International Journal of Electrical, Electronic and Communication Sciences
ISSN: 2517-9438
Vol:4, No:3, 2010

A Reliable FPGA-based Real-time Optical-flow Estimation

M. M. Abutaleb, A. Hamdy, M. E. Abuelwafa, and E. M. Saad

***Abstract*—**Optical flow is a research topic of interest for many years. It has, until recently, been largely inapplicable to real-time applications due to its computationally expensive nature. This paper presents a new reliable flow technique which is combined with a motion detection algorithm, from stationary camera image streams, to allow flow-based analyses of moving entities, such as rigidity, in real-time. The combination of the optical flow analysis with motion detection technique greatly reduces the expensive computation of flow vectors as compared with standard approaches, rendering the method to be applicable in real-time implementation. This paper describes also the hardware implementation of a proposed pipelined system to estimate the flow vectors from image sequences in real time. This design can process 768 x 576 images at a very high frame rate that reaches to 156 fps in a single low cost FPGA chip, which is adequate for most real-time vision applications.

***Keywords*—**Optical flow, motion detection, real-time systems, FPGA.

I. Introduction

THIS work presents a robust method to estimate the flow vectors for moving objects appearing in images of the frame sequence. First, a motion detection technique is applied to detect the moving pixels in the image sequence taken from a static camera. Second, optical flow estimation is preformed only w.r.t. the detected moving pixels. This method is suitable for FPGA implementation that can be used in applications which require real-time.

A. Motion Detection

Motion detection can be essential to allow for the optical flow estimation to be performed in real-time. There are three conventional approaches to motion detection: temporal differencing [1]; optical flow analysis [2]; and background subtraction [3-5]. Temporal differencing is suitable in dynamic environments, but generally does a poor job of extracting all relevant feature-pixels. Optical flow can be used to detect independently moving objects in the presence of camera motion; however, most flow-computation methods are complex and inapplicable in real-time. Motion detection by background subtraction can be divided into adaptive and non-adaptive background methods. Non-adaptive methods need off-line initialization; errors in the background accumulate over time.

A common method of adaptive backgrounding is to average the frames over time [6]. This creates an approximate background. This is effective where objects move continuously and the background is visible for a significant portion of time. It is not robust for scenes with slowly-moving objects. It cannot handle a multimodal backgrounds caused by the repetitive motion of the background. Hence, in this work a fast motion detection algorithm based on a multi-modal distribution, modeling each pixel as a mixture of normal distributions, is used to detect the moving objects with a small number of calculations, as presented in [7], to achieve a high frame rate for real-time requirements. This method deals robustly with slowly-moving objects as well as with repetitive background motions of some scene-elements.

B. Optical flow Estimation

Optical flow approach is hard to apply in real-time due to its high computational cost. Hence, it is a reasonable idea to combine the motion detection with the optical flow estimation in a way that flow vectors are calculated only w.r.t. moving pixels to allow flow-based analyses of moving entities in real-time. Applying this flow technique to moving entities provides some straight forward primitives for analyzing the motion of those objects such as analyzing rigidity and cyclic motion using residual flow; and determining self-occlusion and disambiguating multiple, mutually occluding entities using pixel contention.

In this way the computation time can be reduced in most practical cases, making the method feasible in the real-time. In this work, an improved algorithm of optical flow estimation by the region-based matching is proposed and preformed only w.r.t. the detected moving pixels to obtain a reliable flow vectors.

C. Hardware Implementation

High complexity algorithms to estimate the motion field from image sequences have already been developed and successfully implemented in software [2]. Very few of these algorithms have been incorporated into today's video surveillance systems, due to computational cost and lack of real-time capability. This makes the development of such algorithms on the hardware timely. Known algorithms have been restricted to small frame sizes, low frame rates, and solely implemented in software running on general-purpose

Authors are with department of electronics, communications, and computer, faculty of engineering, Helwan University, Cairo, Egypt.

This work was supported by Department of Electronics, Communications, and Computer, Faculty of Engineering, Helwan University, Cairo, Egypt.

computers. Real-time needs of such systems can be improved by a significant amount with the use of hardware.

