Contour/Texture Approach for Visual Tracking

Lucie Masson, Frédéric Jurie, and Michel Dhome

LASMEA - CNRS UMR 6602 - Université Blaise-Pascal
Campus Scientifique des Cézeaux 63170 Aubière
masson,jurie,dhome@lasmea.univ-bpclermont.fr

Abstract. In this article, the problem of real-time hybrid *contour/texture tracking* for planar objects is addressed. On one hand, a lot of methods have been proposed to track objects from their contours. On the other hand, numerous other tracking algorithms deal with texture. In real situations, objects can unfortunately rarely be divided so clearly. Therefore, an hybrid tracking approach, able to mix texture and contour information, appears to be very useful.
The proposed approach is very simple and efficient. It is based on *image differences*, which are the differences between object aspects in the image and aspects predicted using a parametric transformation model. Knowing a difference image, the proposed algorithm only need a matrix multiplication to estimate motion parameters. This is possible due to the use of an off-line learning stage.

1 Introduction

In a general point of view, template tracking algorithms can be seen as optimization algorithms. They compute the estimate of a state vector in order to explain as well as possible the appearance of an object in an image. The state parameters can consist in information about the position, the orientation or the speed of the object. They may also describe aspect modifications or change in the object properties.

These algorithms can be classified in two categories, depending on the nature of the function to be optimized. In the first one, distances between image primitives (points, lines, b-splines, *etc.*) and model primitives are minimized. In the second one, resemblance is calculated between the global intensity function of the target template and the one observed in the current image.

Let us now illustrate these two classes of algorithms by presenting some relevant works.
Tracking based on distance minimization. To illustrate this class of methods, we may mention the work of Lowe [11], consisting in tracking a 3D model in an image sequence. The goal is to find the object pose which minimizes the distance between line segments of the 3d model projected in image and image line segments (maximization of correspondence probability between primitives).

More recently, Kollnig and Nagel [9] propose to track vehicles using 3d models. The originality of their approach comes from the use of gradient information instead of detecting edges. The 3D pose of the model is optimized in order to maximize the probability for a model point to be projected onto an image edge point (high gradient).

Drummond and Cipolla [2] propose a real-time 3D objects tracking system using contours. They proposed to use an hardware real-time rendering system. It can compute very quickly the aspect of a complex object for a given pose. The function to be minimized is the distance between the projected points and the image edges.

Tracking based on global intensity model. The works based on *template matching* belong to this category. The work of La Cascia *et al.* [10] on human face tracking has to be mentioned. Face texture is mapped on a 3d cylinder, and its pose is optimized to minimize the Euclidean norm of the difference between the human face texture in image and texture of the projection of the cylinder.

Hager *et al.* [3] have developed an approach allowing to track textures in real-time. It was first limited to planar objects and small motions but it was extended by Jurie and Dhome to large motions [8] and 3d surfaces [6].

Jepson *et al.* [5] propose another approach in which the model of the target template is updated on-line. The model of the motif is obtained by applying an on-line version of the EM algorithm. This algorithm is used to model the responses of multi-scaled orientable filters applied in every points of the template.

With the *Mean Shift* algorithm [1], the object is modeled by means of a color histogram. The tracking algorithm moves an analyze window to match – as well as possible – the histogram of the target with the one in the analyzed window.

The main contribution of this article is to propose an hybrid method able to mix informations about contour distances and textures. It can automatically take benefits of the different visual characteristics of the tracked template: a given visual template will be better described using distances to edges, while another one will be better described using texture. We propose here to use a mixed criterion. The method described in this article in an extension of Jurie and Dhome's works [7].

This article is made of three parts. Section 2 shows the bases of the proposed approach. The two phases of our method, learning and tracking, are then developed in section 3. At last, we present some experimental results that prove the validity of our approach.

2 Main concepts

2.1 Template representation

We will first see how templates are stored in our method.

