Tracking objects with generic calibrated sensors : an algorithm based on color and 3D shape features [☆☆]

M. Taiana, J. Santos, J. Gaspar, J. Nascimento, A. Bernardino and P. Lima[a]

[a]*Institute for Systems and Robotics, Instituto Superior Técnico, 1049-001 Lisboa, Portugal*

Abstract

We present a color- and shape-based 3D tracking system suited to a large class of vision sensors. The method is applicable, in principle, to any known calibrated projection model. The tracking architecture is based on Particle Filtering methods where each particle represents the 3D state of the object, rather than its state in the image, therefore overcoming the nonlinearity caused by the projection model. This allows the use of realistic 3D motion models and easy incorporation of 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 by comparing the color distributions inside and outside the object's occluding contour. Since only pixel access operations are required, the method does not require the use of image processing routines like edge/feature extraction, color segmentation or 3D reconstruction, which can be sensitive to motion blur and optical distortions typical in applications of omnidirectional sensors to robotics. We show tracking applications considering different objects (balls, boxes), several projection models (catadioptric, dioptric, perspective) and several

[☆]Parts of this manuscript were previously presented at the British Machine Vision Conference (BMVC'08) and at the Workshop on Omnidirectional Robot Vision, held in conjunction with the SIMPAR 2008 conference.

[☆☆]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 PIDDAC program 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.

Email address: {mtaiana,jsantos,jag,jan,alex,pal}@isr.ist.utl.pt (M. Taiana, J. Santos, J. Gaspar, J. Nascimento, A. Bernardino and P. Lima)

challenging scenarios (clutter, occlusion, illumination changes, motion and optical blur). We compare our methodology against a state-of-the-art alternative, both in realistic tracking sequences and with ground truth generated data.

Key words:
Omnidirectional Vision, Tracking, Particle Filtering

---

1. Introduction

Omnidirectional and wide-angle vision sensors have been widely used in the last decade for robotics and surveillance systems. These sensors gather information from a large portion of the surrounding space, thus reducing the number of cameras required to cover a certain spatial extent. Their classical applications include mobile robot self localization and navigation [20, 11], video surveillance [5] and humanoid foveal vision [18]. One drawback is that images suffer strong distortions 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 are among the most successful in tracking and pose estimation applications [19]. However, classical 3D model-based tracking methods are strongly dependent on the projection models and, thus, are not easily applicable to 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 [19]. This process involves the linearization of the relation between state and measurements, which can be very complex with omnidirectional sensor geometries. These approaches have limited convergence basins requiring either small target motions or very precise prediction models.

In this paper we try to overcome these problems by addressing the pose estimation and tracking problem in a Monte Carlo sampling framework [9], through the use of a Particle Filter (PF).

Particle filters (PF) have become a popular tool in image processing and computer vision communities as a way to infer time varying properties of a scene from a set of input images. PF compute a sampled representation of the posterior probability distribution over the scene properties under interest. It is capable of coping with non-linear dynamics and nonlinear observation equations, being easy to maintain uncertainty with multiple distributions.

The principle of the PF is simple. The goal is to compute the posterior probability distribution $p(x_t|y_{1:t})$ over an unknown state x_t, conditioned on image observations up to that time instant, i.e., $y_{1:t}$. Particle filtering works by approximating the posterior density using a discrete set of samples, i.e., the states. Each state corresponds to some hypothesized set of model parameters. Each sample is typically weighted by its likelihood $p(y_t|x_t)$, the probability that the current state observations were generated by the hypothesized state. Each sample can be considered as a state hypothesis, whose weight depends on the corresponding image data. The method operates by "testing" state hypothesis, thus avoiding the linearization between state and measurements required in gradient optimization techniques. In the context of our problem, this allows the utilization of arbitrarily complex projection models.

More precisely, 3D object localization hypothesis, generated by the particle filter, allow obtaining the 2D appearance of the objects, provided one has the projection models of the imaging sensors. Examples of imaging sensors used in our experiments range from conventional perspective to lens-mirror (catadioptric) and fisheye-lens based omnidirectional cameras. Note that while the perspective projection is already a non-linear model, in the sense that constant 3D motion increments of an object do not imply constant 2D image increments, it is still a simple model in the sense that the projection of any 3D point is ray-traced as a straight line passing through one single point. This is not true in general for fisheye or catadioptric cameras. Fisheye lenses bend the incoming principal optical rays progressively more while moving towards the periphery of the field of view. Catadiotric cameras use simple lenses, but the mirror implies a reflection according to the its local slope. In all cases one can have strong, anisotropic, geometrical distortions associated to the projection which bend and blur the object's visual features in a space variant manner, creating difficulties in the search for object's features. In our approach, adopting a 3D tracking context, we can use the projection model to predict and test the location of features in the images instead of locally searching for them.

1.1. Related Work

Particle filtering methods have been extensively used during the last decade in 2D tracking applications. One of the first algorithms applying particle filters in the 2D context was the Condensation algorithm [15]. In that work targets were modeled with a contour based appearance template. The approach proved not very robust to occlusion and hard to initialize. To address such limitations, more

recent works added other types of features such as color histograms [33] and subspace methods [16]. In [34] a hierarchical particle filter is used to track objects (persons) described by color rectangle features and edge orientation histogram features. Since the use of multiple cues increases the computational demands, two optimized algorithmic techniques are employed: integral images are used to speed-up feature computation and an efficient cascade scheme is used to speed-up particle likelihood evaluation. To deal with occlusions [32] fuses color histogram features acquired from multiple cameras. Cameras are pre-calibrated with affine transformations and the state of the filter is composed by the 2D coordinates and bounding box dimensions of the target in one of the cameras. Each particle in the filter generates a bounding box on each camera and the observation model is composed by the concatenation of the color histograms in all bounding boxes. It is shown that multiple cameras' tracking and data fusion are able to tackle situations when the target is occluded in one of the cameras.

Despite the success of particle filters in 2D tracking applications, not many works have proposed their use in a 3D model-based context. In [6] it is proposed a system for estimating the time varying 3D shape of a moving person having 28DOF. To deal with the high dimensionality, hybrid Monte Carlo (HMC) filter is used. However in that work observations were obtained directly from the 2D projection of suitably placed markers in the human body and, therefore is only applicable in restricted cases. In general settings, the true appearance of an object (e.g. color, shape, contours) must be taken into account. For instance, the work in [17] uses a particle filter [15] to implement a full 3D edge tracking system using fast GPU computation for real-time rendering of each state hypothesis image appearance (visible edges) and for applying edge-detectors in incoming video stream. In [27] the computation of edges in the full image is avoided by grouping line segments from a known model into 3D junctions and forming fast inlier/outlier counts on projected junction branches. A local search for edges around the expected values must be performed at each time step. In [25] a particle filter is used to estimate the 3D positions of humans. The environment is explicitly modelled to handle occlusions caused by fixed objects. Multiple fixed cameras and background subtraction techniques are used for the computation of the likelihood.

