Memory used by the super-resolution algorithm v3.0 for the most common image sizes.

Size	mb_x	mb_y	v1.2 & v1.3 (Kbytes)	v2.0 (Kbytes)	v2.1 (Kbytes)	v3.0 (Kbytes)
SQCIF	8	6	315.19	450.19	459.05	36.01
QAVGA	9	7	413.68	590.87	598.56	40.51
QCIF	11	9	650.07	928.51	931.60	49.51
HAVGA	18	14	1,654.73	2,363.48	2,331.25	81.03
CIF	22	18	2,600.30	3,714.05	3,645.39	99.05
AVGA	36	28	6,618.94	9,453.94	9,198.98	162.12
VGA	40	30	7,879.69	11,254.69	10,936.17	180.15
4CIF	44	36	10,401.19	14,856.19	14,419.55	198.19
16CIF	88	72	41,604.75	59,424.75	57,354.19	396.77

Table 25. Comparative of the memory used by the most significant versions of the SRA for different memory sizes in Kbytes.

Table 25 shows a comparative of the memory amounts used by the most significant super-resolution algorithms, and in Figure 134 is shown a comparative chart where is clear the huge difference between this last version and the previous ones. In average, version v3.0 uses 35.63 times less memory than the iterative versions (v1.2 and v1.3) and 50.89 times less memory than the non-iterative versions for still images (v2.0 and v2.1). The price to pay for this drastic memory reduction is a slight decrease of the image quality, versus the quality obtained by versions v2.0 and v2.1, as will be further explained.

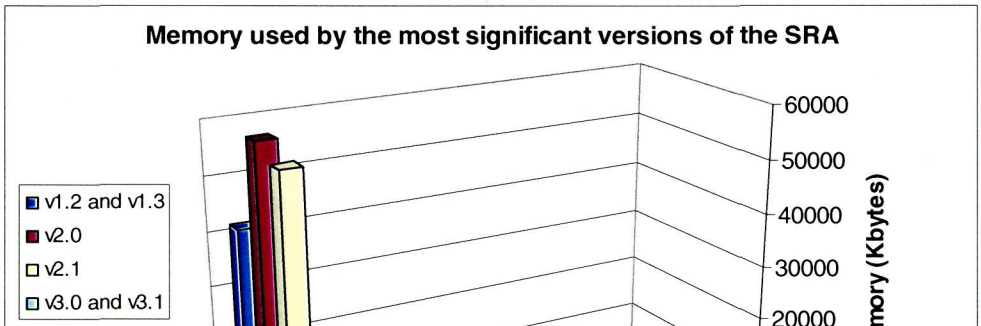


Figure 134. Comparative of the memory used by the most significant versions of the SRA for different memory sizes in Kbytes.

5.3.4.1.4 Simulation results and quality analysis of version v3.0 using ¼ pixel precision

In order to evaluate the image quality that the super-resolution algorithm can achieve, several tests have been carried out, that can be classified in:

a) **Tests with synthetic sequences**. In this case images with artificial shifts and induced aliasing have been used to generate the test sequences. They have the advantage of knowing all the parameters: reference image, shifts and aliasing degree. That allows as to obtain all kind of quantitative measures.

b) **Tests with real movement sequences**. These kinds of sequences are made with real

video sequences and induced aliasing. Although these are the kind of sequences that the system is going to deal with (except the input anti-aliasing sub-system), they have the drawback of not being possible to obtain quantitative quality measures. Only qualitative measures are possible.

Within the set of tests with synthetic sequences, it has been used the KRANT sequence with context change whose motion vectors are shown in Table 21. In Figure 135 is shown the PSNR reached by the luminance signal that is the most important in the perceptual quality. The quality for all the frames is quite higher than the interpolation qualities (a minimal 1.44 dB of improvement with respect to the bilinear interpolation and 5.54 dB as the maximum improvement). The average quality is of 28.64 dB, versus the average 25.3 dB of the bilinear interpolation. This average of 28.64 dB is somewhat lower than the 30.47 dB reached by version v2.0. At the system, application and commercial levels must be studied if a decrease of 1.83 dB will be compensated by a decrease of 50.89 times in the size of the integrated memory required by version v3.0

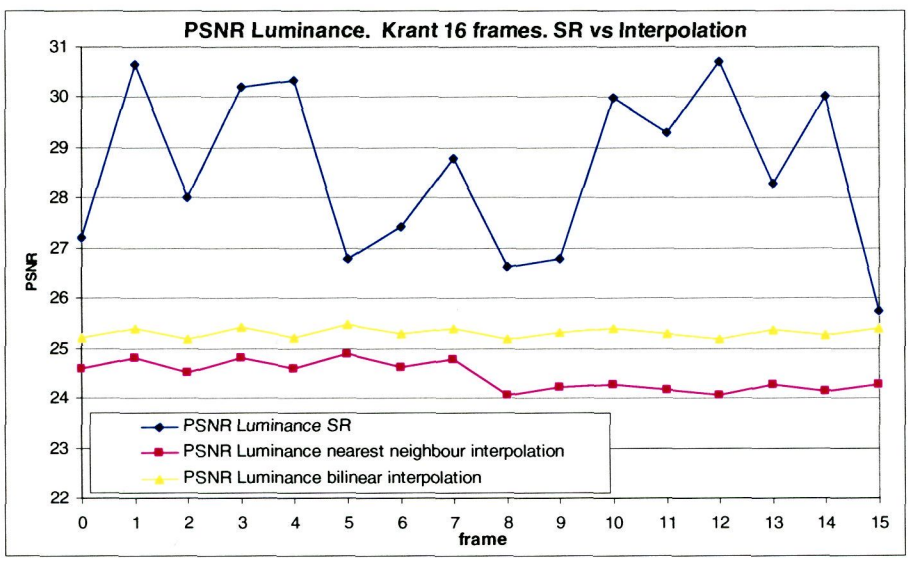


Figure 135. PSNR of the luminance of the KRANT sequence of 16 frames with context change in the frame number 8.

Figure 136 shows that the border effect barely have an effect on this algorithm version and that, as it was expected, the chrominances exhibit better PSNR than the luminance due to their lower entropy.

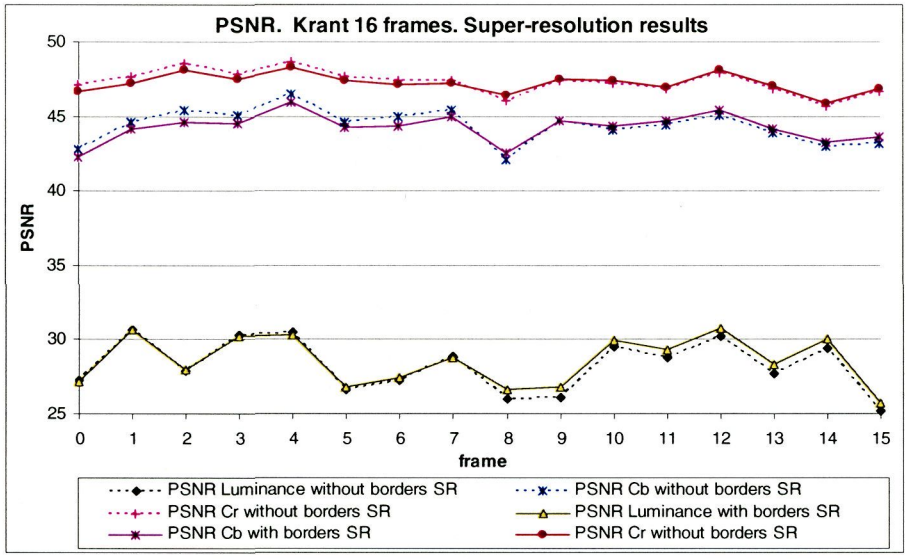


Figure 136. PSNR of the Krant sequence of 16 frames with context change in the frame number 8 with and without borders.

In the Figure 137 the spectral correlation coefficients in magnitude are shown. Frames number 8 and 9 drop below the interpolation quality level due to the context change. The case of frame 8 is because the context change is detected and the image is all interpolated. The case of frame 9 is because the motion vector set does not contribute with a great deal of new information and so, the quality is slighter higher than the interpolation level.

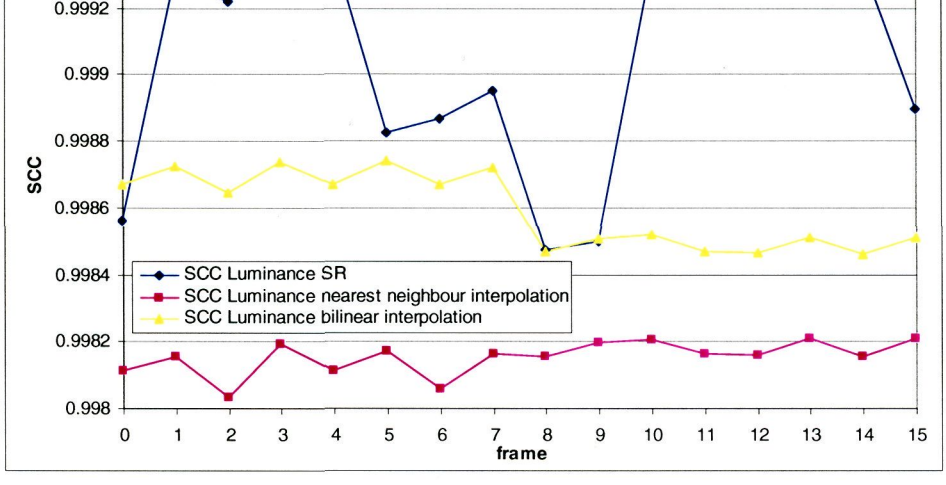


Figure 137. SCC in magnitude of the luminance of the KRANT sequence of 16 frames with context change in the frame number 8.

In Figure 138 the SCC in magnitude, both of the luminance and of the chrominances with and without borders are shown. Equally than with the PSNR the chrominance values exhibit better behavioural and the variation with the borders are minimum before the context change and slightly higher after. In phase the obtained behavioural are very similar.

With respect of applying version v3.0 to sequences with real movements, the following experiment has been designed. The phrase 'Let's make things better' has been disposed in rows and columns with different font sizes (from 6 to 48 points), indicating the size on the left, in vertical and horizontal orientations (Figure 139). This text has been recorded using a low-grade camera (camera for PC ToUcam Vesta PCVC675K from Philips). As the camera was held with the hand, that produced the necessary shifts to increase the images resolution. Several sequences from different distances were taken. The presence of aliasing in the input frames is only due to the camera itself and not to a previous subsample process that artificially introduces the aliasing.

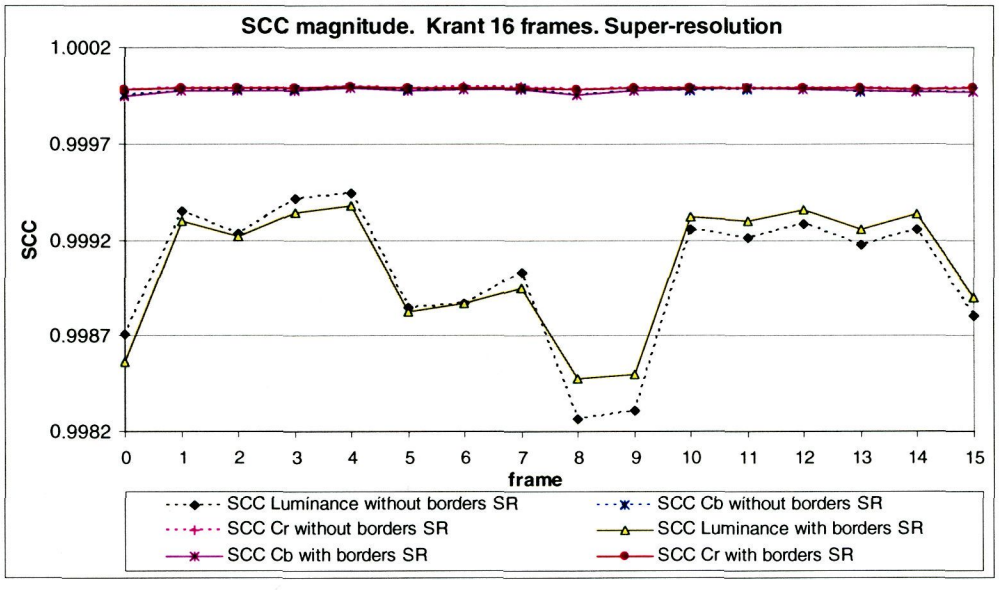


Figure 138. SCC in magnitude of the KRANT sequence of 16 frames with context change in the frame number 8 with and without borders.

In Figure 140 (a) it can be observed the frame number 5 of the input sequence and in Figure 140 (b) the super-resolved image. In general it can be appreciated a recovery in the character edges and a decrease in the background noise, what it is natural as every resulting image comes form the average of some other related images.

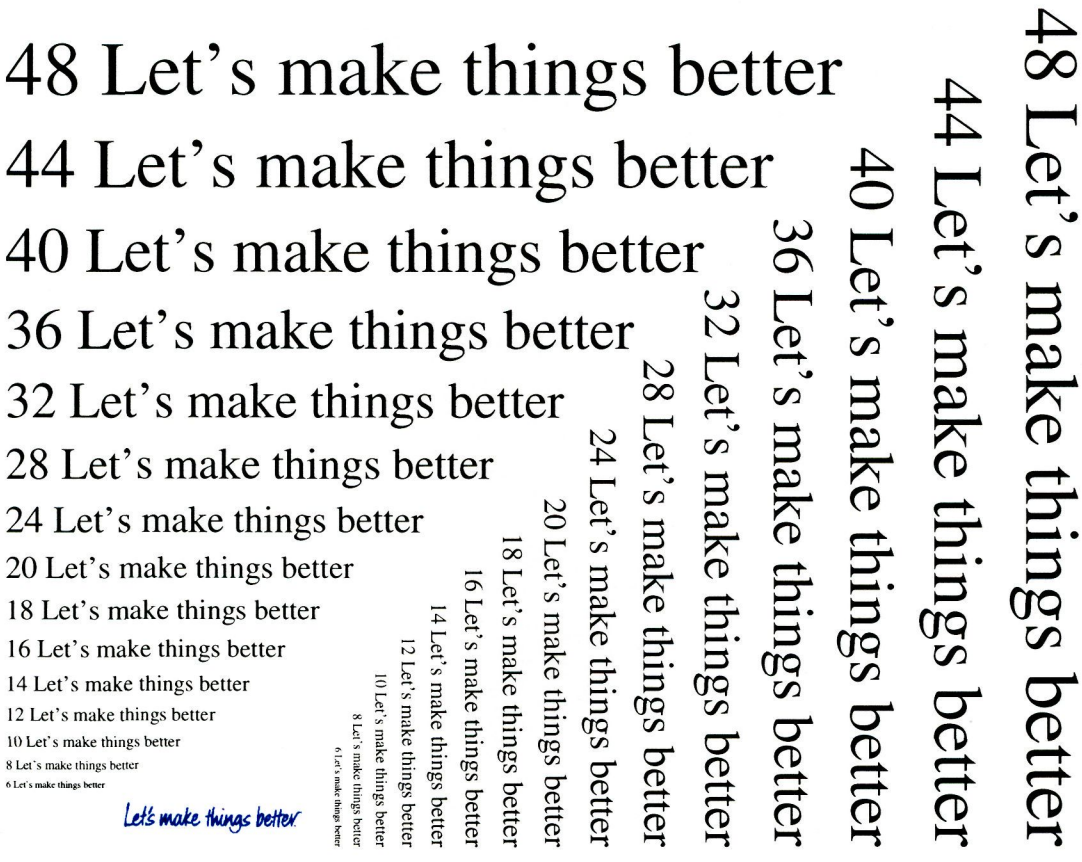


Figure 139. Test text for sequences with real movement.

One relevant aspect in the super-resolution improvements is the size of the fonts that can be clearly read in the input image and in the super-resolved image. In Figure 141 an enlarged detail of Figure 140 can be seen. It is clear how difficult it is to read the fonts of size 28 in the input image (a), while fonts of size 24 are uncertain to be read and fonts of size 20 are impossible to read. Conversely, in the super-resolution image these font sizes are easier to read, even though that the characters of 20 points are still very blurred.

In Figure 142 (a), the frame number 7 of other test sequence is shown, but this time bringing nearer the camera to the text, and the result after applying the super-resolution algorithm (b). In the enlarged detail of Figure 143, it can be clearly seen the readable improvements when applying the SRA on (a) to obtain (b).

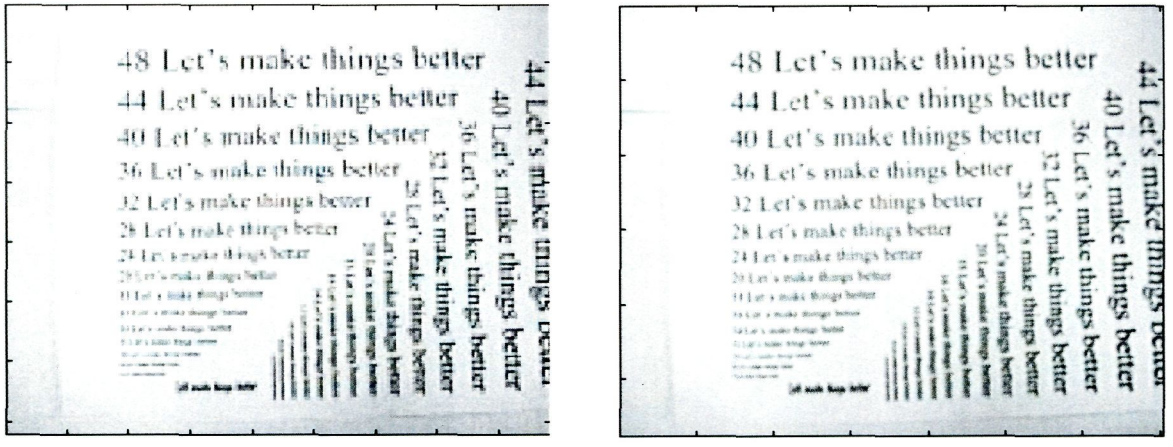


Figure 140. Frame number 5 of the input sequence (a) to the SRA and of the output super-resolved sequence (b).

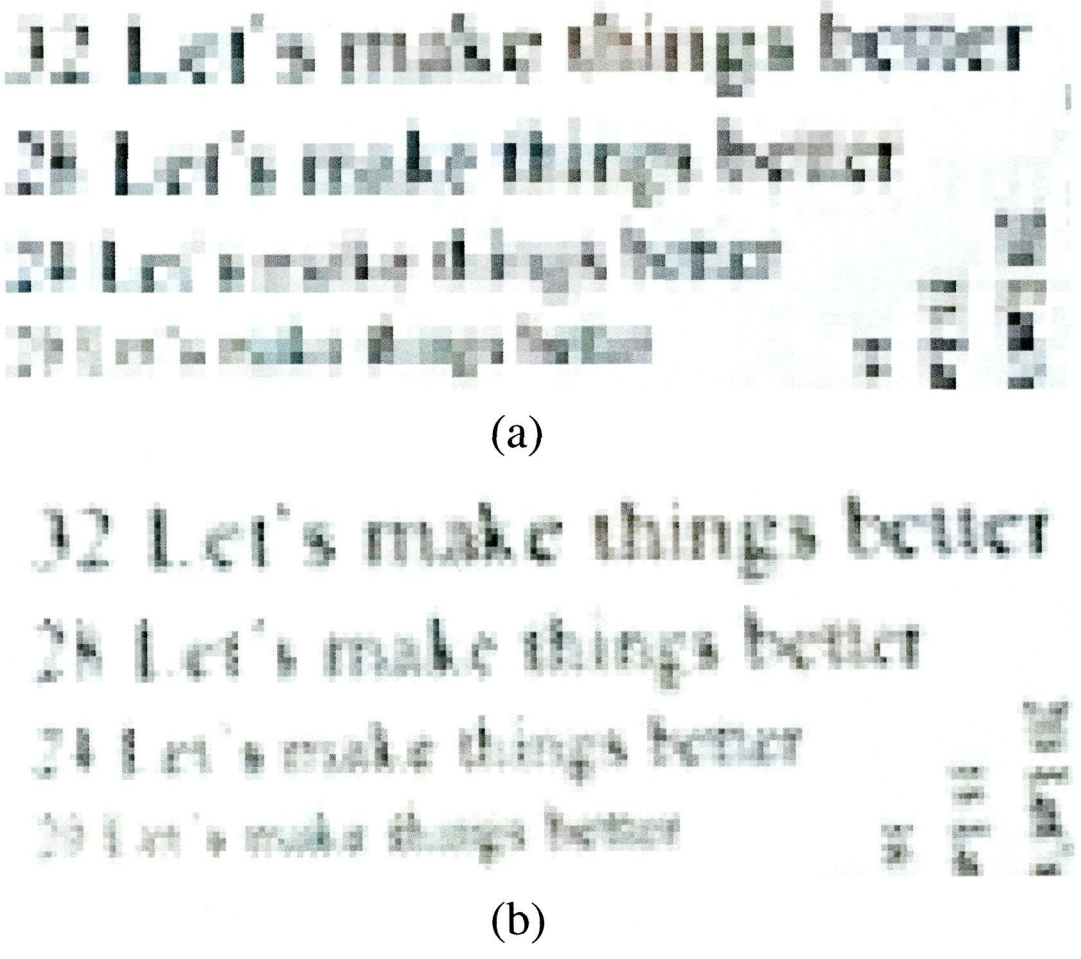
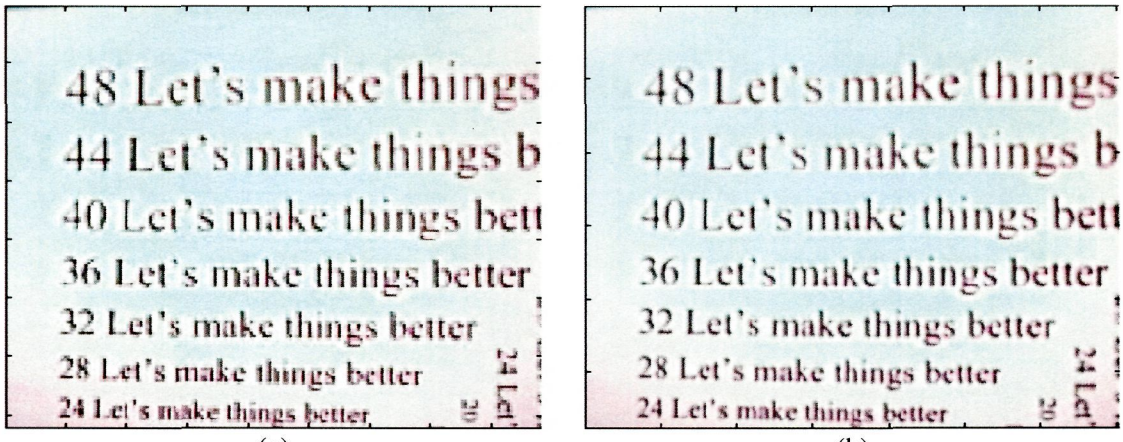
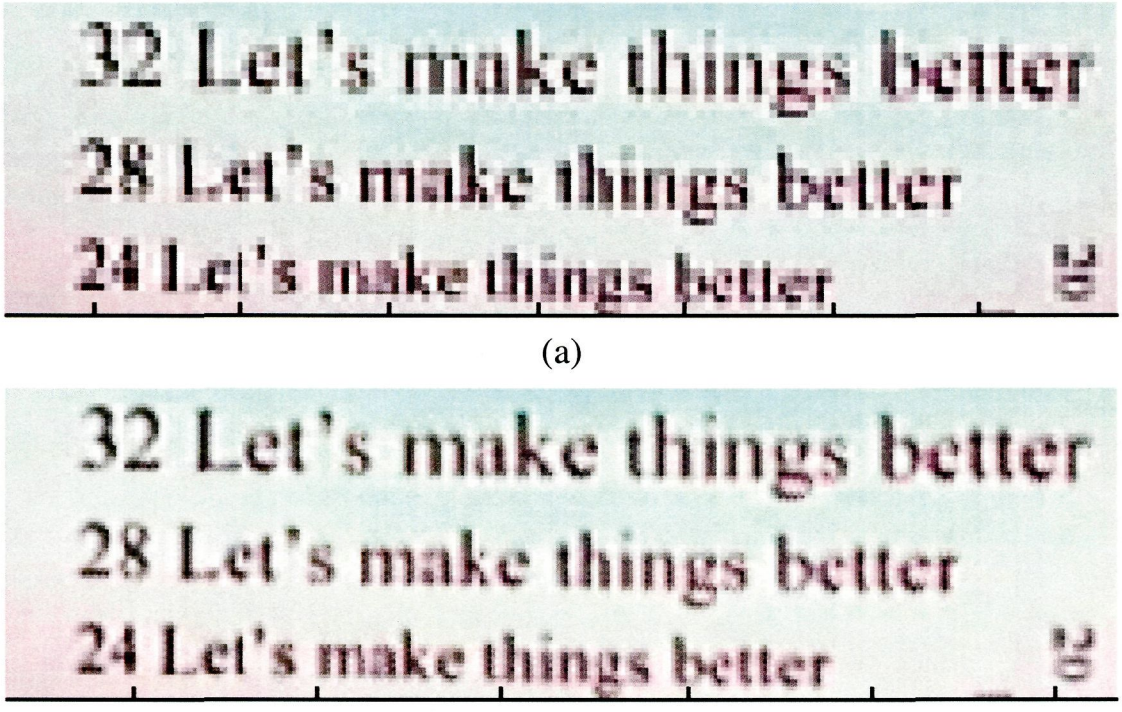


Figure 141. Enlarged detail of frame number 5 of the input sequence (a) and of the output super-resolved sequence (b).



(a) (b)

Figure 142. Frame number 7 of the input sequence (a) and of the output sequence (b) of the SRA.



(a)

(b)

Figure 143. Enlarged detail of the frame number 7 of the input sequence (a) and of the output sequence (b) of the SRA.

5.3.4.2 Modified algorithm for video sequences (v3.1)

This new version v3.1 is also intended for video sequences, where every new input will be the image reference. When the first image comes (Figure 144), it is interpolated using a bilinear interpolation algorithm, and stored in memory. The interpolation is done in such way that the original values are not modified. A more detailed explanation is provided in section 0. Then, this interpolated image is encoded, being the first output of the system.

After that, when a new image comes from the capture system, the following operations are performed:

1. The image is interpolated using bilinear interpolation.
2. Block-base motion estimation is done between this new image and the previous one stored in memory.
3. With the motion vectors obtained from this estimation, a motion compensation of the previous image is done, obtaining the previous image motion-compensated.
4. Using those two images, an upgrade is done mixing pixels of both images as is shown in the Figure 146.
5. This upgraded image is kept in memory.
6. The result of this upgrade is codified and transmitted.

In this way we obtain a continuous video flow in the output and, theoretically and in the best case, we can obtain a high-resolution image after four images in the sequence. The last step is to go back to the first step of the loop, interpolating the input image, when bad motion estimation is detected, i.e. a context change is detected.

We can also show this process in a block view, as it is done in the Figure 145 to Figure 147. Figure 145 shows the case of the first image of the loop, which is interpolated and kept in memory to be the reference for the next motion estimation. Figure 146 shows the process with the remainder images in the loop.

