Proceedings of the 2003 IEEE/RSJ
Intl. Conference on Intelligent Robots and Systems
Las Vegas, Nevada · October 2003

Collaborative tracking for robotic tasks

Antonio Marín-Hernández [⊗], Víctor Ayala-Ramírez[⊕] and Michel Devy[⊗]

[⊗]LAAS-CNRS
7, Av. du Col. Roche
31077 Toulouse CEDEX 4
France

[⊕]Universidad de Guanajuato -FIMEE
Tampico No. 912
36730 Salamanca, Gto.
Mexico

***Abstract*— This paper describes how to integrate tracking functions on a mobile robot. Tracking is required during autonomous robot navigation in different situations: landmark tracking to guarantee real-time robot localization, target tracking for a sensor based motion or obstacle tracking to control an obstacle avoidance procedure. These objects -landmarks, targets, and obstacles- have different characteristics: this paper shows that a single tracking method could not overcome all the tracking tasks. Several methods must be integrated on the robot. A tracker controller is in charge to activate a specific method with respect to the object type and to a transition model used to recover the tracking failures. This method is validated using four trackers, based on: template differences, set of points, lines and snakes. Experimental results provided on numerous image sequences are presented.**

I. Introduction

Under real-world environmental conditions, a single tracking method cannot overcome all different tasks and situations presented in it. Moreover, tracking methods are not robust enough to work under varying environmental conditions. Tracking often fails when illumination undergoes significant changes. Some other situations where tracking methods fail are due to cluttered background, changes in model pose, occlusions of the target image or motion discontinuities in object dynamics. In order to overcome these limitations, we propose to integrate a collaborative tracking controller. This controller is charged to select, the tracking method to be use, in function of the target and environments conditions. In addition, when a specific method has failed, it has to select another method and then to try to recover the target, or if it is not possible, to select another target, depending on the task to achieve. A tracking transition model that uses performance evaluation of the currently selected method and context identification to guarantee robust tracking on a mobile robot determines method switching. Each tracking method is associated with a different type of model so model coherence is a fundamental issue in tracking integration.

The rest of this paper is organized as follows: We review related work in tracking integration and active control in section II. In section III, we present our visual functionality model. Tracking methods included in this functionality are described in section IV. We discuss the tracking transition model in section VI. In section V, results of our experimentations illustrate strategies applied to keep model coherence after tracking method switching. Finally, section VII presents our conclusions and the future work to be done.

[1]Corresponding author: amarin@laas.fr, Phone: +33 5 61 33 64 42, Fax: +33 5 61 33 64 55, Ph.D student funded by Mexican goverment thanks to a CONACyT grant.

[2]This work has been partially funded by Ecos-Nord action M99M01

II. Related work on collaborative tracking

Integration of tracking methods intends to increase robustness of the robotic vision systems for visual tracking tasks. Toyama and Hager [17] have proposed to use a layered hierarchy of visual tracking methods. Some others have proposed fusion of information from different tracking methods to improve reliability of the visual tracker: Kruppa *et al.* [13] takes advantage of a mutual information approach to fuse different tracking models from several tracking methods executed simultaneously. Rasmussen and Hager [16] use a probabilistic approach to control a hierarchy of tracking strategies. Jepson *et al.* [10] perform the integration of several tracking methods limiting its scope to appearance models for face tracking in cluttered natural environment.

Some other works have been focused on performance evaluation to control one or several tracking methods in an active way. Barreto *and al.* [2][4] have developed some performance evaluation measures in order to control active tracking using a stereo head. Resolution control has been developed by Ferrier [6] in order to achieve a stable visual tracking system. Ayala *and al.* [1] have proposed a active tracking functionality that controls acquisitions parameters for a Hausdorff based tracking method. Guibas *et al.* [7] shows how to integrate tracking methods in a global robotic task.

III. Visual functionality

A visual functionality is a block where a visual task is performed. Our visual functionality is modeled as depicted in Fig. 1.

The elements of this model are as follows:

- A set of N visual functions $O = \{O_1, O_2, \ldots, O_N\}$ able to perform the visual task. Each function O_k is associated with a set of parameters $P^t_{k,i}$, $i \in [1, N_{P_k}]$ that can be adapted according to the current image. Each visual function has also an enable signal E^t that determines if the method has to be executed on the current input data. A local adaptation mechanism is provided for local adaptation of the parameter set. This adaptation is done based on a performance evaluation measure m_t of the application of the method to the current image.

