Appeared in The Proceedings of the 3rd International Conference on Entertainment Computing, 2004, LNCS (http://www.springer.de/comp/lncs/index.html)

Optical-flow-driven Gadgets for Gaming User Interface

Zoran Zivkovic

Intelligent Autonomous Systems Group, University of Amsterdam,
1098SJ Amsterdam, The Netherlands
zivkovic@science.uva.nl
http://www.zoranz.net

Abstract. We describe how to build a VIDEOPLACE-like vision-driven user interface using "optical-flow" measurements. The optical-flow denotes the estimated movement of an image patch between two consecutive frames from a video sequence. Similar framework is used in a number of commercial vision-driven interactive computer games but the motion of the users is detected by examining the difference between two consecutive frames. The optical-flow presents a natural extension. We show here how the optical-flow can be used to provide much richer interaction.

1 Introduction

Vision-driven user interfaces are getting close to mass usage because computers are constantly becoming faster and cameras are getting cheaper. Facial and hand gesture recognition from video images remain therefore to be hot topics in the computer vision society [2]. Impressive results are reported and many applications seem possible. However, a real-world user-interface should fulfill a number of extremely difficult requirements: the interface should be very easy and natural to use, there should be no initialization and the system should work in difficult and changing environment conditions. It seems that the only actual commercial real-world vision-based interfaces are currently present in the gaming industry. The available games demonstrate rich and enjoyable interaction using just some simple computer-vision techniques. Although the techniques are often far away from the current state of the art in the computer-vision research, the used methods are fast and robust and fulfill the mentioned difficult real-world requirements. Furthermore, the gaming industry is particularly suitable for application of the computer-vision algorithms since the remaining imperfections of the vision-based interface are tolerated by the game players and often actually considered as an additional challenge.

There are a number of games and other systems that use a vision-based interface. We consider in this paper the "real" real-world systems that fulfill the mentioned requirements and can be actually used by anybody that has a web-cam attached to a PC. As we mentioned, it seems that the only actual commercial usage of the vision-based interfaces is present in the gaming industry. The currently available games use the

VIDEOPLACE-like interaction model [9] where the users see themselves embedded into computer graphics with some computer generated objects that react to their movements. See section 2 for a detailed description. The IntelPlay Me2cam [12] and Vivid Group Gesture Xtreme [11] systems segment the player from the background in order to detect the user gestures. However, the segmentation requires initialization and a special setting. Another standard technique used for vision based interfaces is the detection of the skin colored regions in an image [21], but this is not robust to light conditions. The two commercial game suites of interest are the Reality Fusion Game Cam [3] and the Sony Eye Toy [4]. These games use simple difference between two successive images to detect the motion of the users (see section 3). The detected movements of the players are used to interact with the virtual objects. This is fast and robust and does not require initialization or a special setting. We should mention also the QuiQui game [8] where a computer animated avatar mimics user actions. This game is also using the simple image-differencing.

In this paper we describe a framework for building VIDEOPLACE-like vision-driven user interfaces using the “optical-flow” measurements. The “optical-flow” denotes the movement of an image patch between two frames of a video sequence. There are various techniques for estimating the optical-flow [13,14,20]. The mentioned currently available vision-driven games are using the image-differencing technique to detect user movements. The optical-flow presents a natural extension of the image-differencing. The optical-flow not only detects the movement but also gives us an estimate of the direction and the speed of the movement. This simple extension allows much richer interaction while the described requirements for the “real” real-world systems are still fulfilled. We implement a system that is using the standard image-differencing and a system that is using the optical-flow-based measurements. The two techniques are evaluated and compared both subjectively and objectively. We also present how optical flow can be used to perform some more complex tasks that are not possible using the standard image-differencing technique.

The paper is organized as follows. First, in Section 2 we describe the VIDEOPLACE-like interaction framework that is used for various interactive systems and in the vision-driven computer games. In Section 3 we describe the image-differencing and how it is usually used to detect the movement of the user in the vision-driven games. We also describe a robust version of a standard technique for estimating the optical-flow and how it can be used for vision-based interaction. The experiments are reported in Section 4 and conclusions in Section 5.