1.2. Our Approach

Most previous works on 3D-model based tracking, both the ones based on non-linear optimization and the ones relying in sampling methods, require the extraction of edge points from the images, either by processing the full image with edge detectors or performing local search for edge points. We stress that the

primary disadvantage of edge-based tracking algorithms is the lack of robustness to motion and optical blur. These effects are frequent in robotics applications due to the mobility of the devices and the frequent out-of-focus situations.

We formulate the problem differently. Rather than determining explicitly the location of edge points in the image, we compute differences between color histograms on the inside and outside regions of the projected target surfaces. We do not require explicit image rendering, in the sense of creating an image of the expected appearance of the target as in [17]. Instead we just need to compute the 3D to 2D projection of some selected points inside and outside the target's visible surfaces. This can be easily done with any single projection center sensor, and allows a fast evaluation of each particle's likelihood. Because explicit edge or contour extraction is avoided, the method is more robust to blur arising either from fast object motions or optical defocus. Altogether, our approach facilitates the application to arbitrary non-linear image projections and settings with fast target/robot motion.

In our approach we use color features to compute state likelihoods. 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 suitable for real-time applications; finally, many robotics research problems assume objects with distinctive colors to facilitate the figure-ground segmentation problem, for instance in cognitive robotics [22] or robotic competitions [28]. Notwithstanding, the approach is general and, in other scenarios, additional features could be used.

In [28] we have presented the first application of our method for tracking balls (spheres) and robots (cylinders) in the RoboCup Middle Size League scenario with catadioptric sensors. In [31] we have compared the application of two well known approximations of the projection function for the class of perspective catadioptric mirrors in our tracking framework: the unified projection model and the perspective model. In [29] we extended the approach to consider not only the 3D position but also the orientation of the targets and presented applications with dioptric and perspective cameras with radial distortion to track both spherical and convex polyhedral shapes. In [30] we have extended the observation model to track objects without a initial color model (only the shape model is used) and have extended the motion model to consider the observer's self-motion. In the present work we synthesize the main results derived in previous work and provide a better characterization of the method's performance in challenging scenarios, including the comparison with an alternative state-of-art technique and quantitative

evaluations with ground truth data.

The paper is organized as follows: Section 2 presents common imaging systems used in robotics and corresponding projection models. Section 3 describes the Particle Filtering approach and the state representation in our problem. In Section 4 we detail the 3D shape and color-based observation model used in the tracking filter. In Section 5 we show several experiments that illustrate the performance of the system in realistic scenarios, including a comparison to an alternative approach with ground truth data. In Section 6 we draw conclusions and present directions for future work.

2. Imaging Systems

We start with a brief introduction to the most common imaging geometries employed in robotics that were used in our experiments. We focus on imaging systems with axial symmetry, both dioptric and catadioptric. Cameras with axial symmetry can be described by a projection function $\mathcal{P}$:

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

where the z axis coincides with the optical axis, $[r\ \varphi\ z]^T$ represents a 3D point in cylindrical coordinates, ρ is the radial coordinate of the imaged point (the angular coordinate coincides with angular coordinate of the 3D point, φ) and ϑ is a vector of parameters characterizing the geometry of the system (see Figs. 1a and 1c).

2.1. Dioptric Systems

Conventional lens-only systems, i.e., dioptric systems, have been traditionally described by the pin-hole or perspective projection model (PPM):

$$\rho = k \cdot \frac{r}{z} \tag{2}$$

where k includes the camera's intrinsic calibration parameters. 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 [13, 4]. For example [13] uses a super-fisheye Nikon F8 and proposes the model $\rho = a \cdot \tan(\theta/b) + c \cdot \sin(\theta/d)$, where θ denotes the angle of an incoming optical ray with the optical axis (see Fig. 1c) and (a, b, c, d) are parameters characterizing the optical system, obtained by a calibration procedure. Some recent advances allowed companies to build

super-fisheye lenses having simple projection models. For example the Sunex's DSL215 lens, Fig. 1d, is designed to have a Constant Angular Resolution Projection Model (CARPM, termed *equidistant* in [13]), meaning that the radial coordinate of the image points ρ is in a linear relationship with the direction of the incoming optical ray, θ:

$$\rho = f \cdot \theta = f \cdot \arctan(r/z). \tag{3}$$

2.2. Catadioptric Systems

The conjunction of lens-based systems with mirrors, i.e. catadioptric systems, allows the acquisition of wide-angle and omnidirectional images with relatively simple and cheap setups. As compared to dioptric systems, the mirror introduces a reflection on the incoming optical rays implying that the directions θ observed by the camera effectively correspond to (distinct) directions, ϕ, in the optical rays projecting 3D points towards the mirror (see Fig. 1a).

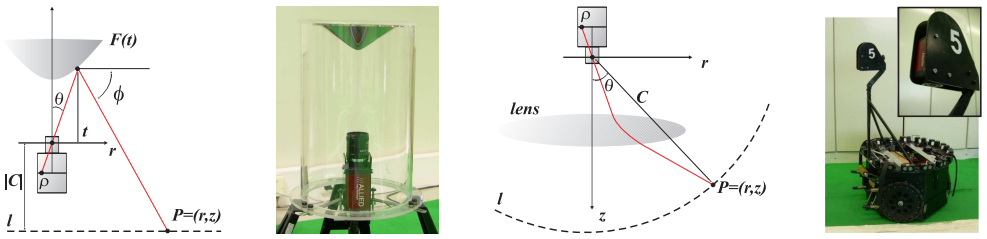
Deriving $\mathcal{P}$ for the complete 3D field of view (FOV) of a catadioptric system involves using the actual mirror shape, F, which is a function of the radial coordinate t. Based on first order optics, particularly on the reflection law at the specular surface of revolution, (t, F), we have:

$$\arctan(\rho) + 2 \cdot \arctan(F') = \phi \tag{4}$$

where $\arctan(\rho) = \arctan(t/F) = \theta$ is the angle of the principal optical ray with the optical axis of the system, $\phi = -\arctan((r-t)/(z-F))$ is the angle of the principal optical ray before the reflection on the mirror, and F' represents the slope of the mirror shape. If F is a known shape then Eq. (4) describes a generic Catadioptric Projection Model (CPM), as it forms an equation on ρ for a given 3D point (r, z).

