Selective Change-Driven Image Processing: A Speeding-Up Strategy

Jose A. Boluda[1], Francisco Vegara[2], Fernando Pardo[1], and Pedro Zuccarello[1]

[1] Departament d'Informàtica, Universitat de València. Avda Vicente Andrés Estellés, s/n. 46100-Burjassot, Spain
[2] Institut de Robòtica, Universitat de València. Polígono de la Coma, s/n. Aptdo. 2085, 46890-València, Spain
{Jose.A.Boluda,Francisco.Vegara,Fernando.Pardo,Pedro.Zuccarello}@uv.es
http://tapec.uv.es

Abstract. Biologically inspired schemes are a source for the improvement of visual systems. Real-time implementation of image processing algorithms is constrained by the large amount of data to be processed. Full image processing is many times unnecessary since there are many pixels that suffer a small change or not suffer any change at all. A strategy based on delivering and processing pixels, instead of processing the complete frame, is presented. The pixels that have suffered higher changes in each frame, ordered by the absolute value of its change, are read-out and processed. Two examples are shown: a morphological motion detection algorithm and the Horn and Schunck optical flow algorithm. Results show that the implementation of this strategy achieves execution time speed-up while keeping results comparable to original approaches.

1 Introduction

Full image processing is usually the classical approach for general image sequence processing, where each image is a snapshot taken at regular intervals. The normal procedure implies the application of the processing algorithm for each image in the sequence. Biological systems work in a different way: each sensor cell sends its illumination information independently. It is possible to reduce processing time by taking into account that images usually change little from frame to frame, especially if the acquisition time is short. This is particularly true in motion detection algorithms with static cameras.

A biologically inspired camera would send pixel information asynchronously when changes are produced, rather than full acquired images. Following this ideas, it is also possible to implement a change-driven data-flow policy in the algorithm execution, processing only those pixels that have changed. Paying attention only to those pixels that change is not new and this principle has been employed to design some image sensors with on-plane compression [1]. These image sensors only deliver the pixels that change, decreasing the amount of data coming from the camera. This strategy will decrease the total amount of data to be processed; consequently also it will decrease the number of instructions and thus the computer execution time.

E. Bayro-Corrochano and J.-O. Eklundh (Eds.): CIARP 2009, LNCS 5856, pp. 37–44, 2009.

A biologically motivated global strategy for speeding-up motion detection algorithms is presented. The system includes a change-driven camera that delivers pixels change instead of synchronous full images, and a data-flow algorithm adaptation for the image processing algorithm.

2 Change-Driven Camera and Processing

2.1 Change-Driven Camera

Biological visual systems has been already partially emulated taking into account its asynchronous nature [2]. Each pixel works independently in this visual sensor and the available output bandwidth is allocated according to pixel output demand. In this kind of sensors the change event signaling depends on a contrast sensitivity threshold, which is also found in biological vision systems. A pixel change greater than this threshold is considered as a change, consequently this pixel is read out and processed. This threshold has already been successfully employed to accelerate differential movement algorithms [3].

A Selective Change-Driven (SCD) camera with pixel delivering for high-speed motion estimation is under construction [4]. In this camera every pixel has an analogue memory with the last read-out value. The absolute difference between the current and the stored value is compared for all pixels in the sensor; the pixel that differs most is selected and its illumination level and address are read out for processing. With this strategy, every pixel that has changed will be sent sooner or later, and thus processed in a data-flow manner, ordered by its illumination change.

2.2 Data-Flow Processing

A generic image processing algorithm can be programmed as an instruction sequence within a classical control flow computing model. The data-flow model works in a totally different way: instructions are fired when the data needed for these instructions are available [5]. One of the main advantages of this model is the reduction of the instructions to be executed when little changes are produced in input data.

Motion detection algorithms (and as a particular case, differential algorithms) greatly benefits from the approach of firing instructions of an algorithm only when data changes. Often only few pixels change from frame to frame and usually there is no need to execute any instruction for unchanged pixels. The classical approach performs the same calculation for all the pixels in an image for every image in a sequence, even if the pixel did not change at all. It is possible to save many calculations if only those pixels that have changed are delivered by the SCD camera and fire the related instructions.

