Semi-automatic video object segmentation basing on hierarchy optical flow

Ming Zhao Jiajun Bu Chun Chen
School of Computer Science, Zhejiang University, Hangzhou, 310027, P.R.China
Contact: {bjj, chenc}@cs.zju.edu.cn

ABSTRACT

Abstract In the new MPEG-4 video coding standard, the semi-automatic video segmentation plays a key role in supporting object-oriented coding and enabling content-based functionalities. A novel hierarchy optical flow based semi-automatic video segmentation method is presented in this paper. The proposed segmentation method contains spatial and temporal segmentation. For the spatial segmentation, a point-based graphic user interface (PBGUI) is presented, with which the user can input easily, and then active contour model and tracking bug algorithm are applied to precisely define the video object of interest to be segmented. With the result of spatial segmentation, the temporal segmentation involves non-rigid object contour tracking and rigid object whole-tracking by hierarchy optical flow algorithm based on the Lucas-Kanade algorithm. And the tracking point selection algorithm is proposed to greatly improve the tracking performance in the rigid object whole-tracking. The experimental results show that the proposed algorithm can precisely segment video objects from video streams.

Keywords: MPEG-4, optical flow, video segmentation, object-oriented coding

1. INTRODUCTION

The emerging MPEG-4 standard has introduced the concept of content-based representation and content-based visual query. The new features of the MPEG-4 standard include enhanced coding efficiency, content-based interactivity and great error robustness for a large range of bit rates. In order to apply the MPEG-4 visual coding standard, each image of video sequences needs to be represented in terms of video objects[1], that is, the objects to be encoded in video sequences should be identified before the encoding process starts. So, video object segmentation has become more and more important.

Recent development of video object segmentation leads to two types of algorithms: automatic segmentation and semiautomatic segmentation [2][3]. In the case of automatic segmentation, the homogeneity of some lower visual features such as color, texture, gradient, motion or some combinations is defined to extract video object. This often leads to poor performance for applications where the video object is not consistent with the predefined homogeneity. Most automatic algorithms adopt motion information as a primary feature for object extraction. For semiautomatic algorithms[4][5][6], the user is required to identify the video objects of interest initially with the help of graphic user interface (GUI). At the same time, other information such as the rigidity of the video object and the quality of the video clip can be passed to the computer. Then semiautomatic algorithms can segment video objects from the following frames automatically. If the user is not satisfied with the results, he can stop the segmentation process and redefine the video object or adjust it.

Chuang Gu et al [4] presented a semi-automatic segmentation algorithm. A muliti-variate watershed algorithm is

used for the spatial segmentation. Then the segmented regions are tracked for the following frames. However, this method can not handle large non-rigid movement, and did not consider rigid objects. Munchurl Kim [6] proposed another semi-automatic segmentation method, which included inter-frame segmentation and intra-frame segmentation. A watershed algorithm is used for intra-frame segmentation. For inter-frame segmentation, boundary tracking and region tracking are used for non-rigid and rigid objects respectively. As the spatial segmentation [5] and intra-frame segmentation largely depend on the result of image segmentation, it is difficult for the user to interact with the system. The boundary tracking [6] is based on block matching algorithm Hence, its accuracy can not be ensured. And the region tracking [6] did not consider the tracking ability of the points inside the objects.

In this paper, a hierarchy optical flow based semi-automatic video object segmentation method is proposed, which can solve the above problems. The algorithm is composed of two modules: spatial segmentation and temporal segmentation. The spatial segmentation can precisely define the video object of interest to be segmented with the proposed point-based graphic user interface (PBGUI) assisted by user interaction. With the result of spatial segmentation, the temporal segmentation involves non-rigid object contour tracking and rigid object whole-tracking by hierarchy optical flow algorithm. Specially, the tracking point selection algorithm is proposed to greatly improve the tracking performance for rigid object whole-tracking. Comparing with other semiautomatic video object segmentation algorithms, this algorithm has the following features: (1) point-based graphic user interface (PBGUI); (2) hierarchy optical flow algorithm is used in both boundary and whole-tracking. (3) Whole-tracking is improved by tracking point selection algorithm

2. SPATIAL SEGMENTATION

Spatial segmentation is to get the accurate contour of the video object with the user's interaction. This is very important, for the result of temporal segmentation depends on spatial segmentation. For automatic segmentation, as there is no such a priori accurate segmentation, the segmentation result is hard to be satisfactory. On the other hand, the video objects are semantic, that is they are defined by people. So, without the user's interaction, the result is hard to meet the user's need. That's why automatic segmentation can not generally give satisfactory results.

