Technische Universität Berlin

Multi-Hypotheses Kalman Filter based Self-Localization for Autonomous Soccer Robots

Masterarbeit
am Fachgebiet Agententechnologien in betrieblichen Anwendungen und der Telekommunikation (AOT)
Fakultät IV Elektrotechnik und Informatik
Technische Universität Berlin

vorgelegt von
Qian Qian

Matriculation Number: 359738

Betreuer: Dr. Yuan Xu,
Prof. Dr.-Ing. habil. Sahin Albayrak,
Prof. Dr. Ben Juurlink

July 24, 2015

Hiermit versichere ich, dass ich diese Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Hereby I declare that I wrote this thesis myself with the help of no more than the mentioned literature and auxiliary means.

Berlin, July 24, 2015

. . . .

Qian Qian

Acknowledgements

First of all, I would like to thank my mentor Dr. Yuan Xu for giving me valuable advice and guidance throughout my thesis work. Also I would like to thank the other DAInamite teams members Martin Berger and Erdene-Ochir Tuguldur who always gave me inspirations and motivations. Lastly, I would like to thank my family and all the friends who supported me throughout the process.

Abstract

Self-localization is a crucial part for autonomous robots, particularly in the RoboCup Standard Platform League (SPL). The robot needs to know where it is in the game field to accomplish the remaining decision making tasks such as passing the ball to the teammates, saving the goal, path planning and so on. The task herein is mainly to estimate the robot's position accurately from its sensor data. However the challenge is that the robot only observes its surrounding world partially at a given time, and it is sometimes not sufficient enough to determine its exact position. In addition, the robot must also identify the landmarks on the game field which could be ambiguous in most of the cases.

In this thesis, two localization methods are investigated and implemented for the NAO robot in the RoboCup SPL, namely optimization based localization and multi-hypotheses Kalman filter localization. Optimization based localization differs from the classic Kalman filter in the measurement update step, where the robot position is estimated by minimizing the measurement error of the field line points in an iterative optimization process. On the other hand, given more landmarks, multi-hypotheses Kalman filter localization describes a multi-modal probability distribution and can fast re-localize the kidnapped robot by incorporating the sensor resetting technique from the particle filter. At the same time, it also inherits the advantages from the Kalman filter of being accurate and efficient in position estimation. The benchmark results presented in this thesis show that multi-hypotheses Kalman filter localization outperforms optimization based localization and particle filter localization in terms of accuracy and efficiency.

Zusammenfassung

Selbstlokalisierung ist unerlässlich für autonome Roboter, insbesondere in der RoboCup Standard Platform League (SPL) spielt sie eine wichtige Rolle. Der Roboter muss seine Position auf dem Spielfeld kennen um Spielmanöver sinnvoll durchführen zu können, wie zum Beispiel einen Pass zu seinen Mitspielern, das Schützen des Tores, das Planen von Wegen und so weiter. Die Aufgabe besteht nun hauptsächlich darin, die Position des Roboters anhand der Daten, welche seine Sensoren liefern, möglichst genau zu schätzen. Jedoch nimmt der Roboter seine Umgebung zu gegebener Zeit nur teilweise wahr und in vielen Fällen reicht es nicht aus, allein die Position des Roboters zu kennen. Des weiteren muss der Roboter Orientierungspunkte auf dem Spielfeld erkennen können, welche in den meisten Fällen in uneindeutiger Weise vom Roboter wahrgenommen werden.

In dieser Arbeit sollen zwei Lokalisierungsmethoden für den NAO-Roboter im RoboCup SPL untersucht und implementiert werden, namentlich die optimierungsbasierte Lokalisierung und die Multi-Hypothesen-Kalman-Filter Lokalisierung. Die optimierungsbasierte Lokalisierung unterscheidet sich vom klassischen Kalman-Filter in der Korrektur, wobei die Position des Roboters durch iterative Minimierung des Messfehlers bezüglich der Feldlinienpunkte bestimmt wird. Andererseits beschreibt der Multi-Hypothesen-Kalman-Filter, stehen weitere Orientierungspunkte zur Verfügung, eine multimodale Wahrscheinlichkeitsverteilung und kann dadurch einen gekidnappten Roboter schnell lokalisieren, dass er die Sensor-Resetting-Methode vom Partikelfilter mitverwendet. Zur gleichen Zeit erbt der Multi-Hypothesen-KalmanFilter vom klassischen Kalman-Filter die Eigenschaft, ein akkurater und effizienter Schätzer der Position zu sein. Die Vergleichswerte, welche in dieser Arbeit präsentiert werden, zeigen die klare Überlegenheit der Multi-Hypothesen-Kalman-Filter Lokalisierung gegenüber der optimierungsbasierten Lokalisierung und der Partikelfilter Lokalisierung bezüglich Genauigkeit und Effizienz.

Contents

List of Figures **XIII**

List of Tables **XV**

Acronyms **XVII**

1. Introduction **1**
- 1.1. Motivation 1
- 1.2. Problem Statement 2
  - 1.2.1. Robot Hardware 2
  - 1.2.2. Standard Platform League Field 3
  - 1.2.3. Problem Formulation 3
- 1.3. Thesis Outline 6

2. Background and Related Work **7**
- 2.1. Bayes Filters 7
  - 2.1.1. Particle Filter 8
  - 2.1.2. Kalman Filter and its Ramifications 9
    - 2.1.2.1. Kalman Filter 9
    - 2.1.2.2. Kalman Filter Variants 10
    - 2.1.2.3. Multi-Model Kalman Filter 11
- 2.2. State of the Art of Robot Localization in RoboCup 12
- 2.3. Software Architecture of DAInamite 13
- 2.4. Vision Perception 13

3. Self-localization Pipeline **17**
- 3.1. Motion Update 18
  - 3.1.1. Motion Model 18
  - 3.1.2. Process Noise Model 19
- 3.2. Sensor Update 20
  - 3.2.1. Optimization Based Model 20
  - 3.2.2. Feature Based Model 21
    - 3.2.2.1. “T” and “X” Junction Detection 21
    - 3.2.2.2. Measurement Model Choice 21
    - 3.2.2.3. Measurement Noise Model 25

3.2.2.4. Landmark Correspondence 27
3.2.2.5. Multiple Simultaneous Measurements 33

4. Multi-Hypotheses Kalman Filter 35
4.1. Hypothesis Model Weighting 35
4.2. Landmark Based Resampling 36
4.2.1. "L" Junction Look Up Table 38
4.3. Best Hypothesis Model and Confidence 39
4.4. Model Pruning 40
4.4.1. Pruning by Weight 40
4.4.2. Merging by Mahalanobis Distance 40
4.4.3. Pruning by Distance to Best Model 42

5. Analysis and Benchmark 43
5.1. Experiments Setup 43
5.1.1. Ground Truth 43
5.1.2. Perception Log 45
5.2. Code Optimization 45
5.3. Benchmarks 46
5.3.1. Accuracy 47
5.3.2. Functionality 58
5.3.3. Efficiency 58

6. Conclusion and Future Work 61
6.1. Conclusion 61
6.2. Future Work 61

Bibliography 63

Appendices 65

A. Appendix 65
A.1. Speed Profiles of Localization Algorithms 65

List of Figures

1.1. Body construction of NAO V4. 3
1.2. The defined coordinate system of physical world frame. 5

2.1. Physical Architecture of DAInamite Code Base. 14
2.2. Example for a typical vision perception result of the robot in the field. . . 15
2.3. Only camera frame for bottom camera is shown 16

3.1. Self-localization pipeline. 17
3.2. Relative odometry change. 19
3.3. All L junctions in the field. 22
3.4. Detection of “L”, “T” and “X” junction in the field. 24
3.5. Observation given in Cartesian coordinates. 24
3.6. Line represented in Hough space. 25
3.7. Measurement noise with respect to landmark distance in robot frame. . . 26
3.8. Landmarks in image plane. 27
3.9. Landmarks in robot frame. 28
3.10. Measurement covariance of the point landmark. 29
3.11. Matching of end points of the detected line. 30
3.12. The criteria of determining the line correspondence. 31
3.13. Matching result of line and penalty area. 32

4.1. “L” junction look up table 39
4.2. Visualization of two Gaussian distributions 41

5.1. The 3D printed support and the pattern marker. 44
5.2. SSL-Vision color and camera calibration result, and robot position result. 44
5.3. Particle filter localization trajectory compared with ground truth trajectory. 47
5.4. Optimization based localization trajectory compared with ground truth trajectory. 48
5.5. Multi-hypotheses Kalman filter localization trajectory compared with ground truth trajectory. 48
5.6. Particle filter localization result and error in θ dimension. 49
5.7. Particle filter localization result and error in θ dimension. 50
5.8. Particle filter localization result and error in θ dimension. 51
5.9. Optimization based localization result and error in x dimension. 52
5.10. Optimization based localization result and error in y dimension. 53

5.11. Optimization based localization result and error in θ dimension. 54
5.12. Multi-hypotheses Kalman filter localization result and error in x dimension. 55
5.13. Multi-hypotheses Kalman filter localization result and error in y dimension. 56
5.14. Multi-hypotheses Kalman filter localization result and error in θ dimension. 57

A.1. Program profile of particle filter localization for executing the perception log on the NAO robot. 66
A.2. Program profile of optimization based localization for executing the perception log on the NAO robot. 67
A.3. Program profile of multi-hypotheses Kalman filter localization for executing the perception log on the NAO robot. 68

List of Tables

1.1. Schematic diagram of the soccer field. 4

2.1. Kalman Filter Algorithm 10
2.2. Extended Kalman Filter Algorithm. 11

5.1. Accuracy comparison between different localization algorithm results . . . 58
5.2. Standard deviation comparison between different localization algorithm results 58
5.3. Average execution time per frame of the perception log for different localization algorithms running on the NAO robot. 59

Acronyms

AI Artifical Intellegence.

EKF Extended Kalman Filter.

FIFO First In, First Out.

FSR Force-sensing Resistor.

GPS Global Positioning System.

HD High Definition.

IMU Inertial Measurement Unit.

MCL Monte-Carlo Localization.

PDF Probability Distribution Function.

SPL Standard Platform League.

UKF Unscented Kalman Filter.

1. Introduction

Robot Soccer World Cup, known as RoboCup, is an annual international robotics competition conducted since 1997. It aims to foster research and development of robotics and Artifical Intellegence (AI), by offering a public appealing but challenging competition. RoboCup consists of five major competition domains, they are RoboCup Soccer, RoboCup Rescue, RoboCup@Home, RoboCup Logistics and RoboCup Junior. Currently the RoboCup Soccer includes several soccer leagues to cover difference research challenges. The main concern in this thesis is the Standard Platform League (SPL) under the sub-category of RoboCup soccer, in which all the teams use identical humanoid robot NAO that is manufactured by Aldebaran Robotics. The robots should operate fully autonomously without external control. The research regarding RoboCup SPL has been actively conducted under various topics. DAInamite, a team from DAI-Labor, TU-Berlin, is dedicated in advancing robot technology; the team has continuous research on humanoid robot NAO and their first participation in the world championship in the SPL was in RoboCup 2013 in Eindhoven, reaching the quarter finals. The code base from DAInamite consists of several modules including motion, vision, behavior, localization, etc, which makes the NAO robot fully functional and competitive in the RoboCup game. The implementation of the localization algorithm proposed in this thesis will be based on the current software infrastructure of DAInamite.

1.1. Motivation

Localization awareness is a central aspect for many pervasive computing applications, especially for autonomous robots playing soccer. Just like human, the robots need to know where themselves are on the field when playing soccer, in order to make decisions and undertake actions. The high level decision making depends highly on the accuracy of the location of the robot, e.g. moving towards a particular direction to split the defense of the defender, or kicking the ball to the goal, or distinguishing the opponent's goal on its own. If the result of localization is inaccurate, it will not only curtail the performance of other algorithms like ball tracking and goal saving, but also prohibit developing advanced techniques for competitive enhancement like opponent modeling or passing the ball to teammates.

In many other applications like self driving car, many sensors including the Global Positioning System (GPS), laser scanners, radar or high end 3D cameras are used to help accomplish the localization task. However, for the NAO robot, with limited number of sensors and restricted computational capability of the processing unit, the localization problem becomes particularly challenging. The robot is able to get odometry information from the body kinematics and Inertial Measurement Unit (IMU) to calculate the supposed walking distance, and using this to help predict its location. Although the odometry information is the most direct measurement for the input of localization calculation, inherited systematic errors like asymmetry of the installation of the joint motors will cause the robot to constantly produce offset in walking, or unpredicted environment factors like foot slipping on the ground, bump into obstacles. The accumulated odometry errors throughout a time period will result in unacceptable localization result.

Besides odometry information, the NAO robot can also use its two High Definition (HD) cameras to sense the surrounding environment, but without depth information. Since the cameras have 60.9° horizontal field of view and 47.6° vertical field of view, the robot can only get local partial perception of the field at any given time. With image processing, landmarks or features can be extacted from the images as another input for localization calculation. However, due to the non-uniqueness of most of the landmarks in the SPL field and false positive results from image processing, the landmarks become severely ambiguous. On the other hand, the image may also be blurred due to the motion of the robot, and the environment is dynamically changing as the other moving robots may occlude the landmark. These challenges impose significant difficulties not only in detecting the features from the noisy image, but also the knowhow of employing ambiguous landmarks for localization.

Furthermore, the soccer field itself is symmetrically structured, which means the two halves of the field are the same. To counter this problem, normally features outside of the field need to be considered to differentiate one side from the other. Not to mention, NAO robot has limited computation power, which requires the localization algorithm to be computationally efficient and not to affect other critical modules like motion which have higher execution priorities.

1.2. Problem Statement

1.2.1. Robot Hardware

The robot used in this thesis for localization algorithm development is NAO V4 robot from Aldebaran Robotics. The robot is 573 mm in height, 275 mm in width. The body construction and the equipped sensors are illustrated in Figure 1.1. Also, the robot has one 2-axis gyrometer and one 3-axis accelerometer inside the body, a Force-sensing Resistor (FSR) on each foot bottom, which is not shown in Figure 1.1.

The two cameras are placed on the face, each camera can capture image of resolution up to 1280 × 960@30fps.

The robot runs a Gentoo version of Linux and the mother board resides in the head of the robot. The specification of the mother board is:

- ATOM Z530 1.6 GHz CPU [1]

---

[1] It is a single core processor, with Intel Hyper-Threading technology.

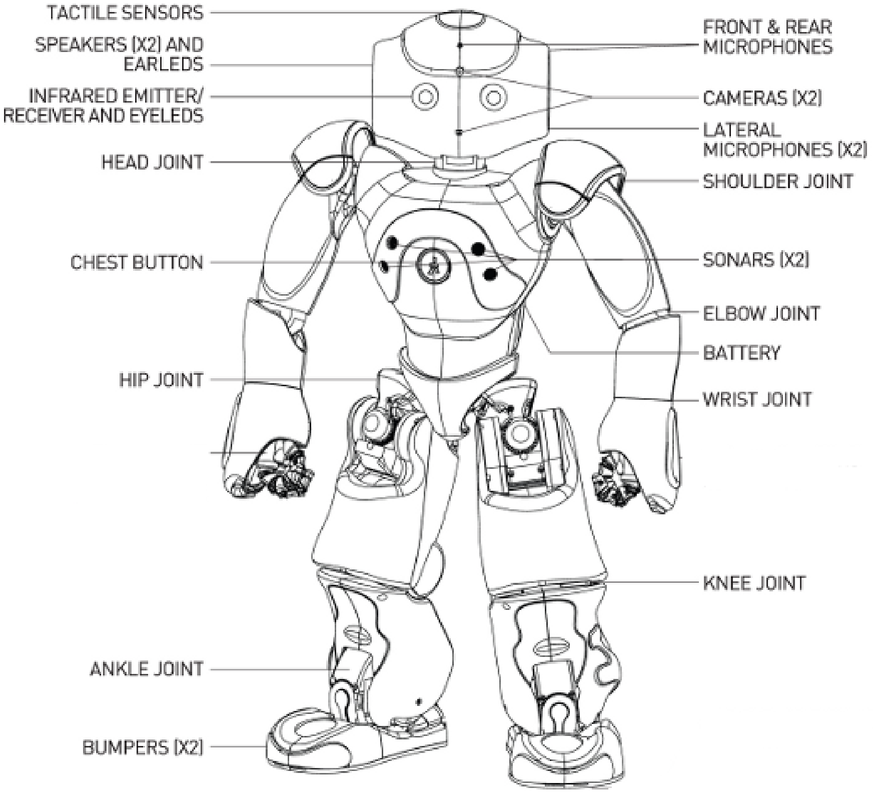

TACTILE SENSORS
SPEAKERS (X2) AND EARLEDS
INFRARED EMITTER/ RECEIVER AND EYELEDS
HEAD JOINT
CHEST BUTTON
HIP JOINT
ANKLE JOINT
BUMPERS (X2)
FRONT & REAR MICROPHONES
CAMERAS (X2)
LATERAL MICROPHONES (X2)
SHOULDER JOINT
SONARS (X2)
ELBOW JOINT
BATTERY
WRIST JOINT
KNEE JOINT

Figure 1.1: Body construction of NAO V4. [1]

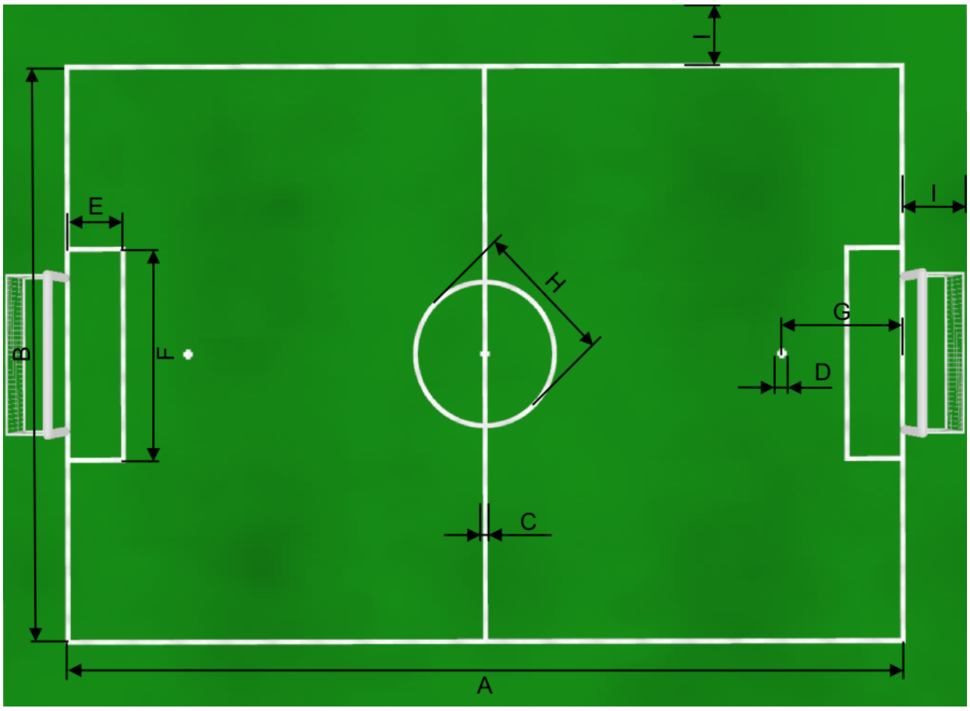
- 1 GB RAM
- 2 GB Flash memory
- 8 GB Micro SDHC

1.2.2. SPL Field

Every year the rules of the SPL are modified on the basis of the old ones to make them closer to the rules of real soccer game and at the same time increase the difficulty levels of the game. Significant changes include the field size being enlarged from $6m \times 4m$ to $9m \times 6m$ in 2013, and the color of goal posts become white instead of yellow in 2015. The schematic diagram of the soccer field and corresponding dimensions are depicted in Table 1.1.

1.2.3. Problem Formulation

In the robotic localization problem, the algorithm is mainly concerned with estimating the “state” of the robot. The “state” in the context of the RoboCup SPL refers to



ID	Description	Length (in mm)
A	Field length	9000
B	Field width	6000
C	Line width	50
D	Penalty mark size	100
E	Penalty area length	600
F	Penalty area width	2200
G	Penalty mark distance	1300
H	Center circle diameter	1500
I	Border strip width	700

Table 1.1: Schematic diagram of the soccer field (not to scale) and corresponding dimensions in mm. Note that measurements on this diagram are made to the center of lines. [2]

the location (2D Cartesian coordinate) and the orientation of the robot in the physical world. In this case the state to be estimated can be represented as a 3 dimensional vector $x(t)$ as follows:

$$x(t) = \begin{bmatrix} x_r(t) \\ y_r(t) \\ \theta_r(t) \end{bmatrix}$$

where $(x_r(t), y_r(t))$ denote the Cartesian coordinates and $\theta_r(t)$ is the orientation of the robot at time t. As illustrated in Figure 1.2, the coordinate system of physical world frame is defined with its origin located at the center of the field, and its positive x-axis axle pointing to the opponent's goal.

For robots in SPL, the localization is intended to achieve the following tasks:

- Global Localization: it happens at the beginning of the game, where the robot has

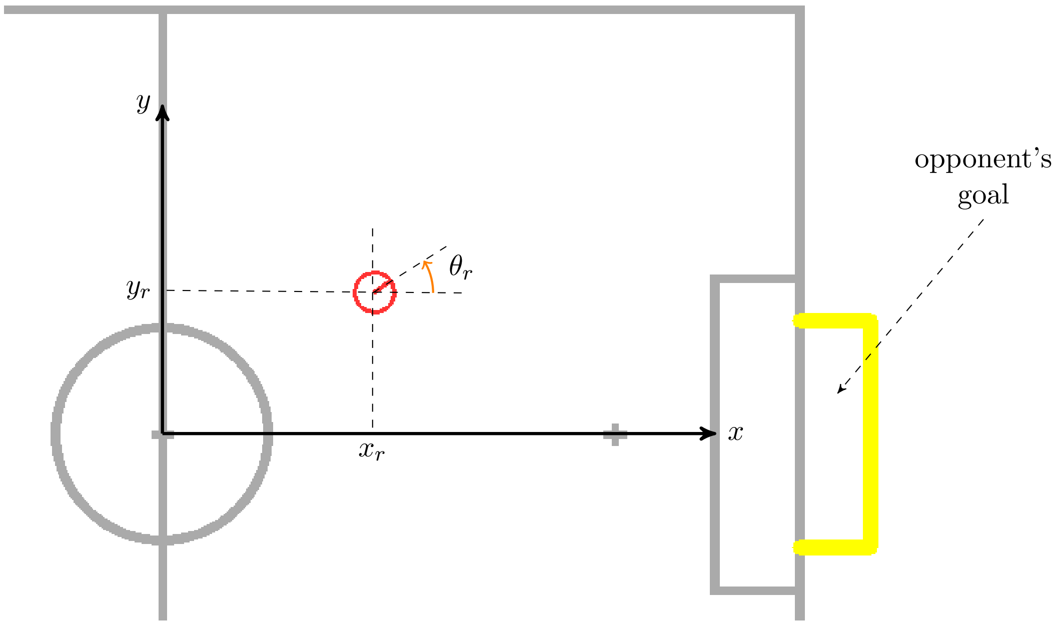


Figure 1.2: The defined coordinate system of physical world frame.

to localize itself without the knowledge of its initial position.

