Reidentificación de Personas y Objetos mediante Aprendizaje Semisupervisado

---

Proyecto Fin de Carrera
Septiembre de 2015

Luis María González Medina

Tutores:
Javier Lorenzo Navarro
Modesto Castrillón Santana

Agradecimientos

En primer lugar he de agradecer a mis tutores, por su paciencia y apoyo. Les agradezco por realizar reuniones semanales que me obligaron a trabajar semana a semana, o al menos, el día antes de la reunión. Gracias por el tiempo y esfuerzo dedicado. Gracias también a los compañeros de las reuniones semanales, Alberto, Eduardo, Pedro, Javier y Daniel que semana a semana han estado ahí.

Por supuesto he de agradecer a mis compañeros de batallas, aguantando incluso el calor de agosto trabajando en el proyecto mientras el resto disfrutaba de la playa. Entre todos nos motivamos y ayudamos para conseguir llegar a este punto. Gracias Eliezer, Gabriel Yeray y Rubén. También al resto de amigos: Yarilo, Aarón, David, Yeray, Omar... Todos me han ayudado a llegar a este punto. Gracias también a ti, Iago, por estar siempre ahí a pesar de la distancia.

Por supuesto al resto de amigos que no menciono pero están ahí y me han apoyado. No creo que lleguen a leer esto, pero aun así gracias.

A mi familia que me ha apoyado y aguantado durante el largo tiempo que ha tomado finalizar este proyecto. Siempre han creído en mí y me han visto capaz de conseguir lo que quisiera. Ese apoyo sincero y desinteresado que sólo la familia puede aportar. Dar las gracias poco significa, pero no por ello dejaré de decirlo: ¡GRACIAS!

Índice general

Bibliografía **147**

Índice de tablas

Índice de figuras

Capítulo 1

Introducción

La reidentificación es el proceso por el cual se intenta volver a identificar un objeto/persona que ya se ha detectado previamente, normalmente en condiciones diferentes a la primera detección. Así, un ejemplo ilustrativo es aquél en el que se quiere identificar a una misma persona desde distintas cámaras: en primer lugar se detecta a esta persona en una cámara, siendo el proceso de reidentificación el encontrar a esta misma persona desde otras cámaras, o desde la misma cámara en momentos posteriores y condiciones diferentes. No se busca asignar una identidad única, si no detectar que se trata de la misma persona en ambas detecciones. En este proyecto se busca realizar este proceso, en medida de lo posible, de forma automática.

Inicialmente se plantea este proyecto para detectar a personas y objetos, pero pronto se decide centrarlo en la reidentificación de personas. Reidentificar a personas y objetos son dos tareas diferentes, por las características propias de éstas. Así, conseguir un sistema general para ambos casos se plantea difícil. La reidentificación de personas plantea grandes retos. Las variaciones de pose o iluminación entre las imágenes de la misma persona tomadas desde diferentes cámaras hacen que esta tarea no sea trivial. Si se tiene en cuenta que su principal aplicación es en escenarios de vídeo vigilancia, se verá como las imágenes que se obtienen suelen ser de baja calidad y resolución, lo que dificulta aun más la reidentificación automática.

A lo largo del desarrollo de este proyecto, se decide enfocar éste a la creación de un Framework para la reidentificación de personas utilizando el lenguaje de programación Python. Se denomina PyReID y se encuentra disponible

en GitHub para que cualquier investigador/desarrollador pueda acceder a su código fuente, utilizarlo y añadir los métodos y modificaciones que desee.

Utilizando esta plataforma que se ha ido construyendo a lo largo del proyecto, se han añadido múltiples métodos usados en el ámbito de la reidentificación de personas. No sólo se añaden métodos ya usados normalmente en el campo, si no que se prueban nuevas técnicas para comprobar su posible utilidad.

Además de la reidentificación de personas, se añade la posibilidad de realizar reordenación. Ésta consiste en que el usuario, una vez el sistema genera los resultados automáticamente, puede darle al sistema un cierto información, de forma que el sistema pueda usar ésta para mejorar los resultados.

Como se verá a lo largo de este documento se consiguen buenos resultados en ciertos aspectos comparando a otros trabajos de reidentificación, como la segmentación de las personas o la combinación de la normalización de la iluminación en distintos espacios de color, que aumentan la precisión de la reidentificación. También se observará cómo los métodos de reordenación son un área interesante en el que seguir explorando en el futuro, ofreciendo resultados esperanzadores.

1.1. Objetivos

Este proyecto se ha centrado en la reidentificación de personas. Los objetivos son los siguientes:

- Implementación de métodos usados en el campo de la reidentificación de personas:
  - Usar métodos ya existentes. Realizar pruebas automáticas variando los parámetros para descubrir su influencia e importancia.
- Proponer algún método que no se haya usado o intentar mejorar alguno existente. En este punto se intentará innovar en medida de lo posible.
- Aplicar técnicas de aprendizaje semisupervisado. Esto se realizará con un método de reordenación.

- Aplicar alguna técnica de Ingeniería del Software. Se tiene un especial interés en este área así que se intentará a la hora de desarrollar el framework utilizar técnicas para escribir un mejor código.

- Creación de un Framework para la reidentificación de personas que permita:

    - Combinar múltiples métodos y técnicas de forma sencilla.
    - Añadir nuevos métodos, siguiendo una estructura predefinida.
    - Obtener estadísticas y resultados en un formato legible, independientemente de las técnicas usadas.
    - Permitir la automatización sencilla para la realización de múltiples pruebas.

1.2. Reidentificación de personas

La reidentificación de personas es una tarea básica para la vídeo vigilancia automática o asistida y se trata de una área de intensa investigación actualmente. Dada una imagen o vídeo de una persona tomada desde una cámara, la reidentificación es el proceso por el cual se detecta a esta misma persona en imágenes/vídeos tomadas por otra cámara diferente, o la misma, en un instante de tiempo posterior y condiciones diferentes. La reidentificación tiene cabida en el proceso de etiquetado de forma consistente entre diferentes cámaras e incluso entre la misma cámara para diferentes grabaciones/imágenes. Aparte del campo de la vídeo vigilancia, la reidentificación de personas también tiene aplicación en robótica, multimedia y para el análisis forense digital.

La reidentificación se trata de un problema muy complejo dada la ambigüedad visual y la incertidumbre espacio-temporal de una persona entre diferentes cámaras. Esto es, los cambios de apariencia en diferentes momentos. Estas dificultades se ven incrementadas por la escasa calidad de vídeo, normalmente de baja resolución, lo que no ayuda al proceso de reidentificación.

La reidentificación de personas pretende asignar un ID único a cada individuo que aparece en una cámara, y que este ID se mantenga cuando el mismo

Figura 1.1: Ejemplos de individuos del conjunto de datos VIPeR desde dos cámaras diferentes [An et al, 2013]

individuo sea capturado por otra cámara diferente. Este concepto es sencillo de entender, ya que las personas lo realizamos continuamente. Nuestro cerebro está entrenado para detectar, localizar, identificar y posteriormente reidentificar personas y objetos en el mundo real. La reidentificación implica que una persona que ya ha sido vista sea identificada con una apariencia diferente, como puede ser la variación de la pose o la iluminación, usando un descriptor único de la persona. Los humanos podemos extraer este descriptor basándonos en la cara de la persona, altura, composición de la persona, la ropa, el pelo, la forma de caminar, etc. La cara de la persona es la característica mas fiable que usan los humanos para identificar a otra persona. Sin embargo en el escenario de la vídeo-vigilancia es difícil de usar dada la baja resolución o calidad de las imágenes. En la figura 1.1 se pueden ver varios ejemplos de las mismas personas captadas desde dos cámaras diferentes.

La automatización de la reidentificación de personas sin embargo es bastante difícil de conseguir sin la intervención humana. Es por ello que muchos trabajos se enfocan en mejorar todo el proceso y no buscan la automatización completa. Estos trabajos parten de la premisa de que conseguir que el sistema funcione de forma completamente autónoma es bastante difícil dada la complejidad de la tarea y las limitaciones técnicas. Por ello en muchos casos estos sistemas muestran los resultados ordenados a un operador humano, que selecciona entre todas las imágenes la correcta. Algunos métodos, explorados en este proyecto, proponen la intervención humana para mejorar los resulta-

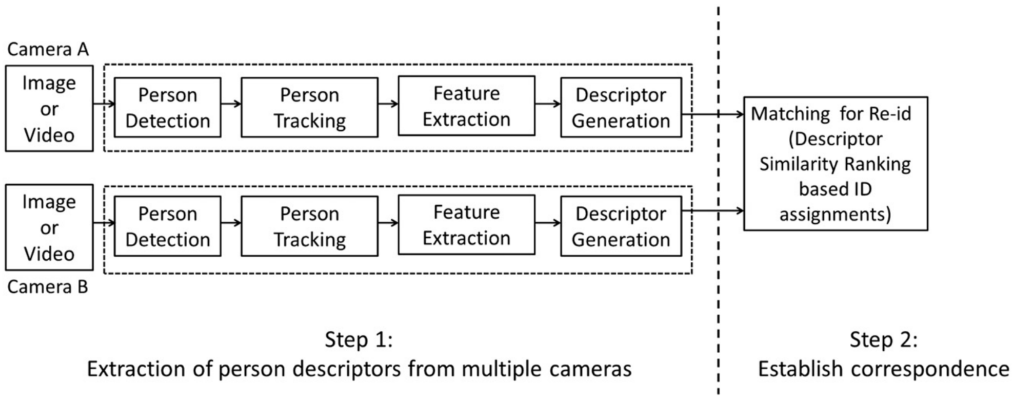


Figura 1.2: Proceso de Reidentificación [Bedagkar-Gala and Shah, 2014].

dos obtenidos previamente de forma automática. Con ello se consigue reducir el tiempo de búsqueda por parte del operador.

Hoy en día grandes redes de cameras se pueden encontrar en muchos sitios públicos como aeropuertos, centros comerciales o campus universitarios. Normalmente monitorizan diferentes zonas que no están superpuestas e intentan cubrir la máxima área posible. Estas cámaras proveen grandes cantidades de vídeo e imágenes, que serán analizadas por las autoridades o utilizadas tras el acontecimiento de un hecho con objetivos forenses, como puede ser la comisión de un delito. El análisis automático se vuelve fundamental para agilizar y optimizar este problema.

En [Bedagkar-Gala and Shah, 2014] se resume de forma esquemática el proceso de la reidentificación de la forma mostrada en la figura 1.2. En la imagen se puede ver cómo se tienen dos cámaras (pueden ser más de dos) y para cada una de ellas se realiza el mismo proceso. Éste consiste inicialmente en detectar, en una secuencia de vídeo o imágenes, a una persona y realizar un seguimiento de ésta, para evitar considerarla como dos personas diferentes y para así obtener más imágenes de este mismo individuo. Una vez se tiene la imagen de la persona, se obtiene un conjunto de características que interesen de ésta. Uniendo estas características, se consigue definir un descriptor para esa persona. A partir de ahí se intenta establecer la correspondencia entre dos individuos de distintas cámaras usando sus descriptores.

Generalmente se suele separar la parte de detección y seguimiento del resto. Así en el problema de la reidentificación se suele trabajar con conjuntos de

datos en los que ya se ha realizado el proceso de detección de las personas. Estos conjuntos de datos son conjuntos de imágenes tomadas por distintas cámaras del entorno en el que se va a trabajar. Para la evaluación se suele dividir en dos conjuntos las imágenes. El primero es considerado *muestra* y el segundo galería. El problema entonces se enfoca en encontrar, para cada elemento de la muestra, su correspondiente en la galería. Normalmente lo que se hace es que para cada elemento de la muestra se le asigna una lista ordenada de los elementos de la galería (ordenación), siendo el primer elemento de esta lista el que se considera como la correspondencia más probable para ese elemento de la muestra en la galería, es decir, son el mismo individuo. Como se verá, para medir la calidad de un método se usarán métricas basadas en esta lista ordenada.

Tanto la muestra como la galería de un conjunto de datos contienen imágenes de múltiples individuos. Sin embargo, si en cada conjunto sólo se tiene una imagen por individuo esto se conoce como caso *SingleShot*. Otro caso posible es que un mismo individuo disponga de múltiples capturas, caso *MultiShot*. Puede darse la combinación de ambos casos, por ejemplo que la muestra sea Single-Shot y la galería Multi-Shot, aunque generalmente la muestra y la galería suelen ser del mismo tipo. Los métodos que se apliquen han de tener en cuenta esta condición, pues son problemáticas ligeramente diferentes.

En [Bedagkar-Gala and Shah, 2014] se comenta que la mayor parte de la investigación en reidentificación actual se centra en la similitud de la apariencia entre dos imágenes, como puede ser los colores de la imagen. Sin embargo este tipo de comparaciones sólo tiene sentido en cortos intervalos de tiempo, pues la apariencia puede cambiar en espacios de tiempo más largo. Un mismo individuo cambiará de ropa entre días distintos, pero es probable que mantenga su apariencia durante el mismo día. Las investigaciones recientes en reidentificación se centran en el problema en el corto espacio de tiempo.

Los métodos para solucionar el problema de la reidentificación se pueden clasificar en dos grupos: métodos contextuales y métodos no contextuales. Se puede ver un diagrama de los tipos de métodos en la figura 1.3.

Los métodos contextuales hacen uso de la información que puedan usar de la cámara y el escenario. Se puede estudiar la geometría de la zona grabada y hacer un mapa. Esto aportará información cuando aparezca una persona, como la altura. También se pueden hacer cálculos probabilísticos para determinar cuánto tiempo tarda una persona en pasar entre dos cámaras. Este

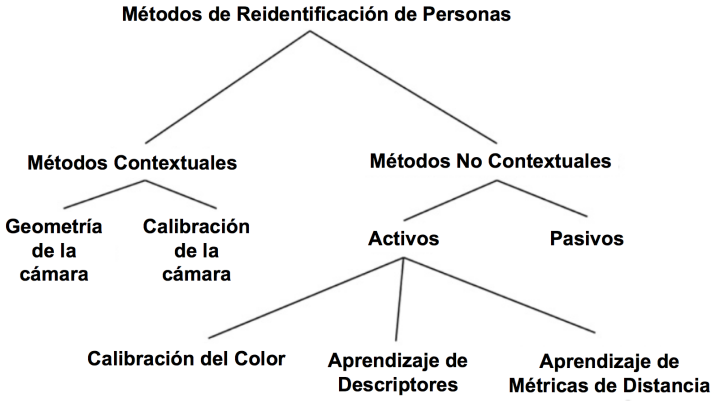


Figura 1.3: Métodos para solucionar el problema de la reidentificación [Bedagkar-Gala and Shah, 2014].

tipo de métodos requieren sin embargo de un esfuerzo previo considerable, pues se necesita realizarlo para cada cámara.

Los métodos no contextuales se basan en el uso de los descriptores visuales sin utilizar ningún tipo de información contextual. Por tanto son de uso general. Es por ello que el esfuerzo investigador suele centrarse en este tipo de métodos. A su vez, éstos se pueden clasificar a grandes rasgos en pasivos y activos.

Los métodos pasivos son aquellos que generan descriptores visuales para categorizar a las personas y luego utilizan medidas de similitud para reidentificar. Estos métodos se conocen como pasivos porque no utilizan ningún tipo de aprendizaje, supervisado o no, para la extracción de las características ni para calcular las similitudes.

Los métodos activos difieren de los pasivos en que utilizan alguna técnica de aprendizaje en la extracción de características o en el cálculo de la similitud. Los denominados como de Calibración del Color intentan modelar la relación entre los colores de dos cámaras diferentes y necesitan una fase de aprendizaje para calcular esta relación. En los métodos de aprendizaje de descriptores se intenta aprender cuáles son las características más discriminantes o un esquema de pesos para múltiples características. Por último existen aquellos que intentan mejorar los métodos de comparación, en lugar de centrarse en las características a comparar. Los métodos de aprendizaje en medidas de

distancias se usan mucho en el reconocimiento de imágenes e intentan aprender una métrica en el espacio definido por las características de la imagen. El mayor problema de estos métodos es que requieren de un aprendizaje. En muchos casos este aprendizaje es supervisado y esto requiere de un etiquetado previo de muchas imágenes. Sin embargo este tipo de método consigue los mejores resultados.

Otro tipo de métodos, menos explorados en el campo, son los procesos de reordenación. Como se ha descrito, en el proceso de la reidentificación se parte de un elemento, de la muestra, y se intenta alcanzar una coincidencia en un conjunto de imágenes, la galería. Así se genera para cada elemento de la muestra una ordenación de los elementos en la galería, es decir, se ordena la galería en base a la similitud con la muestra. Los métodos de reordenación entran en juego en este punto. Parten de este ordenación inicial y permiten la interacción del usuario. Éste evalúa de alguna forma los resultados obtenidos y el método reordena los elementos en base a esta información. Así, se tendrá la ordenación inicial, generada por el método inicialmente, y sobre esta se aplica la reordenación, que suele ser independiente de la ordenación inicial, y modifica los resultados obtenidos usando información provista por un usuario. En este proyecto se explorará este campo y se aplicarán diversas técnicas, como el aprendizaje semisupervisado.

1.3. Estado del arte

El estado del arte en la materia es muy extenso. El interés por el problema de la reidentificación de personas ha sido creciente en lo últimos años. Se comentan algunos trabajos especialmente interesantes, mencionando especialmente aquellos que tienen relación con lo realizado en este proyecto.

Para realizar las pruebas se necesita un conjunto de datos. Uno de los más comunes es VIPeR, introducido por [Gray et al, 2007]. Se compone de 632 pares de imágenes y se describe en la sección 5.1.1 con más detalle.

Uno de los mayores problemas en la reidentificación son los cambios de iluminación y características visuales entre las imágenes de dos individuos capturadas por cámaras diferentes. Teniendo en cuenta que la mayoría de métodos de reidentificación se basan principalmente en encontrar las correspondencias de colores entre imágenes, los cambios en iluminación producen una alta

variación en estos, dificultando la tarea de la reidentificación. Para solucionar este problema se propone buscar las funciones de transferencia de brillo, *BTF* por sus siglas en inglés (*Brightness Transfer Function*). Para calcular esta función se necesitan al menos un par de observaciones, una para cada cámara, con el mismo individuo. Cuando se tienen múltiples observaciones se puede obtener una función de transferencia de brillo.

En [Chong et al, 2008], aunque no específicamente aplicado a la reidentificación, se propone una transformación en el espacio de color para representar las imágenes en un espacio diferente basado en la percepción humana y que es robusto ante cambios de iluminación.

En [Farenzena et al, 2010] se presenta un método, *SDALF*, basado en la apariencia para realizar la reidentificación. Consiste en una combinación de varios criterios que son combinados para obtener resultados mejores. El primero de ellos es utilizar histogramas ponderados, utilizando una distribución Gaussiana, lo que dará mayor importancia a los píxeles más cercanos al centro del individuo. Para ello realizan una separación de la persona en parte alta y parte baja. Otro criterio usado es la búsqueda de regiones de colores estables. El último criterio usado se basa en comparar las texturas. Estos últimos aportan menos información y no son explorados en este trabajo.

[Wei and Lin, 2013] propone retomar la idea de los histogramas ponderados pero utiliza un doble núcleo Gaussiano, mientras en SDALF se usa un núcleo simple. Además introduce la normalización de la iluminación en las imágenes, lo que mejora sustancialmente los resultados. Sus ideas han sido aplicadas en este proyecto.

En [Liu et al, 2013] se realiza un proceso de reordenación en la que el usuario participa activamente. Primero se produce la ordenación inicial, que puede ser cualquier método para ordenar todas las imágenes de la galería con respecto a cada una de la muestra. Este resultado se le muestra al usuario que tiene que etiquetar manualmente algunas muestras. Éstas serán aquellas que el usuario considere parecidas al elemento de la muestra y las que considere poco parecidas. Con esto, el método intenta recalcular la ordenación.

En [Wang et al, 2014] se utiliza un método de reordenación similar al anterior pero basándose en el marcado por parte del usuario sólo de regiones similares de las imágenes, no de la imagen completa. En sus experimentos divide las imágenes en 3 regiones: cabeza, torso y piernas. El usuario selecciona aquellas partes de la imagen que son similares al elemento del Probe, aunque el resto

de la imagen no coincida. Según los resultados que muestran los autores esto facilita el etiquetado al usuario.

En el área de los métodos basados en aprendizaje de distancias, en [Kostinger et al, 2012] proponen el método *KISSME*, un método que necesita supervisión pero consigue muy buenos resultados. Además comparado con métodos anteriores es computacionalmente óptimo. Sin embargo, estos resultados los consigue utilizando la mitad del conjunto de datos como conjunto de entrenamiento. En [Tao et al, 2013] mejoran el método y éste funciona mejor con menos muestras para el aprendizaje. Como ejemplo, en VIPeR hacen pruebas con 100 muestras de entrenamiento (frente a las 316 usadas en KISSME). Sin embargo este tipo de propuestas no son exploradas en este proyecto. En un escenario real estas propuestas necesitan de un etiquetado previo de muchas imágenes de muestra para cada par de cámaras, lo que dificulta su utilidad.

Respecto a la segmentación de la imagen, la separación de la persona con respecto al fondo, se han utilizado diversas técnicas en el campo de la reidentificación. El método propuesto en [Jojic et al, 2009] y usado en el método SDALF, requiere de un aprendizaje. En esencia este método localiza las partes comunes, en este caso el cuerpo de las personas. En SDALF calculan este método sobre una base de datos (no usada para los cálculos) y luego la aplican en el conjunto de datos a utilizar.

Otro método, no muy usado en reidentificación, es segmentar usando GrabCut. En [Bąk et al, 2012] se aplica para obtener la silueta de la persona, aunque los resultados no son muy buenos. Este método, propuesto en [Rother et al, 2004] se basa en los cortes en grafos y necesita una interacción del usuario. En este caso, será una «máscara» inicial que indique al método lo que debe ser fondo y lo que debe ser persona y a partir de esta información realiza la segmentación automática. Será explorado en este proyecto con detalle.

Capítulo 2

Herramientas de desarrollo

Este capítulo se centra en describir todas las herramientas que se han utilizado. Serán fundamentalmente software, enfocándose en describir las principales bibliotecas utilizadas. También se hará hincapié en herramientas que se han usado en el transcurso de este proyecto pero que no tienen que ver con el desarrollo en sí mismo, sino que han sido herramientas de apoyo. Por último se mencionarán los equipos en los que se han realizado las pruebas y el desarrollo.

2.1. OpenCV

OpenCV [Bradski, 2000], Open Source Computer Vision, es una biblioteca de funciones para la visión por computador. Originalmente fue desarrollada

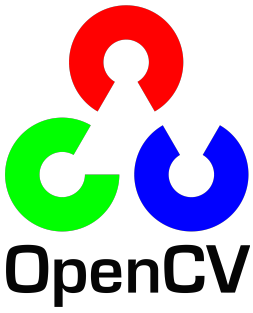


Figura 2.1: Logo de OpenCV [Bradski, 2000].

por Intel y actualmente es mantenida por *Itseez*. Como aparece en su propio nombre, es código abierto y se distribuye bajo la licencia BSD. Su código fuente puede encontrarse en la plataforma GitHub.

Está desarrollada completamente en C++ pero tiene desarrolladas interfaces a otros lenguajes. Estos son Java, MATLAB y Python. Esta última está bastante bien desarrollada y es la versión que usaremos. Además soporta múltiples sistemas operativos, desde los comunes Windows, Linux y OS X a sistemas operativos móviles como Android e iOS.

Su última versión a la fecha es la 3.0, aunque a lo largo de este proyecto se han producido migraciones desde la versión 2.4.10 pasando por versiones alfa y beta de la 3.0.

Esta biblioteca fue diseñada para ser eficiente y aprovecha las ventajas de los procesadores actuales como los múltiples núcleos así como el uso de OpenCL o CUDA para acelerar la ejecución.

El principal uso que se hace en este proyecto será para el cálculo de histogramas, cambios de espacio de color y segmentación de las imágenes.

2.2. Python

Figura 2.2: Logo de Python [PSF, 2015].

Python [PSF, 2015] es un lenguaje de alto nivel de propósito general. Su filosofía es hacer el código legible y por ello permite expresar en una cantidad de lineas menor el código, comparándolos con otros lenguajes como Java o C++. Python es un lenguaje moderno que soporta la orientación a objetos, programación funcional y procedimental y muchos otros paradigmas de programación.

Además la comunidad científica está muy activa, con la creación de bibliotecas como *Scipy* o *Scikit-learn*. Estas bibliotecas y la facilidad de manejar

y entender Python llevó a la decisión de utilizar este lenguaje como base para este proyecto. Si bien se inició en C++, este lenguaje supone una cierta barrera de entrada para personas no especialmente experimentadas en la programación. Uno de los objetivos de este proyecto es crear una plataforma que sea fácilmente reutilizable, ampliada y comprendida por la comunidad. Python es un gran aliado para este objetivo. Además, facilita el desarrollo y la velocidad a la que se genera el código.

Como contrapartida cabe destacar que la velocidad de ejecución es varios órdenes de magnitud inferior a C++. Python es un lenguaje interpretado y su objetivo no es ser un lenguaje rápido. Sin embargo, estos problemas de velocidad pueden ser solventados utilizando *Cython*. Realizando pequeños cambios a nuestro código y compilándolo con *Cython* se obtienen mejoras sustanciales. Por falta de tiempo no se ha hecho uso de esta herramienta, pero sin duda es una de las futuras líneas de mejora (sección 6.2.3).

2.3. Bibliotecas de Python

Se comentarán las principales bibliotecas de Python usadas y el uso principal que se hace en este proyecto.

1. **MatplotLib** [Hunter, 2007]: Biblioteca para dibujar gráficas a partir de los datos que obtenemos. Está basada en la interfaz de Matlab, muy popular para este tipo de tareas.

2. **Numpy** [van der Walt et al, 2011]: Biblioteca matemática que proporciona clases y tipos que facilitan sobremanera los cálculos realizados. Especialmente útil por sus clases de vectores y matrices. Por ejemplo, es la base para la clase imagen que se usa en el proyecto. OpenCV requiere *Numpy* para funcionar en Python.

3. **Scikit-learn** [Pedregosa et al, 2011]: Esta biblioteca proporciona múltiples herramientas para el aprendizaje automático. Se utiliza fundamentalmente en la parte de reordenación para implementar las funciones de aprendizaje.

4. **Scipy** [Jones et al, 2001–]: Se trata en realidad de un «ecosistema» en la que se incluyen múltiples bibliotecas científicas, matemáticas y de ingeniería. Se usará entre otras cosas para la optimización de funciones.

2.4. PyCharm

PyCharm [JetBrains, 2015] es un editor de código muy completo. Éste proporciona soporte para Python y simplifica el desarrollo pues facilita herramientas como el auto-completado de código, marca los errores y permite corregirlos directamente con sugerencias, casi siempre las acertadas.

Dispone a su vez de herramientas muy útiles. Las que más se han utilizado en el desarrollo de este proyecto se mencionan a continuación:

- Soporte para control de versiones: Hoy en día es inconcebible realizar un proyecto (incluso aquellos de pequeña envergadura) que no esté sometido a un control de versiones. Se ha decidido usar Git por su gran potencia y popularidad (Explicado en la sección 2.6). La integración es magnífica y desde los menús se realizarán las acciones más habituales sin necesidad de acudir a un terminal para ejecutar los comandos. Permite navegar a través de la historia incluso de una selección de un fichero, entre otras muchas cosas. Además proporciona soporte nativo para GitHub, el repositorio remoto utilizado en este proyecto. En la figura 2.3 se puede ver la historia de un fichero concreto (con sus diferencias con la versión actual señaladas).

- Potente e intuitivo depurador: El depurador es muy fácil de usar y muy potente. Posee las características usuales de estas herramientas, así como un visualizador del estado de todas las variables. Se pueden cambiar contextos, evaluar expresiones, modificar los valores directamente, etc...

- Autocompletado de código y detección de errores de forma inmediata: *PyCharm* facilita mucho escribir código, pues ayuda en el autocompletado en todo momento, no sólo de funciones o características propias del lenguaje, sino de nuestras propias funciones, ofreciendo por ejemplo sugerencias en base a los parámetros de éstas. También hace comprobaciones de tipos, por lo que detecta cuando se usan mal las variables. Esto es muy útil y ahorra muchos quebradores de cabeza y horas en el depurador para detectar estos errores.

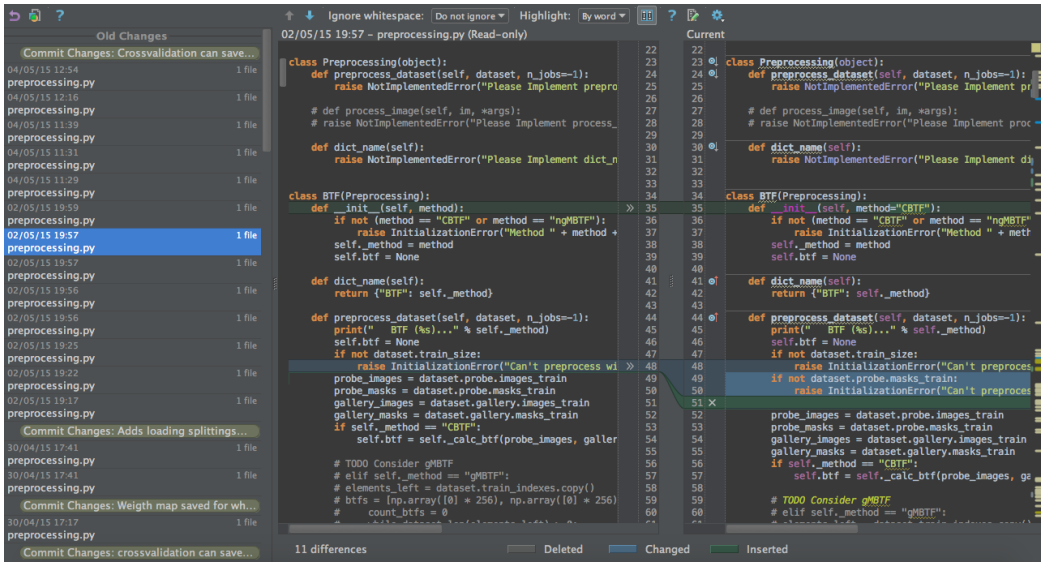


Figura 2.3: Historia de un documento en PyCharm

2.5. Ipython Notebook

Mención especial al utilísimo iPython Notebook [Pérez and Granger, 2007]. Se trata de un entorno interactivo en el que se puede ir creando código y viendo sus resultados sobre la marcha. Se puede dividir en celdas de código, ejecutando cada celda de forma independiente. Es una herramienta que facilita mucho la exploración y el análisis.

Se suele usar para reproducir experimentos e investigaciones. Se pueden generar gráficas y tablas y son mostradas directamente. Además se pueden añadir otro tipo de celdas, no sólo de código. Por ejemplo, se puede añadir una celda en la que se comente lo que se va a hacer en la siguiente celda de código, o comentar un resultado de una gráfica previa. Se puede además generar documentos LaTeX, HTML, etc. a partir de estos *Notebooks*.

En el caso de este proyecto se usa en la parte de pruebas y resultados. Con estos documentos se generan muchas de las tablas y gráficas que son mostrados en la sección 5. Se puede ver el ejemplo de un sencillo Notebook de los usados en este proyecto en la figura 2.4.

In [44]:
```
import pandas as pd
```

In [45]:
```
#df = pd.read_excel("/Users/luigolas/PycharmProjects/PyReid/results/Masksfromfile.xls")
#df = pd.read_excel("/Users/luigolas/PycharmProjects/PyReid/results/MasksHalfDs.xls")
df = pd.read_excel("/Users/luigolas/PycharmProjects/PyReid/results/from DO/MasksFullDs.xls")
```

In [47]:
```
#df_filter = df[df["Preproc1"].isnull()]  # remove those with regions
df_filter = df  # do not remove those with regions

df_list = []
df_list.append(df_filter[df_filter["Preproc0"].isnull()])  #None
df_list.append(df_filter[df_filter["Preproc0"] == "MasksFromMat"])  #aleMask
df_list.append(df_filter[df_filter["Preproc0Params"] == "[2, 'ViperManualMask']"])  #ManualMask
df_list.append(df_filter[df_filter["Preproc0Params"] == "[2, 'ViperOptimalMask']"])  #OptimalMask
df_list.append(df_filter[df_filter["Preproc0Params"] == "[2, 'ViperOptimalMask-4']"])  #OptimalMask-4

table = pd.DataFrame(index=["RGB", "HSV", "IIP", "Total"], columns=["Sin Seg","aleMask", "ManualMask", "OptimalMask", "
#table
```

In [48]:
```
bgr_bins = ["[16, 16, 16]", "[32, 32, 32]"]
for i, df in enumerate(df_list):
    table.ix[0,i] = df[(df["FeColorSpace"] == "BGR") & (df["FeBins"].isin(bgr_bins))]["AUC"].mean()
    table.ix[1,i] = df[df["FeColorSpace"] == "HSV"]["AUC"].mean()
    table.ix[2,i] = df[df["FeColorSpace"] == "IIP"]["AUC"].mean()
    table.ix[3,i] = table.ix[0:-1, i].mean()
table
```

Out[48]:

	Sin Seg	aleMask	ManualMask	OptimalMask	OptimalMask-4
RGB	58.8055	58.8215	58.2465	60.07	59.5725
HSV	77.74611	76.79983	76.27461	77.87667	77.99806
IIP	72.0665	72.9245	71.55425	73.57	73.50313
Total	69.53937	69.51528	68.69179	70.50556	70.35789

In [50]:
```
table.to_excel(excel_writer="/Users/luigolas/Desktop/temp.xls", float_format='%.2f')
```

Figura 2.4: Ejemplo de iPython Notebook

2.6. Control de versiones Git y Github

Cuando se trabaja en proyectos de cierta envergadura, e incluso cuando es con proyectos simples, un sistema de control de versiones puede facilitar el trabajo enormemente. Un control de versiones permite que se pueda ir guardando el progreso de forma gradual, permitiendo volver a cualquier punto anterior. Por supuesto también sirve como copia de seguridad ante posibles pérdidas. Para el trabajo en equipo es indispensable hoy en día, aunque no es el caso de este proyecto.

Git [Torvalds, 2005] es uno de los sistemas de control de versiones más avanzados. Es gratuito y de código abierto y está disponible para todas las plataformas. Está diseñado para manejar de forma distribuida proyectos de todo tipo. En este proyecto se ha utilizado para:

- Controlar el progreso de los distintos ficheros: Git permite ir haciendo copias cuando el desarrollador lo desea. Una buena práctica extendida es hacer una copia (denominada *commit* en Git) cada vez que se realiza

una funcionalidad. De esta forma se puede volver a una versión anterior del código si se quiere eliminar una característica posteriormente, o simplemente visualizar las diferencias entre la versión pasada y la actual.

- Copia de Seguridad: Sin duda, hoy en día los documentos importantes se han de tener en varios sitios para evitar su pérdida. Sin ir más lejos en un momento dado del proyecto el PC principal con el que se desarrollaba ha fallado. Esto no ha supuesto un gran problema porque todo el código estaba bajo control de versión en un repositorio remoto.

Aunque Git puede ser utilizado perfectamente sin necesidad de utilizar repositorios remotos, esto es, guardar una copia en un servidor externo, se perdería una de sus características más útiles que es mantener una copia del repositorio local en un lugar externo. Esto es principalmente para mantener una copia de seguridad. Sin embargo también puede ser utilizado para dar visibilidad al proyecto y que otras personas puedan ver el código. Si se decide mostrarlo como público, cualquier persona puede ver el código.

En nuestro caso se ha hecho uso de GitHub [GitHub, 2015], una de las plataformas más completas y populares para esta tarea. GitHub ofrece gratuitamente la creación de repositorios públicos. Existe la opción de tener repositorios privados también, aunque esta opción es de pago. Sin embargo al ser estudiantes se puede optar gratuitamente por esta opción también. Inicialmente se mantiene el proyecto en un repositorio privado. Sin embargo en cierto punto se decide liberarlo y mostrarlo al público. Está disponible en `https://github.com/Luigolas/PyReid`.

2.7. LyX

Para la redacción de esta memoria se ha recurrido al editor de texto LyX [The LyX Team, 2009]. Como se describe en su página web, LyX es un procesador de documentos que se enfoca en la estructura de los documentos y no en su apariencia.

Es común en muchos proyectos de fin de carrera ese momento en el que se tiene que decidir qué editor de texto usar. La opción más común por su sencillez suele ser un editor como el Microsoft Office Word o LibreOffice.

Este tipo de editores se enfocan en la apariencia (*What You See Is What You Get*). La otra opción común es TEX/LATEX . El enfoque de LATEX es radicalmente opuesto, pues funciona como un lenguaje de markup (como HTML, por ejemplo) y generalmente ofrece resultados muy profesionales. El principal problema de LATEX es que requiere de un período de aprendizaje y en un principio suele ser algo tedioso.

