Circle detection and tracking speed-up based on change-driven image processing

Fernando Pardo, Jose A. Boluda, Julio C. Sosa
Departamento de Informática, Universidad de Valencia
Avda. Vicente Andres Estelles s/n, 46.100 Burjassot, Valencia, Spain
[Fernando.Pardo, Jose.A.Boluda]@uv.es

Xaro Benavent, Juan Domingo
Instituto de Robótica, Universidad de Valencia
Avda. Polígono de la Coma s/n, 46.890 Paterna, Valencia, Spain
[Xaro.Benavent, Juan.Domingo]@uv.es

Abstract

Real-time motion analysis is an image processing task that requires high processing power due to the large amount of data involved. An improved strategy for processing image sequences is presented. This new approach takes only the pixels of the image sequence that are significant for the motion estimation algorithm being implemented, reducing the amount of data to be processed and increasing the algorithm speed. This approach has been tested using one classical algorithm for object detection and tracking. The results show that this method is several times faster than the classical approach.

Keywords: *Change-driven image processing, Hough transform, object tracking, circle recognition.*

1 Introduction

Movement analysis is a challenging image processing task, due to the large amount of data to be processed and the real-time requirements to obtain the computed results. One example of image sequence processing, among others, is the tracking of one or more objects in front of a camera. This specific task requires high speed detection of the object and calculation of its position. Object recognition, especially for simple objects, is a task that can be performed in a relatively short time of few seconds. This time is good for many applications, but not for real-time image sequence processing, in which the time constraint is in the order of milliseconds.

Full image processing is usually the classical approach for general image sequence processing. In the case of object recognition and tracking, the normal procedure implies the application of the detection algorithm for each image in the sequence. This classical approach is enough when the detection algorithm or the processing hardware is capable of detecting the object in few milliseconds. Depending on the complexity of the object to be tracked and the implemented algorithm, the processing time can last even seconds for each image in the sequence.

It is possible to reduce the processing time realizing that images usually change little from frame to frame, especially if the acquisition time is short. The reduction is accomplished executing the instructions only for the pixels which have changed between two consecutive images. In classical optical flow computation algorithms, for example, the spatial and temporal derivatives are calculated for all pixels in the image despite the fact that images could have suffered no change at all from one frame to the next.

Paying attention only to those pixels that change is not new and this same principle has been employed to design some image sensors with on-plane compression [2]. These image sensors only deliver the pixels that change, decreasing the amount of data coming from the camera. This data reduction obtained with these sensors has been successfully employed to speed-up some image processing tasks [4].

Some other sensors, most of them biologically inspired, implement a similar strategy but in the space domain instead of the temporal domain. In those sensors the differences among neighbour pixels is computed and in case this difference is above a fixed threshold, the pixel is read out. It also allows the possibility of asynchronous pixel reading [5], which may open the possibility of processing reduction, since only the interesting part of the image, the most relevant, is processed. Other similar sensors acts like the human cells: if the cell receives more light, it takes less time to fire. These sensors fire high illuminated pixels more often than those at darkness [3]; from the image processing point of view the advantage of this characteristic has to still be proved.

A method for speeding-up classical image sequence processing algorithms is presented. This method is based on pixel change instead of full image processing and it shows a good speed-up. This method do not require any special hardware or sensor, though a larger speed-up could be obtained with a specific sensor or processing architecture.

This methodology has been tested using an algorithm for circle tracking using the Hough transform. The method can work for most video processing algorithms, this circle detection was chosen since it involves many of the operations present in motion detection algorithms, like image differentiation, and it requires operations at different pixel neighbourhoods from local to global operations. This algorithm is also difficult to be implemented at video rates, so any improvement in the speed may allow its use in real-time applications.