- Position Tracking: it assumes that the initial position of the robot is known, and the position tracking can be achieved by accommodating the noise from motion and measurements.
- Kidnapped robot: the robot can be kidnapped and teleported to another position in the game. The problem is similar to global localization, except that the robot might believe it knows where it is while it does not.

As stated in section 1.1, the environment the robot resides in is inherently unpredictable, the sensors are limited in what they can perceive, the joint motors are also to some extent unpredictable. Hence, a probabilistic representation is used to describe the uncertainty of the robot's location, the probability distribution over the state $x(t)$ is called belief, written as $Bel(x_t)$.

In order to estimate the state $x(t)$ and the belief $Bel(x_t)$, the localization algorithm takes the detected objects from vision in the field as one of the inputs. The DAInamite's vision module can detect yellow goal post, center circle, penalty area, lines in the field and boundary of the field, and by projecting the landmarks from the image frame to the robot's local frame, distances to the robot can be estimated. Furthermore, the odometry information is also fed into the localization algorithm. The odometry of the robot is obtained from the motion module by calculating the body kinematics, given that the robot has at least one foot placed on the ground.

1.3. Thesis Outline

The rest of the thesis is structured as follows:

Chapter 2 describes the background and related work concerned to the fundamentals of Bayes theories and localization methods developed for robots in RoboCup, as well as the current software architecture from the team DAInamite, to which the localization algorithm will be integrated.

Chapter 3 explains the motion model and sensor model designed for Extended Kalman Filter (EKF) based localization. It also illustrates the strategies to find landmark correspondence based on ambiguous observations.

Chapter 4 describes the design of multi-hypotheses Gaussian models which encapsulate a EKF in each model. It explains the mechanism of resampling and pruning step to maintain a multi-modal probability distribution.

Chapter 5 comprehensively analyses the functionality, accuracy and efficiency benchmarks of different localization algorithms.

Chapter 6 summarizes the thesis with a conclusion and gives an outlook about the possible future work in this regard.

2. Background and Related Work

In robotics, significant research has been carried out to make the robot localize itself in the environment more accurately. Many localization methods based on probability theories are proposed. Most of them draw their fundamental theory from Bayers filter, and later they evolve into more specific filters such as Particle filter [3] or Kalman filter [4] variants like EKF [5] and Unscented Kalman Filter (UKF) [6]. Each kind of filters has its own intrinsic properties and has advantages and disadvantages depending on the application.

2.1. Bayes Filters

Bayes filter aims to sequentially estimate the belief of the state conditioned on all the information obtained in the sensor data. As illustrated in Equation 2.1, it assumes that, the sensor data has a sequence of time-indexed sensor data. The belief of state $x(t)$, $Bel(x_t)$ can be represented as a posterior probability density conditioned on all the sensor data from the past observations, $z_1, z_2 \dots z_t$.

$$Bel(x_t) = p(x_t | z_1, z_2, \dots, z_t) \tag{2.1}$$

However, the computational complexity of such belief grows exponentially with the increase of observations. To make the computation feasible, Bayes filter assumes that the dynamic system is a Markov process. It means the state belief at time t depends only on the previous state information at time $t-1$. Bayes filter includes two components to estimate the state belief: time update and measurement update.

Time update (prediction)
Firstly, a prediction is made by using the motion model of the robot, as formulated in Equation 2.2.

$$Bel^{-}(x_t) \leftarrow \int p(x_t | x_{t-1}) Bel(x_{t-1}) dx_{t-1} \tag{2.2}$$

Where $p(x_t | x_{t-1})$ depicts the system's dynamics in which a motion model is used to describe how the system state changes due to motion movement.

Measurement update (correction)
By making a measurement from the sensor, the filter corrects the prediction using this observation.

$$Bel(x_t) \leftarrow \alpha_t p(z_t | x_t) Bel^{-}(x_t) \tag{2.3}$$

$p(z_t|x_t)$ in Equation 2.3 represents the probability of making observation z_t given that the robot is at position x_t. It generally describes the perceptual model of the robot. Coefficient α_t is a normalization factor to make sure the belief calculated add up to 1.

Moreover, the update of the belief in Bayes filter is recursive, which means that, the belief at time t is calculated from belief at time $t-1$. Bayes filter provides the theoretical basis for the following filters to be discussed, namely particle filter and Kalman filter.

2.1.1. Particle Filter

Particle filter, also known as Monte-Carlo Localization (MCL), is the localization algorithm that is currently under use for the SPL robot in DAInamite. Particle filter represent a probability distribution by a set of samples or particles. Each particle represents a concrete state in the current system at time t, in our case it will be the one instantiation of coordinates and the orientation. Meanwhile, each particle is also associated with a weight which indicates how probable the state is in the system. In practice, the number of particles is huge in order to provide a close hypothesis for the true world, and normally 1000 particles are needed to approximate the SPL soccer field. Following the principle of Bayes filter, the motion update will move the particles according to the motion model; measurement update will result in a resampling of the particles, in which the particles with higher weight will survive, on the other hand, the ones with lower weight will die out, and in replacement, particles close to the area of the surviving ones will be generated. In the DAInamite team, due to limited computation power of NAO, only 60 particles are instantiated in the system. With such a low number of particles, the system state can hardly be estimated. As an enhancement for this, the technique of particle filter resetting is used to improve the performance of the localization algorithm with low number of particles. It produces particles with high weights for locations where unambiguous observations are made. The high weight particles help to provide relatively reliable base states which result in correct location interpretation for the later recursive processing.

Advantages

- Particle filter is easy to implement.
- Particle filter is robust in localization.
- Particle filter works with multi-modal probability distribution and is applicable in both linear and non-linear systems.
- By scaling up the number of particles, the accuracy of the result could be improved at the cost of more computational effort.

Disadvantages

- Computationally intensive when the number of particles is high.
- Estimation accuracy reduces when the number of particles is low.

2.1.2. Kalman Filter and its Ramifications

2.1.2.1. Kalman Filter

Kalman filter was invented by Swerling (1958) and Kalman (1960) as a technique for filtering and prediction in linear Gaussian systems. In Kalman filter, it assumes the belief of a state conforms to a Gaussian distribution.

A Gaussian distribution for a single random variable x has the form in Equation 2.4.

$$f(x, \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2} \tag{2.4}$$

The parameter μ in this definition is the mean or expectation of the distribution. The parameter σ is its standard deviation; its variance is therefore σ^2. The larger the variance is, the wider the distribution spreads. However, in the problem domain of the robot localization, the state of the robot is a vector with three variables: 2D coordinate and orientation. To represent a multivariate Gaussian distribution, the following definition is used:

$$f_x(x_1, \ldots, x_k) = \frac{1}{\sigma\sqrt{(2\pi)^k |P|}} exp(-\frac{1}{2}(x-\mu)^T P^{-1}(x-\mu)) \tag{2.5}$$

It mimics the form of its one dimensional counterpart. x represents the state vector, k is the dimension of the state, μ is the expectation of the distribution, P represents a covariance matrix which is positive-semidefinite and symmetric. If the state is of three variables, then P is a 3×3 matrix.

The Kalman filter model assumes the true state at time t is evolved from the state at $(t-1)$ according to Equation 2.6.

$$x_t = F_t x_{t-1} + B_t \mu_t + w_t \tag{2.6}$$

Where

- F_t is the state transition model which is applied to the previous state x_{t-1};
- B_t is the control-input model which is applied to the control vector u_t;
- w_t is the process noise which is assumed to be drawn from a zero mean multivariate normal distribution with covariance Q_t.

$$w_t \sim N(0, Q_t) \tag{2.7}$$

At time t, an observation (or measurement) z_t of the true state x_t is made according to Equation 2.8.

$$z_t = H_t x_t + v_t \tag{2.8}$$

Where H_t is the observation model which maps the true state space into the observed space and v_t is the observation noise which is assumed to be zero mean Gaussian white noise with covariance R_t.

$$v_t \sim N(0, R_t) \tag{2.9}$$

Prediction	
Predicted (a priori) state estimate	$\hat{x}_{t\|t-1} = F_t\hat{x}_{t-1\|t-1} + B_t u_t$
Predicted (a priori) covariance estimate	$P_{t\|t-1} = F_t P_{t-1\|t-1} F_t^{\mathrm{T}} + Q_t$
Measurement Update (correction)	
Innovation or measurement residual	$\tilde{y}_t = z_t - H_t\hat{x}_{t\|t-1}$
Innovation (or residual) covariance	$S_t = H_t P_{t\|t-1} H_t^{\mathrm{T}} + R_t$
Optimal Kalman gain	$K_t = P_{t\|t-1} H_t^{\mathrm{T}} S_t^{-1}$
Updated (a posteriori) state estimate	$\hat{x}_{t\|t} = \hat{x}_{t\|t-1} + K_t\tilde{y}_t$
Updated (a posteriori) covariance estimate	$P_{t\|t} = (I - K_t H_t) P_{t\|t-1}$

Table 2.1: Kalman Filter Algorithm

Since Kalman filter is a variant of Bayes filter, it follows the prediction-correction routine defined in Bayes filter. The detailed Kalman filter algorithm is shown in Table 2.1.

Advantages

- It is known from the theory that the Kalman filter is an optimal estimator in case that, a) the model perfectly matches the real system, b) the entering noise is white and c) the covariances of the noise are exactly known.
- It is computationally efficient for the prediction and correction steps, because the calculation can be done efficiently using matrix multiplication.

Disadvantages

- Kalman filter works well only with linear system dynamics. In the SPL, neither the motion model nor the observation model of the robot is linear, thus the state transition equation defined above cannot be straightforwardly used. Other techniques to deal with non-linearity proposed in EKF and UKF need to be applied.
- Kalman filter works with uni-modal probability distribution, in other words, unimodal means there is only one peak in the probability distribution under consideration. However, due to the ambiguity of the vision data input, there could be multiple locations which are the potential candidates. In this case, the probability distribution is multi-modal.

2.1.2.2. Kalman Filter Variants

In our robot system, the sensor model and motion model for robot localization are not linear, thus the Kalman filter is not able to tackle. In order to deal with nonlinearity, Kalman filter variants like EKF and UKF are formulated. EKF adapted the technique from Talyor series expansion to linearize the model at the point of its prior mean, and can capture the posterior mean and covariance accurately to the first order from the nonlinearity. On the other hand, UKF approximates a guassian distribution by generating $2L + 1$ number of sigma points (L is the dimension of the state). By applying transformation on each sigma point and calculating the mean and covariance based on the sigma points, it is proposed in [7] that a higher estimation accuracy than EKF can be obtained. In this thesis, considering the efficiency in calculation of using EKF, the localization algorithm based on EKF will be developed.

EKF requires the state transition and observation models not to be linear functions of the state but instead differentiable. So the state transition and observation can be expressed as follows:

$$\hat{x}_{t|t-1} = f(\hat{x}_{t-1|t-1}, u_t) + w_t \tag{2.10}$$

$$z_t = h(\hat{x}_{t|t-1}) + v_t \tag{2.11}$$

The full EKF algorithm is described in Table 2.2.

Prediction	
Predicted state estimate	$\hat{x}_{t\|t-1} = f(\hat{x}_{t-1\|t-1}, u_t)$
Predicted covariance estimate	$P_{t\|t-1} = F_t P_{t-1\|t-1} F_t^\top + Q_t$
Measurement Update (correction)	
Innovation or measurement residual	$\tilde{y}_t = z_t - h(\hat{x}_{t\|t-1})$
Innovation (or residual) covariance	$S_t = H_t P_{t\|t-1} H_t^\top + R_t$
Near-optimal Kalman gain	$K_t = P_{t\|t-1} H_t^\top S_t^{-1}$
Updated state estimate	$\hat{x}_{t\|t} = \hat{x}_{t\|t-1} + K_t \tilde{y}_t$
Updated covariance estimate	$P_{t\|t} = (I - K_t H_t) P_{t\|t-1}$

Table 2.2: EKF Algorithm.

The prediction and measurement update implementation for EKF are basically the same as Kalman filter, except the transition matrix F_k and observation matrix H_k become the Jacobian matrices defined in Equation 2.12 and Equation 2.13.

$$F_t = \left.\frac{\partial f}{\partial x}\right|_{\hat{x}_{t-1|t-1}, u_{t-1}} \tag{2.12}$$

$$H_t = \left.\frac{\partial h}{\partial x}\right|_{\hat{x}_{t|t-1}} \tag{2.13}$$

2.1.2.3. Multi-Model Kalman Filter

The Kalman filters including EKF and UKF assume that the posterior probability is a Gaussian distribution, it performs rather like a maximum likelihood estimator than as a minimum variance estimator [8]. As a result, it greatly reduces the amount of information which is in the true density distribution which might be multi-modal. To overcome the shortcoming of Kalman filter and its ramifications which perform poorly in multi-modal probability distribution, Multi-model Kalman filter was proposed in [8].

Multi-model Kalman filter represents the probability distribution using the weighted sum of multiple Gaussian distribution models, and each Gaussian distribution model i has its weight α_i, for $\alpha_i \in [0, 1]$.

For each model, the multivariate normal Probability Distribution Function (PDF) is given by:

$$p_i(x) = \alpha_i \frac{1}{\sigma\sqrt{(2\pi)^k |P_i|}} exp(-\frac{1}{2}(x-\mu)^T P_i^{-1}(x-\mu)) \tag{2.14}$$

The overall mixture PDF is therefore:

$$p(x) = \sum_{i=1}^{N} p_i(x) \tag{2.15}$$

The mixture of Gaussian representation allows constructing a sensor model with one Gaussian term for each possible correspondence.

During the measurement update, the weight α_i is updated using the method according to [8]:

$$\alpha_i(x) = \nu \alpha_i' \frac{1}{\sigma\sqrt{(2\pi)^k |P_\eta|}} exp(-\frac{1}{2}\eta^{-1} P_\eta^{-1} \eta) \tag{2.16}$$

Where $\eta = (z_t - \widehat{z_i})$ indicate the m-dimensional innovation vector between the measured observation z_t and the expected observation $\widehat{z_i}$ for the fixed correspondence according the i'th model. P_η is the sum of the measurement and the prediction covariance, and ν is a normalization factor.

There exists two situations to be considered for measurement update for multi-model Kalman filter. One is when the observation is unambiguous, which means when goal post or unique land marks have been detected, so only one approximation of the robot state will be generated from the vision. In this case, the measurement update will be applied on each Gaussian term and result in the same number of terms as before. Another case is when the vision measurement is ambiguous, for example, it generates more than one approximations of the robot state from the vision, because the landmarks are not unique or cannot provide enough information. In this case, supposing initially the system has M Gaussian models, and the robot observes a landmark which has N occurrences in the field, then the models will be split into $M \times N$ Gaussian models after update. With ambiguous measurement update, the terms of Gaussian are increasing multiplicatively. Model merge equations and decisions has been proposed in [9], in which several Gaussian models are merged into one if certain metrics are met. In [10], for efficiency, the author only takes the Gaussian models with maximum likelihood, and compensates the information which is lost when models are pruned by introducing filter resetting technique to reproduce the missing models, which is an ingredient taken from particle filter theory.

2.2. State of the Art of Robot Localization in RoboCup

Most localization algorithms applied in RoboCup SPL falls into the class of either particle filter or Kalman filter based localization. Quinlan and Middleton from Team RoboEireann use multiple model Kalman filters for robot localization [9]. They split the Gaussian model when ambiguous measurement happens to cover the multi-modal hypothesis distribution. The problem of this strategy is the multiplicatively increasing number of Kalman models during model split, therefore a model merge step is proposed to combine the models. Team NaoDevil also developed their localization using a multi-hypothesis UKF. However, instead of model splitting, they adopt sensor resetting technique which is commonly used in particle filter [10]. It argues that the pruning technique in [9] will delete temporarily low possibility Gaussian model which may become more probable in the long term.

However, particle filter or Kalman filter is not the only solution for localization, team Berlin United also proposed a constraint based world modeling for localization [11] to overcome the reduction of landmarks such as beacons or colored goals in RoboCup. They build robot position constrains using the geometry property of the lines in the field. In midsize league, a localization method based on optimization approach [12] is proposed, it tries to calculate the perfect match of the vision with the field.

2.3. Software Architecture of DAInamite

DAInamite adopted modular programming pattern, so the tasks could be isolated in different modules. And more importantly, different people can focus on the development of their own modules without interfering other modules. In addition to C++, Python is mostly used in the DAInamite team's code. The time-critical components for motion, and vision are implemented in C++. The remaining modules such as localization, behavior, and ball-tracking are implemented in Python. The python modules communicate with the C++ modules via python C-bindings. The modules which access the hardware of the robot will be connected to Naoqi. Naoqi is the software framework from Aldebaran, through which the NAO robot can be directly controlled. A brief illustration of the physical architecture of the software is shown in Figure 2.1.

2.4. Vision Perception

As seen in Figure 2.1, localization is one sub-module of pyagent module. In reality, localization is running as a separate thread at 30 Hz. Localization module will be able to get odometry information from motion module and vision results from vision module. To reduce the computation burden, the raw images being processed from the cameras are of resolution 640 × 480. From the images, the vision module extracts features such as field lines, field border, orange balls, yellow goals as shown in Figure 2.2. The localization module is concerned only with the features from the vision results instead of the raw image.

The detected lines result from both sides of the white field line, and each detected line is a vector with direction. For example, the line with number 11 in Figure 2.2 is a vector with its start point indicated by blue and end point indicated by white. Along the direction of the line vector, the right side of it is always the white line.

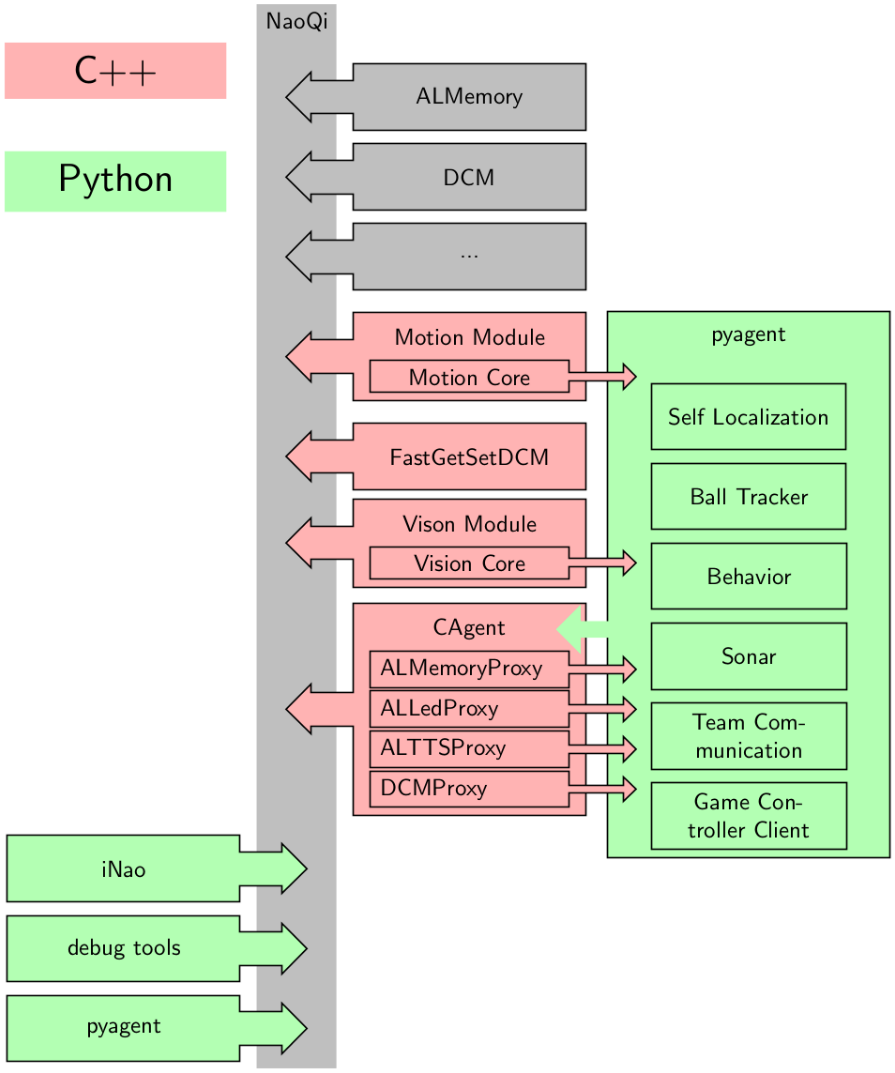

C++
Python
NaoQi
ALMemory
DCM
...
Motion Module
Motion Core
FastGetSetDCM
Vison Module
Vision Core
CAgent
ALMemoryProxy
ALLedProxy
ALTTSProxy
DCMProxy
pyagent
Self Localization
Ball Tracker
Behavior
Sonar
Team Com-
munication
Game Con-
troller Client
iNao
debug tools
pyagent

Figure 2.1: Physical Architecture of DAInamite Code Base.

Among most of the localization methods used in SPL game, the vision results are first transformed from image plane to camera frame, and then projected from the camera frame into the robot local frame in order to do further processing. In the team DAInamite, the camera frame and robot local frame are defined as illustrated in Figure 2.3, where the camera frame has its origin at the place of the camera, and the robot local frame is defined by torso projection on the ground. The camera matrix defines the location and orientation of the camera frame relative to the robot local frame.

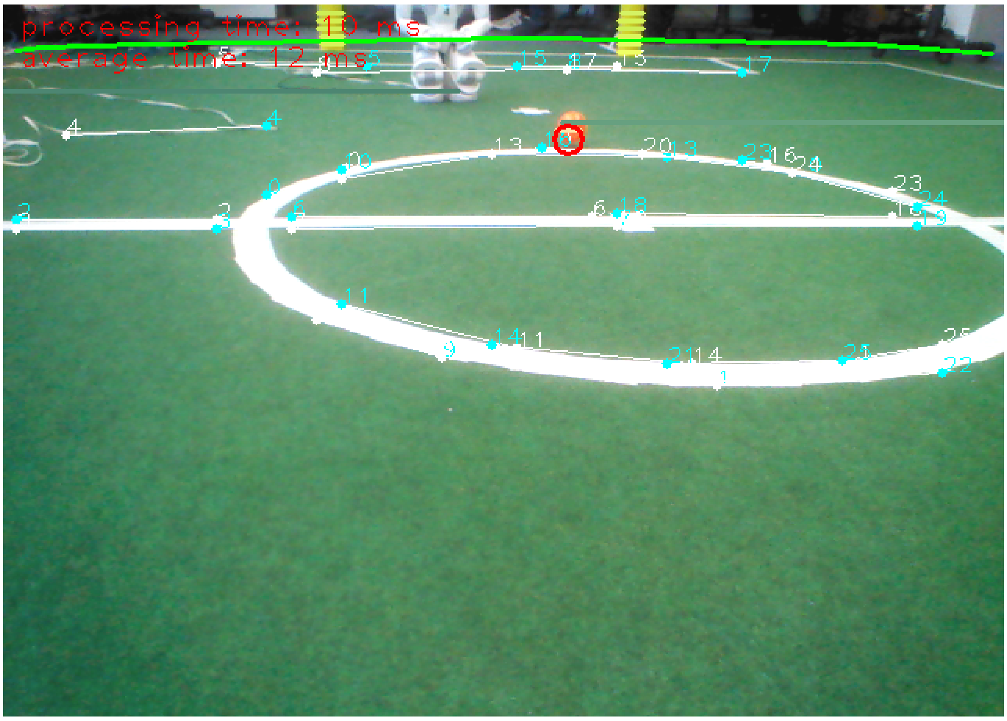

processing time: 10 ms
average time: 12 ms

Figure 2.2: Example for a typical vision perception result of the robot in the field. Detected field lines are indicated by white vectors with start point in blue dot and end point in white dot. Field border is indicted by long green line. The two goal posts are indicted by yellow strips and the ball is indicted by an orange circle.