The main difference with version 3.0 is that the interpolation is performed in the input image instead of in the output image to fill the holes. Another important difference is that contributions have been removed as far as in this new version they only produced improvements in the borders of the image, but not in the main body. This shortens in the

algorithm features saves a great amount of image memory with slight looses in the perceptual image quality.

The first step is to interpolate the new image at the input, and then carry out a motion estimation between this interpolated image and the one kept in memory. With the results of this motion estimation, the previous image kept in memory is compensated and updated with new pixels from the input. This upgraded image is the output of the system and also the new reference for the motion estimation, as is shown in Figure 147.

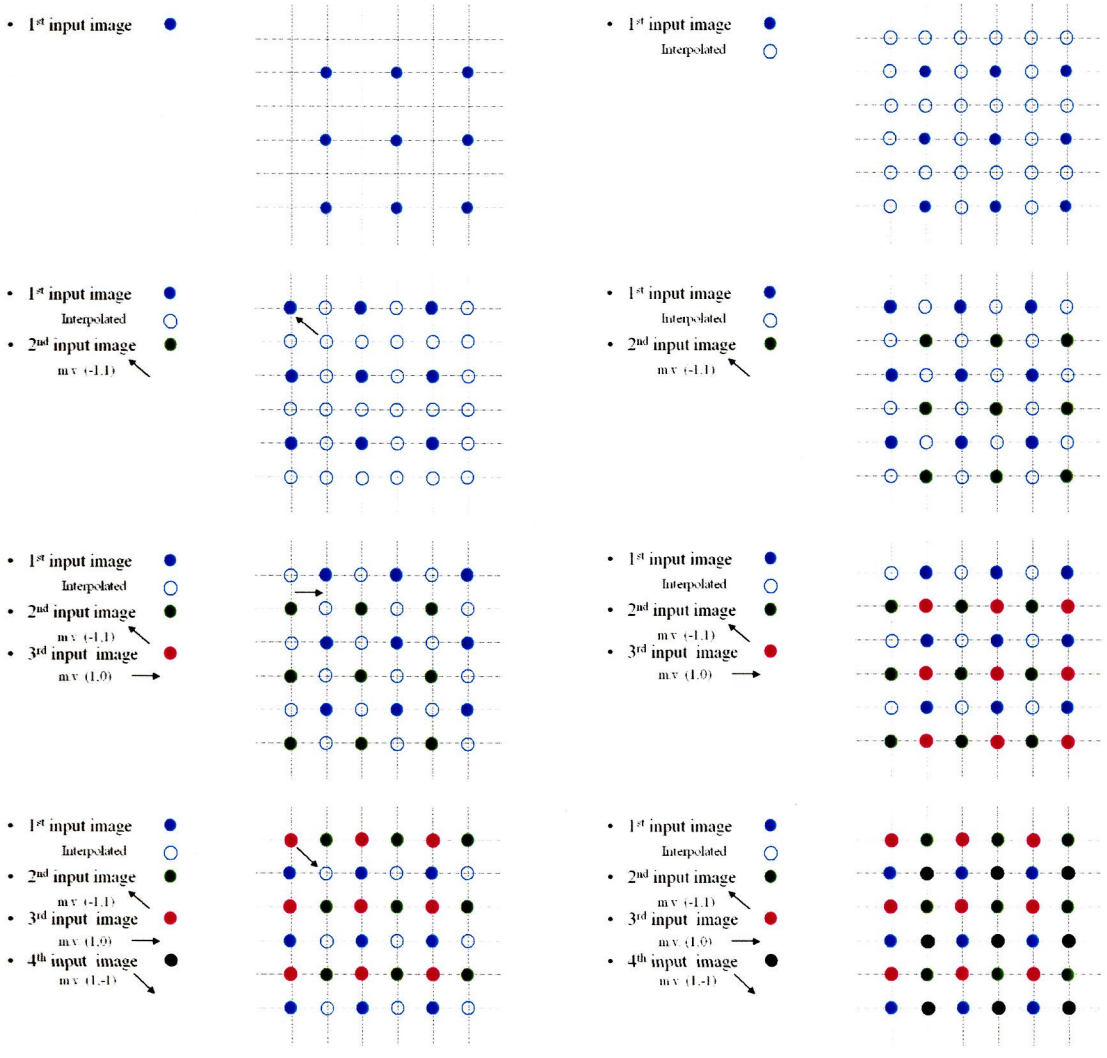


Figure 144. Graphical view of the v3.1 super-resolution approach.

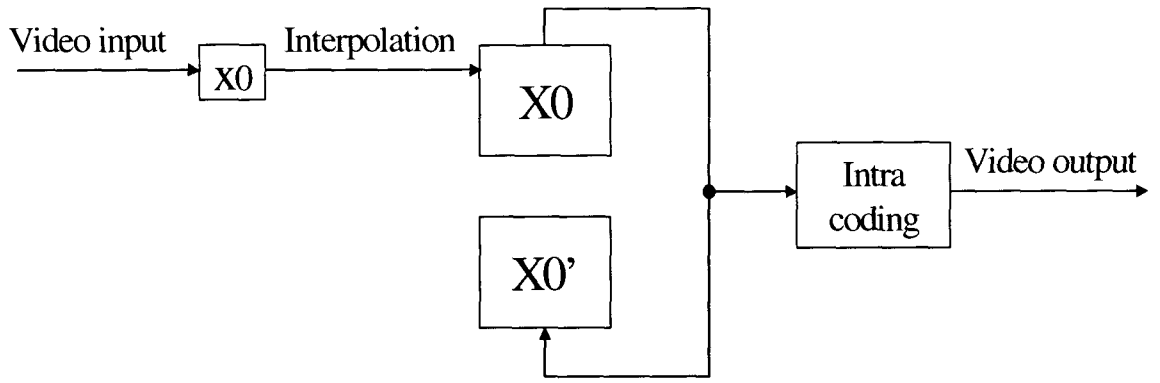


Figure 145. Block view of the first image processing.

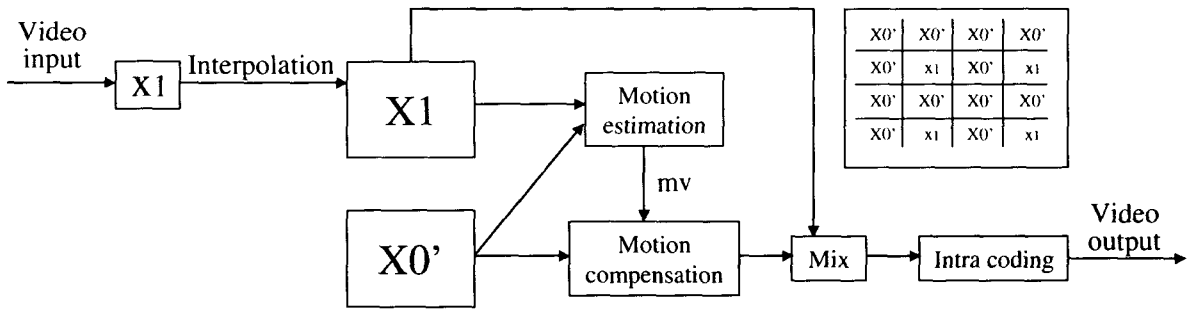


Figure 146. Block view of the loop images processing

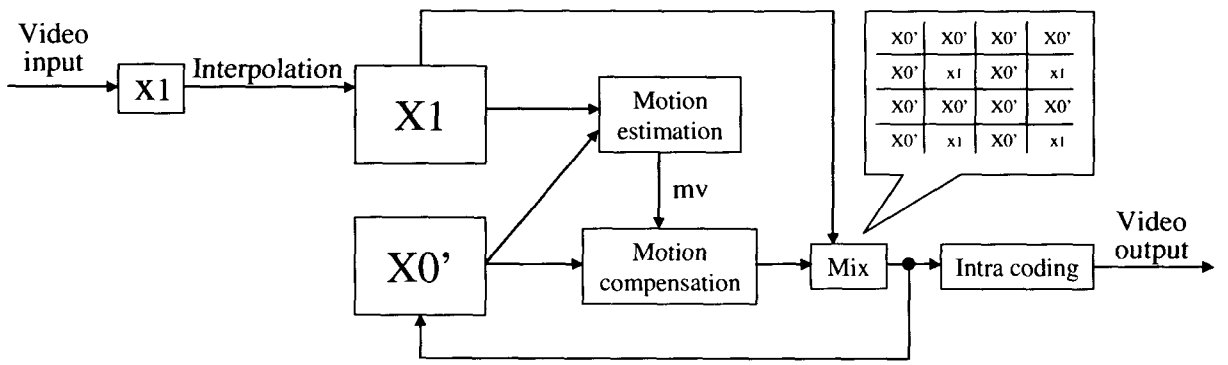


Figure 147. Next step after Figure 146. In this step the upgraded image is kept in memory to be the next reference in the motion estimation.

Once we have this block view, it is possible to identify the main operations to execute in this approach. These operations are:

- Bilinear interpolation.

- Motion estimation and compensation.
- Mixing of both images (current and previous).
- Keeping the upgraded image in the memory.

Therefore, to map the super resolution algorithm in the hybrid video encoder the following blocks will be reused: the motion estimator, the motion compensator, the inverse motion compensator and the loop memory. It is also necessary to implement the interpolation, which is not currently considered in the original hybrid video encoder.

An important issue is that we use only the lossless part of the hybrid encoder to keep the image in the loop memory. For that reason, a new path is established in the hybrid encoder to avoid the DCT, IDCT, Q and IQ blocks, connecting the MC to the IMC directly. The DCT and Q blocks are only used when the image is compressed.

This approach for video was tested using the artificial experimental setup, obtaining good results as are shown in the Figure 148.

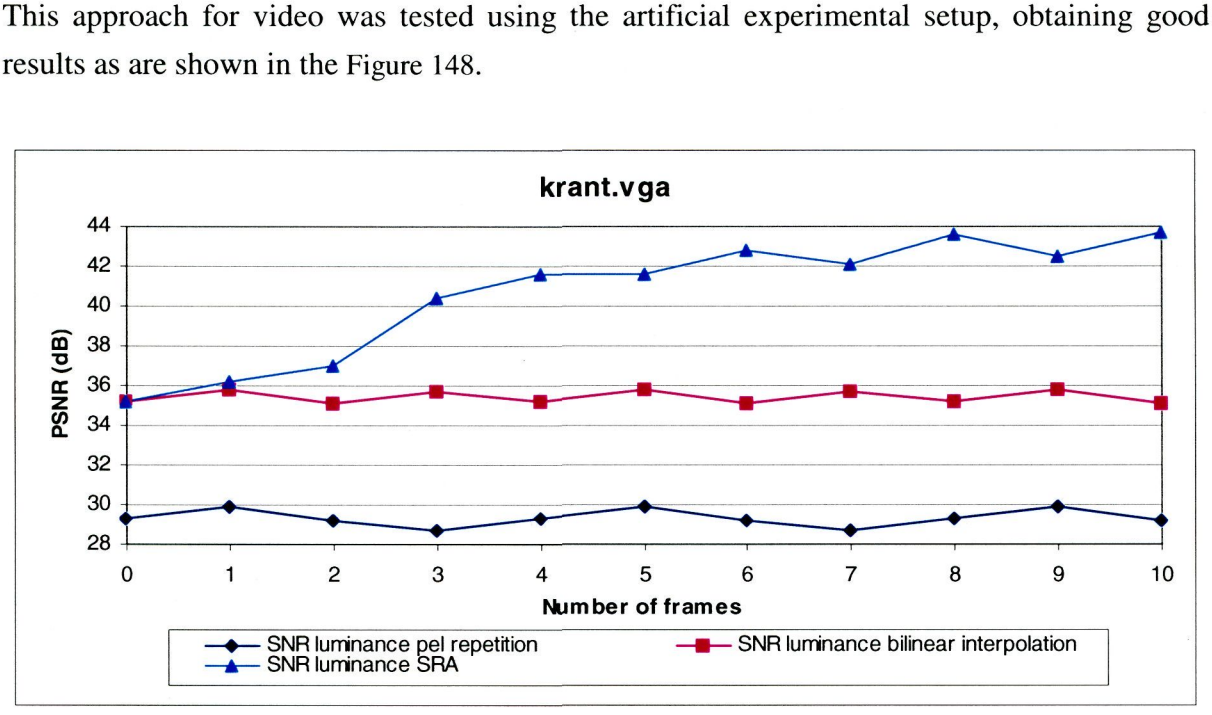


Figure 148. Results using artificial setup with the KRANT sequence.

5.3.4.2.1 Interpolation in the input

A new issue in this version is that the interpolation has been moved from the last step to the first one. Two main ideas have driven this change:

- To decrease the algorithm complexity.
- To reduce the number of hardware architectural changes to support super-resolution.

This target is achieved using interpolation in the input because we are using images with the same size both for compression and for super-resolution.

The interpolation in the input is a bilinear interpolation. It is performed when the data is read from the capture system. The image format is YUV 4:2:0 with interlaced format. It means that we have four pixels of luminance (Y), one for blue chrominance (U) and another one for red chrominance (V), ordered in a YYUYYV structure. This structure is shown in Figure 149.

Figure 149. Interlaced YUV 4:2:0 format.

In consequence, the interpolator reads two lines of luminance and one of chrominance to write four lines of chrominance and two of chrominance.

To achieve a real-time interpolation directly from the input data upon a macro-block basic, it is carried out always using historical data (previously read in other neighbour macro-blocks of the same image). Some positions are interpolated using the current data, and the rest using data kept in memory. The positions interpolated with the current data are depicted as circles in Figure 150 (b). Previous values used to perform the bilinear interpolation are shadowed in Figure 151.

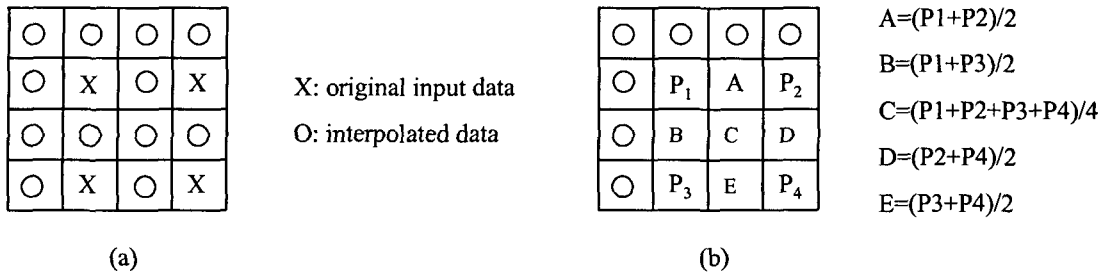


Figure 150. Luminance interpolation: (a) position of original pixels and (b) data interpolation from the original pixels P_i.

With this approach, it is possible to interpolate the entire image without artefacts, except in the top and left borders, where the pixels are simply copied from the nearest position. This problem does not affect to the whole image perception and it is inevitable.

With the chrominance signal, a similar approach is followed, but in this case only one original value is available. Then, the interpolation is done always with values kept in memory.

Doing the interpolation in the input, the contributions concept disappears. This fact simplifies even more the motion compensation process, because only the image has to be motion compensated and not the contributions.

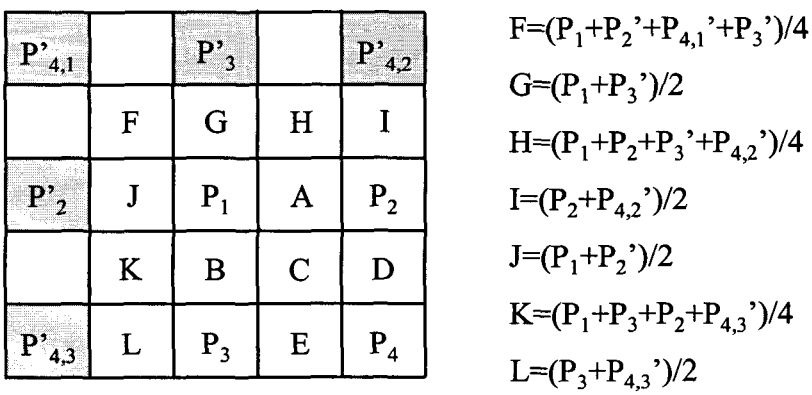


Figure 151. Interpolation using previous values.

5.3.4.3 Motion estimation over real motion sequences

At this point, a new test system was adopted, trying to be closer to the real case. The camera movement becomes real, keeping the aliasing generation in an artificial way. To do that, a set of sequences of 50 frames each one was recorded. In these sequences, the camera was held in the hand, trying to maintain it as quiet as possible in a motionless scene. With the small tremble of the hand, the sequence has the small shifts that we simulated before. The aliasing is generated as in previous steps by sub-sampling.

Finally, three sequences were selected and the system was tested with these sequences. The first frames of those sequences are shown in the Figure 152.

5.3.4.3.1 Motion model problems

After performing the motion estimation we can chose between using directly the obtained motion vectors (one motion vector per macro-block), described as local vectors, or computing the global vector of the scene. Moreover, we can choose as seen in section 4.2.4.1 between local or global vector for every macro-block. In this way the system was tested with several sequences, obtaining the results shown in Figure 153 to Figure 155. In each one of these charts, five curves are shown.

1. Luminance PSNR of each sequence using nearest neighbor interpolation.
2. Luminance PSNR using bilinear interpolation.
3. Luminance PSNR using super-resolution algorithm with control step to choose between global or local motion.
4. Luminance PSNR when the local motion vector is always selected.
5. Luminance PSNR when global motion vector is always selected.

Some conclusions can be taken from these figures.

- PSNR of bilinear interpolation is around 3 dB better than PSNR of nearest neighbor interpolation in all the sequences with aliasing.
- Super-resolution gives around 2 or 3 dB of improvement in PSNR respect to bilinear interpolation.

- Because the global movement is present, the control step selects global motion vector always.
- Calculated global motion vector is not always the best option, as it is shown for example in Figure 155. In this case, even if the global motion is present, the calculated global motion vector does not give the correct motion estimation. Instead of that, local motion vector fits better. The causes of that problem are:
  - Rotations in the camera, which are not present in the motion estimation model (Cartesian model).
  - Objects at different distances. In that case farther objects have more movement in the sequence than closer objects, even with global camera movement.
  - Global vector has influence over the entire image. As consequence of it, a small error in the motion vector can produce a big decreasing of the PSNR.

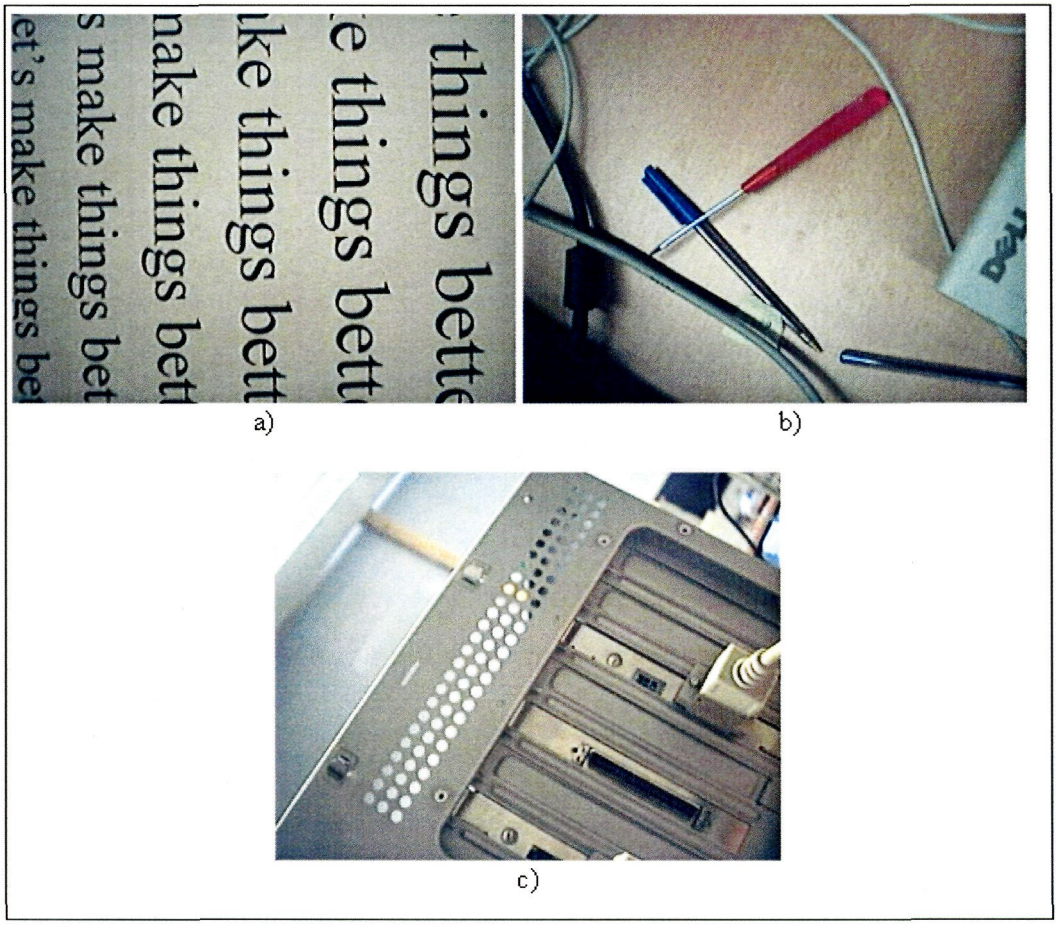


Figure 152. First frame of each recorded sequence, a) 'lmtb', b) 'pen' and c) 'sockets'.

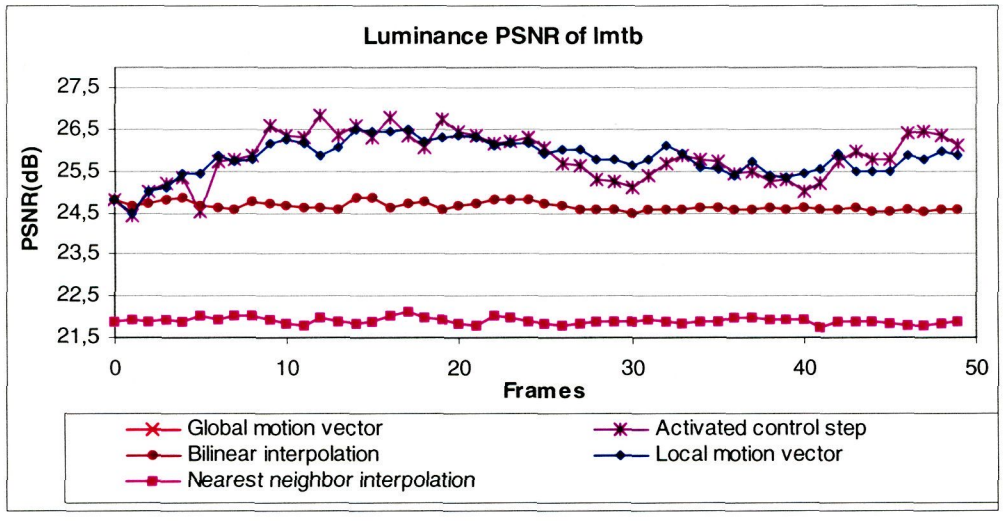


Figure 153. Luminance PSNR of 'lmtb' sequence.

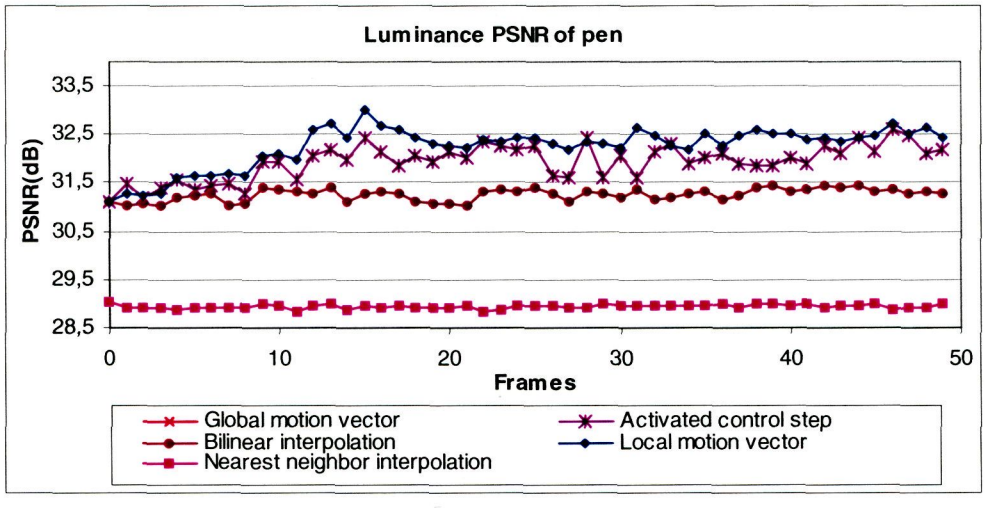


Figure 154. Luminance PSNR of 'pen' sequence.

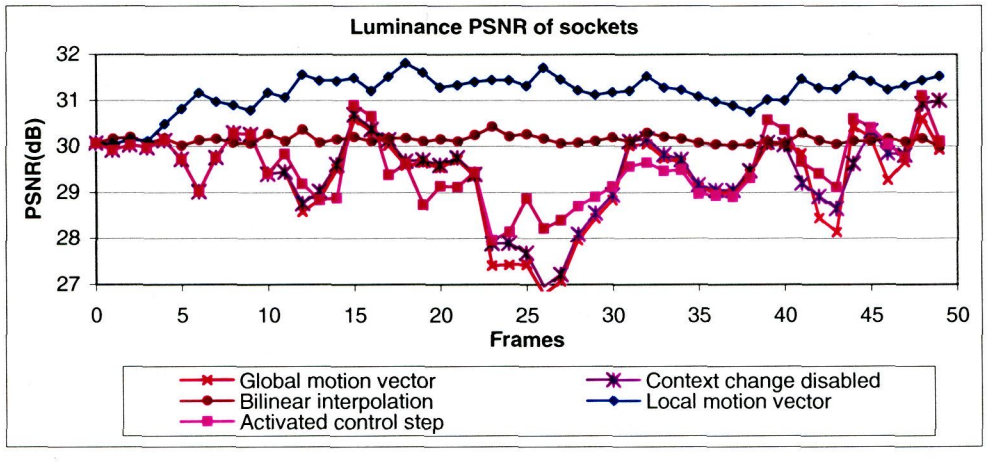


Figure 155. Luminance PSNR of 'sockets' sequence.

The control step was checked again in the next experiment to know its influence in the quality. In this case, context change control should not have any influence in the performance, because no context change is taken place. The context change part of the control step was disabled to check it. The results are shown in the Figure 156 to Figure 157. In this case, the nearest neighbour interpolation curve has been omitted and a new curve with the context change disabled has been included.