2 Change-driven processing

Classical algorithms for image processing consist of a sequence of tasks to be performed over an image or image sequence. Any algorithm can be programmed as an instruction sequence. This list of instructions makes a program that can be easily implemented in any processor, since they are based on program counter. This is one of the reasons why little attention has been focused on the direction of making image processing algorithms dependent on data events instead of on the computer program counter, which points which instruction is executed next. In a change-driven processing, or data-driven processing, or data flow architecture [1], data changes fire the instructions to execute, opening also the possibility of parallel execution. Data-flow architectures are better for change-driven problems, but even with standard processor architectures, it is still possible to successfully use the change-driven principle to speed-up some image processing tasks.

Image sequence processing is a kind of computation that greatly benefits from the approach of firing instructions of an algorithm only when data changes, since many times, only few pixels change from frame to frame and usually there is no need to execute any instruction for those pixels that did not change. Many classical algorithms usually perform the same calculation for all the pixels in an image for every image in the sequence, even if the pixel did not change at all from one frame to the next. It is possible to save some calculations if only those pixels that have changed are taken into account to fire the specific instructions.

Classical algorithms must be completely rebuilt to process only those pixels that change from frame to frame. The implementation of this methodology for an image processing algorithm usually requires extra storage to keep track of the intermediate results of preceding computing stages. In this kind of processing, the speed increases as well as the storage needs. Nowadays, storage is not a problem for modern computers, and even embedded systems use to have enough memory for the extra storage required by this methodology.

2.1 Change sensitivity threshold (CST)

In the implementation of any change-driven processing algorithm a new design parameter appears: it is the change sensitivity threshold (CST). This threshold is the pixel difference between images that fire instruction execution: if the value of a pixel changed equally or above this threshold the change is considered significant and the corresponding instructions are fired; otherwise no action is taken.

A similar threshold parameter can be found in most motion detection algorithms, though the meaning in those cases is slightly different. In general motion detection algorithms, object detection is carried out by applying a threshold to the image [8] using some smart threshold calculation depending on the algorithm to be implemented [9]. The CST discussed in this paper is different since it is the threshold that fires local pixel instructions, thus it is not a global threshold for performing high level tasks as object segmentation.

For most cameras, the intensity level of a pixel is given as an 8-bit binary number, ranging from 0 to 255. The minimum intensity difference is 1 in this case. If the CST is set to 1 (minimum difference), the results of the classical and the change-driven algorithms are exactly the same. It is possible to increase the CST so the algorithm becomes less sensitive to image changes and the number of pixels to fire instructions decreases. The higher this CST is, the faster the execution of the algorithm, but the results will be also less accurate. This CST can be considered as a compression parameter that can be adjusted depending on the accuracy and speed required.

Digital cameras are not perfect and it is possible to see intensity level variations between consecutive frames, even with good illumination conditions. Many times the noise observed among images is higher than 1. Setting a CST of only one unit makes no sense for those images, since it is possible to choose a CST close to the average level of noise among images without loosing accuracy. On the other hand, some algorithms are more sensitive to CST than others; some algorithms give the same results even if the CST is set to a very large value compared to the average noise, obtaining a larger speed-up. Other algorithms do not allow this CST enlargement though it is still possible to obtain a high speed-up compared to the classic implementation. It is necessary to perform some tests to find the highest CST suitable for each specific application or algorithm.

3 Circle tracking using the Hough transform

The experiment employed to evaluate the change-driven technique is the tracking of one circle with a specific radius. The algorithm finds a specific circle in the image and gives its position. The Hough transform [6] is employed to find the circle, though there are other algorithms that can find circles for any radius, but with higher processing requirements [7]. The Hough transform has been chosen because it is one of the most widely used, it is simple and reasonably robust against noise and contour occlusions, nevertheless, it is still computationally expensive and as such, it is a good candidate for improvements using change-driven processing.

3.1 Classical algorithm implementation

The classical implementation of the circle detection algorithm, based on the Hough transform, consists in first detecting the edges of the object in the image. After that, an array of votes is built. For each edge point in the image, a circle is added to the array of votes. The superposition of these circles in the array of votes produces the Hough plane. The contour of any circle in the image with the specified radius produces a maximum at this plane. In the real implementation of this algorithm it is possible to see many maxima; some of them belong to the real circle being recognized, but there are other maxima that do not correspond to any circle and are found in images with many edges. One algorithm refinement consists in looking only for maxima where data changes are sharp, which usually only happens for real circles.

