Sample-Based 3D Tracking of Colored Objects: A Flexible Architecture

M. Taiana, J. Nascimento, J. Gaspar and A. Bernardino
{mtajana,jan,jag,alex}@isr.ist.utl.pt

Abstract

This paper presents a method for 3D model-based tracking of colored objects using a sampling methodology. The problem is formulated in a Monte Carlo filtering approach, whereby the state of an object is represented by a set of hypotheses. The main originality of this work is an observation model consisting in the comparison of the color information in some sampling points around the target's hypothetical edges. On the contrary to existing approaches the method does not need to explicitly compute edges in the video stream, thus dealing well with optical or motion blur. The method does not require the projection of the full 3D object on the image, but just of some selected points around the target's boundaries. This allows a flexible and modular architecture illustrated by experiments performed with different objects (balls and boxes), camera models (perspective, catadioptric, dioptric) and tracking methodologies (particle and Kalman filtering).

1 Introduction

Many applications require the detection, tracking and pose estimation of known 3D objects. Particularly in robotics research, objects often have well known shapes and colors, thus requiring computer vision tools for color-based detection and tracking methods. For example, research in robot grasping and manipulation often assume objects with significantly saturated colors to simplify the foreground segmentation problem [12]. In this paper we blend 3D model-based tracking and color-based measurement models in a practical algorithm for tracking known colored objects.

3D model-based tracking methods have been classically addressed in a non-linear optimization framework [10]. A cost function expressing the mismatch between predicted and observed object points is locally minimized as a function of object's pose parameters, usually by linearizing the relation between state and measurements. These approaches often have limited convergence basins, requiring either small target motions of very precise prediction models. In this paper we overcome this problem by addressing the pose estimation and tracking problem in a Monte Carlo sampling framework [4]. The state of the target is represented by a set of weighted particles, whose weights (or likelihoods) are computed by projecting their features to the image and matching with the assumed model of the object. The method, therefore, does not require the linearization between the state and the measurements, allowing the design of a simple and modular tracking architecture.

Despite the well known advantages of Monte Carlo based methods in maintaining several alternative explanations of the data, not many works have proposed their use in

BMVC 2008 doi:10.5244/C.22.82

a 3D model-based context. This is probably due to the necessity for rendering the object pose in the image plane for each particle in the state space, which constitute a high computational cost. There are, though, a few examples of applications of particle filters in 3D model-based tracking. For instance [9] employs a particle filter [7] and an edge based measurement model for 3D model-based tracking of objects, but requires the explicit computation of edges in the incoming video frames and fast GPU computation for real-time image rendering. In [15], a particle filter was also used for the estimation of camera pose from the measurements of 4 edge junctions on a known planar surface. A local search for edges around the expected values must be performed at each time step.

Our work distinguishes from the above ones by the following facts: (i) our observation model does not require the explicit rendering of predicted images for each particle but just of a set of selected points around the object's visible surface, (ii) it does not require explicit edge detection on the incoming stream of images; and (iii) it uses the color model of the object to enrich and add robustness to the image measurements. The first two facts permit a faster evaluation of each particle's likelihood, even if the cameras projection is non-linear. The last two facts make the method more robust to blur either due to fast motions of the object or to optical defocus. Altogether, our model facilitates the application to arbitrary non-linear image projections. We illustrate this fact with fish-eye lens cameras and catadioptric systems [5], as well as with conventional (perspective) cameras.

The paper is organized as follows. In Section 2 we describe our method's architecture, divided in two major components: sensor and tracking modules. Section 2 details in particular the sensor module, namely how measurements are obtained from the images, how likelihoods are assigned to different pose hypothesis and how different camera projection models can be easily incorporated. Section 3 presents the tracking module. It fuses the information coming from the sensor with the current target's state using a given motion model, and proposes the target's hypotheses for evaluation by the sensor module in the next time step. In Section 4 we present some experimental results in realistic scenarios for tracking different types of objects with different camera projection models and tracking methods, illustrating the flexibility of the framework. In Section 5 we present the conclusions of the paper and directions for future work.