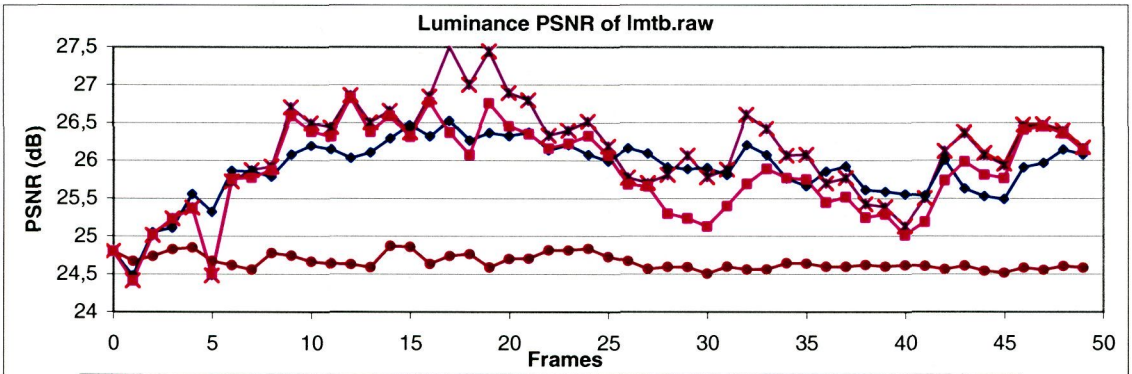


Figure 156. Luminance PSNR of 'lmtb' when context change control is disabled.

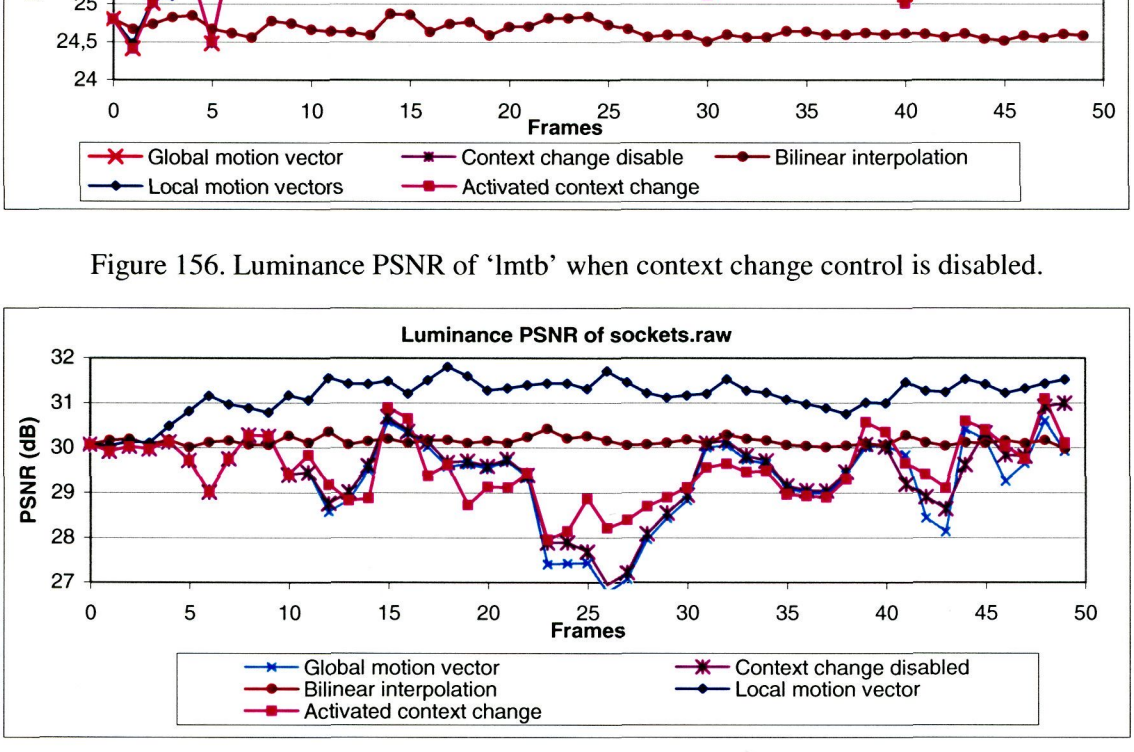


Figure 157. Luminance PSNR of 'sockets' when context change control is disabled.

In these figures, we observe that the context change control step has influence in the PSNR, even when no context change is happening.

With all these results, we optimized the control step. The control is based in a set of metrics, functions and thresholds. The main operation is to optimize the thresholds to obtain the best selection for each macro-block. The objectives of these thresholds optimizations are:

- Select between local and global model, taking the best of both. As was shown before, it has not to take global motion always, even if the global motion is present.
- Avoid any influence of context change when it is not happening.

After some experiments, we detected that the threshold changes only move the performance from global motion model to local motion model, but it never takes the best of both. If the thresholds are moved to promote the local model, the performances are closer to the local model. If the opposite way is taken, we have the global motion performance for all the sequences. A new way to compute the global motion was experimented to better approach the correct global motion vector. The results will prove that the context change influence was solved.

Another criterion to do the motion estimation was applied, as to use the Square Error (SE) instead of the Sum of Absolute Differences (SAD), but we did not find any clear improvement.

5.3.4.3.2 Motion vector filtering

To solve the problem of motion estimation with real movement, a filtering in the motion vectors spatial domain was done. The procedure to perform this filtering was the following:

- Compute the local motion vectors.
- Filter the local motion vectors in a sliding window of nine macro-blocks (3x3), with low-pass filter weights.
- Use this filtered motion vectors as the definitive motion vectors, avoiding the control step.

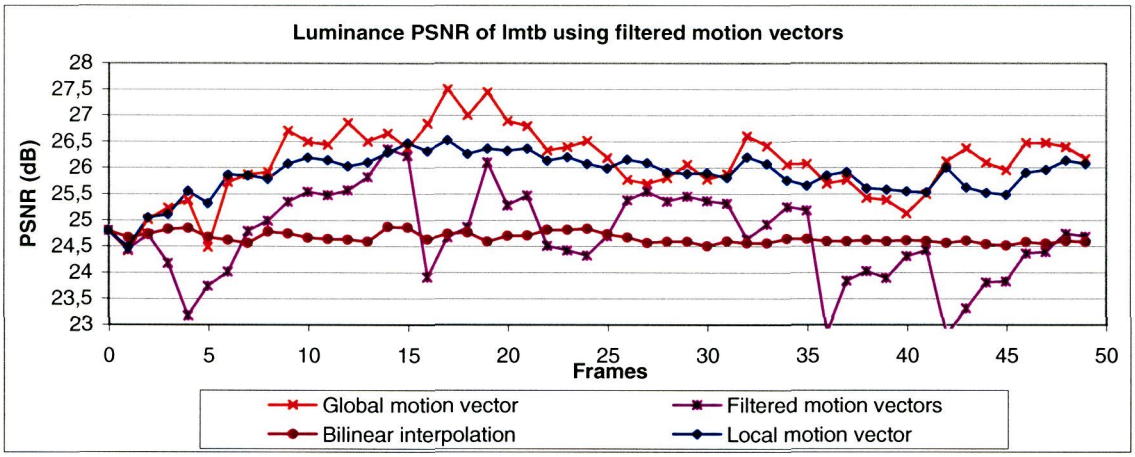


Figure 158. Results filtering the motion vectors field.

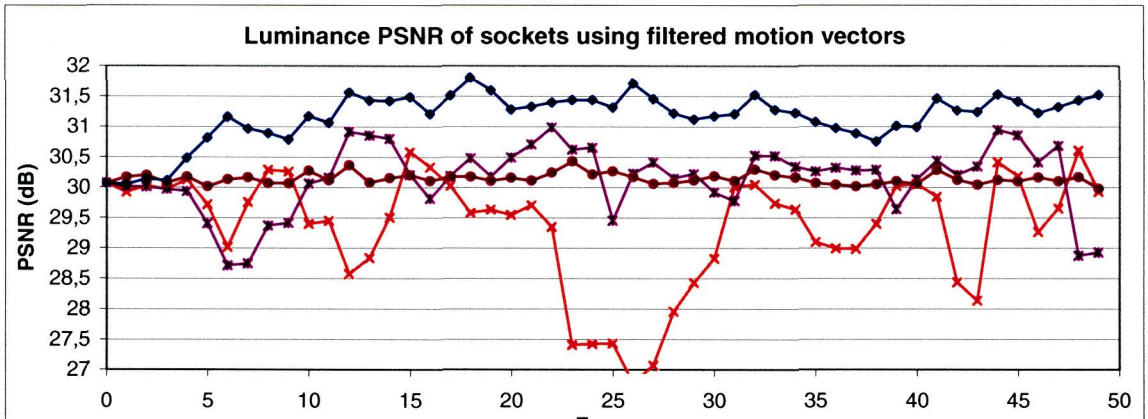
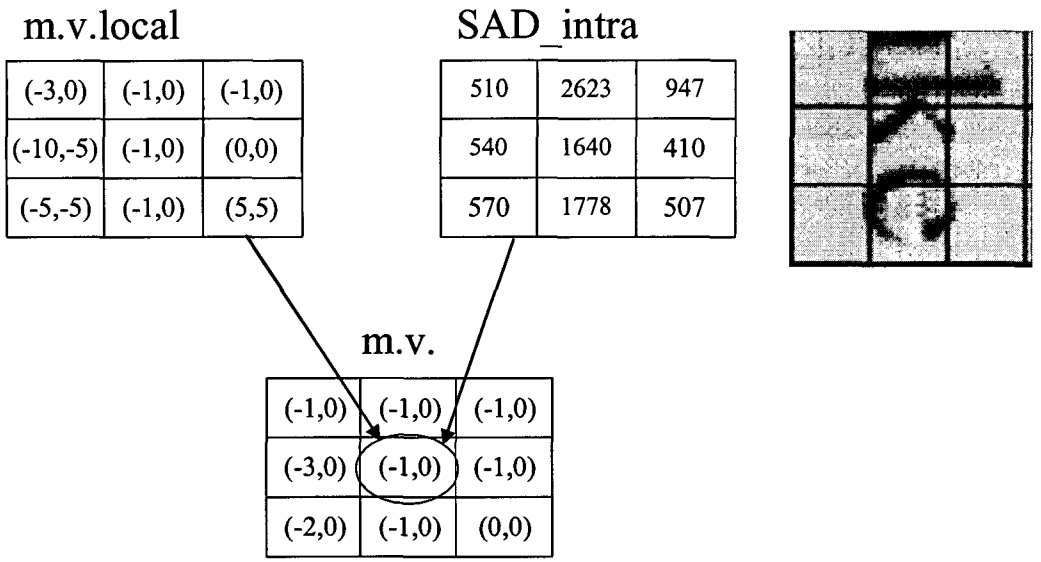


Figure 159. Results filtering the motion vectors field.

In this way, we avoid the control step and we obtain a "global in a region" local motion vector closer to the real one.

An experiment was carried out obtaining the results shown in the Figure 158 to Figure 159. In this step, the accuracy of some motion vectors was very bad. Macro-blocks that contain very few information had motion vectors with bad accuracy. Due to the size of the window, one motion vector very different to its neighbors can affect them very deeply. One way to avoid this motion vectors is to put some weights depending on the amount of information of each macro-block. In this way, a new filter was designed. The new steps are:

- Compute the local motion vectors.
- Filter the local motion vectors in a sliding window of nine macro-blocks (3x3), with a low-pass filter using the SAD_INTRA values as weights.
- Use this filtered motion vectors as the definitive motion vectors, avoiding again the control step.



$$m.v.(x,y) = \frac{\sum_{i=-1}^{1}\sum_{j=-1}^{1} m.v.local(x+i,y+j)\times SAD\_intra(x+i,y+j)\times coeff(i+1,j+1)}{\sum_{i=-1}^{1}\sum_{j=-1}^{1} SAD\_intra(x+i,y+j)\times coeff(i+1,j+1)} \qquad coeff = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Figure 160. Filter using the SAD_intra values as weights.

The experiment was repeated, obtaining this time good results as it is shown in Figure 161 to Figure 163. With this approach, it is possible to keep the best of local and global motion, obtaining the best quality of both.

Another important improvement with this technique is the reduction of the image memory. In the case of motion models, to calculate the local motion vector was necessary to wait until the motion vector of the whole image. Therefore, motion estimation and compression couldn't be done in parallel, and two entire images had to be stored.

With this new approach, it is only necessary to wait until the motion estimation of the next stripe. Motion estimation and other processing can be done in parallel, and the same amount of memory than before is used.

In order to reduce the complexity of this filtering due to the multiplication by SAD_intra, another approach was tried. In this approach, we use different values of coefficients depending in the values of SAD_intra, increasing them where SAD_intra is high and decreasing where SAD_intra is low. In this way, we use some comparisons instead of multiplication. The results are almost similar, achieving the same advantages than with the multiplication, but reducing the computational complexity.

The obtained data are shown in Figure 165. These figures show the first frame of each sequence and the frame 20. The first frame is equivalent to a bilinear interpolation. After 20 frames, the images obtained using this approach with these sequences present good quality.

5.3.4.3.3 Exhaustive search

After achieving these results, we want to know what would be the best performance that we can obtain from these sequences. We have to take into account that in this case the movement is real, and the expected results are not as good as with artificial movement. The main reasons for that are:

- All the shifts are possible, not only with quarter pixel accuracy.
- Not all the shifts that cover all the positions will be present, reducing the final performance.
- The motion estimation errors.

We designed a system to simulate the algorithm using an exhaustive search in the motion vectors. We also supposed a global motion model in all the sequences. The system is represented in the Figure 164, and it is composed by three applications:

- An application (named SHIFTS) that generate all the possible shifted images in a certain range and accuracy. With this application we obtain a sequence of moved images. It is equivalent to a motion compensation using all the possible motion vectors.
- An application (named MIX) that mix all the previously generated shifted images with the new low-resolution image.

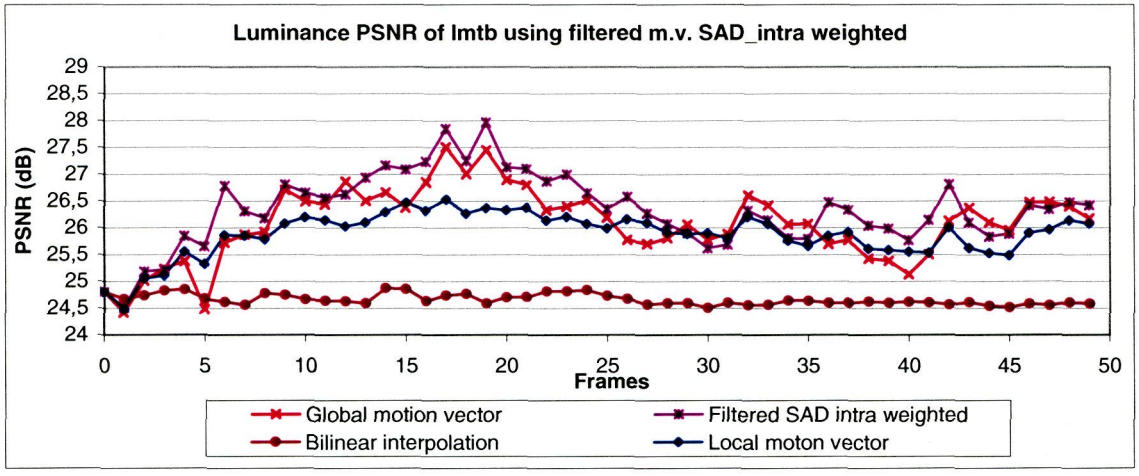


Figure 161. Results of 'lmtb' using filtered motion vectors using SAD_intra as weights.

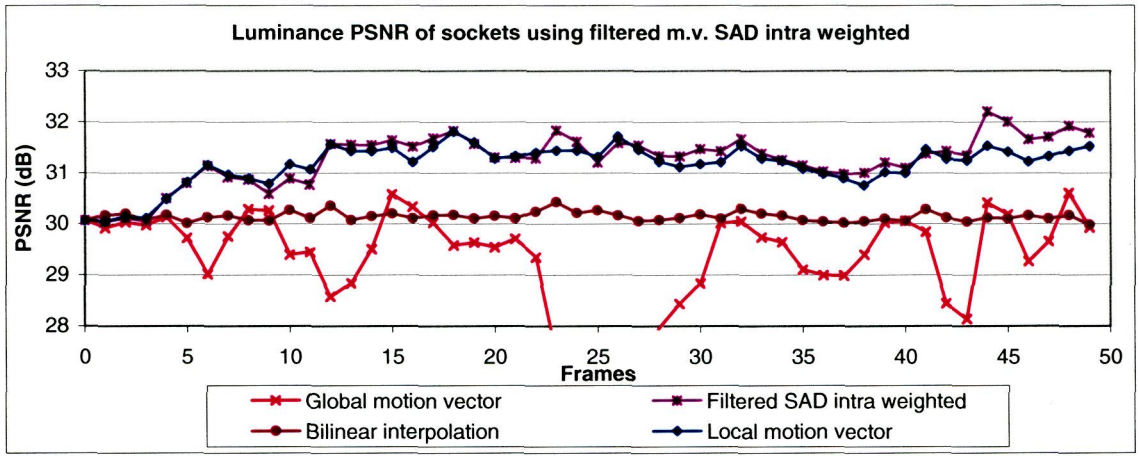


Figure 162.Results of 'sockets' using filtered motion vectors using SAD_intra as weights.

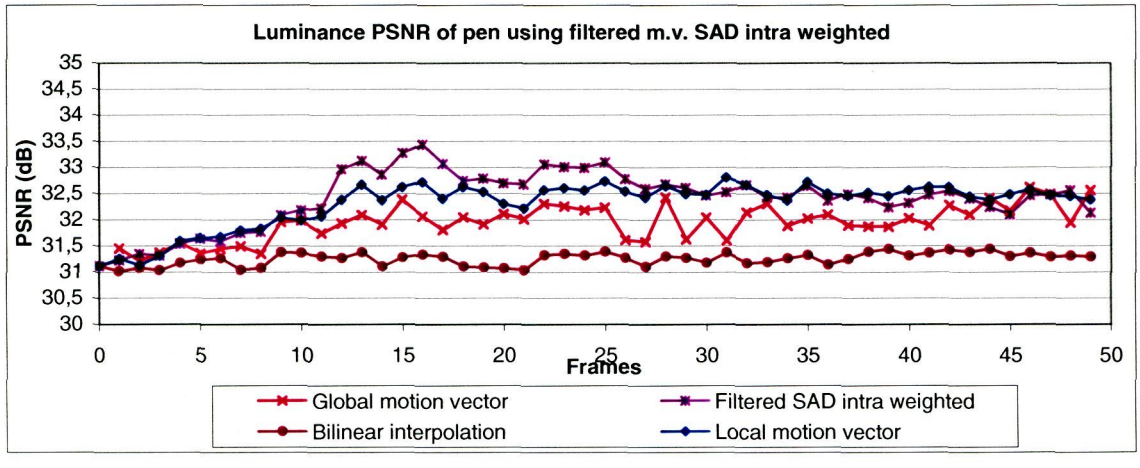


Figure 163 Results of 'pen' using filtered motion vectors using SAD_intra as weights.

- An application (named MATCH) that compares all the new images generated by mixing the new low-resolution image with the previous best match shifted, with the original high-resolution image. The output of this application is the image with best PSNR and we call it the best match. This image will be used by SHIFTS to process the new incoming frame.

With this setup, the same sequences were processed and the results are exposed in Figure 166 to Figure 168. We observed that our results used to be around 70% of the gap between bilinear interpolation and exhaustive search. In this case, 3DRS algorithm is used and sometimes the motion estimation is not perfect, but the results are not very far from the optimum value.

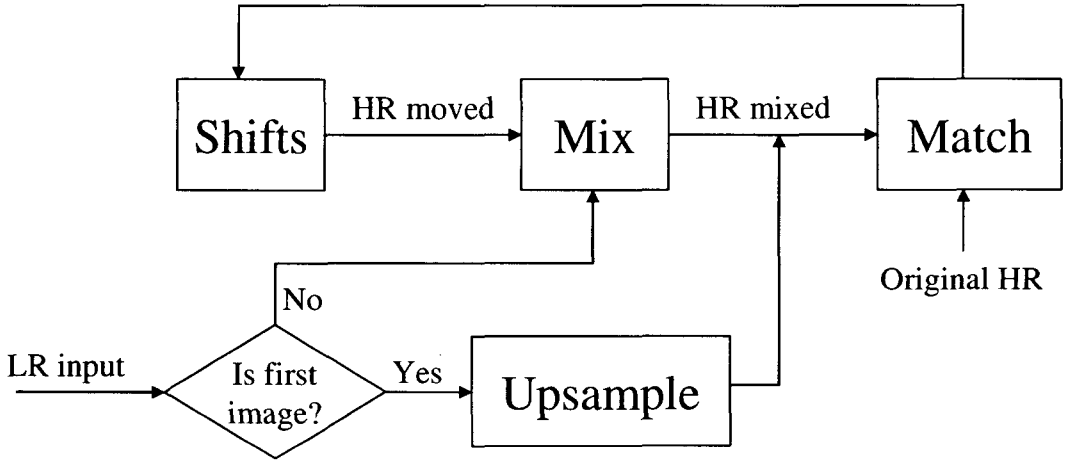


Figure 164. System to simulate the super resolution approach. In this system, an exhaustive search in the full field of global motion vectors is done.

5.3.4.4 HR versus LR motion estimation results

A consequence of the translation of the interpolation from the last step to the input is that the motion estimation can be easily performed in high resolution.

Since the high-resolution images contain the low-resolution images, the motion estimation can be easily performed in low resolution using only the original pixels from low-resolution images.

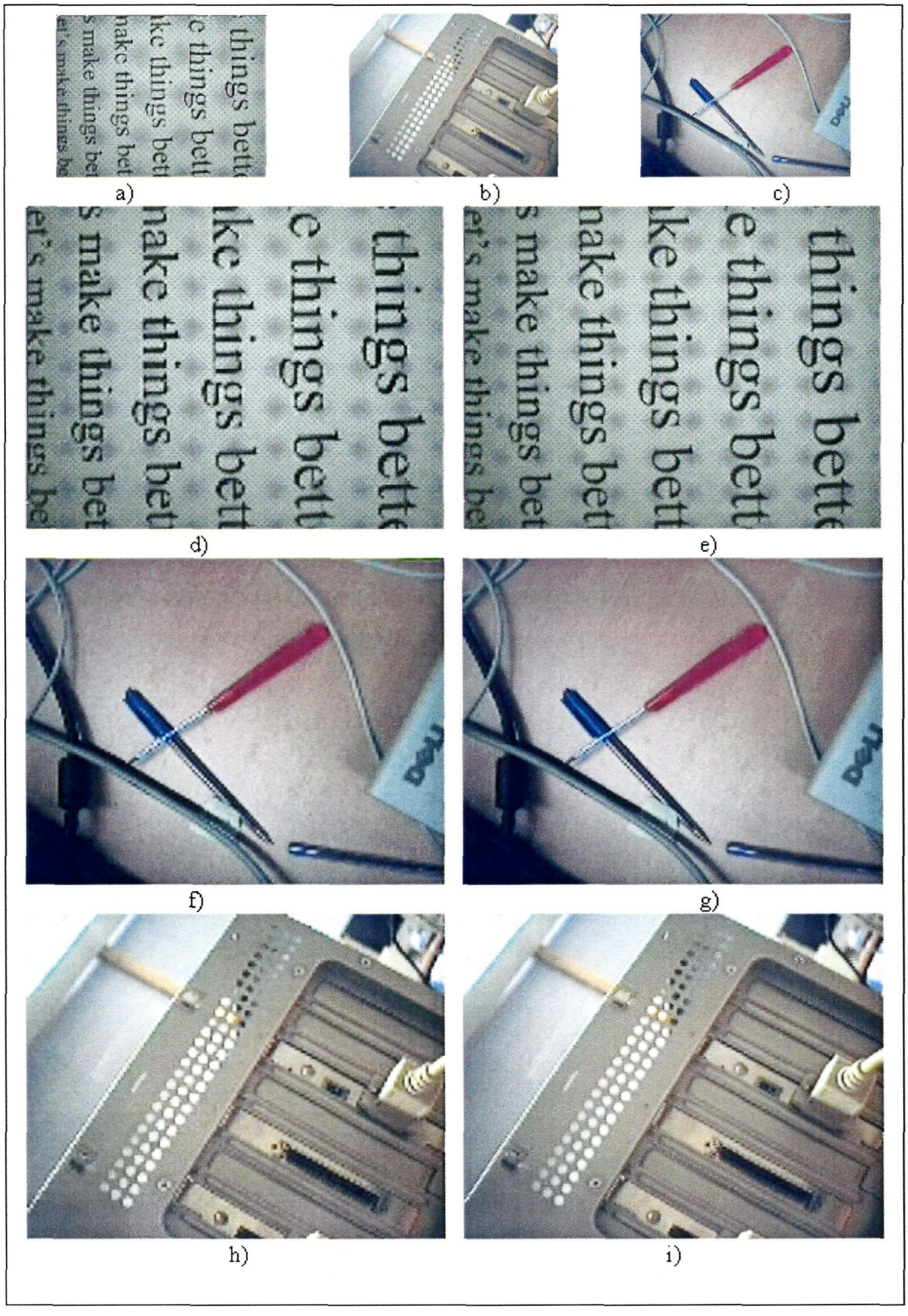


Figure 165. Results with the recorded sequences: a), b) and c) are low resolution images. d), f) and h) are results obtained using bilinear interpolation. e), g) and h) are super resolution images after 20 frames.

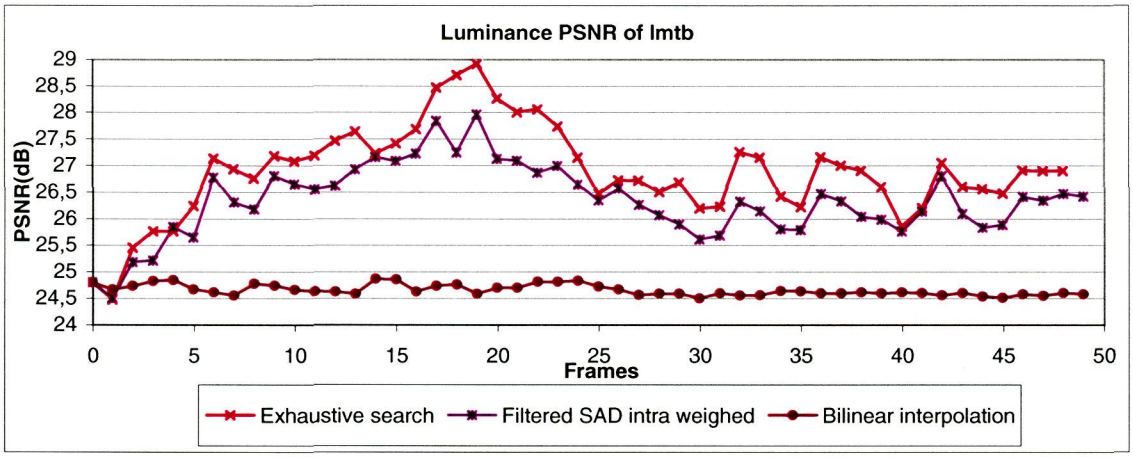


Figure 166. Results of 'lmtb' using exhaustive search.

