Accuracy Analysis of Camera-Inertial Sensor-Based Ball Trajectory Prediction

Oliver Birbach

Diploma thesis at the Faculty 3: Mathematics and Computer Science,
University of Bremen

February 13, 2008

Supervisor: Dr. ing. Udo Frese
Second reviewer: Prof. Dr. rer. hum. biol. Kerstin Schill

Abstract

The RoboCup initiative formulated its long-term goal as winning against the most recent champion of the World Cup in 2050 with a team of autonomous robotic soccer players. At this point of time, the question if the initiative will achieve this goal cannot be answered. Although the progress within the robotic soccer community is remarkable, it is not known whether it will lead to competitive robotic soccer players until then or not. Legitimately, one may also ask the question the other way around: Is there an area of research, in which enough knowledge has already been obtained for the 2050 scenario? This thesis tries to give an answer to the question with respect to the sensing subsystem within the domain of soccer playing robots. More precisely, this thesis will reveal if the accuracy of ball trajectory prediction is sufficient for a hypothetical robotic soccer player under certain constraints.

The sensor setup for this hypothetical robot is a camera-inertial sensor which allows perception of the environment and tracking of the sensor's motion relative to itself. To answer the question if the accuracy is sufficient or not, an experimental-based analysis is performed in the following way: A camera-inertial sensor obtains measurements from a real soccer field, including the ball's position and size and relevant points of the field. These measurements may be used to estimate the state of the ball during each stage of the flight by fitting the ball's and sensor's motion model to the measurements. The accuracy is determined by estimating all in-flight states of the ball and comparing the bouncing points of the predicted trajectories with the real bouncing point.

The results of the analysis indicate that trajectories of flying balls can be predicted reliably over time. Furthermore, under the assumption of certain accuracy constraints that must hold for a hypothetical soccer player, the results show a sufficient trajectory prediction. This makes trajectory prediction feasible for the 2050 scenario.

Zusammenfassung

Das langfristige Vorhaben der RoboCup Initiative ist der Sieg eines Teams bestehend aus autonomen Roboterfußballspielern über den amtierenden, menschlichen Fußballweltmeister im Jahr 2050. Bis zum heutigen Zeitpunkt kann die Frage, ob die Initiative mit diesem Vorhaben Erfolg haben wird, nicht beantwortet werden. Obgleich der Fortschritt innerhalb der Forschungsgemeinschaft des Roboter-Fußballs beachtlich ist, ist nicht klar, ob dieser ausreichend ist, um bis zu diesem Zeitpunkt wettbewerbsfähige humanoide Spieler zu entwickeln. Die Frage darf auch umgekehrt gestellt werden: Gibt es bereits in einem Bereich der Forschung ausreichend Wissen für das Szenario 2050? Diese Arbeit nimmt sich dieser Frage im Bezug auf das Teilgebiet der Wahrnehmung bei fußballspielenden Robotern an. Ziel dieser Arbeit ist die Beantwortung folgender, präzisierter Frage: Ist die Genauigkeit der Flugbahnvorhersage unter gewissen Einschränkungen ausreichend für einen hypothetischen Roboter-Fußballspieler?

Die Sensorik für diesen hypothetischen Roboter ist ein Kamera-Inertialsensor, welcher nicht nur die Wahrnehmung innerhalb der Umgebung, sondern auch das Verfolgen der Bewegung des Sensors relativ zu sich selbst erlaubt. Um die Frage zu beantworten, ob die Genauigkeit ausreichend ist, wurde folgende experimentell gestützte Analyse durchgeführt: Das Kamera-Inertialsensorsystem zeichnet auf einem Fußballfeld Daten inklusive Position und Größe des Balls so wie besondere Punkte des Feldes auf. Diese Messdaten können genutzt werden, um den Zustand des Balles zu jedem Zeitpunkt des Fluges über eine Ausgleichsrechnung zwischen dem Bewegungsmodell des Balles und den Messungen zu ermitteln. Die Genauigkeit wird schließlich bestimmt, indem alle Zustände innerhalb einer Flugbahn geschätzt und die Auftreffpunkte der vorhergesagten Flugbahnen mit dem realen Auftreffpunkt verglichen werden.

Die Resultate haben ergeben, dass die Flugbahn von fliegenden Bällen über die Zeit zuverlässig ermittelt werden kann. Unter der Annahme von gewissen Genauigkeitsbedingungen für einen hypothetischen, fußballspielenden Roboter zeigen die weiteren Resultate, dass eine ausreichende Flugbahnvorhersage bezüglich dieser Genauigkeitsbedingungen vorliegt. Dies macht den Einsatz der Flugbahnvorhersage im Jahre 2050 zu einem realistischen Vorhaben.

Erklärung

Hiermit versichere ich, Oliver Birbach, diese Diplomarbeit ohne fremde Hilfe angefertigt zu haben. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen sind, sind als solche kenntlich gemacht.

Unterschrift

Acknowledgements

First of all, I would like to thank Udo Frese for offering me this exciting topic, for taking the time to discuss problems, answering all my related questions and offering many useful suggestions. Thank you very much.

I also thank Florian Penquitt for setting up the experiments and co-conducting them with me on the field. Further, I would like to thank the two helpers who provided the additional hands that were neccessary for the experiments.

Special thanks goes to the administration of the sport facilities for allowing us to run the experiments during the holidays.

Finally, I would like to thank my parents, especially my mother, for always supporting me in every way.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This thesis is about trajectory prediction of soccer balls observed by a free-moving camera-inertial sensor and whether this trajectory prediction would be sufficient for a hypothetical robotic soccer player or not.

To accomplish this task, a person was wearing a helmet on which the sensor was mounted so that the field of view of the person and of the camera were nearly identical. This person was then moving on a soccer field and tracked different trajectories of ball flights with his eyes while the data of the sensor were recorded. To determine the accuracy of the trajectory prediction, the predicted bouncing points from the recorded sensor data will be compared to the real bouncing points.

1.1 Motivation

Robotic soccer established itself as the benchmark for development within the robotic and artificial intelligence community right after the win of IBM's Deep Blue over the world champion in chess in 1997. Deep Blue's success was quite predictable. Its strength was the exploitation of the deterministic nature of chess by raw computational power. In contrast to chess, robotic soccer is by far more complex since the robots must be able to integrate perception, reasoning and acting within a real soccer environment in real-time.

The idea of soccer-playing robots as a challenge was raised by Sahota and Mackworth [1] along with the emergence of the Situated Agent approach in 1994. This approach suggests that progress within the field of artificial intelligence is achieved by taking Brooks' [2] guiding principles to a specific paradigmatic task domain which allows testing and development. Soccer was chosen to be this paradigmatic task domain and the idea of soccer-playing robots as a standard AI problem led to the formation of the Robot World Cup Initiative (RoboCup) [3] in 1997. Shortly after its formation, the goal of the initiative was

stated:

> By mid-21st century, a team of fully autonomous humanoid robot soccer players shall win the soccer game, comply with the official rule of the FIFA, against the winner of the most recent World Cup.[4]

Since their foundation, world-wide and regional RoboCup competitions, where robotic soccer systems compete each other, are held annually.

Unlike Deep Blue's predictable success, it is not known if the RoboCup initiative can reach its goal within the specified time. Although the progress made in the last decade during the competitions is remarkable, areas exist, in which the progress is quite slow. When thinking of a robot's operation as a series of Sense/Think/Act cycles, the most obvious lack of technology is within the area of actuators, especially the lack of a well-crafted humanoid robot. The current state-of-the-art humanoid robot, the Honda ASIMO [5], is a respectable piece of engineering which is capable of executing simple human tasks. But when it comes to soccer, the robot fails to provide human-like physical skills like fast running and shooting.

Regardless of the fact that breakthrough progress is needed, there might be areas where the substantial knowledge is already obtained or where just a little more research needs to be done. Obviously, the following question arises within this context: Which areas within the robotic research community are already mature enough for the 2050 scenario?

1.2 Goals of the Thesis

This thesis tries to give an answer to the above question within the area of sensing. Actually, the question to be answered here is: Is the accuracy of ball trajectory prediction sufficient for a hypothetical robotic soccer player when the ball is observed by a camera-inertial sensor?

Answering the question by simply stating yes or no will not do the job. In this case, the question implies a set of sub-questions, such as:

- How good is the overall accuracy of trajectory prediction over time?
- Is there a certain moment at which the prediction can be assumed to be sufficient enough?
- How well does the prediction work for different trajectories?

Answering the question requires the definition of accuracy in a robotic soccer context. With respect to this definition, answers to the above questions will be given through the evaluation of various meaningful trajectories obtained by experiments.

Figure 1.1: Camera-inertial sensor attached to a helmet worn by a human. This setup was used during the experiments.

1.3 Experimental and Estimation Procedure

Since no humanoid robot was available, experiments to obtain the required data for the evaluation were conducted using a human wearing a helmet on which the camera-inertial sensor was attached to (see figure 1.1). This procedure is legitimate because a humanoid robot tries to mimic human behaviour. The person was moving on a real soccer field and tracked trajectories of ball flights while the data of the sensor were recorded. This recorded data included the acceleration and angular velocity measured by the inertial sensor and the images obtained from the camera sensor. Since no automatic image processing was available, all relevant objects in the images were extracted manually. This manual vision ensured that all features were extracted reliably from the images. The objects of interest in the images were the ball and certains landmarks defined by line intersections on the field and both goals.

The state that is estimated from the measurements is composed out of two components. First, the ball state is modeled as the ball's position and velocity with respect to a global coordinate system. The other component is the sensor's position, velocity and orientation with respect to the same global coordinate system.

The estimation of this state is performed using two methods. The first method is the

method of least squares where a stack of states is estimated at once using all measurements up to this state. The other method is the so-called unscented Kalman filter which recursively estimates a state from the previously estimated state and the last measurement of the sensors. Because of the free moving sensor in the environment and the monocular vision the estimation becomes particularly difficult.

1.4 Benefits of Camera-Inertial Sensors

The sensing element in this work is the combination of camera and inertial sensor.

Extracting metric information about the environment using a camera is known as photogrammetry. It relies on methods from optics and projective geometry to map geometric objects sensed in the environment into the image plane. Cameras as sensors are often used for self-localization and perception within a known environment. The sensing error from camera measurements is known to be absolute.

Inertial sensors are used to measure the ego-motion of the object on which the sensor is attached to. Inertial sensors usually consist of an accelerometer and a gyroscope. The accelerometer measures linear acceleration, whereas the gyroscope measures angular velocity applied to the object. All sensing occurs in the coordinate system of the inertial sensor and no external reference is required. Applications of inertial sensors are tracking of large-scale vehicles such as aircrafts or spaceships, and low-cost, camera-less motion-capturing. In contrast to the cameras, the sensing error of inertial sensors accumulates over time which has to be compensated.

Combining both sensors is a relatively new sensing discipline. The benefit of the combination is the additional information about the camera's motion from the inertial sensor. Any application requiring spatial information about the sensor, such as localization, will become much more robust, especially when no or almost no visual information are available. Furthermore, the accumulated sensing error problem of the inertial sensor can be compensated by integrating localization-specific measurements from the camera. But this combination bears a new problem. Since the camera and the inertial sensor are operating in different coordinate systems, the parameters for transforming measurements from one system into the other are required. Actually, methods exist for finding these parameters. They will be part of this thesis.

1.5 Related Work

In [6], Kurlbaum presented a proof-of-concept system for tracking and predicting the trajectory of flying balls within a known environment using a camera-inertial sensor. The

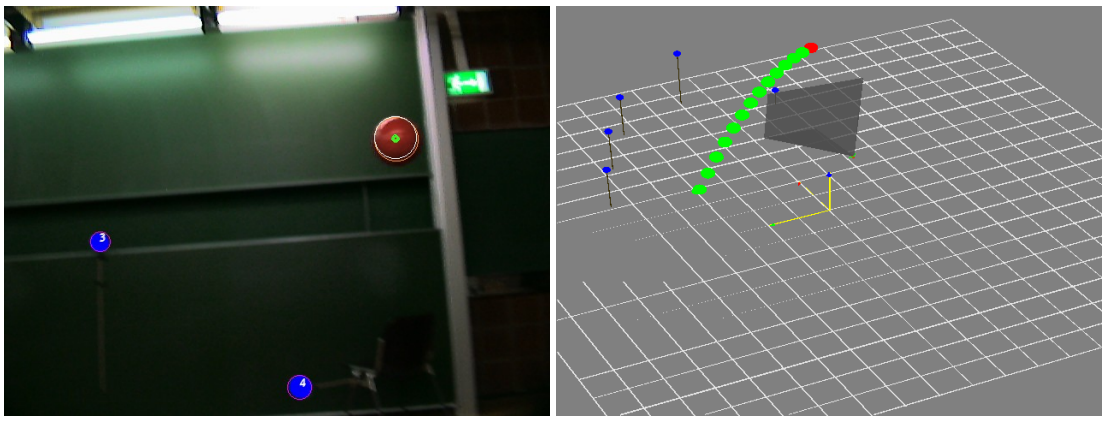

Figure 1.2: Two images of a soccer ball tracking and prediction system using a camera-inertial sensor. (Left) Annotated camera image showing the ball and the visible landmarks. The estimated position of the ball and the landmarks are shown as circles in image coordinates. (Right) Three-dimensional model of the tracking system. The estimated ball position is shown as a red sphere. The predicted trajectory is depicted as a series of green spheres. Images courtesy of [6].

tracking relied on an unscented Kalman filter which estimated the state of the ball by integrating measurements from a color-based vision and an inertial sensor. Although most of the model parameters such as noise parameters and the transformation between the sensors were guessed by the author, experiments showed reasonable estimation results after a couple of measurements. Furthermore, the estimated ball states during the flight showed a visually accurate trajectory prediction. Two pictures of the system's software are shown in figure 1.2.

How to benefit from such a trajectory prediction is shown by Frese et al. [7]. There, a vision-based soft ball tracker, which uses trajectory prediction as the basis for controlling a robotic ball catcher, is presented. Although only a static stereo camera system is used in their setup, this work is an interesting example highlighting the need of accuracy during estimation and prediction. In their work, the thrown ball is extracted out of the images by segmentation based on difference images. The three-dimensional tracking is performed using the extended Kalman filter. After the filter reports a sufficiently small covariance, prediction is used to compute the catch-point for the ball catcher, which is used for controlling the robot's actuators. Experiments were conducted and the catcher was successful in the majority of throws. If the catcher failed, most of the faults were caused by limitations of the experimental setup. A video sequence overlay of a successful throw is shown in figure 1.3.

One usage of trajectory prediction within the context of RoboCup is presented by Simon in [8] as part of the vision system of the former Small Size team FU-Fighters. A method

Figure 1.3: Video sequence overlay of a successful catch by the robotic ball-catcher showing the trajectory of the thrown ball. Image courtesy of [7].

for recognizing chipped kicks (kicks, where vertical and horizontal forces are applied to the ball) along with a method of predicting the bouncing point is implemented using only the measurements from a single overhead camera. A mapping between the measurement of the ball in the image plane and the dynamics of the ball in the world is established. Along with the a-priori information about the height of the camera, three measurements are sufficient to construct a system of equations which solves the problem uniquely. The accuracy of the estimation can be increased by incorporating more measurements and solving the over-determined system. Within their setup, Simon states that ten measurements were sufficient for a good approximation of the trajectory and twenty measurements yielded to an accuracy below $1\ cm$ for the estimated bouncing point. Figure 1.4 shows a plot of the trajectory measured by the overhead camera and the corresponding three-dimensional model of the actual trajectory on the field.

1.6 Guide to the Thesis

The thesis is structured as follows:

Chapter 1 introduces the general question to be answered, outlines used methods and shows some related work.

Chapter 2 gives an in-depth introduction of stochastic methods for state estimation used

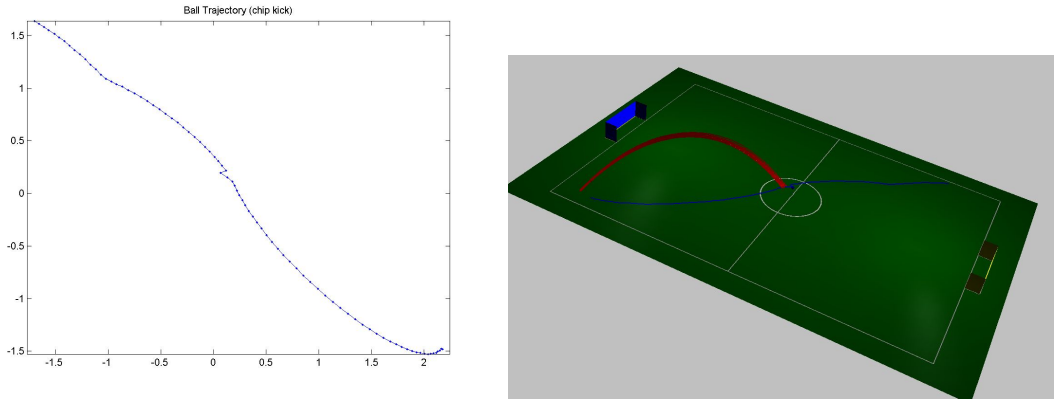

Figure 1.4: Two images visualizing the estimation of chip-kicked ball trajectories used by the FU-Fighters.(Left) Trajectory plot of the ball measured by the overhead camera. (Right) Corresponding three-dimensional model visualizing the actual ball trajectory on the field. Images courtesy of [8].

in this thesis. It also describes how to handle orientation representations during estimation.

Chapter 3 describes the problem of calibration between the rigid combination of camera and inertial sensor. Very high accuracy while transforming measurements between both sensors is vital for the success of an accuracy analysis. An approach how to calibrate the rotational difference and the camera calibration in one procedure will be given.

Chapter 4 introduces the necessary models for observing a ball in flight using a camera-inertial sensor along with the associated noise parameters. An algorithm is introduced to calculate a certain set of noise parameters from the measurements.

Chapter 5 explains the experimental setup in detail and describes how the estimators were implemented using the model.

Chapter 6 analyzes the data obtained from the experiments with respect to accuracy for a hypothetical humanoid robot. A couple of ball flights and their prediction performance will be evaluated.

Chapter 7 finally concludes the thesis and provides an outlook for upcoming work.

Chapter 2

Stochastic Estimation

This chapter introduces the estimation techniques used in this thesis for determining values of unknown parameters in a statistical model. This statistical model may reach from a simple function combined with an uncertainty parameter up to complex dynamical systems involving several functions and uncertainty parameters. Furthermore, some parameters may not be usual vectors, such as orientation representations. This may cause problems during the estimation procedure.

The two estimation techniques introduced here are suitable for estimating nontrivial dynamical systems such as a ball in flight or a free moving camera-inertial sensor. In the upcoming sections it will be shown how they work and how they can be extended to estimate the above mentioned orientation representations as well.

2.1 Nonlinear Least Squares Estimation

The first method is known as nonlinear least squares estimation and is a popular technique for fitting experimentally obtained data to arbitrary functions (including functions describing dynamic processes) depending on nonlinear parameters. Based on the historical work of Gauss' orbit determination of Ceres, the least squares formulae are nowadays used for common tasks like curve-fitting and sensor calibration.

Suppose a set of measurement vectors z_i is obtained depending nonlinearly on the parameters x of the model function f_i, and a normal distribution of the errors ε_i and independence in each observation i is assumed. The relation between measurement and model can then be written as:

$$z_i = f_i(x) + \varepsilon_i \tag{2.1}$$

With respect to the observation of the flight of a ball using a camera-inertial sensor, the measurements z_i would be the vision and inertial sensor measurements, while the pa-

rameter x contains the set of ball and sensor states of the flight. The dynamics of the ball and the sensor are modeled by f_i.

The difference between a measurement z_i and the predicted value $f_i(x)$ is known as the residual r_i:

$$r_i = z_i - f_i(x) \tag{2.2}$$

The goal of least squares estimation is to find the parameters x of f which minimize the sum of squared residuals χ^2

$$\chi^2 = \sum_i r_i^T \ \Sigma_i^{-1} \ r_i = \sum_i (z_i - f_i(x))^T \ \Sigma_i^{-1} \ (z_i - f_i(x)) \tag{2.3}$$

weighted by Σ_i, representing the amount of uncertainty within the measurements z_i. Σ_i is the so-called measurement covariance.

Besides this quite intuitive way of looking onto the problem, a probabilistic point of view can be formulated, too (as done in [9]). The question to be answered here is: Given a set of measurements z, what is the probability that a particular set of parameters x are the rights ones?

$$p(X = x \mid Z = z) \tag{2.4}$$

Applying Bayes' rule to this probability distribution under the assumption that no a-priori information are available leads to:

$$p(X = x \mid Z = z) = \frac{p(Z = z \mid X = x) \ p(X = x)}{p(Z = z)} \tag{2.5}$$

$$\propto \ p(Z = z \mid X = x) \tag{2.6}$$

The probability of the measurements z given the parameters x is known as the likelihood of the parameters x given the measurements z. The most probable parameters are the parameters that maximize the probability density function $p(Z = z \mid X = x)$. The technique of finding parameters in this way is known as maximum likelihood estimation.