2 3D Tracking with Sample-Based Observations

This section presents the proposed architecture, designed to deal with several projection models and tracking methods, on a sampling based sensor paradigm. The important aspects of the framework will be described in the following paragraphs and include the description of the state representation, the observation model and tracking. A graphical description of the architecture is illustrated in Fig. 1.

State Representation: The state vector of the target, denoted as $\mathbf{X}_t$, contains its 3D pose and derivatives up to a desired order. It is represented by a set of state hypotheses, or particles, $\mathbf{X}_t^{(i)}$, $i \in \{1,\ldots,N_p\}$, with N_p the number of particles. This set is generated as an approximation to the state *a priori* density function $p(\mathbf{X}_t|\mathbf{y}_{1:t-1})$, computed from the observation $\mathbf{y}$ based on color histograms in the previous time step or at initialization time.

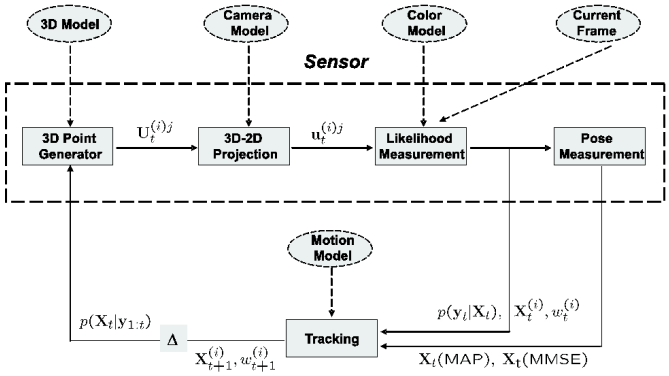

Figure 1: The proposed architecture.

Sensor Module: Since Harris proposed the RAPiD tracking system [6], most 3D model based tracking systems compute image edges as the 2D features to support the pose estimation procedure. Instead, we propose a measurement model based on sampling just some selected points in the images. These points are the projections of 3D points suggested by the pose hypotheses generated in the tracking module. This avoids time consuming edge detection processing or rendering the full object model in the image plane. Also, it facilitates the utilization of non-linear projection models, since only projection of points is required. The sensor model is composed by a chain of four modules (Fig. 1).

3D Point Generator – From the 3D pose hypotheses provided as input to the sensor module, we determine points around the object edges. The idea is that the color and luminance difference around the object edges is an indicator of pose hypothesis likelihood. We consider two different object shapes: spheres and convex polyhedral objects. Notwithstanding the model can be easily extended to general polyhedral objects by exploiting the current knowledge in computer graphics [9]. For each state hypothesis $\mathbf{X}_t^{(i)}$, and given the particular 3D geometric object model, a set of 3D points is generated $\mathbf{U}_t^{(i)j}$, where t is the time step, i is the particle index, and j indexes the points in the objects contour vicinity.

In the case of polyhedra, we use a 3D model that consists of a collection of faces and edges. To each couple (face,edge) we associate a set of 3D points that lie on the specific face, near the edge (Fig. 2.a). To each edge we associate a set of points that lie on the corresponding edge of an expanded polyhedron (Fig. 2.d). The expanded polyhedron is built multiplying by a constant the vectors that join the center of volume of the original polyhedra to its vertices. The 3D points of this model are used to define the areas of the image where the color is sampled in order to build color histograms. This is done by roto-translating the model and then projecting the 3D points onto the image.

For spheres, we define two sets of 3D points that when projected onto the image fall on the internal and external boundary of the sphere's silhouette. These 3D points lie on the intersection between the plane orthogonal to the line connecting the projection center to the center of the sphere and two spherical surfaces, one with a radius smaller than that of the tracked sphere, the other with a radius greater than that.

