Hardware implementation of multi-scale Lucas-Kanade optical flow computation algorithm – a demo

Krzysztof Blachut
AGH University of Science
and Technology Krakow, Poland
E-mail: kblachut@student.agh.edu.pl

Tomasz Kryjak, *Member IEEE*
AGH University of Science
and Technology Krakow, Poland
E-mail: tomasz.kryjak@agh.edu.pl

Marek Gorgon, *Senior Member IEEE*
AGH University of Science
and Technology Krakow, Poland
E-mail: mago@agh.edu.pl

***Abstract*—Motion detection is one of the most important components in image processing and analysis systems. It can be identified as a change in location or size of an object on subsequent frames. Moreover, the position of object, camera or both may change. Motion velocity and direction can be determined on the basis of the displacement of individual pixels and frame acquisition frequency. To calculate the displacement, optical flow methods can be used. One of them – the Lucas-Kanade algorithm – was used in this work. The implementation uses a multi-scale method. This allows the detection of displacements of objects over greater distances. For hardware implementation the FPGA platform was used because of the possibility of parallel calculation and energy efficiency. In this work real-time video stream processing with a resolution of 1280x720 pixels and a frequency of 50 frames per second was obtained. The module could be used in embedded vision systems such as surveillance or autonomous vehicles.**

***Index Terms*—FPGA, hardware implementation, real-time vision system, motion detection, optical flow, Lucas-Kanade algorithm, multi-scale method**

I. Introduction

The information about the displacement of pixels between consecutive frames in a video sequence is described by optical flow. It is a vector field in which each pixel corresponds to two components, defining its shift in the vertical and horizontal direction. Information about the pixel motion can be used in more complex algorithms such as the detection, segmentation and object tracking [2]. The Lucas-Kanade algorithm has already been implemented on hardware platforms such as an FPGA (Field Programmable Gate Array) or a GPU (Graphics Procssing Unit), as well on a GPP (General Purpose Processor).

In some cases it may be beneficial to apply the multi-scale approach suggested in papers [1], [2], [3]. This method is much better than the one-scale version, because it allows the detection of large displacements between frames. This topic has already been presented in the paper [1], where processing a 640x480 @ 32 fps video stream was obtained.

The work presented in this paper was supported by AGH University of Science and Technology project no. 11.11.120.714

II. The used algorithm

In the Lucas-Kanade algorithm (LK), a solution to the Equation (1) is searched. The pixel displacement is denoted as *(u, v)* and I_x, I_y and I_t are spatial and temporal derivatives respectively.

$$I_x u + I_y v + I_t = 0 \tag{1}$$

One of the solutions of Equation (1) was proposed by Bruce D. Lucas and Takeo Kanade in the work [4]. It required an additional assumption – the displacement of pixels in a small neighbourhood is the same.

The equation for pixels in the considered window can be presented in a matrix form (2). If the matrix is reversible, then the solution exists and u, v can be obtained. On both sides of the expression, the *W* matrix may appear, which gives more weight to pixels in the centre of the window.

$$\begin{pmatrix} u \\ v \end{pmatrix} = -\begin{pmatrix} \sum w_i I_x I_x & \sum w_i I_x I_y \\ \sum w_i I_x I_y & \sum w_i I_y I_y \end{pmatrix}^{-1} \begin{pmatrix} \sum w_i I_x I_t \\ \sum w_i I_y I_t \end{pmatrix} \tag{2}$$

However, for fast-moving objects the above presented version of the LK algorithm is insufficient, because in this case the pixel displacement is too large and the condition of analysing a small neighbourhood is violated. In such a situation, a multi-scale approach should be applied. It requires generating a pyramid of images for each frame of the video sequence [3] . The original frame is reduced twice, then the newly created image is again reduced twice, etc.

In the next step optical flow is computed for the image in the smallest scale according to the Lucas-Kanade algorithm. The obtained flow is used to modify the previous image in the lager scale – details in [1]. For the frame after transformation and the current one, the optical flow is calculated again.

The whole procedure is repeated until the flow is calculated for the input image. Figure 1 presents a diagram according to which the calculations are performed on successive scales for the test sequence *Seat* used in [5].