Assuming, as above, that all measurement data is independent and normally distributed, the joint probability density function of all measurements is the product of all individual probability density functions:

$$p(Z = z \mid X = x) = \prod_i p(Z_i = z_i \mid X = x) \tag{2.7}$$

$$= \prod_i p(\varepsilon_i = z_i - f_i(x)) \tag{2.8}$$

Substituting the normal density function for

$$p(\varepsilon_i = z_i - f_i(x)) = \frac{1}{(2\pi)^{n/2}|\Sigma_i|^{1/2}} \exp(-\frac{1}{2}(z_i - f_i(x))^T \Sigma_i^{-1}(z_i - f_i(x))) \tag{2.9}$$

leads to:

$$p(Z = z \mid X = x) = \prod_i \frac{1}{(2\pi)^{n/2}|\Sigma_i|^{1/2}} \exp(-\frac{1}{2}(z_i - f_i(x))^T \Sigma_i^{-1}(z_i - f_i(x))) \tag{2.10}$$

By minimizing the negative logarithm (which is equal to maximizing the positive logarithm) of the probability density function, we get

$$-\log p(Z = z \mid X = x) = \text{const.} + \sum_i^n (z_i - f_i(x))^T \Sigma_i^{-1}(z_i - f_i(x)) \tag{2.11}$$

This equals equation (2.3), which leads to the conclusion that the method of least squares is an instance of maximum likelihood estimation.

Note, without loss of generality $n = 1$ is achieved by stacking z and f and composing Σ diagonally.

Solving the class of least squares problems is well known. While linear least squares problems have a closed-form solution [9], general nonlinear problems can be solved using iterative methods. These methods will be introduced in the following sections.

2.1.1 Gradient Descent Algorithm

One way to solve the problem is to iteratively step in the negative direction of the gradient. Starting from a guess x_i, the next approximation x_{i+1} is calculated by

$$x_{i+1} = x_i - \gamma J^T \Sigma^{-1}(z - f(x_i)) \tag{2.12}$$

where

$$J = \frac{\partial f(x)}{\partial x}\Big|_{x=x_i} \tag{2.13}$$

is the Jacobian of f at x_i and γ is a damping factor controlling the step length. This factor is necessary because the gradient only provides the search direction, but not the length of innovation. Usually, a seperate line search is performed to compute the optimal step length.

While the gradient descent algorithm usually steps fast towards the minimum in the first few iterations, it tends to converge slowly near the actual minimum. Another drawback is the need for careful adjustment of the damping factor γ, which often requires additional function evaluations.

GRADIENT DESCENT ALGORITHM(x_0)

$x \leftarrow x_0$
while not converged
do
  $J \leftarrow \frac{\partial f(\tilde{x})}{\partial \tilde{x}} \Big|_{\tilde{x}=x}$
  $g \leftarrow -J^T \Sigma^{-1}(z - f(x))$
  $\gamma \leftarrow$ LINESEARCH(x, g)
  $x \leftarrow x + \gamma\, g$

return x

Table 2.1: The gradient descent algorithm. It searches the minimum of χ^2 by stepping iteratively in the negative direction of its gradient.

It should not be left unmentioned that gradient descent is a general purpose optimization algorithm that is also applicable to least squares problems. However, it usually performs poorly within this class of problems and is not recommended in practice.

The algorithm is summarized in table 2.1.

2.1.2 Gauss-Newton Algorithm

A better way to find the parameters in a least square sense is the expansion of the function f as a first order Taylor series as done in [10]. Starting from an initial estimate x_i, the linearization

$$f(x) \approx f(x_i) + \frac{\partial f(x)}{\partial x} \Big|_{x=x_i} \cdot (x - x_i) \tag{2.14}$$

at x_i makes x linear in f and therefore applicable to the linear least squares equations, where the minimum can be computed explicitly. The equation for the next estimate reads

$$x_{i+1} = x_i + (J^T \Sigma^{-1} J)^{-1} J^T \Sigma^{-1}(z - f(x_i)) \tag{2.15}$$

where Σ is the amount of confidence assumed for the measurement z expressed as the covariance. The quality of parameter estimation depends on how linear f behaves in the neighbourhood of the linearization point x_i. Usually, the estimation can be improved by using the latest estimation as the new linearization point and iterating until the convergence criterion is reached.

The algorithm is named Gauss-Newton algorithm and is known to converge well when starting from an initial guess near the final minimum but usually fails when starting far off the final minimum.

GAUSS-NEWTON ALGORITHM(x_0)

$x \leftarrow x_0$
while not converged
do
  $J \leftarrow \frac{\partial f(\tilde{x})}{\partial \tilde{x}} \Big|_{\tilde{x}=x}$
  $\alpha \leftarrow J^T \Sigma^{-1} J$
  $\beta \leftarrow J^T \Sigma^{-1} \cdot (z - f(x))$
  $\delta \leftarrow \alpha^{-1} \cdot \beta$
  $x \leftarrow x + \delta$

return x

Table 2.2: The Gauss-Newton algorithm. It solves the least squares problem by linearizing the model function f at x.

The full algorithm is given in table 2.2

2.1.3 Levenberg-Marquardt Algorithm

With both of the above methods in mind, Levenberg [11] and Marquardt [12] developed a method combining the benefits of the Gauss-Newton algorithm and gradient descent while avoiding their most serious limitations. A damping-factor $\lambda > 0$ is introduced and the equation for the next estimate $i + 1$ reads

$$x_{i+1} = x_i + (J^T \Sigma^{-1} J + \lambda I)^{-1} J^T \Sigma^{-1} (z - f(x_i)) \tag{2.16}$$

where J is the Jacobian of f at x_i, Σ the variances of the measurements z and I the identity matrix.

At each iteration step, λ is adjusted to reflect the success in the previous estimation step by the following rules [9]:

1. If $\chi^2(x_{i+1}) < \chi^2(x_i)$, decrease λ by the factor 10 .
2. If $\chi^2(x_{i+1}) \geq \chi^2(x_i)$, increase λ by the factor 10 and set x_{i+1} to x_i.

In words, if the newly obtained parameters minimize the object function better than the ones before, the parameters might be close to the minimum, λ will be reduced and the algorithm behaves like the Gauss-Newton algorithm. Otherwise, if the new parameters

LEVENBERG-MARQUARDT ALGORITHM(x_0)

$x \leftarrow x_0$
while not converged
do
  $J \leftarrow \frac{\partial f(\tilde{x})}{\partial \tilde{x}} \Big|_{\tilde{x}=x}$
  $\alpha \leftarrow J^T \Sigma^{-1} J$
  $\beta \leftarrow J^T \Sigma^{-1} \cdot (z - f(x))$
  $\delta \leftarrow (\alpha + \lambda I)^{-1} \cdot \beta$
  if $f(x + \delta) \geq f(x)$
    then $\lambda \leftarrow \lambda \cdot 10$
    else $\lambda \leftarrow \lambda \cdot 0.1$
      $x \leftarrow x + \delta$

return x

Table 2.3: The Levenberg-Marquardt algorithm for solving nonlinear least squares problems. The algorithm interpolates between the Gauss-Newton algorithm and gradient descent by adjusting the damping factor λ.

show poor perfomance, λ will be increased and $(J^T \Sigma^{-1} J + \lambda I)$ becomes diagonally dominant acting like gradient descent. Furthermore, the new estimate will be rejected and the estimate of the new iteration step remains the old one.

The algorithm therefore combines the ability of gradient descent to approach the final minimum from starting far away and the ability of the Gauss-Newton algorithm to converge rapidly once the parameters are near the final minimum. This robustness made the Levenberg-Marquardt algorithm a popular choice for solving nonlinear least squares problems.

The full algorithm is described in table 2.3.

2.1.4 Estimation of Orientations

Estimation of orientation poses a special problem. An orientation is a member of the so-called rotation group $SO(3)$. $SO(3)$ is a 3-manifold, meaning that an orientation can be locally parameterized with three numbers. However, there are no global parameterizations in $\mathbb{R}^3$ without discontinuities, so-called singularities. Singularity free parameterizations of rotations are achieved by leaving $\mathbb{R}^3$ as done by unit quaternions, which form a sphere S^3 in $\mathbb{R}^4$, or 3×3 special orthogonal matrices $SO(3)$ in $\mathbb{R}^9$.

Applied to the method of least squares, the intuitive idea is to represent the orientation as a vector with nine elements, representing a rotation matrix, or as a vector containing four elements, representing a quaternion. But this bears a substantial problem: the algorithm does not know anything about the underlying constraints of the vector containing the orientation representation. Since the optimization expects parameters in Euclidean space, the resulting parameters in the optimization steps might not be valid rotations. The vectors of the rotation matrix might not be unit vectors or orthogonal to each other anymore. When using unit quaternions, the length of the quaternion might not be 1 anymore.

Since the estimation of the sensor's orientation is required for a ball trajectory estimation, a way to handle this problem is needed. One way is to represent the rotation as a singularity free parameterization in parameter space, which is only changed by small locally defined $\mathbb{R}^3$ rotations [13] [10]. The idea is to let the algorithm operate on the small locally defined $\mathbb{R}^3$ parameterization only. This locally defined parameterization originates from the singularity free parameterization. The algorithm is then instructed to find the $\mathbb{R}^3$ rotation which changes the singularity free parameterization so that the orientation becomes optimal.

These changes are applied by the operator $\oplus$, which adds a small rotation from vector space $\mathbb{R}^n$ to parameter space S. Formally the operation $\oplus : S \times \mathbb{R}^n \rightarrow S$ reads

$$s \oplus d = \begin{cases} s_{\mathbb{R}} + d_{\mathbb{R}} & \text{, if vector part} \\ s_{SO(3)} \cdot \mathrm{Rot}(d_{\mathbb{R}^3}) & \text{, if rotational part} \end{cases} \tag{2.17}$$

where $\mathrm{Rot}()$ defines a valid $SO(3)$ or S^3 rotational representation out of the $\mathbb{R}^3$ vector which is part of d. Since the $\mathbb{R}^3$ vector only holds small locally defined rotation changes, its representation is free from singularities. Therefore, Rot can be applied to compute the corresponding singularity free representation of the vector which is then added to the orientation in the parameter space.

As it can be seen from the definition, the operator performs an ordinary vector addition to the vector portion and only applies the locally defined $\mathbb{R}^3$ rotation to the singularity free rotation representation in the parameter space S.

With the introduction of the $\oplus$-operator, the least squares optimization of a model function f with respect to its parameters $s \in S$ can be reformulated. An additional function g and a mapping from g to the model function f is introduced

$$g_i(d_i) = f(s_i \oplus d_i) \tag{2.18}$$

where $s_i \in S$ is the current parameter estimate, d_i belongs to $\mathbb{R}^n$ and i represents the iteration step. The optimization problem now reduces to the problem of finding the vector d which minimizes g, or, in the context of f, of finding the vector d which minimizes f when applied to f along with s using $\oplus$.

Applied to the Levenberg-Marquardt algorithm (see table 2.3), the change d_i at iteration i is calculated by altering lines 5-7

$$\begin{aligned} \alpha &\leftarrow J^T \Sigma^{-1} J && (2.19) \\ \beta &\leftarrow J^T \Sigma^{-1} \cdot (z - g_i(0)) && (2.20) \\ d_i &\leftarrow (\alpha + \lambda I)^{-1} \cdot \beta && (2.21) \end{aligned}$$

where J is the Jacobian of g_i at 0.

The test if the current step was successful is performed between $g_i(0)$ and $g_i(d_i)$. Finally, if the test was actually successful, the new estimate at $i+1$ is computed by applying the computed change d_i to the current estimate s_i using $\oplus$

$$s_{i+1} = s_i \oplus d_i \tag{2.22}$$

resulting in the new parameter estimate s_{i+1}.

2.1.5 Estimation of Dynamical Systems

A dynamical system is a formalization of how a state x_{t-1} develops into another state x_t over time. A state x consists of a set of real numbers. As mentioned earlier, the pose and velocity of the sensor and the ball and the sensor's orientation are represented by the state. Over the time of the ball's flight the state changes. Describing this change of state is done by dynamical systems.

Estimating the state of a dynamical system using the method of least squares requires to estimate a stack of states up to the current state. For every new measurement at time t, the parameters that need to be estimated are

$$p = \begin{pmatrix} x_0 \\ \vdots \\ x_{t-1} \\ x_t \end{pmatrix} \tag{2.23}$$

where x_t is the state at time t.

The dynamics are modeled by a function which is applied to each adjacent state pair out of this stack. Written as the sum of the squared residuals, a least square optimization with dynamics in mind looks like:

$$\chi^2 = \sum_{i=1}^{t} (x_{i-1} - g(x_i, z_i))^T \Sigma_t (x_{i-1} - g(x_i, z_i)) \tag{2.24}$$

Here, the function g models the dynamics by performing an inverse state transition (computing the state at $t-1$ from the state t and an eventually known dynamic measurement)

to calculate the predicted state at $t-1$ which is compared to the actual state x_{t-1} from p. Since none or only partial measurements are involved, this function will be called virtual measurement function.

A simple example to illustrate the behaviour of equation (2.24) with respect to the experiments of this thesis will be given. For this example, the sensor's state over time should be estimated while its motion is modeled using Newtonian mechanics. The acceleration and angular velocity of the sensor is known from inertial sensor measurements. With respect to the equation x would be the sensor state (position, velocity and orientation), the model function g would represenent inverse state transition function (incorporating Newtonian mechanics) and z would represent the inertial measurements. By obtaining a couple of measurements up to time t from the inertial sensor, it is possible to compute all states of the sensor from time 1 to t using the method of least squares. Similarly, this technique is also applicable for the estimation of a ball flight trajectory.

Since the method of least squares estimates the full dynamical path incorporating all measurements up to the current state, it performs a so-called *full estimation*.

2.2 Estimation of Dynamical Systems Using Kalman Filters

The class of Kalman filters are stochastic state estimation techniques for dynamical systems. They were introduced in [14] to estimate the state (in this case the state of the ball and the camera) of time-discrete linear systems from measurements which are corrupted by noise (for this work, the measurements of the vision and the inertial sensor).

Kalman filters work recursively. Given the input of the last state at time $t-1$, the observed measurements z_t and control data u_t, the state at time t will be computed. No other information is used throughout the recursive estimation, it does not access past states or measurements because every information provided by them is already incorporated in the current state. Because of its recursive nature, Kalman filters perform an *online estimation* of the state which is computationally superior compared to the method of least squares. This property makes Kalman filter capable of estimating the state of the system in real time.

As with the method of least squares, the estimation problem can be formulated in a probabilistic context. The question to be answered here is: What is the probability distribution over the state x_t given all previous measurements $z_{1:t}$ and previous controls $u_{1:t}$? Formally written:

$$p(x_t \mid z_{1:t},\ u_{1:t}) \tag{2.25}$$

The underlying algorithm of all Kalman filters is the so-called Bayes filter. Along with the Markov assumption, which states that the current state is itself a complete summary

of the past, the Bayes filter provides the rules to compute this probability recursively at time t out of the probability at time $t-1$, the current measurement and current control data.

2.2.1 Linear Gaussian Systems

As mentioned above, the Kalman filter is an implementation of the Bayes filter. The key to Kalman filtering is the usage of Gaussians. This means that the filter represents the state of a system at time t as a normal distribution with the mean μ_t along with the covariance Σ_t. To ensure that a state is always a valid Gaussian, certain properties of the system must hold.

First, Kalman filters assume linear dynamics. That is, the state transition from $t-1$ to t must be linear in its arguments plus some Gaussian noise. Formally, it reads

$$x_t = A_t x_{t-1} + B_t u_t + \varepsilon_t \tag{2.26}$$

where x_t and x_{t-1} are the state vectors, A_t is the state transition matrix and B_t is the matrix that relates the eventually known control vector u_t into the state transition. ε_t is a random variable that models the uncertainty that results from the state transition. It is distributed with zero mean and covariance R_t.

Secondly, Kalman filters also assume linearity for the relation between state and measurement. Here, a matrix C_t maps the state x_t into a measurement z_t plus Gaussian noise δ_t:

$$z_t = C_t x_t + \delta_t \tag{2.27}$$

As with the distribution of ε_t, the distribution in this equation has zero mean but the covariance is given by Q_t.

Finally, the initial state of the system must be a valid Gaussian.

2.2.2 Kalman Filter Algorithm

With these three requirements in mind, the Kalman filter algorithm (see table 2.4) can be formulated. The algorithm updates a valid Gaussian state at time $t-1$ with mean μ_{t-1} and covariance Σ_{t-1} to another valid Gaussian at time t with mean μ_t and covariance Σ_t using the current measurement z_t and control u_t.

The algorithm works in a predict/update manner. First, a predicted state with mean $\bar{\mu}_t$ and covariance $\bar{\Sigma}_t$ is computed from the state transition function using the previous state μ_{t-1}, Σ_{t-1} and control u_t (line 1-2).

This predicted state is now the basis for the update step. Here, the Kalman gain is computed in line 3. This value determines how much the measurement z_t should be incor-

KALMAN FILTER ALGORITHM$(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t)$

1 $\bar{\mu}_t = A_t\mu_{t-1} + B_t u_t$
2 $\bar{\Sigma}_t = A_t\Sigma_{t-1}A_t^T + R_t$
3 $K_t = \bar{\Sigma}_t C_t^T (C_t\bar{\Sigma}_t C_t^T + Q_t)^{-1}$
4 $\mu_t = \bar{\mu}_t + K_t(z_t - C_t\bar{\mu}_t)$
5 $\Sigma_t = (I - K_t C_t)\bar{\Sigma}_t$
6 **return** μ_t, Σ_t

Table 2.4: The Kalman filter algorithm. The algorithm updates a state at time $t-1$ to a state at time t using the measurement z_t and the control u_t along with the linear state transition and measurement equations.

porated into the updated state. By computing the measurement residual by subtracting the predicted measurement $C_t\bar{\mu}_t$ from the actual measurement z_t, the new mean μ_t is determined by adding the product of Kalman gain and residual to the predicted mean $\bar{\mu}_t$ (expressed in line 4). Finally the covariance Σ_t of the new state x_t is adjusted in line 5.

2.2.3 Unscented Kalman Filter

Since linear systems rarely exist while observing physical phenomena and because handling nonlinearity was desired, the Kalman filter was extended allowing the estimation of states within a system with underlying nonlinear models. These derivates are known as the extended Kalman filter (EKF) [15] and as the unscented Kalman filter (UKF) [16].

All Kalman filters require that the underlying system is a linear Gaussian system. This requirement ensures the transformation from one valid Gaussian state into another. When nonlinear models are used to describe the dynamics, the transition and measurement equations become

$$x_t = g(x_{t-1}, u_t) + \varepsilon_t \tag{2.28}$$

$$z_t = h(x_t) + \delta_t \tag{2.29}$$

where the nonlinear function g replaces A_t and B_t and the function h replaces C_t.

The idea of nonlinear filtering is to approximate a linear transformation by linearizing the nonlinear functions g and h. A common linearization technique is the Taylor expansion (which has already been introduced within the Gauss-Newton algorithm in section 2.1.2) and is used by the extended Kalman filter. The drawback of this linearization technique and therefore of the EKF is that the filter might perform poorly, especially when using functions which behave highly nonlinear.

A better approach, by using a special transformation, is used by the unscented Kalman filter. This filter linearizes a nonlinear function g at the current Gaussian state by applying a so called unscented transformation. Here, a set of deterministic chosen sample points (also known as sigma points), which represent the mean and the covariance of the Gaussian, are computed. These points are then propagated through the nonlinear function g and recombined to a valid Gaussian state.

In detail, the sigma points of state x with mean μ and covariance Σ are computed by

$$\mathcal{X}_0 = \mu \qquad w_m = \frac{1}{2n+1} \tag{2.30}$$

$$\mathcal{X}_i = \mu + (\sqrt{\Sigma})_i \qquad w_c = \frac{1}{2} \tag{2.31}$$

$$\mathcal{X}_{i+n} = \mu - (\sqrt{\Sigma})_i \tag{2.32}$$

for $i = 1, \ldots, n$ resulting in $2n+1$ sigma points. The values w_m and w_c are the weights of the sigma points and are used for the recombination of the propagated sigma points to a transformed mean and covariance. Note that the actual values of w_m and w_c may differ from application to application, but here all sigma points are weighted equally. The used square-root $\sqrt{\Sigma}$ denotes the matrix square-root (Cholesky factorization) of the positive semidefinite covariance matrix Σ.