Es por lo anterior que se se considera lo siguiente: *¿Y si existiera un editor que uniera lo mejor de ambos mundos?* La facilidad de uso de Word, pero con la potencia de LATEX, que requiera poco tiempo de adaptación. La búsqueda tiene un pronto resultado: *LyX*.

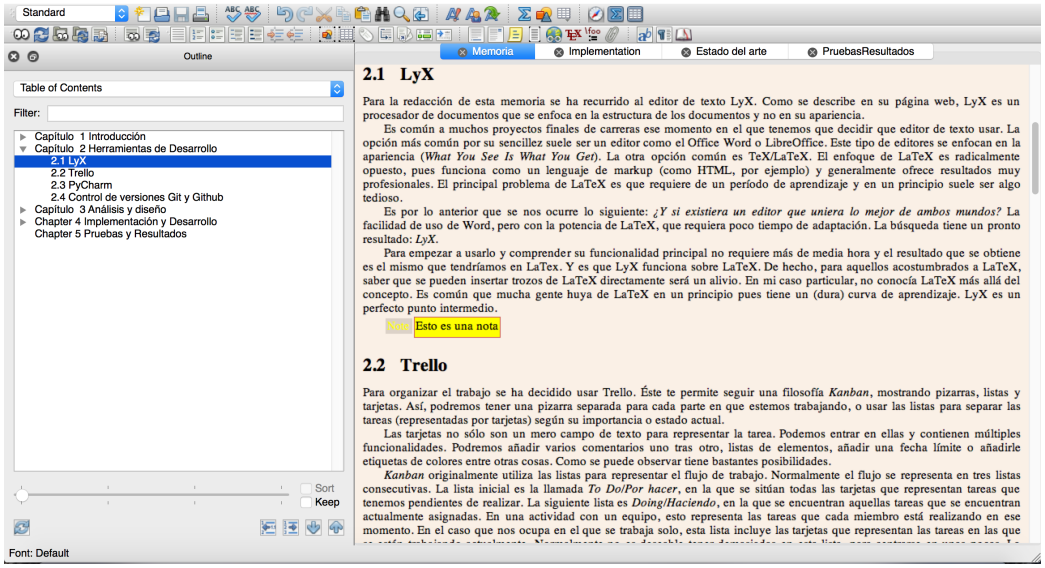


Figura 2.5: Interfaz de LyX

Para empezar a usarlo y comprender su funcionalidad principal no requiere más de media hora y el resultado que se obtiene es similar al obtenido en LATEX. Y es que LyX funciona sobre LATEX. De hecho, para aquellos acostumbrados a LATEX, saber que se pueden insertar trozos de LATEX directamente será un alivio. En mi caso particular, no conocía LATEX más allá del concepto. Es común que mucha gente huya de LATEX en un principio pues tiene una (dura) curva de aprendizaje. LyX es un perfecto punto intermedio.

LyX está disponible en múltiples plataformas, incluyendo las más comunes Windows, Mac OS X y Linux. Se puede ver su interfaz en la figura 2.5.

2.8. Trello

Para organizar el trabajo se ha decidido usar *Trello* [Trello Inc., 2015]. Éste te permite seguir una filosofía *Kanban*, mostrando pizarras, listas y tarjetas. Así, se puede tener una pizarra separada para cada parte en la que se esté trabajando, o usar las listas para separar las tareas (representadas por tarjetas) según su importancia o estado actual.

Las tarjetas no sólo son un mero campo de texto para representar la tarea. Se puede entrar en ellas y contienen múltiples funcionalidades. Se pueden añadir varios comentarios uno tras otro, listas de elementos, añadir una fecha límite o añadirle etiquetas de colores entre otras cosas. Como se puede observar ofrece bastantes posibilidades.

Kanban originalmente utiliza las listas para representar el flujo de trabajo. Normalmente el flujo se representa en tres listas consecutivas. La lista inicial es la llamada *To Do/Por hacer*, en la que se sitúan todas las tarjetas que representan tareas pendientes de realizar. La siguiente lista es *Doing/Haciendo*, en la que se encuentran aquellas tareas que se encuentran actualmente asignadas. En una actividad con un equipo, esto representa las tareas que cada miembro está realizando en ese momento. En este caso, en el que se trabaja solo, esta lista incluye las tarjetas que representan las tareas en las que se están trabajando actualmente. Normalmente no es deseable tener demasiadas en esta lista, para centrarse en unas pocas. La siguiente lista es la de tareas realizadas (*Done*). En esta lista van las tareas que ya se han completado. Especialmente útil para repasar todo aquello que ya se ha realizado y no perderlo de vista.

Todas estas listas representan un flujo, en el que una tarjeta fluye desde la lista inicial de tareas por hacer, hacia la lista de *Doing*, hasta la lista de tareas realizadas una vez se ha completado. En el caso de este proyecto, se usan dos pizarras, una para la creación de la memoria de trabajo, en donde se irán añadiendo las tarjetas relativas a la redacción de ésta, y otra pizarra para el desarrollo de la investigación y de la aplicación. Sin embargo a la pizarra dedicada al desarrollo se le añaden unas cuantas listas extras para representar mejor la forma de trabajar que se ha seguido. El esquema es el siguiente:

Raw Ideas -> To Do -> Doing -> Today -> From Last Meeting -> Done

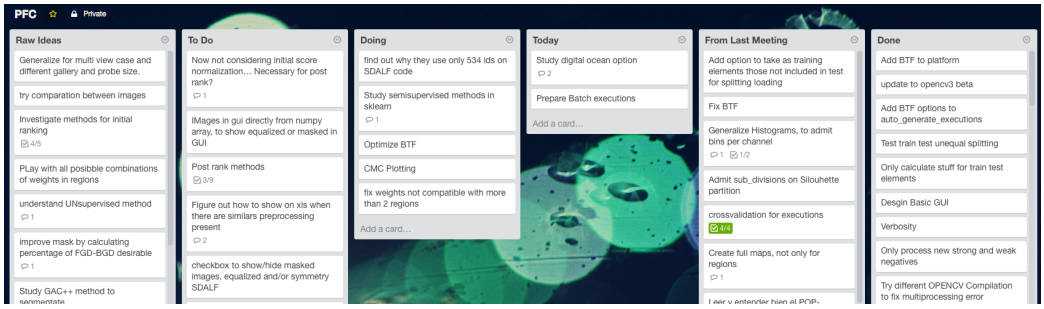


Figura 2.6: Pizarra PFC en Trello

La primera lista de la que se dispone es *Raw Ideas/Ideas en bruto* en la que se añaden todas esas ideas o funcionalidades que pueden ser interesantes. Esta lista va en primer lugar porque se apunta todo tipo de cosas, se piense o no implementarlas. Esa es una decisión que se toma a posteriori según va avanzando el proyecto. Cuando una tarjeta de *Raw Ideas* decide implementarse se pasa a la lista *To Do/Por Hacer*. En esta lista sólo se tienen aquellas tarjetas en las que se tiene planeado trabajar. Si bien alguna vez estas tarjetas pueden viajar de vuelta a *Raw Ideas* si por alguna razón se cambia de idea y no se tiene claro que se vaya a trabajar en esa idea o funcionalidad. Después de esta lista se encuentra la lista *Doing*. En ella se tienen unas 4 ó 5 tarjetas. Esta lista no quiere decir que se esté justo ahora trabajando en ellas, pero sí que están planificadas para realizarse muy pronto. Se añade después de ésta la lista *Today*. Cada día antes de empezar a trabajar se decide a qué se le va a dedicar el tiempo, se escoge una serie de tareas de la lista *Doing* y se traspasa s a la lista *Today*. Normalmente sólo hay 2 ó 3 tarjetas en esta lista.

Cuando las tareas son realizadas, pasan a la siguiente lista que se ha denominado *From Last Meeting/Desde la última reunión*. Esta lista se añade de forma intermedia a la lista *Done* y tiene la misma funcionalidad. Sin embargo como su nombre indica permite controlar qué cosas se han realizado desde la última reunión con los tutores. Así antes de cada reunión se observa cuál ha sido el progreso y es más fácil comunicarlo. Finalmente, una vez han sido comentados en una reunión con los tutores los avances, las tarjetas transitan a la lista *Done.*

Se puede ver en la figura 2.6 como es la interfaz, mostrando el estado en un momento dado de la pizarra dedicada al desarrollo.

2.9. Toggl

Toggl [Toggl OÜ, 2015] es una herramienta que permite un sencillo control del tiempo. En esta herramienta se pueden crear nuestros proyectos y cuando se inice una actividad se puede asignar a alguno de estos. No se comporta como un ToDo. Sólo se puede añadir una actividad en el momento que se empiece a realizarla. Así, es exclusivamente un *Time Tracker*, centrado únicamente en controlar el tiempo. Con estos datos, luego se pueden ver informes para visualizar el tiempo que se ha dedicado a un proyecto, o a cierta tarea.

En este proyecto se ha utilizado precisamente para calcular el tiempo real que se ha utilizado para las diversas tareas del proyecto. Se usa de forma bastante estricta, esto es, sólo se controla el tiempo que se pasa trabajando de forma real, eliminando del temporizador las pequeñas pausas y en general estimando a la baja. Así, por ejemplo, en un día en el que se trabajan 8 horas, en Toggl se puede ver como apenas se llegan a las 6 horas. Con la herramienta se puede visualizar por ejemplo en qué semanas o meses se ha trabajado más, o qué días de la semana son los más que se trabaja. Así se puede intentar detectar las razones y mejorar en la productividad.

2.10. Entorno de desarrollo y pruebas

A lo largo de las distintas etapas por las que ha pasado el desarrollo se han utilizado distintos entornos. El proyecto se inicia haciendo uso de un ordenador personal portátil. Éste funciona con Ubuntu (en distintas versiones que se han ido actualizando). Sus características técnicas son las siguientes:

- *Procesador*: Intel Core i3 330M (2 núcleos a 2.13 GHz)
- *Memoria*: 4 GB de RAM
- *Tarjeta Gráfica*: Nvidia GForce GT330M
- *Sistema Operativo*: Ubuntu

Sin embargo a lo largo del desarrollo se produce un cambio en el ordenador personal. Se pasa a usar un MacBook Pro 13' (Late 2013). Sus especificaciones son:

- *Procesador*: Intel Core i5 (2 núcleos a 2,4 GHz)
- *Memoria*: 8GB de RAM
- *Tarjeta Gráfica integrada*: Iris Graphics de Intel
- *Sistema Operativo*: OS X Yosemite

El principal cambio en este caso es en el sistema operativo. Sin embargo al ser Ubuntu y OS X, ambos sistemas basados en UNIX, el cambio no supone ningún trauma y se produce una transición sencilla.

Además de estos dos equipos para el desarrollo, para la realización de pruebas se ha hecho uso del equipo disponible en el laboratorio del grupo de investigación. Sus características son las siguientes:

- *Procesador*: Intel Core i7
- *Memoria*: 24 GB de RAM
- *Tarjeta Gráfica:* Nvidia GForce
- *Sistema Operativo*: Ubuntu

Así mismo se ha hecho uso de un servicio en la nube que permite tener nuestra propia máquina virtual, configurando la potencia según necesitemos. Este servicio es *DigitalOcean*. La principal ventaja es que se puede acceder de forma remota en cualquier momento. Por contra para acceder al PC del laboratorio hay que usarlo fisicamente. Además se aprovecha la oferta que hace *Github* en su plan para estudiantes. Éste ofrece un pack con acceso a muchos servicios. Uno de ellos es 100$ de crédito para usar en *DigitalOcean*. Así pues hacer uso de esta plataforma no ha supuesto ningún desembolso.

Como se ha dicho, se puede configurar una máquina según nuestras necesidades. Incluso se pueden variar estas características en cualquier momento. Se pueden ver las opciones que ofrece en la figura 2.7. Para acceder a la máquina se usa SSH. Para acceder al sistema de archivos, se puede hacer uso de FTP.

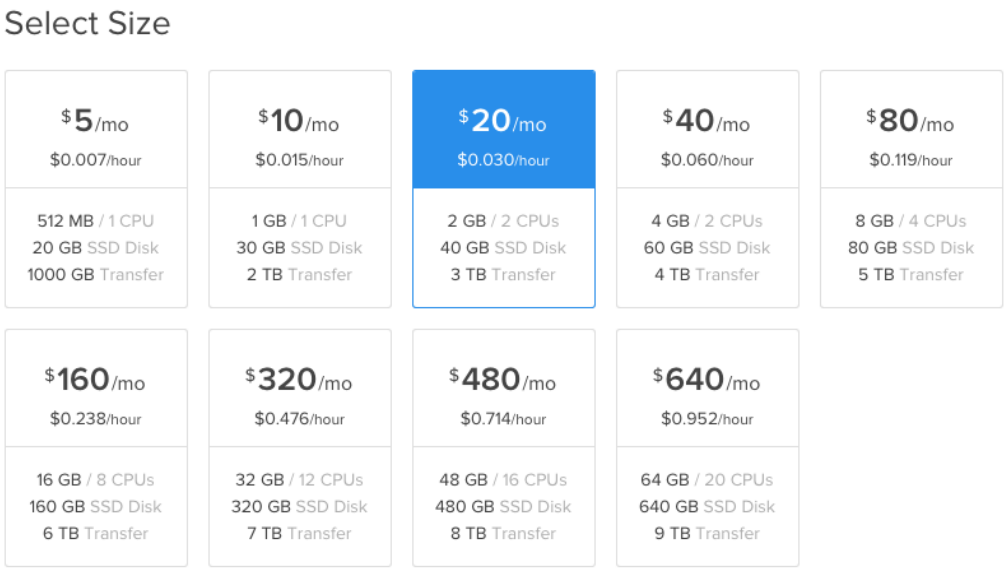


Figura 2.7: Opciones en DigitalOcean

Capítulo 3

Metodología y planificación

En este capitulo se describirá cómo se ha abordado el proyecto desde el punto de vista de la metodología que se ha seguido. Así mismo, se comentará la planificación realizada inicialmente y hasta que punto se ha cumplido ésta.

3.1. Metodología

En este proyecto se ha seguido una metodología evolutiva a grandes rasgos. Así se han producido 3 grandes fases o refactorizaciones. Por ello se podrían dividir las etapas de desarrollo en 3 principales, en las que además se han aplicado distintas técnicas. Esto ha aportado una experiencia muy interesante que cabe la pena analizar.

3.1.1. Cowboy Coding

La primera parte ha sido la menos organizada de todas. En esta fase no se pone el enfoque en que el código sea bonito, sencillo o legible y organizado. En esta fase aun se profundizaba en el conocimiento de Python, sus librerías, el uso de OpenCV y el conocimiento de la propia materia en sí. Este tipo de programación, la más común cuando no se conoce ninguna forma organizada para trabajar en proyectos, es popularmente conocida como *Cowboy Coding.*

Como «metodología» desde luego no es la más adecuada. Este tipo de práctica suele conllevar un código propenso a tener errores y desorganizado. Para

añadir nueva funcionalidad se encuentran muchos problemas. En un inicio, con un conocimiento limitado de Python pero con la necesidad de empezar a obtener resultados pronto, se opta por trabajar de esta forma. En un sólo fichero se tiene todo el código. Este único fichero, sin ninguna orientación a objetos, llega a tener unas 1000 líneas de código. Llegados a este punto, ya con un conocimiento mayor del problema, de las librerías y de Python, se considera pertinente pasar a una siguiente fase.

Aunque es una metodología caótica, y aunque no se tienen datos empíricos para corroborarlo, este tipo de programación sin ningún tipo de orden, es bastante común. Ciertamente tiene su utilidad. Permite arrancar rápido, obtener resultados pronto y que se aprenda sobre un lenguaje y una librería concreta. Es bastante difícil realizar a priori un buen diseño cuando la materia no se controla del todo y las librerías son desconocidas.

Sin embargo queda claro que este tipo de programación no es aconsejable. Es propenso a tener fallos. El diseño se suele obviar totalmente, lo que lleva a problemas posteriormente. Además cambiar una parte del código puede producir errores en otra parte debido al descontrol. La legibilidad y el código limpio suelen verse afectados.

En la siguiente fase se produce la primera refactorización y se aplican técnicas que permiten mantener una mejor estructura.

3.1.2. Test Driven Development

En esta fase lo primero que se realiza es estudiar el lenguaje Python más en profundidad y adentrarnos en la materia de la reidentificación de personas. Con estos conocimientos se puede realizar un diseño mucho mas estructurado. Ver el capítulo 4 para más información.

Como novedad principal se decide aplicar los principios básicos de TDD (Test Driven Development). Este tipo de desarrollo se ha vuelto muy popular y se basa en la repetición de un ciclo, conocido como *Rojo-Verde-Refactorizar.* Inicialmente se decide qué funcionalidad se va a añadir y se crea un test para esa funcionalidad. Se ejecuta este test y debe fallar (*Rojo*) ya que testea una funcionalidad no existente aun. A continuación, se escribe el código mínimo necesario para que se pase el test. Por supuesto el resto de tests ya escritos deben seguir siendo pasados. Esta es la fase *Verde.* A continuación se pone el foco en hacer que el código que se ha escrito sea lo más limpio posible,

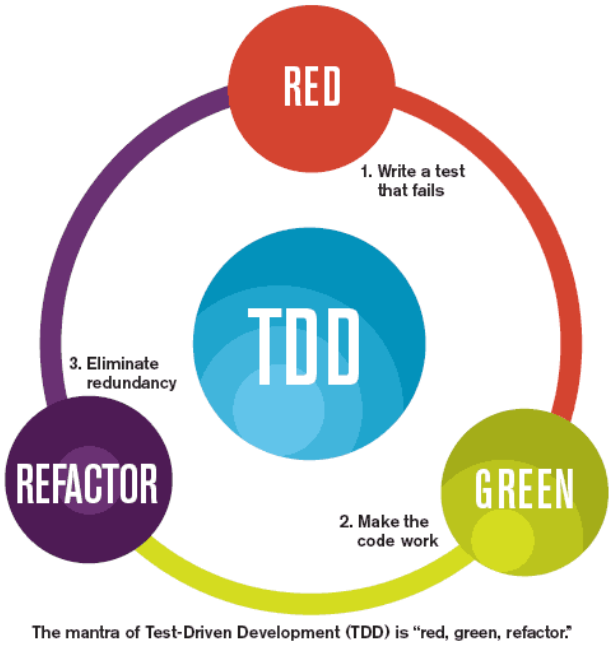


Figura 3.1: Diagrama de TDD [Nintex Labs, 2015]

eliminar duplicidades, renombrar clases y métodos, etc. Esta es la fase de Refactorizar. Se resume en la figura 3.1

Cabe destacar que TDD es casi un arte y dominarlo requiere tiempo y un largo aprendizaje. Además mi experiencia personal con la realización de tests es limitada. Sin embargo se intenta aplicar de la forma más fiel posible, aunque sin llegar a ciertos extremos. Por ejemplo, uno de los principios de TDD es que se debe realizar el código mínimo imprescindible para que pase los test, nada más. La idea es que de esta forma se realizará el código lo más sencillo posible siempre. Si se supone por ejemplo una función que devuelve un valor de tipo *boolean*, el primer test podría ser comprobar un caso en el que debe devolver *True.* En ese caso se podría simplemente escribir en la función *return True*. No sería hasta que se realice el segundo test, para comprobar un caso en el que se ha de devolver *False,* que se habrá de realizar el código «de verdad». En casos así en general he optado por ir directamente a escribir el código que consideraba era necesario.

Sin duda TDD aporta una serie de beneficios. Principalmente te hace pensar en todo el diseño, en cómo se va a usar una funcionalidad antes de ponerte

a implementarla. La fase de refactorización además permite percatarse de partes del código que claramente pueden ser mejorables también y que de otra forma no detectarías. Además no hace falta casi usar el debugger, pues adquieres una conciencia del código mayor.

Sin embargo recientemente se ha puesto en tela de juicio su utilidad. David Heinemeier Hansson [1], conocido comúnmente como DHH en la comunidad, autor de Ruby on Rails y fundador de Basecamp, recientemente escribía un post llamado "TDD is Dead – Long Live Testing"[2] en el que criticaba varios aspectos de TDD. Entre ellos critica el excesivo foco en test unitarios o el intentar testear el 100 % del código. No se va a entrar en detalles, pero dada la influencia de DHH la comunidad se encuentra debatiendo activamente estos puntos.

En mi opinión la experiencia de utilizar TDD ha sido muy enriquecedora. Creo que gracias a TDD el diseño general ha mejorado muchísimo. He aprendido a realizar tests y le he encontrado una gran utilidad. Es sorprendente como he podido darme cuenta de situaciones que de otra forma nunca descubriría. A veces se cambia algo en el código, para pasar un nuevo test y se piensa que todo va a estar correcto. Este código pasa el nuevo test, pero ocurre que falla otro test anterior. Al revisarlo, descubres que habías introducido un error sin darte cuenta. Sin haber realizado test, esta situación hubiera sido mucho más difícil de detectar.

También cabe destacar que la realidad es que muchas veces escribes esbozos de código para probar como implementar la funcionalidad. Luego, creas los test, y después empiezas a implementar el código real. Muchas veces ésta ha sido la situación. El punto negativo de TDD es el tiempo que requiere. Es de suponer que una persona experimentada tardará mucho menos, pero se ha podido apreciar que en esta fase el desarrollo se ralentiza bastante.

Es por ello, ante la falta de tiempo, que para la siguiente gran refactorización se decide abandonar TDD, a pesar de sus ventajas, para poder sacar a tiempo todas las funcionalidades planificadas.

[1] http://david.heinemeierhansson.com/

[2] http://david.heinemeierhansson.com/2014/tdd-is-dead-long-live-testing.html

3.1.3. Cowboy Coding después de TDD

En esta fase se parte del modelo anterior y se realizan una serie de cambios principalmente basados en la experiencia adquirida tanto en Python como en la materia de la Reidentificación. Se busca generalizar más el diseño. Así es como se llegará al diseño descrito en la sección 4.

Dada la falta de tiempo y el interés por sacar nuevas funcionalidades, se dejan de realizar tests y se pierde compatibilidad con la gran mayoría de éstos realizados previamente. Queda como tarea pendiente volver a realizar un conjunto mínimo de tests (apartado 6.2.2).

3.2. Planificación y temporización

Para el Proyecto Final de Carrera se estima una duración de entre 800 a 1000 horas de dedicación. Se parte de una planificación inicial que se intenta cumplir, aunque el desconocimiento inicial del tema y los cambios que se han ido dando a lo largo del desarrollo respecto a la idea original hacen que las diferencias entre el plan inicial y la ejecución final sean inevitables. En la figura 3.2 se puede ver lo originalmente planificado en porcentaje del tiempo total.

Para calcular el tiempo que se ha dedicado realmente se ha hecho uso de la herramienta Toggl, apartado 2.9. Sin embargo los datos no se pueden considerarlos al 100 % exactos. En primer lugar porque inicialmente no se era preciso al definir las tareas y existe cierta ambigüedad entre tareas no quedando claro la categoría a la que pertenecen. Así mismo hay tareas que pueden entrar en varias categorías a la vez.

En la figura 3.3 se muestran los resultados, tomados desde Toggl, procesando las entradas para agruparlas en las categorías mostradas. Aparentemente según los datos se cumplen los porcentajes, excepto el hecho de que parece que se ha dedicado menos tiempo a la investigación y más a la implementación y análisis. La tarea de investigación es sin duda la peor que se ha controlado, pues en muchas ocasiones esta tarea no era temporizada correctamente, así que cabe esperar que el porcentaje de dedicación real sería mayor.

Llevar el control del tiempo ha permitido acumular mucha experiencia en el correcto uso de las herramientas de control del tiempo y productividad y en

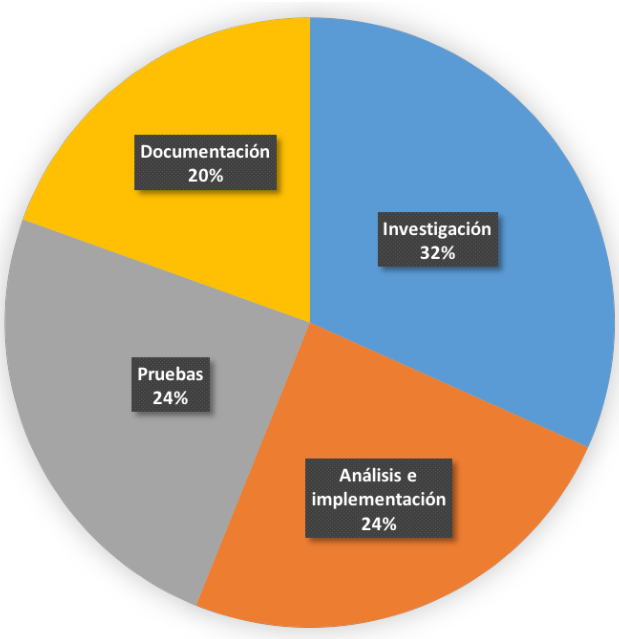


Figura 3.2: Planificación inicial.

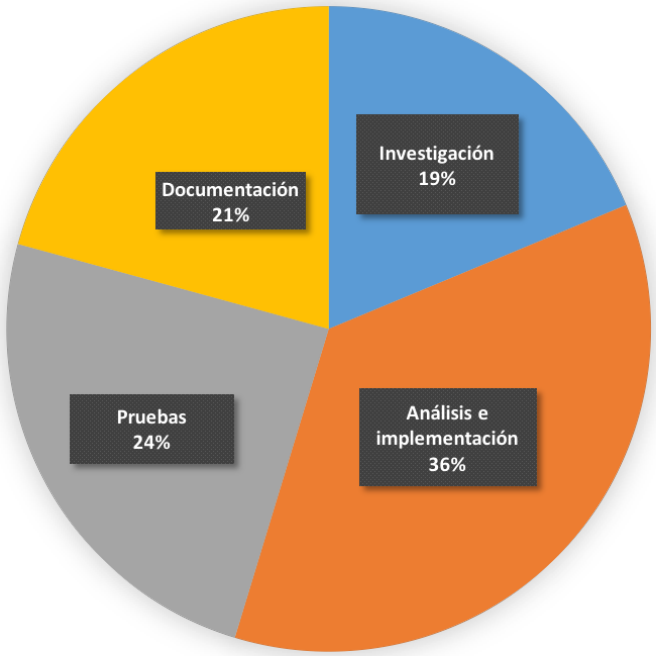


Figura 3.3: Temporización final.

cómo usarlas de la forma más útil. Se ha visto cómo ha supuesto un problema el no usar una nomenclatura concreta a la hora de definir las tareas. Se han tenido que ir renombrando manualmente para poder procesar el tiempo automáticamente a posteriori. También se observa que a veces es difícil definir una tarea dentro de un grupo concreto únicamente. Sin embargo, usar una nomenclatura «estándar» hubiera facilitado mucho el trabajo.

Capítulo 4

Implementación y diseño

A lo largo del proyecto se han ido estructurando los métodos y funcionalidades en torno a un framework que facilite el uso y la adición de nuevas características. En esta sección se explicará la estructura general de PyReID y luego se especificarán detalladamente cada uno de los componentes.

Cabe recalcar que PyReID parte de la base de que el conjunto de datos es proporcionado en forma de imágenes ya etiquetadas. PyReID se centra en realizar pruebas, no es un sistema para ser usado en un entorno de producción. Permite probar diferentes métodos y combinarlos entre sí. Se centra en la reidentificación, por lo que no es capaz de realizar detección o seguimiento de personas (ver figura 1.2). Este framework se enfoca en las fases posteriores del proceso, como se verá en la siguiente sección.

PyReID se divide en distintas partes, cada una de ellas especializada en una de las fases del proceso de reidentificación. El proceso principal se engloba en la clase *Execution* (ejecución), que se encarga de gestionar una ejecución, en la que se tendrá como entrada un objeto *Dataset* (conjunto de datos) y devolverá una matriz denominada *Ranking Matrix* (matriz de ordenación), una matriz con los elementos de la galería ordenados para cada uno de los elementos de la muestra en base a su similitud. Normalmente no sólo se querrá esta matriz, sino que se requiere obtener algunas estadísticas. Para ello existe una clase que facilita el cálculo de estas estadísticas de interés (Sección 4.7).

También se cuenta con la clase *CrossValidation* (validación cruzada), que realiza una validación cruzada (explicada en la sección 4.8). Además de la

validación cruzada, esta clase permite entre otras cosas guardar las estadísticas que se obtengan o guardar los datos en formato hoja de cálculo (excel) para su posterior visualización.

Por supuesto también se cuenta con una clase para realizar la reordenación, a la que se facilitará una matriz de ordenación y el conjunto de datos para que pueda realizar el proceso. Todo ello se puede realizar con una simple interfaz gráfica realizada en Qt. En las siguientes secciones se entrará en detalle de cada una de las partes.

4.1. Ejecución

Ésta es una de las clases más importantes. Controla todo el proceso para realizar lo que se ha denominado la ordenación inicial. El proceso general puede verse en la figura 4.1

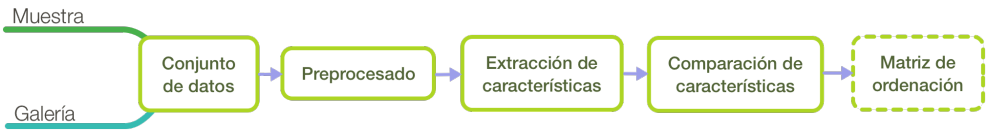


Figura 4.1: Proceso seguido en una ejecución.

La clase *Execution* se puede configurar añadiendo varios (o ningún) objeto *Preprocessing*, que aplican algún tipo de preprocesado como pueden ser la segmentación o normalización de la iluminación. También se ha de establecer un objeto de la clase *Feature_Extractor* (extracción de características) que será el encargado de realizar la extracción de características (Histogramas). Por último hará falta un objeto *Feature_Matching (comparación de características),* que trabajará con la salida del *Feature_Extractor* y se encarga de obtener la similitud entre cada elemento de la muestra con la galería.

En la figura 4.2 se puede ver el diagrama de la clase *Execution*. En ella se observa cómo existe un atributo para cada uno de los elementos mencionados hasta ahora. Cabe destacar los siguientes métodos:

- run(): Función principal en la que se lanzará a ejecutar una vez se haya configurado la clase. A su finalización devuelve la matriz de ordenación, de tipo *ndarray* que será usada por otras clases, como Statistics o PostRanker, explicadas en próximas secciones.

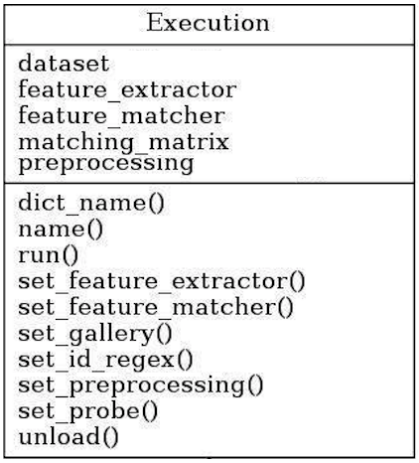


Figura 4.2: Diagrama de la clase *Execution*.

- dict_name(): Usado para devolver un identificador descriptivo, en forma de diccionario en Python. Su principal uso es en la clase *CrossValidation* (sección 4.8) para guardar la información en un fichero excel.

4.2. Imagen

Esta clase es la utilizada para representar una imagen. OpenCV utiliza la clase *ndarray* de *numpy,* librería muy utilizada en Python para cálculos matriciales. Sin embargo se desea añadir alguna funcionalidad extra a las imágenes. La principal es que se quiere añadir un control del espacio de color en que se encuentra una imagen y añadir la funcionalidad para el cambio entre espacios de color como un método de la clase. Además se añade un atributo para saber su nombre, que será la ruta de disco de donde ha sido cargada la imagen. Para mantener la compatibilidad lo que se hará será crear una nueva clase usando herencia de la clase *ndarray*. En la figura 4.3 se puede ver un diagrama resumen de la nueva funcionalidad de la clase.

Como se ha mencionado, una característica de las imágenes que es especialmente interesante en este proyecto es el espacio de color en el que se representan. Es fundamental saber en qué espacio de color se encuentra una imagen, pues como se verá a lo largo del proyecto, el principal método del que se hace uso consiste en comparar imágenes usando características obtenidas

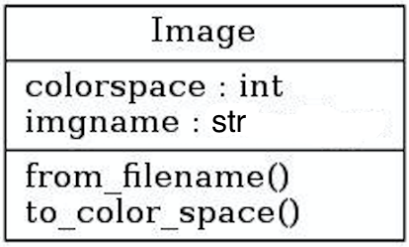


Figura 4.3: Diagrama de la clase *Image*.

desde el color de éstas. A continuación se explican los espacios de color que han sido utilizados en este proyecto.

4.2.1. Espacios de color

Para realizar la reidentificación habrá que comparar unas imágenes con otras. Para ello se tendrá que detectar características en las imágenes, algo que sea comparable y que caracterice a éstas. Una de las más obvias es sin duda el color, la principal característica que se explota en este trabajo y cuyo uso en reidentificación es muy común. Sin embargo el color puede ser representado de múltiples formas. A estas representaciones del color se les conoce como espacios de color.

Más formalmente, un modelo de color es un modelo matemático abstracto que permite representar con tuplas de números el color. Un modelo de color tiene asociada una función de mapeo a un espacio de color absoluto, un espacio de color donde las interpretaciones de los colores en el espacio están definidos colorimétricamente sin referencias a factores externos. Esto es, que un espacio de color ha de relacionarse de alguna forma con un espacio de color en el que la interpretación de los colores sea inequívoca. [1]

Cada espacio de color por tanto representa los colores de una forma diferente. Esto es muy interesante porque habrá que estudiar qué espacio de color es más adecuado para comparar el tipo de imágenes con las que se enfrentará el sistema. En [Marín Reyes, 2015] se puede ver un interesante estudio del comportamiento de los espacios de color en conjuntos de datos utilizados en

[1] Más información sobre los espacios de color absolutos: `http://es.wikipedia.org/wiki/Espacio_de_color_absoluto`

la reidentificación de personas. En este estudio se concluye que HSV es el espacio de color más indicado para la reidentificación. Éste es descrito en la sección 4.2.3 y es uno de los utilizados en este proyecto. En este proyecto se utilizarán algunos espacios de color comunes y ampliamente utilizados como son RGB y HSV, soportados de forma nativa por las bibliotecas de las que se hará uso. También se propondrá un espacio de color no usado previamente en la reidentificación pero del que se espera que por sus características sea de utilidad para el caso que nos ocupa.

4.2.2. Espacio de color RGB

El espacio de color RGB es uno de los más comunes para describir el color. Se trata de un espacio de color aditivo. Esto es así porque se define el color como una adición de los colores rojo, verde y azul (figura 4.4). Con esto se consigue representar una gran cantidad de colores. Su nombre viene de la primera letra de cada color en inglés (*Red, Green, Blue*).

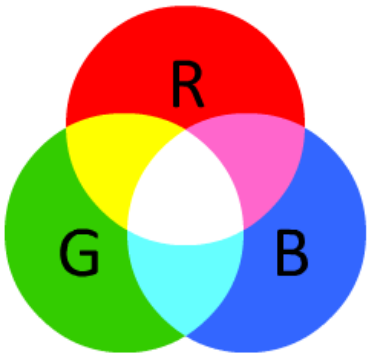


Figura 4.4: Espacio de color RGB [Kirupa.com, 2015].

El principal uso de este espacio de color es para dispositivos electrónicos como pantallas, aunque se usa también para fotografía y otros ámbitos. Este espacio de color ya tenía una buena base teórica antes de la era electrónica basada en la percepción de los colores.

4.2.3. Espacio de color HSV

Este espacio de color es también muy comúnmente usado. Es un espacio de color que usa coordinadas cilíndricas y supone una representación del espacio RGB buscando ser más intuitivo y fácil de percibir. HSV significa *Hue, Saturation, Value* (Matiz, Saturación, Valor).

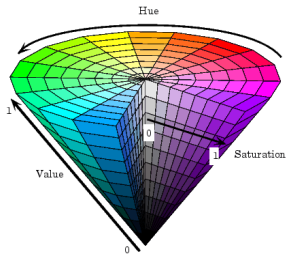


Figura 4.5: Espacio de color HSV [MIT, 2015].

Este espacio de color representa los colores en un cilindro, ver figura 4.5. En este cilindro el ángulo sobre el eje vertical central representa el matiz, la distancia con respecto al eje representa la saturación y la distancia a lo largo del eje el valor o brillo.

4.2.4. Espacio de color IIP

Además de los mencionados espacios de color RGB y HSV se decide probar con este espacio de color propuesto en [Chong et al, 2008]. Este espacio de color está basado en la percepción para el proceso de imágenes sin variación de la iluminación, que se abreviará en IIP, del inglés *Illumination Invariant Perception*. Éste se basa en la percepción del ojo humano para definir una transformación en el espacio de color. Con esta transformación, se obtiene una descripción del color que es aproximadamente invariante a la iluminación.

