Using multiple hypothesis in model-based tracking

Céline Teulière, Eric Marchand, Laurent Eck

***Abstract*— Classic registration methods for model-based tracking try to align the projected edges of a 3D model with the edges of the image. However, wrong matches at low level can make these methods fail. This paper presents a new approach allowing to retrieve multiple hypothesis on the camera pose from multiple low-level hypothesis. These hypothesis are integrated into a particle filtering framework to guide the particle set toward the peaks of the distribution. Experiments on simulated and real video sequences show the improvement in robustness of the resulting tracker.**

I. INTRODUCTION

Knowing the pose of a camera with respect to a specific object or part of the environment is a key requirement in many applications, ranging from augmented reality to robotics. Among the various approaches that have been proposed to address this task, model-based methods have shown growing performances in the past years. The information given by the knowledge of a template or 3D CAD model of the object allows to improve the tracking robustness.

In this paper, the problem is restricted to model-based edge tracking, where a 3D CAD model composed of the linear edges of the object to be tracked is assumed to be known. The task then consists in finding the camera pose which provides the best alignment between the model projected edges and the edges detected in the image [11], [5], [4]. Edges are frequent in industrial environment, indoor, and in urban environment. They offer a good degree of invariance to pose and illumination changes and are easy to detect even in presence of some noise or blurring, which makes them of great interest for visual tracking. However, contrary to feature points which are usually extracted so as to be as specific as possible to allow a robust matching, edges suffer from having very similar appearances. Therefore, some ambiguities can occurr when different edges get close to each other which can lead to wrong matches and tracking failures.

The approaches that have been considered to tackle this issue can be divided into three main categories:

- One way to improve the robustness of the tracker is to use another source of information, by fusing the information given by edges with information given by another kind of feature (such as points of interest [16]) or another sensor [8].
- Various studies propose to improve the low level robustness, by using robust estimation techniques [12] which remove outliers during the registration process [1], [5], [4]. However, since only one hypothesis for the pose is handled, a wrong edge matching can still lead to a tracking failure. [16] tried to overcome this issue by including multiple low level hypothesis in the robust registration method, showing interesting results. These methods have notably improved the performances of the trackers, but still maintain a unique hypothesis on the camera pose.
- Another category of approaches is composed by Bayesian filters, which estimate the camera pose using a dynamic model to establish a prediction and an observation model to correct it. This can be achieved by Kalman filtering when uni-modal Gaussian distributions are considered. More recently, the improvement of computational performances has allowed to consider particle filtering approaches [15], [9], [14]. Instead of going from the low level edges to retrieve the camera pose, particle filtering uses a set of hypothesis on the possible camera poses (the particles). The likelihood of each particle is then measured in the image. Since the space of all possible poses is large, one difficulty is to keep a fair representation of the different modes of the state probability distribution while using few particles.

In this paper we propose a new approach to improve 3D model-based tracking robustness. We first present a robust multiple hypothesis tracker which provides several hypothesis on the camera pose from the low level hypothesis corresponding to low level ambiguities in edge matching. Then we show how this tracker can be used to guide the particles of a particle filter.

The remainder of this paper is organized as follows. Section II describes how low level hypothesis can be integrated in a registration process to give several hypothesis on the camera pose. Experimental results with the resulting tracker are presented in section II-D. Section III shows how to integrate this tracker into a particle filtering framework. Comparative results are also provided.

II. REGISTRATION PROCESS

A. Considering multiple low level hypothesis in the tracking process

In order to track the relative pose between the camera and a known object, the approach we consider relies on a similar basis than the one used in [4], [5] and [16]. Assuming the camera parameters and an estimate of the pose are

C. Teulière and L. Eck are with CEA, LIST, Interactive Robotics Unit, 18 route du Panorama, BP6, Fontenay aux Roses, F- 92265 France firstname.name@cea.fr

E. Marchand is with Université de Rennes 1, IRISA, INRIA, Lagadic Project, Rennes, France firstname.name@irisa.fr

This work was realized in the context of the French ANR national project SCUAV (ANR Psirob 06_174032 SCUAV project ref ANR-06-ROBO-0007-02)

known, the CAD model is first projected into the image according to that pose, which can be the previous one or a prediction obtained from a filter as presented in section III. Each projected edge E_i is then sampled, giving a set of points $\{e_{i,j}\}$ (see figure 1). From each sample point $e_{i,j}$ a search is performed along the edge normal.

In [4] and [5], the point of maximum likelihood with regard to the initial point $e_{i,j}$ is selected. It is denoted by $e'_{i,j}$ in the following. An optimization method is then used to find the camera pose which minimizes the errors between the selected points and the projected edges [11], [4] and [5]. Formally, the quantity to minimize can be expressed by:

$$S = \frac{1}{N_e} \sum_i \sum_j \rho \left(\Delta_{E_i}(e'_{i,j}) \right) \quad (1)$$

where $\Delta_{E_i}(e'_{i,j})$ is the squared distance between $e'_{i,j}$ and the projected edge E_i, N_e is the total number of sampled points, and ρ is a robust estimator. However, as shown in figure 1, ambiguities can occurr when several strong edges are found along the normal to the contour, which can lead to tracking failures. Examples of these situations in tracking sequences are shown in figure 5 and 4.

