VISUAL TRACKING SYSTEM USING A TEMPLATE MATCHING ALGORITHM

Davi Yoshinobu Kikuchi
University of São Paulo – Av. Prof. Mello Moraes, 2231, CEP: 05508-900, São Paulo, SP, Brazil
davi.kikuchi@poli.usp.br

Lucas Antonio Moscato
University of São Paulo – Av. Prof. Mello Moraes, 2231, CEP: 05508-900, São Paulo, SP, Brazil
lamoscat@usp.br

Abstract. *Visual tracking systems are responsible for tracking a moving object through data gained from camera images, using computer vision. The object position should be obtained from an image, allowing the camera to move and keep it inside the image. This work presents an algorithm for image analysis whose objective is to determinate the projected object position on the image, based on the template matching principle. This technique consists of defining an image pattern and searching it inside a new image to discover the position to where it moved, using the sum of squared differences criteria. The algorithm is also extended using incremental and multiresolution estimation techniques. Some possible aplications for this work is videoconference and monitoring cameras.*

Keywords: *visual tracking, computer vision, template matching*

1. Introduction

Computer vision is a technique that utilizes camera images as sensors, allowing to obtain informations from a dynamic system. However, since these information are not given directly, an image should be analysed and interpreted to provide values that can be measured. These data are usually the position of a moving object, which can be used as a reference point. Systems that should follow the object as it moves around are called tracking systems. Their function is to keep the tracked object projection on the center of the image, through camera motion by any kind of external actuator.

This work presents a visual tracking algorithm to be applied to a camera control system. The camera has two rotational degrees of freedom (pan and tilt) and zoom compensation; therefore, it is called PTZ camera. The objective of the algorithm is to determinate the projected position (horizontal and vertical) and size changes (scaling) of a tracked object in the image coordinates system, which the camera controller would receive and use to assign the PTZ displacements.

The algorithm supposes that the tracked object is not pre-defined. Thus, specific shape, geometry, or color informations are not accessible. The basic technique utilized is template matching, used in a large range of applications. It consists of defining a reference image, from which is obtained a window (sub-image) that contains the tracked object. This window corresponds to the reference template and the algorithm should estimate, on each acquired image, the position whose window is the most similar to the template. The criteria used to define the similarity between two windows is the sum of squared differences (SSD) of each correspondent pixels. The most similar window takes place at the position whose SSD value is minimal.

Papanikolopoulos, Khosla, and Kanade (1993) accomplished this technique through a simple search on the image. The algorithm calculates the SSD values (pixel by pixel) of windows at all possible positions. Its disadvantage is the high computational cost if the window size enlarges. Thus, Hager and Belhumeur (1998) presented a technique that models the tracked template movement through a parametric function, developed to provide calculus simplification and increase its execution speed.

Jurie and Dhome (2001) made also use of template matching through a movement model function. Its parameters are obtained through a learning stage, which uses known displacements and their respectives error values (SSD). After that, the algorithm can accomplish the tracking. A visual perception system for car driving applied this technique (Clady et al., 2001).

Odobez and Bouthemy (1995) presented a algorithm applied by Crétual and Chaumette (2000) to submarine visualization. This procedure consists of converting the acquired images to different resolutions and using robust estimators to improve the tracking efficiency.

Fayman, Sudarsky, and Rivlin (1998) developed a specific algorithm for zoom control. Its function is to compensate the size changes of the tracked object projection on the images, through two possibles techniques: using the displacement perspective projections (optical flow) or the camera autofocus sensor.

This work utilizes the sum of squared differences algorithm from Hager and Belhumeur (1998), due its low computational cost. Moreover, two techniques from Odobez and Bouthemy (1995) are incorpored, intending a performance improvement: incremental estimation and multiresolution estimation. The algorithms are tested on example images, which simulate some possible displacements.

2. Visual Tracking Algorithms

2.1 Least Squares Based Algorithm

Let an image window be defined as a *target region* $\Re = \{\boldsymbol{X_1}, \boldsymbol{X_2}, \boldsymbol{X_3}, ..., \boldsymbol{X_N}\}$, with $\boldsymbol{X_i} = [x_1, x_2]^T$ as a pixel position (coodinates system with origin at the image center, horizontal x_1-axis positive to right and vertical x_2-axis positive to up). Denoting the image obtained at the initial time (t_0) as the *reference image*, the pixel brightness values of the target region on the reference image is defined as the *reference template*. The tracked pattern is located inside the reference template.