2 VIDEOPLACE-like Interaction Model

A VIDEOPLACE-like interaction involves an installation where the users see themselves, or some representation of themselves, on a video-projection screen together with some additional computer graphics. A realization is presented in the figure 1. Another simpler version would be just a computer monitor and a web-camera. First experiments using such interaction model were conducted by the artist Myron Krueger in his work VIDEOPLACE as early as 1969. See a description in [9]. This interaction model is used later by other artists for various installations [7], by computer-vision researchers to demonstrate their work [5,21] and finally in the gaming industry [3,4,11,12]. The futuristic interface from the movie “Minority Report”(2002) (designed by John Underkoffler) is another example.

2.1 Virtual Mirror Effect

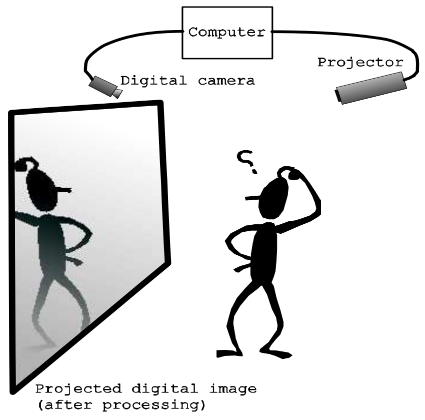


Fig. 1. VIDEOPLACE-like interaction model. A typical simple realization

The VIDEOPLACE-like interaction is based on the illusion of a mirror. Regular mirror is a common physical object in our environment and this makes a VIDEOPLACE-like virtual mirror a natural and easy to use interface [10]. It is also a simple way to provide the users with a certain level of immersion into a virtual world. There are some technical difficulties in realizing a completely realistic mirror illusion:

- If we move in front of a real mirror our view point is changing and consequently also the reflected image we see in the mirror. Since, the digital camera is static it is very difficult to simulate this effect. We would need to track the user’s eyes position and to change the projected images accordingly. Even then, this would be possible only when there is a single user. To the best of our knowledge, there is no such a system that is able to compensate for the view point changes.
- Another related but simpler problem is the eye-contact. Having the eye-contact with yourself when you look at your reflection is important for the mirror illusion. An approximate solution is to use a semitransparent projection screen and to put the camera behind the virtual mirror somewhere at the eye height. See for example [5]. It is also possible to track the person eyes and apply the corrections [6];

Fortunately, in practice even the simplest realization with a web-camera and a computer monitor is enough to produce a reasonable effect and provoke interaction. Furthermore, in the early work of Myron Krueger there was just a shadow of the person and in the QuiQui game [8] there is a computer animated avatar.

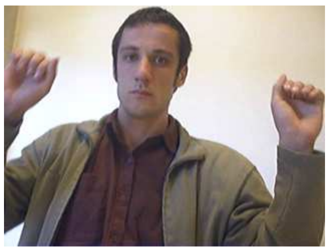
2.2 Vision-driven Gadgets

Beside the mirror-like image of the users, the VIDEOPLACE-like interaction involves some additional computer generated objects that are presented on the screen. The objects react to the movements of the users and we denote them as “vision-driven gadgets”. There is a variety of types of such objects and we will mention the most common ones:

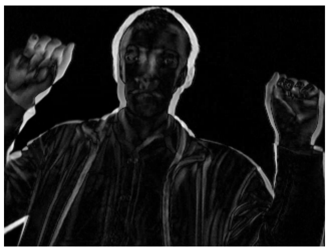
- **Static object:** These objects do not react to the user movements directly. For example there is often some static computer graphics present or there are numbers to show current score in the games etc.
- **Button:** This is the basic component of almost every user interface. In the VIDEOPLACE-like interaction the user should be able to use his movement to press a button. The current games are based mainly on this type of object or a variation of this type. The most common variation is the “moving button”. The object moves around the image and when the user selects it the object changes its behavior. For example in a game from [4] bubbles fly around the screen and they explode when the user selects them. In some other games from [3] and [4] the objects bounce away when selected. The button is selected if there is a movement in the area of the image occupied by the button. In the current games the image-differencing technique is used to detect the movement and we will show how this can be improved by using the optical-flow.
- **Movable object:** Another common part in a user interface. For example the icons in the Windows user interface are dragged using the mouse. In the vision-based interface the user should be able to move the objects of this type using body movements. In the currently available games that use the image-differencing this is not possible. As we mentioned, in some games from [3] and [4] the objects bounce away when selected and this, in a way, controls the movement of the objects but very roughly. The interaction is still button-like. The optical flow provides also information about the direction and the magnitude of the movement and this can be used to move the objects and provide much richer interaction.

