SYCOPHANT WIRELESS SENSOR NETWORKS TRACKED BY SPARSE MOBILE WIRELESS SENSOR NETWORKS WHILE COOPERATIVELY MAPPING AN AREA

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

SEDAT DOĞRU

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

OCTOBER 2012

Approval of the thesis:

SYCOPHANT WIRELESS SENSOR NETWORKS TRACKED BY SPARSE MOBILE WIRELESS SENSOR NETWORKS WHILE COOPERATIVELY MAPPING AN AREA

submitted by **SEDAT DOĞRU** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy in Electrical and Electronics Engineering Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen _____
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. İsmet Erkmen _____
Head of Department, **Electrical and Electronics Engineering**

Prof. Dr. Aydan M. Erkmen _____
Supervisor, **Electrical and Electronics Eng. Dept., METU**

Examining Committee Members:

Prof. Dr. Erol Kocaoğlan
Electrical and Electronics Eng. Dept., METU _____

Prof Dr. Aydan M. Erkmen
Electrical and Electronics Eng. Dept., METU _____

Prof. Dr. Veysel Gazi
Electrical and Electronics Eng. Dept., IKBU _____

Assoc. Prof. Dr. Duygun Erol Barkana
Electrical and Electronics Eng. Dept., Yeditepe University _____

Assist. Prof. Dr. Yiğit Yazıcıoğlu
Mechanical Eng. Dept., METU _____

Date: _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: SEDAT DOĞRU

Signature :

ABSTRACT

SYCOPHANT WIRELESS SENSOR NETWORKS TRACKED BY SPARSE MOBILE WIRELESS SENSOR NETWORKS WHILE COOPERATIVELY MAPPING AN AREA

Doğru, Sedat
Ph.D., Department of Electrical and Electronics Engineering
Supervisor : Prof. Dr. Aydan M. Erkmen

October 2012, 151 pages

In this thesis the novel concept of Sycophant Wireless Sensors (SWS) is introduced. A SWS network is a static ectoparasitic clandestine sensor network mounted incognito on a mobile agent using only the agent's mobility without intervention. SWS networks not only communicate with each other through mobile Wireless Sensor Networks (WSN) but also cooperate with them to form a global hybrid Wireless Sensor Network. Such a hybrid network has its own problems and opportunities, some of which have been studied in this thesis work.

Assuming that direct position measurements are not always feasible tracking performance of the sycophant using range only measurements for various communication intervals is studied. Then this framework was used to create a hybrid 2D map of the environment utilizing the capabilities of the mobile network the sycophant.

In order to show possible applications of a sycophant deployment, the sycophant sensor node was equipped with a laser ranger as its sensor, and it was let to create a 2D map of its environment. This 2D map, which corresponds to a height different than

the follower network, was merged with the 2D map of the mobile network forming a novel rough 3D map.

Then by giving up from the need to properly localize the sycophant even when it is disconnected to the rest of the network, a full 3D map of the environment is obtained by fusing 2D map and tracking capabilities of the mobile network with the 2D vertical scans of the environment by the sycophant.

And finally connectivity problems that arise from the hybrid sensor/actuator network were solved. For this 2 new connectivity maintenance algorithms, one based on the helix structures of the proteins, and the other based on the acute triangulation of the space forming a Gabriel Graph, were introduced. In this new algorithms emphasis has been given to sparseness in order to increase fault tolerance to regional problems. To better asses sparseness a new measure, called Resistance was introduced, as well as another called updistance.

Keywords: SLAM, Connectivity Maintenance, Sycophant Wireless Sensor Networks, Protein, Graph Theory, Resistance, 3D Maps, 3D Maps from 2D maps, Gabriel Graphs, Acute Triangulation, RSSI, Mobile Relay Networks, Leader Follower Networks, Range only tracking

ÖZ

HIBRID DUYARGA AGLARINDA ASALAK SABIT KABLOSUZ DUYARGA ALTAGLARI ILE HAREKETLI DUYARGA ALTAGLARI ARASINDAKI ILETISIME DAYALI VERI DENETIMI VE DUYARGA DAGILIMI

Doğru, Sedat
Doktora, Elektrik ve Elektronik Mühendisliği Bölümü
Tez Yöneticisi : Prof. Dr. Aydan M. Erkmen

Ekim 2012, 151 sayfa

Bu tezde Kablosuz Asalak Duyarga Aglari (KADA) isimli yeni bir kavram tanitilmistir. KADA asalak olarak hareketli bir cisme yerlestirilen, ondan habersiz bir sekilde onun hareket yetenegini kullanan bir kablosuz duyarga agidir. KADAlar sadece birbirleriyle onlari takip eden hareketli duyarga aglari uzerinden haberlesmekle kalmaz ayni zamanda onlarla hibrid bir kablosuz duyarga agi olusturmak icin isbirliginde bulunur. Boyle bir hibrid duyarga aginin problemleri oldugu gibi sundugu yeni imkanlar da vardir. Bu tezde bunlarin bir kismi calisilmistir.

Ortamda direkt konum olcumlerinin her zaman verimli olmadigini varsayarak sadece uzaklik olcumu kullanarak asalak duyarga agi hareketli ag tarafindan farkli haberlesme araliklari icin takip edildi. Devaminda bu altyapi ile hem asalagin hem de hareketli alt agin haritalama yetenekleri birlestirilerek hibrid 2 boyutlu haritalama yapildi.

Asalak altagin olasi kullanimlarini gostermek icin, asalaga duyarga olarak lazer mesafe olcer takildi ve kendi seviyesinde 2 boyutlu bir harita yapmasi saglandi. Yaratilan bu

harita hareketli ag tarafindan baska bir seviyede yaratilan 2 boyutlu harita ile birlestirilerek yeni bir cesit 3 boyutlu harita olusturuldu.

Asalagin konumunu kayboldugu zamanlarda dahi kendi haritasiyle bulma gereksiniminden vazgecerek, takip eden agin olusturdugu iki boyutlu harita ve onun izleme yetenegi asalagin dik ortam taramalari ile birlestirilerek tam bir 3 boyutlu harita elde edildi.

Ve son olarak hibrid agda ortaya cikan baglantililigin korunmasi problemini cozmek icin 2 yeni yontem onerildi. Yontemlerden bir tanesi proteinlerin heliks yapisini kendine ornek alirken, digeri ortami dar acili ucgenlere bolerek Gabriel graf olusturdu. Yeni algoritmalarda agdaki dugumlerin bolgesel problemlere karsi cozum olarak dagitik olarak konuslanmasina ozen gosterildi. Bu cercevede Direnc diye yeni bir olcu onerildi.

Anahtar Kelimeler: Es Zamanli Konumlama ve Haritalama, Baglantililigin Korunmasi, Asalak Duyarga Aglari

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

CHAPTER 1

Introduction

1.1 Motivation

This thesis is mainly motivated by two contradictory application fields, one pertaining to war/military operations and the other to search and rescue. However the proposed architecture will be able to serve both fields.

The first motivation is deployment of a Wireless Sensor Network in a war-zone on friendly troops to collect information on both friendly and enemy lines. The traditional way of collecting information about fronts is usually airplanes or reports from the troops on the front. However such information may not always be accurate, may be blocked by the enemy, or the operation region may not allow aerial view. An alternative way utilizing technology would be deployment of various static or mobile sensor networks. However deployment of mobile sensor nodes could require a large number of nodes for efficient surveillance, monitoring, data harvesting and search tasks. Deployment of static sensors on the other hand is not always possible and deployment of complex nodes in a static manner is too expensive. So as a solution we propose mounting sensor nodes directly on the soldiers, without them knowing, or on machinery, and let those sensors form a network through which information could be related back to a base station. Due to limited sensor power and hostility of the environment such nodes should be supported by some mobile robots that will track them and perform other jobs during an operation. Note that cameras are already deployed on US special force soldiers during operations and live video data is sent to their command center. However we are not aware of any complex backstage, and the

cameras have a satellite connection.

The second motivation is utilization of wireless Sensor Networks in Search and Rescue missions. Disaster, being usually unpredictable do not allow pre-deployment of static WSN, and deployment of static nodes after a disaster is not possible. And though Search and Rescue robots are becoming more and more complex, they are not yet suitable for many disaster areas. However in such places we can utilize some trained animals, like mice utilized with small cameras and/or microphones, which would be able to traverse trough the environment and provide us with information about their environment and state of victims. It is more difficult to find proper mobile sensor nodes, to collect information about a disaster area. Sensors on the hosts could form a wireless network when the mice are close enough, which could be backed up by other mobile robot sensors that are supporting the animals and doing search & rescue in parts allowable to them. For example in a collapsed building some paths are only traversable by small robots, some are traversable only by some small mice. This way we utilize mobility of the mice, hinder their motion as little as possible, and easily gather information about the environment.

1.2 Objectives and Goals

Mobile Sensor networking requires a large number of sensor nodes to be deployed within an environment for efficient surveillance, monitoring, data harvesting and search tasks [33, 34, 46, 72, 16]. Deployment and adaptive coverage are energy devouring processes for each mobile unit of the sensor network. Changing the mobile wireless sensor network into a union of static sensor subnetworks and mobile subnetworks cooperating for efficient deployment and coverage in an energy saving mode, makes this usage of hybrid sensor networks a more tempting architecture. In this thesis, we take the hybrid approach one step further by introducing the novel concept of static sensor subnetworks using only, clandestinely the mobility of an agent they hook parasitically upon. We term such networks, Sycophant Wireless Sensor Networks, in short SWS networks. We find necessary here to clarify the term sycophant by stating the characteristics that the sensor network possesses: a SWS network is a clandestine traveler that passively uses without hostility the mobility of the carrying

agent, to collect, process and communicate data harvested during the agent navigation. But it cannot in any circumstance guide or influence the motion of the carrying agent: this agent performs its navigation for its own tasks without the knowledge of the on-board ”clandestine” SWS network. Depending on the number of carrying agents transporting each SWS subnetworks, various problems of communication between SWS subnets may arise; and difficulties in seamless coverage of a certain area may occur since no guidance can be provided to the carrying agents by SWS subnets. We realize connectivity in communication between SWS nets using mobile units of the mobile sensor networks.

In similar architectures, such as wearable WSN, it is assumed that the sinks, which are the final destinations for the data collected, are within the wireless range so that they are connected [71, 7, 8], or in systems such as on-transporter deployment architectures the static network has an infinite energy source and the wireless sensor network is able to connect to far nodes like satellites [48].

In our work we assume that the sink is far enough from the sycophant WSN so that no direct communication is possible, and the nodes of the sycophant network have scarce energy resources: just enough for periodic data transmission as for static wireless nets. To solve the connectivity problem in this case, we dedicate units of a deployed mobile wireless sensor network to follow the sycophant network and cooperate with it at an acceptable distance. This of course introduces new problems, such as tracking, navigation, path planning, localization and mapping, for which solutions exist individually [69, 10, 44] in the literature. However the hybrid network that includes mobile wireless sensor subnets working cooperatively with SWS subnets has to handle all problems at once. This means that it has to properly attack each problem, applying solutions to each, correctly without generating any interruptions of the communication and of the navigation of mobile units. Since direct line of sight may not always be possible, the mobile network has to be able to locate the sycophant WSN using estimation techniques and be able to process information received from the SWS subnets according to parameters, such as received signal strength indicator, time or angle of arrival. It also has to be able to estimate the trajectory of the sycophant network based on previous knowledge about its motion, which is actually a tracking problem. However, it should be kept in mind that depending on the host

carrying the sycophant WSN, the problem may simplify if the host is a slowly moving one, but may also become more difficult if the host has high maneuvering capabilities. The hybrid network also has to successfully map its own environment which means the fusion of two different mapping abilities with two different navigation capabilities: that of SWS subnets and that of the mobile WSN. This also implies modeling such collaboration of SWS nets and mobile WSN within a ”hybrid” Simultaneous Localization and Mapping (SLAM).

In this hybrid SLAM we have on one hand an autonomous robot that is fully capable of mapping the environment, is able to explore using its range sensors and is even able to make decisions and take actions on how to improve the map by new explorations. However this autonomous robot is constrained by the fact that it has to carry out any self exploration while tracking the sycophant wireless sensor network, without breaking its communication with it. Moreover this mobile robot tracker cannot use its full capability of exploration and mapping while tracking, due to the fact that all these tasks cause high energy consumption. The sycophant sensor network on the other hand navigates completely dependent on the motion of the carrying host and is equipped with restricted sensors so that, the SWS maps the visited portions of the environment in an incomplete constrained manner. The major advantage of this mapping although incomplete is that it is done in a highly energy efficient manner since the motion is done by the hosting agent and the only energy consumption is in data acquisition and transmission. Fusing them into an incomplete map is done on board of the tracker, after reception by the mobile unit. In our work we aim at an energy efficient cooperation of a sparse mobile Wireless Sensor Network and a sycophant Wireless network to map a common area. This task is mainly to complete the incomplete SWS map of a part of a common environment by nodes of a mobile wireless sensor network in a decentralized manner. The mobile nodes also need to execute further exploration for growing the map which is merged.

Though in the previous paragraphs the hybrid SLAM was mentioned for a 2D world, the assumption was that the sycophants and the follower robots map the environment at the same height or the map is height invariant. However this is usually not the case, the sycophants will probably be placed at a different heights compared to teach other and the followers, leading to rough 3D mapping. In this case we will be constructing

3D maps from slices. Such slices will be also used in localizing the sycophants or mobile nodes with respect to each other. The constructed rough 3D maps could also provide information on traversable paths, for example low height obstacles seen by small robots but not traversable by bigger robots could easily be excluded from the paths of the larger robots.

And placing a 3D ranger, or a 2D ranger properly on the sycophant we will be capable of creating a full 3D map of the environment.

Meanwhile, the mobile WSN should also track the SWS using a decentralized strategy. If many nodes of the mobile WSN are going to track the SWS using the same heading strategy, at a passage in the environment that cannot be traversed by all the nodes altogether, some nodes would get stuck in that narrow passage and backtracking by choosing another route could mean loosing the connectivity with the target which is the sycophant network in addition to unnecessary consumption of energy. Another problem worth mentioning, which will make the tracking problem for seamless communication more difficult, is the following: the sycophant network is only able to send periodic signals due to the power scarce nature of its nodes. So localization and tracking should be solved using this sparse information.

1.3 Methodology

1.3.1 Tracking

This thesis starts initially with the single leader single follower tracking case, and the solution consists of estimation of the location of the leader unit and the decision and execution of corresponding motion control for the tracker. In this initial part of the thesis it is assumed that only distance measurements are possible between a sycophant and a mobile robot. Trackability in such a case is highly dependent on the observability of the unit to be tracked. To make position estimation in such a case, a motion control approach that forces the follower to do a high order jerky motion was proposed in order to make position of the sycophant observable.

1.3.2 2D Map Merge

For map merging the posterior probability for single robot SLAM was directly extended to include the sycophant agent. For 2D map merging it is assumed that the sycophant does not have any rangers and so its map consisted of a narrow empty corridor around itself.

1.3.3 3D Maps from sparse horizontal 2D slices

Though 2D laser scanners placed on a mobile robot, and the corresponding sycophant are not capable of performing 3D scans, they scan horizontal cross sections of the 3D world. Utilizing cross sections taken at different levels we can make a rough 3D map of the environment. Concentrating on indoor environments, which usually have orthogonal features like walls, tables, etc. Hough transform of the maps at different levels is utilized to find the rotational differences, and then the projections of the maps on the X and Y axis are used in order to find the displacements along the respective axis.

1.3.4 3D Maps

Placing the 2D laser ranger of a sycophant in such a way that it does not scan in the horizontal axis, we can get non-horizontal slices corresponding to distinct cross sections of the world. If poses of these 2D slices are known, and they are dense enough, a full 3D map could be successfully constructed. These locations are available if a follower robots is able to properly localize the sycophant. Then with time synchronization and data fusion a full 3D map can be constructed, and is constructed in this thesis.

1.3.5 Connectivity Maintenance in Mobile Relay Networks

Though connectivity maintenance for a single sycophant single follower scenario can be achieved by a proper tracking algorithm, more is required for a more crowded

network. For this, 3 different methods were implemented and tested. The first is based on the idea that a robot is connected only to two of its neighbors by virtual springs. Such a system at steady state places the robots on a line/curve. However, this approach lacks in redundancy and robustness to node failures as well as any regional connectivity problems. In order to solve those problems two novel methods are introduced as well as two new performance measures.

The first proposed method is an imitation of the helix structure of proteins, where in addition to having strong connections to two neighbors, amino acid residues form weak connections to two other neighbors. The second method is based on acute triangulation of the plane forming a Gabriel Graph, in which relays are connected using virtual springs. The first performance measure introduced is updistance, and it tells us how far the sycophant can go away from the sink. The second performance measure is resistance, which tells us how sparse the network is. The resistance is calculated on a Gabriel Graph of the network by considering the graph as a resistive network.

1.4 Contribution of this Thesis

In this thesis we propose a new Wireless Sensor Network deployment architecture, called Sycophant Wireless Sensor Network (SWSN). In addition we propose a new tracking and connectivity maintenance algorithm for a single mobile robot single SWSN deployment.

Taking into account the fact a sycophant is probably at a different level compared to a follower, and considering that though continuities are expected to exist between these two or more levels, we propose a different 3D mapping approach, which is creating 3D volumetric maps from 2D maps created at different levels of the same environment.

In this thesis yet another use of the sycophant is shown and proved in hardware, which is creating a full 3D map of the environment by fusing laser data from the sycophant and tracking information from the follower robot.

Two new connectivity maintenance methods for mobile relay networks, one based on imitation of protein structures, and the other based on acute triangulation of the space are proposed. In order to judge the performance of mobile relay networks a measure called updistance, that measure how far the leader can travel, and another measure called resistance, which measures how sparse the network is, are proposed.

1.5 Outline of the Thesis

In this thesis, after carrying an overview of related works in chapter 2, in chapter 3 some theoretical backgrounds on SLAM and Graph theory are given. In chapter 4 range only tracking of the sycophant as well as the hybrid map created by their coordination is given. Later in chapter 5 construction of the novel rough 3D map, as well as the full 3D map of the environment utilizing coordination of the sycophant and the follower robot are given. In chapter 6 two new connectivity algorithms and two new performance measures are introduced. Simulation results are presented and discussed for a crowded network. In the final chapter this work is concluded and some possible future research directions based on the work of this thesis are given.

CHAPTER 2

Literature Survey

In the first section of this chapter we provide literature on on-body sensor deployments and compare them with Sycophant Wireless Sensor Networks in order to show the novelty of this new deployment scheme. The second section discusses range measurement methods used in Wireless Sensor Networks and justifies our choice of Time Difference of Arrival as a distance measurement technique. Afterwards in section three literature on map merging in 2D is reviewed. This review is necessary both for map merging in 2D, and also map merging in 3D since methods in 3D map merging are extensions of 2D methods and also our 3D map is an extrapolation of 2D maps. In the fourth section a briew overview of existing 3D mapping algorithms is given. In the fifth section we provide a detailed review of connectivity maintenance algorithms in leader follower networks as well as a summary of work on consensus, rendezvous, leader following, flocking and formation control.

2.1 Wireless Sensor Networks

Our SWS subnets can be placed incognito on different carrying agents: aerial, underwater or land vehicles as well as a human being. Wearable WSN are now appearing in the literature [7, 8, 76, 51, 41] but contrary to our conceptual approach all are there to get human centered data from the human himself/herself and his/her environment under his/her knowledge.

Volgyesi et al. [71] present a WSN based mobile counter-sniper system consisting of sensor nodes placed on the helmet(s) of soldier(s) and tripods. The sensor node

consists of a microphone array, a custom FPGA based sensor-board for acoustic processing of time of arrival and angle of arrival, and a MICAz [1] mote for inter-node communication. During the test, soldiers wearing the wireless sensor nodes did not move and the system was able to estimate the trajectories of the shots ranging from 50-300m with a 96% success rate using 10 sensor nodes. However the success rate for single sensor nodes was 42%.

Aylward et al. [7, 8] describe a design of custom made high communication rate wearable WSN capturing expressive gestures of dance ensembles in real time. In their design, a sensor node consisted of three orthogonal gyroscopes and accelerometers that were used for capturing load dynamics together with capacitive sensors measuring node to node distances. The sensor nodes can be placed on various parts of the body, like arms or shoes or an accessory worn. Sensors are able to communicate with each other, and as a result extract information about the correlation of gesture patterns of a group of dancers. A group of three persons each having a single sensor node on his right arm were tested and simple correlations for simple arm movements were calculated.

There are also other studies that use wearable wireless sensor networks, but this time to monitor person health, identify fall detection [18], and even to learn context dependent personal preferences [40].

As the most relevant tracking mobile WSN example to our case, Mahlknecht and Madani [48] propose deploying a WSN on containers to track containers used in transportation based on an architecture consisting of internal monitors, container monitors and prime monitors. Internal monitors are sensor nodes monitoring the inside of the containers for temperature, light, humidity, etc. Container monitors are sensor nodes that send data from the internal monitors to prime monitors, and are placed one per container. Prime monitors are sensor nodes having global location awareness and an infinite amount of energy. They have global connectivity and are placed as a single monitor per ship/train.

As our literature survey shows, the vast majority of research works concentrate on networks collecting data related to the agent itself, where the agent is aware of the network it is carrying, and the sink is somehow reachable, either by being in the

vicinity or by letting some of network nodes have a large energy source. Our seminal contribution in this thesis is a SWS network where the agent (carrier) is not aware of the ecto-parasitic SWS network it carries and the SWS network collects information about the environment of the agent independent of the agent task. Moreover in our approaches in this thesis, the sink is assumed to be far away from the sycophant net, and there is a finite source of energy available to the SWS subnet.

2.2 Distance Measurement in Wireless Sensor Networks

In this work, we use range-only measurements to track the sycophant WSN. Received Signal Strength Indicator (RSSI), Time of Arrival (ToA) and Time Difference of Arrival (TDoA) are three methods mentioned in the literature on range measurements for wireless sensor networks [37].

RSSI measurements utilize the fact that signal strength decreases with distance. However, as is mentioned in the literature and also verified by our indoor tests performed using IRIS Wireless Sensor Nodes [1], RSSI is not a reliable range measure.

Time of Arrival techniques assume that the time when a signal has been transmitted is known and use the receive time to find the duration it takes for the signal to arrive, and then the distance is directly calculated. However this approach requires highly synchronized clocks.

Time Difference of Arrival techniques transmit an RF and an ultrasound pulse at the same instant. Taking the fact that an RF pulse reaches almost instantly, a clock is started upon receiving the RF signal and the time it takes for the ultrasound to reach is measured, giving the required range measurement. Very accurate location measurements have been reported in the literature using TDoA [56, 59].

2.3 Map Merging

Map merging in general, assumes that the robots know or can measure their relative poses with respect to each other at some time instants, called rendezvous, during the

mapping process. Then this knowledge is used to find either the transformation or an initial guess for the transformation of the reference frames.

Thrun et al. [68] assume that a leader robot has the global map, and although robots initial poses are not known they start within the map of their leader, and use Monte Carlo localization to localize themselves in this map and join the mapping process. Konolige et al. [39] use a process they call zippering, where common locations between past robot trajectories of robot pairs are found using a sequential algorithm, and then all scans of one map are fused to the other as if all scans were collected by a single robot. Williams et al. [74] let the robots create local maps which are merged into the global one at regular intervals. The transformation between the maps is found by relative robot positions if available, if not it is found by feature matching. Howard [35] uses multi agent SLAM where each robot starts its own SLAM succeeded by their interchanging of measurements. They are first able to measure each others positions and then subsequently extend their particle filter to include states of other robots as well as those of virtual pairs which are assumed to move backward in time, providing the previous measurements. The author assumes that the uncertainty in relative pose measurements is small. Although the method is a direct extension of single robot SLAM, it has some drawbacks: it requires the robots to be always within communication range during mapping; it has latency and requires the robots to move at comparable speeds. Zhou and Roumeliotis [79] use the relative robot pose only as a first estimate of the transformation matrix and use matching landmarks as constraints to further improve the transformation. Leon et al. [45] require very precise inter-robot pose measurements for occupancy grids, then use scan matching to complete the merging process. Andersson and Nygards [6] use a set of synchronized inter-robot measurements at rendezvous points as an additional constraint to the non-linear least squares minimization problem of finding the optimum map and trajectory to find the transformation for landmark based maps.

Other papers in map merging do not require any knowledge of robot poses and make use of various methods that will maximize the match between two maps.

Dedeoglu and Sukhatname [23] use heuristics to select a subset of the available landmarks and then compare properties of same type landmarks to find the matching ones.

Afterwards, they calculate the transformation for each pair of landmarks and choose the one maximizing the number of landmark matches. Birk and Carpin [12] find the transformation that maximizes the match between occupancy grid maps to be an adaptive Gaussian random walk that generates random samples at each iteration, and measure the fitness of these transformations by an image similarity measure. However the method is computationally expensive. Carpin [17] uses sampled cross correlation of Hough spectrum of the occupancy grids to find the rotation component of the transformation, then utilizes cross correlation of the X-, Y- spectra to find the x, y displacements of the transformation. Although the method is brittle when X-, Y- projections of the map are not distinctive, the method is very fast compared to the previous one. Adluru et al. [4] use a virtual robot with local maps and the shapes of real robots as range and odometry sensors respectively. Odometry is found by structure registration.

Our setup is slightly different than the ones in the literature; one of the mobile agents (the SWS) does not have any odometry information and thus, its pose needs to be estimated by the other agents using only range measurements. Therefore our proposed solution needs to comply with this slightly different setup by relying upon a slightly different methodology than those existent in the literature.