Hence, this paper presents a hardware implementation of a proposed method, which takes advantage of data parallelism for a further implementation on a field programmable gate array (FPGA), a re-configurable computing platform. Throughput has been drastically increased while; in contrast, the latency has been decreased, with the use of pipelining. This implementation is used to estimate the flow vectors for moving objects appearing in image sequence at a very high frame rate in a single low cost FPGA chip.

This paper is organized as follows. In section 2, the algorithms are formulated and the suggestions are introduced. In section 3, the hardware implementation design and performance analysis of the proposed system are discussed. Conclusion is given in section 4.

II. Algorithms

A. Motion Detection based on Multi-modal Distribution

A fast and efficient algorithm, presented in [7], is used here to extract the moving objects in each frame for software and hardware implementation. In this algorithm, each pixel is modeled as mixture of three distributions (k = 3) and each distribution is represented by mean-value (μ) and weight-value (ω) maintained at time t. The mixture is sorted every time in decreasing order of weight values. Here, each pixel is checked against the distributions, until the match is found. The matching condition is achieved if the variation of the pixel X_t within $R\%$ (matching ratio) from its mean value.

For the matched distribution, the current pixel is stored as the processed pixel value $X_{P,t}$ ($X_{P,t} = X_t$) to be used in the next time t+1. Also, the temporal differencing is applied where the pixel is considered as foreground pixel if:

$$| X_{\mathrm{t}} - X_{\mathrm{P,t\text{-}1}} | > T \tag{1}$$

where T is the threshold value.

If foreground pixel is detected, the weight of the matched distribution will be updated as in equation (2) and its mean will be kept without any change. While if foreground pixel is not detected, the weight and mean of the matched distribution will be updated as in equation (3) and (4) respectively, where α is the learning factor ($\alpha = 0.005$).

$$\omega_{\mathrm{K,t}} = (1-\alpha)\omega_{\mathrm{k,t\text{-}1}} \tag{2}$$

$$\omega_{\mathrm{K,t}} = (1-\alpha)\omega_{\mathrm{k,t\text{-}1}} + \alpha \tag{3}$$

$$\mu_{\mathrm{K,t}} = (1-\alpha)\mu_{\mathrm{k,t\text{-}1}} + \alpha X_{\mathrm{t}} \tag{4}$$

If the current pixel is not matched with any distribution, this pixel will be classified as foreground pixel. The mean of the third distribution of that pixel will be replaced by its intensity value and its weight will be selected as lower value than other distributions. Also, the processed pixel will be equal to the value of the mean of the first distribution. Real-time performance with high resolution video streams can be achieved by this algorithm.

B. Optical Flow Estimation by Region-based Matching

To analyze the motion of objects over the image sequence and to estimate the 2D-velocity of a certain pixel moving within ROI, the region-based matching is used. The fixed regions as in [8] and the dynamic regions as in [9] have been used for the matching process. The proposed idea is to use the gradient-based measures of texture information for every pixel position. These measures which represent the local gradient activities and the intensity value for each moving pixel are used for the matching process. Hence, the flow computation per pixel is a two-pass process. First, both horizontal and vertical gradient components are generated for each moving pixel to ensure that sufficient information is available to obtain a valid flow vector. Second, a region-matching is performed to actually calculate the flow vectors.

To ensure a good match between regions, it is essential that sufficient texture information around their central pixels x is available in both horizontal and vertical directions. Thus, two images $H(x)$ and $V(x)$ using standard horizontal and vertical Sobel operators are generated for each image $I(x)$ in the frame sequence (I is a pixel intensity value and x=(i,j) represents a pixel location in the image). A region $W(x)I(x)$ as proposed in [10] and regions $W(x)V(x)$ and $W(x)H(x)$ as proposed in this work are defined by multiplying I, V, and H with a 2D masking function $W(x)$ respectively.

