Technische Universiteit Eindhoven University of Technology

4WC00: Bachelor final project

Online mounting calibration around the vertical axis of the Kinect sensor

Student:	J.M. van Willigen
Identity number:	0889156
Project coordinator:	dr.ir. M.J.G. van de Molengraft
Coach:	ir. Y.G.M. Douven
Department:	Mechanical Engineering
Research group:	Control Systems Technology

Eindhoven, 7th July 2017

Contents

List of symbols

Symbol	**Quantity**	**Unit**	**Abbreviated Unit**
x	Distance in the x-direction	meter	m
y	Distance in the y-direction	meter	m
z	Distance in the z-direction	meter	m
θ	Rotation around the z-axis	Radian	Rad
r	Distance in the r-direction	meter	m
σ	Standard deviation	-	-
$\mathbf{B}$	Position vector ball	[meter, meter, meter]	[m, m, m]
$\mathbf{K}$	Position vector Kinect	[meter, meter, meter]	[m, m, m]
$\mathbf{A}$	State transition model	-	-
$\mathbf{B}$	Control-input model	-	-
$\mathbf{C}$	Observation model	-	-
$\mathbf{x}$	state	-	-
$\mathbf{z}$	Measurement	-	-
$\mathbf{u}$	Control-input	-	-
$\mathbf{w}$	Process noise	-	-
$\mathbf{v}$	Measurement noise	-	-
$\mathbf{Q}$	Process covariance	-	-
$\mathbf{R}$	Measurement covariance	-	-
$\mathbf{P}$	Place covariance	-	-
$\mathbf{K}$	Kalman gain	-	-

Subscript	**Meaning**
i	time
r	with regard to the robot coordinate frame
k	with regard to the Kinect coordinate frame
o	with regard to the Omnivision coordinate frame
e	with regard to the expected Kinect coordinate frame
d	difference between Omnivision and Kinect

1 Introduction

Tech united is a team from TU/e specialized in robotics and artificial intelligence [1]. The team participates in international tournaments all around the world in the Robot Soccer World Cup (RoboCup) [2]. RoboCup is a competition for robots that can react to and communicate with an ever-changing environment. A subteam of Tech United participates with much success in the Middle Size League, where teams of autonomous wheel-based soccer robots compete against each other [3]. The robots are required to communicate with each other, independently make decisions and anticipate on the possible future. In order to do so, sensing the environment and its own place in it is essential.

The robots of the Tech United soccer team are called TURTLEs. A key element in playing soccer is locating the ball. For this the TURTLEs are equipped with two vision sensors: the Omnivision camera and the Kinect camera. With the Omnivision camera the TURTLEs are able to get a 360° image of their environment and the Kinect camera gives a 3D image of what is in front of the TURTLE. The information of these sensors should, when fused, give an accurate position of the ball. When the Kinect is mounted on a TURTLE, a mounting calibration of the Kinect camera on the Omnivision is performed. However, in the current situation, the two sensors over time give increasingly differing ball positions, as a result of dislocation of the Kinect obtained during matches, transport, etc.

A possible solution for this problem is to do an online mounting calibration, meaning a calibration during a game. For the calibration of the rotation around the horizontal axes, such a calibration has already been developed, but not yet implemented. This calibration is based on using the ground as a reference [4]. However, for the vertical axis a similar solution is not possible. Therefore this project has the following goal:

Develop an online mounting calibration method of the rotation around the vertical axis of the Kinect on the Omnivision.

For this calibration the position of the ball will be used as a reference.

This report will focus on how this goal is achieved in seven further chapters. In Chapter 2 the current situation will be sketched. In Chapter 3 the approach for the online calibration will be explained. In order to implement this approach a data analysis and a theoretical research about the Kalman filter is performed in respectively Chapter 4 and Chapter 5. In Chapter 6 the actual implementation will be elaborated on. In Chapter 7 the results of the implementation will be presented and in the last chapter conclusions and recommendations will be made.

2 Current situation

2.1 The game

In the RoboCup MSL league teams of five fully autonomous robots play soccer with a yellow regular size ball. The game consists of two halves of 15 minutes. There are certain constraints for the robots such as size and weight. The game is similar to soccer and carries similar rules such as Offside. However, it also carries slightly different rules, e.g. ‘a valid goal can only be scored after the ball has been received or touched by a team mate within the opponent side of the field.’ [5] To enforce the Laws of the Game a Referee is appointed to every match. The Referee communicates with the robots via the RefBox. [5]

The RefBox handles moments where the ball went out of play or a fault has been made. These moments are called RefBox tasks, e.g. the kickoff, penalty and corners [6]. These RefBox tasks are all similarly structured. The RefBox first communicates a stop signal, after which all robots have to stop moving. The ball is then placed on the correct spot by the Referee. Then the RefBox communicates the type of Refbox task. From that moment the robots are allowed to position themselves. Subsequently the RefBox communicates the start signal after which the attacker may kick the ball. [5]

2.2 The TURTLE

The robots of Tech United are called TURTLEs. A TURTLE with corresponding coordinate frame is shown in Figure 2.1. The rotation around the z-axis is defined with θ.

The TURTLE should be able to perform several tasks. Firstly, the TURTLE has an active ball handling system, to control the ball when dribbling. Secondly, the robot is equipped with a shooting mechanism driven by a solenoid. The TURTLE drives around using omniwheels. These are wheels, which make it possible to move in every direction in the horizontal plane. But, most important for this project is how the TURTLE senses its environment. For this the robot uses the Omnivision and Kinect camera.

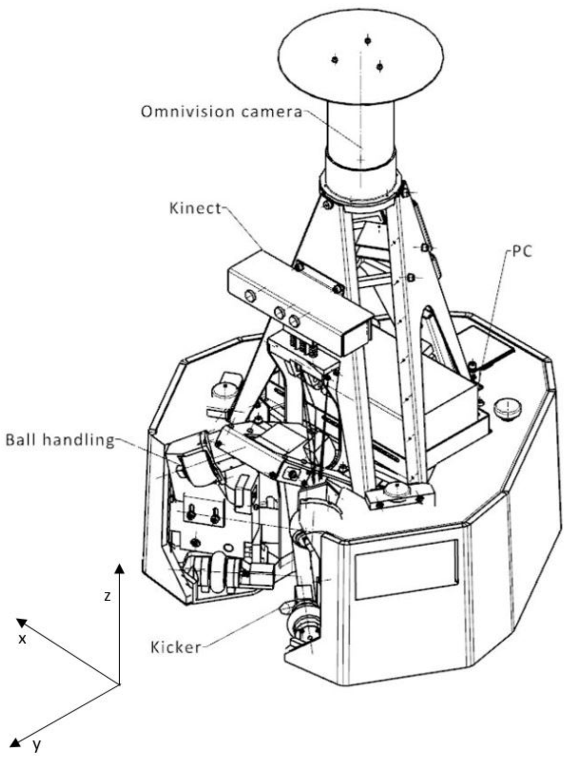


Figure 2.1: *A TURTLE with corresponding coordinate frame. The coordinate frame's origin should actually be exactly beneath the center of the robot*

The TURTLE's software mostly runs on a CPU. However the Kinect runs on a Jetson [7]. This is a computation processor board. The communication between the CPU and the Jetson is done over a communication bus, resulting in a small delay.

2.3 The Omnivision & Kinect camera

The Omnivision camera is a prosillica GC 780 camera pointed directly at a conical mirror, making it possible to get a 360° view around the TURTLE. It can run on 64 Hz [8]. The Kinect V2 camera is a camera by Microsoft which is able to generate a 3D image and can run on 30 Hz [9]. The corresponding coordinate frames of the sensors do not coincide and should be calibrated. At present, this mounting calibration is only performed when a Kinect is newly mounted on a TURTLE. As a result of dislocations of the Kinect obtained during games or transport, the calibration gets less accurate over time. This effect is so significant, that currently, during a game the Kinect sensor data is unused. To solve this, an online mounting calibration for the rotation around the x-axis and the y-axis is already developed based on using the ground as a reference [4]. Before this project a calibration method for the rotation around the z-axis did not yet exist.

2.4 Descriptive data analysis

This project focuses on two sensors: the Kinect and the Omnivision. The data analyzed in this chapter is the location of the ball as measured by both these sensors. This data analysis gives answers to the following questions: what does the data generally look like? What is the measure space? Do the sensors find a lot of false positives or are there any other outliers?

Descriptive analysis of the Kinect

The first conducted measurement session was very straightforward. The TURTLE was put on the field facing a certain direction. Then the ball was placed exactly at the foot of the TURTLE. The Protonect script was ran for about 15 seconds after which the ball was replaced 25 cm further from the TURTLE. This process went on until the Kinect was not able to localize the ball anymore. Then, a similar process was started. However, instead of varying the y-position of the ball, the ball was replaced in x-direction. This was done at two different distances from the TURTLE: approximately one meter and two meters.