In order to achieve the above goals, semi-automatic segmentation techniques must have two functions: first, devise a graphic user interface to convenience the user's input. Second, with the user's input, the video object's contour is obtained automatically. In this paper, a point-based graphic user interface is proposed to accomplish these tasks.

2.1 Graphic user interface to convenience the user's input

Munchurl Kim et al [6] proposed a strip based user interface. Users are required to draw a ring strip around the boundary of the video object. In the method, users are required to input too much data. As a mouse is good at locating, not at drawing, it is difficult to control the locus drawn by a mouse. So drawing a strip is not so easy. Generally speaking, the easiest way for a mouse to input is to draw a rough point, and then is accurate point, rough line, and at last accurate line. Literature [6] had in fact required to input rough line. So it can not be the best way. And what's more, it is nearly the worst way. Based on this observation, a point-based graphic user interface (PBGUI) is proposed in this paper, which only requires the user to input rough points. Needless to say, users can input data easily with PBGUI. The user's operation is very simply. He simply input several points around the boundary of the video object. Then the system can obtain the accurate contour of the video object. If the result is not so accurate, he can simply add some points near the wrong positions. A more accurate contour can be obtained. After a few interactions, the satisfactory contour is there.

2.2 Obtain the accurate contour with the user's interaction

In literature [6], the contour of the video object is obtained in the following way: first, a ring strip around the boundary of the video object is drawn. Then, image segmentation is performed using water shed algorithm. Next, the image segmentation regions are classified basing on the input ring strip. Finally, the boundary of the regions inside the ring strip is used as the contour of the video object. There are two disadvantages for this method: (1) it is difficult to modify the result, i.e. the contour of the video object. A good modification is to modify the wrong things, and retain the right things. Unfortunately, this is not true for this method, because the partial modification of the ring strip is not allowed. So, if the result needs modifying, the user must redraw the entire ring strip. As a result, the accurate areas can not be retained. (2) It is difficult to obtain the accurate contour. As the boundary totally depends on the result of the image segmentation, the wrong contour can not be adjusted to the positions where the user really wants them to be when the result is not accurate enough. Nevertheless, the proposed PBGUI in this paper can overcome these difficulties.

The PBGUI based algorithm to obtain the accurate contour includes four steps:

1) Active contour model or snake [7] is used to adjust the rough points input by the user. As the input points are near their real boundary, they can be adjusted to the boundary by active contour model.
2) An improved boundary tracking algorithm is proposed to get the contour of the video object. Given two points, the boundary between can be obtained with boundary tracking algorithm. In this paper, the tracking bug algorithm is used, for it can reduce the noise effects in the image. The common used tracking bug algorithm begins from a starting point until it reaches the ending point [8]. To make it less susceptible to noise and produce more accurate results, two improvements are made: first, the tracking bug begins from both the starting point and the ending point. All the pixels around the starting and ending point are considered. The next boundary point is taken as the pixel with highest average gradient. In this way, even if one of the starting and ending points are not accurate enough, a fine boundary can be acquired. Yet this is not true for the common used algorithm, which depends largely on the starting point. Second, Limits are enforced on how sharply the boundary can change directions to get a smooth boundary.
3) The mask of the video object is acquired from the contour using region flooding algorithm. To get the mask from the contour is to fill the region inside the contour. Considering that the contour can be at any possible position, the mask image is first expanded with the width of 1 pixel around it, resulting that any contour is within the image. Then, a seed flooding algorithm is applied to the mask image with the seeds selected as the four corners of the image. For pixels which can not be flooded, they are set as grey level 1, otherwise 0. As a result, the mask of the video objects is obtained as a mask image, in which pixels with grey level 1 belong to the video object.
4) If the result is not satisfactory, the user can adjust their input points or add some more points. And then go to step (2).

In conclusion, there are three advantages for the propose PBGUI based spatial segmentation:

- It has a friendly interface, which is easy for the user the input less data.
- It has good interactivity. It is easy for the user to modify the results.
- The result is very good.