The obtained regions for the current and previous images in the frame sequence are then convolved, as in [11]. This is to get a three correlation surfaces (information measures); intensity measure $D_I(x;d)$, vertical-gradient measure $D_V(x;d)$, and horizontal-gradient measure $D_H(x;d)$.

$$D_{\mathrm{I}}(x;d) = \frac{1}{WiWj}\sum_{i=1}^{i=Wi}\sum_{j=1}^{j=Wj} | I_{\mathrm{n}}(i,j) - I_{\mathrm{n\text{-}1}}(i+di, j+dj) | \tag{5}$$

$$D_{\mathrm{V}}(x;d) = \frac{1}{WiWj}\sum_{i=1}^{i=Wi}\sum_{j=1}^{j=Wj} | V_{\mathrm{n}}(i,j) - V_{\mathrm{n\text{-}1}}(i+di, j+dj) | \tag{6}$$

$$D_{\mathrm{H}}(x;d) = \frac{1}{WiWj}\sum_{i=1}^{i=Wi}\sum_{j=1}^{j=Wj} | H_{\mathrm{n}}(i,j) - H_{\mathrm{n\text{-}1}}(i+di, j+dj) | \tag{7}$$

where d=(di, dj) is a linear intra-sequence translation (search area).

Consider a stationary video-camera observing the scene. If the images in the frame sequence are separated in time by Δt, then a particular pixel in the image will move by a distance $v(x)\Delta t$, where $v(x)$=($v_i(x)$, $v_j(x)$) is the 2D image velocity of that pixel. This can be found by matching the corresponding regions in each two successively images in the frame sequence which can be satisfied by minimizing a proposed overall correlation function $D(x;d)$.

$$D(x;d) = \beta_1 * D_{\mathrm{I}}(x;d) + \beta_2 * D_{\mathrm{V}}(x;d) + \beta_3 * D_{\mathrm{H}}(x;d) \tag{8}$$

where β_1 , β_2 , $\beta_3 \in [0\ ,\ 1]$ and $\beta_1 + \beta_2 + \beta_3 = 1$ represent the matching weights of a three correlation components $D_I(x;d)$, $D_V(x;d)$, and $D_H(x;d)$ according to their effective in the

matching. The search for the minimum of *D(x;d)* returns the true pixel displacement value d_{min}.

$$d_{\min} = \min D(x;d) = v(x)\Delta t \quad (9)$$

If the fixed regions have been used for matching, the obtained results by the proposed method is more accurate than the results that obtained by the method in [10] because the gradient-based measures with the intensity measures give sufficient information for matching. The method in [10] can be improved by using dynamic regions that kept growing until there was still enough information. But this improved method is very complex for real time and hardware implementation. So, the proposed reliable method is more proper for real time and hardware implementation which is the aim of this work.

C. Experimental Results

Figure 1 shows the image under study. The next step is the motion detection by our proposed method in [7], which is performed to extract the foreground pixels; moving pixels. Figure 2 demonstrates the extraction of the foreground pixels. Subsequently, the post-processing is performed to fill the holes inside the blobs, as it is demonstrated in Fig. 3. At the next step, the two-pass process of the optical flow estimation is triggered. Figure 4 shows the intensity, horizontal, and vertical evidence information proves the existence of the sufficient local texture.

A region-matching is performed to compute the flow-vectors using a 3x3 support window, as shown in Fig. 5. It is evident from this figure that, there is homogeneity in the flow vectors of the rigid object, while with non-rigid object there is not. It is clear from this figure that, the homogeneity increases with the rigidity and vice versa.

Fig. 1 Single image from the sequence

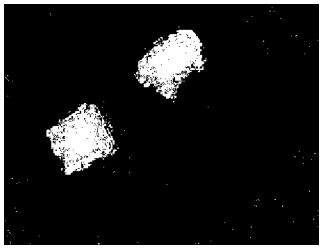

Fig. 2 Pre-processing foreground image

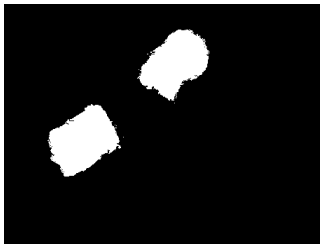

Fig. 3 Post-processing foreground image

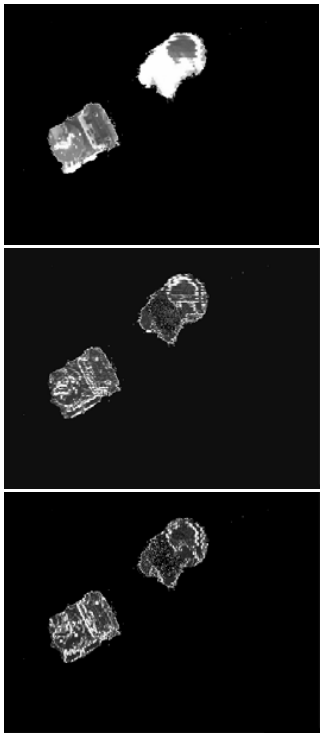

Fig. 4 Masked image (top) and horizontal (middle) and vertical (bottom) components