En el trabajo en que se presenta este espacio se usa para realizar segmentación y para el rellenado de un hueco en una imagen usando otra imagen. En este proyecto resulta interesante por el hecho de ser un espacio de color que es invariante a la iluminación. A continuación se explica la transformación:

$$F(\vec{x}) = A(\hat{ln}(B\vec{x})) \tag{4.1}$$

Donde A y B:

$$A = \begin{bmatrix} 2.707439x10^{1} & -2.280783\times10^{1} & -1.806681 \\ -5.646736 & -7.722125 & 1.286503\times10^{1} \\ -4.163133 & -4.579428 & -4.576049 \end{bmatrix}$$

$$B = \begin{bmatrix} 9.465229\times10^{-1} & 2.946927\times10^{-1} & -1.313419\times10^{-1} \\ -1.179179\times10^{-1} & 9.929960\times10^{-1} & 7.371554\times10^{-3} \\ 9.230461\times10^{-2} & -4.645794\times10^{-2} & 9.946464\times10^{-1} \end{bmatrix}$$

La implementación consiste en realizar los siguientes pasos:

1. Transformar el espacio de color a XYZ.
2. Aplicar las fórmulas expuestas sobre cada píxel de la imagen en el espacio XYZ.

Lo primero es fácil de realizar con OpenCV. Éste ya incluye una función que permite pasar del espacio RGB a XYZ:

```
src_xyz = cv2.cvtColor(src, cv2.COLOR_BGR2XYZ)
```

Con lo obtenido, se puede aplicar la ecuación 4.1. Sin embargo hay que comentar una serie de problemas que se han encontrado a la hora de implementar esta ecuación. A la hora de calcular el logaritmo neperiano, se producen errores dado que hay valores iguales o inferiores a 0. Para estudiar en qué casos se produce se comprobarán los valores que se obtienen para IIP con diversos valores en RGB.

En la tabla 4.1 se puede apreciar el proceso de conversión para diversos píxeles. Cada fila representa un píxel ejemplo, inicialmente con los valores para R, G, B propios del espacio de color RGB y mostrando en cada matriz el cambio en cada fase del proceso de conversión entres espacios de color. Como se ve se encuentra un problema con los valores 0 (Cero) o negativos en el logaritmo neperiano. En este ejemplo se decide mostrar el valor para el logaritmo neperiano si se limita el valor mínimo de entrada a 0.00001, un valor próximo a cero.

$$
\overset{Pixels\,RGB}{\begin{bmatrix} 1 & 1 & 0 \\ 0 & 2 & 0 \\ 2 & 0 & 1 \\ 1 & 0 & 2 \\ 2 & 1 & 0 \\ 2 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}} \rightarrow \overset{Pixels\,XYZ}{\begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 2 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}} \rightarrow \overset{B*XYZ}{\begin{bmatrix} -0.13 & 0.01 & 0.99 \\ 0.29 & 0.99 & -0.05 \\ -0.13 & 0.01 & 0.99 \\ 0.95 & -0.12 & 0.09 \\ -0.26 & 0.01 & 1.99 \\ -0.13 & 001 & 0.99 \\ 0 & 0 & 0 \end{bmatrix}} \rightarrow
$$

$$
\rightarrow \overset{ln(B*XYZ)}{\begin{bmatrix} -23.03 & -4.91 & -0.01 \\ -1.22 & -0.01 & -23.03 \\ -23.03 & -4.91 & -0.01 \\ -0.05 & -23.03 & -2.38 \\ -23.03 & -4.22 & 0.69 \\ -23.03 & -4.91 & -0.01 \\ -23.03 & -23.03 & -23.03 \end{bmatrix}} \rightarrow \overset{A(ln(B*XYZ))}{\begin{bmatrix} -622.3 & 167.87 & 118.37 \\ 8.52 & -289.27 & 110.49 \\ -622.3 & -4.91 & -0.01 \\ 8.07 & 147.47 & 116.58 \\ -623.7 & 171.43 & 112.02 \\ -622.3 & 167.87 & 118.37 \\ -576.66 & 11.60 & 306.72 \end{bmatrix}}
$$

Tabla 4.1: Conversión RGB a IIP con valores próximos a 0.

Existe un problema en estos puntos, pues es difícil saber cual es la interpretación correcta. Así pues, se considera que los valores correctos han de ser aquellos que tengan coherencia con los valores próximos. Es decir, se intentará determinar el comportamiento de la función de transformación cuando se va aproximando a valores bajos de RGB. En la figura 4.6 se puede ver que si se mantienen las componentes al mismo valor ([0, 0, 0], [1, 1, 1], ...), hay un punto en el que repentinamente cambia el comportamiento de cada componente. Esto tiene coherencia con lo que observado en los datos mostrados anteriormente.

Al observarse que el problema surge en valores bajos, la solución que se estimó fue la de limitar los valores BGR antes de la conversión a un valor mínimo. Es un valor que puede ser modificado fácilmente, pero por defecto está fijado en 3 y se trabaja con este valor en todo el proyecto.

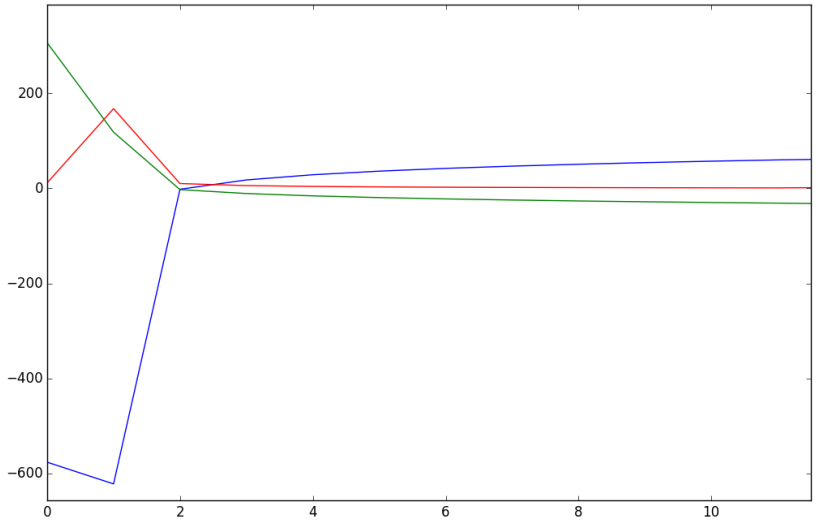


Figura 4.6: Comportamiento IIP según valores RGB. Todas las componentes con el mismo valor.

4.3. Conjunto de datos

Como su nombre indica, en esta clase se controla el conjunto de datos con el que se va a trabajar. La clase *Dataset* (conjunto de datos) mantendrá gran parte de la información necesaria para la ejecución. En ella se tendrán los atributos probe (muestra) y gallery (galería), de tipo ImageSet y que albergarán gran parte de la información. Las imágenes en el conjunto de datos se representan con la clase Image. El diagrama de la clase se puede ver en la figura 4.7.

Sus atributos son:

- probe y gallery: Atributos de tipo ImageSet, explicados a continuación. Son la muestra y la galería.
- id_regex: Expresión regular que define el patrón seguido para nombrar a un fichero. Tiene un valor por defecto, por lo que si se sigue el patrón de nombrado por defecto no habrá que modificar este atributo.

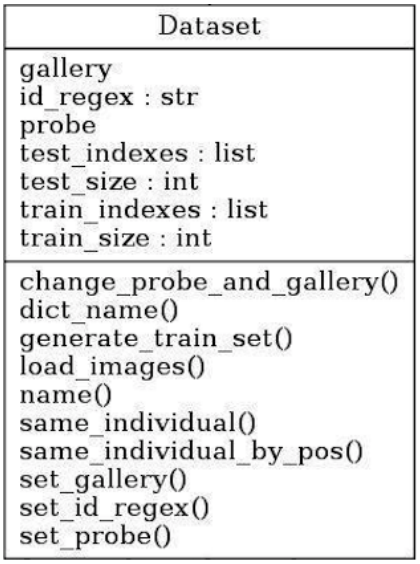


Figura 4.7: Diagrama de la clase *Dataset.*

Es utilizado por los métodos que usan el nombre de las imágenes para determinar si se trata del mismo individuo.

- test y train indexes y size: Almacena los índices (y el tamaño) sobre el conjunto total de los elementos que se encuentran en los conjuntos de entrenamiento o test, necesarios si hay algún método que requiera aprendizaje.

También posee una serie de métodos necesarios para manejar el conjunto de datos. Los más interesantes:

- generate_train_set(): Con este método se puede definir cuantos elementos se quieren en un conjunto de entrenamiento y cuantos en conjunto test. Se puede además definir una semilla (para que siempre salgan los mismos resultados) o hacerlo de forma completamente aleatoria.
- load_images(): Invoca al método *load_ images* de los atributos *probe* y *gallery*. Esto leerá las imágenes de disco.
- same_individual(): Método usado para comparar dos elementos, usando su nombre y el atributo id_regex para determinar si son el mismo individuo.

- dict_name(): Igual que en la clase Execution, sección 4.1.

A continuación se describe la clase ImageSet, clase que dentro de la clase Dataset se usa para representar a la muestra y la galería.

4.3.1. ImageSet

Esta clase es fundamental para este proyecto. Será usada para representar, dentro de la clase Dataset, a los conjuntos muestra y galería, cada uno siendo un ImageSet. Se encarga de mantener todas las imágenes así como de mucha información relacionada, como son las máscaras resultado de la segmentación o los mapas para usar con histogramas ponderados (que se describen más adelante). En la figura 4.8 se puede ver un esquema de la clase.

Figura 4.8: Diagrama de la clase *ImageSet*.

Lo primero que se observa es cómo se dividen los diferentes atributos en conjuntos test y train (entrenamiento). Esto se hace así para simplificar las operaciones en el caso de que sea necesario usar estos dos conjuntos. Si no se necesita el conjunto de entrenamiento, se usa el conjunto test por defecto. Se puede ver cómo se tienen atributos para *mask* (máscara), usados por la segmentación, *regions* (regiones), para la partición en regiones o *maps*

(mapas), para los mapas de pesos en los histogramas ponderados. Todo ello se describe en secciones posteriores.

En los atributos *images* se encuentran las imágenes cargadas. En *files* se guardarán los nombres de las imágenes con su ruta completa en disco. Los atributos *fe* representan los *Feature_Extracted,* las características extraídas de las imágenes. Estas serán rellenadas por el extractor de características *(Feature_Extractor*), explicado en siguientes secciones.

Cuando la clase se crea habrá que pasarle una ruta en disco donde se encuentran todas las imágenes. Al crease la clase, se leen los archivos y se guardan las rutas en los atributos *files* (ficheros). Respecto a los métodos de la clase sólo se tendrá *load_images()* que se encargará de ir leyendo las direcciones de los atributos *files* y de cargar las imágenes.

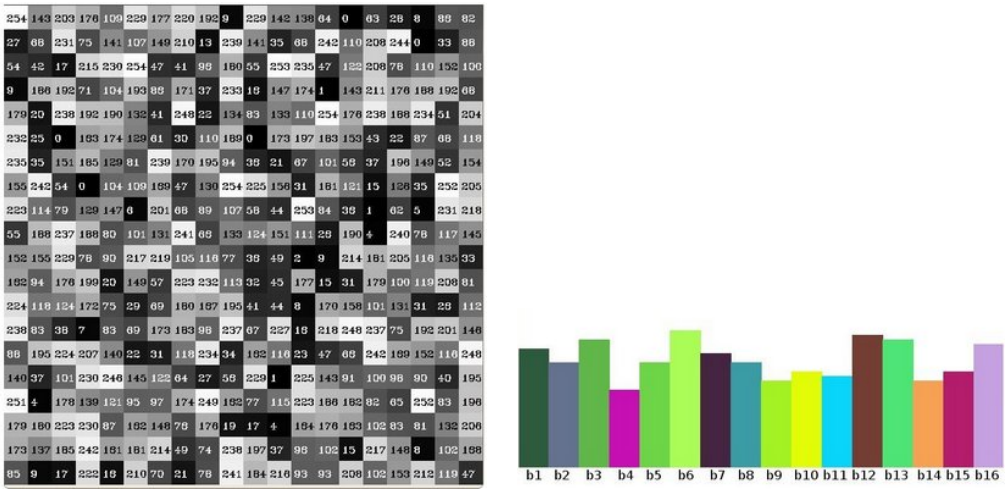
4.4. Extracción de características

Antes de pasar a explicar la fase previa a ésta, el preprocesado, es importante comprender cómo se realiza la extracción de características, pues muchos de los pasos del preprocesado van destinados a complementar o mejorar esta fase. La extracción de características consiste en procesar todas las imágenes, pertenecientes a la muestra y a la galería, y obtener de éstas un vector de características que corresponda a cada una imagen. Este vector será utilizado en la fase posterior para realizar comparaciones entre ellos, por lo que han de ser comparables entre sí.

En PyReID sólo se implementa un método de extracción de características (aunque con varias configuraciones posibles) consistente en obtener histogramas de la imagen. En la literatura se puede ver que los histogramas de color son una de las características más utilizadas [Saghafi et al, 2014]. Queda abierta la posibilidad en posibles mejoras futuras añadir otros métodos de extracción de características y realizar combinaciones entre ellos, tal como se explica en el apartado 6.2.4 sobre posibles mejoras.

4.4.1. Histogramas

Un histograma es una representación probabilística de una distribución numérica. Para construir un histograma se debe determinar el rango de los



(a) Imagen en escala de grises. (b) Histograma de la imagen.

Figura 4.9: Ejemplo de histograma de una imagen en escala de grises. Obtenido de OpenCV.

datos (valores máximo y mínimo) y el número de clases o conjuntos en los que dividir los valores (conocido como «*bins*»). Con estos datos se determina qué valores del rango pertenecen a cada bin y se cuenta la frecuencia de aparición. En el caso de las imágenes, se calculan histogramas en base a los valores de cada canal del espacio de color en el que se encuentre la imagen.

Para entenderlo mejor se usará el ejemplo del tutorial de OpenCV[2]. En la figura 4.9a se puede ver una imagen que se encuentra en escala de grises, con rango [0..255], como podría ser un canal del espacio de color RGB. Si se calcula el histograma de esta imagen usando 16 bins, los valores que entran en cada bin serían los siguientes:

$[0..255] = [0..15] \cup [16..31] \cup ... \cup [240..255]$

$rango = bin_1 \cup bin_2 \cup ... \cup bin_{n=16}$

Ahora calcular el histograma consiste simplemente en contar cuantos píxeles entran en cada bin. Esta información se puede representar en forma de gráfica de barras, como se muestra en la figura 4.9b.

Este método puede ser aplicado a una imagen y con ello se obtiene infor-

[2] `http://docs.opencv.org/doc/tutorials/imgproc/histograms/histogram_calculation/histogram_calculation.html`

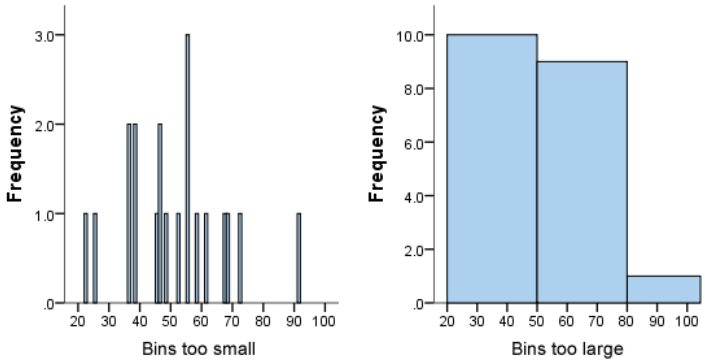


Figura 4.10: Efecto de la cantidad de bins en histograma [Laerd Statistics, 2015].

mación sobre la cantidad de distintos colores que se encuentran en ésta. Sin embargo cabe tener en cuenta ciertas características propias de los histogramas:

- Se carece de cualquier tipo de información espacial. Es decir, se conoce la distribución de colores, pero no su disposición en la imagen. Esto hace que dos imágenes distintas puedan tener el mismo histograma. Esto se intenta solventar en el apartado 4.5.2.

- Dependiendo de la cantidad de bins se tendrá más o menos información de la distribución de valores de la imagen. Como ilustra la figura 4.10 la forma del histograma puede variar según el número de bins.

4.4.1.1. Histogramas ponderados

La idea de los histogramas ponderados [Farenzena et al, 2010] va un paso más allá de los histogramas. Consiste en realizar un histograma de la imagen de la misma forma que se ha descrito anteriormente pero asignando un peso a cada píxel de la imagen. De esta forma, a la hora de contar un píxel para añadirlo a un bin u otro, su contribución no será la unidad (1) como en el caso anterior, sino que contribuirá más o menos en base al peso que tenga ese

píxel. Así, se podrá asignar más importancia a ciertos píxeles y darle menos importancia a otros.

Al poder dar más importancia a unos píxeles u otros se podrá, de alguna forma, intentar mejorar la información que da el histograma. Esto se explota en la sección 4.5.3, donde se usan cálculos de simetría y se usan distribuciones gaussianas para intentar mejorar el resultado de los histogramas.

Uno de los principales problemas que se ha encontrado a la hora de implementar este método es que OpenCV no acepta un mapa de pesos a la hora de calcular un histograma. Así se ha tenido que acudir a *NumPy* para encontrar una implementación que permita esta opción.

Sin embargo, se nota una caída en el rendimiento. Esto es algo de esperar, pues OpenCV está escrito en C++ y NumPy en Python, que es un lenguaje interpretado. Para comprobar la diferencia en el rendimiento, se realiza una pequeña función que calculará el tiempo de ejecución de la misma operación con las dos versiones. Esta función calcula un sencillo histograma de una imagen ejemplo unas 20.000 veces. Los tiempos de ejecución son los siguientes:

Versión OpenCV 0.32 segundos

Versión NumPy 9.57 segundos

Como se puede ver la versión NumPy es aproximadamente unas 30 veces más lenta que la versión de OpenCV. Ante esta situación, se indaga un poco en el código y se buscan posibles mejoras para mejorar la velocidad.

La principal razón es que el método de NumPy realiza múltiples comprobaciones de tipado, tamaño, etc. de los histogramas de entrada. Así se opta por generar nuestra propia versión, usando los distintos recursos online sobre las funciones más eficientes para la tarea. Cuando se realiza la misma prueba con esta versión del código el resultado obtenido es el siguiente:

Versión Propia 3.47 segundos

Como era de esperar, sigue siendo bastante más lento que la versión de OpenCV, pero ahora es «solamente» unas 11 veces más lento, mejorando el resultado de la versión de NumPy en un 64 %.

4.5. Preprocesado

Esta clase es en realidad una interfaz, esto es, otras clases han de heredar de ésta para ser usadas. Son clases que no albergan información por sí mismas, pero proporcionan métodos que modificarán el conjunto de datos. Hay preprocesadores que alteran las imágenes directamente, cambiando los valores de los píxeles. Otros métodos generan información que será usada por procesos posteriores como la extracción de características y comparación de características.

En la figura 4.11 se puede ver el diagrama de la clase, llamada *Preprocessing*. Es una interfaz muy sencilla, cuyo método principal necesita que se facilite un conjunto de datos como parámetro y realizará las modificaciones pertinentes.

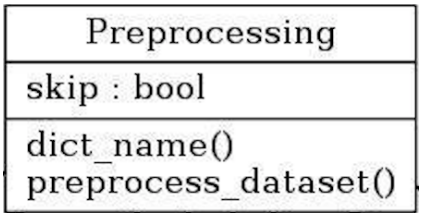


Figura 4.11: Diagrama de la clase *Preprocessing*.

En esta sección se irán describiendo uno a uno las distintas clases de tipo *Preprocessing*, explicando la base teórica y su implementación.

4.5.1. Segmentación

La segmentación consiste en separar un objeto del fondo en una imagen. En el caso de este proyecto el objetivo es separar a las personas del fondo. La segmentación puede resultar muy útil para eliminar ruido en una imagen, descartando la información del fondo y así usar solo la de la persona. Para algunos métodos es necesaria esta segmentación. En la figura 4.12 se puede ver un sencillo ejemplo de segmentación de una persona.

En la literatura muchas veces se obvia este paso por lo costoso que suelen ser estos métodos y por lo poco que aportan respecto al costo en tiempo de ejecución [Saghafi et al, 2014]. Sin embargo en este trabajo se hace uso de métodos que requieren de una segmentación previa. Además se decide

Figura 4.12: Segmentación. De izquierda a derecha: Imagen de la persona; segmentación ideal; segmentación obtenida usando GrabCut .

probar con el método GrabCut proporcionado por OpenCV, cuyo uso no es muy extendido en la reidentificación.

En realidad no hay una clase «Segmentación». Existe la clase GrabCut que implementa el método de segmentación del mismo nombre y otra clase que da la flexibilidad de cargar una segmentación ya realizada y almacenada (*MaskFromMat*).

4.5.1.1. GrabCut

GrabCut [Rother et al, 2004] es un método de segmentación basado en cortes de grafos. Para realizar la segmentación el algoritmo estima la distribución de color del objeto de interés, en este caso una persona, y la del fondo usando modelos de mezcla de gaussianas («*Gaussian Mixture Models*», GMM). El proceso[3] es el siguiente:

- El usuario indica algunas zonas que se consideran fondo y algunas objeto. Estas partes marcadas se consideraran inmutables y se usará esta información para los pasos posteriores. Como se verá más adelante, esta parte es crítica, pues cuanto más preciso sea este punto mejores resultados se obtendrán.
- Se hace un etiquetado inicial dependiendo de los datos proporcionados como objeto de interés y fondo.

[3]Tomado de: http://docs.opencv.org/master/d8/d83/tutorial_py_grabcut.html

- Se hace uso de GMM para modelar el objeto y el fondo.
- Dependiendo de los datos proporcionados, GMM aprende y crea una distribución de píxeles. Esto es, etiqueta los píxeles desconocidos como probablemente objeto o probablemente fondo dependiendo de su relación con los píxeles considerados del objeto o fondo.
- Se construye un grafo desde esta distribución de píxeles, siendo los nodos del grafo los píxeles. Se añaden dos nodos adicionales, nodo fuente («*Source node*») y sumidero («*Sink node*»), siendo conectados los nodo de objeto con el nodo fuente y los nodos fondo con el nodo sumidero.
- A todas las aristas del grafo se les asigna un valor (un peso). Las conexiones con el nodo fuente y sumidero se definen por la probabilidad del píxel de ser parte del objeto o del fondo. Los pesos entre los píxeles se definen por la similitud entre píxeles. Si hay una gran diferencia entre los colores del píxel, la arista entre ambos tendrá un valor bajo.
- Posteriormente se aplica un algoritmo de corte mínimo para segmentar el grafo. Corta el grafo en dos separando el nodo fuente y el nodo sumidero con una función de coste mínimo. La función de coste es la suma de los pesos de las aristas que son cortadas. Después del corte, todos los píxeles conectados al nodo fuente se consideran parte del objeto y los conectados al nodo sumidero se consideran fondo.
- El proceso continua hasta que la clasificación converge.

En la figura 4.13 se puede ver parte del proceso resumido.

Como se ha visto, inicialmente se le da cierta información para que pueda determinar qué considerar fondo y qué considerar objeto. OpenCV proporciona una función a la que se habrá facilitarle esta información, la cual consistirá en una matriz, del tamaño de la imagen a segmentar, con 4 valores posibles: fondo, probable fondo, objeto (persona en este caso) y probable objeto. Con este etiquetado inicial de toda la imagen, el algoritmo intentará separar la zona que interesa, la persona, del fondo. Como se trata de hacerlo de forma automática, se ha de partir de una máscara inicial común para todas las imágenes. Uno de los aspectos importantes que se abordan es la búsqueda de una máscara inicial que ofrezca buenos resultados. Esto será explorado en la sección 5.5.1 del capítulo Pruebas y Resultados.

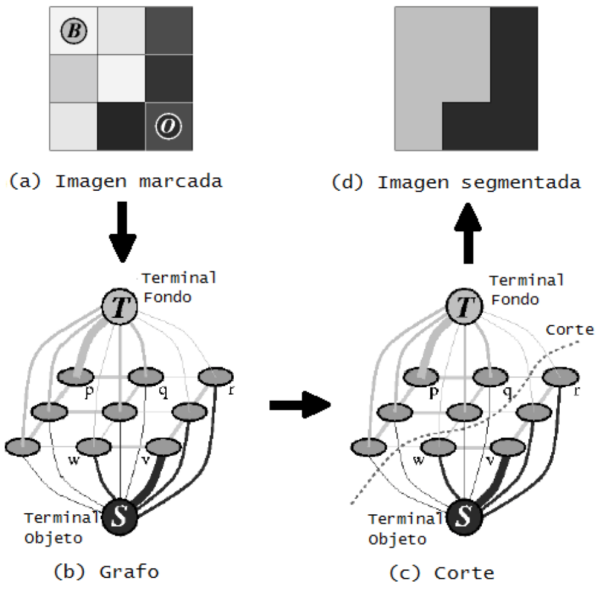


Figura 4.13: Imagen ilustrativa del proceso GrabCut [Greig et al, 1989].

En la figura 4.14 se muestra el diagrama de la clase. Se tiene el atributo *compatible_color_spaces* para controlar los espacios de color que pueden ser usados. En este caso sólo es posible usarlo con RGB. En cuanto a los métodos existen dos importantes. El método heredado *preprocess_dataset* y *preprocess_image*, que permite aplicarlo a una sola imagen en concreto si así se desea. En su inicialización hay que facilitarle la máscara que se usará como semilla, es decir, la que marca qué píxeles se consideran fondo y cuales objeto para que pueda realizar el proceso de segmentación. Los resultados, la segmentación realizada para cada imagen, se guardan en la clase ImageSet (muestra y galería) en el atributo *mask*.

Grabcut
compatible_color_spaces : list name
dict_name() preprocess_dataset() process_image()

Figura 4.14: Diagrama de la clase *GrabCut*.

Este es el único método de segmentación implementado, pero se abre la puerta a importar segmentaciones ya realizadas con la clase descrita a continuación.

4.5.1.2. MaskFromMat

Esta clase se implementa para poder incorporar segmentaciones ya realizadas. Así se ha podido aprovechar la segmentación realizada en otro trabajo. Además, cuando se realizan múltiples pruebas, se puede pre-calcular la segmentación, guardarla y luego cargarla con *MaskFromMat* para cada una de las ejecuciones. Así cada prueba no tiene que volver a calcular la misma segmentación y se ahorra en tiempo de ejecución. En el apartado siguiente se describe el método SCA y cómo se ha utilizado una segmentación ya realizada en otro trabajo para incorporarla y comprarla en las pruebas.

En la figura 4.15 se muestra el diagrama de la clase. El parámetro *invert* indica cómo se deben leer las máscaras a la hora de asignarlas en el ImageSet y saber en qué orden asignarlas.

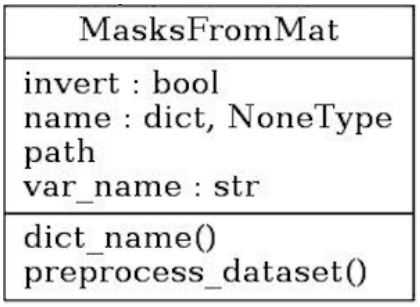


Figura 4.15: Diagrama de la clase MaskFromMat

4.5.1.3. SCA: Stel Component Analysis

Como se describe en [Farenzena et al, 2010], donde se utiliza este método para segmentar, SCA utiliza la noción de elemento de estructura (*«structure element», stel*) que puede entenderse como una porción de imagen cuya topología es consistente en el conjunto de imágenes. Esto significa que dado un conjunto de objetos (caras o imágenes de personas), un *stel* individualiza

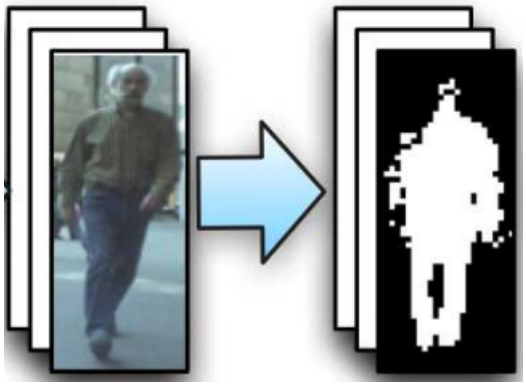


Figura 4.16: Segmentación de una imagen usando el método SCA [Bazzani et al, 2014].

la misma parte en todas las imágenes (por ejemplo el cuerpo de una persona en un conjunto de imágenes en las que aparezca una persona en cada una de ellas). Este sistema fuerza a buscar la división en 2, lo que lleva finalmente a separar la imagen en la persona y el fondo.

Este método es costoso en tiempo de computación, aunque no se ha comprobado en este proyecto. Los autores facilitan la segmentación ya realizada para el conjunto de datos que se va a utilizar y se aprovecha esto para poder comparar con otros métodos. Para ello se hace uso de la clase MaskFromMat. En la figura 4.16 se puede ver un ejemplo de la segmentación que genera el método.

4.5.2. Partición en regiones

Esta técnica consiste en subdividir la imagen en franjas, que se usarán a la hora de realizar la fase de extracción de características y comparación como regiones separadas. La forma básica de actuar es tomar la imagen completa como un todo, aplicar los diversos métodos sobre esta imagen y comparar. Sin embargo así se pierde información espacial que puede resultar interesante a la hora de identificar una característica. Para entender el problema basta con observar la figura 4.17. En ella se pueden ver dos individuos claramente diferentes. Sin embargo, sus histogramas tendrán una apariencia similar. Esto es porque al tratar la imagen completa no se tiene en cuenta algo tan simple como la distribución espacial de los colores.

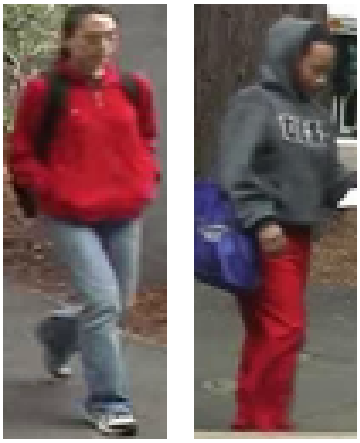


Figura 4.17: Ejemplo del problema de comparación sin regiones.

Una de las técnicas para solucionar esto es subdividir la imagen en regiones [Geng et al, 2013]. Si se divide la imagen, posteriormente se pueden comparar las regiones por separado. Normalmente se divide la imagen en franjas verticales, pues las variaciones en postura y pose hacen que separar la imagen en franjas horizontales no resulte práctico. En la figura 4.18 se observa cómo la partición en regiones hace que la comparación tenga más sentido e intuitivamente lleva a pensar que mejorará la reidentificación. Esto se comprobará en la sección 5 de Pruebas y resultados. En [Marín Reyes, 2015] también se usan franjas verticales y se comprueba cómo éstas mejoran la reidentificación.

A la hora de realizar la comparación se pueden utilizar diversas técnicas si se realiza la partición en regiones, como asignar una importancia diferente a cada región. Estas técnicas se explican en la sección 4.6.

En PyReID se permiten dos métodos para realizar la partición de regiones. Por un lado, se pueden definir directamente las regiones, en cuyo caso se usará esta partición para todas las imágenes por igual. Otra opción es usar el método propuesto en [Farenzena et al, 2010] basada en utilizar la silueta para dividir de forma automática en dos regiones la imagen, correspondientes con el tronco superior e inferior del individuo.

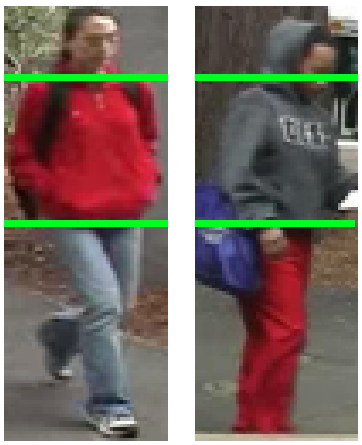


Figura 4.18: Ejemplo de problema donde la partición de regiones es interesante.

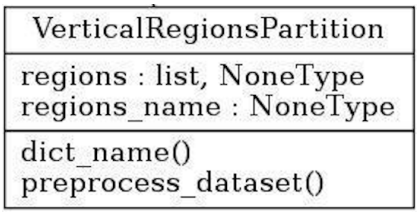


Figura 4.19: Diagrama de la clase *VerticalRegionsPartition*.

4.5.2.1. Partición en franjas horizontales

Esta técnica permite definir las regiones indicando las zonas de interés. Se trata de una partición en franjas horizontales, así que para definir una región se indicará la altura donde comienza la zona y la altura donde termina. Para que sea un método homogéneo ante distintos tamaños de imágenes, se definen usando coordenadas normalizadas entre 0 y 100, que serán luego trasladadas a cada imagen individualmente dependiendo de su tamaño.

Como se ve en el diagrama de la clase en la figura 4.19 el parámetro «*regions*» es dónde se encuentra definida la partición de regiones deseada.

Debido a su continuo uso en las pruebas, si no se define una partición concreta, la clase asume una partición por defecto. Ésta es la resultante de dividir la imagen en 6 franjas iguales y utilizar las 5 inferiores (descartando la superior, la cabeza). Se puede ver esta partición en la figura 4.20.

Figura 4.20: Ejemplo de partición en 5 regiones.

4.5.2.2. Partición basada en la silueta

Esta partición de regiones es introducida en el método SDALF [Farenzena et al, 2010]. Se basa en principios de asimetría para dividir la imagen en dos, que normalmente coincide con la cintura del individuo. Así mismo se descarta la cabeza pues con imágenes de tan baja resolución ésta aporta poca información.

Define dos operadores básicos. El primero es el operador cromático bilateral. Éste establece un eje horizontal y una distancia máxima y calcula la distancia euclídea entre los píxeles equidistantes a cada lado del eje. La suma de estas distancias es el operador cromático bilateral. Esto se realiza en los puntos que formen parte de la persona, pues se utilizará la segmentación en este proceso. Se puede ver en la imagen 4.21. i_{TL}es el eje horizontal de simetría. δ es la distancia máxima en la que se hacen los cálculos.

El segundo operador es el operador de cobertura espacial, que calcula la diferencia de área entre dos regiones. Para ello hace uso de la segmentación, usando el área que es considerada persona. Al igual que con el operador anterior, utiliza un eje horizontal de simetría y una distancia máxima. Combinando estos operadores calcula los ejes de asimetría. El primero de ellos es el eje que pretende separar al individuo aproximadamente por la cintura.

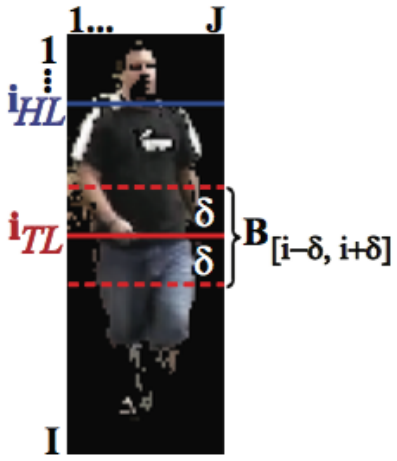


Figura 4.21: Imagen ilustrativa del proceso de partición de la silueta [Farenzena et al, 2010].

Éste se calcula con la ecuación 4.2, siendo C el operador cromático bilateral y S el operador de cobertura espacial.

$$i_{TL} = \underset{i}{argmin}(1 - C(i, \delta)) + S(i, \delta) \tag{4.2}$$

La búsqueda de este eje se centra en la zona intermedia en unos márgenes predefinidos, para acotar la búsqueda. El otro eje de asimetría separa la cabeza del resto del cuerpo. Se calcula con la ecuación 4.3.

$$i_{HT} = \underset{i}{argmin}(-S(i, \delta)) \tag{4.3}$$

Con esta ecuación se consigue separar regiones que tienen grandes diferencias en su área. La búsqueda se limita al intervalo superior, nuevamente para acotar la búsqueda.

Con estos dos ejes se consigue separar la imagen en dos regiones, descartando la cabeza, que suele aportar poca información. Así aproximadamente se tendrá la región del tronco y las piernas.

Como añadido en este proyecto se decide probar con una subdivisión extra. Es decir, se vuelven a dividir las dos regiones que se han obtenido por la

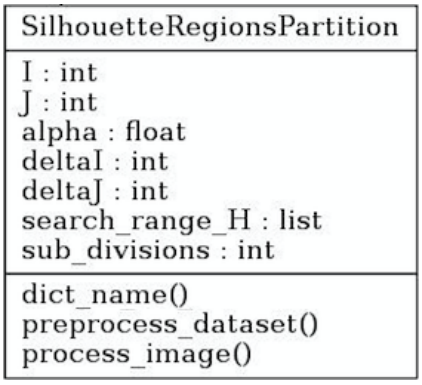


Figura 4.22: Diagrama de la clase *SilhouetteRegionsPartition*.

mitad. Así se obtienen un total de 4 regiones. En realidad se puede dividir en tantas regiones como el usuario decida, aunque esa opción no se estudia en este trabajo.

