Available online at www.sciencedirect.com

SCIENCE DIRECT®

ELSEVIER

Image and Vision Computing 25 (2007) 141–147

image AND vision COMPUTING

www.elsevier.com/locate/imavis

A genetic algorithm for optical flow estimation

Marco Tagliasacchi *

Politecnico di Milano, Dipartimento di Elettronica e Informazione, Piazza Leonardo da Vinci, 32, 20133 Milano, MI, Italy

Received 20 April 2004; received in revised form 4 June 2005; accepted 31 January 2006

Abstract

This paper illustrates a new optical flow estimation technique that builds upon a genetic algorithm (GA). First, the current frame is segmented into generic shape regions, using only luminance and color information. For each region, a two-parameter motion model is estimated using a GA. The fittest individuals identified at the end of this step are used to initialize the population of the second step of the algorithm, which estimates a six-parameter affine motion model, again using a GA. The proposed method is compared with a multi-resolution version of the well-known Lucas–Kanade differential algorithm. Our simulations demonstrate that, with respect to Lucas–Kanade, it significantly reduces the energy of the motion-compensated residual error.
© 2006 Elsevier B.V. All rights reserved.

Keywords: Optical flow; Genetic algorithms; Motion estimation

1. Introduction

We refer to the optical flow as the velocity field which warps one image into another [1]. Optical flow estimation [2] tries to assign to each pixel of the current frame a two-component velocity vector indicating the position of the same pixel in the reference frame. The knowledge of the optical flow is valuable information in several applications, ranging from motion detection, object segmentation, motion compensated encoding, and stereo disparity measurement just to name a few of them. Optical flow should not be confused with the projection of the motion of 3D objects on the 2D image plane. Although, there is a close relationship between true motion and the estimated velocity field, there are other factors affecting the optical flow estimate, such as for example noise (camera noise, quantization noise), reflections and illumination changes.

There are several optical flow estimation algorithms known in the literature. Ref. [2] is a complete survey describing the basic ideas behind the most important algorithms, whereas the authors of [3] compare quantitatively the performance of various techniques. According to the taxonomy proposed in [2], we can cluster the optical flow algorithms in the following categories: differential (Lucas–Kanade [4], Horn-Schunk [5], Nagel [6]), region-based matching (Anandan [7]), energy-based (Heeger [8]) and phase correlation algorithms (Fleet–Jepson [9]). More recent algorithms [10] exploit locally adaptive parametric motion models to drive the optical flow estimation.

In this paper, we chose the Lucas–Kanade algorithm as a benchmark. Although it is not state-of-the-art, it is one of the best performers according to [3] and owing to its simplicity it is widely adopted in the literature. Moreover it provides a full coverage of the video frame, at least when implemented in a multi-resolution fashion, as explained in greater detail in Section 3. Although our algorithm does not match exactly any of the aforementioned categories, it shares similarities with region-based matching methods. In fact both of them try to find the velocity vectors that maximize the correlation between a pair of consecutive frames. The main difference is that our approach replaces a full search with a genetic algorithm driven search and square blocks with generic shape regions. Moreover, an affine motion model is used as in [10] instead of a simple translation model in order to capture more complex motion patterns like zooming and rotation.

In the literature, there are a few works describing applications of genetic algorithms to the problem of motion estimation. In [11], a block matching fast motion estimation technique is proposed, where a genetic algorithm is used to avoid getting trapped in local minima. Ref. [12] describes a multi-resolution optical flow estimation algorithm that uses the same objective function as region-based matching algorithms [7]. The authors claim that the algorithm is able to provide a consistent velocity field also in areas that do not contain

* Tel.: +39 031 332 7341; fax: +39 031 332 7321.
E-mail address: marco.tagliasacchi@polimi.it.

0262-8856/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.imavis.2006.01.021

enough texturing. The algorithm described in this paper shares the general approach of [13,14]. In fact, an affine model is used to describe the motion of the objects composing the scene, and a genetic algorithm is used to search for the optimal model parameters. Nevertheless, in both cases, experimental results are given only for synthetic sequences.