Consider a parametric function $\boldsymbol{F}$, which models the tracked pattern movement:

$$\boldsymbol{F}(\boldsymbol{X_i}; \boldsymbol{\mu}(t_i)) = S.\boldsymbol{X_i} + \boldsymbol{U} \tag{1}$$

where:
$S(t_i)$ = scaling parameter
$\boldsymbol{U}(t_i) = [U_1(t_i), U_2(t_i)]^T$ = translational parameters (horizontal and vertical)
$\boldsymbol{\mu}(t_i) = [U_1, U_2, S]^T$ = parameters vector

The function can determinate the projected displacements on x_1 and x_2 directions, as well as the object scaling. Thus, the algorithm objective is to calculate the parameters vector value $\big(\boldsymbol{\mu}(t_i)\big)$, which gives the position $\boldsymbol{F}$, where the tracked pattern is located at time t_i.

Assuming $\boldsymbol{I}(\boldsymbol{X}, t_i)$ as the $N \times 1$ vector that contains the brightness values of all target region pixels in the image at time t_i, let $\boldsymbol{I}(\boldsymbol{\mu}, t_i)$ be the brightness vector with the target region pixels displaced by $\boldsymbol{F}(\boldsymbol{X_i}; \boldsymbol{\mu}(t_i))$. Thus, the reference template is $\boldsymbol{I}(\boldsymbol{\mu}(t_0), t_0)$.

Consider now an incremental calculus:

$$\boldsymbol{\mu}(t_i) = \boldsymbol{\mu}(t_{i-1}) + \boldsymbol{\delta\mu}(t_i) \tag{2}$$

As there are no displacements at t_0, $\boldsymbol{F}(\boldsymbol{X_i}; \boldsymbol{\mu}(t_0)) = \boldsymbol{X_i}$ and $\boldsymbol{\mu}(t_0) = [0, 0, 1]^T$. Therefore, the $\boldsymbol{\delta\mu}(t_i)$ values need to be determined, allowing the estimation of $\boldsymbol{\mu}$ at each time. Through the sum of squared differences criteria, the objetive function O is defined:

$$O(\boldsymbol{\delta\mu}) = \|\boldsymbol{I}(\boldsymbol{\mu}(t_i), t_i) - \boldsymbol{I}(\boldsymbol{\mu}(t_0), t_0)\|^2 \tag{3}$$

Using least squares estimation, the function is minimized by (Hager and Belhumeur, 1998):

$$\underset{(3\times1)}{\boldsymbol{\delta\mu}} = -(\boldsymbol{M}^T\boldsymbol{M})^{-1}\boldsymbol{M}^T[\boldsymbol{I}(\boldsymbol{\mu}, t_i) - \boldsymbol{I}(t_0)] \tag{4}$$

where: $\underset{(N\times3)}{\boldsymbol{M}} = \boldsymbol{\nabla_\mu I} = \left[\dfrac{\partial \boldsymbol{I}}{\partial U_1} \;\middle|\; \dfrac{\partial \boldsymbol{I}}{\partial U_2} \;\middle|\; \dfrac{\partial \boldsymbol{I}}{\partial S}\right]$

Using Hager and Belhumeur's (1998) procedure, the matrix M can be calculated by:

$$\boldsymbol{M} = \frac{1}{S}\boldsymbol{M_0} = \frac{1}{S}\left[\frac{\partial \boldsymbol{I}(t_0)}{\partial x_1} \;\middle|\; \frac{\partial \boldsymbol{I}(t_0)}{\partial x_2} \;\middle|\; \frac{\partial \boldsymbol{I}(t_0)}{\partial x_1}x_1 + \frac{\partial \boldsymbol{I}(t_0)}{\partial x_2}x_2\right] \tag{5}$$