In the hybrid SLAM problem we propose, we assume that the major map merging is done on a mobile node of the WSN, receiving environmental sensory data from the sycophant sensor network, SWS, creating the mapping of the SWS from the received data. The mobile tracker then fuses this map with its own and grows it by added exploration when necessary, without interrupting its tracking mission. The critical problem in such an incomplete map merging and its subsequent completion arises from the fact that the exact location of the sycophant sensor network is not known and needs to be estimated at all times through the tracker. Therefore the map alignment process has to take this uncertainty into account by using both the generated map and the range measurements. In this work we extend the posterior probability for multi-agent SLAM to include range measurements as well as the state and map of the SWS.

2.4 3D Mapping

Since this thesis works deals with partial reconstruction of the 3D map of a volumetric environment as well as a full 3D map, we find it useful to give an overview of 3D mapping work in the literature, for completeness.

Directly working with 3D laser data is expensive and therefore some authors have concentrated on improving mapping performance by reduction of data, such as reducing the number of samples used in comparing successive scans. For example, [42] extract 2D virtual scans in different planes from 3D range scans and then compare virtual scans of successive measurements using Iterative Closest Point (ICP) and a local registration algorithm to find the 6DoF transformation matrix that results in hundreds-fold performance increase compared to ICP utilizing all 3D range scans. The work in [13] propose using surfaces instead of point clouds for registration.

Zhang et al. [78] propose a 3D sensory system which uses a digital fringe projection and phase shift interferometry to provide real-time 2D and 3D information about the environment at a speed of 30 frames per second. Then at the mapping phase, the authors make use of landmarks, which are extracted using Scale Invariant Feature Transform (SIFT).

Puente et al. [22] develop a feature based 3D mapping approach for semistructured environments using a laser scanner employing a nodding data acquisition system. Geometrical features are extracted using least squares fitting. For 6D localization the maximum incremental probability algorithm based upon the Extended Kalman Filter (EKF) is used.

Others concentrate on improving the storage of maps like Nagatani et al. [52] where they use 3 robots and make use of Digital Elevation Maps (DEM) [32], which are 2D grids where the highest elevation is stored for each grid cell. Nagatani et. al. in their setup having a 3rd robot not capable of holding a rotating laser scanner, mount its 2D laser in a tilted angle, and so its 3D map is formed by combining successive slices at the same angle.

Welle et al. [73] concentrate on improving memory efficiency of a 3D SLAM based

on Rao-Blackwellized Particle Filter (RBPF), where the 2D concept is transferred to 3D. For this, they store obstacle points, without making use of a grid, in a DeltaOctree. This tree stores deltas with respect to the center of the octree instead of absolute positions, and do further improvements when storing the values, such as storing only 2 bytes instead of 4 whenever possible. They use a 2D laser scanner mounted so that it is rotating in the yaw axis (scanning perpendicular to the yaw axis). They test in gazebo an indoor environment, and a large outdoor area, around a building. In both tests the particle filter closes loops successfully, although using 1000 particles are used in a second case, and the best match particle is displayed.

Dryanovski et al. [27] present a multi-volume occupancy grid representation for 3D maps of micro air vehicles, which consists of a grid of cells, each cell containing a list of positive and negative readings corresponding to the starting and ending heights of obstacles and free space respectively at that particular grid cell.

Marder-Eppstein et al. [49] use a full 3D voxel grid representing the whole 3D map for an indoor autonomous robot, but instead of storing probabilities, they store a ternary state at each voxel. The ternary state has occupied, free and unexplored as values and therefore requires only two bits per voxel, which are packed in an integer array.

Song and Jo [63] use Vanishing Points to rectify the perspective images obtained from a stereo camera system. The vanishing points are found from the line features, like doors, bulletin boards etc, of the image. 3D position is found by triangulation of the corresponding lines between the stereo images.

Ryde and Hu [58] use multi resolution occupied voxel lists to represent the map of the environment. In this representation, only the occupied pixels are stored as a hash-list and the value of the pixel corresponds to the times the pixel has been marked as occupied. The authors also mention extracting 2D level maps from 3D maps for use by other robots.

2.5 Connectivity Maintenance

Connectivity maintenance is critical for search & rescue robotics as well as for Sycophant Wireless Sensor Networks. In search & rescue, the robots may loose connectivity with the base station and even with each other due to cluttered highly irregular environment and thus may not be able to convey information to the rescue base station, or may even be trapped without being able to ask for help. To prevent such situations, and increase collaboration between robots, it is necessary that the robots maintain connectivity with each other, as well as with the base station. In Sycophant WSN deployments, the sycophant nodes usually do not have strong RF transceivers and it is also possible that the agent carrying the sycophant subnet wander in a difficult terrain, for example rubbles of an earthquake disaster area for search & rescue, penetrating collapsed buildings. At that point, the communication with the SWS may be highly likley to be lost. We had then to propose in our research work, a hybrid architecture to supplement the sycophant subnet with mobile robots, acting as relays to maintain connectivity with a base station.

Connectivity related issues have been discussed widely both in Wireless Sensor Networks in general, as well as in Mobile Ad Hoc Networks (MANET) more specifically. In this subsection, we overview works on connectivity maintenance in mobile robot networks, and give detailed descriptions of works on leader-follower networks. Search & rescue operations that include some leading robots or sycophant subnets doing the exploration could be classified as leader-follower networks and therefore connectivity maintenance in such networks requires special attention. Connectivity maintenance in leader-follower networks could also be seen as a mobile relaying problem, due to the fact that non-leading robots act as relays.

Nguyen et al. [53] use a convoying strategy where all the relay robots convoy the leading robot as it advances in the environment until the need for a relay shows up, which is detected as the signal level dropping below a predefined threshold. At this point the last robot in the convoy remains at that point, and the rest of the convoy goes on following the leading robot. As the leading robot moves, RF shortcuts may be detected causing a relocalization of the robots that do not relay anymore, which happens by mobbing the robots through the path of the leading robot till they join the

convoy again. The authors show that this strategy, although consuming more power compared to the case when robots are called from a base station as needed, lets the leading robot reach its destination faster, with fewer pauses.

Dynia et al. [29] propose a strategy where mobile relays are placed equidistant to each other along the line connecting the leader and the base station. The authors assume that the terrain is obstacle free. In this strategy the line is formed by each relay moving to the middle of its neighbors using a ”look, compute, move” strategy. Connectivity is maintained by attaching a partner to each mobile relay and the leader. As soon as the leader moves too far away from its neighbors, its partner moves to the middle point between the leader and it neighbors. This strategy requires 2n robots where n is the number of relays + the leader.

Then Dynia et al. [28] improve their previous strategy reported in [29] by injecting new relays directly between their base station and the leader, removing the need for partner robots. This strategy is called Chase Explorer. In this strategy, in addition to keeping close to the line connecting the leader and the base, a relay tries to maintain a predefined distance to its neighbor and to the leader. Authors also handle obstacles, by taking the relay at the obstacle point as a base station for relaying between that relay and the leader.

Chen et al. [19] use a potential field approach consisting of repulsive and propulsive parts where received signal strength is used as the distance measure between the robots for the placement the relays. They also handle obstacles by a virtual potential field between the relays and the obstacles.

Correl et al. [20] propose a probabilistic approach that does not need robot localization and assumes that the relays are able to estimate the number of their neighbors and can avoid obstacles in a bounded environment. In their approach, a relay node makes Brownian motion if either the robot is not connected to the gateway or its neighbors are too much or too few. So the robots mobbing randomly and avoiding each other in a bounded environment are able to uniformly cover the environment.

Dixon and Frew [26] propose using the signal to noise ratio of the communication links instead of the distance between the relays and find the optimum location for the

relays by moving the robots, autonomous aircrafts, on a globally stable limit cycle produced by a Lyapunov Guidance Vector Field.

Stump et al. [66] solve mobile relaying in a leader-follower network in indoor environments by trying to maximize the second largest eigenvalue (Fiedler value) of the weighted Laplacian describing the connectivity of the robot network. During this process, they also aim at keeping the hop count between any two robots below a desired value, which is done using a centralized algorithm. The authors utilize work of Kim and Mesbahi [38], where connectivity constraints are expressed in algebraic terms expressed as a function of the adjacency matrix, in order to maintain connectivity despite obstacles and combine both the connectivity and the obstacle constraint in a convex optimization problem solved at each step.

Tekdas and Isler [67] solve the same problem by providing wireless network connectivity to a mobile user in mind. They utilize dynamic programming to generate relay trajectories that establish a continuous communication link between the base station and the user. The running time of the proposed algorithm is exponential, and processing is central.

Authors of [54, 55, 65], propose using relay bricks that are deployed by the leading robot which is actually the only robot in their scenario, instead of using multiple mobile relays. This deployment is done as soon as received signal strength is below a certain threshold. Nguyen et al take the idea of bricks one step further and propose a system called Automatically Deployed Communication Relays (ADCR) system, where the relay bricks are intelligent and are capable of correcting their position and opening their antenna after being dropped from the leading robot. Although such a relaying using exclusively this way of relay deployment is usually infeasible, mixing mobile deployment of static relays with mobile relays provides new opportunities.

Although not directly related to connectivity maintenance in leader-follower networks, perspectives of works on other multi-agent systems such as consensus, rendezvous [21, 25, 36], leader following [77], flocking and formation control [36], also provide good insight for connectivity. Therefore a brief overview of the methods used in the literature towards the aforementioned connectivity maintenance is appropriate. In such systems, decisions and actions are generally based on local observations and

actions of a connected network, therefore research works focus upon connectivity of the whole robot network. Most research on connectivity maintenance use potential fields assuming large values for the potentials at the boundaries. Work in [24] takes into account agent constraints and proposes a solution based on bounded potential functions. [60] takes into account delays in communication when robots determine actions that would prevent loss of connectivity. [31] still uses potential functions in formation control but take also into account visibility problems caused by obstacles. In [38] instead of using potential functions, authors maximize the second largest eigenvalue of the graph Laplacian to maintain connectivity. In [77] connections between nodes are formed and deleted dynamically using an auction algorithm, whereas other studies assume that any two nodes are connected as soon as one node is within the sensing range of another.

Another distinct class of methods uses virtual springs between robot nodes. Virtual springs approach has been previously used for connectivity management in formation control by [5, 30, 47, 57, 61, 70]. Reif and Wang [57] use a spring based graph to reach a predefined structure. They initially adjust each spring's equilibrium length such that the desired structure is the only equilibrium point. Under local disturbances, robot nodes are found to reach the equilibrium state and the graph achieves a final desired structure. Esin et al. [30] use virtual springs between robots to prevent collisions and keep a desired distance between them during shape formation. MacArthur and Crane [47] use a virtual spring damper system between robots in order to maintain a wedge or vee formation when the formation is following a trajectory. Urcola et al. [70] use linear spring-damper to maintain distance between the nodes, and torsional springs to maintain a given angle between them during leader following, allowing a deformable structure which can comply with the environment. Aizawa and Kubota [5] use directional springs to avoid some equilibrium points during leader following in a triangular formation. Shucker and Bennett [61] use a mesh made of spring edges between selected vertices and the robots are allowed to move under the influence of forces due to these springs that provide damping for motion control during target tracking. However forces are allowed only if the nodes are close enough and they form one side of an acute triangle. In other words, a robot A is allowed to make connections with another robot B, if and only if the triangle formed by ACB where C

is any other robot neighbor, is acute. Later on, Shucker and Bennett [62] extend the approach to track point or diffuse targets where the equilibrium length of the virtual springs connecting nodes that are close to the target is decreased, pulling the mesh towards the target. Such tracking problems are similar to our mobile relaying problem and as will be explained in chapter connectivity, one of our methods will be the application of this approach to connectivity maintenance.

CHAPTER 3

Theoretical Background

3.1 Graph Theory

In this thesis work, graph structures such as Laplacian matrix, algebraic connectivity and Gabriel Graph have been used as a backbone to one of our proposed connectivity maintenance algorithms. These concepts have also been used to develop an evaluation criterion to compare performance. Therefore for the sake of completeness a small introduction to Graph Theory is given in this section. The definitions have been mainly compiled from [9] and [14].

A graph G is an ordered pair $(V(G), E(G))$ consisting of a set V(G) of vertices and a set E(G) of edges consisting of distinct unordered pairs of vertices.

Order and size of a graph are defined as the number of vertices and edges respectively.

A vertex could stand for a robot, a city, a point whereas an edge could stand for a communication link between two robots, a road between two cities, a line between two points.

In this thesis vertices correspond to robots, and edges correspond to either communication links connecting neighboring robots or forces applied by neighboring robots to each other.

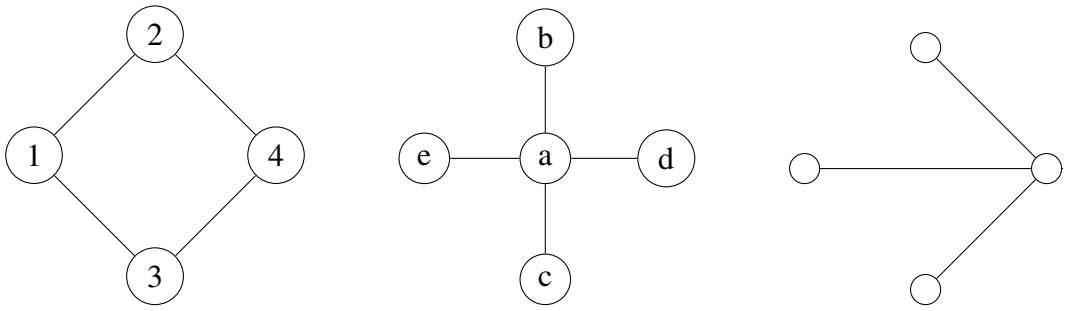


Figure 3.1: Example graphs, the first two have labeled vertices, whereas the third has its vertices not labeled

3.1.1 Laplacian Matrix

Definition 3.1.1 *Let G be a graph with $V(G) = \{1, ..., n\}$ and $E(G) = \{e_1, ..., e_m\}$. The* Laplacian Matrix *of G, denoted by $L(G)$, is defined as the nxn matrix whose columns and rows are indexed by $V(G)$ and (i, j)th entries are defined as -1 if vertices i and j are adjacent, 0 if not. And the (i, i)th entry is defined as the degree of the corresponding vertex, namely vertex i.*

Let $D(G)$ be the diagonal matrix of vertex degrees, and $A(G)$ be the adjacency matrix of G, then obviously $L(G) = D(G) - A(G)$

3.1.1.1 Adjacency Matrix

Definition 3.1.2 *Let G be a graph with $V(G) = \{1, ..., n\}$ and $E(G) = \{e_1, ..., e_m\}$. The* Adjacency Matrix *of G, denoted by $A(G)$, is defined as the nxn matrix whose columns and rows are indexed by $V(G)$ and (i, j)th entry is 1 if vertices i and j are adjacent, 0 if not. (i, i)th entry is 0.*

Definition 3.1.3 *If vertices i and j are endpoints of an edge in a graph, then they are said to be* adjacent *to each other. In other words they are neighbors of each other. Adjacency of two vertices i and j is usually denoted by $i \sim j$.*

Definition 3.1.4 *A graph G is called* disconnected *iff its vertex set $V(G)$ can be partitioned into two disjoint sets V_1 and V_2 in such a way that there is no edge in E with*

one vertex from V_1 and another from V_2. A graph that is not disconnected is called connected.

Definition 3.1.5 *The degree $d(i)$ of a vertex i is defined as the number of vertices its adjacent to, or equivalently the number of edges in which the vertex i is an endpoint.*

Definition 3.1.6 *Let G be a graph with the Laplacian Matrix $L(G)$ whose sorted eigenvalues are $0 = \lambda_1 \leq \lambda_2 \leq ... \leq \lambda_n$. The second smallest eigenvalue λ_2 is called the* algebraic connectivity *of G, and is denoted by $\mu(G)$, or simply μ. The term algebraic connectivity was introduced by Fiedler [].*

A value of 0 for μ means that the graph is disconnected, whereas a value of n means the graph is fully connected, in other words there is an edge from every vertex to every other.

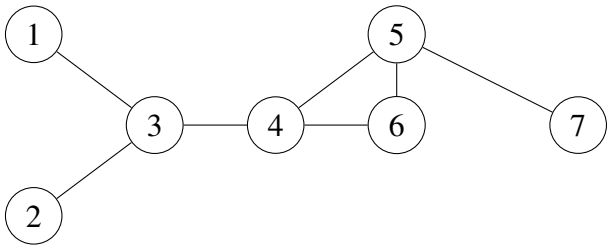


Figure 3.2: A sample graph

Below the adjacency matrix of graph of figure 3.2 is given as an example.

$$A(G) = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

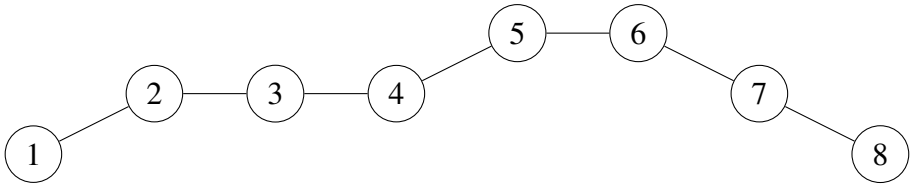


Figure 3.3: A sample graph

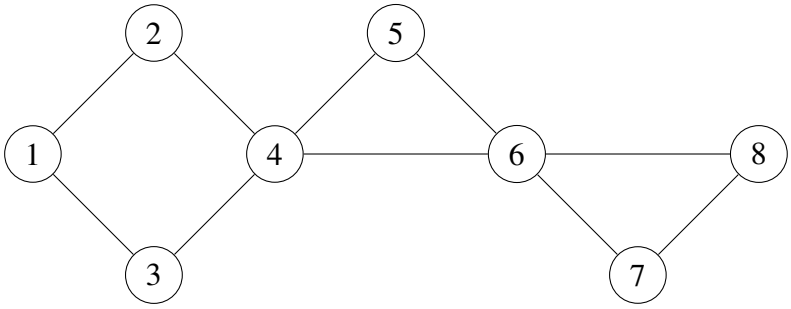


Figure 3.4: A sample graph

Below the Laplacian matrix of graph of figure 3.4 is given as an example.

$$L(G) = \begin{bmatrix} 2 & -1 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 2 & 0 & -1 & 0 & 0 & 0 & 0 \\ -1 & 0 & 2 & -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & -1 & 4 & -1 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 & 2 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 & -1 & 4 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & -1 & 2 \end{bmatrix}$$

Eigenvalues of the graph given in figure 3.3 are $[0.00, 0.15, 0.58, 1.23, 2.00, 2.76, 3.41, 3.85]$ and its algebraic connectivity is $\mu = 0.15$

Eigenvalues of the graph given in figure 3.4 are $[0.00, 0.46, 1.75, 2.00, 3.00, 3.00, 4.24, 5.54]$ and its algebraic connectivity is $\mu = 0.46$

Comparing the algebraic connectivities of the graphs given in figures 3.3 and 3.4, it is seen that the first graph has a smaller algebraic connectivity meaning its connectivity is easier to be lost. Removing any single edge from the first graph makes it disconnected. However the same cannot be said for the second graph. In order to make it disconnected, at least two edges have to be removed.

3.1.1.2 Gabriel Graph

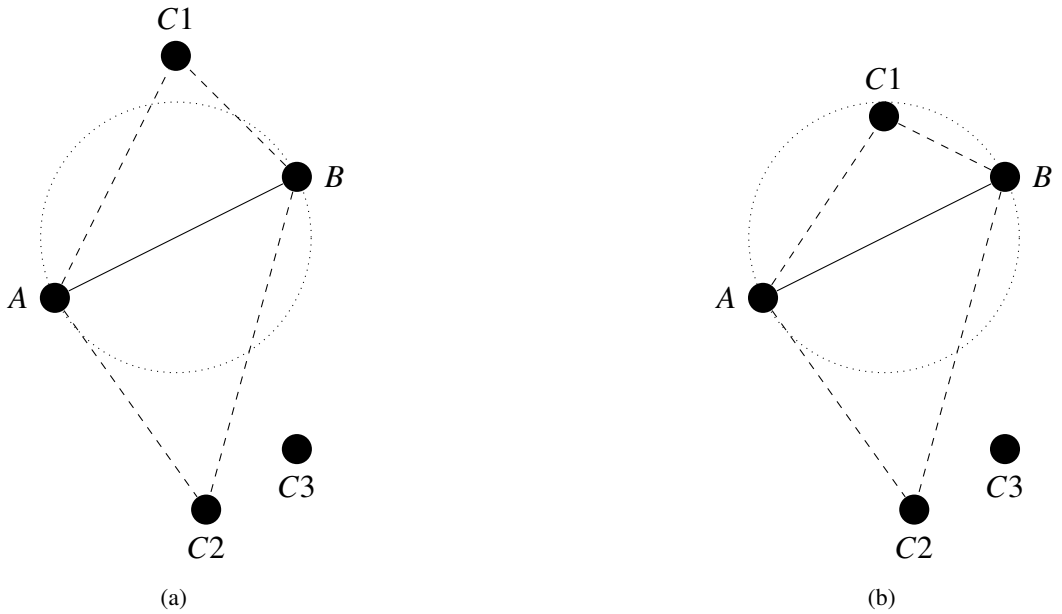


Figure 3.5: The line AB is can be part of an acute triangle since there is no node in the circle with diameter AB. Note that any triangle AB is part of, is acute. However the same could not be said for the second graph. There is a node $C1$ in the circle with diameter AB and the resulting triangle is not acute.

A Gabriel Graph $G(V, E)$, is the graph in which a pair of nodes constitute an edge of that graph if and only if the circle whose diameter is that pair does not encircle any vertex of G. In figure 3.5(b) vertex $C1$ is in the circle whose diameter is the line AB, therefore AB is not an edge of a Gabriel Graph. However in figure 3.5(a), the circle does not contain any vertex and therefore AB is an edge of a Gabriel Graph. Construction of a Gabriel Graph and acute triangulation of the space, where edges on a vertex set are created in such a way that only acute triangles are formed, are the same. An easy way of acute triangulation of the space is the circle test (figure 3.5).

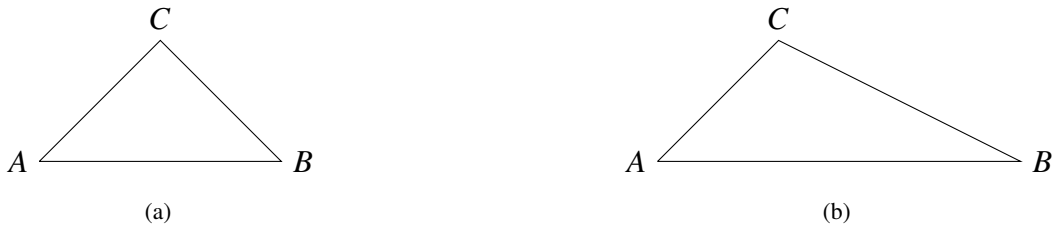


Figure 3.6: (a) An acute triangle's all three angles are less than 90°, (b) whereas a non-acute triangle has one of its angles larger than 90°, note that $\widehat{ACB} > 180°$

CHAPTER 4

Tracking and 2D Map Merging

As an initial part of this thesis work, a network consisting of a single follower and a single leader agent carrying four sycophant sensors is developed and tested in the simulator. Tracking performance of the system as well as its 2D hybrid mapping capabilities are analyzed and presented in this chapter. For a more realistic simulation and realizable hardware implementation, one of the available distance measurement methods, namely the Received Signal Strength Indicator (RSSI), is tested for efficiency indoors and outdoors using wireless Sensor Nodes. However, measurements showed that RSSI is not a good measure of distance in indoors and therefore it is dropped from our approach in favor of the Time Difference of Arrival (TDoA) method.

In the next section, RSSI measurements taken indoors and outdoors are presented and discussed. Then in the second section the hardware framework for this scenario is given, followed by methodology and experimental results for tracking as wells as 2D hybrid mapping.

4.1 RSSI for Distance Estimate

Received Signal Strength Indicator (RSSI) is a measure of the strength of a received RF signal. The farther we go away from a point source, the weaker its received RF signal becomes. And in free space they are correlated by

$$\text{RSSI} \propto R_{emitted}\frac{1}{r^{\beta}}$$

where r is the distance between the emitter and the receiver, $R_{emitted}$ is the signal strength of the emitted signal, and β is an environment dependent constant. We use Crossbow sensors which are able to report the received signal strength in units of dBm and therefore it was decided to take measurements in different indoor environments to see if there is any relationship indoors between RSSI and the distance between the receiver and emitter.

In order to reduce experimental errors in any given indoor/outdoor environment, 5 to 8 RSSI measurements were performed at each distance between the receiver and the emitter. Then mean of the collected data as well as its standard deviation and the corresponding number of measurements for the respective run are given in respective figures.

Data was first collected in a corridor of METU EE's A building, placing the receiver on one of the the centerline and moving the emitter along the center line away from the receiver. The centerline is a virtual line passing through the middle of the corridor and extending from one end to the other. As the plot in figure 4.1 shows, in long corridor like environments, the corridor behaves like a waveguide, and results in a somehow constant signal strength beyond approximately 5 meters. In other words, RSS does not give sufficient information about the distance after 5 meters.

