Tracking of Ball Trajectories with a Free Moving Camera-Inertial Sensor

Oliver Birbach[1], Jörg Kurlbaum[2], Tim Laue[1], and Udo Frese[1]

[1] Deutsches Forschungszentrum für Künstliche Intelligenz GmbH,
Sichere Kognitive Systeme, Enrique-Schmidt-Str. 5, 28359 Bremen, Germany
{oliver.birbach,tim.laue,udo.frese}@dfki.de
[2] Fachbereich 3 - Mathematik und Informatik, Universität Bremen,
Postfach 330 440, 28334 Bremen, Germany
jkur@informatik.uni-bremen.de

Abstract. This paper is motivated by the goal of a visual perception system for the RoboCup 2050 challenge to win against the human world-cup champion. Its contribution is to answer two questions on the sub-problem of predicting the motion of a flying ball. First, if we could detect the ball in images, is that enough information to predict its motion precise enough? And second, how much do we lose by using the real-time capable Unscented Kalman Filter (UKF) instead of non-linear maximum likelihood as a *gold standard*? We present experiments with a camera and an inertial sensor on a helmet worn by a human soccer player. These confirm that the precision is roughly enough and using an UKF is feasible.

1 Introduction

1.1 Motivation: A Vision for 2050

In RoboCup's humanoid leagues, the limiting factor is often the robot hardware, in particular actuation. Current robots can walk and kick the ball, but are far away from running, jumping, or tackling. Comparing the official RoboCup vision to win against the human world champion in 2050 to the last 10 years of hardware development, 40 years to go sound realistic but not overly conservative.

As computer vision researchers, we look at the other end of the causal chain. How long is the road to a visual perception system that meets the RoboCup 2050 challenge? – Not too long, maybe a decade, was our conclusion in a recent analysis [1]. This idea is encouraging. But would it help, if we would build a vision system and then wait until 2050 to try it out? We propose an experiment without a robot instead. We mount a camera and an inertial sensor on a helmet worn by a human soccer player (Fig. 1) and ask, whether this setup and computer vision could provide enough information for a humanoid to take the human's place.

The paper is our first step towards this goal. It contributes a study on the subproblem of predicting a flying ball. The study focuses on the tracking part and hence uses manually processed images from real human soccer. We deliberately chose not to implement a full system because real-time vision of soccer scenes

L. Iocchi et al. (Eds.): RoboCup 2008, LNAI 5399, pp. 49–60, 2009.

Fig. 1. The proposed sensor setup with a wide angle camera and an inertial sensor

from a helmet camera is way beyond the scope of a single paper and instead of presenting incremental progress towards this, we rather want to definitely answer two basic questions on the tracking part: Is the information provided by a camera and an inertial sensor enough to predict a flying ball with the necessery precision for playing soccer? And, how much do we lose by using a real-time capable filter compared to the slow *gold standard* solution of non-linear maximum likelihood estimation? – In short, the answers will be positive: "Yes" and "not too much".

1.2 From Walking to Running

For localization and tracking, it is essential to know the camera pose relative to the ground. Today, it is usually obtained by forward kinematics since the robots have well defined contact with the ground. Once robots start to run and jump this will not be possible anymore. Instead, an additional sensor is needed that perceives the camera's motion, i.e. an inertial sensor measuring acceleration and angular velocity. The insight that motion without ground contact will come up in RoboCup sooner or later additionally motivates our experimental setup, where the camera motion has to be treated as free motion anyway, because "on a human", of course, no kinematic information is available.

1.3 Methods

To show how to track a ball and its observer state with this combined sensor setup, we will use well known methods and provide a quantitative evaluation.

Tracking algorithms exist in several variations. The family of Kalman filters is well understood, reliable and often used. You will mostly find its variations, the Extended Kalman Filter (EKF) [2] and the Unscented Kalman filter (UKF) [3], in RoboCup applications. Also, filters based on Monte-Carlo methods, such as the particle filter, or combinations of both [4] are popular.