After conducting the measurements, the acquired data, in the form of (x, y, z), was processed in Matlab. In Figure 2.2 the measured y-position is plotted against the measured x-position. The outliers have already been filtered out. Also, per data set a covariance ellipse is plotted over the figure, as best seen in Figure 2.3. How these covariance ellipses are determined is explained in Appendix B

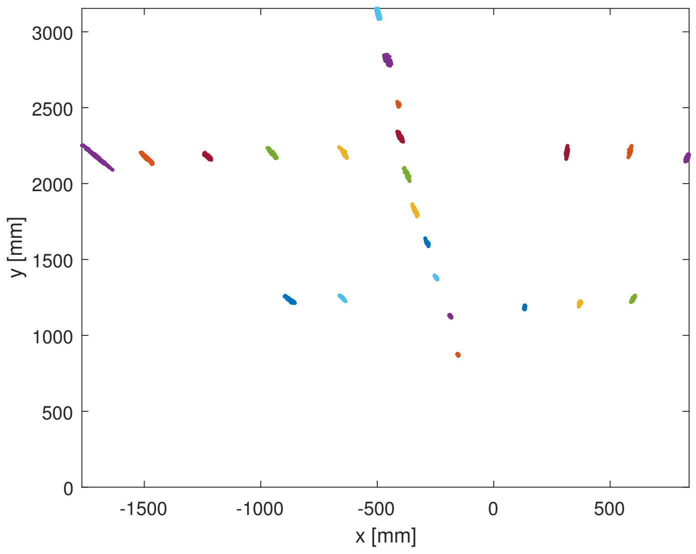


Figure 2.2: *A plot of the y-measurements against the x-measurements of the Kinect, measurement session 1*

From this figure we can get an idea of the measure space. The angle range in which a ball can be detected is approximately $-35° - 35°$. According to this data balls start being visible for the Kinect from about 70 cm in front of the TURTLE to a bit over 3 m. However, from people having experience with the Kinect, it was known the Kinect could actually see balls from further away. Another interesting observation about Figure 2.2 is that although the first 10 balls were all placed directly in front of the TURTLE, these measurements deviate pretty strongly from the line $y = 0$, indicating an offset in the Kinects θ.

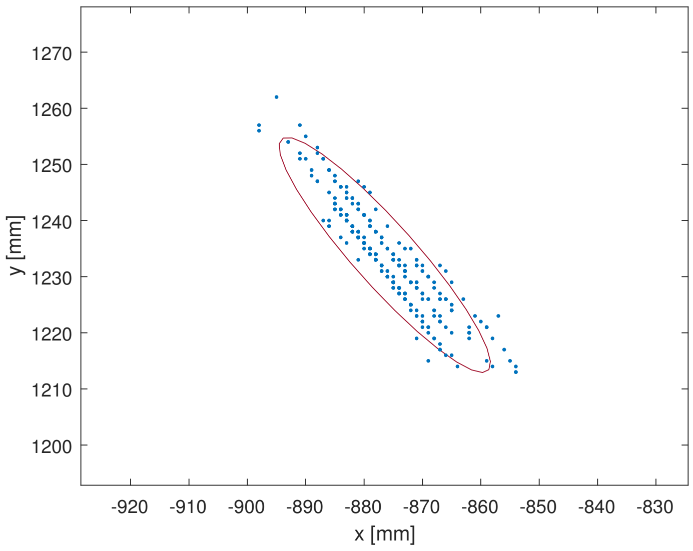


Figure 2.3: *A close-up of one of the measurement sets of figure 2.2*

From both Figure 2.2 and Figure 2.3 it can be seen that the longest covariance axis is pointed towards the origin, or towards the TURTLE. Therefore, it was decided to, later in this report, conduct a new, exploratory analysis in which to find a relation between the position of the ball and the variance of θ. For this analysis a measurement session consisting of more measurements and more random measurements is needed. Another phenomenon noticeable in Figure 2.3 is that the measurements all lay on lines originating from the TURTLE. This is caused by the resolution of the Kinect, which does not directly measure x, y and z.

Descriptive analysis of the Omnivision

The measurement session of the Omnivision was done more randomly. It consists of 47 measurement sets. Both the position and orientation of the ball and the TURTLE were arbitrarily chosen for every set. This was to get rid of external influences such as unevenly distributed light. Every measurement set again took about 15 seconds. Only measurements of the ball in the front half of the TURTLE are performed. Even though the Omnivision camera can see balls 360° around itself, only measurement sets where the ball is in front of the TURTLE are used. After all, the calibration will only be performed on balls in front of the TURTLE. The sets are plotted in Figure 2.4. Again the outliers are filtered out and covariance ellipses are plotted over the figure. A close up of one of the measurements can be seen in Figure 2.5

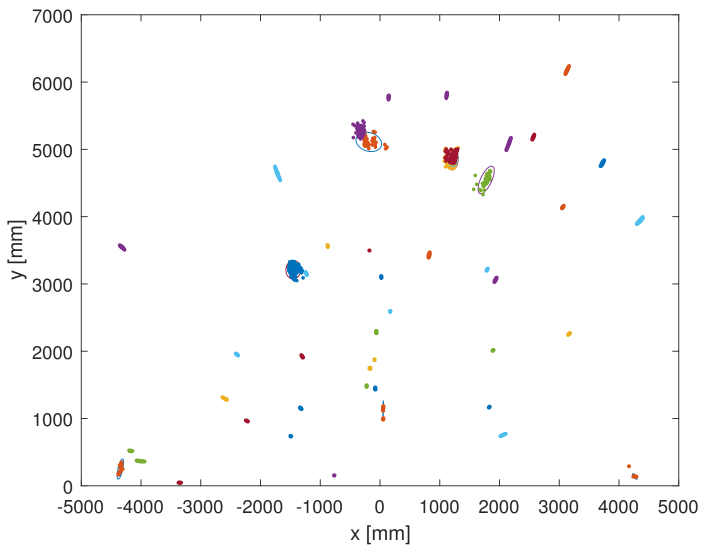


Figure 2.4: *A plot of the y-measurements against the x-measurements of the Omnivision*

From the figure we can determine that the range of the Omnivision is up to approximately 6 m.

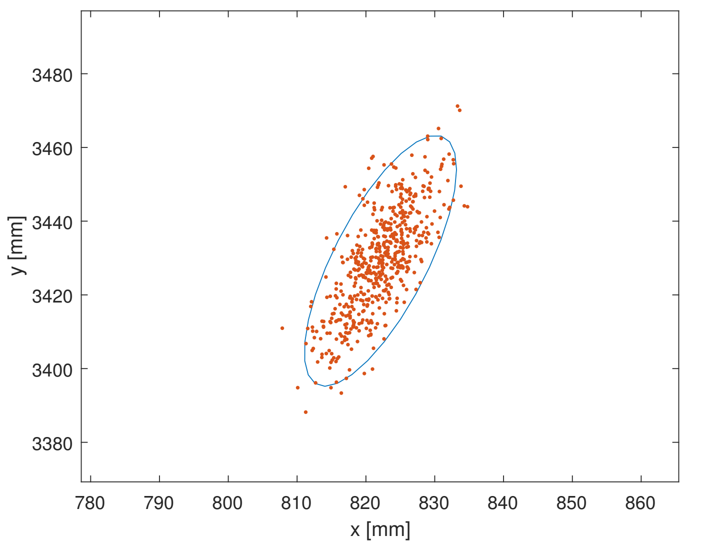


Figure 2.5: *A close up of one of the measurement sets from Figure2.4*

Similar to the Kinect measurements, the longest covariance axis points towards the TURTLE. Also for the Omnivision, later in this report, an exploratory analysis for the Omnivision is conducted, to try and find a relation between the position of the ball and the variance of θ.

3 Approach

The coordinate θ_k of the local coordinate frame of the Kinect does not correspond to the coordinate θ_r of the coordinate frame with regard to the robot. The difference in orientation around the z-axis of the Kinect's coordinate frame and robot's coordinate frame is defined as θ_d and needs to be compensated for. The coordinate θ_o of the coordinate frame with regard to the Omnivision is assumed to correspond to θ_r, thus can be used for calibration. This calibration is now only performed once, when the Kinect is newly mounted on the TURTLE. However, to adjust for dislocations of the Kinect that are obtained during the match, a live calibration is required. To do this the position of the ball can be used.

