3D tracking by Catadioptric Vision Based on Particle Filters. ⋆

Matteo Taiana[1,2], José Gaspar[1], Jacinto Nascimento[1], Alexandre Bernardino[1], and Pedro Lima[1]

[1] IST, Instituto de Sistemas e Robótica – Lisboa, Portugal
{mtajana,jag,jan,alex,pal}@isr.ist.utl.pt
[2] Politecnico di Milano, Italy

Abstract. This paper presents a robust tracking system for autonomous robots equipped with omnidirectional cameras. The proposed method uses a 3D shape and color-based object model. This allows to tackle difficulties that arise when the tracked object is placed above the ground plane floor. Tracking under these conditions has two major difficulties: first, observation with omnidirectional sensors largely deforms the target's shape; second, the object of interest embedded in a dynamic scenario may suffer from occlusion, overlap and ambiguities. To surmount these difficulties, we use a *3D particle filter* to represent the target's state space: position and velocity with respect to the robot. To compute the likelihood of each particle the following features are taken into account: *i*) image color; *ii*) mismatch between target's color and background color. We test the accuracy of the algorithm in a RoboCup Middle Size League scenario, both with static and moving targets.

1 Introduction

In order to carry out complex tasks (e.g. playing football) robots need to extract sufficient information from the environment they operate in. Catadioptric sensors are widely used in robotics, especially for self localization and navigation [8],[1], as they gather information from a large portion of the space surrounding a robot. One drawback is that images are affected by strong distortion and perspective effects, which may force the use of non-standard algorithms for target detection and tracking.

Automated tracking is still an open problem, e.g., surveillance applications [2], sports [5,10] or smart rooms [6]. In general, tracking visual features in complex and cluttered environments is fraught with uncertainty. It is therefore crucial to adopt principled probabilistic models. Over the past few years, particle filters, also known as sequential Monte Carlo (MC), proved to be effective in image processing tracking techniques, e.g., [11,12,13]. The strength of these methods lies in their simplicity and flexibility on nonlinear and non-Gaussian settings [7].

⋆ This work was supported by Fundação para a Ciência e a Tecnologia (ISR/IST pluriannual funding) through the POS-Conhecimento Program that includes FEDER funds. We would like to thank Dr. Luis Montesano and Dr. Alessio Del Bue for the helpful discussions.

We use a 3D particle-filter [9,11] tracker in which the hypotheses are 3D positions and velocities of the object, and whose likelihood is a function of object color and shape. From one image frame to the next, the hypotheses are moved according to an appropriate motion model. Then, for each particle, a likelihood is computed, in order to estimate the object state. To calculate the likelihood of a particle we first project the contour of the object it represents on the image plane (as a function of the object 3D shape, position and orientation) using an approximated model for the catadioptric system, the Unified Projection Model [15]. The likelihood is then calculated as a function of three color histograms: one represents the object color model and is computed in a training phase with several examples taken from distinct locations and illumination conditions; the other two histograms represent the inner and outer boundaries of the projected contour, and are computed at every frame for all particles. The idea is to assign a high likelihood to the contours for which the inner pixels have a color similar to the object, and are sufficiently distinct from outside ones.

A work closely related to this is described in [14], although in that case the tracking of RoboCup Middle Size League (MSL) balls is accomplished on the image plane. Tracking the 3D trajectory of a ball has become relevant in the RoboCup MSL scenario, as robots are now provided with the ability to kick the ball off the ground. Tracking the position of an object in 3D space instead of on the image plane has two main advantages: (i) the motion model used by the tracker can be the actual motion model of the object, while in image tracking the motion model should describe movements of the projection of the object on the image plane and, because of the aforementioned distortion, a good model can be difficult to formulate and use; (ii) with 3D tracking the actual position of the tracked object is directly available, while a further non-trivial step is needed for a system based on an image tracker to provide it.

The paper is organized as follows. In Section 2 we describe the catadioptric sensor and the used projection model. The particle filter is described in Section 3, and customized to our particular problem in 4. The experimental results are shown in Section 5 and, finally, Section 6 concludes the paper and presents ideas for future work.

