ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL OF SCIENCE ENGINEERING AND TECHNOLOGY

COMBINATION OF MCL AND ICP METHODS FOR ACCURATE INDOOR LOCALIZATION

M.Sc. THESIS

Erim VURAL

Department of Control and Automation Engineering

Control and Automation Engineering Programme

DECEMBER 2016

ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL OF SCIENCE ENGINEERING AND TECHNOLOGY

COMBINATION OF MCL AND ICP METHODS FOR ACCURATE INDOOR LOCALIZATION

M.Sc. THESIS

Erim VURAL
(504151110)

Department of Control and Automation Engineering

Control and Automation Engineering Programme

Thesis Advisor: Prof. Dr. Hakan TEMELTAŞ

DECEMBER 2016

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

KESİN İÇ ORTAM LOKALİZASYONU İÇİN MCL VE ICP YÖNTEMLERİNİN BİRLEŞTİRİLMESİ

YÜKSEK LİSANS TEZİ

Erim VURAL
(504151110)

Kontrol ve Otomasyon Mühendisliği Anabilim Dalı

Kontrol ve Otomasyon Mühendisliği Programı

Tez Danışmanı: Prof. Dr. Hakan TEMELTAŞ

ARALIK 2016

Erim VURAL, a M.Sc.student of ITU Graduate School of Science Engineering and Technology student ID 504151110, successfully defended the thesis entitled "COMBINATION OF MCL AND ICP METHODS FOR ACCURATE INDOOR LOCALIZATION", which he prepared after fulfilling the requirements specified in the associated legislations, before the jury whose signatures are below.

Thesis Advisor :	**Prof. Dr. Hakan TEMELTAŞ** Istanbul Technical University	.....
Jury Members :	**Asst. Prof. Dr. Volkan SEZER** Istanbul Technical University	.....
	Asst. Prof. Dr. Janset DAŞDEMİR Yıldız Technical University	.....

Date of Submission : 1 December 2016
Date of Defense : 22 December 2016

To my family,

FOREWORD

Firstly, I would like to thank to my supervisor Prof. Dr. Hakan TEMELTAŞ, who had supported me with his knowledge and experience during this thesis study. Also, I would like to thank to my friends Derya KELOĞLU, Alper Nabi AKPOLAT, Tuğçe ESENDURAN and Abdullah Ömer SEVİL for their supports during my M.Sc. study.

Finally, I would like to thank to my family, who support me every time and in all conditions.

December 2016 Erim VURAL

TABLE OF CONTENTS

Page

FOREWORD ..... **ix**
TABLE OF CONTENTS ..... **xi**
ABBREVIATIONS ..... **xiii**
SYMBOLS ..... **xv**
LIST OF TABLES ..... **xvii**
LIST OF FIGURES ..... **xix**
SUMMARY ..... **xxi**
ÖZET ..... **xxiii**
1. INTRODUCTION **1**
1.1 Problem Statement 2
1.2 Hypothesis 2
2. ROBOTICS MIDDLEWARE AND SOFTWARE TOOLS **5**
2.1 Robot Operating System 5
2.2 MATLAB Robotics System Toolbox 8
3. PATH PLANNING **9**
3.1 Creating the Map of an Environment 9
3.2 Probabilistic Roadmap Path Planner 10
4. TWO-STEP INDOOR LOCALIZATION **13**
4.1 Monte Carlo Localization 13
4.2 Localization with ICP 14
4.2.1 The Iterative Closest Point Algorithm 18
4.2.2 The Trimmed Iterative Closest Point Algorithm 21
4.2.3 The Iterative Closest Point Algorithm with Point-to-Line Metric 24
5. SIMULATION STUDIES **27**
5.1 Pure Pursuit Path Following Controller 27
5.2 Simulation Environment 28
5.3 Case Study 30
5.4 Simulation Results 32
6. CONCLUSION **37**
REFERENCES **39**
APPENDICES **41**
CURRICULUM VITAE **53**

ABBREVIATIONS

AGV	**:** Automated Guided Vehicle
ICP	**:** Iterative Closest Point
KLD	**:** Kullback Liebler Distance
MCL	**:** Monte Carlo Localization
MSE	**:** Mean Square Error
PLICP	**:** Iterative Closest Point with Point-to-Line Metric
PRM	**:** Probabilistic Roadmap
ROS	**:** Robot Operating System
SVD	**:** Singular Value Decomposition
TrICP	**:** Trimmed Iterative Closest Point
2D	**:** Two Dimensional
3D	**:** Three Dimensional

SYMBOLS

L	**:** Length
W	**:** Width
H	**:** Height
$\boldsymbol{F}$	**:** Data set
$\boldsymbol{G}$	**:** Model set
$\boldsymbol{N_F}$	**:** Number of points in *F*
$\boldsymbol{N_G}$	**:** Number of points in *G*
$\boldsymbol{i}$	**:** Index of a point set
$\boldsymbol{j}$	**:** Index of a point set after elimination of outliers
$\boldsymbol{f_i}$	**:** *i* th point in data set
$\boldsymbol{g_i}$	**:** *i* th point in model set
$\boldsymbol{h_i}$	**:** Closest point model set to *i* th point of data set
$\boldsymbol{z_i}$	**:** Second closest point in model set to *i* th point of data set
Corr	**:** Correspondence Operator
$\boldsymbol{H}$	**:** The set of closest points
$\boldsymbol{Z}$	**:** The set of second closest points
$\boldsymbol{R}$	**:** Rotation matrix
$\boldsymbol{t}$	**:** Translation vector
$\boldsymbol{\bar{f}}$	**:** Centroid of *F*
$\boldsymbol{\bar{h}}$	**:** Centroid of *H*
$\boldsymbol{cf_i}$	**:** *i* th point in centered set of *F*
$\boldsymbol{ch_i}$	**:** *i* th point in centered set of *H*
$\boldsymbol{k}$	**:** Iteration number
$\boldsymbol{E}$	**:** Mean square errror before the transformation
$\boldsymbol{E'}$	**:** Mean square errror after the transformation
$\boldsymbol{m_{or}}$	**:** Minimum overlap rate
$\boldsymbol{n_i}$	**:** normal to the line that lies between h_i and z_i

LIST OF TABLES

Page

Table A.1 : Size and weight **43**
Table A.2 : Speed and performance. **43**
Table A.3 : Battery and power system **43**

LIST OF FIGURES

Page

Figure 2.1 : ROS communication structure **6**
Figure 2.2 : Example of a message type **6**
Figure 2.3 : Communication by using topics. **7**
Figure 2.4 : Communication by using services. **7**
Figure 2.5 : Diagram of the connection that is provided by Robotics System Toolbox. **8**
Figure 3.1 : Occupancy grid map of the environment. **10**
Figure 3.2 : Inflated occupancy grid map of the environment. **11**
Figure 3.3 : Probabilistic Roadmap between two locations. **11**
Figure 4.1 : Probabilistic Roadmap of the robot. **15**
Figure 4.2 : Real positions of the robot …. **16**
Figure 4.3 : Real positions of the robot around the goal location. **16**
Figure 4.4 : The difference between the goal location and the real location where the robot stopped…….. **17**
Figure 4.5 : The difference between the goal location and the MCL estimated location where the robot stopped……. **17**
Figure 4.6 : Example for partially overlapping point sets…. **21**
Figure 5.1 : The front view of environment that is used for simulations **28**
Figure 5.2 : The side view of environment that is used for simulations **28**
Figure 5.3 : Occupancy grid map of the environment **29**
Figure 5.4 : The TurtleBot **29**
Figure 5.5 : Target locations **30**
Figure 5.6 : Laser and image data when the robot's angle is 45.3152° **31**
Figure 5.7 : Laser and image data when the robot's angle is 90° **32**
Figure 5.8 : The robot's instantaneous positions during the experiment **33**
Figure 5.9 : The x and y axes position estimation errors **33**
Figure 5.10 : The angular estimation errors **34**
Figure 5.11 : The points where the robot stopped at Target Location 1 ..... **35**
Figure 5.12 : The points where the robot stopped at Target Location 2 ..... **35**

COMBINATION OF MCL AND ICP METHODS FOR ACCURATE INDOOR LOCALIZATION

SUMMARY

With the developing technology, mobile robots are being used more frequently. The use of mobile robots is needed, especially since the aim is to make the production of the industry more efficient.

Nowadays, mobile robots that are widely used in the industry are moving on pre-planned paths. These paths are prepared using wires, magnetic tapes or any other methods that allow the robot to follow the path. However, using these paths restricts the movement of the robot. This is because when it is necessary to change the target locations of the robot in the factory, the paths have to be prepared from the very beginning in accordance with the new task of the robot. Doing this will cause extra cost and time loss. For these reasons, mobile robots used in the industry need to reach the target without any help, instead of moving on predetermined paths as they go to the target position. In order to do this, they need to know their current pose.

In this thesis, the localization study for indoor mobile robots that are used in the industry is performed in the simulation environment. MATLAB program and Gazebo simulation platform are used for this study. Codes written in MATLAB have been implemented in Gazebo using the MATLAB Robotics System Toolbox. The data required to perform robot localization was obtained with the laser sensor of the robot.

A two-stage method has been used for the localization of indoor mobile robots that are used for industrial purposes. In the first stage of the localization method, the pose of the robot was roughly found and then its pose was roughly tracked. In the second stage, the instantaneous pose of the robot was precisely estimated when it was close to a predetermined target point.

In the first stage, the Monte Carlo localization algorithm has been used. This method estimates the pose of the robot using the map of the environment when the pose of the robot is unknown. In order to implement the Monte Carlo localization method, functions that are available in the MATLAB Robotics System Toolbox have been used.

In the second stage, the PLICP algorithm has been used to increase the precision of pose estimates that have been found by Monte Carlo Localization algorithm. PLICP is an algorithm that aligns two given scans. In order to estimate the pose with PLICP, the translation vector and rotation matrix that have been generated with PLICP has been used.

In this thesis, an indoor environment has been prepared in the Gazebo platform to implement the two-stage localization method. In this environment, the robot has been given the task of shuttling between predetermined positions.

In the experiments, the robot did not know its pose at first. In the first stage of localization, robot found and tracked itself in the environment using the Monte Carlo localization method. Later, when the robot was approaching the target, the pose of the robot was estimated precisely using PLICP method. In order to apply this method, model scans were taken with the laser sensor in the target positions before the experiments. During the experiment, the robot computed the translational and rotational movements between the instantaneously taken laser scans and pre-acquired model scans using the PLICP algorithm when it was close to target. Since the position and direction of the robot were known at the time of receiving the model scans, the translation vector and the rotation matrix were used to compute the current location of the robot. The task of shuttling between the two targets of the robot was repeated at the specified number.

KESİN İÇ ORTAM LOKALİZASYONU İÇİN MCL VE ICP YÖNTEMLERİNİN BİRLEŞTİRİLMESİ

ÖZET

Gelişen teknoloji ile birlikte mobil robotlar daha sık kullanılmaya başlanmıştır. Özellikle endüstride üretimin daha efektif bir şekilde yapılabilmesi hedeflendiğinden mobil robotların kullanımına ihtiyaç duyulmaktadır.

Günümüzde endüstride yaygın olarak kullanılan mobil robotlar önceden planlanmış yollar üzerinde hareket etmektedirler. Bu yollar kablolar, manyetik bantlar veya robotun yolu takip etmesini sağlayan herhangi bir işaret yöntemi kullanılarak oluşturulmaktadır. Ancak bu yolları kullanmak robotun hareketini kısıtlamaktadır. Çünkü robotun fabrika içinde görev yaptığı konumların değiştirilmesi gerektiği takdirde bu yolların en baştan robotun yeni görevine uygun şekilde hazırlanması gerekir. Bunu yapmak ise ekstra maliyete ve zaman kaybına sebep olur. Bu sebeplerden dolayı endüstride kullanılan mobil robotların hedef konuma giderken önceden belirlenmiş yollar üzerinde hareket etmeleri yerine herhangi bir yardım almadan hedefe ulaşmaları gerekmektedir. Bunu yapabilmeleri içinse kendi konumlarını anlık olarak bilmeleri gerekir.

Bu tez çalışmasında endüstride kullanılan iç ortam mobil robotlarının simülasyon ortamında lokalizasyonu çalışması yapılmıştır. Bunun için MATLAB programı ve Gazebo simülasyon platformu kullanılmıştır. MATLAB üzerinde yazılan kodlar, MATLAB Robotics System Toolbox kullanılarak Gazebo'da uygulanmıştır.

Robotun lokalizasyonunu yapmak için gereken veriler, robotun üzerindeki lazer sensörü ile elde edilmiştir. Kullanılan lazer sensörü robotun ön tarafına doğru bakmakta olup yüz seksen derecelik lazer taraması yapmaktadır.

Endüstriyel amaçlı kullanılan iç ortam mobil robotlarının lokalizasyonunda iki aşamalı bir yöntem kullanılmıştır. Lokalizasyon yönteminin birinci aşamasında verilen bir haritada robotun konumu ve yönü kabaca bulunmuş ve daha sonra robotun konumu ve yönü kabaca takip edilmiştir. İkinci aşamada ise önceden belirlenmiş bir hedef noktaya yaklaştığı zaman robotun konumu ve yönü hassas bir şekilde hesaplanmıştır.

Birinci aşama için Monte Carlo Lokalizasyon algoritması kullanılmıştır. Bu yöntem robotun konumu ve yönü bilinmediği zamanlarda ortamın haritasını kullanarak robotun konumu ve yönünü bulmaktadır. Monte Carlo Lokalizasyon algoritması robotun muhtemel konumu ve yönünü temsil etmek için parçacıklar kullanmaktadır. Robotun konumu ve yönündeki değişimleri, uzaklık sensöründen gelen verileri ve ortamın haritasını kullanarak zamanla bu parçacıklar robotun gerçek konumu etrafında yoğunlaşmaktadır. Monte Carlo Lokalizasyon yöntemini uygulamak için MATLAB Robotics System Toolbox bünyesindeki hazır fonksiyonlar kullanılmıştır.

İkinci aşamada, Yinelemeli En Yakın Nokta (ICP) algoritmasının bir türü olan Noktadan Doğruya Metrik Yinelemeli En Yakın nokta (PLICP) yöntemi

kullanılmıştır. ICP algoritmaları verilen 2 veya 3 boyutlu iki tane şekli birbirleriyle hizalarlar. Bu iki şekildeki her bir noktanın diğer şekildeki hangi noktaya karşı geldiği bilinmediği için ICP algoritması öncelikle birinci şekildeki her bir nokta için ikinci şekildeki en yakın noktayı bulur. Daha sonra eşleştirilen noktaların arasındaki uzaklığı minimize edecek öteleme vektörünü ve rotasyon matrisini hesaplar. Son adımda ise hesapladığı öteleme vektörünü ve rotasyon matrisini birinci şekle uygular. Bu adımlar belirlenen kıstaslar sağlanıncaya kadar tekrarlanır. Bu kıstaslar bu tez çalışmasında ortalama karesel hatanın değişiminin belirlenen değerden az olması, ortalama karesel hatanın yeterince küçük olması ve belirlenen maksimum yineleme sayısına ulaşılması olarak belirlenmiştir.

ICP algoritmasında eşleştirilen noktaların arasındaki uzaklığı minimize etmek için noktadan noktaya metrik uygulanır. Bu tezde, noktadan noktaya metriği minimize etmek için tekil değer ayrışımı (SVD) tabanlı bir yöntem kullanılmıştır.

