FPGA-Based Implementation of Real Time Optical Flow Algorithm and Its Applications for Digital Image Stabilization

Robert Piotrowski[1], Stanislaw Szczepanski[1], Slawomir Koziel[2]
[1]Faculty of Electronics, Telecommunications and Informatics
Gdansk University of Technology, Narutowicza 11/12, 80-233
Gdansk, Poland
[2]School of Science and Engineering
Reykjavik University, Kringlunni 1, IS-103
Reykjavik, Iceland
Emails: robert@ue.eti.pg.gda.pl, stanisla@ue.eti.pg.gda.pl, koziel@ru.is

Abstract- An efficient simplification procedure of the optical flow (OF) algorithm as well as its hardware implementation using the field programmable gate array (FPGA) technology is presented. The modified algorithm is based on block matching of subsets of successive frames, and exploits one-dimensional representation of subsets as well as the adaptive adjustments of their sizes. Also, an l_1-norm-based correlation function requiring no multiplication/division operations is used. As a result, it was possible to reduce the computational complexity of the algorithm without compromising the processing accuracy. Both the accuracy and the limitations resulting from the introduced simplifications have been verified based on several examples of both synthetic and real movie samples. The presented algorithm has been implemented using VirtexII-1000 FPGA to realize a digital stabilization system for the CMOS camera images. Experimental results fully confirm the efficiency of the presented algorithm when working with limited computational resources. This demonstrates the possibility of using our algorithm in the autonomous navigation and other battery-powered systems.

Index terms*:* Optical flow, correlation algorithm, digital image stabilization, FPGA

I. INTRODUCTION

Practical analysis of the moving objects in the sequence of images is exploited in various applications including security and monitoring systems, automatic robot control [1] and many others [2]. In some of these applications, the motion analysis is realized in battery-powered

systems with limited computational resources. Still, it is required that these systems work in real-time and provide sufficient accuracy with minimum resource usage. In such cases, it is often convenient to exploit a hardware implementation of the analysis procedures, using, e.g., FPGA [3].

One of efficient procedures of motion analysis is based on determining so-called field optical flow [4]. According to the optical flow (OF) algorithm, the previous frame can be restored based on the actual one using appropriate vectors that are determined for each pixel of the latter. The concept of the optical flow is shown in Fig. 1.

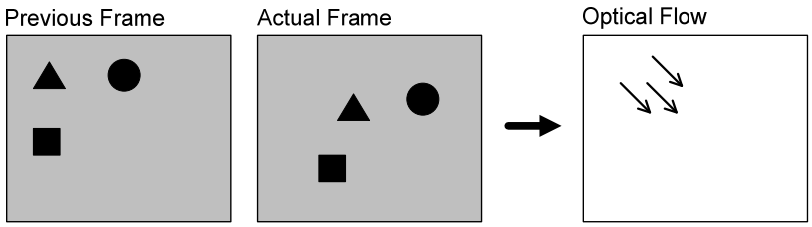


Figure 1. Optical flow concept.

Known OF algorithms can be categorized into three main groups: gradient OF (GOF) [6], frequency OF (FOF) [7] and correlation OF (COF) [13] (see also Appendix A). The first two groups can be characterized by a better processing accuracy than the correlation method as well as other ways of movement detection [5]. However, these methods are complex and their hardware realization (e.g., using FPGA [3]) consumes substantial resources. Also, the accuracy of the gradient algorithm depends on the number of iterations as well as the accuracy of the floating-point operations [3]. Additionally, the gradient method requires complex initial filtering (both in the time and the spatial domain).

Frequency OF algorithms exploit filters detecting the direction of movement in the frequency domain [7]. Filtering in the frequency domain is realized through the gradient-spatial distribution in the Fourier space. Image transformation from the spatial domain to the frequency domain is a complex process with a number of intermediate results (which requires substantial amount of memory) and significant number of multiplication operations. Both the amount of required memory and the processing accuracy have a great influence on the number of the utilized logical cells of the FPGA and the processing time (number of iterations).

Correlation optical flow algorithms exploit simple arithmetic operations, i.e., addition, subtraction, comparison, min and max. Also, they are memory efficient because only two successive frames are stored at any given time. The drawback of the correlation method is its low

accuracy. Also, it is not possible of detect the sub-pixel velocity. Processing time depends on the image resolution, maximum velocity and the subset's size. Additional details on the correlation algorithm and various correlation functions used in practice can be found in Appendix A.

Table 1 shows a qualitative comparison of the three aforementioned types of the OF algorithms. It follows that although both GOF and FOF algorithms provide the best processing accuracy, their high computational complexity, large processing time and memory usage make them unsuitable for resource-efficient hardware implementation working in real-time. COF algorithm, on the other hand, seems appropriate for such an implementation, provided that certain simplifications are introduced with respect to the standard version in order to further reduce its complexity and resource usage.

Table 1: Qualitative Comparison of Optical Flow Algorithms

	GOF	FOF	COF
Reference	[14]	[8]	[5]
Complexity	Large	Large	Small
Accuracy	Subpix	Subpix	Pix
Memory Usage	Large	Large	Small
Processing Time	Average	Large	Small

In this paper, a modification of the correlation OF algorithm is presented together with its FPGA realization. The proposed simplifications allow us to reduce both the computing time and amount of resources used by the algorithm. The adaptive procedure for adjusting the sizes of the sub-regions that are being matched when determining the optical flow was also implemented, which results in the improvement of the processing accuracy. Moreover, a substantial reduction in the computational resources used by the algorithm is obtained by using a Gray code to store the pixel intensity information. The algorithm working in real time is realized using the Virtex-II-1000 FPGA.

Reduction of power consumption and computational resources is of primary importance for wireless and battery-powered systems including autonomous robots, nodes of sensor networks and portable devices. Therefore, the OF algorithm presented here can be useful to realize—in the aforementioned systems—functions such as movement detection, time-to-crash estimation, segmentation, digital image stabilization as well as image compression.

The paper is organized as follows. In Section II the details of our simplified correlation OF algorithm are discussed. Section III presents the results of the numerical verification using several synthetic and real movies. In Section IV, the FPGA realization of our algorithm is demonstrated and compared to other implementations reported in the literature. Section V concludes the paper.