3.1 Challenges

The online calibration brings some challenges along:

- There is a delay between the communication of the Jetson, on which the Kinect runs, and the robot. When the ball is moving, this will result in the Kinect and the Omnivision measuring a different position of the ball.
- The frame of the Kinect and the Omnivision do not coincide in the origin. Therefore, it is not possible to simply correct for the difference between the Kinect and Omnivision measurements.
- Both the Kinect and the Omnivision camera have some inaccuracy.
- The online calibration needs to correct for dislocations of the Kinect obtained during a match. These dislocations tend to occur mainly at collisions between robots.
- Sometimes both or one of the sensors sees a ball where there is none: a false-positive. It also happens that multiple balls are seen, where there is supposed to be only one.
- The ball is not seen by both cameras at every moment during the game. This can be caused by other robots blocking the view or simply because the ball is not in the measurement space.

3.2 Solutions

The problem of delay between the Jetson and the TURTLE is tackled by only looking at moments, where it is known that the ball has a constant value of θ_r. This occurs during RefBox tasks when the ball is laying still.

The second challenge is solved by translating the Omnivision frame to the Kinect frame, which will be explained later in this chapter.

The noise caused by the inaccuracy of the sensors can be filtered out by using a Kalman filter on the difference between the Kinect and the Omnivision measurements.

In order to adjust for dislocations obtained by collisions, it is important to detect these collisions. This can be done using the accelerometer with which the TURTLEs are equipped. On a collision the values of the accelerometer naturally peak. When a collision is detected, a trick with the Kalman filter is performed: the error covariance is reset.

When there are multiple possible balls, the one which corresponds best with the other sensor is evaluated further. However, if all the possible balls seem to be false-positives, they will be treated as a truncated sample, thus they will not be used for the calibration.

Because the ball is not always seen by both cameras and because the calibration can only be performed when the ball has a constant value of θ_r the effective calibration time during a match gets scarcer. It is validated in Appendix C that the effective calibration time is long enough by doing a Greenfield analysis of several real games played by Tech United.

3.3 Elaboration

In this section the solutions will be elaborated on further.

Goniometry

In Figure 3.1 a basic representation of the situation is depicted.

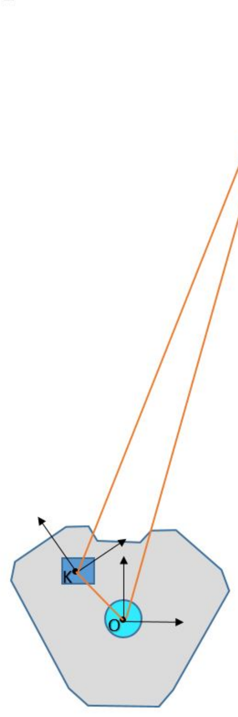


Figure 3.1: *A situation sketch of a TURTLE with Kinect and Omnivision measuring a ball*

Because the origin of the Kinect frame and the Omnivision frame do not coincide, it is not possible to just take the difference between the measured θ_k and θ_o. The position of the Omnivision and the Kinect are known and the orientation of the Omnivision around the vertical axis is assumed correct. Therefore it is possible to calculate the expected ball position of the Kinect sensor from the output of the Omnivision camera by simply translating the Omnivision frame to the origin of the Kinect frame.

The expected ball position and the ball position as measured by the Kinect sensor can be expressed as two vectors: $\mathbf{B}_e$ and $\mathbf{B}_k$. The difference in orientation around the z-axis, θ_d can be determined by combining Equation 3.1-3.2 to form Equation 3.3.

$$|\mathbf{B}_k \cdot \mathbf{B}_e| = |\mathbf{B}_k||\mathbf{B}_e| cos(\theta_d) \tag{3.1}$$

$$|\mathbf{B}_k \times \mathbf{B}_e| = |\mathbf{B}_k||\mathbf{B}_e| sin(\theta_d) \tag{3.2}$$

$$\theta_d = atan2(|\mathbf{B}_k \times \mathbf{B}_e|, |\mathbf{B}_K \cdot \mathbf{B}_e|) \tag{3.3}$$

The outcome is used as input for the Kalman filter.

Collision detection

Collision detection is performed using the accelerometer of the TURTLE. This sensor outputs three values: three different components of the proper acceleration. By taking the absolute value of the proper acceleration vector and putting a threshold on it, a collision can be detected. In Figure 6.1 the absolute value of the proper acceleration is plotted against time.

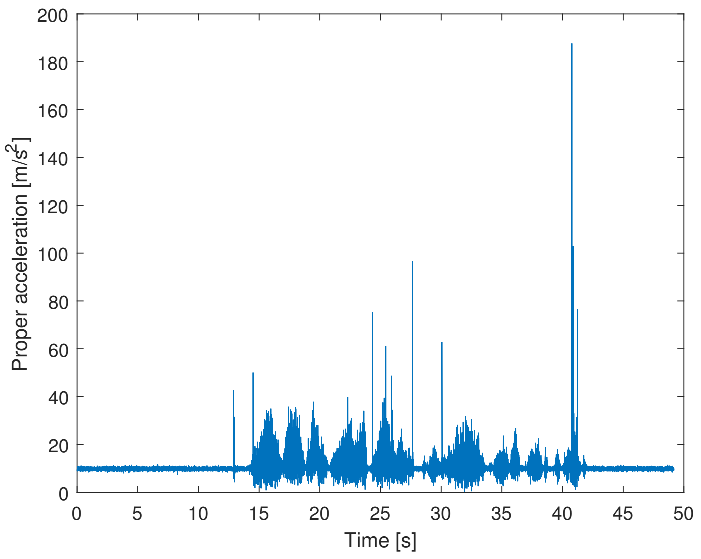


Figure 3.2: *A plot of the absolute value of the proper acceleration versus time. A collision occurs at* ±42 *seconds*

At 42 seconds the TURTLE hits another robot, relatively softly. In the figure it shows a notable peak. Before this collision the TURTLE was made to move around as violently as possible, but nowhere the proper acceleration came close to the collision peak. From this information the threshold, from which we detect a collision, is chosen on 160 m/s^2.

Kalman filter [10]

To get rid of measurement noise, it is decided to use a Kalman filter with state θ_d. In theory the Kalman filter is the best possible filter possible, since it minimizes the state error. However, in order for the Kalman filter to have the property of being optimal, the following conditions must be applicable to the system:

- The system can be described by a linear system model
- The system and measurement noises entering the system are white
- The system and measurement noises entering the system are Gaussian

Luckily, the Kalman filter also gives a good estimate when these three conditions are closely approximated. This applies for the proposed system. To find the covariance of the noise, an exploratory data analysis is performed in Chapter 4.

Thus, the first reason to choose for the Kalman filter is because a lot of information about the system can be determined. Another reason to use the Kalman filter is because the filters inputs and outputs are in the form of covariances and therefore remain insightful. This also makes the filter easily resettable on a collision, by resetting the state covariance.

To be able to implement the Kalman filter, it is important to have a proper understanding of it. To that extend, Chapter 5 is dedicated to the theory of the Kalman filter.

4 Exploratory data analysis

This exploratory data analysis gives an answer to the following question: what is the accuracy with which the ball can be detected and how does this accuracy depend on the position of the ball? This is important information for the implementation of the Kalman filter.

In Appendix A an exploratory analysis focused on the distance coordinate can be found. However, this was outside of the scope of this project.

4.1 Exploratory analysis of the Kinect

For the exploratory analysis the measurement session was performed as randomly as possible: both the TURTLE and the ball were placed arbitrarily on the Robot soccer field. Then the measured position of the ball was logged for about 15 seconds, creating a data set. This was done 70 times. The results can be seen in Figure 4.1.

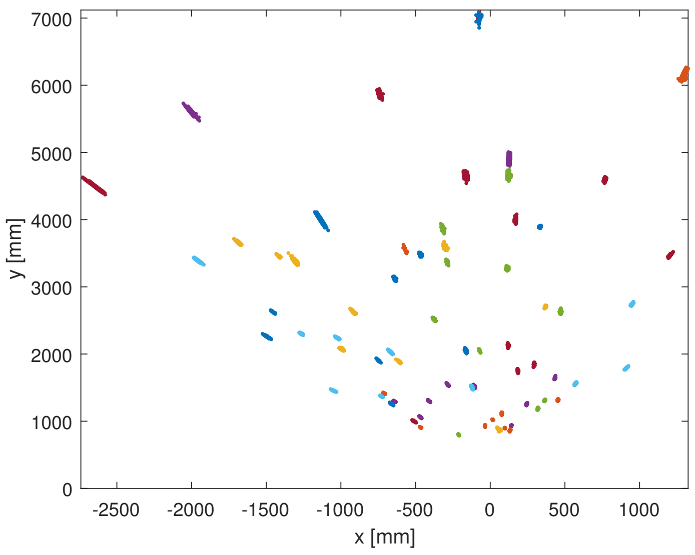
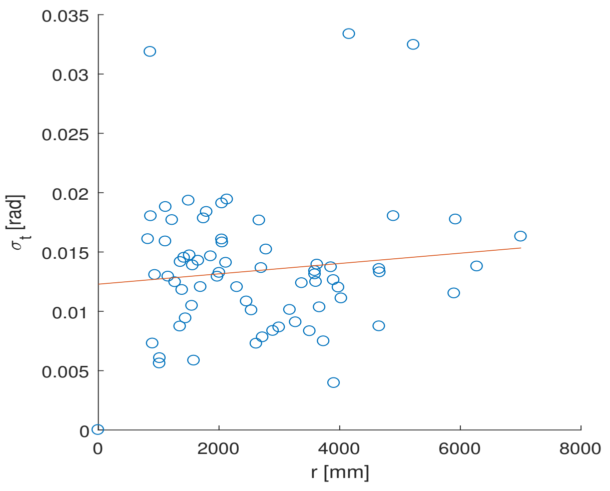


Figure 4.1: *A plot of the y-measurements against the x-measurements of the Kinect, measurement session 2*

In order to find a relation between the position and the standard deviation of θ, σ_t, the position is translated to a polar coordinate system (r, θ). Subsequently, σ_t is plotted versus these two coordinates, see Figure 4.2 a and b. In these figures a first order polynomial is fitted through the data points.