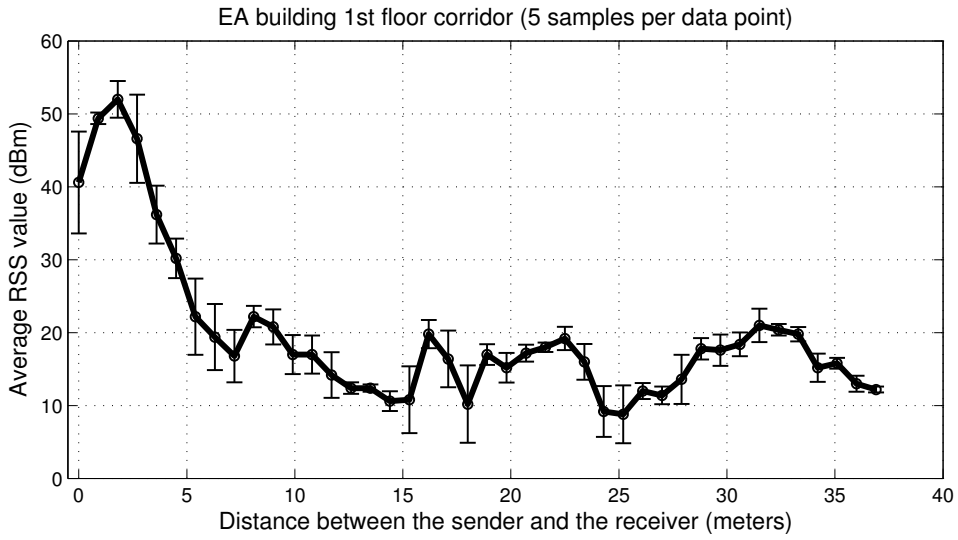


Figure 4.1: RSS measurements taken in the 1st floor corridor of METU EE's A building

The second experiment was done in the alley in front of METU EE's A block. As fig-

ure 4.2 shows, although the resolution is low, RSS provides distance information for a longer range than for the case of the indoor corridor like environment. It is possible to be informed about distance based on RSS values in this outdoor experiment up to 16 m, after which signal strength drops considerably.

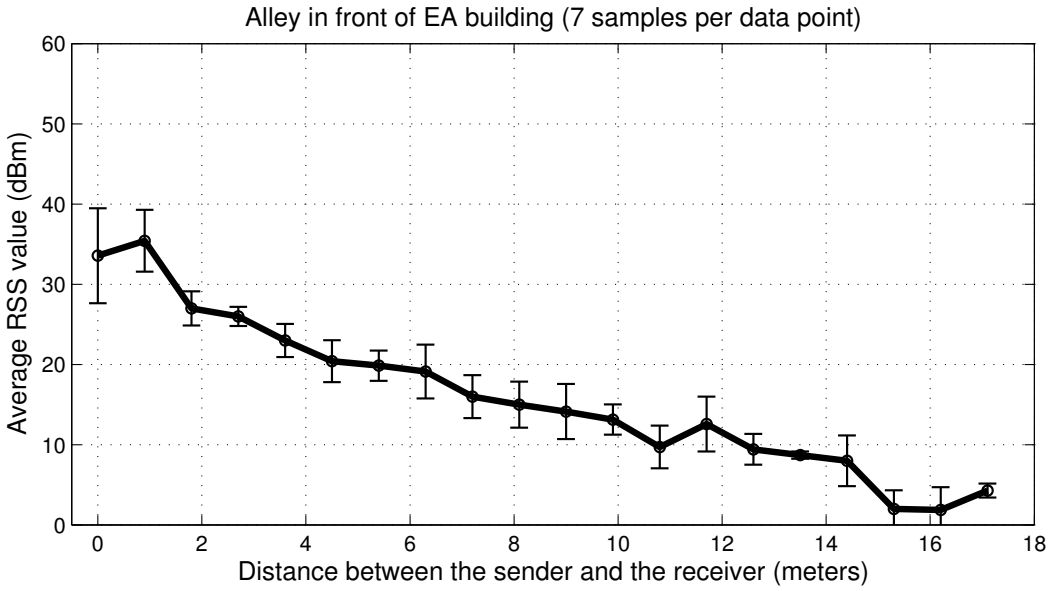


Figure 4.2: RSS measurements in the alley (only the first 18m between two stairways) in front of METU EE's A block

In subsequent experiments data was collected in three different classrooms, for two different configurations. In the first configuration data was collected when the sender and receiver were in direct line of sight of each other. And in the second configuration, an obstacle (my body) was placed between the sender and receiver, and motion started from the end of the class (the motion was a continuation of the motion in the first configuration). First data was collected in two different places of a classroom (EA206) , one on the centerline of the class (figure 4.4), and the others on a line close to the window (figure 4.3). When an obstacle is placed in between, signal strength is found to drop considerably, and measurements become almost unusable for distance information, especially beyond 5 meters away from the receiver.

Then similar measurements were done in another classrom (EA202) and our computer laboratory and the measurements are presented in figures 4.5 and 4.6 respectively. In EA202, the data was collected from the center line of the classroom whereas in the laboratory the data was collected from a line close to the windows. The conclu-

sion is similar: signal strength drops considerably and the measurements are almost unusable for distance estimattion when an obstacle is placed between the receiver and the emitter, and when there is no obstacle usability though the data is usable for the first few meters, it is not usable afterwards.

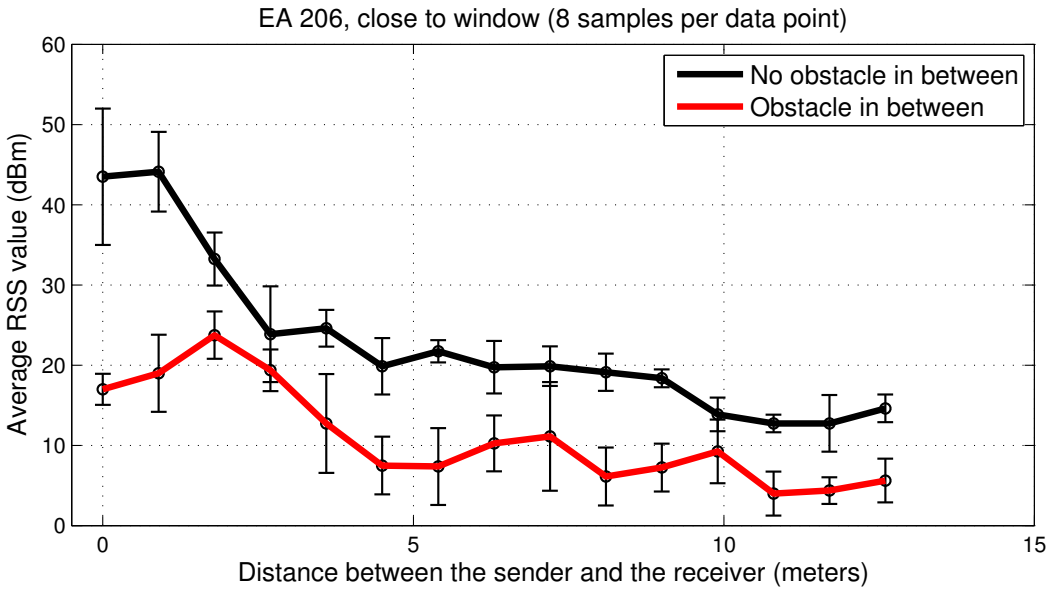


Figure 4.3: RSS measurements on the door side (close to the windows) of class EA 206. The top curve corresponds to the case of no obstacle, and bottom curve to the case of obstacle between the receiver and the sender.

Although RSS is able to provide us with a measure of distance, even in outdoor environments where information is preserved for a longer distance despite reductions of signal strength, the resolution is quite low. In indoor environments on the other hand, RSS revealed to be partially useful, since we got only a very rough approximation of the distance between the receiver and emitter, and the worse case occured for an obstacle in between which did easily upset our estimation of the distance. This made our implementation of RSS to wearable SWS not feasible. And with these measurements we concluded that in order to estimate the distance between an emitter and a receiver in addition to RSS based distance measurements we needed to exploit other methods .

The next section introduces our finalized wearable SWS configuration mounted on a jacket that resulted from the experiments carried in this section.

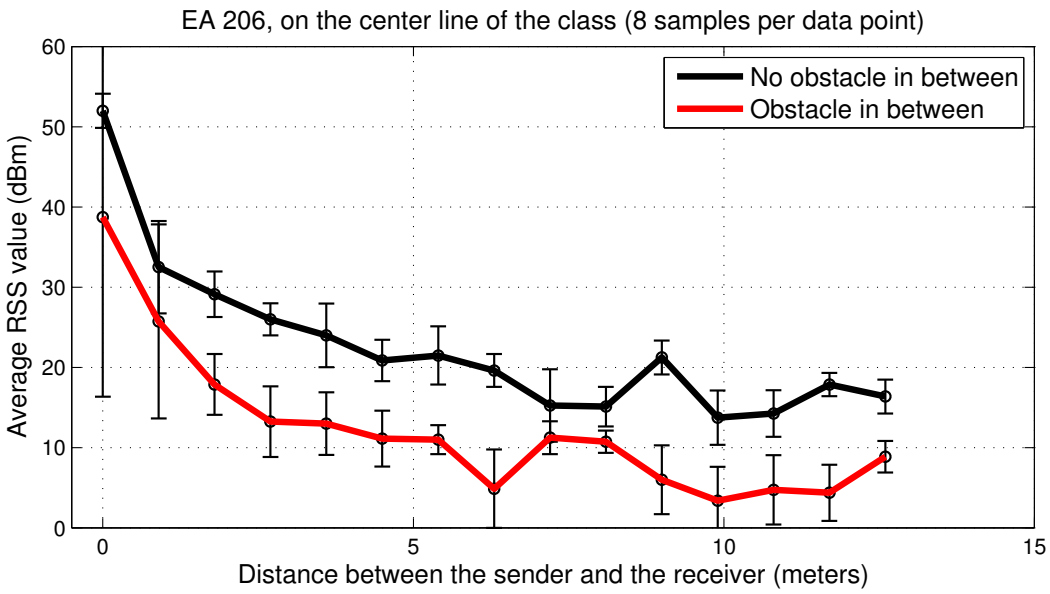


Figure 4.4: RSS measurements on the center line of class EA 206. The top curve corresponds to the case of no obstacle, and bottom curve to the of obstacle between the receiver and the sender.

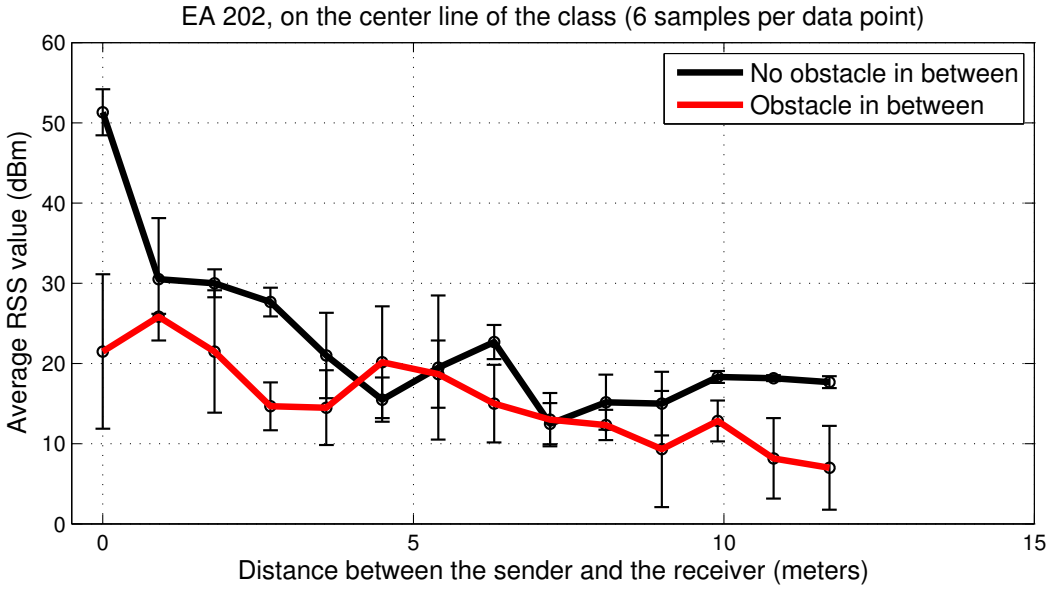


Figure 4.5: RSS measurements on the center line of class EA 202. The top curve corresponds to the case of no obstacle, and bottom curve to the of obstacle between the receiver and the sender.

4.2 Hardware Framework

In this work our sycophant WSN consists of four IRIS nodes [1] mounted on a vest, two on the front two on the back (Fig. 4.8), and a mobile WSN node consisting of

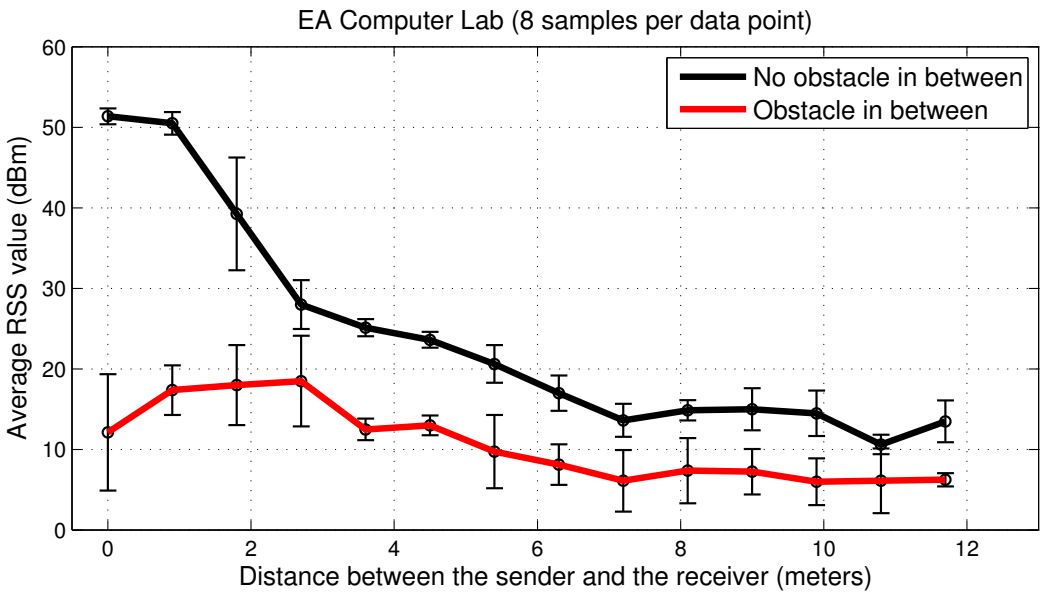


Figure 4.6: RSS measurements on the door side (close to the windows) of the computer lab. The top curve corresponds to the case of no obstacle, and bottom curve to the of obstacle between the receiver and the sender.

Magellan (Fig. 4.9), a differential-drive mobile robot having 16 infrared, and 16 tactile sensors. The IRIS nodes contain temperature, humidity, barometric pressure and ambient light sensors as well as a dual-axis accelerometer. The nodes run TinyOS [3], an open source operating system designed for wireless embedded sensor networks, and are able to run custom programs written in nesC. The wireless communication protocol used is IEEE 802.15.4 and communication occurs at 2.4GHz. To overcome the difficulties of interfacing the mobile node and the sycophant WSN, the mobile robot is connected to an on-top IRIS node through the serial communication channel. The nodes of our Wireless Sensor Network provide SPI or I2C communication protocol interfaces. However our MagellanPro mobile robot provides only an RS232 interface. In order to communicate the robot and the WSN one of the nodes was selected and it was interfaced through an SPI to RS232 converter that was implemented as part of this thesis (figure 4.7). Afterwards a small driver software was implemented to receive and parse the data that comes from the selected sensor node.

On the other hand the simulations are run in Webots [50] simulation environment.

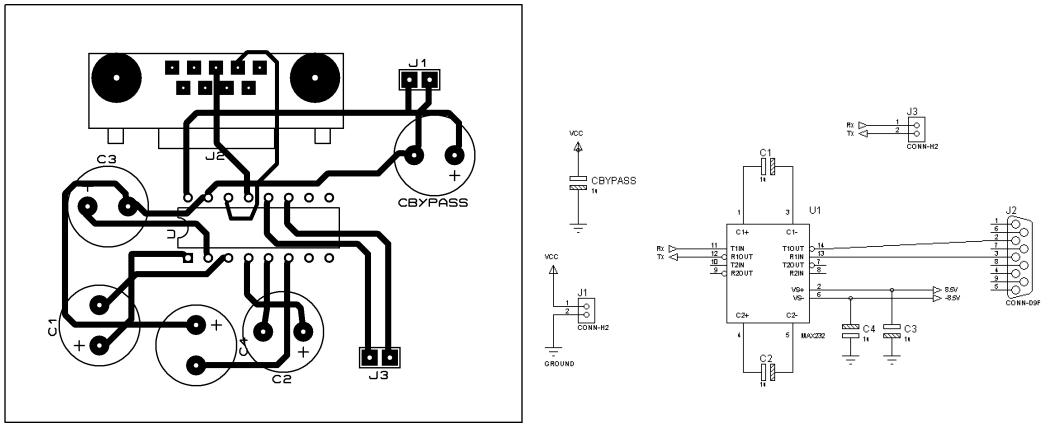


Figure 4.7: An SPI to RS232 convertor used to interface MagellanPro to the Wireless Sensor Network.

Figure 4.8: Front and back view of the vest, which is part of our experimental setup, with sensor network nodes mounted in the pocket areas

(a) (b)

Figure 4.9: (a) Mobile robot Magellan, which is part of our experimental setup, with a wireless sensor network node mounted on the top platform. (b) Lab environment.

4.3 Simulation Framework for 2D Map Merging and Tracking

For 2D map merging and tracking we have used the world given in Fig 4.11. Here both the host carrying the sycophant network and the follower were modeled as MagellanPro robots. The sycophant robot had four transmitters each seeing a different quadrant, and the follower robot had a transmitter seeing all quadrants, imitating deployment of a TDoA hardware (figure 4.10). In figure 4.11 the SWS is represented symbolically as a trio of cicles in a triangular configuration, representative of the SWS nodes of a wearable SWS for example, and the Magellan mobile robot is represented by a single square. Numbered gray rectangles denote obstacles, and the dashed line connecting K and L stands for one of the trajectories that the SWS follows during the simulations. Letters A, B, C, D stand for different positions of the environment and are used below to describe the experimental scenarios.

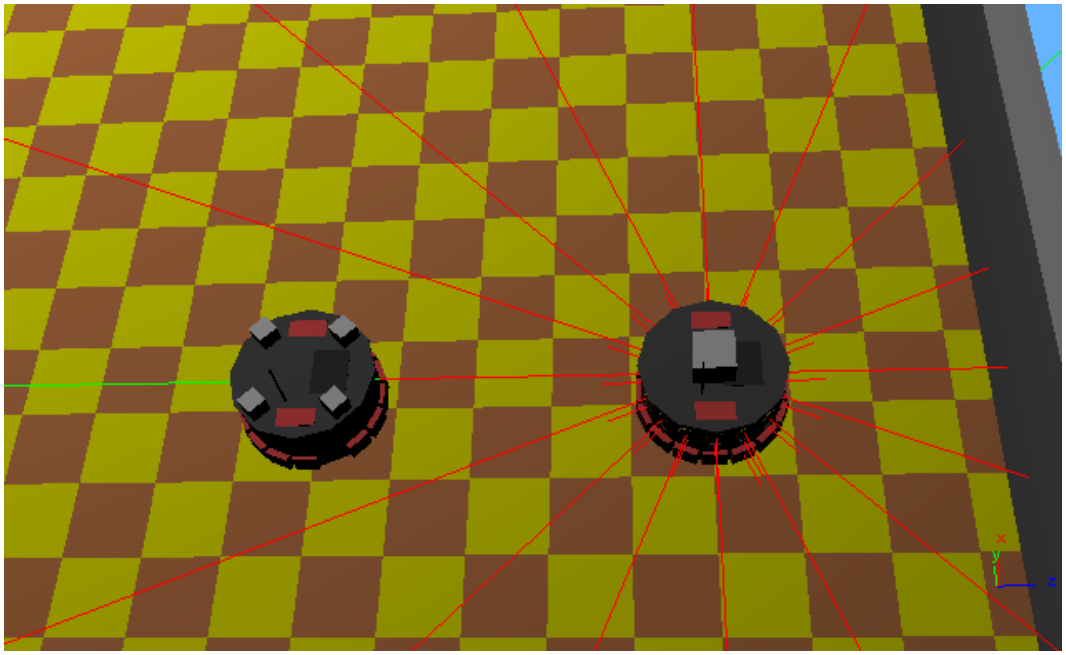
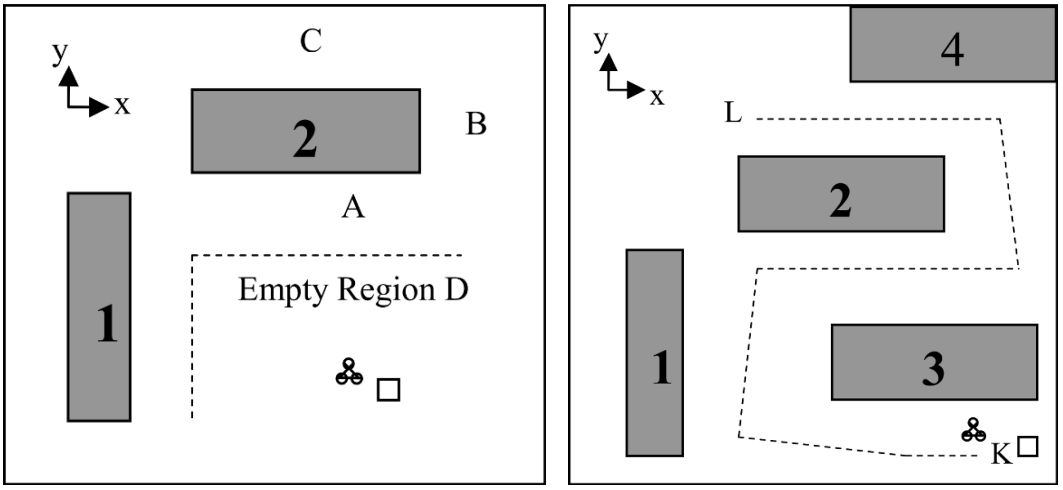


Figure 4.10: The sycophant network mounted on top of a magellan robot imitating the carrying agent (left), and the follower robot (right). The sycophant network consists of four nodes each facing a different quadrant of the environment, whereas the follower robot has a node facing the whole environment.



(a) (b)

Figure 4.11: Worlds used in the simulation

4.4 Methods

In this section, in order to develop the tracking methodology we restrict ourselves to a single mobile node (in our implementation: Magellan) and four SWS nodes mounted on a vest that a person wears. The sensors are placed in different quadrants of the vest, namely front left, front right, back left and back right. This way we make sure that not all ultrasound signals used for Time Difference of Arrival (TDoA) are blocked, and distance measurements in all directions are possible, letting the human move in any directions.

Having a single mobile node and therefore not being able to trilaterate to locate perfectly the SWS subnet we assume that the initial location of the SWS subnet relative to the mobile node is known. Then we use the range measurements obtained using TDoA to detect changes in the location of the SWS subnet.

A small variation in the range measurement would mean that the SWS network is either approaching or receding depending on the gradient of the change. So the mobile node has to either approach or recede. On the other hand an abrupt change in the range measurement, which could also mean completely loosing the signal, would be generated by an obstacle in-between causing high attenuation and the mobile node should then relocate such that the effect of the in-between obstacle is removed and communication with the SWS subnet is re-established.

4.4.1 Building Blocks Of The Problem

In this thesis work, the follower robot is given the task of integrating SWS data with its own data. The tracker therefore takes upon itself most of the computations. The SWS collects data as the carrying agent follows an uncontrollable path. For the sake of efficient discusssions of the results we chose to have a predefined path for the carrying agent. The SWS transmits the collected data to the mobile robot periodically. The block decomposition of the proposed architecture, which runs on the mobile robot, is given in figure 4.12. The algorithms used in each block are explained in the following subsections.

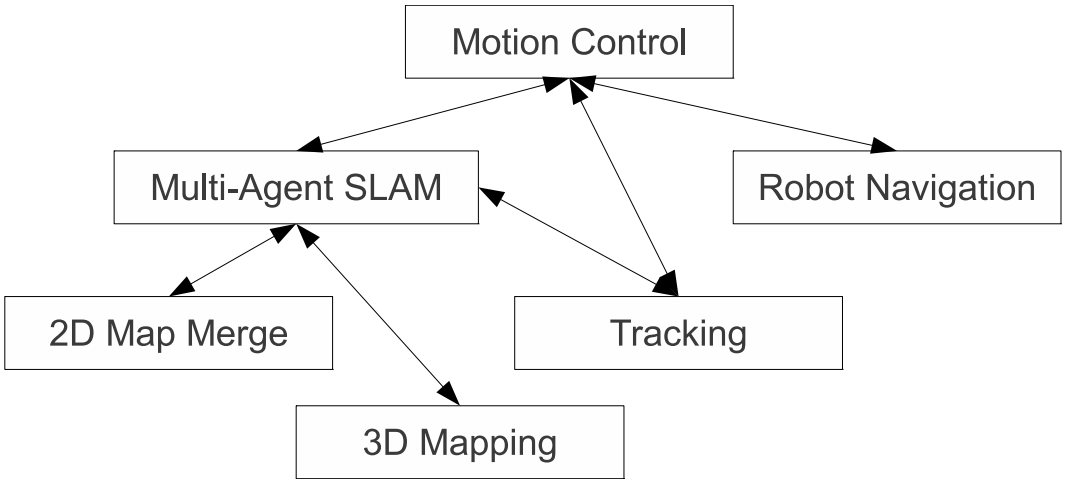


Figure 4.12: Block decomposition of the proposed architecture.