ICP algoritması, birinci şekil ile ikinci şeklin tamamen örtüşen şekiller olduğunu varsaymaktadır. Fakat bu her durumda geçerli olmayabilir. Örneğin, birinci şekildeki her noktanın ikinci şekilde karşılığı olmayabilir. ICP algoritması iki şekildeki her noktayı eşleştireceği için, gerçekte ikinci şekilde karşılığı olmayan birinci şekil noktalarını da eşleştirecektir. Bu da hatayı arttıracaktır. Bu sorunu aşmak için Kırpılmış Yinelemeli En Yakın Nokta (TrICP) yöntemi kullanılmıştır.

TrICP algoritması, en uygun öteleme ve rotasyonu hesaplarken bütün noktaları hesaba katmak yerine sadece belirlenmiş noktaları hesaba katar. Bunu yaparken de birinci şekildeki noktalar ile onlara en yakın ikinci şekil noktalarının arasındaki uzaklıkları hesaplar. Daha sonra bu uzaklıkları küçükten büyüğe doğru sıralar ve belirlenmiş sayıda en küçük uzaklığa sahip çiftleri seçer. Seçilmeyen noktalar ise aykırı değer olarak kabul edilir ve minimize etme adımında kullanılmaz. Dolayısıyla minimize etme adımında belirlenmiş sayıdaki en küçük uzaklıklı nokta çiftleri için en uygun öteleme ve rotasyon değerleri hesaplanır. Buradaki amaç fonksiyonu da ICP algoritmasındaki gibi noktadan noktaya metrik yapısında olduğu için, ICP algoritmasında uygulanan aynı yöntem ile minimize edilebilir. Bu tezde tekil değer ayrışımı tabanlı bir yöntem kullanılmıştır. Minimize etme adımında en iyi öteleme vektörü ve en iyi rotasyon matrisi bulunduktan sonra bulunan bu öteleme ve rotasyon hareketleri birinci şekle uygulanır. Bu adımdan sonra algoritma yine başa döner. Algoritma belirlenen kıstaslar sağlanıncaya kadar tekrarlanır. Bu kıstaslar ICP algoritmasında uygulanan kıstaslara çok benzerdir. ICP algoritmasındaki gibi tüm noktaları hesaba katarak hesaplanan ortalama karesel hata yerine sadece eleme aşamasını geçmiş noktalar için ortalama karesel hata hesaplanır. Bulunan ortalama karesel hata, ICP algoritmasında bahsedilen kıstaslar sağlanıncaya kadar yukarıda anlatılan adımlar tekrarlanır.

Noktadan Doğruya Metrik Yinelemeli En Yakın Nokta (PLICP) algoritması ise yukarıda anlatılan iki algoritmaya benzemektedir. PLICP algoritmasının ICP ve TrICP algoritmalarından farkı noktadan doğruya metrik kullanmasıdır. Bu yüzden birinci ve ikinci şekillerdeki en yakın noktalar eşleştirilirken buna ek olarak birinci şekildeki her bir noktaya ikinci en yakın ikinci şekil noktaları da bulunur. Yani ICP algoritmasında yapıldığı gibi en yakın noktalar arasındaki uzaklığı en aza indirgemeye çalışmak yerine, birinci şekildeki noktalardan ikinci şekilde bulunan ve birinci şekildeki ilgili noktaya en yakın iki nokta arasındaki doğruya olan uzaklık minimize edilmeye çalışılır. Eşleştirmeler yapıldıktan sonra TrICP algoritmasındaki eleme yöntemi uygulanır yani belirlenmiş sayıdaki en küçük uzaklığa sahip

eşleşmeler hesaba katılır ve diğer noktalar dışarıda bırakılır. Bu adımdan sonra elemeyi geçmiş noktalar için noktadan doğruya metrik minimize edilir. Noktadan doğruya metriğin yapısı noktadan noktaya metriğin yapısından farklı olduğu için bu adımda ICP ve TrICP'de yapılandan daha farklı bir yöntem kullanılır. Bir sonraki adımda ise minimize etme işlemi sırasında bulunan en uygun öteleme vektörü ve rotasyon matrisi birinci şekle uygulanır. Bu adımdan sonra algoritma yine birinci adıma döner. Bu tezde PLICP algoritmasını sonlandırmak için TrICP algoritmasını sonlandırmada kullanılan aynı kıstaslar kullanılmıştır.

Bu tezde iki aşamalı lokalizasyon yöntemini uygulamak için Gazebo platformunda bir iç ortam hazırlanmıştır. Burada robota önceden belirlenmiş noktalar arasında gidip gelme görevi verilmiştir. Bunun gerçekleştirilebilmesi için, hazırlanan lokalizasyon algoritmalarına ek olarak MATLAB Robotics System Toolbox bünyesindeki harita çıkarma, yol planlama ve yol takip fonksiyonlarından yararlanılmıştır.

Yapılan deney çalışmalarında robot başlangıçta kendi konumu ve yönünü bilmemektedir. İlk aşamada Monte Carlo Lokalizasyon yöntemini kullanarak konumu ve yönünü kabaca bulmakta ve takip etmektedir. Daha sonra hedef noktalara belirlenen uzaklık kadar yaklaştığında PLICP algoritması kullanılarak robotun konumu ve yönü hassas bir şekilde bulunmuştur. Bu yöntemin uygulanabilmesi için deneylerden önce hedef konumlarda lazer sensörü ile model taramalar alınmıştır. Deney sırasında robot hedefe yeteri kadar yaklaştığında PLICP algoritmasını kullanarak anlık olarak aldığı lazer taramalarının ve önceden alınmış model taramalarının arasındaki öteleme ve rotasyon hareketini hesaplamaktadır. Model taramalarının alındığı esnada robotun konumu ve yönü bilindiği için, bulunan öteleme vektörü ve rotasyon matrisi kullanılarak robotun tam o anda nerede olduğu hesaplanmıştır. Robotun iki hedef arasında gidip gelme görevi belirlenen sayıda tekrarlanmıştır.

1. INTRODUCTION

Localization is an important subject in mobile robotics. In order to fulfill its assigned tasks, the robot must know its position and orientation. Inaccurate position and orientation estimations might cause the robot to miss its target location and fail to complete its task.

Depending on the robot's task and its operating environment, the localization methods and the sensors to be used may vary. For example, in indoor environments, sometimes robots must pass from narrow paths to reach its destination. In such places, high accuracy in pose estimation is needed, for this reason, the localization method and the type of sensor must be chosen to give precise pose estimation results with priority. However, need of more precise results increases the cost of sensors and other components. Therefore, if the operating environment of robot is wide and if the task of the robot does not need too much precision then the sensors with lower precision and cost may be preferred. The same is valid for the localization methods. Since the way of different localization methods to estimate the pose of the robot vary, the properties of the environment is important when choosing localization method. Each localization method has different advantages in different types of environment. In addition to this, their accuracy and speed to estimate the pose of the robot may vary even though they are similar type of methods. In this case, they should be chosen according to the task of the robot. For example, if the primary requirement of the task is high accuracy and if the computation time of the localization algorithm is not main concern then the localization method with higher accuracy should be chosen.

For these reasons, it is better to determine the localization method and sensors to be used according to the conditions of the environment and the task of the robot. In this thesis, the localization study is performed in indoor and for industrial applications.

1.1 Problem Statement

In industrial applications, materials are commonly transported with forklifts driven by human operators or AGVs (Automated Guided Vehicle). Because of the human factor, forklifts are not efficient enough to satisfy the needs of new automated production techniques. Conventional AGVs are also not efficient because they can only operate on predetermined paths. To reach its destination, an AGV should follow a path of wires, magnetic tapes or markers. For this reason, in order to use AGVs, the factories or warehouses must be modified. Modifying the environment for AGVs increases the costs of production and wastes the time that can be used for production.

Consequently, usage of AGVs with designated paths are restricted and they are not suitable for new production techniques. For this reason, a more efficient way than using designated paths is needed to use mobile robots in industry. In this thesis, an industrial localization study is performed to estimate the pose of the robot precisely in order to use mobile robots for industrial applications without restricting the efficiency of production.

1.2 Hypothesis

In industry, the aim is optimizing the efficiency of production. For this reason, the mobile robots should complete their tasks with high speed, high precision and low cost.

In industrial applications, high precision in pose estimation is needed, for example, when the robot is approaching its target in order to place or pick a material. However, when the robot carries load between two locations, the speed of the robot is also important to complete its task earlier. In such cases, relatively low pose estimation accuracy will be enough to maintain the task. By doing so, the computational load and therefore computation time can be decreased when there is no need high accuracy. In addition to these, completing the same task with lower cost is another important subject for the mobile robots that are used in industry.

In this thesis, in order to accurately localize the industrial mobile robots when the high accuracy is needed, the combined localization method in [1] has been used. In [1], the position of the robot is estimated with an accuracy of few millimeters when the robot is approaching predesignated targets for instance in order to pick or place

an object. In this case a scan matching algorithm is used to estimate the pose of the robot accurately. At the time when there is no need such a high accuracy, a global localization method is used instead of scan matching method to track the position of the robot. By using this combined localization method, the position of the robot is estimated accurately at predetermined locations.

2. ROBOTICS MIDDLEWARE AND SOFTWARE TOOLS

In this thesis, ROS and MATLAB Robotics System Toolbox is used to communicate with the simulated TurtleBot robot that is available in Gazebo simulator. The technical details of TurtleBot are available in Appendix A. Also, some of the built in functions that are available in MATLAB Robotics System Toolbox is used in this study.

2.1 Robot Operating System

ROS (Robot Operating System) has a modular structure and consists of nodes that are communicating with each other. Each of these nodes has different tasks. For example, one node might control a camera, one node might publish laser sensor data and another node might perform localization by using this sensor data. In this way different parts of the robot perform its own task and communicate with each other when it is necessary.

Communication between the nodes is unmediated, however, each node should register with the ROS master and advertise its network address to be reachable by other nodes. There is only one ROS master in a ROS network and all of the nodes in the network are connected to that master [2].

Figure 2.1 shows the communication structure of a ROS network with three nodes. Each node registers with the master and declares its own network address. When a node wants to communicate with another node, each node gets the other node's address from the master, then these two nodes communicate directly among themselves [2].

Nodes communicate by sending messages. Each message has a message type that describes its data structure. Message type name consists of package name and type name. Figure 2.2 shows a message of type geometry_msgs/Twist where geometry_msgs is the package name and Twist is the type name [3].

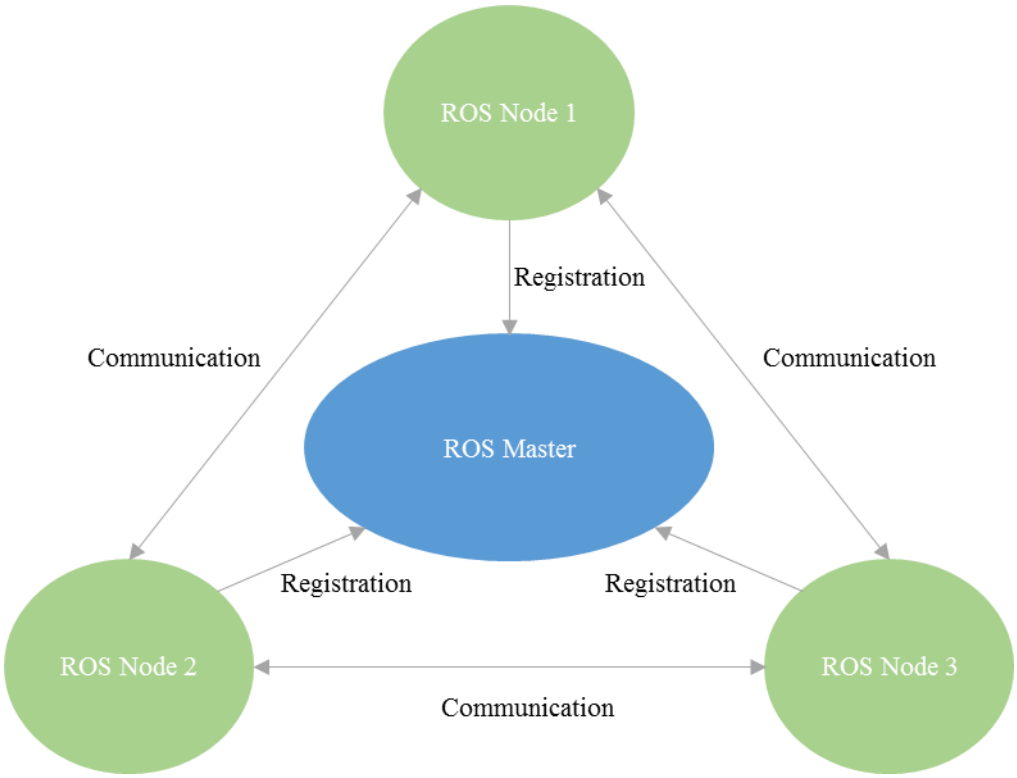

Figure 2.1: ROS communication structure.

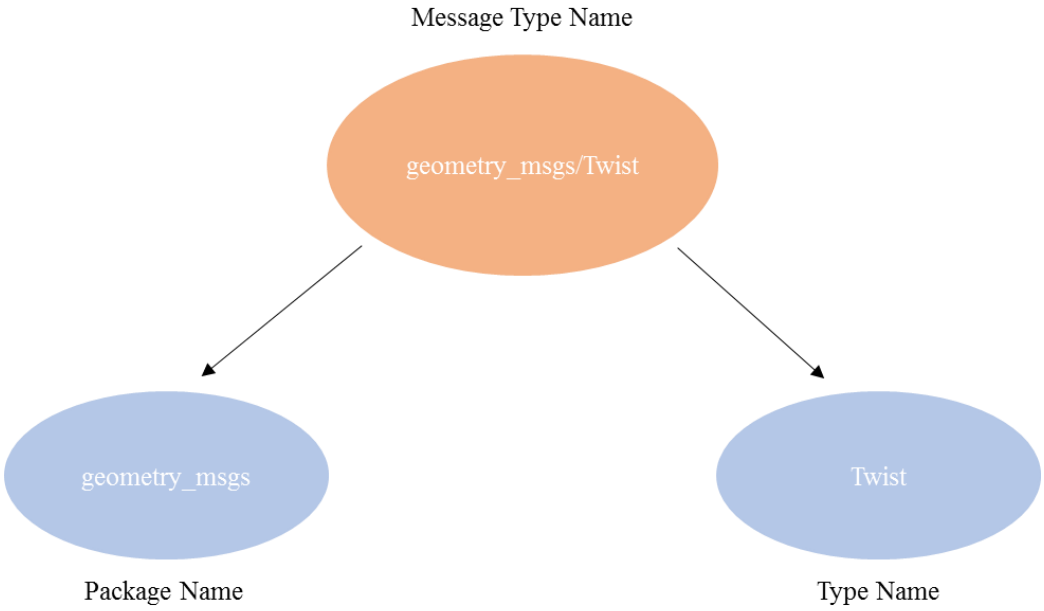

Figure 2.2: Example of a message type.

In ROS, there are two main communication methods. These are topics and services. Topics are used for defining the contents of messages. This means in each topic only one type of message is transmitted. To send messages to topics nodes use the publishers and to receive messages from topics nodes use the subscribers. In a topic there might be multiple publishers and multiple subscribers. Many nodes may send

messages to a topic and many nodes may receive these messages. However, it is not possible for subscribers to request data at a specific time. Publishers send their messages to the subscribers whenever new data is ready. To request data at a specific time services are used. Figure 2.3 shows a topic with 1 publisher and 2 subscribers. In this figure, node 1 sends messages of type sensor_msgs/LaserScan to the /scan topic with its publisher then node 2 and node 3 receive these messages by using their subscribers to the /scan topic [4].

