VOTING-BASED MOTION ESTIMATION

Alexandru ILINU[1]
Cristian AVATAVULUI[2]
Giorgiana Violeta VLĂSCEANU[3]
Costin-Anton BOIANGIU[4]

Abstract: *Motion estimation is an essential topic in computer vision, having a large number of applications, such as tracking and video compression, to name a few. This paper presents a voting-based motion estimation algorithm that combines two categories of methods, namely dense methods (optical flow) and sparse methods (block-matching). The obtained results proved that the proposed approach is fast, robust and reliable, thus being suitable for integration in unsupervised video processing systems.*

Keywords: *motion estimation, optical flow, block matching, motion vectors, voting system, voting methods*

1. Introduction

The main idea of motion estimation is (given a sequence of images, usually a video) to find the motion, namely what moved in the sequence and how or how much it moved (the direction and magnitude) as we step through the images in the sequence [1]. Usually, we want to find the motion between every two consecutive frames at the same point in the image space. We may be interested in finding the motion of some regions of the image, the motion in some specific sampling points, or the motion of all pixels of the image. Thus, the output of motion estimation algorithms is a 2D vector field. The vectors are defined at some points in the image space, or at every pixel, which represents an estimation of the motion at that particular point or in a neighborhood of that point). The vector field is calculated at each frame based on the difference with the previous frame.

1.1. Previous work

In the past years, the domain of motion estimation had evolved considerably [1][2]. There are two major categories of methods for motion estimation mentioned in the literature: direct methods and indirect methods [3][4].

[1] Engineer, University Politehnica of Bucharest, 060042 Bucharest, Romania, alexandru.ilinu@stud.acs.upb.ro
[2] PhD Student., University Politehnica of Bucharest, 060042 Bucharest, Romania, cristianavatavului@gmail.com
[3] Teaching assistant, PhD Student, Eng., University Politehnica of Bucharest, 060042 Bucharest, Romania, giorgiana.vlasceanu@cs.pub.ro
[4] Professor, PhD Eng., University Politehnica of Bucharest, 060042 Bucharest, Romania, costin.boiangiu@cs.pub.ro

In the first category, there are the methods presented in the scientific world:

- **Dense/direct methods** – estimate motion in every pixel based on the temporal and spatial variation of the intensity at that point ([15], [18], [19])
- **Block-matching methods** – divide the image space into blocks of equal size and estimate the motion for each block based on the correlation of the intensities of the pixels in that block and the ones in the other frame. They can further be generalized to so-called "region-matching" methods that use regions of arbitrary shapes instead of rectangular blocks [20][7][8].
- **Phase correlation** – compute an estimate of the global motion between two frames based on their Fourier transform. In some research papers are mentioned as frequency-domain methods [16].
- **Optical Flow** – compute a motion vector between two consecutive frames [12].

The second category is the indirect methods that have at the base the features (corner detection):

- **Feature-based methods** – these methods find feature points and track them across multiple images; they estimate the motion of only the feature points based on the locations they are found in the images [5].

1.2. Problem motivation

Dense methods are based on a Taylor series approximation that only holds for small displacements of the same object point in the two frames. Thus, they are well suited for videos with a "small" amount of motion. On the other hand, block-matching methods can detect more significant motions depending on the size of the search region parameter; however, they are more computationally intensive.

We can obtain a sparse motion estimation method from a dense method by skipping pixels, or equivalently, by sampling the pixels regularly in the image space, at a given interval. Similarly, for a sparse method that computes the motion vector for a specific block, we can shift the block by any number of pixels and apply the same procedure to estimate the motion vector for any pixel in the image space (or decrease the block size up to a few pixels) – and obtain a dense method from a sparse one.

Thus, we can tune the "sparsity" of the methods and combine the advantages of the two methods into one voting-based algorithm.

2. Proposed Method

The method proposed here calculates the motion vector fields at the same "sparsity level" using the two methods described above, and then combines their results into a single output.

Firstly, the two methods will be described in detail, and then the final voting-based method will be presented.