In our case, the filter estimates the orientation of the free-moving camera. The orientation has three degrees of freedom, but suffers from singularity problems when parameterized with three variables, such as Euler angles. We therefore

use quaternions as singularity-free representations and a special technique, the so-called embedding operator $\boxplus$, for handling them in an UKF.

To evaluate the UKF, we compared it to maximum likelihood estimation using Levenberg-Marquardt as a *gold standard*. Furthermore, we evaluated the accuracy by comparing the calculated bouncing point from the tracker to the ground truth bouncing point obtained manually from the images.

1.4 Related Work

In RoboCup, tracking of balls was studied by Voigtländer et al. [5]. Their system recognizes balls in the Middle-Size League and tracks their state over time with a position error of $\leq 0.5m$. Their sensor setup consists of a perspective as well as a catadioptric camera. Within the Small-Size League Rojas et al. [6] developed a system to detect and track lifted kicks with a statically mounted overhead camera. After a couple of frames, this system allowed a reliable prediction of the ball's bouncing point. Another interesting work was done by Frese et al. [7]. There, the catch point for a robotic softball-catcher was determined by predicting the ball trajectory which was tracked using stereo vision and an EKF. Similarly, in [8] a system was developed that tracks a flying dart and repositions the dartboard to always hit the bull's eye.

Recently, even a commercial system has been introduced [9]. The so-called RoboKeeper employs cameras to track the ball. It intercepts the ball with a plastic plate rotated around the lower end by a single motor. The system is intended for public entertainment, so the plate shows the image of a goal keeper.

All these approaches either have a static camera or a camera fixed relative to the ground plane. In contrast, we are dealing with a freely moving observer. This requires to track the observer itself, making the problem much harder.

1.5 Outline

This paper is structured as follows. We introduce the proposed camera-inertial system and the measurements obtained from it. Then, the calibration of the alignment of the inertial-sensor and the camera will be discussed. The description of the system variables and the underlying model lead then to an overview of the UKF implementation. Finally, we present experimental results and a conclusion.

2 Experimental Setup

As mentioned above, we propose to use a combination of a monocular camera and an inertial sensor as the sensor setup for answering the question if the motion of flying balls can be predicted sufficiently.

2.1 Camera-Inertial Sensor System

In our setup, the monocular camera is a Basler A312fc (54 Hz) with a Pentax H(416)KP lens (86° wide-angle). Attached to this camera is a XSens MTx inertial

Table 1. Measurements from the sensor setup and their corresponding units

Accelerometer	**Gyroscope**	**Monocular Camera**				
Acceleration in $\frac{m}{s^2}$	Ang. vel. in $\frac{rad}{s}$	Ball (pixel)			Landmark (pixel)	
$\boldsymbol{a}$	$\boldsymbol{\omega}$	x_b	y_b	r_b	$x_{l,n}$	$y_{l,n}$

sensor synchronized to the camera shutter. The camera provides information about the environment and the inertial sensor about the setup's motion (Tab. 1).

What information can these sensors provide? In principle, the pose of the camera can be obtained by integrating gyro and accelerometer measurements, however with accumulating error. To compensate this error, vision measurements of landmarks, e.g. lines on the field, are used. The camera provides the direction towards the ball from its image position. The image radius provides the depth, however with a large error since the ball is small. Interestingly, implicit depth information also comes from the effect of gravity over time. After a time of t the ball has fallen $\frac{1}{2}gt^2$ relative to motion without gravity. This distance is implicitly observed and, after some time, provides preciser depth than the ball radius.

One might also consider a stereo vision setup instead of a monocular one to obtain the distance by triangulating at the stereo baseline. However, we use the image radius of the ball implicitly triangulating at the ball diameter which is rather even a bit larger. So we expect no great benefit from stereo.