We suppose that the tracked planar template is made of N points $(p_1, p_2, \ldots, p_N)$, where $p_i = (x, y)$, included in the $\mathbf{P}$ vector. These coordinates are expressed in a reference frame linked to the template (different from the image reference frame), and are supposed to be constants. These p_i points are selected in order to respect two constraints: being well spread on the template and being placed on image spots where spatial gradient is high.

Figure 1 explains these concepts of template reference frame and image reference frame. In fact, tracking a template is equivalent to estimating the transformation from the template reference frame to the current image reference frame.

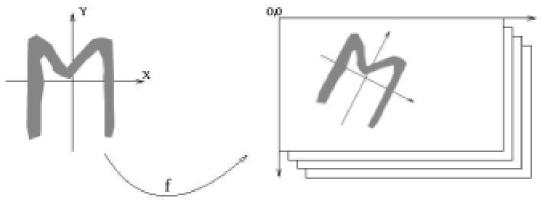

Fig. 1. From reference frame to image coordinates systems

In practice, the template is split in buckets and a minimal number of points is chosen in each bucket. Selection is done by drawing points randomly in the buckets and selecting the ones having the highest gradients.

Each selected point is then classified as edge point or texture point. Classification is done with the following criterion: let x and y be the coordinates of point p in the image, and $I(x, y) = I(p)$ be the image intensity at this point. We define $\mathbf{M}$ as being:

$$\mathbf{M} = \begin{pmatrix} \sum_{p \in w} \frac{\partial I(p)}{\partial x}^2 & \sum_{w} \frac{\partial I(p)}{\partial x} \frac{\partial I(p)}{\partial y} \\ \sum_{p \in w} \frac{\partial I(p)}{\partial x} \frac{\partial I(p)}{\partial y} & \sum_{w} \frac{\partial I(p)}{\partial y}^2 \end{pmatrix} \tag{1}$$

where w is a square window centered on the point. Let α and β be the two eigenvalues of M. Point p is considered as an edge point if $\alpha + \beta$ is great and if simultaneously $\alpha * \beta$ is low. In other cases, it is considered as being a texture point. It assumes that in case of contour points the autocorrelation function of the surface is curved in only one direction. This criterion is very close to the one proposed by Harris and Stephens [4] for corners detection.

At the end of this stage, each point p_i of the template is classified as being an edge point or a texture point, written respectively $C(p_i) = 1$ and $C(p_i) = 0$.

Template is characterized by a shape vector called $\mathbf{V} = (v_1, \ldots, v_N)$ where v_i are defined by:

$$v_i = V(p_i) = \begin{cases} I(p_i, t) & \text{if } C(v_i) = 0 \\ DI(p_i, t, nx, ny) & \text{if } C(v_i) = 1 \end{cases} \tag{2}$$

where $I(p_i, t)$ represents the gray level value of the image at time t.

$DI(p_i, t, nx, ny)$ represents the distance from the point to the closest edge, measured in image I at time t. Practically, for each edge point p_i, we have saved a second point n_i, and these two points define a direction perpendicular to the edge. If we have $\mathbf{N} = (n_1, \ldots, n_N)$, and if we call $\mathbf{F}$ the matrix describing the transformation applied to the template (see 2.2), the new direction is obtained by applying $\mathbf{F}$ on both $\mathbf{N}$ and $\mathbf{P}$, and subtracting their coordinates. We measure the distance d to the nearest edge in the image, which we take as the point with the highest well signed gradient on the considered direction, moved along step by step. We obtain the distance D in the model coordinates system by computing $D = \frac{d}{\|\mathbf{n}'\|}$. We denote $\mathbf{V}$ as $V(p_i, t)$ to show the link of the shape vector with time and the template vector.

2.2 Motion representation

We now have to model the template motion in the image.