Any image processing algorithm would need to be rebuilt in order to be implemented following the SCD strategy. Extra storage to keep track of the intermediate results of preceding computing stages is needed by this methodology.

3 Motion Detection Algorithms

The change-driven delivering and processing has been tested in several motion detection algorithms that have been rebuilt in a data-flow manner. Two examples are included in this paper.

3.1 Traffic Detection Algorithm

This motion detection algorithm has already been utilized serving as an example of how the use of a change sensitive threshold can accelerate differential motion detection algorithms [3].

A detailed description of the original sequential procedure can be seen at [6], where I_t is the input sequence and M_t is the estimated background value. The estimate background is increased by one at every frame when it is smaller than the sample or decreased by one when it is greater than the sample. The absolute difference between I_t and M_t is the first differential estimation Δ_t, that is used to compute the pixels motion activity measure, employed to decide whether the pixel is moving or static. V_t is used as the dimension of a temporal standard deviation. It is computed as a $\Sigma - \Delta$ filter of the difference sequence. Finally in order to select pixels that have a significant variation rate over its temporal activity, the $\Sigma - \Delta$ filter is applied $N = 4$ times. A simple common edges hybrid reconstruction is performed to enhance Δ_t as shown in equation (1). The inputs are the original image I_t and the $\Sigma - \Delta$ difference image Δ_t.

$$\Delta'_t = HRec_{\alpha}^{\Delta_t}(Min(\|\nabla(I_t)\|, \|\nabla(\Delta_t)\|)) \tag{1}$$

The gradient modules of Δ_t and I_t are computed by estimating the first Sobel gradient and then computing the Euclidean norm. $Min(\|\nabla(I_t)\|, \|\nabla(\Delta_t)\|)$ acts as a logical conjunction, retaining the edges that belong both to Δ_t and I_t.

The common edges within Δ_t and with α as structuring element (a ball with radius=3) are reconstructed in order to recover the object in Δ_t. This is done by performing a geodesic reconstruction of the common edges (marker image) with Δ_t as reference. Thus, after Δ_t has been reconstructed, V_t and D_t are computed.

Change-Driven Data-Flow Algorithm. The original algorithm has been modified using the change-driven data-flow processing strategy. With this procedure as soon as the SCD camera delivers a pixel that has changed $\Delta I(x, y)$ the related instructions are fired, and the intermediate images are updated. Fig. 1 shows the data-flow and the intermediate stored images.

An initial image is stored in the computer as the current image. Any absolute difference of the current image pixel with the stored image fires the intermediate images computation. Moreover, these updates must be done taking into account that the change of an input pixel may modify several output variables. For example, if a pixel is modified then its contribution to 6 pixels for the Sobel gradient image G_x and also for 6 pixels for image G_y must be updated. It may appear that a single pixel modification can produce too many operations, but

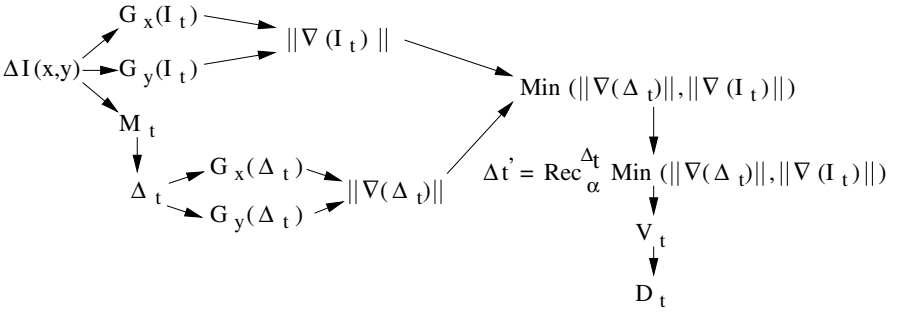

Fig. 1. Change-driven motion detection algorithm

theese are simple additions (and sometimes a multiplication by two) per pixel and they can be reutilized. In the original algorithm for each pixel the Sobel gradient images G_x and G_y are computed in any case; with six additions per pixel with the corresponding multiplications.