II. RESOURCE-EFFICIENT CORRELATION OPTICAL FLOW ALGORITHM

In this section modifications of the correlation OF algorithm are described that aim at reducing its computational complexity and making the algorithm resource-efficient.

A. Correlation Function

The COF algorithm is based on determining an extremum of a correlation function with respect to the two successive frames. In particular, this extremum determines so-called velocity vector **VOF** [5] which estimates the shift of a given pixel between the subsequent frames. The velocity vector for the pixel (or the group of pixels) of coordinates x and y is obtained as

$$\mathbf{VOF}[x, y] = \arg\min_{i,j} \mathbf{MC}[i, j] \tag{1}$$

where **MC** is a correlation function. The correlation function affects both the processing accuracy, computing time and the resources used by the COF algorithm. Here we exploit a simple function of the form [9], based on l_1-norm:

$$\mathbf{MC}[i, j] = \sum_{a,b=0,\ldots,n-1} \left| \mathbf{A}[a, b] - \mathbf{B}[i + a, j + b] \right| \tag{2}$$

where **A** is an $n \times n$ surround of pixel (or group of pixels) of coordinates x and y in the previous frame, **B** is the corresponding $m \times m$ surrounding in the actual frame ($m > n$), and $i, j \in \{0, 1, \ldots, m - n\}$. According to (1), a subset **A** of the previous frame is being matched to a shifted subset **B** of the actual frame; the optimal shift $[i,j]$ that realizes the extremum (here, minimum) of the correlation function (2) is determined. This shift is referred to as the velocity vector **VOF**. The process of determining the velocity vector using the correlation function (2) is illustrated in Fig. 2.

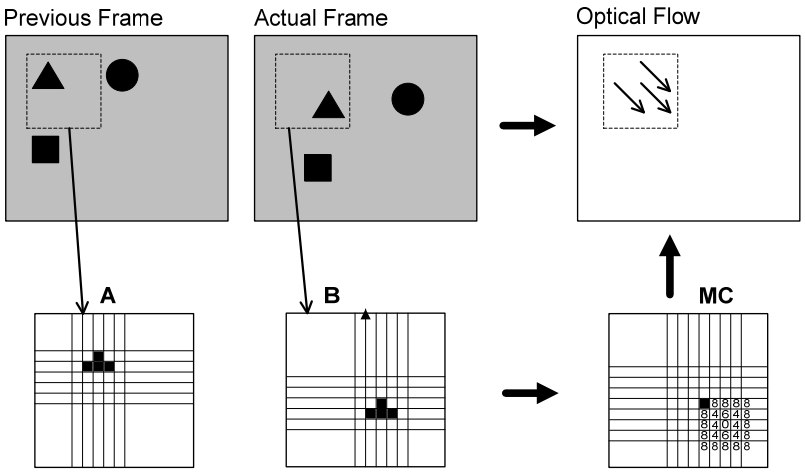


Figure 2. Illustration of determining the velocity vector using a simple correlation function (2): The subset **A** of the previous frame is being matched to a shifted subset **B** of the actual frame; the optimal shift (velocity vector) that realizes the minimum of the correlation function (2) is determined.

The correlation function (2) uses the sum of absolute differences of matrix elements. Thus, it avoids multiplication and division operations present in other commonly used functions that can be found in the literature (see Appendix A). Therefore, the function (2) realizes an efficient way of determining the correlation both with respect to the computational complexity and utilization of the system resources. Moreover, as indicated in Section 3, other correlation functions give only a marginal improvement of processing accuracy.

B. Gray-Coded Bit-Plane Matching

The number of bits required to store the pixel intensity has a major influence on the amount of resources used by the OF algorithm [11]. One of the basic methods of reducing the bit-resolution of images, exploited in this paper, is thresholding [10]. The drawback of this approach is that 8-bit comparators have to be used in the process. In order to make it more efficient, the pixel intensity can be encoded using a Gray code (CG), in which case the image segmentation can be implemented using only 2-bit XOR units. Bits of the frame encoded this way retain the information about the edges, which is essential for COF algorithms.

Figure 3 shows the subsequent bits of the image encoded using CG. CG bits carry various useful data that can be efficiently used in the adaptive OF algorithm. In particular, if the

correlation between subsequent image frames is below the required threshold, the COF algorithm is executed again using a different CG bit which contains different set of edges (see Section 2.*D*).

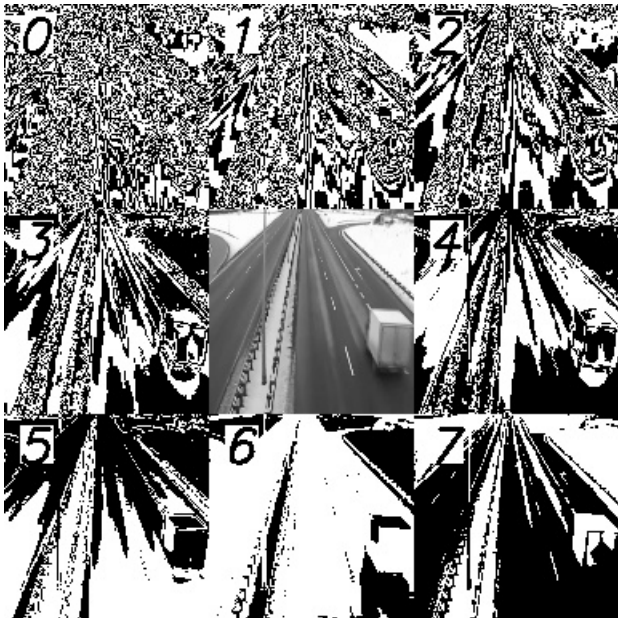


Figure 3. Gray code representation of the traffic frame. Pictures marked with numbers 0 to 7 correspond to the subsequent bits of the image presented in a central picture encoded using a Gray code.