The remainder of this paper is organized as follows: Section 2 defines the metrics that can be used to assess the quality of the optical flow estimate. Section 3 summarizes the Lucas–Kanade differential algorithm. Section 4 explains the details of the proposed genetic algorithm. Section 5 contains experimental results of our simulations on both synthetic and real sequences.

2. Metrics to assess estimation

In order to assess the quality of the optical flow estimate we can use either one of the following two metrics:

- average angular deviation [3];
- energy of the displaced frame difference.

The former can be applied only when the testing conditions involve synthetic sequences and the 'ground truth' optical flow is know in advance. The angular deviation is not computed as the simple Euclidean distance between two velocity vectors $\mathbf{v}^a = (v_x^a, v_y^a)$ and $\mathbf{v}^b = (v_x^b, v_y^b)$. Both are first converted to three component vectors having unitary norm, applying the following transformation:

$$\mathbf{v}_e = \left(\frac{v_x}{\|\mathbf{v}\|}, \frac{v_y}{\|\mathbf{v}\|}, \frac{1}{\|\mathbf{v}\|}\right) \tag{1}$$

Then, the angular deviation between the two transformed vectors is computed as:

$$\Psi_e = \arccos\langle \mathbf{v}_e^a, \mathbf{v}_e^b \rangle \tag{2}$$

The displaced frame difference (DFD) is computed as the difference between the pixel intensities of the current and the reference frame, following the motion trajectories identified by the optical flow. Stated formally:

$$\mathrm{DFD}(x_i,y_i) = I(x_i,y_i,t) - I(x_i - v_x(x_i,y_i), y - v_y(x_i,y_i), t-1) \tag{3}$$

In order to assess the estimate, we can either compute the energy of the DFD (MSE, mean square error) or the MAD (mean absolute differences). The lower is the MSE or MAD, the better the estimate. It is worth pointing out that, despite the average angular deviation, the DFD can be applied as a metric even if the real optical flow is not known, as it is the case for natural imagery. Furthermore, it is more suitable as far as video coding applications are concerned, since our ultimate goal is to reduce the energy of the prediction residuals.

3. Lucas–Kanade differential algorithm

Lucas–Kanade estimation algorithm is one of simplest, yet powerful methods to compute optical flow. For this reason, it is one of the most widely used. It builds upon the assumption that the image intensity remains unchanged along motion trajectories (brightness constraint):

$$\frac{\mathrm{d}I(x,y,t)}{\mathrm{d}t} = 0 \tag{4}$$

Expanding the total differential into partial derivatives we get the well known optical flow equation

$$I_x(x,y)v_x + I_y(x,y)v_y + I_t(x,y) = 0 \tag{5}$$

where I_x, I_y and I_t, respectively, the horizontal, vertical and temporal gradients. In order to enforce such a constraint, the sequence is pre-filtered along time and space with a Gaussian kernel. Eq. (5) represents a line in the velocity space spanned by v_x and v_y. To find a unique solution we impose that the equation needs to be satisfied for all pixels falling in a window centered on the current pixel, yielding the following over-determined system, whose solution is computed using least squares (8):

$$\begin{bmatrix} I_{x_1} & I_{y_1} \\ I_{x_2} & I_{y_2} \\ \cdots & \cdots \\ I_{x_M} & I_{x_M} \end{bmatrix} \begin{bmatrix} v_x \\ v_y \end{bmatrix} = \begin{bmatrix} -I_{t_1} \\ -I_{t_2} \\ \cdots \\ -I_{t_M} \end{bmatrix} \tag{6}$$

$$A\mathbf{v} = \mathbf{b} \tag{7}$$

$$\mathbf{v} = (A^T A)^{-1} A^T \mathbf{b} \tag{8}$$

Lucas–Kanade algorithm suffers from the so-called aperture problem, thus it is unable to produce an accurate result when there is not enough texture within the observation window. In this situation, it is able to estimate only the component that is parallel to the local gradient. The minimum eigenvalue of the matrix $A^T A$ is usually employed as a good indicator [15]. Only when it is greater than a given threshold, a full velocity estimate can be accurately computed.