En la figura 4.22 se puede ver un diagrama de la clase. Se observan múltiples métodos y atributos. Todos ellos son usados internamente, excepto los de funcionalidad heredada y preprocess_image, usado para realizar el proceso en una imagen.

4.5.3. Mapas gaussianos

En el apartado 4.5.2.2 se describe el proceso de partición de la persona en regiones basado en la silueta realizado en [Farenzena et al, 2010]. Otra parte fundamental del proceso es el uso de histogramas ponderados (sección 4.4.1.1), generando un mapa de pesos basándose en la simetría de la imagen y una distribución gaussiana.

Este método necesita disponer de una partición en regiones de la imagen (sección 4.5.2) así como de una segmentación (sección 4.5.1). Aunque en su descripción original aparece ligado con la partición en regiones, se separa de éste para poder ser aplicado con cualquier otra partición de regiones. Sin embargo se hará uso de operadores descritos en el apartado 4.5.2.2.

Así pues, el primer paso consiste en encontrar un eje de simetría en cada región k. Para ello se utiliza el método descrito en la ecuación 4.4:

Figura 4.23: Ejemplo de ejes de simetría [Bazzani et al, 2014].

$$j_{LRk} = \underset{j}{argmin}\ w{*}C(j,\delta) + (1-w) * S(j,\delta) \tag{4.4}$$

Así para cada región k se busca el eje de simetría vertical j. C es el operador cromático bilateral y S es el operador de cobertura espacial. δ es la distancia máxima a la que se realizan los cálculos y se fija a un valor proporcional al ancho de la imagen. w representa el peso que se le da a cada operador. Un ejemplo del resultado obtenido se muestra en la figura 4.23.

El siguiente paso consiste en utilizar estos ejes de simetría para generar un mapa de pesos. Estos pesos se asignarán a cada píxel, representando su «importancia» y será usado para calcular el histograma ponderado, explicado en el apartado 4.4.1.1. Con esta técnica se pretende solventar en parte los errores en la segmentación, que es propensa a errores especialmente en los bordes de la imagen. Se verán dos opciones para generar estos mapas.

En [Farenzena et al, 2010] se propone usar un kernel gaussiano unidimensional $\mathcal{N}(\mu, \sigma)$ donde μ es la coordenada y de j_{LRk} (eje de simetría) y σ es variable para ajustar los valores. Cuanto más próximo esté un píxel al eje de simetría más importante será.

Otra alternativa, que busca mejorar los resultados obtenidos en la propuesta anterior, se presenta en [Wei and Lin, 2013]. En este trabajo se comenta que utilizar un sólo kernel para la región superior e inferior no es lo suficientemente bueno. Su propuesta pasa por proponer una combinación de modelos gaussianos (*«GMM»,* del inglés *Gaussian Mixture Models).* Ésta queda reflejada en la ecuación 4.5:

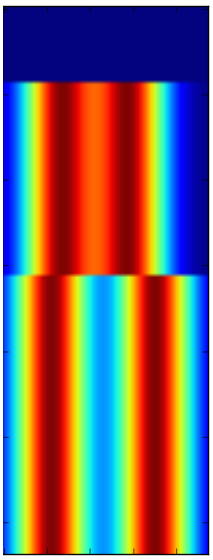


Figura 4.24: Efecto de la variación de la desviación en GMM. Valor de σ: 8; Desviación superior: 8; Desviación inferior: 12.

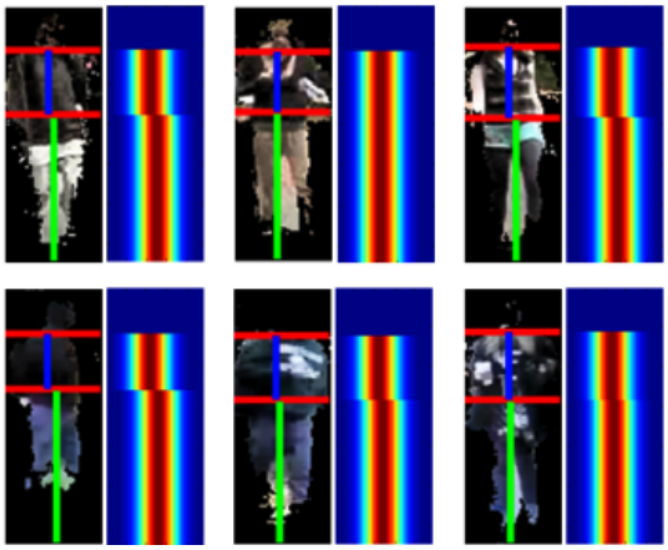
$$G_{MIX}(\mu, \sigma) = w_1 G(\mu - x_1, \sigma_1) + w_2 G(\mu + x_2, \sigma_2) \tag{4.5}$$

Donde w_1 y w_2 representan el peso de cada núcleo gaussiano y x_1 y x_2 el desplazamiento de μ. En la implementación se acaba simplificando y se establecen los pesos w_1 y w_2 a 0.5. También se igualan x_1 y x_2 de forma que se mantenga la simetría. Para entender la idea, se puede ver en la figura 4.24 el efecto de variar la desviación, manteniendo el mismo valor de sigma (σ). El color azul representa el valor nulo y el valor rojo el máximo valor posible. Como se ve se pueden conseguir estructuras más ricas y complejas.

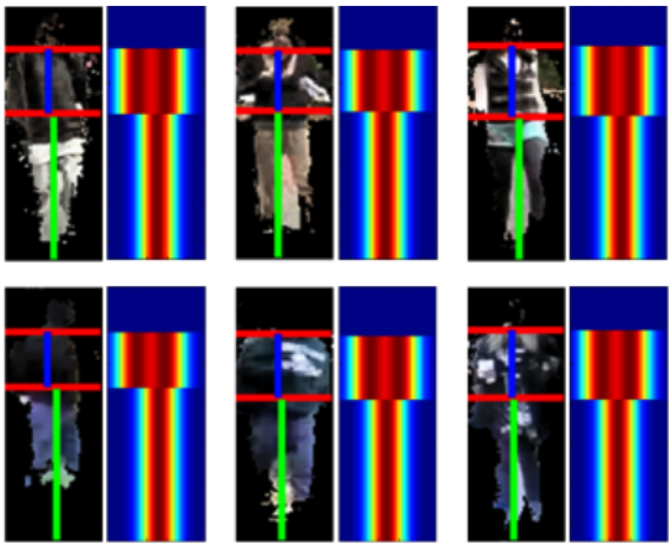
En la figura 4.25 se muestran ejemplos de mapas obtenidos con las dos técnicas descritas. Aquí se puede ver la diferencia entre ambos tipos de mapas.

A la hora de implementarlos se decide utilizar una única clase que se encargue de ambos métodos, usando los atributos para definir si se usa uno u otro. En la figura 4.26 se observa el diagrama de la clase. Posee muchos atributos, la mayoría para cálculos internos. A la hora de inicializar la clase, se han de definir los siguiente:

- Método a usar, mediante una ristra: «GMM» o «Gaussian»



(a) Mapas obtenidos con la técnica del kernel simple [Farenzena et al, 2010].



(b) Mapas obtenidos con la técnica de mezcla de kernels gaussianos[Wei and Lin, 2013].

Figura 4.25: Mapas gaussianos.

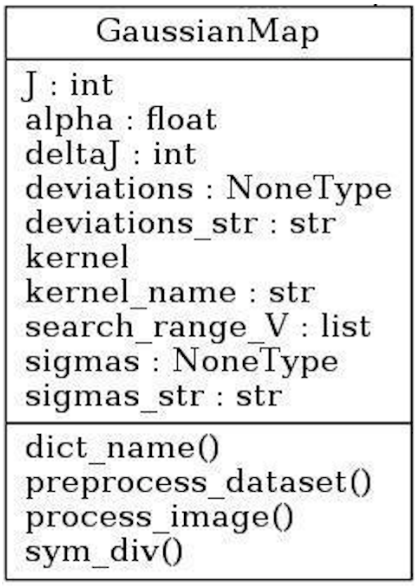


Figura 4.26: Diagrama de la clase *GaussianMap*.

- *alpha*: Define el peso que se le da a cada operador a la hora de calcular la simetría, ecuación 4.4.
- *sigmas*: Valor de σ en el kernel gaussiano. Se definen tantos como regiones hayan.
- *deviations*: En el caso de G_{MIX}, ecuación 4.5, se debe definir la desviación con respecto al eje de simetría en el que calcular los kernel gaussianos. Tantos como regiones haya.

4.5.4. Función de transferencia de brillo (BTF)

En un escenario real es poco probable que dos cámaras tengan las mismas condiciones de iluminación al estar ubicadas en sitios diferentes. Una de las posibles técnicas para solucionar esto es la conocida como función de transferencia de brillo, BTF, propuesta originalmente en [Grossberg and Nayar, 2003]. En este trabajo se aplica el BTF para recuperar imágenes que capturan una escena estática con diferentes tiempos de exposición. El método del BTF es aplicado en reidentificación en [Javed et al, 2005]. El objetivo de esta

técnica es mapear valores de brillo entre dos cámaras distintas. Se presupone que las condiciones de brillo de la escena capturada por una cámara no varían, aunque en un escenario real esta presunción no es valida. Sin embargo, se simplifica el problema considerablemente. Hay variantes de este método que intentan tener en cuenta esta situación [Javed et al, 2008], pero en este proyecto no se abordarán.

Para calcular el BTF se necesitan un par de observaciones de la misma persona obtenidas en dos cámaras distintas. Dado que las imágenes serán diferentes unas de otras, con variaciones en la pose y ángulo, no se puede hallar una correspondencia píxel a píxel que permita calcular la función de transferencia. La propuesta de este método se basa en la suposición de que al tratarse del mismo individuo la distribución de los histogramas (explicados en la sección 4.4.1) entre ambas imágenes será relativamente similar. Dadas las diferencias de brillo, se asumen histogramas similares aunque escalados. En un escenario real esta función de escalado será no lineal, pero seguirá siendo una función monótona.

Se puede definir la función de transferencia de brillo como se muestra en la ecuación 4.6:

$$f_{ij}(B_u) = B_v \tag{4.6}$$

Donde i y j son las observaciones del mismo individuo desde diferentes cámaras. B_u representa el brillo en la imagen i y B_v el brillo en la imagen j. Con las suposiciones anteriores sobre la relación entre histogramas se puede desarrollar la función como se muestra en las ecuaciones 4.7 y 4.8:

$$H_i(B_u) = H_j(B_v) = H_j(f_{ij}(B_u)) \tag{4.7}$$

$$f_{ij}(B_u) = H_j^{-1}(H_i(B_u)) \tag{4.8}$$

Esto se ha de hacer para cada canal de la imagen (RGB) por separado.

Este método requiere por tanto de un conjunto previo de entrenamiento, sobre el que se calculará el BTF para cada par de imágenes facilitado. En este punto existen diferentes formas de unir los BTF de cada par. Por un lado, existe el método conocido como MBTF (*mean Brightness Transfer Function*), que se calcula de la forma mostrada en la ecuación 4.9:

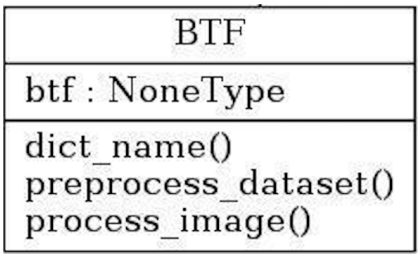


Figura 4.27: Diagrama de la clase BTF.

$$MBTF = \frac{1}{n} \sum_{i=1}^{n} f_i \tag{4.9}$$

Como se ve, consiste en calcular todos los BTF individuales y luego hallar la media [Javed et al, 2005].

El otro método es CBTF (*cumulative Brightness Transfer Function*) [Prosser et al, 2008] en el que lo que se hará será acumular toda la información de las observaciones y luego calcular sobre esto el BTF. Esto es, se calculan los histogramas de las imágenes de cada cámara, se suman en uno solo y luego se calcula el BTF.

En el diagrama de la clase, figura 4.27, se observa que tiene los métodos comunes, y sólo un atributo para indicar el tipo de BTF que se va a realizar («CBTF» o «MBTF»).

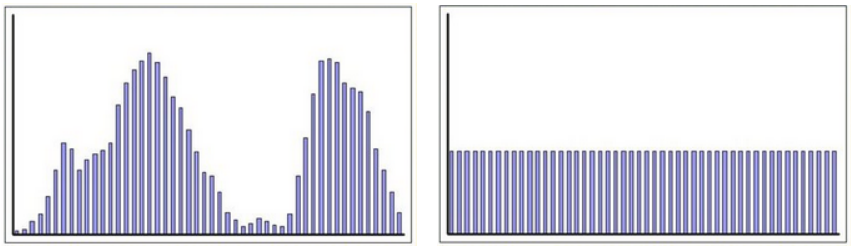
4.5.5. Normalización de la iluminación

La normalización de la iluminación es una idea que se toma de [Wei and Lin, 2013]. En este trabajo se menciona cómo se realiza una normalización del canal de iluminación para mejorar los resultados. Se refiere a la técnica conocida como ecualización de histogramas (descritos en el apartado4.4.1). Con esta técnica se genera una imagen con unos niveles de intensidad igualmente probables y que cubren el rango total de valores (desde el mínimo al máximo). Con ello se consigue que se aumente el rango dinámico, consiguiendo un mayor contraste.

La ecualización de histogramas se aplica sobre un sólo canal, por lo que suelen verse ejemplos de aplicación en imágenes en valores de grises. Sin embargo

puede trasladarse fácilmente a imágenes a color. La idea consiste en utilizar espacios de color en los que exista un canal que represente la luminancia o brillo y utilizar este canal para realizar la ecualización. Así, los colores originales no se transforman, pero el brillo de las imágenes sobre las que se aplique será muy similar. Por tanto, se espera paliar en parte el problema de la variación de iluminación en el problema de la reidentificación.

Como se ha mencionado, la idea de la ecualización de histogramas consiste en lograr que la distribución de valores en el histograma sea uniforme. Esto se puede ver representado en la figura 4.28. A la izquierda se tiene el histograma original y a la derecha el histograma ecualizado.



(a) Histograma original. (b) Histograma ecualizado.

Figura 4.28: Efectos de la ecualización del histograma.

Para conseguirlo lo primero que se hace es calcular el histograma acumulado. Éste es igual que el histograma, pero se acumulan los valores de los bins anteriores. En la figura 4.29 se muestra un histograma normal junto a uno acumulado. Este histograma tiene como valor final el número de píxeles de la imagen.

A continuación se debe realizar una normalización de el histograma acumulado de tal forma que el valor mínimo sea 1 y el máximo 255 (o el rango máximo en el que se mueva el canal que se está normalizando). Esto es consigue aplicando la ecuación 4.10:

$$h(v) = round\left(\frac{cumhist(v) - cumhist_{min}}{P - cumhist_{min}}) * (L-1)\right) \tag{4.10}$$

En esta ecuación se muestra el nuevo valor de un píxel con valor v, donde $cumhist$ es el histograma acumulado, P es el numero de píxeles de la imagen

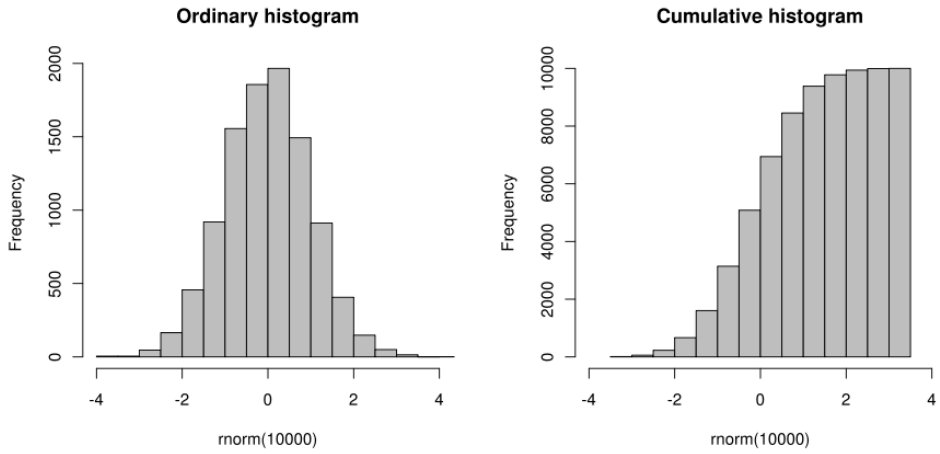


Figura 4.29: Histograma normal frente a histograma acumulado [Kierano, 2015].

(su tamaño) y L es el valor máximo posible en el canal (normalmente 255, aunque depende de como se represente el espacio de color).

Aplicando este método a cada píxel de la imagen se consigue realizar la ecualización en la imagen. Como se ha mencionado, esto sólo puede ser aplicado en un canal. En una imagen en escala de grises se puede aplicar directamente, pero en una imagen representada en RGB, con tres canales, no se puede aplicar directamente a cada uno de los canales, pues afectaría a los colores de la imagen. Por tanto, en PyReID se permite la normalización de la imagen en espacios de color en los que se tenga un canal que represente el brillo o luminancia, aplicando la ecualización a estos canales.

Se dispone de dos espacios de color en los que aplicarlo: *HSV* e *YCbCr.* Ambos espacios de color tienen un canal específico para luminancia, V en el caso de HSV e Y en el caso de YCbCr. Cuando se aplica la normalización, se normaliza este canal y el resto de canales no se ve afectado. Un ejemplo del efecto de la normalización se puede observar en la figura 4.32.

Esto no limita al sistema a tener que trabajar únicamente con estos espacios de color en caso de que se quiera aplicar la normalización de la iluminación. Como se puede ver en la figura 4.30 el proceso consiste en convertir la imagen al espacio de color a normalizar, realizar la normalización y luego volver a convertir la imagen a su espacio de color original.

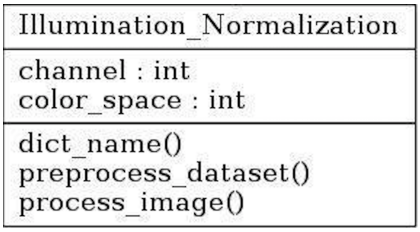


Figura 4.31: Diagrama de la clase *Illumination_Normalization*.

Figura 4.30: Proceso de normalización de la iluminación.

En el diagrama de la clase, figura 4.31, se observa que tiene la estructura esperada, cuyas únicas peculiaridades consisten en los atributos para indicar el espacio de color y el canal sobre el que se desea realizar la normalización.

4.6. Comparación de características

Una vez se han extraído las características del conjunto de datos, llega el momento de compararlas. En este paso, se compararán las características extraídas (histogramas en este caso) de cada elemento de la muestra con cada elemento de la galería. Como resultado en esta etapa se tendrá una matriz de comparación, sobre la que luego se puede calcular la ordenación para obtener la matriz de ordenación, objeto que será devuelto por *Execution* (apartado 4.1).

Como sólo se implementan histogramas como método de extracción de características, el método de comparación implementado es *Comparación de Histogramas*. Como se verá, se trata en realidad de múltiples métodos de comparación entre histogramas.

4.6.1. Comparación de histogramas

Para comparar dos histogramas se ha de definir primero una métrica. Serán usadas las medidas de distancia facilitadas por OpenCV, con su función

Figura 4.32: Normalización de la iluminación para dos imágenes. En la primera columna se muestra la imagen original, en la segunda la normalización en el espacio YCbCr y en la última columna en el espacio HSV.

compareHist. Ésta admite un parámetro en el que se podrá indicar el método de comparación a utilizar. Estos métodos de comparación operan sobre dos histogramas (H y H') que poseen los mismos bins. Los métodos que facilita son los siguientes:

- **Intersección** [Swain and Ballard, 1991]: En este método se calcula, bin a bin, la intersección entre estos, es decir, las partes comunes del histograma. Viene dado por la ecuación 4.11:

$$d(H, H') = \sum_{i=1}^{n} min(H_i, H'_i) \tag{4.11}$$

- **Chi-cuadrado** [Pele and Werman, 2010]: Distancia de origen estadístico. En esta distancia se tiene en cuenta que la diferencia entre dos conjuntos pequeños es más importante que la diferencia entre dos grandes conjuntos. Se define en la ecuación 4.12:

$$d(H, H') = \sum_{i=1}^{n} \frac{2(H_i - H'_i)^2}{(H_i + H'_i)} \tag{4.12}$$

- **Correlación** [Marín Reyes, 2015]: Relación estadística entre bins equivalentes, que implica dependencia entre ambos. La ecuación 4.13 muestra esta distancia:

$$d(H, H') = \frac{\sum_{i=1}^{n}(H_i - \bar{H})(H'_i - \bar{H'})}{\sqrt{\sum_{i=1}^{n}(H_i - \bar{H})^2 \sum_{i=1}^{n}(H'_i - \bar{H'})^2}} \tag{4.13}$$

- **Bhattacharyya** [Bhattacharyya, 1946]: En esta medida de distancia se comparan los elementos de ambos conjuntos, midiendo la similitud de las dos distribuciones de probabilidad. Se refleja en la ecuación 4.14:

$$d(H, H') = 1 - \sqrt{\sum_{i=1}^{n} \frac{\sqrt{H_i H'_i}}{\sqrt{\sum_{i=1}^{n} H_i \sum_{i=1}^{n} H'_i}}} \tag{4.14}$$

Además de los métodos disponibles en OpenCV, también se añade la distancia **Euclídea** [Deselaers et al, 2004]. Ésta es una comparación bin a bin

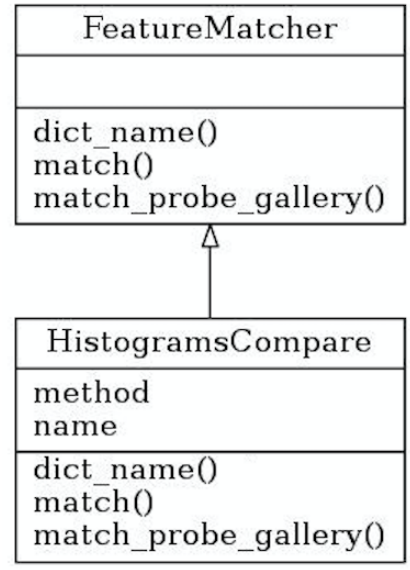


Figura 4.33: Diagrama de la clase *FeatureMatcher*.

pero que no tiene en cuenta si hay valores similares en bins subyacentes. Esta distancia se define en la ecuación 4.15:

$$d(H, H') = 1 - \sqrt{\sum_{i=1}^{n} (H_i - H_i')^2} \tag{4.15}$$

A la hora de realizar la comparación de histogramas se ha de tener en cuenta si existe una partición de regiones. En este caso se puede establecer un peso para cada región. De esta forma, se calcula la comparación entre los histogramas de cada región por separado, realizando una sumatoria pesada para obtener un resultado ponderado. Si existe partición en regiones pero no se facilita ningún peso, los histogramas de las distintas regiones se concatenan y son considerados como uno solo a la hora de realizar la comparación.

En la figura 4.33 se muestra el diagrama de la clase. Se muestra la herencia desde la clase *FeatureMatcher*. Cuando se quiere comparar un conjunto de datos entero se usa el método *match_probe_gallery*, usando el método *match* cuando sólo se quiere comparar dos histogramas.

4.7. Estadísticas

Una vez se ha obtenido la matriz de ordenación cuando se ha finalizado una ejecución, se puede tratar ésta para obtener una serie de datos que muestren de forma resumida cuál ha sido el resultado y cuán buena es una determinada ejecución, pudiéndose comparar diferentes técnicas y métodos de forma sencilla. A esta clase se le facilita el conjunto de datos y la matriz de ordenación. Cuando se ejecuta, calcula y almacena la información que resume la calidad de una ejecución concreta. Los datos que calcula y almacena son los siguientes:

CMC Para obtener la matriz de ordenación se compara un elemento de la muestra con todos los elementos de la galería. Esto se repite para cada elemento de la muestra y así se obtiene la matriz, donde las filas son los elementos de la muestra y las columnas los elementos de la galería. La curva CMC (*cumulative matching characteristic*) [Gray and Tao, 2008] representa la expectativa de encontrar la correspondencia correcta (un individuo de la muestra con uno de la galería) entre las x primeras posiciones. Así se forma una curva para cada valor de x. En el caso ideal, una reidentificación perfecta, se consigue el 100 % en la posición $x = 1$.

Rango-X Esta medida indica el valor de la curva CMC en la posición x. Cuando se requiera calcularlos basta con consultar en la curva CMC la posición x.

AUC Área bajo la curva CMC. El AUC (*«Area Under the Curve»)* permite resumir la curva CMC en un sólo valor. Aunque no es lo mas representativo, pues dos curvas distintas pueden tener la misma área, es la medida que se usará principalmente para medir la calidad de una configuración pues además es independiente del tamaño del conjunto de datos si está normalizada.

Valor Medio Valor que indica la posición media en la matriz de ordenación en la que se encuentra la correspondencia. Se calcula sumando la posición de la correspondencia en la galería para cada elemento de la muestra y dividiendo entre el número total de elementos en la muestra.

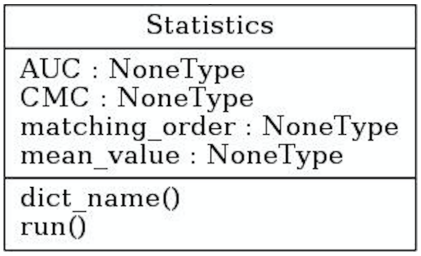


Figura 4.34: Diagrama de la clase *Statistics*.

En la figura 4.34 se muestra el diagrama de la clase *Statistics*. Se observan los atributos que se encargan de almacenar la curva CMC, el valor AUC y el valor medio. «*matching_order*» es un atributo que se usa para cálculos intermedios. No se almacena un atributo independiente para almacenar el *Rango-X*, pues se obtiene directamente desde la curva CMC. Éste se calcula cuando se usa el método *«dict_name()»*, al que se le puede pasar un parámetro para indicar los diferentes *Rango-X* que se desea que devuelva.

4.8. Validación cruzada

Esta clase surge por la necesidad de realizar múltiples pruebas. Cuando se desea probar cómo funciona una configuración en un conjunto de datos y obtener unas estadísticas (como el CMC) de esta ejecución se puede realizar de forma sencilla. Se configura la clase *Execution* y a la salida de esta clase (matriz de ordenación) se le aplica la clase *Statistics*. Pero ¿Qué ocurre si se desea realizar la prueba con, por ejemplo, la mitad de los elementos de un conjunto de datos? ¿O cuando se tienen elementos de entrenamiento (como en el método BTF)? En estos casos lo deseable es que los elementos sean escogidos de forma aleatoria, para evitar sesgos en los resultados obtenidos. Pero entonces, cada ejecución da un resultado distinto. ¿Cuál es el resultado que se debe escoger como bueno?

Para solucionar estos casos se utiliza la validación cruzada. Concretamente se ha implementado la versión de validación cruzada aleatoria (figura 4.35). La validación cruzada aleatoria consiste en realizar la misma ejecución una serie de veces, escogiendo cada vez los elementos de test y entrenamiento de forma aleatoria y luego promediar los resultados. Cuando no se utiliza el conjunto entrenamiento, consiste simplemente en variar los elementos que se escogen

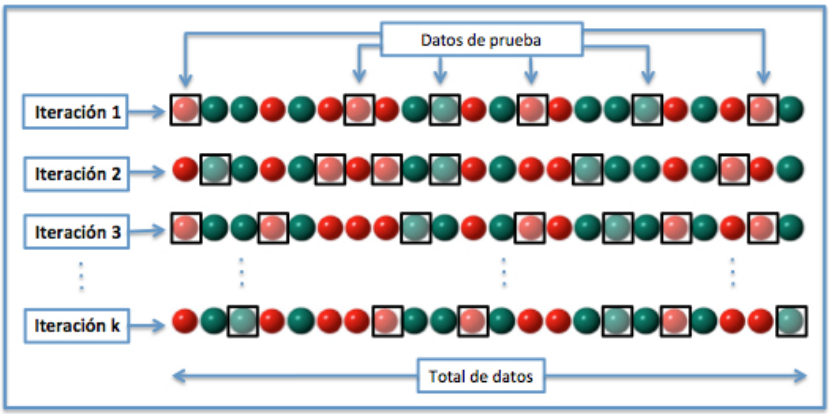


Figura 4.35: Validación Cruzada Aleatoria [Joan.domenech91, 2011].

para *test*. Esto se produce en el caso de que no se use el conjunto completo del conjunto de datos, sino una parte de éste. Como se verá en la sección 5 de Pruebas y resultados se utilizará la mitad del conjunto de datos para evaluar el sistema, realizando 10 ejecuciones y promediando el resultado obtenido.

Como se ha dicho, se escogen los datos de cada ejecución de forma aleatoria. Esto sin embargo tiene la desventaja de que habrá muestras que pueden quedar sin evaluar, u otras muestras que se evalúan muchas veces. Esto puede llevar a que los resultados entre varias validaciones cruzadas sean ligeramente distintas. Para poder realizar una prueba en las mismas condiciones exactas se necesitaría usar los mismos datos. Esto puede interesar cuando se comparan dos métodos distintos.

Esta clase, denominada *CrossValidation,* también permite utilizar un fichero de texto en el que se especifiquen los conjuntos que se usarán para cada una de las ejecuciones de validación cruzada. De esta forma, se puede realizar la validación cruzada usando siempre los mismos conjuntos. Así la comparación entre distintas configuraciones es más justa.

Otra de las características fundamentales de la clase *CrossValidation* es que permite almacenar los datos en una hoja de cálculo en formato excel. Para ello es fundamental la implementación en cada clase de la función *dict_name*, pues esta será llamada para cada clase. La clase se encarga de devolver la información pertinente de ser guardada en la hoja de cálculo, con un formato concreto. Si el nombre del fichero donde guardar estos datos ya existe, no lo sobrescribe, si no que añade los datos a continuación. Así se podrá generar

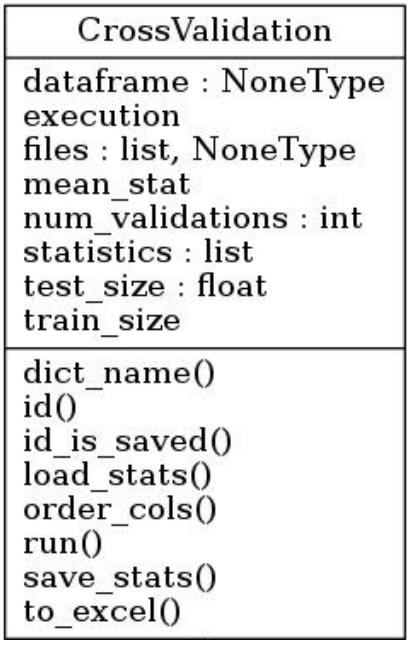


Figura 4.36: Diagrama de la clase *CrossValidation*.

una sola hoja de cálculo en la que tener todos los datos de las distintas ejecuciones.

El diagrama de la clase se puede ver en la figura 4.36.

4.9. Reordenación

Hasta ahora se ha visto como se genera la ordenación inicial. Es decir, para cada elemento de la muestra se tiene una lista ordenada en base a la similitud, de todos los elementos de la galería. El enfoque de la optimización por reordenación es el de realizar una mejora de estos resultados obtenidos en el paso anterior. Por tanto, se obtiene una ordenación inicial para un elemento de la muestra concreto, y se realiza algún proceso extra para mejorar esos resultados.

Este proceso no está incluido como parte de la ordenación inicial. En nuestro caso se debe a que se utilizará información facilitada por el usuario una vez se produce la ordenación inicial. Existen múltiples técnicas para realizar la

Figura 4.37: Ejemplos de selecciones del usuario [Liu et al, 2013].

reordenación, algunas de ellas no supervisadas. La propuesta en este proyecto se engloba en la semisupervisada, pues se pide al usuario que etiquete unas pocas muestras, y con esta información se intenta mejorar los resultados de la ordenación inicial.

Para la implementación de este sistema se ha utilizado como inspiración el sistema propuesto por [Liu et al, 2013], el método POP (*Post Ranking Optimization*). El otro trabajo que se tiene en cuenta es el propuesto por [Wang et al, 2014], RIRO (*Region-based Interactive Ranking Optimization*). Ambos son explicados a continuación.

4.9.1. POP (Post Ranking Optimization)

En POP se propone un modelo en el que el usuario, tras la realización de una ordenación inicial automática, sólo tiene que marcar en las imágenes de la galería una como ejemplo negativo (*no similar*) y opcionalmente ejemplos de imágenes similares a la de la muestra.

Una imagen negativa marcada por el usuario se refiere a una imagen que se encuentra entre los primeros resultados de la ordenación pero el usuario ve claras diferencias con la imagen objetivo. Una imagen marcada positiva hace referencia a una que no es la imagen objetivo pero tiene claras similitudes con la imagen que se busca. En la figura 4.37 se muestran ejemplos de imágenes seleccionada por el usuario.

En POP también se propone un método denominado *Expansión Visual*, por el cual, usando un conjunto de entrenamiento, es capaz de generar muestras pseudo positivas, con las que mejorar el proceso. Consiste en generar muestras *similares* a partir de un sistema de regresión entrenado previamente.

El proceso es el siguiente:

1. Se obtiene una ordenación inicial (ésta es independiente del método POP).
2. El usuario selecciona al menos una muestra no similar. Puede seleccionar más de una. Opcionalmente puede seleccionar también muestras similares.
3. El sistema genera muestras por el método de expansión visual si hay más no similares que similares, para que haya la misma cantidad total.
4. Se construye la matriz de afinidad. Para ello, se utiliza un método de clustering. Se usa el método *Clustering Forest*. Con ello se obtiene un valor de afinidad entre cada par de imágenes.
5. Las imágenes afines a las no similares son penalizadas y las afines a las similares son premiadas.
6. Se utiliza el método *Laplacian SVM* para generar una función que propague la información, recalculando los valores de cada imagen.

El proceso que se sigue en este proyecto es similar, aunque se opta por simplificar la última parte, no usando el método de Laplacian SVM. Esto se debe principalmente a la escasa explicación que se provee y la dificultad de su implementación. En su lugar se decide tomar este método como inspiración y explorar otros métodos.

A continuación se describe el otro trabajo del que se han tomado prestadas algunas ideas.

4.9.2. RIRO

Este método se diferencia del método POP principalmente en que basa el marcado de las imágenes en regiones, no en la imagen completa. Según sus

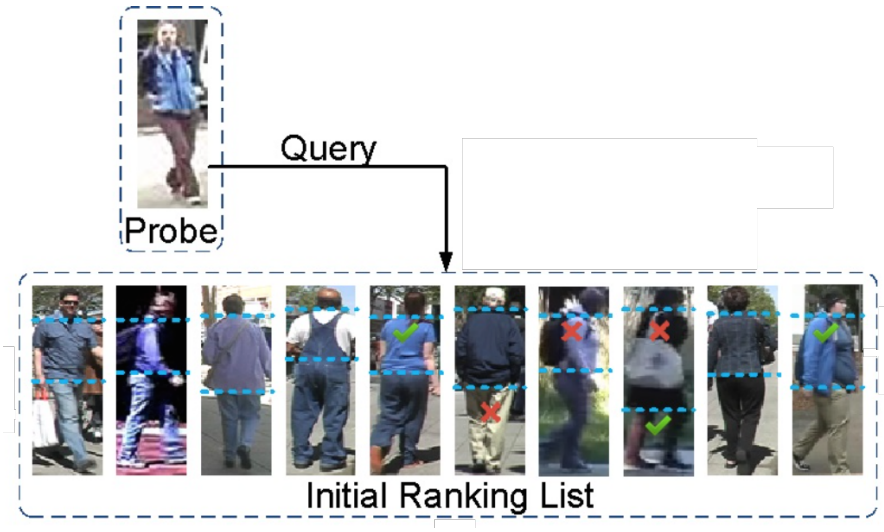


Figura 4.38: Muestra simplificada del proceso de marcado con el método RIRO [Wang et al, 2014].

propias pruebas con usuarios reales, muestran cómo es mucho más fácil para el usuario elegir muestras no similares o similares si pueden seleccionar específicamente una región de la imagen, no la imagen completa.

Así la idea es similar a POP, pero trabajando con regiones de imágenes, no con imágenes completas. La figura 4.38 muestra un ejemplo. En ésta se puede ver la ordenación inicial para un elemento de la muestra. Las lineas discontinuas azules marcan las regiones y se pueden ver marcadas muestras positivas y negativas.

El método para modificar la ordenación en RIRO es algo diferente al método en POP:

1. Tras la ordenación inicial, el usuario elige regiones similares y no similares. El sistema no genera elementos por expansión visual ni métodos similares, así que se exige que haya la misma cantidad de ambos tipos de muestras.