Let $\mu(t) = (\mu_1(t), \mu_2(t), \ldots, \mu_n(t))$ be the parameters vector describing the template position at time t. These parameters are the parameters of a function f that brings points from the *model* reference to the *image* one (see figure 1). This function f, describing the template position, explains how a point of the model is mapped in the image.

$$p' = f(p, \mu(t)) \tag{3}$$

This expression means that point p comes to p' if we apply the transformation f with parameter μ. f can represent simple motions like planar translations, 3D motions (homography, for example), or motions that imply deformations . We also write $V(\mu(t), t) = V(f(\mathbf{P}, \mu(t)))$ where $f(\mathbf{P}, \mu(t)) = (f(p_1, \mu(t)), \ldots, f(p_N, \mu(t)))$. We suppose that V is differentiable in time.

We will call *object motion* the variation of the motion parameters between two images, and we will denote it $\delta\mu(t)$. By definition we have $\delta\mu(t) = \mu(t) - \mu(t-1)$. We suppose that motion is linear in the projective 2D space. In this case, will be able to represent motions like 2D translations, planar rotations, affinities or homographies by linear relationships.

Transformations will be expressed by matrix products, after having written points in homogeneous coordinates. We will write this matrix $\mathbf{F}(\mu(t))$. Relation (3) can be written

$$p' = \mathbf{F}(\mu(t)) \times p \tag{4}$$

where p and p' are written with homogeneous coordinates.

3 Learning and tracking stages

3.1 Theory

Using previous definitions, we propose a tracking method based on the minimization of a SSD-like criterion. Tracking an object consists in giving at each time t its position $\mu(t)$, such that $\mu(t)$ minimizes the criterion:

$$\epsilon(t) = \| V(\mu_0, t_0) - V(\mu(t), t) \|, \tag{5}$$

where $\mu_0 = \mu(t_0)$ is the position of the template in first image of the sequence. We suppose that μ_0 is known at the first image of the sequence.

We are looking for an iterative method that gives state $\mu(t)$ knowing state $\mu(t-1)$ and image at time t. The main principle is to consider that in the neighborhood of a position of the tracking contour, the relation between motion and changes in shape vector is linear.

If we suppose that the state at time $t-1$ is known, meaning that $\epsilon_{t-1} = 0$,

$$V(\mu_0, t_0) = V(\mu(t-1), t-1), \tag{6}$$

V is a function with respect to t and μ ; we can write its first order Taylor expansion as:

$$V(\mu(t-1) + \delta\mu, t-1+\delta t) = V(\mu(t), t)$$

$$= V(\mu(t-1), t-1) + (\mu(t) - \mu(t-1))V_\mu(t) + \delta t V_t(t)$$

where $V_\mu(t)$ represents the derivative of V with respect to μ and $V_t(t)$ its derivative with respect to time. These two derivatives are computed at time t. We suppose that time between two definitions is by definition 1. If we suppose that

$$V_t(t) = -V(\mu(t-1), t-1) + V(\mu(t-1), t)$$

denoted more simply $\delta V(t)$, and if we consider that we will get the minimum for $\bigtriangledown\epsilon(t) = 0$ in (5), and if we use (6), we obtain:

$$0 = V(\mu(t-1), t-1) - V(\mu(t), t) = (\mu(t) - \mu(t-1))V_\mu(t) - \delta V(t)$$

This can also be written:

$$\mu(t) = \mu(t-1) + V_\mu^{-1}(t)\delta V(t), \text{or} \quad \delta\mu(t) = V_\mu^{-1}(t)\delta V(t)$$

where $\delta\mu(t)$ represents object motion at time t, and $V_\mu^{-1}(t)$ represents the inverse of the jacobian matrix. With this relation we can obtain the object motion knowing a variation of the shape vector. The Jacobian matrix $V_\mu(t)$ gives the variations of the shape vector parameters knowing an object motion $\delta\mu$. This is a $N \times n$ matrix where n is the number of parameters describing the motion and N the shape vector dimension. Jacobian matrix is consequently not invertible. We showed in [7], and it is an important contribution of our approach, that using pseudo-inverse $V_\mu^\dagger(t)$ with equation:

$$\delta\mu(t) = V_\mu^\dagger(t)\delta V(t), \tag{7}$$

gives worse results than those obtained by computing **H** such:

$$\delta\mu(t) = H(t)\delta V(t), \tag{8}$$

with direct learning.