Another drawback of this solution is that it fails to estimate large displacements because the brightness profile cannot be assumed to be linear far from the observation point. In order to overcome this limitation we can apply a multi-resolution approach. A low-pass pyramid is built and the Lucas–Kanade algorithm runs on the lowest resolution copy. The estimated optical flow is then interpolated and refined at the next higher resolution level.

4. GA-based optical flow estimation algorithm

The proposed algorithm starts by computing a complete segmentation of the current frame, grouping together those pixels sharing the same spatial location and having similar brightness. We accomplish this task performing a watershed algorithm on the morphological gradient of the current frame, as explained in [16]. Nevertheless, the segmentation method does not significantly affect the optical flow estimation, thus it will not be further described in this paper. Fig. 1 shows one frame of the synthetic *Yosemite* sequence, together with the corresponding segmentation map. After the segmentation phase, the algorithm considers each region independently.

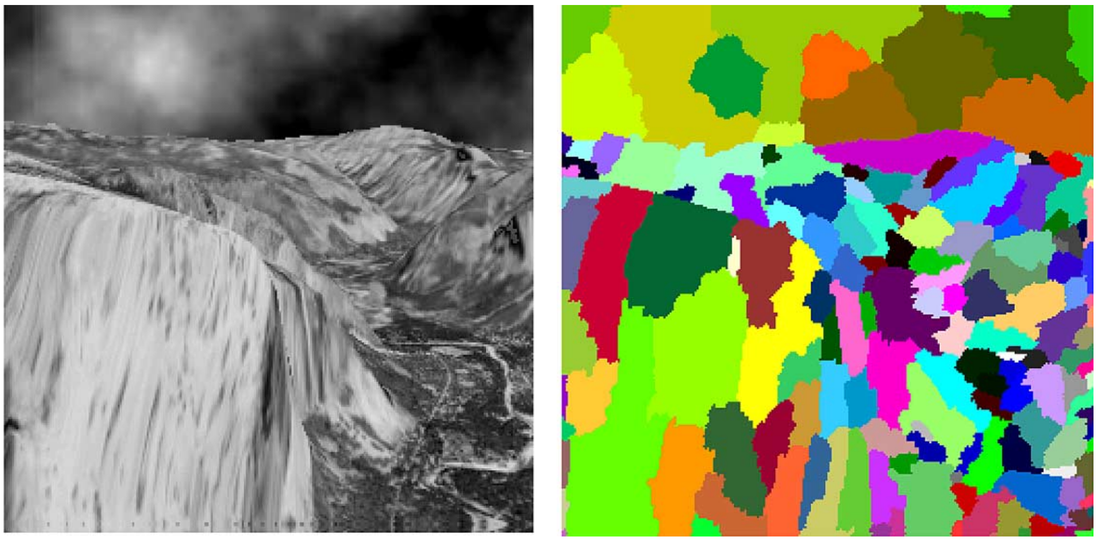


Fig. 1. Left: one frame from the *Yosemite* sequence. Right: segmentation map obtained with the watershed algorithm described in [16].

4.1. Affine motion model

We assume that the velocity field is smooth apart from abrupt changes along object boundaries. Therefore, pixels belonging to the same region are characterized by a coherent motion that can be described with a limited set of parameters. In this paper, we adopt a six-parameter affine motion model. An affine model is able to capture the motion of a rigid planar surface, which moves on the 3D space, projected on the 2D image plane. Although real objects are not planar, this is still a good approximation, since it allows describing complex motion such as zooming, rotation and shear. Once the six motion parameters are known, the full velocity vectors at any point (x,y) of the current region can be computed as

$$v_x(x_i,y_i) = a_1 + a_3\frac{x_i}{C_x} + a_5\frac{y_i}{C_y} \qquad (9)$$

$$v_y(x_i,y_i) = a_2 + a_4\frac{x_i}{C_x} + a_6\frac{y_i}{C_y}$$

where $\mathbf{a} = (a_1, a_2, a_3, a_4, a_5, a_6)$ is the motion model parameters vector, C_x and C_y the region centroid coordinates. Having fixed the motion model, it is matter of finding the vector a which minimize the MSE

$$\mathbf{a} = \arg\min_{\mathbf{a}} \frac{1}{M} \times \sum_{i=1}^{M} \left|I(x_i,y_i,t) - I(x_i - v_x(x_i,y_i), y - v_y(x_i,y_i), t-1\right|^2 \qquad (10)$$

where M is the number of pixels in the current region.

