Comparison of Accumulative Computation with Traditional Optical Flow

Antonio Fernández-Caballero, Rafael Pérez-Jiménez, Miguel A. Fernández, and María T. López

Departmento de Sistemas Informáticos, Universidad de Castilla-La Mancha
Escuela Politécnica Superior de Albacete, Albacete, Spain
caballer@dsi.uclm.es

Abstract. Segmentation from optical flow calculation is nowadays a well-known technique for further labeling and tracking of moving objects in video streams. A likely classification of algorithms to obtain optical flow based on the intensity of the pixels in an image is in (a) differential or gradient-based methods and (b) block correlation or block matching methods. In this article, we are going to carry out a qualitative comparison of three well-known algorithms (two differential ones and a correlation one). We will do so by means of the optical flow obtaining method based on accumulated image differences known as accumulative computation.

Keywords: Optical flow, Accumulative computation method, Image difference.

1 Introduction

One of the most interesting and productive techniques in the field of image sequence motion analysis is the technique known as optical flow [7]. Indeed, segmentation from optical flow calculation is nowadays a well-known technique for further labeling and tracking [10],[15],[17],[11] of moving objects in video streams, as motion is a major information source for segmenting objects perceived in dynamic scenes. Optical flow can be defined as the apparent displacement of the pixels in the image when there is relative motion between the camera and the objects under focus. Another possible definition is considering optical flow as the 2-D motion field obtained from the projection of the velocities of the three dimensional pixels, corresponding to the surfaces of a scene, onto the sensor's visual plane [8].

A possible algorithm classification to obtain optical flow [2] based on pixel intensity in the image would be (a) differential methods and (b) block correlation methods (matching). In this article, we are going to carry out a qualitative comparison of three well-known algorithms with our optical flow obtaining method, known as accumulative computation [6],[16]. Our method presents a new way of looking at optical flow and describes it as a measure of the time elapsed since the last significant change in the brightness level of each pixel in the image [12].

B. Apolloni et al. (Eds.): KES 2007/WIRN 2007, Part I, LNAI 4692, pp. 447–454, 2007.

Differential Methods are also called gradient-based methods. These techniques calculate the flow from the space-time derivatives of the intensities in the image, through the expression known as the brightness constancy equation of the optical flow computation. This method has become the most frequent approximation used in computer vision applications because of its swiftness and its good velocity estimation.

Horn and Schunck [9] propose a method based on first order derivatives and add a smoothness condition on the flow vectors to the general conditions. They assume that object motion in a sequence will be rigid and approximately constant, that a pixel's neighborhood in said objects will have similar velocity, therefore, changing smoothly over space and time. Nevertheless, this condition is not very realistic in many cases and it yields bad results [14] since the images' flow has a lack of continuity, especially in the boundaries between different objects. Therefore the results obtained in these areas will not be correct. Poor results are also obtained in the sequences where there are multiple objects, each having different motion.

Barron, Fleet and Beauchemin [2] suggest a modification where a Gaussian space-time pre-smoothing is done to the images and where the derivatives are calculated using the differential method with a coefficient mask. On the other hand, they introduce a gradient thresholding method in algorithm implementation and decide whether a velocity will be accepted or rejected. This decision is taken based on the gradient module; when it does not exceed the threshold value, the velocity in said pixel will be rejected. A great number of inaccurate results can be eliminated this way.

Lucas and Kanade's algorithm [13] is similar to Horn and Schunck's. Horn and Schunck use a global approach, whereas Lucas and Kanade use a local approach to an environment. The algorithm devised by Lucas and Kanade adds a flow smoothing constraint in local neighborhoods to the intensity conservation restraint. The method expects the velocities to be constant in a relatively small environment basing this on the fact that it is logical to expect pixels from the same object to have identical velocities.

Block Correlation Methods, also known as a block matching-based method, assume that the distribution of the intensity for the region which surrounds the pixel, whose motion is to be evaluated, is maintained. Thus, for each pixel whose flow is to be computed at a certain time, a window of pixels which surrounds that pixel is created. The purpose, in the following time, is to look for the maximum correspondence between said window and a set of windows of equal resolution within a neighborhood defined by a higher window, called a search a window in the following time.

