# Color 3D Model-Based Tracking with Arbitrary Projection Models

M. Taiana, J. Santos, J. Gaspar, J. Nascimento, A. Bernardino and P. Lima*

Institute for Systems and Robotics, IST, 1049-001 Lisboa, Portugal
{mtajana,jsantos,jag,jan,alex,pal}@isr.ist.utl.pt

**Abstract.** We present a color- and shape-based 3D tracking system suited to a large class of vision sensors. The method is, in fact, applicable with any projection model, provided that it is calibrated and the projection function is known. The tracking architecture is based on Particle Filtering methods where each particle represents the 3D state of the object, rather that its state in the image, therefore bypassing the nonlinearity caused by the projection model. This allows the use of realistic 3D motion models and easy integration of the sensor self-motion measurements. All nonlinearities are concentrated in the observation model that, for each particle, projects a few tens of special points onto the image, on (and around) the 3D object's surface. The likelihood of each state is then evaluated using color histograms. Since only pixel access operations are required, the method does not involve costly image processing routines like edge/feature extraction, color segmentation or 3D reconstruction, that can be cumbersome with omnidirectional projection models. The tracking system copes well with motion and optical blur. We show applications of tracking various objects (balls, boxes) in mobile robots with catadioptric and dioptric omnidirectional sensors.

## 1 Introduction

Omnidirectional and wide-angle vision sensors are becoming increasingly used in robotics and surveillance systems. Since these sensors gather information from a large portion of the surrounding space, they reduce the number of required cameras for a certain spatial coverage, thus sparing resources. Their classical applications are in mobile robot self localization and navigation [16,9], surveillance systems [4] and humanoid foveal vision setups [14]. One drawback is that images suffer strong distortion and perspective effects, demanding non-standard algorithms for target detection and tracking.

In scenarios where the shape of objects can be modeled accurately *a priori*, 3D model-based techniques have been among the most successful in tracking and pose estimation applications [15]. However, classical 3D model-based tracking methods are strongly dependent on the projection models and, thus, are not easily applicable to

* 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 pluriannual 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, Dr. Alessio Del Bue and Giovanni Saponaro for the fruitful discussions.

omnidirectional images. Often non-linear optimization methods are employed: a cost function expressing the mismatch between the predicted and observed object points is locally minimized with respect to the object's pose parameters [15]. This process involves the linearization of the relation between state and measurements, which can be very complex with omnidirectional sensors. These approaches have limited convergence basins requiring either small target motions or 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 [7]. The state of the target is represented by a set of weighted particles, whose weights are computed by projecting target features onto the image and matching them with the known object's model. The method, therefore, does not require the linearization between the state and the measurements, facilitating its use with arbitrary projection models. Here we use color features to compute state likelihoods, but in principle other features like corners or edges could be used. The reasons for using color features are the following: color features cope well with motion and optical blur, which are frequent in the scenarios we consider; color features do not require local image processing for extracting edges or corners, but simply pixel evaluations, which makes the system work in real-time; finally, many robotics research problems assume objects with distinctive colors to facilitate the figure-ground segmentation problem, for instance in cognitive robotics [17] or robotic competitions [20]. The algorithms presented in this paper are thus suited to such scenarios, using arbitrary projection models. In [21] we showed the utilization of this algorithm with omnidirectional and perspective sensors. In this paper we concentrate on omnidirectional sensors and formulate the tracking problem in the case of a moving robot.

The paper is organized as follows: Section 2 presents the imaging systems and respective projection models. Section 3 describes the Particle Filtering approach. In Section 4 we detail the color-based, sample-based observation model used in the filter. In Section 5 we describe experiments in which we tracked balls and polyhedra with dioptric and catadioptric imaging systems. In Section 6 we draw conclusions and present directions for future work.

## 2 Imaging Systems

In this section we describe the imaging systems used in our tracking methodology. In fact the methodology is applicable with any projection model, under the assumption that it is calibrated and the projection function is known. Considering robotic applications, our imaging systems are in general designed to have a wide-angle constant-resolution view of the ground plane [12,8] or an omnidirectional view in the azimuth direction [2]. These imaging systems have axial symmetry in most of the state of the art designs, thus we focus the following description on this assumption.

