EFFICIENT REAL-TIME LOCAL OPTICAL FLOW ESTIMATION BY MEANS OF INTEGRAL PROJECTIONS

Tobias Senst, Volker Eiselein, Michael Pätzold and Thomas Sikora

Technische Universität Berlin
Communication Systems Group
EN 1, Einsteinufer 17, 10587 Berlin, Germany

ABSTRACT

In this paper we present an approach for the efficient computation of optical flow fields in real-time and provide implementation details. Proposing a modification of the popular Lucas-Kanade energy functional based on integral projections allows us to speed up the method notably. We show the potential of this method which can compute dense flow fields of 640x480 pixels at a speed of 4 fps in a GPU implementation based on the OpenCL framework. Working on sparse optical flow fields of up to 17,000 points, we reach execution times of 70 fps. Optical flow methods are used in many different areas, the proposed method speeds up current surveillance algorithms used for scene description and crowd analysis or Augmented Reality and robot navigation applications.

Index Terms**—** optical flow, feature tracker, KLT, integral projection, real-time

1. INTRODUCTION

Computation of optical flow is a common topic in the computer vision community whose applications range from motion estimation to point tracking. Based on the seminal works of Lucas-Kanade [1] and Horn-Schunck [2] a diverse range of optical flow estimation techniques have been developed.

Motion is an important cue in many computer vision tasks. Applications as robot navigation, Augmented Reality, visual attention and camera self-calibration require very fast detection of interest points (feature points) and the subsequent search for potential correspondences in real-time. Therefore, methods with excellent runtime performance exploiting local optical flow techniques as the popular KLT tracking algorithm [3] in many cases are still applied and can outperform more accurate but also time consuming feature tracker e.g. SIFT [4]. Recent work was done to improve the run-time performance by parallelising the algorithm and porting it onto a GPU [4, 5, 6, 7]. In the approach of the II-LK method [8] the run-time performance was enhanced by reducing the computational complexity drastically using integral images and applying an additional approximation, which results in an decreased accuracy but is more in reliable applications as robot navigation that are restricted by limited hardware. In this paper we present a modification of the Lucas-Kanade method based on the use of integral projections with a significantly improved performance by reducing the computational complexity compared to the standard PLK and increase the accuracy compared to the II-LK. A parallelized GPU implementation allows then the computation of large flow fields in real-time.

The paper is organized as follows: In section 2 we explain our approach of using integral projections within the Lucas-Kanade method. In section 3 we show our results, evaluate the performance compared to recent techniques and section 4 concludes the paper.

2. MODIFYING THE LUCAS-KANADE METHOD BY INTEGRAL PROJECTIONS

The method we describe combines the Lucas-Kande algorithm with the technique of integral projections which was applied by [9] for global motion estimation.

The common starting point for optical flow algorithms is the well known brightness constancy assumption which defines the constancy of the image intensities in the past and the current image warped according to the optical flow:

$$I(x,y,t) = I(x+u\delta t, y+v\delta t, t+\delta t), \tag{1}$$

where $\mathbf{d} = (u,v)^T$ denotes the unknown motion vector of the vector field.

Lucas and Kanade formulated the brightness constancy assumption in combination to the assumption that all pixels in a region Ω of the size $w_x \times w_y$, where both are odd, are subject to a constant movement. To compute the displacement $\mathbf{d}$ the energy functional based on the the linearisation and first order Taylor-approximation of Eq. 1 has to be minimised:

$$E_{local} = \sum_{x\in\Omega}\sum_{y\in\Omega}(u \cdot I_x + v \cdot I_y + I_t)^2 \tag{2}$$

with the spatial derivatives I_x, I_y and the temporal derivative I_t. The summation of each row and column are separated for a better readability. We define integral projections where