For simplicity, the experiments consider only gray-scale image sequences.

2.1. Optical flow

This method [13][14] considers a sequence of images represented by a function $I = I(x, y, t)$, where x and y are the spatial coordinates (the locations of the pixels) and t represents time (or the frame number). The final goal is to estimate the motion vector $[u, v]$ for every sampling point (x, y) in the image space and between every two consecutive frames (t and $t + 1$).

It was considered a specific point (the blue point in Figure 1) at coordinates (x, y) in the frame t, moving along the motion vector $[u, v]$, so in the frame $t + 1$, the blue point will be at the location $(x + u, y + v)$. A reasonable assumption is that the point will appear in the same color (or the same intensity value) in the frame $t + 1$ at the new location, so this gives the brightness constancy constraint (equation (1)).

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

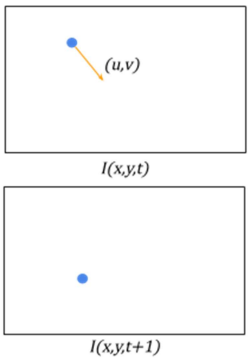

Figure 1 – Optical Flow Example

Now, if the motion vector $[u, v]$ is sufficiently small, the following Taylor series expansion holds:

$$I(x+u, y+v, t+1) \simeq I(x,y,t) + \frac{\partial I}{\partial x} \cdot u + \frac{\partial I}{\partial y} \cdot v + \frac{\partial I}{\partial t} \quad (2)$$

From (1) and (2) it can be obtained:

$$I_x \cdot u + I_y \cdot v + I_t = 0 \quad (3)$$

In (2) and (3), $\frac{\partial I}{\partial x}$, $\frac{\partial I}{\partial y}$, $\frac{\partial I}{\partial t}$ or, I_x, I_y, I_t represent, respectively, the estimates of the derivatives of the image function I (concerning $x, y,$ and t).

The motion vector at (x, y) can be computed by solving equation (3) for u and v. The equation has two unknowns (underconstrained linear system of equations). A unique solution, in this case, is not available, and there are infinitely many. To overcome this, there are some approaches:

- Horn & Schunck [19] – impose a smoothness constraint on the motion vector field across the image; together with (II.3) this becomes a *minimization problem*: we need to find the motion vector field $u(x, y), v(x, y)$ such that the following expression is minimized:

$$\iint \left(I_x u + I_y v + I_t\right)^2 + \lambda\left(u_x^2 + u_y^2 + v_x^2 + v_y^2\right) dx dy \quad (4)$$

  where u_x, u_y, v_x, v_y represent the derivatives of the motion components and λ is a tunable parameter.

- Lucas & Kanade [14] – they assume a *local* smoothness of the motion vector field in the neighborhood of point (x, y); more precisely, the assumption is that the motion vector $[u, v]$ is *the same* for every pixel in a small window around that point; the size of this window is a parameter that can be modified. This is the method that is used in the presented voting-based algorithm.

By writing (3) for every pixel i in the window, the result is an *overconstrained system* of N equations (the number of pixels in the window) with only 2 unknowns, u and v:

$$\underbrace{\begin{bmatrix} I_{x_1} & I_{y_1} \\ I_{x_2} & I_{y_2} \\ \vdots & \vdots \\ I_{x_N} & I_{y_N} \end{bmatrix}}_{A} \begin{bmatrix} u \\ v \end{bmatrix} = -\underbrace{\begin{bmatrix} I_{t_1} \\ I_{t_2} \\ \vdots \\ I_{t_N} \end{bmatrix}}_{b} \tag{5}$$

where the subscript represents the pixel number. By making the above notations it results:

$$A \cdot \begin{bmatrix} u \\ v \end{bmatrix} = b \tag{6}$$

The above is an *overconstrained* system of N equations and 2 unknowns and in general, has no solution. However, it can find the "closest solution", the one that minimizes the error:

$$E = \left\| A \begin{bmatrix} u \\ v \end{bmatrix} - b \right\|^2 \tag{7}$$

by solving the *normal equation*:

$$(A^T A) \begin{bmatrix} u \\ v \end{bmatrix} = A^T b \tag{8}$$

If $A^T A$ is invertible, we have:

$$\begin{bmatrix} u \\ v \end{bmatrix} = (A^T A)^{-1} A^T b \tag{9}$$

which gives the following final expressions:

$$\begin{cases} u = \dfrac{\sum I_y I_t \cdot \sum I_x I_y - \sum I_x I_t \cdot \sum I_y^2}{\sum I_x^2 \cdot \sum I_y^2 - \left(\sum I_x I_y\right)^2} \\ v = \dfrac{\sum I_x I_t \cdot \sum I_x I_y - \sum I_y I_t \cdot \sum I_x^2}{\sum I_x^2 \cdot \sum I_y^2 - \left(\sum I_x I_y\right)^2} \end{cases} \tag{10}$$

where the summations are performed over the whole window around the point (x, y).

If the point (x, y) corresponds to a *flat region*, then the motion vector cannot be computed, as the matrix

$$M = A^T A = \begin{bmatrix} I_x^2(x,y) & I_x(x,y)I_y(x,y) \\ I_x(x,y)I_y(x,y) & I_y^2(x,y) \end{bmatrix} \quad (11)$$

(which is the *second-moment matrix* of the *Harris corner detector*) is not invertible (or close to singular).

It can be seen from (10) that the complexity of calculating the value of one motion vector for 1 frame is $O(W^2)$, where W is the size of the window over which the summations are done.

To combine this technique with the following one, a consideration was made. A rectangular grid is considered over the image space and computes the motion vector at the center of each block/grid cell.

2.2. Block matching

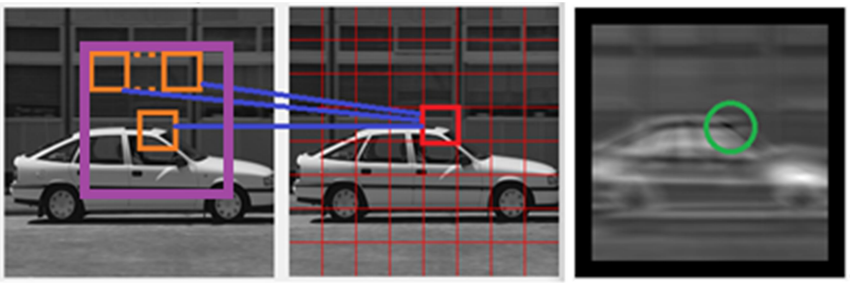

Figure 2 – From left to right: a. reference frame; b. current frame; c. (dis)similarity function. Where: red – current block, orange – sliding window, purple – search region, green – minimum of the dissimilarity function

The image is divided at time t (*current frame*) into a rectangular grid of blocks, as in Figure 2. Then, for every block, the aim is to find out the location in image space where that block was in the previous frame $t-1$ (*reference frame*) – that is, the location that best matches the current block [11]. To do so, it was considered a *sliding window* [10] (having the same size as the block) that swipes the reference frame and for each location in the reference frame, it was computed similar the region in the sliding window is to the current block. For that, it was used a *(dis)similarity function* such as the *mean squared error (MSE)*.

So, for a block at coordinates (x, y), and a particular position of the sliding window, displaced by (d_x, d_y) from (x, y), the *dissimilarity/cost function* of the two regions is:

$$\epsilon(d_x, d_y) = \sum_{i,j} \left(I_t(x+i, y+j) - I_{t-1}(x+d_x+i, y+d_y+j) \right)^2 \quad (12)$$

where the summation is performed on a block-sized region around point (x, y) show in Figure 2. As the difference between the intensities of the two regions at

corresponding locations is bigger, the cost function is bigger. So, the challenge is to find the displacement (d_x, d_y) that minimizes the cost function [9]. To do that, the motion vector (u, v) is chosen to be:

$$(u, v) = -argmin_{(d_x,d_y)}\, \epsilon(d_x, d_y) \tag{13}$$

Alternatively, it can be used a *similarity* function such as cross-correlation and try to *maximize*:

$$\epsilon(d_x, d_y) = \sum_{i,j} I_t(x+i, y+j) I_{t-1}(x+d_x+i, y+d_y+j) \tag{14}$$