In general there is no closed-form solution for such model. 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 ρ from (r, z). An interesting solution is proposed in [14, 10]), where mirrors are designed to have a constant resolution, wide-angle view of the ground plane. Replacing $\rho = t/F$ and $\phi = -\arctan((r-t)/(z-F))$, 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 using numerical integration methods [14, 10].

A closed form model that provides good approximations to the general CPM is given by the Unified Projection Model (UPM) [12]. It represents all omnidirectional cameras with a single center of projection and consists of a two-step



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

Figure 1: Catadioptric and dioptric cameras. (a,b) CPM geometry and setup. FOV of about $5m \times 9m$ and camera $0.6m$ above the ground. (c,d) CARPM geometry and setup.

mapping via a unit-radius sphere [12]: (i) a 3D world point, $P = [r\ \varphi\ z]^T$, is projected orthogonally to the sphere surface onto a point P_s; (ii) projected 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:

$$\rho = \frac{l+m}{l\sqrt{r^2+z^2}-z} \cdot r \tag{5}$$

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) $F = const$ [1, 3]. Note that the PPM is a particular case of the UPM, obtained by setting $l = 0$ and $m = -k$.

Concluding, in the context of our tracking methodology we can use arbitrarily complex projection models. Examples of models used in our experiments comprise the PPM, CARPM, CPM and UPM, described by Eqs.(2) to (5), or by the general expression Eq.(1).

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 $\{\mathbf{x}_t^{(i)}, w_t^{(i)}\}_{i=1}^{M}$ [9]:

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

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:

$$\hat{\mathbf{x}} \doteq \frac{1}{M} \sum_{i=1}^{M} w_t^{(i)} \mathbf{x}_t^{(i)} \approx \mathbb{E}(\mathbf{x}_t \mid \mathbf{y}_{1:t}), \tag{8}$$

or the maximum likelihood estimate:

$$\hat{\mathbf{x}}_{\mathrm{ML}} \doteq \arg\max_{\mathbf{x}_t} \sum_{i=1}^{M} w_t^{(i)} \delta(\mathbf{x}_t - \mathbf{x}_t^{(i)}). \tag{9}$$

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 distribution, giving no privilege to any direction of motion:

$$p(\mathbf{X}_t|\mathbf{X}_{t-1}) = \mathcal{N}(\mathbf{X}_t|\mathbf{X}_{t-1}, \Lambda) \tag{10}$$

where $\mathcal{N}(\cdot|\mu, \Sigma)$ stands for a 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 been widely used (e.g. [2, 26]).

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:

$$\mathbf{X}_{t+1} = \begin{bmatrix} \mathbf{I} & (\Delta t)\mathbf{I} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \mathbf{X}_t + \begin{bmatrix} (\frac{\Delta t^2}{2})\mathbf{I} \\ (\Delta t)\mathbf{I} \end{bmatrix} a_t \tag{11}$$

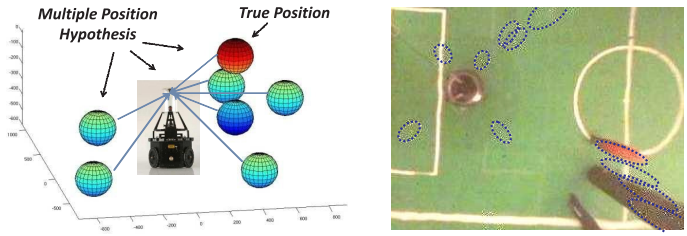
where I is the 3×3 identity matrix, $\Delta t = 1$, and a_t is a 3×1 white zero mean random vector corresponding to an acceleration disturbance. The covariance matrix of the random acceleration vector is usually set experimentally as $\text{cov}(a_t) = \sigma^2\mathbf{I}$.

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, η_p is Gaussian noise and η_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.

4. Observation model

In this section we describe the observation model, as expressed by $p(y_t|x_t)$ in (6). We propose a methodology that associates likelihood values to each of the



(a) (b)

Figure 2: The observation process projects the boundaries of the multiple 3D target hypothesis (a) into the image plane (b) and assigns likelihood values to each of them, as a function image color distributions inside and outside their boundaries.

samples in the particle filter using the target's 3D shape and color models. Recall that each particle represents an hypothesis of the target's position and pose. The likelihood function will assign high weights to a particle if the image information is coherent with its 3D position and pose hypothesis, and will be low otherwise. For instance, when tracking a red ball, an hypothesis whose projection is within a certain image region will have low likelihood if few red pixels are in that region. It will also be low if many pixels outside the region are equally red.

Figure 2 illustrates this process. Boundaries of the multiple 3D pose hypothesis generated by the particle filter are projected onto the image plane with the sensor's projection function (we will explain latter that only some points inside and outside the boundary are projected, not the whole boundary). For visualization purposes, only a limited number of samples is shown – in practice a much higher number of samples is generated.

The likelihood function takes into account the 3D shape and the color properties of the targets. The rationale is to measure the similarity/dissimilarity between the colors of the target and the ones in the image regions corresponding to the boundaries of each hypothesis in the particle filter. High likelihood values should only be assigned to particles if many pixels inside the region, and few outside, match the target's color model. High likelihood particles will survive and be replicated in the next time step while low likelihood particles will be rejected. An appropriately designed likelihood function is thus a crucial aspect in the convergence properties of the particle filter. It should be selective enough to quickly converge to a good approximation of the posterior (6). However it must not be too selective, otherwise particles close to the solution (but not close enough) will receive very low weights and will be rejected. Small mismatches in position and posture should nevertheless be accepted, despite having lower likelihood values

that the optimal solution. We take these criteria into account in the definition of the likelihood function.

In practice we do not project the whole boundary contour in the image plane since boundary parameterizations may be hard to compute in arbitrary non-linear sensors. Instead we define sets of 3D points, corresponding to specific regions of the target's surface or around it. These points are computed as a function of the object 3D shape and the particle's position and orientation with respect to the imaging system. These sets of 3D points, are projected onto the image plane, generating corresponding sets of points in 2D image coordinates. Then 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. The only image information required in such process is the color of a few hundreds of pixels per particle. On the contrary to other existing methods, it only uses pixel values thus avoiding any other image processing operations, 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. In the consecutive sections we explain in depth each of the steps leading to the computation of particle's likelihood.