After obtaining all sigma points, these are propagated through the nonlinear function:

$$\bar{\mathcal{X}}_i = g(\mathcal{X}_i) \quad \text{for } i = 0, \ldots, 2n. \tag{2.33}$$

The mean $\bar{\mu}$ and covariance $\bar{\Sigma}$ for the new Gaussian are calculated by a weighted mean and covariance of the propagated sigma points:

$$\bar{\mu} = \sum_{i=0}^{2n} w_m \bar{\mathcal{X}}_i \tag{2.34}$$

$$\bar{\Sigma} = \sum_{i=0}^{2n} w_c (\bar{\mathcal{X}}_i - \bar{\mu})(\bar{\mathcal{X}}_i - \bar{\mu})^T \tag{2.35}$$

The transformation of the sigma points using the nonlinear function is visualized in figure 2.1.

Applying the unscented transformation to the Kalman filter algorithm results in the unscented Kalman filter, which is described in table 2.5. First, sigma points of the previous state are determined as mentioned above (line 1). These points are then propagated through the nonlinear state transition function g in line 2. The predicted mean $\bar{\mu}_t$ and covariance $\bar{\Sigma}_t$ are then computed by recombining the propagated points where the noise R_t is added to the covariance to resemble the uncertainty in the dynamic process (line 3 and 4).

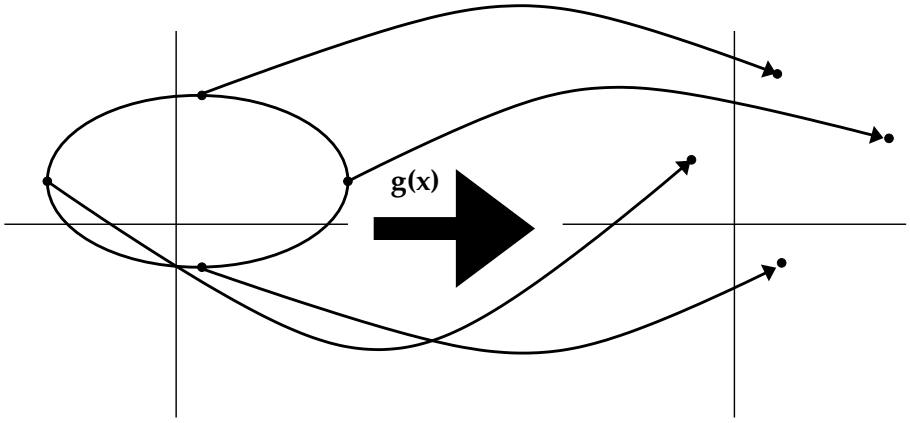

Figure 2.1: The unscented transformation transforms a set of deterministically obtained points by propagating them through the nonlinear function g. Image taken from [6].

In line 5, a new set of sigma points is extracted from the newly obtained predicted state. These sigma points, called $\bar{\mathcal{X}}_t$, are propagated through the nonlinear measurement function h and the resulting values are recombined to a predicted observation $\bar{z}_t$ and its uncertainty S_t (line 6-7). In line 8 and 9, the cross covariance is computed from which the Kalman gain is determined. From here, the computation of the new state is straightforward using the equations from the original Kalman filter.

2.2.4 Filtering Orientations

Similiar to the estimation of orientations using the method of least squares mentioned in section 2.1.4, a way to handle the 3-manifold of orientations during filtering needs to be established. The problem remains the same: The filter does not know anything about the singularity-free orientation and that it cannot be handled like a vector. Actually, this problem can be solved (as described in [10] and [6]) in a similar way as the least squares orientation estimation.

In addition to the $\oplus$ operator, the inverse operator $\ominus$ is needed:

$$\oplus : S \times \mathbb{R}^n \to S \tag{2.36}$$

$$\ominus : S \times S \to \mathbb{R}^n \tag{2.37}$$

The following property holds for both operators:

$$s_1 \oplus (s_2 \ominus s_1) = s_2 \tag{2.38}$$

Here, the operator $\ominus$ is defined as

$$s_1 \ominus s_2 = \begin{cases} s_1 - s_2 & \text{, if vector part} \\ \mathrm{aRot}(s_1^{-1} \cdot s_2) & \text{, if rotational part} \end{cases} \tag{2.39}$$

UNSCENTED KALMAN FILTER ALGORITHM$(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t)$

1 $\mathcal{X}_{t-1} = (\mu_{t-1} \quad \mu_{t-1} + \sqrt{\Sigma_{t-1}} \quad \mu_{t-1} - \sqrt{\Sigma_{t-1}})$
2 $\bar{\mathcal{X}}_t^* = g(u_t, \mathcal{X}_{t-1})$
3 $\bar{\mu}_t = \sum_{i=0}^{2n} w_m \bar{\mathcal{X}}_t^{*[i]}$
4 $\bar{\Sigma}_t = \sum_{i=0}^{2n} w_c (\bar{\mathcal{X}}_t^{*[i]} - \bar{\mu}_t)(\bar{\mathcal{X}}_t^{*[i]} - \bar{\mu}_t)^T + R_t$
5 $\bar{\mathcal{X}}_t = (\bar{\mu}_t \quad \bar{\mu}_t + \sqrt{\bar{\Sigma}_t} \quad \bar{\mu}_t - \sqrt{\bar{\Sigma}_t})$
6 $\bar{\mathcal{Z}}_t = h(\bar{\mathcal{X}}_t)$
7 $\hat{z}_t = \sum_{i=0}^{2n} w_m \bar{\mathcal{Z}}_t^{[i]}$
8 $S_t = \sum_{i=0}^{2n} w_c (\bar{\mathcal{Z}}_t^{[i]} - \hat{z}_t)(\bar{\mathcal{Z}}_t^{[i]} - \hat{z}_t)^T + Q_t$
9 $\Sigma_t^{x,z} = \sum_{i=0}^{2n} w_c (\bar{\mathcal{X}}_t^{[i]} - \bar{\mu}_t)(\bar{\mathcal{Z}}_t^{[i]} - \hat{z}_t)^T$
10 $K_t = \Sigma_t^{x,z} S_t^{-1}$
11 $\mu_t = \bar{\mu}_t + K_t(z_t - \hat{z}_t)$
12 $\Sigma_t = \bar{\Sigma}_t - K_t S_t K_t^T$
13 **return** μ_t, Σ_t

Table 2.5: The unscented Kalman filter algorithm uses a deterministic transformation using samples to linearize the nonlinear transformation functions g and h.

where $s_1, s_2 \in S$ are states containing rotational representations in $SO(3)$. The function $\mathrm{aRot}()$ is the opposite of the $\mathrm{Rot}()$ function mentioned earlier and computes the corresponding $\mathbb{R}^3$ rotational representation from a $SO(3)$ representation. The result of this operation is the difference of two states s_1 and s_2, while their rotational difference is parameterized as a three-dimensional vector.

Both operations integrate almost transparently into the unscented Kalman filter. Only the calculation of the mean out of the propagated sigma points needs to be handled in a special way because adding within S is not possible. Solutions to this problem can be found in [10] and [6].

Since unit quaternions are used for orientation representation throughout this work and angle-axis vectors have been chosen as the locally defined three-dimensional rotation representation, the formulae for both operators can be defined.

The definition of $\oplus$ for the rotational part is

$$q = q_s \cdot \left(\cos(\frac{|r|}{2}), \sin(\frac{|r|}{2}) \frac{r}{|r|} \right) \tag{2.40}$$

where q_s is the quaternion from the state S and r is the three-dimensional axis-angle rotation representation.

The resulting rotational portion of the $\ominus$ operation is

$$\theta = 2\arccos(q_0) \tag{2.41}$$

$$\omega = \frac{[q_1 q_2 q_3]}{\sin(\theta/2)} \tag{2.42}$$

whereas q is the result of a previous quaternion multiplication $q = s_1^{-1} \cdot s_2$ with s_1, s_2 representing the quaternions in the state S. The scale θ and the axis ω are combined to a single vector r by multiplication: $r = \theta \cdot \omega$.

Chapter 3

Sensor Calibration

Combining the data from two sensors requires not only calibration of each sensor itself but also a calibration between them. For the sensors used in this thesis, an inertial sensor coupled to a camera, the calibration goal would be a proper translational and rotational representation of the difference between the coordinate systems of camera and inertial sensor.

Obtaining the translational displacement of both sensors is not a trivial task. Therefore, we simplify the problem by neglecting the translational difference. This is a legitimate option when the physical distance between both sensors is small and the system is not rotating fast.

Two approaches that cope with the problem of finding the rotational displacement should be mentioned. First, in [17] the relation between both coordinate systems is established by moving the coupled sensor and fitting the measured rotational differences of each sensor to a rotational representation. The second approach [18] is static and uses gravity as vertical reference. Here, the inertial sensor measures the gravity using its accelerometers and the camera measures the gravity by performing a camera calibration procedure on a vertically placed checkerboard pattern. The resulting vector pairs are then used to compute the rotational transformation as a closed-form solution.

3.1 Calibration Principle

The calibration technique used here shares the principle with the second approach: gravity will be used as a vertical reference. But instead of separating the procedure into two distinct steps, the camera calibration and the calibration between both coordinate systems is performed simultaneously by fitting the data into a combined model using least squares estimation. Two advantages appear with this approach in contrast to [18]. First,

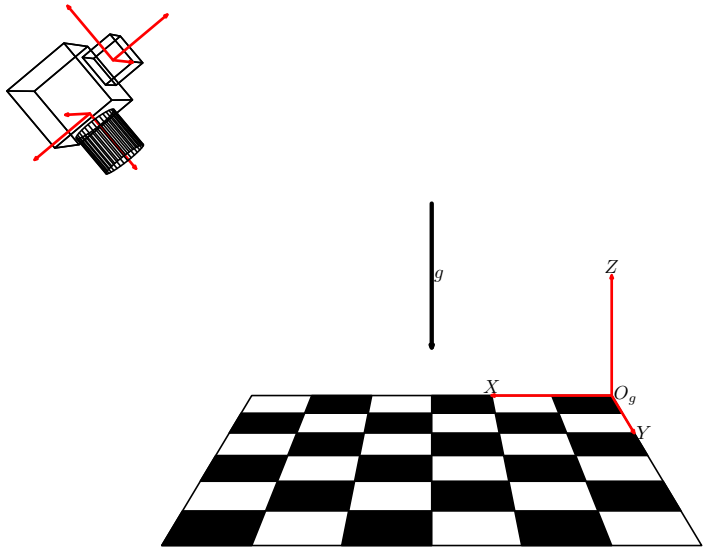

Figure 3.1: Schematic view of a camera observing the checkerboard pattern in its coordinate system (marked as red within the camera) whereas the plane of the checkerboard is aligned orthogonal to gravity g. The feature points on the plane are extracted to determine the camera-centric gravity. The attached inertial sensor measures the gravity g in its coordinate system (marked as red within the inertial sensor). Using these measurements at different poses is sufficient to compute the rotation between both sensors.

the least squares estimation does take the measurement noise into account. This leads to much better estimation results than closed-form solutions provide. Secondly, due to the combination of the camera and the rotational displacement model, not only the measurement noise of the inertial-sensor has impact on the estimation of the rotational difference, but also the uncertainty of the extraction of the camera calibration features. Of course, the estimation of camera parameters during calibration is another welcomed side effect.

The required initial parameters for the estimation are obtained by computing the camera calibration parameters as a closed-form solution. The resulting extrinsic parameters are then used along with the accelerometer measurements to compute the initial rotational difference of both coordinate systems in another quasi-linear closed-form computation.

The calibration principle is visualized in figure 3.1.

3.2 Camera Model

Before using a camera to extract information about the environment, a model representing the optical behaviour of the camera is needed. A common way to model this behaviour is using the geometry of a pinhole camera with radial distortion [19] [20]. Here, a relation between 3D-coordinates of a point in the world and coordinates of its corresponding projection onto the image plane is known as perspective projection.

If we represent the world coordinates space in 3D-coordinates as $X = [X\ Y\ Z]^T$ and the points on the image plane as $m = [u\ v]^T$, this projection can be written as:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = q_r\ (0 \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}^T)^T\ q_r^{-1} + t \tag{3.1}$$

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} / z \tag{3.2}$$

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} f_x \\ f_y \end{pmatrix} \cdot \begin{pmatrix} x' \\ y' \end{pmatrix} + \begin{pmatrix} u_0 \\ v_0 \end{pmatrix} \tag{3.3}$$

The first equation transforms the world point into camera space according to the quaternion q_r and the vector t. These values are known as the extrinsic parameters since they describe the camera pose in the world. Equation (3.2) is the actual perspective projection, where the three-dimensional point is projected onto the image plane as a line that passes the camera center. The parameters in the last equation are the so-called intrinsic parameters and transform points from the camera coordinate system to the image plane. f_x and f_y represent the effective focal distance and u_0 and v_0 the image center for both axes. The geometric relationship between the parameters in the last two equations is depicted in figure 3.2.

The distortion of the camera is taken into account by assuming a rotationally symmetric distortion originating from the optical axis. Each circle with radius $r^2 = x^2 + y^2$ is mapped to a circle with radius r' according to the following equation:

$$r' = 1 + \frac{a_2 r + a_3 r^2}{1 + b_1 r + b_2 r^2 + b_3 r^3} \tag{3.4}$$

In this model, the distortion will be calculated after the transformation to the camera coordinate system but before the transformation to image coordinates, since the effect of distortion is only caused by the lens.

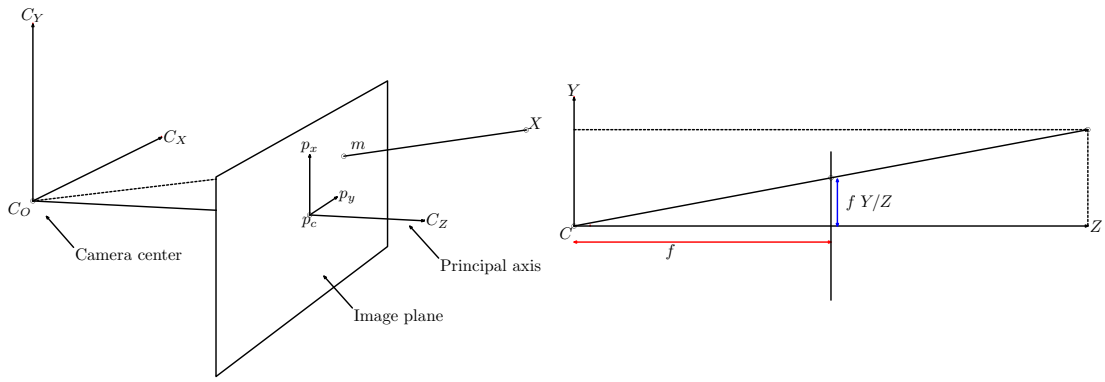

Figure 3.2: Simplified illustration of the pin-hole camera model. (Left) A point X in camera space is projected onto the image plane p at m as a line that passes the camera centre C_O. (Right) The pose on the image plane p is computed by dividing the camera coordinates of the point (here just the Y coordinate) by its depth Z and multiplying it with the focal length f. These illustrations were redrawn from [20].

3.3 Rotation Between the Camera Coordinate System and the Inertial Coordinate System

As mentioned in the beginning of the chapter, we neglect the translational difference and try to minimize the transformation error by reducing the physical distance between camera and inertial sensor. The rotational difference between both coordinate systems is modeled as a unit quaternion.

The transformation from the coordinate system of the camera to the coordinate system of the inertial sensor reads

$$(0\ v_{inert}^T)^T = q\ (0\ v_{cam}^T)^T\ q^{-1} \tag{3.5}$$

where q is the quaternion defining the rotational difference between their coordinate systems, v_{inert} is a three dimensional vector in inertial sensor coordinates and v_{cam} a three dimensional vector in camera coordinates.

3.4 Least Squares Estimation

Assuming that all image points are corrupted by the same but independent noise σ_{vis} and all inertial sensor measurements are corrupted by another constant but indepen-

dent noise σ_{acc}, nonlinear least squares estimation techniques can be used for parameter estimation. The target function Q, which has to be minimized, is defined as a sum of quadratic deviations between the measurements and the values which are predicted by the measurement functions p and t.

$$Q = \sum_i \left(\frac{1}{\sigma_{vis}^2} \sum_j \left(m_{i,j} - p(f_x, f_y, u_0, v_0, k, q_{r,i}, t_i, M_j)\right)^2 + \frac{1}{\sigma_{acc}^2} (g_i - t(q_{r,i}, q, g))^2 \right) \tag{3.6}$$

Here, p is the function that projects the world point M_j of image i onto the image plane according to equations (3.1)-(3.4) and $m_{i,j}$ is the corresponding point on the image plane. Note that the distortion parameters are combined in the vector k.

Function t represents the transformation of the constant gravity vector $g = [0\ 0\ -9.81]^T$ from the world coordinate system to the inertial coordinate system. First, g will be transformed to camera coordinate space using $q_{r,i}$ of image i. After that, the result will be transformed to the inertial sensor space using equation (3.5). g_i is the measured gravity by the inertial sensor belonging to image i.

Due to the iterative nature of nonlinear least squares estimation techniques an initial guess is required. The methods to obtain these values will be described in the following sections.

3.5 Initial Camera Parameters

First, the initial parameters of the camera parameters need to be calculated. The method used here was first presented in [21] and will be applied to the camera model introduced above.

Representing the world and image points as homogenous vectors, e.g. $M = [x\ y\ z\ 1]^T$ for the world points and $m = [x\ y\ 1]^T$ for the image points, the projection can also be written as

$$s\ m = A\ [R\ t]\ M \tag{3.7}$$

where s is an arbitrary scale factor, R a rotation matrix, t are the extrinsic parameters defining the camera coordinate system in world coordinates and

$$A = \begin{pmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3.8}$$

contains the intrinsic parameters.

By observing that all the world points lie on a plane with $Z = 0$, equation (3.7) reads

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = A \begin{bmatrix} r_1 & r_2 & t \end{bmatrix} \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix} \tag{3.9}$$

The term $A[r1\ r2\ t]$ can also be expressed as a homography H (see [20]) between the image plane and the plane in the world leading to

$$H = \lambda A \begin{bmatrix} r_1 & r_2 & t \end{bmatrix} \tag{3.10}$$

where λ is a scale factor.

By exploiting the fact that r_1 and r_2 are orthonormal, two constraints out of equation (3.9) can be deduced for each image:

$$h_1^T A^{-T} A^{-1} h_2 = 0 \tag{3.11}$$

$$h_1^T A^{-T} A^{-1} h_1 = h_2^T A^{-T} A^{-1} h_2 \tag{3.12}$$

$$\tag{3.13}$$

Denoting $A^{-T}A^{-1}$ as the symmetric matrix B leads to

$$B = A^{-T}A^{-1} = \begin{pmatrix} f_x^{-2} & 0 & -(f_x^2 u_0)^{-1} \\ 0 & f_y^{-2} & -(f_y^2 v_0)^{-1} \\ -(f_x^2 u_0)^{-1} & -(f_y^2 v_0)^{-1} & \frac{u_0^2}{f_x^2} + \frac{v_0^2}{f_y^2} + 1 \end{pmatrix} \tag{3.14}$$

for our camera model. Knowing that $h_i^T\ B\ h_j = v_{ij}^T b$, where $b = [B_{11}B_{12}B_{22}B_{13}B_{23}B_{33}]^T$ and $v_{ij} = [h_{i1}h_{j1}, h_{i1}h_{j2} + h_{i2}h_{j1}, h_{i2}h_{j2}h_{i3}h_{j1} + h_{i1}h_{j3}, h_{i3}h_{j2} + h_{i2}h_{j3}, h_{i3}h_{j3}]^T$,allows to form two equations for each image of the world plane:

$$\begin{pmatrix} v_{12}^T \\ (v_{11} - v_{22})^T \end{pmatrix} b = 0 \tag{3.15}$$

By stacking up $2n$ equations of all n images, the solution for b is the right singular vector of the stacked up equation matrix belonging to the smallest singular value.

With the knowledge of b, it is now possible to compute the intrinsic parameters according to equation (3.14):

$$v_0 = -\frac{b_{2,3}}{b_{2,2}} \tag{3.16}$$

$$u_0 = -\frac{b_{1,3}}{b_{1,1}} \tag{3.17}$$

$$\lambda = b_{1,3}\, u_0 + b_{2,3}\, v_0 + b_{3,3} \tag{3.18}$$

$$f_x = \frac{\sqrt{b_{1,1}\lambda}}{b_{1,1}} \tag{3.19}$$

$$f_y = \frac{\sqrt{b_{2,2}\lambda}}{b_{2,2}} \tag{3.20}$$

$$\tag{3.21}$$

Assembling the intrinsic matrix A out of these values makes it possible to deduce the extrinsic parameters from (3.10):

$$r_1 = \lambda A^{-1} h_1 \tag{3.22}$$

$$r_2 = \lambda A^{-1} h_2 \tag{3.23}$$

$$r_3 = r_1 \times r_2 \tag{3.24}$$

$$t = \lambda A^{-1} h_3 \tag{3.25}$$

$$\tag{3.26}$$

with $\lambda = \frac{1}{|A_{-1} h_1|}$.
$Q = [r_1 r_2 r_3]$ might not necessarily be a valid orthogonal matrix due to measurement noise. One way to compute the best rotation matrix R for Q is to find the smallest Frobenius norm of their difference. Decomposing Q to $U\Sigma V^T$ using singular value decomposition and multiplying

$$R = UV^T \tag{3.27}$$

leads to the desired result. Finally, R is converted to a quaternion q_r according to [22].