Cameras with axial symmetry can be described by (see Figs. 1a and 1c):

$$\rho = \mathcal{P}\left([r\ z]^T; \vartheta\right) \tag{1}$$

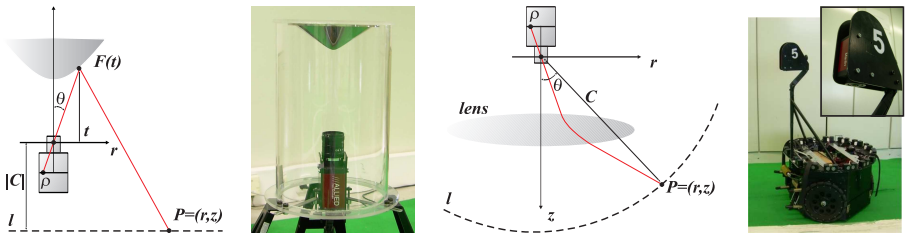
where the $z$ axis coincides with the optical axis, $[r\ z]$ represents a 3D point without its azimuthal coordinate and $\vartheta$ contains the system parameters.

In the case of a constant resolution design, $\mathcal{P}$ is trivial for the ground-plane, as it is just a scale factor between pixels and meters [12,8]. Deriving $\mathcal{P}$ for the complete 3D field of view (FOV) of a catadioptric system involves using the actual mirror shape, $F$, which is a function of the radial coordinate $t$. Based on first order optics, particularly on the reflection law at the specular surface of revolution, $(t, F)$, we have:

$$\arctan(\rho) + 2 \cdot \arctan(F') = -\frac{r-t}{z-F} \tag{2}$$

where $\phi = -(r-t)/(z-F)$ is the system's vertical view angle, $\theta = \arctan(\rho)$ is the camera's vertical view angle, and $F'$ represents the slope of the mirror shape. When $F$ denotes an arbitrary function and we replace $\rho = t/F$, (2) becomes a differential equation, expressing the constant horizontal resolution property, $\rho = a \cdot r + b$, for one plane $z = z_0$. $F$ is usually found as a numerical solution of the differential equation (see details and more designs in [12,8]).

If $F$ is a known shape then (2) describes a generic Catadioptric Projection Model (CPM), as it forms an equation on $\rho$ for a given 3D point $(r, z)$. Some simple shapes $F$, such as a hyperboloid having one focus coinciding with the center of the pin-hole camera capturing the mirror image, allow deriving a closed-form solution for finding $\rho$ from $(r, z)$. In general, however, there is no closed-form solution. In these cases, the CPM is usually implemented as an optimization procedure where $\rho$ is iteratively changed to move the corresponding back-projection ray and approximating the 3D point $(r, z)$. This is computationally expensive and thus justifies finding simpler approximate models for real-time applications.




(a) (b) (c) (d)

**Fig. 1.** Catadioptric and dioptric cameras. (a,b) CPM geometry and setup. FOV of about $5m \times 9m$ and camera $0.6m$ above the ground. (d,e) CARPM geometry and setup. Lens DSL215 by Sunex.

Next we introduce two known models that approximate the CPM: the Unified Projection Model (UPM) and the standard Perspective Projection Model (PPM). The UPM consists of a two-step mapping via a unit-radius sphere [10]: (i) a 3D world point, $P = [r\ \varphi\ z]^T$, is projected orthogonally to the sphere surface onto a point $P_s$; (ii) project to a point on the image plane, $P_i = [\rho\ \varphi]^T$, from a point $O$ on the vertical axis of the sphere, through the point $P_s$. The mapping is defined by:

$$\rho = \frac{l+m}{l\sqrt{r^2+z^2}-z} \cdot r \tag{3}$$

where the $(l, m)$ parameters describe the type of camera. The UPM is a widely used representation for CPM when $F$ describes (i) a hyperboloid or ellipsoid with focus at $(0, 0)$; (ii) a paraboloid combined with a telecentric lens ($\theta = 0$) or (iii) $F = const$ [2]. In our case, $F$ is computed numerically to have the constant resolution property, and therefore does not correspond to any of the former cases. Thus, the UPM can only approximate the CPM.