Fig. 5 Resulted optical flow (motion field)

III. Design

A. Data Flow

The data flow of the proposed algorithm is divided into three modules as shown in Fig.6. In the *Motion Detection* module, images are read from memory into the FPGA and aligned to extract the foreground images (masks). This module is previously implemented in [7]. Then, each image in the frame sequence is masked by the foreground images (masks) to generate the horizontal, vertical, and intensity components for each masked image in the *Moving Object Components* module. Finally, these components are fed into the *Optical Flow Estimation* module to compute the final result according to equations 5, 6, 7, 8, and 9 and then written back to memory. Of importance, there is no iterative processing in this design. Therefore, the computation process can be fully pipelined in hardware to improve its processing throughput and thereby enable real-time use.

In the block diagram of Fig. 6, the connections between modules consist of only unidirectional data and a corresponding data valid signal. Once a set of data is generated in a module, it is registered into the downstream module for processing. At the end of the pipeline, the results are written back to memory for further processing, display, or analysis.

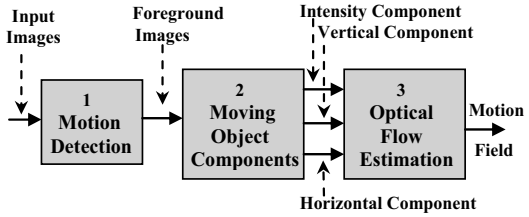

Fig. 6 Data flow of the design

B. Architectures

FPGAs have been used to process larger images at faster speed because of their configuration flexibility and high data processing speed. An algorithm proposed by Horn and Schunck [12] was implemented in [13]. It is an iterative algorithm where the accuracy depends largely on the number of iterations. The classical Lucas and Kanade approach [14] has also been implemented in [15] for its good tradeoff between accuracy and processing efficiency. A fast and accurate motion estimation algorithm has been modified as proposed in [16] for efficient FPGA hardware implementation. This design was accurate and fit for pipeline hardware implementation. It was able to process images of size 640 × 480 at 64 frames per second.

The main objective of this work is to create a prototype system that can generate a reliable flow-vector at every pixel in the moving blobs at a much higher rate as compared to previous proposed works [13, 15, 16]. This approach can be applied in different applications such as tracking systems, robot navigation, and object classification in real time. Here, FPGA-based architectures for the proposed algorithm of optical flow estimation by dynamic region matching at every pixel in the moving blobs are presented.

For *Moving Object Components* module, two identical pixel-based architectures are used in parallel to compute the components of each pixel in the previous and current images in the frame sequence. Each one consists of *Obj_Ext* block, *Obj_Reg* block, *Ver_Comp* block, and *Hoz_Comp* block, as shown in Fig. 7. A control signal *rst* initializes all parameters and registers and *rd_in* signal is active when *mask* signal (foreground pixel) from *Motion Detection* module is ready in the input. *Obj_Ext* block is used to generate *J* signal (intensity component) which is the same value of *F* signal (gray value of the processing pixel) in case of the foreground pixel (*mask* = 1) and zero value in case of the background pixel (*mask* = 0). The masked pixel *J* is stored in *Obj_Reg* block to be used in the next time for its neighbors within the selected window. The previously stored masked pixels *P1* to *P8* (neighbors of the processing pixel) in *Obj_Reg* block are used to generate *V* (vertical component) and *H* (horizontal component) signals using vertical and horizontal Sobel operators in *Ver_Comp* and *Hoz_Comp* blocks respectively.