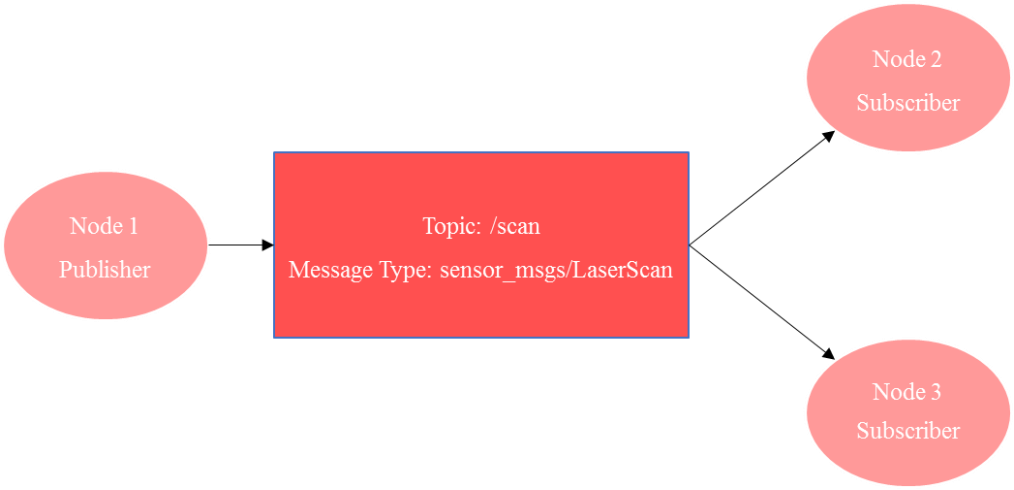

Figure 2.3: Communication by using topics.

As it is explained earlier, services allow requesting data at a specific time. Each service consists of two message structures. These are request and response messages. A service client sends request message to a service server. Then the service server processes the information in the request and replies this request by sending response message to the service client. In services, only one to one communication is possible. Figure 2.4 illustrates the service based communication [5].

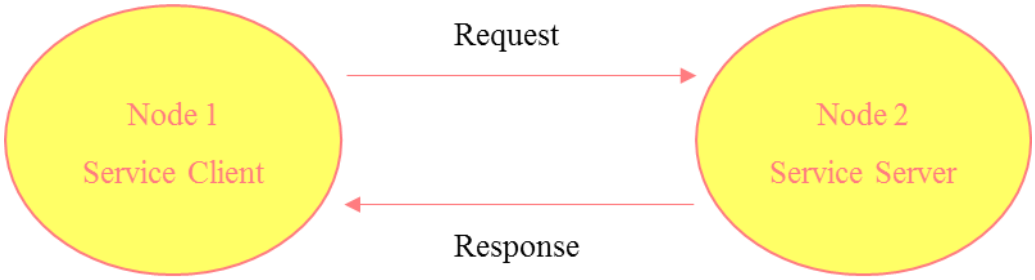

Figure 2.4: Communication by using services.

2.2 MATLAB Robotics System Toolbox

MATLAB Robotics System Toolbox allows to implement robotics algorithms by using MATLAB and Simulink. The toolbox connects MATLAB and Simulink to ROS. With this connection it is possible to develop robotics applications in MATLAB environment and implement them on robot simulators or robots that are using ROS. Figure 2.5 illustrates this connection.

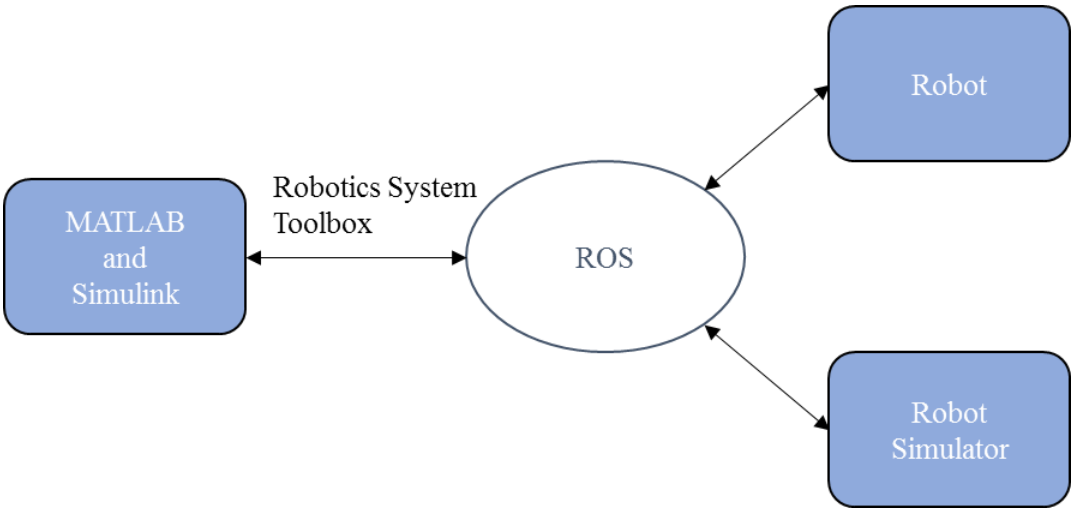

Figure 2.5: Diagram of the connection that is provided by Robotics System Toolbox.

Robotics System Toolbox could be used to connect to an external ROS master or it could launch the ROS master inside the MATLAB. The second method allows to use Robotics System Toolbox when there is no robot or robot simulator to work on. In either events, MATLAB communicates rest of the ROS network by creating its own ROS node called MATLAB global node. This node is registered with the ROS master like the other ROS nodes. Creating the global node and launching the ROS master inside the MATLAB is done by calling *rosinit* function without passing arguments. However, if the Robotics System Toolbox is used to connect to an external ROS master, then the address of the master should be specified when calling the *rosinit* function [6].

3. PATH PLANNING

In mobile robotics, to drive the robot from an initial position to a goal position a path planning algorithm is needed. Path planning algorithms enable to robot reach its target destination without colliding with obstacles and walls. For example, consider an industrial mobile robot that carries load between two locations. In order to reach its destination, it should not collide with obstacles. Also it should follow the shortest path between these locations since the time and energy loss are not desired in many applications especially in industrial applications. For this reason, a path planning algorithm is essential to reach the target location as soon as possible while avoiding the obstacles.

As it is stated in Chapter 1, the main purpose of this thesis is performing accurate localization for industrial applications. Therefore, for the path planning, Probabilistic Roadmap (PRM) [7] algorithm that is available in MATLAB Robotics System Toolbox was used. It uses the map of the environment to determine the path. For this reason, before explaining the path planner, mapping of the environment was explained.

3.1 Creating the Map of an Environment

In mobile robotics, maps of the environments are used for various purposes. In this thesis study, map of the environment was required for path planning and localization. For these reasons, an occupancy grid map of the environment with 20 cells per meter resolution is created by following the steps in [8] and using the *robotics.OccupancyGrid* class of MATLAB Robotics System Toolbox. This map is shown in Figure 3.1. As it can be seen from the figure, the colors of some locations are black, some locations are white and some locations are grey. These colors indicate the probability values of cells. In occupancy grid maps each cell has a probability value between 0 and 1 depending on the possible presence of an obstacle at that cell. The probability value is higher for a cell when it is more likely to be occupied by an obstacle. On the other hand, in Figure 3.1, the darkness of a cell

increases when its probability value increases. Which means darker locations are more likely to be occupied by an obstacle [9].

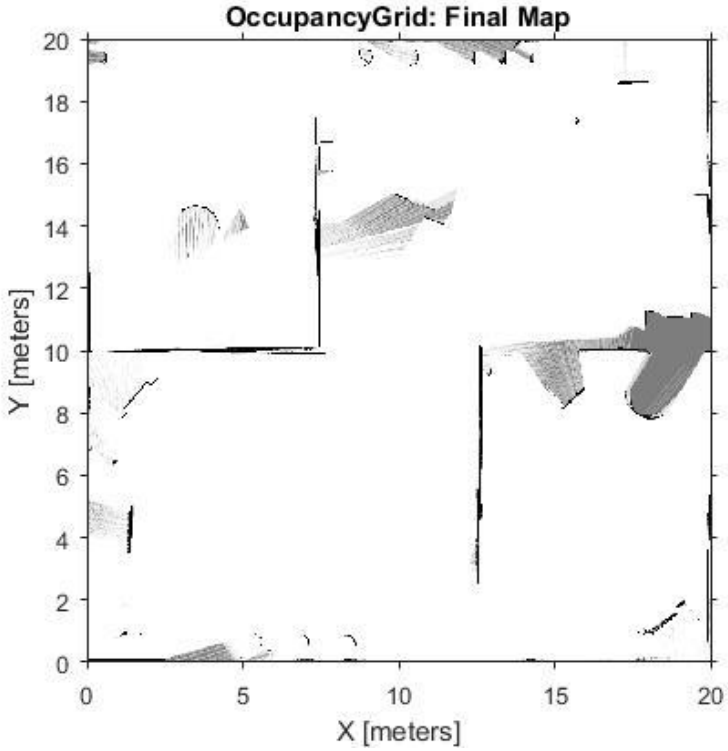

Figure 3.1: Occupancy grid map of the environment.

3.2 Probabilistic Roadmap Path Planner

In this study, PRM algorithm [7] was used in order to shuttle between predetermined locations without colliding with obstacles. It produces nodes randomly in free places of the environment. After the nodes are produced, it links these nodes to find a suitable path for the robot to reach its destination. PRM algorithm uses the map of the environment and determine which locations are obstacle free when producing and connecting nodes. Therefore, it does not produce nodes in the locations that are occupied by obstacles and it does not link two nodes if there is an obstacle between these nodes. As it is stated earlier, for the path planning part of this study, *robotics.PRM* class of MATLAB Robotics System Toolbox was used. This class allows to create PRM [7] path planner. In addition to this, Robotics System Toolbox's *inflate* command was used to inflate the map with the robot's dimension since the PRM path planner algorithm does not consider the robot's dimension [10].

For the path planning, same map in Chapter 3.1 is used. The inflated version of this map is shown in Figure 3.2.

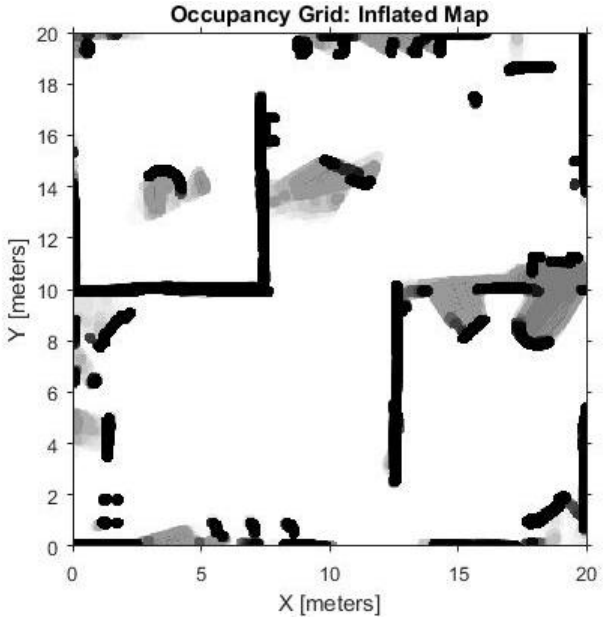

Figure 3.2: Inflated occupancy grid map of the environment.

After the map is inflated with robot's dimension, the number of nodes and longest allowed distance between two linked nodes must be determined. Increasing the number of nodes also increases the possibility of finding a path. On the other hand, it also increases the computation time. Likewise, increasing the longest allowed distance increases the possibility of finding a path in exchange for increased computation time [10]. An example for probabilistic roadmap that is created by using *robotics.PRM* class is shown in Figure 3.3. In this figure, a path is shown between two locations. For this example, 75 nodes are used. Also, the longest allowed distance between two nodes to connect with each other is 5 meters.

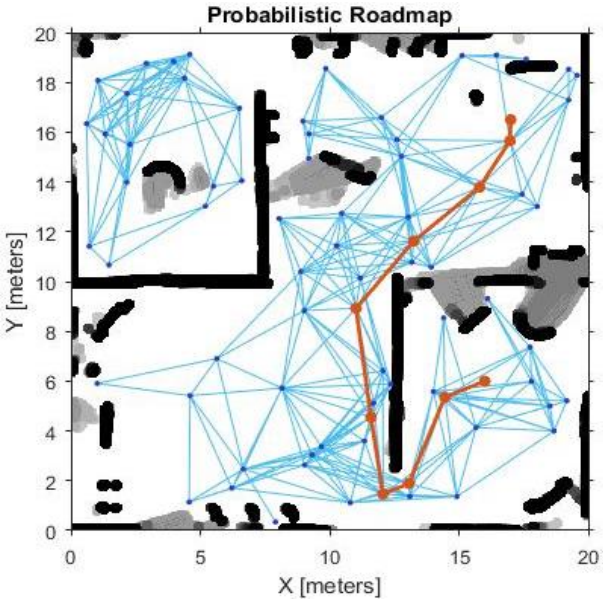

Figure 3.3: Probabilistic Roadmap between two locations.

4. TWO-STEP INDOOR LOCALIZATION

Localization is an important topic in mobile robotics because in order to follow a path a robot needs to know its pose which enables robot to determine whether its following the path correctly or not. Loss of information about its pose or incorrect pose estimations might cause robot to deviate from its path which may end up with missing the goal positions or even worse colliding with a human or an obstacle. For these reasons it is very important to localize the robot accurately. Accuracy in localization becomes even more important for industrial applications since even a minor mistake may interrupt the production in a factory because of the problems due to inaccurate localization that described above.

In order to localize the robot precisely, the combined localization technique in [1] is performed in this thesis study. In [1] Röwekämper et al. have combined the Monte Carlo Localization [11] with KLD sampling [12] and ICP variant with point-to-line metric [13] for the localization part of their works. They used Monte Carlo Localization (MCL) for global localization and position tracking, and used ICP variant with point-to-line metric (PLICP) to improve the pose estimates of MCL at goal positions. In order to apply PLICP, they have taken reference laser scans at goal positions. When the distance between the robot's MCL estimated position and the reference scan's position is decreased to 25 cm they used PLICP to improve the precision of pose estimates. In this way, they have localized the robot with a precision of few millimeters at predefined positions.

For this reason, in this thesis study, localization is divided into two parts. In the first part, global localization and position tracking by using Monte Carlo Localization algorithm is explained. In the second part, precise localization by using Iterative Closest Point algorithm and its variants is explained.

4.1 Monte Carlo Localization

In this study, Monte Carlo Localization [11] was used to localize the robot globally and track its pose continuously. As it is stated earlier, the purpose of this thesis study

is improving the precision of localization by using ICP and its variants. For this reason, the Monte Carlo Localization algorithm that is available in MATLAB Robotics System Toolbox was used in this study.