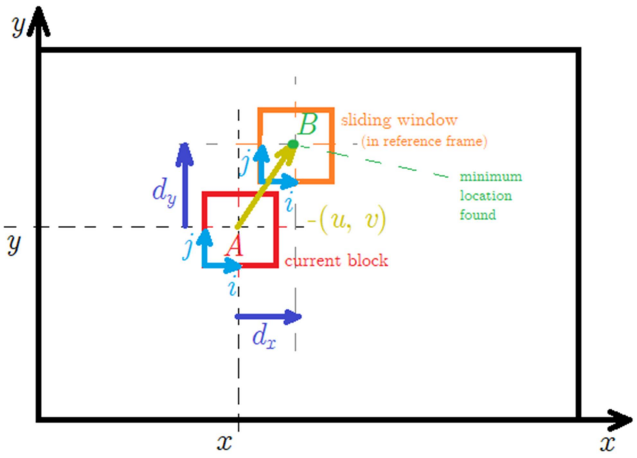

Figure 3 - The *motion compensation vector* $\overrightarrow{AB}$

If it is denoted by A the center of the current block and B the location that gives the minimum cost, it can be seen that the point has moved from B to A, so the motion vector at point A is $\overrightarrow{BA}$ (Figure 3):

$$\begin{bmatrix} u \\ v \end{bmatrix} = \overrightarrow{BA} = \begin{bmatrix} x_A - x_B \\ y_A - y_B \end{bmatrix} \tag{15}$$

If it is known that the motion is less than a specific value u_{max} pixels on x direction, d_x can be restricted to vary in the interval $[-u_{max}, u_{max}]$ (similarly for y direction; thus, the restriction is the search for a block to a *search region* around (x, y) – see Figure 2).

The complexity of the method is $O(S^2B^2)$ for 1 motion vector and for 1 single frame, where S is the size of the search region and B is the block size.

2.3. The Voting method

This section presents the output of the two previous methods with (u_1, v_1) and (u_2, v_2). Then the voting method considers the output of the voting method as the arithmetic means of the two:

$$(u_{voting1}, v_{voting1}) = \left(\frac{u_1 + u_2}{2}, \frac{v_1 + v_2}{2}\right) \tag{16}$$

Another possibility is to take the mean of both magnitude and angle, such as in Figure 4.

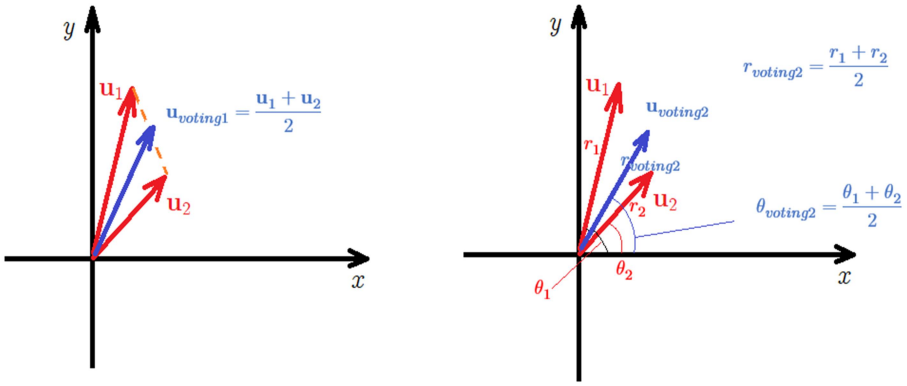

Figure 4 – The two cases for the voting method proposed

3. Implementation details

The motion estimation algorithms presented above were implemented in MATLAB.