3.2 Learning

As we have seen it in the preceding section, we have to avoid computing $\mathbf{H}(t)$ for each new image. Using relation (8) in the model reference frame instead of the image one, we can assume that $\mathbf{H}$ is constant. This is why, in the tracking stage, we have to bring the image in the model reference. Therefore f have to be invertible.

We will compute matrix $\mathbf{H}$ in the neighborhood of the *identity* function. We make N_{Pert} random perturbations $\delta\mu^i$ and then we get N_{Pert} corresponding values of δV^i. Motions $\delta\mu^i$ are stored in matrix $\mathbf{DMU}_{N_{Pert} \times n}$, and shape vectors in matrix $\mathbf{DI}_{N_{Pert} \times N_{Pt}}$.

Then we only have to compute the "best" matrix $\mathbf{H}$, which is the one for which

$$\sum_{i=1}^{i \leq N_{Pert}} (\delta\mu^i - \mathbf{H} \times \delta V^i)^2$$

is minimal. This least square minimization can be done using the relation: $\mathbf{H} = \mathbf{DMU}.\mathbf{DI}^{\dagger}$ where $\mathbf{DI}^{\dagger}$ is the pseudo-inverse of $\mathbf{DI}$.

The result of the learning stage, $\mathbf{H}$, is a matrix that will be used to find motions corresponding to shape vector differences (difference of texture and distance to edge).

3.3 Tracking

During the tracking stage, we do not really use a prediction stage. We suppose that the target position will be the same than in the last image. It may be considered true if the template motion is small. In fact, the convergence area of the proposed algorithm is large enough to track the template even with large displacements.

We build the shape vector difference δV in the model reference as seen previously (section 2.1), taking into account whether it is a texture point or an edge point. Then we can obtain $\delta\mu$, which is an estimation of the motion parameters, in the model reference, using $\delta\mu = \mathbf{H}.\delta V$. The motion is at last given in the image reference, using $f(\mu(t)) = f(\mu(t-1)) \circ f(\delta\mu(t))$. f is modeled by a linear transformation in the projective plane, $f(\mu(t))$, and written with matrix $\mathbf{F}(\mu(t))$, estimated as:

$$\mathbf{F}(\mu(t)) = \mathbf{F}(\mu(t-1)) \circ \mathbf{F}(\delta\mu(t))$$

It is easy to understand that the complexity of this algorithm is rather low: only one matrix multiplication and some simple mathematical operations (additions, multiplications, *etc.*).

4 Experimental results

We have implemented this algorithm on a personal computer, with a numeric SONY video camera giving images in real-time with a FireWire (IEEE1394) bus.

Templates are described with about one hundred of points. The transformation is an homography of the projective plane. Figure 2 shows images taken from real-time processing. Figure 3 shows the contribution of the method compared to

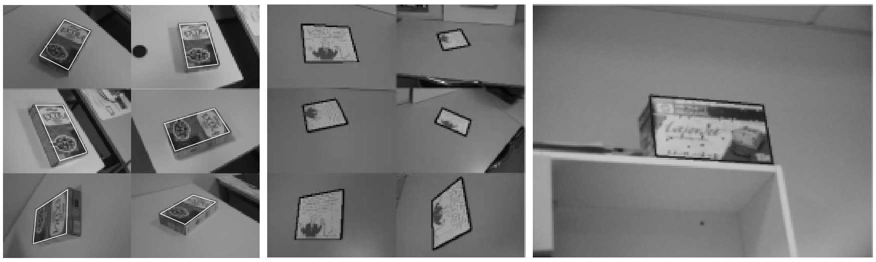

Fig. 2. Examples of textured template tracking

a classical tracking. The tracked template is a paperboard box with bad textures but with some strong edges (figure 2, right). This template is hard to track because it owns some areas that look similar for a classical algorithm. Convergence area is observed for planar translations up to 30% of the template size. Graphs represent error between estimated and real movement. Tracking with texture only is a bit weak (left), when the one with contour only is bad (right). We obtain the best result with the hybrid method (bottom), even if the convergence area contains discontinuities due to edges tracking. These discontinuities come from the use of steps during the distance computation.