C. One-Dimensional Subset Representation

The next step of simplifying the correlation algorithm is focused on reducing the computation time. The drawback of correlation algorithms is the necessity of performing a large number of minimizations (1), corresponding to all pixels in the image. The computing time T of COF algorithm is given by

$$T(r,k) = r^2 \cdot [T_{cor}(k^2) + T_{\arg\min}(k^2)] \tag{3}$$

where $k = m - n + 1$, r is a resolution of the original image, $T_{\text{argmin}}(x)$ is the time necessary to solve the minimization problem (1) with x being the search space size, whereas the time T_{cor} necessary to determine the correlation matrix (cf. (2)) is

$$T_{cor}(k^2) = T_{const}(m-n+1)^2 \tag{4}$$

One of possible ways of reducing the computational complexity of COF algorithm is by using one-dimensional representations of subsets **A** and **B** (cf. (2)). The computing time is then given by

$$T_{1D}(r,k) = 2r^2 \cdot [T_{cor}(k) + T_{\arg\min}(k)] \tag{5}$$

This approach was described in [12], where only the central row and column of the matrix **MC** was determined (winner-take-all or 1D-2D). In this case, the *x* component of the velocity vector **VOF** is obtained assuming the zero value for the *y* component of **VOF**. Similarly, the *y* component is calculated assuming the zero value for the *x* component. The principal drawback of this approach is its larger sensitivity for large velocity (||**VOF**|| > 3). Moreover, algorithm [12] requires substantial sizes of subsets **A** and **B**, which affects both the processing time and the accuracy (for images with the large number of edges).

In this paper, we propose an alternative method of representing two-dimensional sub-regions in one-dimensional space. Our method is based on summing elements of subsets **A** and **B** along rows and columns. This approach results in reduced correlation algorithm (RCOF), where matrices **A** and **B** are replaced by the two vectors **H** and **V** given by

$$\mathbf{V_A}[i] = \sum_{j=0}^{n-1} \mathbf{A}[i, j] \tag{6}$$

$$\mathbf{H_A}[j] = \sum_{i=0}^{n-1} \mathbf{A}[i, j] \tag{7}$$

$$\mathbf{V_B}[k] = \sum_{j=0}^{m-1} \mathbf{B}[k, j] \tag{8}$$

$$\mathbf{H_B}[l] = \sum_{i=0}^{m-1} \mathbf{B}[i, l] \tag{9}$$

where $i, j = 0, \ldots, n-1$, and $k, l = 0, \ldots, m-1$. The correlation between vectors (6) to (9) is determined according to (2). Subsequently, the velocity vector **VOF** is obtained using (1).

D. Adaptive Adjustment of Subset Sizes

Similarly as for the 1D-2D algorithm, the accuracy of the RCOF algorithm is dependent on the proper choice of parameters *m* and *n* of subsets **A** and **B**. It is expected that too small *m* and *n* will result in degrading the accuracy of velocity estimation for large shifts, while large *m* and *n* may degrade the accuracy for the movies with a large number of edges and containing independently moving objects. Having this in mind, an adaptive correlation algorithm (ARCOF) has been

proposed to address the above issues to some extent. The ARCOF algorithm dynamically adjusts the values of parameters *n* and *m* according to the following rules:

- Both *n* and *m* are set to their initial values: $n = n_{\min}$ and $m = m_{\min}$;
- If the correlation parameter than realizes (2) is smaller than $\text{corre}_{\min}$ (the user-defined threshold value), *n* and *m* are incremented by 1 and the matrix **MC** is determined again as in (2);
- Exception 1: If *m* exceeds a user-defined maximum value $m_{\max}$, only *n* is incremented;
- Exception 2: If $m = m_{\max}$ and $n = m_{\max} - 1$, *n* and *m* are reset to $n = n_{\min}$ and $m = m_{\min}$, respectively; the adaptation procedure is restarted with a different CG bit.

It should be emphasized that the improvement in accuracy due to increasing *m* and *n* is obtained at the expense of the longer computing time. The trade-off between these two factors is determined, in the adjustment procedure used in the ARCOF algorithm, by the assumed accuracy level through the threshold value.

III. EXPERIMENTAL RESULTS

In this section we present the results of experiments carried out to verify the performance of the modified COF algorithm described in Section 2.

A. Error Measure

Definition of the processing error is often a problem in case of image processing algorithms. The reason is that the analytical relations are not known for the optical flow of the real images, and estimators must be used as a reference value of **VOF**. Therefore, the processing error obtained using the OF estimator depends on the quality of this estimator. Moreover, OF algorithms are based on assumptions regarding the maximum pixel shift between the subsequent frames, as well as the assumptions regarding the frame structure (cf. the aperture problem [6]). Often, these assumptions are not satisfied, e.g., in case of insufficient number of good matchable features in the image, which results in a large absolute error of the estimator.

In this paper, the following formula describing the processing error has been used to qualitative analysis of the algorithms

$$Err = \frac{1}{p^2}\sum_{i=0}^{p}\sum_{j=o}^{p}\left|\mathbf{I}_{\mathbf{Prev}}[i,j] - \mathbf{I}'_{\mathbf{Act}}[i,j]\right| \tag{10}$$

where p is an initial frame size, $\mathbf{I}_{\mathbf{Prev}}$ is the previous frame, whereas $\mathbf{I'}_{\mathbf{Act}}$ is a restored previous frame obtained as a result of processing the current frame and the corresponding **VOF**. This approach is based directly on the formulation of the OF algorithm where the previous frame can be restored (approximated) using the current frame and the optical flow field.