To overcome this issue, the idea of keeping several low level hypothesis has been proposed in [16]. Different hypothesis $\{e'_{i,j,l}\}$ corresponding to local extrema of the image gradient along the edge normal in $e_{i,j}$ are memorised (see figure 1). They are included into the registration process by introducing a multiple hypothesis estimator ρ^* defined by:

$$\rho^*(x_1, ..., x_n) = \min_i \rho(x_i) \quad (2)$$

Equation (1) becomes:

$$S^* = \frac{1}{N_e} \sum_i \sum_j \rho^* \left(\Delta_{E_i}(e'_{i,j,1}), ..., \Delta_{E_i}(e'_{i,j,n_{i,j}}) \right) \quad (3)$$

where $n_{i,j}$ is the number of selected hypothesis for the sample point $e_{i,j}$. The multiple hypothesis robust estimator ρ^* determines the hypothesis to reject, allowing robustness improvement.

In our approach, we also consider different hypothesis $\{e'_{i,j,l}\}$ corresponding to potential edges. The main difference is that we go from these multiple low level hypothesis to multiple hypothesis on the camera pose itself instead of chosing between the hypothesis during the registration. The next section will explain how this is achieved.

B. Segmenting the low level hypothesis into edge hypothesis

In order to get multiple hypothesis on the camera pose corresponding to the detected low level hypothesis, several minimizations can be performed, using different sets of points in (1). Since considering all the possible combinations of points is obviously not an option, our first step is to determine the underlying lines of the set of points $\{e'_{i,j,l}\}$, to group the points into different sets corresponding to potential edges (see figure 2). This is achieved using a k-mean classification algorithm [6]. For each projected edge E_i, the algorithm segments the candidate points $\{e'_{i,j,l}\}$ into

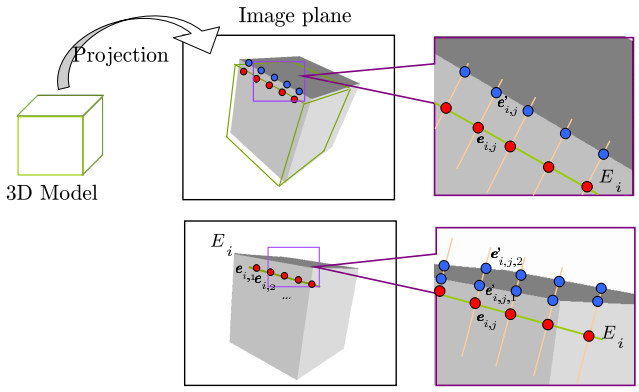

Fig. 1. In classic edge based tracking, the model is projected into the image plane and points are sampled on the projected edges. A search is performed along the normal (top). When multiple strong edges are close in the image, ambiguities can occur when searching along the normal (bottom). Multiple low level hypothesis can be considered.

k_i sets of points or classes $(c^i_1, ..., c^i_{k_i})$. The mean of each of the k_i classes is in our case the line which best fits the points of that class, obtained by a robust least square minimization. To initialise the algorithm, the number k_i of classes for the edge E_i is set to the maximum number of candidate points detected, that is: $k_i = max_j\{n_{i,j}\}$. The classes $(c^i_1, ..., c^i_{k_i})$ are initialised using the order in which the hypothesis have been found on the normal. That is for each class c^i_m: $c^i_m = \{e'_{i,j,m}\}_j$. This initialisation is often close to the good segmentation, allowing the algorithm to converge faster (see figure 2 (a)). At each iteration of the algorithm, the mean of each class is computed (figure 2 (b)). Each point is then assigned to the class with the nearest mean line. The algorithm is deemed to have converged when the assignments no longer change. Since the potential edges are not supposed to be normal to the initial edge, we add the constraint that two hypothesis e'_{i,j,l_1} and e'_{i,j,l_2} of a same initial sample point $e_{i,j}$ cannot belong to the same class.

Finally, the k-mean algorithm corresponding to the initial edge E_i provides us with a set of classes $c^i_m = (\{e'_{i,j,m}\}_j, r^i_m)$ where r^i_m is the residue of the least square minimization, and represents a likelihood criterium that will be used in the next step. In practice, only lines with a sufficient number of points are taken into account. Figure 2 shows a simple example of the process.

Although the contours considered have been restricted to lines in this study, the approach can be easily adapted to other kinds of contours.

In most cases k_i does not exceed two or three. Figure 4 gives an example of the lines detected from the teabox sequence.