$P_x[I]$ denotes the horizontal and $P_y[I]$ denotes the vertical projection of any data I, with $P_x[I], P_y[I] : \Re^2 \mapsto \Re^1$ in a domain Ω, we obtain:

$$P_x[I] = \sum_{x \in \Omega} I(x,y) \tag{3}$$

$$P_y[I] = \sum_{y \in \Omega} I(x,y) \tag{4}$$

in relation to Eq. 2. With the purpose of not losing information, we derive a similar equation by changing the order of the projection terms of $\sum_{x\in\Omega}$ and $\sum_{y\in\Omega}$. Finally, we obtain the following equivalent system of equations which has to be minimized:

$$E_{local} = \sum_{y \in \Omega} P_x \left[(u \cdot I_x + v \cdot I_y + I_t)^2 \right] \tag{5}$$

$$E_{local} = \sum_{x \in \Omega} P_y \left[(u \cdot I_x + v \cdot I_y + I_t)^2 \right] \tag{6}$$

Let us now assume the use of gray value images of the range of $[0, 255]$ and spatial derivatives to be approximated as a discrete difference operator, $I_x = I(x-1, y) - I(x, y)$ and$I_x = I(x, y-1) - I(x, y)$. These assumptions allow for the following conclusions regarding integral projections of image derivatives within the domain Ω:

$$\begin{aligned} \underset{y_0 \cdot\cdot y_{w_y}}{P_x[I_x]} &= \textstyle\sum_{i\in\Omega} (I(i-1,y) - I(i,y)) \\ &= I(x_0,y) - I(x_1,y) + I(x_1,y) - I(x_2,y) + ... \\ &+ I(x_{w_x - 1}, y) - I(x_{w_x}, y) \\ &= I(x_{w_x}, y) \ - I(x_0, y) \in [0; 255] \end{aligned} \tag{7}$$

whereas in the other dimension

$$\underset{y_0 \cdot\cdot y_{w_x}}{P_y[I_x]} = \sum_{i \in \Omega} (I(x, i-1) - I(x, i)) \tag{8}$$

cannot be simplified and its co-domain is $\in [-255 \cdot N; 255 \cdot N]$. In an analogical manner, these considerations hold by changing the directions of the projection and the gradient calculation respectively.

We thus conclude that using a large region Ω and images including sufficient texture in x- and y-direction, which is an important assumption by Lucas and Kanade, the contributions of the projection term $P_x[I_y]$ and $P_y[I_x]$ are in general more significant than $P_x[I_x]$ and $P_y[I_y]$. In theory, calculating the sum of an image gradient is equivalent to applying a low-pass filter followed by a high-pass filter, which will cut off both the very low and the very high frequencies and thus reduces the energy of the signal. As an approximation, we can thus ignore these parts of Eq. 5,6 and set $P_x[I_x] = 0$, $P_y[I_y] = 0$.

Along with this assumption it is now possible to minimise the equation for each motion component separately.

$$E_{local} = \sum_{y \in \Omega} P_x \left[(v \cdot I_y - I_t)^2 \right] \tag{9}$$

$$E_{local} = \sum_{x \in \Omega} P_y \left[(u \cdot I_x - I_t)^2 \right] \tag{10}$$

As shown each component depends from the corresponding projected spatial and temporal derivatives. To get an accurate solution, a pyramidal iterative implementation solving Eq. 9,10 in a Newton-Raphson fashion as proposed in [10] is applied. In practice it has been shown that using a learning rate $\tau \in [0, 1]$ enhances the robustness of the solution by avoiding overshooting and thus potential divergence. The iterative scheme is given by:

$$u_{k+1} = u_k + \tau \cdot \frac{\sum_{x\in\Omega} P_y\left[I_x \cdot I_{t,k}\right]}{\sum_{x\in\Omega} P_y\left[I_x \cdot I_x\right]} \tag{11}$$

$$v_{k+1} = v_k + \tau \cdot \frac{\sum_{y\in\Omega} P_x\left[I_y \cdot I_{t,k}\right]}{\sum_{y\in\Omega} P_x\left[I_y \cdot I_y\right]} \tag{12}$$

with $I_{t,k} = I(x, y, t) - I(x + u_k, y + v_k, t + 1)$ and k as the number of iterations. The iteration is done until reaching convergence or a maximum number of iterations.