The spatial derivates applied to the reference template can be obtained using Sobel convolution masks (Young, Gerbrands, and van Vliet):

$$\frac{\partial \boldsymbol{I}}{\partial x_1}(x_1, x_2) = \frac{1}{8}[\boldsymbol{I}(x_1{+}1, x_2{+}1) - \boldsymbol{I}(x_1{-}1, x_2{+}1) + 2\boldsymbol{I}(x_1{+}1, x_2) - 2\boldsymbol{I}(x_1{-}1, x_2) + \boldsymbol{I}(x_1{+}1, x_2{-}1) - \\ -\boldsymbol{I}(x_1{-}1, x_2{-}1)] \tag{6}$$

$$\frac{\partial \boldsymbol{I}}{\partial x_2}(x_1, x_2) = \frac{1}{8}[\boldsymbol{I}(x_1{+}1, x_2{+}1) - \boldsymbol{I}(x_1{+}1, x_2{-}1) + 2\boldsymbol{I}(x_1, x_2{+}1) - 2\boldsymbol{I}(x_1, x_2{-}1) + \boldsymbol{I}(x_1{-}1, x_2{+}1) - \\ -\boldsymbol{I}(x_1{-}1, x_2{-}1)] \tag{7}$$

Substituting Eq. (5) in Eq. (4):

$$\boldsymbol{\mu}(t_i) = \boldsymbol{\mu}(t_{i-1}) - (\boldsymbol{M_0}^T\boldsymbol{M_0})^{-1}\boldsymbol{M_0}^T[\boldsymbol{I}(\boldsymbol{\mu}, t_i) - \boldsymbol{I}(t_0)]S(t_{i-1}) \tag{8}$$

Fig. 1 contains a summary of the procedure.

Offline:

- Store $\boldsymbol{I}(t_0)$
- Calculate $\boldsymbol{\nabla_X I}(t_0)$ using Eq. (6) and (7)
- Calculate $\boldsymbol{M_0} = \left[\frac{\partial \boldsymbol{I}(t_0)}{\partial x_1} \;\middle|\; \frac{\partial \boldsymbol{I}(t_0)}{\partial x_2} \;\middle|\; \frac{\partial \boldsymbol{I}(t_0)}{\partial x_1} x_1 + \frac{\partial \boldsymbol{I}(t_0)}{\partial x_2} x_2 \right]$
- Store $\boldsymbol{\Lambda} = (\boldsymbol{M_0}^T \boldsymbol{M_0})^{-1} \boldsymbol{M_0}^T$
- Initialize $\boldsymbol{\mu}(t_0) = \begin{bmatrix} U_1(t_0) \\ U_2(t_0) \\ S(t_0) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$

Online:

FOR $i = 1, 2, 3 \ldots$

$$\boldsymbol{\mu}(t_i) = \boldsymbol{\mu}(t_{i-1}) - \boldsymbol{\Lambda}[\boldsymbol{I}(\boldsymbol{\mu}(t_{i-1}), t_i) - \boldsymbol{I}(t_0)] S(t_{i-1})$$

END

Figure 1. Least squares algorithm scheme.

2.2 Extended Algorithm by Incremental Estimation

The previous technnique is extended through a incremental step, as the one used by Odobez and Bouthemy (1995). The approximations applied to the least squares based algorithm make it accurate only for small displacements. However, assuming that $\boldsymbol{\delta\mu}$ does not give the exact $\boldsymbol{\mu}$ value but provide a **closer** estimate, another application of the algorithm can give a still more satisfactory estimate (because the relative displacements decrease). Therefore, this algorithm extension consists of iterating the previous one for the same acquired image, until $\boldsymbol{\delta\mu}$ becomes smaller than a pre-defined error value or a determined number of iterations is reached.

With respect to the parameter S, each value update would need a pixels transform on the image (scaling), which consumes a considerable time. Thus, it is considered that scaling displacements are small and S is not iterated. Then the image do not need to be transformed and the target region is deslocated at each iteration only by U_1 and U_2.

The algorithm summary can be found in Fig. 2.

Apply offline initializations from the previous algorithm
Define $n\_iterations$ and $error$

FOR $i = 1, 2, 3 \ldots$

$\boldsymbol{\mu}(t_i) = \boldsymbol{\mu}(t_{i-1})$

$iteration = 0$

DO

$$\boldsymbol{\delta\mu} = -\boldsymbol{\Lambda}[\boldsymbol{I}([U_1(t_i), U_2(t_i), S(t_{i-1})]^T, t_i) - \boldsymbol{I}(t_0)] S(t_{i-1})$$

$$\boldsymbol{\mu}(t_i) = \boldsymbol{\mu}(t_i) + \boldsymbol{\delta\mu}$$

$$iteration = iteration + 1$$

WHILE $(iteration < n\_iterations)$ OR (max($\boldsymbol{\delta\mu}$) $> error$)

END

Figure 2. Scheme of the extended algorithm by incremental estimation.