The PPM is a particular case of the UPM, obtained by setting $l = 0$ and $k = -m$ :

$$\rho = k \cdot r/z. \tag{4}$$

Eq. (4) shows that the PPM has constant resolution, i.e., it enforces a linear relationship between $\rho$ and $r$, at all $z$-planes. Simulations show that the PPM is more accurate for the ground plane, as expected by its design, but the UPM allows better approximations for planes above the ground, involving however more computations.

Lens-only systems, i.e., dioptric systems, have been traditionally described by the PPM. This is valid only for narrow fields of view, as otherwise the radial distortion becomes too significant. Modeling super-fisheye fields of view (views larger that $180^o$ angles) involves using other projection models [11,3]. For example [11] uses a super-fisheye Nikon F8 and proposes the model $\rho = a \cdot \tan(\theta/b) + c \cdot \sin(\theta/d)$. Some recent advances allowed companies to build super-fisheye lenses having simple projection models. For example the Sunex's DSL215 lens, Fig. 1c, is designed to have a Constant Angular Resolution Projection Model (CARPM, termed *equidistant* in [11]), meaning the image points $\rho$ have a linear relationship if 3D elevation angles $\theta$:

$$\rho = f \cdot \theta = f \cdot \arctan(r/z). \tag{5}$$

## 3 3D Tracking with Particle Filters

We are interested in estimating, at each time step, the 3D pose of the target. Thus, the state vector of the target, denoted as $\mathbf{X}_t$, contains its 3D pose and derivatives up to a desired order. It represents the object evolution along time, which 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 $\{\mathbf{y}_t; t \in \mathbb{N}\}$, $\mathbf{y}_t \in \mathbb{R}^{n_\mathbf{y}}$, are conditionally independent given the process $\{\mathbf{x}_t; t \in \mathbb{N}\}$ with 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 *a posteriori* distribution of the state given all observations $p(\mathbf{x}_t \mid \mathbf{y}_{1:t})$. Under the Markov assumption:

$$p(\mathbf{x}_t|\mathbf{y}_{1:t}) \propto p(\mathbf{y}_t|\mathbf{x}_t) \int p(\mathbf{x}_t|\mathbf{x}_{t-1})\, p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1}) d\mathbf{x}_{t-1}. \tag{6}$$

The above equation shows that the *a posteriori* distribution can be computed recursively, using the estimate at the preceding time step, $p(\mathbf{x}_{t-1} \mid \mathbf{y}_{1:t-1})$, the motion-model, $p(\mathbf{x}_t \mid \mathbf{x}_{t-1})$ and the observation model, $p(\mathbf{y}_t \mid \mathbf{x}_t)$.

We use Particle Filtering methods in which the probability distribution of an unknown state is represented by a set of M weighted particles $\{\mathbf{x}_t^{(i)}, w_t^{(i)}\}_{i=1}^M$ [7]:

$$p(\mathbf{x}_t \mid \mathbf{y}_{1:t}) \approx \sum_{i=1}^{M} w_t^{(i)} \delta(\mathbf{x}_t - \mathbf{x}_t^{(i)}) \tag{7}$$

where $\delta(\cdot)$ is the Dirac delta function. Based on the discrete approximation of $p(\mathbf{x}_t \mid \mathbf{y}_{1:t})$, one can compute different estimates of the best state at time $t$. We 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, $\hat{\mathbf{x}}_{\mathrm{ML}} \doteq \arg\max_{\mathbf{x}_t} \sum_{i=1}^{M} w_t^{(i)}\delta(\mathbf{x}_t - \mathbf{x}_t^{(i)})$.

The tracking algorithm is composed by four steps:

1. *Prediction* - computes an approximation of $p(\mathbf{x}_t \mid \mathbf{y}_{1:t-1})$, by moving each particle according to the object's motion model
2. *Observation* - computes the likelihood of each particle, based on image data
3. *Update* - updates each particle's weight $i$ using its likelihood $p(\mathbf{y}_t \mid \mathbf{x}_t^{(i)})$, by the means of $w_t^{(i)} \propto w_{t-1}^{(i)} p(\mathbf{y}_t \mid \mathbf{x}_t^{(i)})$
4. *Resampling* - replicates the particles with a high weight and discards the ones with a low weight

For this purpose, we need to model in a probabilistic way both the motion dynamics, $p(\mathbf{x}_t \mid \mathbf{x}_{t-1})$, and the computation of each particle's likelihood $p(\mathbf{y}_t \mid \mathbf{x}_t^{(i)})$ (steps 1 and 2). We discuss the model for the motion dynamics in the rest of this section, while we describe the observation model in Section 4.