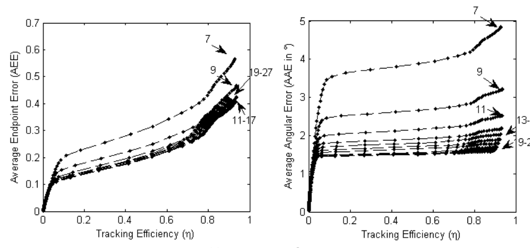
Observe that the projected spatial derivatives $P_y[I_x]$ and $P_x[I_y]$ are computed only once at the beginning of the iteration scheme. This is a higher effort than using integral images (see [8]) but avoids the second Taylor approximation. The solution shows that by the proposed approximation it is theoretically possible to reduce the computational effort for n feature points, k iterations and a quadratic region Ω of the size $w \times w$ from quadratic $\mathcal{O}(k \cdot n \cdot w^2)$ to linear $\mathcal{O}(k \cdot n \cdot w)$.

Practical aspects of this approach, e.g. region sizes and other parameters will be discussed in the following sections.

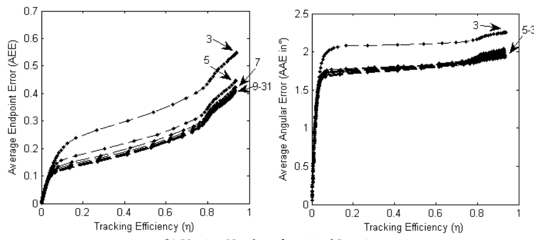
3. EVALUATION

In this section we present the results of the implementations of our proposed method. In a first experiment we test our CPU-implementation with various parameter settings concerning accuracy and execution time in comparison to the state-of-the-art pyramidal Lucas-Kanade implementation in Intel's OpenCV library[1]. Evaluation was done on the Middlebury benchmark sequences [11] without using color information. All tests were conducted on a PC with an 3.00 GHz Intel Core2Duo CPU using C/C++ implementation. The evaluation is based on two criteria: the common performance measures for dense flow fields as the average angular error (AAE) and the average endpoint error (AEE)[11] and

[1]available at http://sourceforge.net/projects/opencvlibrary



(a) Varying Window Size



(b) Varying Number of maximal Iterations

Fig. 1. Comparing tracking performances plot (*Grove2*) by (a) varing region sizes with 12 maximal iterations and (b) varying number of maximal iterations with fixed window size of 15. The convergence of the algorithms is reached in very few iterations ($\tau = 0.6$).

the tracking efficiency η, which was introduced by Singh *et al.* [12] and also used in [7]. This measure is defined as the quotient of the number of successfully tracked features and the total number of features initialized. Both criteria are combined into a plot called tracking performance, see Fig. 1 which is a measurement to describe the ability to track features including their tracking errors (AAE, AEE). Combining both criteria is important as the Lucas Kanade feature tracker has the possibility to detect lost features, and also outliers are rejected differently by each method.

Varying tracking efficiency is produced by comparing the calculated motion vectors from image $I(t) \rightarrow I(t+1)$ and reverse $I(t+1) \rightarrow I(t)$. The vector will be rejected if the residual motion $\left\|\mathbf{d}_{I(t)\rightarrow I(t+1)} - \mathbf{d}_{I(t+1)\rightarrow I(t)}\right\|$ is bigger than a threshold.

To determine a parameter set, the numbers of maximal iterations, region sizes and learning rate are varied and compared in Fig. 1 and Fig. 2. The result shows that the convergence can be achieved by a small number of iterations. In further experiments a maximal number of 12 iterations and a region size of 15 with a learning rate $\tau = 0.6$ will be used.