2. Se calcula la similitud de cada imagen de la galería con las imágenes marcadas. Se comparan sólo las regiones, no la imagen completa.

3. También se calcula la similitud basada en vecindad. Esta es similar a la aplicada en POP. Se utiliza un método de clustering con las imágenes de la galería. Los clusters en los que caigan imágenes con regiones no similares serán influenciados negativamente, decreciendo el valor de las imágenes que se encuentren en este cluster. Lo contrario ocurrirá para muestras similares.

4. Se combinan ambos métodos y se modifica la puntuación de cada imagen de la galería, lo que propicia un cambio en las posiciones de la ordenación, idealmente para atraer a posiciones superiores la imagen objetivo.

4.9.3. Sistema propuesto

El sistema que se propone en este proyecto toma un poco de cada uno de los trabajos, ofreciendo además cierta libertad de configuración, lo que aumenta las posibilidades. En resumen el sistema tiene las siguientes características:

- Se puede elegir si trabajar con regiones o con la imagen completa.
- El usuario puede escoger si quiere que las selecciones de las muestras sean balanceadas o no. Esto es, si se fuerza que existan la misma cantidad de similares como de no similares. A diferencia de los métodos vistos se decide experimentar con esta posibilidad para ver qué resultados se obtienen.
- Se ofrece opción de generar expansión visual, tanto si se trabaja con regiones como si es con la imagen completa.
- El método general de modificación de la puntuación para cambiar la ordenación consiste en:
    - Se calcula la similitud de cada imagen de la galería con cada una de las muestras similares o no similares. Esto genera la puntuación de similitud. Se usa para ello el método de comparación de histogramas, sección 5.2.
    - Se realiza la matriz de afinidad de los elementos de la galería. Para ello se usa el método *Ramdon Trees Embedding*, un método

de clustering (apartado 4.9.5) basado en árboles. Con ello se puede calcular la afinidad entre las distintas imágenes. Según la afinidad de cada imagen con las muestras similares o no similares éstas verán modificadas su puntuación.

- Se entrena el método Label Propagation/Spreading (apartado 4.9.6) con las muestras similares y no similares. Posteriormente se utiliza el método para predecir sobre todas las muestras si estas son del tipo similares o no similares. Se usa esta puntuación para estimar si son similares o no a las marcadas manualmente.
- Usando la similitud, la afinidad y la puntuación obtenida con el método de Label Propagation/Spreading se calcula una nueva puntuación para cada imagen de la galería. Esto hace que el orden de las imágenes de la galería en la ordenación cambie.

- El proceso anterior se puede repetir múltiples veces. Idealmente la imagen objetivo se acercará a las primeras posiciones.

En los siguientes apartados se describen cada una de las distintas características mencionadas.

4.9.4. Expansión visual

Como se ha comentado en apartados anteriores, POP establece que se necesitan tantas muestras no similares como similares a la hora de realizar la reordenación. Ésta no es una limitación que esté impuesta en el sistema que se usará en este proyecto, por lo que realmente no es necesaria. Sin embargo se ofrece como opción y se estudiará si ofrece algún beneficio.

Este método se basa en crear nuevos elementos partiendo de la imagen original de la muestra. Se utilizará un *Random Forest* de regresión para crear nuevas muestras. Los Random Forest [Breiman, 2001], o bosques aleatorios, son combinaciones o *ensembles* de árboles predictores en los que cada árbol se entrena con los valores de un vector aleatorio de características con la misma distribución para cada uno de los árboles. La salida o respuesta del bosque es el promedio de la salida de cada uno de los árboles que lo compone.

Así pues, se entrenará este bosque con pares de imágenes de individuos previamente etiquetados, cada una correspondiendo al mismo individuo desde

las dos cámaras para las que se realiza la reidentificación (muestra y galería). Al tener el random forest entrenado, se podrá usar como entrada la imagen de la muestra. Como salida se obtiene la predicción de cómo sería la correspondiente muestra en la galería.

Esta predicción realizada por el random forest por sí misma no es la que interesa. Sin embargo, como se ha descrito los bosques no son más que un ensamblado de distintos árboles. Si el bosque se compone de T árboles, se podrá tomar una parte de estos árboles para generar una nueva muestra que sea tomada como positiva, en lugar de utilizar el 100 % de los árboles disponibles. En Liu et al [2013] proponen tomar $\frac{2}{3}T$ árboles. Estos árboles serán seleccionados aleatoriamente cada vez, por lo que el resultado obtenido será ligeramente distinto cada vez, ofreciendo distintas muestras que podrán ser usadas para generar nuevas muestras.

Los parámetros que permiten modificar el comportamiento del bosque de regresión usado para la expansión visual son rfr_estimators y rfr_leafs. El primero define el número de árboles en el bosque. Por defecto establecido a 20.

El segundo parámetro, rfr_*leafs*, define el número mínimo de elementos que han de existir en una hoja recién creada. Si una hoja no contiene este numero mínimo de elementos, se descarta. Esto permite controlar la cantidad de hojas para cada árbol y asegura que no se creen clases con muy pocos elementos. El valor por defecto es 5.

4.9.5. Clustering: *Random Trees Embedding*

Para realizar el cálculo de la afinidad o vecindad entre imágenes, utilizado tanto en POP como en RIRO se hace uso de clustering, En este caso, se decide usar la técnica *Random Trees Embedding* [Moosmann et al, 2007]. Esto se decide por varias razones. En primer lugar debido a que se encuentra implementado en la biblioteca *sklearn*. En segundo lugar, porque en POP proponen usar un random forest cuyos árboles sean *Clustering Trees*, pues comentan que el mecanismo de selección de características implícito es beneficioso para mitigar el ruido en las características. Si bien no se trata de exactamente el mismo tipo, también se trata de un método de clustering basado en árboles.

Este tipo de bosque toma las entradas y las clasifica en base a sus características. Cada árbol termina en una serie de hojas finales, donde una entrada

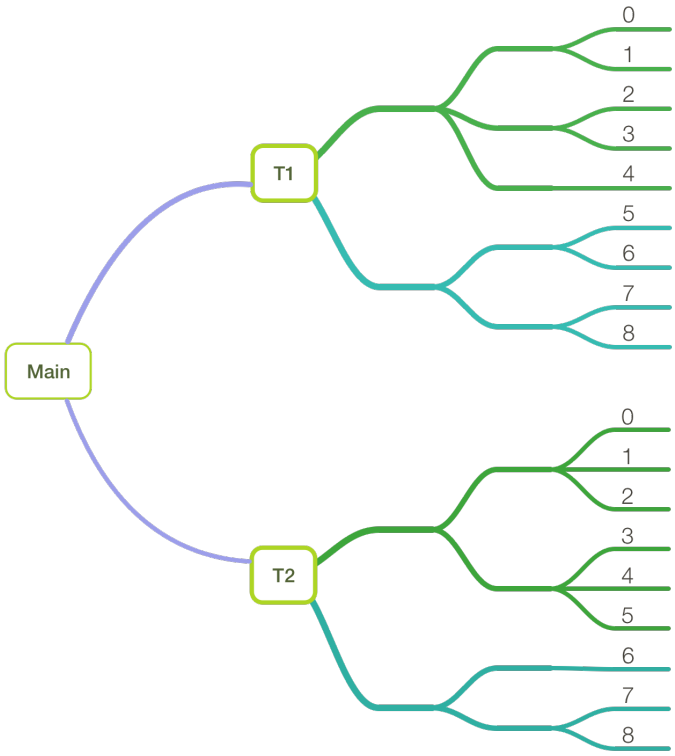


Figura 4.39: Ejemplo de Bosque compuesto de dos árboles con 9 hojas cada uno.

es clasificada en una de estas hojas. Estas hojas finales pueden ser consideradas los clusters. El bosque realiza esto en cada uno de los árboles que lo componen. Por tanto se obtienen en total T clasificaciones, siendo T el número total de árboles que componen el bosque. Esto es, para cada entrada se puede clasificar su salida como la *hoja* o *cluster* en que cae para cada uno de los árboles. Se obtiene así un vector de hojas.

A continuación se puede calcular la afinidad como el número de árboles en los que dos elementos coinciden en su salida. Si dos elementos caen en la misma hoja en uno de los árboles, suman uno en su afinidad. La afinidad máxima es por tanto el número total de árboles del bosque.

En la figura 4.39 se muestra un sencillo ejemplo de bosque con dos árboles. Cada árbol acaba en 9 hojas. Así, cuando se introduzca un elemento (imagen de la galería) en el nodo raíz cada árbol, *T1* y *T2,* clasificará éste de una

	T1	T2	T3	T4	T5
i1	0	1	1	3	2
i2	0	3	1	2	2
i3	2	3	1	2	2

(a) Respuesta para cada árbol del Bosque.

	i1	i2	i3
i1	5	3	2
i2	3	5	4
i3	2	4	5

(b) Afinidad de los elementos.

Tabla 4.2: Afinidad para tres elementos. Se usa un bosque de 5 árboles.

forma distinta. Se obtendrán dos respuestas, en este caso entre 0 y 8. Posibles salidas son [0, 5] ó [5, 8]. Cuando para dos imágenes, sus salidas coinciden en una posición, ésta suma para su afinidad.

Un sencillo ejemplo se puede ver en la tabla 4.2, en el que se calcula la afinidad entre tres elementos usando un bosque con 5 árboles. Cada elemento es completamente afín consigo mismo, esto es, tiene la afinidad máxima, en este caso 5. Como se puede ver, basta con contar las veces en las que los dos elementos tengan la misma salida en cada uno de los árboles. La matriz de afinidad como se ve es simétrica.

Para trabajar con estos valores, se normalizará entre 0 y 1 la afinidad. Simplemente se divide el valor de afinidad obtenido entre el número de árboles que compone el bosque.

Los parámetros que pueden modificarse para variar el comportamiento del bosque son *rte_ estimators*, y *rte_ leafs*, equivalentes a los usados en el random forest para la expansión visual. El primero define el número de árboles (estimadores) que compondrán el bosque en total. El valor por defecto es de 20 árboles. El segundo define el número mínimo de elementos en una hoja recién creada. El valor por defecto es 1.

4.9.6. Label Spreading/Propagation

Label propagation [Zhu and Ghahramani, 2002] es una técnica de aprendizaje semisupervisado. Esta es una situación en la que el conjunto de entrenamiento consta de unos elementos etiquetados y otros no etiquetados, por lo que se desconoce la clase a la que pertenecen. Label Propagation hace uso de los datos que no están etiquetados para capturar la forma de la distribución de

datos subyacentes y generalizar mejor. Generalmente funciona mejor cuando hay una gran cantidad de datos que no están etiquetados.

El proceso simplificado consiste en asignar de forma aleatoria valores a las muestras no etiquetadas. Se calculan los vecinos para cada elemento en base a la similitud y se calcula cual es la etiqueta mayoritaria en los vecinos. Si es una etiqueta distinta ésta se cambia. Se sigue iterando hasta obtener una convergencia.

Label Propagation y Label Spreading difieren en modificaciones en la matriz de similitud y el efecto de *sujeción* de la distribución de las etiquetas. Este efecto de sujeción permite cambiar hasta cierto punto el valor de las etiquetas ya presentes. Así, Label Propagation utiliza sujeción estricta, esto es, no cambia los valores de elementos ya etiquetados.

Label Propagation utiliza la matriz de similitud construida con los datos de entrada directamente. En Label Spreading minimiza una función de pérdida con el objetivo de reducir el ruido.

A la hora de calcular los vecinos se utilizan los kernels para medir la similitud entre elementos. En este caso se utilizan dos kernels:

- El kernel RBF (Radial Basis Function): Produce un grafo completamente conectado, lo cual se representa en memoria como una matriz densa. Esta matriz puede ser especialmente grande, lo que sumado al coste de realizar multiplicaciones matriciales para cada iteración puede acarrear tiempos de ejecución algo mayores. Se ajusta con el parámetro γ. Ecuación:
$$\exp(-\gamma|x-y|^2)$$
- El kernel Knn (K Nearest Neighbor): Produce una matriz mucho más pequeña y suele ser más rápida. Consiste en tomar los k elementos más similares.

En el caso particular de este proyecto se utilizan las muestras no similares y similares como muestras etiquetadas, y el resto de elementos de la galería serán las muestras no etiquetadas. Al aplicar este método todas las muestras son etiquetadas automáticamente. Así los elementos que queden etiquetados como no similares serán penalizados, y aquellos etiquetados como similares son premiados. Además la implementación de sklearn facilita la probabilidad

de que un elemento pertenezca a una clase u otra, así que se puede jugar con este valor para cambiar la puntuación de manera proporcional.

Para este método se pueden configurar los siguientes parámetros:

- **method**: Define el método que se va a usar, esto es, si es Label Propagation o Label Spreading
- **kernel**: Como se define anteriormente, se aceptan los kernels Knn y RBF.
- **n_neighbors** y **gamma**: n_neighbours define el número de vecinos para Knn y gamma es utilizado por el kernel RBF.
- **alpha**: define el factor de sujeción utilizado para el efecto de sujeción. Regula la facilidad que tiene una etiqueta ya establecida de ser cambiada por el algoritmo.

4.9.7. Método de puntuación

En este apartado se describe con mayor detalle la metodología seguida para establecer la puntuación. En primer lugar, se cuenta con tres factores que afectan a la modificación de la puntuación que se va a realizar:

- Similitud: Compara, usando el método de comparación utilizado en la ordenación inicial, las muestras con las seleccionadas como similares y no similares.
- Afinidad: Resultado obtenido desde la matriz de afinidad, previamente calculado. Su valor depende de la afinidad con las muestras similares y no similares.
- Lab_score: Se denominará así al valor obtenido por la predicción del método Label Propagation/Spreading utilizado.

Todas estas medidas se toman para cada región. Es decir, existirá una similitud, una afinidad y un lab_score para cada región (1 si se usa la imagen completa). Una vez obtenidas para cada región, se promedian para obtener

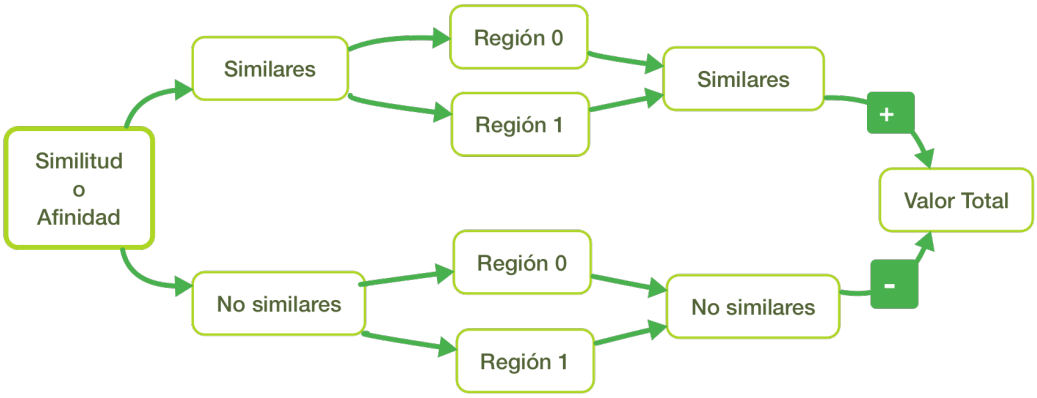


Figura 4.40: Proceso para obtener puntuación para Similitud y Afinidad. Ejemplo con 2 regiones.

un sólo valor. De esta forma, aunque se tengan más muestras en una región que en otra, se consigue equilibrar el resultado.

Se puede ver la metodología seguida para calcular el valor de afinidad y similitud resumida en la figura 4.40. Este proceso se realiza para cada elemento. En ésta se observa cómo se realiza el proceso separado para las muestras similares y no similares. Para cada tipo de muestra, se calcula por separado el valor para cada región. Luego, se promedia, para obtener un valor total. Finalmente, el valor obtenido por las comparaciones con las muestras similares suman, mientras que las no similares restan.

En el caso de Lab_score, el método implementado en sklearn permite obtener no sólo la predicción de la etiqueta, si no un valor probabilístico de pertenencia a cada una de las clases (similares o no similares). Así, se entrena primero Label Propagation/Spreading y posteriormente para cada elemento se obtiene un valor de probabilidad de pertenecer a la clase similares o no similares. Así se resta al valor de probabilidad de pertenecer a la clase similares el valor de probabilidad de pertenecer a la clase no similares. Esto se realiza para cada región, y posteriormente se promedia para cada elemento.

El valor total de Similitud, Afinidad y Lab_score, se combina mediante 3 pesos que el usuario proporciona para cada uno. Esto hace que se obtenga

un único valor total para cada elemento. Con este valor, se puede modificar el valor que tiene el elemento en la ordenación.

Para ello, se define un valor alfa α, que indica la proporción o peso que se le da a la nueva puntuación obtenida con el proceso descrito. Ésta se combinará de alguna forma con el valor que tiene cada elemento de la galería en la ordenación inicial (obtenido con el método de comparación de características) como referencia. En este punto se ofrecen dos opciones:

Proporcional El cambio de puntuación se hace de manera proporcional al valor que tiene cada elemento en la ordenación, esto es:

$$newScore = oldScore + oldScore * \alpha * SAL$$

Donde $newScore$ es el nuevo valor que tendrá el elemento, $oldScore$ es el valor que tenía antes de modificarlo, α es el valor proporcionado por el usuario que regula la importancia que se le da al valor obtenido por el método de reordenación. SAL es la abreviatura de Similitud, Afinidad y Lab_score, promediados con sus respectivos pesos, como se describe en el punto anterior.
Con esta fórmula se asegura que el nuevo valor es una modificación en un cierto porcentaje sobre el valor inicial, haciendo que los cambios sean más controlados

No proporcional En este caso la nueva puntuación se suma con la anterior puntuación:

$$newScore = oldScore + \alpha * SAL$$

Como se ve, en este caso se suma directamente a la anterior puntuación el valor que es obtiene, SAL, sólo controlado con el valor de α.

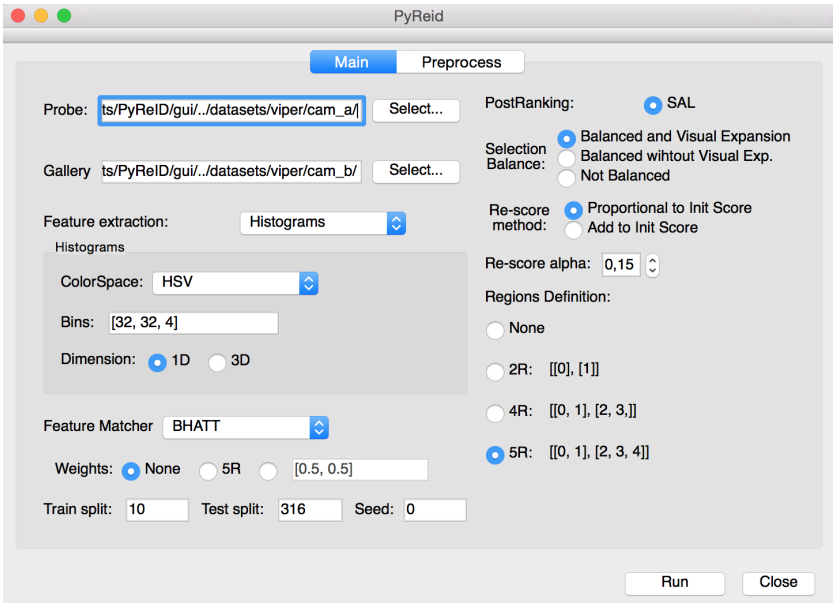
4.9.8. Interfaz gráfica

Para facilitar el uso por parte de un usuario, se decide crear una sencilla interfaz gráfica. Con ella se pueden configurar los parámetros tanto para la ejecución que genera la ordenación inicial como para el proceso de reordenación. En la figura 4.41a se puede ver cómo es ésta. También permite definir

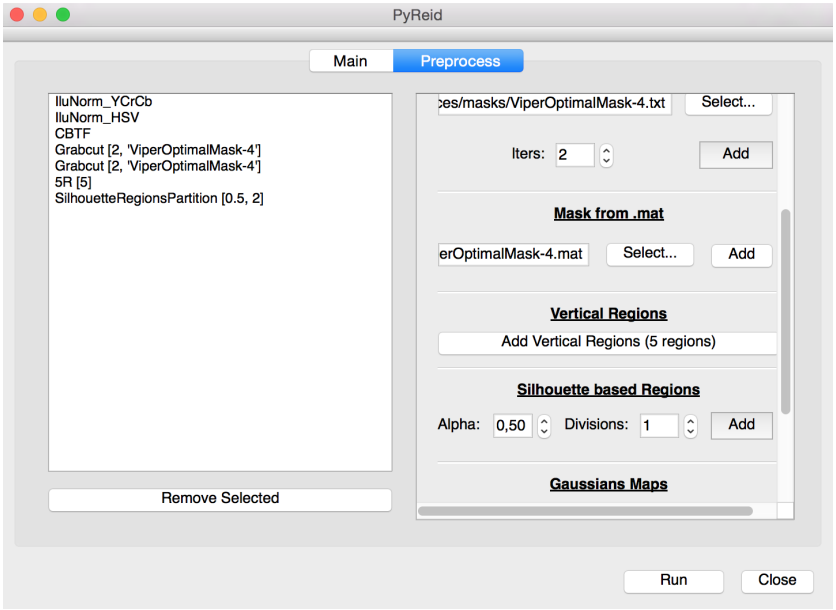
con precisión todos los elementos de preprocesado como se observa en la figura 4.41b.

Cuando se ejecuta, enseña un elemento de la muestra escogido al azar y los elementos de la galería ordenados en base a la ordenación inicial. Ahora el usuario puede seleccionar con el botón izquierdo del ratón las muestras similares y con el derecho las no similares. Cuando se hayan seleccionado, se puede elegir iterar para que se produzca la reordenación. En la figura 4.42 se puede ver como es esta parte de la interfaz.

En la próxima sección, se realizarán múltiples pruebas para comprobar cómo funciona cada método, incluida la reordenación.



(a) Ventana principal.



(b) Ventana de preprocesado.

Figura 4.41: Interfaz de usuario de PyReid.

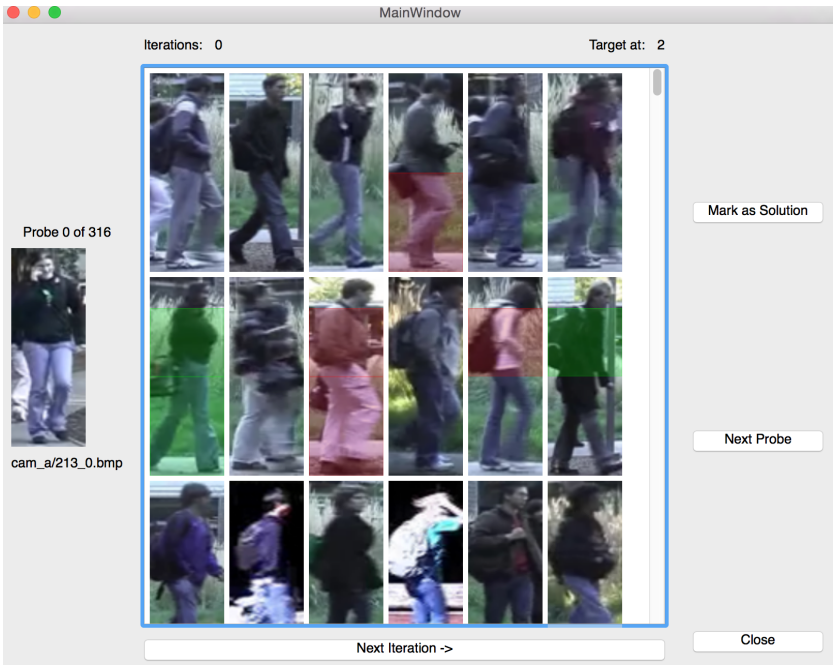


Figura 4.42: Interfaz de PyReid para la reordenación.

Capítulo 5

Pruebas y resultados

A lo largo de este apartado se irán realizando diferentes pruebas con el fin de analizar el comportamiento del sistema. Se irá paso a paso analizando la influencia de cada proceso y las distintas opciones que ofrecen para encontrar las mejores configuraciones. Se analizará cada módulo de preproceso, extracción de características y comparación de características. A continuación se compararán los mejores resultados obtenidos con otros métodos existentes. Por último se estudiará como afecta la reordenación.

5.1. Diseño de experimentos

A continuación se irán realizando pruebas con cada uno de los métodos. Generalmente, los resultados que se obtengan en pruebas previas podrán servir para acotar las pruebas a realizar a continuación. Esto es, se procura siempre utilizar los mejores resultados previos. Por ejemplo, a la hora de evaluar la normalización de la iluminación, se escogen los mejores métodos de comparación y los espacios de color que mejores resultados han obtenido previamente. De esta forma, se acota la cantidad de pruebas a realizar y no se produce una saturación de información irrelevante. Al inicio de cada conjunto de pruebas se especifican las condiciones en las que éstas se llevan a cabo.

Para la evaluación se va a utilizar el AUC, área bajo la curva CMC. Normalmente se mostrará el AUC promediado de varias ejecuciones. Esto se hace

para agrupar varias ejecuciones que cumplen un mismo requisito, como pertenecer al mismo espacio de color. Se empezará explicando el conjunto de datos que se ha utilizado para las pruebas. Posteriormente se explica brevemente el procedimiento seguido para las pruebas.

5.1.1. Conjunto de datos: VIPeR

Para probar todo lo que se va a desarrollar, se necesita tener unos datos con los que experimentar y poder testear. Se usará un conjunto de datos ampliamente utilizado en el campo de la reidentificación de personas.

El conjunto de datos que se usará en este proyecto es de tipo SingleShot, es decir, para cada individuo se tiene una única imagen en la muestra y su correspondiente en la galería. Existe también la posibilidad de Multi-Shot, que consiste en que en algunos de los conjuntos (muestra o galería) se tiene más de una imagen del mismo individuo. Este no ha sido el problema que intenta resolver el proyecto. Sin embargo en la sección futuras mejoras se contempla esta posibilidad (apartado 6.2.1)

El conjunto de datos con el que se trabajará para las pruebas se trata del conocido VIPeR [Gray et al, 2007]. Consiste en imágenes desde dos cámaras diferentes de personas, de las que sólo se dispone de una imagen por cámara (SingleShot). Este conjunto de datos fue generado para poner a prueba el reconocimiento de personas desde distintos ángulos. Por ello cada par de imágenes del mismo individuo están en ángulos distintos, lo que complica la reidentificación. Los ángulos de visión de las personas están cuantizados en 45^{o}, por lo que hay 8 ángulos en total. El conjunto de datos contiene 632 pares de imágenes (1264 en total) tomadas desde dos cámaras distintas. Existen variaciones de ángulos como se ha explicado, pero también de condiciones de iluminación. Las imágenes están escaladas a 128x48 píxeles. Se trata de uno de los conjuntos de datos más complicados para la reidentificación automática. En la figura 5.1 se pueden ver algunas muestras de individuos de este conjunto de datos desde las dos cámaras.

Para realizar las pruebas se utilizará un subconjunto de individuos ya prefijado. Éste es el mismo usado en [Farenzena et al, 2010] y es comúnmente utilizado en el campo de la reidentificación para que los distintos métodos

Figura 5.1: Imágenes de ejemplo del conjunto de datos Viper [Gray et al, 2007].

sean comparables, pues utilizan exactamente las mismas muestras. Se trata de un *crossvalidation* que se realiza 10 veces, utilizando 316 muestras en la muestra (cámara a) y sus equivalentes en la galería (cámara b). No proporcionan elementos para entrenamiento. A este conjunto se le denominará *File.*

5.2. Métodos de comparación de histogramas

Inicialmente se comprobará cual de los métodos de comparación con los que se puede trabajar ofrece un mejor resultado. En [Marín Reyes, 2015] se realiza un estudio sobre los mejores métodos de comparación para el problema de la reidentificación de personas. En él se concluye que los mejores métodos de comparación son Bhattacharyya, Chi Cuadrado e Intersección. Estos métodos serán usados en esta prueba, además de Correlación y distancia Euclídea.

Se realizará esta primera prueba con la comparación directa, sin usar segmentación ni partición de regiones, algo que se realizará en experimentos posteriores. Se usarán todos los espacios de color admitidos por PyReID. En el caso de RGB se usan los mismos bins para cada canal y sus valores irán de 4 a 32 con saltos de 4 (4, 8, 12...). En el caso de HSV se usan 8, 16 y 32 bins

	RGB	HSV	IIP	Media
Bhattacharyya	60.32	**82.35**	73.87	**72.18**
Chi Cuadrado	60.26	77.48	73.17	70.30
Correlación	**61.09**	79.10	75.21	71.80
Euclídea	59.09	80.59	**75.36**	71.68
Intersección	58.81	80.44	73.65	70.97
Media	59.91	**79.99**	74.25	

Tabla 5.1: AUC para métodos de comparación de histogramas en los distintos espacios de color.

para H y S (mismos bins para cada canal) y 0 para V, representando el 0 que ese canal es ignorado. Esto se hace así porque el canal V representa el brillo o luminancia e interesa descartar el brillo para evitar un exceso de sensibilidad en los cambios de iluminación. En el caso de IIP se desconoce su comportamiento así que se probarán las mismas combinaciones que para RGB. En la sección 5.3 se profundiza más en el número de bins de cada espacio de color. En este punto lo importante son los métodos de comparación.

En la tabla 5.1 se puede ver como se comporta cada método con cada espacio de color. Los resultados que se se muestran son el AUC para cada espacio de color. Como se puede observar el método con mejores resultados es Bhattacharyya. Sin embargo llama la atención cómo cada espacio de color tiene un método que se comporta mejor. En el caso del RGB el método que mejor se comporta es Correlación y para el caso de IIP es la distancia Euclídea. Para HSV Bhattacharyya es el mejor con cierta diferencia.

En la siguiente sección se estudia más a fondo los espacios de color, como afecta el número de bins y los métodos de comparación que mejor han resultado.

5.3. Espacios de color y número de bins

En PyReID se permiten de forma nativa 3 espacios de color. Se analiza en este apartado cómo se comportan y cómo afectan a la reidentificación. Para las pruebas no se utilizará ningún preprocesado como segmentación o partición en regiones. Los métodos de comparación utilizados serán Bhattacharyya, Correlación y la distancia Euclídea.

En todas las pruebas se limitarán los bins entre los rangos 0 y 32 con saltos de 4 u 8. Sin embargo en PyReID se tiene flexibilidad a la hora de definir los bins en los histogramas. No sólo se hará uso de la forma común de analizar los histogramas, usando los bins de forma homogénea para cada canal, sino que se utilizarán combinaciones desiguales de éstos. Por ejemplo, se podrán definir bins del tipo: «16, 32, 4» ó «0, 4, 28» (representando el 0 que ese canal es ignorado). Con ello se puede realizar un análisis más profundo en ciertos casos y se verá si esto tiene alguna repercusión.

RGB

El espacio de color RGB, explicado en la sección 4.2.2 consta de 3 componentes. Se utilizan los mismos bins para cada canal. Por las características de RGB no se va a realizar un estudio de la influencia de cada canal por separado, pues es un espacio de color aditivo donde cada canal representa un porcentaje de un color. El hecho de que el rojo o el azul influencie más los resultados, puede simplemente indicar que en este conjunto de datos concreto se usen más o menos prendas de color rojo. Esto no sería trasladable a otros conjuntos de datos.

En la tabla 5.2 se puede observar el comportamiento de este espacio de color para cada tamaño de bin y los tres métodos de comparación usados. En este caso parece mejor resultado usar el método Correlación con 32 bins.

HSV

Este espacio de color es el más común y parece demostrar buenos resultados en el ámbito de la reidentificación de personas [Marín Reyes, 2015]. Se encuentra descrito en la sección 4.2.3. Lo que se suele producir con este espacio de color es que se utilizan las dos primeras componentes y se desprecia la última (o se usa un número bajo de bins). Esto se debe a que el último parámetro está íntimamente ligado a la iluminación. factor que afecta negativamente en la reidentificación.

bins	BHATT	CORRE	EUCLI	Media
4	60.25	60.50	**59.86**	60.20
8	60.07	59.84	59.12	59.67
12	**60.48**	60.68	59.16	60.11
16	60.23	61.12	59.16	60.17
20	60.33	61.47	59.05	60.28
24	60.43	61.39	58.79	60.20
28	60.30	61.69	58.74	60.24
32	60.45	**62.04**	58.83	**60.44**
Media	60.32	**61.09**	59.09	

Tabla 5.2: AUC para bins iguales en cada canal en RGB frente a los distintos métodos de comparación.

Bhattacharyya		**Euclídea**		**Correlación**	
bins	AUC	bins	AUC	bins	AUC
[32, 8, 0]	**83.25**	[16, 8, 0]	81.88	[16, 8, 32]	81.36
[24, 8, 0]	83.07	[16, 8, 32]	81.75	[16, 8, 0]	81.28
[16, 8, 0]	82.98	[8, 8, 32]	81.72	[24, 8, 32]	81.19
[32, 16, 0]	82.57	[8, 8, 0]	81.70	[24, 8, 0]	81.14
[16, 16, 0]	82.50	[16, 8, 24]	81.61	[16, 8, 24]	81.00

Tabla 5.3: 5 Mejores resultados para HSV.

Para comprobar esto, se realizará una extensa prueba en este espacio de color. Se comprobarán los resultados con todas las combinaciones posibles de bins (entre 0 y 32 y con saltos de 8) para ver cuales son los mejores resultados.

Estos resultados se resumen en la tabla 5.3. Se muestran los 5 mejores resultados usando las tres distancias escogidas. Claramente Bhattacharyya es el mejor método para HSV. Con esta distancia se consigue aproximadamente 1.5 puntos más que el siguiente mejor método. Se confirman las sospechas y en los 5 mejores resultados el canal V se encuentra a 0. H oscila entre 32 y 16 y S entre 8 y 16. Si se miran en los otros métodos de comparación llama mucho la atención cómo aparece el canal V y éste varia entre valores extremos (0 y 32).

A continuación se realiza un estudio más exhaustivo del comportamiento de cada canal, estudiando la influencia de cada componente (canal) de forma

bins	H	S	V
0	67.40	75.01	**79.88**
8	79.65	**78.67**	77.25
16	79.93	78.27	77.06
24	80.20	78.09	76.97
32	**80.40**	77.84	76.91

Tabla 5.4: Influencia de los canales HSV con Bhattacharyya.

independiente. Para ello, se mantienen fijos los bins del canal a estudiar y se varían los otros dos componentes en todo su rango. Finalmente se calcula la media del AUC de todas las muestras. Así se podrá tener una idea de cómo de influyente es cada canal con unos bins concretos independientemente del resto de canales. Esta prueba se realiza con el método de comparación Bhattacharyya, que como se ha visto muestra mejores resultados para este espacio de color.

En la tabla 5.4 se pueden sacar rápidas conclusiones. Efectivamente, el canal V muestra el comportamiento esperado. El mejor resultado se da con 0 bins, empeorando el resultado cuanto más alto es el valor. Así mismo parece que el canal H tiene ligeramente más relevancia que el canal S, pues los mejores bins se encuentran entre 16, 24 y 32 y la media en el AUC es casi 2 puntos superior. Para el canal S lo mejor parece ser que oscile entre los 8 y 16 bins.

IIP

Este espacio de color, introducido de forma novedosa aplicado a la reidentificación de personas en este proyecto, presenta también tres componentes. Se realizará un proceso similar al realizado con HSV, aunque no se parte con información previa de cómo puede comportarse cada componente en el ámbito de la reidentificación de personas.

En la tabla 5.1 se veía como para este espacio de color el método Bhattacharyya no parecía el más adecuado. Sin embargo en la tabla 5.5 se observa que los 3 mejores resultados se obtienen para esta distancia. Además llama la atención que el segundo canal es en estos resultados el único usado, siendo los otros dos ignorados. En los otros métodos de comparación los resultados son más dispares. Esto parece mostrar que el usar el primer y tercer canal

Bhattacharyya		**Euclídea**		**Correlación**	
bins	AUC	bins	AUC	bins	AUC
[0, 32, 0]	**80.52**	[0, 16, 0]	78.91	[0, 16, 0]	79.58
[0, 24, 0]	79.98	[32, 16, 0]	78.18	[0, 16, 32]	78.70
[0, 16, 0]	79.74	[0, 16, 32]	78.09	[32, 16, 0]	78.63
[8, 32, 0]	79.32	[24, 16, 0]	78.03	[0, 16, 24]	78.37
[0, 32, 8]	78.99	[0, 16, 24]	77.96	[24, 16, 0]	78.36

Tabla 5.5: 5 Mejores resultados para IIP.

bins	0	1	2
0	**74.01**	57.65	**74.42**
8	71.79	73.28	71.68
16	71.13	75.14	70.99
24	70.99	75.71	70.92
32	70.95	**76.43**	70.87

Tabla 5.6: Influencia de los canales IIP.

no afecta tanto como a Bhattacharyya, en cuyo caso empeora el resultado cuando más altos son estos dos canales.