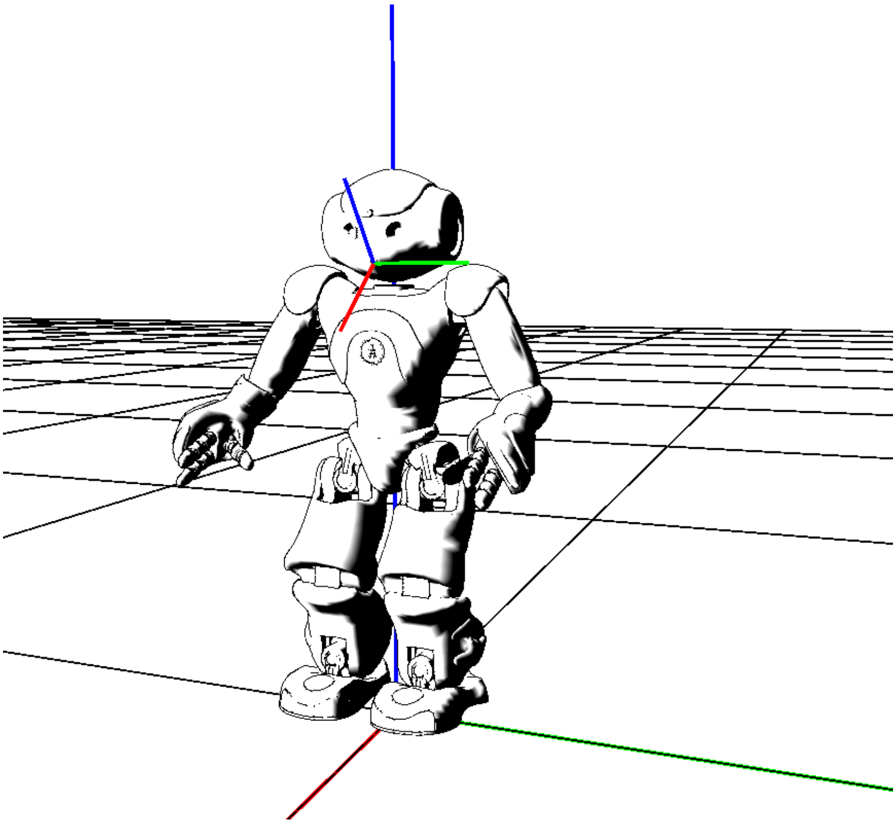


Figure 2.3: camera frame and robot local frame. In both frames, x, y, z axis are represented in red, green, blue respectively. (Only camera frame for bottom camera is shown.)

3. Self-localization Pipeline

As stated in subsection 2.1.2.1, Kalman filter gives an optimal estimation of the current state variable by combining prediction step and measurement update. To be more specific to the problem, prediction step will be referred as motion update and measurement update as sensor update in the following text. In this thesis, odometry information from motion module is used to perform the motion update. From odometry, the information of how much distance the robot moved relative to the pose of last time stamp is provided, thus an estimate of the current position can be established. Likewise, an estimate from the sensor update is performed based on what the robot has observed, or in a nutshell, the vision results from the vision module. In the end, Kalman filter updates the state variable by a weighted average of the estimates. The behavior module then decides the behavior of the robot by the given robot position. The illustration of the pipeline of the localization process is shown in Figure 3.1.

The choice of motion model and observation model is fundamental and crucial for both Kalman filter's prediction step and measurement update. The more precise and comprehensive the model describes the system, the more accurate the estimate of the state will be.

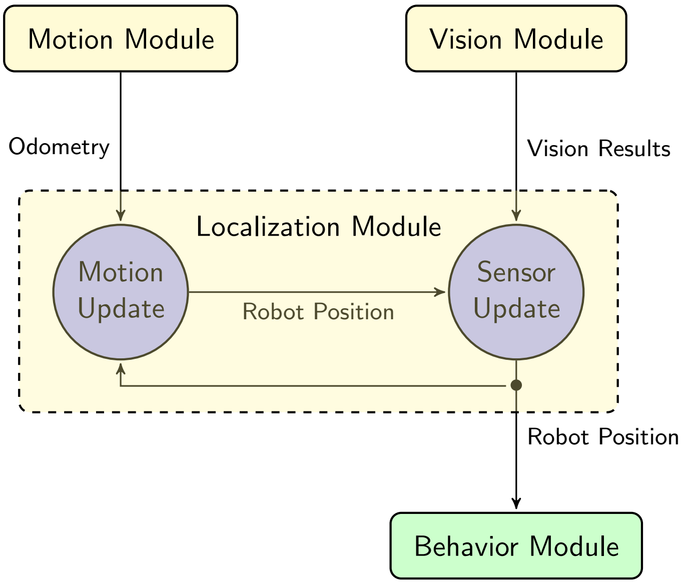

Motion Module
Vision Module
Odometry
Vision Results
Localization Module
Motion Update
Robot Position
Sensor Update
Robot Position
Behavior Module

Figure 3.1: Self-localization pipeline.

3.1. Motion Update

3.1.1. Motion Model

In general, motion models can be categorized by two models: velocity motion model and odometry motion model. Practical experience suggests that odometry, while still erroneous, is usually more accurate than velocity. Both suffer from drift and slippage, but velocity additionally suffers from the mismatch between the actual motion controllers and its mathematical model [13]. It is especially true for humanoid robot like NAO, whose moving velocity is difficult to obtain.

Since the transformation between the coordinates used by the odometry measurement and the physical world frame is unknown, the odometry measurement in this motion model is relative. To be specific, in the time interval $(t-1, t]$, the robot moves from a position x_{t-1} to position x_t, and meanwhile the odometry reports us a related movement from $\bar{x}_{t-1} = (\bar{x}, \bar{y}, \bar{\theta})$ to $\bar{x}_t = (\bar{x}', \bar{y}', \bar{\theta}')$. The bar here indicates that these are odometry measurements. We use the relative difference of $\bar{x}_{t-1}$ and $\bar{x}_t$ as an estimation of the difference between the true position $x_{t-1} = (x, y, \theta)$ and $x_t = (x', y', \theta')$. Therefore, the odometry information u_t can be given by the pair:

$$u_t = \begin{pmatrix} \bar{x}_{t-1} \\ \bar{x}_t \end{pmatrix} \tag{3.1}$$

Let Equation 3.2 denotes the rotation matrix with angle α, which is the notation assumed in the rest of the thesis.

$$\Omega(\alpha) = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} \tag{3.2}$$

In this motion model, to obtain the relative odometry change, u_t can be treated as a translation and then followed by a rotation. Figure 3.2 demonstrates the decomposition of the odometry measurement. δ_{trans} is the translation and δ_{rot} is the rotation after translation. Both the translation and the rotation are considered under the coordinate O, which is relative to $\bar{x}_{t-1}$. The relative translation and rotation is calculated using Algorithm 1.

Algorithm 1 subtract_poses $(\bar{x}_{t-1}, \bar{x}_t)$

```
1: $\begin{bmatrix} \delta_x \\ \delta_y \end{bmatrix} = \Omega(-\bar{\theta}) \begin{bmatrix} \bar{x}' - \bar{x} \\ \bar{y}' - \bar{y} \end{bmatrix}$
2: $\delta_{rot} = \bar{\theta}' - \bar{\theta}$
3: $return(\delta_x, \delta_y, \delta_{rot})$
```

Once the estimated translation δ_{trans} and rotation δ_{rot} relative to pose x_{t-1} is obtained, it can be applied to update the robot position in the physical world frame. The full motion update using odometry measurement is shown in Algorithm 2.

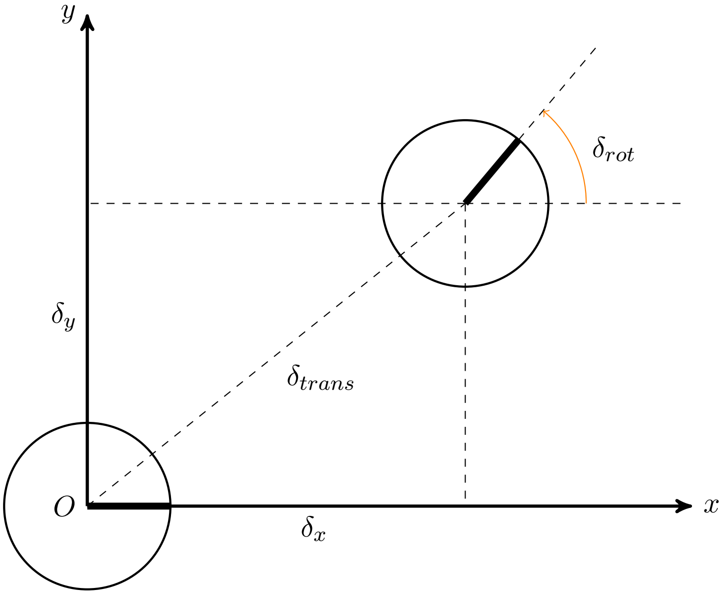


Figure 3.2: Relative odometry change. Translation δ_{trans}, rotation δ_{rot} are considered in coordinate system O.

Algorithm 2 odometry_motion_update(x_{t-1}, u_t)

```
1: $(\delta_x, \delta_y, \delta_{rot}) = substract\_poses(\bar{x}_{t-1}, \bar{x}_t)$
2: $\begin{bmatrix} x' \\ y' \end{bmatrix} = \Omega(\theta) \cdot \begin{bmatrix} \delta_x \\ \delta_y \end{bmatrix} + \begin{bmatrix} x \\ y \end{bmatrix}$
3: $\theta' = \theta + \theta_{rot}$
4: $return[x', y', \theta']^\top$
```

The function *odometry_motion_update* in Algorithm 2 corresponds to the transition function $f(x_{t-1}, u_t)$ described in EKF algorithm in Table 2.2. Then the related Jacobian matrix F_t can be calculated by:

$$F_t = \begin{bmatrix} 1 & 0 & -sin(\theta) \cdot \delta_x - cos(\theta) \cdot \delta_y \\ 0 & 1 & cos(\theta) \cdot \delta_x - sin(\theta) \cdot \delta_y \\ 0 & 0 & 1 \end{bmatrix} \tag{3.3}$$

3.1.2. Process Noise Model

The process noise covariance matrix Q_t is modeled to be directly proportional to the absolute change of odometry $(|\delta_x|, |\delta_y|, |\theta_r|)$. Since the more the robot moves, the more it suffers from unpredictable noises like slippage and drift. Equation 3.4 is proposed to calculate the process noise covariance matrix.

$$Q_t = \begin{bmatrix} |\delta_x| & 0 & 0 \\ 0 & |\delta_y| & 0 \\ 0 & 0 & |\delta_{rot}| \end{bmatrix} \cdot Sc^2 \cdot \begin{bmatrix} |\delta_x| & 0 & 0 \\ 0 & |\delta_y| & 0 \\ 0 & 0 & |\delta_{rot}| \end{bmatrix}^\top \tag{3.4}$$

$$Sc = \begin{bmatrix} 0.8 & 0.2 & 0.2 \\ 0.2 & 0.8 & 0.2 \\ 0.2 & 0.2 & 0.8 \end{bmatrix} \tag{3.5}$$

Sc is a scaling matrix and Sc^2 is the element-wise square of Sc. The values chosen for Sc was determined from a set of experiments by visual inspection. It intuitively means, when the robot walks 1 m in x or y direction, it has an error range of ± 0.8 m. When it rotates 1 rad, it has an error of ± 0.8 rad in orientation. The value of Sc depends on the accuracy of odometry measurement from the motion module, and a better tuning of the values can be argued.

With all the essential elements for motion model ready, we plug in the formula from Table 2.2 and perform the EKF prediction step for the localization algorithm.

3.2. Sensor Update

The robot can also measure its position by sensing the surrounding environment. In the case of NAO robot, the measurement is made by observing the environment from its two cameras. The images from the cameras are preprocessed by the DAInamite's vision module to extract the landmarks as shown in Figure 2.2. During the work of this thesis, two sensor update models are proposed to accomplish the localization task. One is based on an optimization approach and another is based on features. We will discuss both models in this section, but with more focus on feature based model.

3.2.1. Optimization Based Model

In the early phase of the thesis, the only unique landmark can be detected by vision module is the yellow goal posts, however, from 2015 the yellow goal posts will be replaced by white goal posts according to the rule [2]. Apart from yellow goal posts, the landmark can be detected in team DAInamite is the white line in the field. Field lines are of high ambiguity, in the case of feature based Kalman filter localization, a wrong matching of the landmarks could lead to the divergence of the tracking which can hardly be recovered. Thus, a robust and accurate algorithm for sensor update is implemented taking the reference from Lauer [12]. The algorithm is based on Rprop [14] optimization method, which uses only the line points from the detected lines, to find the robot position which matches the field map best.

One of the challenges to design Kalman filter using optimization based model is to define the covariance of the measurement model. Unlike a classic Kalman filter measurement update step, the optimization based model uses Rprop to refine the target variable by an iterative error minimizing process, in which no covariance is considered. The idea proposed in [12] is to use second order error derivative to approximate the value of covariance. However, without a consistent physical meaning of the covariance throughout the system, the motion model and sensor model can not be stably combined. Another challenge regarding this model is the intensive computation required by the optimization step which is not applicable on the robot.

3.2.2. Feature Based Model

Although the optimization based localization have relatively the same accuracy as the particle filter localization, it is not computationally efficient to run on the robot (comparison in section 5.3). During the preparation of German Open 2015, DAInamite team developed more feature detections like center circle detection, penalty area detection and "L" junction detection. Provided the richness of features, the development of feature based sensor update for localization is motivated.

3.2.2.1. "T" and "X" Junction Detection

In the SPL soccer field, there are 36 "L" junctions, 14 "T" junctions, 2 "X" junctions. As illustrated in Figure 3.3, the detection of "L" considers both sides of the field line as well as the line direction, but "T"and "X" junctions do not consider line side and direction. The counting includes "L", "T", "X" junctions at center circle. Compared to "L" junction, "T" and "X" junctions are much stronger landmarks. Especially the "T" junction at the penalty area, it can significantly help the robot localize itself in front of the goal.

Both the "T" and "X" junctions are detected in the robot frame instead of the image plane, since in the image plane the perpendicularity of the lines can not be easily observed. When the detected lines are projected to the robot frame, every two lines are compared according to their angles. If the differences of the angles are close to 90°, then the intersection point of the two lines are computed assuming the lines have infinite length. After the intersection is calculated, it should be satisfied that the intersection lies on at least one of the line segments. The distance from line end points to the intersection are further checked if they satisfy the threshold according to T or X geometry. The pseudocode of the TX junction detection is illustrated in Algorithm 3, and the detection result is shown in Figure 3.4.

3.2.2.2. Measurement Model Choice

Given the various kinds of features, different sensor models can be distinguished for different features. In general, they can be divided into point landmarks and line landmarks. Assume the robot position $x_t = (p_x, p_y, p_\theta)$, and the landmark represented as l in physical world frame. The objective is to describe the expected measurement $\bar{z}$ by the observation function $h(x_t)$ and derive the Jacobian matrix H.

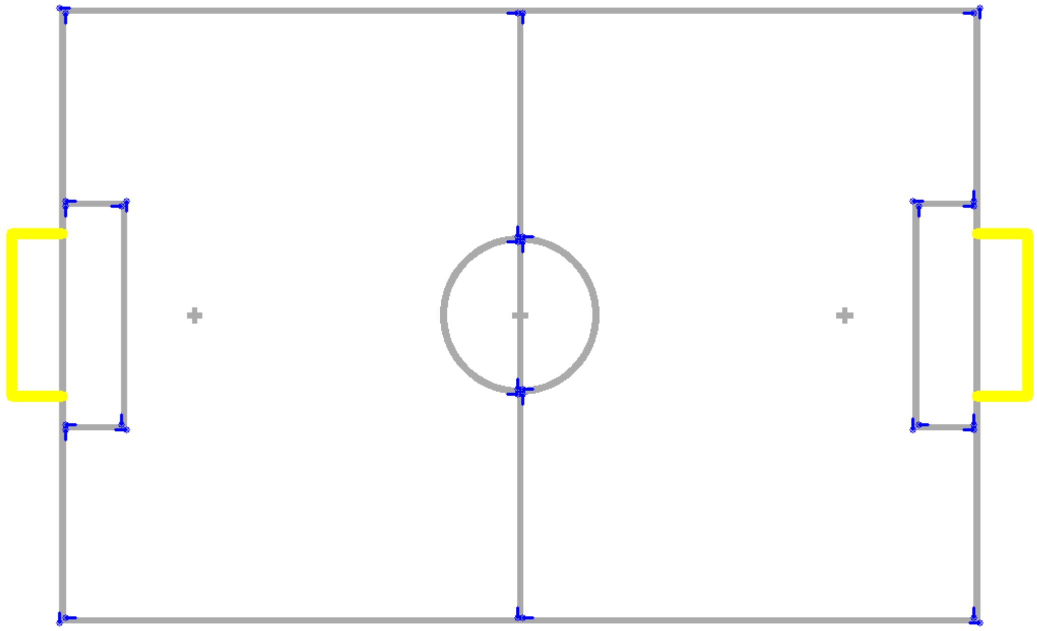


Figure 3.3: All L junctions in the field, the blue pointer indicates the position of the “L” junctions as well as their directions.

Point Landmark without orientation Under the Cartesian coordinate system in robot frame. The expected point landmark can be described as $\bar{z} = (\bar{z}_x, \bar{z}_y)^\top$ (illustrated in Figure 3.5, and regards the ball as a point landmark.) and its corresponding landmark in physical world frame $l = (l_x, l_y)^\top$. The observation function $h(x_t)$ can be described in Equation 3.6, which transforms the landmark from physical world frame to the robot frame. The Jacobian matrix H is the partial derivative of $h(x_t)$ at x_t, as illustrated in Equation 3.7.

$$\bar{z} = \begin{pmatrix} \bar{z}_x \\ \bar{z}_y \end{pmatrix} = h(x_t) = \Omega(-p_\theta) \cdot \left[\begin{pmatrix} l_x \\ l_y \end{pmatrix} - \begin{pmatrix} p_x \\ p_y \end{pmatrix} \right] \tag{3.6}$$

$$H = \frac{\partial h(x_t)}{\partial x_t} = \begin{bmatrix} -\cos p_\theta & -\sin p_\theta & -(l_x - p_x)\sin p_\theta + (l_y - p_y)\cos p_\theta \\ \sin p_\theta & -\cos p_\theta & -(l_x - p_x)\cos p_\theta - (l_y - p_y)\sin p_\theta \end{bmatrix} \tag{3.7}$$

The feature which follows this model is center circle without the center line being detected.

Point Landmark with Orientation When the point landmark has not only position but also orientation, it can be described as $l = (l_x, l_y, l_\theta)^\top$ in the physical world frame,

Algorithm 3 TX_junction_detection ()

```
for each $Line(i)$ do
    for each $Line(j)$ do
        if AngleDifference($Line(i)$, $Line(j)$) $\approx 90^\circ$ then
            $intersection \leftarrow$ Intersection($Line(i)$, $Line(j)$)
            if $intersection$ lies on both $LineSegment(i)$ and $LineSegment(j)$ then
                if CheckDistance($LineEndPoints(i)$, $LineEndPoints(j)$, $intersection$) within ThresholdX then
                    $LineSegment(i)$ and $LineSegment(j)$ is an X junction
                end if
            else if $intersection$ lies on either $LineSegment(i)$ or $LineSegment(j)$ then
                if CheckDistance($LineEndPoints(i)$, $LineEndPoints(j)$, $intersection$) within ThresholdT then
                    $LineSegment(i)$ and $LineSegment(j)$ is a T junction
                end if
            end if
        end if
    end for
end for
```

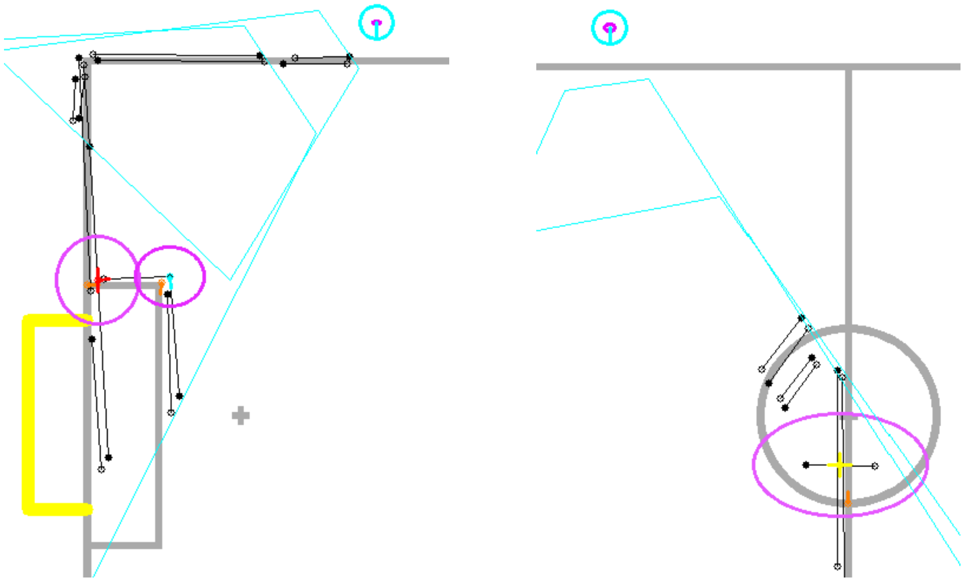
and the expected point landmark can be described as $\bar{z} = (\bar{z}_x, \bar{z}_y, \bar{z}_\theta)^\top$. With addition of orientation, the observation function $h(x_t)$ can be described in Equation 3.8 and the Jacobian matrix H in Equation 3.9.

$$\bar{z} = \begin{pmatrix} \bar{z}_x \\ \bar{z}_y \\ \bar{z}_\theta \end{pmatrix} = h(x_t) = \begin{pmatrix} \Omega(-p_\theta) \cdot \left[\begin{pmatrix} l_x \\ l_y \end{pmatrix} - \begin{pmatrix} p_x \\ p_y \end{pmatrix} \right] \\ l_\theta - p_\theta \end{pmatrix} \tag{3.8}$$

$$H = \frac{\partial h(x_t)}{\partial x_t} = \begin{bmatrix} -\cos p_\theta & -\sin p_\theta & -(l_x - p_x)\sin p_\theta + (l_y - p_y)\cos p_\theta \\ \sin p_\theta & -\cos p_\theta & -(l_x - p_x)\cos p_\theta - (l_y - p_y)\sin p_\theta \\ 0 & 0 & -1 \end{bmatrix} \tag{3.9}$$

The landmarks which comply with this measurement model are “L”,“T”,“X” junctions, and center circle with center line detected.