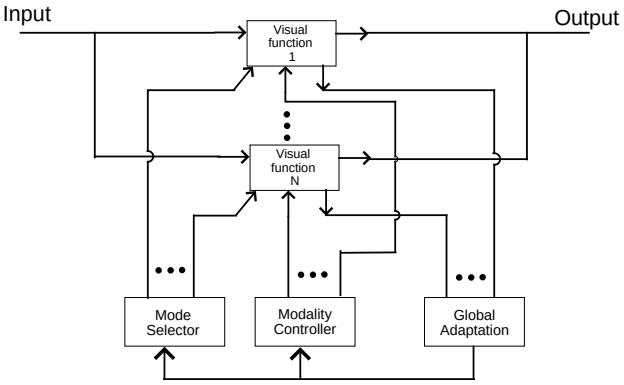

Fig. 1. Visual functionality block diagram.

- A mode selector that determines which methods have to be executed on the current input data.
- A modality controller that determines a set of parameters for the selected visual function to perform the visual task.
- A global adaptation mechanism that controls both the mode selector and the modality controller using performance evaluation information from all the currently active methods.

In this paper, we apply this model for a visual target tracking task that can be concurrently performed by several tracking methods.

IV. The tracking methods

Tracking is used on a mobile robot during a navigation task, in order to follow a target or to keep a landmark in the view field. Four trackers are considered here, based on template differences, set of interest points, lines and snakes.

For every tracker, only a small image region is examined to obtain new target position, as opposed to the entire image. In this manner, the computation time is decreased significantly. The idea behind a local exploration of the image is that if the execution of the code is quick enough, the new target position will then lie within a vicinity of the previous one. In this way, the robustness of the method is increased to handle target deformations, since it is less likely that the shape of the model will change significantly in a small δt. We do not describe here, the prediction step based on Kalman filtering, used to estimate the predicted target position at $t+1$ knowing its position at t

A. Template difference tracker

Template tracking is an interesting region based approach based on template differences ([8], [11]). This method consider that, process rate frequency is high (∼20Hz), to make linear approximations of a planar patch movements. Let $\mathbf{R} = \{x_1, x_2, ..., x_N\}$ the set of N image locations which define target region, and let $\mathbf{I}(\mathbf{R}, t)$ the vector of the intensities of region $\mathbf{R}$ at time t. Taking the reference image at time t_0 = 0, we define the reference template as $\mathbf{I}(\mathbf{R}, t_0)$. Motion of reference template over time produces deformations, rotations and/or translations in images at $t > 0$. Considering that the motion could be model by a parametric function $f(\mathbf{x}; \mu)$, called *motion model* and parameterized by $\mu = (\mu_1, \mu_2, ..., \mu_n)$ with initial condition μ_0 at t_0. Then to track an object in the image sequence is to determine the vector $\mu(t)$. Considering that $N >> n$ and that f and μ are both differentiable. The *motion parameter vector* μ can be estimated at time t by minimizing the following objective function:

$$O(\mu) = \sum_{x \in \mathbf{R}} (I(\mathbf{f}(\mathbf{x}; \mu), t) - I(\mathbf{f}(\mathbf{x}; \mu_0), t_0))^2$$

Considering that, at time $t > t_0$ the motion parameter vector can be approximated by $\mu(t+\tau) = \mu(t) + \delta\mu$, τ been a very small increase of time t, the objective function O can be expressed in terms of $\delta\mu$, as :

$$O(\delta\mu) = ||I(\mu(t) + \delta\mu, t+\tau) - I(\mu_0, t_0)||$$

We can linearized the problem expanding $I(\mu(t) + \delta\mu, t + \tau)$ in a Taylor series around μ and t, as is described in [8]. Solving the set of equations for $\nabla O = 0$, we get:

$$\mu(t+\tau) = \mu(t) - (\mathbf{M^t M})^{-1} \mathbf{M^t} \delta\mathbf{i}$$

where $\mathbf{M}$ is the *jacobian matrix* of $\mathbf{I}$ with respect to μ, and $\delta\mathbf{i}$ is the intensity error vector defined as :