C. From multiple edge hypothesis to multiple hypothesis on the camera pose

Once candidates have been obtained for each edge in the form of sets of points associated to a residue, random weighted draws are performed. Weights w^i_m considered for

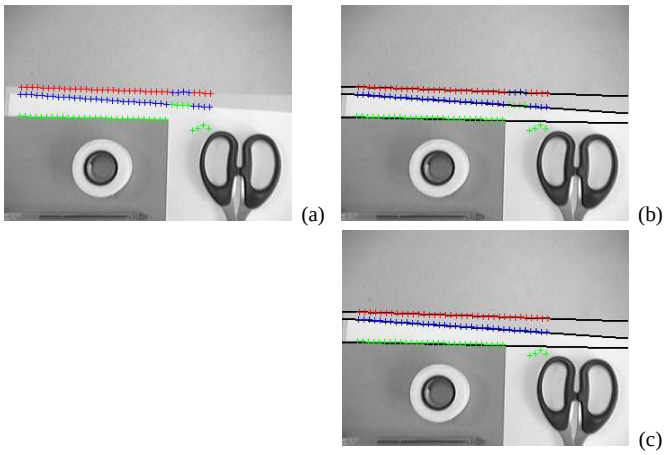

Fig. 2. An example of the k-mean computation, with $k = 3$. Each class is represented by a different color. (a) Initialisation of the classes of points. (b) Mean lines computation. (c) The final segmentation is obtained in one step.

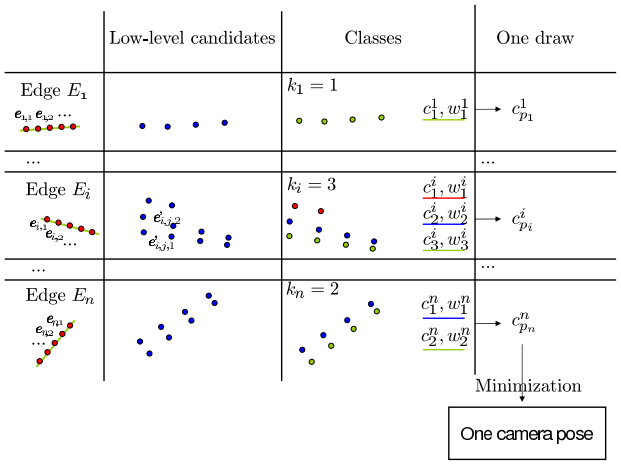

Fig. 3. From low level hypothesis, classes of points are extracted. For each projected edge a random weighted draw is performed among the classes to determine the points that will be used for the minimization process. The minimization provides an hypothesis on the camera pose. A different draw would lead to another candidate pose.

each candidates are deduced from the residues by:

$$w_m^i = \begin{cases} e^{-\lambda\left(\frac{r_m^i - r_{min}^i}{r_{max}^i - r_{min}^i}\right)^2} & \text{if } r_{max}^i \neq r_{min}^i \\ 1 & \text{otherwise.} \end{cases} \quad (4)$$

where λ is a parameter that can be tuned according to the selectivity that is desired.

One weighted draw denotes here the draw of one candidate per edge, that is, for each edge E_i a class $c_{p_i}^i$ is drawn from the k_i classes. From each draw, a numerical non-linear minimization is performed according to (5), using the set of points corresponding to the picked classes, resulting in a camera pose.

$$S = \frac{1}{N_e} \sum_i \sum_{e'_{i,j,l} \in c_{p_i}^i} \rho\left(\Delta_{E_i}(e'_{i,j,l})\right) \quad (5)$$

Since the optimization is deterministic, it is only computed when the sets of candidates are different. The weighted draw allows to favour, among all the possible combinations, the ones with the candidates of lowest residue, which are more likely to correspond to a real edge. Several hypothesis on the camera pose are thus obtained from the low level detected hypothesis.

The process is illustrated in figure 3.

In practice, since the number of candidate lines per edge is small, so will be the number of optimizations to be performed and thus the number of pose candidates obtained.

D. Experimental results

To illustrate the interest of this approach in terms of robustness, figure 4 shows a concrete example where the multiple hypothesis allow to avoid failure. At this particular frame, extracted from a video sequence, a single hypothesis tracker fails. While running the multiple hypothesis algorithm, it appears (see figure 4) that only one candidate is found for almost every projected edge, except the top back one. For this edge, two candidates have been found, which lead to two different draws. Whereas the single hypothesis tracker fails due to a wrong match, the multiple hypothesis tracker finds the correct pose (figure 4-(2-b)).

To validate the proposed approach, we also used a simulated sequence, for which the ground truth is known. The comparative results between the classic registration method and our multiple hypothesis method are shown in the figure 5.

In the single hypothesis case, only the maximum likelihood point is selected in the search along the normal. The tracking fails when confronted to ambiguities, that is especially when two edges get close from each other, or when a new face appears (figure 5-(a) and (c)).

In the multiple hypothesis case, the output considered at this stage is the camera pose which gives the lowest residue in the minimization process. The object is successfully tracked even in cases of ambiguities. However, at some ambiguous frames where two candidates gives almost the same residue, the tracker selects the wrong one as the best, which results in some jitter on the camera trajectory (figure 5 (d)). In the same way, the minimizations which lead to figure 4 (1-b) and (2-b) correspond to minima giving almost the same residues. By selecting only the ”best” one, some information given by other candidates can be lost.
Moreover the tracker still needs frame to frame motion to be small to converge and could benefit from a prediction.