3. TEMPORAL SEGMENTATION

3.1 Contour tracking for non-rigid object segmentation

For non-rigid object segmentation, Munchurl Kim et al [6] proposed an object boundary tracking algorithm basing on

block matching. However, the tracking result is not so good. Referring to literature [9], block matching is not good at motion estimation for small movement. So literature [6] used boundary refinement to adjust the tracking result. Nevertheless, the accuracy of boundary refinement itself can not be guaranteed, especially under the circumstances with other boundaries.

As indicated by literature [9], under the circumstances of small movement, the Fleet-Jepson method is the best for optical flow estimation, and the next is Lucas-Kanade method. And what's more Lucas-Kanade method is most efficient. So we choose Lucas-Kanade method for contour tracking as a balance of performance and efficiency. A Laplace hierarchy model is used in order that this method can be used under the circumstances of large movement.

3.1.1 Lucas-Kanade optical flow model

Lucas-Kanade method [10] assumes the motion vector remains unchanged over a particular block of pixels, denoted by B , that is:

$$v(X,t) = [v_x(t) \quad v_y(t)]^T = [a \quad d] \qquad X \in B \tag{1}$$

where $X = (x, y)$ is the coordinate of a pixel in block B . Obviously, this is a translation model. Although this model cannot handle rotational motion, it is possible to estimate a purely translational motion vector uniquely under this model provided that the block of pixels contains sufficient gray level variation. But if block B is small enough, any movement can be approximately processed. Lucas and Kanade defined the error in the optical flow equation over the block of pixels B as [10]:

$$E = \sum_{x \in B} \left(\frac{\partial I(X,t)}{\partial x} a + \frac{\partial I(X,t)}{\partial y} d + \frac{\partial I(X,t)}{\partial t} \right)^2 \tag{2}$$

where $I(X,t)$ is the intensity function. In order to minimize the error, we compute the partials of the error E with respect to a and d respectively, and set them equal to zero, that is:

$$\sum_{x \in B} \left(\frac{\partial I(X,t)}{\partial x} a + \frac{\partial I(X,t)}{\partial y} d + \frac{\partial I(X,t)}{\partial t} \right) \frac{\partial I(X,t)}{\partial x} = 0 \tag{3}$$

$$\sum_{x \in B} \left(\frac{\partial I(X,t)}{\partial x} a + \frac{\partial I(X,t)}{\partial y} d + \frac{\partial I(X,t)}{\partial t} \right) \frac{\partial I(X,t)}{\partial y} = 0 \tag{4}$$

Solving these equations simultaneously, we have

$$\begin{pmatrix} a \\ d \end{pmatrix} = \begin{bmatrix} \sum_{x\in B} \frac{\partial I(X,t)}{\partial x}\frac{\partial I(X,t)}{\partial x} & \sum_{x\in B} \frac{\partial I(X,t)}{\partial x}\frac{\partial I(X,t)}{\partial y} \\ \sum_{x\in B} \frac{\partial I(X,t)}{\partial x}\frac{\partial I(X,t)}{\partial y} & \sum_{x\in B} \frac{\partial I(X,t)}{\partial y}\frac{\partial I(X,t)}{\partial y} \end{bmatrix}^{-1} \begin{bmatrix} -\sum_{x\in B} \frac{\partial I(X,t)}{\partial x}\frac{\partial I(X,t)}{\partial t} \\ -\sum_{x\in B} \frac{\partial I(X,t)}{\partial y}\frac{\partial I(X,t)}{\partial t} \end{bmatrix} \tag{5}$$

where the matrix

$$\begin{bmatrix} \sum_{x\in B} \frac{\partial I(X,t)}{\partial x}\frac{\partial I(X,t)}{\partial x} & \sum_{x\in B} \frac{\partial I(X,t)}{\partial x}\frac{\partial I(X,t)}{\partial y} \\ \sum_{x\in B} \frac{\partial I(X,t)}{\partial x}\frac{\partial I(X,t)}{\partial y} & \sum_{x\in B} \frac{\partial I(X,t)}{\partial y}\frac{\partial I(X,t)}{\partial y} \end{bmatrix} \tag{6}$$

must not be odd. If it is odd, it means that the block does not contain sufficient gray level variation to estimate the motion vector.