2.3 Extended Algorithm by Multiresolution Estimation

This technique is also extracted from Odobez and Bouthemy (1995). If the tracked object displacement is huge and the template goes outside the target region, the tracking is not possible. But if the image is acquired at a lesser resolution and the target region size is constant, the **relative** size of the target region increases (see Fig. 3).

Thus, the algorithm can track in a larger area of images with smaller resolutions. However, lower resolutions bring also lower number of pixels representing the tracked object. As the pattern knowledge decreases, also does the algorithm precision. Therefore, the technique is applied to three resolution levels: original, 1:2 and 1:4. The incremental estimation algorithm is first used on the 1:4 image (smaller resolution), then on the 1:2 and at last on the original image. At lower resolutions, the tracking of higher displacements can be improved. And at higher resolutions the displacements are already

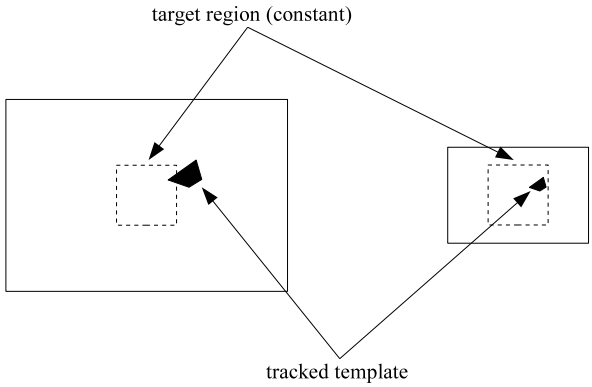

Figure 3. Multiresolution principle estimation example.

smaller and the accuracy is increased.

Besides, when the algorithm shifts from a lower to a higher image resolution, U_1 and U_2 must be multiplied by 2, because the resolution doubles. The same happens from the original to the 1:4 resolution: they must be divided by 4. The scaling parameter S is not directed altered by resolution changes and is updated only in the end, as the incremental algorithm.

In Fig. 4 there is the algorithm summary. The half resolution images are obtained splitting up each four pixels neighbourhoods from the original one. The pixels from the lower resolution image are the mean of each four pixels block.

Offline initiations from the previous algorithm for the three resolutions (the "1", "2" and "3" subscripts refer to 1:4, 1:2 and original resolution, respectively):

- Store $\boldsymbol{I_1}(t_0)$, $\boldsymbol{I_2}(t_0)$ e $\boldsymbol{I_3}(t_0)$
- Calculate and store the matrix $\boldsymbol{\Lambda_k}$, using $\boldsymbol{I_k}(t_0)$, $k = 1, 2, 3$
- Initializate $\boldsymbol{\mu}(t_0) = [0, 0, 1]^T$
- Define $n\_iterations$ and $error$

FOR $i = 1, 2, 3 \ldots$

$\boldsymbol{\mu}(t_i) = \dfrac{\boldsymbol{\mu}(t_{i-1})}{4}$

FOR $k = 1$ to 3

$iteration = 0$

DO

$\boldsymbol{\delta\mu} = -\boldsymbol{\Lambda_k}[\boldsymbol{I_k}([U_1(t_i), U_2(t_i), S(t_{i-1})]^T, t_i) - \boldsymbol{I_k}(t_0)]S(t_{i-1})$

$\boldsymbol{\mu}(t_i) = \boldsymbol{\mu}(t_i) + \boldsymbol{\delta\mu}$

$iteration = iteration + 1$

WHILE $(iteration < n\_iterations)$ OR (max$(\boldsymbol{\delta\mu}) > error$)

IF $k < 3$ THEN $U_1(t_i) = 2U_1(t_i)$ and $U_2(t_i) = 2U_2(t_i)$

END

END

Figure 4. Scheme of the extended algorithm by multiresolution estimation.