Since our approach provides us with multiple high level hypothesis, particle filtering framework is a natural choice to ensure temporal coherence while keeping a multi-modal probability distribution. The integration of our multiple hypothesis tracker into particle filtering is described in the next section.

III. Oriented particle filter

At this stage, a multiple hypothesis tracker has been presented. We show here how it can be used to guide the particles of a particle filter.

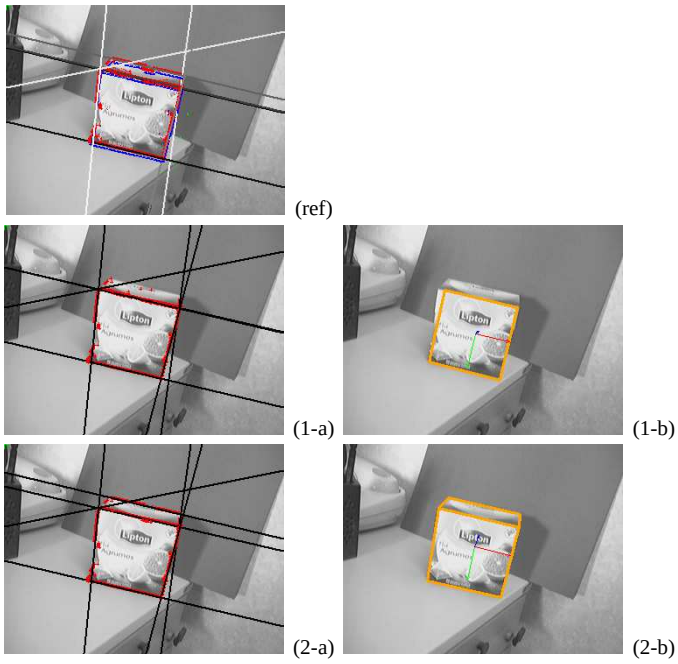

Fig. 4. An example of multiple hypothesis. On the top frame, all the candidate lines and their corresponding points have been represented. The gray level of the line corresponds to its likelihood. When tracking the top-back edge, two hypothesis have been found, one of them corresponding to the top-front edge. (1-a) and (2-a) show two different draws from the initial set of candidates, resulting in two different camera poses (1-b) and (2-b). In the first draw, the back edge has been mixed up with the front one, leading to a tracking failure. Using multiple hypothesis allows to be more robust to such situations.

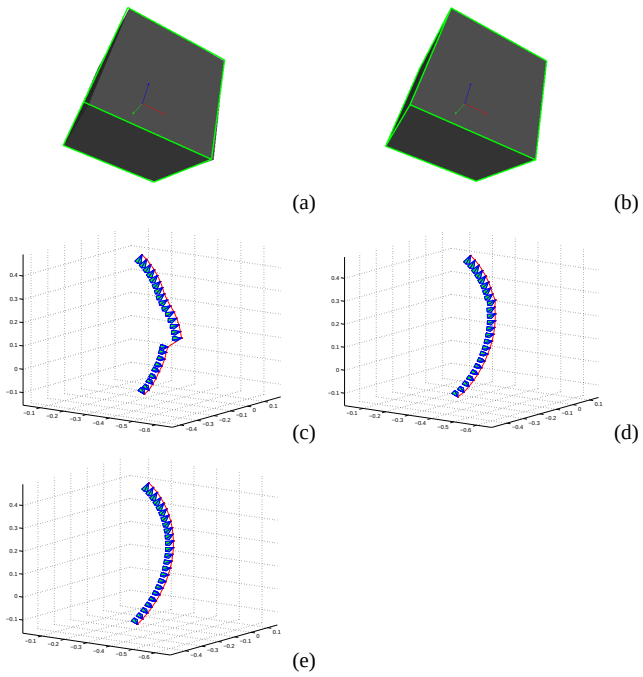

Fig. 5. Comparative results on a simulated sequence with auto-occlusion. The basic algorithm with single hypothesis (a), (c) fails when a new face appears. While considering multiple hypothesis (b), (d), the object is successfully tracked. The camera pose ground truth is represented in (e).

A. Overview

As mentionned in the introduction, particle filtering approaches [7] have been recently introduced in model-based tracking as an alternative to numerical optimization methods, showing promising performances [15], [9], [14].

As for classic particle filters, the main idea is to represent the probability density function $p(\mathbf{x}_k \mid \mathbf{z}_{1:k})$ of the state $\mathbf{x}_k$ at frame k, by a finite set $\{(s_k^{(i)}, \pi_k^{(i)})\}_{i=1..N}$ of N samples, or particles, $s_k^{(i)}$ associated with the weights $\pi_k^{(i)}$. Each particle $s_k^{(i)}$ represents a potential camera pose and $\mathbf{z}_{1:k}$ are the observations until frame k. For each new frame, the particles first evoluate according to a given dynamic model. Then, the likelihood of every particle is measured in the image and a weight is derived. The output considered is usually the weighted mean of the resulting set of particles. The particle set is updated by performing a random weighted draw among the particles.