Image Projection – This module converts 3D point coordinates $\mathbf{U}_t^{(i)j}$ to corresponding

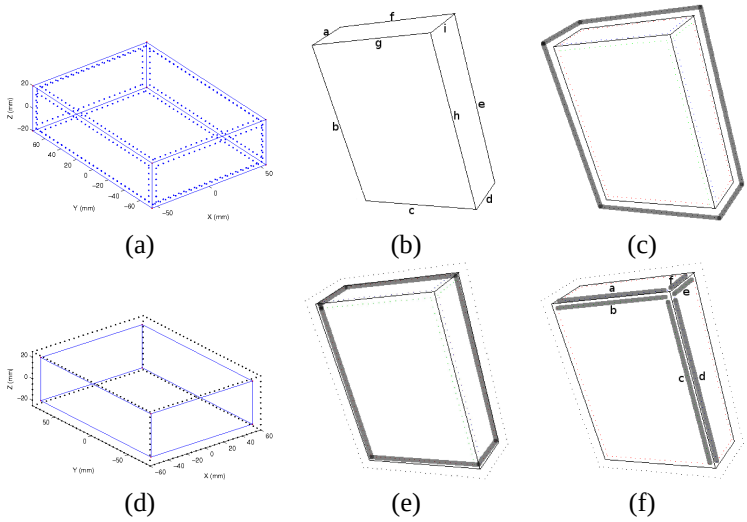

Figure 2: Polyhedron 3D model. (a) and (d) show the location of the 3D points of the model. (b) shows one particular projection of the object in which edges a, b, c, d, e and f form the silhouette of the projected figure, while g, i and h are non-silhouette edges. (c), (e) shows the areas sampled in the image to build he internal and external histograms. (f) shows the couples of areas associated with non-silhouette edges.

projections in the image plane $\mathbf{u}_t^{(i)j}$. One of the advantages of our approach is that any projection model can be employed. In this work we use the following types of cameras: (i) a catadioptric camera modeled by the unified projection model [5]; (ii) a camera with a fish-eye lens modeled by constant angular resolution; and (iii) and a standard camera modeled by the perspective projection model. Defining ρ as the radial distance in the image sensor, $\rho = |\mathbf{u}_t^{(i)j}|$, the above projection models are respectively:

$$\rho = \frac{l+m}{l\sqrt{r^2+z^2}-z}\cdot r \quad \text{or} \quad \rho = f\cdot atan(r/z) \quad \text{or} \quad \rho = k\cdot r/z \tag{1}$$

where l, m, k and f are parameters, and z, r are the cylindrical coordinates of $\mathbf{U}_t^{(i)j}$.

Likelihood Measurement – The 2D points coordinates generated by the previous process are sampled in the current image, and their photometric information is used to obtain each particle's likelihood $w_t^{(i)}$. This approximates the state *posterior* probability density function, represented by the set of weighted particles $\mathbf{X}_t^{(i)}, w_t^{(i)}$. For color modeling we use independent normalized histograms in the HSI color space, which decouples the intensity (I) from color (H and S). We denote by $\mathbf{h}_{ref}^c = (h_{1,ref}^c, \ldots, h_{B,ref}^c)$ the B-bin reference (object) histogram model in channel $c \in \{H, S, I\}$. An estimate for the histogram color model, denoted by $\mathbf{h}_{\mathbf{x}}^c = (h_{1,\mathbf{x}}^c, \ldots, h_{B,\mathbf{x}}^c)$, can be obtained as

$$h_{i,\mathbf{x}}^c = \beta \sum_{\mathbf{u}\in\mathscr{U}} \delta_a(b_{\mathbf{u}}^c), \qquad i = 1, \ldots, B. \tag{2}$$

$\mathscr{U}$ is the region where the histogram is computed; $b_{\mathbf{u}}^c \in \{1, \ldots, B\}$ denotes the histogram

bin index associated with the intensity at pixel location $\mathbf{u}$ in channel c; δ_a is a Kronecker delta function at a; and β is a normalization constant such that $\sum_{i=1}^{B} h^c_{i,\mathbf{x}} = 1$.