### 3.1 Object Motion Dynamics

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

$$p(\mathbf{X}_t|\mathbf{X}_{t-1}) = \mathcal{N}(\mathbf{X}_t|\mathbf{X}_{t-1}, \Lambda) \tag{8}$$

where $\mathcal{N}(.|\mu, \Sigma)$ stands for a Gaussian distribution with mean $\mu$ and covariance $\Sigma$, and $\Lambda$ stands for the diagonal matrix with the variances for random walk models on the components of the object state model. This approach has been widely used (e.g. [1,19]).

In this work we consider two kinds of objects: (i) spherical and (ii) polyhedral. 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 time sample:

$$\mathbf{X}_{t+1} = \begin{bmatrix} \mathbf{I} & (\Delta t)\mathbf{I} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \mathbf{X}_t + \begin{bmatrix} (\frac{\Delta t^2}{2})\mathbf{I} \\ (\Delta t)\mathbf{I} \end{bmatrix} a_t \tag{9}$$

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

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, $\eta_p$ is Gaussian noise and $\eta_q$ is a quaternion generated by sampling Euler angles from a Gaussian distribution.

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. The fact of using a constant velocity model is not limiting for cases where the objects can undergo sudden direction changes, e.g., in a RoboCup scenario. Using an adequately chosen acceleration noise, we can cope with arbitrary accelerations.

### 3.2 Tracking with Moving Robots

To use the tracker on moving robots we need to distinguish between two reference frames for the 3D coordinates of a point: the world reference frame $\{\mathrm{W}\}$, which is inertial, and the robot reference frame $\{\mathrm{R}\}$, which is not inertial as the robot undergoes accelerations during its motion. The state of a tracked object is expressed in terms of the world reference frame, so that a motion model based on the laws of physics can be expressed in the simplest possible way.

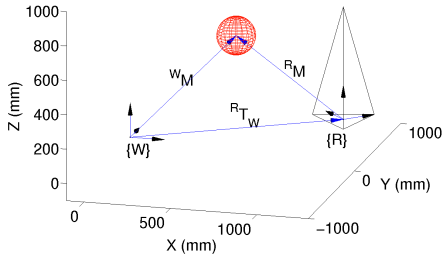



**Fig. 2.** Position of one sphere in the world and robot reference frames.

In order to project a 3D point onto the image plane, its relative position with respect to the imaging system of the robot must be known. This means that coordinates expressed in the world reference frame ${}^{W}M = [{}^{W}X, {}^{W}Y, {}^{W}Z, 1]$ must be transformed to the robot reference frame ${}^{R}M = [{}^{R}X, {}^{R}Y, {}^{R}Z, 1]$ by means of the transformation matrix ${}^{R}T_{W}$, which comprises the rotation matrix ${}^{R}R_{W}$ and the translation vector ${}^{R}t_{W}$.

$$ {}^{R}T_{W} = \begin{bmatrix} {}^{R}R_{W} & {}^{R}t_{W} \\ 0 & 1 \end{bmatrix} \tag{10} $$

The transformation matrix is computed at every time step based on the difference between the initial pose of the robot and the current one. The 2D coordinates of the point $m$, corresponding to the point ${}^{W}M$, are thus computed as:

$$ m = \mathcal{P}({}^{R}M) = \mathcal{P}({}^{R}T_{W} \cdot {}^{W}M). \tag{11} $$

In other words, we model the dynamics of the tracked object in the world reference frame, while the observer position is taken into account in the observation model.

## 4 Observation model

To calculate the likelihood of one particle we first define sets of 3D points, as a function of the object 3D shape and of its position and orientation with respect to the imaging system. Then we project these 3D points onto the image plane. Eventually we build a color histogram for each set of points and compute the likelihood of the particle as a function of the similarities between pairs of color histograms. This process 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 the projection of isolated points is required.