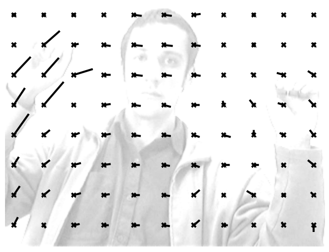
3 Detecting and Measuring Motion

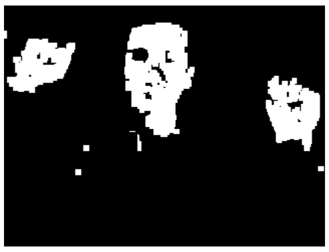
A vision-based interface is driven by the movements of the users that are observed by a digital camera. Robust techniques are needed to detect and measure the motion of the users. We will discuss the common motion detection techniques in this section. The techniques are illustrated in figure 2. In the VIDEOPLACE-like framework the presented image consists of a mirror-like image of the users and a number of augmented computer generated vision-driven gadgets. The gadgets react to the user movements. We will denote the region of the image occupied by a "vision-driven-gadget" by W. We need to detect and measure the movement in this region.


a) a frame from an image sequence

b) the next frame from the sequence


c) the difference D between two frames


d) optical flow for some image points (lines present displacements)


e) skin color segmentation

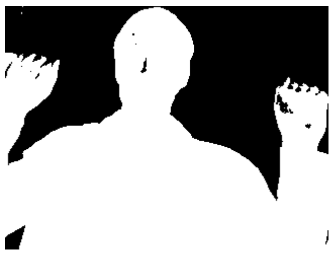

f) background subtraction

Fig. 2. Common motion detection techniques used for vision-driven interaction

3.1 Image Differencing

We will usually have RGB images. Let us use $R_0(\mathbf{x})$, $G_0(\mathbf{x})$, $B_0(\mathbf{x})$ and $R_1(\mathbf{x})$, $G_1(\mathbf{x})$, $B_1(\mathbf{x})$ to denote the RGB values of a pixel at position $\mathbf{x}$ in two consecutive images obtained from the camera. Here $\mathbf{x}=[x\ y]^T$ is a vector, where x and y are the coordinates of the pixel within an image. The change of the image values within a region W can be described by:

$$D=\sum_{\text{all x withinW}} \{(R_1(\mathbf{x})- R_0(\mathbf{x}))^2+(G_1(\mathbf{x})- G_0(\mathbf{x}))^2+(B_1(\mathbf{x})- B_0(\mathbf{x}))^2\} \tag{1}$$

The motion can be simply detected by checking if D is greater than a predefined threshold value. The currently available games [3][4] use this or a similar technique to detect the motion of the users. The motion is used for a “button-like” interaction we described in the previous section. If motion is detected within W, the button is activated. Note that in this way we do not extract the direction of the motion and the value D is just weakly related to the magnitude of the motion. See figure 2c for an example.

3.2 Optical Flow

If a vision-driven-gadget covers a region W we would like to estimate the 2D displacement $\mathbf{d}=[d_x\ d_y]^T$ (optical flow) of this patch between two consecutive images that are captured by the camera. We will present a standard techniques for estimating the optical flow from [13][14] and modify it to get more robust results.

For simplicity we will use the gray intensity images. If we have RGB images we can get the intensity $I(\mathbf{x})$ of a pixel by $I(\mathbf{x})=(R(\mathbf{x})+G(\mathbf{x})+B(\mathbf{x}))/3$. For an intensity image I_0 and a patch W, the goal is to find the patch in the next image I_1 that is the most similar to the initial patch. If we put this into equations the goal would be to find the displacement **d** that minimizes:

$$J(\mathbf{d})=\sum_{\text{all x withinW}} (I_1(\mathbf{x}+\mathbf{d})- I_0(\mathbf{x}))^2 \tag{2}$$

If we use a truncated Taylor expansion approximation of (2) we get:

$$Z\mathbf{d}=\mathbf{e}, \tag{3}$$

where the 2x2 matrix Z and the vector **e** are given by:

$$Z=\sum_{\text{all x withinW}} \begin{bmatrix} g_x^2 & g_x g_y \\ g_x g_y & g_y^2 \end{bmatrix} \tag{4}$$

$$\mathbf{e}=\sum_{\text{all x withinW}} (I_1- I_0)[g_x\ g_y]^T \tag{5}$$

Here g_x and g_y present the derivatives of the initial intensity image I_0 in x and y direction. We compute the derivatives using a simple Sobel operator [19].