Anandan's algorithm [1] fits into the matching methods. It proposes that, in a discrete case, the sum of the squared differences (SSD) is closely related to the correlation coefficient. To attain sub-pixel accuracy and to avoid problems due to aperture or great displacements, Anandan used a hierarchical scheme based on Gaussian or Laplacian pyramids, estimating velocity from the lowest to the highest resolution level. This way, sub-pixel displacements are estimated in

two different phases. Anandan also proposes smoothing the resulting velocities, since it is expected for velocities present in a sequence to be fairly homogenous. In the final algorithm, a matching and smoothing of the resulting velocities is carried out for each level in the pyramid created, from the lowest to the highest resolution level.

2 Optical Flow Through Accumulative Computation

Accumulative computation is based on the allocation of charge levels assigned to every image pixel related to the history of a studied feature of the pixel. The general formula which represents the charge in an image pixel, due to accumulative computation [5],[4] is:

$$Ch[x,y,t] = \begin{cases} \min(Ch[x,y,t-\Delta t]+C, Ch_{max}), \text{ if "}property\ is\ fulfilled\text{"} \\ \max(Ch[x,y,t-\Delta t]-D, Ch_{min}), \text{ otherwise} \end{cases} \tag{1}$$

In the LSR mode of operation (length-speed ratio) [3], $C = C_{Mov}$ is called a charge increase value. The idea behind is that if there is no motion in pixel (x, y), which is estimated as a change in the grey level between two consecutive times, charge value $Ch[x, y, t]$ increases up to a maximum value Ch_{max}. And if there is motion, there is a complete discharge (a minimum value Ch_{min} is assigned). In general, Ch_{max} and Ch_{min} take values of 255 and 0, respectively. Notice that charge value $Ch[x, y, t]$ represents a measure of the time elapsed since the last significant change in the image pixel's (x, y) brightness.

$$Ch[x,y,t] = \begin{cases} Ch_{min}, \text{ if "}motion\ is\ detected\ in\ (x,y)\ in\ t\text{"} \\ \min(Ch[x,y,t-1]+C_{Mov}, Ch_{max}), \\ \qquad \text{otherwise} \end{cases} \tag{2}$$

Once the image's charge map is obtained for the current time t, the optical flow considered as the velocity estimated from the stored charge values is obtained as detailed next. (1) $Ch[x, y, t] = Ch_{min}$: Motion is detected in pixel (x, y) in t. The map's value is the minimum charge value. (2) $Ch[x, y, t] = Ch_{min} + k \cdot C < Ch_{max}$: Motion in pixel (x, y) is not detected in t. Motion was last detected in $t - k \cdot \Delta t$. After k increments, the maximum charge has not yet been reached. (3) $Ch[x, y, t] = Ch_{max}$: Motion is not detected in pixel (x, y) in t. It is not known when motion was last detected. The map's value is the maximum charge value.

It is important to point out that the velocity obtained by these means is not the velocity of an object pixel, which is occupied by pixel (x, y) in time t, but the velocity of an object pixel responsible for motion detection when it went by pixel (x, y) $k = \dfrac{Ch[x,y,t] - Ch_{min}}{C_{Mov}}$ units of time ago. Therefore, a given charge has the same value in all pixels where motion was detected at the same time.

Now then, velocity is calculated in axis x, v_x, as well as in axis y, v_y. To calculate velocity in x, the charge value in (x, y), which an object is currently crossing, is compared to the charge value of another coordinate in the same image

row $(x+l, y)$, where the same object is crossing. At best, that is when both values are different to Ch_{max}, the time elapsed from the last motion detection in (x, y) to the time when motion is detected in $t - k_{(x+l,y)} \cdot \Delta t$ en $(x + l, y)$ can be calculated as:

$$\begin{aligned} &Ch[x, y, t] - Ch[x + l, y, t] = \\ &= (Ch_{min} + k_{(x,y)} \cdot C_{Mov}) - (Ch_{min} + k_{(x+l,y)} \cdot C_{Mov}) = \\ &= (k_{(x,y)} - k_{(x+l,y)}) \cdot C_{Mov} \end{aligned} \tag{3}$$