It is interesting to note that whereas the tracker presented in the previous section was a bottom-up approach, in which multiple hypothesis on the camera pose were derived from low level hypothesis, particle filtering does the opposite. Multiple hypothesis are made on the camera pose at start, and the likelihood of these hypothesis is measured at the low level, in the image.

The main difficulty with this top-down approach, as proposed in [9] and [14], results from the great size of the state space considered. For the tracking to be accurate enough, a large number of particles is required. [9] and [14] use particle annealing method as a hierachical approach to reduce the particle number. However, the likelihood functions proposed still need to be very fast to compute, since they have to be called for each particle. These functions may not distinguish enough between low level ambiguities.

In this paper, we propose to use particles resulting from our multiple hypothesis tracker to guide the particle set towards the local maxima of the distribution. The coherence is ensured using the so-called importance sampling method [7].

B. State space

The state space considered is the special Euclidean group $SE(3)$ of all possible pose matrices which transform points from homogeneous world coordinates to the camera coordinate frame.

$$ {}^c\mathbf{M}_o = \begin{bmatrix} {}^c\mathbf{R}_o & {}^c\mathbf{t}_o \\ 0 & 1 \end{bmatrix} $$

where ${}^c\mathbf{R}_o \in SO(3)$ is a rotation matrix and ${}^c\mathbf{t}_o \in \mathbb{R}^3$ is a translation vector. $SE(3)$ is the group of rigid body transformations.

Since $SE(3)$ is not a Euclidean space, but rather a Lie group, the notions of distance, mean, or Gaussian distributions are not obvious. The question of designing a particle filter on a Lie group has been addressed in detail in [3] and [10]. We recall here two necessary elements: how to propagate the particles and how to compute their mean.

1) Particles propagation on $SE(3)$: As in [9], the propagation model that has been considered in the experiments

is a simple Gaussian noise centered on the previous pose. Since $SE(3)$ is a Lie group, there exists an exponential map between $SE(3)$ and its Lie algebra $se(3)$. Gaussian noise is first added on the canonical exponential coordinates and the resulting pose matrix is computed using the exponential map. Formally,

$$\mathbf{x}_{pred} = \mathbf{M}_{\text{noise}}.\mathbf{x} \tag{6}$$

where $\mathbf{M}_{\text{noise}} = exp(\sum_{i=1}^{6} \alpha_i \mathbf{G}_i), \boldsymbol{\alpha} \sim \mathcal{N}(0, \sigma^2 \mathbf{I}_6)$, the matrices $\mathbf{G}_i$ being the basis of the exponential coordinates in $se(3)$.

2) Mean and averaging on $SE(3)$: Since the addition is not a binary operation on $SO(3)$ (and thus $SE(3)$), the arithmetic mean $\overline{\mathbf{R}} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{R}_i$ of a set of rotation matrices $\mathbf{R}_i$ is obviously not a rotation. However, [13] shows that a meaningful average rotation can be computed as the arithmetic mean $\overline{\mathbf{R}}$, followed by the unique projection onto SO(3) given by the unique polar factor in the polar decomposition of $\overline{\mathbf{R}}$. Let $\overline{\mathbf{R}} = \mathbf{U\Sigma V}$ be the singular value decomposition of $\overline{\mathbf{R}}$, then the mean rotation $\mathbf{R}_m$ is given by:

$$\mathbf{R}_m = \begin{cases} \mathbf{VU}^\top & \text{if } det(\overline{\mathbf{R}}) > 0 \\ \mathbf{VHU}^\top & \text{otherwise,} \end{cases} \tag{7}$$

where $\mathbf{H} = diag[1, 1, -1]$. Using this result, the weighted mean of the particle set considered is composed of this average rotation and the arithmetic mean of the translations.

C. Using hypothesis from low level to guide the particles

To integrate our multiple hypothesis tracker within particle filtering framework we use an approach inspired from [2] where some particles are moved to local maxima of the likelihood by a local optimization. Here, the optimization corresponds to the multiple hypothesis tracker described in section II. To reduce the computational complexity, the optimization is not applied on each particle but only on a subset of particles whose likelihood is greater than a given percentage of the maximum likelihood. The resulting set of particles still provides a good representation of the main modes of the density. However, as underlined in [2] the new set cannot be used directly since it is not sampled from the prior distribution $f_k(\mathbf{x}_k) = p(\mathbf{x}_k \mid \mathbf{z}_{1:k-1})$ as required in particle filtering theory.