Monte Carlo Localization was introduced in [11] and it is a widely used method to estimate and track the robot's pose by using the map of the environment when there is no preliminary knowledge about the pose of the robot. It uses particles to indicate possible poses of the robot. At the beginning, it distributes particles randomly and uniformly to the everywhere in the map if there is no preliminary knowledge about the robot's pose. When robot moves, the particles are sampled depending upon robot's motion. When robot senses the environment, the algorithm assigns weights to particles based on the measurement. After this, the particles are resampled proportionally to their weights. By repeating these steps, the particles are concentrated on the position of the robot [14].

In order to apply the Monte Carlo Localization algorithm in MATLAB, *robotics.MonteCarloLocalization* object of the Robotics System Toolbox was used. The object uses the KLD sampling [12] method to improve the performance. KLD sampling method changes the number of particles during the process. Therefore, the more particles are used when there is more uncertainty about the robot's pose, and less particles are used when the uncertainty about the robot's pose is decreased.

4.2 Localization with ICP

In this study, it is aimed to estimate the position of the robot at specified goal locations with error less than 1 centimeter which is necessary in some cases when using the mobile robots in industrial applications. However, as it is experimentally shown below, it is not possible to reach this precision always by using the Monte Carlo Localization algorithm alone.

Figure 4.1 shows a PRM [7] to make the robot reach its target from its current position. In this figure, map is the occupancy grid map of the environment with 20 cells per meter. Occupancy grid map is created by using *robotics.OccupancyGrid* class and Probabilistic Roadmap is constructed using *robotics.PRM* class of the MATLAB Robotics System Toolbox. In order to construct the PRM, 500 nodes are

used. Also, in order to count the robot's dimension in PRM, map is inflated with the robot's dimension by using *inflate* command.

In the figure below, robot's starting position is $(x, y) = (10, 10)$ and the coordinates of the goal location is $(x, y) = (14.50, 15.40)$. The objective of the robot is stopping at the goal location with a maximum distance of 1 cm from the goal location.

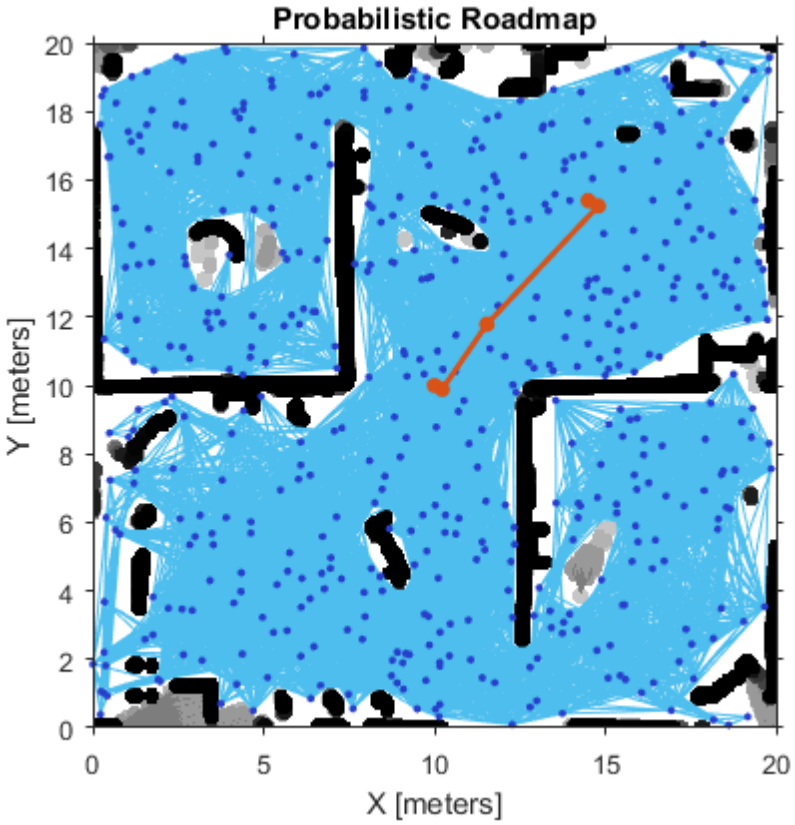

Figure 4.1: Probabilistic Roadmap of the robot.

By following the path that is shown above, it is aimed to stop the robot at the target location within a maximum distance of 1 cm from the target by using the pose estimations of MCL. In Figure 4.2**,** the real path of the robot is shown. This path is generated by obtaining the real positions of the robot from the simulator during the experiment.

In Figure 4.3, the zoomed version of the Figure 4.2 around the goal location is shown. As it can be seen, the robot is drawing circles around the target location because the MCL estimated positions are not enough accurate (less accurate than stopping condition which is 1 cm as stated above) and they misdirect the controller of the robot with this relatively inaccurate position values.

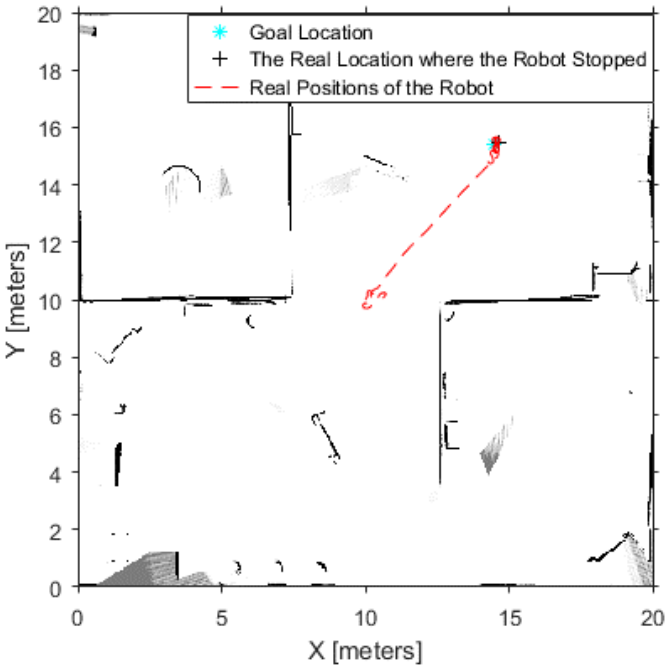

Figure 4.2: Real positions of the robot.

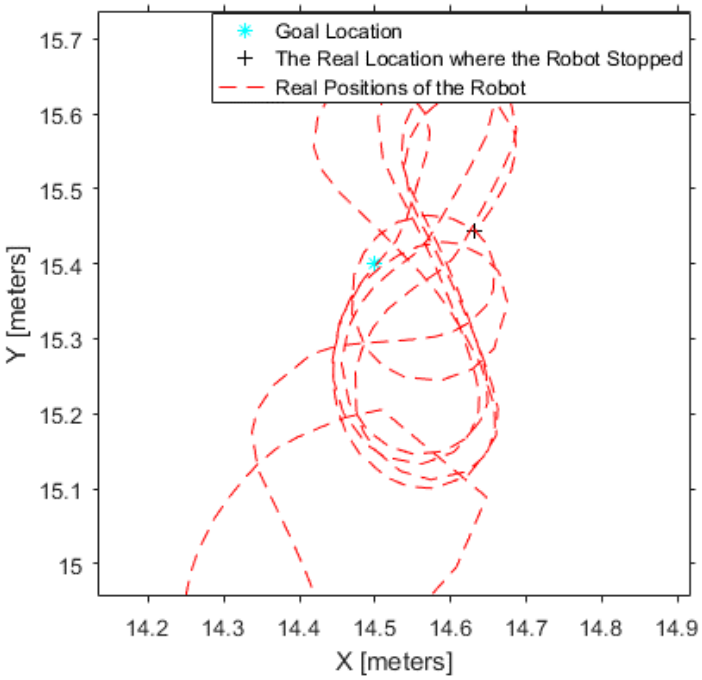

Figure 4.3: Real positions of the robot around the goal location.

In addition to this, the estimated positions are not enough accurate the check the stopping condition (the stopping condition is, stopping at a location within a maximum distance of 1 cm from the target) of the robot correctly. This causes the robot to stop in a wrong location. As it can be seen in Figure 4.4, the location where the robot stopped is 13.92 cm away from the goal location which is quite higher than the threshold of 1 cm.

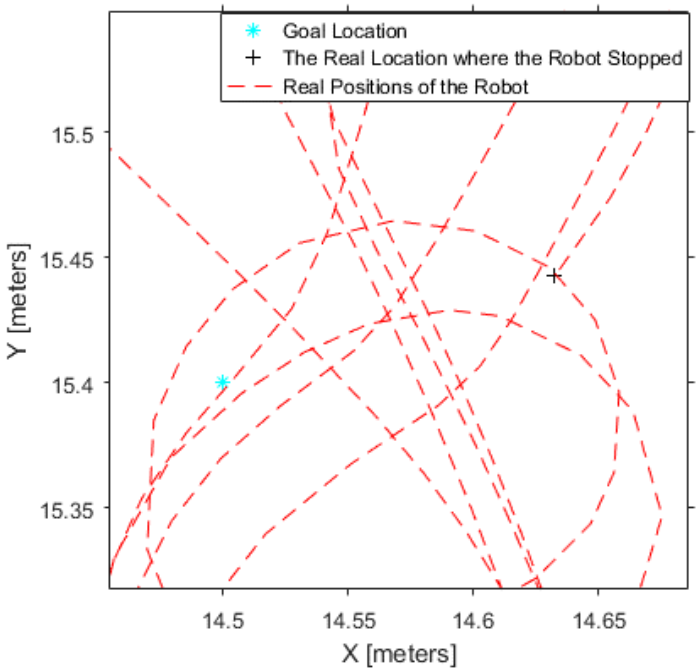

Figure 4.4: The difference between the goal location and the real location where the robot stopped.

However, the robot does not know the fact that it stopped in wrong location because it checks the stopping condition by using its estimated position values. According to these position values it stopped 7.5 mm far away from the target location which is lower than the threshold. This can be seen in Figure 4.5.

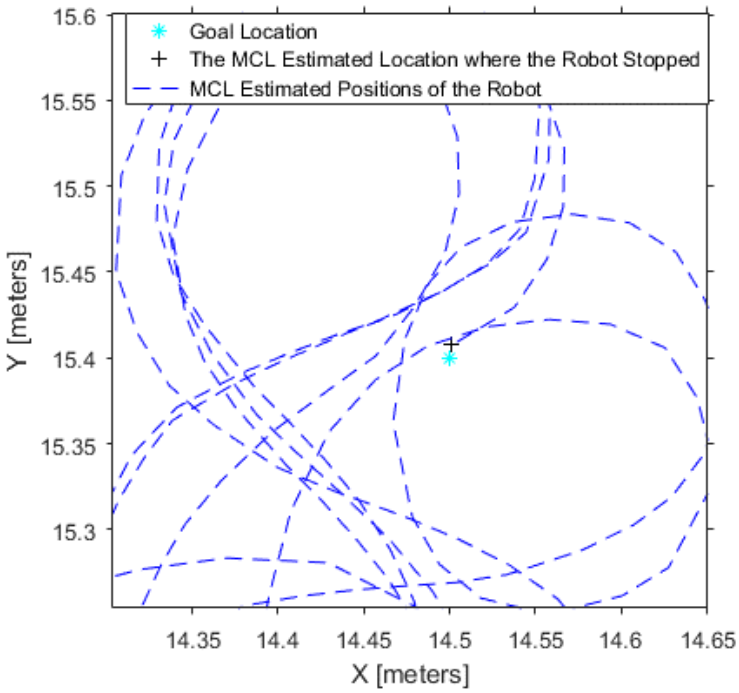

Figure 4.5: The difference between the goal location and the MCL estimated location where the robot stopped.

As it is experimentally shown above, achieving a sub centimeter precision with Monte Carlo Localization (MCL) is not possible, for this reason the combined localization technique in [1] is applied in this study. By following the steps of [1], the ICP with point-to-line metric (PLICP) algorithm is used to improve the position estimates of Monte Carlo Localization algorithm. To do this, when the robot gets close enough to goal location, the robot's pose is started to be estimated by using PLICP. To estimate the position, the transformation matrices that are generated with PLICP used because PLICP [13] computes the transformation that aligns two given scans. In this study, these scans are 2D point sets that are obtained by using the laser scanner of the robot. As it is explained in [1], the model scans that have been taken previously at the determined goal locations and scans that are taken at the robot's current position during the experiments are aligned by using PLICP. Then the resultant transformation matrix from the PLICP and the coordinates of the goal locations where the model scans were taken are used to estimate the position of the robot.

The PLICP [13] is a variant of Iterative Closest Point algorithm [15]. It uses point-to-line metric instead of point-to-point metric of original ICP [15]. In this chapter, in order to explain the PLICP more clearly, the original version of ICP and the trimmed ICP is explained firstly.

4.2.1 The Iterative Closest Point Algorithm

The ICP algorithm was developed by Besl and McKay [15] to iteratively align two 3D shapes. Consider 3D data point set *F* and model point set *G*. Since it is not known at the beginning that which point in *F* corresponds to which point in *G*, the algorithm matches the closest points in these two sets to each other. Then the transformation that minimizes the distance between matched closest points in *F* and *G* is computed. At the last step, this transformation is applied to *F*. These three steps are iterated. After each iteration, these two point sets get closer to each other and overlaps at the end of the algorithm [15].

These three steps can be shown as below:

1 - Match each point of *F* with the closest point in *G*.

2 - Compute the transformation that minimizes the distances between the matched points.

3 - Apply the transformation to *F*.

These three steps are iterated until converging to an optimal transformation.

In the first step of the algorithm a correspondence search procedure is applied. Let N_F and N_G are the number of points in *F* and *G* respectively. Then the data set *F* is formed as:

$$F = \{f_1, f_2, f_3, \ldots, f_{N_F}\} \tag{4.1}$$

and the model set *G* is formed as:

$$G = \{g_1, g_2, g_3, \ldots, g_{N_G}\} \tag{4.2}$$

The Euclidean distance between a point f_i in data set and a point g_j in model set is:

$$d(f_i, g_j) = \|f_i - g_j\| \tag{4.3}$$

The distance between a data point f_i and the model set *G* is:

$$d(f_i, G) = \arg\min_{g \in G} d(f_i, g) \tag{4.4}$$

Solving the equation 4.4 gives the closest point in *G* that has the minimum distance between the point f_i . Let h be the closest point in *G*. In this situation the equation 4.4 can be written as follows:

$$d(f_i, h) = d(f_i, G) \tag{4.5}$$

where $h \in G$. By denoting *Corr* as the correspondence operator which pairs the closest points of *F* and *G* as explained above, the set of closest points *H* can be shown as follows [15]:

$$H = Corr(F, G) \tag{4.6}$$

In the second step, the transformation that minimizes the distance between the closest points is computed. Classical ICP [15] uses point-to-point distance metric that sums squared distances between corresponding data and model points to compute rotation matrix *R* and translation vector *t* which minimize the distance between corresponding points. This distance function can be shown as follows:

$$J(R, t) = \sum_{i=1}^{N_F} \|h_i - Rf_i - t\|^2 \tag{4.7}$$

where h_i is the closest point in model point set that corresponds to the point f_i in data point set. In this thesis, to find the *R* and *t* that minimizes the point-to-point metric a

Singular Value Decomposition (SVD) based method [16] is used. This minimization method can be found in Appendix B.

In the third step of the ICP algorithm, the rotation matrix *R* and translation vector *t* which are computed in the second step of the ICP, are applied to data point set *F*.

$$F = RF + t \tag{4.8}$$

After the step three, algorithm returns the step 1. These three steps are repeated until change of mean square error is smaller than designated value [15].