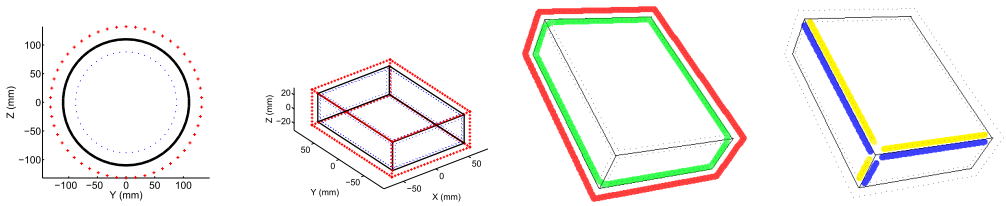
4.1. 3D Points Generation and Their Projection onto the Image

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 polyhedra by exploiting the current knowledge in computer graphics [17].

For each state hypothesis $\mathbf{X}_t^{(i)}$, and given the particular 3D geometric object model, a few sets of 3D points are generated $\mathbf{U}_t^{(i)j}(\alpha)$, where t is the time step, i is the particle index, j indexes the points in the objects contour vicinity, and α represents a part of the object, usually the inside or outside of a surface. Then we convert the sets of 3D points $\mathbf{U}_t^{(i)j}(\alpha)$ to the corresponding sets of 2D points $\mathbf{u}_t^{(i)j}(\alpha)$ from which the likelihood measurements will be collected.

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. To each edge we associate a set of points that lie on the corresponding edge of an expanded polyhedron (see Fig. 3b). The 3D



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

Figure 3: Sphere and polyhedron 3D model. (a) Location of the internal (blue dots) and external (red crosses) 3D points sets for the sphere, with respect to the great circle of the sphere (black line). (b) Location of the 3D points of the polyhedron model: the points lying on the faces of the object are represented as blue dots, while the points lying on the edges of the expanded object are represented as red crosses. (c) One particular projection of the polyhedron.The areas sampled in the image to build the internal and external histograms are highlighted. (d) The same projection of the polyhedron with the pairs of areas associated with non-silhouette edges highlighted.

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 and 3d) . 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 (see Fig. 3a).

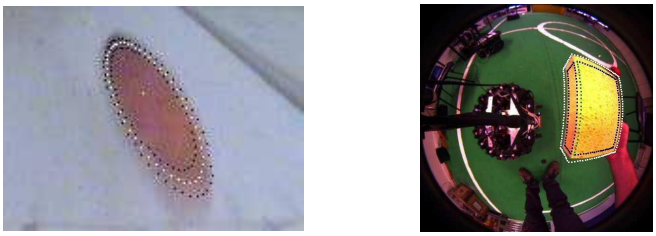
The projection of the 3D points onto the image is performed using the appropriate projection model, as detailed in Section 2:

$$\mathbf{u}_t^{(i)j}(\alpha) = \mathcal{P}(\mathbf{U}_t^{(i)j}(\alpha)) \tag{12}$$

Fig. 4 shows examples of the obtained image points for a particle of the exact posture.

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 the B-bin reference (object) histogram



(a) (b)

Figure 4: Examples of image points used to build the color histograms used for particle likelihood computation. Obtained from the projection of the 3D points shown in Fig. 3. (a) For spheres. (b) For cuboids.

model in channel $c \in \{H, S, I\}$ by $\mathbf{h}^c_{ref} = (h^c_{1,ref}, \ldots, h^c_{B,ref})$. An estimate for the histogram color model, denoted by $\mathbf{h}^c_{\mathbf{x}} = (h^c_{1,\mathbf{x}}, \ldots, h^c_{B,\mathbf{x}})$, can be obtained as

$$h^c_{i,\mathbf{x}} = \beta \sum_{\mathbf{u} \in \mathcal{U}} \delta_a(b^c_{\mathbf{u}}), \quad i = 1, \ldots, B. \tag{13}$$

$\mathcal{U}$ is the region where the histogram is computed; $b^c_{\mathbf{u}} \in \{1, \ldots, B\}$ 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 $\mathbf{h}^{\text{model}}$, $\mathbf{h}^{\text{in}}$ and $\mathbf{h}^{\text{out}}$ as a reference (object) histogram, the inner boundary points and the outer boundary points histogram, respectively. We define $\mathbf{h}^{\text{sideA}}_i$ and $\mathbf{h}^{\text{sideB}}_i$ as the histograms of each of the two sides of the i^{th} non-silhouette edge (see Fig. 3d). To measure the difference between histograms we use the Bhattacharyya similarity as in [7, 24]:

$$\mathcal{S}(\mathbf{h}_1, \mathbf{h}_2) = \sum_{i=1}^{B} \sqrt{h_{i,1} h_{i,2}} \tag{14}$$

We define:

$$\mathcal{S}_0 = \mathcal{S}(\mathbf{h}^{\text{model}}, \mathbf{h}^{\text{in}}), \quad \mathcal{S}_1 = \mathcal{S}(\mathbf{h}^{\text{out}}, \mathbf{h}^{\text{in}}), \quad \mathcal{S}_2 = \frac{\sum_{i=0}^{n} \mathcal{S}(\mathbf{h}^{\text{sideA}}_i, \mathbf{h}^{\text{sideB}}_i)}{n} \tag{15}$$

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

$$\mathcal{D} = \left(1 - \frac{\mathcal{S}_0 + \kappa_1(1 - \mathcal{S}_1) + \kappa_2(1 - \mathcal{S}_2)}{\kappa_1 + \kappa_2 + 1}\right) - \gamma \tag{16}$$

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

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 κ_1 and κ_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 and second experiment and to 0 and 0 in the third one (thus ignoring the reference color model). The term γ is useful when tracking an object whose projection lies only partly on the image. 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 $\epsilon = 1/30$.

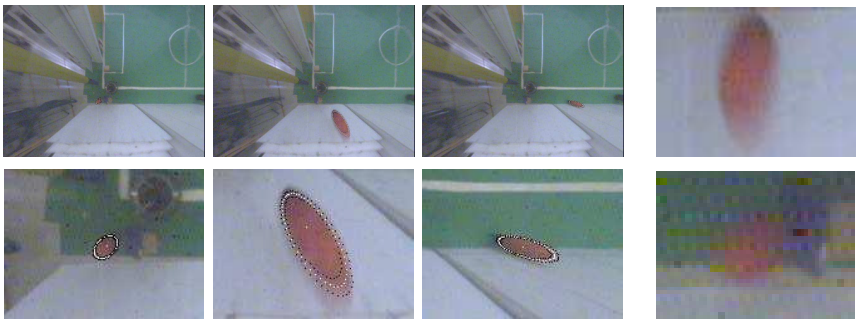
5. Experimental Results

This section presents an evaluation of the proposed methods. Firstly we present results taken with omnidirectional cameras: the tracking of a ball performed with a catadioptric setup and the tracking of a cuboid in a dioptric setup. Secondly we present results with conventional cameras and perform a comparison between our method and a competing alternative based on 2D tracking followed by 3D reconstruction. Is this set of experiments we show results with real and artificial images, both with ground truth. Finally we show results taken from a mobile platform equipped with a dioptric vision system, tracking a ball of unknown color.