2 Catadioptric Imaging System

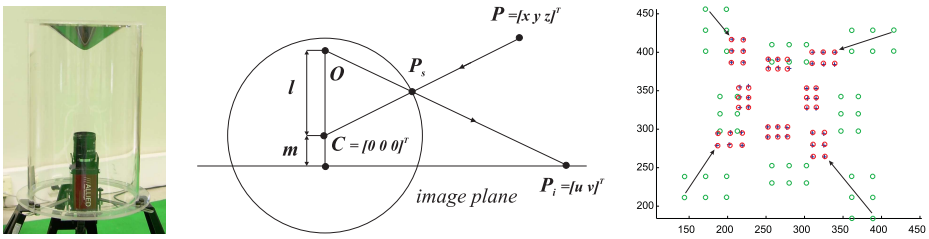
In this section we describe the imaging system, its projection model and the used calibration method. Our catadioptric vision system, see Fig.1a, combines a camera looking upright to a convex mirror, having omnidirectional view in the azimuth direction [16]. The system is designed to have a wide-angle and a constant-resolution view of the ground plane [17,18]. The system has the constant-resolution property at one reference plane, the ground plane, and has only approximately constant-resolution at planes parallel to the reference one. As compared to perspective cameras, the constant-resolution design is a good compromise between approximating ubiquitous constant-resolution and enlarging the field of view. Note that perspective cameras can have constant-resolution

for all planes orthogonal to the optical axis only for narrow view fields. Large fields-of-view imply using small focal length lenses which introduce large radial distortions.

Let the projection model, $\mathcal{P}$ of the constant-resolution system represent the transformation of a 3D point, $[X\ Y\ Z]^T$ into the 2D coordinates of its projection on the image plane, $[u\ v]^T$, considering the parameters θ:

$$[u\ v]^T = \mathcal{P}\left([X\ Y\ Z]^T; \theta\right). \tag{1}$$

$\mathcal{P}$ is trivial for the ground-plane, as it is just a scale factor between pixels and meters. Deriving $\mathcal{P}$ for the complete 3D field of view is complex as it involves using the actual mirror shape [18]. Here we assume that the system approximates a single projection center system, considering that the mirror size is small when compared to the distances to the imaged-objects. Hence, we can use a standard model for catadioptric omnidirectional cameras, namely the Unified Projection Model (UPM) pioneered by Geyer and Daniilidis [15].



(a) (b) (c)

(d) (e) (f)

Fig. 1. (a) Catadioptric camera. (b) The Unified Projection Model. (c) Calibration result: observed image points (blue crosses), 3D points projected using initial projection parameters (green circles) and using calibrated parameters (red circles) - the arrows show the calibration effect at four points. (d) The OmniISocRob robotic platform. (e) Image used for calibration. (f) Sample image taken in a RoboCup MSL scenario.

The UPM represents all omnidirectional cameras with a single center of projection [15]. It is simpler than the model which takes into account the actual shape of the mirror and gives good enough approximations for our purposes. The model consists of a two-step mapping via a unit-radius sphere: (i) project a

3D world point, $P = [x\ y\ z]^T$ to a point P_s on the sphere surface, such that the projection is normal to the sphere surface; (ii) project to a point on the image plane, $P_i = [u\ v]^T$ from a point, O on the vertical axis of the sphere, through the point P_s. This mapping is graphically illustrated in Fig.1b. The mapping is mathematically defined by:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \frac{l+m}{l\sqrt{x^2+y^2+z^2}-z} \begin{bmatrix} s_u & 0 \\ 0 & s_v \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} u_0 \\ v_0 \end{bmatrix} \tag{2}$$

where (l, m) parameters describe the type of camera, (s_u, s_v, u_0, v_0) represent pixel scaling and offsetting in the image plane, and $[x\ y\ z]^T$ is a 3D point in the camera coordinate system, whose relationship to world coordinates is given by the 3D rigid transformation, $[x\ y\ z]^T = R[X\ Y\ Z]^T + [x_0\ y_0\ z_0]^T$.

To calibrate the model we use a set of known non-coplanar 3D points $[X_i\ Y_i\ Z_i]^T$ and measure their images $[u_i\ v_i]^T$. Then, we minimize the mean squared error between the measurements and the projection with the parametric model $\mathcal{P}$:

$$\theta^* = \arg_\theta \min \sum_i \left\| [u_i\ v_i]^T - \mathcal{P}\left([X_i\ Y_i\ Z_i]^T; \theta\right) \right\|^2 \tag{3}$$

where θ contains the 3D rigid transformation from world to camera coordinates, pixels scaling and offsetting, and the camera type parameters (l, m). We set the calibration patterns coordinate system in accordance with the robot frame by aligning the patterns with the center of the robot, see Fig.1e.