As illustrated in Fig. 2, with a reduced theoretical complexity of $\mathcal{O}(k \cdot n \cdot w)$ compared to the $\mathcal{O}(k \cdot n \cdot w^2)$ by PLK, the AEE of the proposed method is slightly higher than in the original Lucas-Kanade algorithm which is due to the approximation error of using integral projections but obviously has a higher accuracy than the II-LK [8], which computational complexity is constant $\mathcal{O}(k \cdot n)$ due to the usage of integral images.

In practice the theoretical computational effort could be varying dramatically i.e. the convergence is reached before the maximal numbers of iterations or not. From Fig. 2 left the convergence behaviour can be derived. But to display the overall computational effort the PLK, and modified PLK are compared in Fig. 3 bottom, the results, obtained for the execution time by different region sizes, emphasize that the computational complexity is practically reduced from the quadratic complexity to the linear complexity. Finally, we compared the execution time of our GPU implementation using OpenCL with the pyramidal Lucas-Kanade-implementation of Marzat [6][2], which was implemented using the CUDA interface to show the ability for parallelisation of our method. We performed this evaluation on a NVIDIA GTX275 graphic device. Figure 3 right illustrates the execution time of the two implementations. As already mentioned during the evaluation of the CPU-implementation, these results show the reduction of the computational complexity.

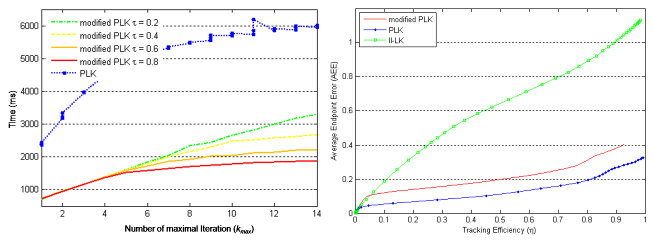


Fig. 2. Comparing execution time and AEE by the CPU implementation of the pyramidal Lucas-Kanade (PLK) and the integral projected modified PLK (window size 15). The convergence of the algorithms is reached in very few iterations (left). The AEE (right) shows that the modified PLK is less accurate than the PLK but more accurate than the II-LK.

This yields a huge speed-up of up to 248ms per frame compared to 672ms of the Cuda implementation. The top diagram of Fig. 3 shows the execution time of the modified PLK for a smaller number of feature points. The scalability of our implementation makes it possible to decrease significantly the execution time if the number of features to track is reduced. That can be useful e.g. in surveillance scenarios, Structure from Motion or Augmented Reality. By computing the motion vectors for every sixth pixel of the given video sequence (640x480 pixels), the execution time of our algorithm decreases from 248ms (applying to 276624 feature points) to 14 ms (applying to 17289 feature points). Due to the constant computational complexity of the not optimised preprocessing kernels, the execution time can not fall below 4ms per frame.

4. CONCLUSION

In this paper we present an approach to speed up the motion estimation of sparse and dense sets of features. The classical Lucas-Kanade algorithm is modified by minimizing the

[2]available at http://www.nvidia.com/object/cuda_home.html#state=home

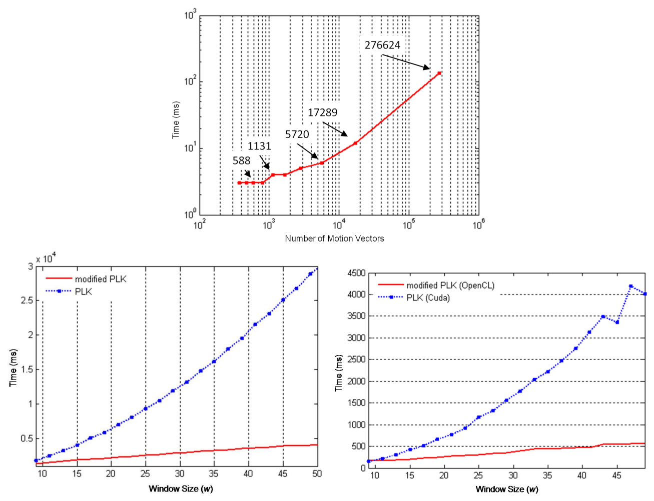


Fig. 3. Comparing the execution time of CPU (bottom left) and the GPU (bottom right) implementation of the pyramidal Lucas-Kanade (PLK) and the integral projected modified PLK. Varying the number of motion vectors (emphasised by arrows) for the modified PLK decreases significantly the execution time (top).