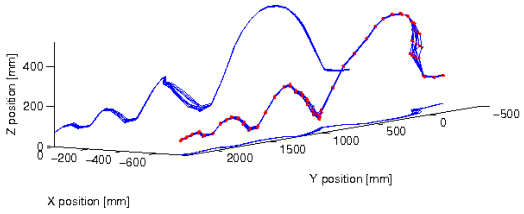
The first and the last sets of experiments are taken in very unconstrained scenarios and their evaluation is not supported by ground truth values of the target's trajectories. Evaluation of the results is made only on a quantitative basis by empirical observations of the system's performance. On the contrary, the second set of experiments is supported by ground truth values and results are evaluated quantitatively with the mean squared tracking error. In the experiments with real images, ground truth is obtained manually by measuring the 3D coordinates of some points along the trajectory. In the experiments with artificial images, target's true 3D coordinates are provided by the simulation engine at each time step.

5.1. Omnidirectional Cameras

Here we illustrate qualitatively the performance of the methods with omnidirectional images with large nonlinear distortions, blur and noise. The sequences



(a) (b)



(c)

Figure 5: 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.

were collected from setups usually employed in RoboCup Middle Size league scenarios.

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 series 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 640×480 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. Processing one frame took 11 seconds on a 2GHz Pentium 4 CPU, i.e. 1.1ms per particle. 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. 5a), due to the nature of the catadioptric system used. The images are affected by both motion blur and heavy sensor noise (see Fig. 5b). We repeated the tracking 10 times on the same image sequence to assess how the stochastic component of the tracker influences the precision of the 3D estimated paths (see Fig. 5c), with satisfactory results.

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 640×480 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. Processing one image took 33 seconds on a 2GHz Pentium 4 CPU, i.e. 6.6ms per particle. The tracker managed to follow the cuboid along rotations and translations that greatly affect its projection onto the image, see Fig. 6.

5.2. Comparative Study: Perspective Cameras with Ground Truth

Here we evaluate our algorithm quantitatively with ground truth data and compare its performance against an alternative approach: a 2D tracker followed by a 3D reconstruction algorithm. The 2D tracker is based on a MMDA (Multiple Model Data Association) methodology [23] and outputs a collection of points uniformly sampled at the target's contour. In each image, the tracked contour is expanded in the 3D cone generated by the camera optical center and the sphere's occluding contour. Then, knowing the sphere's radius, simple trigonometric calculations allow the reconstruction of its 3D position.

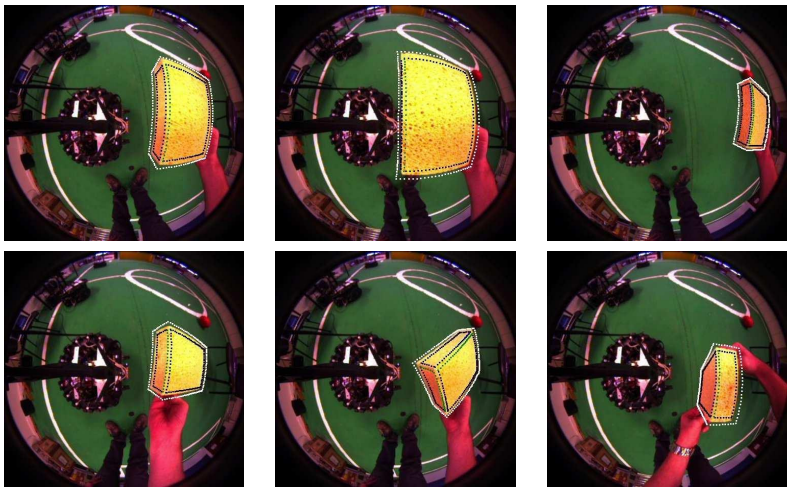


Figure 6: Cuboid tracking in the dioptric setup. Six frames of the sequence with the pixels used to build color histograms highlighted.

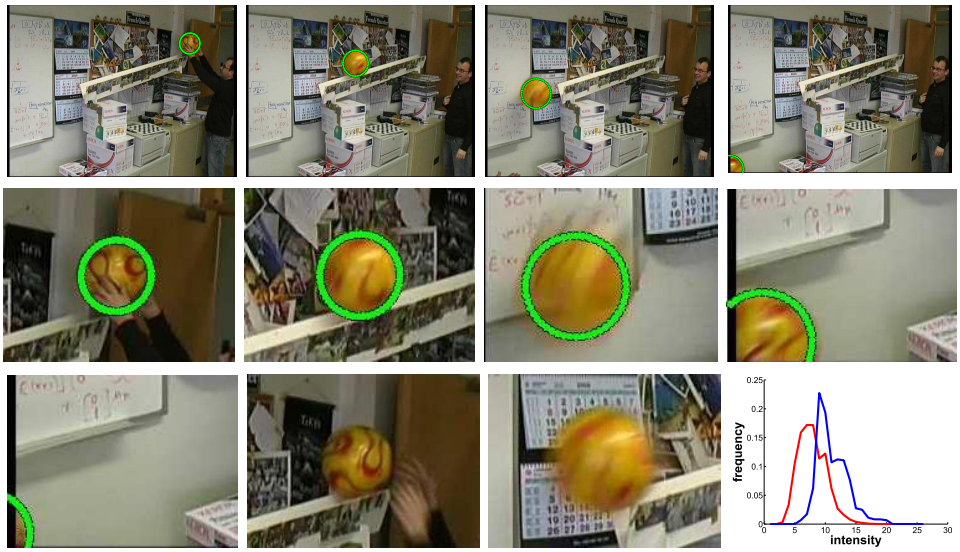


Figure 7: Non-uniformly colored ball tracking on cluttered background. Four frames of the sequence with the estimated contour highlighted in green (top row). Close-up's on the ball showing the behaviour of the tracker in challenging conditions: partial occlusion, heavy background clutter, motion blur, projection of the ball lying partly outside the image (middle row). One close-up showing the behaviour of the tracker when only a small part of the projection of the ball lies on the image. Two frames showing the different illumination conditions at the beginning and at the end of the ramp. A plot of the intensity histogram built on the area occupied by the ball at the beginning of the ramp, in red, and at the end of the ramp, in blue (bottom row).