We define h^{model}, h^{in} and h^{out} as a reference (object) color model, the inner boundary points and the outer boundary points histogram, respectively. We define h_i^{sideA} and h_i^{sideB} as the histograms of each of the two sides of the i^{th} non-silhouette edge (see Fig.2f). To measure the difference between histograms we use the Bhattacharyya similarity as in [3,13]:

$$\mathscr{S}(\mathbf{h}_1, \mathbf{h}_2) = \sum_{i=1}^{B} \sqrt{h_{i,1} h_{i,2}} \tag{3}$$

The resulting distance used herein is

$$\mathscr{D} = (1 - \frac{1 - \mathscr{S}_0 + \kappa_1(1 - \mathscr{S}_1) + \kappa_2(1 - \mathscr{S}_2)}{\kappa_1 + \kappa_2 + 1}) - \gamma \tag{4}$$

where γ is a coefficient that modulates the distance based on the number of projected 3D points that fall onto the image, $\gamma = \ln(\frac{\textit{used points ratio}}{\varepsilon})$ and

$$\mathscr{S}_0 = \mathscr{S}(h^{\text{model}}, h^{\text{in}}), \quad \mathscr{S}_1 = \mathscr{S}(h^{\text{out}}, h^{\text{in}}), \quad \mathscr{S}_2 = \frac{\sum_{i=0}^{n} \mathscr{S}(h_i^{\text{sideA}}, h_i^{\text{sideB}})}{n} \tag{5}$$

are respectively, the object-to-model, object-to-background and mean-side-to-side (non-silhouette edges) similarities.

The rationale for this definition is that the distance metric should be high when candidate color histograms are different from the reference histogram and similar to the background. The distance metric should also be high when there is little or no difference in color on the sides of non-silhouette edges. Parameters κ_1 and κ_2 allow to balance the influence of the different contributions. They were set to 1.5 and 0.6 respectively, for the case of the polyhedron while for the sphere they were set to 1.5 and 0. The data likelihood function $\mathscr{L}$ is modeled as a Laplacian distribution over the distance metric: $p(\mathbf{y}_t \mid \mathbf{X}_t^{(i)}) \propto e^{-\frac{|\mathscr{D}|}{\varepsilon}}$. In our experiments we set $\varepsilon = 1/30$.

Pose Measurement – Despite it is more informative to have a description of the state as a weighted set of particles, for some purposes it may be useful to condensate this information in a single measurement. We take either the best weighted particle, i.e. *maximum a posteriori* (MAP) or the average of the most significant ones, that is, *minimum mean square error* (MMSE). We term this process a particle fusion, which allows us to obtain a single observation, that will be used to maintain the unimodality (Gaussian assumption) needed for the linear propagation, e.g. in the Kalman filter. Also, for visualization and evaluation purposes, it is often convenient to consider a single value representative of the system state.

Tracking Module: We use classical Bayesian inference to propagate the state information between two consecutive time steps. This will be detailed in Section 3.

3 The Tracking Module

In this work several trackers are possible to be used. In this paper we use two filtering techniques: (i) particle filtering (PF), and (ii) Kalman filtering (KF). In following we detail each one of the filtering techniques, and the common motion model.

3D particle based tracker: In this section we describe the method used for 3D target tracking with particle filters (see Fig. 3). We are interested in computing, at each time $t \in N$, an estimate of the 3D pose of the target. We represent the hidden state of the object ("state-vector") defined by a random variable $\mathbf{X}_t \in \mathbb{R}^{n_\mathbf{x}}$ whose distribution is unknown (non-Gaussian); $n_\mathbf{x}$ is the dimension of the state vector. The state sequence $\{\mathbf{X}_t; t \in \mathbb{N}\}$ represents the state evolution along time and is assumed to be an unobserved Markov process with some initial distribution $p(\mathbf{X}_0)$ and a transition distribution $p(\mathbf{X}_t|\mathbf{X}_{t-1})$.

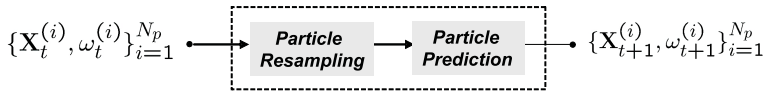

Figure 3: Modules of the Particle filter tracker.