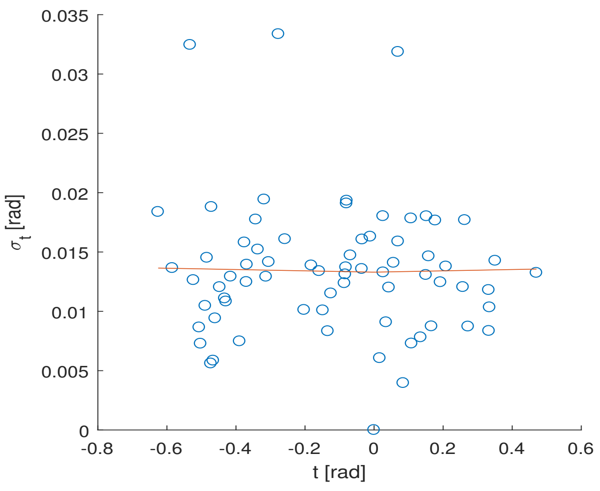
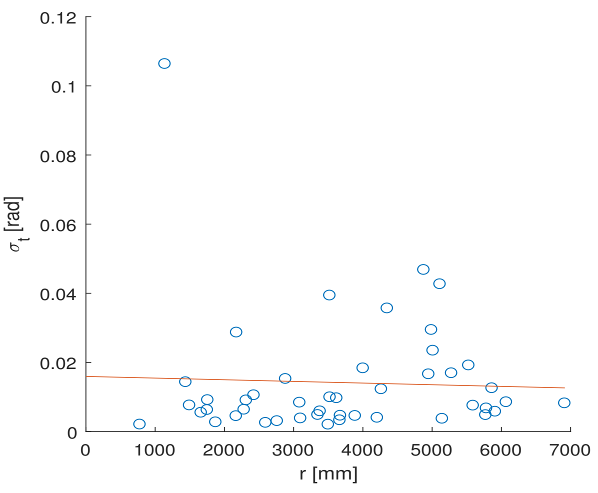


Figure 4.2: *The relation between σ_t and the location of the ball (r, θ) measured by the Kinect: r left, θ right*

It can be concluded that there is no real dependency of σ_t on either θ or r. This standard deviation varies around 0.014 rad or a variance of $2.0 * 10^{-4}$ rad^2.

4.2 Exploratory analysis of the Omnivision

For the exploratory analysis of the Omnivision the same measurement session is used as for the descriptive analysis from Section 2.4/ Again, the same plots as in the previous part are constructed, resulting in Figure 4.3.



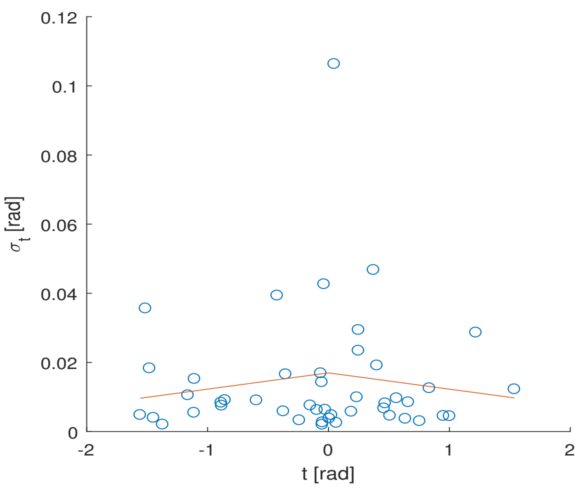


Figure 4.3: *The relation between σ_t and the location (r, θ) of the Omnivision: r left, θ right*

It can be concluded that there is no real dependency of the standard deviation of θ on either θ or r. This standard deviation varies around 0.018 rad or a variance of $3.2 * 10^{-4}$ rad^2.

5 Kalman filter theory

The Kalman filter is a powerful algorithm that produces estimates of a signal, containing noise and other inaccuracies. The outputs of the Kalman filter tend to be more accurate than those based on a single measurement. This chapter is based on [11] and [12].

5.1 The process

The Kalman filter works with a linear discrete-time system, as represented in Equation 5.1 and 5.2.

$$\mathbf{x}_i = \mathbf{A}\mathbf{x}_{i-1} + \mathbf{B}\mathbf{u}_i + \mathbf{w}_i \tag{5.1}$$

$$\mathbf{z}_i = \mathbf{C}\mathbf{x}_i + \mathbf{v}_i \tag{5.2}$$

with $\mathbf{x}_i$ being the state, $\mathbf{u}_i$ being the input and $\mathbf{z}_i$ being the measurement. The random variables $\mathbf{w}_i$ and $\mathbf{v}_i$ respectively represent the system noise and the measurement noise according to Equation 5.3 and 5.4.

$$\mathbf{w}_i \sim N(0, \mathbf{Q}) \tag{5.3}$$

$$\mathbf{v}_i \sim N(0, \mathbf{R}) \tag{5.4}$$

The process covariance $\mathbf{Q}$ and the measurement covariance $\mathbf{R}$ can change with every time step, but in this case they are assumed to be constant. This corresponds to the process of this project, as investigated in Chapter 4: Exploratory data analysis.

5.2 The algorithm

The algorithm loops through two phases for every time step: the time update and the measurement update. The time update estimates the current state based on previously acquired information about the system. It also calculates the corresponding uncertainty. This state and uncertainty is called the a priori estimate and the a priori covariance, respectively represented by $\hat{\mathbf{x}}_{i|i-1}$ and $\mathbf{P}_{i|i-1}$. The measurement update then incorporates a new measurement into the a priori estimate to determine a more accurate a posteriori estimate and covariance: $\hat{\mathbf{x}}_{i|i}$ and $\mathbf{P}_{i|i}$. This process is schematically depicted in Figure 5.1.

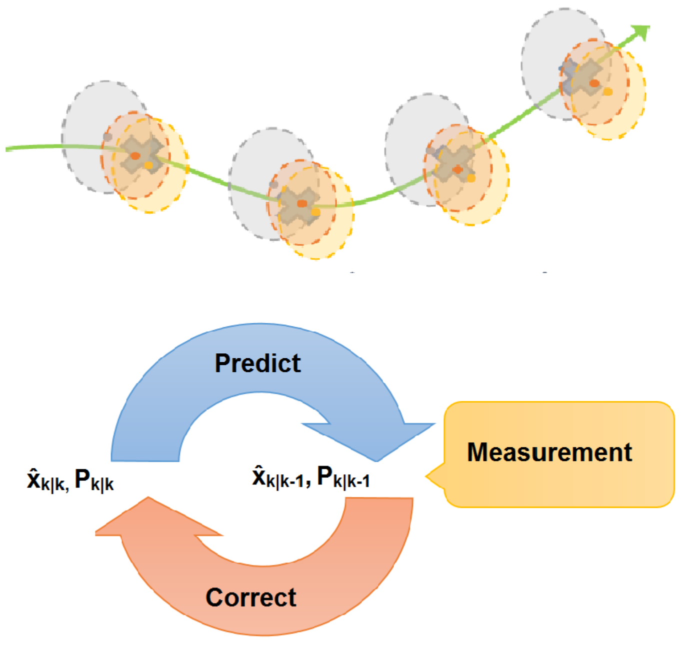


Figure 5.1: *The Kalman filter cycle; the real state is represented by the green line with the blue crosses. The a priori state is represented with blue, the measurement with yellow and the a posteriori state with red. [12]*

The equations for the time update are shown in Equation 5.5 and 5.6.