Line Landmark Line is a special landmark, it needs two end points to describe it in Cartesian space. However, with only two end points of the line, it is very hard to find the corresponding points in the field map. For example, a short line can be matched to any segment within a long line. Therefore, to counter this ambiguity and extract the intrinsic property from lines, the representation from Hough transform is used where a line can be represented as (ρ, θ), as shown in Figure 3.6. While Hough representation discards the length information of the line, it keeps the direction and distance information from



(a) "L", "T" junction detection.

(b) "X" junction detection.

Figure 3.4: Detection of "L"(Blue), "T"(Red) and "X"(Yellow) junctions in the field. The purple ellipses indicate the measurement covariance. The orange pointers indicate the correspondent junctions on the field.

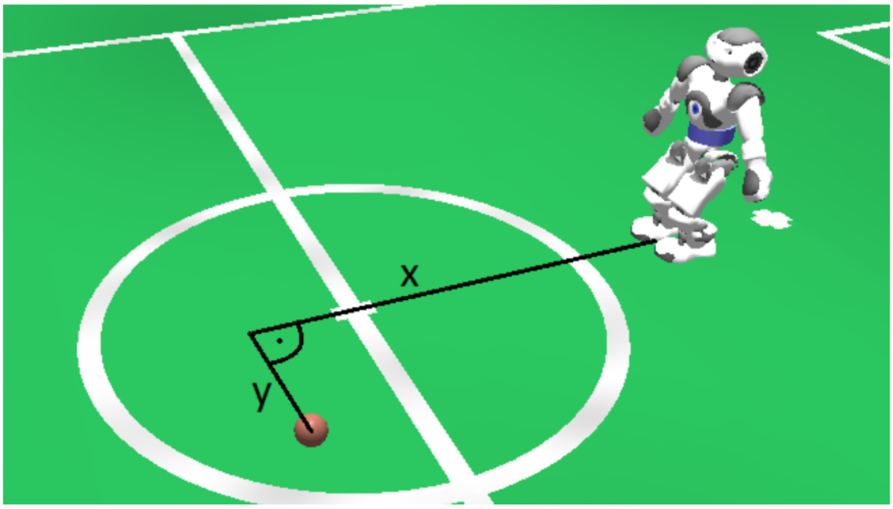


Figure 3.5: Observation given in Cartesian coordinates. [15]

the line intact. It means, for example, when a vertical line is matched, the robot's orientation can be determined and it will also be sure about its position in x axis, but keep uncertain about its position in y axis until a horizontal line is matched.

Using Hough representation, the expected line landmark can be described as $\bar{z} =$

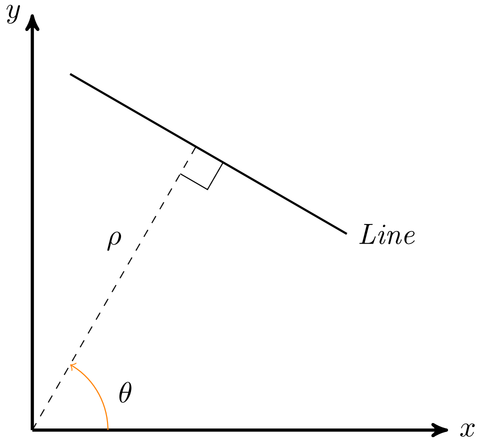


Figure 3.6: Line represented in Hough space.

$(\bar{z}_\rho, \bar{z}_\theta)^\top$, and depending on whether the corresponding line in the field map is vertical or horizontal, different measurement models should be adjusted. If the matched line is horizontal, the line landmark in physical world frame could be described using only y axis value $l = l_y$. Then the observation function $h(x_t)$ can be described in Equation 3.10 and the Jacobian matrix H in Equation 3.11.

$$\bar{z} = \begin{pmatrix} \bar{z}_\rho \\ \bar{z}_\theta \end{pmatrix} = h(x_t) = \begin{pmatrix} |l_y - p_y| \\ atan2(\cos p_\theta \cdot (l_y - p_y), \sin p_\theta \cdot (l_y - p_y)) \end{pmatrix} \tag{3.10}$$

$$H = \frac{\partial h(x_t)}{\partial x_t} = \begin{bmatrix} 0 & \frac{l_y - p_y}{|l_y - p_y|} & 0 \\ 0 & 0 & -1 \end{bmatrix} \tag{3.11}$$

Similarly for vertical lines, the line landmark in physical world frame can be described using only x axis value $l = l_x$. The observation function $h(x_t)$ can be described in Equation 3.12 and the Jacobian matrix H in Equation 3.13.

$$\bar{z} = \begin{pmatrix} \bar{z}_\rho \\ \bar{z}_\theta \end{pmatrix} = h(x_t) = \begin{pmatrix} |l_x - p_x| \\ atan2(\cos p_\theta \cdot (l_x - p_x), -\sin p_\theta \cdot (l_x - p_x)) \end{pmatrix} \tag{3.12}$$

$$H = \frac{\partial h(x_t)}{\partial x_t} = \begin{bmatrix} \frac{l_x - p_x}{|l_x - p_x|} & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix} \tag{3.13}$$

The features that fall into this measurement model are the detected lines as well as the penalty area. Because the penalty area detection basically is just the detection of two parallel lines which satisfies certain criteria. The penalty area detection result is given in the form of Hough representation, thus it is also treated as a line.

3.2.2.3. Measurement Noise Model

The measurement noise indicates how accurate the measurement made for each feature is, and by assuming the noise distribution is Gaussian, the measurement noise covariance is represented by the term R_t in Table 2.2. The measurement noise in case of NAO robot in SPL game mainly due to:

- noise of the camera matrix.
- blur of the image due to motion.
- discretization of the physical environment into the pixels of an image.
- distortion of the image due to rolling shutter of the camera.

While many kinds of noise listed above result from the hardware intrinsic of the camera, to compensate these error, specific error models have to be developed for each kind. In this thesis, we focus only on developing the noise model for noise from the camera matrix.

Since all the landmarks are measured in robot frame, the further the landmarks from the robot (the origin of robot frame coordinate) are, the larger the measurement noise will be. Because, the error introduced by the camera matrix will be magnified when the detected landmarks are projected from image plane to robot frame. Assuming in 2D situation and the camera matrix has an error of 0.02 radian in angle, the corresponding measurement noise for a point landmark can be approximated in Figure 3.7. This phenomenon is demonstrated in Figure 3.8 and 3.9. For example, the line (both sides) which is being detected from the penalty area in Figure 3.8 become two lines with big gap (shown in Figure 3.9) when projected to the robot frame. Both lines suffered significant error in distance.

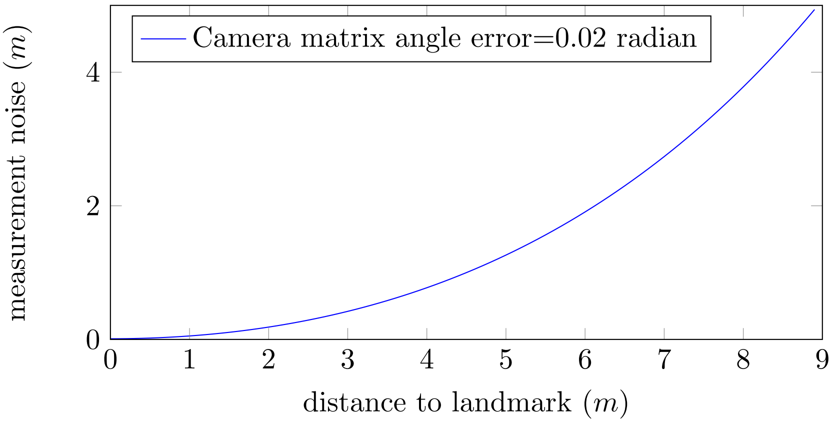


Figure 3.7: Measurement noise with respect to landmark distance in robot frame.

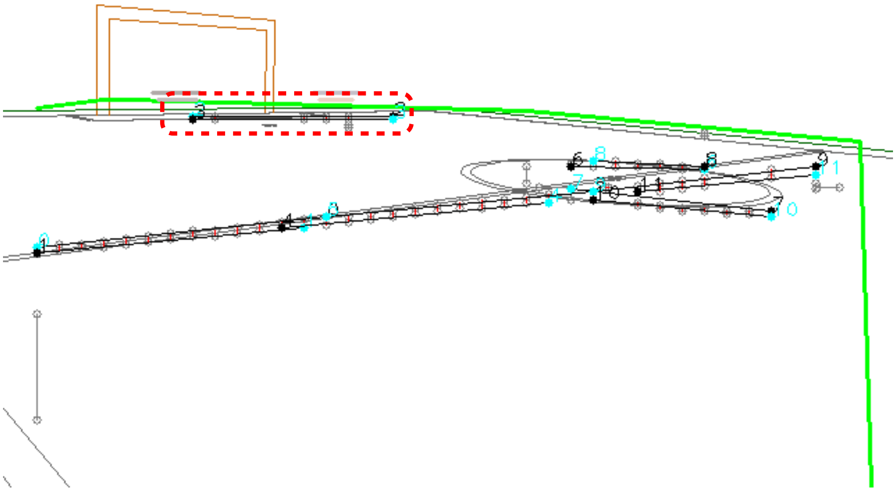


Figure 3.8: Landmarks in image plane. Detected field border are indicated by a long green line. Detected lines are indicated in black with starting point in blue dot and end point in black dot. The red box indicates the parts which are detected from the front line of the penalty area.

In this thesis, the noise model which is used by team B-Human is adopted [16]. The basic idea of their noise model is illustrated in Figure 3.10. In Figure 3.10, a point landmark without orientation is observed at location L in robot frame xOy. In this model, it assumes the robot has certain angle noise deviation in camera matrix's pitch and yaw, namely θ_x and θ_y. By adding the angle noise deviations and re-projecting the landmark onto the ground, the measurement noise indicated by the purple line LB and LA can be calculated.

On the other hand, line landmark is represented as (ρ, θ) according to its measurement model. Its corresponding measurement noise is calculated by taking the center point of the detected line landmark and using it as a point landmark to calculate the noise. The noise in x direction is regarded as the measurement noise for ρ. Based on the assumption that the longer the detected line is, the more confidence the robot will have for line angle θ, the measurement noise for θ is calculated by $atan(measurementNoise(\rho)/halfLineLength)$.

In the end, for point landmarks with orientation, the noise model of point landmark mentioned above is firstly applied. Then the measurement noise in orientation can be obtained by taking the line from the landmark which determines the landmark's orientation, and calculate its noise in angle using the noise model for line landmark.

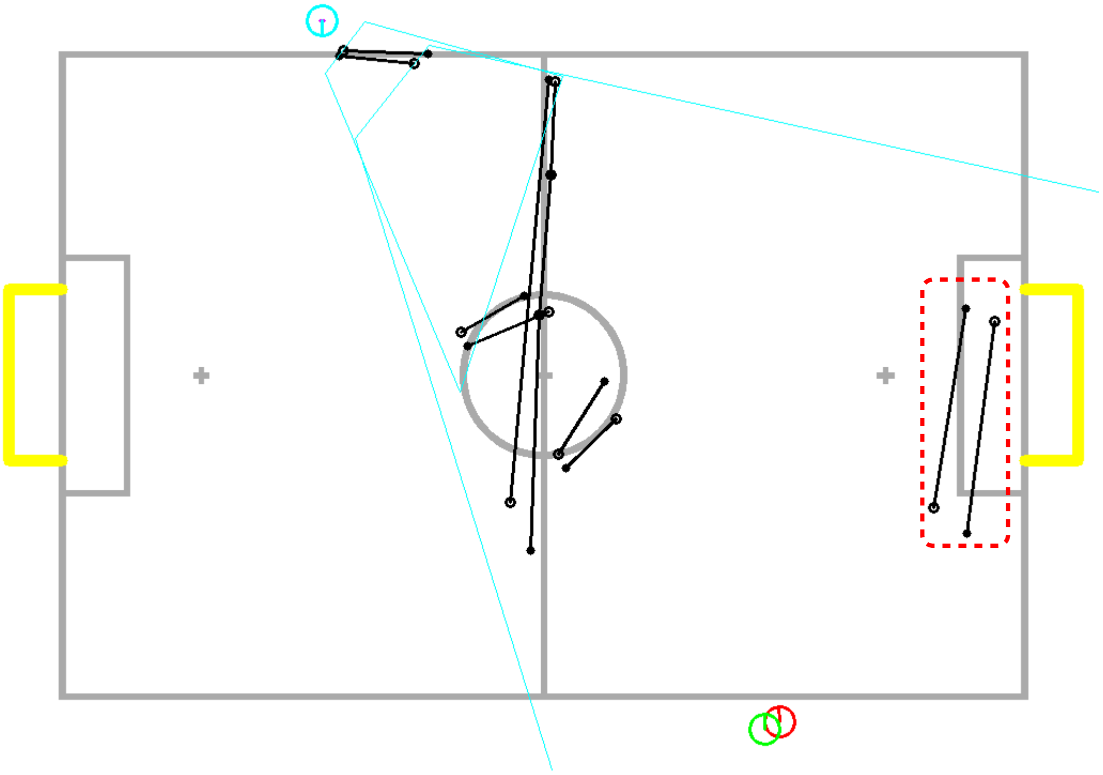


Figure 3.9: Landmarks in robot frame. Black lines indicate the detected lines after projection, the red box indicates the lines which are detected from the front line of the penalty area.

3.2.2.4. Landmark Correspondence

Due to the high ambiguity of the features in SPL, one of the difficulties of using Kalman filter for localization is to find the correspondence between the observations and the landmarks. In other words, it is to find the corresponding landmark l in physical world frame, when a landmark z_t (in Equation 2.11) is observed in robot frame.

Center Circle The only unique landmark that currently can be used is the center circle with center line detected. Its correspondence is easily fixed to $l = (0, 0, 0)$, meaning position $(0, 0)$ in physical world coordinate and 0° in orientation. However, because the center circle is symmetrical, it can not provide the information of the field side, so the robot could be in either one position or its mirrored one. In this case, the side of the current robot position is considered when using center circle to update robot position, then the robot position can be uniquely defined. With the help of unique landmark like center circle, the global localization problem stated in subsection 1.2.3 can be solved.

On the other hand, when the center circle is detected without the middle line, the robot position can still not be decided, since it can be at any position surrounding the

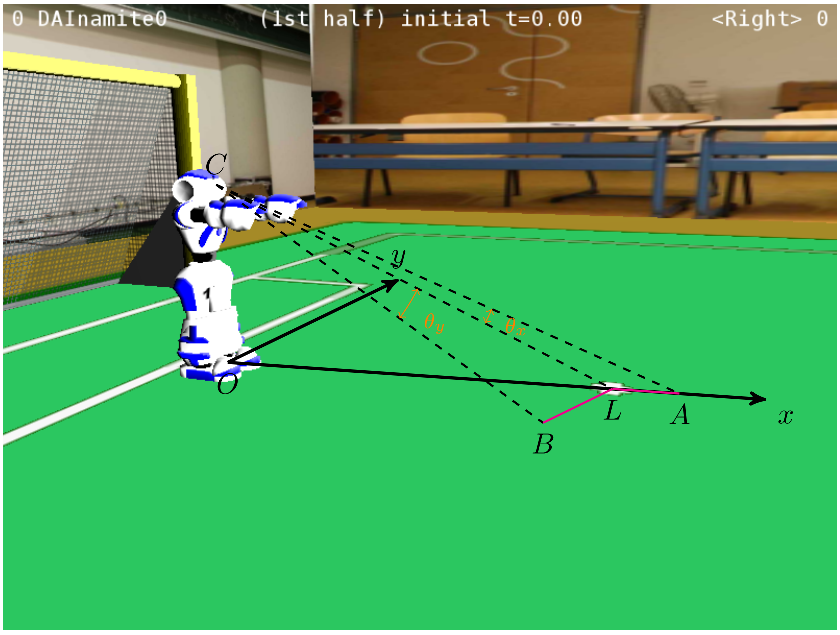


Figure 3.10: Measurement covariance of the point landmark. θ_x and θ_y are the measurement angle deviations in x, y direction. LA and LB indicate the measurement noises.

center circle. So in this case, the center circle can only be treated as a point landmark without orientation and its correspondence is $l = (0, 0)$.

Junctions For junction features which include “L”, “T”, “X” junctions, they are ambiguous features and the detected junctions are represented as point landmark with orientation $z_t = (z_x, z_y, z_\theta)$. Within the junctions, “L” junctions are the most ambiguous ones, with 36 appearances in the field (illustrated in Figure 3.3). It means one detected “L” junction can be matched to 36 “L” junctions in the field. “T”, “X” junctions are less pervasive, with 14 and 2 appearances respectively. Given the ubiquitous distribution of the junctions in the field, it is hard to find the correspondence between the junctions seen and the junctions in the field.

In this thesis, a *nearest neighbor search method with threshold* is used to find the junction correspondences. Assume the robot position is not of high inaccuracy, then we transform the junctions from robot frame into physical world frame. Given the junction in physical world frame, the associative correspondence is found by searching the nearest junction in the field.

Instead of looping through all the possible junctions, the k-d tree algorithm [17] is adopted to do the search efficiently. Two k-d trees are constructed, one for “L” junctions and one for “T” junctions. No k-d tree is needed for “X” junction, since the “X” junctions can be compared directly with the two in the field. The distance of measurements between the junctions is Euclidean distance.

Once the associative junction is found by nearest distance, the distance and the difference in orientation are checked. If the distance to the associative junction is larger than a threshold[1], or the difference in orientation is exceeding 45°, the associative junction is discarded and this landmark will not contribute to the robot position update. Because in this case, the assumption that current robot position is relatively accurate no longer holds, more than one possible correspondences can be chosen. In other words, *nearest neighbor search with threshold* acts more like a local position optimization to track the robot pose rather than global localization. In section 4.2, we will see the discarded landmarks can be utilized for global localization by using multi-hypotheses Kalman filter. The matching result has been shown above in Figure 3.4, in which the orange pointer indicates to which junction the detected junction is corresponding.

Lines Lines are the most ambiguous landmarks among all the landmarks in the field, because a single line can be matched to any field line in the field. Due to the high ambiguity, the lines also can not be used for global localization if features are not extracted from them. To find the line correspondence, a similar methodology as the nearest neighbor search is adopted. Each detected line is represented by a vector of two end points $z_t = (P_{start}, P_{end})$. If the nearest neighbor search is directly applied, the two end points could be erroneously matched to two different field lines, as illustrated in Figure 3.11.

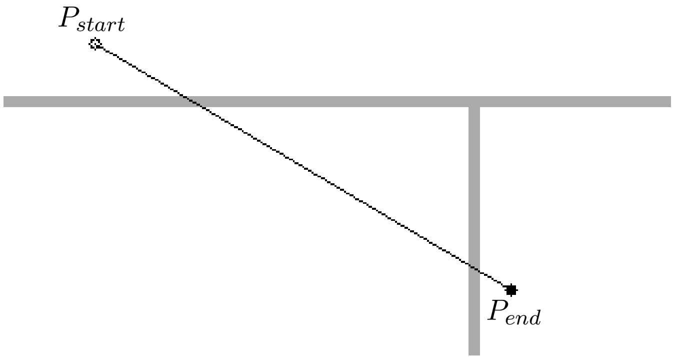


Figure 3.11: The end points of the detected line (black) would be matched to two different field lines using nearest neighbor search, which is not the desired result.

Inspired by the method used by B-Human [16] to find line correspondence, a *nearest line neighbor search* is implemented. Given a detected line, after projecting it to physical world frame, it is compared with each of the field lines to check if certain criteria are satisfied. Suppose we start with the outer side of the horizontal field line (indicated by red in Figure 3.12), the criteria are as follows:

---

[1] The threshold depends on which associative junction on the field is matched. For example, the junctions from the side lines will have larger threshold than the ones from the center circle.

- Find the nearest distance from the end points of detected line to the field line, in this case, distance $P_{start}A$ and $P_{end}B$ are checked. They should be below a threshold[2] respectively, otherwise, this field line is not the correspondence.

- Start from each end point of the detected line, make a perpendicular line pointing to the field line, *i.e.* line $P_{start}C$ and $P_{end}D$. If both perpendicular lines have intersections with the field line, the distance of $P_{start}C$ and $P_{end}D$ are checked. Each of them should also be below the threshold. Moreover, if either orthogonal line starting from the end point fails to have an intersection with the field line. This field line is regarded as not the correspondence. (For example, P_{start} fails to have an orthogonal intersection with the vertical field line in Figure 3.12)

- When both of the criteria above are satisfied, the direction of the detected line is checked with the field line. They should have more or less the same direction. And this is done by dot multiplication of the two vectors, the one with result larger than 0 satisfies.

- Only the field line which satisfies all the criteria above is treated as the correspondence. If more than one field line happen to satisfy the criteria, then all of them are discarded due to the ambiguity.

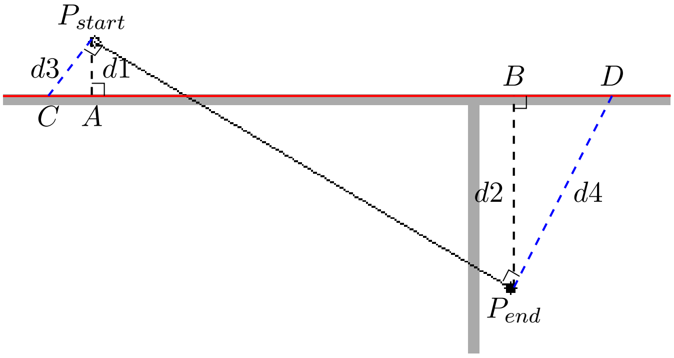


Figure 3.12: The criteria of determining the line correspondence.

Another important property of finding correspondence of the lines is that short lines can match to long lines and short lines, but long lines can not match to short lines. Thus, the long lines and short lines are distinguished, the lines with length exceeding the penalty area width (defined in Table 1.1) are treated as long lines, the rest are short lines. With this classification, long lines are limited to the four field lines on the border and the one field line in the center. The benefit is discernible, not only the process of *nearest line neighbor search* can be speeded up for long lines, but also the threshold in the matching criteria can be enlarged, since the ambiguity is reduced.

---

[2] The threshold is chosen to be half of the penalty area length defined in Table 1.1, since it is the largest distance that can distinguish the ambiguity between lines.