The observations are represented by the random variable $\{\mathbf{y}_t; t \in \mathbb{N}\}$, $\mathbf{y}_t \in \mathbb{R}^{n_\mathbf{y}}$. For tracking, the distribution of interest is the posterior $p(\mathbf{X}_t|\mathbf{y}_{1:t})$, i.e., the filtering distribution, where $\mathbf{y}_{1:t} = (\mathbf{y}_1, \ldots, \mathbf{y}_t)$ are the observations collected up to time instant t. In Bayesian sequential estimation, the posteriori $p(\mathbf{X}_t|\mathbf{y}_{1:t})$ is computed following the two step recursion

$$\begin{aligned} \textit{prediction} \quad & p(\mathbf{X}_t|\mathbf{y}_{1:t-1}) = \int p(\mathbf{X}_t|\mathbf{X}_{t-1}) p(\mathbf{X}_{t-1}|\mathbf{y}_{1:t-1}) d\mathbf{X}_{t-1} \\ \textit{filtering} \quad & p(\mathbf{X}_t|\mathbf{y}_{1:t}) \propto p(\mathbf{y}_t|\mathbf{X}_t) p(\mathbf{X}_t|\mathbf{y}_{1:t-1}) \end{aligned} \tag{6}$$

This recursion requires the specification of the motion model $p(\mathbf{X}_t|\mathbf{X}_{t-1})$ and likelihood model $p(\mathbf{y}_t|\mathbf{X}_t)$ along with the assumptions $\mathbf{X}_t \perp \mathbf{y}_{1:t-1}|\mathbf{X}_{t-1}$ and $\mathbf{y}_t \perp \mathbf{y}_{1:t-1}|\mathbf{X}_t$, where $\perp$ stands for independence. A more rigorous and broad description of Monte Carlo probabilistic estimators can be found in [1,11]. The basic idea is simple, starting from a weighted set of samples $\{\mathbf{X}_{t-1}^{(i)}, w_{t-1}^{(i)}\}_{i=1}^{N}$ approximately distributed according to $p(\mathbf{X}_{t-1}|\mathbf{y}_{1:t-1})$, new samples are generated from a chosen proposal distribution $q(.)$. In this work the proposal follows the state dynamics, and the new set of weighted particles, $\{\mathbf{X}_t^{(i)}, w_t^{(i)}\}$, is approximately distributed according to $p(\mathbf{X}_t|\mathbf{y}_{1:t})$, i.e.

$$w_t^{(i)} \propto w_{t-1}^{(i)} \frac{p(\mathbf{y}_t|\mathbf{X}_t^{(i)}) p(\mathbf{X}_t^{(i)}|\mathbf{X}_{t-1}^{(i)})}{q(\mathbf{X}_t^{(i)}|\mathbf{X}_{t-1}^{(i)}, \mathbf{y}_t)} \longrightarrow w_{t-1}^{(i)} p(\mathbf{y}_t|\mathbf{X}_t^{(i)}) \tag{7}$$

3D Kalman based tracker: The motivation for the Kalman based tracker (see Fig. 4) comes from control applications where it is required to have a smooth state estimation in each frame. Using PF's the state estimation is usually taken to be the mean or the median of the particles, which are not particularly accurate estimates [10]. This can result in motion estimates that are not as smooth as desired, and thus are not suitable for the generation of control signals, leading to saturation or the so called "bang-bang" effect.

The KF consists in recursive computation of the mean and covariance as follows (see (6))

$$\begin{aligned} \textit{prediction}: \quad & \hat{\mathbf{X}}_t^- = \mathbf{A}\hat{\mathbf{X}}_{t-1} & \qquad \textit{filtering}: \quad & \hat{\mathbf{X}}_t = \hat{\mathbf{X}}_t^- + \mathbf{K}_t(\mathbf{X}_t^M - \mathbf{H}_t\hat{\mathbf{X}}_t^-) \\ & \mathbf{P}_t^- = \mathbf{A}\mathbf{P}_{t-1}\mathbf{A}^T + \mathbf{Q} & & \mathbf{P}_t = (\mathbf{I} - \mathbf{K}_t\mathbf{H}_t)\mathbf{P}_t^- \end{aligned} \tag{8}$$

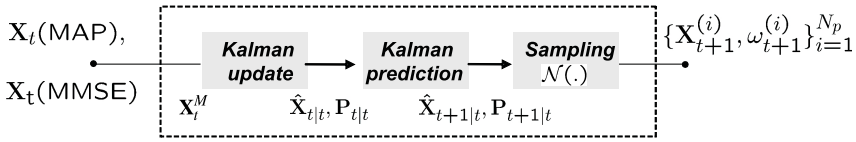

Figure 4: Modules of the Kalman filter tracker.