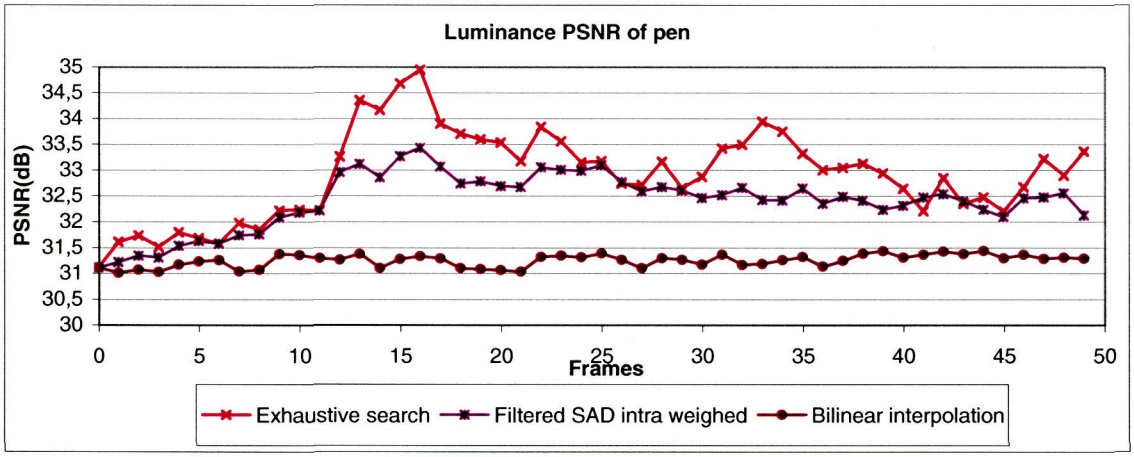


Figure 167. Results of 'pen' using exhaustive search.

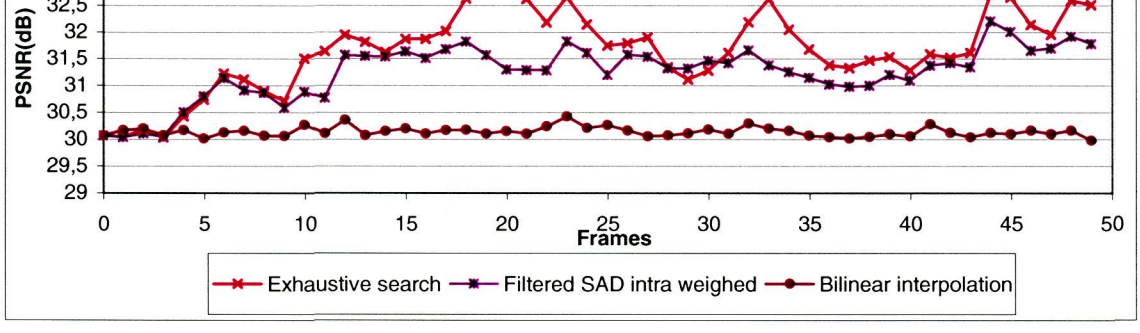


Figure 168. Results of 'sockets' using exhaustive search.

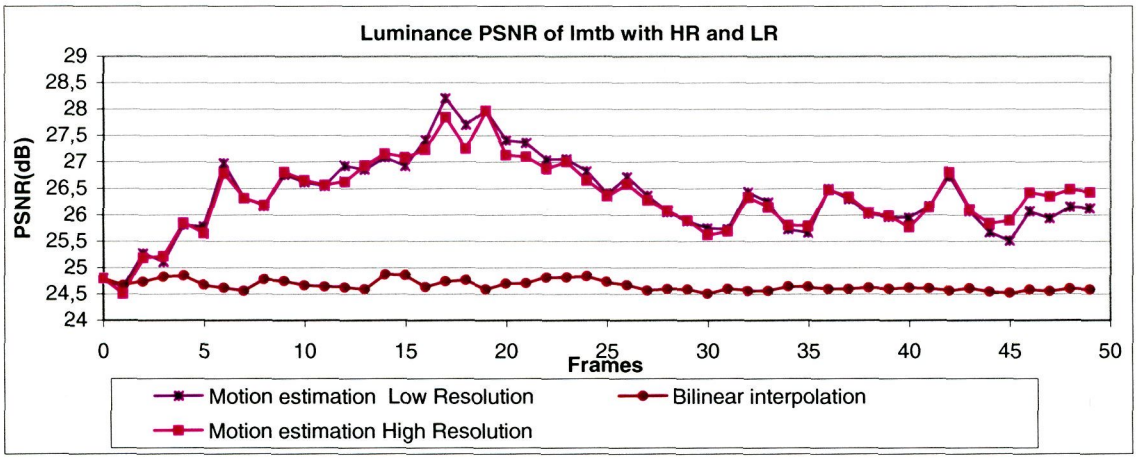


Figure 169. Results with high and low resolution motion estimation.

Some experiments were done to know the performance of motion estimation in low and high resolution. The results are shown in Figure 169 and Figure 170. As was explained before, this feature requires a trade-off between regularity and computing power. As regularity and no hardware changes have driven all the process, and the results are almost similar in both cases, motion estimation in high resolution has been chosen.

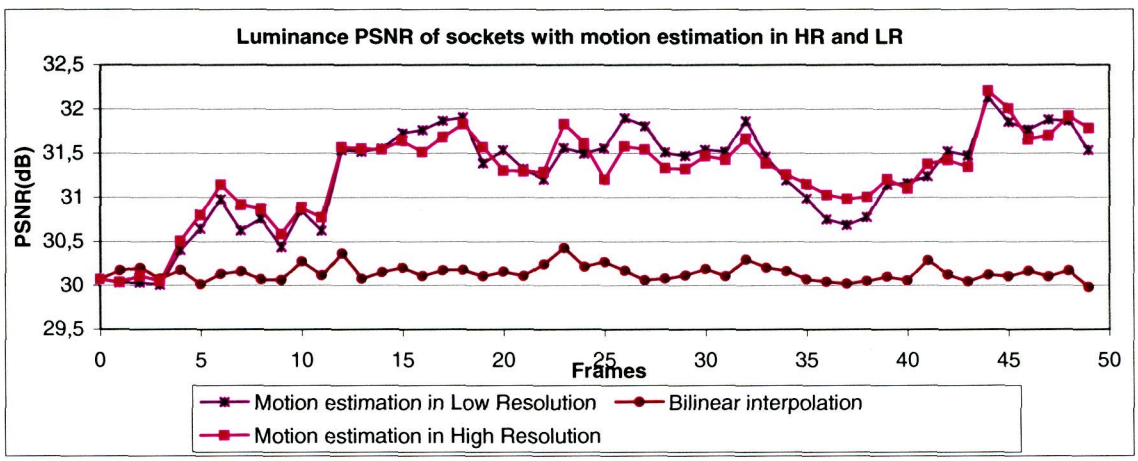


Figure 170. Results with high and low resolution motion estimation.

5.3.4.5 Quarter pixel versus half pixel results

In order to increase the accuracy in the estimation of the real motion between scenes, a new refinement step in the motion estimation was introduced. In this way, the 3DRS algorithm performs one search step plus three refinements steps to achieve quarter pixel accuracy, namely: a full-pixel refinement, a half-pixel refinement and a quarter-pixel refinement. This accuracy is increased when the motion estimation is performed in high resolution. Due to the increase of size, quarter-pixel motion estimation in high resolution is equivalent to one-eight pixel accuracy in low-resolution. When passing from high-resolution to low resolution the motion vectors must be divided by a factor of two. So, one-quarter precision motion vectors are equivalent to one-eight precision from the low-resolution point of view, where the input images came from.

Some experiments were done to check this feature. In these experiments, the motion estimation was done in high resolution, therefore multiplying by two the resolution of the vectors. In this case the motion estimator can detect motion with one-eight-pixel accuracy.

Two main experiments were carried out using the sequences 'lmtb' and 'sockets'. In the first experiment, the motion estimation was performed with half-pixel precision and quarter pixel precision. The results are shown in Figure 171 and Figure 172.

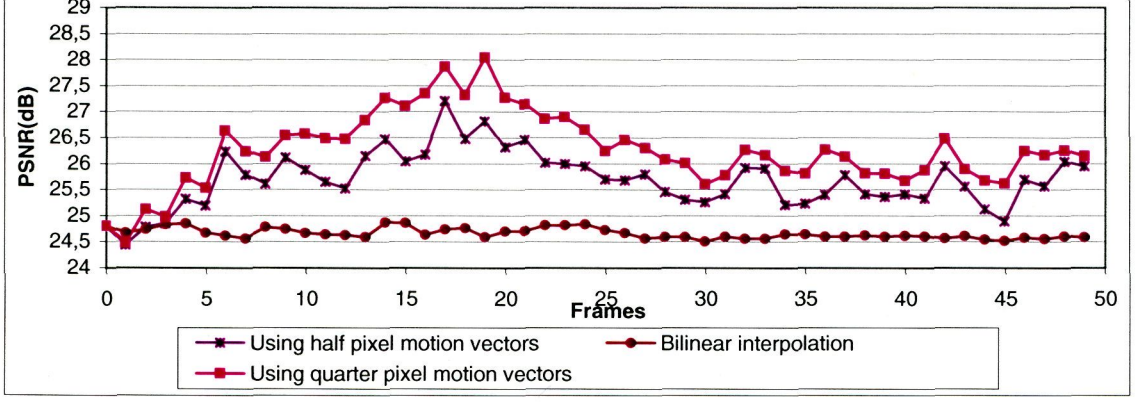


Figure 171. Results with quarter and half pixel precision motion estimation.

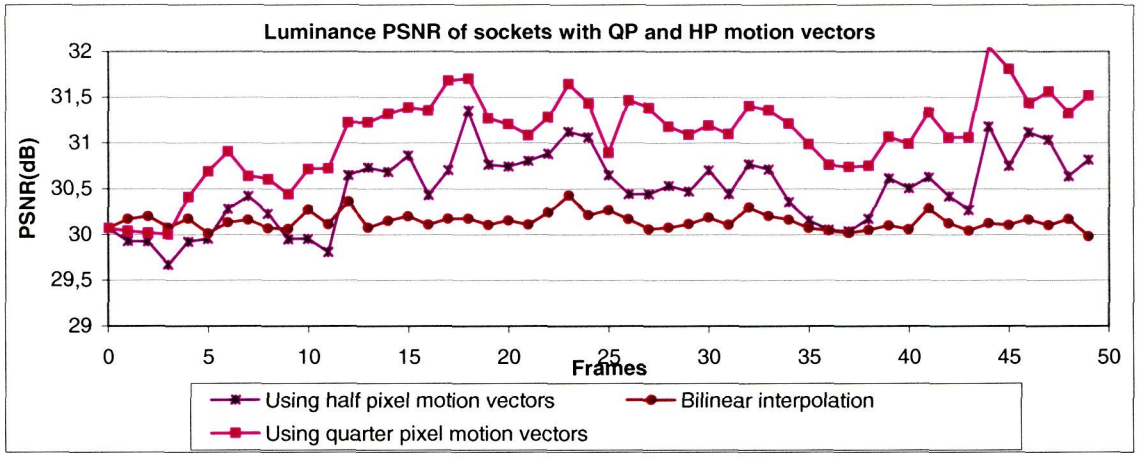


Figure 172. Results with quarter and half pixel precision motion estimation.

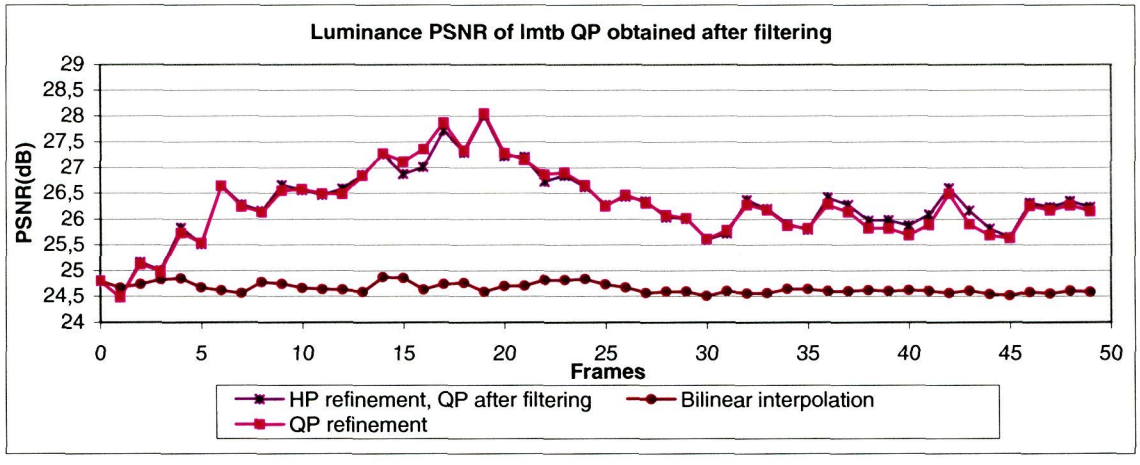


Figure 173. Results with quarter pixel precision motion estimation with two steps of refinement and with filtering.

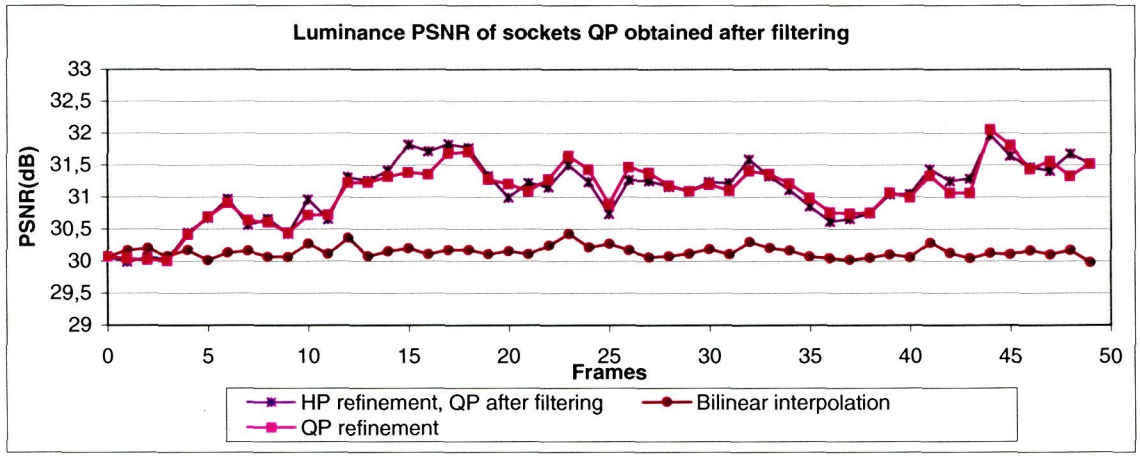


Figure 174. Results with quarter pixel precision motion estimation with two steps of refinement and with filtering.

Since after filtering the motion vectors it is possible to obtain motion vectors with quarter pixel precision avoiding the second refinement step, a second experiment to test this feature was carried out. In this case, half-pixel motion-vectors were filtered, allowing an artificial precision of one-quarter pixel. Results are shown in Figure 173 and Figure 174. Using this second technique results probed to be very similar as with quarter-pixel refinement but with a one refinement step less in the motion estimation.

5.4 Conclusions

The quality of the super-resolved images depends, among other factors, on the sampling of the input image. If the image set used to generate the super-resolved image has identical samples, the super-resolution process has less new information available, generating lower quality images.

When the input image sequence contains a great amount of aliasing, the motion estimation process is highly improved by previously performing a low-pass filtering.

Table 26 shows a summary of the qualities obtained for the different versions of the super-resolution algorithms. For the iterative algorithms the PSNR is computed after 8 iterations, as this number of iterations is a good trade-off among quality, quality drop in case of data missing (images of type 'b', 'c' or 'd') and computing effort. Chrominances exhibit higher PSNR due to their lower entropy, and version v1.3 (reference is the first frame) shows an improvement of 5 dB with respect to version v1.2 (reference is the average frame). Nevertheless, these results for the first versions strongly rely on the number of iterations and in the position of the input samples.

These iterative algorithms using only the resources founded in a hybrid video encoder suppose a first step to the target of a real-time super-resolution algorithm. Among the iterative algorithms developed, one set (version two) is intended for still images and the other set (version three) is for video sequences. Both sets are suitable to work in real time onto the hybrid video encoder addressed in this work. The first set is also known as the static super-resolution solution and the second one as the dynamic super-resolution solution.

Versión	Type	Average PSNR Luminance	Average PSNR Red chrominance	Average PSNR Blue chrominance
v1.2	Iterative. Static SR	23.19 dB	38.45 dB	40.79 dB
v1.3	Iterative. SR static	28.24 dB	38.24 dB	39.88 dB
v2.0	Non Iterative. SR static	30.47 dB	45.42 dB	48.77 dB
v2.1	Non Iterative. SR static	34.63 dB	46.90 dB	49.66 dB
v3.0	Non Iterative. SR dynamic	28.64 dB	44.22 dB	47.22 dB
v3.1	Non Iterative. SR dynamic	26.28 dB	41.28 dB	44.25 dB

Table 26. Summary of the PSNR for the most significant versions of the developed super-resolution algorithms.

Although the static super-resolution algorithms demonstrate a high image quality, they have the drawback of demanding a large amount of memory to store intermediate image results. Depending of the application, this can be a problem or not. For a photography environment, the increase in the quality can justify the memory increase, as the system will decrease the overall costs using lower resolution sensors.

These last sets of algorithms (version three) are intended for video and its functioning is based on a feed back of the last super-resolved image. As it does not gather information from various images, the pixels suffer from a temporal dilution process, producing a slight quality loss (< 2 dB) when compared to the previous versions. Nonetheless, the decrease in the memory required can compensate in many cases the quality losses, depending on the application and on the available resources. These kinds of algorithms exhibit a high independence of the metrics with respect to the border effect, whenever the contributions are used.

The memory requirements of the most significant super-resolution algorithms are presented in Table 27.

Size	mb_x	mb_y	v1.2 and v1.3 (Kbytes)	v2.0 (Kbytes)	v2.1 (Kbytes)	v3 (Kbytes)
SQCIF	8	6	315.19	450.19	459.05	36.01
QAVGA	9	7	413.68	590.87	598.56	40.51
QCIF	11	9	650.07	928.51	931.60	49.51
HAVGA	18	14	1,654.73	2,363.48	2,331.25	81.03
CIF	22	18	2,600.30	3,714.05	3,645.39	99.05
AVGA	36	28	6,618.94	9,453.94	9,198.98	162.12
VGA	40	30	7,879.69	11,254.69	10,936.17	180.15
4CIF	44	36	10,401.19	14,856.19	14,419.55	198.19
16CIF	88	72	41,604.75	59,424.75	57,354.19	396.77

Table 27. Comparative table of the memory requirements for the most significant versions of the developed super-resolution algorithms.

A new super-resolution algorithm version v3.1 is presented, which is more suitable for low-cost implementations onto video encoders based on some simplifications of version v3.0. This last algorithm not only has shown good performance with artificial sequences but also with real sequences.

Last but not less, the algorithm has been tested with video sequences where the movement is real, but the aliasing has been artificially produced. One of the key factors of this algorithm when dealing with real sequences is the motion estimation. Several options have been evaluated, finally adopting a solution based on filtering of the motion-vector, weighted with the SAD_INTRA as a measure of the high frequencies amount, and therefore of the goodness of the estimated motion. This solution shows an excellent behavioral in many real scenes. An exhaustive motion search solution has also been assessed and both solutions probe to be quite similar.

Finally, the influence of some parameters on the super-resolved quality, as the image resolution for the motion estimation or the precision of the motion vectors, has been analyzed. The choice has been to perform the motion estimation in high-resolution and with a ¼ pixel precision as obtained from the filtering process.

Chapter 6

Results! Because, man, I have gotten a lot of results, I know several thousand things that won't work.

Thomas A. Edison (1847 - 1931)

Results

6.1 Introduction

Among the different results of this work, the final architecture as well as some applications of super-resolution can be highlighted. These applications have been tested over real sequences taken from a prototyping system. In these sequences, both the movements and the aliasing are provided by the capture system, as in a real application case.

6.2 Final architecture

The final architecture is based in the initial proposal, where some additional changes have been added to support super-resolution for video sequences. Some hardware coprocessors have been modified and also some software tasks have been added. The main rule followed in the entire mapping process to obtain this architecture was to avoid as many changes in hardware as possible and to try to implement super-resolution as a software plug-in. This main rule also had to consider the ability to reach real time. As a consequence of the established, the final architecture has very small changes in hardware coprocessors and most of the work is done by software executed in the general processor.

The hardware coprocessors that have been changed are:

- The input coprocessor.
- The motion estimator.
- The texture codec.

Each ones of these hardware changes, as well as the software ones, are explained in next sections.

6.2.1 Input coprocessor

The input coprocessor was changed in order to support the bilinear interpolation of the input image. The input processor reads only one quarter of the input image from the capture system and then, puts in its output an image four times larger as a result of the bilinear interpolation of the input. As the original input processor, it works stripe by stripe composed by 16 image lines. Because it performs a bilinear interpolation, it reads 8 lines at the input and writes 16 lines in the output. The amount of pixels per line depends of the image size, but it also doubles the amount of pixels per line. The process followed to perform this interpolation was already explained in section 0.

6.2.2 Motion estimator

The motion estimator hardware block has been changed by adding an anti-aliasing filter, with the transfer function shown in Figure 175. The objective of this filter is to perform a low pass filtering of each block in the pixel domain before to evaluate it. By this way, the amount of aliasing is reduced and then better motion vectors are obtained.

6.2.3 Texture codec

Texture codec coprocessor is responsible to carry out most of the operations in the compression algorithm. In super-resolution, it is responsible to do the motion compensation, the mixing of both images and the compression. To do this mixing, it is necessary to perform

some minor modifications in the compressor hardware. Then, when the motion compensation is done, the output image will be the previous motion compensated image updated with pixels of the new image. These hardware modifications include the updating into the motion compensation process. In Figure 23 the tasks performed in the texture codec are shown.

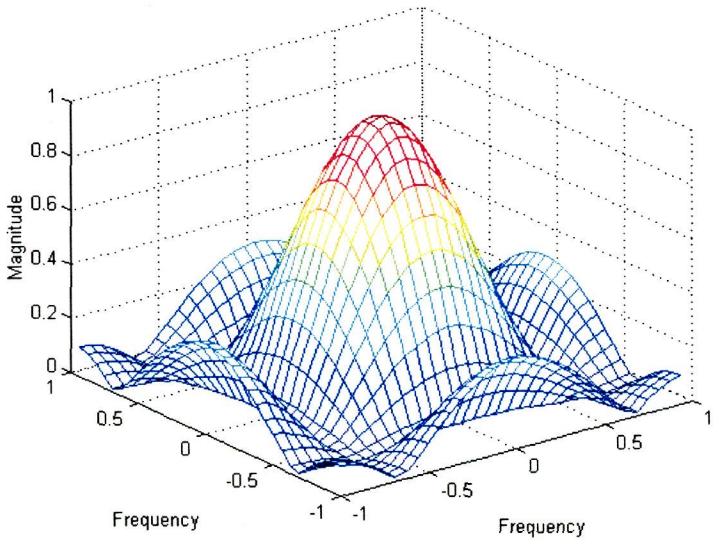


Figure 175. Impulse response of anti-aliasing filter.

6.2.4 General processor software

As it was discussed in section 5.3.4.3.2, the final taken approach was to filter the motion vectors. This filtering is performed by software, keeping the motion vectors as software variables. This software performs the following operations:

- Get both the motion vectors and the SAD_INTRA from the motion estimator coprocessor.
- Keep in memory these data.
- Perform the filtering in the motion vectors field, obtaining a new set of motion vectors.

In all the processing, only the lossless part of the hybrid video encoder is used. The motion compensated image is passed to the inverse motion compensation (IMC) and then stored in memory. Since the general processor is responsible for the communication, this feature is taken into account by the software running in the general processor. It is important to note that the DCT, Q, IQ and IDCT remove the higher frequencies in the image, and so,

they should be avoided. These blocks are only used to compress the image, but are not kept into the loop.

6.3 Applications

In this section, the performances of super-resolution algorithms are tested in real situations. As it was shown in the previous section, the quality improvement depends on two factors:

- Movement in the sequence.
- Presence of aliasing in the image.

The first factor was tested in the section 0. In this section, a method to obtain the necessary shifts was described and also the system was optimized to support real movements. The next step is to obtain aliasing in the image without any artificial system. This aliasing is obtained from the capture system directly. In the next sections, a common image capture system will be described. How to obtain the necessary aliasing and new approaches to take advantage of that feature will also be discussed.

One important point to take into account at this moment is the lack of a reference image. In previous steps, an original image as a reference was always considered. At this point, as the aliasing is obtained from the capture system, there is not a reference. Therefore, all the measurements about the quality and the improvement obtained using super resolution are subjective.

6.3.1 Capture system

A common capture system is shown in Figure 176. In front of the system, there is a lens to focus the scene on the sensor plane. This plane is commonly called image plane. The most common technique to obtain a color image is to sense red, green and blue colors. In the high quality products, three sensors are placed, one for each of these colors. Using some mechanical or optical system, the image plane is focused on the three sensors, obtaining three colors with the sensor resolution.

In cheaper applications, only one sensor is used. In that case, in order to obtain three colors a Color Filter Array (CFA) is placed in front of the sensor. By this way, each sensor cell senses a different color. The most commonly used is the RGB Bayern pattern, which is represented in the Figure 176.

After sensing the colors, a color reconstruction algorithm is applied to obtain the colors where they are not sensed. There are several color reconstruction algorithms [PKT83], [Jai89], [PRA91], most of them based on different types of interpolation. After the color reconstruction, another image processing is performed to obtain the definitive YUV signal, which is the input of the compression system.

In order to obtain the YUV 4:2:0 signal, the following operations are done:

- To calculate the YUV 4:4:4 signal as:
  - $Y = 0.3 \cdot R + 0.59 \cdot G + 0.11 \cdot B$.
  - $U = Y - R$
  - $V = Y - B$
- To perform a low pass filtering on U and V signal.
- Sub-sample the U and V signals to obtain the YUV 4:2:0 format.

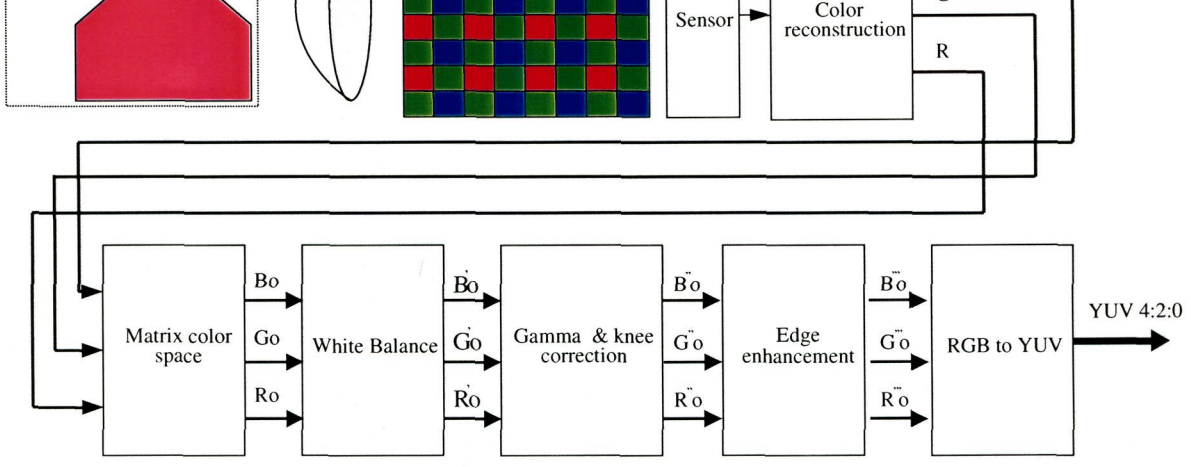


Figure 176. Capture system for imaging.