Para confirmarlo se realizar un estudio de la influencia de cada componente. Al igual que se realiza con HSV, se fijará una de las componentes a un valor, se calcula el AUC para todas las combinaciones variando las otras dos componentes y se obtendrá la media. Este proceso se repite para cada valor de bins de cada componente. Como se ha visto que los mejores resultados son obtenidos con Bhattacharyya se hará el estudio con este método de comparación.

En la tabla 5.6 se confirma lo que parecían indicar los mejores resultados. Claramente parece que el canal más interesante en IIP es el segundo. Lo más destacable sin duda es que el no usar el segundo canal empeora mucho los resultados (0 bins en el canal 1), cuyo AUC cae a 57.65.

	Sin Norm.	HSV	YCrCb	HSV + YCrCb	YCrCb + HSV
HSV	82.79	83.07	83.25	**83.87**	83.35
IIP	80.52	80.31	79.09	**81.11**	79.34
Media	81.65	81.69	81.17	**82.49**	81.34

Tabla 5.7: Efectos de la Normalización de la Iluminación en el AUC medio.

5.4. Normalización de la iluminación

Como se describe en la sección 4.5.5, uno de los elementos de preprocesado es la normalización de la iluminación. En este proyecto se realiza la normalización en dos espacios de color diferentes, HSV y YCrCb, para los canales V e Y respectivamente. Como el sistema lo permite, se pueden perfectamente encadenar las dos normalizaciones, una tras la otra. Se verá si esto consigue mejorar los resultados y si cambia el hecho de aplicarlas en un orden concreto. Así mismo se compara con la opción de no realizar normalización de la iluminación en absoluto. Hay que recordar que a la hora de realizar la normalización se producen dos conversiones (normalmente) de espacio de color. Primero se transforma al espacio en que se va a normalizar la imagen y luego se vuelve a transformar al espacio de color original.

Para las pruebas se utiliza el método de comparación Bhattacharyya. No se utilizará preprocesado, más allá de la normalización de la iluminación que se está evaluando. Viendo los resultados de las pruebas anteriores se limitan las pruebas a los parámetros que mejor resultado han dado. Si no se hiciera esta selección la cantidad de pruebas a realizar sería muy grande. Concretamente las pruebas se realizan con los siguientes parámetros:

- Métodos de comparación de histogramas: Bhattacharyya.
- Espacios de color:
  - HSV: bins H: 32, 24, 16; bins S: 16, 8; bins V: 0
  - IIP: bins: [0, 32, 0] respectivamente para cada canal.

En la tabla 5.7 se observan los resultados, separados para cada conjunto, de aplicar las diferentes combinaciones de la normalización de la iluminación (Sin normalización, Normalización con HSV, con YCrCb y las combinaciones

entre ambas). Para cada espacio de color se calcula el promedio. Además se añade la fila *Media,* que es el promedio de los espacios de color con los que se trabaja.

En el caso de HSV cualquiera de las normalizaciones mejora los resultados. La mejor es la combinación de la normalización HSV con YCrCb. Se observa que es importante el orden en el que se realicen las normalizaciones de la iluminación. Para el caso de IIP el mejor resultado se da con la misma combinación que para HSV, pero llama la atención cómo cualquier otra normalización de la iluminación produce peores resultados que no usar la normalización.

5.5. Segmentación

En esta sección se realizarán dos pruebas diferenciadas. Por un lado, se buscará una máscara inicial para el método GrabCut que obtenga los mejores resultados posibles. Por otro lado, se probarán las distintas segmentaciones, entre ellas las obtenidas en el paso anterior, para comprobar cómo se comportan con cada espacio de color.

5.5.1. Máscara inicial para Grabcut

Para la utilización del método Grabcut se necesita proveer una máscara inicial que indique y delimite la imagen. En la modalidad interactiva de GrabCut un usuario marca las imágenes y corrige la segmentación. En este caso, se parte de una máscara inicial y se espera que se realice una segmentación lo más optima posible. La máscara inicial será la misma para todas las imágenes, por lo que es un punto clave para que la segmentación obtenida sea fiable, puesto que la pose puede diferir entre las distintas imágenes.

De forma inicial se pueden marcar cuatro tipo de zonas: fondo, probable fondo, objeto (persona en este caso) y probable objeto. Con esta separación inicial, el algoritmo intentará separar la zona que nos interesa, la persona, del fondo. Para obtener esta máscara, se han utilizado las siguientes aproximaciones.

Figura 5.2: Máscara manual. Colores: fondo (azul), persona (celeste), fondo probable (amarillo), persona probable (rojo).

5.5.1.1. Máscara manual

La primera aproximación consiste en realizar esta máscara inicial a mano. Tras observar múltiples imágenes se intuye que, por lo general, los individuos están centrados y las esquinas superiores suelen ser pertenecientes al fondo. Así mismo los laterales son una buena zona para marcar como fondo. Esta primera aproximación de máscara manual se puede observar en la figura5.2.

Se marcan las esquinas y una pequeña zona lateral como fondo (azul), y en el centro de la imagen se marca desde la altura estimada de la cintura hasta la cabeza como persona (celeste). A su lado se añaden dos zonas de persona probable (rojo). El resto, es considerado como fondo probable (amarillo). Es muy difícil acotar las zonas de forma más precisa, pues hay una gran variedad en la disposición de las imágenes, lo que lleva inicialmente a una distribución más conservadora de este tipo. Esta máscara será denominada *ManualMask*.

5.5.1.2. Máscara promedio (OptimalMask)

Como primera aproximación la máscara manual descrita en el punto anterior no parece mala idea. Sin embargo, para obtener mejores resultados se propone utilizar un método más fiable que la simple observación. La idea consiste en

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

Figura 5.3: Segmentación ideal para varias imágenes.

recoger una serie de imágenes de forma aleatoria, calcular manualmente los limites (segmentación ideal) y luego hallar una media de entre todas estas.

En el conjunto de datos Viper se han escogido 40 imágenes de forma aleatoria de los dos conjuntos disponibles. Una muestra de cómo se realiza el marcado se observa en la figura 5.3

Con estas máscaras marcadas a mano se realiza un promedio muy simple. Para cada píxel, se mira en cada imagen y se suma +1 cada vez que ese punto pertenezca al primer plano o se resta -1 cada vez que pertenece al fondo. Se obtiene finalmente una matriz de puntos cuyo valor oscila en el rango de imágenes utilizadas (-40 y 40). Con este resultado se puede discretizar entre los cuatro valores posibles para Grabcut: fondo, fondo probable, probable persona, persona. El resultado obtenido, y la versión discretizada, se muestran en la figura 5.4. Se puede observar claramente la silueta de la persona. La parte más difusa es la de las piernas, ya que entre imágenes varia mucho la posición en la que son capturados los individuos. A esta máscara se le denomina *OptimalMask*.

5.5.1.3. Alteraciones a la máscara promedio (OptimalMaskMod)

Si bien la máscara promedio aparentemente parece la idónea, se observa que en la parte inferior es muy difusa y parece que el método de segmentación recorta bastante la parte de las piernas. Una alternativa que se propone, es ser más permisivos con esa zona. Se prueba con otra máscara en la que se realizan las siguientes modificaciones:

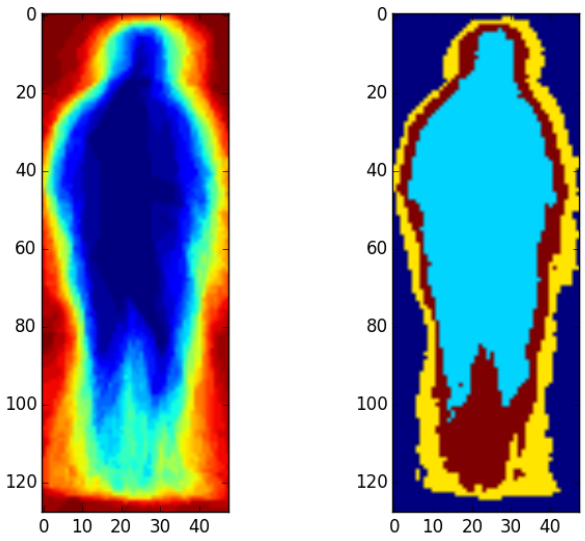


Figura 5.4: Máscara promedio (OptimalMask). A la izquierda versión sin discretizar, derecha versión discretizada.

- Por debajo de la altura de la cintura, se considerarán todos los puntos probablemente persona como persona.
- Los laterales se cortan ligeramente hacia dentro los puntos considerados persona, trasformándolos en probablemente persona.
- La región de la cabeza se convierte en probablemente persona.

El resultado puede observarse en la figura 5.5. A la izquierda se muestra la máscara promedio original y a la derecha las modificaciones realizadas. A esta máscara se le denominará *OptimalMaskMod*.

5.5.1.4. Resultados de las pruebas

En este punto se va a comprobar el comportamiento de las máscaras que se han presentado en los apartados anteriores. Para ello, se han escogido otras imágenes del conjunto de datos, se ha realizado el marcado de la persona y fondo a mano (segmentación ideal), se calcula la máscara resultante aplicando

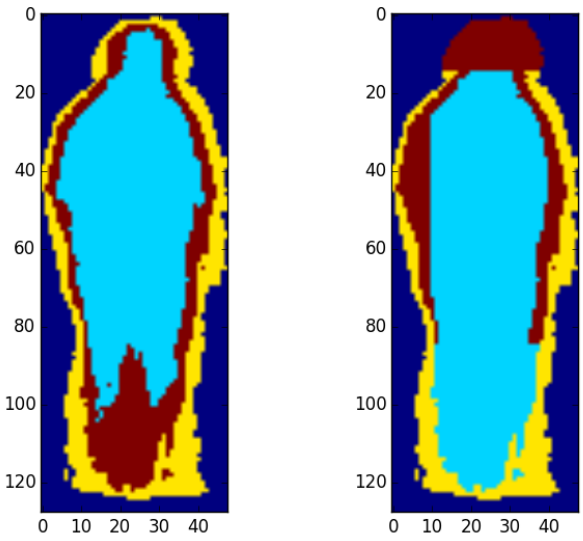


Figura 5.5: Mascara promedio (izquierda) junto a máscara promedio modificada (OptimalMaskMod).

GrabCut con las distintas máscaras a probar y se compara con el marcado realizado a mano. Así se podrán obtener unas estadísticas que indiquen cuál parece ser la mejor máscara para utilizar con GrabCut en este conjunto de datos.

Se pueden observar los resultados en la tabla 5.8. Se muestra tanto el ratio de acierto con el fondo como con la persona. De esta forma se puede detectar a qué se debe el fallo a la hora de realizar la segmentación. Claramente parece que la Máscara manual es la peor, pues acierta menos en fondo y en persona. La máscara promedio parece que muestra los mejores resultados, aunque la máscara modificada mejora en los aciertos del fondo, a costa de disminuir los aciertos en persona (esto significa que considera como persona píxeles que pertenecen al fondo). En el siguiente apartado se estudiará cómo se comportan en la reidentificación.

Num. iteraciones	Aciertos Fondo %	Aciertos Persona %
1	73,19	77,46
2	74,54	73,86
3	74,77	73,19
4	75,11	72,65

(a) Máscara Manual.

Num. iteraciones	Aciertos Fondo %	Aciertos Persona %
1	78,78	86,47
2	78,79	86,53
3	78,77	86,54
4	78,78	86,53

(b) Máscara Promedio.

Num. iteraciones	Aciertos Fondo %	Aciertos Persona %
1	80,73	82,84
2	80,81	82,81
3	80,47	82,69
4	80,56	82,82

(c) Máscara Promedio Modificada.

Tabla 5.8: Resultados Máscaras para el conjunto de datos ViPER.

	Sin Seg	SCA	ManualMask	OptimalMask	OptimalMaskMod
HSV	82.79	83.66	83.25	**84.62**	84.21
IIP	80.52	81.09	80.37	**81.58**	81.09
Media	81.65	82.38	81.81	**83.10**	82.65

Tabla 5.9: AUC con distintos métodos de Segmentación.

5.5.2. Pruebas métodos de segmentación

En esta sección se describe cómo se comportan los diferentes métodos de segmentación. Las condiciones del experimento serán las mismas que hasta ahora en cuanto a los espacios de color y método de comparación escogido. No se utilizará ningún preprocesado más allá de la propia segmentación que va a ser probada. Condiciones:

- Métodos de comparación de histogramas: Bhattacharyya.
- Espacios de color:
  - HSV: bins H: 32, 24, 16; bins S: 16, 8; bins V: 0
  - IIP: bins: [0, 32, 0] respectivamente para cada canal.
- Sin preprocesado.

En primer lugar se va a analizar el resultado de usar diferentes métodos para la segmentación. Concretamente se usarán las máscaras obtenidas por el método SCA (apartado 4.5.1.3) y el método GrabCut usando las diferentes máscaras descritas en 5.5.1. Cabe mencionar que se fija el número de iteraciones de GrabCut a 2, pues no se aprecian diferencias notables al variar este parámetro.

En la tabla 5.9 se pueden ver los resultados para cada espacio de color. Se observa que la máscara con mejores resultados es la OptimalMask. La máscara ManualMask apenas consigue mejorar los resultados de no usar segmentación y es la peor, algo que ya se intuía, llegando incluso a empeorar los resultados en el caso de IIP. SCA y OptimalMaskMod ofrecen a su vez buenos resultados. Esto es especialmente interesante dado que SCA requiere de un proceso costoso. Con Grabcut se han mejorado los resultados y la eficiencia computacional, pues requiere de menos cálculo. GrabCut sin embargo requiere de

una máscara inicial. En el caso de este conjunto de datos es bastante intuitiva y sencilla de obtener, pero en otros conjunto de datos puede volverse una alternativa no viable.

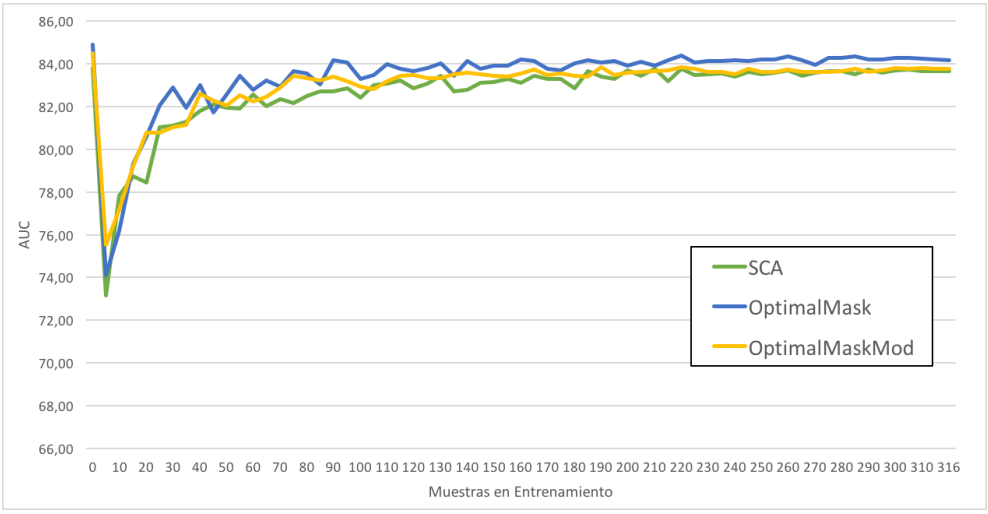
5.6. Función de transferencia de brillo (BTF)

Una vez se han probado los distintos métodos de segmentación se empezará a probar los preprocesados que necesiten una segmentación previa. Éste es el caso de la función de transferencia de brillo. Se probará con el método acumulativo (CBTF) y el método promedio (MBTF). Las condiciones de los experimentos de esta sección son las siguientes:

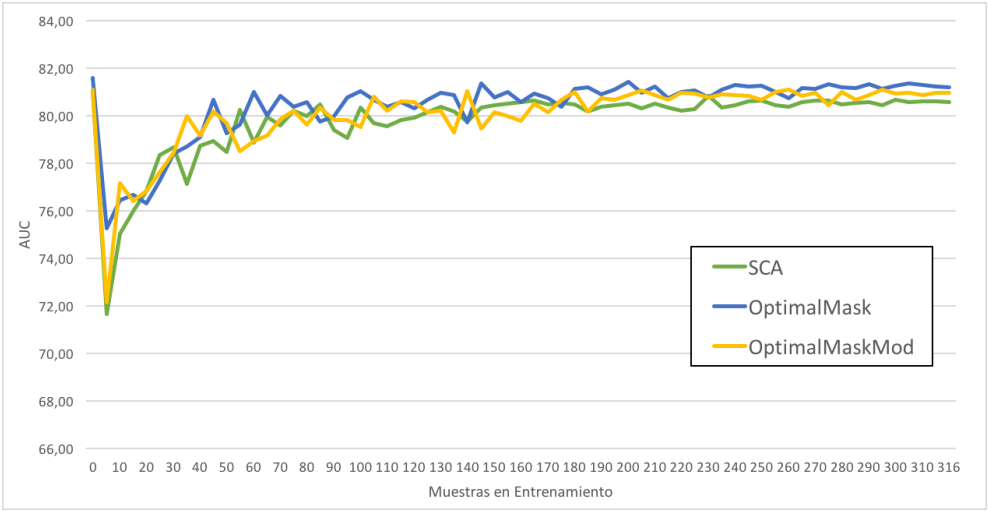
- Método de comparación de histogramas: Bhattacharyya.
- Espacios de color:
  - HSV: bins [32, 16, 0] (1 combinación)
  - IIP: bins: [0, 32, 0] respectivamente para cada canal. (1 combinación)
- Preprocesado:
  - Segmentación: GrabCut (OptimalMask y OptimalMaskMod) y SCA.

Se ha decidido simplificar la cantidad de bins, pues aporta poco probar con tantas combinaciones distintas y apenas afecta al resultado. Teniendo en cuenta que cada vez se hacen mas pruebas, se han de mantener las combinaciones en las mínimas posibles para que no se produzca una explosión combinatoria.

Para ambos métodos se seguirá la misma metodología. Se cambiará la cantidad de individuos que se usan en el conjunto de entrenamiento y se verá cómo afecta esto a los resultados. El rango de pruebas irá desde 0 elementos a la mitad (316) con saltos de 5 en 5. Como en todas las pruebas, se usarán 316 elementos en el conjunto de test. En cada ejecución se escogen los elementos de entrenamiento de forma completamente aleatoria.

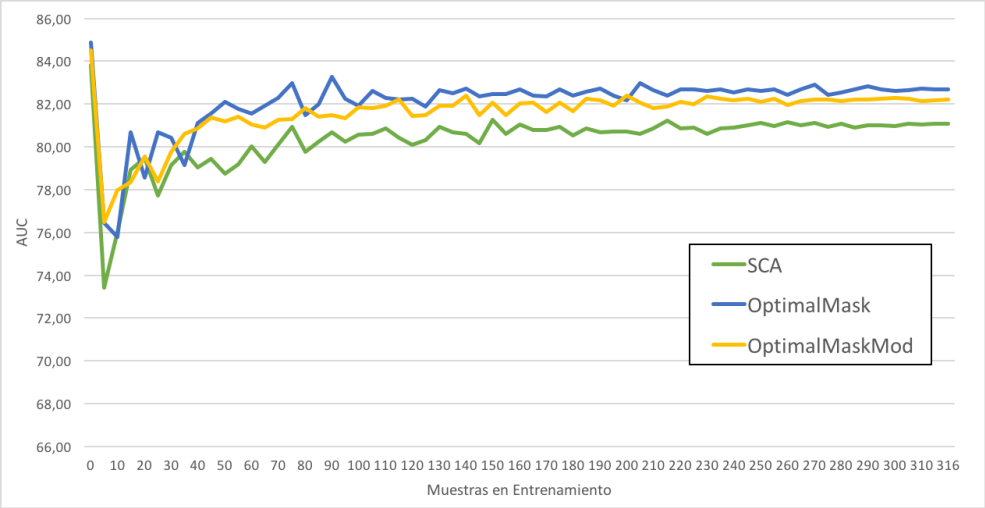


(a) CBTF HSV.

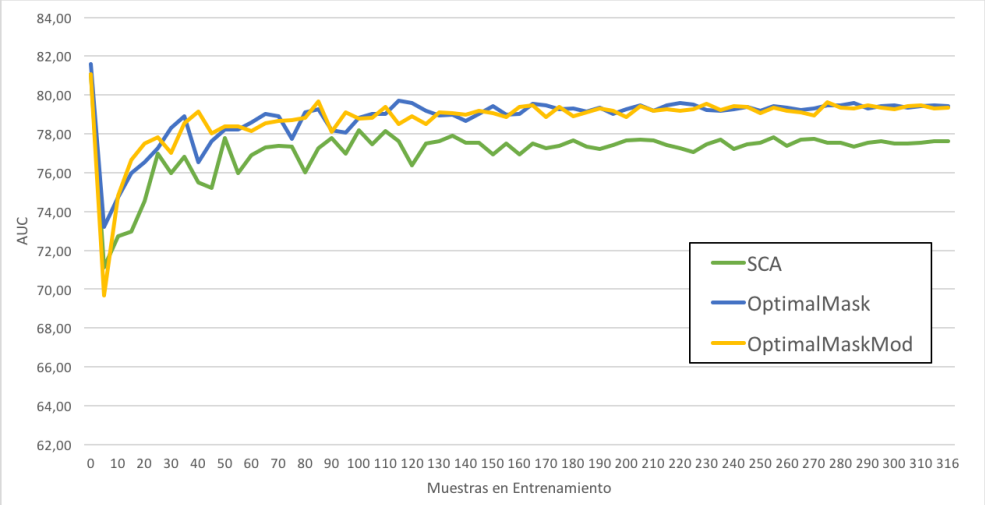


(b) CBTF IIP.

Figura 5.6: Comportamiento de CBTF en base al número de elementos en el conjunto de entrenamiento y la segmentación utilizada.



(a) MBTF HSV.



(b) MBTF IIP.

Figura 5.7: Comportamiento de MBTF en base al número de elementos en el conjunto de entrenamiento y la segmentación utilizada.

En la figura 5.6 se observa el comportamiento de CBTF. Inicialmente se produce una gran caída. Parece que no utilizar CBTF es mejor que usarlo con valores de entrenamiento bajo. Sin embargo se ve como cuantos más elementos son usados para el entrenamiento mejor es el resultado. Con MBTF , figura 5.7 se observa un comportamiento similar. Esto se produce para ambos espacios de color. En cualquier caso, el mejor comportamiento se produce cuando no se usa este método, aunque se usen todos los elementos restantes como entrenamiento.

En [Wang et al, 2013] comprueban cómo en el caso del conjunto de datos VIPeR, al igual que ocurre en nuestras pruebas, los resultados son peores al usar el método BTF. Piensan que puede deberse al hecho de que no existen grandes variaciones en la iluminación y que en este conjunto de datos la mayor dificultad proviene de los cambios en la pose, por lo que BTF puede causar más problemas que ayuda.

5.7. Partición en regiones

A continuación se probará cómo se comporta la partición en regiones. Como se describe en la sección 4.5.2 existe la opción de definir las secciones de la imagen manualmente y también se puede usar el método de *Partición de la Silueta.* Para estos experimentos se usa la división en 6 regiones ignorando la parte superior (la cabeza) y que se denominará *5R.* También se probará el método de la partición de la silueta (*SilPart*) para el cual se consideran dos versiones: la normal (2 secciones) y la subdividida, que consiste en dividir cada región de las anteriores en 2 secciones más, obteniendo 4 en total (SilPart2). El método de la *Partición de Silueta* depende de la segmentación, pues usa ésta para hacer los cálculos de la partición ideal, así que se probará con las distintas máscaras para ver los mejores resultados. Además se ha de tener en cuenta que se se puede asignar un peso a cada región a la hora de hacer la comparación de características. Esto puede variar muchos los resultados y será estudiado en este apartado.

Las condiciones de los experimentos de esta sección son las siguientes, salvo que se especifique lo contrario:

- Método de comparación de histogramas: Bhattacharyya.

Máscaras	Sin Part.	5R	SilPart	SilPart2	Media
Sin Seg.	82.79	84.16	-	-	83.47
SCA	83.66	87.65	86.70	87.58	86.40
OptimalMask	84.62	88.45	87.43	88.38	87.22
OptimalMaskMod	84.21	**88.86**	87.84	88.85	**87.44**
Media con Seg.	84.17	**88.32**	87.32	88.27	

(a) Espacio de color HSV.

Máscaras	Sin Part.	5R	SilPart	SilPart2	Media
Sin Seg.	80.52	83.18	-	-	81.85
SCA	81.09	86.24	84.61	85.57	84.38
OptimalMask	81.58	86.36	84.57	85.88	84.60
OptimalMaskMod	81.09	**86.92**	85.51	86.64	**85.04**
Media con Seg.	81.26	**86.50**	84.89	86.03	

(b) Espacio de color IIP.

Tabla 5.10: Partición en regiones con distintas máscaras y espacios de color. Sin utilizar pesos en la comparación de características.

- Espacios de color:
  - HSV: bins [32, 16, 0] (1 combinación)
  - IIP: bins: [0, 32, 0] respectivamente para cada canal. (1 combinación)
- Preprocesado:
  - Segmentación: GrabCut (OptimalMask y OptimalMaskMod) y SCA.

En la primera prueba se comprobará el comportamiento de las 5 regiones (5R) con y sin segmentación, y la partición de silueta (SilPart y SIlPart2) con segmentación (recordar que este método requiere de segmentación, por lo que no puede ser usado sin ella). En esta prueba no se usará ningún peso para la comparación de características.

En la tabla 5.10 se resume el resultado para ambos espacios de color. En total se han necesitado 168 ejecuciones. En ambos el comportamiento es similar,

donde la máscara OptimalMaskMod supera ligeramente a OptimalMask. Esto puede deberse a que la OptimalMaskMod, como se menciona en la sección 4.5.1, produce resultados con un mayor «hueco» en la parte inferior de la imagen. Esto hace que aumente la cantidad de fondo que se considera parte de la imagen, pero parece que con la partición en regiones esto puede resultar beneficioso.

Nuevamente **HSV** ofrece el mejor resultado frente a IIP (unos 2 puntos porcentuales mejor). Esto se corresponde con los resultados previos. Así mismo la mejor partición parece ser 5R, seguido muy de cerca por SilPart2. Cabe destacar que la mejora de usar partición de regiones a no usarla en el caso de IIP mejora en casi 7 puntos. En HSV la mejora es de unos 4 a 5 puntos.

La siguiente prueba consiste en jugar con los pesos en la comparación de características. Hasta ahora sólo se ha probado sin usar peso alguno, por lo que se trata todo el conjunto de histogramas como uno sólo, concatenados y comparándolos directamente. Como se describió en la sección 4.5.2 se pueden asignar pesos cuando se realiza la comparación a cada una de las regiones (o grupos de regiones). De esta forma se da más importancia o menos a las regiones de la partición realizada, comparando éstas por separado.

En las siguientes secciones se estudiará el comportamiento de la variación de pesos para los tres tipos de particiones. La condición que han de cumplir todas las combinaciones de pesos es la mostrada en la ecuación 5.1:

$$\sum_{x=1}^{i} w_x = 1.0 \tag{5.1}$$

En ésta se refleja que la sumatoria de todos los pesos, siendo i el total de pesos a asignar (5 para 5R y 4 para SilPart2) ha de sumar 1.

5.7.1. 5R

Para 5R se han de usar 5 pesos distintos para cada región. En este caso se va a probar con múltiples combinaciones de pesos que cumplan las condiciones mostradas en la ecuación 5.1. Así mismo para limitar los posibles valores, se utiliza el siguiente rango, definido en la ecuación 5.2:

$$0 < w_x < 0.3 \;\; x\epsilon[1..5] \tag{5.2}$$

Pesos \ Regiones	0	1	2	3	4
0.0	86.13	86.79	86.90	88.10	88.85
0.05	86.83	87.21	87.37	88.21	88.91
0.1	87.42	87.59	87.66	**88.25**	**88.98**
0.15	87.83	87.82	87.91	88.20	88.51
0.2	88.14	88.02	88.01	87.89	88.13
0.25	**88.22**	**88.08**	**88.07**	87.78	87.58
0.3	88.09	88.01	87.90	87.32	86.91

Tabla 5.11: Influencia de los pesos en 5R. Espacio de color HSV y segmentación OptimalMaskMod.

Esto es, el rango en el que se probará cada peso es entre 0 y 0.3. Concretamente se probará con los valores siguientes:

0 , 0.05 , 0.1 , 0.15 , 0.2 , 0.25 , 0.3

Dada la cantidad de posibilidades, se limitan las pruebas al espacio de color HSV y a la segmentación OptimalMaskMod. En la tabla 5.11 se muestran los resultados obtenidos en esta prueba. Para obtenerla se han realizado 753 ejecuciones.

Se observa claramente que las 3 primeras regiones (regiones superiores) tienen más peso. Esto significa que son más importantes en la reidentificación. Se puede confirmar esto al comprobar los 5 mejores resultados que se obtienen de todas las ejecuciones. Estos se muestran en la tabla 5.12.

5.7.2. SilPart

En SilPart se pueden usar 2 pesos en total, uno para la parte superior y otro para la parte inferior. El peso que tendría la parte inferior es complementario (en total los pesos de ambas partes suman 1). Se limitan los posibles valores en el siguiente rango (ecuación 5.3):

$$\begin{cases} 0.3 < w_1 < 0.7 \\ w_2 = 1 - w_1 \end{cases} \tag{5.3}$$

Regiones					
0	1	2	3	4	AUC
0.25	0.25	0.3	0.1	0.1	89.33
0.25	0.2	0.3	0.15	0.1	89.30
0.25	0.25	0.25	0.15	0.1	89.30
0.25	0.25	0.3	0.15	0.05	89.29
0.2	0.3	0.25	0.15	0.1	89.27

Tabla 5.12: Mejores resultados para 5R con pesos. Espacio de color HSV y segmentación OptimalMaskMod.

Donde w_1 representa el peso de la parte superior y w_2 el peso de la parte inferior. Se discretiza el rango en saltos de 0.05 (0.3; 0.35; 0.4; ...).

En la figura 5.8 se muestra cómo varía el AUC según el peso que se le da a la parte superior. Se calcula para HSV e IIP así como las 3 mejores máscaras y el valor promedio de éstas. El número total de ejecuciones es de 54.

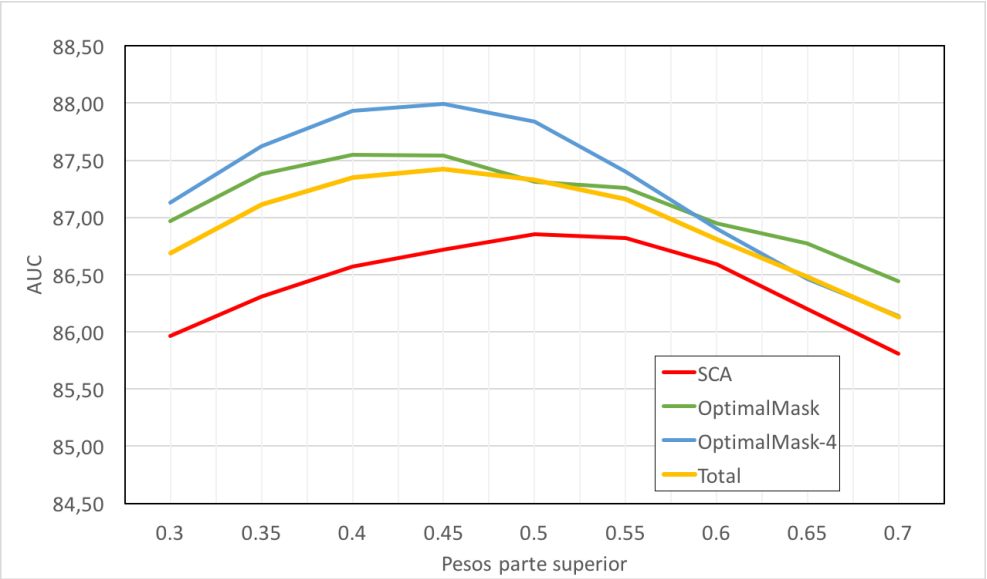
En el caso de HSV la mejor máscara, como se esperaba, es OptimalMaskMod. Curiosamente los mejores valores se dan cuando la parte superior tiene un peso entre 0.4 y 0.45. Por tanto parece que la parte inferior tiene algo más de relevancia que la superior (aunque las diferencias son mínimas). En HSV además se aprecia como las máscaras obtenidas con GrabCut dan mejores resultados que SCA.

Con IIP ocurre que el mejor resultado es exactamente 0.5, igualdad de peso para la parte inferior y superior, también con la máscara OptimalMaskMod. Sin embargo por la forma de la gráfica de OptimalMaskMod parece que tiene algo más de relevancia la parte superior que la inferior, pues según aumenta el peso el AUC no decae tan rápido como en el caso de HSV. Con SCA se aprecia el comportamiento opuesto.

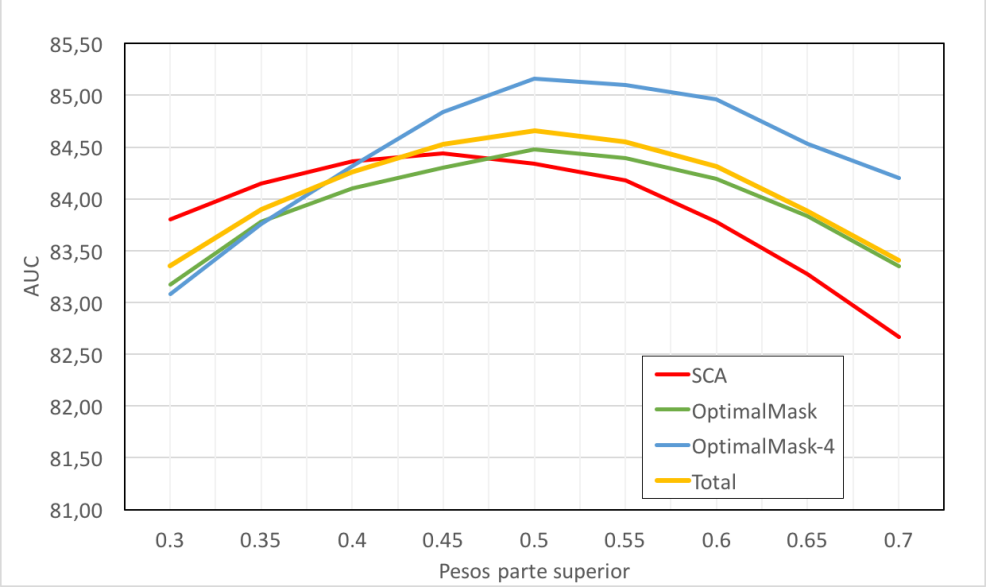
En general se concluye que los mejores resultados se dan en torno a 0.5, es decir, equilibrados los pesos y con la segmentación OptimalMaskMod.

5.7.3. SilPart2

El caso de SilPart se va a estudiar desde dos puntos diferentes. Por un lado, se pueden asignar, al igual que con SilPart, solamente 2 pesos. Esto agrupará



(a) Espacio de color HSV.



(b) Espacio de color IIP.

Figura 5.8: Influencia de los pesos de la parte superior en SilPart en el AUC.

las regiones en dos grupos, estando cada grupo compuesto de 2 regiones que serán tratadas como una sola. La otra posibilidad es tratar cada región por separado, usando 4 pesos, uno para cada región.

En la figura 5.9 se muestra el resultado de usar únicamente 2 pesos. Se sigue la misma metodología que con SilPart (apartado 5.7.2). Los resultados no son nada malos y se observa un comportamiento muy similar al obtenido en SilPart como era de esperar. Sin embargo mejora de media un punto los resultados de SilPart.