In this thesis, the stopping conditions that are described in [17] are used:

- change of mean square error (MSE) is smaller than designated value
- the designated number for maximum iterations has been reached.
- MSE is small enough to terminate the iterations.

For the stopping conditions MSE is found as follows:

$$\text{MSE} = \frac{1}{N_F}\sum_{i=1}^{N_F}\|h_i - f_i\|^2 \tag{4.9}$$

which is the mean of squared distances between data point f_i and its corresponding model point h_i (closest point to f_i in model point set). In [15] Besl and McKay have proved that ICP algorithm always converges monotonically to a local minimum in terms of mean square error. This proof is shown in Appendix C.

In addition to ICP steps described above there might be need of an initial registration to align *F* and *G* before applying the ICP algorithm. The algorithm only guarantees the converging to a local minimum, hence, to converge to a global minimum good initial states are necessary as it is stated in [15]. In Chapter 6, how the initial registration has been done in this study will be explained.

Since the ICP steps were described in details, now the ICP algorithm can be stated. Let *k* denote the iteration number of the ICP algorithm. In this situation, H_k denotes the set of closest points and F_k denotes the data point set at the *k* th iteration of the algorithm. Also R_k and t_k are the rotation matrix and translation vector that applied to data point set at the *k* th iteration. Then the ICP algorithm is as shown below [15]:

Initial registration: $F_1 = R_0F_0 + t_0$ where $F_0 = F$

Initialize the iteration: $k = 1$

1. Match each point of *F* with the closest point in *G*.

   $H_k = Corr(F_k, G)$

2. Compute the transformation that minimizes the sum of the squared distances between the matched points.

   $$(R_k, t_k) = min \sum_{i=1}^{N_F} \left\| h_{i,k} - R_k f_{i,k} - t_k \right\|^2$$

3. Apply the transformation to *F*.

   $F_{k+1} = R_k F_k + t_k$

4. If one of the stopping conditions described before are satisfied terminate the iteration; otherwise return to step 1 and increase the iteration number *k* by '1'.

4.2.2 The Trimmed Iterative Closest Point Algorithm

The classical ICP [15] described above assumes that the data point set *F* and model point set *G* fully overlap, however, for example when working with laser scan data which is the case in this thesis, this assumption will no longer be true since the two scans might be taken from different locations which means they are not fully overlap with each other [18]. An example for this situation is shown in Figure 4.6 below.

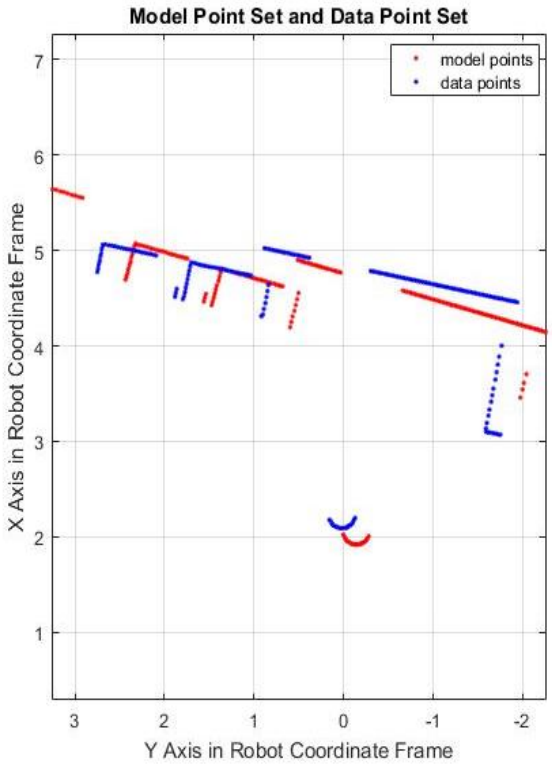

Figure 4.6: Example for partially overlapping point sets.

As it can be seen in above figure, some data points have no correspondences in model point set and using the classical ICP will give incorrect results because it will also match these data points with model points even though these model points are not the true correspondences of the outlying data points. In addition, in the presence of measurement noise the number of these incorrect matchups will increase and therefore accuracy of results will decrease more.

Since the classical ICP algorithm does not consider outliers as explained above, Chetverikov et al. [19] proposed a new algorithm called the Trimmed ICP (TrICP) to increase the robustness. The difference between the ICP and TrICP is that the TrICP computes the optimal transformation for the determined number of data points instead of considering all of the data points. The number of data points that are used in minimization step is determined according to minimum overlap rate of data points. Minimum overlap rate is the least known rate of data points that have true correspondences in model set. Let m_{or} be the minimum overlap rate. In that case, the number of data points that are going to be used in minimization is [19]:

$$N_{FM} = N_F m_{or} \tag{4.10}$$

As it is stated in [19], the value of minimum overlap rate can be determined by repeating the same experiment with different m_{or} values and selecting the m_{or} which gives the best result. Also a method for automatically determining the m_{or} is given in [19]. In this thesis study, m_{or} is determined according to the experiment results.

After deciding the how many data points have true correspondences in model points, which data points and their corresponding model points are used in minimization step is decided. This selection is done by sorting the squared distances between each data point f_i and its corresponding model point h_i in ascending order and selecting the N_{FM} number of matchups with smallest squared distances. This means $N_F - N_{FM}$ number of data points are considered as outliers. These outliers and their corresponding model points are discarded and is not going to be used in minimization step [19].

In the minimization step, objective function is very similar to the objective function that is used in classical ICP. However, this time the objective function consists of remaining data and model points after the outliers are discarded. Also, it must be

noted that the point indexes are changed after elimination of the outliers. By considering these differences, the objective function is:

$$J(R,t) = \sum_{j=1}^{N_{FM}} \left\| h_j - Rf_j - t \right\|^2 \tag{4.11}$$

where j is the new index after the elimination. Same minimization technique that is used in original ICP can be applied to this objective function because both of the objective functions are in point-to-point metric form [19].

After the minimization step, the rotation matrix R and translation vector t which are computed in equation 4.11, are applied to data point set F.

$$F = RF + t \tag{4.12}$$

After this step, algorithm returns the first step. These steps are repeated until one of the designated stopping conditions are satisfied as it is described in [19]:

- the trimmed MSE is small enough to terminate the iterations.
- the trimmed MSE difference between consecutive iterations are small enough to terminate the iterations.
- the designated number for maximum iterations has been reached.

where

$$\text{trimmed MSE} = \frac{1}{N_{FM}} \sum_{j=1}^{N_{FM}} \left\| h_j - f_j \right\|^2 \tag{4.13}$$

In [19] it is stated that the TrICP algorithm always converges monotonically to a local minimum in terms of trimmed mean square error.

Since the steps of the TrICP algorithm has been described the algorithm can be stated as shown below [19]:

Initial registration: $F_1 = R_0 F_0 + t_0$ where $F_0 = F$

Initialize the iteration: $k = 1$

1. Match each point of F with the closest point in G.
   $H_k = Corr(F_k, G)$
2. Sort the squared distances between each data point $f_{i,k}$ and its corresponding model point $h_{i,k}$ in ascending order and select the N_{FM} number of matchups with smallest squared distances.

3. Compute the transformation that minimizes the sum of the squared distances between the remaining matched points.

$$(R_k, t_k) = min \sum_{j=1}^{N_{FM}} \left\| h_{j,k} - R_k f_{j,k} - t_k \right\|^2$$

j = index after eliminating the outliers

4. Apply the transformation to *F*.

$$F_{k+1} = R_k F_k + t_k$$

5. If one of the stopping conditions described before are satisfied terminate the iteration; otherwise return to step 1 and increase the iteration number *k* by '1'.

4.2.3 The Iterative Closest Point Algorithm with Point-to-Line Metric

ICP with Point-to-Line Metric (PLICP) is a variant of ICP which uses point-to-line metric instead of point-to-point metric that is used in classical ICP. PLICP is stated by Censi [13] and according to the experimental results in [13], it is more accurate and need less iterations. It uses the trimming process [19] described in Chapter 4.2.2 to eliminate outliers. However as it is stated in [13], the ICP is more robust than PLICP when the rotational displacement between the two scans are large. The steps below describe the PLICP by comparing it with ICP.

The differences between PLICP and ICP come from their objective functions. As it is stated before, ICP uses point-to-point metric, however, PLICP uses point-to-line metric. Consider the data point f_i in point-to-point metric, the distance error for each point is the distance between data point f_i and its corresponding closest point in the model point set *G*. On the other hand in point-to-line metric, the distance error for each point is the distance between data point f_i and the line which lies between f_i's closest point correspondence and f_i's second closest point correspondence in the model point set *G*. As it was stated earlier, h_i is the f_i's corresponding closest point in model point set and *H* is the set of the closest points. In that case, let z_i be the f_i's second closest point correspondence in model point set and *Z* be the set of second closest points. Also, by taking into consideration that *Corr* was denoted as correspondence operator, the set of closest points *H* and the set of second closest points *Z* can be written as follows [13]:

$$(H, Z) = Corr(F, G) \tag{4.14}$$

In order to find the optimal transformation, the point-to-line metric objective function is given as follows [13]:

$$J(R,t) = \sum_{i=1}^{N_F} [n_i^T (h_i - Rf_i - t)]^2 \quad (4.15)$$

where n_i is the normal to the line that lies between h_i and z_i. To find the *R* and *t* that minimizes the point-to-line metric an exact closed form solution is stated in [13]. This solution is given in Appendix D.

Since the other steps of PLICP is same as the steps of TrICP, the algorithm of PLICP can be shown as [13]:

Initial registration: $F_1 = R_0 F_0 + t_0$ where $F_0 = F$

Initialize the iteration: $k = 1$

1. Match each point of *F* with the closest and second closest points in *G*.
   $(H_k, Z_k) = Corr(F_k, G)$
2. Sort the squared distances between each data point $f_{i,k}$ and its corresponding model point $h_{i,k}$ in ascending order and select the N_{FM} number of matchups with smallest squared distances.
3. Compute the transformation that minimizes the sum of the squared distances between the point $f_{j,k}$ and the line that lies between points $h_{j,k}$ and $z_{j,k}$

   $$(R_k, t_k) = \sum_{j=1}^{N_{FM}} [n_j^T (h_{j,k} - R_k f_{j,k} - t_k)]^2$$

   j = index after eliminating the outliers
4. Apply the transformation to *F*.
   $F_{k+1} = R_k F_k + t_k$
5. If one of the stopping conditions described before are satisfied terminate the iteration; otherwise return to step 1 and increase the iteration number *k* by '1'.

5. SIMULATION STUDIES

In this chapter, simulation studies are performed for the two-step indoor localization method this is explained in Chapter 4. As it is stated earlier, the purpose of this localization method is estimating the pose of the robot accurately at predefined target locations. However, in order to travel between these target locations, the robot needs a path planner and a path following controller. For this reason, a path planning algorithm was explained in Chapter 3. Also, in the following chapter a path following controller that enables the robot travel between target locations is explained.

5.1 Pure Pursuit Path Following Controller

In Chapter 3, a path planning algorithm has been described. The output of that path planning algorithm is an array of waypoints where the first waypoint is robot's current position and the last waypoint is robot's goal position. In order to reach its goal position, the robot must follow that waypoints. Following the path that consists of these waypoints enable robot to reach its goal position while avoiding the obstacles in the environment. In order to follow the path, the speed and direction of the robot must be adjusted accordingly. For this reason, a controller is needed to continuously adjust the robot's linear and angular velocities. In this study, the Pure Pursuit path tracking algorithm [20] that is available in MATLAB Robotics System Toolbox was used.

The Pure Pursuit controller [20] allows the robot to follow the path by driving the robot from its current position to a target point on the path by adjusting the steering angle of the robot. When the robot moves, the target point also moves, so there is a distance between the robot and the target point. This distance is called look ahead distance. Look ahead distance is the only parameter in the controller. For this reason, it is very important to determining a proper value for look ahead distance. Choosing a short look ahead distance might cause more overshoots on the path however it enables robot to retrieve the path sooner if the robot is not following the path

accurately at that time. On the other hand, longer look ahead distance enables smoother path following but the robot will retrieve the path slower if it has missed the path already [21].

In order to apply the controller, *robotics.PurePursuit* object of MATLAB Robotics System Toolbox was used.

5.2 Simulation Environment

In this study, simulations were performed on Gazebo Simulator. The connection between the Gazebo and the MATLAB was made by using MATLAB Robotics System Toolbox.

For the experiments, an environment which is shown in Figure 5.1 and Figure 5.2 was designed in Gazebo.

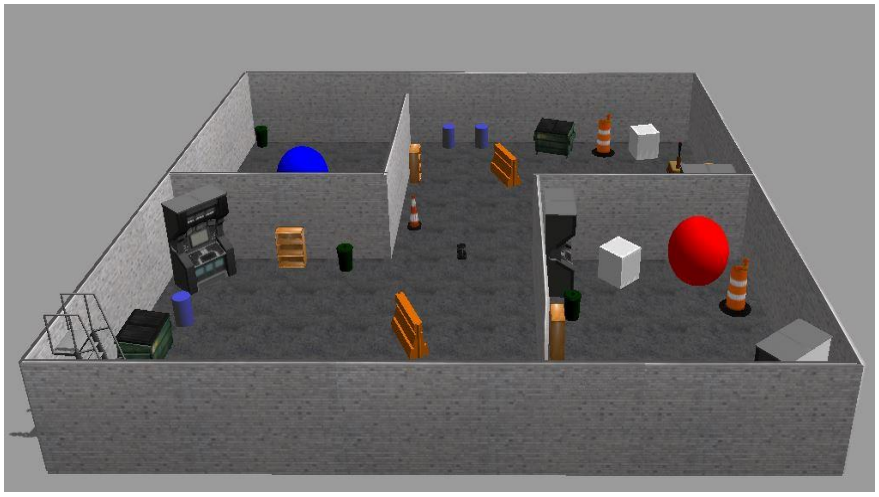

Figure 5.1: The front view of environment that is used for simulations.

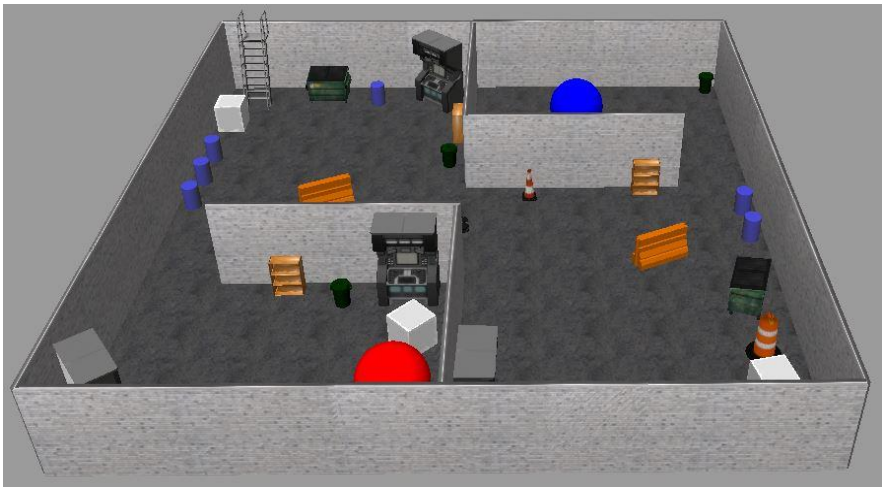

Figure 5.2: The side view of environment that is used for simulations.

The occupancy grid map of the environment can be seen in Figure 5.3. It is created by using *robotics.OccupancyGrid* class of the MATLAB Robotics System Toolbox.