Obviously, this cannot be calculated if either of the values is equal to Ch_{max}, since it is not known how many time intervals have elapsed since the last motion detection. Therefore, for valid charge values, we have:

$$\Delta t = \frac{(k_{(x,y)} - k_{(x+l,y)}) \cdot C_{Mov}}{C_{Mov}} = k_{(x,y)} - k_{(x+l,y)} \tag{4}$$

From equations (3) and (4):

$$\Delta t = \frac{Ch[x, y, t] - Ch[x + l, y, t]}{C_{Mov}} \tag{5}$$

Since $v_x[x, y, t] = \dfrac{\delta x}{\delta t} = \dfrac{l}{\Delta t}$, we finally have:

$$v_x[x, y, t] = \frac{C_{Mov} \cdot l}{Ch[x, y, t] - Ch[x + l, y, t]} \tag{6}$$

Velocity is calculated in the same way in y from values stored as charges:

$$v_y[x, y, t] = \frac{C_{Mov} \cdot l}{Ch[x, y, t] - Ch[x, y + l, t]} \tag{7}$$

3 Data and Results

Once the methods have been described, we go on to present the results obtained in the qualitative comparison of the different algorithms: (a) Barron, Fleet and Beauchemin, (b) Lucas and Kanade, (c) Anandan and (d) accumulative computation. For this experimental level comparison, different image sequences have been selected for each algorithm. The results show in a qualitative manner those pixels where some velocity different from zero is obtained.

Yosemite Sequence. This is a complex case in the synthetic sequence bank used in numerous benchmarks. It shows a virtual flight over the Yosemite valley. The clouds on the upper right of the image move at a velocity of 2 pixels/frame from left to right. The rest of the flow is divergent, with velocities of up to 5 pixels/frame in the lower left corner. This is an interesting sequence since it displays different types of motion, slightly different boundaries and it can resemble a real situation. In Fig. 1, we see the result of applying each of the four

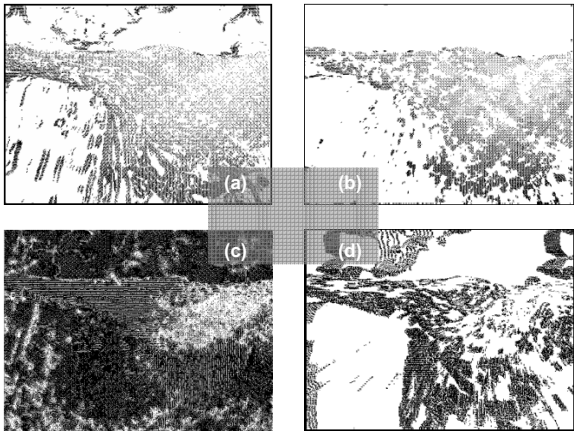

Fig. 1. Results obtained in the Yosemite sequence. (a) Barron, Fleet and Beauchemin's (BFB) method. (b) Lucas and Kanade's (LK) method. (c) Anandan's (A) method. (d) Accumulative computation (AC) method.

methods to the Yosemite sequence. In the first place, we are struck by Anandan's method's poor performance. It detects much more (and inaccurate) flow than other methods. We can also verify that both Barron's and the accumulative computation methods are able to detect cloud motion as opposed to Lucas and Kanade's which cannot.

Hamburg Taxi Sequence. This sequence is a classic in the computer vision field. There are four objects in motion: (1) the white taxi turning the corner, (2) a dark car in the lower left corner, moving from left to right, (3) a van, also dark, moving from right to left, and, (4) a pedestrian, who is fairly far away from the camera, in the upper left corner. In the foreground and slightly to the right, we see tree branches. The approximate velocity for each object is: 1.0, 3.0, 3.0 and 0.3 pixels/frame, respectively. The fields obtained for the Taxi sequence (Fig. 2) in general show all the displacements mentioned in its description, with the exception of the pedestrian's movement which can only be obtained with accumulative computation-based algorithm. This method also "outlines" objects better than others. In every case there is a lot of noise in the scene. We are also struck by the fact that the vehicles are not excessively well segmented (this would belong to an advanced level analysis). The vehicle closest to the right hand side is detected the worst because it is partially hidden by part of a tree.