2.2 Sensor Calibration

Both sensors operate in different coordinate systems. For that, a transformation, mapping measurements between both, is needed. Our setup is not rotating too fast. Therefore, we can neglect the translation and the problem reduces to finding the rotational displacement. It is parameterized as a quaternion q, leading to the following mapping of a vector v_{cam} from camera to v_{iner} in inertial coordinates:

$$(0, v_{\text{iner}}) = q_{\text{iner}}^{\text{cam}} \cdot (0, v_{\text{cam}}) \cdot q_{\text{iner}}^{\text{cam}\ -1} \tag{1}$$

The rotational displacement $q_{\text{iner}}^{\text{cam}}$ is found by calibration. The idea here is to use a horizontal checkerboard calibration plate so both camera and inertial sensor know the direction "down" in their respective coordinate systems [10]. The camera parameters are computed by nonlinear least squares estimation following Zhang's method [11] which is extended to estimate the rotational displacement q with (1). The camera equations are described in Sec. 3.2. For our setup, the calibration had ≈ 0.2 pixels (camera) and $\approx 0.26°$ (inertial sensor) residual.

3 System Model

The flight of a ball observed by a camera-inertial sensor can be formally represented by a dynamical system. Within this system, the state of the ball and sensor changes as the ball flies and time progresses. The composition of the system's state and its variables is given in Tab. 2.

Table 2. Tabulated summary of the systems's state $\in S$

Ball		**Sensor**		
Position in m	Velocity in $\frac{m}{s}$	Position in m	Velocity in $\frac{m}{s}$	Orientation as quaternion
$\boldsymbol{x}_b$	$\boldsymbol{v}_b$	$\boldsymbol{x}_s$	$\boldsymbol{v}_s$	$q^{\text{iner}}_{\text{world}}$

3.1 Dynamic Equations

The evolution of the state, i.e. the motion, is given by ordinary differential equations. For the ball, these obey classical mechanics with gravitation and air drag.

$$\dot{\boldsymbol{x}}_b = \boldsymbol{v_b} \tag{2}$$

$$\dot{\boldsymbol{v}}_b = \boldsymbol{g} - \alpha \cdot |\boldsymbol{v_b}| \cdot \boldsymbol{v_b}, \quad \alpha = \frac{c_d A \rho}{2m} \tag{3}$$

where $\boldsymbol{g}$ represents gravity. The effect of air drag is determined by the factor α, where c_d is the drag coefficient, ρ is the density of air, A is the cross-sectional area and m the mass of the ball. Our football had a cross-sectional area of $A = 0.039\ m^2$ and mass of $m = 0.43\ kg$, resulting in $\alpha = 0.011\ m^{-1}$.

Similarly, the motion of the sensor follows differential equations that incorporate the measurements from the inertial sensor (Tab. 1 and 2).

$$\dot{q}^{\text{iner}}_{\text{world}} = q^{\text{iner}}_{\text{world}} \cdot \frac{1}{2}(0,\ \boldsymbol{\omega}) \tag{4}$$

$$\dot{\boldsymbol{x}}_s = \boldsymbol{v_s} \tag{5}$$

$$\dot{\boldsymbol{v}}_s = \boldsymbol{g} + q^{\text{iner}}_{\text{world}} \cdot (0, \boldsymbol{a}) \cdot q^{\text{iner}\ -1}_{\text{world}} \tag{6}$$

Basically the inertial measurements are converted from body- to world coordinates and integrated once (q from $\boldsymbol{\omega}$) or twice ($\boldsymbol{x}$ from $\boldsymbol{a}$).

3.2 Measurement Equations

The measurement equations model the measurements from the monocular camera, i.e. image position and radius of the ball and of point landmarks on the soccer field. We model the projection of these features from the state using a pin-hole camera model plus radial distortion. The first function simply projects a point $\boldsymbol{x}_{world}$ from a 3D-scene into the image plane at $(u, v)^T$:

$$\begin{pmatrix} X_{cam} \\ Y_{cam} \\ Z_{cam} \end{pmatrix} = q^{\text{cam}\ -1}_{\text{world}} \begin{pmatrix} 0 \\ \boldsymbol{x}_{world} - \boldsymbol{x}_s \end{pmatrix} q^{\text{cam}}_{\text{world}} \tag{7}$$

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} X_{cam} \\ Y_{cam} \end{pmatrix} / Z_{cam} \tag{8}$$

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} f_x \\ f_y \end{pmatrix} \cdot \begin{pmatrix} x' \\ y' \end{pmatrix} + \begin{pmatrix} u_0 \\ v_0 \end{pmatrix} \tag{9}$$

The first equation transforms the world point into camera space according to the quaternion $q^{\text{cam}}_{\text{world}} = q^{\text{imu}}_{\text{world}} \cdot q^{\text{cam}}_{\text{imu}}$ and the sensor's position $\boldsymbol{x}_s$. Equation (8)

is the actual perspective projection. The last equation transforms the result to pixels according to f_x, f_y (effective focal distance) and u_0,v_0 (image center).