$$\hat{\mathbf{x}}_{i|i-1} = \mathbf{A}\hat{\mathbf{x}}_{i-1|i-1} + \mathbf{B}\mathbf{u}_i \tag{5.5}$$

$$\mathbf{P}_{i|i-1} = \mathbf{A}\mathbf{P}_{i-1|i-1}\mathbf{A}^T + \mathbf{Q} \tag{5.6}$$

The equations use the **A** and **B** from Equation 5.1 and the **Q** from Equation 5.3 to determine the a priori state and covariance. The equations for the measurement update are shown in Equation 5.7-5.9.

$$\mathbf{K}_i = \mathbf{P}_{i|i-1}\mathbf{H}^T(\mathbf{H}\mathbf{P}_{i|i-1}\mathbf{C}^T + \mathbf{R})^{-1} \tag{5.7}$$

$$\hat{\mathbf{x}}_{i|i} = \hat{\mathbf{x}}_{i|i-1} + \mathbf{K}_i(\mathbf{z}_i - \mathbf{C}\hat{\mathbf{x}}_{i|i-1}) \tag{5.8}$$

$$\mathbf{P}_{i|i} = (\mathbf{I} - \mathbf{K}_i\mathbf{C})\mathbf{P}_{i|i-1} \tag{5.9}$$

Firstly, the Kalman gain $\mathbf{K}_i$ is calculated. This is a gain that minimizes the error in the a posteriori state by balancing how much the measurement is trusted versus how much the a priori state is trusted. Then, with this Kalman gain, the a posteriori state and covariance can be determined.

6 Implementation

The code is implemented in the main loop of the script 'Protonect.cpp'. It goes through the following sequence for every loop:

1. Collision detection
2. RefBox check
3. Ball selection and transformation
4. Calculate θ_d
5. Kalman implementation
6. Correction

6.1 Collision detection

The collision detection is performed on the robot and when a collision is detected a counter variable is updated. This counter is sent to the Protonect script on the Jetson, where it is compared to a tracker variable. When the counter differs from the tracker, the state covariance is reset and the tracker is updated.

6.2 RefBox check

For the calibration method to work, the θ_r-coordinate of the ball must be constant. A moment in the game this happens is during a RefBox task, after the TURTLE has positioned itself and before the start signal. Step 3 to 6 are only performed if this is the case.

6.3 Ball selection transformation

The input of the script consists of two possible balls as detected by the Omnivision camera and one possible ball from the Kinect sensor. However, before using these balls for the calibration, it needs to be verified whether or not these possible balls represent the real ball. Firstly, it is checked if both the Kinect and Omnivision detect a possible ball. Secondly, the detected Omnivision balls are expressed in a coordinate frame, which is translated from the Omnivision to the Kinect. Then, the possible Omnivision ball that is closest to the possible Kinect ball is selected, after which it is checked if the selected Omnivision ball and Kinect ball are within a certain range of each other.

6.4 Calculate θ_d

After verifying the 'realness' of the detected balls, the rotation around the z-axis can be determined according to Equation 3.3. The outcome is used as input for the Kalman filter.

6.5 Kalman implementation

The implementation of the Kalman filter for this situation is not particularly difficult: the system with state $\hat{\theta}_d$ can be described according to Equation 5.1 and 5.2, with **A**=1, **B**=0 and **C**=1. The measurement covariance, **R**, is attainable from Chapter 4. The process covariance, **Q**, is taken 0, since the system is rigid and assumed to only change as a result of a collision. This results in the state covariance, **P**, approaching 0 until a collision is detected. This process is visualized in Figure 6.1.

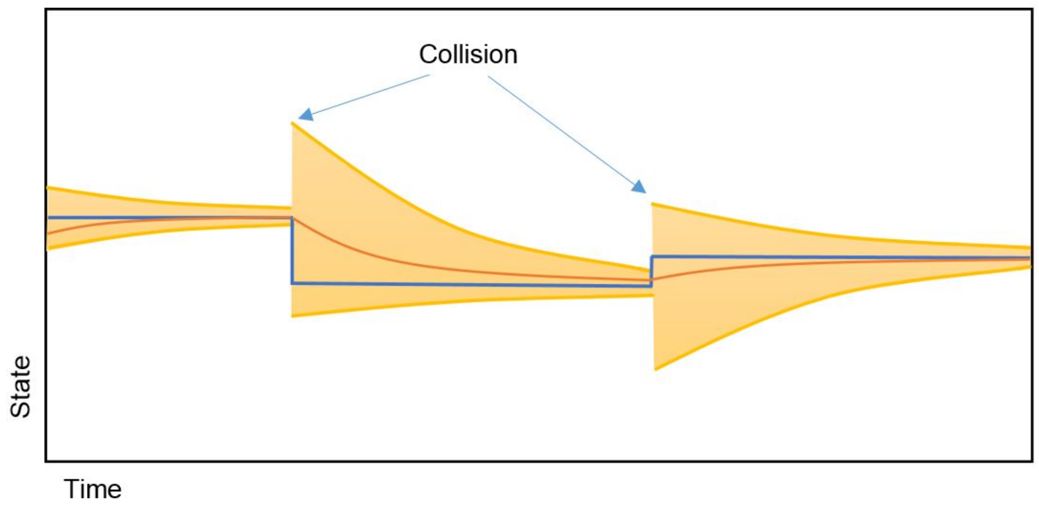


Figure 6.1: *The functioning of the Kalman filter when collisions happen; yellow represents the state covariance, red represents the state approximation by the Kalman filter and blue represents the actual state.*

6.6 Correction

By using the output of the Kalman filter, the ball as measured by the Kinect can easily be expressed in the robot's coordinate frame. This is done with a translation and a rotation matrix, according to Equation 6.1.

$$\begin{bmatrix} \mathbf{B}_r(x) \\ \mathbf{B}_r(y) \\ \mathbf{B}_r(z) \\ 1 \end{bmatrix} = \begin{bmatrix} cos(\hat{\theta}_d) & -sin(\hat{\theta}_d) & 0 & 0 \\ sin(\hat{\theta}_d) & cos(\hat{\theta}_d) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & \mathbf{K}_r(x) \\ 0 & 1 & 0 & \mathbf{K}_r(y) \\ 0 & 0 & 1 & \mathbf{K}_r(z) \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{B}_k(x) \\ \mathbf{B}_k(y) \\ \mathbf{B}_k(z) \\ 1 \end{bmatrix} \tag{6.1}$$

$\mathbf{K}_r$ and $\mathbf{B}_r$ are respectively the position of the Kinect and the ball expressed in the robot's coordinate frame.

7 Verification

In this chapter the implementation will be tested in two scenarios. The first will test whether the online calibration works when the ball is laying still and whether it acts as supposed after a collision happens. The second will test whether the online calibration also works when shooting or passing a ball.

7.1 Case 1: ball laying still with a collision

For case 1, an experiment consisting of eight consecutive steps was conducted.

1. The TURTLE and ball are both placed on the field with a screen, blocking the line of sight between them.
2. The Protonect script is activated.
3. The screen is removed, thus starting a calibration.
4. The screen is put back, resulting in the calibration being stopped.
5. Another robot collides (relatively softly) with the TURTLE.
6. The screen is removed, initializing another calibration
7. The screen is put back, stopping the calibration.
8. The Protonect script is deactivated.

The corresponding output of the Kalman filter is plotted versus time in Figure 7.1. This includes $\hat{\theta}_d$ and a shaded error bar around it.

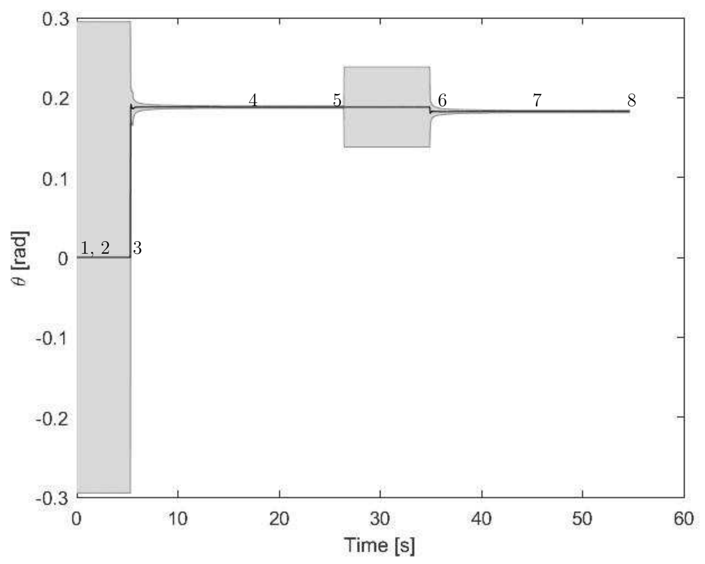


Figure 7.1: *A plot of the Kalman output versus time. This output includes estimated state and standard deviation visualized as a shaded error bar; case 1.*

The Kalman filter's output is in confirmation with the expected output: the estimate starts at zero with a big uncertainty. After the removal of the screen, the estimate quickly converges to a value with a quickly decreasing uncertainty. After collision the uncertainty is set to a bigger value and when a calibration starts the estimate again converges, however to a slightly different value. To check whether the $\hat{\theta}_d$ is accurate in Figure 7.2 the measured Omnivision is compared with the corrected Kinect measurement. For reference the real Kinect measurement has also been included in the plot.