Eq. (10) represents an unconstrained, non-linear optimization problem in a six-variable space, characterized by the following features:

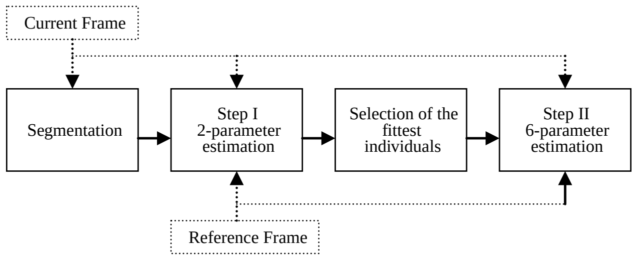
- the function to be minimized cannot not be expressed in an closed form in terms of the parameters vector **a**;
- the search space is large;
- there are several local optima;
- a good solution, even if it is not the global optimum, might be satisfactory.

Although other multivariate optimization algorithms (i.e. simulated annealing) might be applied, here we investigate the use of a genetic algorithm to address the optimization problem in (10), since they are well suited to this class of problems.

In order to speed up the convergence, the algorithm is divided into two steps. Step I computes the estimate of a simpler two-parameter model, which is able to describe only rigid translations. The result is used to initialize Step II, which refines the solution estimating the whole six-parameter affine model. Only the fittest half of the population is selected at the end of Step I, and it is mixed with a randomly generated population. Fig. 2 shows the block diagram of the overall algorithm. In the following, we summarize the details of the various phases of the algorithm.

4.2. Chromosome encoding

Each six-parameter vector **a** represents an individual of the population explored by the genetic algorithm. The individuals are encoded as a 48 bits binary string chromosome, where each parameter a_i is represented with a precision of 8 bits. This allows to span the interval $[-16, +15]$ with 1/8 pixel accuracy.


Current Frame
Segmentation
Step I 2-parameter estimation
Selection of the fittest individuals
Step II 6-parameter estimation
Reference Frame
Fig. 2. Block diagram summarizing the proposed optical flow estimation algorithm.

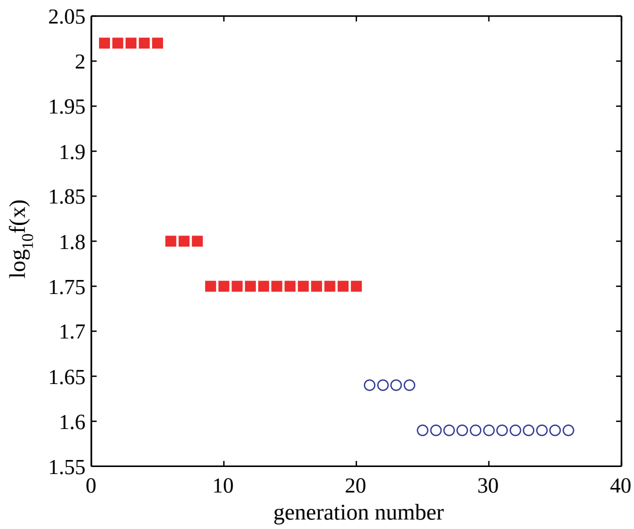


Fig. 3. Evolution of the value of objective function of the fittest individual over generation time. Squares, step I; circles, step II.

We decided to opt for a quantized representation of the parameters **a** in order to speed up the convergence of the algorithm, with respect to using real-valued parameters. Furthermore, the quality of the motion-compensation tends to saturate when the accuracy of the motion field goes beyond 1/8.