where $\hat{\mathbf{X}}_t, \mathbf{P}_t$ are the mean and covariance matrices; $\mathbf{A}, \mathbf{Q}$ are the system dynamics and its uncertainty; $\mathbf{H}, \mathbf{K}_t = \mathbf{P}_t^- \mathbf{H}_t^T (\mathbf{H}_t \mathbf{P}_t^- \mathbf{H}_t^T + \mathbf{R})^{-1}$ are the observation matrix and Kalman gain; $\mathbf{X_t^M}$ is the state measurement computed in the sensor model, X(MAP) or X(MMSE). The noise covariance $\mathbf{R}$ is assumed to be constant, although it can be be computed at each time step.

Finally, the predicted state $\hat{\mathbf{X}}_{t+1|t}$ is spread in a number of N_p particles following the uncertainty of the predicted value of the covariance matrix. This is the main difference in our tracking module regarding a classical Kalman filter, and is required due to our particular sample-based sensor module. This is similar to the Unscented Kalman Filtering [8], in the sense that that filter computes the covariance by propagating a set of deterministically chosen points, i.e., *sigma points*, chosen in such a way that their sample mean and sample covariance correspond to the mean and the covariance of the Gaussian being estimated. However, in our case these points do not suffice to cope with the pdf of the likelihood which is non Gaussian and largely multi-modal.

Motion model: In order to accommodate to any real object motion, we use a Gaussian distributed one, giving no privilege to any direction of motion

$$p(\mathbf{X}_t|\mathbf{X}_{t-1}) = \mathcal{N}(\mathbf{X}_t|\mathbf{X}_{t-1}, \Lambda) \tag{9}$$

where $\mathcal{N}(.|\mu, \Sigma)$ holds for Gaussian distribution with mean μ and covariance Σ, and Λ stands for the diagonal matrix with the variances for random walk models on the components of the object state model. This approach has widely been used (e.g. [2,14]).

In this work two cases are addressed: (i) spherical object and (ii) polyhedral object. For the first case, the state vector consists of the 3D Cartesian position and linear velocities of the ball, $\mathbf{X}_t = [x\ y\ z\ \dot{x}\ \dot{y}\ \dot{z}]^T$. The motion is modeled by a constant velocity model, i.e. the motion equations correspond to a uniform acceleration during one sample time:

$$\mathbf{X}_{t+1} = \begin{bmatrix} I & (\Delta t)I \\ \mathbf{0} & I \end{bmatrix} \mathbf{X}_t + \begin{bmatrix} (\frac{\Delta t^2}{2})I \\ (\Delta t)I \end{bmatrix} a_t \tag{10}$$

where I is the 3×3 identity matrix, $\Delta t = 1$, and a_t is a 3×1 white zero mean random vector corresponding to an acceleration disturbance. The covariance matrix of the random acceleration vector is usually set experimentally as $cov(a_t) = \sigma^2 I$.

For the polyhedral object, the state vector is $\mathbf{X}_t = [\mathbf{p}_t;\ \mathbf{q}_t]$ where $\mathbf{p}_t = [x\ y\ z]^T$ denotes the position of the mass-center of the object and $\mathbf{q}_t = [q_w\ q_x\ q_y\ q_z]^T$ is a quaternion representing the object orientation. To model the dynamics, in this case we use a constant pose model, $\mathbf{p}_{t+1} = \mathbf{p}_t + \eta_p$ and $\mathbf{q}_{t+1} = \mathbf{q}_t * \eta_q$, where $*$ stands for quaternion product, η_p is Gaussian noise and η_q is a quaternion generated by sampling Euler angles from a Gaussian distribution.