While each line detected from a vertical or horizontal field lines can have a correspondence, the lines detected from the center circle can hardly have. The center circle can be treated as consisting of infinite short lines, and each line can be of different length. The algorithm implemented checks if a short line is in the vicinity of the center circle, if so, the line is simply discarded. Therefore, in this case, localization depends more on center circle detection.

Penalty Area As we discussed earlier in section 3.2.2.2, detected penalty area can be regarded as a line. We can assume on the field, there is a virtual line which is vertical and goes through the center of each penalty area box. Then the problem becomes the matching of the penalty area with the virtual lines. Unlike the *nearest line neighbor search* for lines, the penalty area matching criterion is simpler. Since there are only two penalty area virtual lines in the field, the nearest distance from the center of the detected penalty area to both virtual lines are calculated. The virtual line with shorter distance becomes the correspondence. The matching result for both detected lines and penalty area is illustrated in Figure 3.13.

3.2.2.5. Multiple Simultaneous Measurements

In practical implementation of the Kalman filter, the routine of one motion update and one sensor update described in theory may not hold. Instead, there can be multiple motion updates without sensor updates, if no result is obtained from the observation. Or on other cases, several observations are made at once, since several landmarks are extracted in the same time step. Assuming the landmarks extracted from the observations are all point landmarks without orientation, according to the measurement noise model discussed in subsection 3.2.2.3, the measurement noise covariance can be represented as a 2-dimensional matrix in Equation 3.14

$$R = \begin{bmatrix} r_x^2 & 0 \\ 0 & r_y^2 \end{bmatrix} \tag{3.14}$$

The stochastically correct sensor update for n detected landmarks is to execute a $2n$-dimensional measurement update. However, when the landmarks are stochastically independent of each other, the off-diagonal inter-landmarks entries of the $2n \times 2n$ measurement covariance are zero. Then a single $2n$-dimensional measurement update is approximated well by n 2-dimensional updates [15]. Although, the assumption of the observed landmarks are independent of each other is not completely true in SPL game, in this thesis, the sensor update is performed independently for each observed landmark. As a result, it sacrifices the localization accuracy as a trade-off to increase the computation efficiency.

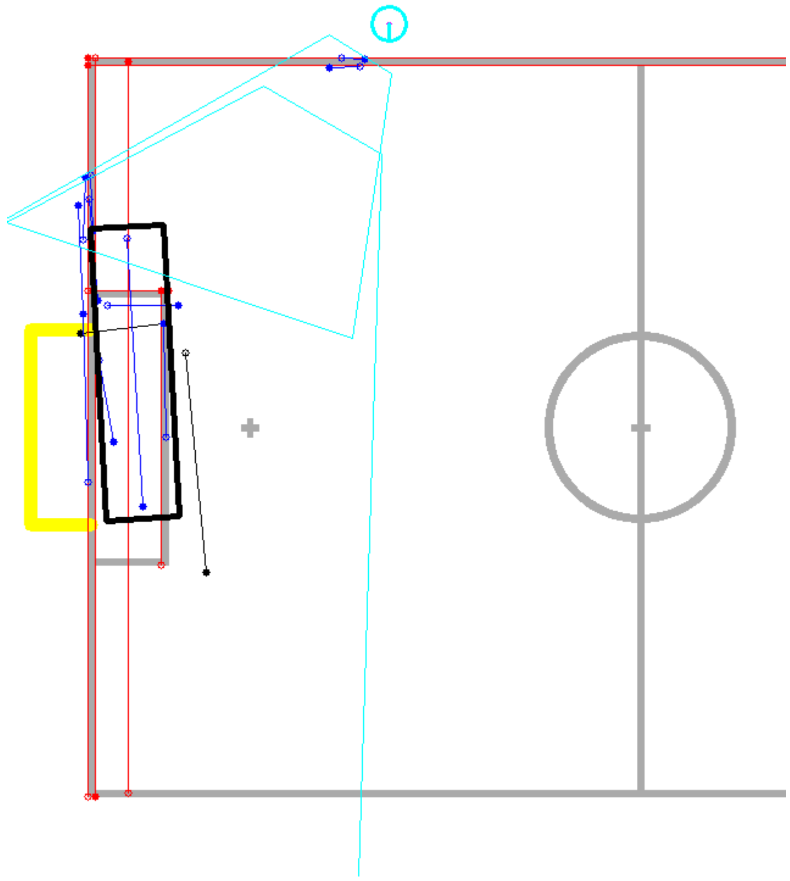


Figure 3.13: Matching result of line and penalty area. The black box indicates the detected penalty area. The line in blue indicates it is matched with the field line in red. The line in black indicates it fails to find a correspondence.

4. Multi-Hypotheses Kalman Filter

While the Kalman filter can only deal with uni-modal probability distribution, it does not suffice to handle the ambiguous landmarks that occur in robot's observation, which requires a multi-modal distribution. Therefore, a multi-hypotheses Kalman filter is adopted to handle the ambiguous situations by describing the robot position in the field by a Gaussian sum distribution. The problem of multi-hypotheses Kalman filter lies mainly in the domain of when and where to add new hypothesises into the Gaussian sum distribution, as well as when to prune or merge certain hypotheses to restrain the number of hypotheses within a limit, so it does not consume too much of the computation resources.

4.1. Hypothesis Model Weighting

As illustrated in subsection 2.1.2.3, each normal probability distribution from the Gaussian sum distribution is represented as Equation 2.14. In addition to uni-modal Gaussian distribution, the weight α_i has to be determined for each Gaussian distribution model i in the Gaussian sum distribution. The weight, at the same time, illustrates the quality of the state hypothesis. Possible update method like Equation 2.16 has been proposed in [8]. However, this update method is only suitable for landmarks with known correspondence. If the correspondence is unknown or ambiguous, like in our case, the "T" junctions, the expected observation $\widehat{z_i}$ can not be determined.

In this thesis, a voting buffer is used to measure the weight of each model. The voting buffer is constructed by a First In, First Out (FIFO) circular buffer of size 60. The basic idea is to vote 1 to the buffer when the observed landmark is matched, and 0 when not matched. The matching of landmark correspondence is using the nearest neighbor algorithm with threshold discussed in subsection 3.2.2.4. The advantage of using a circular buffer is that, after 60 observed landmarks, regardless matched or not, the old voting in the buffer will be flushed and will not be counted. In other words, it tries to approximate the current robot state by keeping the memory of the most recent history. The model weight is determined by the average value of the voting buffer. Therefore, the more matches with landmarks the Kalman model makes, the higher weight it will have.

Unlike the implementation of particle filter that the total number of particles are fixed, in multi-hypothesis Kalman filters, the number of hypothesis model is dynamic. The adjustment made here is that the weights of all the Gaussian models do not add up to

1, instead they behave independently. So the weights of the existing hypothesis models will not be affected by the newly generated ones.

As discussed in subsection 3.2.2.4, some landmarks are unique landmarks, *i.e.* center circle, and some are ambiguous landmarks, *i.e.* junctions and lines. To depict the quality of the robot localization state, the vote should depend only on the globally unique landmarks instead of ambiguous landmarks. Assume a scenario, where the robot is not moving and constantly observing a “T” junction, and the “T” junction is perfectly matched with its nearest neighbor in the field. In this case, the state of the robot position is local optimal. The weight of the model will keep increasing and reach 1, but this weight is not truly describing the quality of localization state in global.

To better express the localization state globally, in the localization algorithm, landmarks are classified differently to update the weight. In observation, center circle with/without orientation and penalty area are treated as unique landmarks, they can directly vote 1 to the voting buffer when the threshold requirement in the nearest neighbor algorithm is satisfied, or vice versa.

On the other hand, “L”, “T”, “X”junctions are classified as non-unique landmarks, the voting buffer can be voted by 1, only when at least two junctions are matched, and in addition they must each belong to different junctions in the field. The junctions under consideration do not distinguish “L”, “T” or “X”, and they can come from different observation frames, as long as there is no failure match (beyond threshold) in between. By using the combination of the different observations of junctions, we assume it represents a global unique landmark, thus helping to overcome the local optimal problem. While the requirement of voting positive using junctions is strict, the condition of voting 0 is the same as the unique landmarks, *i.e.* when a failure match happens for the junction, a 0 will be voted to the buffer.

For the observed line landmarks, they are currently not contributing to the voting buffer. To utilize the lines, certain feature structures need to be extracted from them, in order to use the same strategy above to update the voting buffer. However, the extraction of other features out of the lines is not an easy task, further work may be required in this regard in the future.

4.2. Landmark Based Resampling

Similar to the augmented particle filter’s [13] sensor reseting step to recover the situation of robot getting kidnapped, multi-hypotheses Kalman filter uses sensor reseting based on landmarks to recover from position tracking failure. Assume uni-modal Kalman filter for position tracking, the filter can lose track of the robot’s position when the robot is hit by another robot which results in an error in robot’s orientation; or the robot is kidnapped by a referee during manual replacement, etc. Augmented particle filter initiates the sensor resetting step to generate random particles when the fluctuation of the weights from the particles is high, multi-hypotheses Kalman filter designed in this thesis will start resampling when an observed landmark fails to match the one on the field. Here, a correspondence match failure is regarded as a signal for potential loss of position tracking.

The resampling is achieved by calculating all the possible poses of the robot with respect to the observation of the landmark. The landmarks which can trigger resampling are the landmarks with orientation, *i.e.* center circle with center line and all kinds of junctions. The landmarks without orientation like lines, center circle without center line and penalty area can generate infinite possible robot positions, therefore currently they are not used for resampling. The problem of calculating the possible poses can be formulated as follows:

Given an observed landmark $l_R = (x_R, y_R, \theta_R)$ in robot frame, and assume its correspondent landmark $l_G = (x_G, y_G, \theta_G)$ in physical world frame, the corresponding robot pose in physical world frame is what is needed to be calculated. The calculation concerns mainly with the coordinate frame transformation which can be described by homogeneous transform matrix.

A homogeneous transform matrix in our context is a 3×3 matrix. It is structured by a 2×2 rotation matrix R_B^A and a 2×1 translation matrix P_B^A denoted in Equation 4.1.

$$T_B^A = \begin{bmatrix} R_B^A & P_B^A \\ \mathbf{0} & 1 \end{bmatrix} \tag{4.1}$$

The sub and super-script in R_B^A indicate that it is the rotation of frame B relative to frame A. Translation matrix P_B^A denotes the translation of the origin of frame B in frame A. In the following equations, we represent the robot pose in physical world frame in the form of homogeneous transform matrix as T_R^G, landmark in robot frame as T_L^R, landmark in physical world frame as T_L^G. According to *Composition Rule for Homogeneous Transformations*, the transformation between T_R^G, T_L^R and T_L^G can be described in Equation 4.2.

$$T_R^G \cdot T_L^R = T_L^G \tag{4.2}$$

To obtain T_R^G, we multiply $(T_L^R)^{-1}$ from the right for both side of the equation, then obtain Equation 4.3.

$$T_R^G = T_L^G \cdot (T_L^R)^{-1} \tag{4.3}$$

Substituting the homogeneous transform matrix using Equation 4.1, and calculating the inverse of matrix, results in the following:

$$T_R^G = \begin{bmatrix} R_L^G & P_L^G \\ \mathbf{0} & 1 \end{bmatrix} \cdot \begin{bmatrix} (R_L^R)^\top & -(R_L^R)^\top P_L^R \\ \mathbf{0} & 1 \end{bmatrix} \tag{4.4}$$

$$T_R^G = \begin{bmatrix} R_L^G (R_L^R)^\top & -R_L^G (R_L^R)^\top P_L^R + P_L^G \\ \mathbf{0} & 1 \end{bmatrix} \tag{4.5}$$

By using the rotation matrix representation stated in Equation 3.2, the rotation matrix and translation matrix in T_R^G is calculated in Equation 4.6 and Equation 4.7.

$$R_R^G = R_L^G (R_L^R)^\top = \Omega(\theta_G - \theta_L) \tag{4.6}$$

$$P_R^G = -R_R^G P_L^R + P_L^G = -\Omega(\theta_G - \theta_L) \cdot \begin{bmatrix} x_R \\ y_R \end{bmatrix} + \begin{bmatrix} x_G \\ y_G \end{bmatrix} \tag{4.7}$$

Therefore, given the correspondence of l_R and l_G, the robot x-y position is P_R^G and orientation is $\theta_G - \theta_L$. The number of possible robot poses generated depends on the occurrence of the landmarks in the field. For example, a mismatch of a “T” junction would trigger resampling which would generate 14 possible robot poses. The weight for the newly generated hypothesis model is 1 divided by the number of occurrences of the landmarks in the field. So, in the case for resampling by “T” junction, the weight for the new hypothesis model is $1/14 = 0.0714$. However, the model resampled by center circle is an execution as only one model will be generated. The weight of this model will be directly proportional to the detection confidence of the center circle.

4.2.1. “L” Junction Look Up Table

A special case worths discussing is the observation of the “L” junctions, as there are 36 occurrences of “L” junctions in the field, a match failure can resample as much as 36 possible robot positions, the weight for the newly generated hypothesis model will be $1/36 = 0.027$ which is extremely low. Such low weight model can be easily pruned in the pruning step which we will discuss in section 4.4. However, due to the large number of occurrences of “L” junctions, the possibility of observing more than one “L” junctions at the same frame is also high. When more than one “L” junctions are observed in one frame, the number of possible positions can also be reduced. Given the “L” junctions observed, the implementation to generate possible positions due to multiple “L” junctions is described in the following steps:

1. Take one observed “L” junction, use Equation 4.6 and Equation 4.7 to generate 36 possible robot positions based on all the “L” junctions in the field.
2. Take another observed “L” junction, at each robot position generated in Step 1, project the “L” junction to the physical world frame.
3. Check if at the place where the observed “L” junction is projected, there exists an “L” junction in the field. If so, this generated robot position is valid, otherwise it is an invalid position.
4. If there are remaining “L” junctions in observation, steps from Step 2 are repeated. Otherwise, the possible robot positions which are valid are returned.

To speed up the computation, Step 3 above is implemented using a pre-computed look up table. Given a pose (x_l, y_l, θ_l) of the “L” junction in physical world frame, the look up table will return boolean value, *i.e.* true or false, depending on whether in the vicinity of pose (x_l, y_l, θ_l), there exists an “L” junction. Although it is a 3-dimensional look up table, the resolution can be configured coarse, so it will not consume too much memory. In return, the checking of 36 “L” junctions in physical world frame can be done together

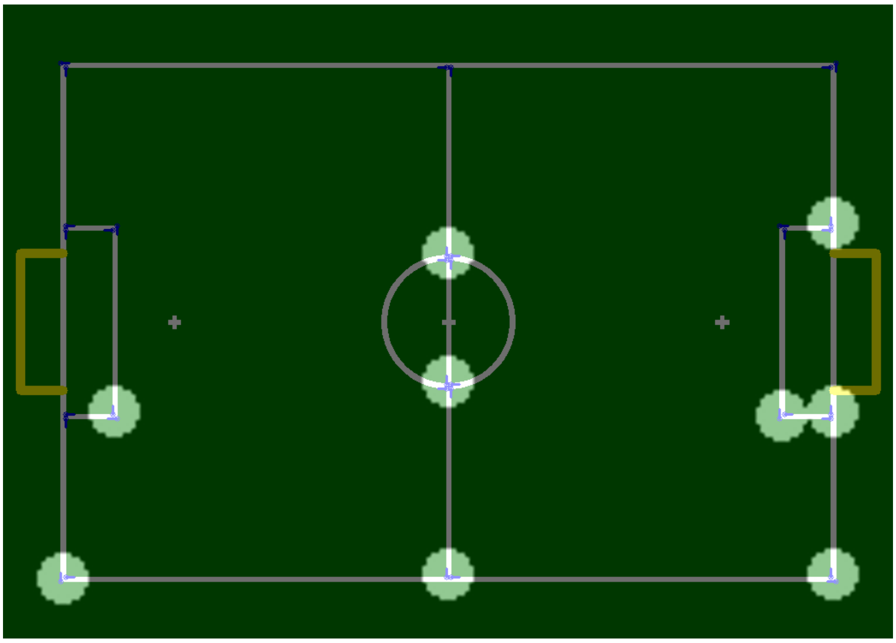


Figure 4.1: “L” junction look up table with angle dimension at 90°. Bright parts indicate true, dark parts indicate false

at once. A visualization example of the look up table is illustrated in Figure 4.1. Since it is a 3-dimensional look up table, Figure 4.1 only shows the look up table when the angle dimension is at 90°. It means, if given an “L” junction in physical world frame with angle being 90°, only the “L” junctions which fall into the bright area is valid.

To further reduce the number of sampled positions by “L” junction, we retain only the positions which are close enough to the last robot position. With the filtering of the positions generated by “L” junctions, the weight of the sampled position is set to be the same as the ones generated by “T” junctions.

4.3. Best Hypothesis Model and Confidence

In multi-hypothesis Kalman filter, there are multiple hypothesis models to form the possibility distribution. From the robot position probability distribution, one position $(x_r(t), y_r(t), \theta_r(t))$ has to be determined as the end result of localization algorithm. In the algorithm designed, the position is chosen to be the mean of the “best” hypothesis model in the probability distribution. The “best” hypothesis model should satisfy the following two criteria:

- Firstly, the hypothesis model should be globally best, which can be indicated by the weight of the model, the one with the highest weight is believed to be globally the best.

- Secondly, when multiple models have the same weights, then the quality of local belief is compared which can be indicated by the covariance of the Gaussian model. The one with the lowest covariance is believed to be the best.

Covariance describes the confidence of the belief of the current Gaussian model. To compare the covariance value between the models, it is re-structured as *Cov* in Equation 4.8. *Cov* is the sum of the diagonal elements of the covariance matrix. Since the error in orientation can cause more negative influence on robot position, *i.e.* the larger the error in orientation, the further the projected observed landmark will be away from its global correspondence, thus harder to match. Therefore, the variance value for orientation is doubled to have a higher weight in comparison.

$$Cov = Cov_x + Cov_y + 2 * Cov_\theta \tag{4.8}$$

When the "best" model is chosen, its covariance is also set to be the covariance of the robot position, and the weight is set to be the confidence for the position.

4.4. Model Pruning

While the resampling step generates hypothesis models into the Gaussian sum distribution, pruning step is also necessary to remove the hypothesis models which are redundant or have little contribution to the whole distribution.

4.4.1. Pruning by Weight

When the weight of hypothesis model is lower than a certain threshold, this model is considered to have little contribution to the distribution, so that it can be removed. Moreover, the maximum number of models is limited to a number N set by the localization algorithm. Once the maximum number of models is exceeded, the models are sorted by weight from high to low in a list. Only first N models in the list will remain, the others will be deleted. In this thesis, the maximum number of models is set to be 16.

4.4.2. Merging by Mahalanobis Distance

When the hypothesis models are close enough to each other, in other words, they represent almost the same probability distribution, they could also be merged into one. One way is to apply Euclidean distance directly between the mean of the models to calculate the distance as shown in Equation 4.9. μ_1 and μ_2 represent the mean of two different Gaussian distributions.

$$D_{euclidean} = \sqrt{(\mu_1 - \mu_2)^T(\mu_1 - \mu_2)} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (\theta_1 - \theta_2)^2} \tag{4.9}$$

However, the drawback of using Euclidean distance is that the covariance information of the distribution is not utilized. In this thesis, an adapted version of Mahalanobis distance is proposed to measure the distance between two hypothesis models. Mahalanobis distance is defined as follows in Equation 4.10 [18], which uses covariance as one factor to calculate the distance. In Equation 4.10, x is the observation, μ is the mean of the distribution and S is the covariance matrix.

$$D_M(x) = \sqrt{(x-\mu)^T S^{-1}(x-\mu)} \tag{4.10}$$

Given two Gaussian distributions G_1 and G_2, where G_1 has mean and covariance (μ_1, S_1) and G_2 has mean and covariance (μ_2, S_2), the formula in Equation 4.10 can not be directly applied to measure the distance. In order to measure the distance between G_1 and G_2, assume μ_2 is the observation for G_1, and μ_1 is the observation for G_2, therefore, by combining two Mahalanobis distances, the adapted Mahalanobis distance to measure distance between two Gaussian hypothesis models is defined as following:

$$D_{mahalanobis} = \sqrt{((\mu_2-\mu_1)^T S_1^{-1}(\mu_2-\mu_1) + (\mu_1-\mu_2)^T S_2^{-1}(\mu_1-\mu_2)) \cdot 0.5} \tag{4.11}$$

An example is illustrated in Figure 4.2, where the Gaussian distribution in blue has mean $\mu = -3.0$, deviation $\delta = 0.2$, and the Gaussian distribution in green has mean $\mu = -2.5$, deviation $\delta = 2.0$. Although the Euclidean distance between the mean of the two Gaussian distributions are close ($D_{euclidean} = 0.5$), the Mahalanobis distance is relatively larger ($D_{mahalanobis} = 1.77$) due to the significant difference between their deviations. As a result, Mahalanobis distance is more suitable for describing the model distance than Euclidean distance.

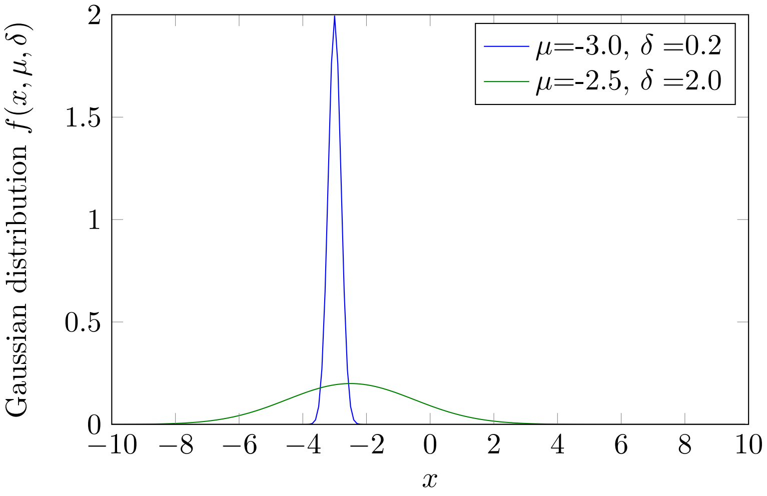


Figure 4.2: Visualization of two Gaussian distributions

To prune hypothesis models, the Mahalanobis distance is calculated between each pair of the models. If the Mahalanobis distance is under certain threshold, the two models are considered to be close to each other, and the one with lower weight will be removed. If both weights are the same, the model with higher covariance will be removed.

4.4.3. Pruning by Distance to Best Model

As we discussed in section 4.3, the model whichever is the “best” is chosen to represent the robot position. To further prune hypothesis models, the models which are very close to the “best” model are also removed. This distance is measured by Euclidean distance and the threshold is set to be a small value 2 cm. It means the error of maximum 2 cm in robot position is tolerated by the localization algorithm.