For *Optical Flow Estimation* module, a pixel-based architecture is used to compute the true pixel displacement by matching each specified region in the current masked image with all regions in the previous masked image using *J*, *V*, and *H* components. It consists of *Pixel_Corr_Calc* block and *Match_Loc* block, as shown in Fig. 8. A control signal *rst* initializes all parameters and registers while the control signals *En1, En2, En3* are used to enter the pixels of the current and previous images through the matching operation and output the final result at the end of the operation. Three identical *Pixel_Corr_Calc* blocks are used in parallel to compute the correlation components for each pixel in the previous and current images in the frame sequence according to equations 5, 6, and 7. *Match_Loc* block is used to compute the overall correlation function and to search for its minimum in each location [Cx,Cy] according to equations 8 and 9 to return the matching location [Xm,Ym].

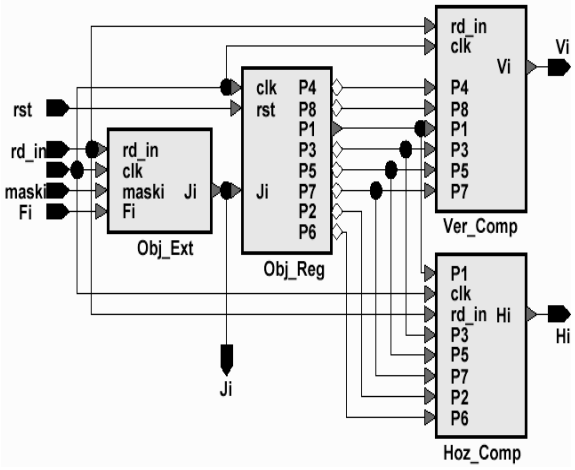

Fig. 7 The structure of *Moving Object Components* module

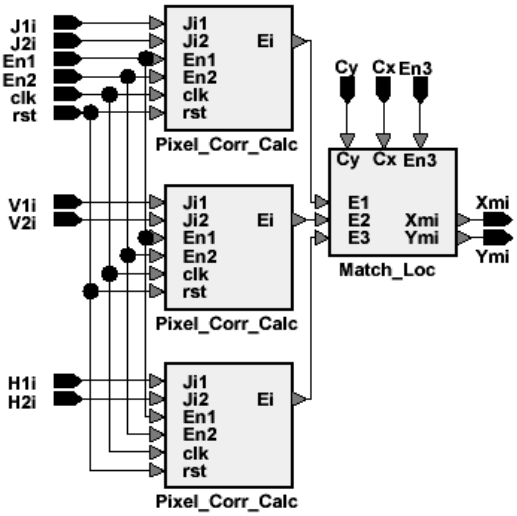

Fig. 8 The structure of *Optical Flow Estimation* module

C. Hardware Optimization

For maximum hardware performance, several hardware optimizations were made:

- *Pipeline structure*. A heavily pipelined hardware structure was used to maximize throughput. Once the pipeline is full, the hardware can produce a result on every clock cycle.
- *Optimizations of Motion Detection module.* Memory bandwidth reduction is achieved by utilizing distribution similarities in succeeding neighboring pixels and data flow reduction is also achieved by processing only one distribution at time through the hardware, as presented in [17].
- *Fast memory access.* Due to the pipelined hardware architecture used, the major system bottleneck turned out to be memory access. A high speed multi-port memory controller is used to increase the data throughput of the design.
- *Bit width trimming*. Fixed-point numbers are used in the design. To maintain accuracy with saving hardware resources, the bit widths used are custom selected at each stage of the processing pipeline.

D. Experimental Results

This system has been implemented on low cost available Spartan-II development system with Xilinx chip 2s200fg456 which has 2352 CLB Slices. The system clock rate is 150 MHz. This clock is converted into different levels which are required for the units within the system. Module 1 uses 418 CLB slices with 17.8% utilization [17], module 2 uses 88 CLB slices with 3.74% utilization, and module 3 uses 383 CLB slices with 16.28% utilization. The module numbers are those from Fig. 6. This system is able to process images of size 768x576 (442368 pixels) at 156 fps. The output results are the same results that are obtained by the software version. To speed up this system, the bigger area and higher performance FPGA chip or more than one FPGA chip can be used.

IV. Conclusion

In this paper, a new algorithm of optical flow estimation by the region-based matching has been proposed and preformed only w.r.t. the detected moving pixels to obtain a reliable optical flow that allows flow-based analyses of moving entities in real-time. The proposed idea is to use the gradient-based measures of texture information for every moving pixel position with its intensity variation in the matching process. Real-time performance and high accuracy are achieved by this algorithm.