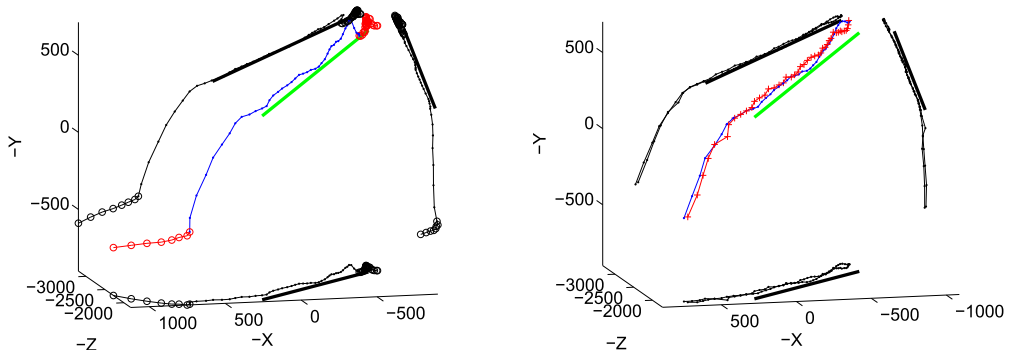


Figure 8: (a) 3D plot of the trajectory estimated with PF along the whole image sequence. The parts of the trajectory during which the ball is occluded are plotted with red circled lines, while the part in which the ball is fully visible is plotted as a blue dotted line. Ground truth along the ramp is represented with a green line, the projection of each line on the ground, lateral and vertical plan is shown in black. (b) 3D plot of the trajectories estimated with PF (blue dotted line) and the MMDA-based tracker (red crossed line). Because the MMDA-based method can not deal with occlusion, both the beginning and the end of the sequence (where occlusions are significant and the MMDA tracker fails) are not shown in this plot. Ground truth is plotted in green, projections on the plans are plotted in black.

Tracking a Non-Uniformly colored Ball in 3D on Cluttered Background. In this experiment we tracked a non-uniformly colored ball rolling down a ramp. The path of the ball consists of four different parts: in the first part the ball is on top of the ramp an partially occluded by the hand of an operator, in the second part the ball rolls down the ramp, in the third part the ball performs a free fall, while in the fourth it is caught by an operator and is gradually moved out of the image. The image sequence is affected by heavy background clutter, occlusions, motion blur and changes in illumination (see Fig. 7). It was acquired with a Canon XM1 camera with a frame rate of 25fps and a resolution of 640×480 pixels. The tracker used 10000 particles, a version of the Perspective Projection Model that takes into account radial and tangential distortion, and the constant velocity motion model described in Section 3.1. Processing was done off-line, with Matlab. Processing one image took 21 seconds on a 2GHz Pentium 4 CPU, i.e. 2.1ms per particle. The tracker was provided with the color model for the ball.

The proposed method successfully tracked the ball along the whole sequence (see Fig. 8a). We compared the performance of the proposed tracker with the MMDA-based method, finding that they produce similar results (see Fig. 8b).

Method	Precision (RMSE)
PF	0.035m
MMDA	0.017m

Table 1: Precision of the 3D tracking methods (PF - Particle Filter, MMDA - Multiple Model Data Association) during the linear part of the ramp descent experiment. Precision is measured by the root mean squared error (RMSE) between the measured trajectory and the best fit linear trajectory. Notice that this error metric does not take into account constant bias terms (systematic errors).

This comparison, though, could only be performed on the part of the sequence in which the ball is fully visible, as the MMDA tracker is not robust to occlusions.

The proposed method is able to cope with occlusions, although the 3D estimate is less accurate: the gradual occlusion during the last part of the path induces an increasing localization error, but that is difficult to quantify as the motion of the ball was constrained by an operator catching it.

To evaluate the precision of both methods during the linear part of the ramp descent, we have computed the root-mean-squared error of the estimated position with respect to a "ground truth trajectory" obtained by fitting a line to the observed 3D points. Results are presented in table 1. Furthermore, we compared the performance of the two methods along the first and second half of the linear path, assessing that the increasing motion blur does not affect it. The RMS error for the PF approach is smaller during the second half of the linear path (0.039m and 0.032m, for the first and second halves respectively), as it is for the MMDA method (0.019m and 0.016m, respectively). Eventually, we measured the precision of the PF approach during the frames in the beginning of the sequence while the ball is partially occluded and still, obtaining a value of 0.033m.

We notice that both methods have low tracking errors. The PF filter is less precise that the MMDA method in this particular experiment, this is due to an arbitrary choice of the tuning parameters. By appropriately tuning the acceleration covariance noise it would be possible to change the precision of the methods. What we would like to stress in this experiment is that the PF method is able to cope better with significant levels of occlusion whereas the MMDA method fails.

Tracking a Non-Uniformly colored Ball in 3D in Simulated Images with Ground Truth. In this experiment we tracked a non-uniformly colored ball along a spi-

Method	Accuracy (RMSE)
PF	0.0192m
MMDA	0.0712m

Table 2: Accuracy of the 3D tracking methods (PF - Particle Filter, MMDA - Multiple Model Data Association) for the virtual ball tracking experiment. Accuracy is measured by the root mean squared error (RMSE) between the measured trajectory and the true synthesized trajectory. Both systematic errors (bias) and random errors (variance) are taken into account on the error metric.

raling path. The images were generated with a virtual reality software[1], so the intrinsic parameters of the camera and the 3D position of the ball at each given time are known exactly. The image sequence was generated with a resolution of 576×380 pixels. The tracker used 10000 particles, the Perspective Projection Model, and the constant velocity motion model. Processing was done off-line, with C++. Processing one image took 0.26 seconds on a 2GHz Pentium 4 CPU, i.e. 0.026ms per particle. The tracker was provided with the color model for the ball. The results from simulation experiments can be seen in Figure 9.

The PF and MMDA methods were compared in terms of root-mean-squared error to the true trajectory. Whereas in the previous experiment we were able to evaluate only the precision of the methods (repeatability), in this experiment we are able to quantify accuracy (truthfulness).

In comparison to the MMDA-based method, we notice a better match with the ground truth, despite a slightly larger variance. Quantitatively, the average error is much smaller with the proposed tracking method. Table 2 presents the obtained accuracy values.