The camera's radial distortion is considered by scaling $(x, y)^T$ according to

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} \left(1 + \frac{a_2 r + a_3 r^2}{1 + b_1 r + b_2 r^2 + b_3 r^3}\right), \quad r^2 = x^2 + y^2. \tag{10}$$

To model the ball measurements, we project the ball from a 3D-scene as a circle into the image plane. For this, we use the introduced projection to calculate the position of four outer points of the ball state, which are computed from the always known ball diameter, on the image plane. After that, we recombine the projected points to center and radius by computing mean and std. deviation.

4 Unscented Kalman Filter (UKF) for Ball Tracking

For tracking the ball and sensor state over time, we use an UKF incorporating the preceding equations into the dynamic update and measurement update step.

4.1 Dynamic and Measurement Equations

The UKF's dynamic update step uses a numerical solution of (2), (3), (5), (6). For (4) an analytic solution is used. The camera measurements are integrated in the measurement update step using the perspective projection (7)-(10).

Kalman filters assume noisy processes. Therefore, the dynamic noise R and measurement noise Q need to be defined. We set (2) and (5) to zero noise assuming that every error in position comes from an error in velocity. The dynamic noise of the inertial sensor, affecting (4) and (6), was calibrated from a resting sensor. The dynamic noise of the ball (3) was tuned by looking at the recorded trajectory. This parameter is important since it models side wind and bended trajectories caused by ball spin. Based on these values, the measurement noise for the ball and landmark measurements were calibrated by maximizing their likelihood in an recorded series of a ball flight [12].

4.2 Quaternions in the UKF State

To track the state of the system described above, we use a customized UKF. While tracking $\boldsymbol{x}_s, \boldsymbol{v}_s, \boldsymbol{x}_b, \boldsymbol{v}_b$ is straight-forward, the orientation $q^{\text{iner}}_{\text{world}}$ poses a special problem. In Sec. 3, we modeled the orientation in the state space S using a unit quaternion to avoid singularities. Unfortunately, such a state cannot be treated as a vector $\mathbb{R}^4$ because of the constraint $|q^{\text{iner}}_{\text{world}}| = 1$. In mathematics such a structure is called a 3-manifold in $\mathbb{R}^4$. The filter doesn't know this structure and assumes to operate on a flat vector. Hence, after a measurement update, usually $|q^{\text{iner}}_{\text{world}}| \neq 1$ and the result is not an orientation anymore.

Our idea to solve this problem is to use the mean-state from the UKF as a reference and parameterize small deviations from that reference as a flat vector

with a dimension corresponding to the dimension of the manifold (i.e. 3 for $q^{\text{iner}}_{\text{world}}$). As the deviations are small, this is possible without singularities.

The operation of applying such a small change to the state is encapsulated by an operator $\boxplus$ and the inverse operator $\boxminus$. They provide an interface for the UKF to access the state, which is then treated as a black-box datatype.

$$\boxplus : S \times \mathbb{R}^n \to S \tag{11}$$

$$\boxminus : S \times S \to \mathbb{R}^n \tag{12}$$

$$s_1 \boxplus (s_2 \boxminus s_1) = s_2 \tag{13}$$

$$s \boxplus (a + b) \approx (s \boxplus a) \boxplus b \tag{14}$$

To summarize the interaction between the UKF and the blackbox state datatype, dynamic and measurement equations access the state's internal structure as they depend on the concrete problem. However, the generic UKF equations for sigma point propagation and measurement update access the state only through $\boxplus$ and $\boxminus$ with a flat vector. Thereby, the state's internal structure, in particular the manifold structure underlying quaternions, is hidden from the UKF. Notably, this corresponds to a common implementation issue, that one would prefer the state to have named members sometimes and sometimes a flat vector.