In order to permit comparison of our algorithm with other COF approaches described in the literature, an additional error measure—so-called average angle error (AAE)—introduced in [4] and [20] will also be used. AAE is defined as

$$AAE = \frac{1}{p^2}\sum_{i=0}^{p}\sum_{j=o}^{p}\mathbf{E}_A[i,j] \tag{11}$$

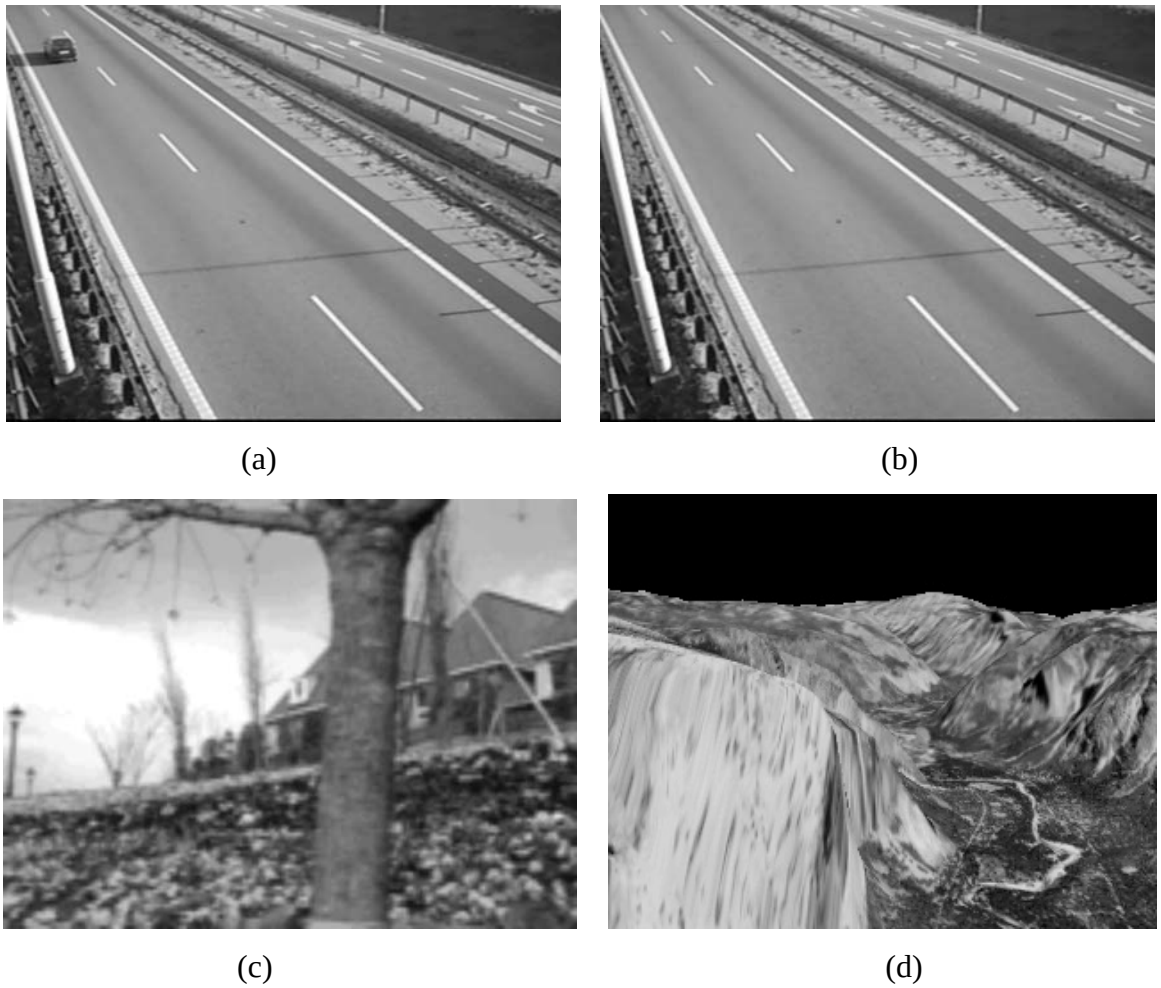
with $E_A[i, j]$ given by

$$\mathbf{E}_A[i,j] = \cos^{-1}(\hat{\mathbf{c}}[i,j] \bullet \hat{\mathbf{e}}[i,j]) \tag{12}$$

Here, $\boldsymbol{c}$ is the correct motion vector, $\boldsymbol{e}$ is the estimated optical flow vector. Symbol ˆ denotes vector normalization, whereas symbol $\bullet$ denotes the inner product.

B. Test Movies

The processing quality of the OF algorithms is also affected by the maximum velocity ($\mathbf{VOF}_{\max}$) and the number of characteristic elements (e.g., edges, corners). In order to verify the algorithm proposed in this work, several types of movies have been used to analyze the processing error. Movie #1 contains a piece of road together with the vehicles moving along, whereas Movie #2 contains the same piece of road, however, without the vehicles. Movie #3 is characterized by large shifts, and Movie #4 is a synthetic image sequence for which the ground-truth optical flow is known. Figure 4 shows the 8th frame from all the movies.



(a) (b)

(c) (d)

Figure 4. The movies used for OF algorithms verification (frame number 8): (a) Movie #1: a piece of road together with the vehicles moving along, (b) Movie #2: the same piece of road, no vehicles, (c) Movie #3: flower-garden sequence, (d) Movie #4: Yosemite sequence.

C. Results and Discussion

We have performed a number of experiments in order to verify the influence of the modifications of the COF algorithm introduced in Section 2 on the processing accuracy as well as the two factors important from the point of view of the hardware realization of the algorithm: the computing time and the resource usage.

Table 2: Exploited Resources and Processing Accuracy for The Correlation Optical Flow Algorithm versus Image Resolution

	Image Resolution			
	1 bit	2 bits	4 bits	8 bits

Resources	Adder	4 slice	5 slice	5 slice	6 slice
	Comparator	4 slice	5 slice	5 slice	6 slice
	Memory	2 BRAM	4 BRAM	8 BRAM	16 BRAM
Processing Accuracy*	Movie #1	12.1	11.9	10.8	8.1
	Movie #2	12	11.8	10.4	7.9

* Accuracy according to formula (10)

Table 2 shows the dependence of the processing accuracy and the required resources on the image bit-resolution for the standard COF algorithm. In practice, selection of the image grayscale depth is determined by the necessity of reducing the number of exploited memory blocks (BRAM) [17], which, in turn, is a consequence of limited resources of fast BRAMs available in FPGA circuits. Of course, reduction of exploited BRAM units can be achieved at the expense of a processing accuracy as indicated in Table 2. The number of logic units (slices), on the other hand, is only slightly dependent on the grayscale depth of the image.

Based on the data presented in Table 2, a binary (1-bit resolution) image was selected for further realization. In this case only 2 BRAM units are necessary. Also, the processing accuracy is almost the same as for 2- and 4-bits resolution (cf. Table 2).