$$\delta\mathbf{i} = \mathbf{I}(\mu(t), t+\tau) - I(\mu_0, t_0)$$

Hager and al [8], propose different models to reduce Jacobian matrix computation, Jurie and al [11], consider an *interaction matrix* $\mathbf{A}$, defined by:

$$\mu(t+\tau) = \mu(t) + \mathbf{A}(t+\tau)\delta\mathbf{i}$$

which, could be learnt in an offline computation stage, by making a set of N_p small perturbations $\delta\mu$ and calculating the intensity error vector $\delta\mathbf{i}$ for each one, in a local reference system, which makes easier the computation of $\mathbf{A}$. Making $N_p > N$, is possible to obtain the interaction matrix $\mathbf{A}$ from the N_p couples $(\delta\mathbf{i}, \delta\mu)$, such that next term be minimal.

$$\sum_{k=1}^{k=Np} (\delta\mu^k - \mathbf{A}\delta\mathbf{i}^k)^2$$

We have implemented this learning method: figure 2 shows an example of such a target, here a poster.

(a) (b) (c)

Fig. 2. (a) the initial template; (b) the template after a little motion $\delta\mu$ (c) the template difference

B. The set of points tracker

This tracking method, has been presented in [9], [1]. The tracking is done using a comparison between two sets of points: on the one hand, points extracted from the current image: discontinuity points, detected by a classical edge detector or interest points extracted by Susan, Harris detectors; and on the other hand, points given as the target model. The Hausdorff distance is used to measure the similarity between these sets of points. Other works have proposed to track only points, without applying constraints on their relative positions, by using a correlation function, independently for several small windows (typically 10×10 pixels), selected around interest points.

A partial Hausdorff distance is used as a resemblance measurement between the target model and its predicted position in an image. Given two sets of points P and Q, the partial Hausdorff distance is defined as:

$$H(P,Q) = \max(h(P,Q), h(Q,P))$$

where

$$h_k = K^{th}_{p \in P} \min_{q \in Q} \| p - q \|$$

$\| p - q \|$ is a given distance between two points p and q. $K^{th}_{p \in P} f(p)$ denotes the K^{-th} ranked value of $f(p)$ over the set P.

The function $h(P,Q)$ (distance from set P to Q) is a measure of the degree in which each point in P is near to some point in Q. The Hausdorff distance is the maximum among $h(P,Q)$ and $h(Q,P)$; it gives the distance between the most mismatched points on the two sets, rejecting the K worse matchings, considered as outliers.

At step $t+1$ of the sequence, the first task is to search the position of the model M_t in the next image I_{t+1}, around the predicted position. The minimum value of the unidirectional partial distance from the model to the image, $h_{k1}(M_t, I_{t+1})$, identifies the best "position" of M_t in I_{t+1}, under the action of some group of translations G. The target search is stopped at the first translation g, such that its associated $h_{k1}(M_t, I_{t+1})$ is no larger than a given threshold τ, considering a target search strategy that sweeps space of possible translation following a spiral trajectory around the predicted position.

Having found the position of the model M_t in the image I_{t+1} of the sequence, the second task consists in updating the target model M_t: the new model M_{t+1} is built by determining which pixels of the image I_{t+1} are part of the target. The model is updated by using the unidirectional partial distance from the image to the model as a criterion for selecting the subset of images points I_{t+1} that belong to M_{t+1}. In order to allow scale change, the scale is increased whenever there are a significant number of nonzero pixels near the boundary and is decreased in the contrary case.

The model M_{t+1} can be filtered applying some shape constraints on the target by morphological operators (for example, rather elliptical target to track a face). Fig. 3 presents an example of this tracker on a person.

(a) (b)

Fig. 3. Tracking on a person: (a) image; (b) target

C. Line tracker

The line tracker [5] is focused on straight lines; in our implementation, the tracker can search in every image I_{t+1} of a sequence, n lines $({l^i}_t, i = 1, n)$ without verifying constraints between these lines (for example, parallelism or convergence on a vanishing point).

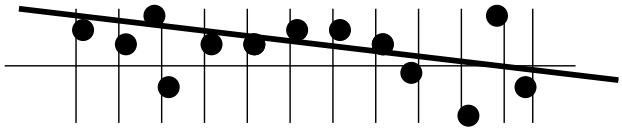

Fig. 4. 1D correlation principle.

