2007 IEEE/RSJ International Conference on Intelligent Robots and Systems,
San Diego, USA, October 2007

On the use of perspective catadioptric sensors for 3D model-based tracking with particle filters

M. Tajana, J. Gaspar, J. Nascimento, A. Bernardino and P. Lima

***Abstract*— We present a model-based 3D tracking system, using wide angle perspective catadioptric sensors. These sensors acquire 360^{o} views of the environment and the projection from 3D world points to the image plane is approximated by a perspective model. This is a major advantage in structured environments because straight lines on specific surfaces are not deformed by the sensor, allowing the application of standard computer vision algorithms. Objects off the surface are distorted according to a complex projection model, but can be approximated by a simple wide angle perspective mapping. This is exploited here to develop a robust tracking system for autonomous robots using a 3D shape and color-based object model. The use of particle filters allows tracking to be done with 3D realistic motion models and tackling object occlusion, overlap and ambiguities. We show that the use of the perspective model is advantageous over more standard catadioptric projection models, since it renders a very good approximation to the true model, being simpler and more efficient to use, in particular with 3D particle filtering methods.**

I. Introduction

Wide angle catadioptric sensors have often been used in robotics, especially for self localization and navigation [4],[2], as they gather information from a large portion of the space surrounding a robot. One drawback is that images are affected by strong distortion and perspective effects, which may force the use of non-standard algorithms for target detection and tracking. In structured scenarios, however, custom sensor designs have been proposed to avoid distortions in certain parts of the environment. For example, [11], [12] propose systems with wide-angle and constant-resolution view of the ground plane, such that structures in the floor can be treated as in conventional perspective images.

For objects off the ground floor, the constant resolution property does not hold anymore. If precise measurements are required, the exact projection model should be employed. Because the exact model involves complex non-linear and non closed-form relationships between 3D points and their 2D projections, approximations are often used. A widely used approximation for catadioptric systems is the Unified Projection Model (UPM) pioneered by Geyer and Daniilidis [9]. The UPM models all omnidirectional cameras with a single center of projection and provides a good approximation to wide-angle constant resolution sensors. We show, however, that a simple perspective projection model is as good as the UPM, with additional advantages of simplicity and computational efficiency.

To fully evaluate our system, we perform both simulations studies and experiments in real scenarios. We have developed a Particle Filtering method [5], [6], that is able to estimate the 3D position and velocity of structured objects, from a single catadioptric sensor. Particle filtering methods are among the state-of-the-art techniques for object tracking. Due to their principled probabilistic approach, they are able to tackle the uncertainty, ambiguity and complexity of cluttered environments. We fully describe the employed 3D particle filtering and observation models. Since PF's rely on the test of multiple hypothesis (usually several hundreds), using simple and computationally efficient measurement methods will have a high impact on the overall system computational cost. We show that the employed perspective model requires about half the computations for the same level of precision, when compared to the standard UPM.

The tracking system was inspired by the work of [8], which proposed a PF method based on 2D image coordinates. On the contrary, we perform tracking in 3D coordinates. This is advantageous in several scenarios, e.g. in the RoboCup MSL (Medium Size League), as robots are now provided with the ability to kick the ball off the ground. 3D tracking has two main advantages over 2D methods: (i) since the projection to 2D introduces non-linear effects, the 3D motion models are closer, and more physically grounded, to the actual object motion; (ii) with 3D tracking the actual position of the tracked object is directly available, while in image tracking a further non-trivial step is needed to compute it.

The paper is organized as follows. Section II describes the catadioptric sensor and projection models. The 3D particle filter tracking algorithm and image measurement models are described in Sections III and IV. Experimental results are shown in Section V and, finally, Section VI concludes the paper and presents ideas for future work.

II. Catadioptric Imaging System

In this section we describe the imaging system, detail its model, and propose two approximations using well known projection-models. Our catadioptric vision system, see Fig.1a, combines a camera looking upright to a convex mirror, in order to have omnidirectional view in the azimuth direction [10]. The system is designed to have wide-angle and constant-resolution view of the ground plane [11], [12].

Let $m = \mathcal{P}_0(M;\vartheta_0)$ represent the projection of a 3D point in cylindrical coordinates, $M = [r\ \varphi\ z]^T$ to 2D polar coordinates on the image plane, $m = [\rho\ \varphi_0]^T$, with ϑ_0

This work was partially supported by the Fundação para a Ciência e a Tecnologia (ISR/IST pluriannual funding) through the POS_Conhecimento Program that includes FEDER funds.