The edge detection of the algorithm first stage has been implemented by simply computing spatial derivatives. Afterward, edges are detected if the derivative intensity level at the pixel is above some fixed threshold. Once the edges have been detected, a circle for each pixel belonging to and edge is added to the array of votes. Afterward, maxima are found looking for those places where data change is higher. The classical implementation of this algorithm applies all these steps to every pixel of any image in the sequence.

3.2 Change-driven algorithm implementation

The algorithm is significantly different when change-driven processing is applied. In the first stage of the algorithm, only the spatial derivatives of the pixels that have changed and their neighbours are computed. It is necessary to have extra storage for the previous frame spatial derivatives. The stages of the algorithm are computed for single pixels or single data streams, instead of calculating one complete stage for the full image. The Hough transform is computed right after the spatial derivative of the pixel that has just been computed. Once the spatial derivative has been calculated for the pixel that has changed, the edge condition of that point must be verified; if this point is an edge, and it was already an edge, nothing has to be done; if it was not an edge and now it is not an edge yet, nothing has to be done, but if the condition changes the array of votes (Hough plane) must change accordingly. Therefore, there are four situations depending on the current spatial derivative and the last calculated derivative: if both are above or below the edge threshold, there is no need to calculate the circle for the Hough plane; if the new derivative is above the threshold and the old is below, a new circle must be added to the Hough plane (a new edge point appeared), but if the new derivative is below the threshold and the old is above, the circle must be removed (an edge point disappeared). In this way, the Hough plane is calculated based on the previous and adding and removing circles from it depending only on the pixels that have changed. The difference with the classical algorithm is clear: in the classical implementation the Hough plane is calculated for every image starting each time from an empty array, while the change-driven implementation takes the Hough array of the previous image and changes it depending only on the image pixel changes.

Once the Hough plane has been calculated, the same strategy of change-driven processing is applied to this plane: only those points in the Hough plane that have changed are employed to calculate the final coordinates of the circle. For those points, neighbour differences are calculated to detect maxima. For these maxima there is a similar situation as in the edge detection step: if the new difference value and the old one are above or below the circle detection threshold, nothing changes; but if the new one is above and the old below, there is a new point that must be considered as the possible circle centre being tracked; on the other hand, if the new difference is below the threshold and the old is above, this point must be removed from the point list of possible circle centres. There is a list of possible circle centre coordinates and depending on the last process, coordinates are added or removed from this list. At the end of the process, the calculation of the average of points in the list (after removal of isolated non significant points) gives the calculated coordinates of the circle in the image.

4 Experimental results

The change-driven processing and the classical algorithms have been implemented. Both algorithms yield the same results (correct coordinates of the circle being tracked) but at significantly different execution speeds. The test computer platform has been a PC, with Intel Pentium IV at 2.3 GHz. Both algorithm implementations have been tested using several sequences of a pendulum. Each sequence has been recorded at different distances, thus giving a moving circle of different radius for each image sequence. The reason for this set of sequences is to have different sizes of the object being tracked to see the impact of the object size in the speed-up of the algorithm. Figure 1 shows four images of the last four test sequences, specifically for radius of 24, 35, 46 and 58 pixels. Images have a resolution of 320x240 pixels and have been recorded using a progressive scan digital still camera at 25 fps.

4.1 Execution time analysis

Execution time has been measured for each algorithm. Figure 2 shows a plot with the time required for each algorithm to process one single image. The change-driven algorithm can be implemented with different CST values. The change-driven algorithm with CST=1 is totally equivalent to the classical implementation, nevertheless the value employed for comparison has been CST=5 since this is the average noise measured among images and the results are the same (see speed-up discussion in next section).