The method is based on 1D correlation to look for a line ${l^i}_t$ in a predicted position (figure 4). (a) Points are regularly sampled on ${l^i}_t$. (b) Every point is matched with the closer discontinuity found on the normal to the line on this point: the optimal discontinuity position is found using a correlation along this normal, with a generic step model. (c) A RANSAC fitting method allows to generate the line position ${l^i}_{t+1}$, keeping only the aligned points, rejecting the outliers. The line is tracked if at least 50% of the sampled points are matched with aligned points.

In contrast to a classical segment tracker, this method is very robust: it can be applied also to track curves defined by some analytical model in order to apply the RANSAC fitting (a quadric or a cubic curve for example). The initial lines $({l^i}_0, i = 1, n)$ are generally provided by the projection of 3D segments on the image (for example, the wall-ground or wall-ceiling edges), or by the edges of an object recognized in an initial image.

D. Snake tracker

Snakes or active contours [12] have been extensively used for object tracking. This method is based on energy minimization along a curve, which is subject to internal and external forces. The total energy for an active contour $\mathbf{v}$ described with a parametric representation $\mathbf{v} = (x(s), y(s))$ can be written as:

$$E_{tot}(\mathbf{v}) = \sum E_{int}(\mathbf{v}(s)) + E_{ext}(\mathbf{v}(s))ds$$

Marin *and al* [15] propose to consider the active contour as the surface of an electrical charged conductor; these electrical charges generate a new internal force. As a

result of these repulsive forces, the control points are redistributed along the curve with higher density in zones of high convex curvature. Then internal energy can be written as:

$$E_{int}(v) = \sum \omega_1(s)(\frac{\delta v}{\delta s})^2 + \omega_2(s)(\frac{\delta^2 v}{\delta s \delta s})^2 + k\frac{\sigma^2}{(\frac{\delta^2 v}{\delta s \delta s})^2}ds$$

where $\omega_1(s)$ and $\omega_2(s)$ are weights given to elasticity and smooth-ness energies, k a constant and σ the charge density.

The evolution of the curve is determined by the equation of movement derived from energies terms.

A parametric shape-model could be defined (deformable template) when the object to be track is well known, which will be matched to an image, in a very similar way like snakes [3]. Another way to introduce shape information in active contours is applying constraints in order to deform the snake in a suitable way. Relationships between control points can be incorporated as energy terms and multiple control points can be introduced to describe vertices or discontinuities [14].

Fig. 5 presents an example of a person tracking using a snake initialized by an operator.

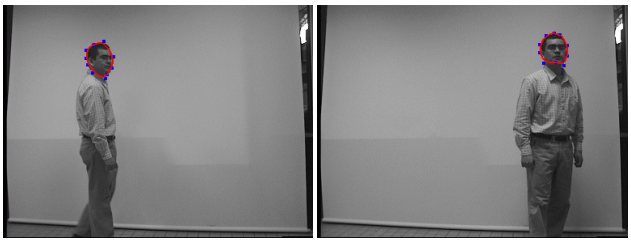

Fig. 5. Tracking on a face by the snake tracker

V. Tracker characterization

The tracker methods must be described, mainly for the initialization conditions, the confidence measurement to be used for failure detection and information required by the tracker selection mechanism. The selection will be based on intrinsic properties of the tracker (for example, template differences tracking must use a textured target), but also on more contextual information (for example, the image background is textured or uniform).

A. Template differences tracker

Template differences tracking can be used only for planar textured templates, or for quasi-planar textured templates (for example, a face), when the movements do not change a lot planar perspective; it is a fast method (at least 20 Hz).

A knowledge basis must be built on planar templates to be tracked during an offline computation; the interaction matrix $\mathbf{A}$ must be learnt off line, because the execution time is too important for on line computation. In this case, the image from which $\mathbf{A}$ is learnt, needs to be acquired from a viewpoint close to the first camera online position.

One of the major problems with template tracker is that, once the target is lost, it is nearly impossible to find it again, by the method itself. A confidence measure is the norm of the μ vector. So when, this norm overpasses a threshold (typically, the largest learnt variation), it could be considered as a tracker failure.

B. Set of points tracker

This tracker has good performances in non-structured environments or with deformable objects, the appearance of which could change gradually. This method is based on a matching process, so it is rather slow (from 3 to 5Hz), but a target lost can be detected again.