4.4.2 Generating The Multi-Agent SLAM

Having noisy odometry, and being in a crowded environment with obstacles, the mobile agent has to properly localize itself and map the environment. Consequently, it has to be able to plan paths to be taken during navigation and reason about them when modifications are needed. Uncorrected odometry will certainly degrade tracking performance, adversely affecting the on-board mapping of the mobile unit. To improve mapping performance, we also take into account the mapping capability of the SWS subnet during the map building process. This capability is highly constrained but energy efficient. In the context of this thesis the SWS subnet is only deployed on one host (carrying agent), so that the sycophant subnet is assumed to be a single node for

mapping purposes. In order to join the maps created by the sycophant as well as the mobile robot we develop the following approach.

The single robot full SLAM problem is defined as finding the conditional posterior probability $p(x_{1:t}, m|z_{1:t}, u_{1:t})$ where x is the robot state, m is the map, z is the vector of sensor measurements, and u is the input. This posterior probability is decomposed into:

$$\begin{aligned} p(x_{1:t}, m|z_{1:t}, u_{1:t}) = \eta \cdot p(z_t|x_t, m) \cdot p(m|x_{1:t}, z_{1:t-1}) \\ \cdot p(x_t|x_{t-1}, u_{t-1}) \\ \cdot p(x_{1:t-1}, m|z_{1:t-1}, u_{1:t-1}) \end{aligned}$$

where $p(m|x_{1:t}, z_{1:t-1})$ and $p(x_t|x_{t-1}, u_{1:t-1})$ are state updates, and $p(z_t|x_t, m)$ is the measurement update.

The posterior probability to be used when solving a two robot SLAM problem is $p({}^1x_{1:t}, {}^2x_{1:t}, m|{}^1z_{1:t}, {}^2z_{1:t}, {}^1u_{1:t}, {}^2u_{1:t})$. However, in our own scenario, there is another measurement r, which is the distance between the tracker and the SWS. r is measured using TDoA, and is the only measurement between the tracker and the SWS. Taking also into account that the SWS does not have any odometry, the posterior probability for this problem becomes $p({}^1x_{1:t}, {}^2x_{1:t}, m|{}^1z_{1:t}, {}^2z_{1:t}, {}^1u_{1:t}, r_{1:t})$ where ${}^1x$ and ${}^2x$ belong to the mobile robot and the SWS respectively. Decomposing this posterior probability we get:

$$\begin{aligned} p({}^1x_{1:t}, {}^2x_{1:t}, m|{}^1z_{1:t}, {}^2z_{1:t}, {}^1u_{1:t}, r_{1:t}) = \\ \eta \cdot p({}^1z_t|{}^1x_t, m) \cdot p({}^2z_t|{}^2x_t, m) \\ \cdot p(r_t|{}^1x_t, {}^2x_t) \cdot p(m|{}^1x_{1:t}, {}^2x_{1:t}, {}^1z_{1:t-1}, {}^2z_{1:t-1}) \\ \cdot p({}^1x_t|{}^1x_{t-1}, {}^1u_{t-1}) \cdot p({}^2x_t|{}^2x_{t-1}) \\ \cdot p({}^1x_{1:t-1}, {}^2x_{1:t-1}, m|{}^1z_{1:t-1}, {}^2z_{1:t-1}, {}^1u_{1:t-1}, r_{1:t-1}) \end{aligned}$$

In this thesis this expression is implemented by extending the grid based SLAM [69] algorithm to include the new terms; such as the state and measurement updates for the SWS, which are $p({}^2x_t|{}^2x_{t-1})$ and $p(r_t|{}^1x_t, {}^2x_t)$ respectively. Since in this work we do not possess any heading information due to the fact that the sycophant network does not have any node with a ranger/camera, we assume that its mapping consists of marking a rectangular area around its pose as empty.

4.4.3 Tracking

In this work two different types of tracking algorithms are used. The first one, which is utilized to find tracking performance in the first two cases to be described in section on experimental results, is a simple Kalman Filter. The second tracking algorithm is done within the multi-agent SLAM framework described in the previous section. When tracking the SWS we are trying to find the posterior probability $p({}^2x_{1:t}|{}^1x_{1:t}, r_{1:t})$, which can be further decomposed as

$$p({}^2x_{1:t}|{}^1x_{1:t}, r_{1:t}) = \eta \cdot p(r_t|{}^2x_t, {}^1x_t) \cdot p({}^2x_t|{}^2x_{t-1}, {}^1x_t) \cdot p({}^2x_{1:t-1}|{}^1x_{1:t-1}, r_{1:t-1})$$

where $p(r_t|{}^2x_t, {}^1x_t)$ and $p({}^2x_t|{}^2x_{t-1}, {}^1x_t)$ are respectively measurement and state updates of the tracking problem. Since ${}^2x_t$ and ${}^1x_t$ can be taken as independent, the state update becomes $p({}^2x_t|{}^2x_{t-1})$

These state and measurement updates are terms included in the posterior probability generated for the Multi-Agent SLAM. This inclusion is also intuitively expected, because the posterior probability for Multi-Agent SLAM aims to find the robot and SWS trajectories as well as the map, given the sensor measurements and robot controls as well as the relative distance measurement between the robot and the SWS node.

Tracking using range-only measurements is possible only if the corresponding tracking system is fully observable. Full observability for a range-only tracking system is achievable only if the tracker's dynamic equations are at least one degree higher than that of the targets', as is shown by Song [64]. For example, if the target is following a constant velocity trajectory, the tracker has to have an acceleration. In our implementation, the tracker dynamics was considered jerky (3rd order) in order to follow any type of acceleration/deceleration of the human wearing the SWS built-in vest.

4.4.4 Robot Navigation

In this block we use Rapidly-exploring Random Trees (RRT) [43] to generate the path that the robot units will follow using motion control commands. RRT are randomized

data structure designed for a broad class of path planning problems. RRT explore the free space by randomly selecting points and adding them to the tree. The exploration is biased towards unexplored regions of the space and is able to quickly cover the empty space. RRT are able to handle nonholonomic constraints, including dynamics, and high degrees of freedom.

RRT are fast and being a random path generation technique can automatically create jerky paths that satisfy the observability requirement mentioned in the previous subsection.

Though there are many different variations on generating RRT, the following algorithm was used throughout this work. To speed up path generation and avoid problems related to incompatibilities between the ideal RRT control inputs and noisy robot control inputs, dynamic and kinematic parameters were not included in the state. The state had only three components: x and y position as well as the heading of the robot.

The algorithm given in algorithm 1 starts with an empty tree of vertices where each vertex stores the state and a link to its previous state so that by back-tracing through these back-links, one can get all the intermediate states leading to the current one. Then the initial state is added as a vertex to the tree. Afterwards, the following steps are undertaken untill the maximum iteration count is reached or a vertex close enough to the final state is obtained: First a random state is selected from the configuration space. Then a node from the tree that is closest to this random state or the final state is selected depending on the outcome of a random number which is compared to a value, called *GOAL BIAS*. This bias value is used to grow an RRT towards the final state, so that the algorithm converges quicker. After this, a new node that is between the random node and the new node is created. This final node created is added to the tree if it does not fall into an obstacle space. After enough iterations the algorithm outputs the tree which can then be traced to read out the desired path.

A few words about the parameters used in the algorithm are as follows: Selection of the *goal bias* is environment dependent and having a large value is usually not recommended because it can degrade the performance in a crowded environment. Distance of the final node to its closest node depends on *RANDOM_EPS*, and this parameter also affects convergence speed. It is generally taken as a uniform random number

between 0 and an ϵ_{max}. If ϵ_{max} is taken too large, then the steps of the algorithm will be large and the algorithm may never get close enough to the final state. On the other hand if it is too small, one would need many iterations to reach the final state.

Algorithm 1 A Basic RRT Algorithm

1: **procedure** BASIC RRT($x_0, x_f, map_{obstacle}$) ▷ Initial and destination states, map of the environment
2: Tree.add(x_0);
3: $i \leftarrow 0$
4: **repeat**
5: $x_r \leftarrow RANDOM\_STATE()$
6: **if** ($rand \leq GOAL\_BIAS$) **then**
7: $x_c \leftarrow Tree.getClosestTo(x_f)$
8: **else**
9: $x_c \leftarrow Tree.getClosestTo(x_r)$
10: **end if**
11: $x_n \leftarrow x_c + RANDOM\_EPS * (x_r - x_c)$
12: **if** ($x_n \notin map_{obstacle}$) **then**
13: $Tree.add(x_i)$
14: **end if**
15: **until** ($i \geq MAX\_ITERATION\_COUNT$) or ($\|x_n - x_f\| < \epsilon$)
16: **return** Tree
17: **end procedure**

An RRT expanding in free space could be seen in Fig 4.13, and an RRT generated to connect two points in a simulation environment could be seen in Fig 4.14

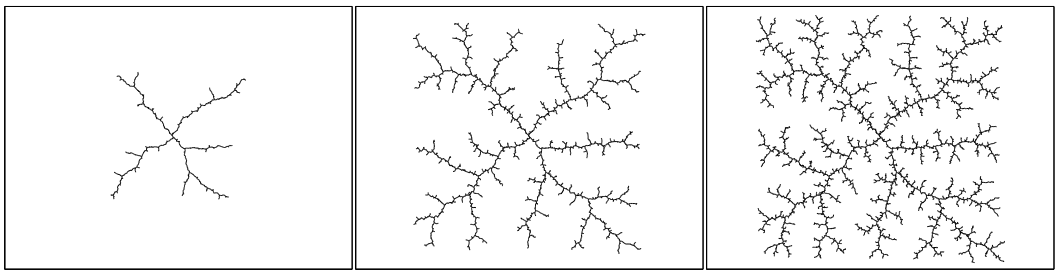


Figure 4.13: A Rapidly-exploring Random Tree expanding in free space.

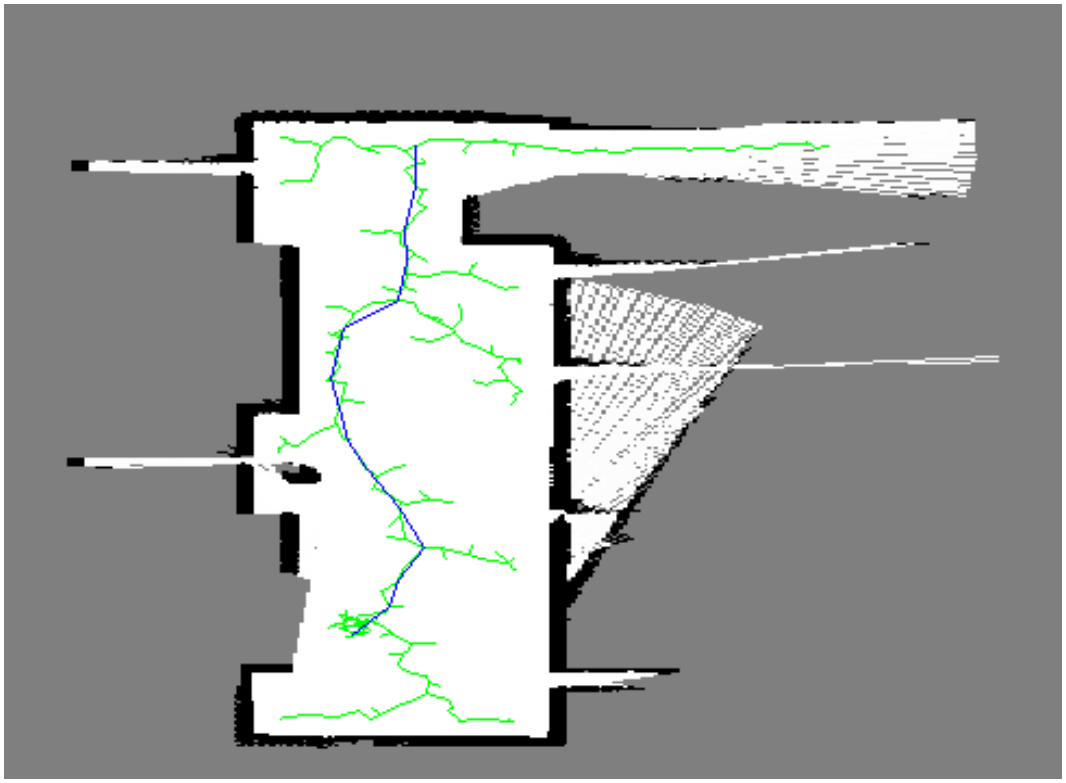


Figure 4.14: An RRT from one of the simulation runs, the green and the blue lines form the RRT, the blue line is the path the robot will follow as it goes down.

4.4.5 Motion Control of the Mobile Robot

This block contains our proposed motion control methodology as the main high level loop in our framework. The methodology that integrates the hybrid multi agent SLAM, RRT based path generation and tracking is best explained using the block diagram of figure 4.15.

Note that, the SWS network sends periodic signals to the tracker which may have a large period due to scarcity of energy, so that getting a distance measurement for the SWS (target) by the tracker at every time step is not possible, thus decreasing the quality of target state estimation.

The solution in obstacle free regions is straightforward: as the signal strength decreases, the mobile robot moves closer to the SWS network. Getting closer is allowed when the distance is above a predefined threshold and this allowance stops when the distance is below another predefined threshold. When the mobile robot is too close, instead of standing still, the mobile robot begins to explore its surrounding without

loosing contact with the SWS. This exploration increases tracking performance as well as mapping performance. In order to avoid high oscillations and prevent quick maneuvers of the follower robot, the interval between thresholds should be kept large enough. In our implementation the lower and higher thresholds were varied for different scenarios between 0.5 to 2 meters and 2 to 4 meters respectively.

When obstacles are present in the medium, the behavior of the tracker becomes a bit different. Assuming that range measurements are interrupted due to an obstacle in the communication path, the mobile robot does not have to go on performing a complex motion (here 1 degree higher than the SWS dynamics) to preserve observability of the system, instead it goes to the just estimated position of the SWS using the shortest path, not to loose the SWS due to missing data that are causing missing measurement updates in the tracker. When communication is reestablished the mobile agent resumes its standard tracking and map building operations.

So the main motion characteristic of the mobile tracking unit is to keep the SWS always within a predefined distance range and be visible to it for uninterrupted communication, to make jerky motions for everlasting observability of the SWS, and to explore its immediate neighborhood when possible.

4.5 Experimental Results and Discussion

In this thesis four different cases were run on the same simulation environment of our mechatronics lab (figure 4.11). The first three cases whose results are presented in the next subsection were run to demonstrate tracking performance, and therefore a high quality mapping with little sensor and odometry noise was assumed.

4.5.1 Tracking Results

In the first case the SWS network and the mobile robot are in an obstacle free portion of the environment (denoted as empty region D in figure 4.11), and there is no adverse effects for their mutual communication. The agent carrying the SWS network moves and stands still for certain time slots. For a pictorial description of the scenario and

the results, the real trajectories of the mobile node and the SWS subnet as well as the SWS trajectory as it is estimated by the mobile tracking robot are given in figure 4.16, and time evolution of these trajectories are given in figure 4.17.

As can be seen in the figures, the tracker follows a random trajectory which is the outcome of the RRT algorithm that guides the robot motion. The path has intentionally not been smoothed in order to provide the jerky motion required to make the SWS system observable. The estimated trajectory of the SWS by the tracker is seen to be close to the original one in form, while the average measurement error is 0.31 meters.

Wireless sensor nodes do not continuously broadcast RF signals, making it necessary to run the experiments using different broadcast intervals. Therefore, this scenario was also run using different measurement update intervals in order to find a correlation between the error and measurement update intervals. The results are presented on table 4.1.

Table 4.1: Average error for different measurement update intervals, where the state update interval was chosen to be 0.064 seconds (first case in text)

Measurement update interval (seconds)	Average Error after 10 runs (meters)	Standard deviation
0.064	0.25	0.05
0.320	0.34	0.09
0.640	0.46	0.09

The first column of table 4.1 gives the measurement update interval in seconds; the second column shows the average error of 10 standard runs in meters. As the table shows by comparing its first and third rows, having a measurement update interval 10 times that of the state update causes larger errors.

In the second case, the SWS network starts from region A and goes through region B to region C. The mobile node follows the SWS network but communication is deliberately and randomly blocked in region B for a distance of approximately 1 meter in the y-axis. The trajectories of the mobile node, the SWS subnet, and its estimated position for a sample run are given on figure 4.18.

Despite the interruption in communication the tracker is able to successfully estimate

SWS network's trajectory. The form of the estimated trajectory is very close to the real one and the average error is 0.22 meters. The average error in many runs of the simulations was found to be 0.22 meters with a very small standard deviation, which is 0.01.

Simulations were also run for different maximum minimum allowed proximity values. For the first case described above, the maximum allowed distance between the SWS and its tracker was increased from 1.5 meters to 3 meters, and then it was observed that the mean error increased upto 0.6 meters. And as communication is broken, the SWS is easily lost. Although following the target at a close distance is the best option in terms of tracking, the allowed distance thresholds certainly depend on the mission for which an efficient proximity value interval is to be assigned.

In the third case, the SWS moves in a more complex environment incorporating more obstacles, following the trajectory KL(figure 4.11). The mobile unit (Magellan) follows it as a tracker. For a pictorial description of the scenario and the results, we provide the x-y trajectories of the mobile node and SWS subnet as well as its estimated position by the tracker for one of the simulation runs in figure 4.19, and the time evolution of each trajectory in figure 4.20. The average errors for various measurement update intervals for this case are given on table 4.2. Comparing the results of the third and first two scenarios, the average error is larger due to the fact that the tracker has to adapt to more maneuvers in the 3rd case compared to the previous ones. The results are still satisfactory: the robot can properly track the SWS using range measurements only.

Table 4.2: Average error for different measurement update intervals, where the state update interval was chosen to be 0.064 seconds (third case in text)

Measurement update interval (seconds)	Average Error after 20 runs (meters)	Standard deviation
0.064	0.33	0.07
0.320	0.51	0.11
0.640	0.68	0.11

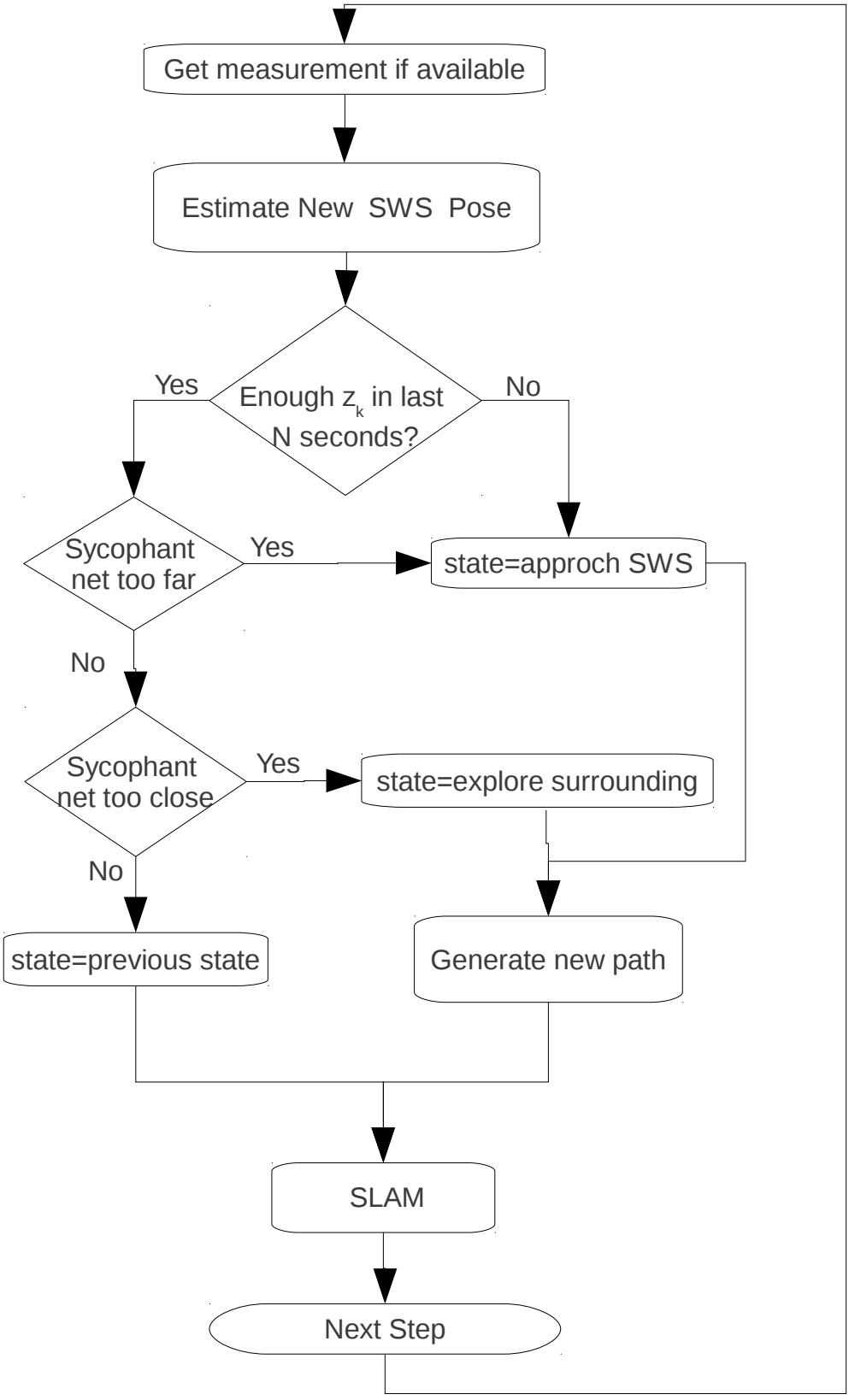


Figure 4.15: Block diagram showing algorithm for Motion Control block. z_k stands for measurements

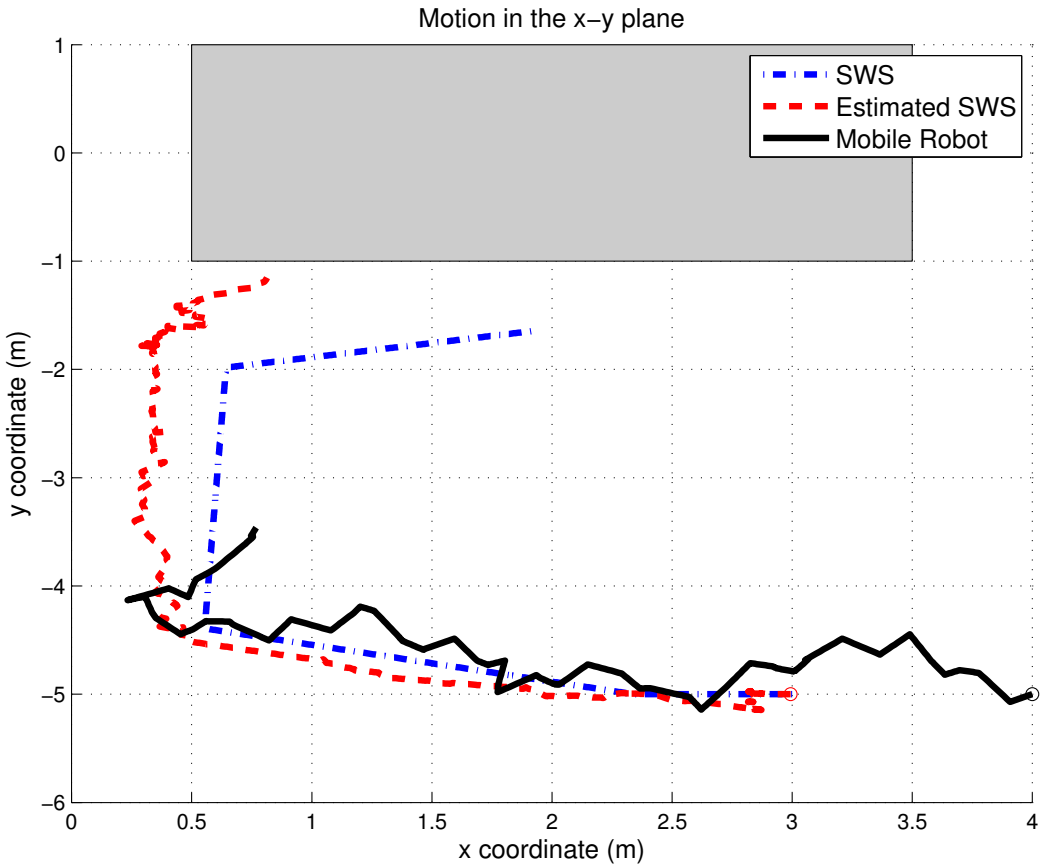


Figure 4.16: Trajectories of the mobile robot, the SWS network and its estimated pose in obstacle free region. Measurement update is executed every 0.64 seconds, whereas the state update is done every 0.064 seconds.