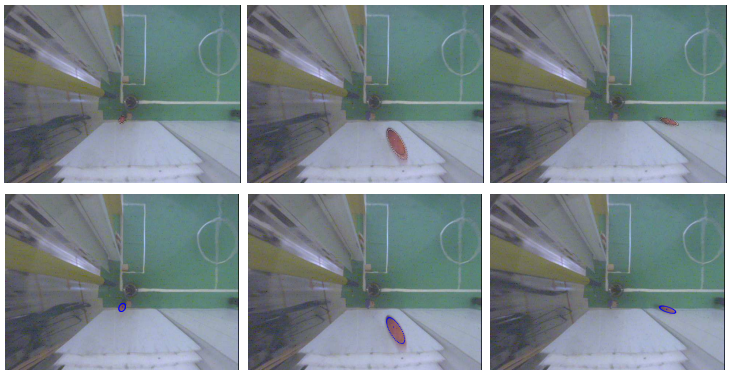

Figure 5: Tracking a jumping ball in a catadioptric sensor. Top: Particle Filter. Bottom: Kalman filter. Frames: 113, 122, 146.

4 Experimental Results

We have conducted an extensive evaluation under the proposed framework in realistic scenarios. We illustrate the performance of the methods in three experiments using in each one a distinct camera [1]: (i) catadioptric, (ii) fisheye (constant angular resolution), and (iii) perspective.

In the **first experiment** we illustrate the performance of the tracker on images acquired with the catadioptric vision system. This sensor is composed by a perspective camera looking upright to a convex mirror, providing omnidirectional view in the azimuth direction and an orthographic view of the ground plane. This type of systems is characterized by strong geometrical distortions and optical blur (see Fig.5). The tracked object is a ball on a typical RoboCup Middle Size League (MSL) scenario. We use the constant velocity motion model defined in (10). The covariance matrix of the random acceleration vector was experimentally tuned to $cov(a_t) = \sigma^2 I$, with $\sigma = 120$mm/frame^2. Fig. 5 shows some snapshots of the sequence, illustrating the tracking results with both the Kalman and the Particle Filters.

In the **second experiment** we use a fisheye vision system (a dioptric camera coupled with a fisheye lens) to track a ball in the RoboCup MSL scenario. Figure 6 shows the tracking performance under partial occlusion and the case where the target passes close to another identical ball. The motion model is the same as the first experiment. It is interesting to note that the Kalman filter, as compared to the Particle filter with MAP/MMSE pose estimation, takes advantage of the motion model to return a more centered estimate of the position in the cases of large occlusions or confusion to nearby objects (see cols. 4 and 6 of Fig. 6; the MAP/MMSE are shown by the black/white dots).

In the **third experiment** we use an off-the-shelf web camera to track a box being manipulated by the user. We use the constant pose motion model defined before. The standard deviation for the positional noise was set to 15 mm/frame, while that of the rotational noise was set to 0.1 rad/frame, in any direction. The results are presented in Fig. 7 where we can observe good performance even in the case of partial occlusion.

[1] The results can be also evaluated through the accompanying videos.

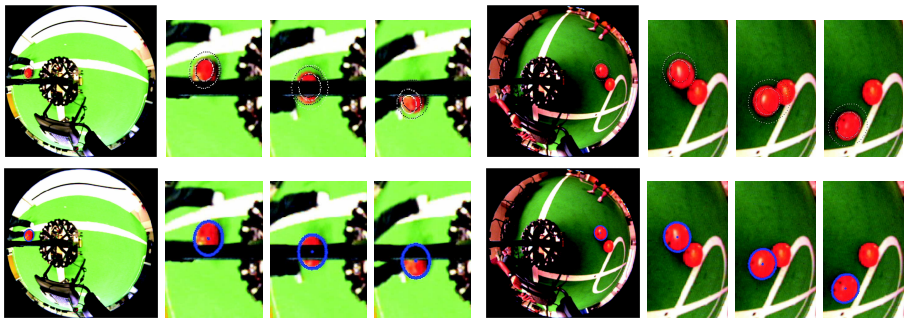

Figure 6: Ball tracking under occlusion (cols. 1..4) and having two close targets (cols. 5..8) in a fisheye lens camera. Columns 1 and 5 are full-size images (the other ones just show the region of interest). Top/bottom compare Particle vs Kalman filters.