### 4.1 3D Points Generation and Their Projection onto the Image

From one 3D pose hypothesis for the tracked object we determine sets of 2D points that lie on the image, around the object edges and silhouette. The idea is that the color and luminance differences around the object edges are indicators of the likelihood of the hypothetical pose. We consider two different object shapes: spheres and convex polyhedral objects. However the model can be easily extended to general polyhedral objects by exploiting the current knowledge in computer graphics [13].

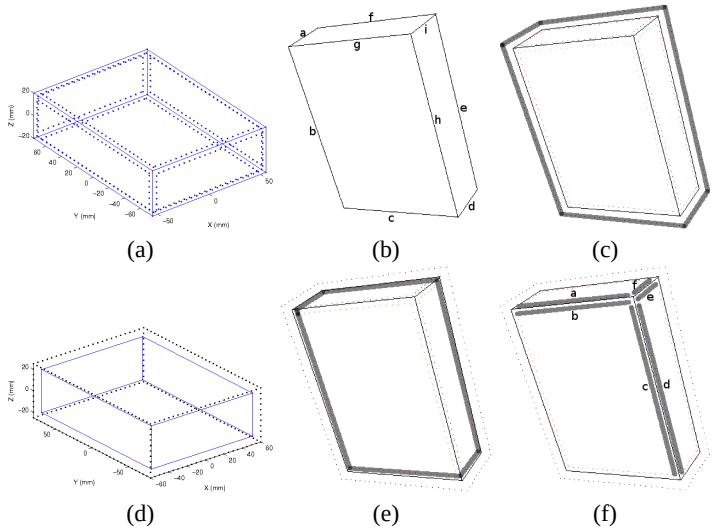



**Fig. 3.** Polyhedron 3D model. (a) and (d) show the location of the 3D points of the model. (b) shows one particular projection of the object in which edges $a, b, c, d, e$ and $f$ form the silhouette of the projected figure, while $g, i$ and $h$ are non-silhouette edges. (c), (e) show the areas sampled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

In the case of polyhedra, we use a 3D model that consists of a collection of faces and edges. To each pair $(face, edge)$ we associate a set of 3D points that lie on the

specific face, near the edge (Fig. 3a). To each edge we associate a set of points that lie on the corresponding edge of an expanded polyhedron (Fig. 3d). 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 (see Figs. 3c, 3e and 3f) . 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.

The projection of the 3D points onto the image is performed using the appropriate projection model, as detailed in Section 2.

### 4.2 Color-Based 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 *a posteriori* 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 intensity from color. 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

$$h_{i,\mathbf{x}}^c = \beta \sum_{\mathbf{u} \in \mathcal{U}} \delta_a(b_{\mathbf{u}}^c), \qquad i = 1, \ldots, B. \tag{12}$$

$\mathcal{U}$ 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 $c$; $\delta_a$ is a Kronecker delta function at $a$; and $\beta$ is a normalization constant such that $\sum_{i=1}^{B} h_{i,\mathbf{x}}^c = 1$.

We define $\mathbf{h}^{\text{model}}$, $\mathbf{h}^{\text{in}}$ and $\mathbf{h}^{\text{out}}$ as a reference (object) histogram, the inner boundary points and the outer boundary points histogram, respectively. We define $\mathbf{h}_i^{\text{sideA}}$ and $\mathbf{h}_i^{\text{sideB}}$ as the histograms of each of the two sides of the $i^{th}$ non-silhouette edge (see Fig. 3f). To measure the difference between histograms we use the Bhattacharyya similarity as in [5,18]:

$$\mathcal{S}(\mathbf{h}_1, \mathbf{h}_2) = \sum_{i=1}^{B} \sqrt{h_{i,1} h_{i,2}} \tag{13}$$