The procedure form [13] repeats the Taylor approximation (2) iteratively. If $\mathbf{d}(k)$ presents the estimated displacement at k-th iteration the iterative solution is given by:

$$\mathbf{d}(k+1)= \mathbf{d}(k)+Z^{-1}\,\mathbf{e}(k) \tag{3}$$

At each iteration **d**(k) is used to wrap I_1 and recalculate **e**(k). The matrix Z remains the same. The procedure stops when the displacement estimate does not change any more or the maximum number of iterations is reached. The estimated displacement **d** can be used in various ways. See the experimental section for some examples.

Optical-flow estimates are often very noisy. Texture in an image is important to reliably estimate the position of an image patch W. This is known as the "aperture problem" [20]. However, we can avoid this in the following way. If the patch is in an area of the image with uniform intensity, the motion estimate will be very noisy. This can be detected using the eigenvalues e_1 and e_2 of the matrix Z. See also [15][16]. Both eigenvalues will be small and can set the motion to zero since it can not be reliably estimated. If the W is on a line-like structure in the image, the motion is only well defined in the direction normal to the line. This situation is detected when one of the eigenvalues is small and the other large. We then calculate the motion only in the direction of the eigenvector that correspond to the larger eigenvalue. This direction is the direction normal to the line. Only when both eigenvalues are larger than a threshold we calculate the full optical flow displacement as described.

Calculating the optical-flow in real-time for the whole image might require a lot of computing power. However, we only need optical-flow measurements for the visual-driven gadgets which can be done very fast.

3.3 Other common motion detection methods

Among other common techniques for detecting and measuring motion of the user, we will mention the "background subtraction" and the "skin color detection".

Background subtraction is a method that tries to distinguish the user from a static background. See an example in figure 2e. The IntelPlay Me2cam [12] and Vivid Group Gesture Xtreme [11] systems use this technique, but a special setting and initialization is needed. Background subtraction is also sensitive to environment changes and camera movements. In such cases the system should be re-initialized. Some automatic adaptive methods exist [22,23], but they are slow in adapting and they assume that the camera most of the time observes background which is not the case when a user is using a vision-based user interface.

Skin color detection is a common technique used for building vision based interfaces [21]. Regions in the image that have the color similar to that of the human skin are detected. These regions usually correspond to the hands and faces of the users. See figure 2f. The movement of these regions can be used for interaction. See [21] for some examples. Unfortunately, the observed colors depend heavily on the light conditions and also the camera that is used. Therefore, for good performance a careful calibration is required.

In conclusion, both techniques do not fulfill the difficult "real" real-world requirements and will not be considered further in the experiments.

4 Experiments

In this section we will evaluate and compare the performance of the image-differencing and the optical-flow techniques.

4.1 Qualitative Comparison

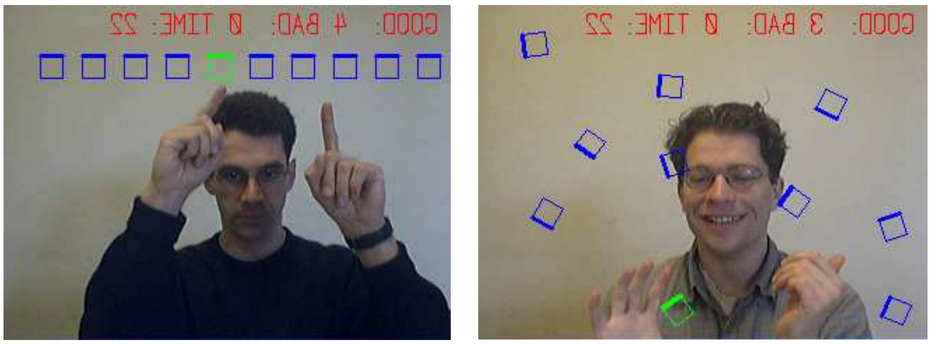
Both techniques comply with the requirements for the “real” real-world algorithm mentioned in the introduction. Light conditions are not important, there is no initialization etc. This is probably why the commercial vision-driven games often use the image-differencing. Optical-flow measurements preserve the good properties of the image-differencing technique but provide more information. A summary is given in the table 1.

Both techniques present a simple low-level analysis of the scene. However, the optical-flow might be in perspective used also for some higher semantic level understanding the user actions. Jahansson, in his famous experiments [1], filmed people in darkness wearing small light bulbs on their body. Moving cloud of identical bright dots on a dark field was enough to detect people in the scene and even assess their activity, gender and age. Inspired by [1], computer vision algorithms for detecting human activities from optical-flow were reported in [17] and [18]. These results might be important for further extension of the interface we present in this paper.