In order to obtain aliasing, three points are crucial:

- Lens.
- Sensor configuration.
- Color reconstruction algorithm.

In this study, some sequences were recorded using a high quality lens. The resolution of this lens was selected to be higher than the sensor resolution. By this way, we try to avoid the lens optical low-pass filtering.

The aliasing appears in the sampling process. In this process, there are two main factors:

- Sensor fill factor.
- Color sampling.

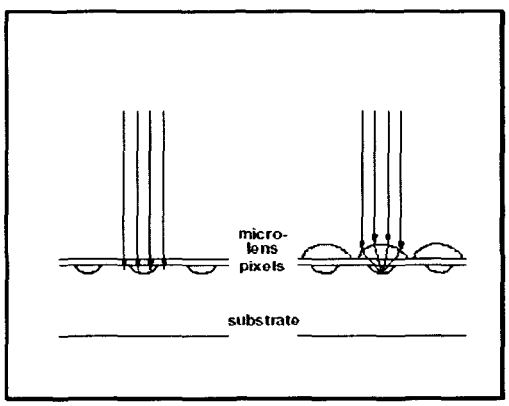


Figure 177. Micro lenses disposition.

The first one describes the ratio between the light sensitive area per pixel and the total pixel area. In the new CMOS active pixel sensors, called SeeMOS, this factor can be as low as 25%. That means that only one quarter of the pixel is light-sensitive. This factor is increased using a small micro-lens in top of each pixel. With these lenses, some of the incident light in the non-sensitive area is concentrated to the light-sensitive area. Those micro lenses can increase the fill factor until 60 or 70%, but still remain some amount of aliasing.

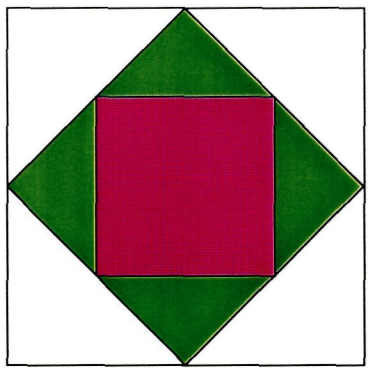


Figure 178. Color-sampled frequencies in the frequency space.

The second factor is the color sampling. In fact, each color signal is a reconstructed signal. The maximum sampled frequency is represented in Figure 178. The axes of this figure are the horizontal and vertical space frequencies, with the low frequencies located in the middle of the square. The color squares represent the maximum sampled frequency for each color: the bigger square with diamond shape represents the frequencies of the green color and the smaller inner squares represent the frequencies of the red and blue colors. It is important to note that the green signal has higher frequencies than red and blue ones. This limitation in the frequency distribution produces some aliasing in the color spectral domain.

In the experiments carried out, the sensor used was the UPA1021 chip sensor, which has the previously described characteristics.

6.3.2 Resolution enhancement and electronic zoom

The first experiment consisted in the use of the signal in YUV 4:2:0 as in previous steps. In this case, the signal follows all the processing, but we choose different algorithms to reconstruct the color signal. This processing must keep as much aliasing as possible in the luminance signal, to obtain a good performance in super-resolution. Three algorithms were tried:

- Standard reconstruction
- SmartGreen1
- SmartGreen3

Standard reconstruction algorithms perform a bilinear interpolation of each color. This reconstruction performs a strong low pass filtering, removing most of the aliasing present in the luminance signal. Super resolution algorithm was applied using the obtained signal in YUV 4:2:0. The results are shown in Figure 179 to Figure 182. After twenty frames, only a small improvement is achieved due to the limited aliasing presence in the luminance signal.

SmartGreen1 performs a bilinear interpolation over the red and blue signal. If high frequencies are present, the green signal is guessed from the red or blue signal in places where there are no green samples. In those places, the green signal is equal to the red or blue sampled signal multiplied by some predefined coefficients. If this approach works well, the green signal can achieve pixel resolution, and therefore, increases the luminance resolution. This algorithm works better in gray areas, where R=G=B. SmartGreen1 seems to keep more aliasing information, and we expect better performance. Results are shown in the Figure 183 to Figure 186. Nevertheless, no important quality improvement was achieved using this reconstruction technique.

The last algorithm used to reconstruct the color space was the SmartGreen3. This algorithm requires more computational power. In previous versions of SmartGreen, the computed green signal from the red and blue values was used directly. This approach works well in gray areas but made colorful images grayer. In SmartGreen3, this signal is only used to enhance the edges and to help in the detection of false color due to color aliasing. The main problem with this algorithm is that it performs a band-pass filtering of the luminance signal, avoiding any option of aliasing. Then, we will not increase the quality of the image using this algorithm in conjunction with super-resolution. Due to the good performance of this technique, it can be a reference of sharpness for SR as one of the best qualities that the color reconstruction algorithms can achieve. The results obtained using these algorithms are shown in the Figure 187 to Figure 190.

The performance obtained using standard reconstruction or SmartGreen1 with super resolution are not better than using SmartGreen3. Moreover, the improvement of using super-resolution in conjunction with SmartGreen3 is very small, so it is not very interesting to apply super resolution in that case.

Figure 179. Standard reconstruction and zoom with super resolution of 'paper', frame 1. For the first frame, no SR is possible and so it is equal to bilinear interpolation.

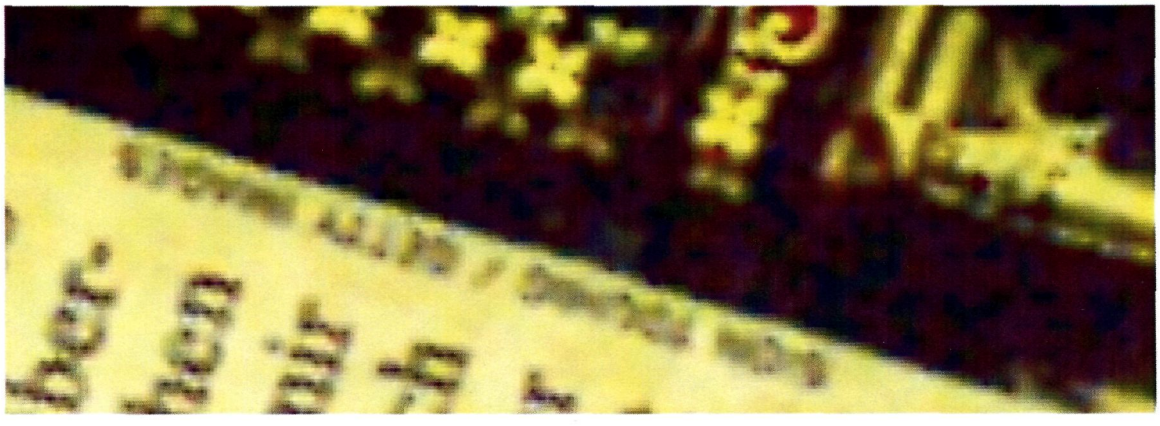


Figure 180. Standard reconstruction and zoom with super resolution of 'paper', frame 1 detail. For the first frame, no SR is possible and so it is equal to bilinear interpolation.

Figure 181. Standard reconstruction and zoom with super resolution of 'paper'. Frame 20.

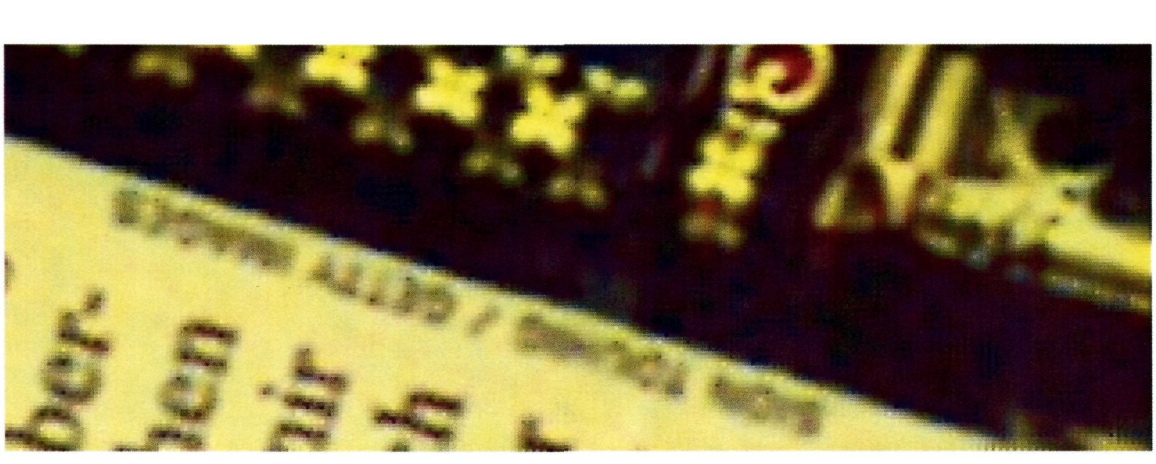


Figure 182.Standard reconstruction and zoom with super resolution of 'paper'. Frame 20 detail.

Figure 183. SmartGreen1 reconstruction and zoom with super resolution of 'paper', frame 1. For the first frame, no SR is possible and so it is equal to bilinear interpolation.

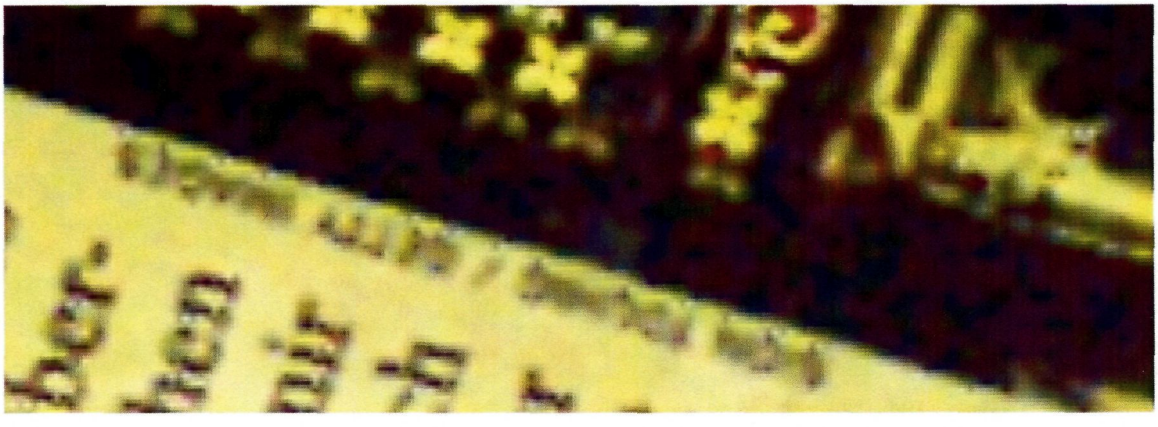


Figure 184. SmartGreen1 reconstruction and zoom with super resolution of 'paper', frame 1 detail. For the first frame, no SR is possible and so it is equal to bilinear interpolation.

Figure 185. SmartGreen1 reconstruction and zoom with super resolution of 'paper'. Frame 20.

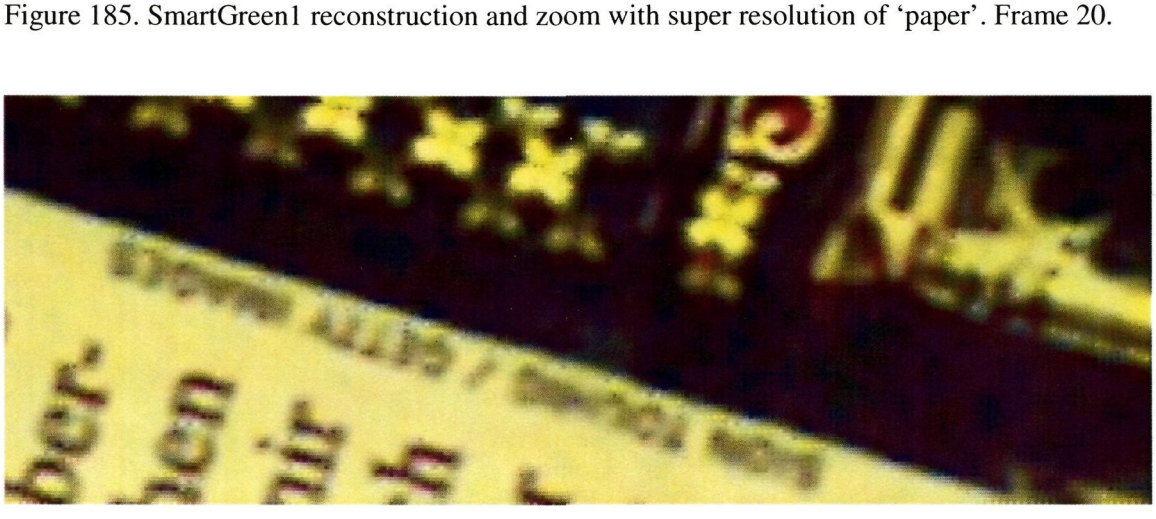


Figure 186. SmartGreen1 reconstruction and zoom with super resolution of 'paper'. Frame 20 detail.

Figure 187. SmartGreen3 reconstruction and zoom with super resolution of 'paper', frame 1. For the first frame, no SR is possible and so it is equal to bilinear interpolation.

Figure 188. SmartGreen3 reconstruction and zoom with super resolution of 'paper', frame 1 detail. For the first frame, no SR is possible and so it is equal to bilinear interpolation.

Figure 189. SmartGreen3 reconstruction and zoom with super resolution of 'paper'. Frame 20.

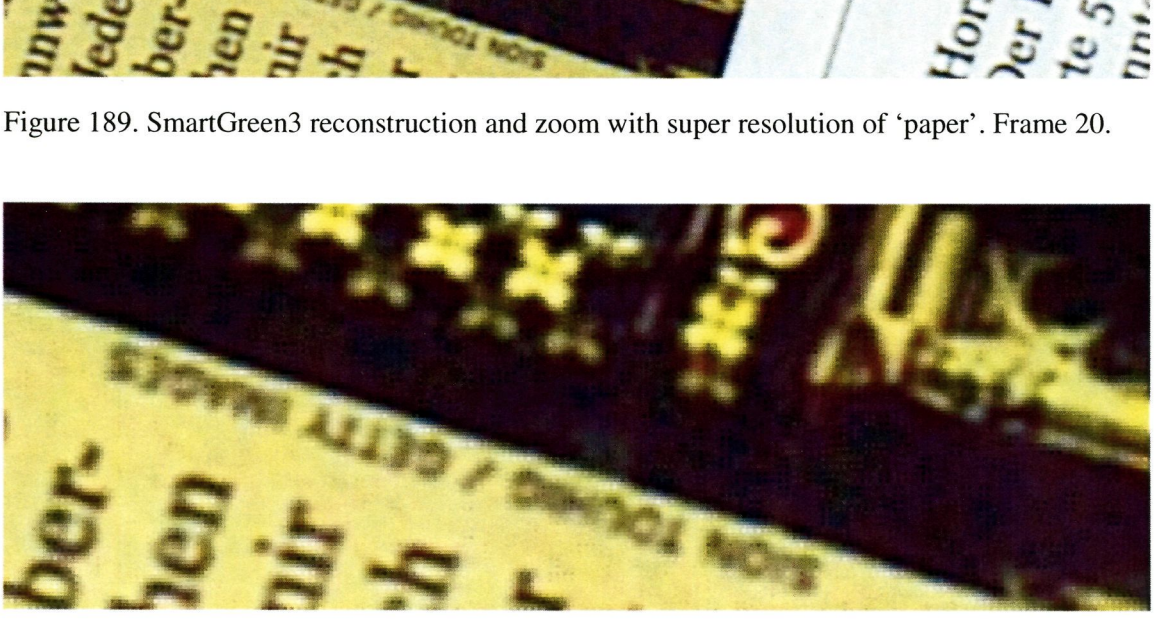


Figure 190. SmartGreen3 reconstruction and zoom with super resolution of 'paper'. Frame 20 detail.

6.3.3 Colour reconstruction

As the results obtained were not very satisfactory using the previous approaches, a new application was tried. Since each color signal is sub-sampled, it is easier to achieve aliasing in each one of these signals. In fact, the luminance signal is never a full sensed signal, because it is composed by sampled values of one color and reconstructed values from the other two color signals. Because the sensor works in RGB, it is better to work directly in this domain.

Following this idea, it can be seen that the way that the sensor works is similar to our approach to generate aliasing in previous steps. Indeed, the red and blue signal are sub-sampled by a factor of two in both directions, and the green signal is sub-sampled, but loosing only half of the samples.

Therefore, the idea is to reconstruct each color signal as color reconstruction algorithms do, but obtaining pixel resolution using super resolution. This idea is exposed in Figure 191. In this application, each color signal is treated as the luminance signal was before, but the motion estimation is done based only on the green values. In this way, super-resolution can substitute the color reconstruction algorithm.

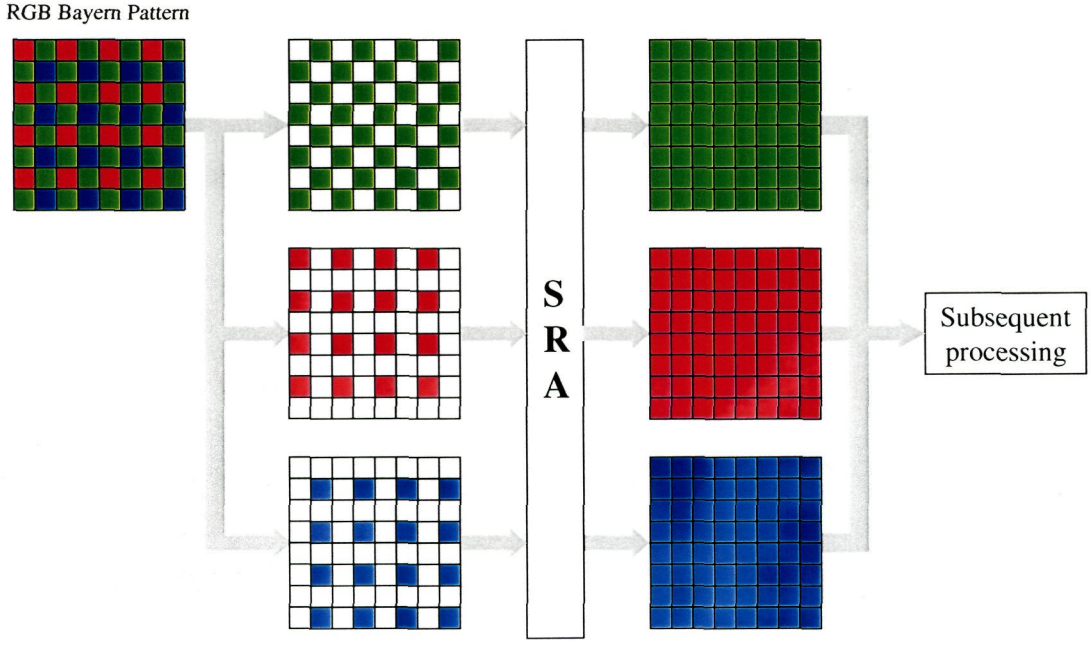


Figure 191. Color reconstruction with SRA.

The first image in the sequence is equivalent to standard reconstruction, but when new images come to the system, the quality increases and good results are obtained.

To avoid a great amount of calculations, the motion vectors are computed over the green values and the same motion vectors are applied to the red and blue matrixes. Another reason to do it, is that different motion vectors for different colors can produce color aliasing due to different calculated movements of each color. For this reason, the motion vectors must be coherent over the three color planes.

In this case, super-resolution achieves good results. The images obtained are comparable with SmartGreen3 ones, but there are some problems left:

- More than one image is needed to increase the quality.
- Errors in the motion estimation can produce bad results. The motion estimation algorithm must be very accurate, or it has to be used only in certain scenes.
- A lot of calculations are needed to perform the motion estimation.

On the other hand, the most important advantages obtained are:

- It works better in colorful images, while SmartGreen has a lot of problems, because this last one shift the false colors to the gray scale.
- In case of good motion estimation, the color aliasing is better suppressed.

Also another approach was tried, where the SRA is applied after the overall processing in the image. This processing increases the differences between the values in the RGB domain. For example, the matrix correcting process and the gamma correction provide more differentiate values. This can be useful for the motion estimator to perform a better motion estimation. The proposed process in this case is to follow the next steps:

- To reconstruct the color using standard reconstruction algorithm, what keeps the original values. It is necessary because other processing need R, G and B data in all the positions.
- To do all the processing (matrix correction, gamma correction, white balance, etc.)
- To take the sampled values from the original positions. These values are only modified by the previous processing and not from the color reconstruction algorithms. Using these values, we carry out the super resolution algorithm and reconstruct the lost colors.

With this approach, the obtained results are shown from Figure 192 to Figure 194. As it usual in this kind of processing, where no reference image is available, the results are only qualitative, but in Figure 194 a better color reconstruction is appreciated, with less color aliasing in all the colors, but in special in the green color, which is less shifted to the gray scales.

Figure 192. Standard color reconstruction.

6.3.4 Colour reconstruction and resolution enhancement

Following the line of the last two applications of sections 6.3.2 and 6.3.3, it is possible to join both of them in a single one. The basic idea is to increase the pixel resolution, but using each color signal instead of the luminance signal.

In section 6.3.1 was shown that there are two main sources of aliasing: the color sampling and the fill factor of the sensor. Using this idea, we can exploit both sources of aliasing in a new application.

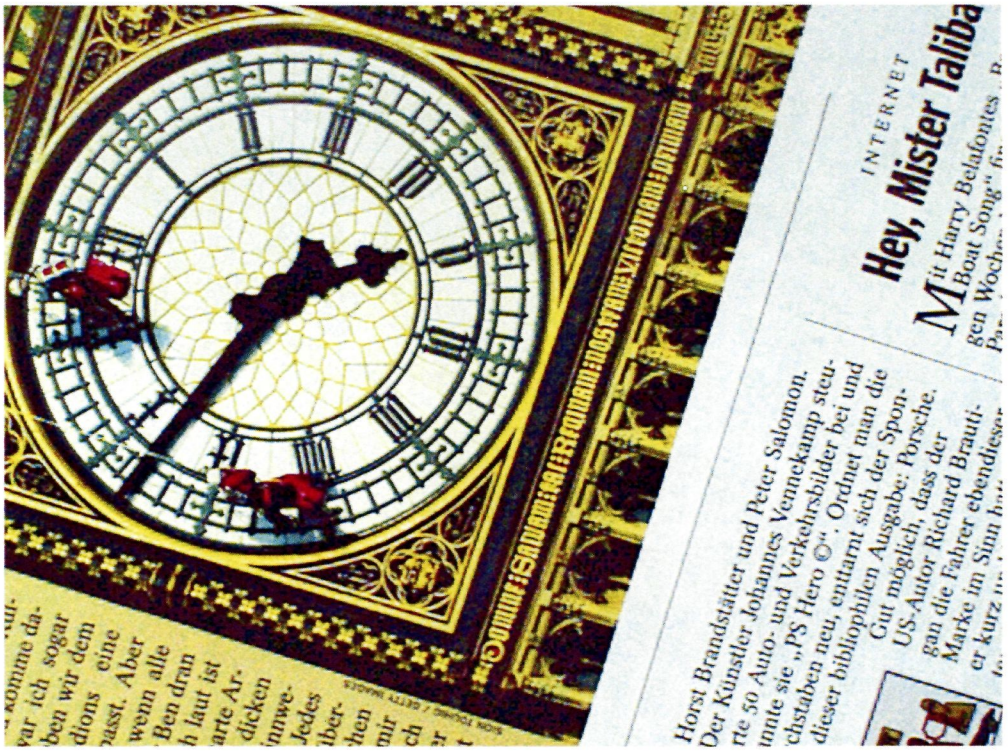


Figure 193. SmartGreen3 color reconstruction with false color detection and edge enhancement.

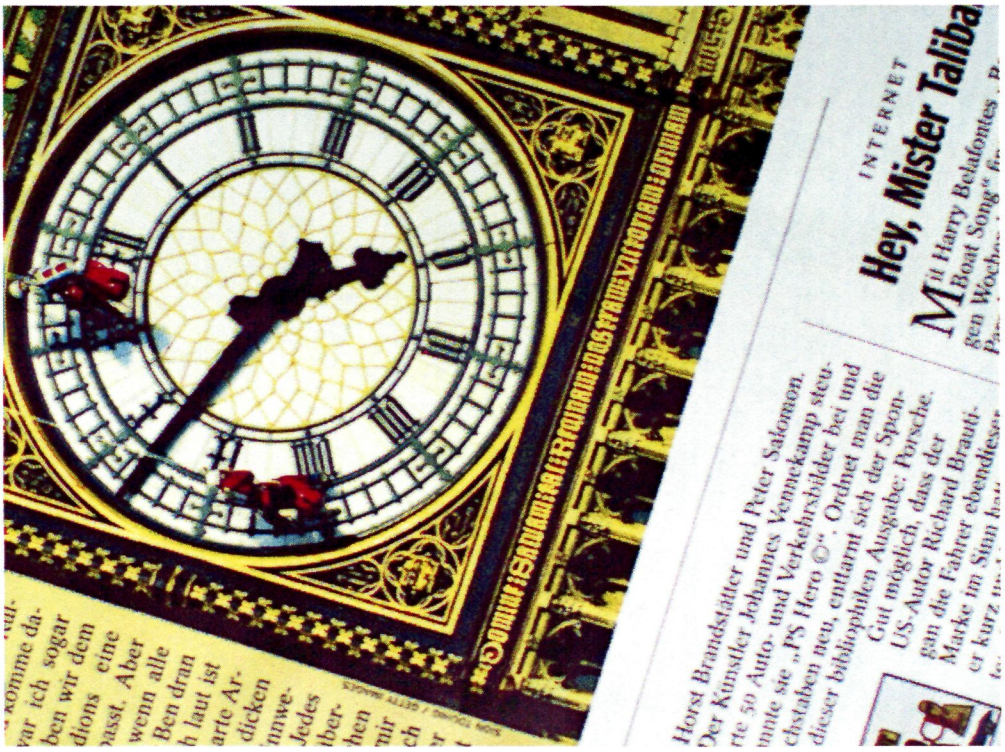


Figure 194. Super resolution color reconstruction.

We will apply the idea exposed in the Figure 195. With this application, we reconstruct 16 pixels from two green values and from one blue and red value. Therefore, more images will be necessary to obtain a good quality, 8 in the ideal case.

The processing followed in this case consisted in:

- To reconstruct the color using a standard reconstruction algorithm, keeping the original values.
- To perform all the other processing (matrix correction, gamma correction, white balance, etc.)
- To take the sampled values from the original positions. These values are only modified by the previous processing and not from the color reconstruction algorithms.
- To apply super-resolution, but using a 4x4 matrix instead of a 2x2 as in the previous cases.

In this case, the motion vectors are also calculated using the green values and reused for the other color matrixes.