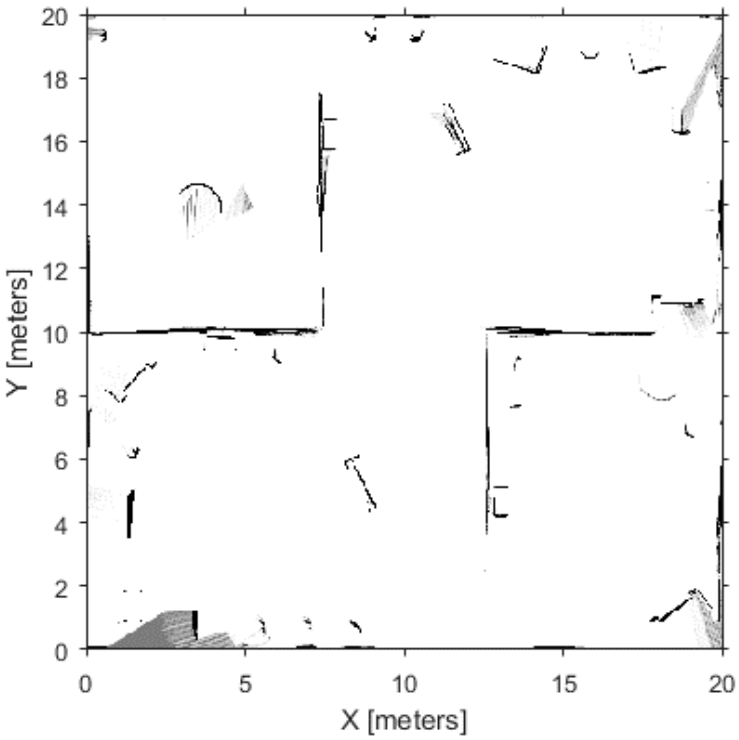

Figure 5.3: Occupancy grid map of the environment.

The TurtleBot that is shown in Figure 5.4 was chosen as the robot to perform given tasks in simulation environment. Technical details of TurtleBot can be found in Appendix A.

Figure 5.4: The TurtleBot.

5.3 Case Study

In this chapter, a case study was performed in the environment that is described in Chapter 5.2. In this case study, the robot traveled between Target Location 1 and Target Location 2. It has been assigned with the task of visiting each location 100 times and stopping at each target location with a maximum distance of 1 cm from the target point. Also in order to make the simulations more realistic, noise which has normal distribution was applied on laser scans. The noise has zero mean and 0.003 standard deviation. It was applied both of the x and y coordinate values of each point.

At the beginning, the robot started its journey from the middle of the map. However, it was unaware of its pose. By using the Monte Carlo Localization algorithm [11] it has found its pose in the environment and then started to travel between Target Location 1 and Target Location 2.

The coordinates of the Target Location 1 are:

$$(x_1, y_1) = (14.50, 16.00) \tag{5.1}$$

The coordinates of the Target Location 2 are:

$$(x_2, y_2) = (16.00, 6.00) \tag{5.2}$$

The target locations can be seen in Figure 5.5.

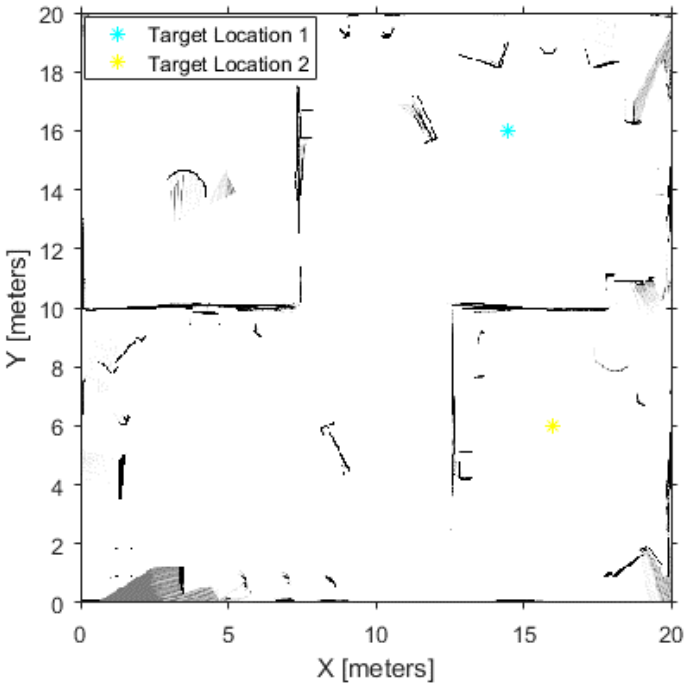

Figure 5.5: Target locations.

In order to apply the PLICP [13] algorithm to estimate the pose of the robot, the steps that are described in [1] were applied. By following these steps, firstly, model laser scans had been taken at the target locations before the experiments. This was done by sending the robot to each target location and scanning the environment by rotating the robot. In this thesis study, each model scan was taken by rotating the robot 5° with respect to its previous position. This value was determined experimentally. At the time the first scan was taken the robot's angle had been set to 0° and at the time the last scan was taken the robot's angle had been set to 180° with respect to world coordinate frame. For this reason, there were 37 model scans saved at each target.

Figure 5.6 and Figure 5.7 shows two example model scans taken at the Target Location 1. In Figure 5.6, the robot's angle is 45.3152° and in Figure 5.7 the robot's angle is 90°. The laser data are shown in robot coordinate frame. The image data are shown in order to visualize the robot coordinate frame more clearly. The selection of which model scan is to be used in PLICP was done by using MCL's orientation estimations about robot's angle at that time. This is because the Monte Carlo Localization algorithm's position and orientation estimations are used as initial guess in PLICP. For this reason, the conditions of in which cases the PLICP is used to estimate the robot's pose are determined by MCL estimations [1].

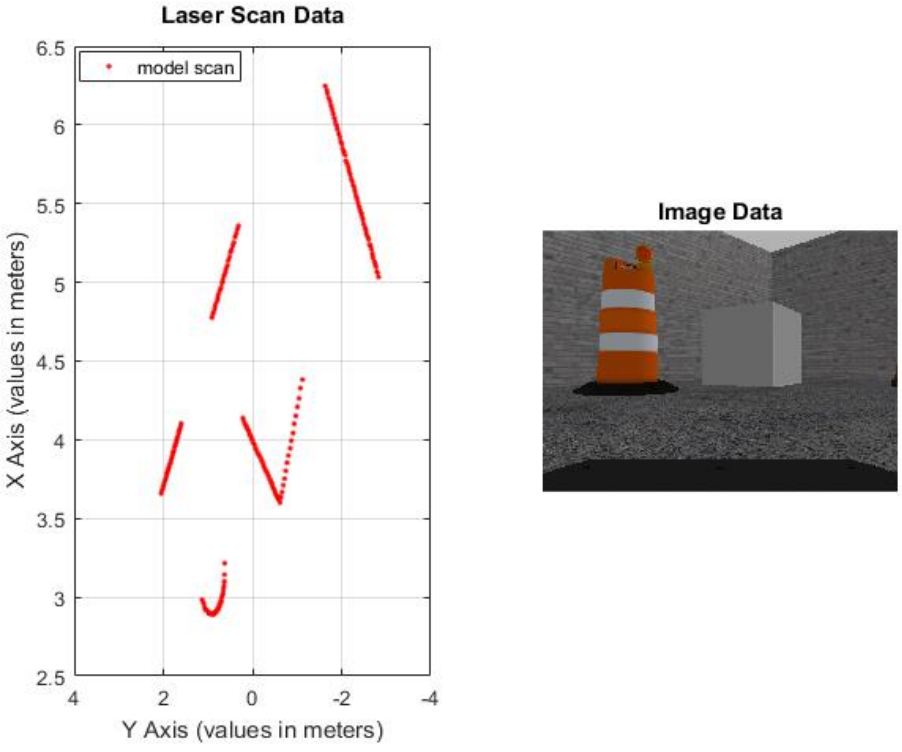

Figure 5.6: Laser and image data when the robot's angle is 45.3152°.

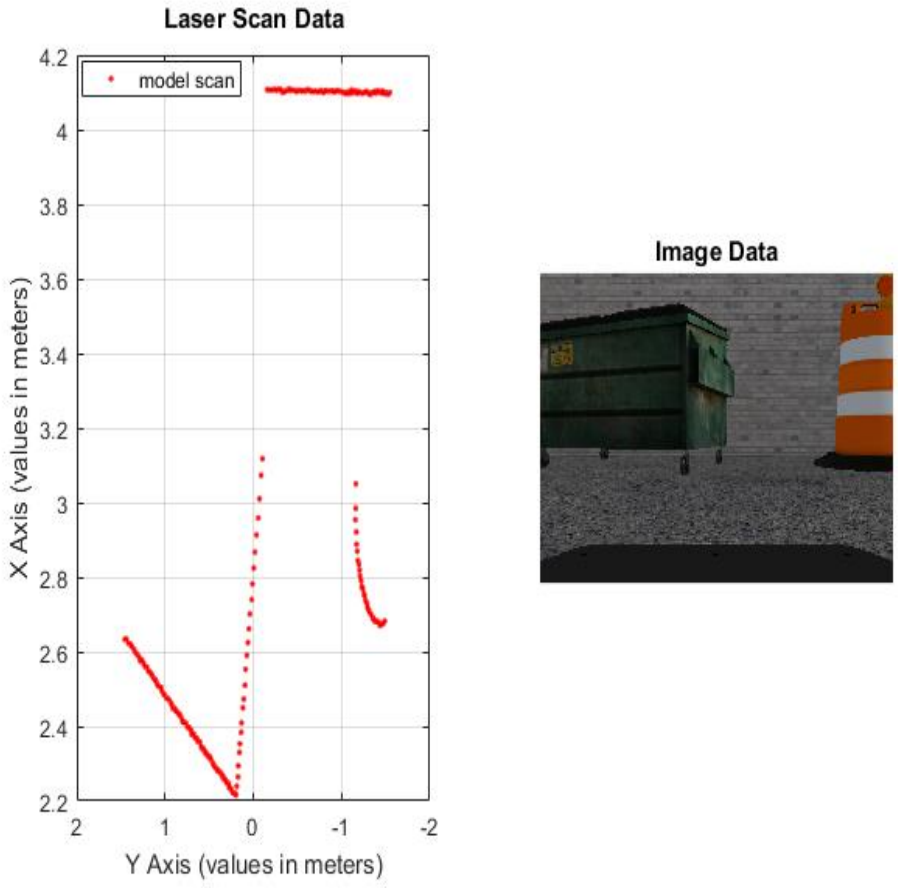

Figure 5.7: Laser and image data when the robot's angle is 90°.

In this thesis study, in order to obtain the robot's pose from PLICP, the method that is proposed in [22] was applied. Since the laser sensor is mounted on the robot, the coordinate values of points in laser scans are with respect to the robot's coordinate frame. Also, since the PLICP was used to generate the translation vector and rotation matrix between the laser scans, then the robot's pose in world coordinate frame can be found by transforming the translation and rotation values from robot's coordinate frame to world coordinate frame [22].

5.4 Simulation Results

As it is stated earlier, the robot's task was 100 times shuttling between Target Location 1 and Target Location 2. Also it must have stopped itself at each target with a maximum distance of 1 cm from the target point. The robot's instantaneous positions during the experiment can be seen in Figure 5.8. The robot's starting position which is the middle of the map can also be seen in this figure.

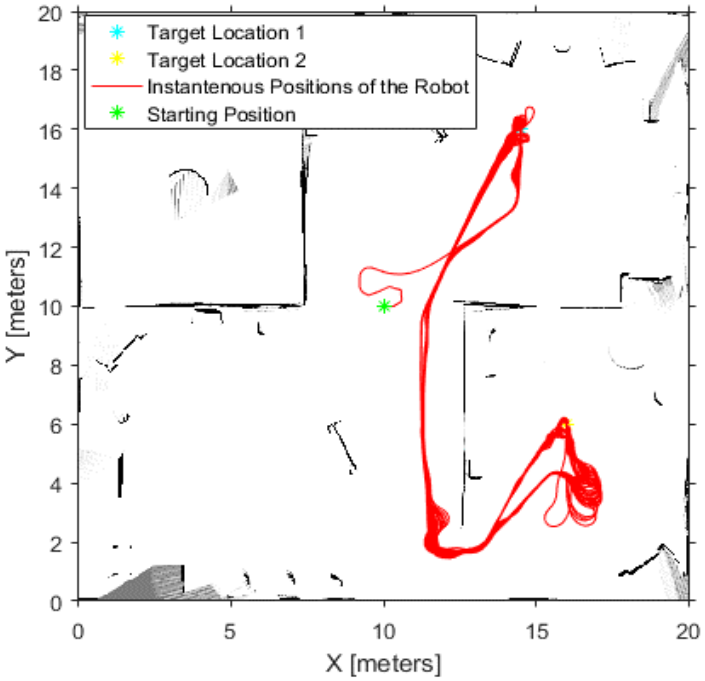

Figure 5.8: The robot's instantaneous positions during the experiment.

In Figure 5.9, the robot's average and maximum x, y position estimation errors are shown for Target Location 1 and Target Location 2. For the Target 1, the average x axis position estimation error is 4.2 mm and the average y axis position estimation error is 1.9 mm. Also, the maximum x axis position estimation error is 13.9 mm and the maximum y axis position estimation error is 10.6 mm. For the Target 2, the average x axis position estimation error is 4.8 mm, the average y axis position estimation error is 3.3 mm. On the other hand, maximum x axis position estimation error is 17.6 mm and the maximum y axis position estimation error is 12.5 mm.

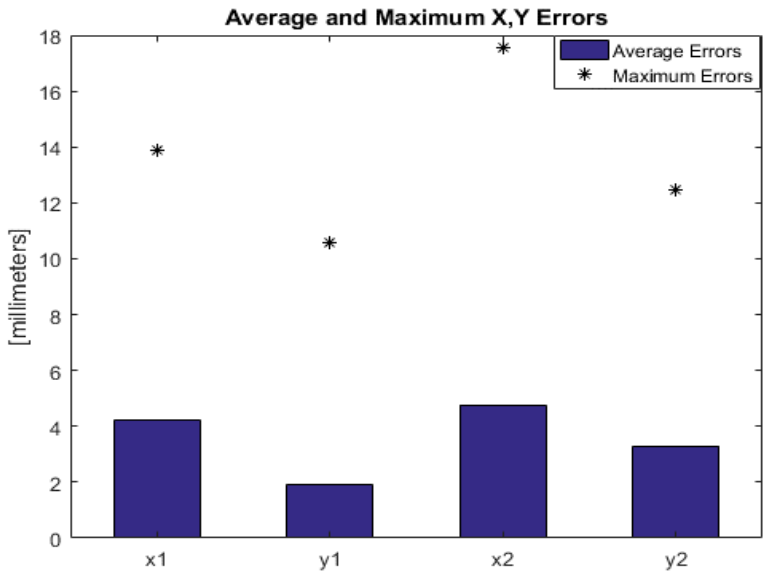

Figure 5.9: The x and y axes position estimation errors.

In Figure 5.10, the robot's average and maximum angular estimation errors are shown for Target Location 1 and Target Location 2. For the Target 1, the average angular error is 1.75° whereas the maximum angular error is 3.1907°. For the Target 2, the average angular error is 1.6975° and the maximum angular error is 2.8999°.