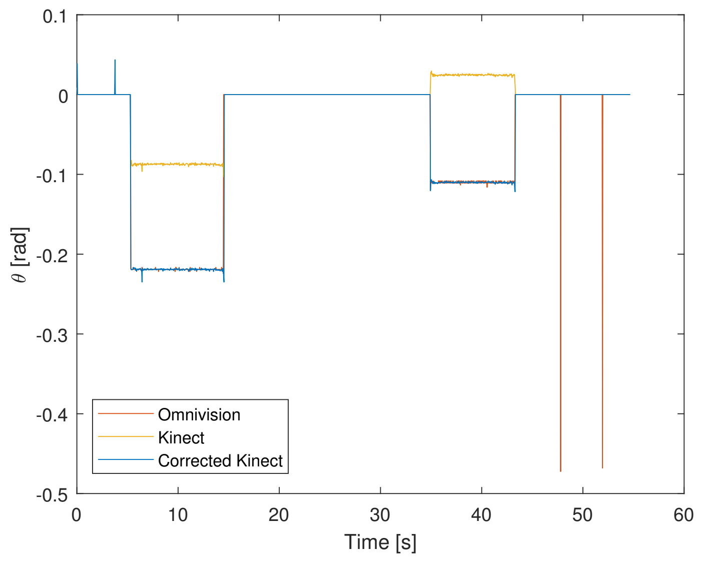


Figure 7.2: *A plot of the Omnivision and Kinect measurement and the corrected Kinect measurement.*

In the figure it is clearly visible, the Kalman filter gives an accurate $\hat{\theta}_d$. However, the output of the Protonect script actually is the location of the ball in cartesian coordinates. For the Omnivision, Kinect and corrected Kinect, x- and y-value are plotted versus time in Figure 7.3.

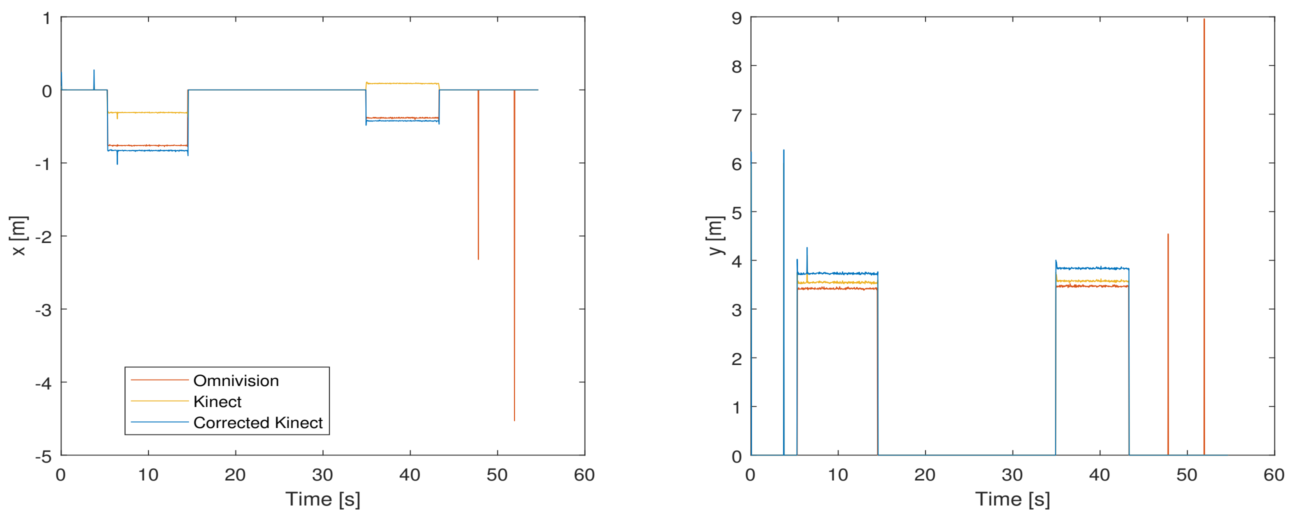


Figure 7.3: *A plot of the Omnivision and Kinect measurement and the corrected Kinect measurement in cartesian coordinates, left x, right y.*

It is noticeable that for the x-coordinate the Omnivision and corrected Kinect closely approximate each other. Yet this does not apply to the y-coordinate. However, because the measurement was executed with the ball almost exactly in front of the TURTLE, this problem is not likely to originate from the performed calibration, but from either the Kinect or Omnivision giving a inaccurate distance to the ball.

7.2 Case 2: shooting/passing the ball

When shooting or passing the ball, its θ_r-coordinate remains approximately constant. Therefore a second experiment was conducted in which it was tested whether the implementation would also work when shooting or passing the ball:

1. The ball is placed in the ball handler of the TURTLE.
2. The Protonect script is activated.
3. The ball is shot/passed.
4. The Protonect script is deactivated.

The output of the Kalman filter is again plotted versus time in Figure 7.4. Unfortunately, this method does not seem to work for shooting. The Omnivision and Kinect see the ball at the same time only twice and the output does not correspond at all to the expected ±0.2 rad. A pass has a lower ball speed, resulting in more calibration time. Therefore the same experiment was conducted for a pass. The results are shown in Figure 7.5.

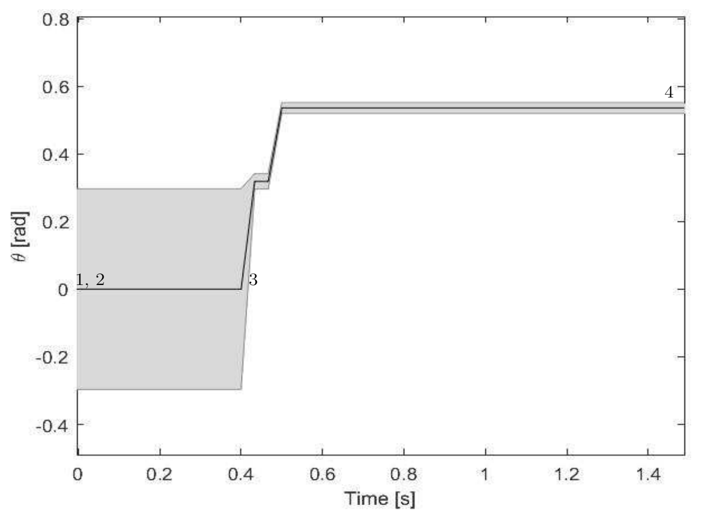


Figure 7.4: *A plot of the Kalman output versus time. This output includes estimated state and standard deviation visualized as a shaded error bar; case 2, shot.*

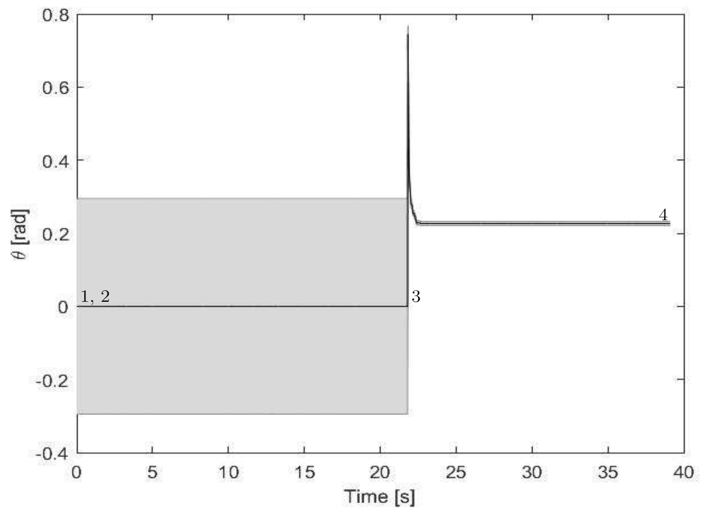


Figure 7.5: *A plot of the Kalman output versus time. This output includes estimated state and standard deviation visualized as a shaded error bar; case 2, pass.*

The method works better for passing, but still gives an inaccurate $\hat{\theta}_d$. Right after the ball has been shot, the estimate is completely off. When the ball gets further away from the TURTLE a better estimate is established, however still with a small offset. This is caused by the fact that the calculation of θ_d is performed between two coordinate systems with the origin at the Kinect. The ball originates from a slightly different place. Because the Omnivision measurement comes with a delay, this results in an offset in θ_d, especially when the ball is close to the TURTLE.