4.3. Objective function

The initial population is selected at random, dragging samples from a Gaussian distribution. For each individual, the objective function indicates the quality of the motion-compensation given the set of parameters **a** used, expressed in terms of the energy of the displaced frame difference:

$$f(x) = \frac{1}{M} \times \sum_{i=1}^{M} \left| I(x_i,y_i,t) - I(x_i - v_x(x_i,y_i), y_i - v_y(x_i,y_i), t-1) \right|^2 \tag{11}$$

$$f(x) = \frac{1}{M}\sum_{i=1}^{M}\left| I(x_i,y_i,t) - I\left(x_i - \left(a_1 + a_3\frac{x_i}{C_x} + a_5\frac{y_i}{C_y}\right), y_i - \left(a_2 + a_4\frac{x_i}{C_x} + a_6\frac{y_i}{C_y}\right), t-1\right)\right| \tag{12}$$

We make the assumption that the magnitude of v_x and v_y cannot be larger than 20 pixels. If this event is detected for some **a**, the computation of the objective function is stopped prematurely, in order to speed up the evaluation.

4.4. Fitness evaluation

The fitness of an individual is calculated from the objective function using linear ranking, with selective pressure equal to 2

$$F(p_i) = 2\frac{p_i - 1}{N_{\text{ind}} - 1} \tag{13}$$

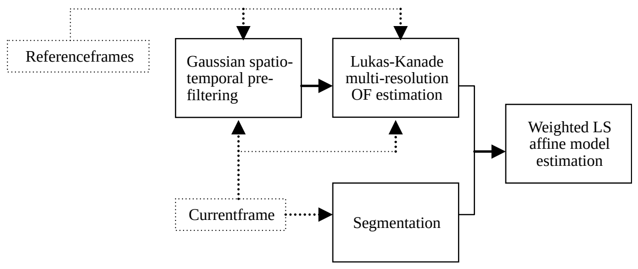
where p_i is the position in the ordered population of individual i, using the value of the objective function as a ranking criterion. N_{ind} is the number of individuals in the current population, which has been set to 20 in our experiments. The choice of the population size plays an important role in tuning the performance of the algorithm. While a larger population increases the chances of finding the optimum in fewer iterations, it also affects the computational complexity of each iteration. The value $N_{\text{ind}} = 20$ has been selected empirically as it strikes a good balance between these two competing requirements.

4.5. Selection, crossover and mutation

The selection phase extracts the individuals that have to be reproduced using stochastic universal sampling [17], which guarantees zero bias and minimum spread. A generation gap of 0.9 is chosen in order to maintain the fittest individuals. Multipoint crossover is performed on pairs of chromosomes with probability 0.7. The crossover points have fixed locations, corresponding to the boundaries between the variables, in order to preserve the building blocks. Mutation probability is set to $0.7/L_{\text{ind}}$, where L_{ind} is the length of the chromosome ($L_{\text{ind}} = 48$). This value is selected as it implies that the probability of any one element of a chromosome being mutated is approximately 0.5 [18].

4.6. Stopping criterion

Each step of the two steps of the algorithm (see Fig. 2) is terminated when the objective function computed for the fittest individual has not changed during the last 10 iterations. Experiments demonstrate that with this configuration, the



Referenceframes
Gaussian spatio-temporal pre-filtering
Lukas-Kanade multi-resolution OF estimation
Weighted LS affine model estimation
Currentframe
Segmentation

Fig. 4. Block diagram illustrating the Lucas–Kanade LK2 version. It computes the six-parameter affine model for each segmented region which best matches the optical flow provided by LK1.

Table 1
Yosemite sequence

	Avg. ang. dev.	PSNR (dB)	PSNR gain
GA	12.13	31.11	–
LK1	14.06	31.12	−0.01 dB
LK2	9.74	30.79	+0.33 dB

Averages over the first 16 frames.