However, the N^* new particles $\{(s_k^{*(i)})\}_{i=1..N^*}$ can be regarded as sampled from a function $g_k(\mathbf{x}_k)$, with $g_k(\mathbf{x}_k)$ being a sum of Gaussians centered in the optimized particles. As in (6) the Gaussian functions are computed using the exponential coordinates associated to the poses. To compensate the fact that the new set is sampled from g instead of f, a corrective term f/g is applied to the weights of the particles according to the importance sampling theory [7]. As in [2], we combine the set of propagated particles with the optimized ones, and $f_k(\mathbf{x}_k)$ and $g_k(\mathbf{x}_k)$ are approximated by Gaussian mixtures to evaluate the corrective term:

$$f_k(\mathbf{x}_k) = \frac{1}{N}\sum_{i}^{N} \mathcal{N}(s_k{}^{(i)}, \Sigma)(\mathbf{x}_k) \tag{8}$$

$$g_k(\mathbf{x}_k) = \frac{N}{N+N^*}\left(\frac{1}{N}\sum_{i=1}^{N} \mathcal{N}(s_k{}^{(i)}, \Sigma)(\mathbf{x}_k) + \frac{1}{N^*}\sum_{i=1}^{N^*} \mathcal{N}(s_k^{*(i)}, \Sigma)(\mathbf{x}_k)\right) \tag{9}$$

where $\mathcal{N}(s, \Sigma)$ denotes the 6-dimensional normal distribution of covariance Σ centered on the exponential coordinates of the pose s.

D. Likelihood evaluation

Each particle represents a potential camera pose which has to be evaluated according to what is observed in the image. In [9] the contours are projected according to the particle s to evaluate, and the ratio between the number n of pixels of the projected contours which do correspond to an edge in the image, and the total number v of pixels on the visible contours is computed. The likelihood of the particle s is then derived from that ratio by:

$$p(\mathbf{z} \mid \mathbf{x} = s) = e^{\left(\lambda \frac{n}{v}\right)} \tag{10}$$

where λ is a parameter to tune. To decide whether a pixel does correspond to an edge in the image, a distance map is first computed, providing for each pixel of the image the distance to the closest edge and its direction. (see Figure 6). Then a threshold on the distance has to be set to determine the inlier/outlier count. The distance map has to be computed only once per frame, which make the likelihood value very fast to compute. [9] shows that the computation can be performed in real time on a graphics processing unit (GPU).

In this paper, the distance map is directly used to compute a mean distance:

$$d(s) = \frac{1}{N}\sum_{i} d_i \tag{11}$$

where d_i is the distance given by the distance map for the pixel i, that is the distance between the pixel i and the closest contour in the image. The pixels i are the pixels of the projected edges. The use of the direction to the nearest edge could improve the discriminativity of the distance function. However, we found that this measure was accurate enough in our experiments. Figure 7 shows the shape of the distance d with respect to in-plane translations for the frame of figure 6. The likelihood is derived from this distance by:

$$p(\mathbf{z} \mid \mathbf{x} = s) = \begin{cases} e^{-\lambda\left(\frac{d(s)-d_{min}}{d_{max}-d_{min}}\right)^2} & \text{if } d_{max} \neq d_{min} \\ 1 & \text{otherwise.} \end{cases} \tag{12}$$

As in [9], hidden-edge removal is performed independently for each particle.

The whole algorithm is summarized in the next section.

E. Algorithm summary

Given the set $\left\{(s_{k-1}{}^{(i)}, \frac{1}{N})\right\}_{i=1..N}$ of N particles of equal weights $\frac{1}{N}$ at frame $k-1$, the algorithm goes as follow:

- **Propagation** of the particles according to (6), giving the new set: $\left\{(s_k^{'(i)}, \frac{1}{N})\right\}_{i=1..N}$.

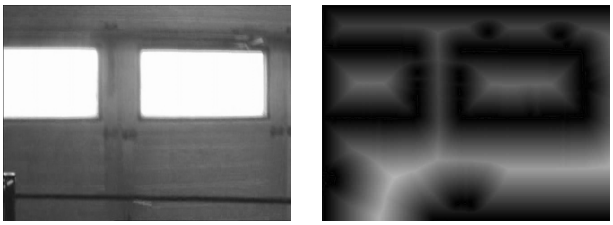

Fig. 6. Window frame (left) and its distance map (right). The darkest values correspond to the smallest distances.

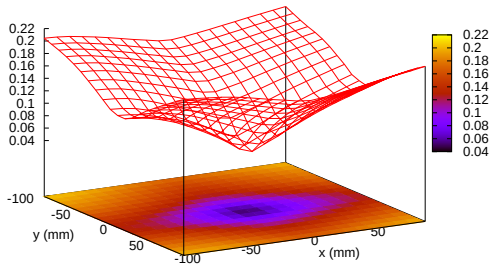

Fig. 7. Distance function with respect to x and y translations for the window frame. The zero position corresponds to the true camera pose.