In contrast to [21], no initial guesses will be computed for the distortion parameters. In practice, setting the distortion parameters to zero will be sufficient for the least squares method to obtain the actual distortion coefficients.

3.6 Initial Rotation Between Camera and Inertial Sensor

Out of the extrinsic parameters of each observed plane, the quaternion which minimizes the error between the gravity in camera coordinates and the measured gravity of the sensor can be computed using the method provided by [23].

First, the constant gravity g is transformed into camera coordinate space using the extrinsic parameter q_r for every image.

After that, the problem of finding the quaternion, which rotates from camera to inertial space by minimizing the error between inertial measurements v_{acc} and the rotated gravities from camera space v_{vis} to inertial space, can be written as finding the maximum of the sum of their products

$$\max \sum_i (q\ v_{vis,i}\ q^{-1}) \cdot v_{acc,i} \tag{3.28}$$

in which the vectors are augmented by a preceding 0. This term can also be written as:

$$\max \sum_i (q\ v_{vis,i}) \cdot (v_{acc,i}\ q) \tag{3.29}$$

By writing the quaternion product as a matrix-vector multiplication

$$q\ v = \begin{pmatrix} 0 & -v_x & -v_y & -v_z \\ v_x & 0 & v_z & -v_y \\ v_y & -v_z & 0 & v_x \\ v_z & v_y & -v_x & 0 \end{pmatrix} q = Vq \tag{3.30}$$

$$v\ q = \begin{pmatrix} 0 & -v_x & -v_y & -v_z \\ v_x & 0 & -v_z & v_y \\ v_y & v_z & 0 & -v_x \\ v_z & -v_y & v_x & 0 \end{pmatrix} q = Vq \tag{3.31}$$

the sum can be written as:

$$\max\ q^T \left(\sum_i V_{vis,i} V_{acc,i} \right)\ q\ . \tag{3.32}$$

The unit quaternion that maximizes this expression is known to be the eigenvector corresponding to the largest positive eigenvalue of the sum of $V_{vis}\ V_{acc}$ multiplications.

A better way to compute this sum is possible by exploiting certain properties of the matrices and is given in [23].

3.7 Calibration Procedure

The calibration procedure that was used for calibration of the used sensor can be described in five distinct steps:

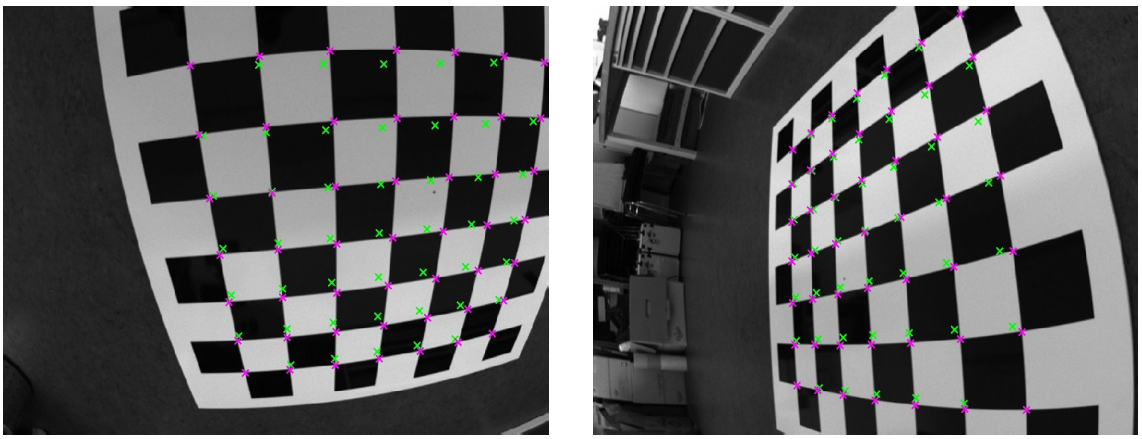

Figure 3.3: Two example images of the camera directed to the checkerboard pattern with annotations of the feature point extraction. The green crosses show an estimated projection established from a homography which was estimated out of the pattern's four corner points. These points were fed to the feature point detector algorithm, resulting in the set of purple crosses which were used for the calibration. This procedure was inspired by [25].

1. Multiple images of a checkerboard pattern lying orthogonal to gravity were taken (see figure 3.3) and the accelerometer measurements along with the corresponding image files were saved.

2. The feature points were extracted using [24], and a homography between all feature points on the image plane and the world plane was estimated for every image.

3. These homographies were then used to determine the initital camera parameters. In contrast to the described camera model above, the model was slightly simplified. Only one parameter for the focal length of both axes is estimated. This simplification is based on the fact that the focal length for the used camera in this procedure should have the same value for both axes.

4. Once the extrinsic parameters were obtained, the initial quaternion representing the rotation between camera space and inertial sensor space was computed.

5. After obtaining all initial parameters, a nonlinear least squares estimation using the Levenberg-Marquardt algorithm was performed to refine the results.

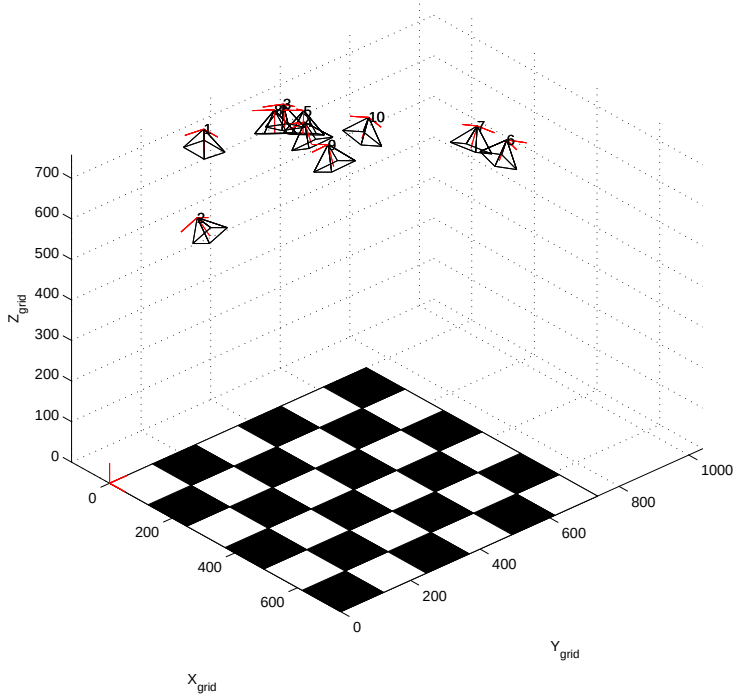

Figure 3.4: Three dimensional view of the extrinsic parameter results obtained through the calibration procedure. The ten different poses of the camera, which were used for the calibration, are shown in the coordinate system of the checkerboard grid originating at a corner of the pattern. The unit of the axes is milimeters.

3.8 Results

The calibration procedure has been successfully applied to the used camera-inertial sensor. The results of the extrinsic parameters are shown in figure 3.4 to give an impression how the system was positioned while the calibration data were obtained.

Since the actual values of the parameters depend on the chosen sensor and calibration setup (and therefore will not be mentioned here), the calibration perfomance can be measured by evaluating the root mean square. The root mean square of the residuals for both the projection of the feature points and the rotation of the gravity into inertial sensor coordinates are shown in table 3.8.

As expected, the least squares estimation improves the initial estimation considerably. While the vision results speak for themselves, the root mean square of the rotation corresponds to a mean rotational difference of $0.26°$ between measurements of gravity by the

RMS	Vision [pixel]	Rotation $[m/s^2]$
initial estimate	15.8843 17.0174	1.2326 1.0543 0.1708
least squares estimate	0.2128 0.2119	0.0307 0.0315 0.0386

Table 3.1: The root mean square values before and after the least squares refinement.

inertial sensor and the earth's gravity rotated into the coordinate system of the inertial sensor using the estimated parameters.

Chapter 4

Modeling a Ball in Flight

This chapter describes the fundamental models that are used to estimate the state within the dynamic system of a ball in flight which is observed by a free moving camera-inertial sensor.

Starting with a description of the to-be-estimated state, this chapter will explain the underlying dynamic and measurement models and their representation in both estimators. Futhermore, a brief description of the noise parameters within the model will be given and an algorithm for computing these parameters from sensor data will be presented. Finally, the variables and parameters of the model and the differences in the models for the unscented Kalman filter and the least squares estimation will be summarized.

4.1 System State

The state of a flying ball observed by a free moving camera-inertial sensor is modeled as a combination of ball and sensor state. With respect to a global coordinate system, the ball state is given by position and velocity

$$\vec{x}_{ball} = (x_b, y_b, z_b)^T \tag{4.1}$$
$$\vec{v}_{ball} = (vx_b, vy_b, vz_b)^T \tag{4.2}$$

while the sensor state is given by

$$\vec{x}_{sensor} = (x_s, y_s, z_s)^T \tag{4.3}$$
$$\vec{v}_{sensor} = (vx_s, vy_s, vz_s)^T \tag{4.4}$$
$$q_{sensor} = (q_0, q_1, q_2, q_3)^T \tag{4.5}$$
$$\tag{4.6}$$

denoting position, velocity and orientation of the sensor. The orientation is encoded as a unit quaternion which avoids singularities. This makes a total number of 16 components in the state, from which four are not in Euclidean space. However, using the techniques from section 2.1.4 and 2.2.4, small changes can be applied to the state by $\oplus$ with d, in which d is a $\mathbb{R}^{15}$ (the orientation is represented in $\mathbb{R}^3$) vector. In this way, the problems of estimating orientations are hidden from the estimation algorithms.

4.2 Underlying Dynamic Model

The dynamic model describes the change of state in a system over time. Here, the used state contains two parts, the ball's state and the sensor's state.

The dynamics of the ball state are described by classical mechanics along with the consideration of air drag, which influences the ball in flight. This rather precise model is necessary since the quality of tracking and predicting a flying ball relies on how well the model reflects the actual properties of the flight.

The dynamics of the ball in flight can be expressed as two first order differential equations [7]:

$$\dot{\vec{v}} = \vec{g} - \alpha \, |\vec{v}| \, \vec{v} \tag{4.7}$$

$$\dot{\vec{x}} = \vec{v} \tag{4.8}$$

with g representing the gravity. α represents the effect of air drag and is composed of $\alpha = \frac{c_d A \rho}{2m}$, where c_d is the characteristic drag of the object moving in a fluid (also known as the drag coefficient), A is the cross-sectional area of the moving object, ρ is the density of the fluid in which object is moving, and m is the object's mass. From [26], the actual values for a flying soccer ball are $A = 0.039\ m^2$ for the cross-sectional area, $\rho = 1.2\ kg\ m^{-3}$ for the density of air, $c_d = 0.2$ for the drag coefficient of a sphere and $m = 0.43\ kg$ resulting in $\alpha = 0.011\ m^{-1}$.

Table 4.1 shows how important it is to incorporate air drag into the dynamic model of the ball. Here, flights at different initital velocities with and without consideration of air drag are compared.

Initial velocity $[m/s]$	5	7	10	15	20	30	40
Length in $[m]$ with $\alpha = 0\ m^{-1}$	2.51	4.95	10.12	22.90	40.70	91.67	163.04
Length in $[m]$ with $\alpha = 0.011\ m^{-1}$	2.45	4.78	9.32	19.22	30.59	53.85	74.65

Table 4.1: Comparision of the length of ball flight trajectories with different starting velocities. The initial release angle is $45°$.

At lower ball speeds, the air drag might be negligible, but the higher the speed the more significant the drag becomes. At an initial velocity of $40\ m/s$, which is comparable to the initial velocity of a common goal kick, the lengths of the flights differ by more than the double. Therefore, state estimations and predictions of fast flying balls without this phenomenon in mind would become excessively incorrect. This makes the incorporation of air drag a vital requirement for the precision of model and prediction.

It should not be left unmentioned that the model might be extended by integrating the Magnus effect [27], which describes the interaction of a spinning ball with the air, resulting in a curved flight. Practically, this effect would be hard to integrate since measuring the spin of the ball with a simple sensor setup would be very inaccurate and additional dynamic equations modeling this spin would be required, which would make the model as a whole quite complex.

Unlike the ball, the dynamics of the sensor are not influenced by the environment. The dynamics are again obeying classical mechanics, which are coupled to the measurements of the inertial sensor. The only trick here is to rotate the accelerometer measurements into world coordinates before applying them since all inertial measurements are made in local coordinates.

According to [28], the change of the orientation forward in time is expressed as the mapping of the angular velocity into the tangent space of the used parameterization. For a quaternion parameterization, the mapping is expressed as

$$\dot{q} = \frac{1}{2}(0,\ \omega^T)^T \cdot q \tag{4.9}$$

where q is the quaternion and ω represents the angular velocity measured by the inertial sensor.

With the updated orientation, the accelerometer values of the sensor are used to update velocity and position. First, the above mentioned rotation into world coordinates is performed. After that, the gravity of earth is subtracted from the measured acceleration, resulting in the geometric acceleration of the moving sensor.

Therefore, the differential equations expressing the relation between velocity, position and measured acceleration read

$$\dot{\vec{v}} = \vec{g} + q \cdot (0, \vec{a}^T)^T \cdot q^{-1} \tag{4.10}$$

$$\dot{\vec{x}} = \vec{v} \tag{4.11}$$

where $\vec{x}$ is the position, $\vec{v}$ is the velocity, $\vec{a}$ is the measured acceleration and $\vec{g}$ is the gravity vector.

4.2.1 Integration Into the Unscented Kalman Filter

Nonlinear Kalman filters require the dynamic model to be a function, which is usually denoted as g. This function serves as the state transition function and predicts the actual state at time t from the previous state at time $t-1$.

This prediction is performed by integrating the differential equations over time. Assuming constant measurements during a certain time interval δt, a simple although not too accurate but computationally efficient way to compute this integration is Euler's method. Here, the solution for a first order differential equation is approximated by a first order Taylor expansion.

Using Euler's method, the state transition from time $t-1$ to t for the ball reads

$$\vec{v}_t = \vec{v}_{t-1} + (\vec{g} - \alpha \, |\vec{v}_{t-1}| \, \vec{v}_{t-1})\delta t \tag{4.12}$$

$$\vec{x}_t = \vec{x}_{t-1} + \vec{v}_{t-1}\delta t \tag{4.13}$$

where $\vec{x}$ is the position and $\vec{v}$ is the velocity. Their subscripted indices denote the timestep whereas δt is the elapsed time between both timesteps.

The equations for the state transition of the sensor based on the differential equations read:

$$q_t = q_{t-1} \cdot \mathrm{Rot}(\omega \cdot \delta t) \tag{4.14}$$

$$\vec{v}_t = \vec{v}_{t-1} + (\vec{g} + q_{t-1} \cdot (0, \, \vec{a}_t^{\,T})^T \cdot q_{t-1}^{-1})\delta t \tag{4.15}$$

$$\vec{x}_t = \vec{x}_{t-1} + \vec{v}_{t-1}\delta t \tag{4.16}$$

The transition to $\vec{v}_t$ and $\vec{x}_t$ follows right out of the differential equations. The calculation of the quaternion at time t is done by a quaternion multiplication. The first factor is the quaternion q_{t-1} from the previous state. The second factor is the quaternion resulting from applying $\mathrm{Rot}()$ to the multiplication result of the measured angular velocity ω and the temporal difference δt between both states. The function $\mathrm{Rot}()$ is known from section 2.2.4. The solution for this integration was taken from [17].

4.2.2 Integration Into the Least Squares Method

Based on the function g of the unscented Kalman filter, the functions for the least squares method are obtained by rearranging the integrated function equations to obtain the inverse state transition function. For the ball, the virtual measurement function h_{dynB} reads:

$$\vec{v}_{t-1} = \frac{-v_t + g \; \delta t}{-1 + \frac{\frac{1}{2} - \frac{1}{2}\sqrt{1 - 4 \; \alpha \; \delta t |-\vec{v}_t + g \; \delta t|}}{\delta t}} \tag{4.17}$$

$$\vec{x}_{t-1} = \vec{x}_t - \vec{v}_{t-1}\delta t \tag{4.18}$$

The first equation looks quite complicated in contrast to its counterpart in the forward dynamic function of the Kalman filter. Because norm and vector of the velocity exist within the differential equation 4.7, rearranging is not trivial and makes the equation look deformed.

The semi-virtual measurement function for the sensor dynamics h_{dynS} can be expressed as

$$\vec{x}_{t-1} = \vec{x_t} - (\vec{v}_t - \vec{v}_{t-1})\delta t \tag{4.19}$$

$$\vec{a}_t = q_{t-1}^{-1} \cdot [0, (\vec{v}_t - \vec{v}_{t-1})^T \delta t - g^T]^T \cdot q_{t-1} \tag{4.20}$$

$$\omega = \frac{\mathrm{aRot}(q_{t-1}^{-1} \cdot q_t)}{\delta t} \tag{4.21}$$

where $\vec{x}_{t-1}$ is the position of the sensor at time $t-1$ and a and ω are the predicted acceleration and angular velocity at time t. The predicted angular velocity is computed using the function aRot, which calculates the the three dimensional representation of the rotational difference of the quaternions q_{t-1} and q_t. Dividing the rotational difference by δt results in the predicted angular velocity between both states at $t-1$ and t.

In contrast to the virtual measurement function for the ball dynamics, this function is not completely virtual since measurements are involved. These measurements are not sufficient since each measurement of the inertial sensor provides six degrees of freedom, but the dynamics are described by nine parameters. The missing three degrees of freedom are gained by computing the predicted position of the sensor out of the state variables at time $t-1$ and t, which makes the function semi-virtual.

4.3 Measurement Model

For the purpose of estimating the state of a flying ball using a camera-inertial sensor, model functions for the non-dynamical measurements done by the camera need to be defined. First, the measurement of the ball within the camera image has to be included into the estimation process. Furthermore, one might want to stabilize the sensor pose estimation by evaluating known points of the environment within the image. This is necessary since the error in position from the inertial sensor accumulates over time.

These functions are known as measurement functions and are the same for both estimators. The functions are based on the projective geometry-based camera model, which was introduced in section 3.2, and were taken from the work of [19]. Here, the measurement function for the ball calculates the center point and diameter of the ball's circle in the image from the current system state and the always known ball diameter. This function is denoted as h_{measB}. The second measurement function, the relation between the known landmark points in the environment and their corresponding position in the image, is

modeled by a transformation of three-dimensional world coordinates into image coordinates using the equations given in the projective camera model. This function is denoted as h_{measS}.

Within this work, the orientation of the inertial sensor with respect to the field is used as the orientation in the state. To incorporate the measurement functions of the camera correctly, the orientation needs to be transformed into the camera coordinate system. Here, the estimated rotational difference between both sensors estimated from the calibration procedure comes into play. The orientation in the camera coordinate system is computed by applying this rotational difference to the inertial sensor orientation.

4.4 Noise

So far, the state transition and measurement process are handled deterministically as the functions $x_t = g(x_{t-1})$ and $z_t = h(x_t)$ respectively. In order to get the above model functions to work within both estimators, it is necessary to expand this deterministic approach.

From equations (2.1), (2.28) and (2.29) it is known that the state transition and the measurements need to be modeled with noise in mind. For the unscented Kalman filter, the noise of the system is given by three covariance matrices R, Q_{measS} and Q_{measB}. R is known as the process noise, and its diagonal contains the squared standard deviations of the uncertainty in the state transition (given in the corresponding units of the state). The other covariance matrices, Q_{measS} and Q_{measB}, represent the uncertainty of their corresponding measurement functions in each timestep. The diagonal contains the squared standard deviations of the assumed error in pixels.

Similar to Kalman filters, the uncertainty in the least squares method is represented by a set of covariance matrices. Here, every measurement function h_n has a corresponding covariance matrix Σ_n. Since four measurement functions exist in the model to describe the system, the four covariance matrices Σ_{dynS}, Σ_{dynB}, Σ_{measS} and Σ_{measB} need to be defined within the least squares estimation.