M. Tajana, J. Gaspar, J. Nascimento, A. Bernardino and P. Lima are with the Institute for Systems and Robotics, IST, 1049-001 Lisboa, Portugal {mtajana,jag,jan,alex,pal}@isr.ist.utl.pt

M. Tajana is also with the Politecnico di Milano, 20133 Milano, Italy

containing the system parameters. Considering cameras with axial symmetry and aligning the coordinate systems such that $\varphi \equiv \varphi_0$, one obtains the radial model (see Fig.1b):

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

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

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

where $\phi = -(r-t)/(z-F)$ is the system's vertical view angle, $\theta = \operatorname{atan}(\rho)$ 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 [11], [12]).

If F is a known shape then (2) describes a generic Catadioptric Projection Model (CPM), as it forms an equation on ρ for a given 3D point (r, z). In general ρ has a non-closed-form solution. Therefore, the CPM is usually implemented by finding one 3D point from ρ, i.e. doing a back-projection. We propose the following back-projection algorithm:

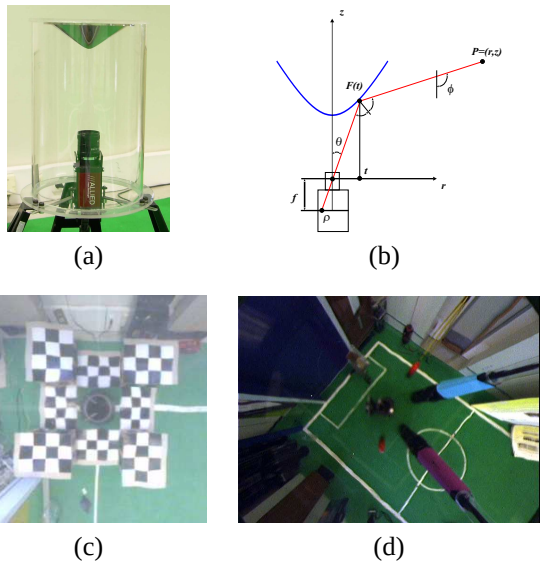
1) remove the intrinsic parameters from the image coordinates, i.e. convert an image point to the polar coordinates $[\rho\ \varphi]^T$ and then to the vertical angle θ,
2) compute, or interpolate from a look-up table, (t, F, F') given that $t/F = \tan(\theta)$, and finally
3) obtain ϕ using Eq.(2).

Note that, to fully define a 3D optical ray, both the direction ϕ and the point on the mirror surface, (t, F) are needed.

Next we will introduce two known models that approximate the CPM: the UPM and the standard Perspective Projection Model (PPM). The UPM consists of a two-step mapping via a unit-radius sphere [9]: (i) project a 3D world point, $P = [r\ \varphi\ z]^T$ to a point P_s on the sphere surface, such that the projection is normal to the sphere surface; (ii) project to a point on the image plane, $P_i = [\rho\ \varphi]^T$ from a point, O on the vertical axis of the sphere, through the point P_s. The mapping is mathematically 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) an hyperboloid or ellipsoid with focus at $(0, 0)$; (ii) a paraboloid combined with a telecentric lens ($\theta = 0$) or (iii) $F = const$ [10]. In our case, F is computed numerically to have the constant resolution property, and therefore does not correspond to any of the former cases. The comparison between CPM and UPM must be done in simulation.



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

Fig. 1. (a) Catadioptric camera. (b) The Generic Catadioptric Model. (c) Image used for calibration. (d) Sample taken in RoboCup MSL scenario.

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

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

Eq.(4) shows that the PPM has constant resolution, i.e. linear relationship between ρ and r, at all z-planes.

In Section V-A, we will use the CPM, with F for constant horizontal resolution, as a simulation methodology for evaluating the approximations by the UPM and the PPM models. The approximations allow assessing the resolution properties of the system, as e.g. the linear relationship between ground- and image-measured distances.

III. 3D Tracking with Particle Filters

This section introduces a model-based 3D tracking system using particle filters. In particular, we will point out where computational gains arise from having a simple projection model. As a case study, we consider the RoboCup MSL scenario and use a spherical object-model for tracking balls, but the system copes with any rotationally symmetric object. In previous work [13] we have also considered cylindrical objects representing RoboCup MSL robots.