- **Distance measurement** for each particle, computed from equation (11), to determine which particle to optimize.
- **Optimization of the best particles:** the multiple hypothesis tracker is applied from the camera poses corresponding to the best particles. One optimization can lead to several hypothesis. A set of optimized particles $\left\{(s'^{*(i)}_k, \frac{1}{N^*})\right\}_{i=1..N^*}$ is obtained.
- **Distance measurement** for the optimized particles.
- **Combination** of the particles $s'^{(i)}_k$ and $s'^{*(i)}_k$ to get a set $\left\{(s^{(i)}_k, \frac{1}{N+N^*})\right\}_{i=1,..N+N^*}$.
- **Weight computation** for each particle using a corrective term: $\pi^{(i)}_k \propto \frac{f_k(s^{(i)}_k)}{g_k(s^{(i)}_k)} p(\mathbf{z}_k \mid \mathbf{x}_k = s^{(i)}_k)$, with $\sum_{i=1}^{N+N^*} \pi^{(i)}_k = 1$. See III-D. It gives the set $\left\{(s^{(i)}_k, \pi^{(i)}_k)\right\}_{i=1..N+N^*}$
- **Estimation** of the tracking result as the weighted mean of the particle set (see III-B).
- **Resampling** by performing a weighted draw of N particles among the $N + N^*$ particles.

Experiments have been conducted on video sequences using this algorithm. They are described below.

F. Experimental results

The tracker presented in this paper was tested on different video sequences. Comparative results are presented in the figures 8 and 9. In the teabox sequence (figure 8), the classic registration method gets mistaken at some point and recovers at the end. Our tracker is successful during the whole sequence.

To underline the improvement in robustness brought by the particle filtering framework, the tracker was tested on a sequence taken from a UAV, with important frame to frame motion and occlusions. This sequence also presents great illumination changes. Results are shown in figure 9. The registration process alone fails when the occlusion is too important (figure 9 1-c). Embedded in a particle filtering framework, the window is tracked all along the sequence. Thanks to the optimization of some of the particles, a small number of particles is needed. For the window sequence of figure 9, only 100 particles were used. The complete video sequences of the tracking results are provided with this paper.

IV. CONCLUSIONS

Classic registration methods for model-based tracking are subject to wrong low-level matches, due to the similarity of different edges appearance. In this paper we presented a method to go from different low-level candidates to multiple hypothesis for the camera pose. This is achieved by performing several minimizations, corresponding to different sets of points. The relevant sets of points are obtained thanks to a k-mean like classification algorithm and usuallly the resulting number of minimizations does not exceed 4 or 5, which makes the algorithm suitable for real-time application. By selecting the pose giving the lowest minimization residue, we showed that the robustness of the resulting multiple hypothesis tracker was improved with respect to classic registration techniques. However, this bottom-up approach still presents some limitations. By selecting only one candidate per frame, the information given by the others is lost. Moreover, as for any registration method, the frame to frame motion needs to be small for the tracker to converge, and it would benefit from temporal filtering. To keep the advantages of our multiple hypothesis tracker, we proposed to embed it into a particle filtering framework. The multiple hypothesis tracker is applied to the best particles to move them to the local maxima of the likelihood function. The particle set is therefore guided toward the candidate poses emerging from the multiple hypothesis tracker. Although the state space is large, a small number of particles are needed. Experiments have been conducted on different video sequences, and the tracker succesfully performed in presence of ambiguities, large displacements, occlusions and illumination changes.

REFERENCES

[1] M. Armstrong and A. Zisserman. Robust object tracking. In *In Proc. Asian Conference on Computer Vision*, volume 1, pages 58–61, 1995.

[2] M. Bray, E. Koller-Meier, and L. Van Gool. Smart particle filtering for 3d hand tracking. *IEEE International Conference on Automatic Face and Gesture Recognition*, 1:675–680, 2004.

[3] A. Chiuso and S. Soatto. Monte-carlo filtering on lie groups. In *Proceedings of IEEE Conf. Decision and Control*, volume 1, pages 304–309, 2000.

[4] A.I. Comport, E. Marchand, M. Pressigout, and F. Chaumette. Real-time markerless tracking for augmented reality: The virtual visual servoing framework. *IEEE Transactions on visualization and computer graphics*, 12(4):615–28, July/August 2006.

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

[6] J.A. Hartigan. *clustering algorithms*. John Wiley & Sons, 1975.

[7] M. Isard and A. Blake. Icondensation: Unifying low-level and high-level tracking in stochastic framework. In *European Conference on Computer Vision (ECVV'98)*, volume 1, pages 893–908, 1998.