3. Experiment Description

This work utilizes a AXIS 2130 PTZ Network Camera, with IP communication, and position and velocity control for pan–tilt angles and zoom. The development of a control software is accomplished on Microsoft Visual C++ 6.0 environment, including an image analysis stage that contains the previous three algorithms. Each algorithm is applied to two groups of some grayscaled image examples and their results are compared. The first images group show a created

feature, which was produced only for testing (Fig. 5). The second one are real images that were obtained from the camera, for practical situations simulation (Fig. 6). The first images of Fig. 5 and 6 correspond to the reference image and the others the previous one after some displacements. The software runs in a Pentium 3 – 800 Mhz computer, with Microsoft Windows 2000 and 160 MB RAM.

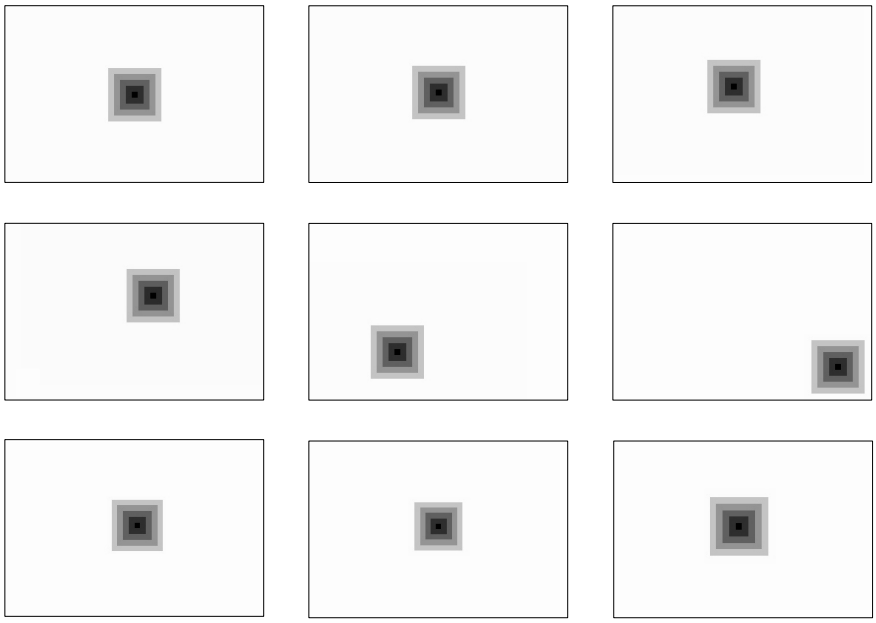

Figure 5. First group of image examples for the tracking algorithms.

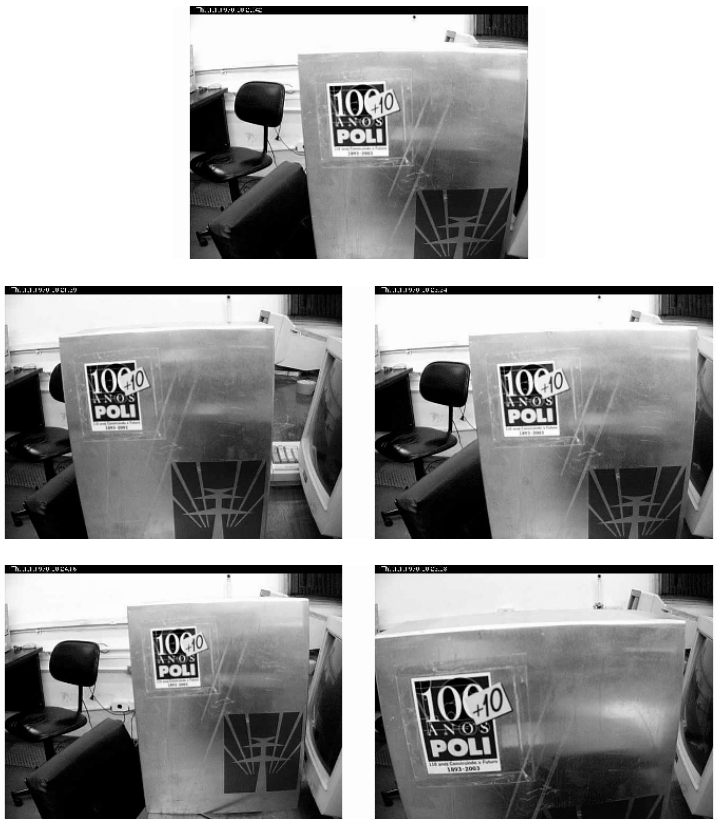

Figure 6. Second group of image examples for the tracking algorithms.