3.1.2 Non-rigid object segmentation algorithm

The segmentation algorithm begins with the input of the video object's mask $Mask_t$ for the current frame F_t, and will output the video object's $Mask_{t+1}$ for the next frame F_{t+1}. It works as follows:

1) For each point on the boundary of $Mask_t$, its motion vector is computed using equation (5). If the motion vector of a point is not available, it will be interpolated from the other vectors.
2) Use the boundary and the motion vectors in the current frame to produce the boundary in the next frame. And interpolation is used to ensure that the boundary is closed.
3) $Mask_{t+1}$ is obtained from the boundary using the region flooding algorithm discussed in section 2.2.

In order to adjust to large movement and restrain noise, a Laplace hierarchy model is implemented. For each layer of the hierarchical pyramid, the above steps are executed. And the result of the bottom layer is $Mask_{t+1}$.

3.2 Whole-tracking for rigid object segmentation

If the video object's boundary is clear enough, the above contour tracking based segmentation method can produce good result. But if it is blurry, the result will be bad. It is because that, in this condition, there is not sufficient gray level variation so that the matrix (6) is odd or almost odd. As a result, the motion vectors computed from equation (5) is apt to be incorrect. This problem can not be completely solved by the contour tracking itself.

But for the rigid object, the whole object moves in the same way. So it is possible to use the whole motion as the boundary motion. Hence, a whole-motion model is used in this paper. The whole-motion model includes a

whole-translation model and a whole-affine model.

3.2.1 The whole-translation model

In fact, equation (1) is a translation model for block B , i.e. a block translation model. But it is not a whole-translation model. If we substitute block B with the region occupied by the whole object, equation (1) becomes a whole-translation model. In order to differentiate the whole-translation model from block translation model, the whole-translation model is denoted as:

$$M_T=[\, m_T^a, m_T^d \,]^T=T^{-1}B_T \qquad (7)$$

Where $M_T=[\, m_T^a, m_T^d \,]^T$ is the whole-motion vector, T^{-1} and B_T are the whole-translation version of matrix (6) and the right hand side matrix in equation (5).

3.2.2 The whole-affine model

The whole-affine model assumes that the motion parameters remains unchanged over the whole object region, also denoted by B , similar with equation (1), that is:

$$v(X,t)=[v_x(t) \quad v_y(t)]^T=[a+bx+cy \quad d+ex+fy]^T \quad X \in B \qquad (8)$$

Like equation (2), the error in the optical flow equation over the whole video object region B becomes:

$$E=\sum_{x\in B}\left(\frac{\partial I(X,t)}{\partial x}(a+bx+cy)+\frac{\partial I(X,t)}{\partial y}(d+ex+fy)+\frac{\partial I(X,t)}{\partial t}\right)^2 \qquad (9)$$

In order to minimize the error, we compute the partials of the error E with respect to a , b , c , d , e , and f respectively, and set them equal to zero, that is:

$$\begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix} = \begin{bmatrix} \sum I_xI_x & \sum xI_xI_x & \sum yI_xI_x & \sum I_xI_y & \sum xI_xI_y & \sum yI_xI_y \\ \sum xI_xI_x & \sum x^2I_xI_x & \sum xyI_xI_x & \sum xI_xI_y & \sum x^2I_xI_y & \sum xyI_xI_y \\ \sum yI_xI_x & \sum xyI_xI_x & \sum y^2I_xI_x & \sum yI_xI_y & \sum xyI_xI_y & \sum y^2I_xI_y \\ \sum I_xI_y & \sum xI_xI_y & \sum yI_xI_y & \sum I_yI_y & \sum xI_yI_y & \sum yI_yI_y \\ \sum xI_xI_y & \sum x^2I_xI_y & \sum xyI_xI_y & \sum xI_yI_y & \sum x^2I_yI_y & \sum xyI_yI_y \\ \sum yI_xI_y & \sum xyI_xI_y & \sum y^2I_xI_y & \sum yI_yI_y & \sum xyI_yI_y & \sum y^2I_yI_y \end{bmatrix}^{-1} \begin{bmatrix} -\sum I_xI_t \\ -\sum xI_xI_t \\ -\sum yI_xI_t \\ -\sum I_yI_t \\ -\sum xI_yI_t \\ -\sum yI_yI_t \end{bmatrix} \qquad (10)$$

(All sums are over $x \in B$.)