5. Analysis and Benchmark

In this thesis, two localization algorithms are implemented, namely optimization based localization and multi-hypotheses Kalman filter localization. Optimization based localization and multi-hypotheses Kalman filter localization share the same motion update model discussed in section 3.1, but different sensor update models discussed in section 3.2. As the name already indicates, optimization based localization uses the optimization based sensor model, while multi-hypotheses Kalman filter localization adopts the feature based sensor model. Moreover, multi-hypotheses Kalman filter is only enabled for the latter, because the resampling step discussed in section 4.2 depends highly on landmarks, which optimization based localization do not have during the time of its implementation. In this chapter, the two localization algorithms will be analyzed and benchmarked along with the particle filter localization algorithm from team DAInamite.

5.1. Experiments Setup

5.1.1. Ground Truth

To assess the quality of the localization algorithm, the true position of the robot needed to be obtained in order to do the comparisons and benchmarks with the position calculated by the algorithm. The true position of the robot or the so called *ground truth* can not be obtained from the robot itself, since it does not have built in GPS or other position tracking sensors. Moreover, the accuracy within centimeters is required for this purpose.

The approach adopted in this thesis is SSL-Vision [19], the vision system used in RoboCup Small Size League to obtain the position of the robots. SSL-Vision requires a camera mounted on the ceiling, and a marker with specific pattern on top of the robot. By detecting the marker and the field through the ceiling camera, the ground truth can be obtained. For NAO robot, since its head could be scanning left and right, the marker can not be directly attached on its head, otherwise the robot's orientation obtained is not correct. To counter this, a plastic support is printed using a 3D printer. As illustrated by Figure 5.1, the support is worn by the robot from the back, and the marker is attached on the top of the support.

In the SSL-Vision software [20], first set the field size, robot height, camera height and the corners of the field to calibrate the camera, so a point in the image plane can be mapped to the coordinate of the physical world frame. Then the colors in the field and the colors from the marker have to also be calibrated. Shown in Figure 5.2a is the

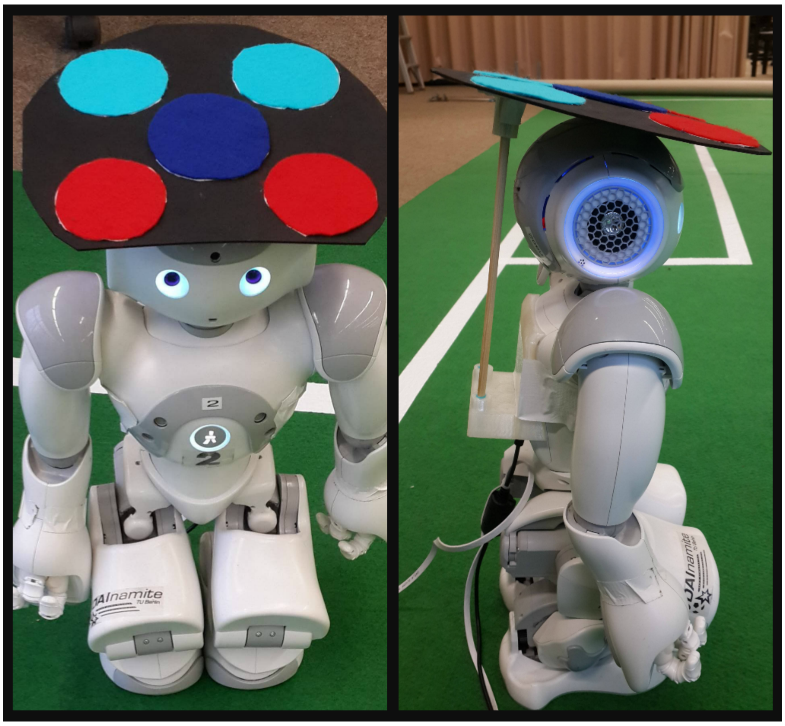
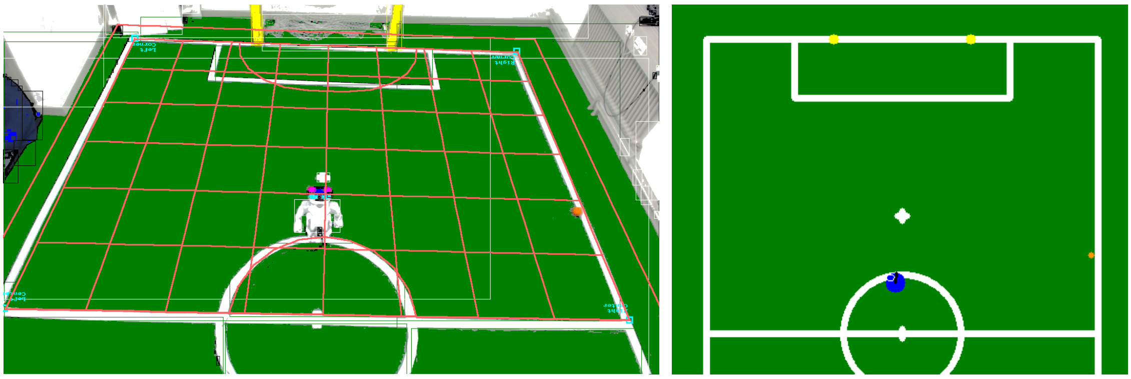


Figure 5.1: The 3D printed support and the pattern marker.

visualization result after calibration. When the marker is detected by SSL-Vision, the coordinate of the robot position in global frame will be broadcasted via network. The detected robot position is drawn in the field GUI in Figure 5.2b.



(a) Calibration result.

(b) Position in field.

Figure 5.2: SSL-Vision color and camera calibration result (a), robot position result (b).

Advantages of SSL-Vision

- Low-cost hardware for setting up SSL-Vision.
- Open-source project with manual and support from community.
- Able to achieve a position accuracy of better than 15 mm in RoboCup Small Size League [21].

Disadvantages of SSL-Vision

- The system highly depends on the light of the environment, once the surrounding light changes, the color metrics need to be recalibrated.
- If the marker is printed using normal paper, it may cause reflection at certain angles from the view point of the camera, then the pattern can not be detected. For this reason, fuzzy materials are specially chosen to manually create the marker.
- The system can not detect the pattern when the robot has fallen down.

5.1.2. Perception Log

For the purpose of debugging and benchmarks, the robot can store the perception as logs for future replay. The log contains the necessary data needed to re-run the localization algorithm on another computer. The log includes vision results, odometry, IMU, sonar data, time stamp, robot posture, etc. In order to benchmark the quality of the localization algorithm result, the ground truth data has to be stored as well, and when the log is replayed, the ground truth data should be synchronized with the perception log. The solution for this is to store the broadcasted ground truth from SSL-Vision at the same time of recording the perception, and consequently the ground truth becomes part of the perception log. The replayed perception log with ground truth is already illustrated in Figure 3.9, in which the ground truth position is indicated by red, calculated position in blue, mirrored calculated position in green.

5.2. Code Optimization

As illustrated in subsection 1.2.1, the motherboard that the robot equips is an embedded platform with only limited computational power. This requires the localization algorithms developed to be computationally efficient in order to run in real time in SPL game. In this section, the methods used to enhance the performance of the two localization algorithms, namely optimization based localization and multi-hypotheses Kalman filter localization, will be discussed.

By the intrinsics of the Rprop algorithm, optimization based localization imposes significant overhead in computation. On the one hand, it uses all the detected line points instead of line segments to calculate the measurement error, on the other hand, the optimization procedure takes several iterations to converge to the optimal result. Although the error function being used by Rprop has been implemented by an error look up table and an error gradient look up table, which saves the on-line computation time by pre-calculating the error and error gradient off-line, the computation time is still high. Initially the implementation is using the combination of python and Numpy. After optimizing the computation heavy part using Cython and reducing the number of optimization iterations, the computation speed has increased by 84%. However, with this speed up after optimization, it is still almost 3 times slower than the implementation of particle filter as illustrated in Table 5.3, which is far from acceptable to run on the robot.

The major drawback pertains to the optimization based localization is that when the size of look up table is big, the requests to the look up table can often cause CPU cache misses if the data requested in memory is not contiguous. Since the line points in observation are often scattered over the field, and the look up table is constructed row by row along the field, a request to the look up table by these line points will often result in the penalty of cache misses. Moreover, two look up tables have to be requested repeatedly for several iterations during the optimization step, which magnifies the overhead.

Taking the experience during implementing optimization based localization, the principle is to implement the computational heavy part using lower level programming language to gain performance and the other parts in higher level language to have design flexibility. For multi-hypotheses Kalman filter, the code pertaining to motion update and sensor update are implemented in C++. The other parts of code regarding observation pre-processing and feature detection are implemented using the combination of python, Numpy and Cython. The interface between C++ code and python code is bridged by Cython. The performance of this design turns out to be satisfactory. As illustrated in Table 5.3, it is 22.61% faster than the speed of particle filter.

5.3. Benchmarks

To benchmark the quality of localization algorithms, we use the same perception log with ground truth throughout this section as the input for the different algorithms. The log consists of 3226 perception frames and is recorded when the robot is initially placed at the side of the field and walks into the goalie position. In the end, the robot is kidnapped to a position near the center circle[1]. The quality of the localization algorithm is judged by the following criteria:

1. Initial global localization.
2. Accuracy of robot position tracking.
3. Recovery from tracking failure or kidnap.
4. Efficiency of algorithm.

Three localization algorithms are compared, namely, particle filter, optimization based localization, multi-hypotheses Kalman filter localization.

---

[1] The kidnapping happens at around frame 2900 of the log, where a robot position “jump” is illustrated by the ground truth trajectory in Figure 5.3

5.3.1. Accuracy

As illustrated in Figure 5.3, 5.4 and 5.5, firstly, the resulting robot position trajectory for each algorithm is compared with the ground truth trajectory. In addition to trajectories, which only give a general impression of the quality of the corresponding localization method, the error is quantitatively measured for each dimension of the position x, y, θ respectively. Figure 5.6, 5.7 and 5.8 show the particle filter localization result at each time frame, as well as its corresponding error with the ground truth. Similarly for optimization based localization and multi-hypotheses Kalman filter localization, the results and errors are shown in Figure 5.9, 5.10, 5.11 and Figure 5.12, 5.13, 5.14 respectively. The average error and standard deviation for each localization algorithm result are shown in Table 5.1 and 5.2. Average error and standard deviation are calculated for each dimension of the robot position x, y, θ.

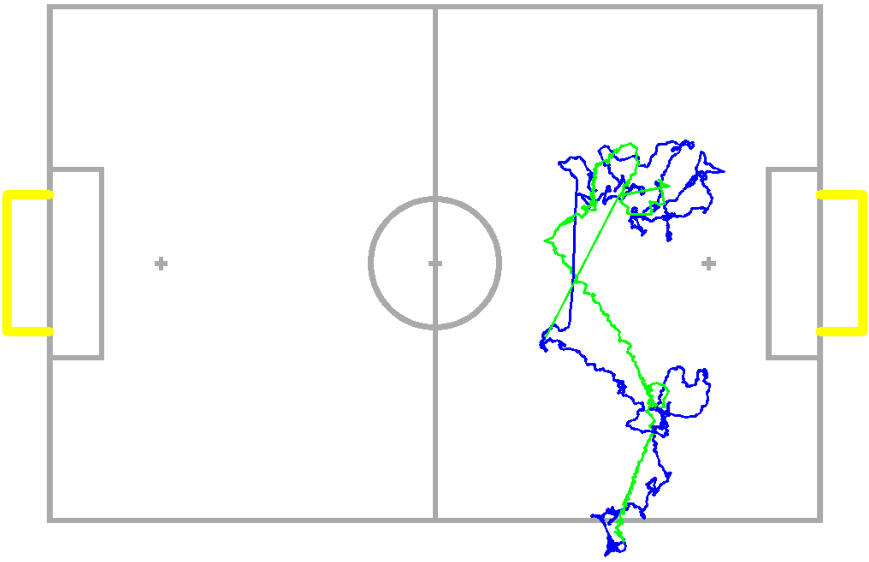


Figure 5.3: Particle filter localization trajectory (blue) compared with ground truth trajectory (green).

From Table 5.1 and 5.2, we can see that multi-hypotheses Kalman filter out-performs the other localization algorithms in term of accuracy, but particle filter has less standard deviation in x and y dimension compared to multi-hypotheses Kalman filter. Optimization based localization tracks the robot position with high accuracy at the beginning, but diverges into a wrong localization when the robot get kidnapped, which results in its large average error in the end. By comparing Figure 5.3 and 5.5, another benefit of Kalman filter based localization is that the trajectory of the position is more smooth than the one from particle filter.

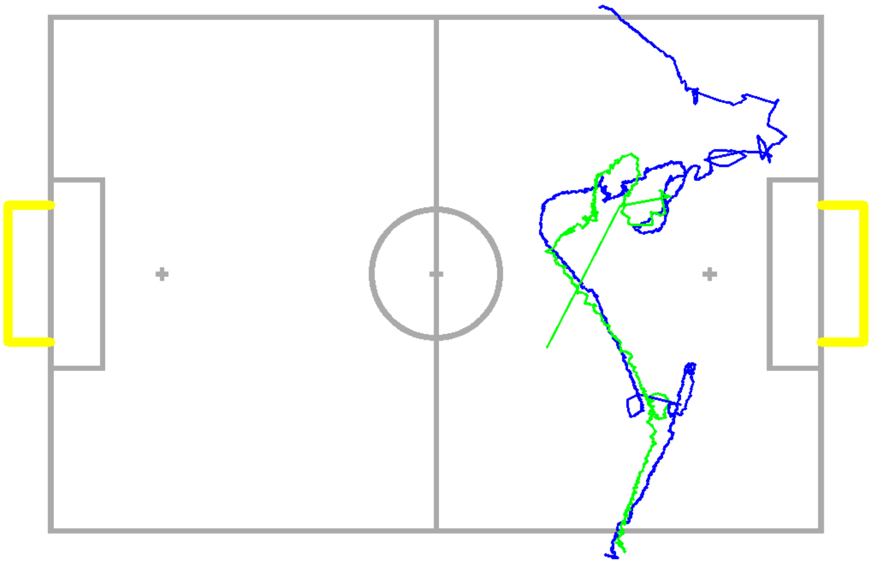


Figure 5.4: Optimization based localization trajectory (blue) compared with ground truth trajectory (green).

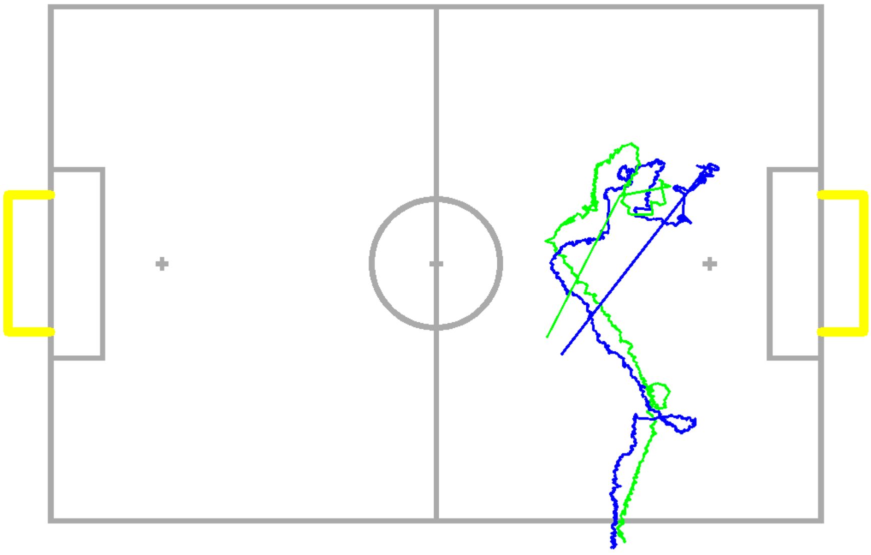
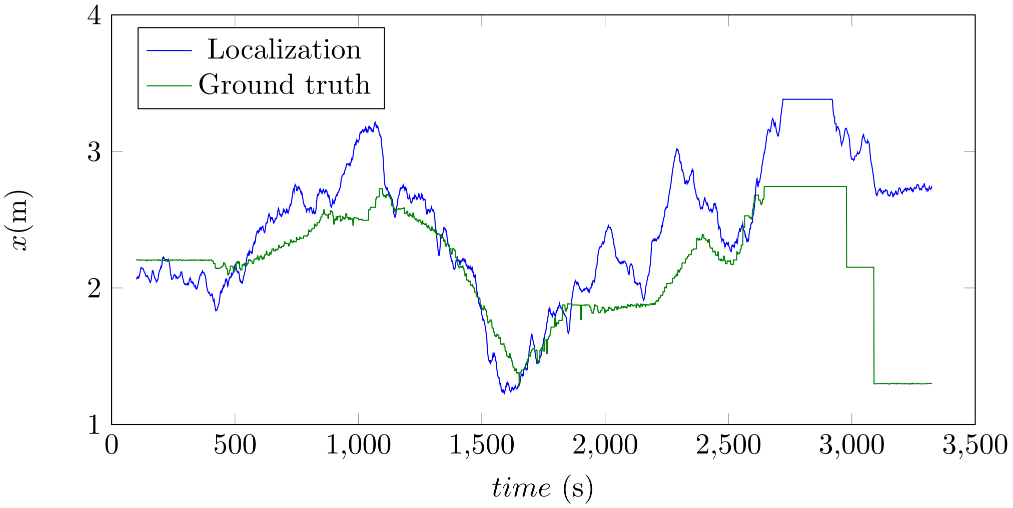


Figure 5.5: Multi-hypotheses Kalman filter localization trajectory (blue) compared with ground truth trajectory (green).


localization result in x dimension compared with ground truth.

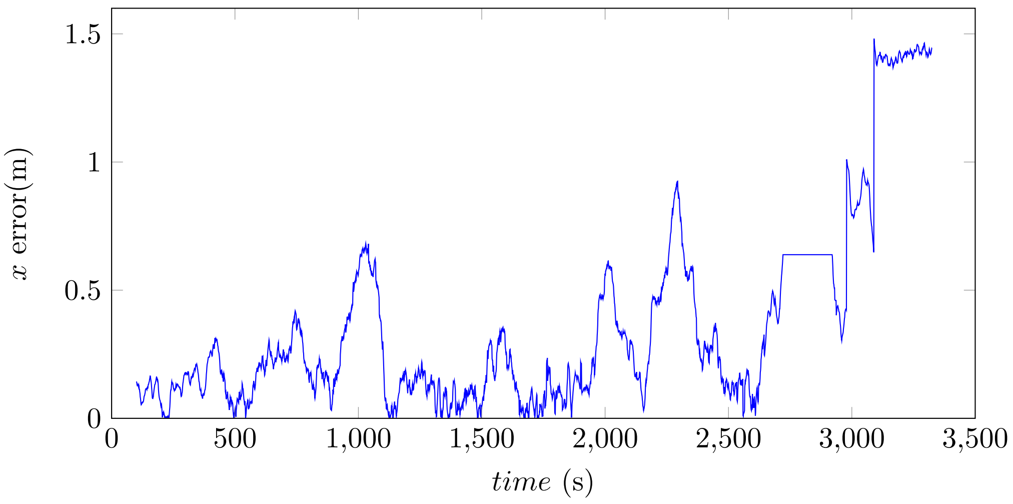
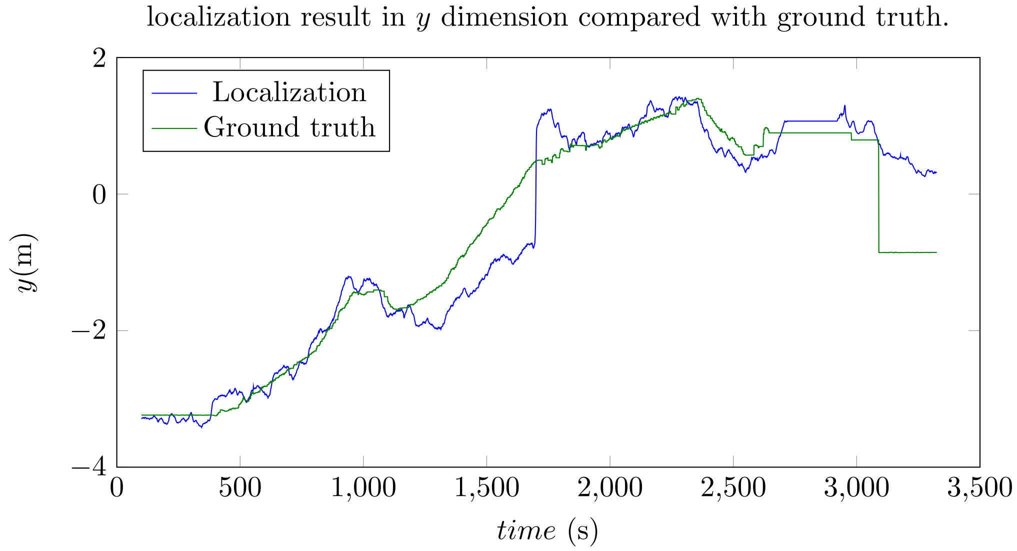

Error in localization x dimension.

Figure 5.6: Particle filter localization result and error in θ dimension.



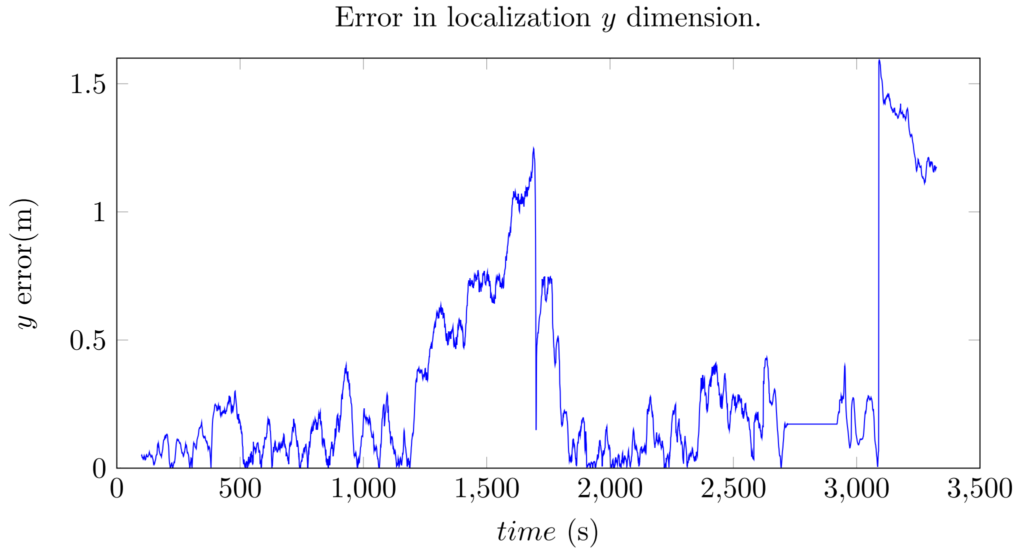
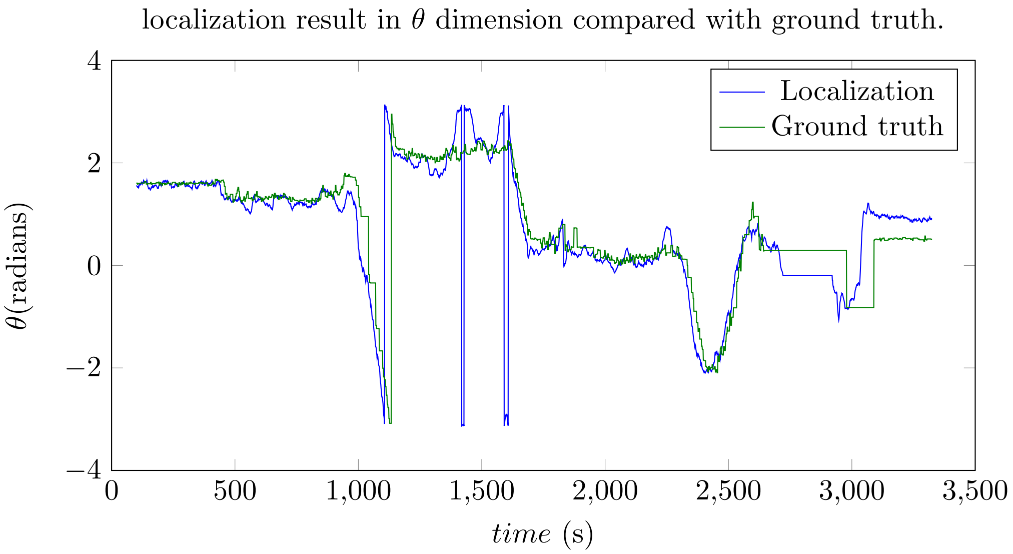


Figure 5.7: Particle filter localization result and error in θ dimension.