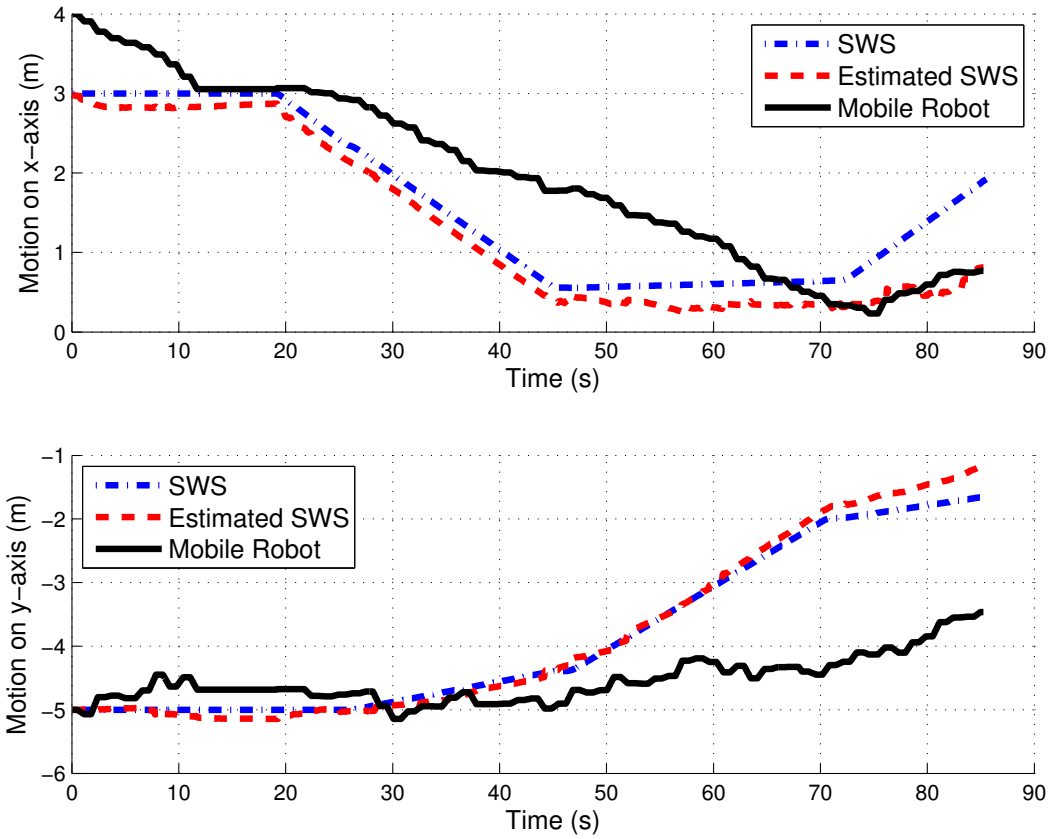


Figure 4.17: Time evolution of the trajectories given in Fig. 4.16.

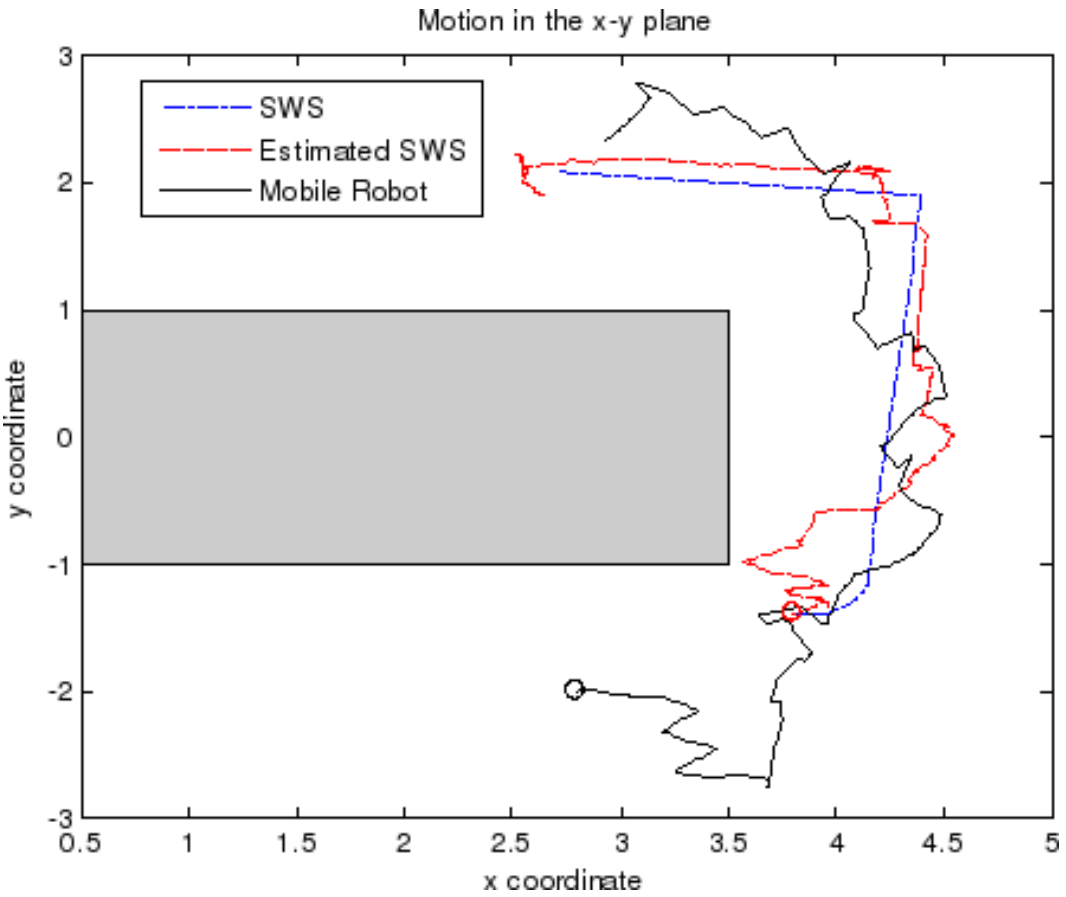


Figure 4.18: Trajectories of the mobile robot, the SWS network and its estimated pose starting from region A through B to C. Time interval between two measurement updates is 0.64 seconds and the state update interval is 0.064 seconds.

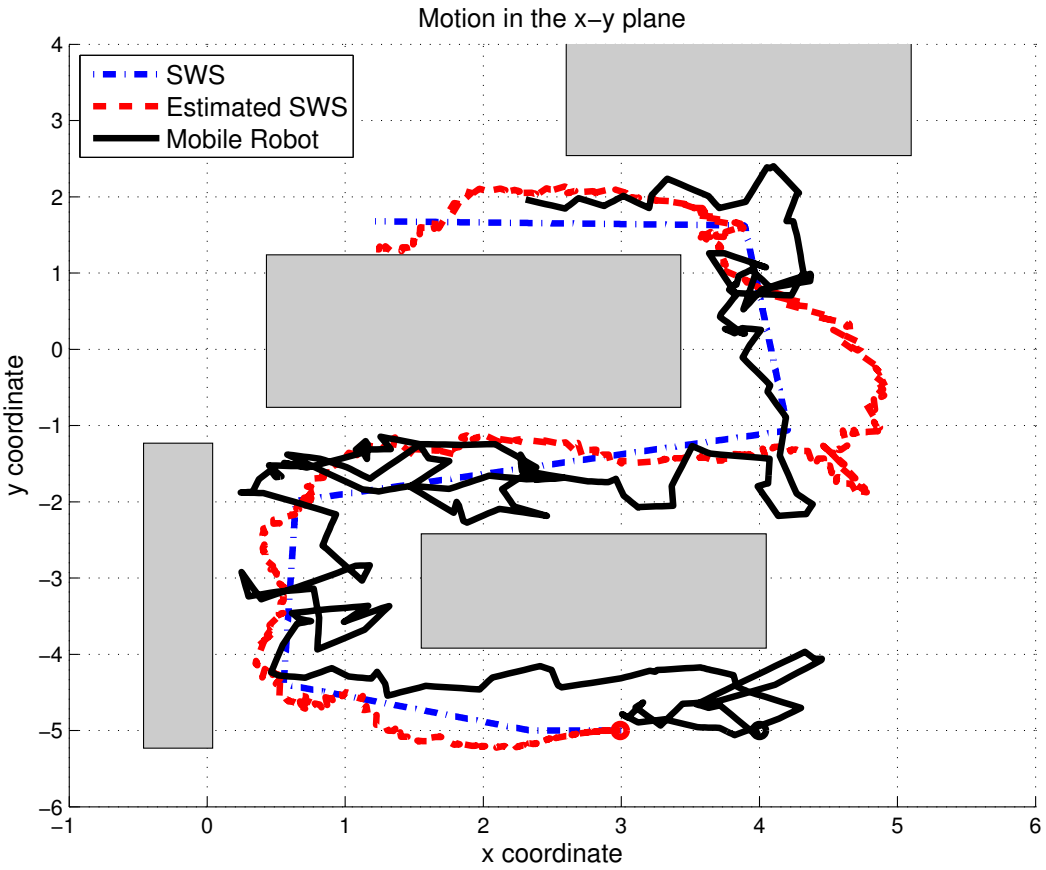


Figure 4.19: Trajectories of the mobile robot, the SWS network and its estimated pose as it follows the trajectory KL. Measurement update is done every 0.64 seconds, and the state update is executed every 0.064 seconds.

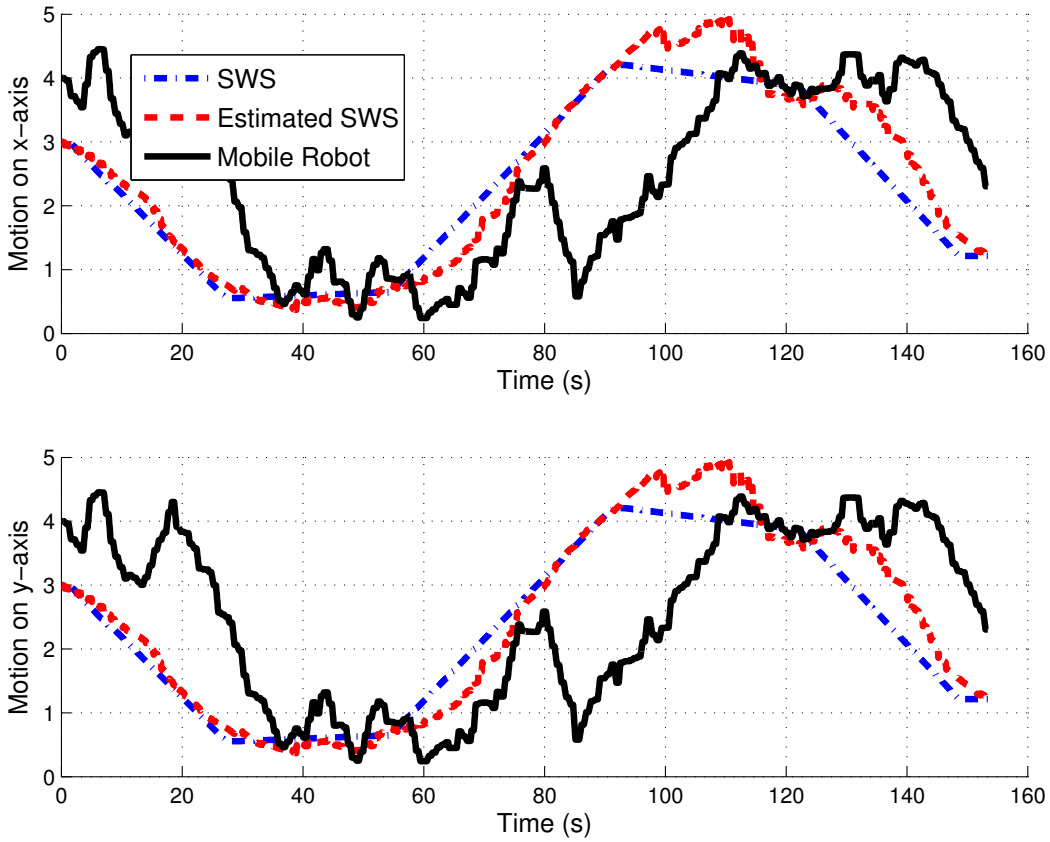


Figure 4.20: Time evolution of the trajectories given in Fig. 4.19.

4.5.2 2D Map Merge Results

In the fourth case, we show how hybrid SLAM (multi agent SLAM) improves mapping using the world in figure 4.11(a) and a minimum allowed proximity of 2 meters between the two agents.

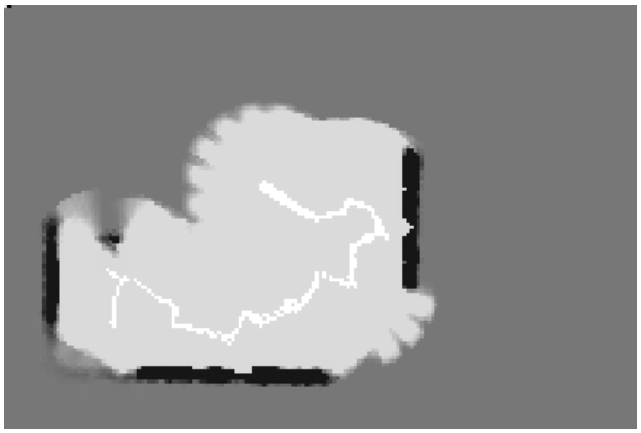
In this section we dwell on extending the primary mission of the mobile robot which is to properly track the SWS that we analyzed in previous sections by improving the generated map of the environment, utilizing mapping capabilities of the SWS too. This way a larger area is covered using less energy, and areas traversable only by the SWS are mapped too. Results in this section are obtained using the multi agent SLAM approach described in section 4.4.2. Note that the map term in the SLAM algorithm described in section 4.4.2 contains the merged 2D map of the environment created by the SWS and the follower robot. However in order to show the advantages of this hybrid mapping approach, non merged separate maps were also created, and they are shown together with the merged map in the figures of this subsection (figures 4.21, 4.22, 4.23 and 4.24).

In figure 4.21 the top left region is not yet explored by the mobile robot, but this region is already traversed and partially mapped by the SWS. This partial map is merged with the mobile robot's map, and as could be seen in figure 4.22, a little later the mobile robot explores the small region left between the SWS and itself.

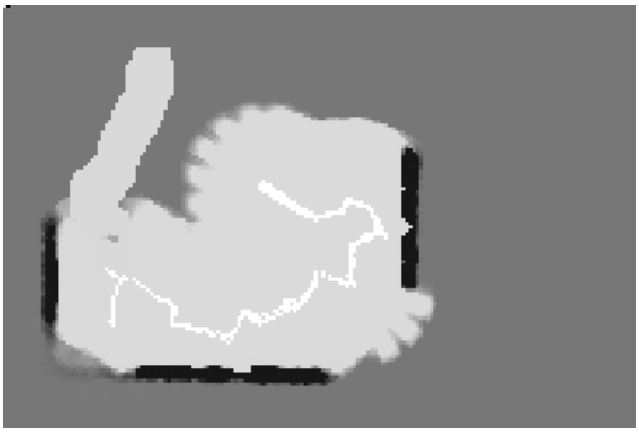
Due to large proximity constraint, the follower robot keeps exploring the empty region as the SWS traverses region B in figure 4.23. A few steps later, the SWS enters region C, in which it is not visible to the follower but the tracking algorithm estimates its pose properly and its map is growed accordingly (figure 4.24). With the knowledge of this extra map patch at hand, the mobile robot can easily generate a path that will lead it to the SWS. This obstacle free extra patch easies path pllaning by telling possible directions of growth for RRT during path planning.

Results presented in this section are preliminary. A large hybrid WSN with many mobile robots and many SWS networks will make advantages of hybrid mapping more obvious.

(a)



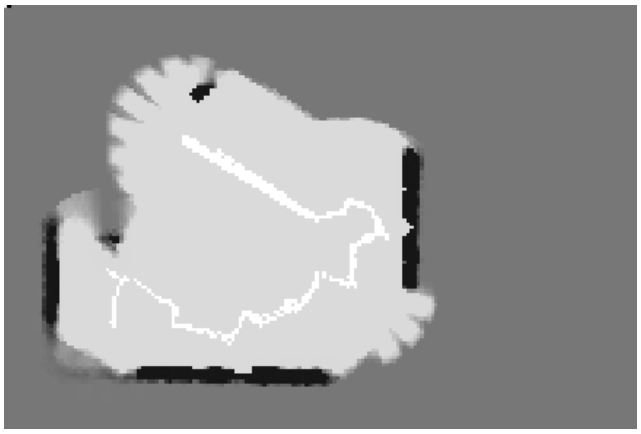
(b)



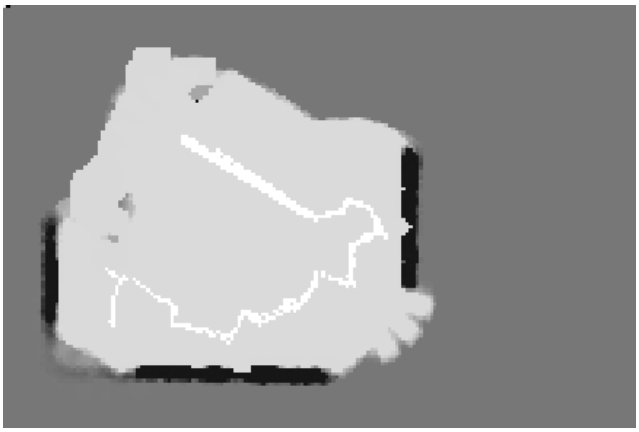
(c)

Figure 4.21: Starting from top, map of the SWS, map of the robot and both maps merged. Both the sycophant and the follower robot are in the empty region. The sycophant has mapped the upper left corner whereas the bottom parts are mapped by the follower robot.

(a)

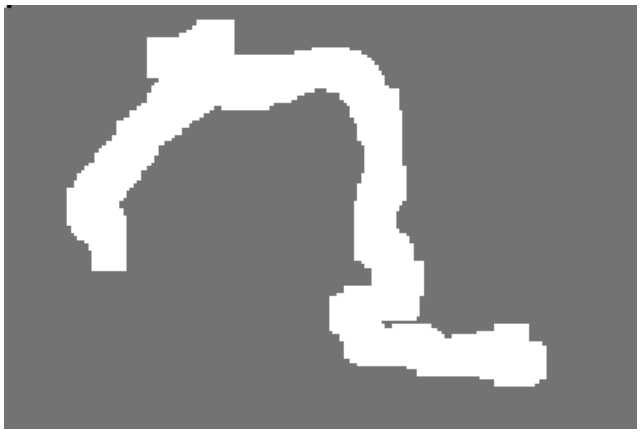


(b)

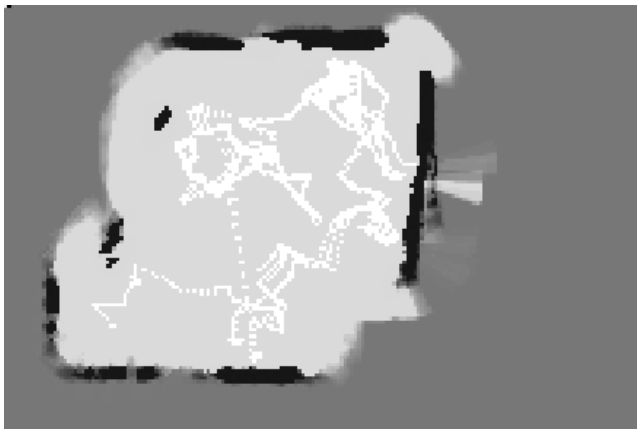


(c)

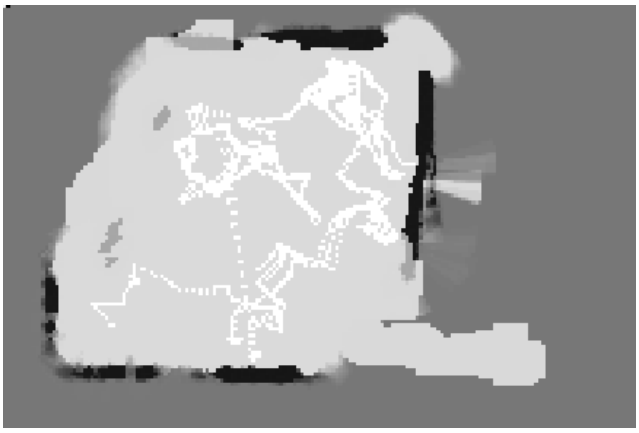
Figure 4.22: Starting from top, map of the SWS, map of the robot and both maps merged. Both the sycophant and the follower robot are still in the empty region. The sycophant has started going to the right, mapping a little more. And in the meantime the follower robots move a little closer to the SWS, completing map of the missing patch (compare to figure 4.21)



(a)

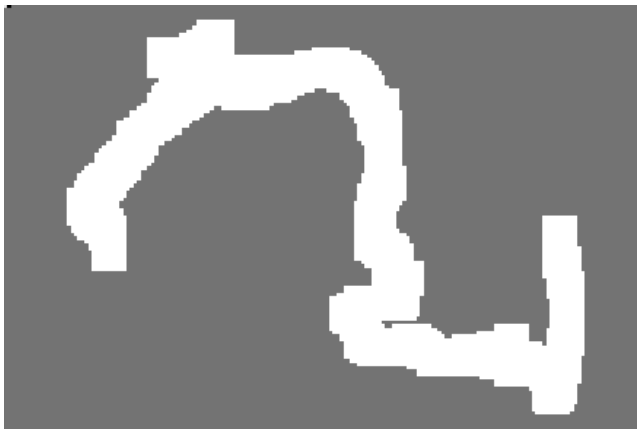


(b)

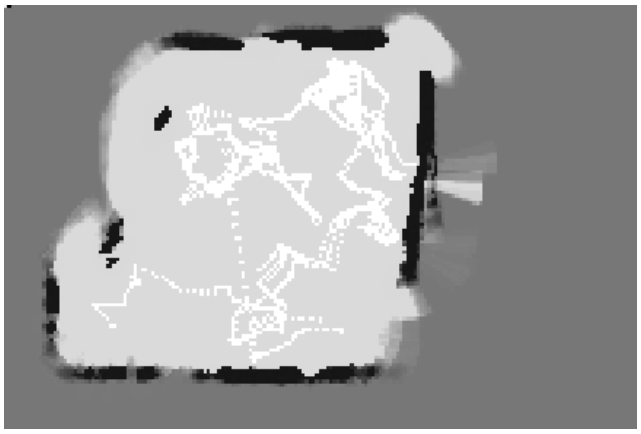


(c)

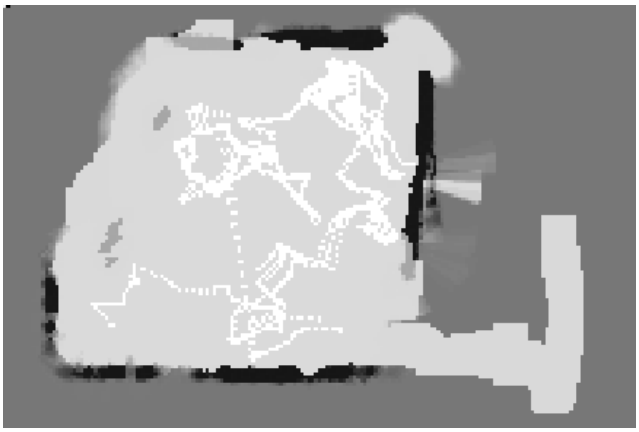
Figure 4.23: Starting from top, map of the SWS, map of the robot and both maps merged. The sycophant is now travelling in region B, whereas the follower robot is still wandering in the empty region.



(a)



(b)



(c)

Figure 4.24: Starting from top, map of the SWS, map of the robot and both maps merged. The SWS is finally in region C, not visible to the follower any more. However, the tracking algorithm estimates its pose and grows its map accordingly. Using the new map the follower is easily able to calculate a route passing the free regions and leading to the sycophant.

CHAPTER 5

3D Mapping

In this chapter, we introduce two approaches in the generation of 3D maps: the first one is a novel 3D map constructed out of 2D map slices created from different levels of the corresponding 3D environment. The 2D slices are aligned by the aid of common features present at the individual levels where each of the 2D map slices were created. In the scenarios of the first type one map is created at the sycophant level, and the other one at the follower mobile robot level and then merged. The second approach is a complete 3D map created again by the cooperation of the tracking mobile robot with the sycophant. This time the sycophant, instead of scanning the world parallel to the follower, scans the environment at slanted angles and more specifically perpendicular. Though this way the SWS maps loose their correlation at each of their time steps and it is only by the aid of the 2D horizontal map of the follower, that is the follower is able to construct the complete 3D map of the visited places.

5.1 3D maps out of 2D maps

For 3D mapping we assume that the sycophant has a laser ranger, and is able to take high resolution range scans of its environment. We also assume that the environment is indoors, having enough line features for both laser rangers of the individual scans done at two different heights, namely at the levels of the sycophant and that of the follower robot's. Under these assumptions we can create a rough 3D map of the environment by comparing the common features. In this work we merge complete 2D maps instead of continuously growing a 3D map. This decreases the uncertainty

in map merging.

While the map at the robot level could be generated using odometry and laser scans, since there is no odometry available to the sycophant, its map could be generated only from the laser scans. In order to create the 2D map of the sycophant, and also improve the odometry of the follower robot, laser scan matching was used. The scan matching method used is based on hough transform and its details are explained in the next section. The same scan matching method is also used with some modifications to merge the 2D maps into a 3D map. The modifications are sub-sampling to improve the matching speed as well as the inputs.

5.1.1 Scan Matching using Hough Transform

In scan matching the aim is to find the displacement of a robot, usually equipped with a laser ranger, by comparing scans from its two successive poses. For the scan matching algorithm to work properly the scans compared should have considerable overlap, in other words those two scans should correspond roughly to the same part of the environment. Due to this overlap by comparing the scans it is possible to find the transformation matrix $T(\Delta X, \Delta Y, \Delta\theta)$, or in other words the displacement, between the two poses. This displacement could be used instead of odometry in a robot, or it could be used to improve available odometry considerably. A high quality odometry increases the performance of SLAM algorithms by improving the quality and increasing the speed of processing by decreasing the number of particles necessary. Since our sycophants do not have any odometer, scan matching is an indispensable tool for mapping using the sycophant.

Various methods have been proposed in the literature for scan matching. The methods could be roughly classified as ones operating on point clouds, i.e. the raw laser scan, and ones operating on features like lines and corners extracted from the raw laser scan. To keep the implementation simple we have decided to utilize a method based on point clouds. Again with the aim of simplicity and traceability, but without compromising from accuracy we decided to utilize a Hough Transform based method for scan matching. In this method the rotational component, $\Delta\theta$, is found by cross correlations of the Hough Spectrum, and then translational components, $\Delta X, \Delta Y$, are

found by cross correlation of the X and Y spectrums of the rotationally aligned raw scans.