Rubik's Cube Sequence. Another well-known sequence is this Rubik's cube rotating counter-clockwise. The velocity field caused by the cube's rotation is less than 2 pixels/frame. The surface on which the cube is placed has a motion between 1.2 and 1.4 pixels/frame. Good results are obtained, in general, in the Rubik's cube sequence (Fig. 3), obtaining the velocity of the cube's sides as well

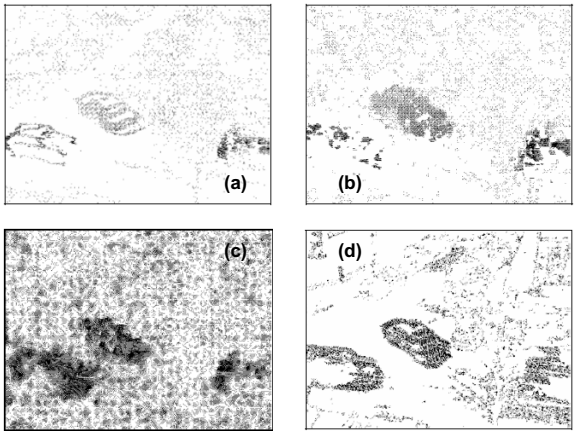

Fig. 2. Results obtained in the Hamburg Taxi sequence. (a) BFB method. (b) LK method. (c) A method. (d) AC method.

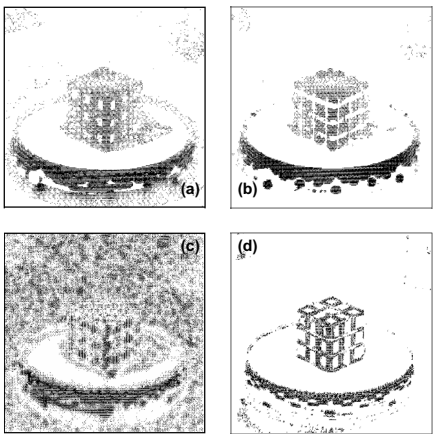

Fig. 3. Results obtained for the Rubik's Cube sequence. (a) BFB method. (b) LK method. (c) A method. (d) AC method.

as that of the rotary base. The cube's shadow motion is detected in Barron-Fleet-Beauchemin's and Anandan's algorithms and it is best filtered with Lucas and Kanade's method and with the accumulative computation-based method. The latter algorithms offer the best results for this sequence, qualitatively speaking. We see at a first glance that the accumulative computation algorithm eliminates the most noise from the scene. Again, Anandan offers poor results.

SRI Trees Sequence. This time, the camera moves from right to left, parallel to the plane in front of the group of trees. This is a complex sequence since it has a great number of occlusions, as well as low resolution. The velocities are

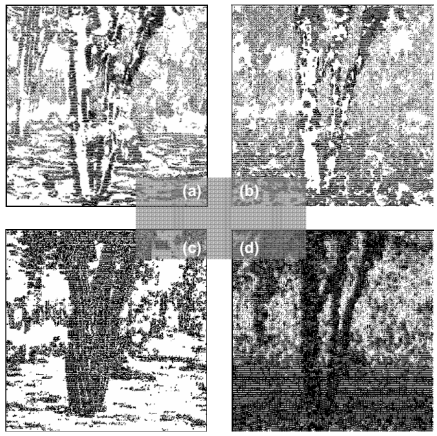

Fig. 4. Results obtained for the SRI Tress sequence. (a) BFB method. (b) LK method. (c) A method. (d) AC method.

greater than 2 pixels/frame. The SRI Trees sequence is very complex. Barron et al's algorithm performs better than the rest since it outlines the trees (Fig. 4). Other methods seem to be inefficient when working with movable cameras on static scenes. This is essentially so in the accumulative computation method.