Let $\mathbf{x}_t = [x, y, z, \dot{x}, \dot{y}, \dot{z}]^T$, be the state-vector, with (x,y,z), ($\dot{x}$,$\dot{y}$,$\dot{z}$) the object 3D cartesian position and linear velocities in a robot centered coordinate system. The goal is to estimate $\{\mathbf{x}_t; t \in \mathbb{N}\}$, which represents the object state at each time, 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 marginal distribution $p(\mathbf{y}_t \mid \mathbf{x}_t)$, where $n_\mathbf{y}$ is the dimension of the observation vector.

In a statistical setting, the problem is posed as the estimation of the *posteriori* distribution of the state given all

observations $p(\mathbf{x}_t \mid \mathbf{y}_{1:t})$. Under the Markov assumption:

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

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

We use Particle Filtering methods in which the probability distribution of an unknown state is represented by a set of M weighted particles $\{\mathbf{x}_t^{(i)}, w_t^{(i)}\}_{i=1}^{M}$ [6]:

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

where $\delta(\cdot)$ is the dirac delta function. Based on the discrete approximation of $p(\mathbf{x}_t \mid \mathbf{y}_{1:t})$, different estimates of the best state at time t are possible to be devised. For instance we 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, $\hat{\mathbf{x}}_{\mathrm{ML}} \doteq \mathrm{argmax}_{\mathbf{x}_t} \sum_{i=1}^{M} w_t^{(i)} \delta(\mathbf{x}_t - \mathbf{x}_t^{(i)})$.

The computation of the *posteriori distribution* is decomposed in 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 motion model;
2) *Observation* - computes the likelihood of each particle, based on image data.
3) *Update* - each particle's weight i is updated using its likelihood $p(\mathbf{y}_t \mid \mathbf{x}_t^{(i)})$, using $w_t^{(i)} \propto w_{t-1}^{(i)} p(\mathbf{y}_t \mid \mathbf{x}_t^{(i)})$
4) *Resampling* - the particles with a high weight are replicated and the ones with a low weight are forgotten.

For this purpose, we need to model probabilistically both the motion dynamics, $p(\mathbf{x}_t \mid \mathbf{x}_{t-1})$, and the computation of each particle's likelihood $p(\mathbf{y}_t \mid \mathbf{x}_t^{(i)})$ (steps 1 and 2). In particular, step 2 involves the sensor model and will benefit from using an adequate projection model. These two steps will be further detailed in the following sections.

A. The Motion Dynamics

Our method assumes that object motion follows a standard autoregressive dynamic model, $\mathbf{x}_t = A\mathbf{x}_{t-1} + \mathbf{w}_t$, where $\mathbf{w}_t \sim \mathcal{N}(0, Q)$. We have chosen a constant velocity model, in which the motion equations correspond to a uniform acceleration during one sample time, i.e. $\mathbf{x}_t = A\mathbf{x}_{t-1} + Ba_{t-1}$, with:

$$A = \begin{bmatrix} \mathbf{I} & (\Delta t)\mathbf{I} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}, \; B = \begin{bmatrix} (\frac{\Delta t^2}{2})\mathbf{I} \\ (\Delta t)\mathbf{I} \end{bmatrix} \tag{6}$$

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 was experimentally tuned to $cov(a_t) = \sigma^2\mathbf{I}$, with $\sigma = 120$mm/frame^2.

Since the tracker uses real-world coordinates, the motion model for an 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.

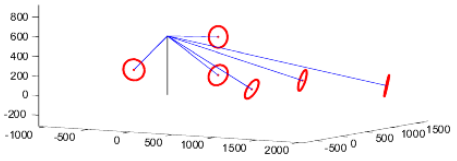


Fig. 2. Plot of the 3D points projected to obtain the 2D contour points for balls at different positions.

IV. Observation model

To calculate the likelihood of a particle we project its corresponding contour on the image (as a function of the object 3D shape, position and orientation) using an approximated model for the catadioptric system. The idea is to determine which points of the 3D model would be projected on the object's contour on the image (see Fig.2) and then create sets of 2D points in the inside and outside boundaries.