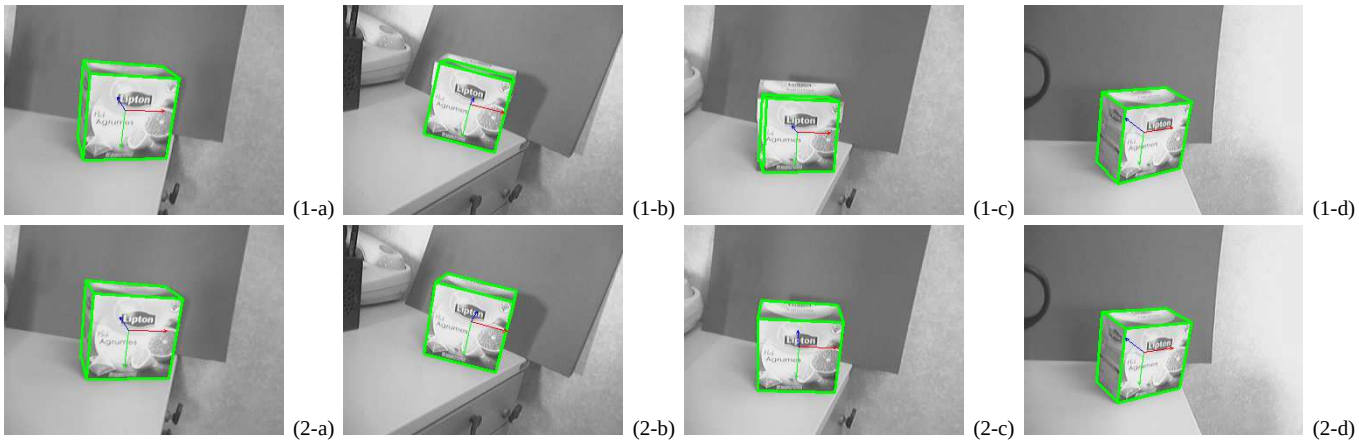

Fig. 8. Teabox sequence. The first row show the results from classic registration tracker. The multiple hypothesis tracker embedded in particle filtering framework (second row) tracks the object successfully.

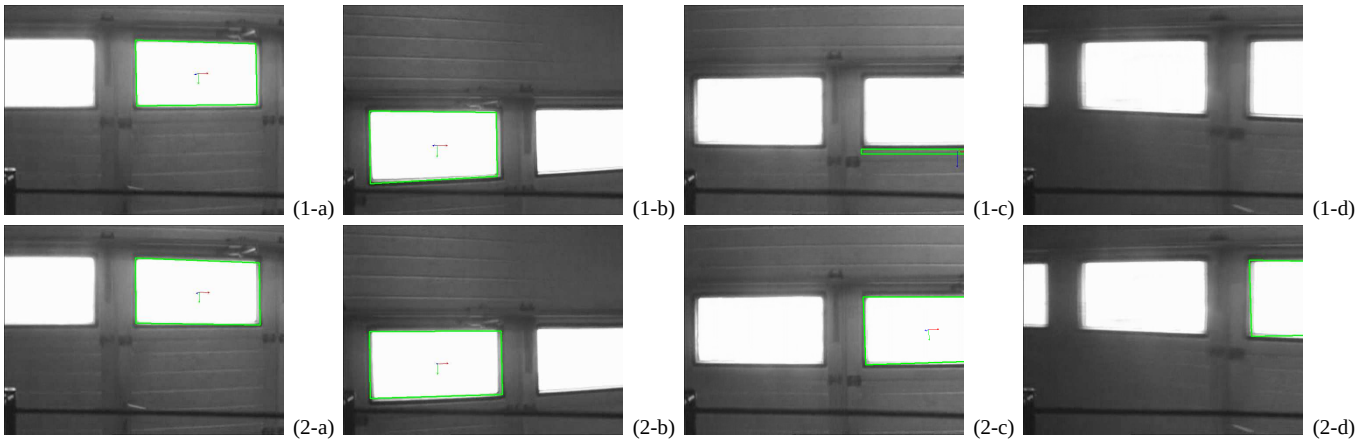

Fig. 9. Window sequence. The registration method alone (first row) fails when large occlusions occurr (1-c). The multiple hypothesis tracker embedded in particle filtering framework (second row) tracks the object successfully.

[8] G. Klein and T. Drummond. Tightly integrated sensor fusion for robust visual tracking. *Image and Vision Computing*, 22(10):769–776, September 2004.

[9] G. Klein and D. Murray. Full-3d edge tracking with a particle filter. In *British Machine Vision Conference*, volume 3, pages 1119–1128, 2006.

[10] J. Kwon, M. Choi, C. Chun, and F.C. Park. Particle filtering on the euclidean group. In *IEEE Int. Conf. on Robotics and Automation*, pages 3552–3557, 2007.

[11] D.G. Lowe. Fitting parameterized three-dimensional models to images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13:441–450, 1991.

[12] E. Malis and E. Marchand. Experiments with robust estimation techniques in real-time robot vision. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, IROS'06*, pages 223–228, Beijing, China, October 2006.

[13] M. Moakher. Means and averaging in the group of rotations. *SIAM journal on matrix analysis and applications*, 24:1–16, 2002.

[14] S. Nuske, J. Roberts, and G. Wyeth. Visual localisation in outdoor industrial building environments. In *IEEE Int. Conf. on Robotics and Automation*, pages 544–550, 2008.

[15] M. Pupilli and A. Calway. Real-time camera tracking using known 3d models and a particle filter. In *Int. Conference on Pattern Recognition*, pages 199–203, August 2006.

[16] L. Vacchetti, V. Lepetit, and P. Fua. Combining edge and texture information for real-time accurate 3d camera tracking. In *IEEE/ACM Int. Symp. on Mixed and Augmented Reality*, pages 48–57, 2004.