We define:

$$\mathcal{S}_0 = \mathcal{S}(\mathbf{h}^{\text{model}}, \mathbf{h}^{\text{in}}), \quad \mathcal{S}_1 = \mathcal{S}(\mathbf{h}^{\text{out}}, \mathbf{h}^{\text{in}}), \quad \mathcal{S}_2 = \frac{\sum_{i=0}^{n} \mathcal{S}(\mathbf{h}_i^{\text{sideA}}, \mathbf{h}_i^{\text{sideB}})}{n} \tag{14}$$

as the object-to-model, object-to-background and mean-side-to-side (non-silhouette edges) similarities, respectively. Finally, the resulting distance is:

$$\mathcal{D} = \left(1 - \frac{\mathcal{S}_0 + \kappa_1(1 - \mathcal{S}_1) + \kappa_2(1 - \mathcal{S}_2)}{\kappa_1 + \kappa_2 + 1}\right) - \gamma \tag{15}$$

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

The rationale for this definition of $\mathcal{D}$ is that the distance metric should be high when candidate color histograms are different from the reference histogram and similar to the background. It should also be high when there is little or no difference in color on the sides of non-silhouette edges. Parameters $\kappa_1$ and $\kappa_2$ allow to balance the influence of the different contributions, up to the extent of ignoring them, by setting such parameters to 0. They were set to 1.5 and 0.6 respectively, for the case of the polyhedron; for tracking the sphere they were set to 1.5 and 0 in the first experiment and to 0 and 0 in the second one. The data likelihood function $\mathcal{L}$ is modeled as a Laplacian distribution over the distance metric: $p(\mathbf{y}_t \mid \mathbf{X}_t^{(i)}) \propto e^{-\frac{|\mathcal{D}|}{\epsilon}}$. In our experiments we set $\epsilon = 1/30$.

## 5 Experimental Results

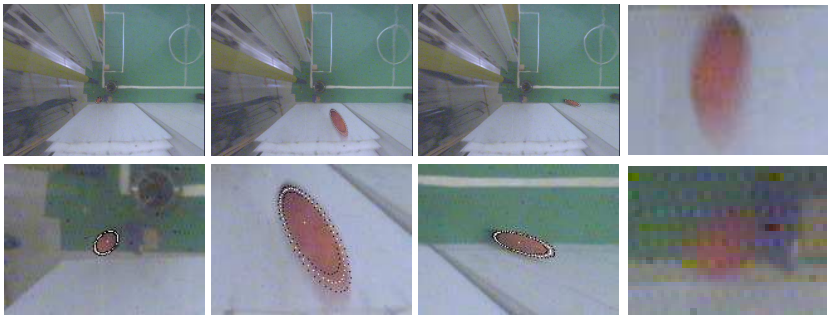
This section presents an evaluation of the proposed methods. Firstly we present the tracking of a ball performed with a catadioptric setup. Then we show the results of the tracking of a cuboid in a dioptric setup. Eventually we show the results of the tracking of a ball performed with a moving robot equipped with a dioptric vision system.

### 5.1 Tracking a Ball in 3D with One Catadioptric Omnidirectional Camera

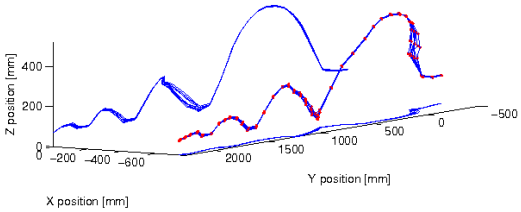
In this experiment we tracked a ball hitting an obstacle on the ground and subsequently performing a sequence of parabolic movements. This image sequence was acquired with a Marlin firewire camera (see Fig. 1b) with a frame rate of 25fps and a resolution of $640 \times 480$ pixels. The tracker used 10000 particles, the Unified Projection Model described in Section 2 and the constant velocity motion model described in Section 3.1. Processing was done off-line, with Matlab. In this case the tracker was provided with the color model for the ball. The projection of the ball on the image plane changes dramatically in size during the image sequence (see Fig. 4a), due to the nature of the catadioptric system used. The images are affected by both motion blur and heavy sensor noise (see Fig. 4b). We repeated the tracking 10 times on the same image sequence to assess how the aleatory component of the tracker influences the precision of the 3D estimated paths (see Fig. 4c), with satisfactory results.