For a ball, the 3D contour points lie on the intersection between its spherical surface and the plane orthogonal to the line connecting the virtual projection center to the center of the sphere. With this model, it is possible to adjust the number of points describing the 2D contour, obtaining faster processing times (less points) or more robustness (more points). For our sensor, we will compare the performance of the tracking system using both the UPM and the PPM projection models.

The computation of the particles' likelihood is based on three color histograms: (i) the object color model; (ii) the inner, and (iii) outer boundaries of the projected contour. The first is computed in a training phase with several object examples taken from different locations and illumination conditions, and the others are computed at each frame for each particle. The idea is to assign a high likelihood to the contours for which the inner pixels have a color similar to the object, and are sufficiently distinct from the outer ones.

Formally, let $\mathbf{x}_t$ represent a target state hypothesis. A set of N points in the 3D object boundary is collected: $\{\mathbf{D}^n_{\mathrm{boundary}}\}$, $n = 1, \ldots, N$. These points are projected on the image, using the selected projection model, resulting in the 2D point set $\{\mathbf{d}^n_{\mathrm{boundary}}\}$, $n = 1, \ldots, N$. Each point $\mathbf{d}$ in the image is represented by its color vector in the HSI representation $\mathbf{y}_t(\mathbf{d})$. We will use this information to compute HSI histograms, with $B = B_h\, B_s\, B_i$ bins.

A. Deformable parametric contours - B-splines

Throughout this paper we use B-splines to represent the object contour. Indeed, this representation is widely used in computer graphics and in computer vision [1].

A B-spline has the following representation

$$f(t) = \sum_{i=0}^{k-m-1} c_i \mathcal{B}_i^m(t), \quad t \in [t_m, t_{k-m}] \tag{7}$$

where $\{\mathcal{B}_k^m(t), k = 0, \ldots, k-m-1\}$ is the set of basis functions such that $\mathcal{B}_i^m(t) \geq 0$, and $\sum_i \mathcal{B}_i^m(t) = 1$; $\{c_0, c_1, \ldots, c_{k-m-1}\}$ is the set of coefficients; and $[t_{i-1}, t_i]$

is an interval in which the spline functions are polynomial and exhibit a certain degree of continuity at the knots.

Planar curves are simply the $\mathbb{R}^2$ version of Eq. (7), i.e. $\mathbf{v}(t) \equiv [x(t) \quad y(t)] = \sum_{i=0}^{k-m-1} c_i \mathcal{B}_i^m(t)$. A discretized spline is a set of N equispaced samples of $\mathbf{v}(t)$ collected as the $N \times 2$ vector, $\mathbf{v} = [\mathbf{v}_0^T, \ldots, \mathbf{v}_{N-1}^T] = [\mathbf{x} \;\; \mathbf{y}], \quad N > k$. If we arrange the coordinates of control points into a parameter $\boldsymbol{\theta}_{(k)} = [c_0^T, \ldots, c_{k-1}^T]^T = [\boldsymbol{\theta}_{(k)}^x \quad \boldsymbol{\theta}_{(k)}^y]$[1], the discretized closed spline $\mathbf{v}$ can be obtained by the matrix product $\mathbf{v} = \mathbf{B}_{(k)}\boldsymbol{\theta}_{(k)} \Leftrightarrow \{\mathbf{x} = \mathbf{B}_{(k)}\boldsymbol{\theta}_{(k)}^x; \;\; \mathbf{y} = \mathbf{B}_{(k)}\boldsymbol{\theta}_{(k)}^y\}$, where the elements of $\mathbf{B}_{(k)}$ are $[\mathbf{B}_{(k)}]_{ij} = \mathcal{B}_j(t_0 + \frac{(t_k - t_0)i}{N})$.

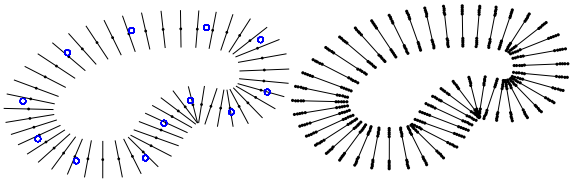
In computer graphics $m = 3$ or 4 is generally found to be sufficient. Herein, we use quadratic B-splines, i.e., $m = 3$.