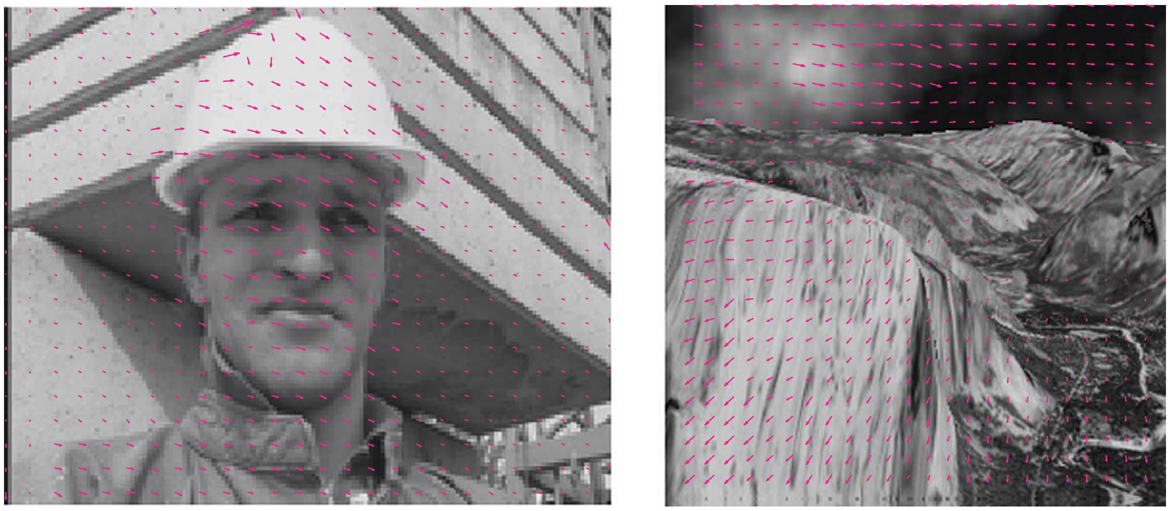


Fig. 5. Estimated optical flow: Left, *Yosemite* sequence, frame 10; right, *Foreman* sequence, frame 30.

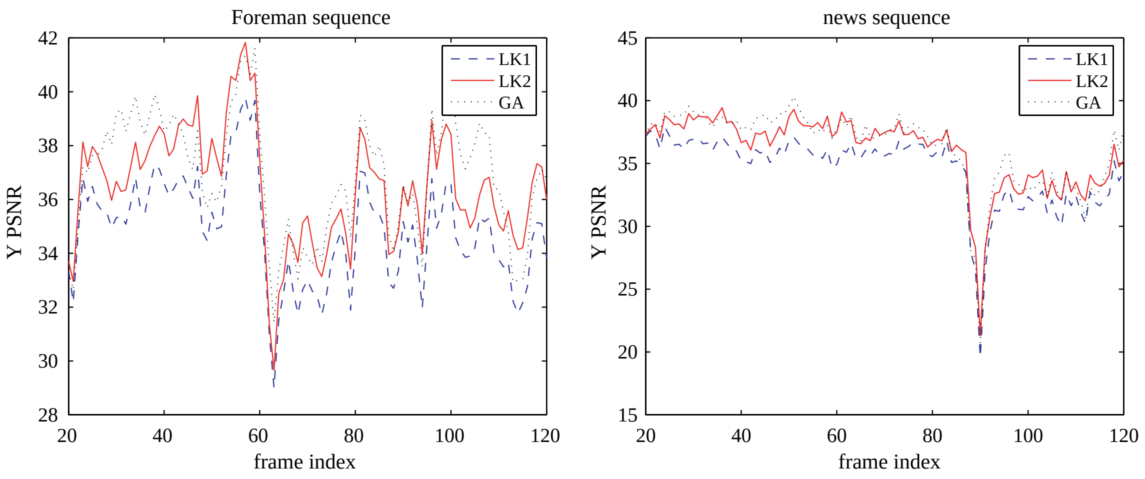


Fig. 6. Left: PSNR vs frame index over 100 frames of the *Foreman* and *News* sequences. Right: average values for the same sequences.

algorithm tends to converge in less than 50 generations, of which 20 are used to validate the solution. Fig. 3 shows an example that illustrates the convergence of the algorithm.