The MMDA technique was designed to track approximately circular shapes in the image. Due to a large focal distance of the camera used in this experience, there is a high perspective distortion and the 3D sphere projects to the image plane into an ellipse whose eccentricity varies significantly with position. This produces a space variant bias in the estimated 3D position that was not so obvious in the previous experiment. On the contrary, our method does not suffer from this effect because it uses the true 3D space invariant shape of the object instead of its 2D space variant projection. It is, therefore, more easily applicable to different kinds of sensors.

[1]The authors would like to thank Simon Day for granting them the possibility to use one of his images as wallpaper in this experiment.

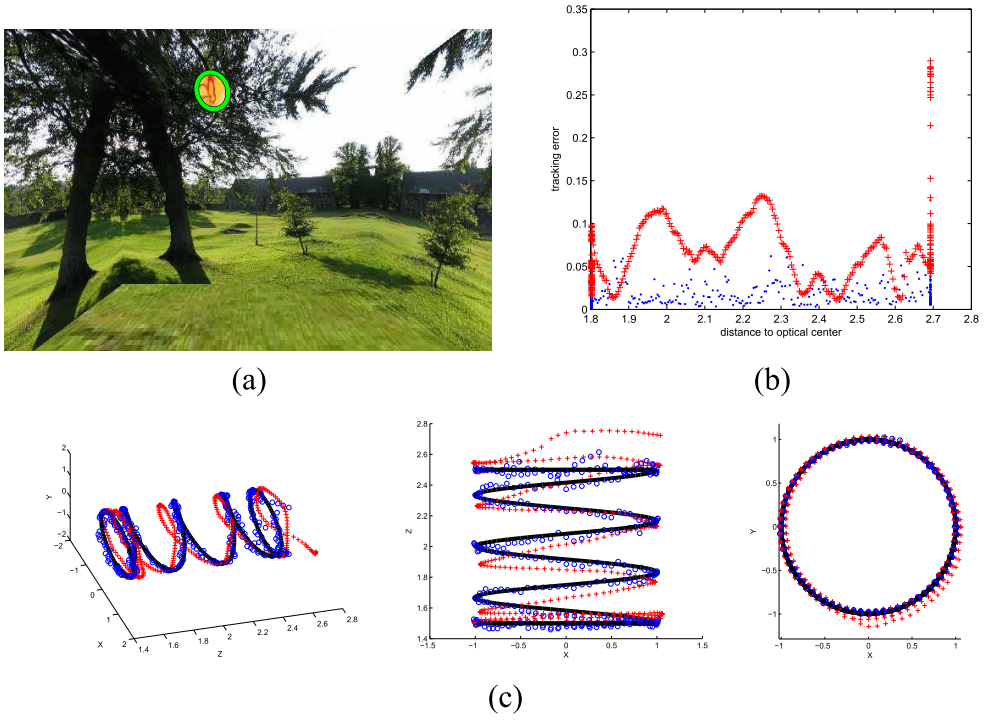


Figure 9: Non-uniformly colored ball tracking with ground truth. (a) One frame of the sequence with the estimated contour highlighted in green. (b) Plot of the tracking error as a function of distance to the projection center for PF (blue dots) and the MMDA-based approach (red crosses). (c) Plot of the tracked path of the ball estimated with PF (blue dots) and the MMDA-based approach (red crosses). Each graph represents a different view of the reconstructed 3D trajectory. The left plot is a general perspective view, the middle plot is a projection in X,Z plane (top view) and the right plot is a projection in the X,Y plane (front view). The ground truth trajectory is represented by the black line.

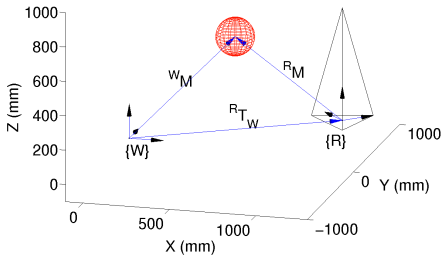


Figure 10: Position of one sphere in the world and robot reference frames.

5.3. Arbitrary Color Ball Tracking with a Moving Robot

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.

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 ${}^{R}t_{W}$.

$$ {}^{R}T_{W} = \begin{bmatrix} {}^{R}R_{W} & {}^{R}t_{W} \\ 0 & 1 \end{bmatrix} \tag{17} $$

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:

$$ m = \mathcal{P}({}^{R}M) = \mathcal{P}({}^{R}T_{W} \cdot {}^{W}M). \tag{18} $$

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.

The tracker was implemented in the soccer robot team ISocRob [8] 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, moved around by a human. The robot is constantly moving towards the ball without catching it.

In this implementation the observation model discards the object-to-model mismatch, described in Eq.(15), 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. 11a, but other colored balls, e.g., orange, could be used.

Images on the robot were acquired with the dioptric omnidirectional camera used in Section 5.1, 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. 11b [2]. In the figure, the robot posture and ball position are plotted in a frame located on the center of the soccer field (*field frame*). The ball coordinates were transformed from the *robot frame* to the *field frame* and the robot self-localization postures (obtained using Monte Carlo Localization [21] – MCL) were expressed in the *field frame*, to plot the ball at each update step. The ball trajectory is not smooth due to the robot self-localization errors. Ball shifts in the plot correspond to equal shifts in the robot posture, when MCL corrects the robot posture, since the transformation of the ball position from the *robot frame* to the *field frame* is based on the current estimate of the robot posture by MCL.

6. Conclusions

We presented a 3D model-based tracking system, based on a Particle Filter framework. The method requires the knowledge of the 3D shape of the target and the imaging sensor calibration. We stress 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. In practice the method can be used with any projection model given the availability of ways to compute the projection of 3D points to sensor coordinates. We have shown applications with different kinds of omnidirectional vision sensors (dioptric and catadioptric), with different objects (spheres and polyhedra) and with moving observers. The proposed methods were evaluated in challenging real and synthetic sequences with clutter, partial occlusions, image blur and non-uniformly colored objects. The observation model assumes distinct target and background color distributions and benefits, but does not depend critically, on their knowledge. We have shown a case

[2] see also a video taken online – at 2 fps – from the robot camera during this same experiment, at http://www.youtube.com/watch?v=szReXrJUeYQ&feature=channel