### 5.2 Tracking a Cuboid in 3D with one Dioptric Omnidirectional Camera

In this experiment we tracked a yellow cuboid in a sequence of 600 frames, acquired using the dioptric omnidirectional setup, comprising a Marlin firewire camera and a Sunex DSL215 lens (see Fig. 1d). The resolution was $640 \times 480$ pixels. In the tracker we used 5000 particles, the Constant Angular Resolution Projection Model described in Section 2 and the constant position motion model described in Section 3.1. Processing was done off-line, with Matlab. The tracker managed to follow the cuboid along rotations and translations that greatly affect its projection onto the image, see Fig. 5.



(a) (b)




(c)

**Fig. 4.** Ball tracking in the catadioptric setup. (a) Three frames of the sequence (top row), the corresponding close-ups (bottom row). In both cases the projection of a ball lying in the estimated position is marked in white, while the pixels used to build the inner and outer color histograms are marked in black. (b) Close-ups of the ball showing motion blur and image sensor noise. (c) 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 blue lines are the projection of these trajectories on the ground and lateral plane.

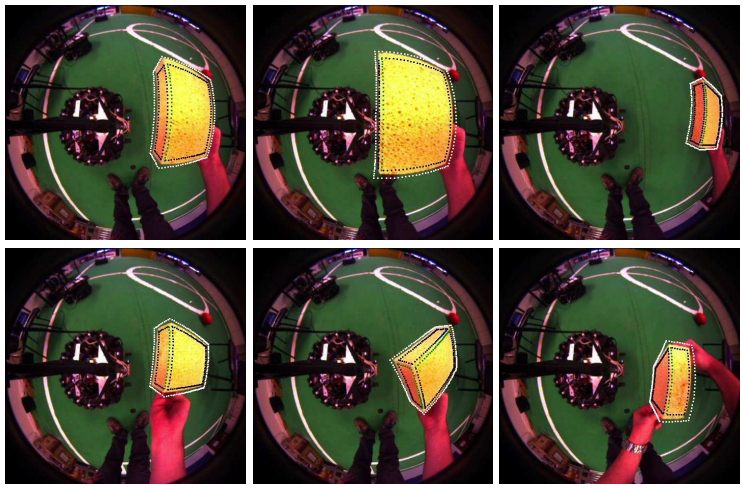
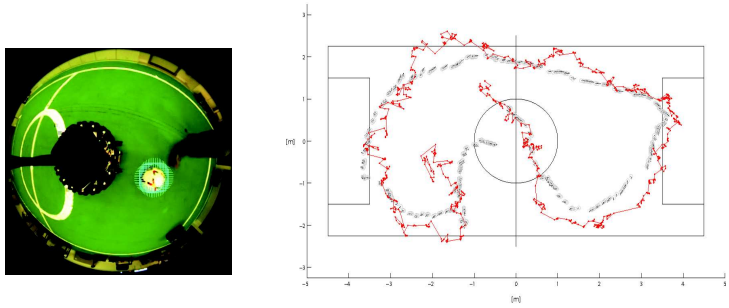


**Fig. 5.** Cuboid tracking in the dioptric setup. Six frames of the sequence with the pixels used to build color histograms highlighted.

### 5.3 Arbitrary Color Ball Tracking with a Moving Robot

The tracker was implemented in the soccer robot team ISocRob [6] software architecture in order to demonstrate a real-time application of the method with robots playing in the RoboCup Middle Size League. In this experiment the omnidirectional robot tracked a moving ball, moving while attempting to catch it. This implementation discards the object-to-model mismatch, described in Eq. 14, and relies strictly on the object-to-background dissimilarity. Therefore, we carried out the experience with an ordinary soccer ball, mainly white colored, as one can see in Fig. 6a, but other colored balls, e.g., orange, could be used.

Images on the robot were acquired using the same omnidirectional camera of the previous experiment with a dioptric setup at 10fps. Odometry motion control measurements were obtained at 25fps and we used only 600 particles in the tracker. Processing was done on-line, at 10fps. The results are visible in Fig. 6b where one can perceive the robot path while pursuing the detected ball (arbitrary trajectory) in a global reference frame (the soccer field centered frame).



(a) (b)

**Fig. 6.** Tracking a white ball with a moving robot. (a) Detection of a white ball based on the object-to-background dissimilarity, where the light green crosses mark the pixels used to build the background histogram. (b) Plot of the paths of robot and ball. The robot starts in the middle circle facing opposite to the ball. The gray circles represent the robot's *pose* while the red dots represent the ball localization, here in a 2D representation only.