It is clear that if one wants to compare both estimators, the same values must be used in the various covariance matrices of both estimators. In the modeled system here, all but two classes of covariances have exactly the same values. The difference between the two variance classes is due to the difference of the dynamic model functions between Kalman filter and least squares estimator. The Kalman filter models the sensor dynamics as a state transition function incorporating the inertial measurement z_t into the state x_{t-1}. Therefore, the uncertainty in this process is expressed as a covariance using the units of the state (m/s and rad). In contrast to the Kalman filter, the dynamic functions of the least squares estimator are inverse state transition functions. The predicted measurement z_t is

computed out of the states x_t and x_{t-1}. Therefore, the uncertainty is expressed in units of the sensor measurements (m/s^2 and rad/s). Conversion of these noise parameters into parameters of the Kalman filter is done by multiplication with δt.

4.5 Model Parameter Determination

In the preceding sections, various parameters within the model were introduced. These parameters include the intrinsic parameters of the camera, the rotational displacement of camera and inertial sensor, the degree of air drag influencing the ball in flight and the noise of the dynamic and measurement process. For some of those parameters, the computation of the actual values were already shown along with their introduction.

It is known that the camera parameters and the rotational difference may be computed from the calibration procedure. Furthermore, the value determining the air drag of a flying ball was mentioned in the introduction of the ball dynamics. The only parameters left unmentioned are the various noise parameters within the model.

Actually, a-priori knowledge may be used to set some of the noise parameteres to reasonable values. First, the noise parameters influencing the dynamics of ball and sensor may be held constant. These values are actually known quite well prior to the evaluation. Under the assumption that the whole dynamics error is only caused by the error in the velocity, the translational error may be set to a very small value (e.g. $(1\ mm)^2$ for the ball and $(2\ cm)^2$ for the sensor). Zero would be the optimal value for the ball but the assignment of zero to the covariances causes numerical problems. The value for the error in velocity in each timestep models disturbances of the assumed physical model. In this evaluation, the covariance of the velocity ranged from $(1\ cm/s)^2$ for rather short to $(2.5\ cm/s)^2$ for longer flights (to model the effects of wind and ball spin). The noise parameters regarding the measurements of the inertial sensor can be computed from observing and analyzing the noise of a resting sensor.

How to obtain suitable values for the remaining parameters of the measurement uncertainty will be addressed in the upcoming section by introducing a parameter learning algorithm.

4.5.1 Algorithm

Having obtained a dataset by observing the flight of a ball in a known environment, the noise parameters of the model can be calculated from the likelihood of the probability density distribution

$$p(Z \mid X, \theta) \tag{4.22}$$

LEARN MEASUREMENT NOISE PARAMETERS$(Z, \Sigma_{measB}, \Sigma_{measS})$

1 $\Sigma_{dynS} \leftarrow$ set manually to a reasonable value
2 $\Sigma_{dynB} \leftarrow$ set manually to a reasonable value
3 **while** not converged
4 **do**
5 compute optimal X out of Z and Σ's
6 $Z_{meas} = Z_{measB} \cup Z_{measS}$
7 $\sigma_{meas} \leftarrow \sqrt{\frac{1}{|Z_{meas}|} \sum_i (Z_{meas,i} - h_{meas}(X))}$
8 $\Sigma_{measB} \leftarrow \sigma^2_{meas}$
9 $\Sigma_{measS} \leftarrow \sigma^2_{meas}$
10 **return** $\Sigma_{measB}, \Sigma_{measS}$

Table 4.2: Iterative algorithm for learning the measurement noise parameters out of the measurement data.

where Z are the obtained measurements, X are the states of the system associated with the measurements and θ denotes the set of unknown noise parameters. By maximizing this likelihood with respect to θ, the answer to the problem of finding suitable values for the parameters can be given. The algorithm that performs this maximization for the model used here computes the optimal values of the ball measurement uncertainty Σ_{measB} and the landmark measurement uncertainty Σ_{measS} using their measurements Z_{measB} and Z_{measS} out of Z and is presented in table 4.2.

The algorithm executes the following three steps until the convergence criterion is reached.

1. Compute the optimal X for the measurements Z using all necessary covariances (line 5). The optimal X may be found using the method of least squares.

2. Unify the measurement classes Z_{measB} and Z_{measS} into one class Z_{meas} (line 6). Compute the root mean square of the residuals, resulting from the subtraction of $h_{meas}(X)$ from Z_{meas}, where h_{meas} is either the measurement function of the ball or the landmark (line 7). The result of this operation is a scalar.

3. Set the diagonals of the covariance matrices of the ball and landmark measurement to the obtained value. (lines 8-9).

After the convergence criterion is reached, the resulting values for Σ_{measB} and Σ_{measS} are the covariances that maximize the probability density function from the equation 4.22.

Please note that the computed scalar is assigned to all diagonal components of both covariance matrices. Why this has to be the case will be explained in the derivation below.

4.5.2 Mathematical Derivation

This section provides a mathematical derivation of the algorithm. As it is known from above, the problem is to maximize the probability density function 4.22 with respect to the model parameters θ.

The iterative structure of the algorithm is based on the Expectation-Maximization algorithm [29]. It is used for computing the maximum likelihood estimates of the parameters X in probabilistic models which depend on unknown variables (Σ_{measB} and Σ_{measS} in this case). The maximization is performed by computing an expected value from the measurements and an initial guess of the variables (this computation is found in line 5). The maximum likelihood of the parameters is then computed by adjusting θ in such a way that the expected likelihood becomes a maximum (performed in line 7). By assigning the values found in the last step as the new variable estimates, the overall maximum likelihood is found by iteration.

For the calculation of the parameters that maximize the likelihood, the split-up of the whole problem into several sub problems should be considered first. The measurements Z and the optimal states X are being held fixed, so the problem reduces from the maximization of one complex function to the maximization of the individual probability density function of each measurement class:

$$p(Z \mid X, \theta) = \prod_i p(Z_i \mid X,\ \theta_i) \tag{4.23}$$

Here, Z_i contains the measurements of measurement class i (e.g. ball measurement through computer vision) and θ_i are the noise parameters belonging to this class.

Denoting a set of measurements belonging to a certain measurement class as Z and representing the uncertainty in the measurement as σ, the to be maximized function becomes

$$p(Z \mid x, \sigma) = \prod_i p(z_i \mid x,\ \sigma) \tag{4.24}$$

$$= \prod_i \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{(z_i - f(x))^2}{2\sigma^2}) \tag{4.25}$$

where f is the function computing the expected measurement from the optimal state X.

By computing the positive logarithm, which preserves the maximum of the function, the

function then reads

$$\log p(Z \mid x,\ \sigma) = \sum_i \left(-\frac{1}{2} \log 2\pi\sigma^2 - \frac{(z_i - f(x))^2}{2\sigma^2} \right) \tag{4.26}$$

$$= -\frac{n}{2} \log 2\pi\sigma^2 - \sum_i \left(\frac{(z_i - f(x))^2}{2\sigma^2} \right) \tag{4.27}$$

$$= -\frac{n \log 2\pi}{2} - n \log \sigma - \frac{1}{2\sigma^2} \sum_i (z_i - f(x))^2 \tag{4.28}$$

from which the first derivative with respect to σ is:

$$\frac{d}{d\sigma} \log p(Z \mid x,\ \sigma) = -\frac{n}{\sigma} + \frac{1}{\sigma^3} \sum_i (z_i - f(x))^2 \tag{4.29}$$

The maximum of the function is then obtained by setting the first derivative to zero and solving the resulting equation for σ:

$$\sigma = \sqrt{\frac{1}{n} \sum_i (z_i - f(x))^2} \tag{4.30}$$

Therefore, given a set of measurements and the corresponding optimal state of the system, the maximum of the initial probability density distribution is found by computing the σ of the individual measurement classes. But only one single σ is computed for the covariances of different measurement functions within the algorithm. To explain this fact, a simple example involving two measurements will be given, which shows interesting properties regarding the computation of different covariances at once.

4.5.3 Counter Example

This counter example will show why the estimation of two covariances at once will give paradoxical results.

Considering two measurements of a state X denoted as $z_1 = 0$ and $z_2 = 1$ with the covariances σ_1^2 and σ_2^2, what does the probability density function

$$p(z_1 = 0, z_2 = 1 \mid X, \sigma_1, \sigma_2) \tag{4.31}$$

look like with respect to σ_1 and σ_2 with X optimal?

To answer this, the optimal X is first computed from the function by applying Bayes' rule and omitting the denominator and second factor:

$$p(X \mid z_1 = 0, z_2 = 1, \sigma_1, \sigma_2) \propto p(z_1 = 0, z_2 = 1 \mid X, \sigma_1, \sigma_2) \tag{4.32}$$

$$\tag{4.33}$$

By assuming a Gaussian distribution and independence between all measurements, the function may be written as the product of two Gaussian distributions.

$$p(z_1 = 0, z_2 = 1 \mid x, \sigma_1, \sigma_2) \tag{4.34}$$
$$= \; p(z_1 = 0 \mid x, \sigma_1, \sigma_2)\, p(z_1 = 1 \mid x, \sigma_1, \sigma_2) \tag{4.35}$$
$$= \; \frac{1}{\sqrt{2\pi\sigma_1^2}} \exp\left(-\frac{1}{2}\frac{(0-x)^2}{\sigma_1^2}\right) \cdot \frac{1}{\sqrt{2\pi\sigma_2^2}} \exp\left(-\frac{1}{2}\frac{(1-x)^2}{\sigma_2^2}\right) \tag{4.36}$$

Again, this can be solved by calculating the logarithm of the function:

$$\log p(z_1 = 0, z_2 = 1 \mid x, \sigma_1, \sigma_2) \tag{4.37}$$
$$= \; -\log\sigma_1^2 - \frac{x^2}{\sigma_1^2} - \log\sigma_2^2 - \frac{(1-x)^2}{\sigma_2^2} \tag{4.38}$$

Since the goal is to find the maximum likelihood of the function, computing the first derivative

$$\frac{d}{dx}\log p(z_1 = 0, z_2 = 1 \mid x, \sigma_1, \sigma_2) \;=\; -\frac{x}{\sigma_1^2} + \frac{(1-x)}{\sigma_2^2} \tag{4.39}$$

and setting this derivate to 0 leads to the desired optimal state:

$$x = \frac{\sigma_1^2}{\sigma_1^2 + \sigma_2^2} \tag{4.40}$$

From here, calculating the distribution of the two measurements using the newly obtained optimal state is obvious:

$$p(z_1 = 0, z_2 = 1 \mid \sigma_1, \sigma_2) \tag{4.41}$$
$$= \; \frac{1}{\sqrt{2\pi\sigma_1^2}} \exp(-\frac{1}{2}\frac{(0-\frac{\sigma_1^2}{\sigma_1^2+\sigma_2^2})^2}{\sigma_1^2}) \frac{1}{\sqrt{2\pi\sigma_2^2}} \exp(-\frac{1}{2}\frac{(1-\frac{\sigma_1^2}{\sigma_1^2+\sigma_2^2})^2}{\sigma_2^2}) \tag{4.42}$$

Unlike the previous two occasions, the negative logarithm is computed (because of cosmetic purposes for the visualization) . This will inverse the outcome of the function values and therefore the minimum of the resulting function will be its maximum likelihood. Please note that the preceding factor is just for the elimination of the $\frac{1}{2}$.

$$-2\log p(z_1 = 0, z_2 = 1 \mid x, \sigma_1, \sigma_2) \tag{4.43}$$
$$= \; \text{const.} + \log(\sigma_1^2) + \log(\sigma_2^2) + \frac{\frac{\sigma_1^2}{\sigma_1^2+\sigma_2^2}}{\sigma_1^2} + \frac{1-\frac{\sigma_1^2}{\sigma_1^2+\sigma_2^2}}{\sigma_2^2} \tag{4.44}$$

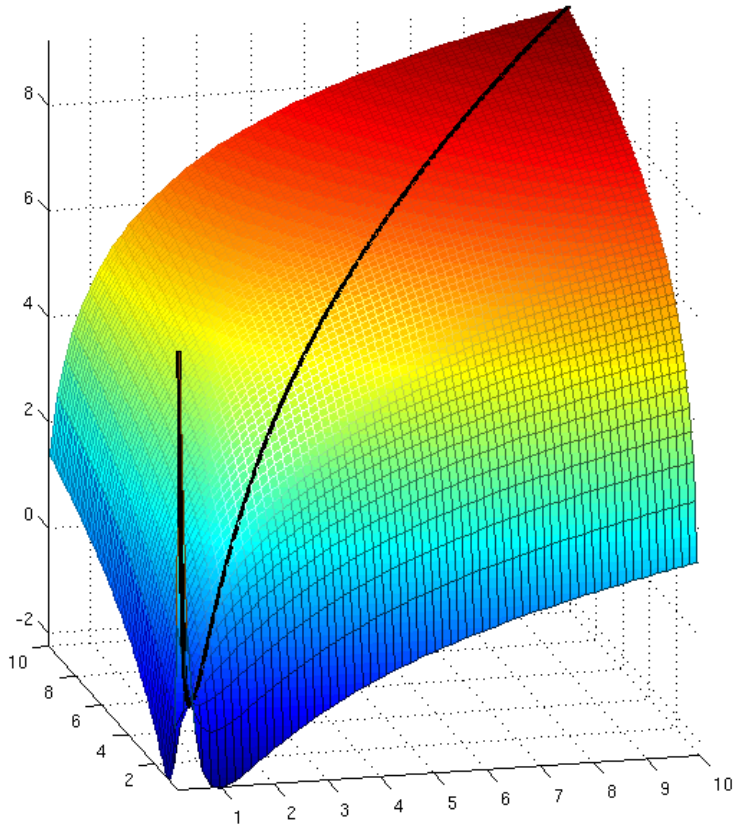

Figure 4.1: Surface plot of the likelihood function with respect to σ_1 and σ_2. The black line illustrates the function value for $\sigma_1 = \sigma_2$.

By expanding and reducing the latter fractions and ignoring the preceding constant term, the function finally becomes:

$$-2\log p(z_1 = 0, z_2 = 0 \mid x, \sigma_1, \sigma_2) = \log(\sigma_1^2) + \log(\sigma_2^2) + \frac{1}{\sigma_1^2 + \sigma_2^2} \tag{4.45}$$

By looking at the function (a surface plot is shown in figure 4.1), it can be seen that there is no minimum but only a saddle point at $\sigma_1 = 0.5$ and $\sigma_2 = 0.5$.

Besides this, the problem that may occur for the algorithm is that if one covariance is approaching zero during the iteration the other covariance may have a significantly larger value, whereas the function value still becomes minimal. It seems that putting all the certainty only into one measurement while ignoring the certainty of the other measurement

is the maximization of the likelihood.

Of course, within this two measurement example, putting all certainty into one measurement while ignoring the other does not make sense. Practically, this makes the problem of finding the maximum likelihood, in which the covariances still make sense, difficult. Explaining this behavior in detail and its impact to any mixed Gaussian is not part of this thesis, but since all measurements that are obtained from the sensors should be used for the estimation purposes of this thesis, a way to handle this problem is required.

A simple solution preventing the algorithm of finding its way into this area is achiveved by combining all single covariances and expressing them as one covariance. For the two-measurement example above, combining the covariances into one ($\sigma_1 = \sigma_2$) leads to the black line in figure 4.1. As can be seen, this constraint solves the problem of misdirection within this example and the minimum of the function is $\sigma_1 = \sigma_2 = 0.5$.

The behavior of this example might easily be expanded to any other function involving several covariances. Therefore, the uncertainties of the ball and landmark measurements are expressed as a single covariance to prevent the computation of wrong values in the algorithm. Actually, finding the real maximum likelihood of mixed Gaussian distributions with respect to real measurement data will fail due to this constraint. Nevertheless, the likelihood found by the algorithm above should reflect the maximum with respect to the constraint of the unified covariances.

4.6 Summary

After describing what is necessary to model a ball in flight which is observed by a camera-inertial sensor, the involved variables and parameters neeeded for the estimation are presented in table 4.3. It can be seen that all model parameters are obtained prior to the estimation process. Of course, the measurement data are gained from the sensors during the experiments in which the flight of a ball is observed. The state is computed during the actual estimation process.

Model variable		Obtained from
Measurements	Inertial: z_{dynS}	Sensors during experiments
	Vision landmark: z_{measS}	
	Vision ball: z_{measB}	
State x	Ball: $\vec{x}_{ball}, \vec{v}_{ball}$	Estimation at runtime
	Sensor: $\vec{x}_{sensor}, \vec{v}_{sensor}, q_{sensor}$	
Noise	Dynamic: Σ_{dynB} and part of Σ_{dynS}	Manual guess
	Inertial measurement: part of Σ_{dynS}	Observation of a resting sensor
	Vision measurement: Σ_{measB} and Σ_{measS}	Measurement noise determination algorithm
Further parameters	Camera parameters and rotation between sensors (used by h_{measB} and h_{measS})	Sensor calibration procedure
	Air drag α	Manual computation

Table 4.3: Tabulated summary of all parameters and variables involved in the process of estimating the state of flying ball observed by a camera-inertial sensor.

To clarify the differences between the necessary model functions of the unscented Kalman filter and the method of least squares, these are summarized in table 4.4.

	Kalman Filter	Least squares
Dynamics	Forward dynamics function g for sensor and ball	Virtual measurement functions h_{dynB} (ball) and h_{dynS} (sensor)
Measurements	Measurement function h_{measB} for ball and h_{measS} for sensor	
Dynamic noise	Covariance matrix R	Covariance matrices Σ_{dynB} (ball) and Σ_{dynS} (sensor)
Measurement noise	Covariance matrices Q_{measB} (ball) and Q_{measS} (sensor)	Covariance matrices Σ_{measB} (ball) and Σ_{measS} (sensor)

Table 4.4: Tabulated comparison of the needed functions and uncertainty parameters for a Kalman filter and the method of least squares to estimate the state of a flying ball using a camera-inertial sensor.

Chapter 5

Experimental Setup and Implementation

In the previous chapters, all vital parts to implement a system that is capable of estimating the state of a flying ball using a camera-inertial sensor by two different estimators were described. Since the goal is to analyze the quality of prediction during a ball flight, comprehensible experiments needed to be performed which served as input data for the estimators.

This chapter deals with how and where the experimental data for the evaluation were obtained. Furthermore, it explains how the actual ball bounce point on the field was calculated from the camera image of the ball bounce, how both of the estimators were implemented using the model functions of the previous chapter and how the upcoming states of the ball can be predicted.

5.1 Sensor Setup

The sensors used in this experiments were a Basler A310f Firewire camera paired with a $4.2\ mm$ wide angle lens and a XSens MTx inertial sensor. Both sensors were mechanically combined on a common bicycle helmet (see figure 5.1). First, the inertial sensor was attached directly onto the top of the camera, so that the distance between both sensor coordinate system origins became as minimal as possible. After that, camera and inertial sensor were mounted onto the helmet so that the view of the camera corresponded to the view of the person wearing the helmet.

Both sensors sent their measurement data at a rate of 54 Hertz to a portable computer which was held by another person who was attending the experiment (alternatively, the computer might be stored in the helmet wearer's backpack). Additionally, the camera

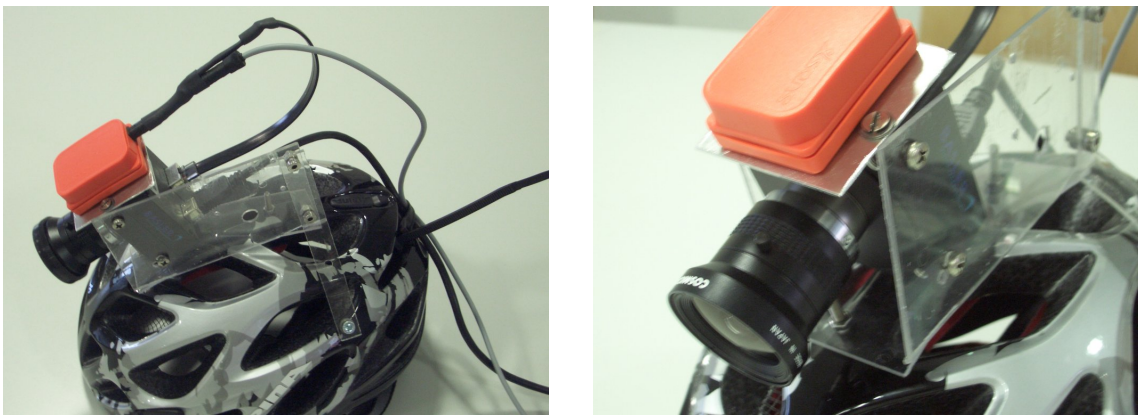

Figure 5.1: Used sensor setup for the experiments. (Left) The sensor setup viewed from the side. The photograph shows how the inertial sensor is mounted on top of the camera, the connection cable of each sensor and the synchronization cable between both sensors. (Right) The sensor setup shown from the front. The camera is mounted inclined onto the helmet so the view of the person wearing the helmet matches the view of the camera.

and the inertial sensor were connected to each other through a synchronization cable. This cable ensured that each inertial sensor measurement corresponded to a single image captured by the camera, which made the recombination of data at a later date more reliable.