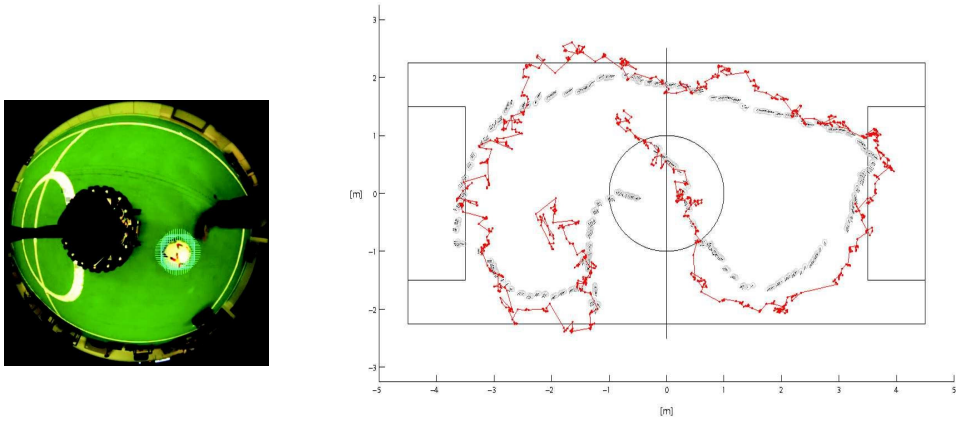


Figure 11: Tracking a white ball with a moving robot. (a) Detection of a white ball based on the object-to-background dissimilarity, where the light green crosses mark the pixels used to build the background histogram. (b) Plot of the paths of robot and ball. The robot starts in the middle circle facing 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.

where the target color model is absent and tracking relies only on the contrast between target and background color. Anyway having known and distinctly colored objects, naturally improves the robustness of the tracker and makes it specially suited to color engineered environments as is the case of robotic competitions.

The proposed method was compared with a state-of-the-art 2D tracker followed by 3D pose estimation, in the problem of tracking spheres. The two methods have similar results whenever the 2D tracker reliably computes the image contour. However, the proposed methodology has proven more robust to partial occlusion. Additionally, our method is advantageous in terms of the customization required to cope with objects of different shapes. In our case we just have to recompute points in the inner and outer object boundaries, compute their visibility and project them to the sensor plane (a large body of computer graphics algorithms and hardware acceleration techniques are available for such purposes). Instead, contour based methods require 3D pose estimation from the 2D silhouette which often involves complex computations and is frequently an ill-posed problem.

In future work we will advance in two directions. First we will consider the use of multiple models for target motion. Despite a constant velocity model, as the one employed in this work, can cope with arbitrary accelerations by increasing

the model's acceleration noise, this has the side effect of increasing also the variance of the estimated trajectories. With multiple motion models we expect to have less noisy position estimates and still be able to cope with sudden target accelerations. Secondly we will investigate the possibility of applying information fusion methods to perform the collaborative tracking of objects in distributed robotics systems. Something we did not address in this paper was the consideration of localization errors in the robot/sensor position information. By appropriately combining the information of both robots' and targets' location and uncertainty, we aim at attaining a globally coherent and precise localization of robot teams.

References

[1] S. Baker and S. K. Nayar. A theory of single-viewpoint catadioptric image formation. International Journal of Computer Vision, 35(2):175–196, 1999.

[2] 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.

[3] R. Benosman and S. Kang. Panoramic Vision. Springer Verlag, 2001.

[4] B.Micusik and T.Pajdla. Estimation of omnidirectional camera model from epipolar geometry. In IEEE CVPR, pages 485–490, 2003.

[5] T. Boult, X. Gao, R. Michaels, and M. Eckmann. Omni-directional visual surveillance. Image and Vision Computing, 22(7):515–534, 2004.

[6] K. Choo and D. J. Fleet. People tracking using hybrid monte carlo filtering. In Proc. Int. Conf. on Computer Vision (ICCV'01), pages 321–328, 2001.

[7] 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.

[8] 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.

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

[10] 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.

[11] J. Gaspar, N. Winters, and J. Santos-Victor. Vision-based navigation and environmental representations with an omnidirectional camera. IEEE Transactions on Robotics and Automation, 16(6), 2000.

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

[13] 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.

[14] R. Hicks and R. Bajcsy. Catadioptric sensors that approximate wide-angle perspective projections. In IEEE CVPR, pages 545–551, 2000.

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

[16] Z. Khan, T. Balch, and F. Dellaert. A rao-blackwellized particle filter for eigentracking. In Proc. of Int. Conf. on Computer Vision and Pattern Recognition (CVPR'04), volume 2, pages 980–986, 2004.

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

[18] 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.

[19] 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.

[20] P. Lima, A. Bonarini, C. Machado, F. Marchese, F. Ribeiro, and D. Sorrenti. Omni-directional catadioptric vision for soccer robots. 2001.

[21] J. Messias, J. Santos, J. Estilita, and P. Lima. Monte carlo localization based on gyrodometry and line-detection. In ROBOTICA2008, 8th Conference on Mobile Robots and Competitions, 2008.

[22] 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.

[23] J. Nascimento and J. S. Marques. Robust shape tracking with multiple models in ultrasound images. IEEE Transactions on Image Processing, 17(3):392–406, 2008.

[24] 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.

[25] Tatsuya Osawa, Xiaojun Wu, Kaoru Wakabayashi, and Takayuki Yasuno. Human tracking by particle filtering using full 3d model of both target and environment. In ICPR (2), pages 25–28. IEEE Computer Society, 2006.

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

[27] 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.

[28] 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.

[29] 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.

[30] M Taiana, J. Santos, J. Gaspar, J. Nascimento, A. Bernardino, and P. Lima. Color 3D model-based tracking with arbitrary projection models. In Workshop Proceedings of SIMPAR 2008: Omnidirectional Robot Vision, pages 206–217, Venice, Italy, 2008.

[31] M. Tajana, J. Gaspar, J. Nascimento, A. Bernardino, and P. Lima. On the use of perspective catadioptric sensors for 3d model-based tracking with particle filters. In IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, pages 2747–2752, 2007.

[32] Ya-Dong Wang, Jian-Kang Wu, and Ashraf A. Kassim. Particle filter for visual tracking using multiple cameras. In Proc. of IAPR Conf. on Machine Vision Applications (MVA'05), pages 298–301, 2005.

[33] Y. Wu. Robust visual tracking by integrating multiple cues based on co-inference learning. Int'l Journal of Computer Vision, 58(1):55–71, 2004.

[34] Changjiang Yang, Ramani Duraiswami, and Larry Davis. Fast multiple object tracking via a hierarchical particle filter. In Proc. of Int. Conf. on Computer Vision and Pattern Recognition (CVPR'05), pages 212–219, 2005.