Figure 7: Tracking a box with a perspective camera. Top: Particle Filter. Bottom: Kalman filter. Frames: 11, 207, 291.

5 Conclusions

We have presented a novel observation model for 3D model based tracking. The method uses color features thus avoiding both explicit edge extraction from images and full object appearance rendering. Rather it is based on the computation and comparison of color histograms obtained from a sparse set of points in the images, arising from likely target's posture hypotheses. Only sparse point rendering (projection) is required, which facilitates the utilization of arbitrary linear or non-linear camera projection models. Also, being a method based on sampled hypotheses, it fits nicely with the particle filtering paradigm. Anyway, it can also be used with more classical methods, like the Kalman filter, whenever Gaussianity is a reasonable assumption to rely on.

In future work we will look at variations of the observation model to improve likelihood estimation, e.g. devising better ways to model the luminance changes induces by orientation changes in the target's surfaces. Also, we will look at ways to reduce the number of used particles. This may be related to varying distance of the sampling points

from the boundaries of the targets. Longer distances may provide smoother likelihood estimates, and an annealed particle filtering methodology might be designed.

Acknowledgments

This work was supported by the European Commission, Project IST-004370 RobotCub, and by the Portuguese Government - Fundação para a Ciência e Tecnologia (ISR/IST plurianual funding) through the POS_Conhecimento Program that includes FEDER funds, and through project BIO-LOOK, PTDC / EEA-ACR / 71032 / 2006. We would like to thank Dr. Luis Montesano and Dr. Alessio Del Bue for the fruitful discussions.

References

[1] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE TIP*, 50(2):174–188, 2002.

[2] P. Barrera, J. M. Cans, and V. Matelln. Visual object tracking in 3d with color based particle filter. In *Proc. of world academy of science, Eng. and Tech.*, volume 4, 2005.

[3] D. Comaniciu, V. Ramesh, and P. Meer. Real-time tracking of non-rigid objects using mean shift. In *Proc. Conf. on Comp. Vision Pattern Rec.*, volume 2, pages 142–149, 2000.

[4] A. Doucet, N. de Freitas, and N. Gordon. *Sequential Monte Carlo Methods In Practice*. Gordon editors, Springer Verlag, 2001.

[5] C. Geyer and K. Daniilidis. A unifying theory for central panoramic systems and practical applications. In *IEEE CVPR*, pages 445–461. 2000.

[6] C. Harris. Tracking with rigid models. In *Active Vision*, chapter 4, pages 59–73. 1992.

[7] M. Isard and A. Blake. Condensation: conditional density propagation for visual tracking. *Int. Journal of Computer Vision*, 28(1):5–28, 1998.

[8] S. Julier and J. Uhlmann. A new extension of the kalman filter to non-linear systems. In *Proc. of Aerosense*, 1997.

[9] G. Klein and D. Murray. Full-3d edge tracking with a particle filter. In *Proc. of BMVC 2006*, page III:1119, Edinburgh, Scotland, 2006.

[10] V. Lepetit and P. Fua. Monocular model-based 3d tracking of rigid objects: A survey. *Foundations and Trends in Computer Graphics and Vision*, 1(1):1–89, 2005.

[11] D. MacKay. Introduction to Monte Carlo methods. In M. I. Jordan, editor, *Learning in Graphical Models*, pages 175–204. Kluwer Academic Press, 1998.

[12] L. Montesano, M. Lopes, A. Bernardino, and J. Santos-Victor. Learning object affordances: From sensory motor maps to imitation. *IEEE Transactions on Robotics, Special Issue on Bio-Robotics*, 24(1):15–26, 2008.

[13] K. Okuma, A. Taleghani, N. de Freitas, J. J. Little, and D. G. Lowe. A boosted particle filter: Multitarget detection and tracking. In *Proc. ECCV*, pages 28–39, 2004.

[14] P. Pérez, J. Vermaak, and A. Blake. Data fusion for visual tracking with particles. *Proc. of the IEEE*, 92(3):495–513, 2004.

[15] M. Pupilli and A. Calway. Real-time camera tracking using known 3d models and a particle filter. In *Proc. of ICPR (1)*, pages 199–203, Hong Kong, 2006.