Three processes were running on the portable computer during the experiments to collect the incoming measurement data. The first process, the camera-server, recorded the incoming images from the camera and saved them onto the disk using the timestamp as the filename. The second process, the inertial-server, accepted all the arriving measurement data on the serial bus from the inertial sensor and sent them over to the third process, the logger, which saved each measurement along with the current timestamp to a file. Therefore, the resulting data of an experiment was a set of images on the hard disk and a set of inertial sensor data stored in a file, which were coupled by timestamps. The software implementation used for controlling and accessing the setup was based on the implementation by Kurlbaum [6].

5.2 Experimental Environment

All experiments were performed on a real-size synthetic turf soccer field. The dimension of the field, the dimension of the lines defining the different areas of the field and the width and height of the goals were measured for the creation of a model of the field which was used by the vision based localization.

The used balls in this experiments were standard soccer balls with a diameter of $0.22\ m$ and a weight of $0.43\ kg$.

5.3 Experimental Procedure

The experiments were performed along the following procedure:

1. The sensor setup was calibrated using the sensor calibration procedure.
2. The noise of the inertial sensor was determined by observing the noise of a resting sensor.
3. The helmet with both sensors was then worn by a person who was moving around the soccer field. On this field, the person observed various ball flights which were initiated by another person. The observed ball flights were part of common soccer situations like corner kicks, shots on goal and passing.

In fact, the determination of the noise parameters of the inertial sensor was done at a much later date, which may have lead to inaccurate results, since the measurements depend on environmental conditions such as the temperature. Better results may be obtained when the noise parameters are obtained directly after or shortly before the experiment.

5.4 Image Preprocessing and Field Model

Before the experiments could be evaluated, the images had to be preprocessed. For this evaluation, the relevant features of the environment needed to be extracted out of the images so that they could be fed to the estimators.

Here, the extraction of features out of the images was done manually. A software (figure 5.2 left) read the set of images of a ball flight scene, and the location and size of the ball were extracted through user interaction. Similarly, the location of relevant field points was obtained by marking their position in the image manually.

The relevant points on the field are defined by a model, which was created from the measurement of the line and goal dimensions of the field. These points, which are also called landmarks, are: all intersections of two lines on the field, the lower end of the goal posts, the intersections of the goal post and the crossbar and the penalty spot. Every landmark is identified by a unique number. The whole set of landmarks on the field is depicted in the right subfigure in figure 5.2.

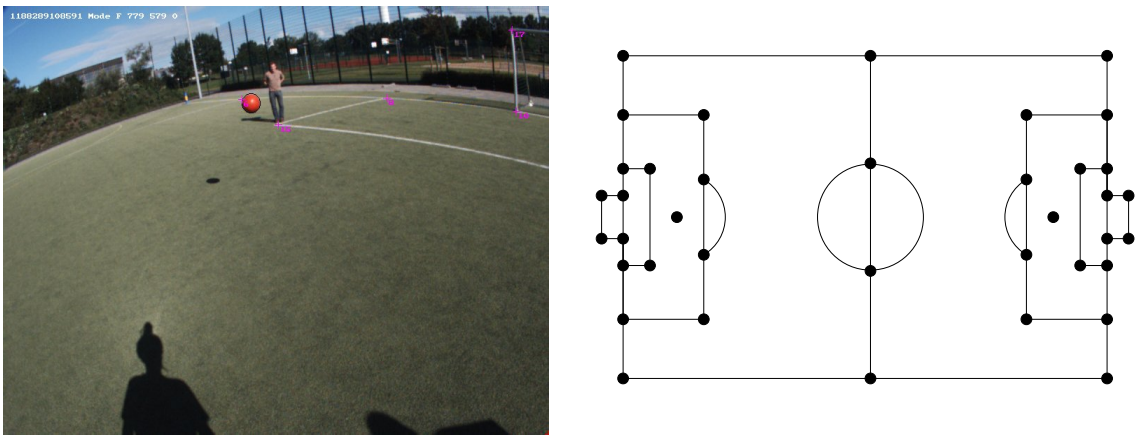

Figure 5.2: Extracting the image information using a field model. (Left) Screenshot of the software used for extracting the ball (marked as black circle) and the landmark points (marked as purple crosses) out of the camera images. (Right) Sketch of a soccer field showing the used landmark points as dots.

Because of this manual image processing, all measurements for the extracted balls and landmarks are given in integers.

5.5 Measuring Ground Truth

In practice, acquiring the real bouncing point of a ball is not that easy to accomplish. Actually, there is no way to measure the real point, since the ball leaves no permanent mark on the field which would allow a manual measurement.

For the evaluation of these experiments, the actual bouncing point of the ball was determined using an estimated homography H between at least four landmarks on the field plane X (given in world coordinates) and their corresponding undistorted points in the image m

$$m = H\ X \tag{5.1}$$

where X and m are given in homogenous coordinates.

With this homography, the world coordinates of any point on the field plane can be computed from measurements within the image plane. For the experimental setup here, the temporarily raised dust caused by the ball's bounce is extracted manually from the image. After that, the corresponding position on the real field is calculated using the estimated homography.

The accuracy of this technique depends highly on the extraction accuracy of the chosen landmarks and the raised dust from the camera. Furthermore, the accuracy depends on

Figure 5.3: Grid overlay created by a homography between points on the image plane and points on the field plane. The four points chosen for the homography estimation are marked by red circles. The space between the lines is $1\ m$. It can be seen (especially near the front line of the goalkeeper's area), how well the grid resembles the actual plane on the field. The image here shows a ball right after it bounced off the ground. The position of the raised dust is used to determine the actual bouncing point of the ball on the field.

the quality of the estimated distortion parameters from the camera calibration. Actually, no quantitive value for the accuracy of this method can be given, but as it can be seen from figure 5.3, which displays a grid created out of a homography of four points on both planes, the grid aligns well to the actual field lines. Measuring a point in image coordinates and transforming it into the field coordinates using a homography is therefore assumed to be quite close to the actual point on the field for the upcoming evaluation.

5.6 Estimator Implementation

The implementations of both estimators were based on the algorithms of chapter 2. It will now be described how the model functions from chapter 4 integrate into both algorithms and further implementation details will be considered. MATLAB / GNU Octave was used for the implementation of the estimators.

5.6.1 Relationship Between Measurement Data and States

Since both sensors are synchronized and collect their data in discrete time, the relationship between sensor measurements and the to-be-estimated states is simple. Every pair of measurements from the inertial sensor $z_{dynS,t}$ and from the vision, which measures the landmarks $z_{measS,t}$ and the ball $z_{measB,t}$, at time t belongs to exactly one state x_t at time t.

5.6.2 Initial State Determination

Both estimators require an initial state x_0 so that the first dynamic functions can be applied to determine the state of the system after the measurement.

To obtain the initial state for the sensor and ball pose, a least squares estimation was performed using the landmark and ball measurements of the camera image right before the first image of the ball flight. A function Q, which had to be minimized, was defined as a sum of quadratic deviations between the predicted ball and landmark measurements (obtained from the measurement function h_{measB} and h_{measB} and the to be estimated state x_0) and the actual measurements $z_{measB,0}$ and $z_{measS,0,j}$:

$$Q = \sum_j \left(\frac{1}{\Sigma_{measS}} (z_{measS,0,j} - h_{measS}(x_0))^2 \right) + \frac{1}{\Sigma_{measB}} (z_{measB,0} - h_{measB}(x_0))^2 \quad (5.2)$$

Since absolute accuracy was not required at this stage of the estimation, the required uncertainty parameters for this optimization were chosen manually.

Once Q was minimized, the resulting x_0 was used as the initial pose for the ball and sensor. The velocities of the ball and sensor in the initial state were set to zero.

5.6.3 Unscented Kalman Filter

The UKF implementation is straightforward. The dynamic function for the Kalman filter g and both measurement functions h_{measB} and h_{measS} integrate seamlessly into the prediction and the correction step respectively. The dynamic noise is modeled using the covariance matrix R. The measurement noise of the manual vision is represented by the covariance matrices Q_{measS} for the landmark measurements and by Q_{measB} for the ball measurements. All values of the covariance matrices are obtained using the model parameter determination method from section 4.5. The obtained standard deviation from the algorithm was at about 1 pixel for the experiments.

Finally, the initial covariance of the dynamic system needs to be chosen. These values depend on the quality of the initial pose estimation and the actual dynamics of the system.

Since the estimation of the initial pose is usually quite accurate, the initial covariance for the sensor position and rotation are set to the value $(0.05\ m)^2$ and $(0.035\ rad)^2$. The covariance for the velocity of the sensor is assumed to be $(5\ m/s)^2$. For the uncertainty of the initial ball position, the fact that the ball is lying on the field at the initial state can be exploited. Therefore, the z-coordinate of the covariance is set to a quite small value of $(0.075\ m)^2$, whereas the x and y coordinate are set to $(1\ m)^2$. The initial covariance of the ball's velocity ranged from $(10\ m/s)^2$ to $(20\ m/s)^2$, depending on initial velocity applied to the ball in the experiment. The implementation of the filter was based on the implementation by Kurlbaum [6].

5.6.4 Least Squares Method

As it is known from section 2.1.5, modeling a dynamic system using the method of least square is not as elegant as in the Kalman filter. Here, the estimation of the state at time t is done by minimizing the target function Q_t. This function is defined as the sum of quadratic deviations between the predicted measurement (in which some are virtual) and the actual measurement for *all* the states from 1 to t. Formally, the function reads

$$\begin{aligned} Q_t \quad = \quad & \sum_{i=1}^{t} \left(\frac{1}{\Sigma_{dynS}} ([z_{dynS,i}, x_{s,i-1}] - h_{dynS}(x_{i-1}, x_i))^2 + \frac{1}{\Sigma_{dynB}} (x_{i-1} - h_{dynB}(x_{i-1}, x_i))^2 \right. \\ & \left. + \sum_{j} \left(\frac{1}{\Sigma_{measS}} (z_{measS,i,j} - h_{measS}(x_i))^2 \right) + \frac{1}{\Sigma_{measB}} (z_{measB,i} - h_{measB}(x_i))^2 \right) \end{aligned}$$

where $[z_{dynS,i}, x_{s,i-1}]$ is the inertial sensor measurement augmented by the sensor position at time $i-1$, h_{dynS} is the measurement function for the sensor dynamics and Σ_{dynS} is the assumed uncertainty in the measurements. The other subscripted symbols in the equation are defined in the same way, where $dynB$ subscribes the virtual measurement function and the uncertainty of the the ball dynamics. $measS$ and $measB$ subscribe the measurement function, uncertainty and actual measurement of the landmark and the ball respectively.

In contrast to the Kalman filter, there is no initial covariance since all states up to t are estimated at once. In order to integrate the known fact of the lying ball at the start of the flight, another measurement function needs to be integrated into the to be minimized function:

$$Q'_t = Q_t + \frac{1}{\Sigma_{initB}} (x_{ball,init} - h_{initB}(x_0))^2 \tag{5.3}$$

Here, x_{init} denotes the initial state, h_{initB} is the measurement function which simply returns the ball state at time 0. Σ_{initB} represents the uncertainty in the determined initial state. As with the initial covariance, the information that the height of the ball on the field

is known at the initial state is encoded here and the covariance values from the Kalman filter may be used.

The least squares optimization was performed using the Levenberg-Marquardt algorithm from section 2.1.3. The initial parameter guesses were calculated from a preceding state estimation of the unscented Kalman filter. The required calculation of the first derivative was obtained through numerical differentiation.

5.6.5 Prediction

Once the state of a ball in flight is estimated, this state may be used to predict the next states and therefore to predict the trajectory of the ball flight up to the point where the ball hits the ground.

While evaluating the data from the experiments, the prediction of the next state from a state x at time t was performed using the forward dynamic function g which is also used in the Kalman filter.

Chapter 6

Analysis

In order to answer the question how well the flight of a ball observed by an camera-inertial sensor can be estimated and predicted, three ball flights recorded during the experiments will be analyzed. The goal of this analysis is not only to find out whether or not the accuracy of the methods presented in this thesis are sufficient for a hypothetical robotic soccer player to interact with the ball. Furthermore, it is also of interest how the performance of prediction behaves over time.

To accomplish the analysis, it is first defined what is needed to successfully interact with a ball on a soccer field with respect to accuracy. After that, the three flights will be evaluated. Finally, the differences between both estimators will be addressed briefly and the obtained evaluation results will be summarized.

6.1 Analysis Goals and Constraints

Interacting with the ball on the field is a quite demanding task for a soccer playing human. The player not only has to anticipate the trajectory of the ball but also needs to position himself to accept or intercept the ball. For the goalkeeper, this task becomes unequally harder since he has to predict ball flights shot by highly skilled players within his range movement which only covers the goal partially.

For the evaluation of these experiments it is assumed that the quality of estimation and prediction is independent of the robot's position. Furthermore, it is assumed that the robot is always in the right position for ball interaction. For the analysis of accuracy with respect to a hypothetical humanoid robot, mental chronometry (i.e. the temporal sequencing of cognitive actions starting from the perception and ending in an action) of humans might be used as an indicator of requirements for ball estimation and prediction. Actually, mental chronometry measurements for the task of playing soccer are not avail-

able. For this analysis, the critical time that is allowed to elapse between the point where the cognition is finished and the point when the motor related actions take place (e.g. the player moves his foot) is $500\ ms$. This value is very conservative and it is assumed that skilled players will actuate faster. Estimation and prediction are assumed to be accurate if the distance between the predicted bouncing point and the real bouncing point is below $0.3\ m$. This value was chosen as a trade-off between the size of the foot, head and chest, the three most commonly used parts of the human body in soccer.

Therefore, for this evalution, ball flight prediction is assumed to be accurate for a hypothetical robotic soccer player if the accuracy is below $0.3\ m$ during the last $500\ ms$ before the ball hits the ground.

This appears to be enough for accepting a common ball kick. Other tasks than accepting, such as volley kicking, will probably require higher accuracy. However, this is impossible to evaluate with this experimental setup because no ground truth with the required higher accuracy is available.

6.2 Expected Results

From the experiments of Kurlbaums work and the general theory of estimation within dynamic systems, the following behavior during the estimation and prediction process is expected:

- Since both estimators are getting enough information about their motion through inertial and landmark measurements, localization of the sensor itself should not be problematic.
- Due to the integer based values of the image processing, the estimation of the ball's position during the first couple of frames should be quite inaccurate. This applies to the estimation of the velocity of the ball, too. Therefore, the prediction should perform poorly in the first couple of frames.
- The estimation of the ball's position and velocity should improve the longer the ball flies, since more measurements are processed and the physics of the ball flight determines the trajectory.
- The unscented Kalman filter and the Levenberg-Marquardt algorithm should perform almost identically. Averaging their performance over many ball flights, the Levenberg-Marquardt algorithm should outperform the Kalman filter slightly. If there is a difference in their trajectory estimation, it should be rather at the beginning of the trajectory.

6.3 Analysis Results

Three types of ball flights will be analyzed:

- A ball is moving from a short distance towards the observer and the prediction accuracy is determined with respect to ground truth. This flight should be an adequate example for short passes directed to the player.
- A ball is passing by the observer. Assuming that the last estimate has only minimal deviation, the prediction accuracy is determined with respect to the predicted bouncing point of the last estimate at the end of the flight. This flight should be an example for the observation of ball flights which may lead to an interception.
- A long distance, banana-shaped flight of a ball towards the observer. The prediction accuracy is determined as in the flight above. This ball flight should represent a long pass, a cross or a corner kick.

6.3.1 Short-Distance Ball Flight Towards the Observer

The first observed trajectory is used to determine the system performance compared to ground truth. 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 is flying almost spin-less and the impact of the side wind to the ball at this range of flight and speed is minimal.

To obtain the ground truth of the ball's bouncing point, five images following the bounce were chosen and the image position of the raised dust caused by the ball is determined manually. The actual ground truth for this ball flight is then obtained by computing the mean of the five field positions from the image positions and the homography describing the transformation between image plane and field plane.

Estimated trajectory

The estimated trajectory of the ball flight by the unscented Kalman filter and the Levenberg-Marquardt algorithm is depicted in figure 6.1. As expected, both methods estimate almost the same set of states for the whole flight and the degree of similarity increases over time. What directly catches the eye of the viewer is that the trajectory of the estimated ball positions is not as smooth as an actual physical flight of a ball. This behavior can be explained by points 2 and 3 from the list of the expected results. The ball diameter is only given in integer values, making the real depth of the ball difficult to estimate, which leads to this strange looking trajectory.

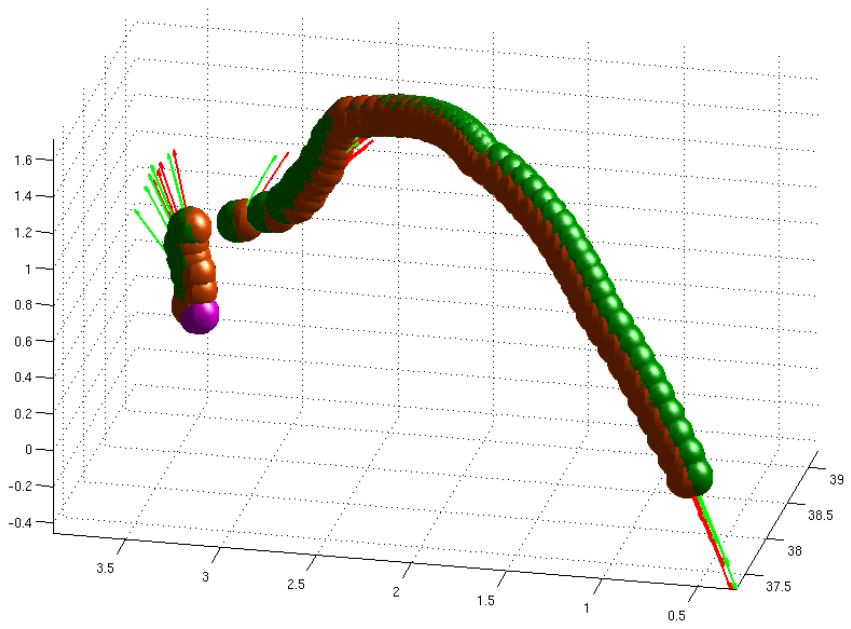

Figure 6.1: Three dimensional view of estimated ball trajectories and velocity vectors while the ball is flying towards the observer. As can be seen, the estimated trajectories of the Unscented Kalman Filter (UKF), shown as orange balls, and the Levenberg-Marquardt algorithm (LM), shown as green balls, differ in the beginning but approach each other as the flight progresses. The purple ball represents the initial ball state.

The unscented Kalman Filter and the Levenberg-Marquard algorithm show another interesting property while estimating the states within this trajectory. In the first few frames, until the diameter of the ball increases by one, reflecting the fact that the ball is moving towards the sensor, the ball is estimated behind the initial ball position. This results in a velocity vector pointing away from the sensor into the opposite direction. Again, this phenomenon can be explained by the integer values of the ball measurements. The inaccurate circle diameter measurement leads to a state estimate in which the ball is wrongly estimated behind the initial ball position. This is not suprising since the monocular camera defines the ball's distance by the ball's diameter. Within the first few frames the diameter is 16 ± 1 pixels, leading to an expected error of 6.25% in the distance.