For both classical and change-driven algorithms, the faster execution is obtained when the moving object is small. The algorithm based on changes shows an almost linear relationship between the circle radius and the execution time per image. This is so because the operations executed in this case depend, almost exclusively, on the pixels which have changed from one frame to the next. These pixels belong to the circle periphery so their number is linearly related to the circle radius. The execution time of the classical algorithm also increases with the circle radius, but it reaches a maximum and it even decreases in the last sequence (big object). The execution time of the classical algorithm depends on many factors, but especially of the number of edge points in the image and the size of the circles to add to the array of votes. For big circles (last sequence) the number of edge points in the image decreases since the circle has no internal edges and compensates the higher circle radius to be drawn.

4.2 Change-driven speed-up

The speed-up of the change-driven algorithm compared to the classical has been tested at different CST values. The CST test values have been 1, 5, 10 and 15. Figure 3 shows the speed-up for these CST values except for CST=1, since it gives the same results than CST=5. For CST values around 5 and below, the circle coordinates obtained from the change-driven algorithm are the same as those given by the classical. For higher CST values the coordinates were slightly differ-

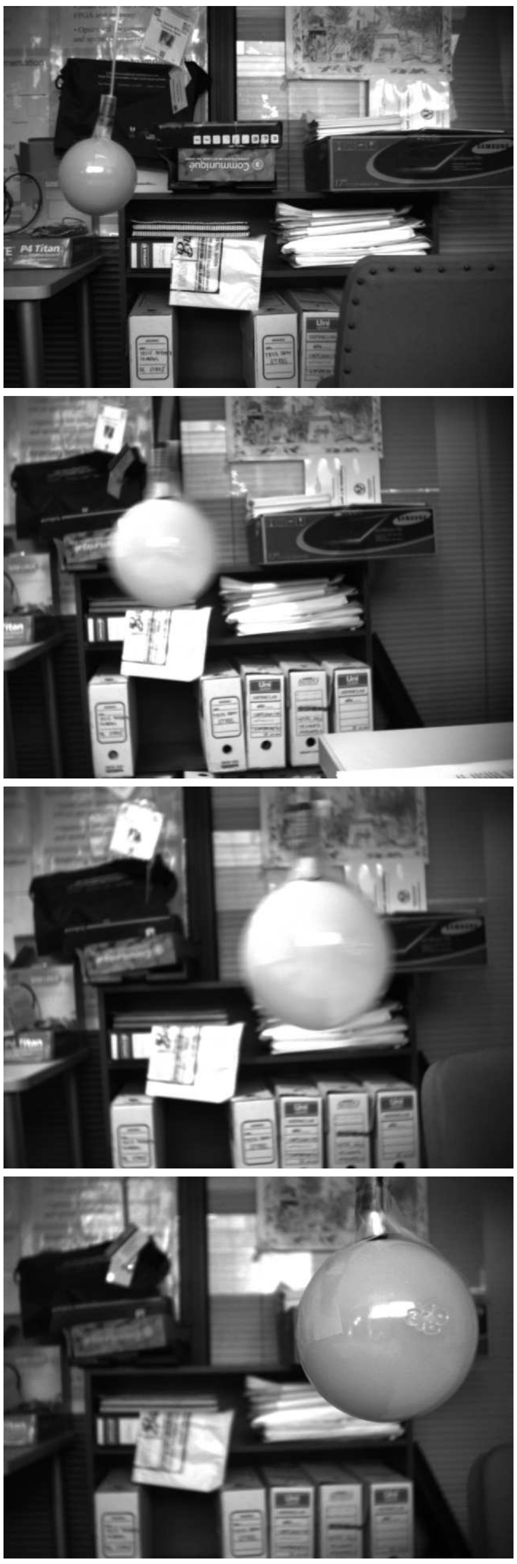
Figure 1: Four images of the last test sequences