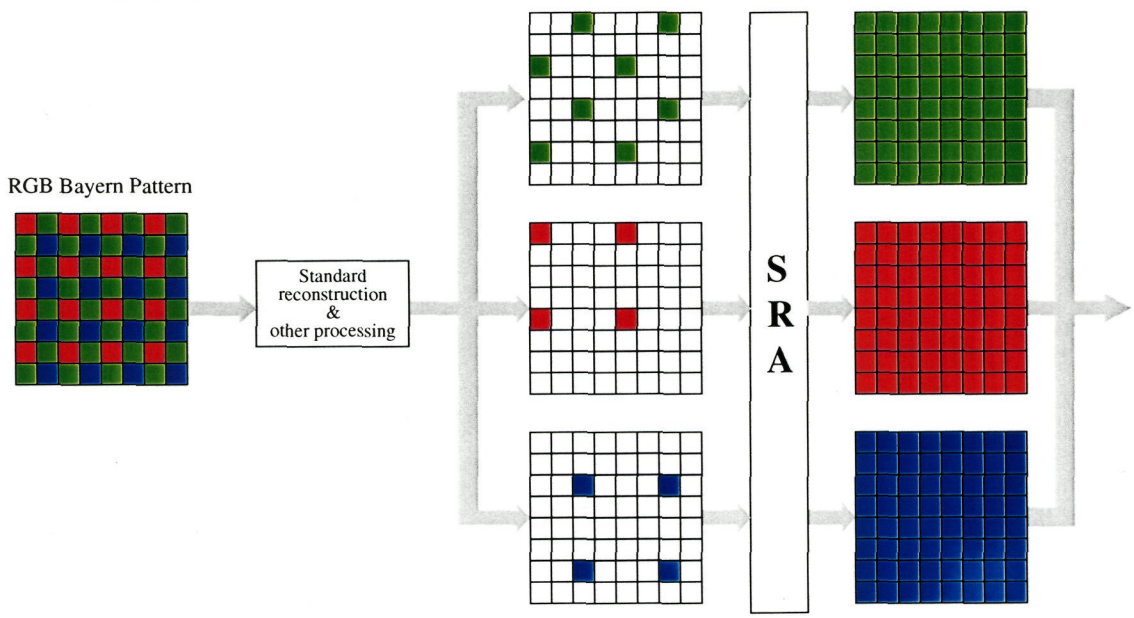


Figure 195. Color reconstruction and zoom using super-resolution.

Some experiments were developed, obtaining the results shown in Figure 198 and Figure 201. The usual way to obtain these images is to first perform a color reconstruction algorithm followed by a bilinear interpolation. This approach was followed in the images of

Figure 196, Figure 197, Figure 199 and Figure 200. These images serve us as a subjective reference to compare against.

The obtained results seem quite good, also some advantages and disadvantages can be found in this approach. The main disadvantages are:

- More than one image is needed to obtain good quality. Nevertheless, this is not a problem for video sequences.
- Bad motion estimation can reduce its quality to bilinear interpolation quality, worse than SmartGreen3 quality.
- It works in RGB data, and it is still necessary to convert to YUV format.
- To suppress the aliasing is a must when super resolution is not done.

Also some advantages are appreciated:

- Better resolution and suppression of color aliasing, achieving a good electronic zooming.
- Reuse of the existing hardware in combination with video compression.
- Take advantage of undesirable characteristics as aliasing and hand-motion.
- Use of signal processing instead of mechanical parts to perform the zoom, contributing in this way to reduce the overall device power dissipation.

6.4 Conclusions

In this chapter the modifications needed to be performed on the hardware coprocessors jointly with some super-resolution applications has been shown. Most of the final modifications have been made in the software running on the ARM7 processor and in the input processor in order to perform the bilinear interpolation.

Several applications have been addressed using a real acquisition system. A study of the system has been carried out, analyzing the processing flow in order to identify the possible sources of aliasing. An important problem in this stage is the absence of a reference image to be used in order to obtain quantitative metrics.

Figure 196. Standard color reconstruction and bilinear interpolation.

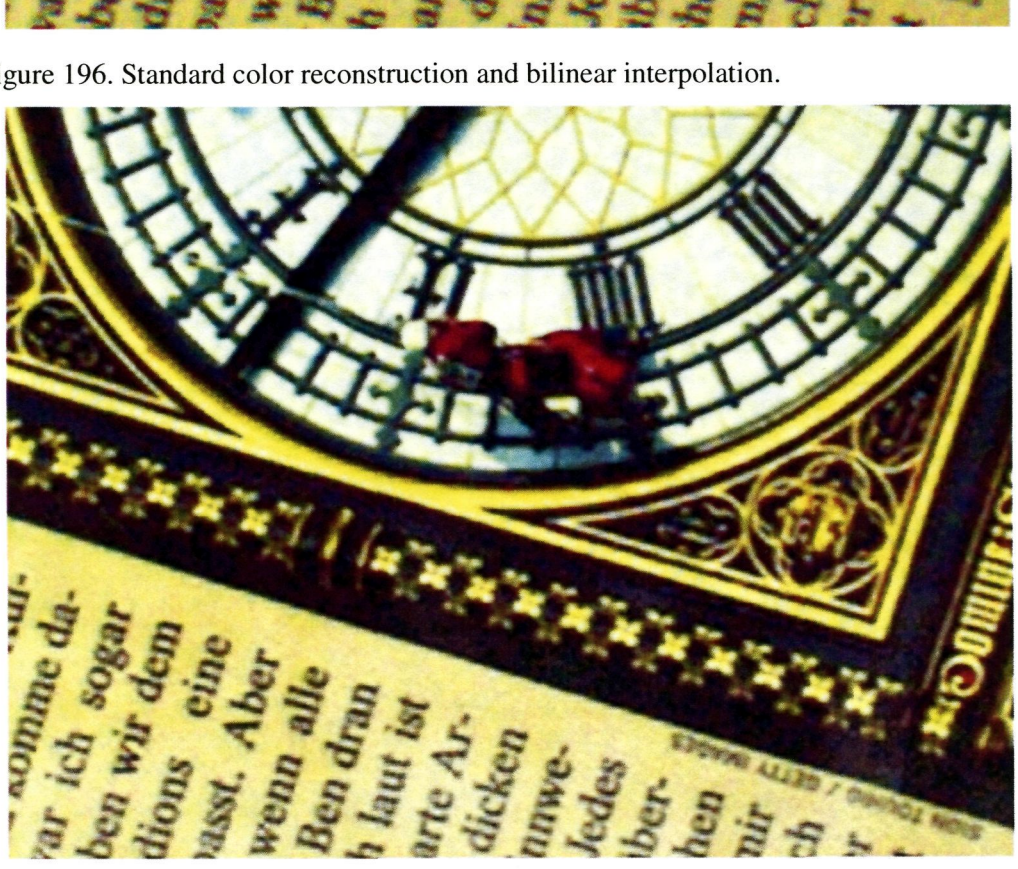


Figure 197. SmartGreen3 color reconstruction followed by false color detection and bilinear interpolation.

Figure 198. Super resolution reconstruction and zoom.

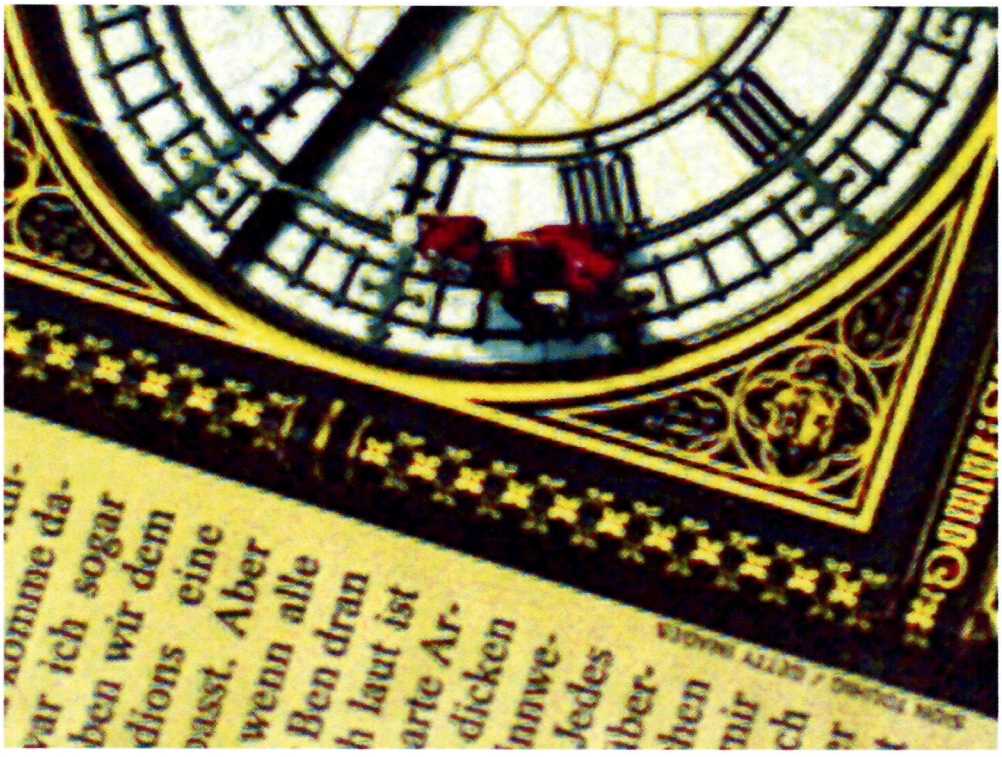


Figure 199. Standard color reconstruction followed by edge enhancement and bilinear interpolation.

Figure 200. SmartGreen3 color reconstruction followed by false color detection and edge enhancement and bilinear interpolation.

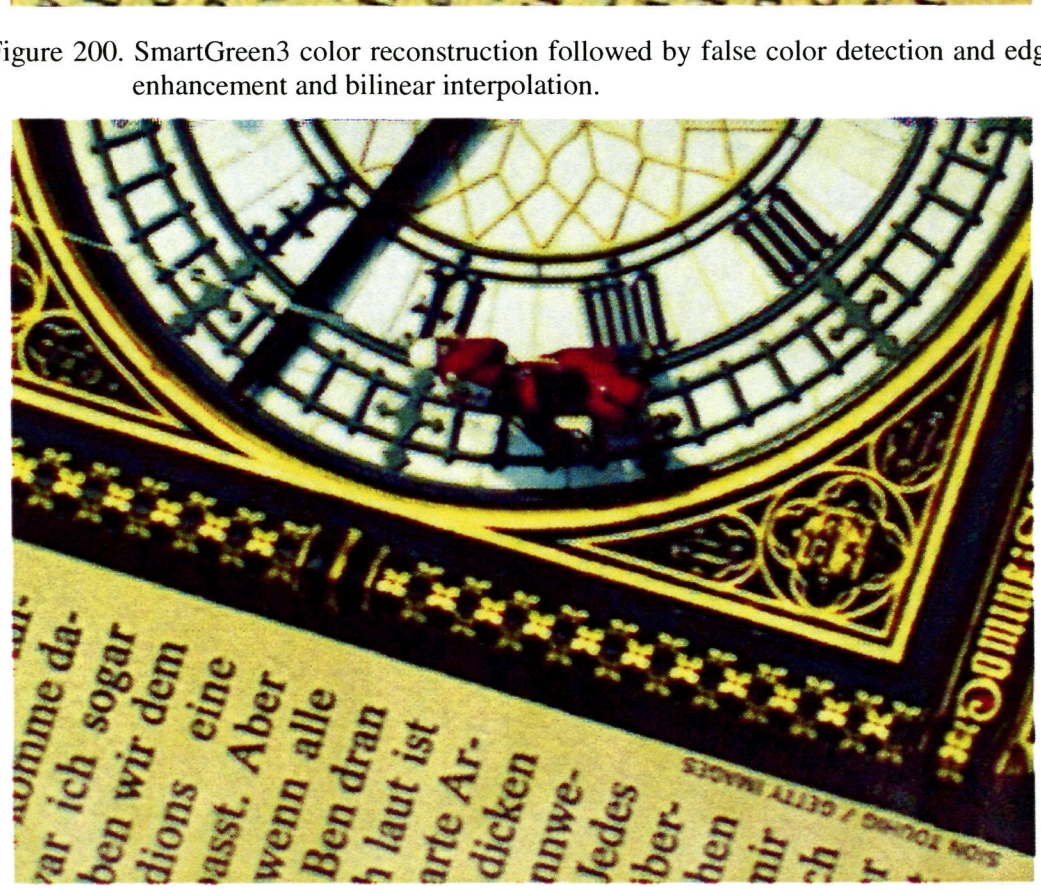


Figure 201. Super resolution reconstruction and zoom and edge enhancement.

Three main applications related with electronic cameras and the image processing and acquisition chain have been addressed:

- **Resolution enhancement and electronic zoom**, where three available color reconstruction algorithms have been studied in order to keep the aliasing at the input. As these algorithms do not allow passing a great amount of aliasing, the super-resolution enhancement is not very noticeable.
- **Color reconstruction**, where it is exposed how to apply super-resolution as a way to reconstruct the missing colors when a single sensor is used with a CFA.
- **Color reconstruction and electronic zoom**, where the two previous applications are combined. In this last case good results are achieved, making recommendable this kind of techniques for such applications.

Chapter 7

When I examine myself and my methods of thought, I come to the conclusion that the gift of fantasy has meant more to me than any talent for abstract, positive thinking.

Albert Einstein (1879 - 1955)

Prediction is very difficult, especially about the future.

Niels Bohr (1885 - 1962)

Conclusions and Further Research

7.1 Conclusions

After the study and classification of the different super-resolution algorithm found in the scientific literature, new iterative versions have been created. These algorithms have been optimized for being executed on modified hybrid video encoders. The iterative algorithms have been modified in order to create non-iterative versions for dynamic super-resolution, also implemented over the video-encoder. These versions have been developed and implemented on the hybrid video-encoder platform named Picasso, developed at Philips Nat.Lab in the ESAS group. The main goal has been to achieve super-resolution improvements for video sequences with real-time and low-cost performances.

This platform supposes a powerful system for hybrid video encoding, where it is interesting to point out the following characteristics:

- It implements the basic compression core for algorithms based on the discrete cosine transform (DCT).
- The system is open to the implementation of new standards for video and still-image

compression.

- The hardware-software partition provides very high performance.
- Memory compression in the DCT domain allows using only on-chip memories. By means of avoiding the use of external memories an important decrease of the power dissipation is achieved at the same time that the costs are minimized to a single integrated circuit (IC).
- The design methodology allows the implementation and easy evaluation of new applications over the platform, as could be the super-resolution function developed.

The Picasso platform is capable of compressing video sequences using the H.263 standard and implementing some the options of H.263+. Likewise, it is also prepared to give support to the main compression core for MPEG-4. It can also compress static images in JPEG format and libraries have been developed to enable multi-thread operations, what will permit the execution of several formerly mentioned applications at the same time following a time-sharing scheme of the resources.

The first algorithm set developed consists of the iterative algorithm for static super-resolution, based in [BK99] and constitutes a first step through the planned objective of obtaining low-cost and real-time super-resolution algorithms. The low-cost objective has been achieved in the sense that the algorithm has successfully been implemented on the existing video platform, with minimal hardware modifications. However, the real-time objective is not accomplished with this kind of algorithm. Following this approach, a new non-iterative algorithm scheme has been created, initially based on the concept on contributive weights, capable of obtain super-resolution improvements in a single step. Moreover, reusing the last super-resolved image, we succeed in keeping the memory sizes within the limits that allow integrating them on-chip.

The first algorithm set is suitable for static super-resolution, i.e. the generation of a single high-resolution image through the combination of several low-resolution images. In Figure 85 the scheme of super-resolution image generation followed in this kind of algorithms is shown. Versions v1.0, v1.1, v1.2 and v1.3 belong to the category of iterative algorithms for static super-resolution. For the used test sequences, the average luminance qualities obtained were of 23.19 dB for version v1.2 and 28.24 dB for version v1.3, as these are the most significant versions among the iterative algorithms.

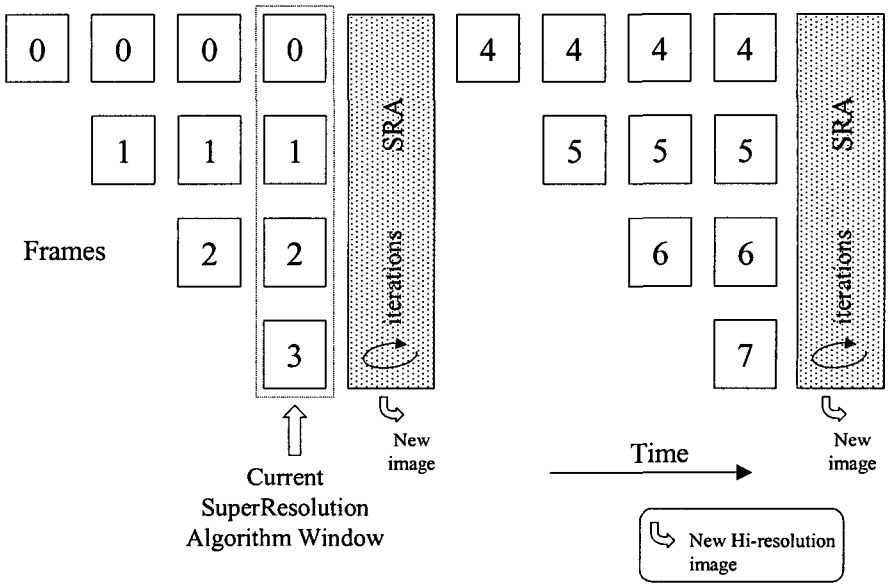


Figure 202. Scheme for the generation of static super-resolution images in the iterative algorithms.

The quality of the super-resolution images strongly depend on the sampling performed for the input images. If the image set used for generating the super-resolution image has redundant samples, then the super-resolution process will be deprived of new information, generating images of lower quality than the quality that could be obtained if every image will contribute new information.

The experiments carried out demonstrate that the 3DRS motion estimation performs equal to full-search for super-resolution enhancements. Therefore, using an exhaustive method of motion estimation of high computational cost that does not have a relevant repercussion in the final image quality makes no sense. A similar quality is obtained with a much cheaper motion estimation method as it is the 3DRS already implemented in Picasso.

However, the use of low-pass filters in the input image before the motion-estimation is indeed decisive, thus achieving important improvements in the quality and stability of the image.

The second algorithm set brought in this work are the non-iterative ones, suitable to work in real time. Inside the non-iterative algorithms we can perform a sub-classification in algorithms for static super-resolution and algorithms for dynamic super-resolution. In Figure 203.a the scheme for the generation of images in the non-iterative algorithms for static super-resolution is presented. Although the quality of the image is quite high, the non-iterative algorithms have the serious drawback of demanding a lot of memory, making very difficult

the objective of keeping the system inside low-cost restrictions. The second kinds of algorithms (Figure 203.b) are thought for video (dynamic super-resolution) and their function is based on the feed-back of the resulting image.

During the evaluation phase of the quality of the obtained images, it was clear the difficulty presented by the image borders to be processed as far as they break the algorithm homogeneity. Initially the procedure to obtain test images was modified to avoid undesirable artefacts in the borders when moving the image. This is simply done by starting from higher image sizes, performing the shifts over them and finally, cropping the central part. After all, through algorithm adjustments it was possible to incorporate the image borders processing jointly with the contributive weights. This is achieved by introducing zeroes in the borders of the shifted images that will be lately filled with new or interpolated data using the present data. Nevertheless, due to the existing trade-off between quality and memory requirements, in the last version 3.1 the decision of just interpolate the image borders was adopted, removing the contributions and therefore obtaining some memory and processing saves.

Through the reusing of the data of the previous image, in versions v3.0 and v3.1 the memory requirements are reduced in a factor of 50 times lower than the versions v2.0 and v2.1, and 35 times lower than the versions v1.2 and v1.3. As these two latter versions do not collect and accumulate the information coming from different images, the pixels suffer from an effect of temporal dilution that produces a quality loss compared with the previous versions. Nevertheless, experimentally the quality loss demonstrates to be lower than 2 dB.

In addition to the pixel filtering (conventional image filtering) the filtering of the motion vectors has shown to be a very powerful technique to both increase the image quality and reduce the control part of the algorithm. As the shifts have different effects on the objects depending on the focal plane where they are located, this technique allows the computation of a pseudo-global motion vector for every zone. To increase the quality of the filtered motion vectors, these have been weighted with the SAD intra. This is a measure of the high frequencies amount contained in every macro-block and therefore a measure of how good the computed motion vectors are.

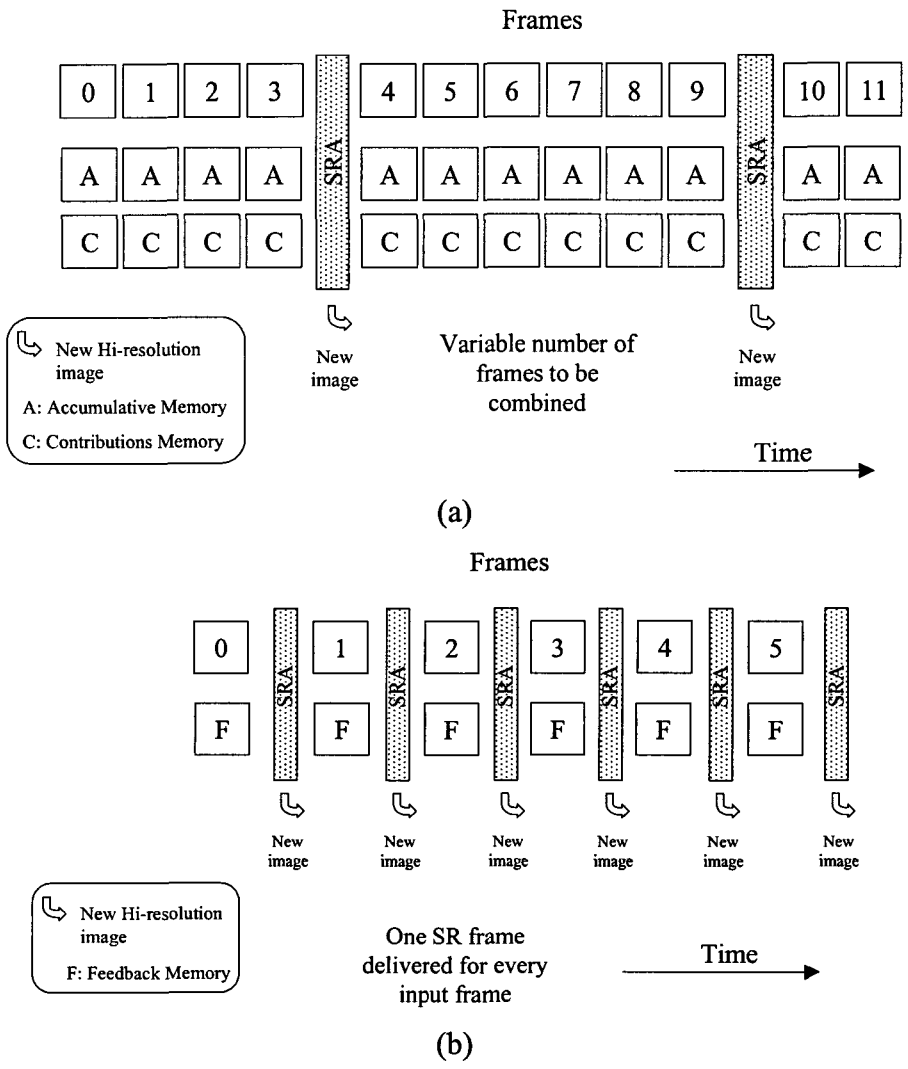


Figure 203. Scheme for the generation of static super-resolution images in the iterative algorithms.

Super-resolution has several applications in the image processing. Among the main applications scope we can highlight performing digital zoom without the use of mechanical-optical components, colour reconstruction from the raw colours delivered by the sensor, increase of the resolution and size of an image or video sequence and some combinations of them as could be the resolution enhancement together with the colour reconstruction.

The input system has a vital importance in the super-resolution process. It is impossible to obtain super-resolution improvements using images where all the aliasing has been removed. Nevertheless, this is the usual scenario in almost all the imaging systems. As the aliasing is an undesirable feature in the input images for typical applications, it is removed (as much as possible) before the imaging system starts the processing, using lenses with

optical low-pass filters (OLP) incorporated, including micro-lenses over every pixel sensor or low-pass filtering the images as they are captured. Removing the aliasing implies loosing a lot of high frequency information, just the information that the super-resolution algorithms can recover.

As a resume of this chapter, Figure 204 depicts a summary of all the developed super-resolution algorithms in this work and the main features of each one. In addition, in Table 28 a summary of the average quality obtained by the versions of the developed super-resolution algorithms is presented. The non-iterative versions for dynamic super-resolution exhibit qualities slightly lower than the static super-resolution but with much lower memory requirements.

All the presented algorithms are intended to be executed onto the Picasso platform, performing minor modifications and reusing some of the hardware. In this manner, the capability of performing super-resolution improvements has been added to the previous existing features of the encoder at a reduced cost.

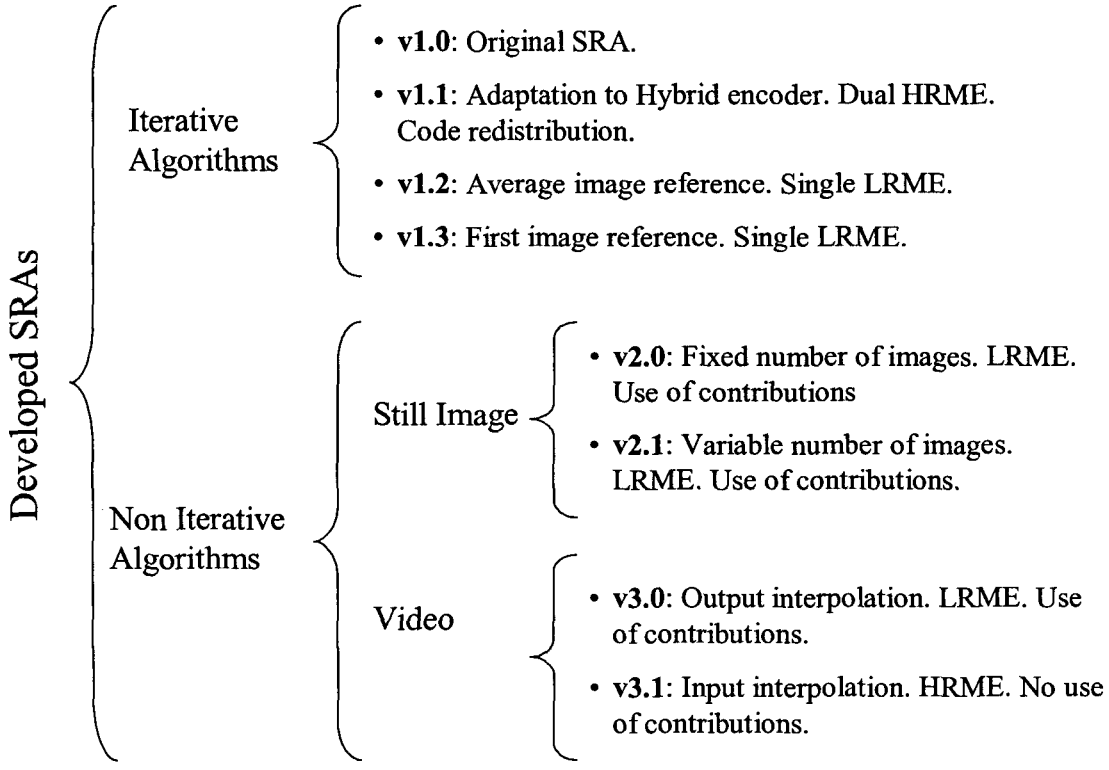


Figure 204. Summary of the different developed super-resolution algorithms.