The application opens a video and then loops through every frame, generates, and displays a motion field between the current and previous frame. In this case, this application was considered a rectangular grid over the image space and perform the motion estimation in the center of every grid cell.

For the first method - optical flow - the spatial derivatives of the image I_x, I_y were computed using a Sobel filter and I_t as the difference between the current and previous frame. The final motion vector components using was computed using equation (10).

As for the second method, for each location was looping through the search region, and compute the cost function using equation (12). The application keeps track of the position that gives the minimum value for the cost function and then calculates the motion vector by using equation (15).

4. Results

The results of the voting approach are displayed in Figure 5.

Figure 5 – Proposed Voting Method. From left to right: a. original image, b. optical flow, c. block matching, d. voting approach 1 e. voting approach 2

The methods presented above are very computationally intensive and only worked for very small size input images. The optical flow method often gives highly inaccurate results, especially at the boundary of objects, where the assumption of constant motion field is violated. The optical flow method is unstable in flat regions (the denominator in (10) goes to zero since the matrix in equation (11) is singular). Moreover, this method is susceptible to noise and only works for tiny motions (up to a few pixels). The Block-matching method is more robust than the previous, but unfortunately, the complexity is higher. However, many optimizations can be added to improve the efficiency of the method, such as 2D logarithmic search, pixel subsampling, pixel projection etc. [4]

Some performance measurements are given below in Table 1 and Table 2.

	Optical flow	Block matching
Parameters:	window size = 2	Search region: 16
88 × 60 × 2 frames	9.46 s	9.62
352× 240 × 2 frames	14.09	14.63

Table 1 – Performance measurements with Block size 8

Block matching	
Search region	**Time**
32	16.65
16	14.63
8	13.78
2	15.06

Optical flow	
Window size	**Time**
2	14.15
3	13.47
1	13.38

Table 2 – Performance measurements in time for block-matching and the optical flow

5. Conclusions

Although motion estimation is an essential topic in computer vision, most of the algorithms are very computationally intensive, and there is still room for improvement, especially for real-time applications. Every class of algorithms is suitable for a specific problem, and one needs to consider the requirements of the underlying application when choosing the appropriate method.

The presented motion detection approach will be used in the future in an unsupervised system containing various voting-based modules [21-23], with the explicit purpose of ensuring increased reliability and robustness, as well as better confidence in the obtained results.

Acknowledgements

This work was supported by a grant of the Romanian Ministry of Research and Innovation, CCCDI - UEFISCDI, project number PN-III-P1-1.2-PCCDI-2017-0689 / „Lib2Life- Revitalizarea bibliotecilor și a patrimoniului cultural prin tehnologii avansate” / "Revitalizing Libraries and Cultural Heritage through Advanced Technologies", within PNCDI III.

References

[1] Bergen J.R., Anandan P., Hanna K.J., Hingorani R., *Hierarchical model-based motion estimation*, In Sandini G. (eds) Computer Vision — ECCV'92. ECCV 1992. Lecture Notes in Computer Science, vol 588. , 1992, Springer, Berlin, Heidelberg, DOI:10.1007/3-540-55426-2_27

[2] Thomas B. Moeslund, Erik Granum, *A Survey of Computer Vision-Based Human Motion Capture*, Computer Vision and Image Understanding, Volume 81, Issue 3, pp 231-268, DOI: 10.1006/cviu.2000.0897, 2010.

[3] Forsyth, David A., Ponce, Jean, *Computer Vision: A Modern Approach*, Prentice-Hall Professional Technical Reference, 2002

[4] Claudette Cédras, Mubarak Shah, *Motion-based recognition a survey*, Image and Vision Computing, Volume 13, Issue 2, pp. 129-155, DOI: 10.1016/0262-8856(95)93154-K, 1995.