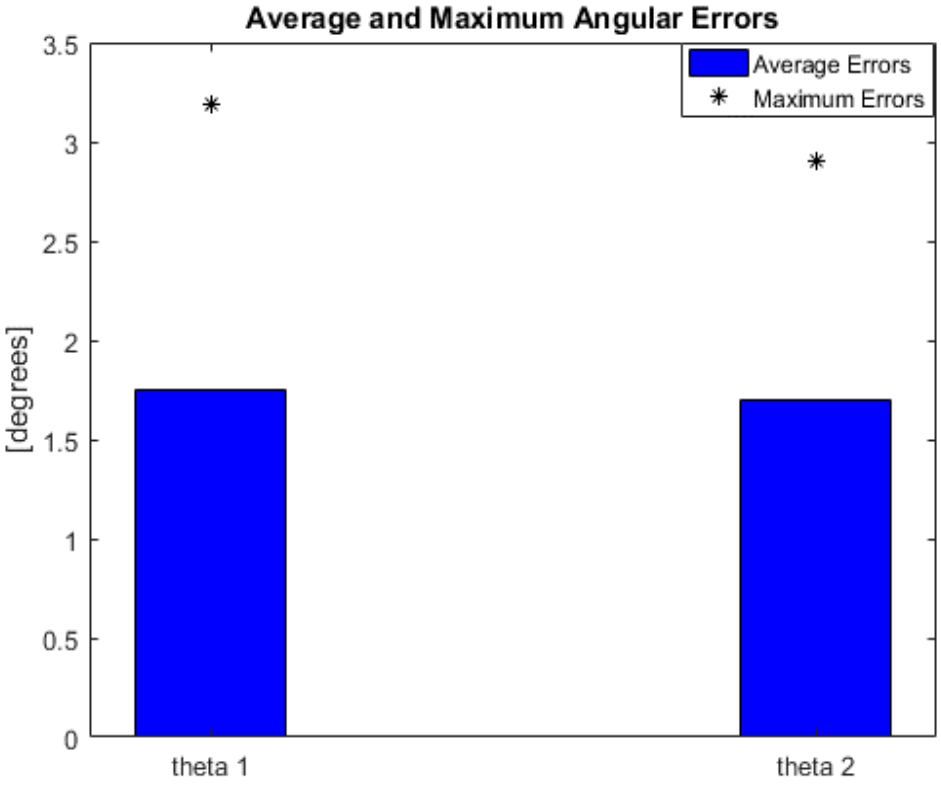

Figure 5.10: The angular estimation errors.

In Figure 5.11 and Figure 5.12, the points where the robot stopped at Target Location 1 and Target Location 2 are shown respectively. The magenta colored circle has a radius of 1 cm. Being inside of that circle means the robot's positioning error is less than 1 cm. Since the robot was given a task of stopping with a maximum distance of 1 cm from each target point, sometimes it has stopped outside the circles even though its position estimation error is less than 1 cm. For example, consider that the robot is 1.5 cm away from the target point, in this situation a position estimation error with 0.6 cm makes the robot believes that it is 0.1 cm inside of the circle.

In Figure 5.11, the farthermost point corresponds to the failure of the PLICP algorithm. When it failed, the result of PLICP was discarded and the MCL estimation at that time was used automatically for the pose estimation. Since the precision of the MCL is lower than PLICP, a much higher error in contrast to PLICP estimations was occurred. In order to prevent such failings, sensor redundancy may be considered.

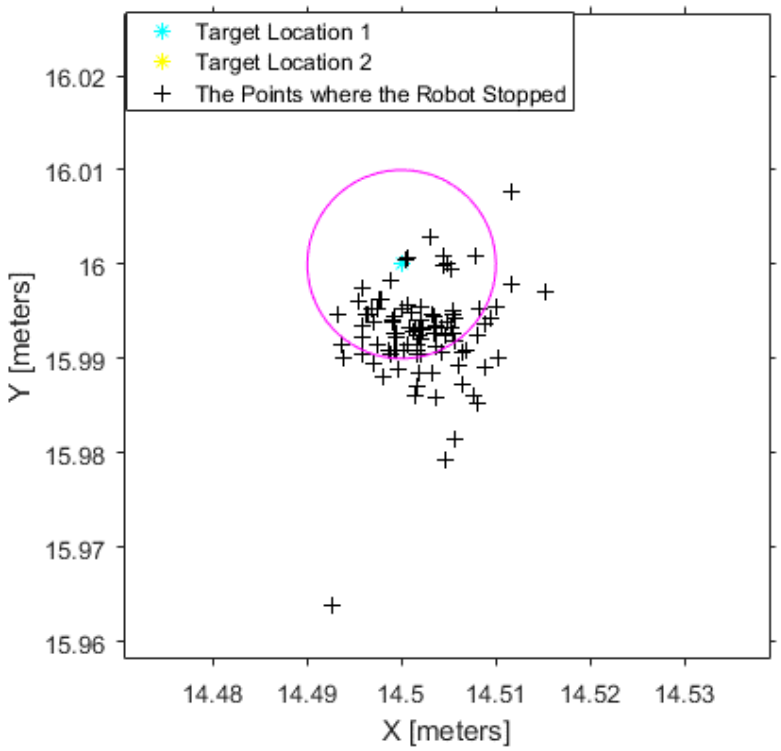

Figure 5.11: The points where the robot stopped at Target Location 1.

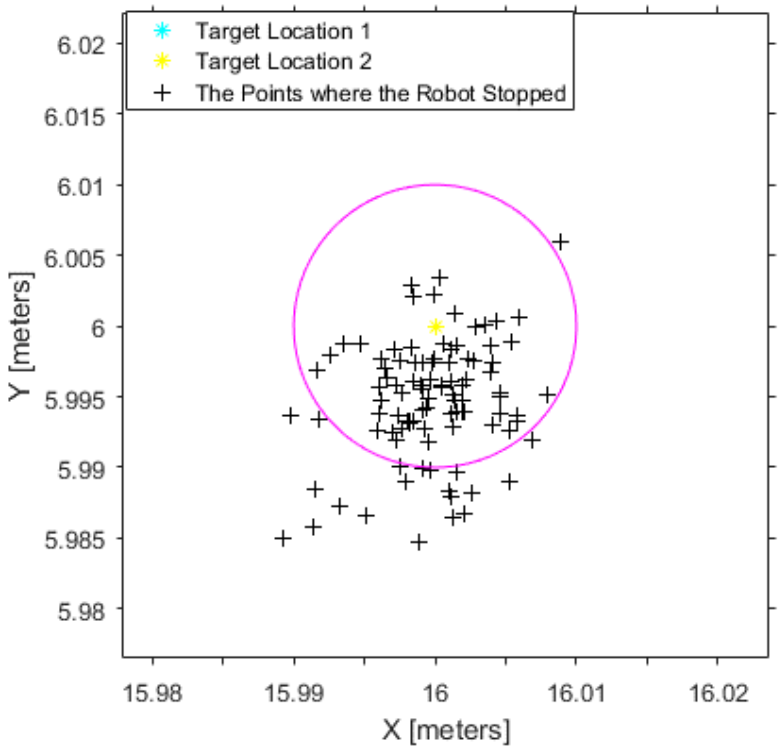

Figure 5.12: The points where the robot stopped at Target Location 2.

6. CONCLUSION

In this thesis, a study for accurate localization of indoor mobile robots was performed. The localization accuracy is very important in industrial applications because in these applications, mobile robots must perform tasks that require positioning accuracy of sub-centimeter. In order to achieve a sub-centimeter positioning accuracy, a localization method that estimates the pose of the robot with same accuracy is needed. However, at the times when there is no need sub-centimeter accuracy, a less accurate localization method can be used to decrease the computational load. For this reason, a two-step indoor localization method was applied in this thesis. In this method, the robot is globally localized and then its pose is tracked coarsely in a known environment by using the MCL algorithm. When the robot approaches the predetermined locations to perform accuracy needed tasks, the PLICP algorithm is used for estimate the pose of the robot accurately.

The simulation studies performed in this thesis show that by using the two-step localization method the position of the robot can be estimated at predefined target locations with sub-centimeter accuracy.

The simulations were performed on static environment. However, by using a proper obstacle avoidance algorithm that allows the robot to avoid obstacles in dynamic environment, the two-step indoor localization method can also be tested in dynamic environments.

Also, it is a well-known problem that the ICP based algorithms might be trapped in local minimum. In this thesis, in order to avoid local minimum in PLICP, the MCL estimations were used to choose the model scans that are closest to given data scans. Even though, this method reduces the possibility of getting trapped in local minimum it is possible to obtain better results with an algorithm that guarantees global minimum.

Finally, in order to increase the robustness of the localization method, a second laser sensor can be used. This allows to perform two independent PLICP, compare their results and detect possible failures.

REFERENCES

[1] **Röwekämper, J., Sprunk, C., Tipaldi, G. D., Stachniss, C., Pfaff, P., & Burgard, W.** (2012). On the position accuracy of mobile robot localization based on particle filters combined with scan matching. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 3158–3164).

[2] **Url-1** <*http://www.mathworks.com/help/robotics/ug/ros-network-setup.html*>, date retrieved 29.08.2016.

[3] **Url-2** <*http://www.mathworks.com/help/robotics/examples/work-with-basic-ros-messages.html*>, date retrieved 30.08.2016.

[4] **Url-3** <*http://www.mathworks.com/help/robotics/examples/exchange-data-with-ros-publishers-and-subscribers.html*>, date retrieved 30.08.2016.

[5] **Url-4** <*http://www.mathworks.com/help/robotics/examples/call-and-provide-ros-services.html*>, date retrieved 31.08.2016.

[6] **Url-5** <*http://www.mathworks.com/help/robotics/examples/connect-to-a-ros-network.html*>, date retrieved 01.09.2016.

[7] **Kavraki, L. E., Svestka, P., Latombe, J-C., Overmars, M. H.** (1996). Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *Robotics and Automation, IEEE Transactions on*, *12*, 566–580.

[8] **Url-6** <*http://www.mathworks.com/help/robotics/examples/mapping-with-known-poses.html*>, date retrieved 20.11.2016.

[9] **Url-7** <*http://www.mathworks.com/help/robotics/ug/occupancy-grids.html*>, date retrieved 20.11.2016.

[10] **Url-8** <*http://www.mathworks.com/help/robotics/examples/path-planning-in-environments-of-different-complexity.html*>,date retrieved 13.11.2016.

[11] **Dellaert, F., Fox, D., Burgard, W., & Thrun, S.** (1999). Monte Carlo localization for mobile robots. *Proceedings 1999 IEEE International Conference on Robotics and Automation*, *2*, 1322–1328.

[12] **Fox, D.** (2003). Adapting the Sample Size in Particle Filters Through KLD-Sampling. *The International Journal of Robotics Research*, *22*, 985–1003.

[13] **Censi, A.** (2008). An ICP variant using a point-to-line metric. In *Proceedings - IEEE International Conference on Robotics and Automation* (pp. 19–25).

[14] **Url-9** <*http://www.mathworks.com/help/robotics/examples/localize-turtlebot-using-monte-carlo-localization.html*>, date retrieved 15.11.2016.

[15] **Besl, P. J., & McKay, N. D.** (1992). A Method for Registration of 3-D Shapes. *IEEE Trans. Pattern Anal. Mach. Intell. (), 14*, 239–256.

[16] **Arun, K. S., Huang, T. S., & Blostein, S. D.** (1987). Least-Squares Fitting of Two 3-D Point Sets. *IEEE Trans. Pattern Anal. Mach. Intell. (), 9*, 698–700.

[17] **Trucco, E., Fusiello, A., & Roberto, V.** (1999). Robust motion and correspondence of noisy 3-{D} point sets with missing data. *Pattern Recognition Letters*, *20*, 889–898.

[18] **Segal, A., Haehnel, D., & Thrun, S.** (2009). Generalized-ICP. In *Robotics: Science and Systems* (Vol. 2, p. 4).

[19] **Chetverikov, D., Svirko, D., Stepanov, D., & Krsek, P.** (2002). The Trimmed Iterative Closest Point algorithm. *Object Recognition Supported by User Interaction for Service Robots*, *3*, 0–3.

[20] **Amidi, O., & Thorpe, C. E.** (1991). Integrated mobile robot control. In W. H. Chun & W. J. Wolfe (Eds.), (pp. 504–523). International Society for Optics and Photonics.

[21] **Coulter, R. C.** (1992). *Implementation of the Pure Pursuit Path Tracking Algorithm* (Report No: CMU-RI-TR-92-01). Robotics Institute, Carnegie Mellon University

[22] **Wang, X., Jia, Y., Xi, N., Zhen, J., & Xu, F.** (2013). Mobile robot pose estimation using laser scan matching based on Fourier Transform. In *2013 IEEE International Conference on Robotics and Biomimetics (ROBIO)* (pp. 474–479). IEEE.

[23] **Clearpath Robotics.** (2015). *TurtleBot* [Data sheet]. Retrieved 1 September 2016 from https://www.clearpathrobotics.com/turtlebot-2-open-source-robot/

APPENDICES

Appendix A: The TurtleBot
Appendix B: Minimization of Point-to-Point Metric
Appendix C: Convergence Theorem of ICP
Appendix D: Minimization of Point-to-Line Metric

Appendix A The TurtleBot

In this study, TurtleBot was used to perform docking tasks and compare different localization algorithms in the simulation environment.

In Table A.1 external dimensions, weight of the robot, diameter of wheels and ground clearance are shown [23].

Table A.1: Size and weight.

External dimensions (L x W x H)	Weight	Wheels (Diameter)	Ground clearance
354 x 354 x 420 mm	6.3 kg	76 mm	15 mm

As it is seen in Table A.2, TurtleBot has a maximum speed of 0.65 m/s. Also it could carry loads up to 5 kg [23].

Table A.2: Speed and performance.

Maximum payload	Maximum speed	Maximum rotational speed
5 kg	0.65 m/s	180 °/S

TurtleBot has a 2200 mAh lithium ion standard battery. To have more execution time, larger battery with 4400 mAh may also be used. Battery and power system of the TurtleBot are shown in Table A.3 [23].

Table A.3: Battery and power system.

Standard battery	Extended battery	User power
2200 mAh lithium ion battery	4400 mAh lithium ion battery	5 V and 19 V (1 A), 12 V (1.5 A and 5 A)

Appendix B Minimization of Point-to-Point Metric

Consider the following objective function in [15] :

$$J(R,t) = \sum_{i=1}^{N_F} \|h_i - Rf_i - t\|^2 \quad \text{(B 2.1)}$$

where h_i is the closest point in model point set that corresponds to the point f_i in data point set.