B. B-spline Fitting: 2-D with known number of control points

After obtaining the 2D points $\{\mathbf{d}_{\text{boundary}}^n\}, n = 1, \ldots, N$, we convert them into a B-spline which best fits to this set. Assuming that we known the knots, the $N \times k$ matrix $\mathbf{B}_{(k)}$ can be computed. Thus, given the vector $\mathbf{d}$ with N data points and a choice of k, a $N \times k$ matrix $\mathbf{B}_{(k)}$ can be built and its pseudo-inverse $\mathbf{B}_{(k)}^{\dagger}$ computed. The estimated control points are given by $\widehat{\boldsymbol{\theta}_{(k)}} = \mathbf{B}_{(k)}^{\dagger}\mathbf{d}$, with $\mathbf{B}_{(k)}^{\dagger} = (\mathbf{B}_{(k)}^T\mathbf{B}_{(k)})^{-1}\mathbf{B}_{(k)}^T$ and the curve is given by $\mathbf{v} = [\mathbf{B}_{(k)}^{\dagger}\mathbf{x} \;\; \mathbf{B}_{(k)}^{\dagger}\mathbf{y}] = \mathbf{B}_{(k)}\widehat{\boldsymbol{\theta}_{(k)}}$. The regions in which the histogram is computed are defined at the points of the normal lines radiating from the discrete B-spline curve $\mathbf{v}$, i.e.

$$\mathbf{H} = \bigcup_{i=1}^{N} (\pm\Delta)\mathbf{v}(s_i)\,\bar{\mathbf{n}}(s_i) \tag{8}$$

where the sign $+$, and $-$ distinguishes whether the inspection is performed inside ($\mathbf{H}^{\text{inner}}$) or outside ($\mathbf{H}^{\text{outer}}$) the reference contour respectively; $\bar{\mathbf{n}}(s_i)$ is the normal vector at the point s_i; $\Delta \in [0, 1]$, Fig. 3 depicts the technique proposed herein. A quadratic B-spline of a generic shape and the orthogonal lines are shown in Fig. 3 (a). Fig. 3 (b) displays the lines at which the histogram values are collected. In some cases, e.g. when using cameras displaying color mosaicking errors at image edges, it may be convenient to avoid sampling at the middle of the line segment. In our case, however, color information at the contour pixels is quite acceptable and one can perform the inspection along the entire line segment.



(a) (b)

Fig. 3. Generic B-spline shape with the orthogonal lines. Control points are represented by circles (a). Bold lines show the image locations where the histogram values are collected (b).

[1] (k) is the total of number of control points

In the experiments of Section V we use 12 control points, 50 orthogonal lines along the contour, each one having 6 points, both inside and outside the object boundary.

C. Color Histograms

Let $b_t(\mathbf{d}) \in \{1, \ldots, B\}$ be the bin index associated with the pixel color at location $\mathbf{d}$ and frame t. The color histogram of a generic set of points can be computed by a kernel density estimate $\mathbf{H} \doteq \{h(b)\}_{b=1,\ldots,B}$ of the color distribution at frame t, and is given by [3]: $h(b) = \beta \sum_n \delta[b_t(\mathbf{d}^n) - b]$, where δ is the Kronecker delta function and β is a normalization constant so that h is a probability distribution.

To compute the similarity between two histograms we apply the Bhattacharyya similarity metric, as in [7]:

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

We adopt a distance metric inspired in [8]. Two quantities are taken into account: (i) distance between the object color model and the color measures inside the contour and; (ii) similarity between regions inside and outside the contour.

Defining $\mathbf{H}^{\text{model}}$, $\mathbf{H}^{\text{inner}}$ and $\mathbf{H}^{\text{outer}}$ as a reference (object) color model, the inner boundary points and the outer boundary points histogram, respectively, we will measure their pairwise similarities using (9). The distance metric should be high when candidate color histograms are different from the reference histogram and similar to the background. This can be expressed by the following quantity:

$$\mathcal{D} = \frac{\left(1 - \mathcal{S}(\mathbf{H}^{\text{model}}, \mathbf{H}^{\text{inner}})\right) + \kappa\mathcal{S}(\mathbf{H}^{\text{outer}}, \mathbf{H}^{\text{inner}})}{\kappa + 1} \tag{10}$$

This allows us to take into account the object-to-model mismatch (first term) and the object-to-background similarity (second term). Parameter κ allows to balance the two terms and was set to $\kappa = 1.5$, from experimental tests. The data likelihood function $\mathcal{L}$ is modeled as a Laplacian distribution over the distance metric: $p(\mathbf{y}_t \mid \mathbf{x}_t^{(i)}) \propto e^{-\frac{|\mathcal{D}|}{b}}$. In our experiments we set $b = 1/30$.