3 3D Tracking with Particle Filters

In this section we introduce the methods employed for 3D target tracking with particle filters. We are interested in computing, at each time $t \in \mathbb{N}$, an estimate of the 3D pose of a target. We represent this information as a "state-vector" defined by a random variable $\mathbf{x}_t \in \mathbb{R}^{n_\mathbf{X}}$ whose distribution in unknown (non-Gaussian); $n_\mathbf{X}$ is the dimension of the state vector. In the present work we are mostly interested in tracking balls and cylindrical robots, whose orientation is not important for tracking. However, the formulation is general and can easily incorporate other dimensions in the state-vector, e.g. target orientation and spin.

Let $\mathbf{x}_t = [x, y, z, \dot{x}, \dot{y}, \dot{z}]^T$, with (x,y,z), $(\dot{x},\dot{y},\dot{z})$ the 3D cartesian position and linear velocities in a robot centered coordinate system. 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 $p(\mathbf{x}_t \mid \mathbf{x}_{t-1})$.

The observations taken from the images are represented by the random variable $\{\mathbf{y}_t; t \in \mathbb{N}\}$, $\mathbf{y}_t \in \mathbb{R}^{n_\mathbf{Y}}$, and are assumed to be conditionally independent given the process $\{\mathbf{x}_t; t \in \mathbb{N}\}$ with marginal distribution $p(\mathbf{y}_t \mid \mathbf{x}_t)$, where $n_\mathbf{Y}$ is the dimension of the observation vector.

In a statistical setting, the problem is posed as the estimation of the *posteriori* distribution of the state given all observations $p(\mathbf{x}_t \mid \mathbf{y}_{1:t})$. Under the Markov

assumption, we have:

$$p(\mathbf{x}_t \mid \mathbf{y}_{1:t}) \propto p(\mathbf{y}_t \mid \mathbf{x}_t) \int p(\mathbf{x}_t \mid \mathbf{x}_{t-1})\, p(\mathbf{x}_{t-1} \mid \mathbf{y}_{1:t-1}) d\mathbf{x}_{t-1} \tag{4}$$

The previous expression tells us that the *posteriori* distribution can be computed recursively, using the previous estimate, $p(\mathbf{x}_{t-1} \mid \mathbf{y}_{1:t-1})$, the motion-model, $p(\mathbf{x}_t \mid \mathbf{x}_{t-1})$ and the observation model, $p(\mathbf{y}_t \mid \mathbf{x}_t)$.

To address this problem we use particle filtering methods. Particle filtering is a Bayesian method in which the probability distribution of an unknown state is represented by a set of M weighted particles $\{\mathbf{x}_t^{(i)}, w_t^{(i)}\}_{i=1}^{M}$ [11]:

$$p(\mathbf{x}_t \mid \mathbf{y}_{1:t}) \approx \sum_{i=1}^{M} w_t^{(i)} \delta(\mathbf{x}_t - \mathbf{x}_t^{(i)}) \tag{5}$$

where $\delta(\cdot)$ is the dirac delta function. Based on the discrete approximation of $p(\mathbf{x}_t \mid \mathbf{y}_{1:t})$, different estimates of the best state at time t are possible to be devised. For instance we may use the Monte Carlo approximation of the expectation, $\hat{\mathbf{x}} \doteq \frac{1}{M} \sum_{i=1}^{M} w_t^{(i)} \mathbf{x}_t^{(i)} \approx \mathbb{E}(\mathbf{x}_t \mid \mathbf{y}_{1:t})$, or the maximum likelihood estimate, $\hat{\mathbf{x}}_{ML} \doteq \text{argmax}_{\mathbf{x}_t} \sum_{i=1}^{M} w_t^{(i)} \delta(\mathbf{x}_t - \mathbf{x}_t^{(i)})$.

To compute the approximation to the *posteriori distribution*, a typical tracking algorithm works cyclically in three stages:

1. *Prediction* - computes an approximation of $p(\mathbf{x}_t \mid \mathbf{y}_{1:t-1})$, by moving each particle according to the motion model;
2. *Update* - each particle's weight i is updated using its likelihood $p(\mathbf{y}_t \mid \mathbf{x}_t^{(i)})$:

$$w_t^{(i)} \propto w_{t-1}^{(i)} p(\mathbf{y}_t \mid \mathbf{x}_t^{(i)}) \tag{6}$$

3. *Resampling* - the particles with a high weight are replicated and the ones with a low weight are forgotten.