The original resolution of the example images is 352×240 pixels. The utilized target region is a 50×50 pixels window centralized at the image center. The $n\_iterations$ value is 50 and $error$ value is 0,01.

4. Results

Fig. 7 to 12 contain the experiment results. The "real" displacements are approximated values, because they were manually obtained by image examination. The algorithms 1, 2, and 3 corresponds to the least squares, the extension by incremental estimation, and the extension by multiresolution estimation, respectively.

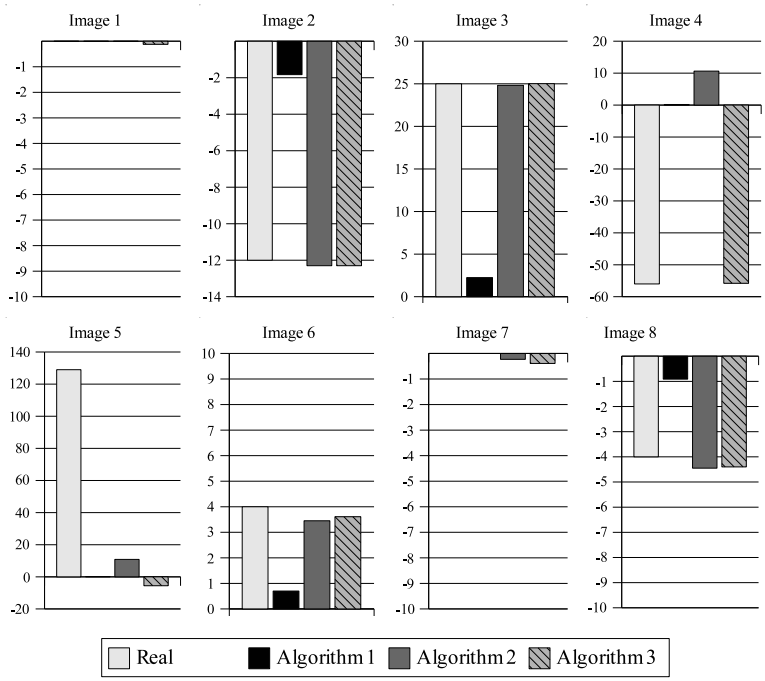

Figure 7. Experimental x_1 (U_1) displacement results for the first image examples group.

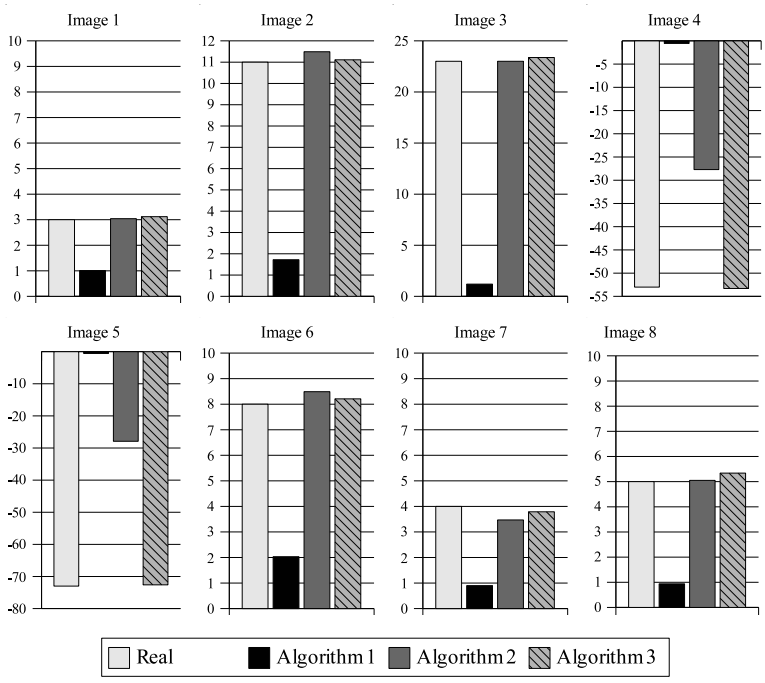

Figure 8. Experimental x_2 (U_2) displacement results for the first image examples group.