3.2 Horn and Schunk Optical Flow Computation

Optical flow is one of the main methods to estimate movement of objects and its calculation provides valuable information to artificial and biological systems. Unfortunately it is computationally intensive which constrains its use in real-time applications; despite of this, its high scientific interest motivates research on new strategies and hardware approaches to reduce its calculation time [8].

Differential techniques are applied under global restrictions in the Horn and Schunck method. Several global approximations are assumed as the conservation of intensity. Additionally, a method of global restriction that minimizes the squared magnitude of the gradient of the optical flow is introduced. A mask is used for Laplacian calculation of the mean values $(\bar{u}, \bar{v})$ of the optical flow components at any point (x, y), which are used in the equations that relate the optical flow vector from the image Laplacian and the spatial-temporal gradients:

$$u = \bar{u} - I_x \frac{I_x\bar{u} + I_y\bar{v} + I_t}{\lambda^2 + I_x^2 + I_y^2} \qquad v = \bar{v} - I_y \frac{I_x\bar{u} + I_y\bar{v} + I_t}{\lambda^2 + I_x^2 + I_y^2} \tag{2}$$

Final classical determination of the optical flow is done from these equations through an iterative full image processing over pairs of consecutive images.

Data-Flow version. The change-driven data-flow implementation of the Horn and Schunck algorithm uses the equations that treat only the data which are involved because of the variation of a given pixel, differently to the calculation of optical flow in the whole image. The procedure is as follows:

– Initial gradient and optical flow maps are computed for the whole image following the classical Horn and Shchunk method described before.

– Then, the changing pixels sent by the SCD sensor are processed in decreasing order of variation and for each received pixel the following operations are performed:
  - Recalculate the spatial and temporal gradients for those pixels of the image that are under the influence of the pixel that has changed.
  - Recalculate $(\bar{u}, \bar{v})$ for all pixel involved by the variation of pixel (i, j).

4 Experimental Results

Since the construction of the SCD camera is still in progress, the synchronous delivering of the pixels that have changed has been simulated by software. The comparison between the original algorithms and the change-based data-flow versions follows.

4.1 Traffic Detection Algorithm Results

The original traffic detection algorithm and the change-driven data-flow versions have been implemented. Both algorithms have been tested using several traffic sequences downloaded from the professor H. H. Nagel public ftp site: `http://i21www.ira.uka.de/image_sequences/` at the University of Karlsruhe. Fig. 2(a) shows a sequence frame of 740×560 pixels. Fig. 2(b) shows the original version with a full frame processing (414,400 pixels per frame). The change-driven data-flow algorithm results are shown in Fig. 2(c) with a mean of 80,000 pixels (roughly a 20% of image pixels). In this experiment, it has been simulated that the SCD camera has delivered out, ordered by the absolute magnitude of its change, a mean of 80,000 pixels.

Both result images shown at Fig. 2 are almost the same. There are no evident differences in terms of detected moving points between the original and the change-driven implementations. The executed time decreases significantly as Fig. 4 (left) shows (speed-up of 1.57). In this way, it must be appointed that the change-driven data-flow implementation gives similar results as the original, but with lower computational cost.

The execution time decreases because only changing pixels are processed. Moreover, if there are bandwidth limitations, the pixels with a bigger change

Fig. 2. (a) Original sequence, (b) Original algorithm results, (c) Change-Driven modified algorithm with 80,000 pixels (20%)

are processed first leaving as no processed the pixels with a lower change. The question in the application to this algorithm is whether there is a limit to decreasing the number of pixels and therefore to the algorithm speed-up versus the original full processing implementation. The answer is that if very few points are processed there is not a systematic background update, and moreover, moving points with a small gray value difference are not detected. This property produces two effects that limit the change-driven data-flow strategy. If there are more changes than pixels that the SCD camera can deliver due to bandwidth limitations, fewer correct moving points are detected and, otherwise, more false positives are detected. Values under a 10% of the image total number of pixels make this approach unfeasible and therefore further pepper noise filtering is required for D_t, disappearing the change-driven data-flow speed-up.