For this purpose, we need to model probabilistically both the motion dynamics, $p(\mathbf{x}_t \mid \mathbf{x}_{t-1})$, and the computation of each particle's likelihood $p(\mathbf{y}_t \mid \mathbf{x}_t^{(i)})$.

3.1 The Motion Dynamics

In the system proposed herein we assume motion dynamics follow a standard autoregressive dynamic model:

$$\mathbf{x}_t = A\mathbf{x}_{t-1} + \mathbf{w}_t, \tag{7}$$

where $\mathbf{w}_t \sim \mathcal{N}(0, Q)$. The matrices A, Q, could be learned from a set of representative correct tracks, obtained previously (e.g., see [3]), however, we choose pre-defined values for these two matrices (see Sections 4 and 5). Since the coordinates in the model are real-world coordinates, the motion model for a tracked object can be chosen in a principled way, both by using realistic models (constant velocity, constant acceleration, etc.) and by defining the covariance of the noise terms in intuitive metric units.

3.2 Observation model

Each state vector $\mathbf{x}_t$ represents a target pose hypothesis. According to target shape, we compute sets of N points in the 3D inner and outer object boundaries: $\{\mathbf{D}_{in}^n\}$ and $\{\mathbf{D}_{out}^n\}$, $n = 1 \cdots N$. These points must be carefully chosen so that their projection in the image plane, using the projection model of section 2, falls in the 2D inside and outside boundaries of the image contour. Then, we obtain sets of 2D points $\{\mathbf{d}_{in}^n\}$ and $\{\mathbf{d}_{out}^n\}$. Each point $\mathbf{d}$ in the image is represented by its color vector in the HSI representation. For the inner and outer boundary point sets, we will compute HSI histograms, with $B = B_h\, B_s\, B_i$ bins.

Let us denote $b_t(\mathbf{d}) \in \{1, \ldots, B\}$ the bin index associated with the color vector at pixel location $\mathbf{d}$ and frame t. Then the histogram of the color distribution of a generic set of points can be computed by a kernel density estimate $\mathbf{H} \doteq \{h(b)\}_{b=1,\ldots,B}$ of the color distribution at frame t, where each histogram bin is given as in [4]

$$h(b) = \beta \sum_n \delta[b_t(\mathbf{d}^n) - b] \tag{8}$$

where δ is the Kronecker delta function, β is a normalization constant which ensures h to be a probability distribution $\sum_{b=1}^{B} h(b) = 1$.

To compute the similarity between two histograms we apply the Bhattacharyya similarity metric, as in [13]:

$$\mathcal{S}\left(\mathbf{H}^1, \mathbf{H}^2\right) = \sum_{b=1}^{B} \sqrt{h^1(b) \cdot h^2(b)} \tag{9}$$

The likelihood of the hypothesis is computed, as a function of two similarities: the similarity between the object color model and the color measured in the inside image boundary, and the similarity between the colors measured in the image inside and outside the contour.

Defining a reference color model for the object as $\mathbf{H}^{\text{model}}$, $\mathbf{H}^{\text{inner}}$ as the inner boundary points color histogram, and $\mathbf{H}^{\text{outer}}$ the outer boundary histogram, we will measure their similarity, using (9). The data likelihood should favor candidate color histograms which are close to the reference histogram and are sufficiently distinct from the background. Therefore we use:

$$p(\mathbf{y}_t \mid \mathbf{x}_t^{(i)}) = \text{pos}\left[\mathcal{S}(\mathbf{H}^{\text{model}}, \mathbf{H}^{\text{inner}}) - k\mathcal{S}(\mathbf{H}^{\text{outer}}, \mathbf{H}^{\text{inner}})\right] \tag{10}$$

where the pos$(\cdot)$ function truncates to zero the negative values. This allow us to cope with the detection of the object (first term) and the detection from the background (second term).