brightness constancy assumption energy functional by means of integral projections. With an OpenCL-based implementation, we achieved a performance of 248 ms per frame on a NVIDIA GTX 275 GPU for a 640×480 dense flow field. Sparse flow fields of about 17,000 points can be computed in an execution time of up to 14 ms. This shows that the trade-off between speed-up and accuracy is a good compromise, especially in real-time scenarios. With respect to the runtime, our method is much faster than state-of-the-art GPU implementations as [6]. Therefore we propose this algorithm for surveillance applications, which need motion vectors or feature trajectories for e.g. crowd description or crowd behaviour analysis and can be speeded up this way or such as Augmented Reality and robot navigation which are restricted by limited hardware. Furthermore, we propose this algorithm as an adequate solution for algorithms which process high-definition video content.

5. REFERENCES

[1] Bruce D. Lucas and Takeo Kanade, "An iterative image registration technique with an application to stereo vision," 1981, pp. 674–679.

[2] Berthold K. P.Horn and Brian. G. Schunck, "Determining optical flow," *Artifical Intelligence*, vol. 17, pp. 185–203, 1981.

[3] C. Tomasi and T.Kanade, "Detection and tracking of point features," Technical report CMU-CS-91-132, CMU, 1991.

[4] Marc Pollefeys Sudipta N. Sinha, Jan-Michael Frahm and Yakup Genc, "Gpu-based video feature tracking and matching," Technical report 06-012, UNC Chapel Hill, 2006.

[5] C. Zach, D. Gallup, and J.M. Frahm, "Fast gain-adaptive klt tracking on the gpu," in *Visual Computer Vision on GPUs workshop (CVGPU 08)*, 2008, pp. 1–7.

[6] J. Marzat, Y. Dumortier, and A. Ducrot, "Real-time dense and accurate parallel optical flow using cuda," in *Proceedings of the 17th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG)*, 2009, pp. 105–111.

[7] Tobias Senst, Volker Eiselein, Rubén Heras Evangelio, and Thomas Sikora, "Robust modified L2 local optical flow estimation and feature tracking," in *IEEE Workshop on Motion and Video Computing (WMVC 11)*, 2011, pp. 685–690.

[8] Tobias Senst, Volker Eiselein, and Thomas Sikora, "II-LK a real-time implementation for sparse optical flow," in *International Conference on Image Analysis and Recognition (ICIAR 10)*, 2010, pp. 240–249.

[9] A. J. Crawford, H. Denman, F. Kelly, F. Piti, and A. C. Kokaram, "Gradient based dominant motion estimation with integral projections for real time video stabilisation," in *IEEE International Conference on Image Processing*, 2004, pp. 3371–3374.

[10] Jean-Yves Bouguet, "Pyramidal implementation of the lucas kanade feature tracker," Technical report, Intel Corporation Microprocessor Research Lab, 2000.

[11] J.P. Lewis Stefan Roth Michael J. Black Simon Baker, Daniel Scharstein and Richard Szeliski, "A database and evaluation methodology for optical flow," Technical report MSR-TR-2009-179, Microsoft Research, 2009.

[12] Meghna Singh, Mrinal K. Mandal, and Anup Basu, "Gaussian and laplacian of gaussian weighting functions for robust feature based tracking," *Pattern Recognition Letters*, vol. 26, no. 13, pp. 1995–2005, 2005.