In this thesis, to find the *R* and *t* that minimizes the objective function a Singular Value Decomposition (SVD) based algorithm is used [16]. By following the steps in [16], first define the centroids of *F* and *H* as follows:

$$\bar{f} = \frac{1}{N_F} \sum_{i=1}^{N_F} f_i \quad \text{(B 2.2)}$$

$$\bar{h} = \frac{1}{N_F} \sum_{i=1}^{N_F} h_i \quad \text{(B 2.3)}$$

Consider that the centroids of data point set *F* and closest point set *H* should overlap when *F* and *H* are aligned correctly. For this reason, the translation vector *t* is chosen to move rotated data point set centroid to closest point set centroid. By doing so, the translation vector will be eliminated from the equation B 2.1 and the objective function will be minimized with respect to *R* matrix only. After that, the translation vector will be computed from the optimal rotation matrix. Let cf_i be the i th point in centered set of *F* and ch_i the i th point in centered set of *H* as shown below:

$$cf_i = f_i - \bar{f} \quad \text{(B 2.4)}$$

$$ch_i = h_i - \bar{h} \quad \text{(B 2.5)}$$

Now, write f_i in terms of cf_i and $\bar{f}$ then write h_i in terms of ch_i and $\bar{h}$. The new form of equation B 2.1 is shown as below:

$$J(R,t) = \sum_{i=1}^{N_F} \left\| ch_i + \bar{h} - Rcf_i - R\bar{f} - t \right\|^2 \quad \text{(B 2.6)}$$

As explained earlier, translation vector that minimizes objective function is chosen to move rotated data point set centroid to closest point set centroid.

$$t = \bar{h} - R\bar{f} \quad \text{(B 2.7)}$$

Put the equation B 2.7 into equation B 2.6. Then the new objective function will be shown as follows:

$$J(R) = \sum_{i=1}^{N_F} \|ch_i - Rcf_i\|^2 \quad \text{(B 2.8)}$$

Since the translation vector was eliminated from the objective function, *R* that minimizes the equation B 2.8 should be found. After finding the optimal *R* the translation vector will be found from the equation B 2.7.

To compute the optimal *R*, expand the equation B 2.8 as shown below:

$$J = \sum_{i=1}^{N_F} (ch_i - Rcf_i)^T (ch_i - Rcf_i) \quad \text{(B 2.9)}$$

$$J = \sum_{i=1}^{N_F} (ch_i{}^T ch_i - ch_i{}^T Rcf_i - cf_i{}^T R^T ch_i + cf_i{}^T R^T Rcf_i) \quad \text{(B 2.10)}$$

$$J = \sum_{i=1}^{N_F} (ch_i{}^T ch_i - 2ch_i{}^T Rcf_i + cf_i{}^T cf_i) \quad \text{(B 2.11)}$$

In order to minimize the objective function, let

$$L = \sum_{i=1}^{N_F} ch_i^T Rcf_i \quad \text{(B 2.12)}$$

Then minimizing the objective function is equal to maximizing the *L*.

$$L = \sum_{i=1}^{N_F} ch_i^T Rcf_i = Trace(\sum_{i=1}^{N_F} Rcf_i ch_i^T) \quad \text{(B 2.13)}$$

$$L = Trace(RA) \quad \text{(B 2.14)}$$

where

$$A = \sum_{i=1}^{N_F} cf_i \, ch_i^T \quad \text{(B 2.15)}$$

Now, the SVD of *A* is considered [16]:

$$A = U\Lambda V^T \quad \text{(B 2.16)}$$

Then the optimal rotation matrix *R* that maximizes the *L* and therefore minimizes the objective function is:

$$R = VU^T \quad \text{(B 2.17)}$$

In some cases, determinant of VU^T might be equal to -1. In this situation *R* is a reflection rather than a rotation. To fix this, the sign of the third column of the *V* is changed as follows:

$$V' = [v_1, v_2, -v_3] \quad \text{(B 2.18)}$$

where v_1, v_2, v_3 are the first, second and third columns of *V* respectively.

$$R = V'U^T \quad \text{(B 2.19)}$$

Since the optimal rotation matrix *R* is found the optimal translation vector *t* is found as described earlier:

$$t = \bar{h} - R\bar{f} \quad \text{(B 2.20)}$$

Appendix C Convergence Theorem of ICP

It is stated and proved in [15] that the ICP algorithm always converges monotonically to a local minimum in terms of mean square error. This theorem is based on two ideas. The first one is that average distance between the data points and their corresponding closest points in the model set is decreasing with each iteration because of the least squares registration. The other idea is that the distance for each point is decreasing separately in the correspondence search procedure which also yields a smaller average distance between data points and model points [15].

Consider the ICP [15] steps explained in Chapter 4.2.1. Let *k* be the iteration number of the ICP algorithm. In this case, F_k denotes the data point set and H_k denotes the set of closest points at the *k* th iteration of the algorithm. Then, the MSE at *k* th iteration can be written as follows:

$$E_k = \frac{1}{N_F}\sum_{i=1}^{N_F}\left\|h_{i,k} - f_{i,k}\right\|^2 \quad \text{(C 3.1)}$$

Let R_k and t_k be the rotation matrix and translation vector that applied to data point set at the *k* th iteration. When the R_k and t_k are applied to F_K the MSE after the transformation will be as follows:

$$E'_k = \frac{1}{N_F}\sum_{i=1}^{N_F}\left\|h_{i,k} - R_k f_{i,k} - t_k\right\|^2 \quad \text{(C 3.2)}$$

$$E'_k = \frac{1}{N_F}\sum_{i=1}^{N_F}\left\|h_{i,k} - f_{i,k+1}\right\|^2 \quad \text{(C 3.3)}$$

where $f_{i,k+1} = R_k f_{i,k} + t_k$ as it is explained in Chapter 4.2.1.

It is clear that the error after the transformation is always smaller than the error before the transformation due to least squares minimization. Hence:

$$E'_k \leq E_k \quad \text{(C 3.4)}$$

Each time after the transformation, a new set of closest points H_{k+1} will be generated in corresponding point search process. In this case MSE is shown as below:

$$E_{k+1} = \frac{1}{N_F}\sum_{i=1}^{N_F}\left\|h_{i,k+1} - f_{i,k+1}\right\|^2 \quad \text{(C 3.5)}$$

Note that the distance will be reduced for each point after the closest point search:

$$\left\|h_{i,k+1} - f_{i,k+1}\right\|^2 \leq \left\|h_{i,k} - f_{i,k+1}\right\|^2 \text{ for each } i = 1, 2, \dots, N_F \quad \text{(C 3.6)}$$

Hence:

$$\frac{1}{N_F}\sum_{i=1}^{N_F}\left\|h_{i,k+1} - f_{i,k+1}\right\|^2 \leq \frac{1}{N_F}\sum_{i=1}^{N_F}\left\|h_{i,k} - f_{i,k+1}\right\|^2 \quad \text{(C 3.7)}$$

This is equal to:

$$E_{k+1} \leq E'_k \quad \text{(C 3.8)}$$

In equation C 3.4, it is shown that $E'_k \leq E_k$. Combining the equations C 3.4 and C 3.8 results:

$$E_{k+1} \leq E'_k \leq E_k \quad \text{(C 3.9)}$$

By generalizing the equation C 3.9 for each iteration the inequality shown below is obtained.

$$0 \leq E'_{k+1} \leq E_{k+1} \leq E'_k \leq E_k \text{ for each } k \quad \text{(C 3.10)}$$

As it is shown in equation C 3.10, the MSE of the algorithm is non-increasing. In this case, it can be said that ICP algorithm always converges monotonically to a local minimum in terms of MSE [15].

Appendix D Minimization of Point-to-Line Metric

Consider the following objective function [13]:

$$J(R,t) = \sum_{i=1}^{N_F}[n_i^T(Rf_i + t - h_i)]^2 \quad \text{(D 4.1)}$$

In this equation n_i is the normal to the line that lies between h_i and z_i where h_i is the closest point and z_i is the second closest point in model point set to the point f_i in data point set. To find the rotation matrix *R* and translation vector *t* that minimizes the objective function in equation D 4.1 an exact closed form solution is introduced by Censi in [13]. By following the steps of [13], the equation D 4.1 can be written as follows:

$$J(R,t) = \sum_{i=1}^{N_F}\|Rf_i + t - h_i\|_{B_i}^2 \quad \text{(D 4.2)}$$

where

$$\|Rf_i + t - h_i\|_{B_i}^2 = (Rf_i + t - h_i)^T B_i (Rf_i + t - h_i) \quad \text{(D 4.3)}$$

In equation D 4.3, B_i is defined as shown below:

$$B_i = w_i n_i n_i^T \quad \text{(D 4.4)}$$

where w_i is defined as weight. The rotation matrix *R* and translation vector *t* are as shown below:

$$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \quad \text{(D 4.5)}$$

$$t = \begin{bmatrix} t_x \\ t_y \end{bmatrix} \quad \text{(D 4.6)}$$

The aim is to find optimal values of θ, t_x and t_y. Then it can be said that the solution will be in the form (t_x, t_y, θ). By using the below relation between $\sin\theta$ and $\cos\theta$:

$$(\sin\theta)^2 + (\cos\theta)^2 = 1 \quad \text{(D 4.7)}$$

The solution can be expressed as follows:

$$a = (a_1, a_2, a_3, a_4) = (t_x, t_y, \cos\theta, \sin\theta) \quad \text{(D 4.8)}$$

$$\text{subject to} \quad a_3^2 + a_4^2 = 1 \quad \text{(D 4.9)}$$

Express the *x* and *y* coordinates of data point f_i as shown below:

$$f_i = [f_{x_i} \quad f_{y_i}] \quad \text{(D 4.10)}$$

Now constitute the matrix P_i as shown below:

$$P_i = \begin{bmatrix} 1 & 0 & f_{x_i} & -f_{y_i} \\ 0 & 1 & f_{y_i} & f_{x_i} \end{bmatrix} \quad \text{(D 4.11)}$$

Then the objective function in equation D 4.2 can be written as follows:

$$J(a) = \sum_{i=1}^{N_F} (P_i a - h_i)^T B_i (P_i a - h_i) \quad \text{(D 4.12)}$$

$$J(a) = \sum_{i=1}^{N_F} (a^T P_i^T B_i P_i a + h_i^T B_i h_i - 2h_i^T B_i P_i \text{a}) \quad \text{(D 4.13)}$$

Since it is a minimization problem with respect to *a*, the constant terms can be ignored:

$$J(a) = a^T \left(\sum_{i=1}^{N_F} P_i^T B_i P_i\right) a + \left(\sum_{i=1}^{N_F} -2h_i^T B_i P_i\right) a \quad \text{(D 4.14)}$$

To simplify the equation D 4.14, *N* and *v* are defined as follows:

$$N = \sum_{i=1}^{N_F} P_i^T B_i P_i \quad \text{(D 4.15)}$$

$$v^T = \sum_{i=1}^{N_F} -2h_i^T B_i P_i \quad \text{(D 4.16)}$$

Therefore, the equation D 4.14 can be shown as below:

$$J(a) = a^T N a + v^T a \quad \text{(D 4.17)}$$

Also the constraint defined in equation D 4.9 can be written as below:

$$a^T Q a = 1 \quad \text{(D 4.18)}$$

where

$$Q = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{(D 4.19)}$$

Now the objective function and its constraint are as shown below [13]:

$$J(a) = a^T N a + v^T a \quad \text{(D 4.20)}$$

$$\text{subject to} \quad a^T Q a = 1 \quad \text{(D 4.21)}$$

Since the objective function given above is subjected to an equality constraint, the Lagrange multipliers method can be used. Then the Lagrange function for the above optimization problem is:

$$L(a) = a^T N a + v^T a + \lambda (a^T Q a - 1) \quad \text{(D 4.22)}$$

The equations below show the necessary conditions for optimal solution:

$$\frac{\partial L}{\partial a} = 2a^T N + v^T + 2\lambda a^T Q = 0^T \quad \text{(D 4.23)}$$

$$a^T Q a = 1 \quad \text{(D 4.24)}$$

To combine the equation D 4.23 and equation D 4.24 rewrite the equation D 4.23 as follows:

$$a = -(2N + 2\lambda Q)^{-T} v \tag{D 4.25}$$

Then the equation D 4.25 and equation D 4.24 are combined as shown below:

$$(-v^T(2N + 2\lambda Q)^{-1})Q(\text{-}(2N + 2\lambda Q)^{-T} v) = 1 \tag{D 4.26}$$

$$v^T(2N + 2\lambda Q)^{-1} Q(2N + 2\lambda Q)^{-T} v = 1 \tag{D 4.27}$$

Let

$$C = (2N + 2\lambda Q) \tag{D 4.28}$$

Hence, the equation D 4.27 can be written as follows:

$$v^T C^{-1} Q C^{-T} v = 1 \tag{D 4.29}$$

The *C* matrix can also be written in following form:

$$C = \begin{bmatrix} Z & Y \\ Y^T & G + 2\lambda I \end{bmatrix} \tag{D 4.30}$$

As it can be seen in equation D 4.19, the *Q* matrix is a sparse matrix. For this reason, in equation D 4.29 there is no need to compute the columns of C^{-1} except the last column:

$$C^{-1} = \begin{bmatrix} * & -Z^{-1}YK^{-1} \\ * & K^{-1} \end{bmatrix} \tag{D 4.31}$$

In this equation *K* equals to:

$$K = (G - Y^T Z^{-1} Y + 2\lambda I) \tag{D 4.32}$$

Now, the equation D 4.29 can be written as follows:

$$v^T \begin{bmatrix} Z^{-1}YK^{-1}K^{-T}Y^T Z^{-T} & -Z^{-1}YK^{-1}K^{-T} \\ (symm) & K^{-1}K^{-T} \end{bmatrix} v = 1 \tag{D 4.33}$$

In equation D 4.32 write *K* as follows:

$$K = M + 2\lambda I \tag{D 4.34}$$

where

$$M = G - Y^T Z^{-1} Y \tag{D 4.35}$$

Then K^{-1} in equation D 4.33 can be shown as below:

$$K^{-1} = (M + 2\lambda I)^{-1} = \frac{\det(M) M^{-1} + 2\lambda I}{\det(M + 2\lambda I)} \tag{D 4.36}$$

In equation D 4.36, the denominator of K^{-1} is a polynomial. Let $q(\lambda)$ denote that polynomial:

$$q(\lambda) = \det(M + 2\lambda I) \tag{D 4.37}$$

Also denote the $\det(M)\, M^{-1}$ in equation D 4.36 as follows:

$$U = \det(M)\, M^{-1} \tag{D 4.38}$$

Now $K^{-1}K^{-T}$ in equation D 4.33 equals to:

$$K^{-1}K^{-T} = \frac{(U+2\lambda I)(U+2\lambda I)^T}{q(\lambda)^2} \tag{D 4.39}$$

$$K^{-1}K^{-T} = \frac{UU^T+4\lambda U+4\lambda^2 I}{q(\lambda)^2} \tag{D 4.40}$$

By using these relations in equation D 4.33, the following equation is obtained:

$$q(\lambda)^2 = \lambda^2 4v^T \begin{bmatrix} Z^{-1}YY^TZ^{-T} & -Z^{-1}Y \\ (symm) & I \end{bmatrix} v + \lambda 4v^T \begin{bmatrix} Z^{-1}YUY^TZ^{-T} & -Z^{-1}YU \\ (symm) & U \end{bmatrix} v + v^T \begin{bmatrix} Z^{-1}YU^TUY^TZ^{-T} & -Z^{-1}YU^TU \\ (symm) & U^TU \end{bmatrix} v \tag{D 4.41}$$

In equation D 4.41, a closed form solution can be found for λ. Then by putting the λ in equation D 4.25, the a is found [13].

CURRICULUM VITAE

Name Surname **:** Erim Vural

Place and Date of Birth **:** Munich, 1993

E-Mail **:** vuraler@itu.edu.tr

EDUCATION **:**

- **B.Sc.** **:** 2015, Istanbul Technical University, Faculty of Electrical and Electronics Engineering , Control and Automation Engineering Department

REWARDS:

- Graduated as 3rd rank B.Sc. student among Control and Automation Engineering Department, 2015
- Istanbul Technical University Members Association Scholarship, 2015
- Istanbul Technical University High Honor List, 2015