4 Implementation of the RoboCup MSL 3D Tracker

The present approach is tested for a ball and robot tracking task, in a typical RoboCup MSL environment. The color model for each object was built collecting

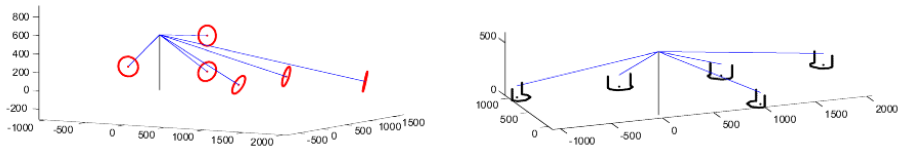
a set of images in which the object is present, and calculating the HSI color histogram on the (hand labeled) pixels belonging to the specific object. For target dynamics, we have chosen a constant velocity model, in which the motion equations correspond to a uniform acceleration during one sample time:

$$x_t = Ax_{t-1} + Ba_{t-1}, \quad A = \begin{bmatrix} \mathbf{I} & (\Delta t)\mathbf{I} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}, \quad B = \begin{bmatrix} (\frac{\Delta t^2}{2})\mathbf{I} \\ (\Delta t)\mathbf{I} \end{bmatrix} \tag{11}$$

where I is the 3×3 identity matrix and a_t is a 3×1 white zero mean random vector corresponding to an acceleration disturbance. We have set $\Delta t = 1$ for all the experiments, whereas the covariance matrix of the random acceleration vector was fixed at:

$$cov(a_t) = \sigma^2\mathbf{I}, \quad \sigma = 90\text{mm/frame}^2 \tag{12}$$

The observation model requires the definition of adequate points in the 3D object inner and outer boundaries, as described in Section 3.2. Our idea was to determine which points of the 3D model would be projected on the object's contour on the image (see Fig.2) and then create the two sets of 2D boundary points by projecting the selected 3D points for a smaller and a larger model of the object (see the close-up's in Figures 3 and 5: the projected contours are drawn in white, while internal and external points are drawn in black). For the ball, for instance, the 3D contour points lie on the intersection between the sphere modelling it and the plane orthogonal to the line which passes through the virtual projection center and the center of the sphere. With this model, it is possible to adjust the number of points describing the 2D contour, obtaining faster processing times (less points) or more robustness (more points).



(a) (b)

Fig. 2. 3D plot of the 3D points projected to obtain the 2D contour points for balls (a) and robots (b), at different positions.

5 Experimental Results

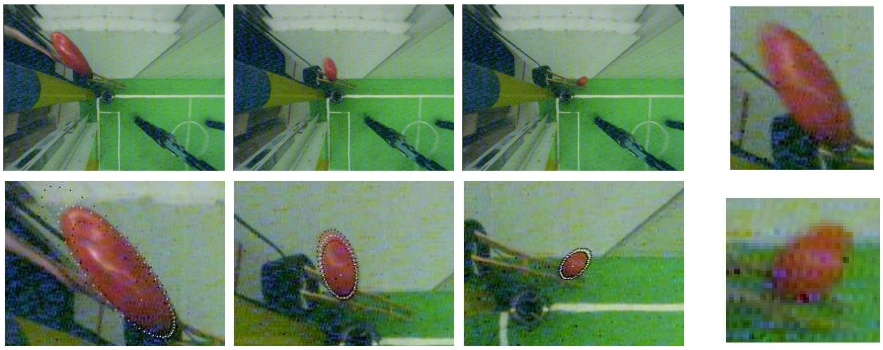
We ran several experiments to assess the accuracy and precision of the proposed tracking method: we tracked a ball rolling down a ramp, a ball bouncing on the floor and a robot maneuvering. We furthermore ran an experiment placing a still ball at different positions around the robot and measuring the error with respect to the ground truth.

5.1 Ball tracking – ramp and bouncing

In the first experiment we tracked a ball rolling down a two-rail ramp. The projection of the ball on the image plane changes dramatically in size after some frames (see Fig.3a), due to the nature of the catadioptric system used. The images are affected by both motion blur and heavy sensor noise (see Fig.3b).

This image sequence was acquired with a frame rate of 20fps and we used 10000 particles in the tracker. The initial position for the particles was obtained sampling a 3D Normal distribution, the mean and standard deviation of which were manually set. The initial velocitiy was also manually set, equal for every particle. To sample the pixels in order to build the inner and outer color histograms for each hypothesis we projected a sphere with respectively 0.9 and 1.1 the radius of the actual ball. For each projection we used 50 points, uniformly distributed on each 3D contour. The parameter k was set to 1.5 for all experiments, meaning that we wanted the difference between inner color and outer color to be more discriminative than the similarity between inner color and model color. We repeated the tracking 10 times on the same image sequence.