The target is only defined by a region of interest in the image, without any a priori learning process. The confidence measure is given by the Hausdorff distance between the current image and target model.

When the tracking is performed over very complex images (too much texture), some errors can happen, because the target model will be polluted by outliers.

C. Line tracker

The exact opposite than the previous tracker: good performances with structured targets, quick method (10Hz) which could detects a target lost by itself.

The initial conditions are a set of segments to be tracked; one problem is that, due to the RANSAC fitting, the length of these segments are always decreasing along a sequence, and some high level information must be used to restore the segment sizes.

This method works very fine with either a uniform target on a textured background, or a textured target on a uniform background. Nevertheless, if the image is too complex, in spite of the RANSAC fitting, some segments could be lost. The number of points integrated in the segments gives the confidence measure.

D. Snake tracker

This tracker is normally used for deformable or free form shapes, with high gradient edges between the background and the target; it means that the environment cannot be very complex. The speed depends on the number of control points on the snake, but typically, it runs at 12Hz.

The control points can be initialized close to the silhouette of the target. In order to guarantee the convergence of the method, internal weights parameters ω_1, ω_2 and k can be modified, to be adapted to the current situation, but such an adaptation need to be very well characterized in a learning processing stage.

As the template tracker, once a snake has lost their target, it is nearly impossible to recover it, without a recognition stage. We consider that snake has lost their target when first area moments of the enclosed region, has a high variation.

VI. Tracking transition model

Once every tracker has been characterized, the system can associate a target with the best tracker, which will be able to maintain the target position along an image sequence. When it is required in a sensor based navigation system, the decisional level will generate a target-tracking request. Depending on the target type, a tracker control

will activate the best method. The transition between different trackers could occur in two situations:

- An error is detected, either by the tracker itself using the confidence measures described in the previous section, or by a low frequency interpretation function (double loop paradigm).
- The tracked target is occluded or moves out of the image due to the target or the camera motion. Depending on the task, another target must be selected and tracked.

Two examples are presented on figures 6 and 7: the red lines are the boundaries of the predicted position, the green lines show the tracker results.

A. Recovering an error tracking

Fig. 6 presents a transition due to a tracker failure. The target is the poster moved here by an operator, on a very complex background. The template differences tracker is selected for this planar target. Tracking is right on the first 60 images ((a) to (d)), but due to the background, the computed $\delta\mu$ on the image 62 is wrong on (e) and the tracker diverges totally on the image 65 (f). The set of points tracker is activated using as initial poster model, the interest points extracted from the last good window detected by the template tracker (g). Thanks to the Hausdorff distance, the poster model is found and the tracking can continue at a low frequency on (h). As template differences tracker is faster, the tracker controller will activate it again and it will be executed concurrently with the set of points tracker during some frames, to verify it converges again. As it is faster, the tracker controller will activate again the template differences tracker executed concurrently with the set of points tracker during some frames, to verify it converges again.

B. Transition between targets

Fig. 7 presents a transition between trackers and between landmarks: the robot is navigating on a corridor. When he enters the visibility area of the poster on the left wall, it tracks this poster on figures (a) to (c), and concurrently, computes the relative position camera-poster, and then robot-environment, using a metric map built off line. We use the template differences tracker in this sequence: the line tracker could be used also, because the wall is uniform.

When the robot goes forward, the boundary of the poster region in the image becomes closer to the image limit; when it is too close in (d), the tracker controller selects another target to be tracked, here the lines corresponding to the edges wall-ceiling. The line tracker is initialized with the projections of the two 3D edges on the current image: the two trackers are executed concurrently on 10 frames (because the template tracker is very fast) and then the poster tracking is stopped to avoid a failure due to the poster output from the image.

VII. Conclusions and future work

This paper has described the tracking module integrated on our mobile robots. Tracking is required for several tasks a robot must perform: object following (another robot, a person), visual based navigation (trajectory defined as a sequence of visual servoing commands) or obstacle following during the execution of a motion (trajectory defined as a curve on the ground).

A collaborative tracking method has been proposed so that, the robot can selects the best tracking method depending on the target type or some contextual knowledge (nature of the background ...). A target controller has been described. This module will request tracker switches, either if a target is lost, by one quick but not so robust method, and a more robust method is activated to find again the target using local image measurements, or if a target switch is necessary, for example because the robot must turn around an edge after a corridor following execution. Every target is associated with the more adapted tracker and with a recovery procedure in case of failure.