Table 3: Processing Accuracy* of RCOF Algorithm versus m and n

	Movie			
	#1	#2	#3	#4
$2n = m = 32$	13.3	13.2	13.0	12.8
$2n = m = 16$	13.4	13.5	14.5	14.1
$4n = m = 32$	13.5	13.5	16.6	16.0
$4n = m = 16$	13.5	13.6	16.8	16.1
$8n = m = 32$	13.5	13.8	18.2	17.9
$8n = m = 16$	13.6	13.9	19.0	18.1

* Accuracy according to formula (10)

Table 3 shows the processing error of the RCOF algorithm for different values of m and n. These results indicate that the best accuracy is obtained for large values of m (i.e., large searching regions A) and large n (large neighborhoods B). Obviously, the consequence of large m is a large

size of **MC** matrix (2) which influences the processing time (cf. (3) and (4)). For the hardware realization of the adaptive OF algorithm (cf. Section 2.*D*) the following parameters were selected: $m_{max} = 16$, $m_{min} = 8$, $n_{min} = 4$. This choice ensures a reasonable trade-off between the accuracy and processing time.

Table 4: Correlation Algorithms: Comparison of Processing Accuracy*

	Movie			
	#1	#2	#3	#4
GOF	4.0	3.6	3.7	2.4
[12]	12.9	12.5	16.1	14.0
COF	11.2	11.3	11.6	11.1
RCOF	13.3	13.2	13.0	12.8
ARCOF**	9.1	9.3	11.0	10.8

* Accuracy according to formula (10)

** ($corre_{min} = 0.96$, $m_{max} = 16$, $m_{min} = 8$, $n_{min} = 4$)

Table 4 compares the processing error for the GOF algorithm, the 1D-2D algorithm [12], as well as RCOF and ARCOF algorithms. As expected the GOF algorithm is characterized by the smallest error for all four test movies considered. The adaptive procedures used in ARCOF algorithm have a clear impact on the processing error as indicated by comparison with the RCOF algorithm. Also, the ARCOF algorithm outperforms the 1D-2D algorithm [12]. It should be stressed that although the accuracy of ARCOF is not as good as the accuracy of GOF or FOF algorithms, the ARCOF approach requires the least amount of resources, and, it is therefore much more suitable for hardware implementation.

IV. HARDWARE REALIZATION IN FPGA

The ARCOF algorithm has been selected for the FPGA realization because it is using the minimum amount of resources and provides sufficient accuracy in computing of the optical flow field. Based on the OF field obtained by means of ARCOF, a digital image stabilization (DIS) algorithm [15] have been realized. The DIS algorithm eliminates unwelcome vibrations in the movies.

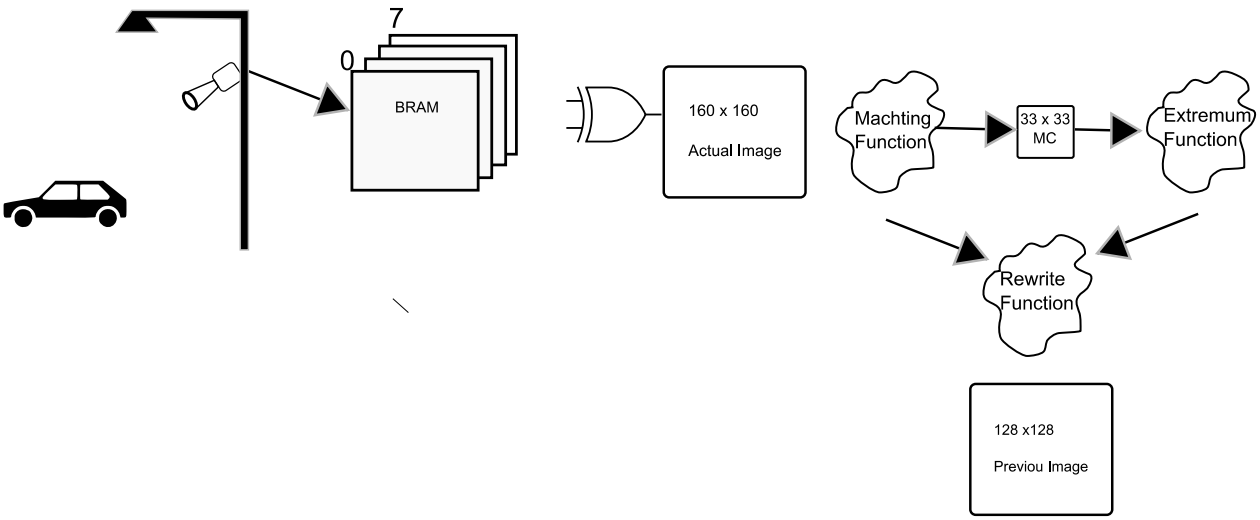


Figure 5. Block diagram of a DIS system realized with FPGA. The image from the camera is stored in BRAM. Image segmentation is performed using XOR. The result of comparing appropriate subsets of the actual and previous frames is stored in the MC module; then, the optical flow vector is determined in the Extremum Function module. Having the optical flow field determined, the DIS algorithm is realized in the Rewrite Function module.

Our system is shown in Fig. 4 and consists of the module computing the optical flow field, module determining the velocity vector (cf. (1)), as well as the module responsible for changing the location of pixel in the output matrix (i.e., actual realization of DIS). The absolute value of the difference has been selected as the correlation function (cf. (2)). The ADS operation was realized using 2-input XOR gate as well as the adder with D flip-flop.

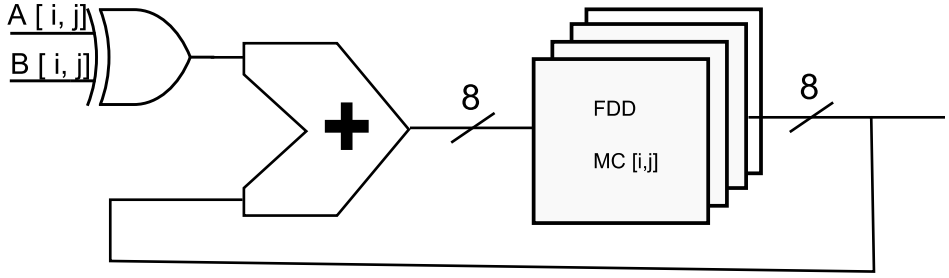


Figure 6. Block diagram of the ARCOF module. The absolute sum of differences is realized using XOR (subtraction) and an adder (with memory implemented with D flip-flops).