4.7. Complexity issues

Most of the complexity load is owed to the computation of the objective function, which grows linearly with the area of the regions. However, once the segmentation of the current frame is performed, the algorithm works independently on each region. This observation suggests that a parallel implementation would be better suited, where the number of simultaneous tasks matches the number of regions. Therefore, the time complexity turns out to be of the order $O(P \cdot H \cdot W/K)$, where P is the average number of iteration for the genetic algorithm to converge, $H$ and W, respectively, the frame height and width, while K is the number of regions. Our algorithm does not require demanding memory requirements, since only two frame buffers are used at a time to store the current and the reference frame.

5. Experimental results

In this section, we compare the performance of the proposed genetic algorithm (GA) with the multi-resolution version of Lucas–Kanade (see Section 3), dubbed LK1 in the following. Moreover, in order to make a fair comparison, we fitted a locally affine motion model to the Lucas–Kanade optical flow, estimating by least squares the affine motion model that best matches the velocity field in each segmented region. In order to improve the estimate of the affine model, each pixel is assigned a weight reflecting its reliability. Pixels located close to the border and those having a non-textured neighbourhood receive a lower weight. Segmentation and optical flow estimation are performed independently and they are merged only at last. We will refer to this algorithm as LK2. Refer to Fig. 4 for a overall diagram illustrating this version of the algorithm.

We performed our tests on the synthetically generated *Yosemite* sequence (see Fig. 6). Since the real optical flow is

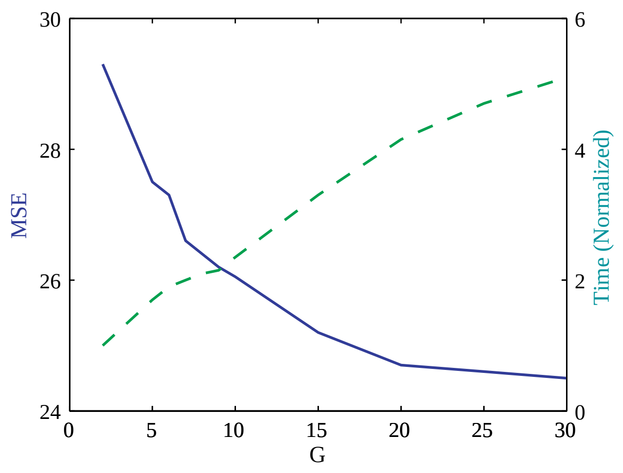


Fig. 7. Effect of the stopping criterion G on the MSE (solid) and time complexity (dashed).

available, we are able to compute both the PSNR[1] of the DFD and the angular deviation, which are listed in Table 1. The tabulated value of the angular deviation for LK1 differs from the one reported in literature ([3]) for two reasons: (a) the latter refers to a coverage of only 40% obtained with a single resolution approach; (b) a 15 pixel border, i.e. where most of the error is localized, is not considered in [3]. GA and LK1 outperform LK2 in terms of PSNR. With respect to the average angular deviation, GA performs better than LK1. A closer inspection demonstrates that this is especially true along image borders, due to the inaccuracy of computing the discrete approximation of the partial derivatives needed by Lucas–Kanade. LK2 perform significantly better using this metric. The reason is that the regularization performed by the affine model filters out outliers, improving the estimate at a cost of higher MSE.

The experiments conducted on the real sequences *Foreman* and *News* (see Fig. 5) confirm that the proposed algorithm outperforms on average both LK1 and LK2 in terms of PSNR.

We carried out another experiment aimed at determining the optimal stopping criterion (Fig. 6). We stated that both Step I and Step II stop when the best individual objective function has not changed during the last G generations. Fig. 7 shows the relation existing between G and the quality of the estimate for the *Foreman* sequence. By increasing G the average number of generations for each region grows, together with the time needed to complete the algorithm. For this reason, we found that setting G equal to 10 is a good trade-off between quality and complexity.