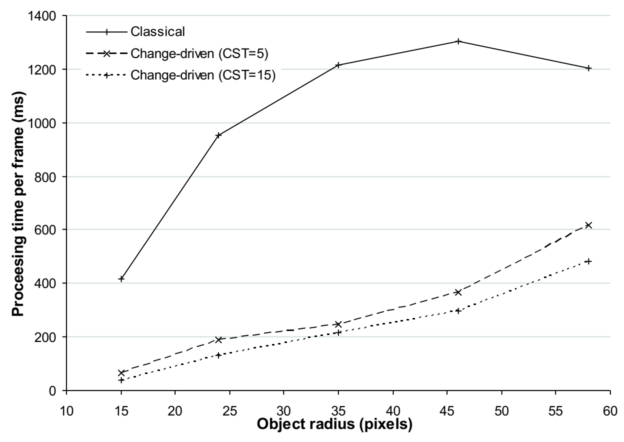

Figure 2: Measured time required for each algorithm to process one image depending on the object radius

ent though they matched the real coordinates of the circle. For values above CST=15 the algorithm starts to give incorrect values for the circle coordinates.

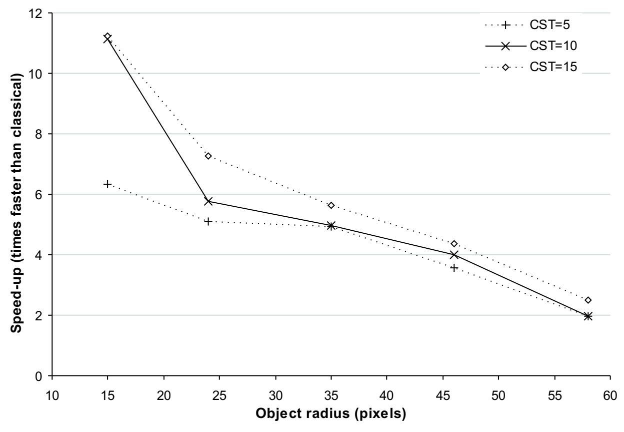

Figure 3: Speed-up obtained with the change-driven processing for several CST

The highest speed-up is obtained setting the highest CST and the smallest object. The evolution is quite similar in all cases. With a CST of 15 the speed-up is always higher, especially for small circles where the speed-up is more than 10 times, while with CST=5 it is around 6. For this experiment, the worst speed-up (lowest CST and largest object) is still two, which means that change-driven processing is at least twice faster than the classical, reaching up to 10 times speed-up for small moving objects.

The plots of the speed-up and execution time with CST=1 have not been included in previous figures since the results are almost the same than CST=5. Figure 4 shows the speed-up for CST=1 and CST=5. In both cases the results are the same except for the smallest circle where the implementation with CST=5 is faster than the implementation with CST=1. The reason is that high CST values filter noise among images and the number of points that change due to noise is usually small compared to the pixels that change due to object movement, except for the case where the object is small and the number of object points is comparable to the number of pixels that changed due to noise.