Prediction performance

To determine the accuracy of the ball state estimation during the flight, the ball states are being predicted until the ball hits the ground of the field. For the estimated states in this flight scene, the predicted ball bouncing points of the states are shown in figure 6.2 as a

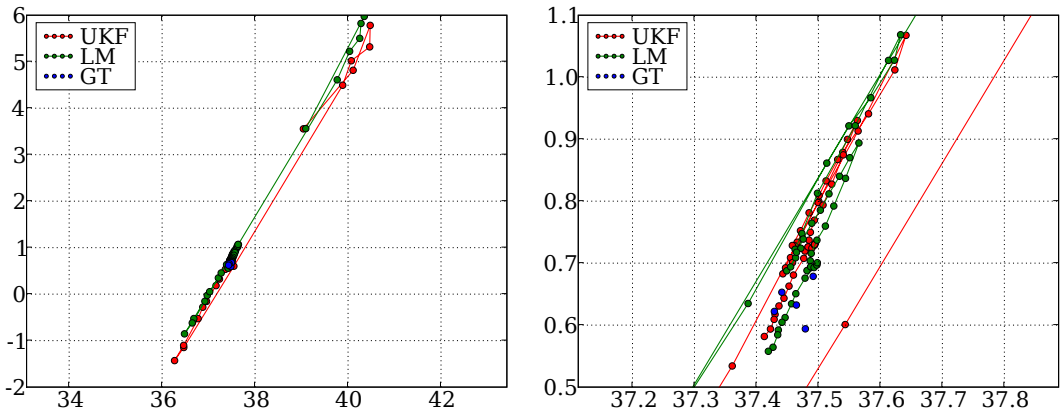

Figure 6.2: Predicted positions on the field of the estimated states' bouncing points as a function of time with respect to the positions determined by the ground truth measurements of the whole trajectory (shown left) and a zoom in near the assumed bouncing point (shown right). The initial ball position was at about $38.5\ m$ in the x-axis and $3.5\ m$ in the y-axis.

function of time along with the five points on the field that were obtained through the ground truth determination using homographies. Comparing the predicted ball bouncing point with the assumed real bouncing point, it can be seen that the accuracy starts poorly in the first few frames but then drastically improves. The poor start is due to the fact that the first estimated states are behind the initial ball position, leading to predicted ball bouncing points which are far away from the actual ball bouncing point. The more information about the depth is gained from the sensor the better the estimation becomes. For the states at the end of the trajectory the predicted ball bouncing points are becoming more accurate. The discrepancy shrinks until the sixth last estimated state of both methods almost reaches the mean of the five ground truth measurements. The upcoming predictions are slightly beyond this mean and the accuracy is therefore decreasing.

However, the visual information contained in figure 6.2 is quite limited. As a representation of the estimator's accuracy, the overlapping estimation results make the actual accuracy hard to extract. To show how the accuracy changes during the estimation of the states over time, the distance between the mean of the five ground truth measurements and the predicted states is plotted in figure 6.3. The plot shows the already observed improvement of accuracy over time more clearly. Starting with the estimation of ball states behind the initial ball position, which lead to very inaccurate ball predictions as described above, the difference between the assumed real bouncing point and the predicted points from both estimates decreases significantly after 7 frames (or about $129\ ms$), reaching a surprisingly good estimate with a difference of just $0.4\ m$. After that, the accuracy

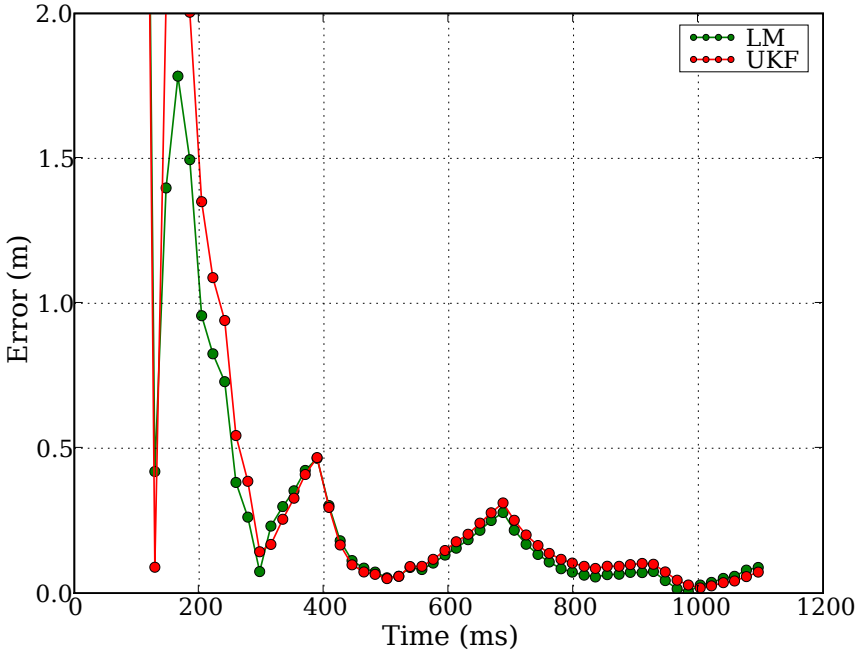

Figure 6.3: Error between the predicted bouncing points computed from the estimated trajectory and the mean of the ground truth measurements over time.

decreases rapidly for a couple of frames before it reaches a next good estimate after 16 frames ($297\ ms$) with a error between 0.1 and $0.15\ m$. From here, the estimates seem to be quite reliable and the prediction performance improves in a wave-like shape with decreasing peaks over time.

Within this wave-like shape, there seem to be certain recurring estimates where the predicted ball bouncing point is very close to the assumed bouncing point. Interestingly, over time the recurring good estimates are the minima of these waves. The reason for this behavior is the integer based image processing: The peaks of the waves in the figures are caused by a change of the ball's measurement of the circle diameter. As time progresses, the ball moves towards the sensor and the diameter increases, representing the visual information about the ball's distance. Intuitively, the integer based depth information is resulting in inaccurate states after and right before a peak, whereas somewhere in between two peaks the estimated state should match the circle diameter exactly, leading to a minimum. Over time, the drawback of the integer based image processing values diminishes as the physical properties of the flight dominate the ball's state estimation.

As can be seen in this figure, the accuracy requirement is fullfilled in this flight. Within the last $500\ ms$, the predicted bouncing point was always within $0.3\ m$. Actually, the predicted trajectory was already sufficiently accurate $670\ ms$ before the ball flight ended.

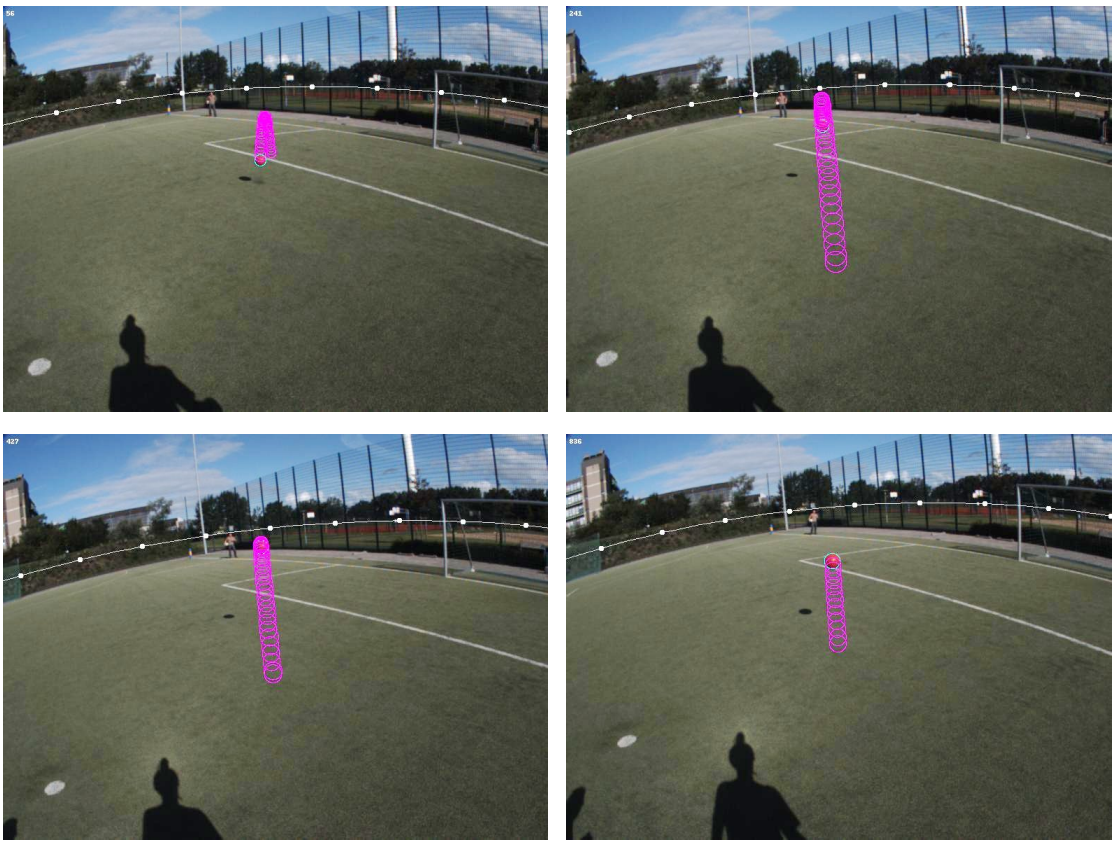

Figure 6.4: Four frames of the ball trajectory captured by the camera while the ball is flying towards the observer. The predicted trajectory from this state is shown in purple circles.

In-Image Visualization

In order to get an impression of the observed trajectory, the scene of the ball flight is summarized using four images from the whole series describing the flight. In addition to the raw images, the result of the ball position estimation of the unscented Kalman filter and the predicted trajectory from this state is projected to image coordinates and drawn into the image. Only the estimation of the unscented Kalman filter is shown, since the difference in the estimation result of both methods is barely visible in image coordinates. Obviously, this visualization does not contribute anything to the accuracy analysis and is just for the reader's convenience.

It can be seen from the series how the estimation of the trajectory is inaccurate in the first image but becomes better in the second snapshot and finds the upcoming trajectory quite accurately in the third image. This is confirmed by the fourth image, in which the

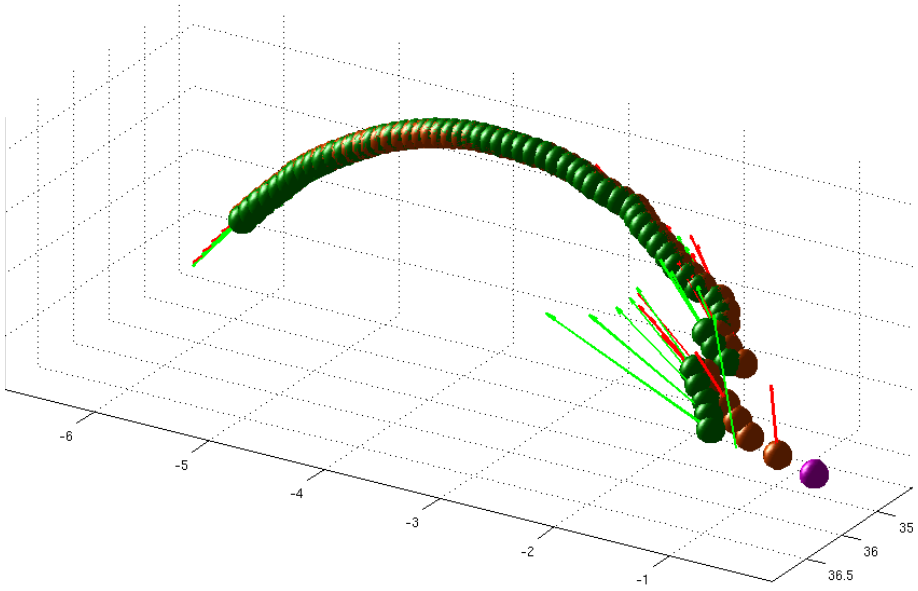

Figure 6.5: Three-dimensional view of estimated ball trajectories and velocity vectors while the ball is passing by the observer. Similar to the previous flight, the estimated trajectories of the unscented Kalman filter (UKF), shown as orange balls, and the Levenberg-Marquardt algorithm (LM), shown as green balls, do not agree in the beginning but come closer to each other as the flight progresses. The purple ball represents the initial ball state.

predicted trajectory visually does not change with respect to the third camera image.

6.3.2 Ball Flight Passing by the Observer

The second flight evaluated here is a ball passing by the observer at a distance of about $5\ m$. The flight itself is about $5.5\ m$ long and the ball reaches a maximum height of $1.5\ m$. In contrast to the first flight, no ground truth measurements are available due to the lack of significant landmarks within the images. Because of the quite good prediction results of the previous flight's last states, the predicted bouncing point of the last state in this flight is assumed to be the actual bouncing point. Again, this ball flew almost spin-less and the impact of sidewind was negligible.

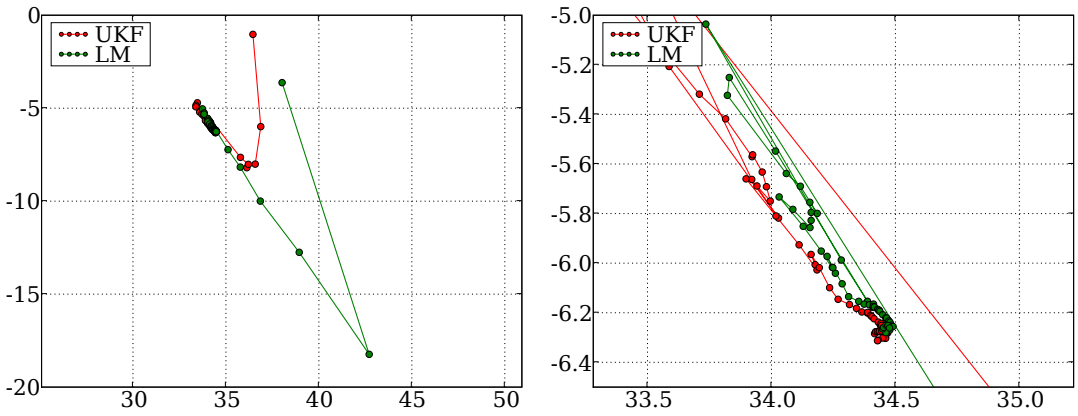

Figure 6.6: Predicted positions on the field of the estimated states' bouncing points over time of a ball passing by the observer. The whole trajectory predictions are shown in the left and a zoom in near the assumed bouncing point is shown in the right subfigure. The initial ball position was at about $36.2\ m$ in the x-axis and $-0.3\ m$ in the y-axis.

Estimated trajectory

Figure 6.5 shows a three-dimensional view of the estimated ball trajectory and velocity vectors. Similar to the trajectory of the previous flight, the estimation performance is poor within the first few frames due to the difficult determination of the ball's depth. As expected, after these couple of frames the estimation becomes more reliable and the unscented Kalman filter and the Levenberg-Marquardt algorithm almost estimate the same set of the states for the remaining trajectory.

Prediction performance

The predicted positions of the estimated states' bouncing points are shown in figure 6.6. As can be seen, the direction of the flight is estimated well and the error seems to lie on the line between starting and bouncing point.

Again, the visual information contained in figure 6.6 is hard to extract. Therefore, the change of the accuracy during the state estimation over time is plotted as the distance between the bouncing point of the last estimated state and the predicted bouncing points from the trajectory's state in figure 6.7. The plot shows similarities to the plot of the previously observed trajectory. The error in the first few frames is large but after 9 frames the Levenberg-Marquardt algorithm made a suprisingly good estimate, in which the error in prediction is just about $0.13\ m$. Both estimators agree at an estimate with an error ranging from $0.5 - 0.7\ m$ at about frame 15 when $278\ ms$ were elapsed. From here, the accuracy

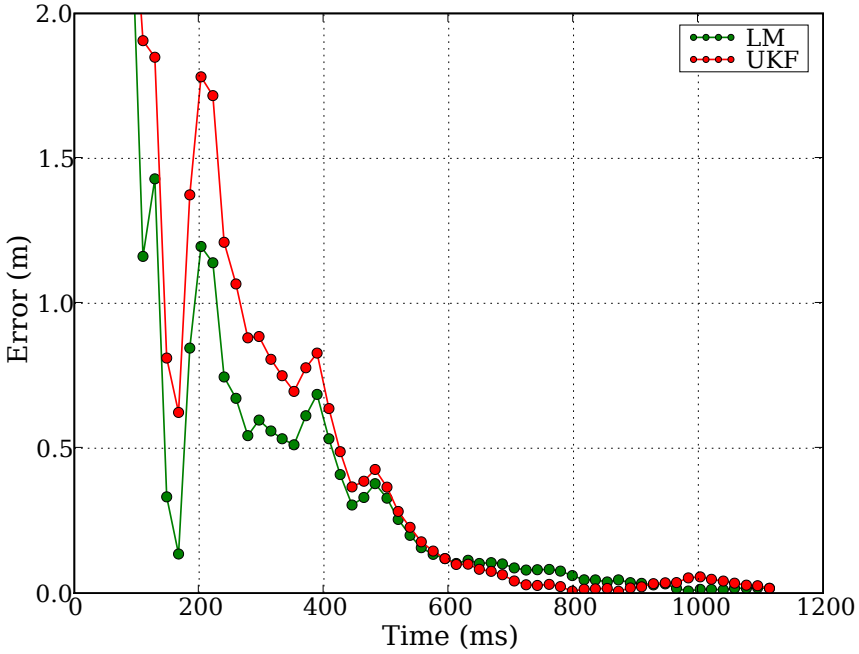

Figure 6.7: Error between the predicted bouncing points computed from the estimated trajectory and predicted bouncing point of the last estimate over time for a ball passing by the observer.

decreases once but recovers right afterwards, resulting in an almost gradually increasing accuracy over time. Obviously, the predicted trajectories of the last states show perfect accuracy since the last state's trajectory was used as the assumed actual bouncing point of the ball.

As with the previous ball flight, the accuracy requirements are fullfilled in this flight. Within the assumed movement-time of $500\ ms$ the ball is always estimated accurately in the defined sense. The sufficiently accurate predicted trajectory was obtained $595\ ms$ before the bounce occured.

In-Image Visualization

Figure 6.8 shows four frames of the captured vision data with additional geometric annotations representing the estimated ball state and the predicted trajectory of the unscented Kalman filter. Similar to the previous flight, the estimation in the first frame is poor, leading to a visually incorrect trajectory prediction. The second frame shows a significant improvement of the predicted trajectory. In the third frame, the predicted trajectory is refined and seems to be quite accurate since it visually does not change much in the fourth

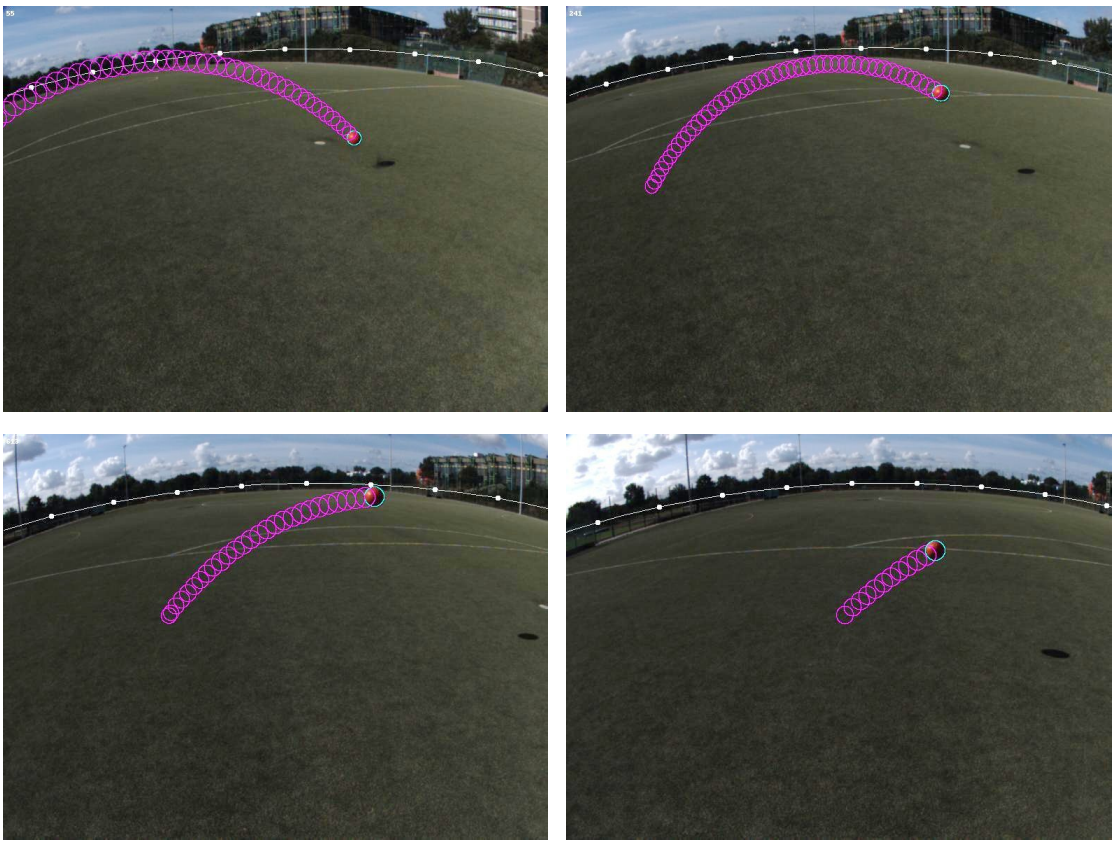

Figure 6.8: Four frames of the ball trajectory (from the top-left to the bottom-right) captured by the camera while the ball was passing by the observer. The predicted trajectory from this state is shown in purple circles.

frame.