In fact, this method allows us to enlarge the convergence area in the case of bad textured templates. The convergence area contains holes, that means that for some movements in the area tracker would be wrong, but globally the system would be more stable than without the contour component.

5 Conclusions

In this paper, we have presented a new hybrid tracking approach, taking advantages of texture and contour informations.

The main idea consists in describing templates with a limited number of points. These points are chosen in order to be equally distributed in the template, and lie on places where intensity variation is important. These points are classified as edge points or texture points. A shape vector that contains template intensity (for texture points) and distance to the nearest edge (for edge points) is then computed. When the template moves, a difference appears in the shape vector. This difference is used to deduce the object motion.

This method, based on an algorithm well known for its efficiency and robustness, have a larger area of convergence in the case of bad textured templates.

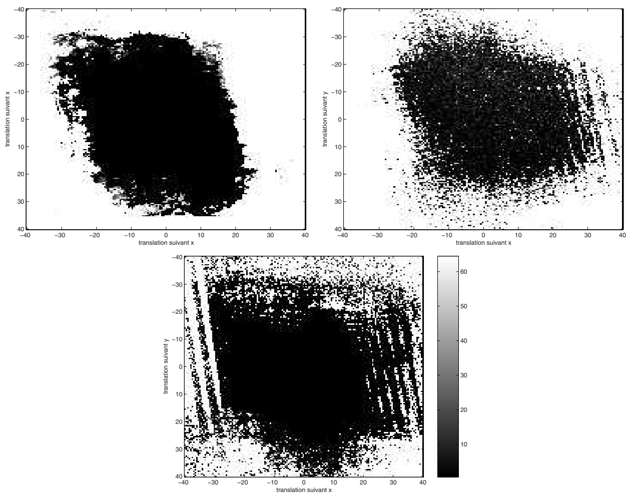

Fig. 3. Convergence area for planar translations: (left) only textures, (right) only contour points, (bottom) hybrid method

References

1. D. Comaniciu, V. Ramesh, and P. Meer. Real-time tracking of non-rigid objects using mean shift. In *CVPR00*, pages II: 142–149, 2000.
2. T. Drummond and R. Cipolla. Real-time tracking of complex structures with on-line camera calibration. *IVC*, 20(5-6):427–433, March 2002.
3. G.D. Hager and P.N. Belhumeur. Efficient region tracking with parametric models of geometry and illumination. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 20(10):1025–1039, October 1998.
4. C. Harris and M. Stephens. A combined corner and edge detector. In *4th Alvey Vision Conference*, pages 147–151, Mancherster, 1988.
5. A. D. Jepson, D.J. Fleet, and T.F. El-Maraghi. Robust on-line appearance models for visual tracking. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages pp. 415–422, 2001.
6. F. Jurie and M. Dhome. Real time 3d template matching. In *Computer Vision and Pattern Recongition*, pages (I)791–797, Hawai, December 2001.
7. F. Jurie and M. Dhome. Real time template matching. In *Proc. IEEE International Conference on Computer vision*, pages 544–549, Vancouver, Canada, July 2001.
8. F. Jurie and M. Dhome. Hyperplane approximation for template matching. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 24(7):996–1000, 2002.
9. H. Kollnig and H.H. Nagel. 3d pose estimation by directly matching polyhedral models to gray value gradients. *International Journal of Computer Vision*, 23(3):283–302, 1997.
10. M. La Cascia, S. Sclaroff, and V. Athitsos. Fast, reliable head tracking under varying illumination: An approach based on registration of textured-mapped 3d models. *PAMI*, 22(4):322–336, April 2000.
11. D.G. Lowe. Robust model-based motion tracking through the integration of search and estimation. *International Journal of Computer Vision*, 8(2):113–122, 1992.