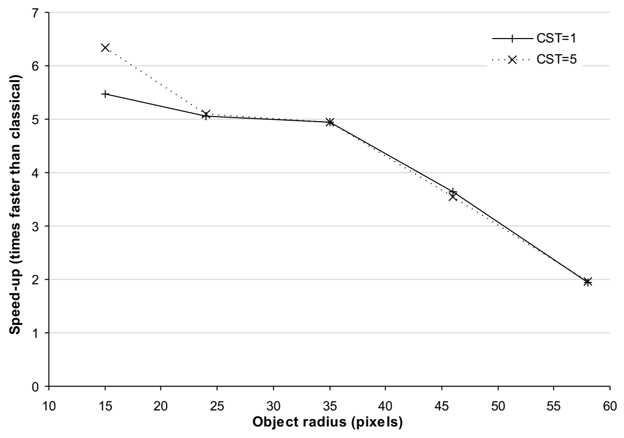

Figure 4: Speed-up comparison for CST=1 and CST=5

4.3 Change-driven algorithm worst case

The experiment of one object tracking while the camera is still is especially good for change-driven processing, since only the object pixels change between two consecutive images. In fact, the smaller the object, the faster the execution. Therefore the question that arises is if the change-driven algorithm would be still good in situations where a significant part of the image changes and not only a small part of it. Another experiment has been performed using a sequence where the camera, instead of the single object, moves. The object to be detected has been the same pendulum, this time still, with radius 40 pixels. The execution time per frame for the classical algorithm has been 667 ms, while the change-driven algorithm took 530 ms. While the speed-up is not very high (20% faster) it is still profitable to use the change-driven processing, even for the case where the whole image changes.

If all pixels change in an image, the change-driven implementation should give a slightly higher processing time than the classical, since it usually requires extra computation. Nevertheless, this last experiment has shown that even when the whole image moves, the change-driven algorithm is faster that the classical for this specific case. The reason is that even when the whole image moves, not all the pixels change, since there could be some objects in the image with a continuous intensity level and, for those objects, a significant change is only found at the object borders.

5 Conclusion

In this article, a procedure for speeding-up the processing of image sequences has been presented. This methodology is based on the data changes from one image to the next. It has been tested using a classical sequence processing task: circle tracking. On this problem, the change-driven algorithm is twice faster than the classical in its worst case, reaching a speed-up of 10 depending on the tracked object size and compression threshold. Even in the case that the whole image changes, this procedure is still advantageous, though the speed-up is not as high as when tracking a single object movement from a still camera. This methodology introduces a new parameter in the processing which allows the adjustment of speed and accuracy; this parameter is the Change Sensitivity Threshold. This CST fixes a compression ratio for the number of instructions to execute and it is directly related to algorithm speed-up. This methodology for classical algorithm image sequence processing implementation is general and can be adapted to many existing algorithms; speed-up is algorithm dependent and must be analyzed case by case.

Acknowledgements

This work has been supported by the project UV-AE-20050206 of the University of Valencia and the project GV2005-184 of the Generalitat Valenciana.

References

[1] E. Acosta, V. Bove jr., J. Watlington, and R. Yu. Reconfigurable processor for a data-flow video processing system. In J. Schewel, editor, *SPIE: FPGAs for Fast Board Development and Reconfigurable Computing*, pages 83–91, Bellingham, Washington, 1995.

[2] K. Aizawa, Y. Egi, T. Hamamoto, M. Hatori, M. Abe, H. Maruyama, and H. Otake. Computational image sensor for on sensor compression. *IEEE Transactions on Electron Devices*, 44(10):1724–1730, October 1997.

[3] Eugenio Culurciello, Ralph Etienne-Cummings, and Kwabena A. Boahen. A biomorphic digital image sensor. *IEEE journal of solid-state circuits*, 38(2), February 2003.

[4] T. Hamamoto, R. Ooi, Y. Ohtsuka, and K. Aizawa. Real-time image processing by using image compression sensor. In *International Conference on Image Processing, ICIP'99*, volume 3, pages 935–939, October 1999.

[5] Thomas A. Holz and John G. Harris. An integrate and fire pixel with contrast outputs for a cmos imager. In *IASTED International Conference on Circuits, Signals, and Systems*, February 2003.

[6] C. Kimme, D. H. Ballard, and J. Sklansky. Finding circles by an array of accumulators. *Communications of the ACM*, 18:120–122, 1975.

[7] Filiberto Pla. Recognition of partial circular shapes from segmented contours. *Computer Vision and Image Understanding*, 63(2):334–343, March 1996.

[8] E. Rivlin, M. Rudzsky, R. Goldenberg, U. Bogomolov, and S. Lapchev. A real-time system for classification of moving objects. In *Proceedings of the 16 International Conference on Pattern Recognition - ICPR'02*, volume 3, pages 688–691, Québec City, Canada, August 2002.

[9] Paul L. Rosin. Thresholding for change detection. In *ICCV*, pages 274–279, 1998.