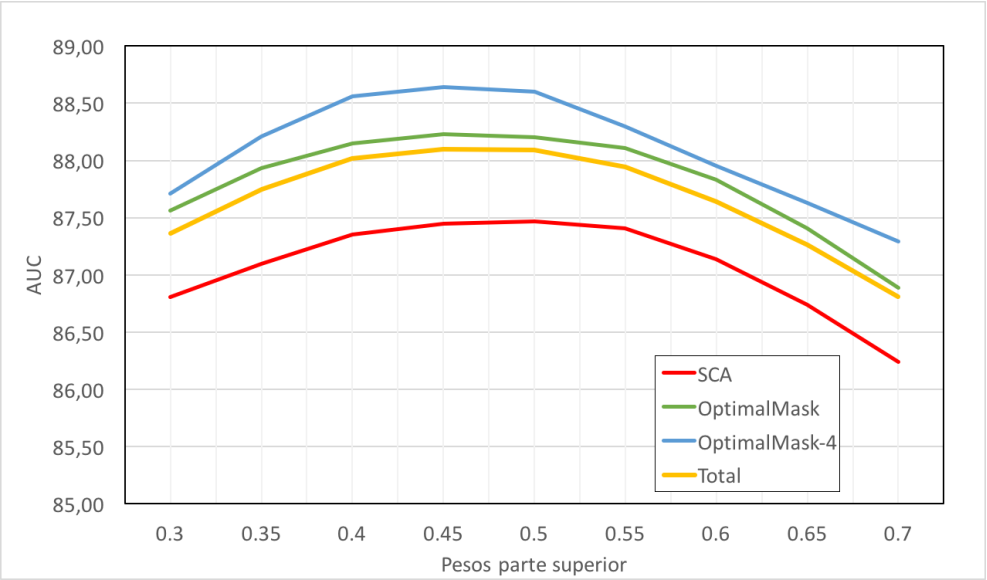
A continuación se realizan pruebas usando 4 pesos. Para obtener el valores de los pesos para cada región se ha utilizado la ecuación 5.4:

$$\begin{cases} w_1 = a * b \\ w_2 = a * (1 - b) \\ w_3 = (1 - a) * c \\ w_4 = (1 - a) * (1 - c) \\ 0,4 \leq a, b, c \leq 0.6 \\ \sum w_x = 1 \end{cases} \tag{5.4}$$

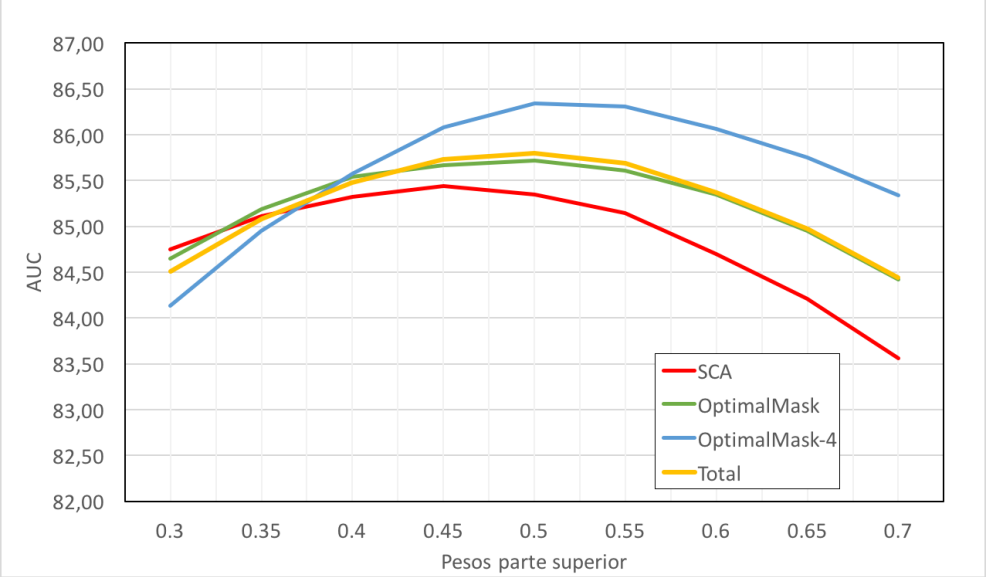
Siendo w_x los pesos de las regiones empezando por la región superior. Esto genera pesos para cada región que oscilan entre 0.16 y 0.36, siempre sumando el total de los pesos 1. También impone la restricción de que las dos regiones superiores y las dos inferiores, en conjunto, no superen 0.6. Así se intenta mantener los pesos equilibrados y limitar el conjunto de experimentos, vistos los resultados obtenidos en el paso anterior, a la zona de interés, evitando una cantidad de ejecuciones excesiva.

Para mostrar los resultados, se realiza un análisis del peso ideal para cada una de las 4 regiones. Para ello se fija un valor del peso y se promedia el resultado obtenido usando todas las combinaciones que tengan ese peso en esa región. Esto se hace para cada posible valor de cada región. En la tabla 5.13 se muestra el mejor peso para cada región obtenido de esta forma. Como en el caso de 5R, se limitan las pruebas al espacio de color HSV y a la segmentación OptimalMaskMod.

Curiosamente en este caso se observa como la región más influyente es la tercera (región 2), que se correspondería con la parte superior de las piernas. Sin embargo, la diferencia entre los distintos pesos no es muy relevante.



(a) Espacio de color HSV.



(b) Espacio de color IIP.

Figura 5.9: Influencia de los pesos de la parte superior en SilPart2 en el AUC. Usando 2 pesos.

Pesos \ Regiones	0	1	2	3
0.16	88.26	88.40	87.29	87.81
0.18	88.35	**88.52**	87.40	87.91
0.2	88.33	88.36	87.82	88.01
0.2025	**88.42**	88.50	87.95	88.13
0.22	88.21	88.16	87.92	88.03
0.225	88.39	88.46	88.26	88.24
0.24	87.99	87.85	87.92	87.92
0.2475	88.27	88.42	88.37	88.27
0.25	88.36	88.36	88.38	**88.38**
0.27	88.04	88.39	88.23	88.16
0.275	88.20	88.18	88.47	**88.38**
0.3	87.96	87.93	88.45	88.26
0.3025	88.03	88.03	**88.58**	88.37
0.33	87.78	87.79	88.57	88.22
0.36	87.44	87.57	88.56	88.03

Tabla 5.13: Influencia de usar 4 pesos en SilPart2 en el AUC. Espacio de color HSV y segmentación OptimalMaskMod.

Método	Pesos	AUC
5R	[0.25, 0.25, 0.3, 0.1, 0.1]	89.33
SilPart2	Sin Peso	89.13
SilPart	Sin Peso	88.15

Tabla 5.14: Comparación mejores resultados para cada método de partición de regiones.

5.7.4. Mejores resultados

En este apartado se comparan los distintos métodos de particiones para tener una perspectiva de qué método ofrece mejores resultados. Para ello se crea la tabla 5.14 en la que se muestra el mejor resultado para cada método de partición, mostrando los pesos usados. El mejor método parece ser la partición en 5 regiones verticales. Seguido muy de cerca por la partición SilPart2. SilPart parece encontrarse a algo más de distancia.

5.8. Mapas gaussianos

En esta sección se probarán los mapas gaussianos para calcular histogramas ponderados. Las condiciones experimentales serán las siguientes:

- Método de comparación de histogramas: Bhattacharyya.
- Espacios de color:
  - HSV: bins [32, 16, 0] (1 combinación)
  - IIP: bins: [0, 32, 0] respectivamente para cada canal. (1 combinación)
- Preprocesado:
  - Segmentación: GrabCut (OptimalMaskMod) y SCA.
  - Partición en regiones: SilPart2 y 5R

Figura 5.10: Esquema de las pruebas realizadas para Mapas Gaussianos.

Se pasa a usar sólo una de las máscaras en GrabCut para disminuir la cantidad de pruebas a realizar. Serán probados los dos métodos para generar los mapas: la combinación de mapas gaussianos (*«GMM»*) y el kernel unidimensional gaussiano (*«Gaussian»*). El procedimiento seguido para realizar las pruebas se resume en el esquema 5.10.

En éste se refleja fundamentalmente que se realiza siempre la misma partición en regiones inicial para generar el mapa. Con el mapa ya generado, se usa una partición de regiones diferentes para evaluarlo. Esto se hace así para separar la partición de regiones del mapa que usaremos. Se podrá ver, por ejemplo, si un mismo mapa es más efectivo usando una partición u otra. El mapa que será generado usará la partición en regiones SilPart (explicada en el apartado5.7.2) simple (sin subdivisiones). El mapa que se generará será por tanto para dos regiones, idealmente tronco superior y piernas. La particiones en regiones que se usarán para realizar las pruebas serán SilPart2 (partición de la silueta con subdivisión, 4 regiones en total) y 5R (5 regiones verticales).

5.8.1. Gaussiana simple

En primer lugar se estudia el mapa generado con una Gaussiana simple (método «*Guassian*»). Para ello se realizan múltiples ejecuciones con las condiciones experimentales descritas anteriormente variando el valor de Sigma (σ) en el rango [2..10]. Esto se hará para las dos regiones, pues cada región puede tener una guassiana independiente. Se calculará la importancia de cada valor de sigma, fijando éste y calculando la media para todas las ejecuciones en las que éste valor aparezca, de forma similar a como se estudia la importancia de los bins en los espacios de color (apartado 5.3).

En las figuras 5.11 y 5.12 se muestran los resultados en los que se observa el efecto sobre el AUC de variar el valor de sigma (σ) a la hora de calcular la guassiana. Cada una de ellas muestra los resultados para cada espacio de color y para las 2 regiones, tronco superior y tronco inferior. En ellas se muestran 4 líneas, la combinación de las 2 segmentaciones y las dos particiones de

Segmentación	Partición	Sigma Superior	Sigma Inferior	AUC
OptimalMaskMod	5R	4	4	89,08
OptimalMaskMod	5R	4	4,5	89,06
OptimalMaskMod	SilPart2	4	4	89,05
OptimalMaskMod	SilPart2	4	4,5	89,04
OptimalMaskMod	5R	4,5	4,5	89,01

(a) Espacio de color HSV.

Segmentación	Partición	Sigma Superior	Sigma Inferior	AUC
OptimalMaskMod	5R	4	6	86,39
OptimalMaskMod	5R	4	5	86,39
OptimalMaskMod	5R	4	4,5	86,38
OptimalMaskMod	5R	4	5,5	86,35
OptimalMaskMod	5R	4	4	86,33

(b) Espacio de color IIP.

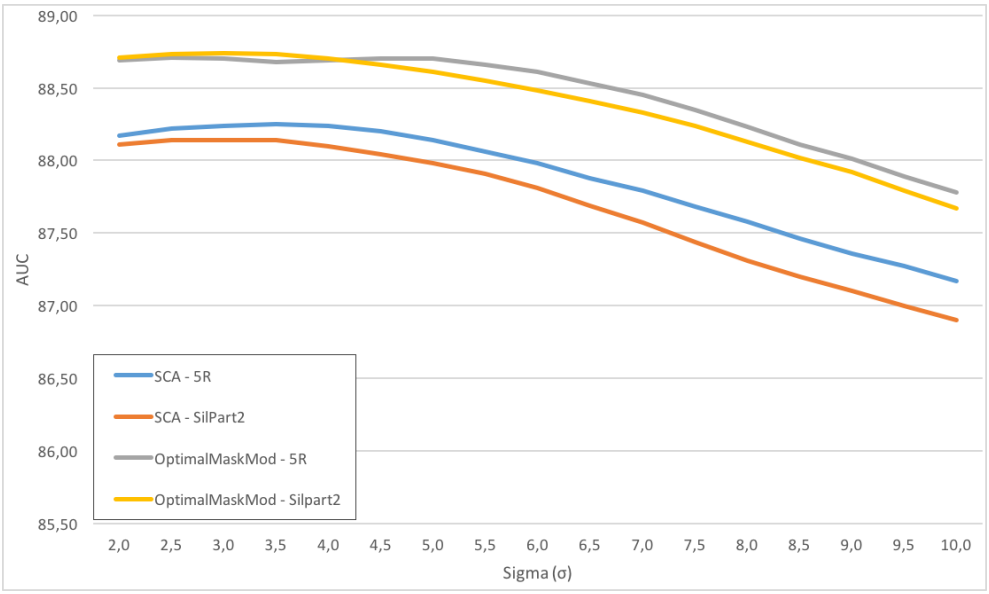
Tabla 5.15: 5 mejores resultados con mapas guassianos.

regiones que se realizan. El número de ejecuciones es de 4624.

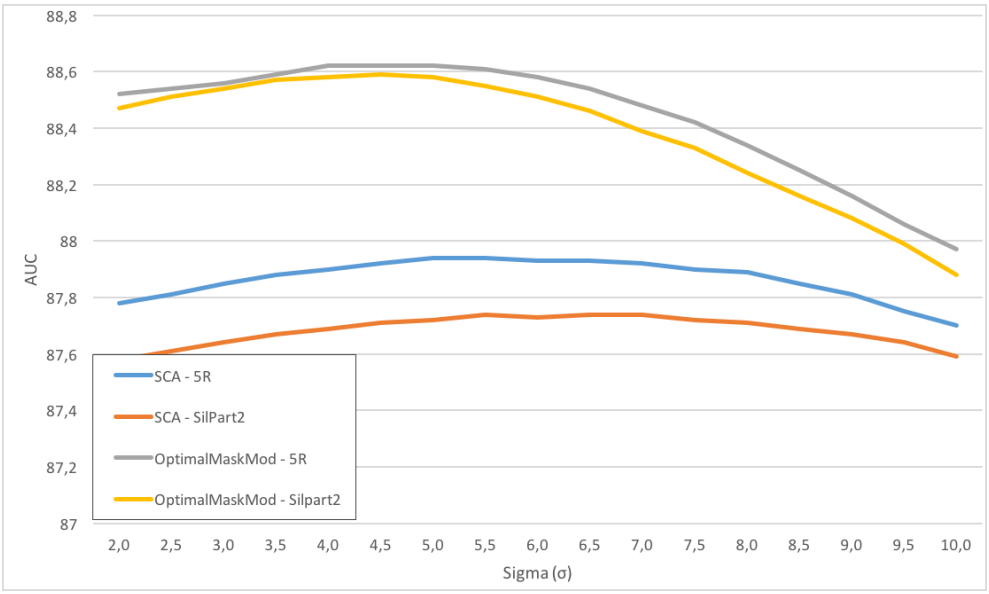
En los resultados se ve que la mejor combinación siempre es usando OptimalMaskMod y la partición en 5 regiones. En el caso de HSV parece que valores bajos de Sigma producen mejores resultados en la parte superior. En la parte superior, un valor intermedio entre 4 y 6 parece ofrecer los mejores resultados. En el caso de IIP nuevamente los valores bajos en la parte superior ofrecen resultados mejores, llamando especialmente la atención el caso de OptimalMaskMod en el que se producen diferencias mayores entre valores bajos y altos. Para la parte inferior, valores intermedios parecen ser los mejores, aunque para SCA parece que valores bajos ofrecen resultados similares que los intermedios.

En la tabla 5.15 se muestran los 5 mejores resultados para cada espacio de color. Parece claro que el mejor valor de Sigma para la parte superior es 4. En la parte inferior es donde se ve una ligera discrepancia según el espacio de color, aunque se encuentran entre 4 y 6 en cualquier caso.

Para entender los valores de sigma y qué representan se muestran en la figura 5.13 varios ejemplos de valores para la parte inferior y superior. En primer lugar se muestra la que parece la mejor combinación, usar 4 como valor de

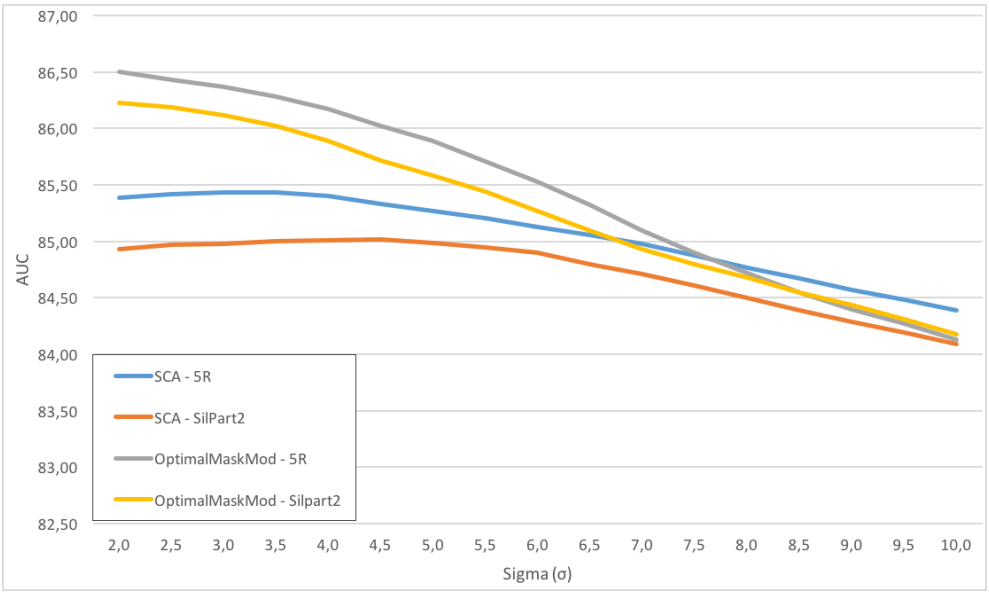


(a) Sigma (σ) varía para la región superior.

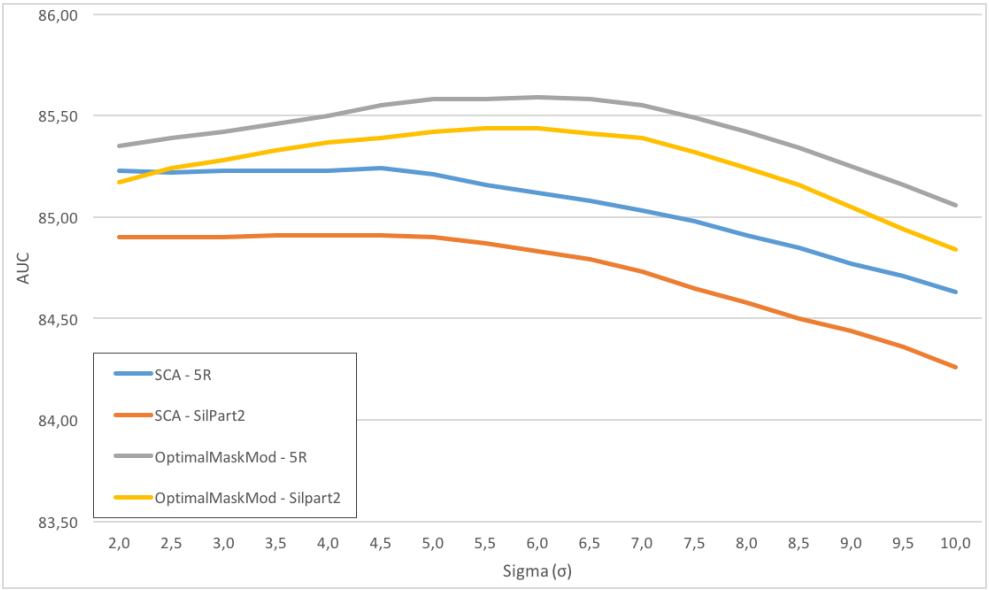


(b) Sigma (σ) varía para la región inferior.

Figura 5.11: Efecto de variar sigma (σ) al calcular mapas guassianos simples. Espacio de color HSV.

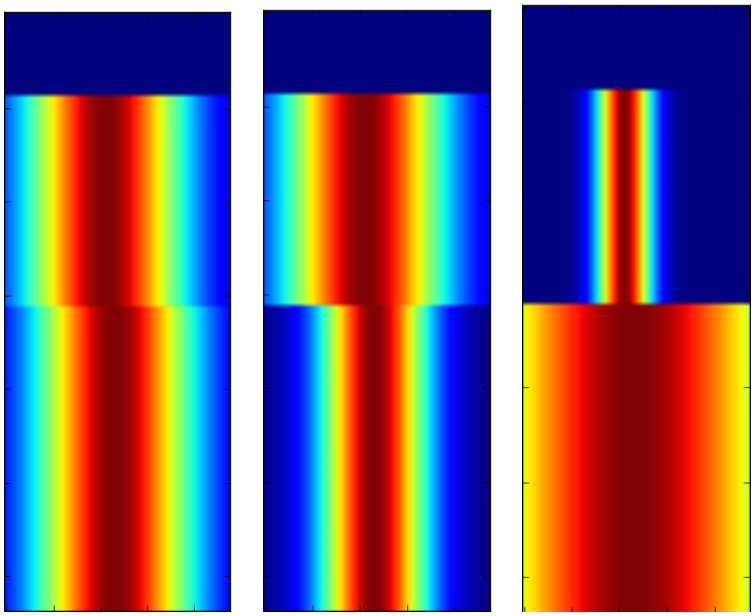


(a) Sigma (σ) varía para la región superior.



(b) Sigma (σ) varía para la región inferior.

Figura 5.12: Efecto de variar sigma (σ) al calcular mapas guassianos simples. Espacio de color IIP.



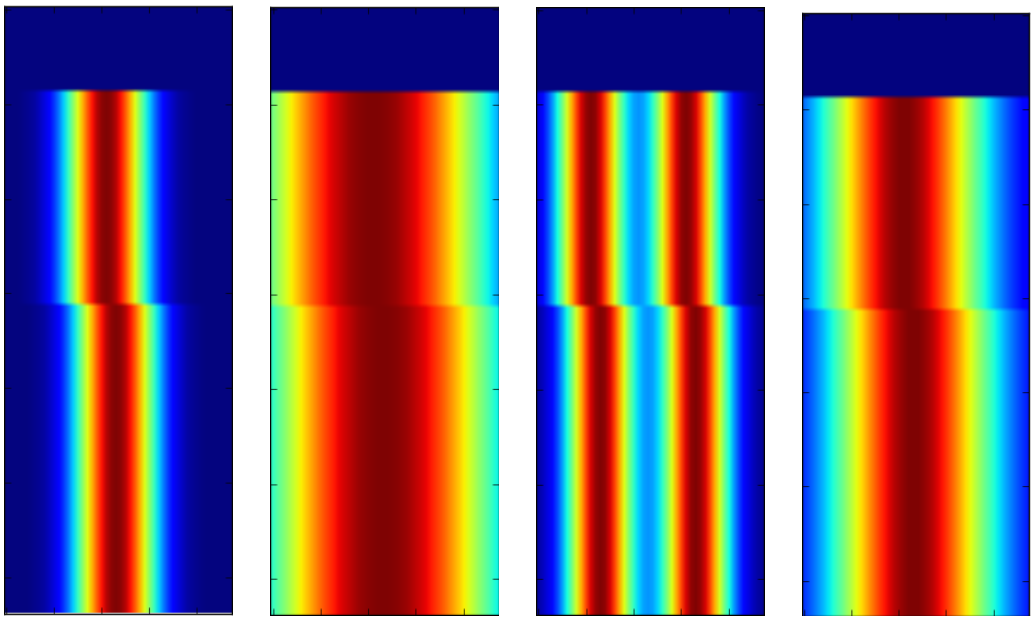
(a) Valor superior: 4 Valor inferior: 4 (b) Valor superior: 4 Valor inferior: 6 (c) Valor superior: 10 Valor inferior: 2

Figura 5.13: Mapas gaussianos obtenidos para diversos valores de sigma. El azul representa al valor nulo. Cuanto más rojo mayor valor.

sigma en la parte superior e inferior. También se muestra la combinación de 4 en la parte superior y 6 en la inferior. Por último, se muestran los casos extremos de los rangos en los que probamos: En la parte superior se usa un gima 10 y en la inferior un sigma 2. En los mapas el valor azul representa el valor nulo y cuanto más cercano al rojo mayor peso de ese píxel. Así se ve que cuanto mayor es el valor de sigma, más estrecha se hace la zona de importancia.

5.8.2. Mezcla de gaussianas (GMM)

A continuación se estudia el método de mezcla de gaussianas. En este caso, se usarán dos gaussianas, desplazadas a cada lado del eje de simetría. Por tanto, ahora además de controlar los valores de sigma (σ), se ha de controlar el valor de desviación. De esta forma, la cantidad de ejecuciones crece exponencialmente. Por ello, se decide mostrar únicamente resultados para el espacio de



(a) Sigma: 9 Desv: 3 (b) Sigma: 4 Desv: 10 (c) Sigma: 9 Desv: 10 (d) Sigma: 4 Desv: 3

Figura 5.14: Mapas generados con distintos valores de sigma (σ) y desviación.

color HSV y la segmentación OptimalMaskMod, que ha demostrado ser la mejor combinación.

En las tablas 5.16 y 5.17 se muestra una tabla para cada región y partición de regiones, en las que se representa el AUC dependiendo del valor de sigma y el valor de desviación. Como se ve, no existe una gran diferencia entre los distintos rangos. Mirando los valores en bruto, entre el mejor y el peor resultado apenas hay un punto de diferencia.

En la figura 5.14 se muestran ejemplos de mapas con distintos valores de sigma y desviación. Ahí se puede apreciar la diferencia entre los mapas. Parece que no afecta demasiado al resultado, aunque el mejor resultado en la parte superior es mantener una apertura grande (entre 10 y 9). El resto de valores parece no ser tan relevante, aunque como se ha visto las diferencias son mínimas.

Desviación \ Sigma (σ)	4.0	5.0	6.0	7.0	8.0	9.0
3.0	88.99	88.92	88.82	88.66	88.49	88.31
4.0	89.00	88.94	88.85	88.76	88.64	88.54
5.0	89.00	88.96	88.89	88.84	88.78	88.74
6.0	89.00	88.97	88.94	88.90	88.87	88.85
7.0	89.01	88.99	88.96	88.92	88.92	88.93
8.0	89.02	88.99	88.94	88.93	88.98	89.01
9.0	89.02	88.99	88.95	89.00	89.06	89.08
10.0	89.02	88.98	89.01	89.08	89.11	**89.14**

(a) Partición de regiones: 5R. Sigmas y desviaciones de región superior.

Desviación \ Sigma (σ)	4.0	5.0	6.0	7.0	8.0	9.0
3.0	89.02	88.89	88.73	88.56	88.39	88.21
4.0	89.03	88.92	88.80	88.66	88.54	88.44
5.0	89.05	88.96	88.86	88.78	88.69	88.62
6.0	89.06	89.00	88.93	88.87	88.82	88.78
7.0	89.08	89.03	88.99	88.94	88.92	88.89
8.0	89.08	89.06	89.02	88.99	89.00	89.00
9.0	89.09	89.06	89.04	89.06	89.04	89.04
10.0	89.09	89.05	89.07	**89.10**	89.07	89.03

(b) Partición de regiones: 5R. Sigmas y desviaciones de región superior.

Tabla 5.16: Variación del AUC en base a sigma y desviación con mapas GMM en la parte superior. Espacio de color HSV y segmentación OptimalMaskMod.

Desviación \ Sigma (σ)	4.0	5.0	6.0	7.0	8.0	9.0
3.0	88.97	88.92	88.83	88.73	88.62	88.49
4.0	88.97	88.94	88.87	88.80	88.74	88.68
5.0	88.98	88.95	88.92	88.88	88.86	88.85
6.0	88.98	88.96	88.95	88.95	88.95	88.95
7.0	88.98	88.97	88.98	88.98	88.98	88.98
8.0	88.98	88.98	88.97	88.97	88.98	88.97
9.0	88.97	88.96	88.95	88.97	88.96	88.92
10.0	88.96	88.94	88.95	88.95	88.90	88.81

(a) Partición de regiones: 5R. Sigmas y desviaciones de región superior.

Desviación \ Sigma (σ)	4.0	5.0	6.0	7.0	8.0	9.0
3.0	88.97	88.93	88.84	88.72	88.58	88.43
4.0	88.96	88.94	88.88	88.80	88.72	88.65
5.0	88.96	88.95	88.92	88.89	88.86	88.85
6.0	88.96	88.96	88.95	88.95	88.94	88.94
7.0	88.96	88.96	88.96	88.96	88.96	88.98
8.0	88.95	88.95	88.94	88.95	88.98	89.00
9.0	88.94	88.94	88.93	88.96	88.98	88.96
10.0	88.93	88.91	88.94	88.96	88.94	88.84

(b) Partición de regiones: 5R. Sigmas y desviaciones de región superior.

Tabla 5.17: Variación del AUC en base a sigma y desviación con mapas GMM en la parte inferior. Espacio de color HSV y segmentación OptimalMaskMod.

Partición	Sigmas		AUC
SilPart2	2	2	89,26
SilPart2	2	2.5	89,25
SilPart2	2	3.5	89,24
SilPart2	2	3	89,23
SilPart2	2.5	3	89,22

(a) 5 mejores resultados usando mapas gaussianos.

Partición	Sigmas		Desviación		AUC
SilPart2	6	8	10	7	89.24
5R	8	4	10	4	89.24
SilPart2	7	9	10	7	89.24
5R	9	5	10	5	89.24
SilPart2	7	9	10	8	89.24

(b) 5 mejores resultados usando mapas GMM.

Partición	AUC
5R	89.17
SilPart2	89.13

(c) Resultados sin usar mapas.

Tabla 5.18: Comparación de los mejores resultados usando mapas gaussianos, GMM y sin usar mapas. Espacio de color HSV y segmentación OptimalMaskMod.

5.8.3. Comparación

En este apartado se van a poner en contexto los resultados obtenidos, comparando el uso de la Gaussiana, GMM o no usar ningún mapa de pesos. En la tabla 5.18 se muestran los mejores resultados, usando sólo el espacio de color HSV y la máscara OptimalMaskMod. A la vista de los resultados, parece que aporta poco al proceso utilizar los mapas gaussianos o GMM.

5.9. Comparativa con otros métodos

En esta sección se van a evaluar los resultados obtenidos y ponerlos en perspectiva con otros trabajos del área de reidentificación. Inicialmente, se buscará la mejor configuración que ofrece la combinación de los distintos métodos vistos hasta el momento. Para ello se realizan pruebas con los siguientes valores:

- Método de comparación de histogramas: Bhattacharyya.

Partición	Pesos	Gaussian	bins	AUC	Rango 1	Rango 20
5R	No	Sí	[32, 32, 4]	91.92	21.71	64.24
5R	Sí	No	[16, 32, 4]	91.91	20.70	65.85
5R	Sí	No	[16, 16, 4]	91.88	21.42	66.46
SilPart2	No	No	[32, 32, 4]	91.87	20.67	64.59
SilPart2	No	No	[16, 32, 4]	91.85	18.96	64.94

Tabla 5.19: 5 mejores resultados.

- Espacios de color:
  - HSV: bins H y S: [32, 16]. Bins V: [0, 4]
- Preprocesado:
  - Segmentación: GrabCut (OptimalMaskMod).
  - Partición en regiones: SilPart2 y 5R
  - Mapas: Sin mapas y Gaussianos: Sigmas 2, 2.
  - Normalización de la Iluminación: HSV + YCrCb
- Pesos Comparación de histogramas:
  - Sin pesos
  - 5R: 0.25, 0.25, 0.3, 0.1, 0.1

En la tabla 5.19 se muestran los 5 mejores resultados obtenidos. Se muestra el AUC, el Rango 1 y Rango 20. Son resultados rozando el 92 % del AUC, con un rango 1 superior al 20 % y con un rango 20 en torno al 65-67 %.

A continuación se va a comparar con los siguientes métodos:

SDALF [Farenzena et al, 2010], de donde se obtiene la idea de usar los histogramas ponderados. En nuestras pruebas los beneficios de añadirlos es bastante limitada. También es de donde se obtiene la partición en regiones SilPart, la cual se ha modificado para obtener SilPart2.

GMMWCH [Wei and Lin, 2013], del que se toma la idea de la normalización de la iluminación (gran aporte) y la mezcla de mapas gaussianos. En este caso no se consigue repetir el éxito que parecen obtener en su trabajo.

KISSME [Kostinger et al, 2012]Este método no se ha implementado, pues requiere de aprendizaje. Se añade a modo de referencia, aunque obtiene resultados bastante mejores. Esto es porque utiliza la mitad de los elementos como entrenamiento. Como se mencionó, esto en un escenario real requeriría de un gran esfuerzo y parece poco realista.

En la figura 5.15 se muestran los resultados usando 316 elementos. En esta prueba se utiliza el segundo mejor resultado de la tabla 5.19, esto es, 5R, usando pesos y sin usar los histogramas pesados. Como se comentaba, KISSME es el mejor método, pero se ha puesto para tener una referencia. Con SDALF parece que la cosa está bastante igualada con nuestro método. Sin embargo, cabe destacar que nuestro trabajo utiliza una partición en regiones estática, no se usan mapas gaussianos y el proceso de segmentación es más simple. Además, SDALF utiliza 2 métodos complementarios para mejorar la precisión, como son la búsqueda de regiones de colores estables y otros criterios basados en comparar las texturas. Es decir, nuestro método requiere de menor necesidad de cálculo.

En la figura 5.16 se muestran el CMC sólo de los 50 mejores resultados. Aquí se incluye también el método GMMWCH, del cual no se dispone de suficientes datos como para ser incluido en la figura anterior. En este caso se observa que los tres métodos están bastante igualados.

Como se ha visto la combinación de métodos que se han implementado para este proyecto consiguen resultados competitivos compitiendo con trabajos más complejos y que utilizan una mayor cantidad de técnicas.

En la siguiente sección, se analizarán los efectos de realizar el proceso de reordenación.

5.10. Reordenación

Para evaluar cómo funciona la reordenación se ha de seguir una metodología diferente a la seguida hasta ahora. El método de reordenación depende de

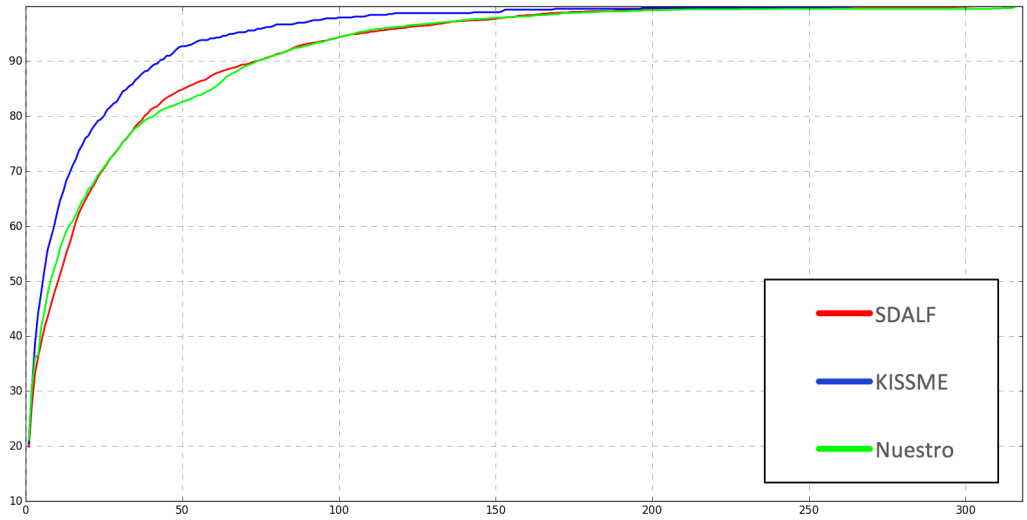


Figura 5.15: Comparativa métodos de reidentificación de personas. Curva CMC.

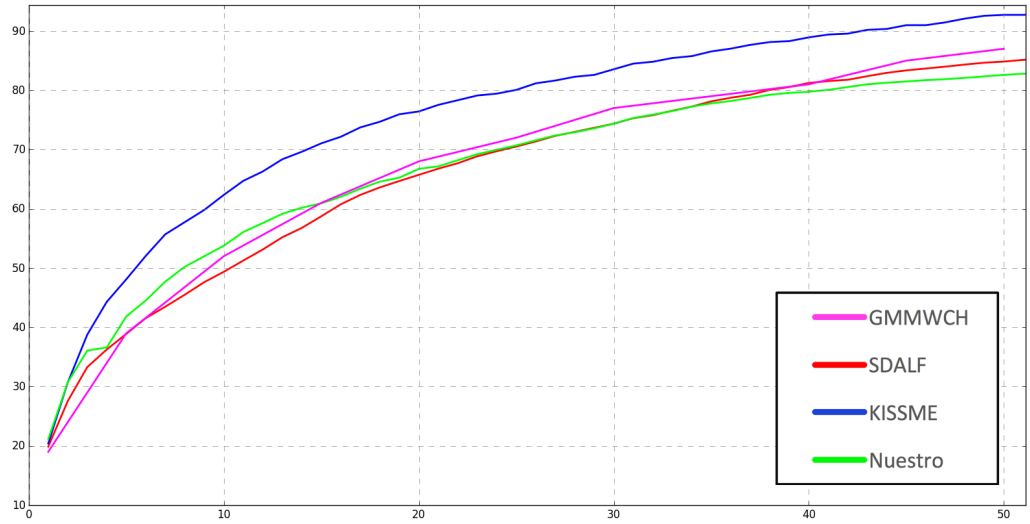


Figura 5.16: Comparativa métodos de reidentificación. Curva CMC para 50 primeros.