V. Experimental Results

This section presents an evaluation of the proposed methods. Firstly, we present a simulation where arbitrarily generated 3D points are projected to the image plane using three projection models: (i) the exact model – CPM; (ii) the UPM approximation; and (iii) the PPM approximation. All models use realistic calibration parameters. We show that both cases have equivalent re-projection errors.

These results are confirmed in non-simulated experiments, with a catadioptric system calibrated using the pattern shown in Fig.1c, with 3D points placed at several heights around the robot. Color models are obtained from manually selected regions in the image sequence containing the object of interest. All object pixels were used to train the color histograms, whereas in run-time, only a subset of points is used. In the first experiment we place a still ball at different positions around the robot and measure the error with respect to the

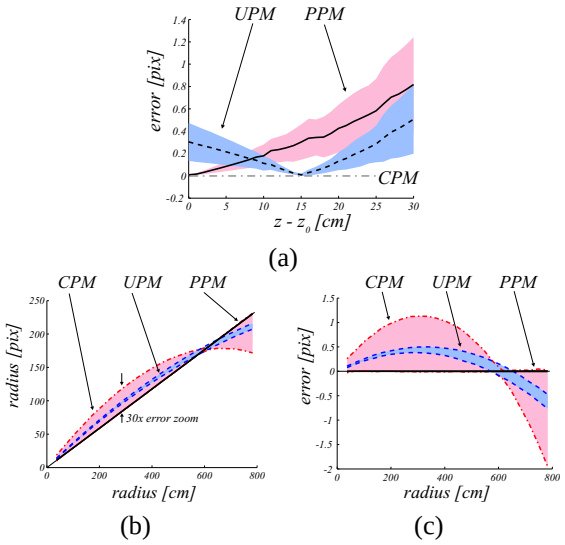


Fig. 4. Comparing projection models. (a) UPM (blue) and PPM (red) approximations to the CPM constant-resolution camera vs the height above the ground plane, $z = z_0$. The colored areas are standard deviations, lines show the mean re-projection error. (b) constant resolution analysis for CPM (red, dash-dot-), UPM (blue, dash-) and PPM (black, solid-line) vs distances measured on the ground plane. Lines show max and min error bounds. (c) zoom of (b) showing only the error differences to the PPM.

ground truth, using the proposed 3D tracking technique. In a final experiment, we track a bouncing ball, illustrating qualitatively the overall 3D tracking performance.

A. Camera-model approximations

Using the back-projection detailed in Sec.II, given the nominal parameters, ϑ and the mirror shape, (t, F) specified as a look-up table, one obtains a set of 3D rays from a set of image points. Sampling the 3D rays at several heights of interest from the ground plane, in our case $z - z_0 \in [0, 30]cm$ for a camera $60cm$ above the ground, results in a set of 3D points and their projections, that represents the exact-model – CPM. From the set of 3D points and their projections, the UPM and PPM can be fitted to the data using least-squares of the re-projection errors. These UPM and PPM approximations can now be compared with the CPM.

Figure 4a shows that the approximation errors are sub-pixel for almost all the considered z range. Notice that the PPM approximates exactly the CPM at the ground plane, as expected since it is a design specification, while the UPM approximates the CPM at a middle height so that the overall mean squared re-projection error is minimized. Qualitatively, the UPM encompasses some radial distortion allowing to approximate better the CPM.

On the other hand, the approximation given by the PPM allows studying how close to perspective the CPM and UPM are. Figure 4b shows that the perspective camera (solid-line) follows exactly a linear relationship $\rho = k_0 Z_0 \cdot r/Z$, while CPM and UPM yield that property only approximately (dashed and dash-dotted bounded areas).