To finish up this research work, it is important to remark that we have demonstrated, qualitatively and quantitatively, the quality improvements in static images and video that can

be obtained using super-resolution algorithms implemented onto a hybrid video encoder with real-time and low-cost features.

Version	Type	Average PSNR Luminance	Average PSNR Red chrominance	Average PSNR Blue chrominance
v1.2	Iterative. Static SR	23.19 dB	38.45 dB	40.79 dB
v1.3	Iterative. SR static	28.24 dB	38.24 dB	39.88 dB
v2.0	No Iterative. SR static	30.47 dB	45.42 dB	48.77 dB
v2.1	No Iterative. SR static	34.63 dB	46.90 dB	49.66 dB
v3.0	No Iterative. SR dynamic	28.64 dB	44.22 dB	47.22 dB
v3.1	No Iterative. SR dynamic	26.28 dB	41.28 dB	44.25 dB

Table 28. Average PSNR for the most significant versions of the developed super-resolution algorithms.

7.2 Further Research

The subjects developed in this work offer the possibility of continuing with the research matters hereafter outlined.

a) **Determine the appropriate input system that enables super-resolution improvements**. It has been clearly established the necessity of having some aliasing in the input images. It is also clear that the aliasing is undesirable in most of the imaging systems and therefore it is largely removed. Removing the obstacles that avoid the aliasing will have a negative impact on the imaging system when super-resolution is not being used, but at the same time these obstacle also suppose a limit on the quality of the images. It is a clear trade-off between the quality of the high frequencies of the image and the image aliasing. An input sub-system that would enable some aliasing when working with super-resolution and that removes (probably through filtering) the aliasing when not working with super-resolution will be highly desirable. The use of fixed micro-lenses clearly precludes the possibility of obtaining super-resolution improvements.

b) **Incorporate temporal motion vector filtering**. As the spatial motion vector filtering has proven to be a powerful technique, some additional research should be done in order to investigate if a temporal filtering of the motion vectors would have an important improvement on the final image quality.

c) **Use of super-resolution in the image decoding**. The decoded images have been low-pass filtered several times (by the loop-filters and by the combination DCT-quantizer that removes the high frequency coefficients). Thus, it would be practically impossible to obtain super-resolution improvements from them. But, if the quantizer coefficients were modified in such a way that they enable to pass some high frequency information, we could sub-sample the image prior coding, sending an image of 25% the size of the original size, and recompose the original size by means of the super-resolution process after decoding it. It must be taken into account that, as some high frequencies would be allowed to pass, the binary stream will probably be slightly longer than the normal size (both streams would be covered by the standards, and so, both could be decoded by any standard decoder, but only in the case that the standard allows to change the quantification matrix). An optional way to obtain SR improvements from a decoded video sequence will be using the enhancement layer described in the MPEG-2 standard for SNR scalability. The enhancement layer contains high-frequency information that can be used in the SR process to increase the size and resolution of the final decoded and improved video sequence.

References

Science is organized knowledge. Wisdom is organized life.
Immanuel Kant (1724 - 1804)

References

[Ack93] B.D. ACKLAND, "A video-codec chip set for multimedia applications," AT&T Technical Journal, pp. 50-65, January 1993.

[Ack94] B.D. ACKLAND, "The role of VLSI in multimedia," IEEE Journal of Solid State Circuits, vol. 29, Issue 4, 1994.

[Ack95] B.D. ACKLAND, "VLSI for multimedia applications," Proc. of 13th Australian Microelectronics Conference, pp. 23-33, Adelaida, Australia del Sur, The IREE Society, July 1995.

[AD97] V. AVRIN and I. DINSTEIN, "Restoration and resolution enhancement of video sequences," ICASSP IV, pp. 549-553, 1997

[Aki94] T. AKIYAMA, "MPEG2 video codec using image compression DSP," IEEE Transantions on Consumer Electronics, vol. 40, issue 3, 1994.

[Aon92] K. AONO, "A video digital signal processor with a vector-pipeline architecture," IEEE Jornal of Solid-State Circuits, vol. 27, issue 12, 1992.

[APM02] Y. ALTUNBASAK, A.J. PATTI and R.M. MERSEREAU, "Super-resolution still and video reconstruction from MPEG-coded video," IEEE Transactions on Circuits and Systems for Video Technology, vol. 12, issue 4, pp. 217-226, April 2002.

[Ara94] T. ARAKI, "Video DSP architecture for MPEG-2 codec," Proceedings of the 1994 Conference on Acoustics, Speech and Signal Processing, vol. 2, pp. 417-420, 1994.

[Bai92] D. BAILEY, "Programmable vision processor/controller," IEEE Micro, vol. 12, issue 5, pp. 33-39, October 1992.

[BBZ96] B. BASCLE, A. BLAKE and A. ZISSERMAAN, "Motion deblurring and super-resolution from an image sequence," in Proceedings of European Conference and Computer Vision ECCV III, Cambridge, VK. Springer-Verlag, pp. 573-582, 1996.

[BCF+97] J.C. BAUER, É. CLOSSE, É. FLAMMAND, M. POIZE, J. PULOU and P. PENIER, "SAXO: A retargetable optimized compiler for DSPs," Proceedings of ICSPAT, 1997.

[Bee97] MARC J. OP DE BEECK , "On the influence of aliasing and low pass filtering on super resolution image reconstruction for video processing," Nat-lab technical note tn-390/97, Philips Research, Eindhoven, 1997. Company Restricted.

[BFB94] J. L. BARRON, D. J. FLEET and S. S. BEAUCHERNIN, "Performance of Optical Flow Techniques," Journals of Computational Vision, vol. 12, pp. 43-77, 1994.

[Bie95] J. BIER, "DSP processors and cores: The options multiply," Integrated System Design Magazine, July 1995.

[BK00] S. BAKER and T. KANADE, "Limits on super-resolution and how to break them," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, vol. 2, pp. 372 -379, 2000.

[BK02] S. BAKER and T. KANADE, "Limits on super-resolution and how to break them," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, issue 9, pp. 1167 -1183, Sep 2002.

[BK96] V. BHASKARAN, K. KONSTANTINIDES, Image and Video Compression Standards: Algorithms and Architectures, second edition, Kluwer Academic Publishers, 1997.

[BK99] MARC J.R. OP DE BEECK and RICHARD P. KLEIHORST, "SuperResolution of Regions of Interest in a Hybrid Video Encoder", Philips Conference on DSP, 1999.

[BK99b] SIMON BAKER and TAKEO KANADE, "Super-resolution optical flow," Tech. Rep. CMU-RI-TR-99-36, Robotics Institute, Carnegie Mellon University, USA, 1999.

[BKV93] N.K. BOSE, H.C. KIM and H.M. VALENZUELA, "Recursive implementation of Total Squares algorithm for image reconstruction from noisy under-sampled multi-frames," Proceedings of the IEEE International Conference ASSP (ICASSP), vol. 5, Minneapolis MN, pp. 269-272, 1993.

[BMC+96] T. BAUTISTA, GUSTAVO MARRERO CALLICÓ, P.P. CARBALLO and A. NÚÑEZ, "Towards a low-cost processor architecture for multimedia," Proceedings of the XI Conference of Design of Integrated Circuits and Systems, pp. 445-450, Universitat Politècnica de Catalunya, CPDA, Barcelona, November 1996.

[BMC+97] TOMÁS BAUTISTA, GUSTAVO MARRERO CALLICÓ, PEDRO P. CARBALLO and ANTONIO NÚÑEZ, "Rapid-prototype of High-performance RISC Cores with VHDL," Fall 1997 Conference. VHDL International Users' Forum (VIUF). Edited by the IEEE Computer Society, Arlinton, Virginia, EE.UU, pp. 43-52, October 1997.

[BNCS92] T. BAUTISTA, A. NÚÑEZ, P.P. CARBALLO AND R. SARMIENTO, "Compilación y realización de una versión con células estándares del TMS32010," VII Congreso de Diseño de Circuitos Integrados, pp. 371-376, Noviembre 1992.

[Bot00] VINCENT BOTTREAU, "Study of Motion Estimation Techniques for Motion Compensated Temporal Filtering," Philips Technical Report No. C2000-729, January 2000. Company Restricted.

[Bov00] AD BOVIK, Handbook of Image & Video Processing, Academic Press, 2000.

[Bro81] J.L. BROWN "Multichannel sampling of low pass signals," IEEE Trans. on Circuits and Systems, vol. CAS-28, no. 2, pp. 101-106, February 1981.

[Bur93] D. BURSKY, "Codec compresses image in real time," Electronic Design, pp. 123-124, October 1993.

[Cam99] L. CAMICIOTTI, "Noise filters for consumer MPEG-2 encoders," Private Communications, 1999.

[Can98] FRANK M. CANDOCIA, "A Unified Superresolution Approach for optical and synthetic Aperture Radar Images," Ph.D. dissertation, University of Florida, 1998.

[CB00] M.C. CHIANG AND T.E. BOULT ,"Efficient super-resolution via image warping," Image and Vision Computing, vol. 18, pp. 761-771, 2000.

[CC90] C. K. CHUI AND G. CHEN, Kalman Filtering, Springer-Verlag, 1990.

[CCu97] C-Cube Microsystems, "One-to-one Digital video: Integration Encoding and Decoding Technology on One Chip," <http://www.c-cube.com>, 1997.

[Cha01] SUBHASIS CHAUDHURI editor, Super-resolution Imaging, Kluwer Academic Publishers, 2001.

[Chi92] T. M. CHIN, "Dynamic Estimation in Computational Vision," Ph.D. Dissertation, Department of Electrical engineering and Computer Science, MIT, February 1992.

[CKK+94] P. CHEESEMAN, B. KANEFSKY, R. KRAFT, J STUTZ and R. HANSON, "Super-resolved surface reconstruction from multiple images," technical report FIA-94-12, NASA Ames Research Center, Moffet Field, CA, December 1994.

[CKM+93] T. M. CHIN, V. C. KARL, A. M. MARIANO and A. S. WILLSKY, "Square Root Filtering in Time- Sequential Estimation of Random Fields," Proc. SPIE, vol. 1903, pp. 51-58, 1993.

[CLL02] YUPING CHENG, YILONG LU and ZHIPING LIN, "A super resolution SAR imaging method based on CSA," IEEE International Geoscience and Remote Sensing Symposium IGARSS '02, vol. 6, pp. 3671 -3673, 2002.

[CM98] DIDIER CALLE and ANNICK MONTANVERT, "Super-resolution inducing of an image," in Proceedings of the IEEE International Conference on Image Processing, Chicago, USA, pp. 742–746, 1998.

[CP98] F. M. CANDOCIA and J. C. PRINCIPE, "A Method using Multiple Models to Superresolve SAR Imagery," Proceedings of SPIE: Algorithms for Synthetic Aperture Radar Imagery V, April 1998.

[CPL85] J.J. CLARK, M. R. PALMER and P.D. LAWRENCE, "A transformation method for the reconstruction of functions from non uniformly spaced samples," IEEE Trans. on Acoustics, Speech and Signal Processing, vol. 33, pp. 1151-1165, October 1985.

[CR99] S. CHAUDHURI and A. N. RAJAGOPALAN, Depth from defocus ed images: A real aperture imaging approach, Springer-Verlag, New York, 1999.

[CRS+98] R. CMAR, L. RIJNDERS, P. SCHAUMONT, S. VERNALDE and I. BOLSENS, "A methodology and design environment for DSP ASIC fixed point refinement," Proceedings of DATE, pp.271-276, Munich, RFA, IEEE Computer Society, 1998.

[CS98] D. CHEN and R.R. SCHULTZ, "Extraction of High-Resolution Frames from MPEG Image Sequences," In Proceeedings of the International Conference of Image Processing, Chicago , IL, October 1998.

[CZ00] D. CAPEL and A. ZISSERMAN, "Super-resolution enhancement of text image sequences," Proceedings of the 15th International Conference on Pattern Recognition, vol 1 , pp. 600-605, 2000.

[CZ01] D. CAPEL and A. ZISSERMAN, "Super-resolution from multiple views using learnt image models," Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2001), vol. 2, pp. II-627-II-634, 2001.

[Das95] B. V. DASARATHY, Image Data Compression, IEEE Computer Society Press, 1995.

[DBP+00] F. DEKEYSER, P. BOUTHEMY, P. PÉREZ and E. PAYOT, "Super-resolution from noisy image sequences exploiting a 2D parametric motion model," Proceedings of the 15th International Conference on Pattern Recognition, vol. 3 , pp. 350-353, 2000.

[DSP] "DSP Overview at Texas Instruments," <http://www.ti.com/sc/docs/products/dsp/overview.htm>

[DWW+96] S. DUTTA, A. WOLFE, W. WOLF and K.J. O'CONNOR, "VLSI Signal Processing, IX," chapter *'Design issues for very-long-instruction-word (VLIW) video signal processors'*, pp. 95-104, IEEE Press, 1996.

[EF95] M. ELAD and A. FEUER, "Recursive Optical Flow Estimation - Adaptive Filtering Approach", Internal Report #1009, the Technion - Israel Institute of Technology, 1995

[EF97a] M. ELAD and A. FEUER, "Restoration of Single Super-Resolution Image From Several Blurred, Noisy and Down-Sampled Measured Images," IEEE Trans. on Image Processing, vol. 6, no. 12, pp.1646-1658, 1997.

[EF97b] M. ELAD and A. FEUER, "Super-Resolution Restoration of Image Sequence – Adaptative Filtering Approach," Tech. Rep. 973, The Technion – Israel Institute of Technologhy, 1994

[Egg00] PETER EGGLESTON, "Quality Photos from Digital Video: Salient Stills' Algorithms Provide a Missing Link," Advanced Imaging, pp. 39-41, May 2000.

[Ela96] MICHAEL ELAD, "Super-Resolution Reconstruction of images," Ph.D. dissertation, Israel Institute of Technology, 1996.

[Eno93] T. ENOMOTO, "A 300 MHz 16 bit programmable video signal processor ULSI for a single chip teleconferencing system," Proceedings of the 19th European Solid-State Circuits Conference (ESSCIRC'93), September 1993.

[Ern97] R. ERNEST, Hardware / software co-design: Principles and Practice, Kluwer Academic Publishers, 1997.

[Ern98] R. ERNEST, "Codesign of embedded systems: Status and trends," IEEE Design & Test of Computers, pp. 45-54, April 1998.

[ES00] K.J. ERICKSON and R.R. SCHULTZ, "MPEG-1 super-resolution decoding for the analysis of video still images," Proceedings of the 4th IEEE Southwest Symposium on Image Analysis and Interpretation, pp. 13 -17, 2000.

[EST97] P. E. EREN, M. L SEZAN and A M. TEKALP, "Robust, object based high resolution image reconstruction from low resolution video," IEEE Transactions on Image Processing, vol. 6, pp. 1446 – 1451, Oct. 1997.

[Fan86] K. M. FANT, "A non aliasing, real time spatial transform technique," IEEE Computer Graphics and Applications, vol. 6, no. 1, pp. 71 – 80, 1986.

[FDF98] P. FARABOSCHI, G. DESOLI and J.A. FISCHER, "The latest word in digital and media processing," IEEE Signal Processing Magazine, vol. 15, issue 2, pp. 59-85, March 1998.

[FJP02] W.T. FREEMAN, T.R. JONES and E.C. PASZTOR, "Example-based super-resolution," IEEE Computer Graphics and Applications, vol. 22, issue 2, pp. 56 -65, March/April 2002.

[FL95] D. J. FLEET and K. LANGLEY, "Recursive Filters for Optical Flow," IEEE Trans. Pattern analysis and Machine Intelligence, vol. 17, pp. 61-67, 1995.

[Fos99] R. FOSTER, "Simplifying design of systems-on-a-chip with rapid silicon prototyping and reusable IP," DATE User Forum, IEEE Computer Society, pp. 147-151, Munich, 1999.

[FP99] W.T. FREEMAN and E.C. PASZTOR, "Learning low-level vision," Proceedings of the Seventh IEEE International Conference on Computer Vision, vol. 2, pp. 1182 -1189, 1999.

[fro] "Frontier Design," <http://www.frontierd.com/artlibrary. htm>

[Fur96] STEVE FURBER, ARM System Architecture, Addison-Wesley, ISBN. 0-201-40352-8, 1996.

[Gal91] D. LE GALL, "MPEG: A video compression Standard for Multimedia Applications," Communications of the ACM, vol. 34, no. 4, April 1991.

[GBA+02] B. K. GUNTURK, A.U. BATUR, Y. ALTUNBASAK, M.H. HAYES and R.M. MERSEREAU, "Eigenface-based super-resolution for face recognition," Proceedings of the International Conference on Image Processing, vol. 2 , pp. 845 -848, 2002.

[Ger74] R. W. Gerchberg, "Super-resolution through error energy reduction," Opt. Acta, vol. 21, pp. 709–720, 1974.

[Ghi84] D. C. GHIGLIA, "Space-Invariant deblurring Given N Independently Blurred Images of a Common Object," Journal of Optical Society in America, vol. 1, pp. 398-402, April 1984.

[Gin00] V. GINZBURG, "Locator with super-scanning phased array and its basic features," Proceedings of International IEEE Conference on Phased Array Systems and Technology, pp. 181-184, 2000.

[GLM+94] J. A. GOYETTE, G. D. LAPIN, MOON GI KANG and A. K. KATSAGGELOS , "Improving autoradiograph resolution using image restoration techniques," IEEE Engineering in Medicine and Biology Magazine, vol. 13, no. 3, pp. 571-574, Sept. 1994.

[GMC+98] ALVARO GONZÁLEZ, GUSTAVO MARRERO, PEDRO P. CARBALLO, TOMÁS BAUTISTA and ANTONIO NÚÑEZ, "Tratamiento de señales en DSPs multiprocesadores. Algoritmo paralelo para MPEG en el sistema SDB TMS320C80," FAVI, International Conference on Automatic Control, PADI2, Octubre 1998.

[GVN+94] D. GAJSKI, F. VAHID, S. NARAYAN and J. GONG, Design of Embedded Systems, Prentice Hall, 1994.

[GW87] R. C. GONZALEZ and P. WINTZ, Digital Image Processing, Addison-Wesley Publishers, 1987.

[GZ97] R. GUPTA and Y. ZORIAN, "Introducing core-based system design," IEEE Desipn & Test, vol. 14, no. 4, October 1997.

[H261] "Line transmission on non telephone signals. Video Codec for audiovisual services at p×64 Kbits/s," ITU-T recommendation H.261, March 1993

[H262] ISO/IEC JTC1/SC29 CD 11544, "Coded Representation of picture and audio information – progressive bi-level image compression," recommendation H.262, November 1993.

[H263++] ITU-T recommendation H.263 Version 2 (H.263++). Video Coding for Low Bitrate Communication. November 2000.

[H263+] ITU-T recommendation H.263 Version 2 (H.263+) Video Coding for Low Bitrate Communication. January 1998.

[H264] ITU-T Recommendation H.264/ISO/IEC 11496-10, "Advanced Video Coding", Document JVT-E002, September 2002.

[H263] "Video Coding for low bit rate communication", ITU-T recommendation H.263, July 1995.

[Ha+99] MICHEL HARRAND et.al, "A single-chip CIF 30-Hz, H261, H263+ video encoder/decoder with embedded display controller," IEEE Journal of Solid-State Circuits, vol. 34, no. 11, pp. 1627-1632, November 1999.

[HB95] G. DE HAAN and P.W.A.C. BIEZEN, "Sub-pixel motion estimation with 3-D recursive search block-matching," Signal Processing on Image Communication, vol. 6, pp. 485-498, 1995.

[HBA97] R.C. HARDIE, K.J. BARNARD and E.E. ARMSTRONG, "Joint MAP registration and high resolution image estimation using a sequence of undersampled images," IEEE Trans. Image Process, vol. 6, pp. 1621-1633, 1997.

[HBB98] RUSSEL C. HARDIE, K. J. BARNARD, J. G BOGNAR, E. E. ARMSTRONG and E. A. WATSON, "Joint high resolution image reconstruction from a sequence of rotated and translated frames and its application to an infrared imaging system," Optical Engineering, vol. 37, no. 1, pp. 247–260, January 1998.

[HBH+93] G. DE HAAN, P.W.A.C. BIEZEN, H. HUIJGEN and O.A. OJO, "True motion estimation with 3-D recursive search block-matching", IEEE Transaction on Circuits and Systems for Video Technology, vol. 3, pp. 368-379, October 1993.

[HK84] B. R. HUNT and O. KUBLER, "Karhunen-Loeve Multispectral Image Restoration, Part I -Theory," IEEE Transaction on Acoustics, Speech and Signal Processing, vol. 32, pp. 592-599, June 1984.

[HK97] MIN-CHEOL HONG, MOON GI KANG and AGGELOS K. KATSAGGELOS, "A regularized multichannel restoration approach for globally optimal high resolution video sequence," in Proceedings of Conference on Visual Communications and Image Processing, San Jose, CA, USA, pp. 1306–1313, 1997.

[HKK97a] M. HONG, MOON GI KANG and A. K. KATSAGGELOS, "Higher resolution reconstruction of video signal based on generalised multichannel regularisation," in Proceedings of SPIE - The international Society for optical Enginnering , VCIP97, March 1997.

[HKK97b] M. HONG, MOON GI KANG and A. K. KATSAGGELOS, "A multichannel restoration for globally optimal high resolution video," IEEE ICIP 97, October 1997.

[HS81] B. K. P. HORN and B. G. SCHUNCK, "Determining Optical Flow," Artificial Intelligence, vol. 17, pp. 185-204, 1981.

[HT84] T. S. HUANG and R. Y. TSAY, "Multiple Frame Image Restoration and Registration," in Advances in Computer Vision and Image Processing, (Editor - T. S. Huang), vol. 1, pp. 317-339, JAI Press Inc., Greenwich, CT, 1984.

[Hua83] T. S. HUANG, "Image Sequence Processing And Dynamic Scene Analysis," Springer-Verlag, Berlin, 1983.

[Hun95] B.R. HUNT, "Super-resolution of images: algorithms, principles and performance," International Journal of imaging Systems and Technology, vol. 6, pp. 297-304, 1995.

[IIT93] Integrated Information Technology, Inc. "Single Chip Video Codec and Multimedia Communications Processor," Preliminary Datasheet, 1993.

[Ike96] M. IKEDA, "A hardware/software concurrent design for real-time SPML MPEG2 video-encoder chip set," Proceedings of the EDTC, pp. 320-326, 1996.

[Ino93] T. INONUE, "A 300-MHz 16-B BiCMOS video signal processor," IEEE Journal of Solid-State Circuits, vol. 28, issue 12, December 1993.

[IP91] M. IRANI and S. PELEG, "Improving resolution by Image Registration," CVGIP: Graphical Models and Image Processing, vol. 53, pp. 231-239, March 1991.

[IP93] M. IRANI and S. PELEG, "Motion Analysis for Image Enhancement: Resolution, Occlusion, and Transparency," Journal of Visual Communication and Image Representation (VCIR), vol. 4, pp. 324-335, December 1993.

[ISD] Integrated System Design, <http://www.isdmag.com/>

[Jai89] A. K. JAIN, Fundamentals In Digital Image Processing, Prentice-Hall, 1989.

[Jai89] A. K. JAIN, Fundamentals In Digital Image Processing, Prentice-Hall, 1989.

[Kat90] A. K. KATSAGGELOS, "A Multiple Input Image Restoration Approach," Journal of Visual Communication and Image Representation, vol. 1, pp. 93-103, September 1990.

[KBV90] S. P. KIM, N. K. BOSE and H. M VALENZUELA, "Recursive Reconstruction of High Resolution Image from Noisy undersampled multiframes," IEEE Trans. on ASSP, vol. 38, pp. 1013-1027, June 1990.

[KC98] R.P. KLEIHORST and F.CABRERA, "VLSI implementation of DCT-domain motion estimation and compensation," In Proceedings of the 19th Symposium on Information Theory in the Benelux, pp 21-28, Veldhoven, The Netherlands, May 1998.

[KDE+89] A. K. KATSAGGELOS, J. N. DRIESSEN, S. N. EFSTRATIADIS and R. L. LAGENDIJK, "SpatioTemporal Motion Compensated Noise Filtering of Image Sequences," in Proceedings of SPIE - The international Society for optical Enginnering, vol. 1199, pp. 61-70, 1989.

[KF02] O. KURSUN and O. FAVOROV, "Single-frame super-resolution by a cortex based mechanism using high level visual features in natural images, "Proceedings of the 6th IEEE Workshop on Applications of Computer Vision (WACV 2002), pp. 112 -117, 2002.

[KG98] AGGELOS KATSAGGELOS and NICK GALATSANOS, "Signal Recovery Techniques for Image and Video Compression and Transmission," ISBN: 0-7923-8298, Ed. Kluwer Academic Publishers 1998.

[KIA+93] TAKASHI KOMATSU, TORU IGARASHI, KIYOHARU AIZAWA and TAKAHIRO SAITO, "Very high resolution imaging scheme with multiple different-aperture cameras," Signal Processing: Image Communication, vol. 5, pp. 511-526, December 1993.

[Kim94] H. C. KIM, "High Resolution Image Reconstruction from Undersampled Multiframes," PhD thesis, The Pennsylvania State University, 1994.

[KKE+91] A. K. KATSAGGELOS, R. P. KLEIHORST, S. N. EFSTRATIADIS and R. L. LAGENDIJK, "Adaptive Image Sequence Noise Filtering Methods," in Proc. SPIE - The international Society for optical Engineering, vol. 1606, pp. 716-727, 1991.

[KL91] S. J. KO and Y. H. LEE, "Nonlinear Spatio-Temporal Noise Suppression Techniques with Applications in Image Sequence Processing," IEEE International Symposium CJS, vol. 5 pp. 662-665, 1991.

[KMT+95] L. KOHN, G. MATURANA, M. TRMBLAY, M. PRABHU and G. ZYNER, "The Visual Instruction Set (VIS) in UltraSPARCTM," UltraSPARC Microprocessor Whitepapers, SPARC Technology Business, 1995.

[Kon96] T. KONDO, "Two-chip MPEG-2 video encoder," IEEE Micro, pp. 51-58, 1996.

[Kou95] WEIDONG KOU, Digital Image Compression Algorithms and Standards, Kluwer Academic Publishers, 1995.

[KPB88] D. KEREN, S. PELEG and R. BRADA, "Image sequence enhancement using sub-pixel displacements," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Ann Arbor, USA, pp. 742-746, 1988.

[KSS+85] D.T.KUAN, A.A. SAWCHUCK, T.C.STRAND and P. CHAVEL, "Adaptive noise smoothing filter for images with signal-dependent noise," IEEE Transaction on Patt. Anal. Machine Intell. PAMI-7, pp.165-177, March 1985.

[KVL99] RICHARD KLEIHORST, RENÉ VAN DER VLEUTEN and RAFAEL PESET LLOPIS, "DCT-domain embedded compression for hybrid video coders", Philips Technical note 301/99. Company Confidential.

[KVP00] R. P. KLEIHORST, R. VAN DER VLEUTEN and R. PESET, "DCT-domain embedded memory compression for hybrid video coders," Journals of VLSI Signal Processing Systems, vol. 24, pp. 31-41, February 2000.

[KW93] S.P. KIM and WEN-YU SU, "Recursive High-Resolution Reconstruction of Blurred Multiframe Images," IEEE Transactions on Image Processing, vol. 2, no. 4, pp 534-539, October 1993.