5.1.1.1 Hough Transform

Scans expressed in Cartesian coordinates allow us to construct, draw and understand the world. However transforming the scans from one reference frame to another, rotation and translation become coupled. Hough Transform, instead of working on points in the Cartesian coordinate system, it works in the polar coordinate system, utilizing decoupling of translation and rotation in the new domain. Let us now introduce the well known methodology for the sake of completeness.

Equation of a line in 2D Cartesian space is given by:

$$y = mx + b$$

where x and y are the Cartesian coordinates, m is lines slope and b is the point the line crosses the y axis.

In 2D Hough space a line is expressed as

$$\rho = x\cos(\theta) + y\sin(\theta)$$

where ρ is the distance of the line to the origin, and θ is the angle between the x-axis and the distance vector of the line.

In Cartesian space we need an infinite set

$$S_L^c = \{(x, y) : y = mx + b\}$$

to represent a line, but this set reduces to a single point in the Hough domain

$$S_L^p = \{(\rho, \theta) : \rho = x\cos(\theta) + y\sin(\theta)\ \&\ (x, y) \in S_L^c\}$$

However a point represented by a single tuple in Cartesian space

$$S_P^c = \{(x, y) : x = x_0 and y = y_0\}$$

is represented by an infinite set representing a sinusoid in Hough space

$$S_P^p = \{(\rho, \theta) : \rho = x\cos(\theta) + y\sin(\theta)\ \&\ (x, y) \in S_P^c\}$$

If instead of the Cartesian line S_L^c, each of its points is transformed into Hough domain, represented by S_i, we get a collection of sinusoids

$$S_S^p = \bigcup_{i=1}^{\infty} S_i$$

This set of sinusoids intersects at a single (ρ, θ), which is the corresponding Cartesian line's Hough representation. Of course this holds for a subset of the points of the line, a line segment, even their discretized version, where they are expressed as a finite set of points. However lines from laser scans are not that perfect due to both noise and discretization error. Taking into account those errors lines can be represented as

$$y = (m + \tilde{m})x + (b + \tilde{b})$$

where $\tilde{m}$ and $\tilde{b}$ are random variables. Therefore when the output of a laser ranger corresponding to a line is transformed into Hough domain, sinusoids do not intersect at a single (ρ, θ) but in a region $(\rho + \tilde{\rho}, \theta + \tilde{\theta})$. Based on these observations the Hough transform is implemented as a 2D accumulator array whose rows and columns correspond to the discretized θ and ρ. This version is also called the Discrete Hough Transform.

The transform works by calculating the discretized (ρ, θ) set for each Cartesian point of the laser scan. Then after each calculation the accumulators corresponding to the (ρ, θ) pair are incremented. When all points of the scan have been processed, we have a 2D array with many sinusoids where Cartesian lines are visible here as intersection of many sinusoids, in other words as the maxima of the 2D accumulator array.

Then the Hough Spectrum, $HS(\theta)$, which will be used to find rotational difference between two successive scans, is calculated by summing over the columns of the Hough transform, in other words the 2D accumulator array $DHT(\rho, \theta)$, obtaining a transform than depends only on θ, which is invariant to translations.

$$HS[i](\theta) = \sum_{k=1}^{M} DHT[k, i](\rho, \theta)^2$$

Algorithm 2 Hough Transform

procedure HOUGHTRANSFORM(M) ▷ Grid map as a list of occupied pixels
  $HT(\rho, \theta) \leftarrow \mathbf{0}$ ▷ 2D MxN array of the transform is initialized
  $\theta_k \leftarrow k\frac{2*\pi}{N}\ k = 1..N$ ▷ N: Number of θ cells
  for $i \leftarrow 1..n$ **do** ▷ n:Number of pixels
    for $k \leftarrow 1..N$ **do**
      $\rho \leftarrow x\cos(\theta_k) + y\sin(\theta_k)$
      $\rho \leftarrow \rho/\rho_{max} * M$ ▷ ρ is discretized
      $HT(\rho, k) \leftarrow HT(\rho, k) + 1$
    end for
  end for
  return HT
end procedure

5.1.1.2 Scan Matching Using Hough Spectrum

In scan matching, two given laser scans S_1 and S_2 are first converted into grid maps, M_1 and M_2 respectively. Then the Hough transform and the hough spectra HS_1,HS_2 of each are calculated. Afterwards these two hough spectra are cross-correlated as

$$\Delta\Theta = \arg\max_{\phi} \sum_{\theta=-\pi}^{\pi} HS_1(\theta)HS_2(\theta - \phi)$$

and the peak value $\Delta\Theta$ is found as the rotational displacement of the two given laser scans. Note that Hough Spectrum is circularly periodic and therefore cross-correlation has to unwrap properly. After $\Delta\Theta$ is found the second map is back-rotated, so that we have a new map M_2'. Then X and Y spectra of the grid maps are calculated by summing column and row wise respectively.

$$S_x^M(i) = \sum_{k=1}^{N} M(k, i)$$

$$S_Y^M(i) = \sum_{k=1}^{N} M(i, k)$$

Finally cross-correlating the spectra

$$\Delta X = dx * \arg\max_{j} \sum_{i=1}^{N} S_x^{M_1}(i) S_x^{M_2}(i - j)$$

$$\Delta Y = dy * \arg\max_{j} \sum_{i=1}^{N} S_Y^{M_1}(i) S_Y^{M_2}(i-j)$$

the loci of the maxima are found as the translational displacements ΔX and ΔY. dx and dy are the quantization steps.

Algorithm 3 Scan Matching Using Hough Spectrum

1: **procedure** SCAN MATCH(S_1, S_2)
2: $M_1 \leftarrow OccupancyGridMap(S1)$
3: $M_2 \leftarrow OccupancyGridMap(S2)$
4: $HS_1 \leftarrow HoughSpectrum(M1)$
5: $HS_2 \leftarrow HoughSpectrum(M2)$
6: $C_{HS} \leftarrow CrossCorrelation(HS_1, HS_2)$
7: $\Delta\theta \leftarrow \arg\max_\theta C_{HS}$
8: $M_2' \leftarrow Rotation(\Delta\theta) M_2$
9: $S_x^{M1} \leftarrow XSpectrum(M_1)$
10: $S_y^{M1} \leftarrow YSpectrum(M_1)$
11: $S_x^{M2} \leftarrow XSpectrum(M_2)$
12: $S_y^{M2} \leftarrow YSpectrum(M_2)$
13: $\Delta X \leftarrow \arg\max_x CrossCorrelate(S_x^{M1}, S_x^{M2})$
14: $\Delta Y \leftarrow \arg\max_y CrossCorrelate(S_Y^{M1}, S_Y^{M2})$
15: **return** ΔX, ΔY, $\Delta\theta$
16: **end procedure**

To visualize the method lets apply it on a sample scan taken from the simulation environment given in figure 5.1. Here we compare two scans taken close to each other, figures 5.2(a) 5.2(b). In the first step we find the hough domain representation of the given scans (figure 5.3). Then θ spectra are calculated and are cross correlated as given in figure 5.4. From here we find the translational component of the transformation matrix. Then the second map is back-rotated, and the X and Y spectra of each map (figures 5.5 and 5.6) are calculated and crosscorrelated, finding the loci of the maxima, which are the displacement between the two scans in the x and y axes.

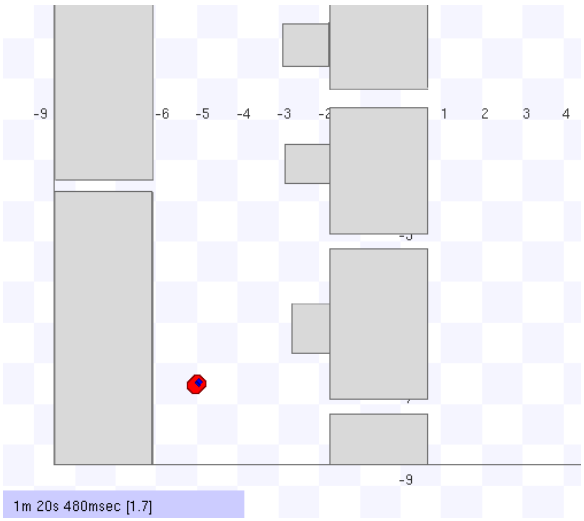
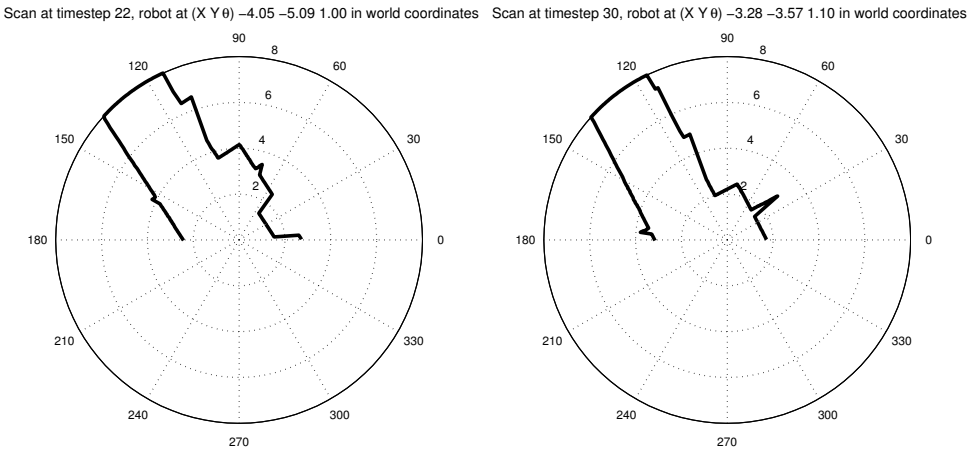
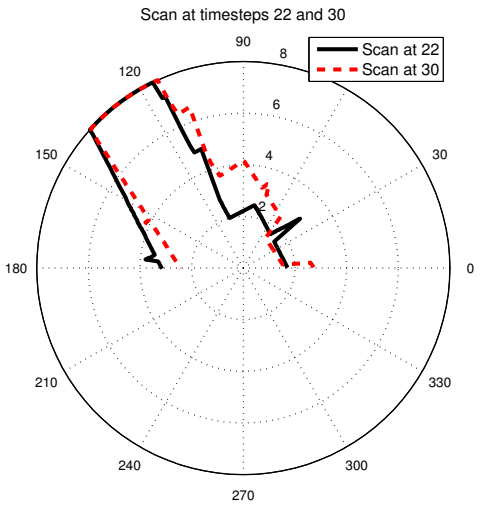


Figure 5.1: Simulation environment from which the two scans are taken

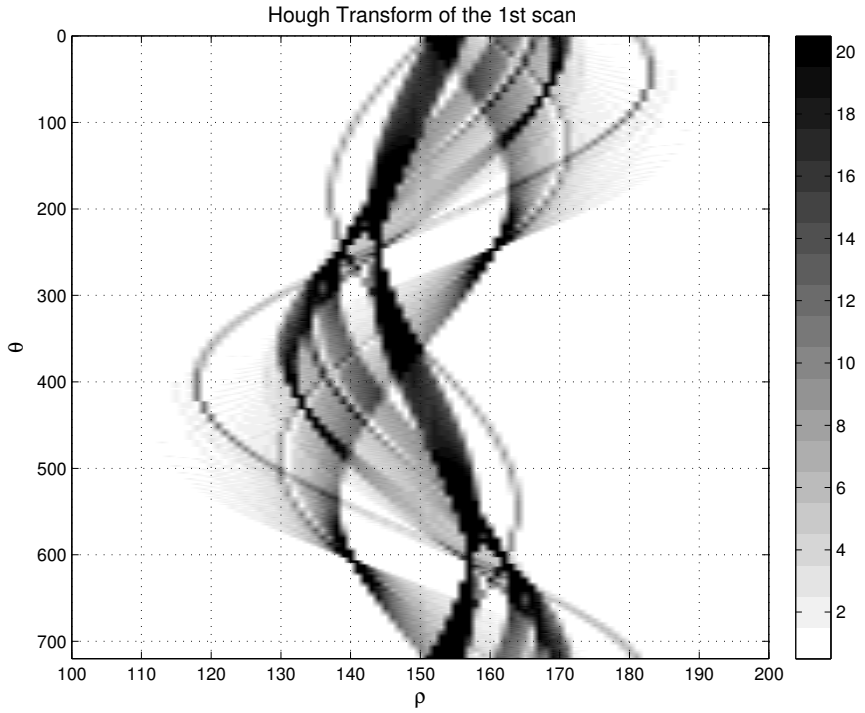


(a) (b)

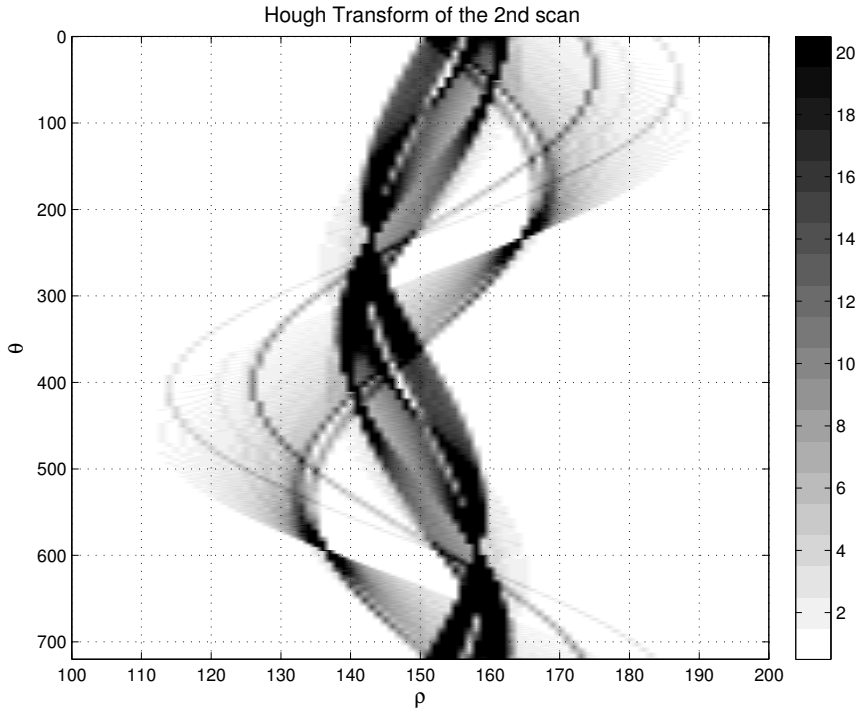


(c)

Figure 5.2: First and second scans shown in polar coordinates



(a)



(b)

Figure 5.3: (a) Hough transform of the first scan given in 5.2(a), (b) Hough transform of the second scan given in 5.2(b).

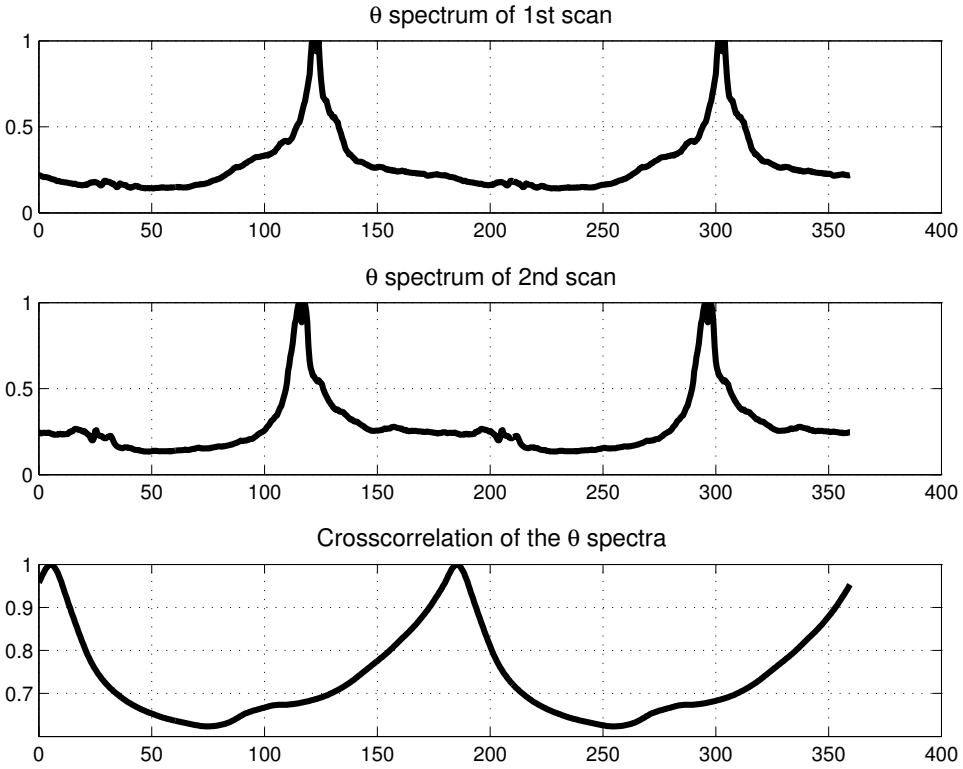


Figure 5.4: θ spectra of the hough transforms of the scans given in 5.2, as well as their cross correlation.

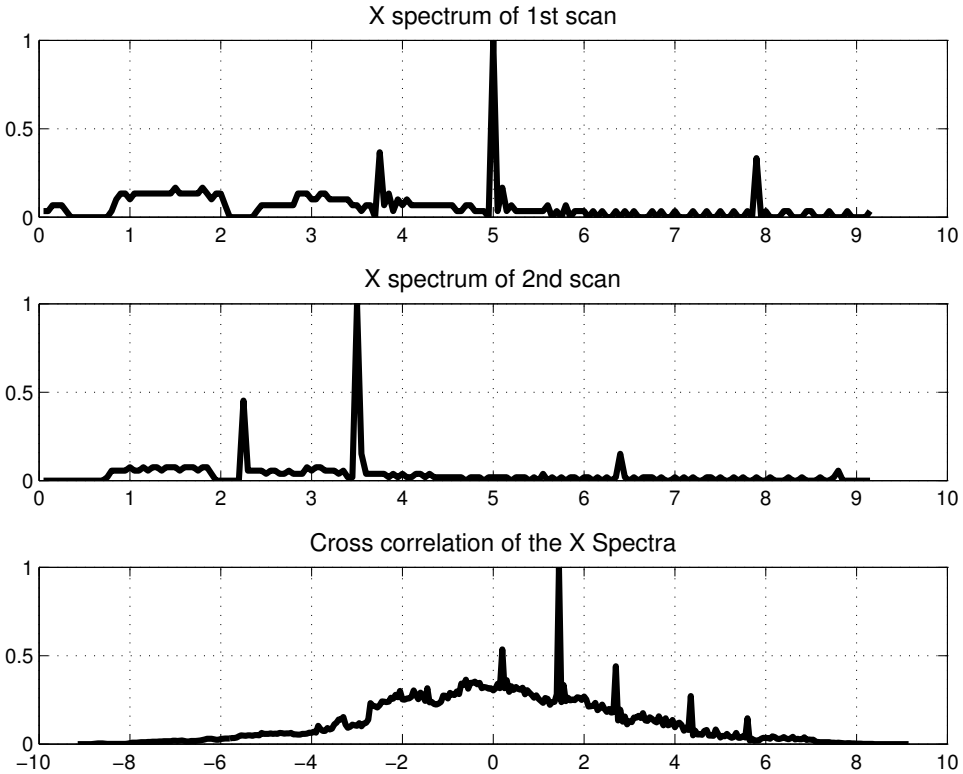


Figure 5.5: X spectra of the scans given in 5.2, as well as their cross correlation.

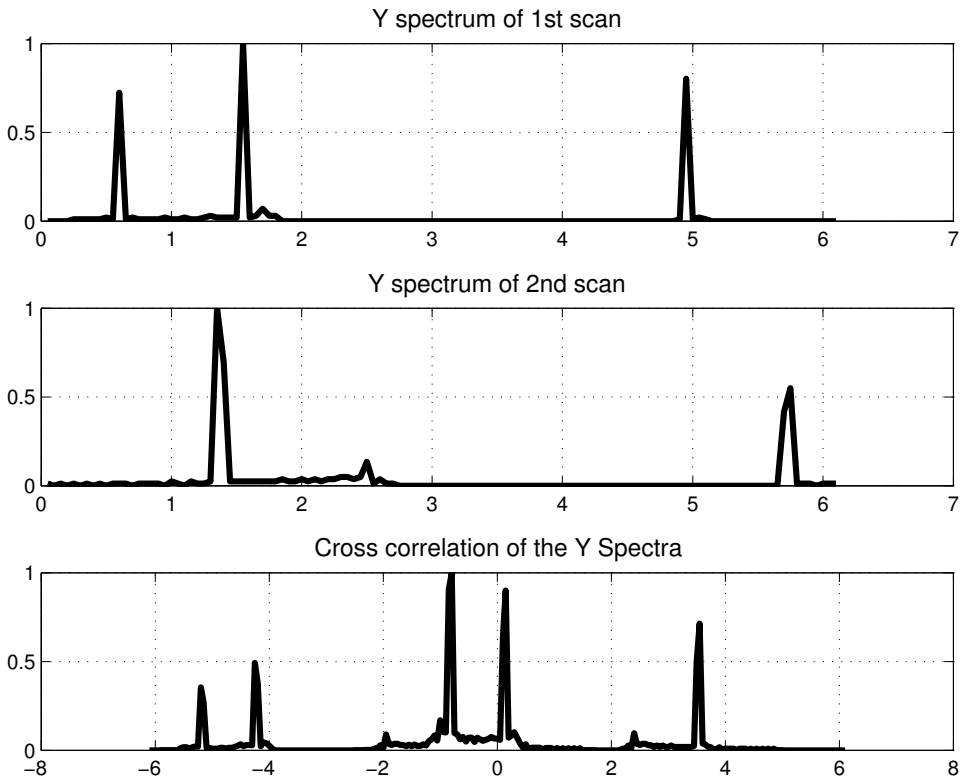


Figure 5.6: *Y* spectra of the scans given in 5.2, as well as their cross correlation.

5.1.2 Map Matching Using Hough Spectrum

As mentioned previously our maps have line features visible at both levels of the sycophant and the follower. Having common features we can apply available map merging methods. Since our indoor,rather structured environment has many orthogonal features, methods based on hough spectrum and introduced by Carpin [17] are applicable. Since we have large 2D maps, in order to improve time performance of the algorithm a subset of the points available are used. The used algorithm is given below and is almost the same as the one used for scan matching, except for the usage of a subset of the available pixels rather than directly using the created maps.

Algorithm 4 Map Matching Using Hough Spectrum

procedure MAP MATCH(M_1, M_2) ▷ Occupancy grids
  $M_1 \leftarrow subsample(M_1)$
  $M_2 \leftarrow subsample(M_2)$
  $HS_1 \leftarrow HoughSpectrum(M1)$
  $HS_2 \leftarrow HoughSpectrum(M2)$
  $C_{HS} \leftarrow CrossCorrelation(HS_1, HS_2)$
  $\Delta\theta \leftarrow \arg\max_\theta C_{HS}$
  $M_2' \leftarrow Rotation(\Delta\theta)M_2$
  $S_x^{M1} \leftarrow XSpectrum(M_1)$
  $S_y^{M1} \leftarrow YSpectrum(M_1)$
  $S_x^{M2} \leftarrow XSpectrum(M_2)$
  $S_y^{M2} \leftarrow YSpectrum(M_2)$
  $\Delta X \leftarrow \arg\max_x CrossCorrelate(S_x^{M1}, S_x^{M2})$
  $\Delta Y \leftarrow \arg\max_y CrossCorrelate(S_Y^{M1}, S_Y^{M2})$
  return ΔX, ΔY, $\Delta\theta$
end procedure

5.1.3 Simulation Framework and Performance Metrics

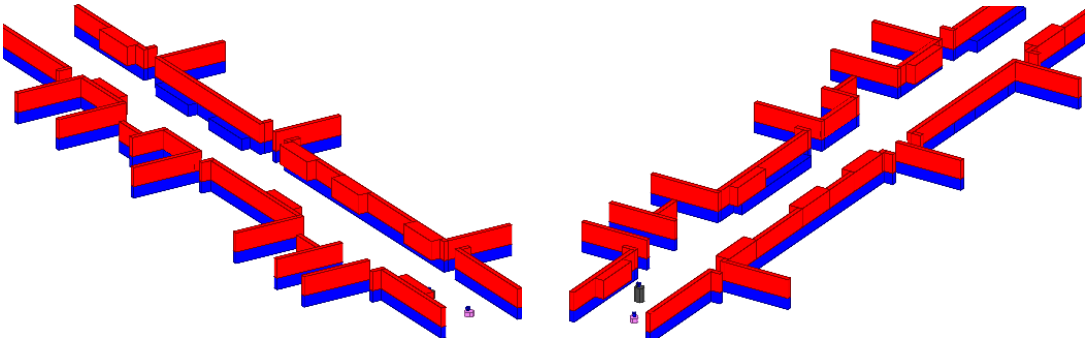
Inspired by one of the corridors of our department, the simulation was run for a corridor like structure with long benches visible to the follower only, exaggerated bulletin boards visible only to the sycophant as well as walls and open doors visible to both the sycophant and the follower. In the simulation the host carrying the sycophant was required to visit some preset way-points, while traversing the whole corridor, and the follower was required to keep an approximate distance of 2m to the sycophant, and localizing it using its laser ranger. In figure 5.7 two views from different angles of the simulation environment are given. In figure 5.8 cross sections at the height of the sycophant's and that of the follower's laser scan level are given respectively. These cross sections are the worlds that they can see with their laser rangers. It can be seen from the figures that, the two levels have many common walls, making 3D mapping possible and plausible.