8 Conclusions & Recommendations

At the start of this report the goal of the project was defined as:

Develop an online mounting calibration method of the rotation around the vertical axis of the Kinect on the Omnivision.

In this chapter conclusions about the functioning of this method are discussed and recommendations are made based on knowledge gained throughout the project.

8.1 Conclusions

Firstly, it can be concluded that the designed method works well for balls laying still. The θ of the corrected Kinect corresponded closely to the θ of the Omnivision. However, the distance of the ball derived from the corrected Kinect data was significantly different from the distance of the ball derived from the Omnivision data.

The calibration does not seem to work well for balls being shot or passed. This is caused by two effects. Firstly, θ_d is calculated between two coordinate systems with the origin at the Kinect. The ball originates from a slightly different place. Because the Omnivision measurement comes with a delay, this results in an offset in θ_d. Secondly, the ball is only visible for a short amount of time for both the Kinect and Omnivision after a shot.

The designed method deals well with collisions. During the experiment for case 1 the collisions intentionally were small to not damage the TURTLEs, but still a change in θ_d was noticeable. After a collision the Kalman filter quickly adjusts to the new value of θ_d.

The algorithm has not yet been tested in a real game, but from the Greenfield analysis, it can be expected that there will be enough moments for the calibration to result in a good $\hat{\theta}_d$ throughout the game.

8.2 Recommendations

To test whether the calibration method will indeed result in a good $\hat{\theta}_d$ throughout the game, the algorithm should be still tested in a real game.

In case the moments of calibration are too scarce, it can be considered to also do a calibration when passing. The algorithm would need some change for this. At the moment θ_d is calculated between two coordinate systems with the origin at the Kinect. To make it applicable for a pass, these origins should be at the place the ball originates from.

Because this report only discussed the online mounting calibration method of the rotation around the vertical axis, an improvement could still be found in also implementing an online mounting calibration around the horizontal axes. A method for this has already been developed, but not yet implemented.

A direction of research could be to investigate why the distance of the ball to the TURTLE as derived from the corrected Kinect data and as derived from the Omnivision data differ so significantly.

Bibliography

[1] "Tech United official site," 2017. [Online]. Available: http://www.techunited.nl/

[2] "RoboCup official website," 2016. [Online]. Available: www.robocup.org/

[3] "Middle Size League - RoboCup Wiki," 2017. [Online]. Available: http://wiki.robocup.org/Middle{_}Size{_}League

[4] Y. Douven, Eindhoven, 2017.

[5] M. Asada, T. Balch, and A. Bonarini, "Middle Size Robot League Rules and Regulations for 2017," no. 4, 2016. [Online]. Available: http://wiki.robocup.org/images/7/72/2017-msl-rules{_}v18{_}4.pdf

[6] B. Coenen and P. Metsemakers, "Refbox - Tech United Eindhoven," 2010. [Online]. Available: http://www.techunited.nl/wiki/index.php?title=Refbox

[7] "Jetson TK1 Embedded Development Kit — NVIDIA," 2017. [Online]. Available: http://www.nvidia.com/object/jetson-tk1-embedded-dev-kit.html

[8] "GC780 GC780C GC780 User Manual," 2010. [Online]. Available: https://www.1stvision.com/cameras/AVT/dataman/GC780{_}User{_}Manual.pdf

[9] "Kinect for Windows Sensor Components and Specifications," 2017. [Online]. Available: https://msdn.microsoft.com/en-us/library/jj131033.aspx

[10] P. S. Maybeck, "Stochastic models, estimation, and control," in *Stochastic models, estimation, and control*, volume 1 ed. Ohio: Harcourt Brace Jovanovich, 1979, ch. 1, Introdu. [Online]. Available: http://www.cs.unc.edu/{~}welch/kalman/media/pdf/maybeck{_}ch1.pdf

[11] G. Welch and G. Bishop, "An Introduction to the Kalman Filter," *In Practice*, vol. 7, no. 1, pp. 1–16, 2006. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.79.6578{&}rep=rep1{&}type=pdf

[12] D. Jurić, "Object Tracking: Kalman Filter with Ease," 2015. [Online]. Available: https://www.codeproject.com/Articles/865935/Object-Tracking-Kalman-Filter-with-Ease

[13] M. M. Wilson, Edwin B., Hilferty, "The distribution of chi-square," *Proceedings of the National Academy of Sciences*, vol. 17, no. 12, pp. 684–688, 1931. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1076144/pdf/pnas01728-0064.pdf

[14] V. Spruyt, "How to draw a covariance ellipse?" 2014. [Online]. Available: http://www.visiondummy.com/2014/04/draw-error-ellipse-representing-covariance-matrix/

A Analysis of the standard deviation of r

In this Appendix a further exploratory analysis is performed with the focus on the standard deviation of the measured ball distance, σ_r, and how this depends on the location of the ball. It exists of two parts: Kinect and Omnivision.

Kinect

For this analysis the data from Section 4.1 is used. In Figure A.1 σ_r is plotted against the position coordinates (r, θ). Subsequently a first order polynomial is fitted through the data points. This results in a clear dependency of the σ_r on r, according to Equation A.1.

$$\sigma_r = 2.1 * 10^{-3} * r + 0.23 \tag{A.1}$$

However, there is no clear dependency visible between σ_r on θ.

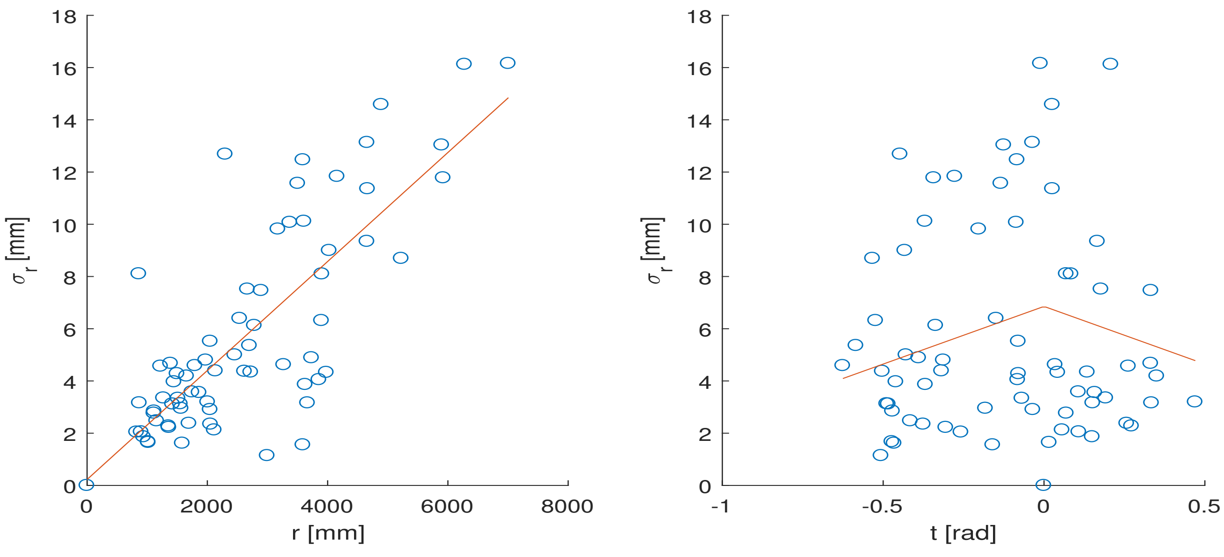


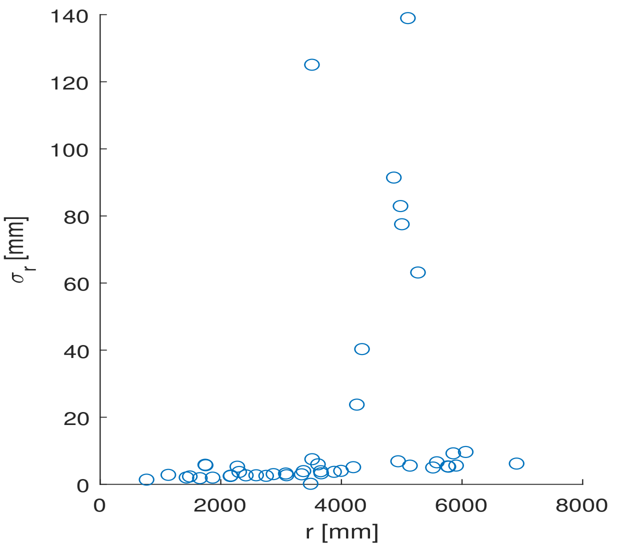
Figure A.1: *The relation between σ_r and the location (r, θ) of the Kinect: r left θ right*

Omnivision

A similar analysis is performed on the Omnivision. The data used is the data from Section 2.4. In Figure A.2 σ_r is plotted against the position coordinates (r, THETA). Notable is that there are eight extreme values for the standard deviation. In Figure A.3 these extreme values are filtered out and a linear polynomial is fitted through the remaining data points. A clear relation between σ_r on r is visible. This relation is expressed in Equation A.2.

$$\sigma_r = 8.8 * 10^{-4} + 1.0 \tag{A.2}$$

There does not seem to be a dependency of σ_r on r.