A block diagram of the ARCOF algorithm is shown in Fig. 5. The module seeking for the maximum of the correlation function stores only a single value of the **MC** matrix as well as the

corresponding argument. Having found the maximum for a single pixel, a module realizing the pixel shift or the change of the CG bit (in case of insufficient correlation) is activated.

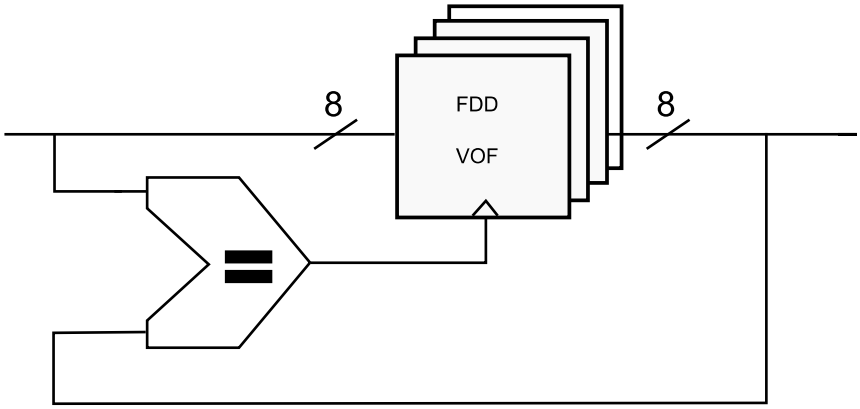
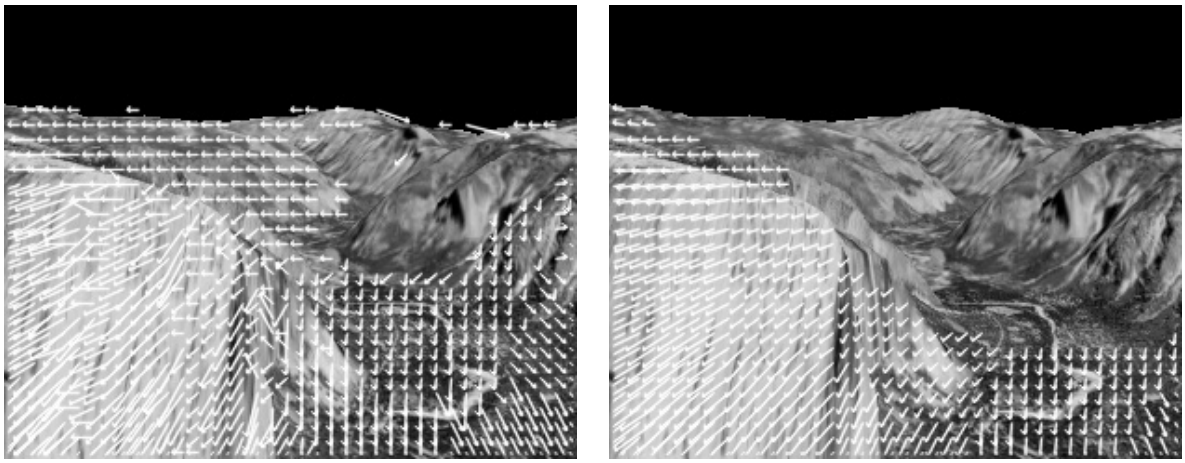


Figure 7. Block diagram of the module finding the maximum of the correlation function. The module uses comparator as well as D-flip-flop-based memory.

A block diagram of the maximum seeking module is shown in Fig. 6. The last element of the system is a module that shifts a pixel from the input frame to the proper position at the output frame (according to the velocity vector **VOF** found using (1)). Figure 7 shows the block diagram of the module finding the maximum of the correlation function. The module uses comparator as well as D-flip-flop-based memory.

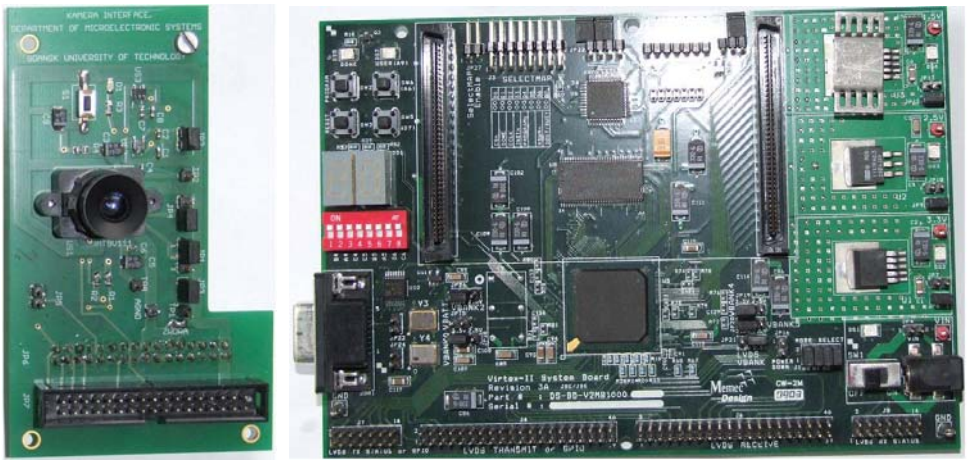
The input image (resolution 640×480, 30 fps) has been stabilized in real time to the output resolution of 128×128 (30 fps). The whole system (DIS + handling the CMOS camera) was using 3105 slice (which is 60% of available FPGA resources, 5120 slice).



(a) (b)

Figure 8. The synthetic movie used for OF algorithms verification (frame number 8): (a) ARCOF algorithm, (b) ground-truth optical flow.

For the sake of illustration, Fig. 8 shows the optical flow field determined with our algorithm as well as the ground-truth optical flow for the test Movie #4. Figure 9 shows the camera and the FPGA circuit used to realize the DIS algorithm.