[5] Torr, P. H. S, Zisserman, A., *Feature Based Methods for Structure and Motion Estimation*, in Vision Algorithms: Theory and Practice, Springer Berlin Heidelberg, pp 278-294, 2000

[6] Rui Xu, David Taubman, Aous Thabit Naman, *Motion Estimation Based on Mutual Information and Adaptive Multi-scale Thresholding*, in Image Processing, IEEE Transactions, vol.25, no.3, pp.1095-1108, March 2016.

[7] G. de Haan, P. W. A. C. Biezen, H. Huijgen, O. A. Ojo, *True-motion estimation with 3-D recursive search block matching*, IEEE Transactions on Circuits and Systems for Video Technology, vol. 3, no. 5, pp. 368-379, DOI: 10.1109/76.246088, Oct. 1993.

[8] Shan Zhu, Kai-Kuang Ma, *A new diamond search algorithm for fast block-matching motion estimation*, IEEE Transactions on Image Processing, vol. 9, no. 2, pp. 287-290, Feb. 2000, DOI: 10.1109/83.821744.
[9] Yao Nie and Kai-Kuang Ma, *Adaptive rood pattern search for fast block-matching motion estimation*, IEEE Transactions on Image Processing, vol. 11, no. 12, pp. 1442-1449, Dec. 2002, DOI: 10.1109/TIP.2002.806251.
[10] Jianhua Lu, M. L. Liou, *A simple and efficient search algorithm for block-matching motion estimation*, IEEE Transactions on Circuits and Systems for Video Technology, vol. 7, no. 2, pp. 429-433, DOI: 10.1109/76.564122, April 1997.
[11] M. Brunig, W. Niehsen, *Fast full-search block matching*, IEEE Transactions on Circuits and Systems for Video Technology, vol. 11, no. 2, pp. 241-247, DOI: 10.1109/76.905989, Feb 2001.
[12] D. Sun, S. Roth, M. J. Black, *Secrets of optical flow estimation and their principles*, IEEE Computer Society Conference on Computer Vision and Pattern Recognition, San Francisco, CA, 2010, pp. 2432-2439, DOI: 10.1109/CVPR.2010.5539939, 2010.
[13] L. Xu, J. Jia, Y. Matsushita, *Motion Detail Preserving Optical Flow Estimation*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 34, no. 9, pp. 1744-1757, Sept. 2012, DOI: 10.1109/TPAMI.2011.236.
[14] Argyriou, V., Vlachos, T, *A Study of Sub-pixel Motion Estimation using Phase Correlation,* In BMVC pp. 387-396, 2006
[15] A. French *Optical Flow*, Computerphile, 2019.
[16] M. Li, M. Biswas, S. Kumar, Truong Nguyen, *DCT-based phase correlation motion estimation*, 2004 International Conference on Image Processing, 2004. ICIP'04. Vol. 1. IEEE, 2004.
[17] Rui Xu, David Taubman, Aous Thabit Naman, *Motion Estimation Based on Mutual Information and Adaptive Multi-scale Thresholding*, in Image Processing, IEEE Transactions, vol.25, no.3, pp.1095-1108, March 2016.
[18] I. Essa *A., Bobick Introduction to Computer Vision*, udacity.com, 2015.
[19] M. Shah, *Optical Flow*, UCF Computer Vision Video Lectures, 2012.
[20] A. K. Katsaggelos, *Fundamentals of Digital Image and Video Processing*, Northwestern University.
[21] Costin-Anton Boiangiu, Radu Ioanitescu, Razvan-Costin Dragomir, *Voting-Based OCR System*, The Journal of Information Systems & Operations Management, Vol. 10, No. 2, 2016, pp. 470-486.
[22] Costin-Anton Boiangiu, Mihai Simion, Vlad Lionte, Zaharescu Mihai – *Voting Based Image Binarization*, The Journal of Information Systems & Operations Management, Vol. 8, No. 2, 2014, pp. 343-351.
[23] Costin-Anton Boiangiu, Paul Boglis, Georgiana Simion, Radu Ioanitescu, *Voting-Based Layout Analysis*, The Journal of Information Systems & Operations Management, Vol. 8, No. 1, 2014, pp. 39-47.