As a result, in all the matrix computations of the UKF the quaternion part of the state $q^{\text{iner}}_{\text{world}}$ simply corresponds to 3 columns just as the vector part $\boldsymbol{x}_s, \boldsymbol{v}_s, \boldsymbol{x}_b, \boldsymbol{v}_b$. The only difference arises when the UKF finally applies the innovation to the state by $\boxplus$, where the vector part is a simple $+$ and the quaternion part is a more complex operation (multiplication with an angle-axis rotation).

This method is more elaborated in [13]. Its beauty lies in the fact that it is obtained from the classical UKF simply by replacing $+$ with $\boxplus$ and $-$ with $\boxminus$.

5 Experiments

We implemented the ball tracker as described above and held a series of experiments on a real soccer field. A human wearing the helmet with the camera-inertial sensor followed the trajectories of several ball flights which where initiated by another person. The data was recorded and the images were manually processed (confer the introduction why). The image center and radius of the ball and the image position of the field's landmark points were extracted. Landmarks were all intersections of two lines on the field, the lower end of the goal posts, the intersections of the goal post with the crossbar and the penalty spot.

5.1 Performance Compared to a Gold Standard

To evaluate the performance of the unscented Kalman filter, we implemented a nonlinear maximum likelihood estimator using the Levenberg-Marquardt (LM) algorithm [14]. It performs the same task and uses the same model as the Kalman filter. In contrast to the UKF, however, estimating the current state requires to estimate all past states as well. This makes maximum likelihood estimation

Fig. 2. Grid overlay of a homography between the image plane and the field plane. Observe the fine correspondence near the line in front of the goal.

computationally expensive and unsuitable for real-time tracking. The question we want to answer here is, whether the UKF is good enough or whether we will have to think about better real-time capable alternatives in the future.

5.2 Obtaining Ground Truth

To analyze the accuracy of the tracker, we wanted to compare the predicted bouncing points of the tracked ball states with the real bouncing point. This turned out to be not that easy as the ball leaves no permanent mark on field, making a manual measurement infeasible. However, at least temporarily raised dust is visible on the video. So for the evaluation here, the actual bouncing point was determined from the image position of the dust in the video using a homography between at least four landmarks on the field and their corresponding points in the image plane (correcting for distortion). If the dust was visible in several images, we took the mean of these results. While we have no quantitive value, the homography looks visually rather precise (Fig. 2).

5.3 Results

In order to show how well the ball flight observed by a moving camera-inertial sensor can be estimated and predicted, results from one fully filtered ball flight recorded during the experiments will be given. For this, a ball flying about 3 m (reaching an maximum height of 1.6 m) towards the observer from about 7 m away was chosen. The ball was flying almost spin-less and the impact of the side wind to the ball at this range of flight and speed was minimal. The performance of tracking and predicting the trajectory is obtained by comparing the predicted bouncing points of the tracked trajectory with the ground truth bouncing point.

Figure 3 shows the trajectory and velocity vectors tracked by the unscented Kalman filter and the LM algorithm from this flight. As can be seen, both methods calculate almost the same series of state estimates. But the tracked trajectories do not resemble the smoothness of the actual trajectory. Note that,

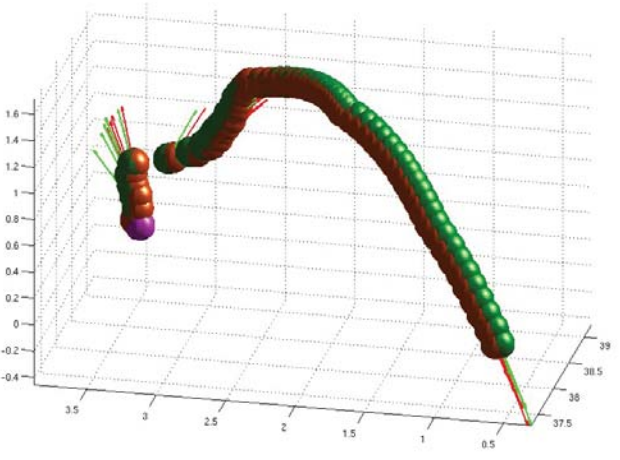

Fig. 3. Tracked trajectories and velocity vectors of the unscented Kalman filter and the Levenberg-Marquardt algorithm. As can be seen, both methods estimate almost the same set of states.

since we plotted the estimate over time, Fig. 3 does not show how the tracker believes the ball flew, but how the belief of the tracker on where the ball is changed. So the jerkiness in the beginning is due to new information arriving.