4.2 Change-Driven Optical Flow Results

The well known Rubik sequence has been downloaded from the ftp public site of the Department of Computer Science (`ftp://ftp.csd.uwo.ca/pub/vision/`) at the University of Ontario. Each frame has 61,440 pixels (256×240). Classical parameters have been used to calculate the optical flow: 10 iterations for each frame; only the flow vectors with modulus bigger or equal than 0.2 have been represented; the value of the Lagrange multiplier for the regularization term has been taken as $\lambda = 5$. Results are shown in the Fig. 3 for the classical algorithm implementation and for the change-driven implementation with 4,000 pixels, roughly a 7% of the image size.

Table 1 shows the mean angular deviation (in degrees) between the original Horn and Schunk algorithm and the change-driven implementation for different number of processed pixels. This angular deviation Ψ_E between the real velocity vector components (u_c, v_c) and the calculated velocity vector (u_e, v_e) has been computed through the optical flow error equation:

$$\Psi_E = \arccos\left(\frac{u_c u_e + v_c v_e + 1}{\sqrt{(u_c^2 + v_c^2 + 1)(u_e^2 + v_e^2 + 1)}}\right) \tag{3}$$

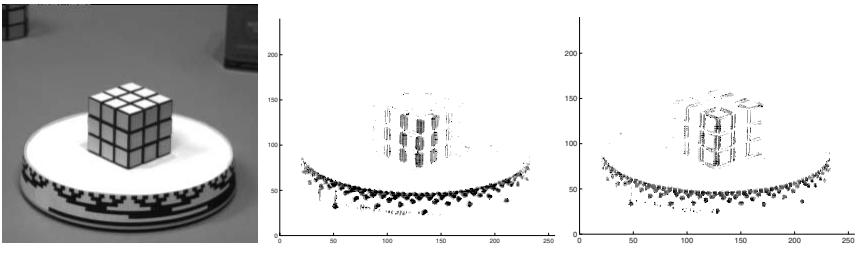

Fig. 3. (a) Original sequence, (b) Original optical flow results, (c) change-driven optical flow computed with 4000 pixels

Table 1. Mean error and standard deviation with different number of pixels

N^o of pixels	Mean error (^{o})	Standar deviation (^{o})
1000	18	8
2000	16	8
3000	15	8
4000	14	7
5000	13	7

The error shown at table 1 decreases as long as the number of pixels increases. In this algorithm, for a number of pixels under a 10%, result seems not very good but are not far from most optical flow algorithms with greater complexity. In this case, a lower number of pixels processed than in the traffic algorithm can give acceptable results. The Rubiks sequence has been taken in a controlled environment, giving less changing pixels.

Experimental measured speed-up of the change-driven optical flow algorithm referenced to the classical one is shown in Figure 4 (right). The optical flow is calculated for every received pixel as long as there is sufficient time until the next integration period. If not all received pixels can be processed, those with a bigger change in their luminance are processed first, since this is the way they will arrive from the sensor. This can be interpreted as an optical flow calculation at the pixel level instead of at the frame level. As expected, for a low number of processed pixels, there is a significant speed-up. If the number of pixels increases, the speed-up decreases. With 4,000 pixels there is still a speed-up of 1.2 with an error of 14° which can be useful for real-time applications.