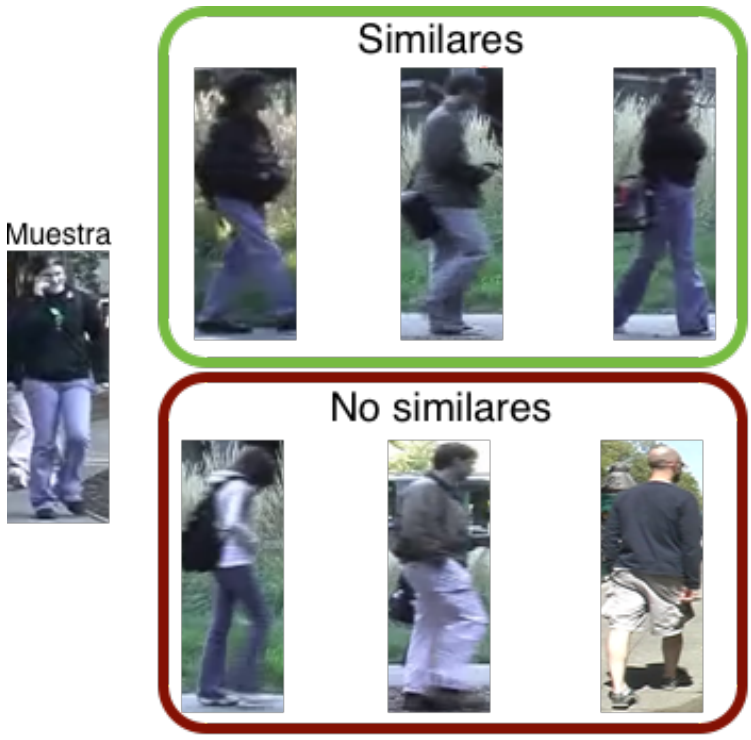


Figura 5.17: Ejemplo de imágenes escogidas como similares y no similares para un elemento de la muestra. Imágenes para región de imagen completa.

la interacción de un usuario, que indicará muestras similares y no similares. Para poder simular este efecto, se realiza previamente una selección manual de estas muestras. Esto se realiza para 11 elementos de la muestra. Se puede ver un ejemplo en la figura 5.17.

Para evaluar por tanto la calidad del método, se evalúa en base a la mejora que se consiga en estas 11 muestras. La medida consistirá en la suma de las mejoras en la posición en la ordenación de la imagen objetivo. Así, si la imagen correspondiente a la muestra aparece en la posición 50, y con el método de reordenación pasa a la posición 40, se contabilizará la mejora como +10. Sin embargo, si pasara a la posición 65, la mejora se contabilizará como -15. Esto se realiza para los 11 elementos de la muestra y se suman estas mejoras. Un resultado positivo, indicará que en general se ha producido una mejora en la ordenación. Un resultado negativo indicará que los resultados

han empeorado.

Como en algunos métodos existe cierto grado de aleatoriedad los resultados no siempre son iguales para los mismos parámetros. Por ello se realizan las pruebas 10 veces y se promedian para cada uno de los elementos de la muestra. También se mostrará en algunos casos la desviación estándar, que permitirá visualizar si los resultados en general están cerca de la media o varían mucho.

En primer lugar, se prueba usando sólo la Similitud y la Afinidad (a Lab_score se le da un peso de 0). Lab_score es más pesado para computar, así que se intentará afinar los parámetros para los dos primeros y luego se harán pruebas con Lab_score.

Así pues, estos son los valores con los que se va a realizar las pruebas:

- Ordenación inicial igual en todas las pruebas.
- Pesos para cada método:
  - Similitud: Entre 0 y 1.
  - Afinidad: Entre 0 y 1.
  - Lab_score: 0.
- Siempre balanceados.
- Modificación de la puntuación*:*
  - *Proporcional*: alpha entre 0.15 y 0.45
  - *No Proporcional*: alpha entre 0.15 y 0.45
- Regiones:
  - 2 regiones usando 5R.
  - Imagen completa.
- Sin expansión Visual.
- Parámetros afinidad:

Alpha	Regiones	Simil.	Afin.	rte_estims.	rte_leafs	Mejora	Desv.
0.2	No	1	0	80	3	17.3	3.66
0.3	No	1	0	100	3	16.8	5.27
0.25	No	0.8	0.2	80	3	16.6	3.75
0.35	No	0.6	0.4	60	5	16.6	6.51
0.4	No	0.6	0.4	40	8	16.2	8.38

(a) No Proporcional.

Alpha	Regiones	Simil.	Afin.	rte_estims.	rte_leafs	Mejora	Desv.
0.4	No	1	0	20	8	16	5.55
0.45	No	1	0	100	3	15.7	4.94
0.45	No	0.8	0.2	20	8	15.2	4.12
0.45	No	0.8	0.2	100	8	15.2	5.17
0.4	No	1	0	80	8	15.1	4.74

(b) Proporcional.

Tabla 5.20: 5 mejores resultados para la reordenación usando Similitud y Afinidad.

  - *rte_ estimators*: entre 20 y 100.
  - *rte_ leafs*: entre 3 y 8.

- Número de repeticiones de la prueba: 10

En la tabla 5.20 se muestran los 5 mejores resultados obtenidos. En primer lugar, llama la atención que el uso de las regiones, tal como se han aplicado, no ha dado un buen resultado. Parece que se comporta de una forma más efectiva cuando se utiliza la imagen completa. También se aprecia en los resultados una importancia mayor del factor similitud que del de afinidad. Puede deberse a que la estructura de los datos usados no sea la más adecuada para este tipo de técnica de clustering. En cuanto a los valores de *rte_ estimators* y *rte_ leafs* no parece haber un patrón claro, aunque en general valores altos en la cantidad de estimadores y en el número de hojas parece ofrecer mejores resultados.

A continuación se realizarán pruebas añadiendo el Lab_score. En vista de los resultados anteriores, se decide simplificar las pruebas:

- Pesos para cada método:
    - Similitud: Entre 0.9
    - Afinidad: Entre 0
    - Lab_score: 0.1
- Modificación de la puntuación*:*
    - *Proporcional*: alpha 0.5
    - *No Proporcional*: alpha 0.4
- Sin regiones.
- Parámetros Label Propagation/Spreading:
    - Kernels:
        - RBF: gamma entre 15 y 24.
        - Knn: número de vecinos entre 3 y 9.
    - alpha: entre 0.2 y 1

Como se ve se usará sólo la similitud unida a la puntuación de Lab_score. En la tabla 5.21 se muestran los 5 mejores resultados, donde *Param* indica el gamma o el número de vecinos, según el kernel que se use. Lo más destacable sin lugar a dudas son los altos valores de la desviación estándar. Esto es fruto de la aleatoriedad de los resultados obtenidos con el método de Label Propagation/Spreading. En general parece que el kernel RBF funciona mejor para estos datos y se pueden conseguir buenos resultados, pero de nuevo, los altos valores de la desviación estándar indican que no es un método muy estable.

Por último se realizarán pruebas para evaluar el funcionamiento de la expansión visual. En este caso se exige que los conjuntos estén balanceados y se activa la expansión visual. En esta prueba las condiciones son las siguientes:

- Pesos para cada método:
    - Similitud: Entre 0.8

Método	Lab-Alpha	Kernel	Param	Mejora	Desv.
spreading	0.73	RBF	18	19	7.34
propagation	1	RBF	24	16	8.28
propagation	1	RBF	21	14.4	11.83
spreading	0.2	RBF	18	14	7.69
spreading	0.47	RBF	18	12.4	6.79

(a) No Proporcional.

Método	Lab-Alpha	Kernel	Param	Mejora	Desv.
spreading	1	RBF	18	11.9	11.43
spreading	1	Knn	7	11.8	17.16
spreading	0.2	RBF	21	11.8	14.71
spreading	0.47	RBF	15	11.7	10.50
spreading	0.47	RBF	21	11	10.13

(b) Proporcional.

Tabla 5.21: 5 mejores resultados para la reordenación usando Similitud y Afinidad.

  - Afinidad: Entre 0.2
  - Lab_score: 0

- Modificación de la puntuación*:*
  - *Proporcional*: alpha 0.5
  - *No Proporcional*: alpha 0.4
- Sin regiones
- Parámetros afinidad:
  - *rte_estimators*: 80.
  - *rte_leafs*: 5.
- Parámetros expansión visual:
  - Número de estimadores: entre 20 y 100

Num. Estimadores	Num. hojas	Mejora	Desv.	Mejoras sin E.V.
80	8	7	2.15	16.8
40	8	6.9	2.55	15.7
80	3	6.8	3.16	14.5
20	3	6.6	3.90	14.2
80	5	6.6	1.80	12.8

(a) No Proporcional.

Num. Estimadores	Num. hojas	Mejora	Desv.	Mejoras sin E.V.
80	3	10.6	1.56	15.4
80	8	10.1	1.22	14.4
100	5	10	0.77	14.1
20	3	9.9	1.76	14
20	5	9.9	1.30	13.9

(b) Proporcional.

Tabla 5.22: 5 mejores resultados para la reordenación usando Similitud y Afinidad.

- Número de elementos por hoja: entre 3 y 8

En la tabla 5.22 se muestran los resultados obtenidos. Como se puede observar, al igual que en el caso de Lab_score, no aparece un patrón claro que indique cuál es la mejor configuración para la expansión visual. Sin embargo se observa cómo el hecho de utilizar la expansión visual parece no ofrecer tan buenos resultados como se esperaba. Parece que es preferible mantener los conjuntos desequilibrados a añadir elementos «de relleno» generados artificialmente. La expansión visual se utiliza en [Liu et al, 2013] porque el método que utilizan exige que los conjuntos de similares y no similares estén compensados. No es el caso en la implementación de este proyecto.

En general, se ha visto que la modificación de la ordenación inicial usando información proporcionada por el usuario puede ofrecer buenos resultados. Sin embargo, hay mucho que mejorar en este aspecto. Se ha de ser cuidadoso con la metodología a seguir. Sin duda, es un campo en el que futuros proyectos podrían encontrar mucho que explorar.

Capítulo 6

Conclusiones y posibles mejoras

En este capítulo se empezará comentando las conclusiones, en las que se hace un repaso los resultados obtenidos, qué se ha aprendido y otros temas diversos. A continuación se detallan algunas ideas sobre posibles líneas futuras en las que ampliar y mejorar este trabajo.

6.1. Conclusiones

Este proyecto ha ido evolucionando mucho a lo largo de su desarrollo. Se partía con una base muy amplia y genérica de la Reidentificación. Como su título indica, la idea consistía en reidentificar personas y objetos. Pronto se descubre lo genérico de la propuesta y lo complicado de ésta. Así, se decide centrarlo en la reidentificación de personas.

Posteriormente se decide enfocar el proyecto hacia una plataforma que facilite la incorporación de métodos y la realización de pruebas. Como se ha visto en el documento, han sido múltiples y diversas las técnicas usadas (con mayor o menor acierto) y eso complica mucho el mantener una cohesión entre todo lo realizado. La idea de un framework nace para facilitar estas tareas, para automatizar muchas pruebas, combinaciones entre métodos, variar el orden y parámetros de cada método, etc. Todo ello definiendo una serie de estructuras que faciliten la interacción a un desarrollador/investigador. Si bien queda camino por recorrer, errores que corregir y aspectos que mejorar, creo que este objetivo se ha cumplido.

PyReID se encuentra como proyecto de software libre disponible para todo el mundo en GitHub[1]. Se planea crear una documentación mínima, además de crear una serie de ejemplos que muestren su uso y versatilidad. Sin embargo la falta de tiempo hace que esta sea una tarea pendiente, pues en la actualidad la documentación es escasa.

Se han implementado una gran cantidad de métodos y se han realizado pruebas exhaustivas con cada uno de ellos. Es uno de los aspectos a los que más tiempo se ha dedicado. En primer lugar, por la ambición de realizar muchas pruebas para obtener una mayor visión y un mayor entendimiento de cómo afecta cada método al proceso de la reidentificación. En segundo lugar el querer realizar tantas pruebas ha llevado a optimizar muchas partes del código, realizando *profiling* para detectar aquellas partes más lentas. Inicialmente se desean realizar muchas pruebas. Por ello se optimizan ciertas partes que eran demasiado lentas. Sin embargo, también se aprende a priorizar las pruebas a realizar. Muchas de las pruebas han quedado descartadas por falta de tiempo y porque aportarían poco.

De entre todos los métodos, una de las mayores aportaciones sin duda es la segmentación con GrabCut. Se ha generado una máscara genérica para todo el conjunto de datos y se consiguen resultados superiores a los vistos en otros trabajos, con una mayor eficiencia. También se ha constatado que combinar la normalización de la iluminación en dos canales puede resultar en una mejora de los resultados.

También cabe resaltar los múltiples métodos que se usan y que no resultan en grandes resultados. El BTF no dio los resultados esperados. Quizá es algo de esperar dada la dificultad del conjunto de datos con el que se realizan las pruebas. Otro método que no consigue las mejoras esperadas son los histogramas ponderados. Inicialmente prometían mejorar los resultados, pero finalmente muestran que al menos en el caso de este proyecto no suponen apenas mejora. En un momento se llegó a pensar que estarían mal implementados, pero tras realizar pruebas y comprobaciones exhaustivas se ve que no es ese el problema. Del espacio de color IIP, al estar basado en la percepción del color, se esperaban buenos resultados. Sin embargo no mejora los resultados del espacio de color HSV. Queda pendiente el uso de este espacio de color en combinación con otros. Aquí quizá si se puedan conseguir encontrarle una mayor utilidad.

[1]PyReID: `https://github.com/Luigolas/PyReID`

Así mismo en el caso de la reordenación se han obtenido resultados dispares. Por un lado, se ha demostrado que es posible mejorar los resultados usando la retroalimentación del usuario. Por otro lado, se ha visto que es una tarea compleja. La medida de similitud, la más *ingenua* de todas, demuestra ser la más relevante de las tres. Usar la afinidad (aprendizaje no supervisado) y el método Label Propagation/Spreading (aprendizaje semisupervisado) no resultan tan interesantes como se esperaba. Es un área en la que hay que seguir explorando y mejorando para conseguir mejores resultados. Queda como tarea pendiente. La falta de tiempo ha hecho que no se pueda explorar en mayor profundidad, pero no cabe duda de que es un área prometedora.

Sin duda, debido a su componente de exploración, este proyecto podría alargarse mucho más. Quedan sin cubrir algunos aspectos como las técnicas de aprendizaje en la ordenación inicial, que parecen en auge en este campo. Sin embargo, como se indica desde el inicio del proyecto, el enfoque no va hacia estas técnicas pues requieren de un gran esfuerzo previo de etiquetado que limita su utilidad a algunos escenarios muy concretos.

Con este proyecto espero que otros puedan encontrar una pequeña ayuda que les facilite sus investigaciones en el campo de la reidentificación de personas. Es mi deseo que este trabajo realizado sea continuado, ampliando sus posibilidades y en definitiva aprovechando una gran cantidad de trabajo ya realizado.

Sin duda, doy este proyecto final de carrera como completo. He aprendido muchas cosas con un proyecto de esta envergadura. He encontrando dificultades en la implementación que he tenido que aprender a solventar y se han producido varios cambios de rumbo con respecto a la planificación inicial que han derivado en este resultado final. Este proyecto tiene una gran parte de exploración así que esa posibilidad se contemplaba. He aprendido la dificultad de realizar un proyecto por mi cuenta sin un «plazo de entrega» preestablecido, administrando el tiempo para conseguir terminarlo. Sin duda, la ayuda de los tutores ha resultado inestimable en esta empresa. Sin su ayuda y motivación, este proyecto no habría salido adelante. Gracias.

6.2. Posibles mejoras

Éste es un proyecto que no muere con la conclusión y defensa de esta memoria. PyReID es una plataforma Open Source disponible para cualquier persona

que quiera usarla, modificarla y experimentar con ella. PyReID puede ser mejorado en muchos aspectos. Muchas de las cosas previstas han tenido que quedarse en el tintero, pues no entrarían en el ámbito de este proyecto y abarcarían demasiado. Es el deseo del autor y tutores que esta plataforma siga creciendo en el futuro.

En esta sección se describen todas aquellas líneas en las que se contempla seguir mejorando la plataforma.

6.2.1. Soporte para conjuntos de datos Multi-Shot

Como se ha visto, PyReID se ha limitado a los conjuntos de datos de tipo Single-Shot, esto es, que un individuo sólo aparece una vez en la muestra y otra en la galería. Esto ha permitido simplificar la estructura y centrar los esfuerzos en otros aspectos. En un inicio se contempla el escenario Multi-Shot, pero se descarta para dedicar más tiempo a la implementación de nuevos métodos y algoritmos. Sin embargo sería muy interesante añadir este soporte para ampliar las posibilidades.

Para implementarlo se tendrían que cambiar varias cosas de la estructura. En varias partes el código asume que sólo va a encontrar una imagen por individuo. Hay algunos métodos que habría que replantear. Por ejemplo, en la reordenación, habría que contemplar si el usuario tiene que marcar las X muestras de un individuo o sólo una, o para realizar la *Expansión Visual* qué se haría con todas las imágenes.

6.2.2. Suite de tests completo

En una de las fases del desarrollo se utiliza TDD para implementar el código. Esto conllevó la creación de cientos de tests. En una fase posterior en aras de crear más funcionalidades, se abandona este modelo. Una mejora deseable sería volver a realizar un conjunto mínimo de tests para tener controlado siempre el estado del código.

6.2.3. Mejora de rendimiento: Cython

Para mejorar el rendimiento en Python se puede hacer uso de Cython [Behnel et al, 2011]. Éste, entre otras opciones, compila el código Python a C++, lo

que permite que su ejecución mejore drásticamente. Hay varios algoritmos implementados en puro Python, como el caso del BTF. Otros muchos se basan principalmente en funciones de OpenCV (que ya es C++) por lo que no se beneficiarían, pero desde luego ciertas partes podrían ser mejoradas.

6.2.4. Añadir nuevos métodos de extracción de características

Por limitaciones de tiempo, no se ha podido añadir nuevos métodos de extracción de características. Se ha decidido usar los histogramas e histogramas ponderados porque aparece en la literatura como uno de los métodos más utilizados [Saghafi et al, 2014]. Sin embargo, existen otras técnicas que pueden aportar características interesantes, como los descriptores de texturas [Kostinger et al, 2012] u operadores como el *Maximally Stable Color Regions* de [Farenzena et al, 2010].

Por supuesto, lo ideal es poder combinar múltiples extractores de características, darles diferentes pesos e incluso combinar dos iguales pero configurados de forma diferente. Por ejemplo, combinar histogramas HSV con histogramas IIP y ver si la combinación produce mejores resultados que por separado.

6.2.5. Explorar más métodos de reordenación

Sin duda uno de los aspectos en los que más se puede investigar es en los distintos métodos de reordenación. En este proyecto se ha tratado de realizar una pequeña aportación, pero es un campo poco explorado. En vista de los resultados actuales y de la dificultad que supone realizar una reidentificación casi perfecta, los métodos de reordenación que aprovechen la información proporcionada por el usuario podrían ser soluciones más aplicables en un escenario real. Normalmente para este tipo de aplicaciones siempre existe la supervisión de un operador humano, por lo que se podría aprovechar la información que éste provee para refinar los resultados.

6.2.6. Documentación

Para facilitar su visibilidad y utilidad para otros desarrolladores/investigadores se planea realizar una documentación online, en inglés, disponible para

todo aquel que lo requiera. Es uno de los planes de futuro y sin duda de los puntos mas importantes para mejorar su usabilidad y visibilidad.

Glosario

CMC Cumulative Matching Curve. Representa la expectativa de encontrar la correspondencia correcta (un individuo de la muestra con uno de la galería) entre las x primeras posiciones.

AUC Área bajo la curva CMC. El AUC (*«Area Under the Curve»)* permite resumir la curva CMC en un sólo valor.

Conjunto de datos Conjunto de datos que se van a utilizar. En este proyecto, hace referencia a los conjuntos de imágenes de personas que se van a utilizar. Un conjunto de datos tiene que estar compuesto al menos de imágenes tomadas desde dos cámaras distintas, en las que aparezcan los mismos individuos desde diferentes cámaras.

SingleShot Este es un tipo de conjunto de datos en el que se encuentra una única imagen por individuo para cada cámara. Una en el conjunto de muestra y otra en el conjunto de la galería, si sólo hay dos cámaras.

MultiShot Al contrario que en el caso SingleShot, en este caso se pueden encontrar múltiples imágenes de un mismo individuo desde una misma cámara.

Muestra Conjunto que se toma como referencia para realizar la reidentificación. Se intenta buscar una correspondencia para los elementos de este conjunto en otro conjunto, la galería. A veces se utiliza para referirse a un único elemento de este conjunto.

Galería Conjunto en el que se buscan las correspondencias. Para cada elemento de la muestra se busca su correspondencia en este conjunto.

Histograma Representación probabilística de una distribución numérica. En el proyecto se utilizan con los espacios de color. Según el número de bins representan la cantidad de píxeles que entran en un determinado rango.

Bin En los histogramas, representan el número de intervalos en el que se divide la distribución original.

Weak negative Usado cuando se habla de reordenación, se trata de un elemento que el usuario ha seleccionado como similar a la imagen de la muestra, sin ser la correspondencia correcta.

Strong negative Similar a Weak negative, pero en este caso el usuario indica que se trata de una imagen que no es similar al elemento de la muestra para el que se busca la correspondencia.

Clustering Es la tarea de agrupar un conjunto de objetos de tal forma que los objetos de un mismo grupo (cluster) son más similares entre ellos que con los elementos de otro grupo. Se utiliza en la reordenación, intentando aprovecharlo para mejorar los resultados.

Reidentificación de personas Dada una imagen o vídeo de una persona tomada desde una cámara, la reidentificación es el proceso de identificar a esta misma persona en imágenes/vídeos tomadas por otra cámara diferente.

Interfaz gráfica de usuario Se trata de un entorno gráfico que facilita a un usuario interactuar con un programa, en este caso, usar PyReID directamente sin necesidad de ser un desarrollador.

Reordenación Proceso por el cual se modifica una ordenación inicial previa, con intención de mejorarla. En este proyecto se intenta mejorar los resultados usando información facilitada por un usuario.

Ordenación Inicial Cuando se habla de proceso de reordenación, se considera la ordenación inicial como la ordenación realizada previamente, sobre la cual se realizará el proceso de reordenación.

Profiling Proceso en el cual se intenta detectar las partes más lentas del código. En este proyecto se utiliza para optimizar aquellas funciones en las que se tarda más.

Bibliografía

[An et al 2013] An, Le ; Kafai, M. ; Yang, Songfan ; Bhanu, B.: Reference-based person re-identification. In: *Advanced Video and Signal Based Surveillance (AVSS), 2013 10th IEEE International Conference on*, Aug 2013, p. 244–249

[Bąk et al 2012] Bąk, Sławomir ; Charpiat, Guillaume ; Corvée, Etienne ; Brémond, François ; Thonnat, Monique: *Learning to Match Appearances by Correlations in a Covariance Metric Space*. Volume 7574. p. 806–820. In: Fitzgibbon, Andrew (Editor) ; Lazebnik, Svetlana (Editor) ; Perona, Pietro (Editor) ; Sato, Yoichi (Editor) ; Schmid, Cordelia (Editor): *Computer Vision – ECCV 2012* Volume 7574, Springer Berlin Heidelberg, 2012. – URL `http://dx.doi.org/10.1007/978-3-642-33712-3_58`

[Bazzani et al 2014] Bazzani, Loris ; Cristani, Marco ; Murino, Vittorio: *SDALF: Modeling Human Appearance with Symmetry-Driven Accumulation of Local Features*. p. 43–69. In: Gong, Shaogang (Editor) ; Cristani, Marco (Editor) ; Yan, Shuicheng (Editor) ; Loy, Chen C. (Editor): *Person Re-Identification*, Springer London, 2014. – URL `http://dx.doi.org/10.1007/978-1-4471-6296-4_3`

[Bedagkar-Gala and Shah 2014] Bedagkar-Gala, Apurva ; Shah, Shishir K.: Editor's Choice Article: A Survey of Approaches and Trends in Person Re-identification. In: *Image Vision Comput.* 32 (2014), april, Nr. 4, p. 270–286. – URL `http://dx.doi.org/10.1016/j.imavis.2014.02.001`. – ISSN 0262-8856

[Behnel et al 2011] Behnel, S. ; Bradshaw, R. ; Citro, C. ; Dalcin, L. ; Seljebotn, D.S. ; Smith, K.: Cython: The Best of Both Worlds. In:

Computing in Science Engineering 13 (2011), March, Nr. 2, p. 31–39. – URL http://cython.org/. – ISSN 1521-9615

[Bhattacharyya 1946] Bhattacharyya, A.: On a Measure of Divergence between Two Multinomial Populations. In: *Sankhyā: The Indian Journal of Statistics (1933-1960)* 7 (1946), Nr. 4, p. pp. 401–406. – URL http://www.jstor.org/stable/25047882. – ISSN 00364452

[Bradski 2000] Bradski, G.: The OpenCV Library. In: *Dr. Dobb's Journal of Software Tools* (2000). – URL http://opencv.org/

[Breiman 2001] Breiman, Leo: Random forests. In: *Machine learning* 45 (2001), Nr. 1, p. 5–32

[Chong et al 2008] Chong, Hamilton Y. ; Gortler, Steven J. ; Zickler, Todd: A Perception-based Color Space for Illumination-invariant Image Processing. In: *ACM SIGGRAPH 2008 Papers*. New York, NY, USA : ACM, 2008 (SIGGRAPH '08), p. 61:1–61:7. – URL http://doi.acm.org/10.1145/1399504.1360660. – ISBN 978-1-4503-0112-1

[Deselaers et al 2004] Deselaers, Thomas ; Keysers, Daniel ; Ney, Hermann: *Features for Image Retrieval: A Quantitative Comparison*. Volume 3175. p. 228–236. In: Rasmussen, CarlEdward (Editor) ; Bülthoff, HeinrichH. (Editor) ; Schölkopf, Bernhard (Editor) ; Giese, MartinA. (Editor): *Pattern Recognition* Volume 3175, Springer Berlin Heidelberg, 2004. – URL http://dx.doi.org/10.1007/978-3-540-28649-3_28

[Farenzena et al 2010] Farenzena, M. ; Bazzani, L. ; Perina, A. ; Murino, V. ; Cristani, M.: Person re-identification by symmetry-driven accumulation of local features. In: *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, June 2010, p. 2360–2367. – ISSN 1063-6919

[Geng et al 2013] Geng, Yanbing ; Hu, Hai-Miao ; Zheng, Jin ; Li, Bo: A person re-identification algorithm by using region-based feature selection and feature fusion. In: *Image Processing (ICIP), 2013 20th IEEE International Conference on*, Sept 2013, p. 3363–3366

[GitHub 2015] GitHub, Inc.: *GitHub: Build software better, together*. 2015. – URL https://github.com/

[Gray et al 2007] Gray, Doug ; Brennan, Shane ; Tao, Hai: Evaluating appearance models for recognition, reacquisition, and tracking. In: *In IEEE International Workshop on Performance Evaluation for Tracking and Surveillance, Rio de Janeiro*, 2007

[Gray and Tao 2008] Gray, Douglas ; Tao, Hai: *Viewpoint Invariant Pedestrian Recognition with an Ensemble of Localized Features*. Volume 5302. p. 262–275. In: Forsyth, David (Editor) ; Torr, Philip (Editor) ; Zisserman, Andrew (Editor): *Computer Vision – ECCV 2008* Volume 5302, Springer Berlin Heidelberg, 2008. – URL http://dx.doi.org/10.1007/978-3-540-88682-2_21

[Greig et al 1989] Greig, D. M. ; Porteous, B. T. ; Seheult, A. H.: Exact maximum a posteriori estimation for binary images. (1989)

[Grossberg and Nayar 2003] Grossberg, M.D. ; Nayar, S.K.: Determining the camera response from images: what is knowable? In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 25 (2003), Nov, Nr. 11, p. 1455–1467. – ISSN 0162-8828

[Hunter 2007] Hunter, J.D.: Matplotlib: A 2D Graphics Environment. In: *Computing in Science Engineering* 9 (2007), May, Nr. 3, p. 90–95. – URL http://matplotlib.org/. – ISSN 1521-9615

[Javed et al 2005] Javed, O. ; Shafique, K. ; Shah, M.: Appearance modeling for tracking in multiple non-overlapping cameras. In: *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on* Volume 2, June 2005, p. 26–33 vol. 2. – ISSN 1063-6919

[Javed et al 2008] Javed, Omar ; Shafique, Khurram ; Rasheed, Zeeshan ; Shah, Mubarak: Modeling inter-camera space–time and appearance relationships for tracking across non-overlapping views. In: *Computer Vision and Image Understanding* 109 (2008), Nr. 2, p. 146 – 162. – URL http://www.sciencedirect.com/science/article/pii/S1077314207000100. – ISSN 1077-3142

[JetBrains 2015] JetBrains: *PyCharm: The Most Intelligent Python IDE*. Septiembre 2015. – URL https://www.jetbrains.com/pycharm/

[Joan.domenech91 2011] Joan.domenech91: *Validación cruzada aleatoria*. December 2011. – URL `https://commons.wikimedia.org/wiki/File:Random_cross_validation.jpg`

[Jojic et al 2009] Jojic, N. ; Perina, A. ; Cristani, M. ; Murino, V. ; Frey, B.: Stel component analysis: Modeling spatial correlations in image class structure. In: *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, June 2009, p. 2044–2051. – ISSN 1063-6919

[Jones et al 2001–] Jones, Eric ; Oliphant, Travis ; Peterson, Pearu et al: *SciPy: Open source scientific tools for Python*. 2001–. – URL `http://www.scipy.org/`. – [Online; accessed 2015-09]

[Kierano 2015] Kierano: *Cumulative vs normal histogram.svg*. 2015. – URL `https://upload.wikimedia.org/wikipedia/commons/5/53/Cumulative_vs_normal_histogram.svg`

[Kirupa.com 2015] Kirupa.com: *RGB*. 2015. – URL `http://www.kirupa.com/design/little_about_color_hsv_rgb.htm`

[Kostinger et al 2012] Kostinger, M. ; Hirzer, M. ; Wohlhart, P. ; Roth, P.M. ; Bischof, H.: Large scale metric learning from equivalence constraints. In: *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, June 2012, p. 2288–2295. – ISSN 1063-6919

[Liu et al 2013] Liu, Chunxiao ; Loy, C.C. ; Gong, Shaogang ; Wang, Guijin: POP: Person Re-identification Post-rank Optimisation. In: *Computer Vision (ICCV), 2013 IEEE International Conference on*, Dec 2013, p. 441–448. – ISSN 1550-5499

[Marín Reyes 2015] Marín Reyes, Pedro A.: *Estudio comparativo de medidas de distancia para histogramas en problemas de reidentificación*. http://hdl.handle.net/10553/12759, Instituto de Sistemas Inteligentes y Aplicaciones Numéricas en Ingeniería (SIANI), Master thesis, 2015

[MIT 2015] MIT, Massachusetts Institute of T.: *HSV Cone*. 2015. – URL `http://courses.csail.mit.edu/6.831/2014/readings/L16-color/`

[Moosmann et al 2007] Moosmann, Frank ; Triggs, Bill ; Jurie, Frederic: Fast discriminative visual codebooks using randomized clustering

forests. In: *Twentieth Annual Conference on Neural Information Processing Systems (NIPS'06)* MIT Press (event), 2007, p. 985–992

[Nintex Labs 2015] Nintex Labs: *TDD*. 2015. – URL `http://labs.nintex.com/code-dojo-test-driven-development/`

[Pedregosa et al 2011] Pedregosa, F. ; Varoquaux, G. ; Gramfort, A. ; Michel, V. ; Thirion, B. ; Grisel, O. ; Blondel, M. ; Prettenhofer, P. ; Weiss, R. ; Dubourg, V. ; Vanderplas, J. ; Passos, A. ; Cournapeau, D. ; Brucher, M. ; Perrot, M. ; Duchesnay, E.: Scikit-learn: Machine Learning in Python. In: *Journal of Machine Learning Research* 12 (2011), p. 2825–2830. – URL `http://scikit-learn.org/stable/`

[Pele and Werman 2010] Pele, Ofir ; Werman, Michael: *The Quadratic-Chi Histogram Distance Family*. Volume 6312. p. 749–762. In: Daniilidis, Kostas (Editor) ; Maragos, Petros (Editor) ; Paragios, Nikos (Editor): *Computer Vision – ECCV 2010* Volume 6312, Springer Berlin Heidelberg, 2010. – URL `http://dx.doi.org/10.1007/978-3-642-15552-9_54`

[Pérez and Granger 2007] Pérez, Fernando ; Granger, Brian E.: IPython: a System for Interactive Scientific Computing. In: *Computing in Science and Engineering* 9 (2007), may, Nr. 3, p. 21–29. – URL `http://ipython.org`. – ISSN 1521-9615

[Prosser et al 2008] Prosser, B. ; Gong, S. ; Xiang, T.: Multi-camera Matching using Bi-Directional Cumulative Brightness Transfer Functions. In: *Proceedings of the British Machine Vision Conference*, BMVA Press, 2008, p. 64.1–64.10. – doi:10.5244/C.22.64. – ISBN 1-901725-36-7

[PSF 2015] PSF: *Python*. Septiembre 2015. – URL `https://www.python.org/`

[Rother et al 2004] Rother, Carsten ; Kolmogorov, Vladimir ; Blake, Andrew: GrabCut -Interactive Foreground Extraction using Iterated Graph Cuts. In: *ACM Transactions on Graphics (SIGGRAPH)* (2004), August. – URL `http://research.microsoft.com/apps/pubs/default.aspx?id=67890`

[Saghafi et al 2014] Saghafi, M.A. ; Hussain, A. ; Zaman, H.B. ; Md Saad, M.H.: Review of person re-identification techniques. In: *Computer Vision, IET* 8 (2014), Nr. 6, p. 455–474. – ISSN 1751-9632

[Statistics 2015] Statistics, Laerd: *https://statistics.laerd.com/statistical-guides/understanding-histograms.php*. 2015

[Swain and Ballard 1991] Swain, MichaelJ. ; Ballard, DanaH.: Color indexing. In: *International Journal of Computer Vision* 7 (1991), Nr. 1, p. 11–32. – URL `http://dx.doi.org/10.1007/BF00130487`

[Tao et al 2013] Tao, Dapeng ; Jin, Lianwen ; Wang, Yongfei ; Yuan, Yuan ; Li, Xuelong: Person Re-Identification by Regularized Smoothing KISS Metric Learning. In: *Circuits and Systems for Video Technology, IEEE Transactions on* 23 (2013), Oct, Nr. 10, p. 1675–1685. – ISSN 1051-8215

[The LyX Team 2009] The LyX Team: *LyX 1.6.1 - The Document Processor [Computer software and manual]*. Internet: http://www.lyx.org. 2009. – URL `http://www.lyx.org/`. – Retrieved February 16, 2009, from http://www.lyx.org

[Toggl OÜ 2015] Toggl OÜ: *Toggl: The ultimate Timer. It's insanely simple.* Septiembre 2015. – URL `https://www.toggl.com/`

[Torvalds 2005] Torvalds, Linus: *git –distributed-is-the-new-centralized*. 2005. – URL `https://git-scm.com/`

[Trello Inc. 2015] Trello Inc.: *Trello es la manera gratuita, flexible y visual de organizarlo todo con cualquiera.* 2015. – URL `https://trello.com/`

[van der Walt et al 2011] Walt, S. van der ; Colbert, S.C. ; Varoquaux, G.: The NumPy Array: A Structure for Efficient Numerical Computation. In: *Computing in Science Engineering* 13 (2011), March, Nr. 2, p. 22–30. – URL `http://www.numpy.org/`. – ISSN 1521-9615

[Wang et al 2013] Wang, Yimin ; Hu, Ruimin ; Liang, Chao ; Zhang, Chunjie ; Leng, Qingming: Camera compensation using feature projection matrix for person re-identification. In: *Multimedia and Expo (ICME), 2013 IEEE International Conference on*, July 2013, p. 1–6. – ISSN 1945-7871

[Wang et al 2014] Wang, Zheng ; Hu, Ruimin ; Liang, Chao ; Leng, Qingming ; Sun, Kaimin: *Region-Based Interactive Ranking Optimization for Person Re-identification*. Volume 8879. p. 1–10. In: Ooi, WeiTsang (Editor) ; Snoek, CeesG.M. (Editor) ; Tan, HungKhoon (Editor) ; Ho, Chin-Kuan (Editor) ; Huet, Benoit (Editor) ; Ngo, Chong-Wah (Editor): *Advances in Multimedia Information Processing – PCM 2014* Volume 8879, Springer International Publishing, 2014. – URL http://dx.doi.org/10.1007/978-3-319-13168-9_1

[Wei and Lin 2013] Wei, Yu-Lun ; Lin, Chang H.: Single-shot person re-identification by Gaussian mixture model of weighted color histograms. In: *Intelligent Signal Processing and Communications Systems (ISPACS), 2013 International Symposium on*, Nov 2013, p. 47–50

[Zhu and Ghahramani 2002] Zhu, Xiaojin ; Ghahramani, Zoubin: Learning from Labeled and Unlabeled Data with Label Propagation. 2002. – Research Report