In the simulations the sycophant was imitated by a high robot with a ranger following a preset way-points, imitating our independently moving agent not being guided by the SWS it carries, which is scanning at a height of 0.8m, whereas the follower, whose objective is to keep an approximate distance of 2m to the sycophant, is another robot with a ranger scanning at a height of 0.3m.

In order to test the performance for different initial conditions, i.e. different orientations, obtained maps are rotated upto 15 degrees in both directions, and independently translated as much as 5m in both directions for a set angle. The percent error which is defined as

$$\frac{abs(\text{real displacement} - \text{estimated displacement})}{\text{real displacement}} * 100$$

is given both for the rotation and displacement in fig 5.13. As can be seen in the graphs the hough transform method is quite immune to rotations, the error being almost zero. As for the performance in finding the translational component, the method is successful, keeping the error in the order of few per one thousand.



(a) (b)

Figure 5.7: Perspective views of the simulation environment used in 3D mapping taken from different angles. The blue walls are obstacles with a height of 0.6 and the red walls are on a higher level, between 0.6 and 1.0.

(a) (b)

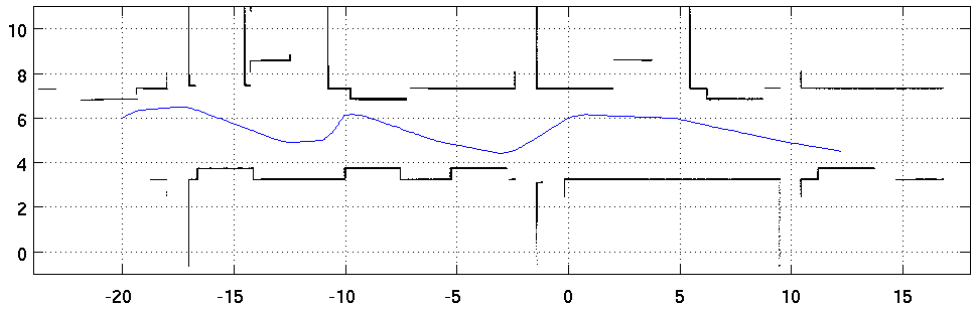
Figure 5.8: Cross sections of the simulation environment given in Fig. 5.7. (a) corresponds to the follower robots level (blue in previous figure), where the black thick lines correspond to benches. (b) corresponds to the sycophant's level (red in previous figure), where the black thick lines correspond to the bulletin boards.

5.1.4 Simulation Results and Performance Analysis

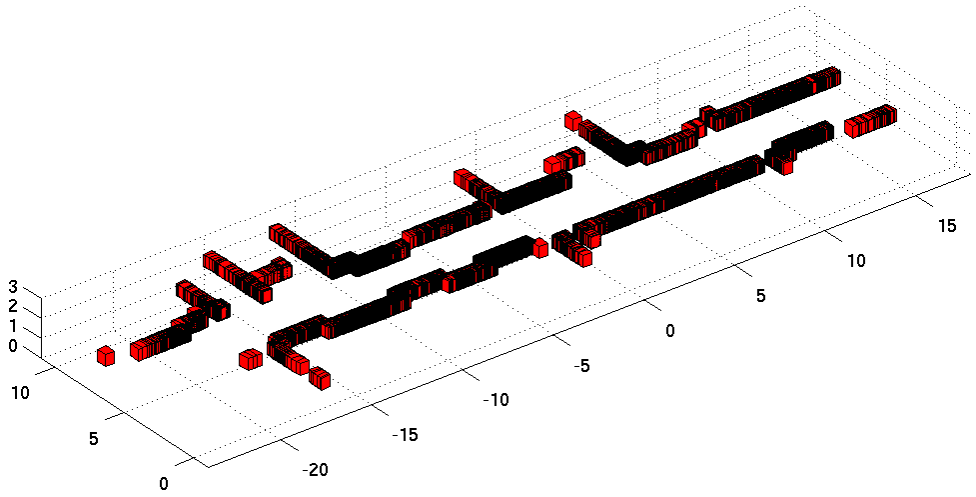
Obtained maps of the sycophant and the follower are given in figure 5.9 and figure 5.10 respectively, together with their extruded version for illustrative purposes. To assess performance of the alignment, the maps were aligned using only the described hough transform method, without using knowledge of the sycophant's pose in the followers reference frame. As can be comparatively seen in figure 5.11 the maps align quite well. The merged 3D map is given in figure 5.12, which is quite close to the simulated world of figure 5.8.

5.1.5 Hardware Implementation and Results

In order to prove the usability of the 3D map as well as its construction technique we now run the hardware system in a classroom of our department (figure 5.14). The



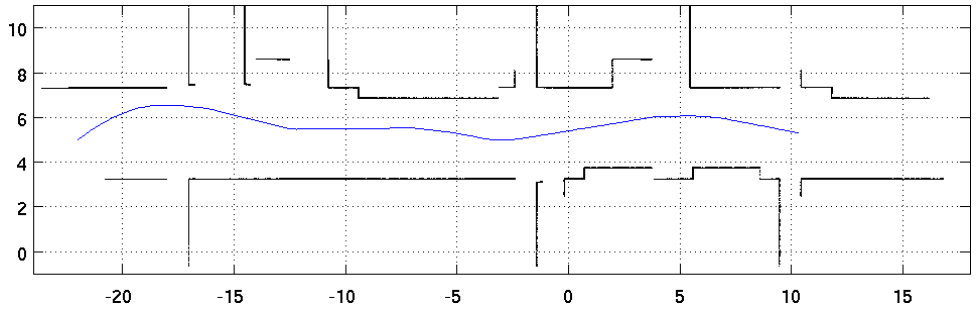
(a)



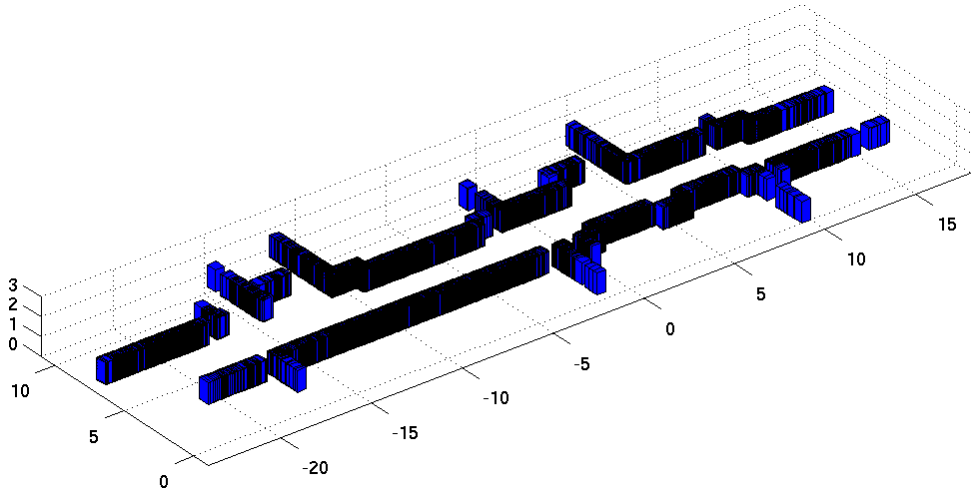
(b)

Figure 5.9: (a) 2D map of the simulated world as created by the sycophant, and its trajectory. (b) Extruded form of the map.

classroom has windows and window frames visible only to the sycophant (carried by me), under-window walls and radiators visible only to the follower robot and wall segments as well as pillars visible to both the follower and the sycophant. The sycophant sensor was a hokuyo RB-Hok-07 laser ranger (figure 5.15(a)) scanning an area of 270 degrees, with a range of 5.6m, scanning at a height of 155cm. The follower was a MagellanPro (figure 5.15(b)) mobile robot equipped with a SICK ranger scanning an area of 180 degrees, with a range of 25m, scanning at a height of 35cm. The sycophant ranger was attached to the shoulder of a human host, which traversed a large rectangle in the classroom. The follower was required to follow the sycophant at an approximate distance of 2m using its ranger. The maps obtained by the sycophant (figure 5.16) and the follower mobile robot (figure 5.17) were merged at the end of the mission using the described algorithm. Figure 5.18 shows that map alignment performed well.

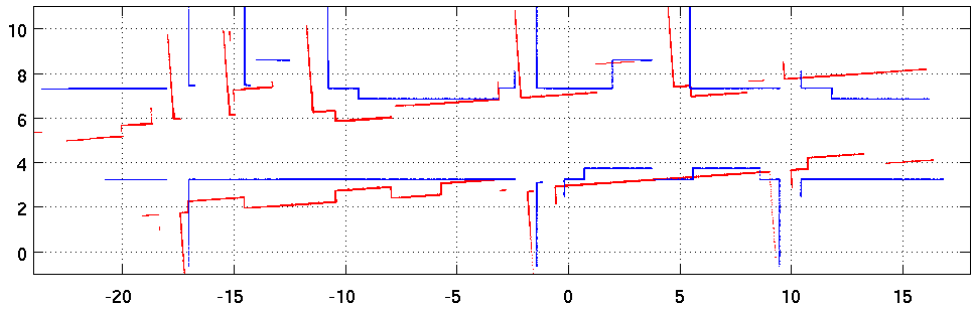


(a)

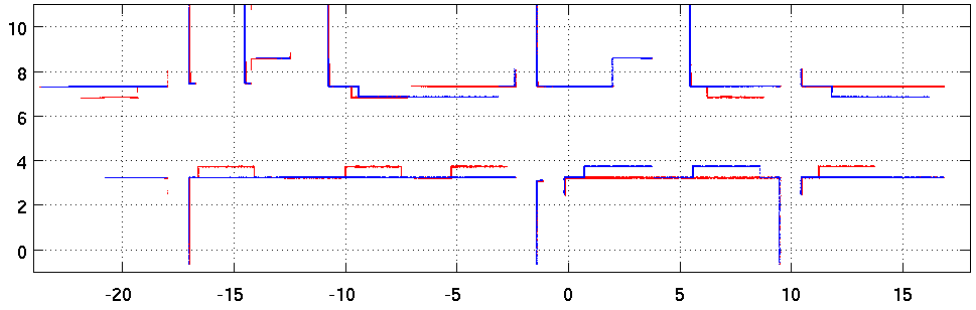


(b)

Figure 5.10: (a) 2D map of the simulated world as created by the follower robot, and its trajectory. (b) Extruded form of the map.



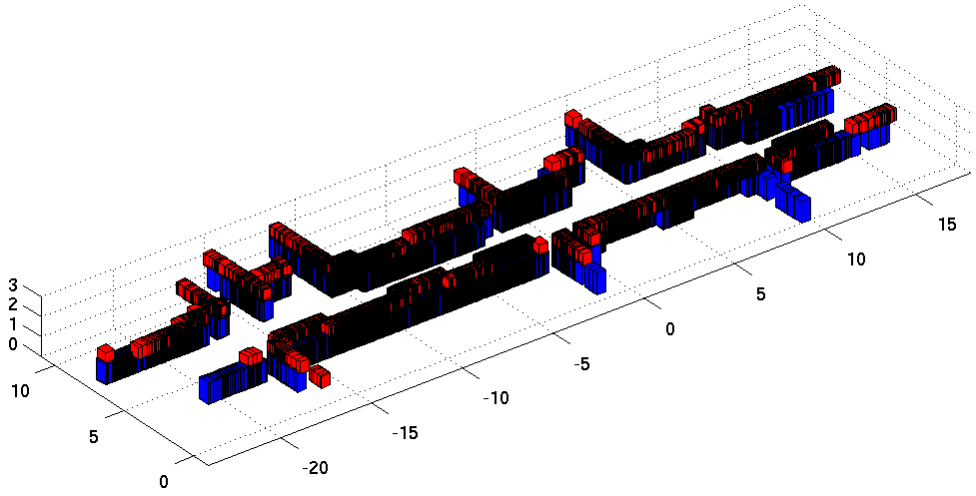
(a)



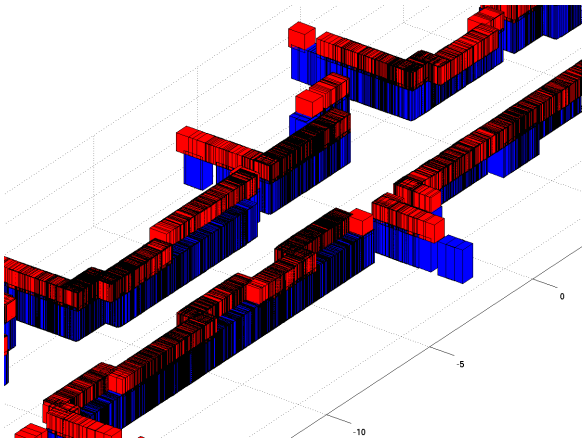
(b)

Figure 5.11: (a)unaligned maps of the sycophant and the follower robot (b) aligned maps of the sycophant and the follower robot

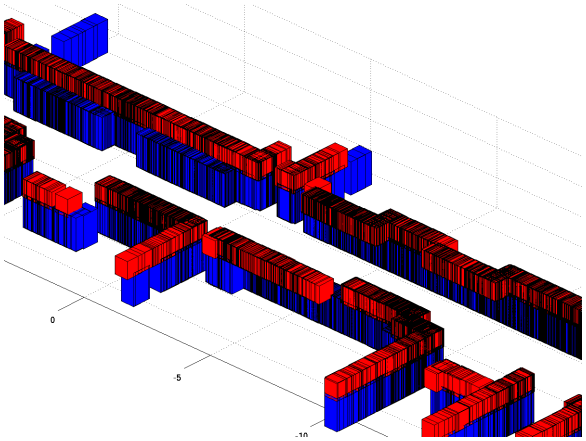
As could be seen the common wall segments at the beginning middle and end of the maps coincide well whereas some of the window frames are misaligned, which is mainly due to the noise and inaccuracy of the 2D maps. What the 2D maps really tell us is given in fig 5.19. But since our indoor environment is regular we can easily extrude each level and obtain the 3D map given in figure 5.20, which is quite informative of the classroom, yielding an approximated view of the actual structured environment.



(a)

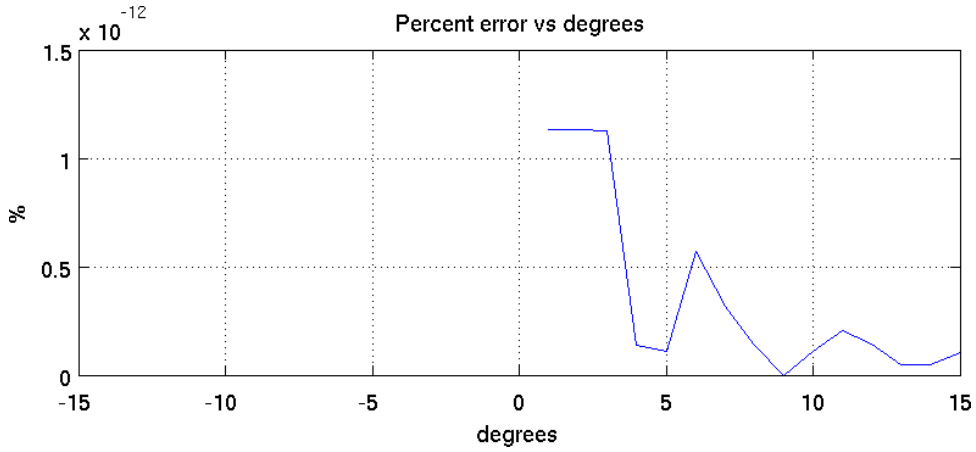


(b)

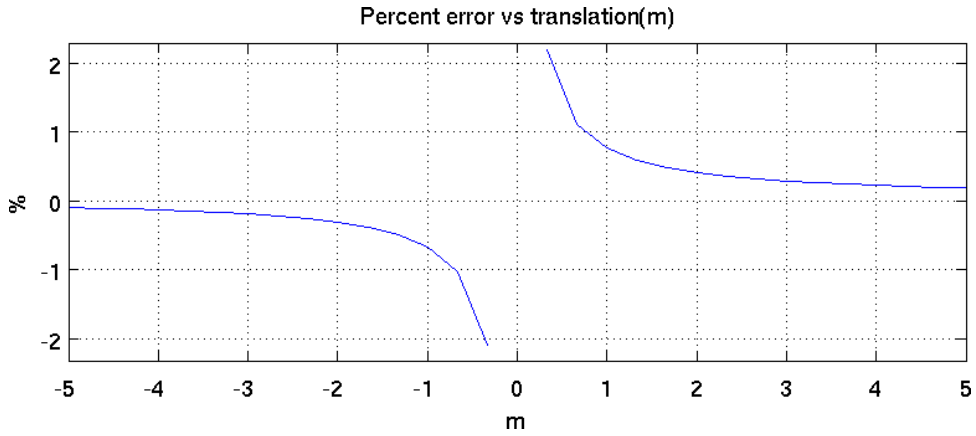


(c)

Figure 5.12: Maps of the sycophant (Fig 5.9) and the follower mobile robot (Fig 5.10) merged into a 3D map, and displayed from different angles.

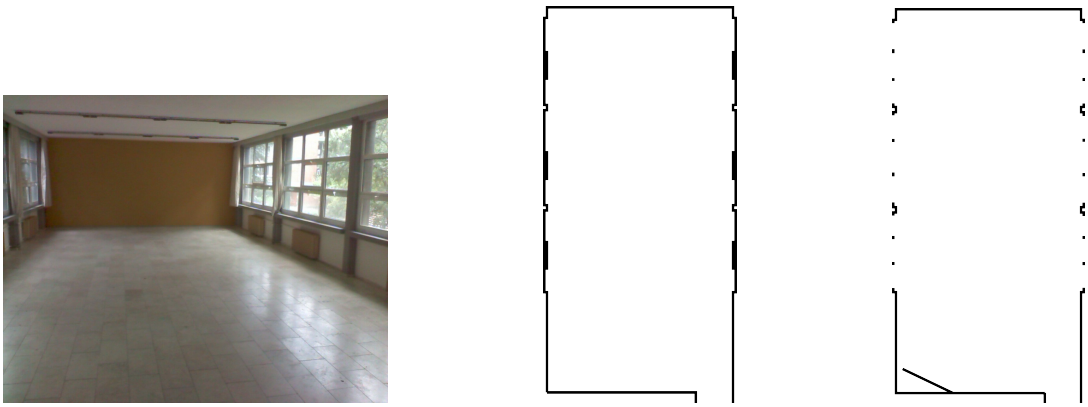


(a)



(b)

Figure 5.13: Performance of the method in finding rotational(a) and translational(b) displacements. The measure is the error in percent between the real and estimated displacements.

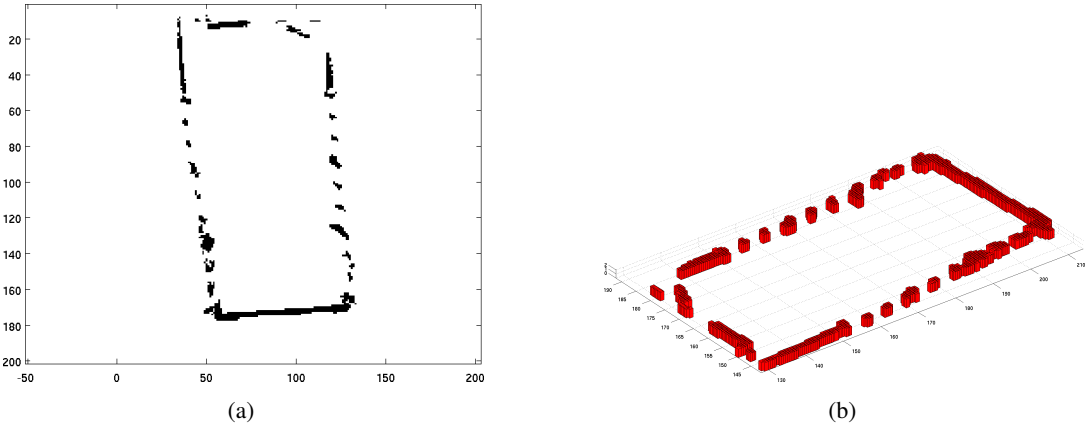


(a) (b) (c)

Figure 5.14: (a) A view of the classroom whose 3D map is obtained, (b) draft of the classroom at the level of MagellanPro, the follower robot, (c) draft of the classroom at the level of the sycophant sensor

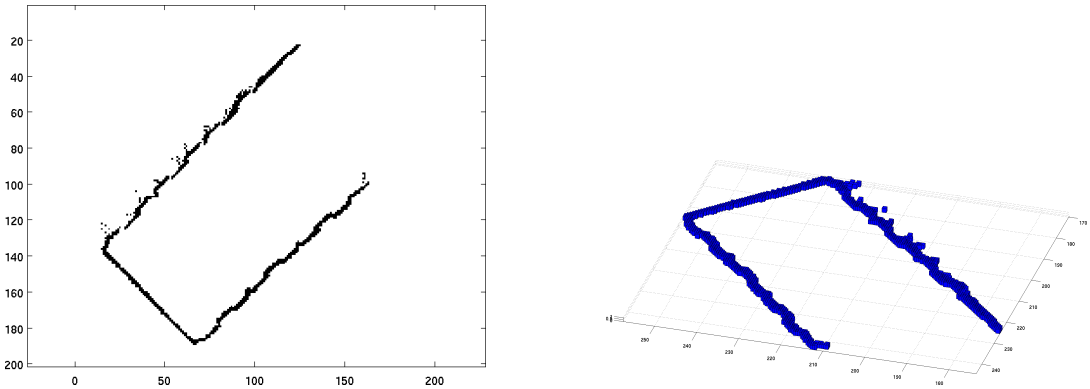
(a) (b)

Figure 5.15: The robot used as a follower to the sycophant (a), and the laser used as a sycophant sensor (b)



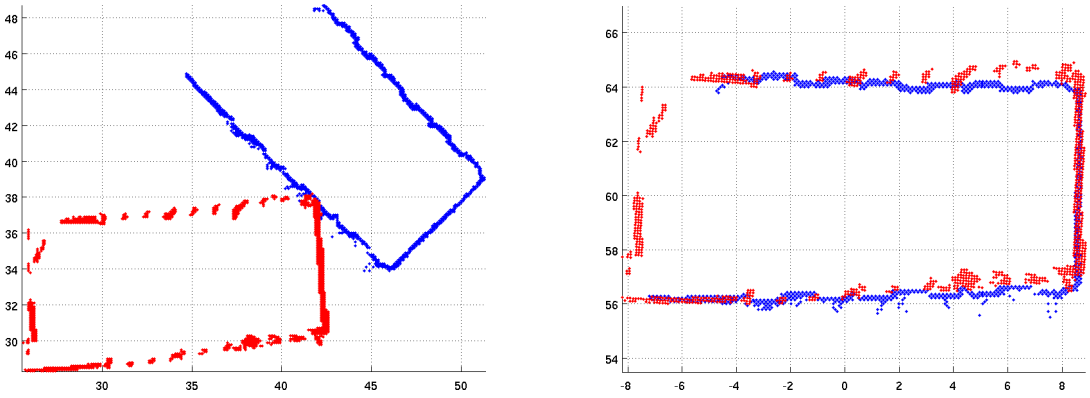
(a) (b)

Figure 5.16: (a) 2D map of the class as seen by the sycophant. (b) Extruded form of the map.



(a) (b)

Figure 5.17: (a) 2D map of the class as seen by the follower robot. (b) Extruded form of the map.