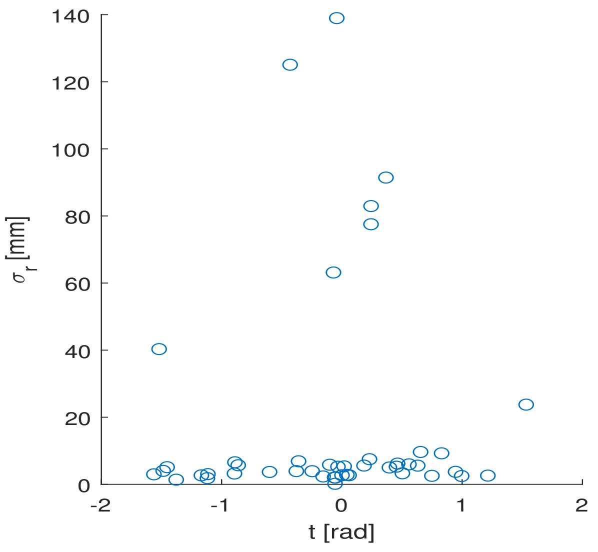
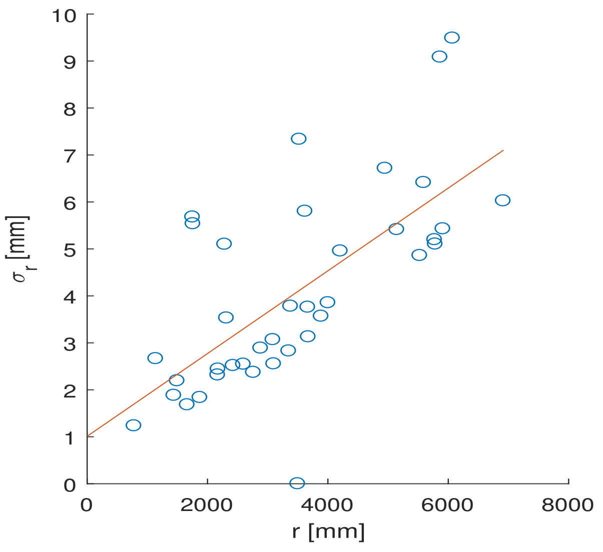


Figure A.2: *A plot of the σ_r against the location (r, θ) of the Omnivision: r left θ right*



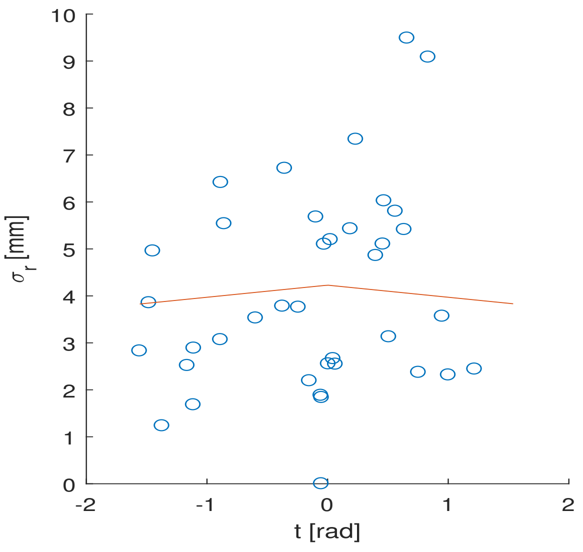


Figure A.3: *The relation between σ_r and the location (r, θ) of the Omnivision: r left θ right, with big values left out.*

B Covariance ellipses

This appendix is based on [14]. In order to draw a covariance ellipse, four properties must be obtained from the measurement set:

- The mean
- The longest axis length
- The shortest axis length
- The orientation of the longest axis length

The axis lengths should correspond to the 95%-confidence interval.

The mean can immediately be determined in Matlab with the mean-function. For the other three properties we need the covariance matrix, which also can be directly obtained from Matlab.

A characteristic of a covariance matrix is that the largest eigenvector, v_1, always points in the direction of the largest variance of the data. Naturally, the other eigenvector, v_2, which is orthogonal to the largest eigenvector, points in the direction of the smallest variance of the data. The magnitude of the variances equals the corresponding eigenvalues, λ_1, λ_2.

From this theory we can determine the longest and shortest axis length according to respectively Equation B.1 and B.2.

$$axis1 = \sqrt{5.991 * \lambda_1} \tag{B.1}$$

$$axis2 = \sqrt{5.991 * \lambda_2} \tag{B.2}$$

The value 5.991 is obtained from a Chi-Square Probability table with two degrees of freedom at 95% [13]. The orientation of the longest axis length can be determined with Equation B.3.

$$\alpha = -atan2(v_1(x), v_1(y)) \tag{B.3}$$

With this information it is trivial how to plot the ellipses.

C Greenfield analysis

In this appendix a greenfield analysis is performed, with as a goal to check whether there are enough opportunities to perform the calibration. In order to do this the matches of the Robocup 2016 Leipzig tournament are analysed. Firstly, the amount of Refbox moments per half of a match is counted for six matches. The results are depicted in the box plot in Figure C.1.

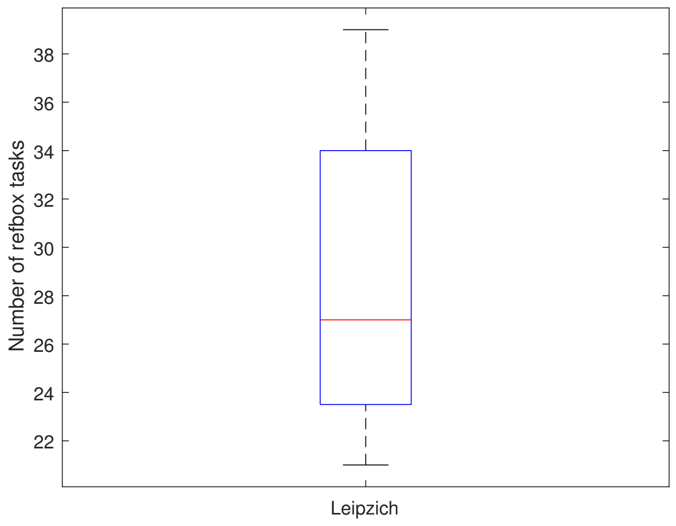


Figure C.1: *A box plot of the number of Refbox tasks during one half of a match*

The mean is 27 moments per 25 minutes, which seems to be quite often. However, not during every Refbox task, every robot can perform a calibration. Another requirement is that both the Kinect and the Omnivision camera see the ball. In Figure C.2 a bar plot is shown in which the percentage of this happening given that a Refbox task is happening is represented.

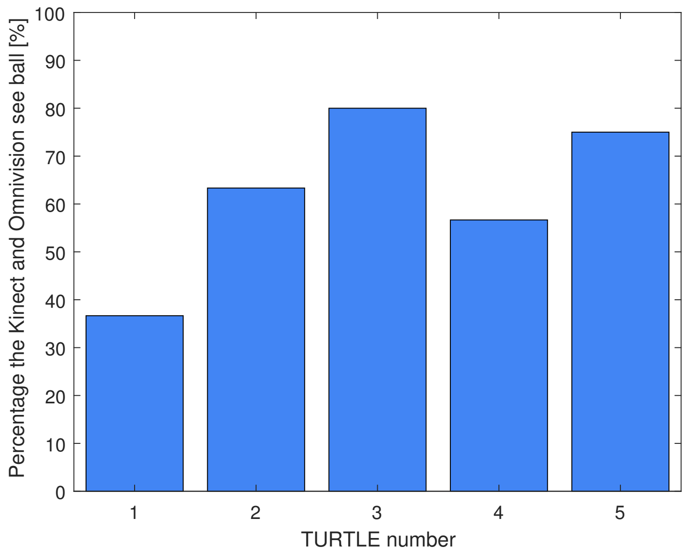


Figure C.2: *A bar plot of the percentage that a TURTLE sees the ball with both the Kinect and the Omnivision*

The most critical TURTLE is the keeper: TURTLE 1, which sees the ball with both sensors in 37% of the Refbox tasks. This means an opportunity for a calibration every two to three minutes. All the other TURTLEs will have significantly more opportunities. In conclusion, during a match there are sufficient opportunities to perform a calibration.