This equation represents the whole-affine model. If we use M_A, A^{-1} and B_A represent the three matrixes from left to right respectively, equation (10) can be rewrite as:

$$M_A=[m_A^a, m_A^b, m_A^c, m_A^d, m_A^e, m_A^f]^T=A^{-1}B_A \quad (11)$$

3.2.3 The tracking point selection algorithm

As indicated by J. Shi et al [11], some points on the video object are hard to track. If they are used to track the object, the result tends to be erroneous. So it is necessary to select the easy-to-track points for object tracking. J. Shi et al [11] also pointed out that the easy-to-track points must satisfy two requirements: first, both eigenvalues of matrix (6) must be large; second, they cannot differ by several orders of magnitude. Two small eigenvalues mean a roughly constant intensity profile within a window. A large and a small eigenvalue correspond to a unidirectional texture pattern. Only two large eigenvalues can represent patterns that can be tracked reliably. But there are only a few easy-to-track points inside the video object. If we only use them for tracking, the result will not be robust. As a large and a small eigenvalue correspond to a unidirectional texture pattern, this pattern is easy to track in one direction. If a pair of such points, which have different directional tracking pattern, is selected, they will enhance the tracking performance in every direction. So the proposed tracking point selection algorithm is as follows:

1) All the easy-to-track points all firstly selected.
2) If the number of these easy-to-track points is too small, pairs of unidirectional easy-to-track points, which have different directional tracking pattern, are selected, until the number of selected points reach a maximum value or there are no such point any more.

3.2.4 Rigid object segmentation algorithm

The segmentation algorithm begins with the input of the video object's mask $Mask_t$ for the current frame F_t, and will output the video object's $Mask_{t+1}$ for the next frame F_{t+1}. It works as follows:

1) Initialize translation and affine motion vectors: $M_T=[0,0]$ and $M_A=[0,0,0,0,0,0]$.
2) Tracking point selection algorithm is used to select the tracking points inside $Mask_t$.
3) The whole-translation model is used to compute the change in translation vector, i.e. $\triangle M_T$; then let $M_T=M_T+\triangle M_T$.
4) $-\triangle M_T$ is used to translate the next frame F_{t+1}.

5) If $||\triangle M_T|| > Thresh_T$ and the number of translation operation is less than $Iter_T$, go to step (3).

6) If there is only translation movement for the object, go to step (10).

7) The whole-affine model is used to compute the change in affine vector, i.e. $\triangle M_A$; then let $M_A=M_A+\triangle M_A$.

8) $-\triangle M_A$ is used to affine the next frame F_{t+1}.

9) If $||\triangle M_A|| > Thresh_A$ and the number of affine operation is less than $Iter_A$, go to step (7).

10) If $||\triangle M_A|| > Thresh_A$, go to step (3).

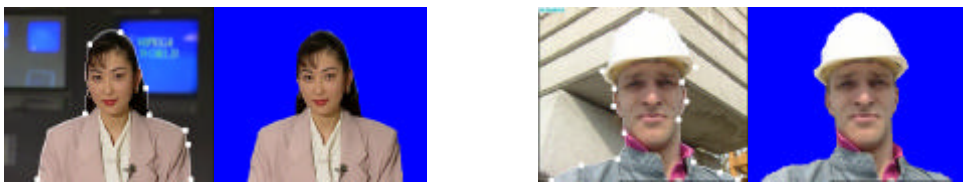
11) Let $M=[\, m_T^a + m_A^a, m_A^b, m_A^c, m_T^d + m_A^d, m_A^e, m_A^f \,]^T$. Then M is used to affine all the boundary points of $Mask_t$ to get the object boundary in the next frame. Interpolation is used to ensure that the boundary is closed. At last, $Mask_{t+1}$ is obtained from the boundary using the region flooding algorithm discussed in section 2.2.

In order to adjust to large movement and restrain noise, a Laplace hierarchy model is implemented. For each layer of the hierarchical pyramid, the above steps are executed. And the result of the bottom layer is $Mask_{t+1}$.

4. EXPERIMENTATL RESULTS

4.1 Spatial segmentation results

The spatial segmentation results are illustrated in Figure 1. We use AKIYO and FOREMEN as the test video sequences. The big white dots in the left image of (a) and (b) are the points input by the user. And the white lines around the video objects are the contours obtained by the system. As for AKIYO, its boundary is very clear. So the user only needs to input 9 points. As for FOREMAN, however, its boundary is not so clear and 19 points is still enough. The segmentation results are very good.