Fig. 7 to 9 show that the first algorithm produces bad results for the first image group, even for not so large displacements. The incremental estimation technique can track the feature at averaged distant positions and the multiresolution algorithm covers the highest range. The scaled images (last three ones) produces similar results for x_1 and x_2 displacements, but the scale factors are not so accurate.

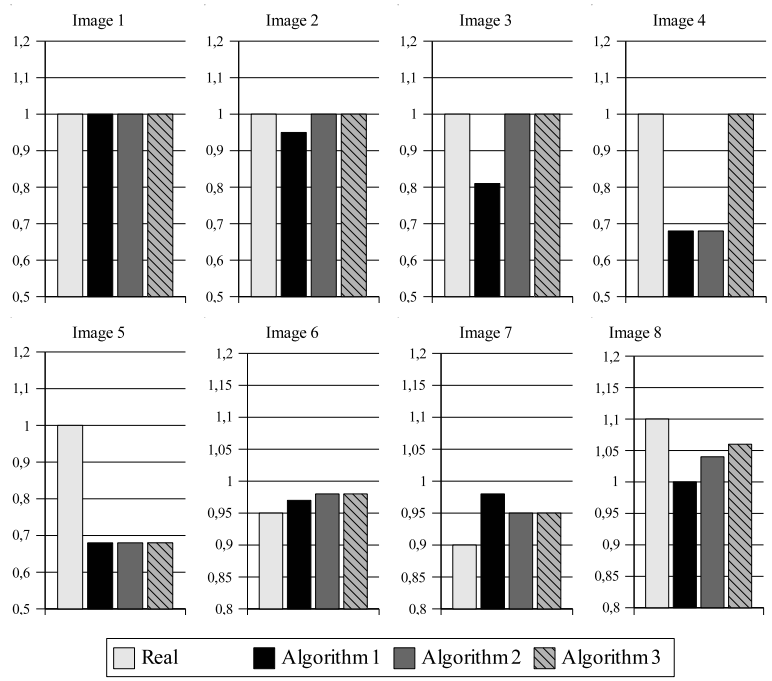

Figure 9. Experimental scaling (S) displacement results for the first image examples group.

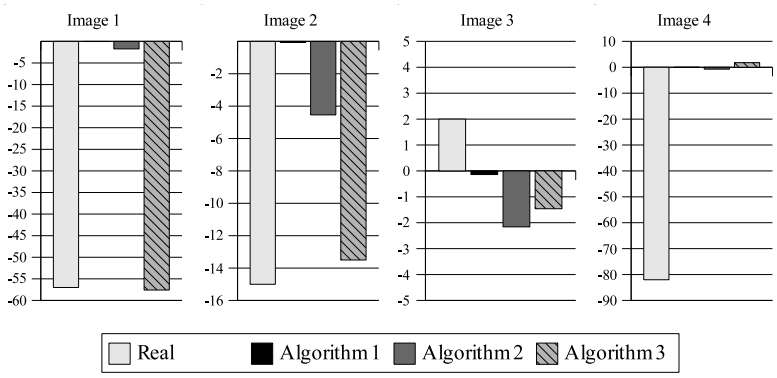

Figure 10. Experimental x_1 (U_1) displacement results for the second image examples group.

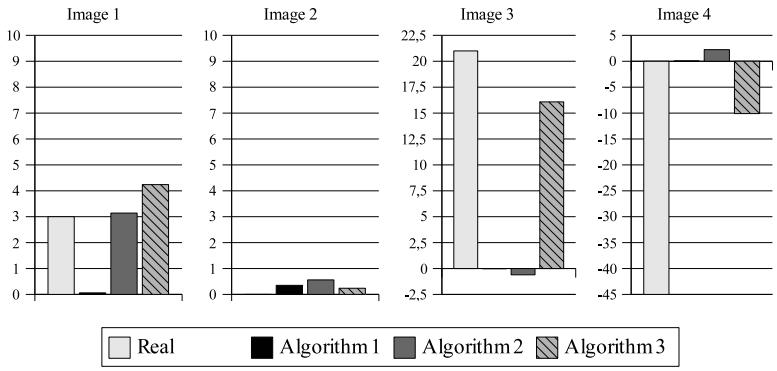

Figure 11. Experimental x_2 (U_2) displacement results for the second image examples group.