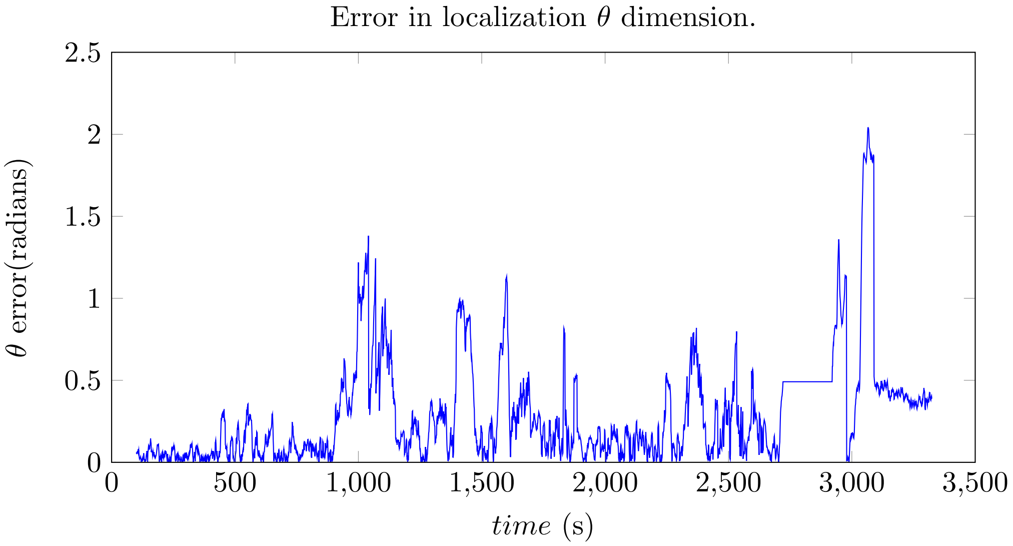


Figure 5.8: Particle filter localization result and error in θ dimension.

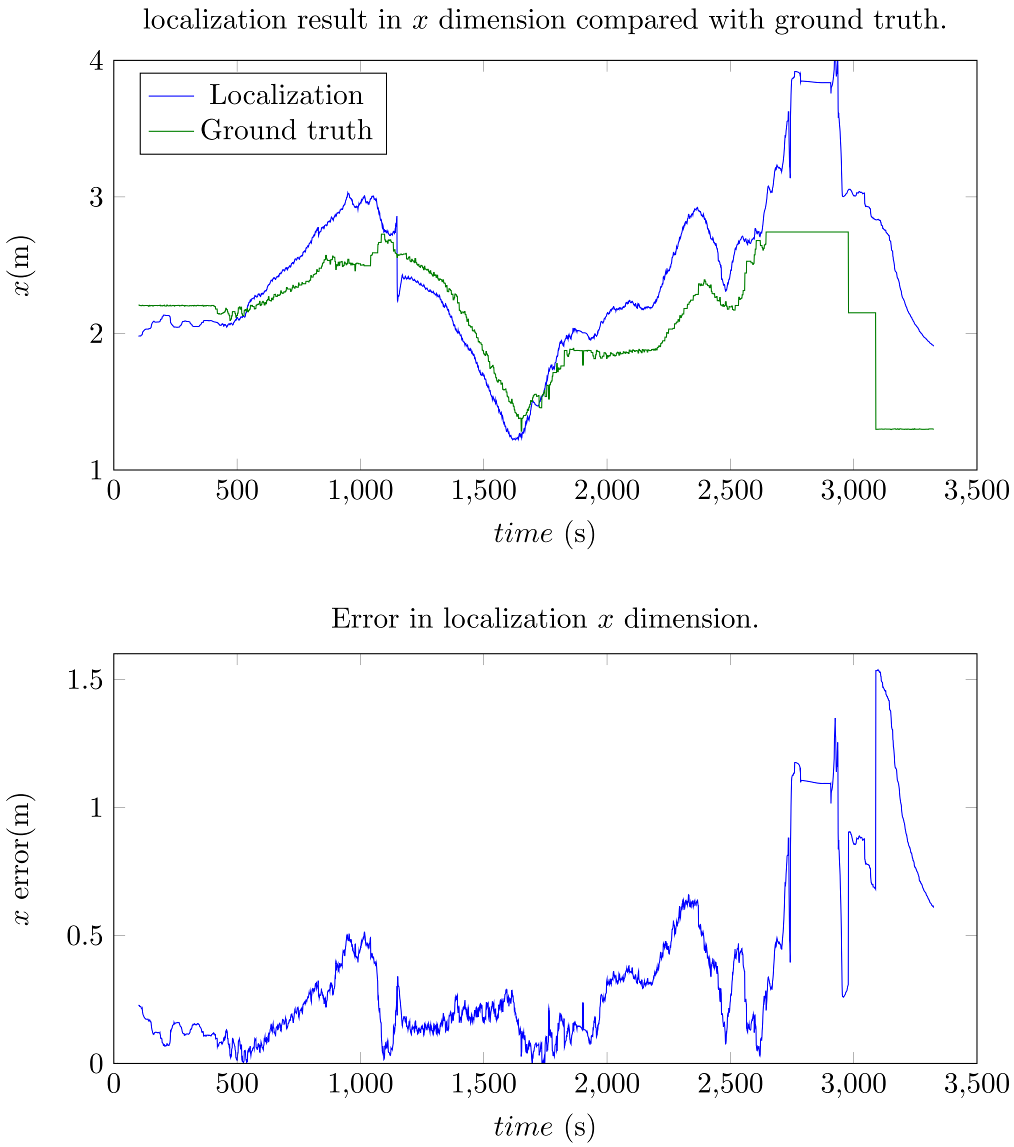
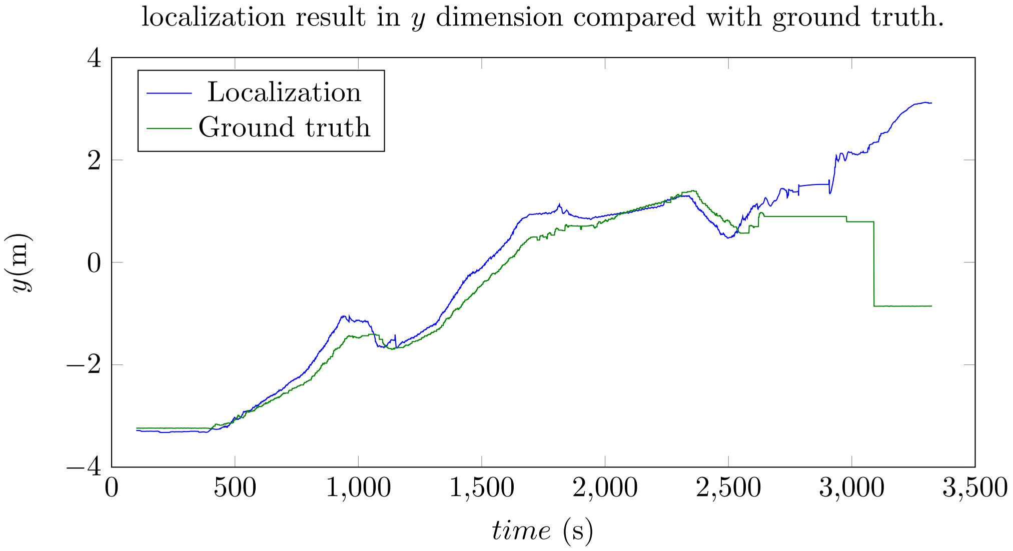


Figure 5.9: Optimization based localization result and error in x dimension.



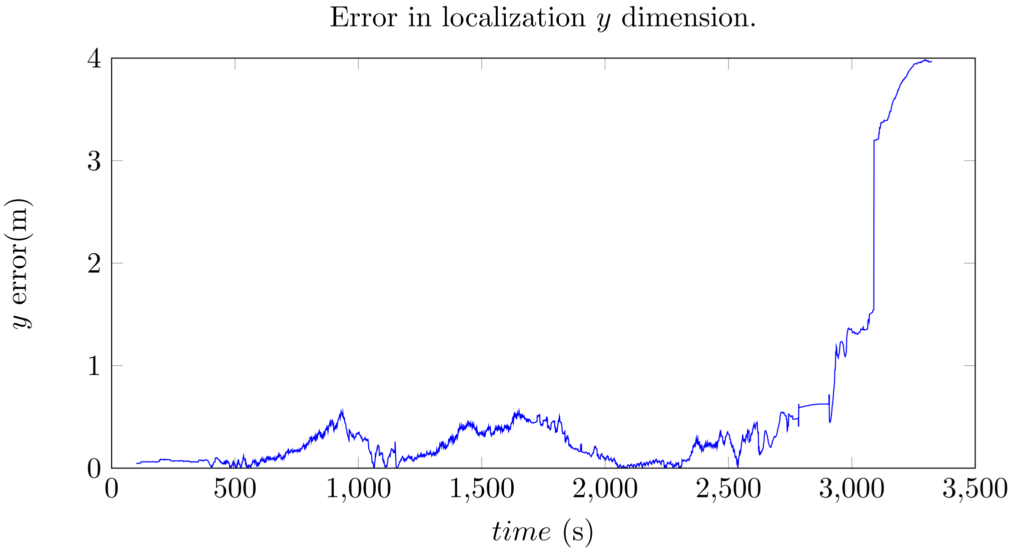


Figure 5.10: Optimization based localization result and error in y dimension.

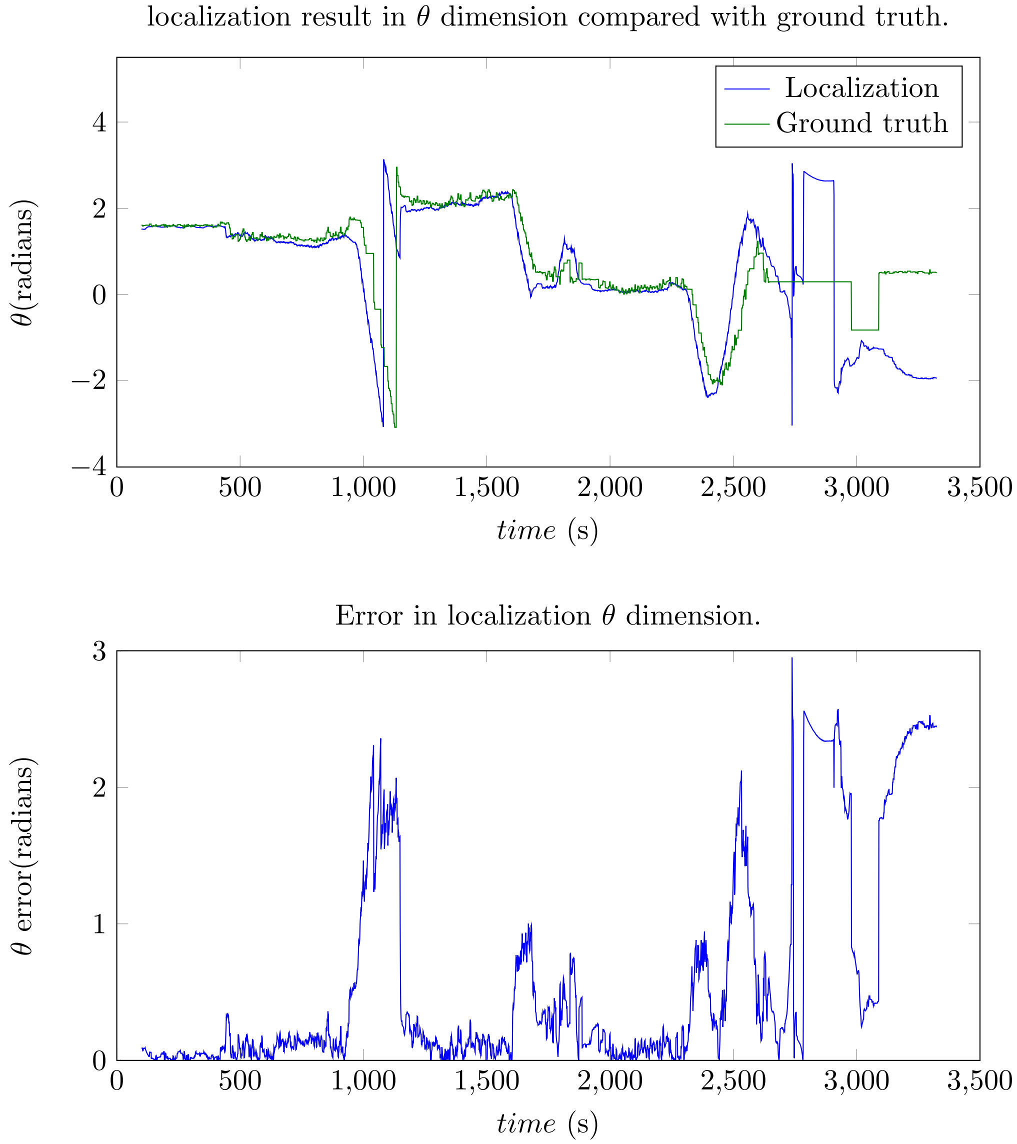
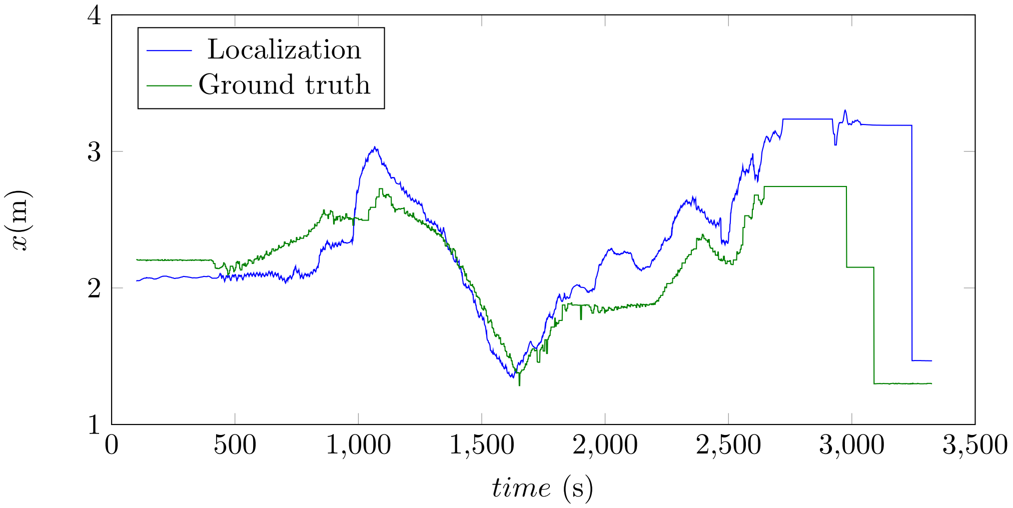


Figure 5.11: Optimization based localization result and error in θ dimension.

localization result in x dimension compared with ground truth.



Error in localization x dimension.

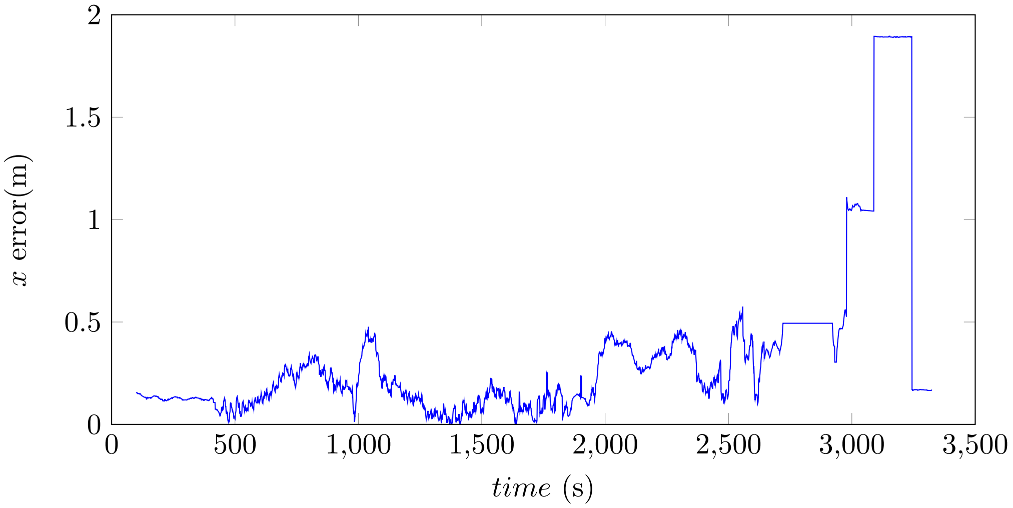
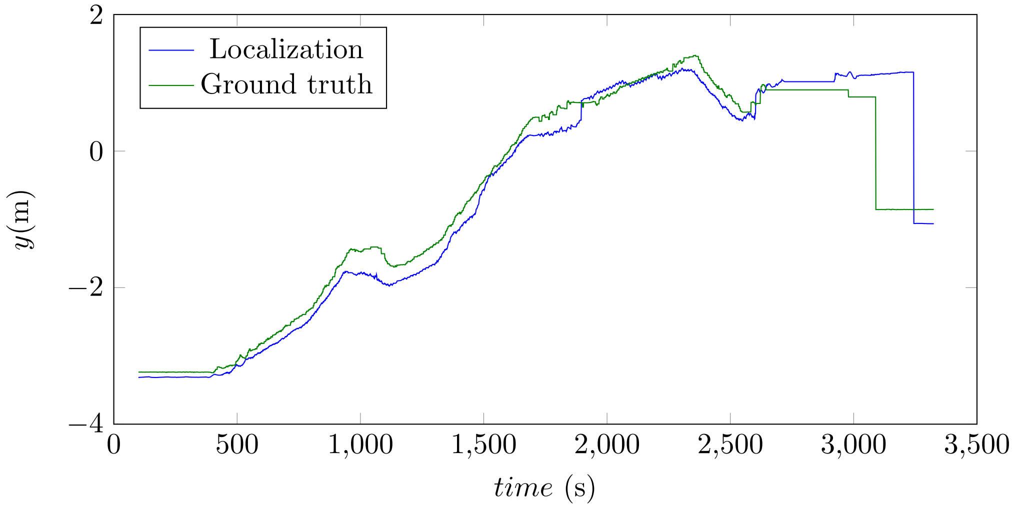


Figure 5.12: Multi-hypotheses Kalman filter localization result and error in x dimension.

localization result in y dimension compared with ground truth.



Error in localization y dimension.

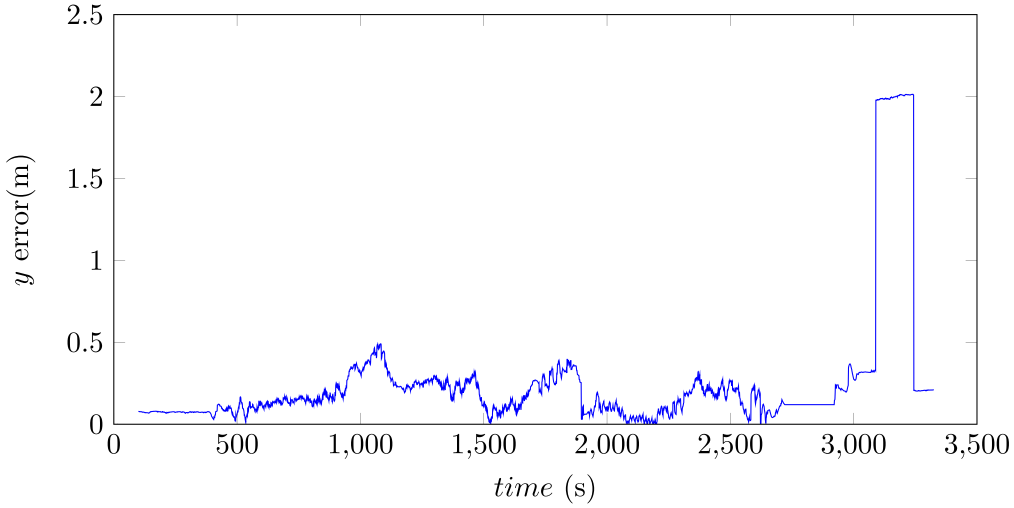
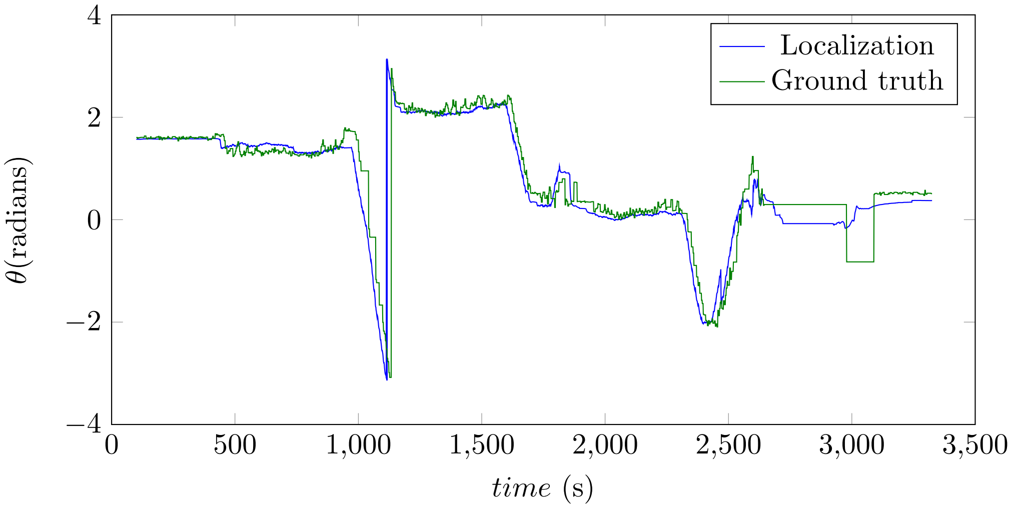
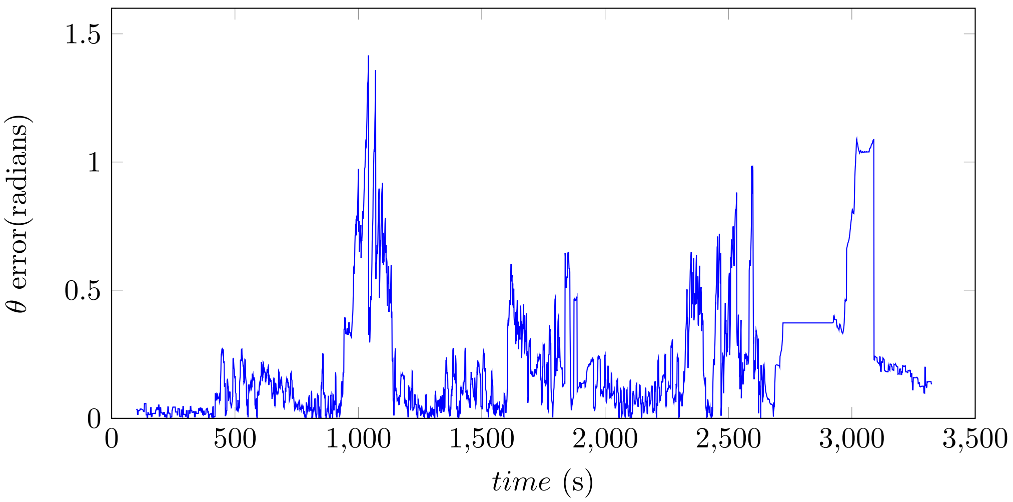


Figure 5.13: Multi-hypotheses Kalman filter localization result and error in y dimension.



localization result in θ dimension compared with ground truth.