Another interesting behavior is the estimation of the velocity vectors pointing away from the actual flight direction. This behavior is not suprising since the monocular camera defines the ball distance by the ball diameter. Within the first few frames the diameter is 16 ± 1 pixels, leading to an expected error of 6.25% in the distance. So, initially estimating the distance is very difficult, the situation only gets better after the ball has been observed for some time.

5.4 Prediction Performance Compared to Ground Truth

To show how the prediction performs over time, Fig. 4 depicts the error between the prediction of the tracked ball state and the assumed bouncing points as a function of time. It can be clearly seen that the UKF and the LM method perform similar. The error of both methods decreases with time. Interestingly, the prediction performance improves in a wave-like shape, caused by the integer based image radius measurements. The peaks of the waves decrease, since the influence of the radius diminishes and the distance becomes more defined by gravity (Sec. 2.1).

A quantative analysis reveals that the estimation of ball states behind the initial ball position lead to very inaccurate ball predictions in the beginning. The error decreases significantly after 7 frames (or about 129 ms), reaching a surprisingly good estimate with a difference between 0.1 m and 0.4 m. Then, the accuracy decreases rapidly for a couple of frames before it reaches a next good estimate after 16 frames (297 ms) with an error between 0.1 and 0.15 m. From

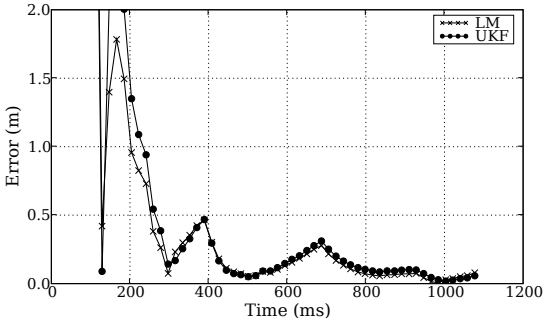

Fig. 4. Error between the predicted bouncing points computed from the estimated trajectory and the mean of the ground truth measurements over time

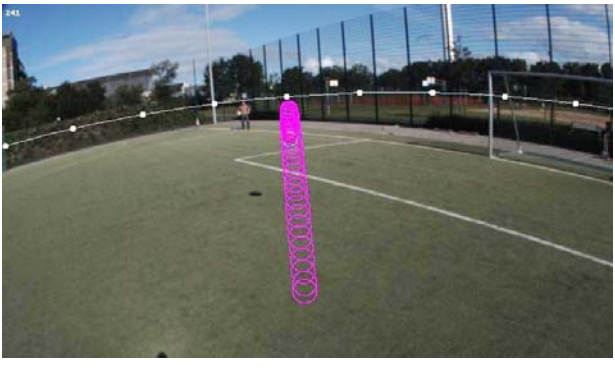

Fig. 5. Predicted trajectory of a flying ball observed by a moving camera-inertial sensor projected into image coordinates. See `http://www.informatik.uni-bremen.de/agebv/en/Vision2050`.

here, the estimates seem to be quite reliable and the prediction performance improves in the above mentioned wave-like shape.

Figure 5 shows a predicted trajectory of a ball from its tracked state projected into images coordinates. This image is part of a video visualizing the prediction performance within the recorded images from the experiments.

The results observed within this ball flight match the results from the two other ball flights that were evaluated. All three flights showed a suprisingly good predicted trajectory after $8 - 10$ frames. But all of these predictions only lasted for one or two frames, assuming that there is still a large amount of uncertainty at this point of the time during tracking. A couple of frames later, namely after $15 - 16$ frames ($278 - 296\ ms$), all examined flights show a quite accurate indication of the predicted bouncing point. This number of frames seem to be the point in time in which the distance is more defined by gravity than by the image radius.