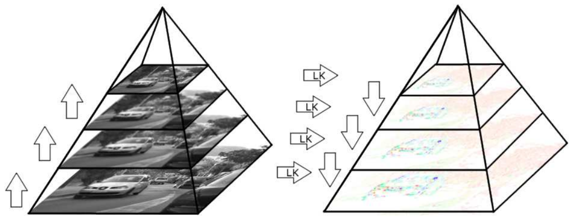

Fig. 1. A pyramid of images for one frame

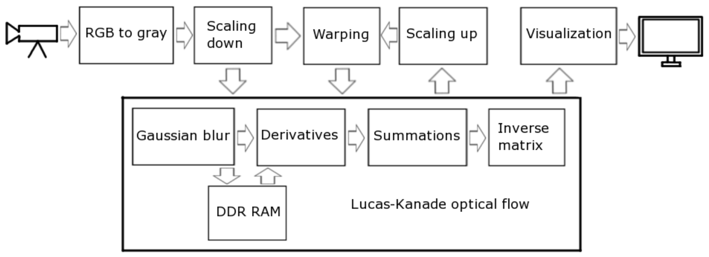

Fig. 2. Scheme of the LK computing system

III. Hardware implementation

The described Lucas-Kanade module was verified on the VC707 development board equipped with a Virtex-7 FPGA device from Xilinx and external DDR RAM memory resources. The scheme of the system is presented in Figure 2.

The single-scale version of the LK algorithm consists of several modules. All of them were prepared in Vivado Desing Suite using Verilog language. Firstly the incoming frame is converted from the RGB colour space to gray-scale.

Then, the frame is convoluted with a one-dimensional mask [1 4 6 4 1]/16, which is the approximation of a Gaussian distribution. Blurring of the image is applied in rows and then in columns.

In the next module, two frames are analysed simultaneously – the previous one (stored in external RAM) and the current from the camera. Then the spatial derivatives I_x, I_y on the previous frame and the temporal one I_t (between previous and current frame) are computed. To calculate spatial derivatives the mask [-1 0 1]/2 (calculated for rows and columns) is used, while the mask [-1 1]/2 is used to compute temporal derivatives.

Then the corresponding derivatives are multiplied by themselves and added in the window. As part of the compromise between resource consumption and accuracy, the window size was set to 5x5. Weights w_i in Equation (2) are values of a two-dimensional Gaussian function. The last step before calculating the flow for each pixel is matrix inversion as in [5]. In addition, thresholding is applied, which allows to filter results that may be incorrect.

The described modules enables LK computations in one scale. The masks' sizes used in the algorithm were chosen as a compromise between accuracy and resource consumption. For the multi-scale method, it was necessary to implement some additional modules.

The first of them is responsible for reducing the size of the image twice. For this purpose the pixels in rows and columns with odd indexes are chosen. The flow *(u, v)* obtained in a smaller scale is then up-scaled using bilinear interpolation. A displacement along the x and y axes is specified for each pixel, so it is possible to modify the picture in a larger scale by initially shifting pixels. The use of this method enables motion compensation. Therefore, the assumption about small displacement used in the LK method is not violated.

The flow obtained for each pixel is shown graphically – (*Visualisation*). In the case of hardware implementations the results are often presented in the form of colours in the HSV space. Pixel colour corresponds to the direction in which they move and its saturation corresponds to the speed of the movement. Firstly, the length of the vector is calculated. Then with the help of the pre-computed values of the arc tangent function (look-up table), the angle specified by *(u, v)* is determined. In addition the conversion to the RGB color space is made to correctly display the results on the screen.

In Table I the total consumption of resources on the VC707 board is presented. In addition to the Lucas-Kanade algorithm module, the resources were used to support the external memory controller and video pass-through.

TABLE I
FPGA resource utilization for the VC707 board

Resource type	Used	Available	Percentage
LUT	41167	303600	13,56%
Flip-Flop	52324	607200	8,62%
Block RAM	186	1030	18,01%
IOB	177	700	25,29%

IV. Summary

In this paper a hardware implementation of the multi-scale version of the Lucas-Kanade optical flow computation algorithm using FPGA platform was presented. Thanks to an appropriate parallel-pipelined architecture, real-time processing of a 1280x720 @50 fps video stream has been achieved. The multi-scale method enables the detection of large displacements between frames, which allows correct operation for fast-moving objects.

References

[1] F. Barranco, M. Tomasi, J. Diaz, M. Vanegas, E. Ros. ”Parallel Architecture for Hierarchical Optical Flow Estimation Based on FPGA”. In: IEEE 20.6 (2012), pp. 1058-–1067.

[2] J.Y. Bouguet. ”Pyramidal Implementation of the Affine Lucas Kanade Feature Tracker. Description of the algorithm”. Tech. note, Intel Corporation, 2000.

[3] J. Bergen, P. Anandan, K. Hanna, R. Hingorani. ”Hierarchical model-based motion estimation”. In: Computer Vision ECCV'92 588 (1992), pp. 237—252.

[4] B.D. Lucas, T. Kanade. ”An Iterative Image Registration Technique with an Application to Stereo Vision (DARPA)”. In: Proceedings of the 1981 DARPA Image Understanding Workshop. 1981, pp. 121-–130.

[5] D. Bagni, P. Kannan, S. Neuendorffer. ”Demystifying the Lucas-Kanade Optical Flow Algorithm with Vivado HLS”. Tech. note XAPP1300. Xilinx, 2017.