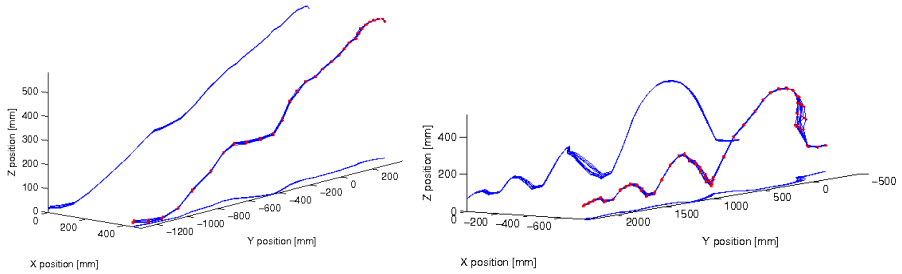
The results of the tracking are visible in Fig. 4a.



(a) (b)

Fig. 3. Ball rolling down a ramp. (a) frames 1, 11 and 21 of the sequence and three corresponding close-ups of the tracked ball with the contour of the best hypothesis drawn in white (bottom row). The pixels marked in black are the ones used to build the color histograms. (b) Close-ups of the ball showing motion blur and noise.

In the second experiment we tracked a ball bouncing on the floor. The experimental setting was exactly the same as for the first experiment described, but for the frame rate, which in this case was of 25fps. The image sequence begins with the ball about to hit an obstacle on the ground, while moving horizontally. The collision triggers a series of parabolic movements for the ball, which is tracked until it hits the ground for the fourth time. We repeated the tracking process ten times on the same image sequence. The results are visible in Fig. 4b.



(a) (b)

Fig. 4. (a) Ball rolling down the ramp: plot of the tracked paths resulting from 10 runs of the algorithm performed on the same image sequence. The 10 blue lines with red points represents the 10 3D estimated trajectories of the ball, the two blue lines are the projection of these trajectories on the ground and lateral plane. (b) Ball jumping: same kind of plot for the estimated trajectories of the ball in the jump image sequence.

5.2 Robot tracking.

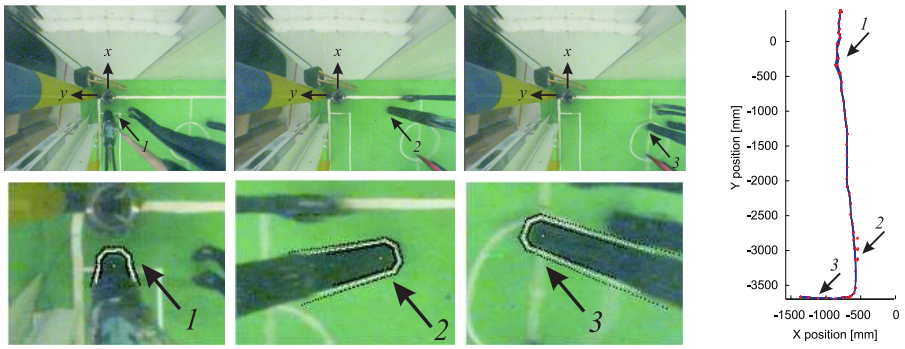
In this experiment we tracked a robot moving along a straight line, turning by 90 degrees and continuing its motion along the new direction (Fig. 5a). In this experiment the setting differs from the previous two: the vertical position and speed of the tracked object were constrained to be null. The injected velocity noise was, thereafter, distributed as a 2D Normal. To sample the pixels in order to build the inner and outer color histogram for each hypothesis we projected the contour of an 8-sided-prism with respectively 0.75 and 1.25 the size of the actual robot. The size difference between the projected models and the actual one is greater than in the case of the ball due to the fact that the model for the robot does not exactly fit its actual shape. For each projection, 120 points were used. We repeated the tracking 10 times and results are shown in Fig. 5b.

5.3 Error evaluation.

In this experiment we placed a ball at various positions around a robot and confronted the positions measured with our system against the ground truth. The positions were either in front of the robot, on its right, on its left or behind it, and either on the floor or at a height of 340mm.

Both the ground truth and the estimated ball positions are shown in Figure 6. It is noticeable some bias mainly in the vertical direction, due to miscalibration of the experimental setup. However, we are mostly interested in evaluating errors arising in the measurement process. Distance to the camera is an important parameter in this case, because target size varies significantly. Therefore, we have performed a more thorough error analysis evaluating its characteristics as a function of distance to the camera.