Concluding, despite the fact that CPM has by design the constant-resolution property just for the ground plane, simulations show that the property holds for a reasonable

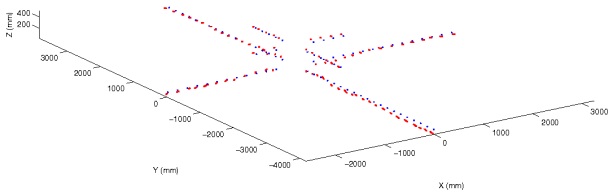


Fig. 5. A 3D view of ground truth (blue) and measurements obtained with the PPM (red) for error evaluation.

part of the 3D FOV and the CPM is well approximated in that volume by both the UPM and the PPM. Simulations also show that the PPM is the most accurate model for the ground plane, as expected due to the design, while the UPM allows more accurate approximations for planes above the ground, involving however more computations.

We would like to stress that the constant-resolution design is a good compromise between approximating ubiquitous constant-resolution and enlarging the field of view, w.r.t perspective cameras. Note that perspective cameras only have constant-resolution on all planes orthogonal to the optical axis, for narrow view fields. Large fields-of-view imply small focal length lenses that introduce radial distortions.

B. Error evaluation

The previous results show the validity of UPM and PPM model approximations using simulated 2D and 3D target positions. Here we use the real catadioptric camera and image measurements to estimate target's 3D positions. The camera is calibrated either with the PPM or the UPM models. Both calibrations share the same 3D- and imaged-points calibration-data. Ín [13] we show that UPM produces small re-projection errors. In this section we assess the quality of PPM calibration combined with the ball localization method.

We have placed a ball at various positions around a robot and confronted the positions measured with our system against the ground truth. The positions were either on the floor or at a height of 340mm, as illustrated in Fig. 5.

Measurements were made with the proposed 3D tracking system, using both the UPM and PPM projection models. Particles were initialized with a Gaussian distribution with large variance around the ground truth positions and zero velocity. We have computed the errors between ground truth and measurements, in spherical coordinates γ, ϕ and ψ, respectively distance, elevation and azimuth. Results for the errors' mean and standard deviation are presented in Table I. As can be observed from the table, there are no significant differences between the two projection models. However, the computation time for PPM is about half the time taken by the UPM (last line of Table I). Therefore, the usage of the PPM is advantageous for the overall 3D tracking system, since it reduces computation time for the same level of precision.

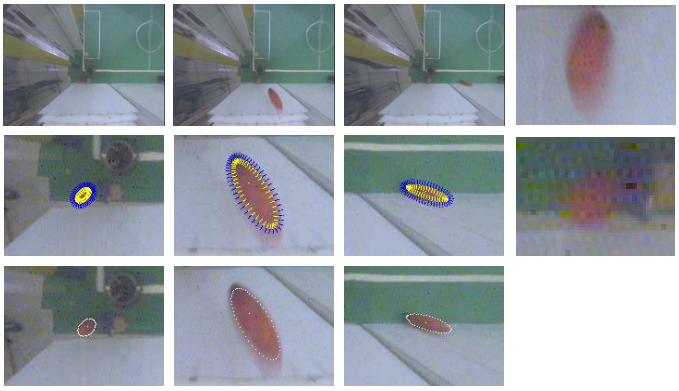
C. Ball tracking

In this experiment we track a bouncing ball. The ball's projection on the image plane changes dramatically along time (see Fig.6a), due to the nature of the catadioptric system

	UPM	PPM
mean γ error (mm)	-19.8712	-18.2616
std.dev γ error (mm)	48.1137	47.9484
mean ϕ error (rad)	0.0005	0.0006
std.dev ϕ error (rad)	0.0310	0.0276
mean ψ error (rad)	0.0072	0.0072
std.dev ψ error (rad)	0.0314	0.0312
Computation Time (ms)	11.2	6.3

TABLE I

COMPARISON OF ERRORS MEASURED WITH THE UPM *vs* PPM. LAST LINE SHOWS THE COMPUTATION TIMES FOR PROJECTING 50000 POINTS, ON A P4 2.6GHZ COMPUTER.



(a) (b)

Fig. 6. Ball jumping. (a) Three frames of the sequence (top row), the corresponding close-ups of the tracked ball with the pixels used to build the inner and outer color histogram marked in yellow and blue, respectively (middle row), the same close-ups with the contour projected by a ball placed in the estimated position drawn in white (bottom row). (b) Close-ups of the ball showing motion blur and noise.

used. The images are affected by both motion blur and heavy sensor noise (see Fig.6b). Images were acquired at 25fps and we used 10000 particles in the tracker. Particles were initialized by a Gaussian distribution with large variance, centered at rough estimates of ball's initial position and velocity. To compute the inner and outer color histograms for each hypothesis, we have used $N = 50$ points on the sphere's contour, as described in Section IV. We repeated the tracking 10 times on the same image sequence to illustrate the variance of the estimated trajectories on different runs. In previous work [13] we presented results on this sequence using the UPM model. In this paper we run the same experiment using the PPM model. A comparison between the two models is shown in Fig. 7. No noticeable differences are observed between the two models, but computational savings justify the choice of the PPM model.