(a) (b)

Figure 9. Hardware components: (a) CMOS imager, (b) Virtex-II-1000-Board.

Table 5 shows a comparison between our algorithm and algorithms presented in [3], [18] and [19]. The comparison was performed with respect to the amount of resources necessary to implement the algorithm, power consumption as well as the accuracy using error measure (12). It should be emphasized that our algorithm uses substantially less resources and is characterized by the smallest power consumption which is crucial for low-power and battery-powered systems. Obviously, the processing accuracy is somewhat compromised, particularly in comparison with [18], however, this was to be expected according to the trade-offs discussed in Section 3.

Table 5: Exploited Resources and Power Consumption for The Correlation Optical Flow Algorithm versus Show In [18], [19], [3]

Algorithm		This work	[18]	[19]	[3]
Resources	OF module	2700 slice	8520 slice	10288	not indicated
	Others	315 slice	12961	3681	not indicated
	Total	3105 slice	21481 slice	13969 slice	19000 slice

Power	480 mW	not indicated	2 W	not indicated
Resolution	640 × 480	640 × 480	640 × 480	320 × 240
Frame per sec	30	60	64	30
Processing Accuracy* for Movie #4	17.3°	6.5°	12.7°	18.3°

* Accuracy according to formula (12)

V. CONCLUSION

The paper presents the algorithm for determining the field optical flow with reduced computational complexity and resource usage. The proposed approach is based on determining the correlation between one-dimensional representations of the image subsets and on the adaptive adjustment of the subset sizes. It has been demonstrated, using several movie examples, that our algorithm ensures processing accuracy only slightly worse than the COF algorithm. The presented FPGA implementation exploits substantially less resources and exhibits lower power consumption than other implementations reported in the literature. Therefore, our OF algorithm is suitable for realizing various functions involving movement detection, digital image stabilization as well as image compression in battery-powered systems using limited computational resources.

APPENDIX A: CORRELATION FUNCTIONS FOR OPTICAL FLOW ALGORITHMS

The processing accuracy, processing time and the hardware resources used by the COF algorithm depend on the correlation function. A simple correlation function (2) using the sum of absolute differences of matrix elements has been shown in Section 2. Several commonly used correlation functions can be found in the literature. For example, [16] describes the following set of functions:

$$\mathbf{S}[i,\, j] = \sum_{a=0}^{n-1} \sum_{b=0}^{n-1} \left(\mathbf{A}[a,b] - \mathbf{B}[i+a,\, j+b] \right)^2 \tag{13}$$

$$\mathbf{S}[i,\, j] = \frac{\sum_a \sum_b \left(\mathbf{A}[a,b] - \mathbf{B}[i+a,\, j+b] \right)^2}{\sqrt{\sum_a \sum_b \left(\mathbf{A}[a,b] \right)^2 \cdot \sum_a \sum_b \left(\mathbf{B}[i+a,\, j+b] \right)^2}} \tag{14}$$

$$\mathbf{C}[i,\, j] = \sum_{a=0}^{n-1}\sum_{b=0}^{n-1}\left(\mathbf{A}[a,b]\cdot\mathbf{B}[i+a,\, j+b]\right)^2 \tag{15}$$

$$\mathbf{C}[i,\, j] = \frac{\sum_a\sum_b\left(\mathbf{A}[a,b]\cdot\mathbf{B}[i+a,\, j+b]\right)^2}{\sqrt{\sum_a\sum_b\left(\mathbf{A}[a,b]\right)^2\cdot\sum_a\sum_b\left(\mathbf{B}[i+a,\, j+b]\right)^2}} \tag{16}$$

$$\mathbf{C}[i,\, j] = \sum_{a=0}^{n-1}\sum_{b=0}^{n-1}\left(\tilde{\mathbf{A}}[a,b]\cdot\tilde{\mathbf{B}}[i+a,\, j+b]\right)^2 \tag{17}$$

$$\mathbf{C}[i,\, j] = \frac{\sum_a\sum_b\left(\tilde{\mathbf{A}}[a,b]\cdot\tilde{\mathbf{B}}[i+a,\, j+b]\right)^2}{\sqrt{\sum_a\sum_b\left(\tilde{\mathbf{A}}[a,b]\right)^2\cdot\sum_a\sum_b\left(\tilde{\mathbf{B}}[i+a,\, j+b]\right)^2}} \tag{18}$$

$$\tilde{\mathbf{B}}[a,b] = \tilde{\mathbf{B}}[a,b] - \bar{\mathbf{B}}[i,\, j] \tag{19}$$

$$\tilde{\mathbf{A}}[a,b] = \mathbf{A}[a,b] - \bar{\mathbf{A}} \tag{20}$$

$$\bar{\mathbf{B}}[i,\, j] = \frac{1}{n^2}\sum_{a=0}^{n-1}\sum_{b=0}^{n-1}\mathbf{B}[i+a,\, j+b] \tag{21}$$

$$\bar{\mathbf{A}} = \frac{1}{n^2}\sum_{a=0}^{n-1}\sum_{b=0}^{n-1}\mathbf{A}[a,b] \tag{22}$$

and $i, j = 0, ..., m - n + 1$.

Table 6: Processing Accuracy* of Correlation Optical Flow Algorithms versus Correlation Function Used

Correlation Function	Movie			
	#1	#2	#3	#4
(2)	12.1	12.0	12.3	11.9
(13)	11.8	11.8	11.7	11.5
(14)	11.8	11.8	11.7	11.3
(15)	11.5	11.6	11.7	11.2
(16)	11.4	11.6	11.7	11.2
(17)	11.4	11.4	11.6	11.1
(18)	11.2	11.3	11.6	11.1

* Accuracy according to formula (10)