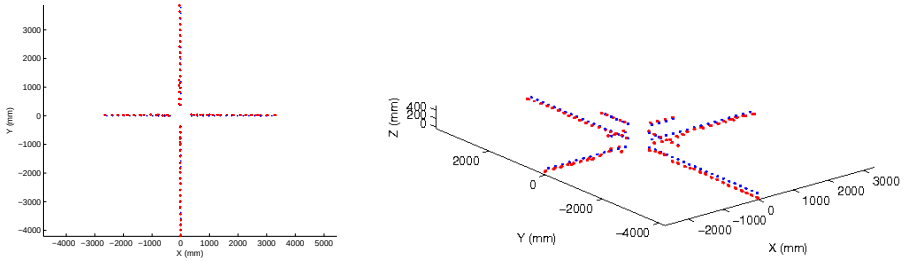
In Figure 7, we plot the measured error in spherical coordinates (ρ–distance, ϕ–elevation, θ–azimuth), as a function of distance to camera's virtual projection



(a) (b)

Fig. 5. Robot tracking. (a) three different frames of the sequence (top) and three close-ups of the tracked robot with the contour of the best hypothesis (white) and regions used to build the color histograms (black) drawn (bottom). (b) reconstructed robot trajectory - a view from the top of 10 estimated trajectories.

center. The first plot shows that radial error characteristics are mostly constant along distance, with a systematic error (bias) of about $46mm$, and standard deviation of about $52mm$. This last value is the one we should retain for characterizing the precision of the measurement process. The second plot shows the elevation error, where it is evident a distance dependent systematic error. This has its source on a bad approximation of the projection model for distances close to the cameras. Finally, the third plot shows the azimuthal error. It can be observed that there is a larger random error component at distances close to the camera, but this just a consequence of the fact that equal position errors at closer distances produce larger angular errors. Therefore we conclude that the precision of the observation model is in average of $52mm$ and do not depend significantly on distance to the camera in the tested range.



(a) (b)

Fig. 6. Two views of ground truth (blue) and measurements (red).

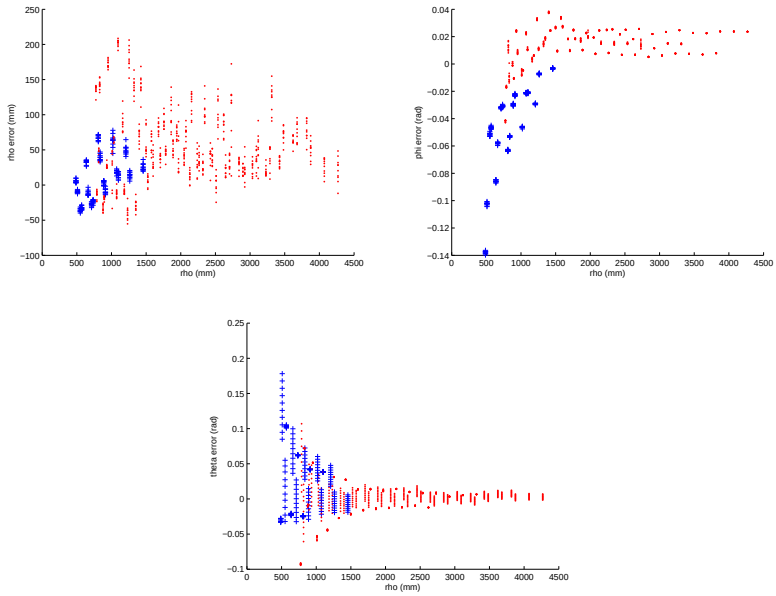


Fig. 7. ρ, ϕ, θ error for balls laying on the ground (red dots) and flying at 340mm (blue crosses).

6 Conclusions

In this paper we have presented a tracking system for MSL Robots equipped with omnidirectional cameras, capable of tracking both targets on or above the floor, to consider the possibility of flying balls. The tracker uses a 3D shape and color model of the targets, and uses particle filtering methods to estimate their 3D location with respect to the Robot. Each hypothesis in the filter represents the 3D pose of an object. Even though in this paper we only consider targets with simple shapes (spheres and cylinders for representing balls and robots), the proposed model is general and copes with arbitrary shapes.