6.3.3 Long-Distance Ball Flight Towards the Observer

The last of these three ball flights is somewhat special and is primarily presented to reveal the limits of the underlying model. This flight is a corner kick directed to the goal with the observer near the goal. The length of the flight was $40\ m$ and it reached its peak at $5.5\ m$. While shooting, a spin was applied to the ball so it bended towards the goal, resulting in a banana-shaped flight trajectory. As with the previous example, no ground truth measurements were available and the bouncing point of the ball was assumed to be the predicted bouncing point of the last estimated state.

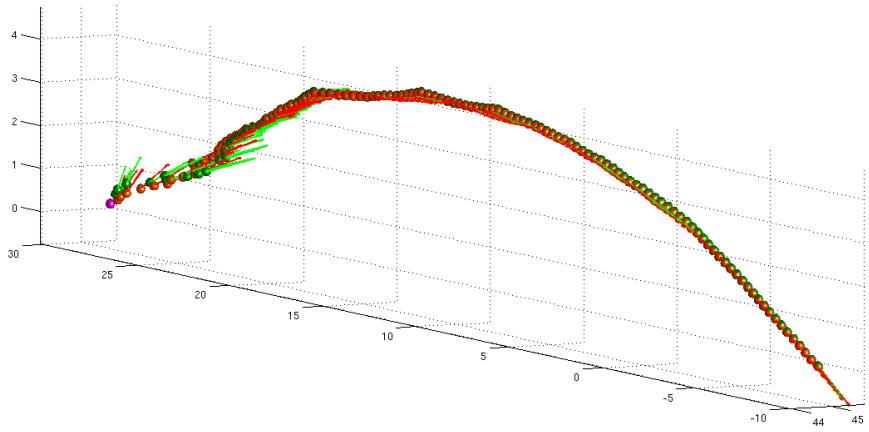

Figure 6.9: Three dimensional view of estimated ball trajectories and velocity vectors of a ball that is flying towards the observer from a long distance. The estimated trajectory of the Unscented Kalman Filter (UKF) is shown as orange spheres. The trajectory estimated by the Levenberg-Marquardt algorithm (LM) is shown as green spheres.

Estimated trajectory

The trajectory of the ball flight estimated by both methods is shown in figure 6.9. Again, in the first few frames the estimate does not resemble the actual trajectory of this flight. The reason for this behavior is similar to the behavior of the state estimation in the first few frames of the first evaluated ball flight, but this time it is the other way around. Due to the quite inaccurate integer-based values of the ball's diameter measurements, the ball is estimated several centimeters in front of its actual position. The accompanying estimated velocity resulting from this position is quite large and does not correspond in any way to the actual velocity during this stage of flight. Once enough information is gained, better estimates by both methods are obtained and the banana-shaped trajectory is estimated quite well. Since the model does not take the Magnus effect into account, the augmented noise parameters of the ball dynamics take care of the bended trajectory.

Prediction performance

Both of the described behaviors during the state estimation within the trajectory are clearly visible when predicting the estimated state up to the point the ball hits the ground (shown in figure 6.10). At first, the estimation of the ball's position up to $5\ m$ in front of its starting point leads to predicted bouncing points far beyond the position of the observer. Since the estimation becomes more reliable a few frames later, the predicted bouncing

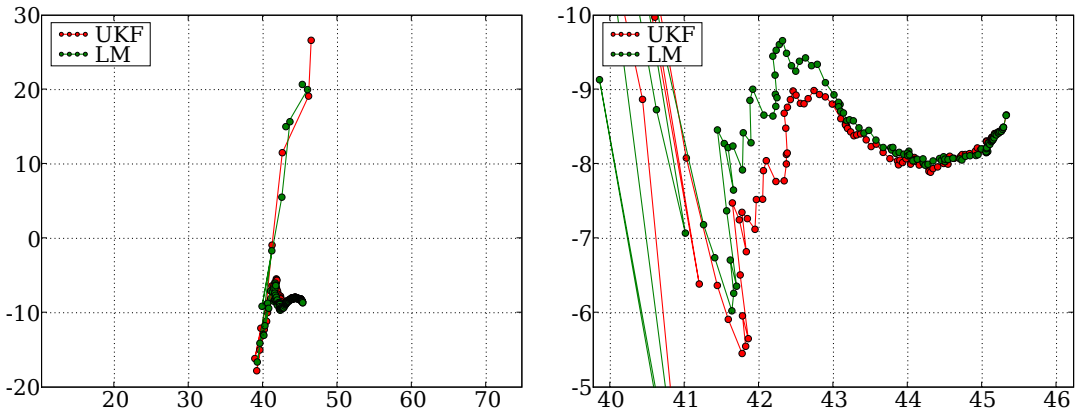

Figure 6.10: Predicted positions on the field of the estimated states' bouncing points as a function of time of a ball approaching the observer from a long distance. The initial ball position was at about $45.7\ m$ in the x-axis and $30\ m$ in the y-axis.

points settle themselves at a much more plausible position. Secondly, since the underlying dynamic model of the ball does not take the spin and the resulting interaction with air into account, the predicted trajectories are not bended. This results in an inaccuracy during prediction which reduces over time since the effect of bending becomes less significant as the ball progresses in flight. This effect is clearly visible in the right plot of figure 6.10.

Figure 6.11 shows the accuracy over time as the distance between the predicted bouncing point and the predicted bouncing point of the last estimate. The plot is limited to a deviation of $10\ m$ so the huge inaccuray of over $20\ m$ during the first few frames is not visible. The accuracy increases as more information is processed: The prediction only deviates $4.4\ m$ after 9 frames for the UKF and $4.7\ m$ after 10 frames for the LM, decreases in the upcoming framess until it settles between $4.3-4.6\ m$ for a couple of frames starting from frame 15 at about $279\ ms$. From here, the prediction becomes only more accurate. This effect is due to the bended trajectory as described in the paragraph above, and as can be seen in the figure, the accuracy increases gradually as the ball moves towards its bouncing point.

Obviously, the accuracy requirements for the hypothetical robot are not met in this flight. The necessary $0.3\ m$ are not achieved during the $500\ ms$ before the ball hits the ground. At this point of time the predicted bouncing point is about $0.73\ m$ away. About $186\ ms$ before the actual bounce occurs the required accuracy is reached.

It should not be left unmentioned that the wave-like shape of the change in accuracy over time, which was observerd in the plot of the first evaluated ball flight (see figure 6.3), is also present in this ball flight. At the balls' peak in the three-dimensional visualization

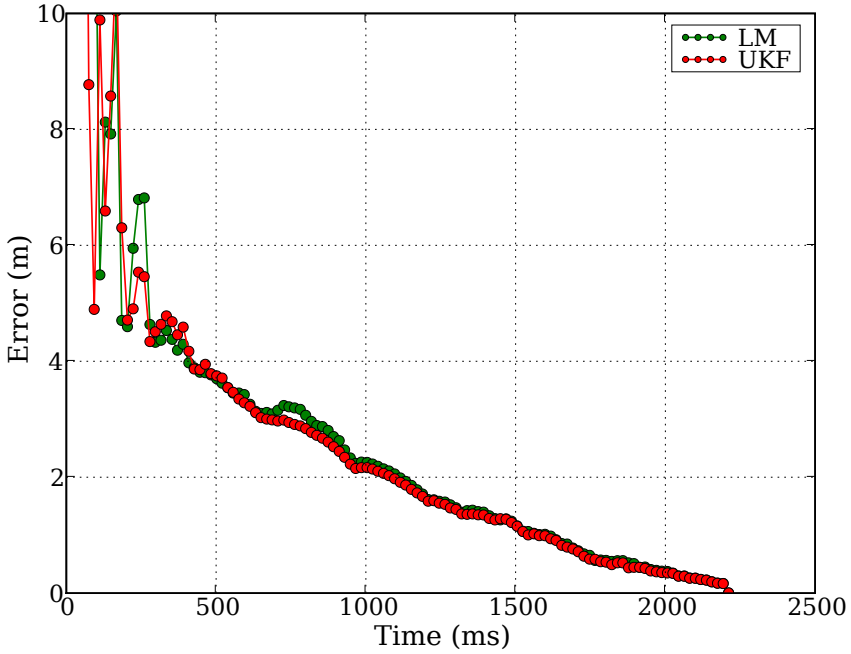

Figure 6.11: Error between the predicted bouncing points computed from the estimated trajectory and the predicted bouncing point of the last estimate over time of a ball approaching the observer from a long-distance.

two small peaks are visible, which resemble changes in the ball's diameter measurement. The effect of the ball's diameter change can easily be seen in the right subfigure of figure 6.10, in which it causes the change in the length of the estimation. Paired with the decrease of the effect caused by the curve of the flight, the predicted bouncing points approach the assumed bouncing point in a zigzag-shape for a couple of frames.

In-Image Visualization

In figure 6.12, four frames of the captured vision data with annotations visualizing the estimated ball state and the predicted trajectory of the unscented Kalman filter are given. As with both of the previous flights, the prediction starts poorly in the first frame, becomes significantly better in the second frame, seems to be robustly estimated in the third frame and shows a visually accurate prediction in the last frame.

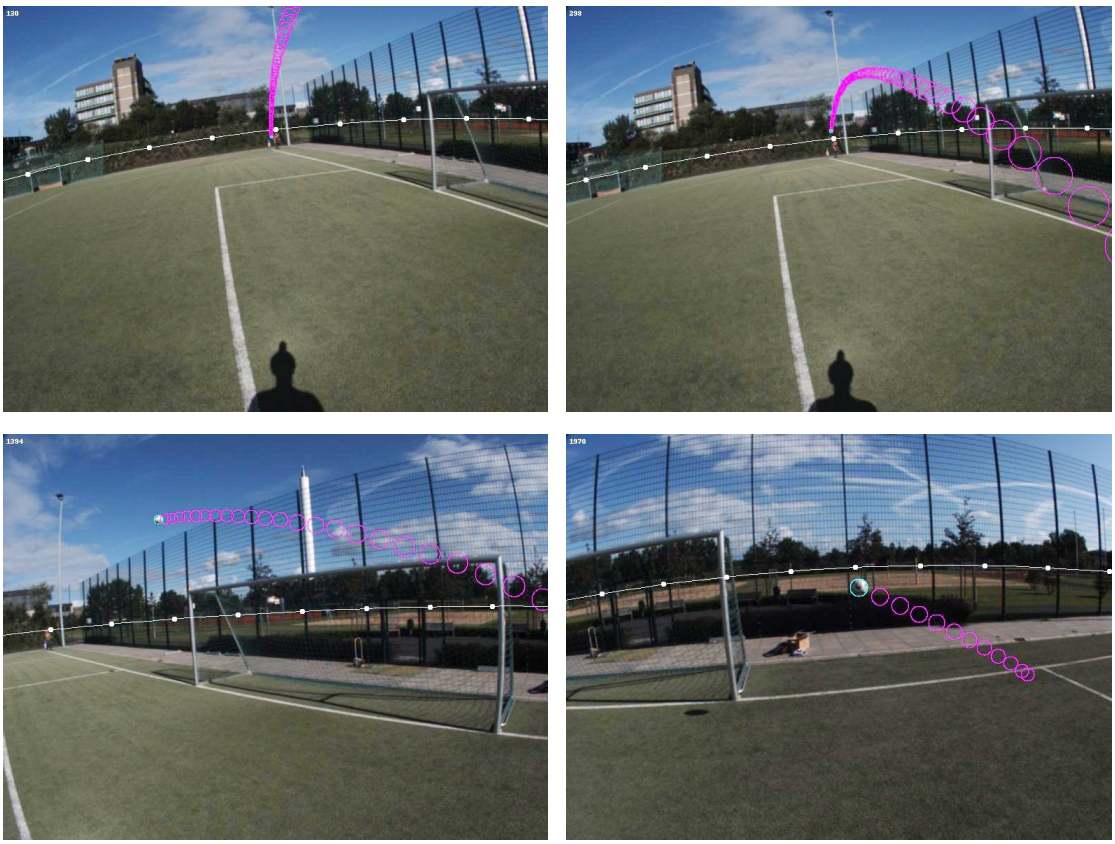

Figure 6.12: Four frames of the ball trajectory (from the top-left to the bottom-right) captured by the camera while the ball is flying towards the observer from a long distance. The predicted trajectory from this state is shown in purple circles.

6.4 Estimator Difference

There is a discrepancy in the estimated states and their associated predicted trajectories of the unscented Kalman filter and the Levenberg-Marquardt algorithm. Especially during the first few frames the difference is clearly visibile.

This behavior is not uncommon. It is assumed to be the result of the performed linearization during each estimators nonlinear function approximation. Furthermore, the initial covariances given to the unscented Kalman filter before the computation may lead to a large amount of uncertainty in the estimation during the first few frames. Detailed investigation of this behavior is out of the scope of this thesis but may become the subject of further research.

6.5 Summary

Two out of three evaluated ball flights fulfilled the required accuracy constraint. Although the tracking of the third flight seemed to be correct and the trajectory was estimated quite well, the lack of the model to predict bended trajectories prevented an accuracy within the defined constraint. This result was expected and it is clear that the integration of the Magnus effect into the model will compensate this inaccuracy. Therefore, the question if the accuracy of trajectory prediction is sufficient for a hypothetic robotic soccer player can be answered with yes, provided that only ball flights are observed that are correctly modeled by the system.

The answers for the other questions are found by comparing the results of all three evaluated ball flights. The performance over time can be separated into three distinct phases:

1. As expected, due to the monocular setup all ball flights show inaccurate predictions during the first few frames. All three flights showed a suprsingly good predicted trajectory after $8 - 10$ frames. But these predictions lasted only for one or two frames, showing that there is still a large amount uncertainty in the estimation process. Only the information about the direction of the ball flight seems to be useful.

2. The second phase shows a first accurate indication of the predicted bouncing point after a constant time. All examined flights agree that about $15 - 16$ frames ($278 - 296\ ms$) have to be processed to achieve this indication. These number of frames seem to be the point in time in which the modeled physical properties of flight determine the most of the ball's trajectory.

3. In the last phase, the accuracy usually improves gradually over time until the accuracy of the predicted trajectories almost reaches the actual bouncing point at the end of the flight.

Another interesting observation is that the accuracy constraint of the two successfully predicted trajectories was fulfilled about 95 and $170\ ms$ earlier than required.

With respect to the question how well the prediction works for different kinds of trajectories, it should be mentioned that these flights do not represent the whole set of possible ball flights that might occur on a soccer field. But the results are convincing and suggest that the prediction of most ball flights would behave like the ones evaluated here.

Chapter 7

Conclusion and Outlook

Trajectory predictions for three different ball flights were performed and evaluated within this thesis. The results clearly show that prediction can be performed sufficiently for a hypothetic humanoid soccer player if the observed ball flight is correctly modeled within the underlying dynamic system used by the estimators. Although the third flight failed to achieve the accuracy goal, the prediction indicated the area of the bouncing point.

Vital parts that contributed to the achieved accuracy are the sensor calibration and the parameter learning algorithm. The sensor calibration ensured that measurements obtained in the different coordinate systems could be transformed between each other almost error-free. The parameter determination algorithm computed the most optimal noise parameters of the visual sensor from a set of already obtained measurements. Without this automated process the parameters would have been guessed manually making the parameter determination error-prone.

Although the experiments show sufficient results, it might be of interest to investigate the source of the estimation error. Actually, during the analysis of this work an error in the estimation of the sensor height was experienced. While the system knew where the ball was relative to the sensors on the head, it was unsure where the head was relative to ground. One way to handle this specific problem would be the parallel usage of a wearable self-contained motion capturing system. By tracking the information about the body's joint structure, the height of the sensor could be obtained resulting in a reduction of the uncertainty.

Obviously, before benefiting from the results of this work in real soccer scenarios more work needs to be done. Research within the near future may include:

- Replace the manual vision by a computer vision, making the process of extracting features from the images autonomous.
- Development of a multi-state estimator, which is not only capable of estimating the

state of a flying ball but also of balls which are rolling or just lying.

- Build a robust ball recognition which may be used in conjunction with the estimator for better reliability. Such a system might rely on bioliogically inspired image processing methods to mimic human recognition abilities.
- Use not only points between lines and goal posts, but also the lines and goal posts themselves for vision-based localization. This will improve the localization, especially in places where only a few lines are in the field of view of the camera.
- Perform the computation of the state estimation in real-time using the unscented Kalman filter.
- Add the Magnus effect to the dynamic model so the estimation and prediction of banana-shaped trajectories will perform more accurate.
- Presentation of the information obtained through state estimation and prediction in an augmented reality application. For example, a head mounted display may be used to show the predicted trajectory as an overlaid graph within the camera or the real image. A possible application would be the observation of a real soccer match by a human player equipped with such an augmented reality setup.

Bibliography

[1] M. Sahota and A. K. Mackworth. Can Situated Robots Play Soccer? In *Proc. Artificial Intelligence'94*, pages 249–254, Banff, AB, May 1994.

[2] R. A. Brooks. Intelligence Without Reason. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI-91)*, pages 569–595, 1991.

[3] H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, and E. Osawa. RoboCup: The Robot World Cup Initiative. In *Proceedings of the First International Conference on Autonomous Agents (Agents'97)*, pages 340–347, New York, 5–8, 1997. ACM Press.

[4] RoboCup Federation. RoboCup Website, 2008. `http://www.robocup.org/overview/22.html`. Accessed January 24, 2008.

[5] Honda Worldwide. ASIMO, 2008. `http://world.honda.com/ASIMO/`. Accessed January 24, 2008.

[6] J. Kurlbaum. Verfolgung von Ballflugbahnen mit einem frei beweglichen Kamera-Inertialsensor. Master's thesis, Universität Bremen, 2007.

[7] U. Frese, B. Bäuml, S. Haidacher, G. Schreiber, I. Schaefer, M. Hähnle, and G. Hirzinger. Off-the-Shelf Vision for a Robotic Ball Catcher. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1623–1629, 2001.

[8] M. Simon. Ein robustes Echtzeit-Vision-System für die Robocup F180 Small-Size Liga. Master's thesis, Institut für Informatik, Freie Universität Berlin, 2006.

[9] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, New York, NY, USA, 1992.

[10] U. Frese and L. Schröder. Theorie der Sensorfusion, 2007. Lecture notes.

[11] K. Levenberg. A Method for the Solution of Certain Non-Linear Problems in Least Squares. *The Quarterly of Applied Mathematics*, 2:164–168, 1944.

[12] D. W. Marquardt. An Algorithm for Least-Squares Estimation of Nonlinear Parameters. *Journal of the Society for Industrial and Applied Mathematics*, 11(2):431–441, June 1963.

[13] A. Ude. Nonlinear Least Squares Optimisation of Unit Quaternion Functions for Pose Estimation from Corresponding Features. In *Proceedings 14th International Conference on Pattern Recognition*, pages 425–427, 1998.

[14] R. E. Kalman. A New Approach to Linear Filtering and Prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[15] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, 2005.

[16] S. J. Julier and J. K. Uhlmann. A New Extension of the Kalman Filter to Nonlinear Systems. In *In The Proceedings of AeroSense: The 11th International Symposium on Aerospace/Defense Sensing, Simulation and Controls, Multi Sensor Fusion, Tracking and Resource Management II*, 1997.

[17] P. Lang and A. Pinz. Calibration of Hybrid Vision / Inertial Tracking Systems. In 2^{nd} *InverVis 2005: Workshop on Integration of Vision and Inertial Systems*, Barcelona, April 2005.

[18] J. Lobo and J. Dias. Relative Pose Calibration Between Visual and Inertial Sensors. In *ICRA 2005 Workshop on Integration of Vision and Inertial Sensors (InerVis2005)*, 2005.

[19] U. Frese. Camera Sensor Model, 2007. Not published.

[20] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2004.

[21] Z. Zhang. A Flexible New Technique for Camera Calibration. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 22, pages 1330–1334, 2000.

[22] E. Salamin. Application of Quaternions to Computation with Rotations. Technical report, Stanford AI Lab, 1979.

[23] B.K.P Horn. Closed-Form Solution of Absolute Orientation using Unit Quaternions. *Journal of the Optical Society of America*, 4(4):629–642, April 1987.

[24] U. Frese. Local Marker Detector, 2007. Not published.

[25] J.-Y. Bouguet. Camera Calibration Toolbox for Matlab, 2007. `http://www.vision.caltech.edu/bouguetj/calib_doc/`. Accessed January 24, 2008.

[26] J. Wesson. *The Science of Soccer*. IOP Publishing Limited, 2002.

[27] R. G. Watts and R. Ferrer. The Lateral Force on a Spinning Sphere: Aerodynamics of a Curveball. *American Journal of Physics*, 55:40–44, January 1987.

[28] F. S. Grassa. Practical Parameterization of Rotations Using the Exponential Map. *The Journal of Graphic Tools*, 3.3, 1998.

[29] A. Dempster, N. Laird, and D. Rubin. Maximum Likelihood From Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society*, 1977.