(a) (b)

Figure 1 spatial segmentation results based on PBGUI

60th frame 120th frame 180th frame 240th frame 298th frame

Figure 2 the temporal segmentation results for AKIYO based on contour tracking

8th frame 12th frame 20th frame 55th frame 100th frame

Figure 3 the temporal segmentation results for FOREMAN based on contour tracking

100th frame 150th frame 200th frame 250th frame 298th frame

Figure 4 the segmentation results for COASTGUARD based on whole-tracking

4.2 Contour tracking results for non-rigid object segmentation

The results of non-rigid object segmentation are given in Figure 2 and 3. We still use AKIYO and FOREMAN as the test video sequences. The results shown in Figure 1 are used as the spatial segmentation for the first frames. For the AKIYO sequence, the segmentation results are very good, for the images are clear with small movement. The results are satisfactory for all the frames in the video stream. For the FOREMAN sequence, the images are not so clear with large movement. So its segmentation results are not as good as AKIYO. The results are acceptable for the first 100 frames.

4.3 Whole-tracking results for rigid object segmentation

The results of rigid object segmentation basing on whole-tracking are shown in Figure 4. We use COASTGUARD as the test sequence. The ship is the object to be segmented. The segmentation started from the 100th frame, in which the ship totally appeared. It can be seen from Figure 4 that the segmentation results is satisfactory for all the following frames. We achieve these good results because the tracking point selection algorithm is used.

5. CONCLUSIONS

A semi-automatic video object segmentation technique basing on hierarchy optical flow is proposed in this paper. This technique comprises of spatial and temporal segmentation modules. In the spatial segmentation stage, a point-based graphic user interface (PBGUI) is presented. With PBGUI, it is very convenient for the user to input data and the spatial

segmentation results are very good. With the result of spatial segmentation, the temporal segmentation involves non-rigid object contour tracking and rigid object whole-tracking all by hierarchy optical flow algorithm based on the algorithm proposed by Lucas and Kanade. Specially, the tracking point selection algorithm is proposed to greatly improve the tracking performance in the rigid object whole-tracking. The experimental results show that the proposed algorithm can precisely segment video objects from video clips and can be applied to object-oriented coding, content-based functionality and multimedia database indexing. In this paper, the color information has not been used. The future work of this paper is how to use the color information to improve its performance.

6. REFERENCES

1. MPEG Video and SNHC Groups. "Committee draft of MPEG-4, part 2, 14496-2" Technical Report ISO/IEC JTC/SC29/WG11/N1902, ISO/IEC, Fribourg, Switzerland, October 1997.
2. T.Meier, K.N.Ngan, Automatic segmentation of movings for video object plane generation. IEEE Transactions on Circuits and Systems for Video Technology, 1998, 8(5):525-538.
3. D.Wang. Unsupervised video segmentation based on watersheds and temporal tracking. IEEE Transactions on Circuits and Systems for Video Technology, 1998, 8(5):539-546.
4. Chuang Gu, Ming-Chieh Lee. Semiautomatic segmentation and tracking of semantic video objects. IEEE Transactions on Circuits and Systems for Video Technology, 1998, 8(5):572-584.
5. J.G.Choi, M.Kim, M.H.Lee, C.Ahn. A user-assisted segmentation method for video object plane generation. Proc. ITC-CSCC'98, Sokcho, Korea, 998,7-10.
6. Munchurl Kim, J.G.Jeon, J.S.Kwak, M.H.Lee, C.Ahn. Moving object segmentation in video sequences by user interaction and automatic object tracking. Image and Vision computing, 2001, 19(5):245-260.
7. Kass M,Witkin A,Terzopoulous D. Snakes: active contour models. International Journal of Computer Vision,1988, 1(4):321-331.
8. Castleman K.R. Digital Image Processing. Englewood Cliffs, New Jersey: Prentice – Hall, 1996:460-461.
9. S.S. Beauchemin, J.L. Barron, D.J. Fleet. On Optical flow. Int. Conf. on Artificial Intelligence and Information-Control Systems of Robots. Bratislava, Slovakia, 1994, 3-14.
10. B.D.Lucas, T.Kanade. An iterative image registration technique with and application to stereo vision. Proceedings of the 7th International Joint Conference on Artificial Intelligence. Vancouver, 1981,674-679.
11. J. Shi, C. Tomasi. Good Features to Track.IEEE Conference on Computer Vision and Pattern Recognition. Seattle, 1994, 593--600.