The main advantage of our approach is the use of a full 3D model in which the targets' motion dynamics is naturally expressed. Previous image based (2D) tracking methods require non-linear motion models (image projection is often non-linear) or must rely on approximations. This non-linearity becomes even more drastic in the case of omnicamera systems when targets are not lying on the floor. An additional advantage of the proposed method is related to a direct computation of 3D pose, whereas 2D models compute an image based pose that still must be mapped to world coordinates.

We have performed extensive experiments with real robots in a RoboCup MSL scenario. This paper showed the performance of our method in tracking maneuvering robots, rolling and jumping balls, demonstrating its ability to deal with off-the-floor targets and sudden trajectory changes. Additionally, we eval-

uated the precision of the system in static scenarios with ground truth measurements.

Since it is becoming more frequent to have robots kicking balls off the floor, the presented method constitutes a solution to improve ball position estimation, which, in the case of the goal-keeper, may be of fundamental importance.

References

1. J. Gaspar, N. Winters, J. Santos-Victor: Vision-based Navigation and Environmental Representations with an Omnidirectional Camera. IEEE Transactions on Robotics and Automation, Vol. 16, 6, December 2000
2. D. Koller, J. Weber, J. Malik: Robust Multiple Cars Tracking with Occlusion Reasoning. European Conf. on Computer Vision, pp. 186-196, 1994.
3. Gilles Celeux, J. Nascimento, J. S. Marques: Learning switching dynamic models for objects tracking Pattern Recognition, vol. 37, no. 9, pp. 1841-1853, Sep. 2004.
4. D. Comaniciu, V. Ramesh, P. Meer: Real-Time Tracking of Non-Rigid Objects using Mean Shift. CVPR (2000), pp. 142-151.
5. T. Misu, M. Naemura, Wentao Zheng, Y. Izumi, K. Fukui: Robust Tracking of Soccer Players based on Data Fusion. IEEE 16th Int. Conf. on Patern Recognition, pp. 556-561, vol. 1, 2002.
6. S. S. Intille, J. W. Davis, A. F. Bobick: Real-Time Closed-World Tracking. IEEE Conf. on Compuiter Vision Pattern Recognition, pp. 697-703, 1997.
7. Z. Khan, T. Balch, and F. Dellaert: MCMC Data Association and Sparse Factorization Updating for Real Time Multitarget Tracking with Merged and Multiple Measurements. IEEE Trans. on PAMI, vol. 28, no. 12, pp. 1960-1972, Dec. 2006.
8. Lima, P., Bonarini, A., Machado, C., Marchese, F., Ribeiro, F., Sorrenti, D.: Omnidirectional catadioptric vision for soccer robots Robotics and Autonomous Systems, 36(2–3), pp. 87–102, 2001.", year = "2001", (2001).
9. Thrun, S., Burgard, W., Fox, D.: Probabilistic Robotics. MIT press (2005).
10. K. Okuma, Ali Taleghani, N. de Freitas, J.J. Little, and D. G. Lowe: A Boosted Particle Filter: Multitarget Detection and Tracking. European Conf. on Computer Vision, pp. 28-39, 2004.
11. A. Doucet, N. de Freitas, N. Gordon: Gordon editors: Sequential Monte Carlo Methods In Practice. Springer Verlag (2001).
12. M. Isard, and A. Blake: Condensation: conditional density propagation for visual tracking. Int. Journal of Computer Vision, vol. 28, no. 1, pp. 5-28, 1998.
13. P. Perez, C. Hue, J. Vermaak, M. Gangnet: Color-Based Probabilistic Tracking using Unscented Particle Filter. CVPR (2002).
14. Olufs, S., Adolf, F., Hartanto, R., Plöger, P.: Towards probabilistic shape vision in robocup: A practical approach. RoboCup International Symposium (2006).
15. Geyer, C., Daniilidis, K.: A unifying theory for central panoramic systems and practical applications. European Conf. on Computer Vision, (2000) pp. 445-461.
16. Benosman, R., Kang, S.B., eds.: Panoramic Vision. Springer Verlag (2001).
17. Hicks, R., Bajcsy, R.: Catadioptric sensors that approximate wide-angle perspective projections. CVPR (2000) 545–551
18. Gaspar, J., Deccó, C., Jr, J.O., Santos-Victor, J.: Constant resolution omnidirectional cameras. 3rd IEEE Workshop on Omni-directional Vision (2002) 27–34