The proposed algorithm has been implemented in Matlab using the Genetic Algorithm toolbox described in [19]. As a rough indication, it takes approximately 2 min to compute the optical flow estimate for QCIF[2] sequences on a Pentium M 1.6 Ghz computer. Nevertheless, the source code is not optimized and further improvements are possible.

[1] PSNR $= 10 \log_{10} 255^2/\text{MSE}$.

[2] Spatial resolution: 176×144 pixels.

6. Conclusions

In this paper, we introduced a new optical flow estimation method that takes advantage of a two-step genetic algorithm. Experiments proved that it yields results comparable to the Lucas–Kanade algorithm. We are currently working on a multi-resolution version of the algorithm aiming at speeding up the computation. The presented method is used as the core component of a motion segmentation algorithm whose goal is to isolate distinct moving objects composing the scene.

References

[1] A. Verri, T. Poggio, Motion field and optical flow: qualitative properties, IEEE Transactions on Pattern Analysis and Machine Intelligence 11 (5) (1989) 490–498.

[2] S.S. Beauchemin, J.L. Barron, The computation of optical flow, ACM Computing Surveys 27 (3) (1995) 433–466.

[3] D.F. J.L. Barron, S. Beauchemin, Performance of optical flow techniques, International Journal of Computer Vision 12 (1) (1994) 43–77.

[4] B.D. Lucas, T. Kanade, An iterative image registration technique with an application to stereo vision, in: Proceedings of the International Joint Conference on Artificial Intelligence, Vancouver, BC, Canada, August 1981, pp. 674–679.

[5] B. Horn, B. Schunk, Determining optical flow, Artificial Intelligence 17 (1985) 185–203.

[6] H. Nagel, On the estimation of optical flow: Relations between different approaches and some new results, Artificial Intelligence 33 (1987) 299–324.

[7] P. Anandan, A computational framework and an algorithm for the measurement of visual motion, International Journal of Computer Vision 2 (1989) 283–310.

[8] D. Heeger, Optical flow using spatiotemporal filters, International Journal of Computer Vision 1 (1988) 279–302.

[9] A.J.D.J. Fleet, Computation of component image velocity from local phase information, International Journal of Computer Vision 5 (1990) 77–104.

[10] M.J. Black, A.D. Jepson, Estimating optical flow in segmented images using variable-order parametric models with local deformations, IEEE Transactions on Pattern Analysis and Machine Intelligence 18 (10) (1996) 972–986.

[11] S. Li, W.-P. Xu, H. Wang, N.-N. Zheng, A novel fast motion estimation method based on genetic algorithm, in: Proceedings of the International Conference on Image Processing, Kobe, Japan, October 1999.

[12] M. Gong, Y.-H. Yang, Multi-resolution genetic algorithm and its application in motion estimation, Pattern Recognition 1 (2002) 644–647.

[13] M. Zaim, A.E. Ouaazizi, R. Benslimane, Genetic algorithms based motion estimation, in: Vision Interface Annual Conference, Ottawa, Canada, June 2001.

[14] E.L. Dixon, C.P. Markhauser, K.R. Rao, A new object motion estimation technique for video image, based on a genetic algorithm, IEEE Transactions on Consumer Electronics 43 (3) (1997).

[15] E.H.A.E.P. Simoncelli, D.J. Heeger, Probability distributions of optical flow, in: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Maui, HI, 1991.

[16] D. Wang, A multiscale gradient algorithm for image segmentation using watersheds, Pattern Recognition 678 (12) (1997) 2043–2052.

[17] J.E. Baker, Reducing bias and inefficiency in the selection algorithm, in: Proceedings of the Second International Conference on Genetic Algorithms and their Application, Hillsdale, NJ, 1987, pp. 14–21.

[18] L. Booker, Improving Search in Genetic Algorithms, Genetic Algorithms and Simulated Annealing 1987 Morgan Kaufmann, 61–73.

[19] A. Chipperfield, P. Fleming, H. Pohlheim, C. Fonseca, Genetic Algorithm Toolbox (for use with Matlab). Department of Automatic Control and System Engineering, University of Sheffield, Available from: http://www.shef.ac.uk/acse/research/ecrg/gat.html.