Table 1. Qualitative comparison of the two simple methods. A summary

	Image-differencing	Optical-flow
No initialization	+	+
Works under different environment conditions	+	+
Simple to compute	+	+
Motion detection	+	+
Motion magnitude	≈	+
Motion direction	-	+

4.2 Buttons Experiment



a) simple menu

b) complex menu with randomly distributed buttons

Fig. 3. Buttons experiment. Two situations are presented.

We analyzed the performance of the two techniques on the basic element of any user interface, the button. (see section 2.2). We used simple static buttons that are selected when the movement in the area W of the button is detected. We implemented a system that is using the standard image-differencing technique. To filter out some possible noisy detection we wait until the motion is detected for two consecutive frames. We implemented also a system that is using the optical-flow. The button that is driven by the optical-flow will react only to the movement in the right direction. This filters out many unwanted motions. According to the users in our experiment, this is a very natural way of pressing a virtual button.

The experiments were performed using 14 people. The task was to press the button presented using green color. See figure 3. We counted number of well selected buttons and the number of wrongly selected buttons during a period of 30 seconds. After a button was selected there was a 0.5 second pause before a new button is randomly chosen and highlighted using green color. The users had 2 trials to learn to use the interface and the results from the third trial are reported in table 2. We observe a big improvement when the optical-flow is used. In the complex menu case (figure 3b) image-differencing was completely useless.

Table 2. Results of the experiments. We report the difference between the number of well selected buttons and the number of wrongly selected buttons. Mean value, standard deviation, maximum and minimum are reported

	Image-differencing (simple menu)	Optical-flow (simple menu)	Image-differencing (complex menu)	Optical-flow (complex menu)
Mean	11.2	19.2	-20.9	5.7
Std	6.6	2.9	17.8	4.9
Min	1	13	-54	1
Max	20	24	0	14

4.3 Movable Objects

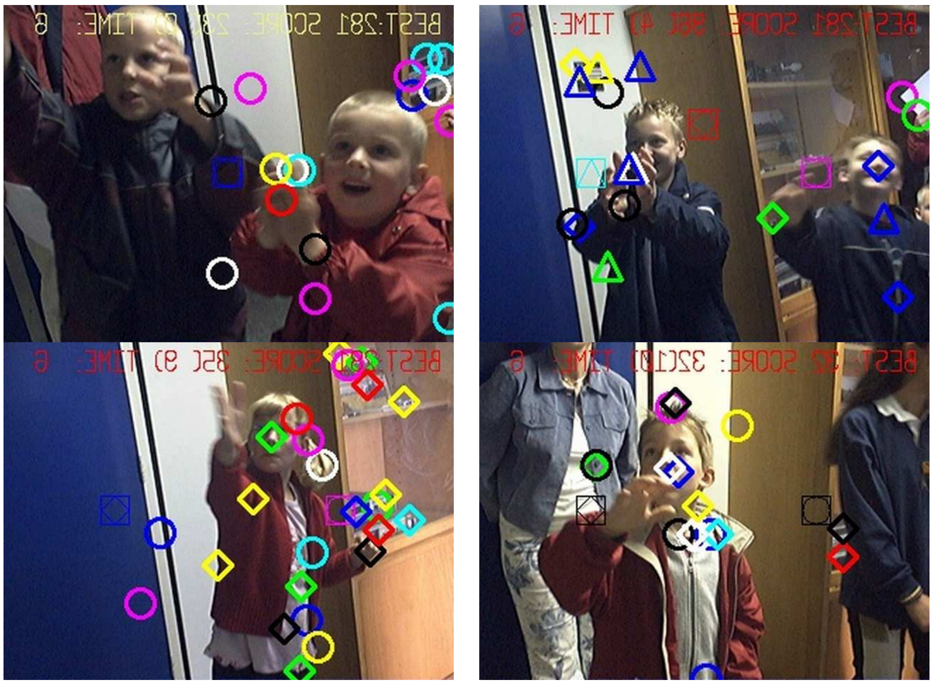


Fig. 4. Moving objects. Some images from a public performance are presented. Children sorted virtual objects on color and shape

Optical flow can be used to perform another basic user-interface task: dragging a virtual object. A vision-driven-gadget is moved to the new position according to the calculated displacement **d**. We realized an interactive vision-driven game where the task was to sort virtual objects on shape and color. We had a large number of users during a set of public performances. It turned out that it was easy and fun for most people to use this interface. We had also a large number of children participants. The children enjoyed the game enormously and they were also able to learn to use the interface very fast. See some screen-shots in figure 4.