VI. CONCLUSIONS

We have presented a model-based 3D tracking system with particle filters, using a wide angle perspective catadioptric sensor. Although in most of the 3D space these sensors only provide an approximation to the perspective model and are often approximated by the more general UPM model, we have shown experimentally that the perspective model

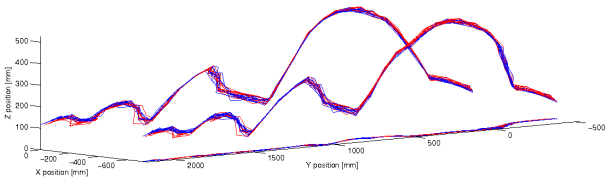


Fig. 7. Ball jumping: plot of the tracked paths resulting from 10 runs of the algorithm performed on the same image sequence. The estimated trajectories using the PPM and the UPM models are shown, respectively, with blue and red lines. Both the 3D trajectory and its projections on the ground and lateral planes are shown.

is advantageous in terms of simplicity and computational efficiency, yielding a comparable accuracy.

We have described in detail the proposed 3D tracker, including both the motion and observation models. We have performed extensive experiments with real robots in a RoboCup MSL scenario. We showed the performance of our method in tracking jumping balls, demonstrating its ability to deal with off-the-floor targets and sudden trajectory changes. Additionally, we evaluated the precision of the system in static scenarios with ground truth measurements.

The results show that the combination of wide angle sensors with 3D model-based tracking methods is able to cope with complex target motions in challenging observation conditions. In future work we will further characterize the proposed observation model, evaluating its robustness to occlusions and ambiguities. Also, we will extend the particle filter method to address the following issues: (i) include object pose in the state vector, such as to deal with non rotationally symmetric objects; (ii) use multiple motion models to characterize common objects' maneuvers; and (iii) take into account robot self-motion and express particle's state in a fixed reference frame.

REFERENCES

[1] A. Blake and M. Isard, *Active Contours*, Springer, 1998.

[2] J. Gaspar, N. Winters and J. Santos-Victor, Vision-based Navigation and Environmental Representations with an Omnidirectional Camera, *IEEE Transactions on Robotics and Automation*, Vol. 16, 6, Dec. 2000.

[3] D. Comaniciu, V. Ramesh and P. Meer, Real-Time Tracking of Non-Rigid Objects using Mean Shift. *IEEE CVPR*, pp. 142-151, 2000.

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

[5] S. Thrun, W. Burgard, D. Fox, *Probabilistic Robotics*, MIT press, 2005.

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

[7] P. Perez, C. Hue, J. Vermaak and M. Gangnet, Color-Based Probabilistic Tracking using Unscented Particle Filter, *IEEE CVPR*, 2002.

[8] S. Olufs, F. Adolf, R. Hartanto and P. Plöger, Towards probabilistic shape vision in robocup: A practical approach, *In: RoboCup Int. Symposium*, Bremen, Germany 2006.

[9] C. Geyer and K. Daniilidis, A unifying theory for central panoramic systems and practical applications, *IEEE CVPR*, pp. 445-461, 2000.

[10] R. Benosman, S. Kang, Panoramic Vision, *Springer Verlag*, 2001.

[11] R. Hicks and R. Bajcsy, Catadioptric sensors that approximate wide-angle perspective projections, *IEEE CVPR*, pp. 545-551, 2000.

[12] J. Gaspar, C. Deccó, Jun Okamoto Jr, J. Santos-Victor, Constant resolution omnidirectional cameras, *3rd Int. IEEE Workshop on Omni-directional Vision at ECCV*. pp. 27-34, 2002.

[13] M: Tajana, J. Gaspar, J. Nascimento, A. Bernardino, P. Lima, 3D tracking by Catadioptric Vision Based on Particle Filters, Proc. of the Robocup Symposium, Atlanta, July 9-10, 2007.