For more information on our experiments and results on further scenes, see [12].

6 Conclusion

In this paper, we have shown how to track and predict the trajectories of flying balls using a freely moving monocular camera-inertial sensor. We have evaluated the precision by comparing the predicted bouncing points over time with the real bouncing point. The results indicate that after half the time of flight the error is about $0.3m$ which we judge to be enough for playing soccer. The real-time capable UKF and Levenberg-Marquardt, the *gold standard*, perform very similar. We can conclude that with such a setup, flying balls can be predicted roughly precise enough for playing soccer and the UKF is a feasible algorithm for that.

The next step is, of course, to replace the manual vision by computer vision. This is difficult because of the uncontrolled lighting and background on a real soccer field. Our idea [1] is to use, in turn, the tracking component as context for the ball detection. In real images, many things may look like a ball, e.g. a head, a white spot, or reflections, but few of them move according to the physics of a flying ball. So by treating the problem as a combined likelihood optimization with respect to both the physical trajectory model and visual appearance, we hope to obtain a reliable vision system.

Furthermore, we want to go away from predicting a single pass to tracking the ball during a full scene. Therefore, we need a multi-state tracker which not only tracks the state of a flying ball, but also a ball which is rolling or just lying.

References

1. Frese, U., Laue, T. (A) VISION FOR 2050 – The Road Towards Image Understanding for a Human-Robot Soccer Match. In: International Conference on Informatics in Control, Automation and Robotics (2008)
2. Welch, G., Bishop, G.: An Introduction to the Kalman Filter. Technical Report TR 95-041, Chapel Hill, NC, USA (1995)
3. Julier, S., Uhlmann, J.: A new extension of the Kalman filter to nonlinear systems. In: Int. Symp. Aerospace/Defense Sensing, Simul. and Controls, Orlando, FL (1997)
4. Kwok, C., Fox, D.: Map-based multiple model tracking of a moving object. In: Nardi, D., Riedmiller, M., Sammut, C., Santos-Victor, J. (eds.) RoboCup 2004. LNCS, vol. 3276, pp. 18–33. Springer, Heidelberg (2005)
5. Voigtländer, A., Lange, S., Lauer, M., Riedmiller, M.: Real-time 3D Ball Recognition using Perspective and Catadioptric Cameras. In: Proc. of the 3rd European Conference on Mobile Robots, ECMR (2007)
6. Rojas, R., Simon, M., Tenchio, O.: Parabolic Flight Reconstruction from Multiple Images from a Single Camera in General Position. In: Lakemeyer, G., Sklar, E., Sorrenti, D.G., Takahashi, T. (eds.) RoboCup 2006: Robot Soccer World Cup X. LNCS, vol. 4434, pp. 183–193. Springer, Heidelberg (2007)
7. Frese, U., Bäuml, B., Haidacher, S., Schreiber, G., Schaefer, I., Hähnle, M., Hirzinger, G.: Off-the-Shelf Vision for a Robotic Ball Catcher. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Maui, pp. 1623–1629 (2001)
8. Kormann, B.: Highspeed dartboard (2008), `http://www.treffsichere-dartscheibe.de/`

9. 4attention GmbH. RoboKeeper web site (2006), http://www.robokeeper.com/
10. Lobo, J., Dias, J.: Relative pose calibration between visual and inertial sensors. In: ICRA Workshop InerVis (2004)
11. Zhang, Z.: A Flexible New Technique for Camera Calibration. IEEE Transactions on Pattern Analysis and Machine Intelligence 22(11), 1330–1334 (2000)
12. Birbach, O.: Accuracy Analysis of Camera-Inertial Sensor-Based Ball Trajectory Prediction. Master's thesis, Universität Bremen (2008)
13. Kurlbaum, J.: Verfolgung von Ballflugbahnen mit einem frei beweglichen Kamera-Inertialsensor. Master's thesis, Universität Bremen (2007)
14. Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P.: Numerical Recipes in C: The Art of Scientific Computing, ch. 15.5. Cambridge University Press, New York (1992)