[KWS+99] R. KLEIHORST, P. WIELAGE, R. SALTERS, R. PESET LLOPIS and R. VAN DER VLEUTEN, "Variable-length encoding and decoding with content-addressable memories", Philips Technical Note 357/99. Company Confidential.

[Lap95] P. LAPSEY, "NSP shows promise on the Pentiurn and PowerPC," Microprocessor Reports, vol. 8, pp. 11-15, 1995.

[LB91] R. L. LAGENDIJK and J. BIEMOND, Iterative Identification And Restoration Of Images, Kluwer Academic Publishing, Boston, 1991.

[LBSL97] P. LAPSEY, J. BIER, A. SHOHAM and E.A. LEE, "DSP Processor Fundamentals," IEEE Press, 1997.

[Lee80] J.S. LEE, "Digital image enhancement and noise filtering by use of local statistics," IEEE Transaction on Patt. Anal. Machine Intell., PAMI-2, pp. 165-168, March 1980.

[Lee94] B.W. LEE, "Data flow processor for multi-standard video codec," Proceedings of the IEEE 1994 Customs Integrated Circuits Conference, pp. 103-106, 1994.

[Lee95] R.B. LEE, "Accelerating multimedia with enhanced processors," IEEE Micro, April 1995.

[Lee96] R.B. LEE, "Subword parallelism with MAX-2," IEEE Micro, vol. 16, issue 4, pp. 51-59, August 1996.

[Lio91] M. L. LIOU, "H.261 overview," Communications of the ACM, vol. 34, no. 3, April 1991.

[Lip97] P. E. R. LIPPENS, "C-HEAP: CPU-controlled heterogeneous embedded architectures for signal processing," Private Communications, 1997.

[LK81] B. LUCAS and T. KANADE, "An Iterative Image Registration Technique with application to Stereo Vision," Proc. DARPA Image Understanding Workshop, pp. 121-130, 1981.

[LKL+01] RAFAEL PESET LLOPIS, RICHARD KLEIHORST, PAUL LIPPENS, MARCEL OOSTERHUIS and RENE VAN DER VEULEN, "Architecture Overview of Picasso Designs", Natlab Technical note 336/99, 2001. Company Confidential.

[LN97] PAUL LIPPENS and ANDRE NIEUWLAND, "C-HEAP, CPU-controlled Heterogeneous Embedded Architectures for Signal Processing", Philips Technical Note 175/97. Company Confidential.

[LP95] E. A. LEE and T.M. PARKS, "Dataflow process networks," Proceedings of the IEEE, vol. 83, no. 5, pp. 773-799, May 1995.

[LS01] ZHOUCHEN LIN and HEUNG-YEUNG SHUM, "On the fundamental limits of reconstruction-based super-resolution algorithms," Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2001), vol. 1, pp. I-1171 -I-1176, 2001.

[LSI96] "L64110/120/130 VISC Encoder Chipset," LSI Press Release, 1996.

[LSV+96] L. LAVAGNO, A. SANGIOVANNI-VINCENTELLI and HSIEH, "Embedded System Codesign: Synthesis and Verification", Kluwer Academie Publishers, 1996.

[LTZ+01] XINGDONG LIANG, RAN TAO, SHIYONG ZHOU and YUE WANG, "Multidimensional real aperture radar imaging," Proceedings of the 2001 CIE International Conference on Radar, pp. 675 -678, 2001.

[Mad95] V.K. MADISETTI, "Virtual prototyping of embedded DSP systems," Proc. of IEEE International Conference Acoustics, Speech, and Signal Processing, 1995.

[Mar95] GUSTAVO MARRERO CALLICÓ, Diseño y Evaluación de Arquitecturas SPARC en VHDL, Simulación y Síntesis, Master Thesis in Telecommunication Engineer, November 1995.

[MCE+02] M. MARRERO, P.P. CARBALLO, A. M. ESCUELA, GUSTAVO MARRERO CALLICÓ and ANTONIO NÚÑEZ, "A new approach for IPs design in microelectronics design," 4th European Workshop on Microelectronics Education. (EWME 2002), Vigo, Spain , pp. 153-156, May 2002.

[MCM+02a] M. MARRERO, P.P. CARBALLO, GUSTAVO MARRERO CALLICÓ and ANTONIO NÚÑEZ, "A design and reuse methodology for IP soft-cores with built-in performance metrics," IEEE Design and Diagnostics of Electronic Circuits and Systems (5th International Workshop IEEE DDECS 2002), Brno, Czech Republic, pp. 286-289, April 2002.

[MCM+02b] M. MARRERO, P.P. CARBALLO, GUSTAVO MARRERO CALLICÓ and ANTONIO NÚÑEZ, "A Quantitative Approach to Analyze and Bound the Synthesis-to-Layout Performance-Spread of Soft IP Cores," Proceedings of the XVII Conference on Design of Circuits and Integrated Systems (DCIS'02), pp. 449-454, Santander, Spain, November 2002.

[MF02] B. MARTINS and S. FORCHHAMMER, "A unified approach to restoration, deinterlacing and resolution enhancement in decoding MPEG-2 video, " IEEE Transactions on Circuits and Systems for Video Technology, vol.12, issue 9, pp. 803 -811, 2002

[MIP97] MIP Technologies, Inc., <http://www.mips.com/>, MIPS Extension for Digital Media with 3D," March 1997.

[ML85] D. MARTINEZ and J.S.LIM, "Implicit motion compensated noise reduction of motion video scenes," IEEE Proc. Int. Conf. Acoust. Speech and Signal Processing. pp. 375-378, Tampa, Florida, USA, 1985.

[MPEG1] ISO/IEC JTC1 CD 11172. "Coding of moving pictures and associated audio for digital storage media up to 1.5 Mbp/s". ISO 1992.

[MPEG4] MPEG-4 Part 2: Visual (IS 14496-2), doc. N2502, Atlantic City, N.J., USA, October 1998.

[MPF+96] JOAN L. MITCHELL, WILLIAN B. PENNEBAKER, CHAD E. FOGG and DIDIER J. LEGALL, MPEG Video Compression Standard, Chapman & Hall, 1996.

[MPN+02] GUSTAVO MARRERO CALLICÓ, RAFAEL PESET LLOPIS, ANTONIO NÚÑEZ, RAMANATHN SETHURAMAN and MARC OP DE BEECK, "A Low-Cost Implementation of Super-Resolution based on a Video Encoder," 28th Annual Conference of the IEEE Industrial Electronics Society (IECON 2002), vol. 2 , pp.1439-1444, Seville, Spain, November 2002.

[MPN+03a] GUSTAVO MARRERO CALLICÓ, RAFAEL PESET LLOPIS, ANTONIO NÚÑEZ and RAMANATHAN SETHURAMAN, "Low-Cost and Real-Time Super-Resolution over a Video Encoder IP," Proceedings of the 2003 4th International Symposium on Quality Electronic Design, ISQED'2003, pp. 79-84, San Jose, California, USA, March 2003.

[MPN+03b] GUSTAVO MARRERO CALLICÓ, RAFAEL PESET LLOPIS, ANTONIO NÚÑEZ ORDÓÑEZ and RAMANATHAN SETHURAMAN, "Mapping of Real-Time and Low-Cost Super-Resolution Algorithms on a Hybrid Video Encoder, " Proceedings of the SPIE 's 1st International Symposium on Microtechnologies for the New Millennium 2003, vol. 5117, pp. 42-52, Maspalomas, Spain, May 2003.

[MT97] J.D.MELLOT and F. TAYLOR, "Very long instruction word architectures for digital signal processing," Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP), 1997.

[NASA99] NASA Ames super-resolution on Mars Pathfinder IMP images, <http://ic.arc.nasa.gov/ic/projects/bayes-group/super-res/2d/mpf/ >

[NC89] A. NUÑEZ and D. CARNAL, "MVM : a GaAS microprocessor for critical real time applications," Microprocessing and Microprogramming, vol. 25, pp. 289-298, North Holland Elsevier Science Publishers, 1989.

[NMG01] NHAT NGUYEN, P. MILANFAR and G. GOLUB, "A computationally efficient superresolution image reconstruction algorithm," IEEE Transactions on Image Processing, vol. 10, issue 4 , pp. 573-583, Apr 2001.

[Nun95] A. NUÑEZ, "Tradeoffs in VLSI architectures for high performance signal processing," Proceeding of the 13th Australian Microelectronics Conference, pp. 123-135, Adelaida, The IEEE Society, 1995.

[NYY00] H. NAGAHARA, Y. YAGI and M. YACHIDA, "Super-resolution from an omnidirectional image sequence," 26th Annual Conference of the IEEE Industrial Electronics Society (IECON 2000), vol. 4 , pp. 2559 -2564, 2000.

[NYY02] H. NAGAHARA, Y. YAGI and M. YACHIDA, "Resolution improving method for a 3D environment modeling using omnidirectional image sensor," Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '02), vol. 1, pp. 900 -907, 2002.

[Oli98] S. OLIVIERI, "Enhanced Hybrid Recursive Motion Estimation for H.263 Video Coding", Philips Technical Note 253/98. Company Confidential.

[Pap77] A. PAPOULIS, "Generalized Sampling Theorem," WEE Trans. Circuits and Systems, vol. 24, pp. 652-654, November 1977.

[Per96] F. PEREIRA, "MPEG-4: A new challenge for the representation of audio-visual information," Proceedings of the Interntional Picture Coding Symposium, pp. 7-16, Melbourne, March 1996.

[PG98] R. PESET LLOPIS and K. G. W. GOOSSENS, "The Petrol Approach to High-Level Power Estimation," International Symposium on Low Power Electronics and Design, Monterey, CA, pp. 130-132, 1998.

[PH97] S. PELEG, J. HERMAN, "Panoramic mosaics by manifold projection," Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 338 -343, June 1997.

[Pir98] PETER PIRSCH, VLSI Implementations for Digital Signal Processing, John Wiley, 1998.

[PKL+99] R. PESET LLOPIS, R. KLEIHORST, P. LIPPENS, M. OOSTERHUIS and R. VAN DER VLEUTEN, "Architecture Overview of Picasso Design," Nat.Lab. Technical Note 336/99, December 1999. Company Restricted

[PKS87] S. PELEG, D. KEREN and L. SCHWEITZER, "Improving Image Resolution Using Subpixel Motion," Pattern Recognition Letters, vol. 5, pp. 223-226, March 1987.

[PKT83] J.A. PARKER, R.V. KENYON and D.E. TROXEL, "Comparison of interpolating methods for image resampling," IEEE Transactions on Medical Imaging, vol. 2, pp. 31-39, 1983.

[PLF+01] D. PASTINA, P. LOMBARDO, A. FARINA and P. DADDI, "Super-resolution of polarimetric SAR images of a ship," IEEE 2001 International Geoscience and Remote Sensing Symposium, IGARSS '01, vol. 5, pp. 2343-2345, 2001.

[POR+00] R. PESET LLOPIS, M. OOSTERHUIS, S. RAMANATHAN, P. LIPPENS, R. KLEIHORST, R. VAN DER VLEUTEN and J. LIN, "A Low-Cost Low-Power H.263 Video Encoder for Mobile Applications," Second International Symposium on Mobile Multimedia Systems & Applications, Delf, The Netherlands, Nov. 2000.

[POR+01] R. PESET LLOPIS, M. OOSTERHUIS, S. RAMANATHAN, P. LIPPENS, A. VAN DER WERF, S. MAUL and J. LIN, "HW-SW Codesign and Verification of a Multi-Standard Video and Image Codec," IEEE ISQED, San Jose, California, pp. 393-398, March 2001.

[Pra91] W. K. PRATT, Digital Image Processing, Wiley-Interscience Publishing, 1991.

[PRK98] J. PARK, S. RHEE and MOON GI KANG, "Multichannel dealiasing technique for superresolution," ITC98, 1998.

[PRRZ00] S. PELEG, B. ROUSSO, A. RAV-ACHA and A. ZOMET, "Mosaicing on adaptive manifolds," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, issue 10, pp.1144-1154, Oct 2000.

[PST94] A. J. PATTI, M. I. SEZAN and A. M. TEKALP, "High-Resolution Image Reconstruction from a Low-Resolution Image Sequence in the Presence of Time-Varying Motion Blur," Proceedings of TCIP, Austin - Texas, pp. 343-347, November 1994.

[PST95] A. J. PATTI, M. I. SEZAN and A. M. TEKALP, "High-Resolution Standards Conversion of Low Resolution Video," IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Detroit, MI., vol. II, pp. 2197-2200, May 1995.

[PST97] ANDREW J. PATTI, M. IBRAHIM SEZAN and A. MURAT TEKALP, "Super-resolution video reconstruction with arbitrary sampling lattices and nonzero aperture time," IEEE Transactions on Image Processing, vol. 6, no. 8, pp. 1064 – 1076, August 1997.

[PTS93] A. J. PATTI, A. M. TEKALP and M. T SEZAN, "Image Sequence Restoration and De-Interlacing by Motion-Compensated Kalman Filtering," SPIE, vol. 1903, 1993.

[Pur98] S. PURCELL, "The impact of Mpact 2," IEEE Signal Processing Magazine, vol. 15, issue 2, pp.102-107, March 1998.

[PW96] A. PELEG and U. WEISER, "MMX technology extension to the Intel architecture," IEEE Micro, August 1996.

[RAS] RASSP, "Rapid Prototyping of Application Specific Signal Processors," <http://rassp.scra.org/>

[RC01a] D. RAJAN and S. CHAUDHURI, "Simultaneous estimation of super-resolved intensity and depth maps from low resolution defocused observations of a scene," Proceedings of the 8th IEEE International Conference on Computer Vision (ICCV 2001), vol. 1 , pp. 113 -118, 2001.

[RC01b] D. RAJAN and S. CHAUDHURI, "Generation of super-resolution images from blurred observations using Markov random fields," Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, vol. 3 , pp. 1837 -1840, 2001.

[Rj01] DEEPU RAJAN, "Some new approaches to generation of super-resolution images," Ph.D. thesis, School of Biomedical Engineering, Indian Institute of Technology, Bombay, 2001.

[RK96] K. RÖNNER and J. KNEIP, "Architecture and applications of the HiPAR video signal processor," IEEE Transactions on Circuits and Systems for Video Technology, vol. 6, issue 1, February 1996.

[RPF97] B. ROUSSO, S. PELEG and I. FINCI, "Mosaicing with Generalized Strips," in DARPA Image Understanding Workshop, pp. 255-260, May 1997.

[RPFR98] B. ROUSSO, S. PELEG, I. FINCI and A. RAV-ACHA, "Universal mosaicing using pipe projection," 6th International Conference on Computer Vision, pp. 945 -950, January 1998.

[RS98] S. RATHNAM and G. SLAVENBURG, "Processing the new world of interactive media: The Trimedia VLIW CPU architecture," IEEE Signal Processing Magazine, vol.15, issue 2, pp. 108-117, March 1998.

[SAM+01] S.M. SHRESTHA, I. ARAI, T. MIWA, Y. TOMIZAWA, "Signal processing of ground penetrating radar using super resolution techniques", Proceedings of the 2001 IEEE Radar Conference, pp. 300 -305, 2001.

[SBZ+96] HASSAN SHEKARFOROUSH, MARE BERTHOD, JOSIANE ZERUBIA and MICHAEL WERMAN, "Sub-pixel bayesian estimation of albedo and height," International Journal of Computer Vision; vol. 19, no. 3, pp. 289–300, 1996.

[SBZ95] HASSAN SHEKARFOROUSH, MARE BERTHOD and JOSIANE ZERUBIA, "3D super-resolution using Generalized Sampling Expansion," in Proceedings of the International Conference on Image Processing, Washington D.C., pp. 300–303, 1995.

[Sch02] R.R. SCHULTZ, "Super-resolution enhancement of native digital video versus digitized NTSC sequences," Proceedings of the 5th IEEE Southwest Symposium on Image Analysis and Interpretation pp. 193 -197, 2002

[SCMM00] V.N. SMELYANSKIY, P. CHEESEMAN, D.A. MALUF and R.D. MORRIS, "Bayesian super-resolved surface reconstruction from images," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, vol. 1, pp. 375 -382, 2000.

[Ses98] N. SESHAN, "High VelociTI processing," IEEE Signal Processing Magazine vol. 15, issue 2, pp. 86-101, 117, March 1998.

[SHN93] P. SEMENTILLI, B. HUNT and M. NADAR, "Analysis of the limit to super-resolution in incoherent imaging," Journal of Optical Society of America, Series A, vol. 10, pp. 2265 – 2276, 1993.

[Sin92a] A. SINGH, Optic Flow Computation: A Unified Perspective, WEE Computer Society Press, 1992.

[Sin92b] A. SINGH, "Incremental Estimation of Image Flow Using the Kalman Filter," J. Visual Communication and Image Representation, vol. 3, pp. 39-57, 1992.

[SK95] W. SUNG and K. KUM, "Simulation-based word-length optimization method for fixed-point digital signal processing systems," IEEE Transactions on Signal Processing, vol. 43, pp. 3087-3090, December 1995.

[SLJ+00] WANG SHENGLI, ZHU LI, NI JINLIN and ZHANG GUANGYI, "The study and experiment on azimuth super-resolution using phase information," 5th International Conference on Signal Processing Proceedings WCCC-ICSP 2000, vol. 3, pp. 1868 -1871, 2000.

[SM95] G.A. SHAW and V.K. MADISETTI, "Assessing and improving current practice in the design of application specific signal processors," The RASSP Digest, vol. 2, issue 1, January 1995.

[SO89] HENRY STARK and PEYMA OSKOUI, "High-resolution image recovery from image-plane arrays, using convex projections," Journals of the Optical Society of America, vol. 6, no. 11, pp 1715-1726, November 1989.

[SO89] H. STARK and P. OSKUI, "High-resolution image recovery from image-plane arrays using convex projections," J. Optical Society of America, vol. 6, no. 11, pp. 1715 – 1726, Nov. 1989.

[Sor93] N. SOREK, "Image Motion Compensation Using Multiple exposures," Masters Thesis, The Technion - Israel Institute of Technology, 1993.

[SPM98] "IEEE Signal Processing Magazine, The Latest Word in Multimedia," vol. 15, 1998.

[SS90] C. SRINIVAS AND M. D. SRINATH, "A Stochastic Model-Based Approach for Simultaneous Restoration I-Multiple Miss-registered Images," SPIE, vol. 1360, pp. 1416-1427, 1990.

[SS94] R. R. SCHULTZ and R. L. STEVENSON, "A Bayesian approach to image expansion for improved definition," IEEE Transactions on Image Processing, vol. 3, no. 3, pp. 233–242, May 1994.

[SS95] R. S. SCHULTZ and R. L. STEVENSON, "Improved Definition Video Frame Enhancement," IEEE Conference on Acoustics, Speech and Signal processing (ICASSP), Detroit, MI., vol. 10, pp. 2169-2172, May 1995.

[SS96] R. S. SCHULTZ and R. L. STEVENSON, "Extraction of High-Resolution Frames from Video Sequences," IEEE Transaction on Image Processing, vol. 5, pp. 996-1011, June 1996.

[SSO99] A. SMOLIC, T. SIKORA and J.-R. OHM, "Long-term global motion estimation and its application for sprite coding, content description, and segmentation," IEEE Transactions on Circuits and Systems for Video Technology, vol. 9 Issue 8 , pp. 1227 -1242, Dec 1999.

[SSS00] D. SALE, R. R. SCHULTZ and R.J. SZCZERBA, "Super-resolution enhancement of night vision image sequences, " IEEE International Conference on Systems, Man, and Cybernetics, vol. 3 , pp. 1633 -1638, 2000.

[SVR+97] P. SCHAUMONT, S. VERNALDE, L. RIJNDERS, M. ENGELS and I. BOLSENS, "A programming environment for the design of complex high speed ASICS," Proceedings of DAC, Anaheim, 1997.

[SW01] A. SMOLIC and T. WIEGAND, "High-Resolution Video Mosaicing," Proceedings of the IEEE International Conference on Image Processing ICIP2001, Thessaloniki, Greece, October 2001.

[SZ99] N. R. SHAH and AVIDEH ZAKHOR, "Resolution enhancement of colour video sequences," IEEE Transactions on Image Processing, vol. 8, no. 6, pp. 879–885, June 1999.

[Sze96] R. SZELISKI, "Video mosaics for virtual environments," IEEE Computer Graphics and Applications, vol. 16, issue 2, pp. 22 -30, Mar 1996.

[TA+98] M. TAKAHASHI et.al, "A 60-mW MPEG-4 video codec with clustered voltage scaling with variable supply voltage scheme," IEEE Journals of Solid-State Circuits, vol. 33, no. 11, pp. 1772-1780, November 1998.

[Tek95] A. M. TEKALP, Digital Video Processing, Prentice Hall Signal Processing Series, 1995.

[Ter86] D. TERZOPOELOS, "Image Analysis Using Multigrid Relaxation Methods," IEEE Trans. Pattern analysis and Machine Intelligence, vol. 8, pp. 129-139, 1986.

[TFN+00] Y, TANABE, K. FUJISHIMA, T. NISHIMURA, Y. OGAWA and T. OHGANE, "An adaptive antenna for spatial-domain path-diversity using a super-resolution technique," Proceedings of Vehicular Technology Conference VTC, IEEE 51st, vol. 1, pp. 1-5, Tokyo, 2000

[Tho94] SGS-Thomson Microelectronics, "MPEG-2/CCIR 601 Video Decoder," Preliminary Notes, June 1994.

[TK95] BRIAN C. TOM and AGGELOS K. KATSAGELOS, "Reconstruction of a high-resolution image by simultaneous registration, restoration and interpolation of low-resolution images," in Proceedings of International Conference on Image Processing, Washington D.C., pp. 539-542, 1995.

[TLA+01] A. J. TATEM, H.G. LEWIS, P.M. ATKINSON, M.S. NIXON, "Super-resolution mapping of multiple-scale land cover features using a Hopfield neural network," IEEE International Geoscience and Remote Sensing Symposium, IGARSS '01, vol. 7, pp. 3200 -3202, 2001.

[TNB02] C.A. TAN, A.R. NIX and M.A. BEACH, "Dynamic spatial-temporal propagation measurement and super-resolution channel characterisation at 5.2 GHz in a corridor environment," Proceedings of the Vehicular Technology Conference VTC 2002-Fall, IEEE 56th, pp. 797 -801, vol. 2 , 2002.

[TONH96] M. TREMBLAY, J.M. O'CONNOR, V. NARAYANAN AND L. HE, "VIS speeds new media processing," IEEE Micro, vol. 16, issue 4, pp. 10-20, August 1996.

[TOS92] A. M. TEKALP, M. K. OZKAN and M. I. SEZAN, "High-Resolution Image Reconstruction from Lower-Resolution Image Sequences and Space Varying Image Restoration," IEEE International Conference on Acoustics, Speech and Signal processing (ICASSP), San-Francisco, CA., vol. 3, pp. 169-172, March 1992

[Toy94] M. TOYOKURA, "A video DSP with a macroblock-level-pipeline and a SIMD type vector-pipeline architecture for MPEG-2 CODEC," IEEE Journal of Solid-State Circuits, no. 29, vol. 12, 1994.

[Tri01] B. TRIGGS, "Empirical filter estimation for subpixel interpolation and matching," Proceedings of the 8th IEEE International Conference on Computer Vision ICCV, vol.2, pp. 550 -557, 2001.

[UE90] M. UNSER and M. EDEN, "Weighted averaging of a set of noisy images for maximum signal to noise," IEEE Transaction on ASSP (Acoustic Speech and Signal Processing) no. 5, vol. 38. pp. 890-895, May 1990.

[UG92] H. UR and D. GROSS, "Improved Resolution from Sub-pixel Shifted Pictures," CVGIP: Graphical Models and image Processing, vol. 54, pp. 181-186, March 1992.

[VH99] LUCAS J. VAN VLIET and CRIS L. LUENGO HENDRIKS, "Improving spatial resolution in exchange of temporal resolution in aliased image sequences," in Proceedings of the 11th Scandinavian Conference on Image Analysis, Kaugerlussauaq, Greenland, pp. 493-499, 1999.

[VIS] Sun Microsystems, Inc. "The Visual Instruction Set: on-chip support for new media processing". Whitepaper 95-022, <http://www.sun.com/sparc/vis>.

[WDM99] WIRAWAN, PIERRE DUHAMEL and HENRI MAITRE, "Multi-channel high resolution blind image restoration," in Proceedings of IEEE ICASSP, Arizona, USA, pp. 3229-3232, 1999.

[WL00] R. WU and J. LI, "Autofocus and super-resolution synthetic aperture radar image formation", IEE Proceedings on Radar, Sonar and Navigation, vol. 147, issue 5 , pp. 217 -223, Oct 2000

[WN94] D. O. WALSH and P. NIELSEN-DELANEY, "Direct method for super-resolution," Journal of Optical Sac. of America, Series A, vol. 11, no. 5, pp. 572 – 579, 1994.

[WWV+97] A. VAN DER WERF , E. WATERLANDER, M. VERSTRAELEN, T. FRIEDRICH, F. BRÜLS and R. TAKKEN, "I.McIC: a single-chip MPEG2 video encoder for storage," IEEE Journals of Solid-State Circuits, vol. 32, no. 11, pp. 1817-1823, November 1997.

[XFH+01] YI XU, WEI FENG FENG, JUN YAN HAO, H.K. HWANG, "Direction of arrival estimation using super-resolution algorithm," MTS/IEEE Conference and Exhibition OCEANS, vol. 2, pp. 749 -755, 2001.

[Yen56] L. J. YEN, "On Nonuniform Sampling of Bandwidth Limited Signals," IRE Trans. Circuits Theory, vol. 3, pp. 251-257, April 1956.

[You78] D. C. YOULA, "Generalized Image Restoration by the Method of Alternating orthogonal Projections," IEEE Transactions on Circuits & Systems, vol. 25, pp. 694-702, 1978.

[YYR+01] H. YAMADA, Y. YAMAGUCHI, E. RODRIGUEZ, Y. KIM and W.M. BOERNER, "Polarimetric SAR interferometry for forest canopy analysis by using the super-resolution method," IEEE 2001 International Geoscience and Remote Sensing Symposium, IGARSS '01, vol. 3 , pp. 1101 -1103, 2001.

[Zer92] M. B. ZERVAKIS, "Optimal Restoration of Multichannel Images based on Constrained Mean-Square Estimation," Journal of Visual Communication and Image Representation, vol. 3 pp. 392-411, December 1992.

[ZP00] A. ZOMET and S. PELEG, "Efficient super-resolution and applications to mosaics," Proceedings of the 15th International Conference on Pattern Recognition, vol. 1 , pp. 579 -583, 2000.

[ZP02] A. ZOMET and S. PELEG, "Multi-sensor super-resolution," Proceedings of the 6th IEEE Workshop on Applications of Computer Vision (WACV 2002), pp. 27 -31, 2002.

[ZP98] A. ZOMET and S. PELEG, "Applying super-resolution to panoramic mosaics," Proceedings of the 4th IEEE Workshop on Applications of Computer Vision (WACV '98), pp. 286 -287, Oct 1998.

[ZRP01] A. ZOMET, A. RAV-ACHA and S. PELEG, "Robust super-resolution," Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2001), vol. 1, pp. I-645 -I-650, 2001.

[ZSHS02] W. ZHAO, H. SAWHNEY, M. HANSEN and S. SAMARASEKERA, "Super-fusion: a super-resolution method based on fusion," Proceedings of the 16th International Conference on Pattern Recognition, vol. 2, pp. 269 -272, 2002.