4 Conclusion

In this work, we have presented a qualitative comparison of the different traditional optical flow computation methods with our new accumulative computation technique. Other methods have high computational costs as opposed to our accumulative computation method, based on simple additions and subtractions.

In this paper, accumulative computation is based on the allocation of charge levels assigned to every image pixel related to the history of motion presence detection of the pixel. Our accumulative computation method is new in the sense that it calculates the optical flow as a measure of the elapsed time since the last significant change in the brightness level for each pixel in the image.

In the results obtained in the segmentation of the shape of figures due to the motion inherent to he camera capture, we see that for most of the sequences tested, specifically Yosemite and Hamburg Taxi, the accumulative computation method offers similar or better quality than the other methods. We are currently working to offer a quantitative comparison of the results obtained, with regard to execution time and success rate in the results.

Acknowledgements

This work is supported in part by the Spanish CICYT TIN2004-07661-C02-02 grant and the Junta de Comunidades de Castilla-La Mancha PBI06-0099 grant.

References

1. Anandan, P.: A computational framework and an algorithm for the measurement of visual motion. International Journal of Computer Vision 2, 283–310 (1989)
2. Barron, J.L., Fleet, D.J., Beauchemin, S.S.: Performance of optical flow techniques. International Journal of Computer Vision 12(1), 43–77 (1994)
3. Fernández, M.A., Fernández-Caballero, A., López, M.T., Mira, J.: Length-speed ratio (LSR) as a characteristic for moving elements real-time classification. Real-Time Imaging 9, 49–59 (2003)
4. Fernandez, M.A., Mira, J., Lopez, M.T., et al.: Local accumulation of persistent activity at synaptic level: application to motion analysis. From Natural to Artificial Neural Computation, 137–143 (1995)
5. Fernández, M.A., Mira, J.: Permanence memory - A system for real time motion analysis in image sequences. In: Proceedings of the IAPR Workshop on Machine Vision Applications, pp. 249–252 (1992)
6. Fernández-Caballero, A., Fernández, M.A., Mira, J., Delgado, A.E.: Spatio-temporal shape building from image sequences using lateral interaction in accumulative computation. Pattern Recognition 36(5), 1131–1142 (2003)
7. Gibson, J.J.: The Perception of the Visual World. Houghton Mifflin (1950)
8. Horn, B.K.P.: Robot Vision. MIT Press, Cambridge (1986)
9. Horn, B.K.P., Schunck, B.G.: Determining optical flow. Artificial Intelligence 17, 185–204 (1981)
10. Liang, K.H., Tjahjadi, T.: Multiresolution segmentation of optical flow fields for object tracking. Applied Signal Processing 4(3), 179–187 (1998)
11. Lodato, C., Lopes, S.: An optical flow based segmentation method for objects extraction. Transactions on Enginnering, Comuting and Technlogy 12, 41–46
12. López, M.T., Fernández-Caballero, A., Fernández, M.A., Mira, J., Delgado, A.E.: Motion features to enhance scene segmentation in active visual attention. Pattern Recognition Letters 27(5), 469–478 (2005)
13. Lucas, B., Kanade, T.: An iterative image registration technique with an application to stereo vision. In: Proceedings of the DARPA IU Workshop, pp. 121–130 (1981)
14. Lucena, M.: Uso del flujo óptico en algoritmos probabilísticos de seguimiento. Tesis Doctoral, Departamento de Informática. Universidad de Jaén (2003)
15. Macan, T., Loncaric, S.: Hybrid optical flow and segmentation technique for LV motion detection. Proceedings of SPIE 4321, 475–482 (2001)
16. Mira, J., Delgado, A.E., Fernández-Caballero, A., Fernández, M.A.: Knowledge modelling for the motion detection task: The algorithmic lateral inhibition method. Expert Systems with Applications 2, 169–185 (2004)
17. Zitnick, C.L., Jojic, N., Kang, S.B.: Consistent segmentation for optical flow estimation. In: Proceedings of the Tenth IEEE International Conference on Computer Vision, vol. II, pp. 1308–1315 (2005)