Then, a hardware implementation of the proposed method has been presented in the form of pipeline and parallel processing to achieve maximum speeding up. Several hardware optimizations were also made to obtain maximum hardware performance. At a clock rate of 150 MHz, this design could estimate the flow vectors for moving objects appearing in 768x576 image sequence taken from a static camera at a very high frame rate of 156 frames per second in a single low cost FPGA chip. The proposed system can be considered as higher frame rate, lower cost, and smaller area than other recent presented systems [13, 15, 16] for the flow-vector estimation.

References

[1] C. Anderson, P. Burt, and G. van der Wal, "Change detection and tracking using pyramid transformation techniques", in Proceedings of SPIE. Intelligent Robots and Computer Vision, vol. 579, pp. 72-78, 1985.

[2] J. Barron, D. Fleet, and S. Beauchemin, "Performance of optical flow techniques", International Journal of Computer Vision, vol. 12, pp. 42-77, 1994.

[3] A. Kasinski and A. Hamdy, "Efficient Separation of mobile objects on the scene from the sequence taken with an overhead camera". Proc. Int. Conf. on Computer Vision and Graphics, Zakopane, vol. 1, pp. 425-430, Sept. 2002.

[4] C. Ridder, O. Munkelt, and H. Kirchner, "Adaptive Background Estimation and Foreground Detection Using Kalman-Filtering". Proc. Int. l Conf. Recent Advances in Mechatronics, ICRAM .95, pp. 193-199, 1995.

[5] Y. Ivanov, A. Bobick, and J. Liu, "Fast Lighting Independent Background Subtraction". Technical Report no. 437, MIT Media Laboratory, 1997.

[6] G. Halevy and D. Weinshall, "Motion of disturbances: detection and tracking of multi-body non-rigid motion", Machine Vision and Applications, vol. 11, Issue 3, pp. 122-137, 1999.

[7] E.M. Saad, A. Hamdy, and M.M. Abutaleb, "FPGA-based Implementation of a Low Cost and Area Real-time Motion Detection", 15th IEEE Conference in Mixed Design of Integrated Circuits and Systems, Poznan, Poland, pp. 249-254, June 19-21, 2008.

[8] P. Anandan, "A Computational Framework and an Algorithm for the Measurement of Visual Motion", International Journal of Computer Vision, vol. 2, 1989.

[9] M. Okutomi and T. Kanade, "A Locally Adaptive Window for Signal Matching", International Journal of Computer Vision, vol. 7, no. 2, 1994.

[10] A. J. Lipton, "Local Application of Optic Flow to Analyze Rigid versus Non-Rigid Motion", ICCV Workshop on Frame-Rate Vision, 1999.

[11] K. N. Ngan, T. Meier, and D. Chai, "Advanced Video Coding: Principles and Techniques", Elsevier, 1999.

[12] B. Horn, B. Schunck, "Determining optical flow", Artificial Intelligence, vol. 17, pp. 185-203, 1981.

[13] J. L. Martín, A. Zuloaga, C. Cuadrado, J. Lázaro, and U. Bidarte, "Hardware implementation of optical flow constraint equation using FPGAs", Computer Vision and Image Understanding, vol. 98, pp. 462-490, 2005.
[14] B. D. Lucas, T. Kanade, "An iterative image registration technique with an application to stereo vision", Proc. DARPA Image understanding Workshop, pp. 121-130, 1984.
[15] J. Díaz, E. Ros, F. Pelayo, E. M. Ortigosa, and S. Mota, "FPGA-based real-time optical-flow system", IEEE Trans. Circuits and Systems for Video Technology, vol. 16, no. 2, pp. 274-279, Feb 2006.
[16] Z.Y. Wei, D.J. Lee, B.E. Nelson, and M.A. Martineau, "A fast and accurate tensor-based optical flow algorithm implemented in FPGA", IEEE WACV, Austin, Texas, USA, p. 18 (6 pages), Feb 21-22, 2007.
[17] M.M. Abutaleb, A. Hamdy, and E.M. Saad, "FPGA-Based Real-Time Video-Object Segmentation with Optimization Schemes", International Journal of Circuits, Systems, and Signal Processing, vol. 2, issue 2, pp. 78-86, 2008.