6 Conclusions

We proposed a way to use optical flow for vision-based user interfaces and presented how various basic user-interface tasks can be realized. The optical-flow calculation is fast, simple and robust. It does not require initialization or some special setting and it is potentially very interesting for making “real” real-world vision-driven user interfaces. An example is the gaming industry where optical flow presents a natural extension of the currently used simple and robust techniques.

References

1. Johansson, G.: Visual Perception of Biological Motion and a Model for its Analysis, In: Perception and Psychophysics, Number 14, (1973) 201-211
2. Gong S., Zhang, H-J. (eds.): Proceedings of the IEEE International Workshop on Analysis and Modeling of Faces and Gestures – in connection with the International Conference on Computer Vision 2003, (2003)
3. Sony Computer Entertainment Inc.: Sony Eye Toy, www.eyetoy.com, (2003)
4. Reality Fusion Inc: GameCam, (1999)
5. Darrel, T., Baker, H., Crow, F., Gordon, G., Woodfill, J.: Magic Morphin Mirror. Presented at SIGGRAPH 1997- Electronic Garden (1997)
6. Yang, R., Zhang, Z.: Eye gaze correction with stereovision for video tele-conferencing. In: Proceedings of the Seventh European Conference on Computer Vision, (2002) 479–494
7. Rokeby, D.: Transforming Mirrors: Subjectivity and Control in Interactive Media. In: Simon Penny (ed.): Critical Issues in Electronic Media, Leonardo Electronic Almanac, Vol. 3, No. 4 (1995)
8. Höysniemi, J., Hämäläinen, P., and Turkki, L.: Using peer tutoring in evaluating the usability of a physically interactive computer game with children. In: Interacting with Computers, Vol. 15, Issue 2, (2003) 141-288
9. Krueger, M., Gionfriddo, T., Hinrichsen, K.: VIDEOPLACE-An Artificial Reality, In: Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI '85) (1985)
10. Ishii, H., Ullmer, B.: Tangible Bits: Towards Seamless Interfaces between People, Bits and Atoms, In: Proceedings of ACM Conference on Human Factors in Computing Systems (CHI '97) (1997) 234-241
11. Vivid Group Inc.: Gesture Xtreme System, www.vividgroup.com (1986)
12. D'Hoge H.: Game Design Principles for the IntelPlay Me2Cam Virtual Game System, Intel Technology Journal (2001)
13. Lucas B., Kanade T.: An iterative image registration technique with an application to stereo vision. In: Proceedings IJCAI81 (1981) 674–679
14. Baker S., Matthews I.: Lucas-Kanade 20 years on: A unifying framework part 1: The quantity approximated, the warp update rule, and the gradient descent approximation, In: International Journal of Computer Vision (2003)
15. Thomasi C., Kanade T.: Detection and tracking of point features. In: Carnegie Mellon University Technical Report CMU-CS-91-132, (1991)
16. Zivkovic Z., van der Heiden F.: Improving the selection of feature points for tracking, In: Pattern Analysis and Applications, accepted for publication
17. Davis J., Bobick A., The Representation and Recognition of Action Using Temporal Templates, In: Proceedings of IEEE Conference CVPR, (1997), 928-934
18. Song Y., Concalves L., Perona P.: Unsupervised learning of human motion. In: IEEE Transactions Pattern Analysis and Machine Intelligence, 25(7), (2003) 814-827
19. Horn B.: Computer Vision. Cambridge, Mass.: MIT Press, (1986)
20. Beauchemin S.S., Barron J.L.: The Computation of Optical Flow, In: ACM Computing Surveys 27(3) (1996) 433-467
21. Pentland A.: Looking at People: Sensing for Ubiquitous and Wearable Computing, In: IEEE Transactions Pattern Analysis and Machine Intelligence, 22(1), (2000) 107-119
22. Stauffer C., Grimson W.: Adaptive background mixture models for real-time tracking, In: Proceedings of IEEE Conference CVPR,(1999) 246-252
23. Zivkovic Z.: Improved adaptive Gausian mixture model for background subtraction, In: Proceedings of the International Conference Pattern Recognition, (2004)