The second image group shows similar results. The first algorithm does not work properly and the third one produces the best results. Scaling displacements obtained are also not so precise, and for larger values (last two images) the results are very poor.

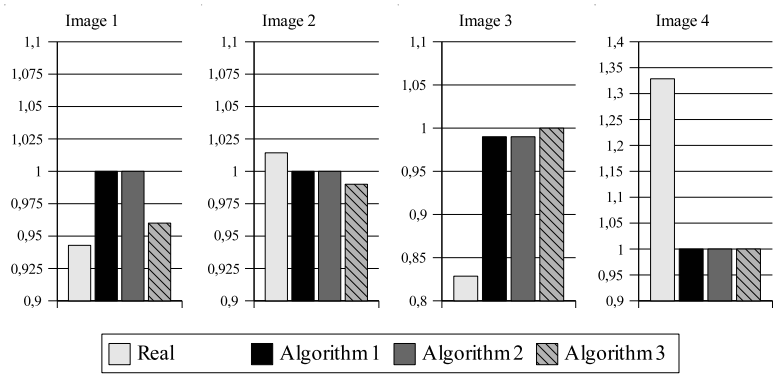

Figure 12. Experimental scaling (S) displacement results for the second image examples group.

5. Conclusions

This work presented three visual tracking algorithms based on template matching. The least squares based algorithm produced poor results, but contains the basic principle utilized by the others techniques. The incremental estimation extension tracked a template at considerable distant positions and the multiresolution estimation extension covered the largest area inside an image. The scaling displacement tracking from the first algorithm is the same used by the others. Therefore, it is quite limited to small variations.

Though the multiresolution technique has presented the best performance, its application in the camera control system maybe do not show the best results, due its computational cost. The processing time of the incremental algorithm is nearly 70 ms and of the multiresolution algorithm is 120 ms. If the processing time is reduced, the delay between the image acquisition and the displacements calculation is also reduced. Thus, the camera motion can be sooner adjusted, improving the tracking accuracy. Moreover, for a higher processing time a higher sample period should be used, and the control system becomes slower. Therefore, an analysis for the best performance can be only achieved by the camera control system application.

The next step in the work is to use the obtained tracked object displacements to feed a control system, allowing the camera motion. Then the application of proper camera models and control strategies with the previous visual tracking algorithms can give a whole robotic visual tracking system.

6. References

Clady, X., Collange, F, Jurie, F., and Martinet, P., 2001, "Object Tracking with a Pan-Tilt-Zoom Camera: Application to Car Driving Assistance", International Conference on Robotics and Automation, pp. 1653-1658.

Crétual, A. and Chaumette, F., 2000, "Dynamic Stabilization of a Pan and Tilt Camera for Submarine Image Visualization", Computer Vision and Image Understanding, Vol.79, No.1, pp. 47-65.

Fayman, J. A., Sudarsky, O., and Rivlin, E., 1998, "Zoom Tracking", International Conference on Robotics and Automation, pp. 2783-2788.

Hager, G. D. and Belhumeur, P. N., 1998, "Efficient Region Tracking With Parametric Models of Geometry and Illumination", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol.20, No.10, pp. 1025-1039.

Jurie, F. and Dhome, M., 2001, "A Simple and Efficient Template Matching Algorithm", International Conference on Computer Vision, Vol.2, pp. 544-549.

Odobez, J.M. and Bouthemy, P., 1995, "Robust Multiresolution Estimation of Parametric Motion Models", International Journal of Visual Communication and Image Representation, Vol.6, No.4, pp. 348-365.

Papanikolopoulos, N., Khosla, P. K., and Kanade, T., 1993, "Visual Tracking of a Moving Target by a Camera Mounted on a Robot: A Combination of Control and Vision", IEEE Transactions on Robotics and Automation, Vol.9, No.1, pp. 14-35.

Young, I. T., Gerbrands, J. J., and van Vliet, L. J., "Image Processing Fundamentals", http://www.ph.tn.tudelft.nl/Courses/FIP/noframes/fip.html.

7. Responsibility notice

The authors are the only responsible for the printed material included in this paper