Table 7: Correlation Optical Flow Algorithms: Properties and Exploited Resources

Correlation Function	**VOF** implementation	Relative Execution Time	Resources		
			Comp	Sum	Mult
(2)	$\mathbf{VOF}[x,y] = \text{argmin}\{i,j : \mathbf{S}[i,j]\}$	1.0	6 slice	6 slice	-
(13)	$\mathbf{VOF}[x,y] = \text{argmin}\{i,j : \mathbf{S}[i,j]\}$	1.6	11 slice	11 slice	4 slice
(14)	$\mathbf{VOF}[x,y] = \text{argmin}\{i,j : \mathbf{S}[i,j]\}$	3.1	11+8 slice	19 slice	8 slice
(15)	$\mathbf{VOF}[x,y] = \text{argmax}\{i,j : \mathbf{C}[i,j]\}$	1.7	22 slice	11 slice	4 slice
(16)	$\mathbf{VOF}[x,y] = \text{argmax}\{i,j : \mathbf{C}[i,j]\}$	3.3	22+8 slice	19 slice	16 slice
(17)	$\mathbf{VOF}[x,y] = \text{argmax}\{i,j : \mathbf{C}[i,j]\}$	1.8	22 slice	11 slice	4 slice
(18)	$\mathbf{VOF}[x,y] = \text{argmax}\{i,j : \mathbf{C}[i,j]\}$	3.4	22+8 slice	19 slice	16 slice

The above expressions contain multiplication and division operations, which, in case of FPGA realization, is a serious problem because of a limited number of modules performing multiplication [17]. Table 6 shows the processing accuracy for the correlation functions (2) and (13)-(18), whereas Table 7 summarizes the resources used in their realization. This data shows that accuracy improvement is not proportional to the amount of resources. Moreover, the normalized functions (12), (14) and (16) give a marginal improvement of accuracy. The correlation function (2) is the most efficient method of determining the **MC** matrix both with respect to the processing time and utilization of the system resources.

ACKNOWLEDGEMENT

This work was supported in part by the Ministry of Science and Higher Education of Poland under the Research and Development Project N N515 423034.

REFERENCES

[1] S. Chhaniyara, P. Bunnun, L. Seneviratne and M. Ichikawa, ”Optical Flow Algorith for Velocity Estimation of Ground Vehicles: A Feasibility Study”, International Journal on Smart Sensing and Intelligent Systems, Vol. 1, No. 1, March 2008.

[2] G. Barrows, C. Neely: ”Mixed-mode VLSI optic flow sensors for inflight control of a micro air vehicle”, Proc. SPIE Vol. 4109, Critical Technologies for the Future of Computing; pp. 52-63, 2000.

[3] Diaz, J., Ros, E., Pelayo, F., Ortigosa, E. M., Mota, S.: ”FPGA based realtime optical-flow system”; IEEE Transactions on Circuits and Systems for Video Technology, 16, 2 (2006), 274-279.

[4] B. McCane, K. Novins, D. Crannitch, and B. Galvin, ”On benchmarking optical flow”, Comput. Vis. Image Understanding, vol. 84, pp. 126-143, 2001.

[5] Barron J. L, Fleet D. J, and Beauchemin S. S. , ”Performance of optical flow techniques.” International Journal of Computer Vision, Vol. 12, pp. 43-77, 1994.

[6] B.D. Lucas and T. Kanade, ”An Iterative Image Registration Technique with an Application to Stereo Vision (IJCAI),” Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI ’81), April, 1981, pp. 674-679.

[7] Jangheon Kim, Thomas Sikora: ”Hybrid recursive energy-based method for robust optical flow on large motion fields.”, ICIP (1) 2005: 129-132

[8] Bruhn, J. Weickert, C. Feddern, T. Kohlberger, C. Schnrr, “Real- Time Optic Flow Computation with Variational Methods”, CAIP 2003, LNCS, vol. 2756, pp. 222-229, 2003.

[9] P. Cobos and F. Monasterio, ”FPGA implementation of the Horn & Schunck optical flow algorithm for motion detection in real time images”, Proc. DCIS’98 XIII, pp.616-621, 1998.

[10] S.J. Ko, S.H. Lee, and K.H. Lee, ”Digital image stabilizing algorithms based on bit-plane matching”, IEEE Trans. Consumer Electron., vol. 44, no. 3, pp.796-800, Aug., 1998.

[11] L. Xu and X. Lin, ”Digital image stabilization based on circular block matching”, IEEE Transactions on Consumer Electronics, vol. 52, no. 2, pp. 566-574, 2006.

[12] N. Ancona and T. Poggio, ”Optical Flow from 1D Correlation: Application to a simple Time-To-Crash Detector”, Fourth International Conference on Computer Vision, IEEE Computer Society Press, May 11-14, 1993, Berlin, Germany, pp. 209-214.

[13] K. Janschek, V. Tchernykh, M. Beck: ”Optical Flow based Navigation for Mobile Robots using an Embedded Optical Correlator”, Preprints of the 3rd IFAC Conference on Mechatronic Systems - Mechatronics 2004, 6-8 September 2004, Sydney, Australia, pp.793-798.

[14] P. C. Arribas and F. M. H. Maciá. FPGA implementation of the Horn & Shunk Optical Flow Algorithm for Motion Detection in real time Images. Proceedings of the XIII Design of Circuits and Integrated Systems Confer- ence, pages 616_621, 1998.

[15] S. Erturk, ”Digital image stabilization with sub-image phase correlation based global motion estimation”, IEEE Trans. Consumer Electron., vol. 49, no. 4, pp.1320-1325, Nov., 2003.

[16] Intel Corporation, ”Open Source Computer Vision Library”, Intel Corporation, http://developer.intel.com, 2000.

[17] Xilinx Company, ”Virtex-II Pro and Virtex-II Pro X FPGA User Guide”, 2004.

[18] Z. Wei, D.-J. Lee, B. Nelson, James K. Archibald, and Barrett B. Edwards, ”FPGA-Based Embedded Motion Estimation Sensor”, International Journal of Reconfigurable Computing Volume 2008 (2008).

[19] Z. Wei, D.-J. Lee, B. Nelson, and M. Martineau, “A fast and accurate tensor-based optical flow algorithm implemented in FPGA,” in Proceedings of the IEEE Workshop on Applications of Computer Vision (WACV ’07), p. 18, Austin, Tex, USA, February 2007.

[20] S. Baker, D. Scharstein, J.P. Lewis, S. Roth, M. J. Black, and R. Szeliski, “A Database and Evaluation Methodology for Optical Flow”, IEEE International Conference on Computer Vision, Oct. 2007.