Error in localization θ dimension.

Figure 5.14: Multi-hypotheses Kalman filter localization result and error in θ dimension.

Average error	x (m)	y (m)	θ (rad)
Particle filter	0.363	0.323	0.296
Optimization based	0.366	0.534	0.604
Multi-hypotheses Kalman filter	0.331	0.255	0.225

Table 5.1: Accuracy comparison between different localization algorithm results

Standard deviation	x (m)	y (m)	θ (rad)
Particle filter	0.371	0.372	0.326
Optimization based	0.338	0.936	0.796
Multi-hypotheses Kalman filter	0.404	0.402	0.243

Table 5.2: Standard deviation comparison between different localization algorithm results

5.3.2. Functionality

From the illustrations of the localization trajectory in Figure 5.3, 5.4 and 5.5, all the three localization algorithms solve the initial global localization problem and tracks the robot position after global localization.

Concerning the ability to recover from kidnapped situation, from the three approaches, only multi-hypotheses Kalman filter localization recovered the robot position at the end of the log. That is because the robot observed an “X” junction near the center circle, and resampling is triggered to recover the robot position. However, “X”, “T” junction detection and resampling is only used in multi-hypotheses Kalman filter localization algorithm, not in particle filter and optimization based localization, therefore the other algorithm can not recover in this case. Particle filter localization is also capable to do resampling when “L” junction, penalty area or center circle are seen.

5.3.3. Efficiency

Last but not the least, the average execution time per perception frame of the log is measured for each algorithm to compare the efficiency. For the purpose of benchmark, the NAO robot is configured only to run the localization algorithm. In this thesis, the execution time of particle filter can be regarded as a base line for other algorithms, as we know that particle filter algorithm could be run seamlessly on the robot together with other necessary SPL modules. Table 5.3 depicts the average execution time per perception frame for each localization algorithm[2]. Within Table 5.3, multi-hypotheses Kalman filter localization out-performs the other localization algorithms in computation speed, optimization based localization being the slowest.

---

[2] For detailed profiles of algorithm execution time, refer to Appendix A.1

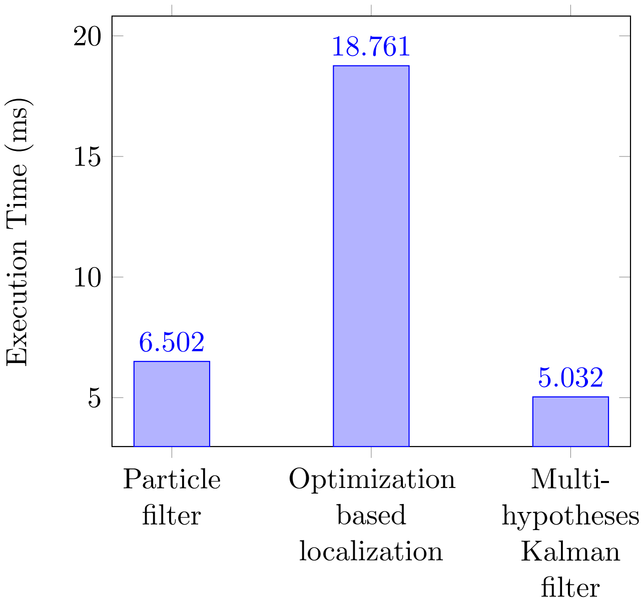


Table 5.3: Average execution time per frame of the perception log for different localization algorithms running on the NAO robot.

6. Conclusion and Future Work

6.1. Conclusion

In this thesis, two localization methods are presented, one is optimization based localization and the other is feature based multi-hypotheses Kalman filter localization. While the most significant problem with optimization based localization is the high computational requirement which prohibits it from running on the robot, this thesis focuses more on feature based multi-hypotheses Kalman filter localization due to the aforementioned reasons.

For feature based multi-hypotheses Kalman filter localization, special characteristics of SPL game have been considered for designing its motion model and sensor model. To overcome the ambiguity of landmarks, different strategies are adopted to find the correspondence between the observed landmark and the landmark on the field. By incorporating landmark based resampling, multi-modal probability distribution can be described, and the localization of the robot becomes more robust and is able to recover from tracking failure and kidnapped situations. In the end, both localizations are compared with DAInamite's particle filter localization. The feature based multi-hypotheses Kalman filter localization out-performs the other localization algorithms in terms of accuracy, efficiency and functionality.

The multi-hypotheses Kalman filter localization is tested in the real game during Night of Science Frankfurt 2015 at Goethe University, and team DAInamite won one game out of three. During the game, the situation was more unpredictable and complex than the test environment, sometimes the robot could localize itself well, and sometimes not. Although the winning of a game depends on many factors, the localization has a significant influence on it. Therefore, the performance of the localization still have large space of improvement.

6.2. Future Work

Future work will focus on improving the robustness of the localization. This can be accomplished by incorporating more landmark features like goal posts or lines which are perpendicular to each other, the lines do not have to be “L”, “T”, “X” junctions, but have the potential to form one. On the other hand, the confidence of the robot position currently depends only on the unique landmarks. Methods which incorporate ambiguous landmarks like line landmarks or combine the states of multiple hypotheses can be investigated.

Moreover, resampling step could be smarter. Currently the resampling is done by a single landmark which results in multiple possible robot positions. The resampling step can be improved by combining all the types of junctions observed to generate robot positions, which can significantly reduce the number of possible positions.

Collaborative localization using ball information can also help enhance the result of localization. Since every robot which observes the ball has a belief of the ball's location. By gathering the information of the ball, it can serve as a globally unique landmark which can help localization.

Bibliography

[1] *Nao body version and type.* Available at `http://doc.aldebaran.com/2-1/family/body_type.html#nao-version-bodytype`.

[2] R. T. Committee, "RoboCup Standard Platform League (NAO) Rule Book," pp. 1–29, 2013.

[3] F. Dellaert, D. Fox, W. Burgard, and S. Thrun, "Monte carlo localization for mobile robots," in *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, vol. 2, pp. 1322–1328, IEEE, 1999.

[4] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of Fluids Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[5] S. J. Julier and J. K. Uhlmann, "New extension of the kalman filter to nonlinear systems," in *AeroSense'97*, pp. 182–193, International Society for Optics and Photonics, 1997.

[6] R. Van Der Merwe and E. A. Wan, "The square-root unscented kalman filter for state and parameter-estimation," in *Acoustics, Speech, and Signal Processing, 2001. Proceedings.(ICASSP'01). 2001 IEEE International Conference on*, vol. 6, pp. 3461–3464, IEEE, 2001.

[7] E. a. Wan and R. Van Der Merwe, "The unscented Kalman filter for nonlinear estimation," *Technology*, vol. v, pp. 153–158, 2000.

[8] D. L. Alspach and H. W. Sorenson, "Nonlinear bayesian estimation using gaussian sum approximations," *Automatic Control, IEEE Transactions on*, vol. 17, no. 4, pp. 439–448, 1972.

[9] M. J. Quinlan and R. H. Middleton, "Multiple model Kalman filters: A localization technique for RoboCup soccer," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 5949 LNAI, pp. 276–287, 2010.

[10] G. Jochmann, S. Kerner, S. Tasse, and O. Urbann, "Efficient multi-hypotheses unscented kalman filtering for robust localization," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 7416 LNCS, pp. 222–233, 2012.

[11] D. Gohring, H. Mellmann, and H.-D. Burkhard, “Constraint based world modeling in mobile robotics,” *2009 IEEE International Conference on Robotics and Automation*, 2009.

[12] M. Lauer, S. Lange, M. Riedmiller, and M. R. Martin Lauer, Sascha Lange, “Calculating the perfect match: an efficient and accurate approach for robot self-localization,” *Robocup 2005: Robot soccer world cup . . .* , vol. 4020, no. c, pp. 142–153, 2006.

[13] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. MIT press, 2005.

[14] M. Riedmiller and H. Braun, “A direct adaptive method for faster backpropagation learning: the RPROP algorithm,” *IEEE International Conference on Neural Networks*, 1993.

[15] S. Tasse, M. Hofmann, and O. Urbann, “On Sensor Model Design Choices for Humanoid Robot Localization,” *RoboCup 2012: Robot Soccer World Cup XVI*, pp. 380–390, 2013.

[16] *The official 2014 B-Human code release.* Available at `https://github.com/bhuman/BHumanCodeRelease.git`.

[17] J. L. Bentley, “Multidimensional binary search trees used for associative searching,” *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975.

[18] R. De Maesschalck, D. Jouan-Rimbaud, and D. L. Massart, “The mahalanobis distance,” *Chemometrics and intelligent laboratory systems*, vol. 50, no. 1, pp. 1–18, 2000.

[19] S. Zickler, T. Laue, O. Birbach, M. Wongphati, and M. Veloso, “Ssl-vision: The shared vision system for the robocup small size league,” in *RoboCup 2009: Robot Soccer World Cup XIII*, pp. 425–436, Springer, 2010.

[20] *SSL-Vision modified for Standed Platfrom league.* Available at `https://github.com/xuyuan/ssl-vision`.

[21] D. Ball, G. Wyeth, and S. Nuske, “A global vision system for a robot soccer team,” in *2004 Australasian Conference on Robotics and Automation*, Australian Robotics and Automation Association Inc, 2004.

A. Appendix

A.1. Speed Profiles of Localization Algorithms

The following program profiles are the total time (seconds) for executing the perception log file refered in section 5.3 with repect to each localization algorithm. For the purpose of benchmark, the execution of the log is performed on a NAO V4 robot without other CPU intensive modules running. The total execution time is indicated at the top left corner of the profile. Each box inside the profile is the function which is invoked by the localization algorithm, and the size of the box corresponds to the occupation of execution time.

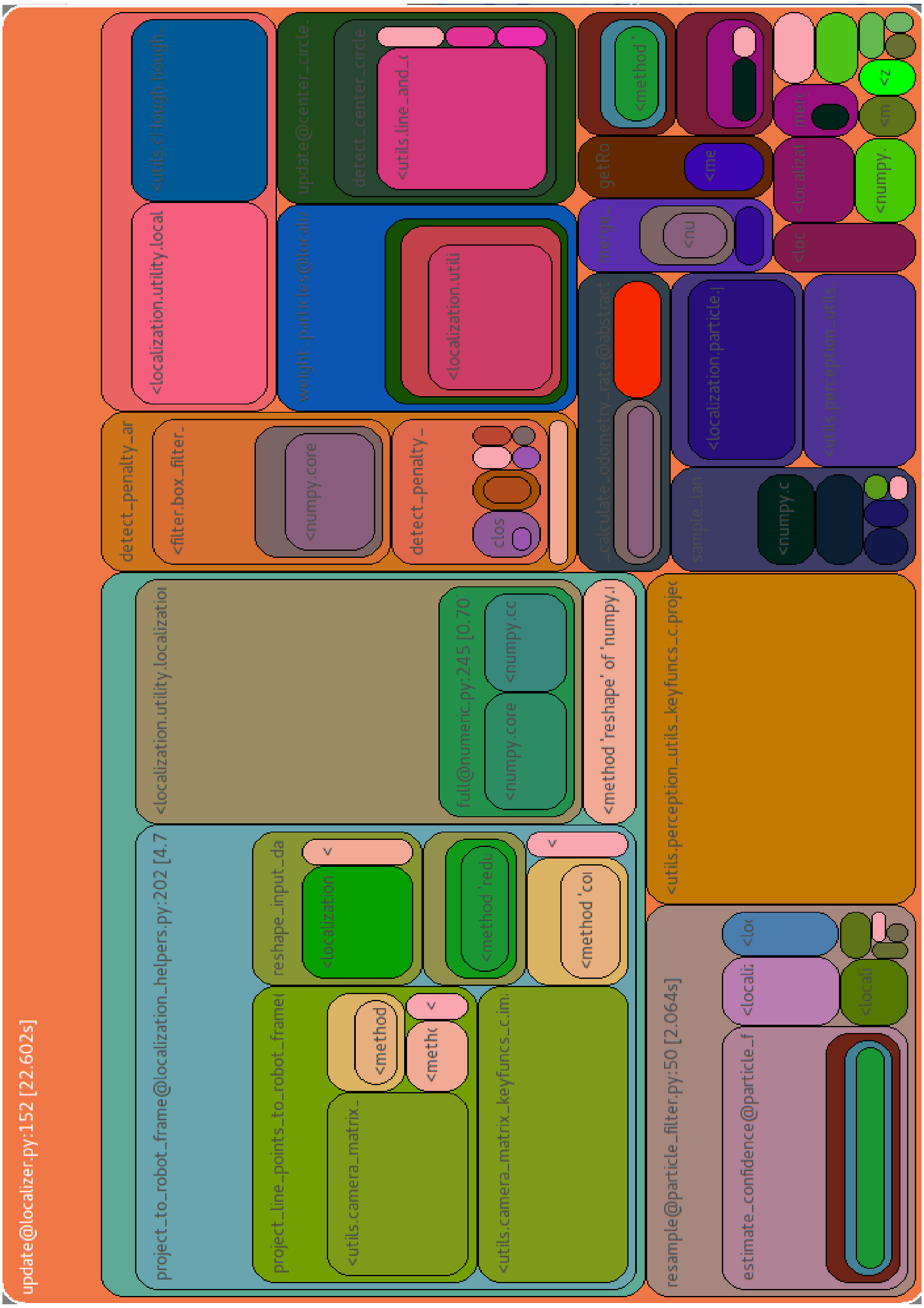


Figure A.1: Program profile of particle filter localization for executing the perception log on the NAO robot.

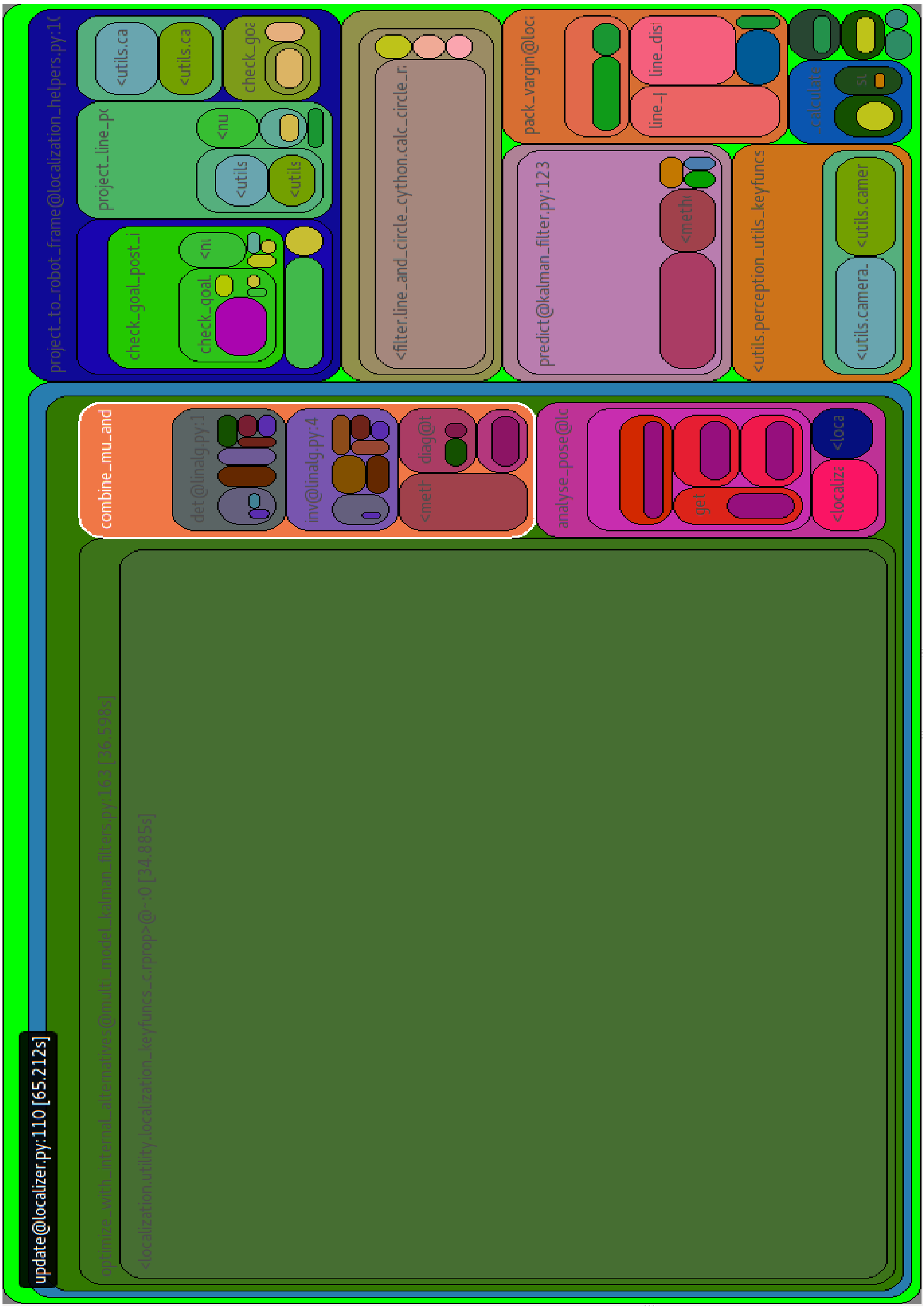


Figure A.2: Program profile of optimization based localization for executing the perception log on the NAO robot.

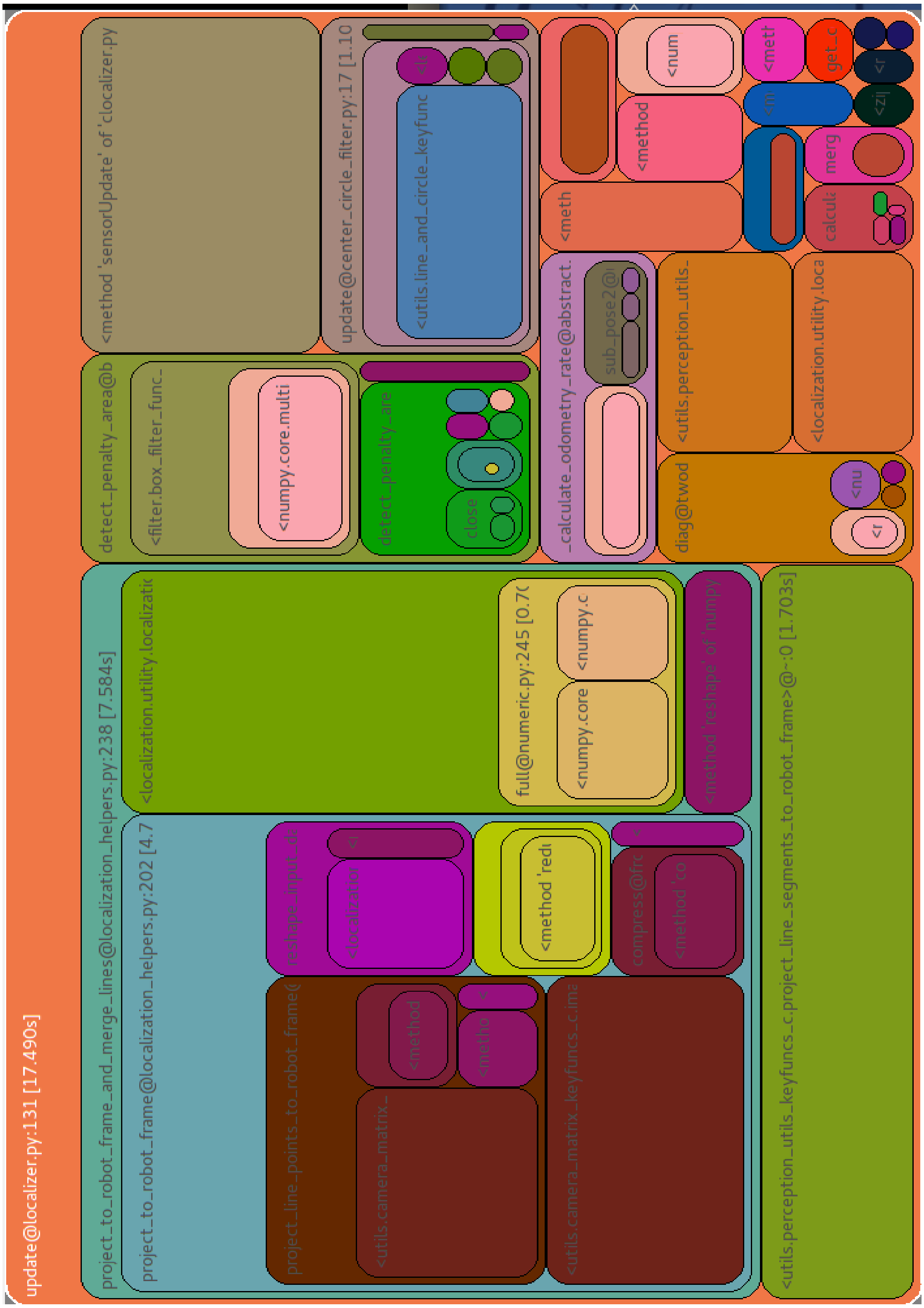


Figure A.3: Program profile of multi-hypotheses Kalman filter localization for executing the perception log on the NAO robot.