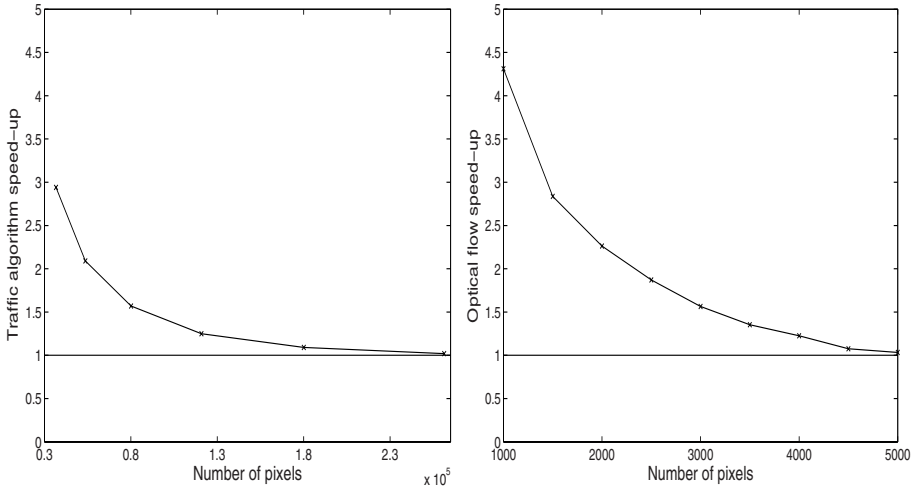

Fig. 4. Speed-up for the traffic detection (left) and the optical flow (right) algorithms

5 Conclusion

A biologically inspired strategy for speeding-up motion detection algorithms has been presented. The system includes an SCD camera that sends the pixel changes instead of sending sequentially full frames. Following these ideas, a generic image processing algorithm must be rebuild in a data-flow manner. The change-driven data-flow strategy is based on processing the pixels that have changed ordered by its absolute difference value.

The implementation of this methodology requires several algorithm adaptations and extra storage to keep track of the intermediate results. Two motion analysis algorithms have been chosen to test the change-driven data-flow policy: a morphological traffic detection algorithm and the Horn and Schunk optical flow algorithm. Change-driven data-flow algorithm implementations show a useful speed-up giving similar results of the original implementation.

Acknowledgments

This work has been supported by the projects TEC2006-08130/MIC and TEC2009-12980 of the Spanish *Ministerio de Ciencia e Innovación* and FEDER funds.

References

1. Özalevli, E., Higgins, C.M.: Reconfigurable Biologically Inspired Visual Motion Systems Using Modular Neuromorphic VLSI Chips. IEEE Transactions on Circuits and Systems 52(1), 79–92 (2005)
2. Lichtsteiner, P., Posch, C., Delbruck, T.: A 128x128 dB 15 μs Latency Asynchronous Temporal Contrast Vision Sensor. IEEE Journal of Solid-State Circuits 43(2), 566–576 (2008)
3. Boluda, J.A., Pardo, F.: Speeding-up differential motion detection algorithms using a change-driven data-flow processing strategy. In: Kropatsch, W.G., Kampel, M., Hanbury, A. (eds.) CAIP 2007. LNCS, vol. 4673, pp. 77–84. Springer, Heidelberg (2007)
4. Pardo, F., Boluda, J.A., Vegara, F., Zuccarello, P.: On the advantages of asynchronous pixel reading and processing for high-speed motion estimation. In: Bebis, G., Boyle, R., Parvin, B., Koracin, D., Remagnino, P., Porikli, F., Peters, J., Klosowski, J., Arns, L., Chun, Y.K., Rhyne, T.-M., Monroe, L. (eds.) ISVC 2008, Part I. LNCS, vol. 5358, pp. 205–215. Springer, Heidelberg (2008)
5. Silc, J., Robic, B., Ungerer, T.: Processor architecture: from dataflow to superscalar and beyond. Springer, Heidelberg (1999)
6. Manzanera, A., Richefeu, J.C.: A new motion detection algorithm based on Σ-Δ background estimation. Pattern Recognition Letters 28, 320–328 (2007)
7. Teng, C.H., Lai, S.H., Chen, Y.S., Hsu, W.H.: Accurate optical flow computation under non-uniform brightness variations. Computer Vision and Image Understanding 97(3), 315–346 (2005)
8. Díaz, J., Ros, E., Pelayo, F., Ortigosa, E.M., Mota, S.: FPGA-based real-time optical-flow system. IEEE Transactions on Circuits and Systems for Video Technology 16(2), 274–279 (2006)