Only sequential executions are proposed: another strategy could be to activate concurrently several tracking methods when a target must be tracked, and to fuse at the control level, the different target representations and positions. This solution could be the best one, but at this time, is not compatible with the real time constraints.

VIII. References

[1] V. Ayala-Ramírez, C. Parra, and M. Devy. Active tracking based on Hausdorff matching. In *Proc. of ICPR'2000*, volume 4, pages 706–709, Barcelona, Spain, September 2000. IAPR.

[2] J. P. Barreto, P. Peixoto, J. Batista, and H. Araujo. *Robust vision for vision-based control of motion*, chapter Evaluation of the robustness of visual behaviors through performance characterization. IEEE Press, 1999.

[3] A. Blake and M. Isard. *Active Contours*. Springer, 1998.

[4] S. Chroust, J. Barreto, H. Araujo, and M. Vincze. Comparison of control structures for visual servoing. In *Proc. ICAR'2001*, pages 235–240, Budapest, August 2001.

[5] T. Drumond and R. Cipolla. Real-time visual tracking of complex structures. *IEEE. Trans. on Pattern Analysis and Machine Intelligence*, 24(7):932–946, July 2002.

[6] N. J. Ferrier. *Workshop on Modelling and Planning for Intelligent Systems*, chapter Active control of resolution for stable visual tracking. LNCS. Springer, 1999.

[7] L. G. Guibas and al. Model building, target finding and target tracking: Performance data on the Stanford TMR Project. Monthly report of Tactical Mobile Robots project, Stanford Robotics Lab, August 1999.

[8] G. Hager and P. Belhumeur. Efficient region tracking with parametric models of geometry and illumination. *IEEE. Trans. on Pattern Analysis and Machine Intelligence*, 20(10):1025–1039, October 1998.

[9] D. Huttenlocher, G. Klanderman, and W. J. Rucklidge. Visually guided navigation by comparing two dimensional edge images. Technical Report TR-94-1407, Stanford University, Stanford, California, January 1994.

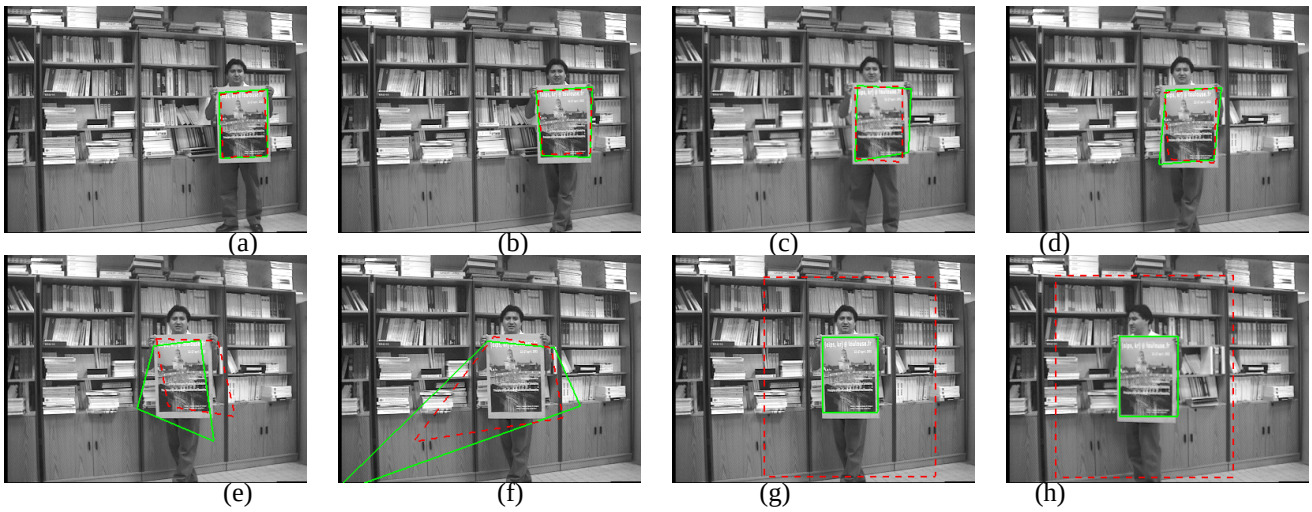

Fig. 6. A transition after the detection of a tracker failure.