## 6 Conclusions

We presented a 3D model-based tracking system, designed on a Particle Filter framework. We illustrated that the system is particularly suited for omnidirectional vision systems, as it only requires the projection of isolated points arising from likely posture hypotheses for the target. We showed that the system is general and can be used with different kinds of omnidirectional vision sensors (dioptric and catadioptric), with different kinds of tracked objects (spheres and polyhedra) and in the cases of static and

moving observer. The results demonstrate that the tracker is able to cope with complex target motions in challenging observation conditions.

In future work we will investigate the possibility of applying information fusion methods to perform the collaborative tracking of objects in distributed robotics systems.

## References


1. P. Barrera, J. M. Cañas, and V. Matellán. Visual object tracking in 3d with color based particle filter. In *Proc. of world academy of science, Eng. and Tech.*, volume 4, 2005.
2. R. Benosman and S. Kang. *Panoramic Vision*. Springer Verlag, 2001.
3. B.Micusik and T.Pajdla. Estimation of omnidirectional camera model from epipolar geometry. In *IEEE CVPR*, pages 485–490, 2003.
4. T. Boult, X. Gao, R. Michaels, and M. Eckmann. Omni-directional visual surveillance. *Image and Vision Computing*, 22(7):515–534, 2004.
5. 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.
6. H. Costelha, N. Ramos, J. Estilita, J. Santos, M. Taiana, J. Torres, T. Antunes, and P. Lima. Isocrob 2007 team description paper. Technical report, Instituto Superior Técnico, 2007.
7. A. Doucet, N. de Freitas, and N. Gordon. *Sequential Monte Carlo Methods In Practice*. Gordon editors, Springer Verlag, 2001.
8. J. Gaspar, C. Deccó, Jun Okamoto Jr, and J. Santos-Victor. Constant resolution omnidirectional cameras. In *IEEE Workshop on Omni-directional Vision*, pages 27–34, 2002.
9. J. Gaspar, N. Winters, and J. Santos-Victor. Vision-based navigation and environmental representations with an omnidirectional camera. *IEEE T-RA*, 16(6), 2000.
10. C. Geyer and K. Daniilidis. A unifying theory for central panoramic systems and practical applications. In *IEEE CVPR*, pages 445–461, 2000.
11. H.Bakstein and T.Pajdla. Panoramic mosaicing with a 180 deg field of view lens. In *IEEE Workshop on Omnidirectional Vision*, pages 60–67, 2002.
12. R. Hicks and R. Bajcsy. Catadioptric sensors that approximate wide-angle perspective projections. In *IEEE CVPR*, pages 545–551, 2000.
13. G. Klein and D. Murray. Full-3d edge tracking with a particle filter. In *Proc. of BMVC 2006*, page III:1119, Edinburgh, Scotland, 2006.
14. Yasuo Kuniyoshi, Nobuyuki Kita, Sebastien Rougeaux, and Takashi Suehiro. Active stereo vision system with foveated wide angle lenses. In *ACCV '95: Invited Session Papers*, pages 191–200, London, UK, 1996. Springer-Verlag.
15. 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.
16. P. Lima, A. Bonarini, C. Machado, F. Marchese, F. Ribeiro, and D. Sorrenti. Omnidirectional catadioptric vision for soccer robots. 2001.
17. L. Montesano, M. Lopes, A. Bernardino, and J. Santos-Victor. Learning object affordances: From sensory motor maps to imitation. *IEEE Trans. on Robotics*, 24(1):15–26, 2008.
18. 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.
19. P. Pérez, J. Vermaak, and A. Blake. Data fusion for visual tracking with particles. *Proc. of the IEEE*, 92(3):495–513, 2004.
20. M. Taiana, J. Gaspar, J. Nascimento, A. Bernardino, and P. Lima. 3D tracking by catadioptric vision based on particle filters. In *Proc. of the Robocup Symposium*, Atlanta, 2007.
21. M. Taiana, J. Nascimento, J. Gaspar, and A. Bernardino. Sample-based 3D tracking of colored objects: A flexible architecture. In *Proc. of BMVC 2008*, Leeds, 2008.