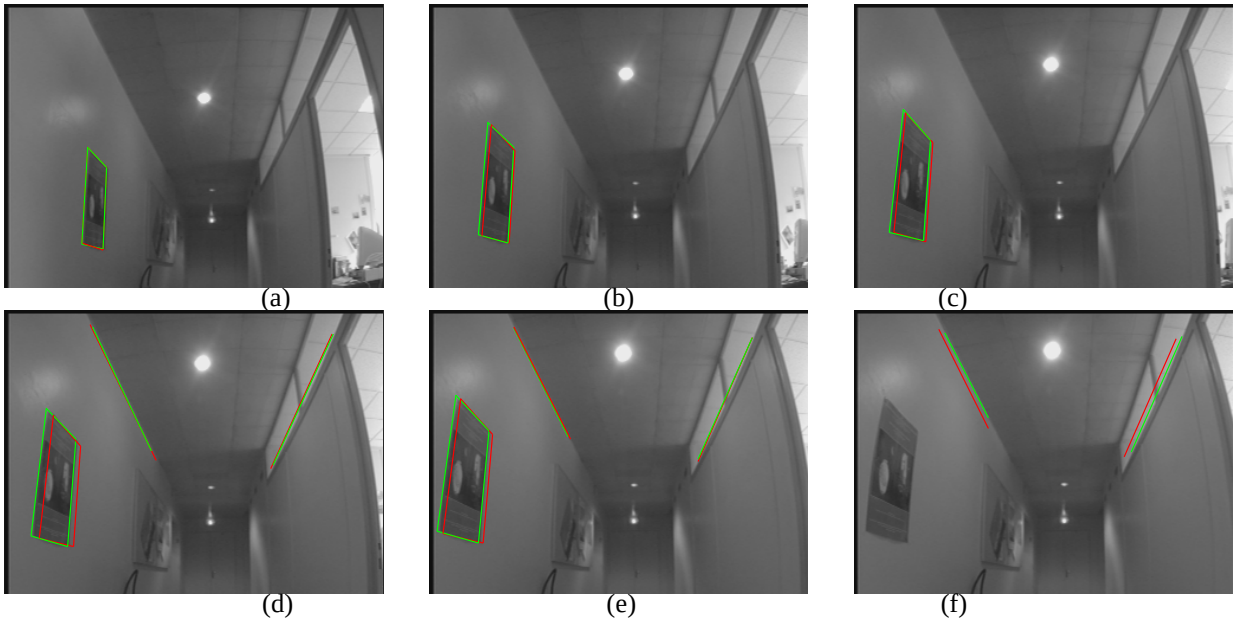

Fig. 7. A transition between tracked features: a poster, then corridor corners.

[10] A. D. Jepson, D. J. Fleet, and T. F. El Maraghi. Robust online appeareance models for visual tracking. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition CVPR'01*, volume 1, pages 415–422, Kauai, 2001. IEEE Computer Society.

[11] F. Jurie and M. Dhome. A simple and efficient matching algorithm. In *Proc. Int.Conference on ¿Computer Vision(ICCV'99)*, pages 265–275, 1999.

[12] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1988.

[13] H. Kruppa, M. Spengler, and B. Schiele. Context-driven model switching for visual tracking. In *Proc. 8th Int. Symp. on Intelligent Robotic Systems SIRS'00*, 2000.

[14] A. Marin-Hernandez and M. Devy. Model-based active contour for real time tracking. In *Proc. International Symposium on Robotics and Automation 2002 (ISRA'02)*, Toluca, Mexico, September 2002.

[15] A. Marin-Hernandez and H. Rios-Figueroa. Eels: Electric snakes. *Computacin y Sistemas*, 2:87–94, 1999.

[16] C. Rasmussen and G. Hager. Probabilistic data association methods for tracking complex visual objects. *IEEE Trans. on Pattern Analysis and machine Intelligence*, 23(6):560–576, June 2001.

[17] K. Toyama and G. Hager. Incremental focus of attention. In *Proc. IEEE Int. Conf. on Computer Vision and Pattern Recognition*. IEEE Computer Society, 1996.