(a) (b)

Figure 5.18: (a) unaligned maps of the sycophant and the follower robot (b) aligned maps of the sycophant and the follower robot

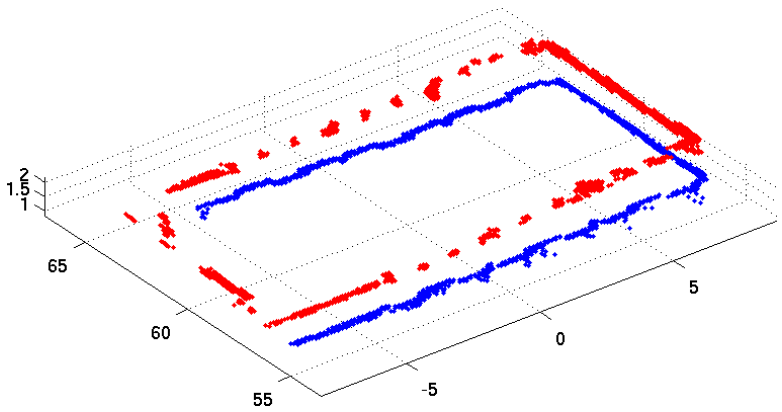
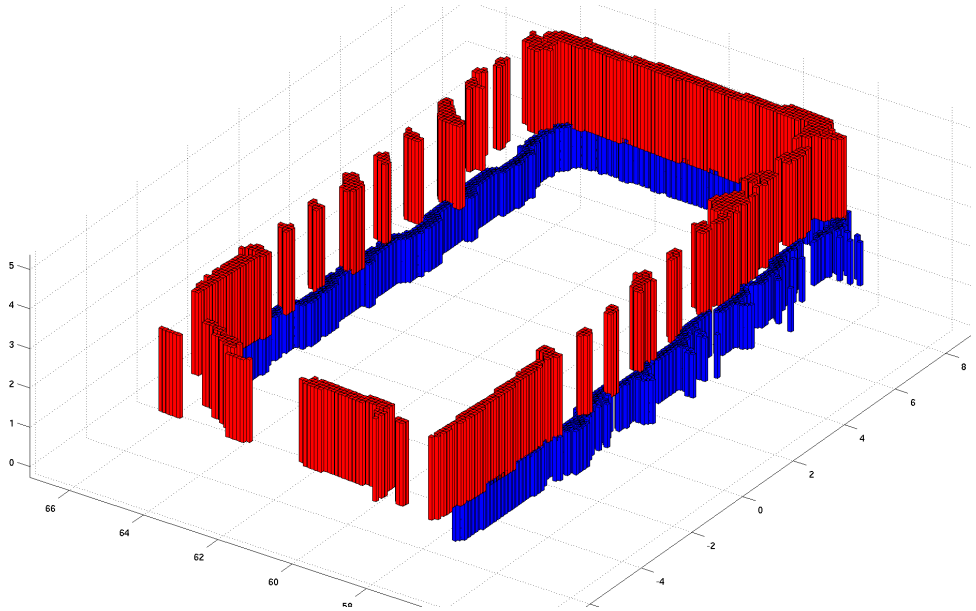
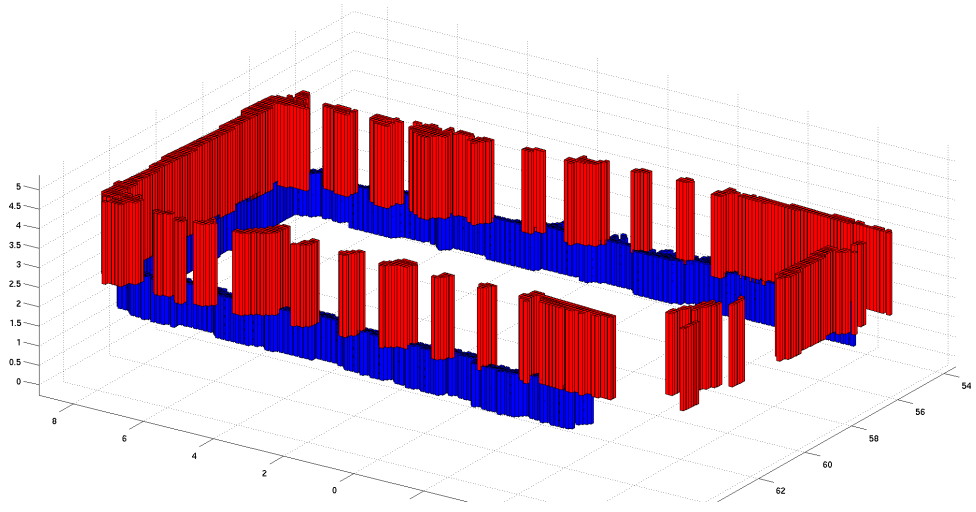


Figure 5.19: Maps of the sycophant (figure 5.16) and the follower robot (figure 5.17) in their actual scanning levels in the 3D world.



(a)



(b)

Figure 5.20: Maps of the sycophant (Fig 5.16) and the follower mobile robot (Fig 5.17) merged into a 3D map, displayed from different angles.

5.2 3D Mapping

In this section we describe yet another 3D mapping achievable using a sycophant and a follower robot. Although we make use of the same hardware, we are able to reconstruct much more 3D features compared to the previous section. This time sycophant's laser ranger is placed perpendicular to the ground, which is parallel to the scanning plane of the follower's robot, as well as perpendicular to the direction of its own motion. This way it is possible to scan extra features that are not completely visible by the mobile robot, which are walls for example, then it is possible to use map of the follower and its belief on the pose of the sycophant together with its 2D perpendicular scans to create complete 3D maps of the environment. However this comes at a price: we loose the 2D map of the sycophant thus loosing the opportunity to improve map of the follower robot based on common features. The follower also looses the chance to localize the sycophant at times when the SWS is not visible to the follower by comparing the 2D maps, and the scans obtained during non-communication or non-visibility.

5.2.1 Method

The method is customized to fit the hardware available. In our setup the Sycophant has a small and new ranger that is capable of scanning at 10Hz, whereas the follower robot is able to scan only at 2.5Hz. Although we are able to localize the SWS carrier properly, there is still a considerable amount of error in the localization, which can distort the 3D map. To prevent this distortion, the 3D mapping algorithm has to be run at an even lower frequency in order to wait for enough data whose average combined with motion knowledge can prevent misalignments of the 3D scans. So the algorithm has four different loops and corresponding repetition intervals. The follower robot is able to scan the environment every Δt^{F}_{Scan} seconds, the sycophant is able to scan the environment every Δt^{S}_{Scan} seconds and relay this information every $\Delta t^{SF}_{Communication}$ seconds. The follower runs the 3D map growing algorithm every Δ_{3Dmap} seconds. $\Delta t^{SF}_{Communication}$ is not necessarily the same as Δt^{S}_{Scan}, and it is allowed to be larger to preserve RF power. And Δ_{3Dmap} is larger than all time constants, large enough to show the average behavior of the system.

The follower robot reads its ranger and odometry data every Δt^F_{Scan} seconds. After every reading the follower scan matches the last two laser scans to improve the odometry. Then it uses its final odometry and scan data to grow its own 2D map. Afterwards it localizes the sycophant in its current scan by using a human detector algorithm, whose detail are explained later. After the carrying agent is localized in the scan, the location is transformed into the reference frame of the 2D map.

The human detector algorithm works by finding a hole in the 2D scan of the environment. This hole is a close obstacle with predefined angular widths for varying ranges. So if there is a narrow obstacle of with say 20 degrees at 2 meters, it is identified as a human in the algorithm. Apparently the algorithm has the disadvantage that other similar objects may be present in the environment. There are two ways to handle other obstacles. The first one is to compare location of the human candidate objects in different scans, and choose the candidate object which seems to be moving across various scans. However this alone has the disadvantage that it can fail if the human stands still for a while. This can be remedied by marking and keeping track of the object in various frames. Human detector works under the assumption that the human is not too close to walls or other obstacles, which would kill its laser signature in the environment.

Algorithm 5 Follower robots 2D map growing

1: **procedure** GROW 2D FOLLOWER MAP
2: **every** Δt^F_{Scan} seconds **do**
3: $S^F_{Current} \leftarrow$ Read Laser Scan
4: $T(\Delta X, \Delta Y, \Delta\theta)^F \leftarrow \text{ScanMatch}(S_{Previous}, S_{Current})$
5: $M^F \leftarrow M^F + T(\Delta X, \Delta Y, \Delta\theta) S^F_{Current}$
6: $Sx(t), Sy(t), St(t) \leftarrow$ HUMANDETECTOR$((S^F_{current}))$
7: $Sx(t), Sy(t), St(t) \leftarrow T(\Delta X, \Delta Y, \Delta\theta)[Sx(t), Sy(t), St(t)]$
8: $S^F_{Previous} \leftarrow S^F_{Current}$
9: **end every**
10: **end procedure**

Another time loop runs every $\Delta t^{FS}_{Communication}$ seconds and receives all the ranger data collected since the last time a connection was established.

Algorithm 6 Collect data from sycophant

1: **procedure** COLLECT DATA FROM SYCOPHANT
2: $N \leftarrow \frac{\Delta t^{FS}_{Communication}}{\Delta t^{S}_{Scan}}$
3: **every** $\Delta t^{FS}_{Communication}$ seconds **do**
4: $S^{S}_{t-\Delta t^{FS}_{Communication}:t} \leftarrow$ Read Last N Laser Scans
5: **end every**
6: **end procedure**

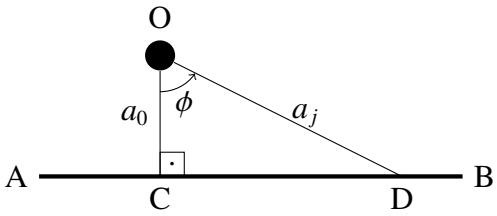


Figure 5.21: Finding ϕ at turnings

A final loop running on the follower every Δt_{3DMap} seconds grows the 3D map of the sycophant. At every iteration it calculates the mean velocity for the last part of the route as well as the average heading. It also checks whether any turning action has been taken by the sycophant. This is detected by checking first the speed of the sycophant, which should be zero at turning points, as well as the time derivative of the scans. At a turning point, successive scans are going to get either shorter or longer depending on the direction of the turning with respect to the current wall. If no turnings are detected, the algorithm calculates the pose of each scan of the sycophant by a uniform velocity model, and it also calculates the direction of the laser scans as the direction perpendicular to the heading. If turnings are detected, the amount is calculated, as described in the next paragraph. in order to find the speed and the amount of turning, which is calculated for each scan of the last scans of the sycophant. Then this amount is used to adjust the direction of the corresponding laser scan. After the final angle and position are calculated, either with or without turning, the transformation matrix is calculated and applied to each laser scan, which are then added to the existing 3D map.

The amount of rotation at a turning is calculated by simple trigonometry: Assume that point **O** in figure 5.21 is the agent carrying the sycophant and it is rotating around its own axis in the direction shown by the angle ϕ. Assume that $\|OC\|$ and $\|OD\|$ are

planes of the laser scans taken at times t_0, the last time the scan was perpendicular to the wall, and t_j, the current time. Also assume that a_0 and a_j are distances of the robot to the wall it is scanning at respective times t_0 and t_j. Then ϕ, direction of the scan at t_j can be easily found as

$$\phi = \arccos(\frac{a_0}{a_j})$$

Algorithm 7 Create 3D Map

```
1:  procedure GROW 3D MAP OF THE SYCOPHANT
2:      N ← Δt_3DMap / Δt^S_Scan
3:      every Δt_3DMap seconds do
4:          v^S_mean ← average(S(t − Δt_3DMap : t)/Δt_3DMap)
5:          if no rotation then
6:              for i ← 1, N do
7:                  (ΔX, ΔY) ← v_mean * Δt^S_Scan * i
8:                  (Δθ) ← AverageHeading
9:              end for
10:         else
11:             (ΔX, ΔY) ← 0
12:             Δθ ← φ
13:         end if
14:         S ← T(ΔX, ΔY, Δθ)S
15:         M^S_3D ← M^S_3D + S
16:     end every
17: end procedure
```

5.2.2 Hardware Implementation & Results

The 3D map is implemented in the same class using the same hardware as the previous section. However this time the sycophant is not scanning parallel to the ground, but perpendicular both to the ground and its own direction of motion. This way the sycophant is scanning the walls, and MagellanPro is following it by using the human detector algorithm and trying to keep a constant distance to it. The 2D map of the follower robot as well as the sycophant's estimated location during the test run are

(a)

(b)

(c)

Figure 5.22: The robot used as a follower to the sycophant (a), and the laser used as a sycophant sensor (b)

given in figure 5.23. As seen in the figure the agent carrying the sycophant traverses a rectangular trajectory parallel to the walls of the classroom, and although the motion of the sycophant is smoother than the one seen in the figure, it was not reflected due to the errors in human detection, as well as the map.

Screenshots of the 3D map of the classroom taken from different angles are given in figures 5.24, 5.25, 5.26, 5.27, 5.28, 5.29, 5.30. Note that one side of the classroom is not visible in the created 3D maps, this is due to the path taken by the sycophant, which does not include a path seeing the corresponding walls. Despite some irregularities the 3D map resembles the classroom quite closely. Not only the ground and the ceiling are visible, but also details like the radiators on the walls (figure 5.26, encircled by red), the window frames with all their structures are visible. Even the lamps at the ceiling of the classroom are visible (figure 5.25, encircled by red).

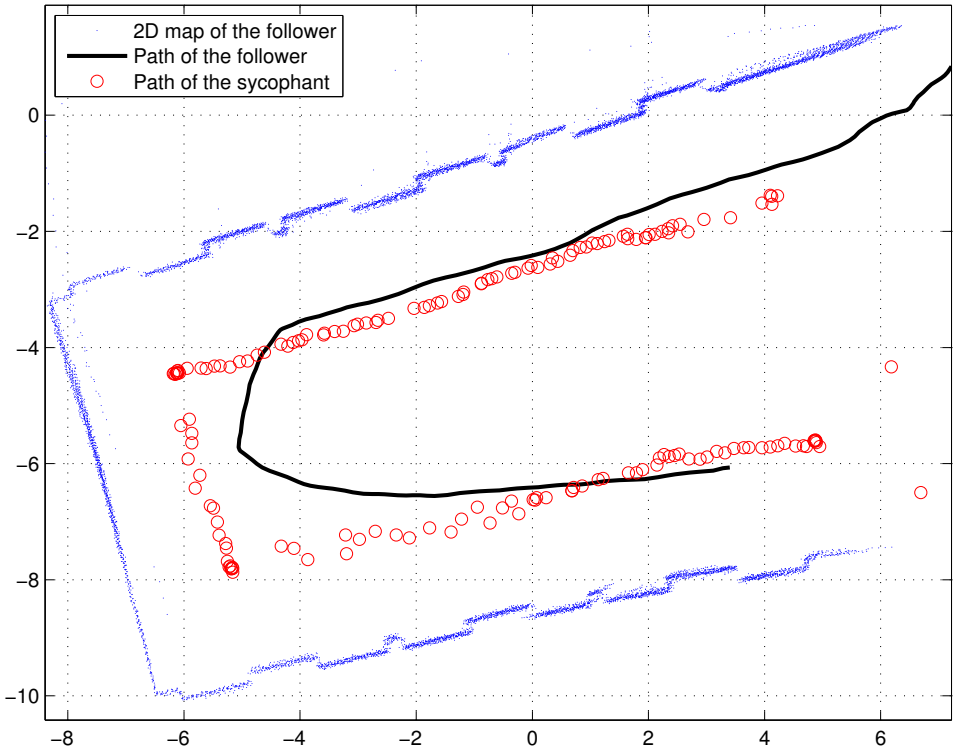


Figure 5.23: The 2D map of the follower as it tracked the sycophant, and trajectory of the sycophant.

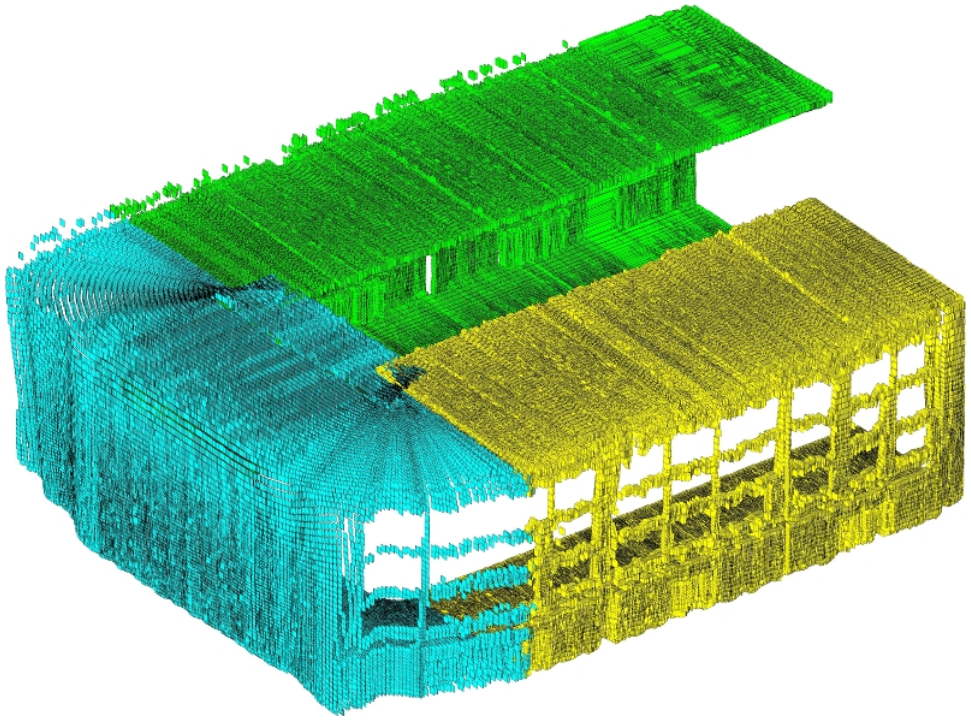


Figure 5.24: 3D map of EA211 seen from outside, above back upper right corner.

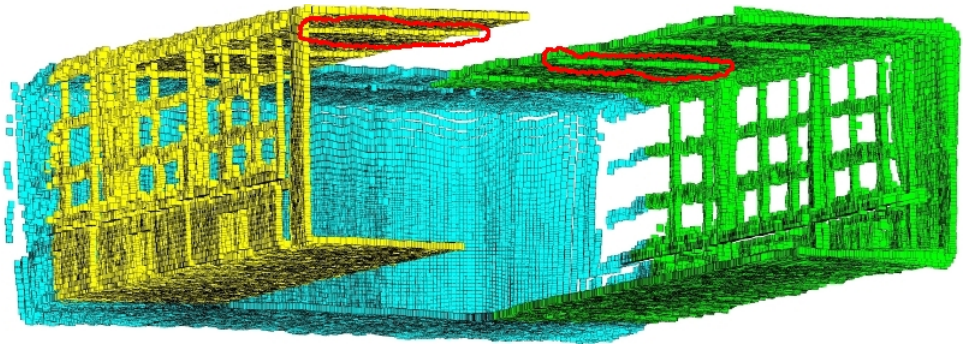


Figure 5.25: 3D map of EA211 seen from outside, below front lower left corner. Two of the lamps on the ceiling are encircled.

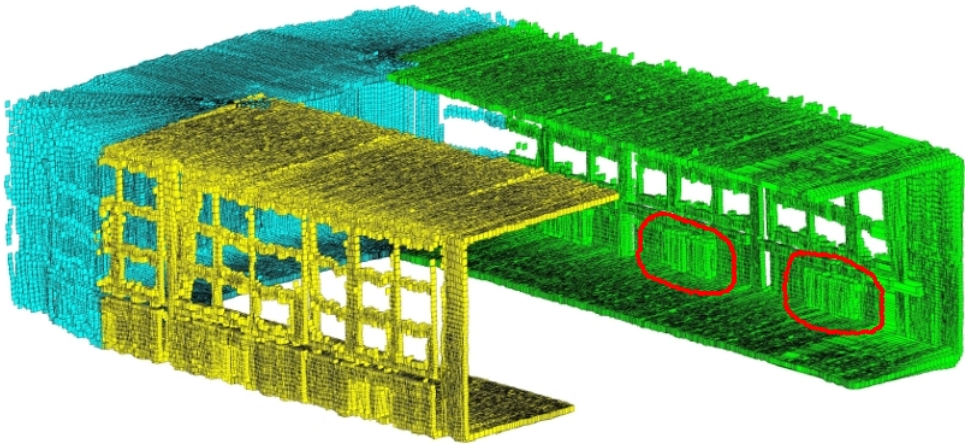


Figure 5.26: 3D map of EA211 seen from outside, above front upper left corner. Two of the radiators seen are encircled.

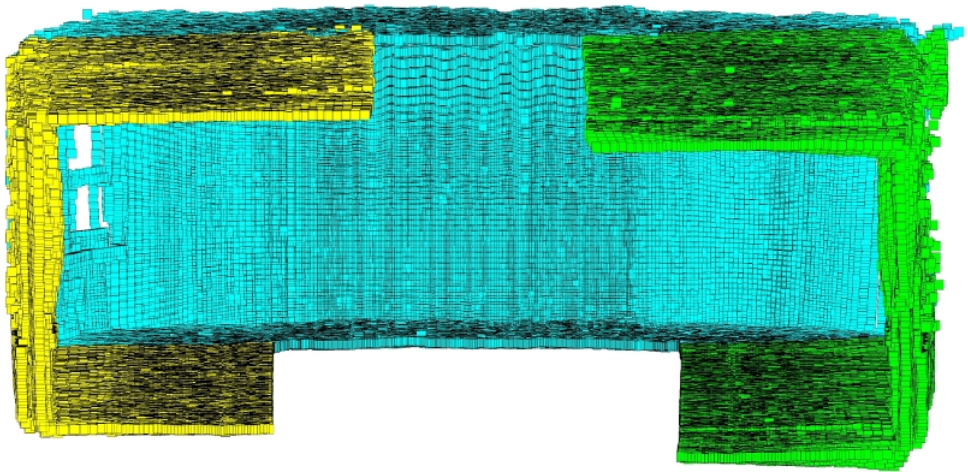


Figure 5.27: 3D map of EA211 seen from outside, above the front middle section.

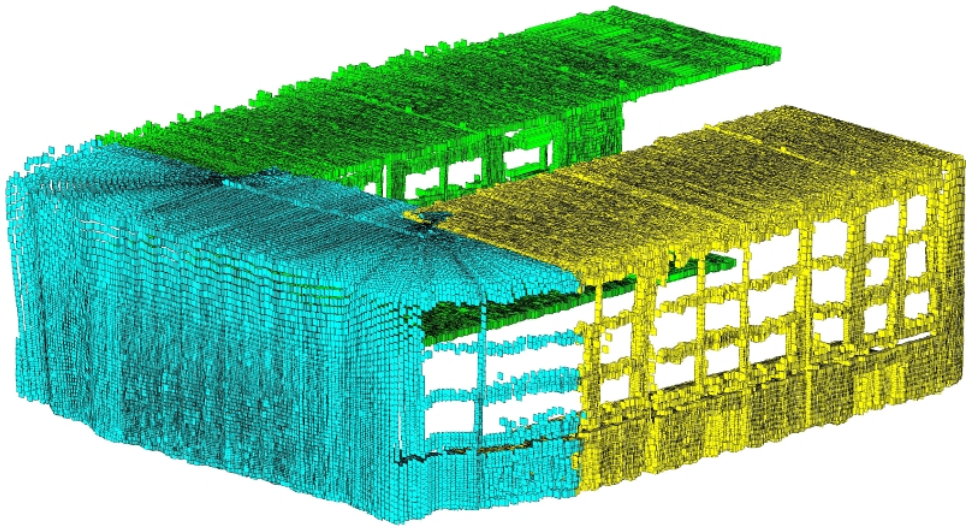


Figure 5.28: 3D map of EA211 seen from outside, above back upper right corner.

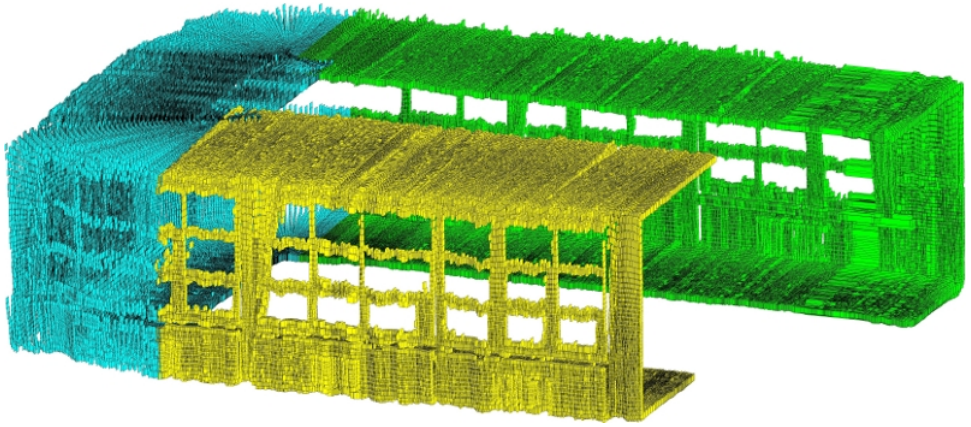


Figure 5.29: 3D map of EA211 seen from outside, upper left side.

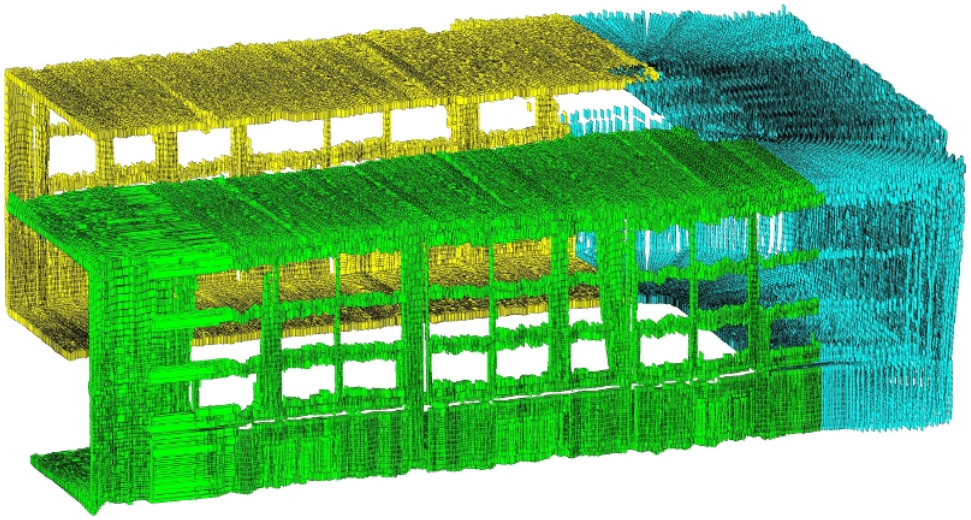


Figure 5.30: 3D map of EA211 seen from outside, upper right side.

CHAPTER 6

Connectivity Maintenance

In the previous chapters we have concentrated more on the first step of our scenario, namely a single sycophant and a single mobile robot network. The next step is the problem involving many sycophants and many mobile robots. Utilization of many mobile robots in a robot network is necessary both due to redundancy and connectivity as well as mission requirements. However deployment of those mobile agents poses another problem, that of achieving a fault tolerant network providing a continuous communication link from the sycophants to a fixed base station. Note that the mobile robots' sole mission is not mobile relaying; at the same time they will be required to participate in search and rescue missions with individual mapping, surveillance and monitoring duties. The robot and sycophant network in this chapter is called a *mobile relay network* or *leader follower network*. This network has a leader that is free to move and which in the context of this thesis corresponds to the sycophant, whereas in the general sense it could refer to any agent capable of communication with the rest of the network, and even a tracked target which pulls the network behind itself. Other agents of the mobile relay network are robots whose mission is to relay information gathered from the sycophant to a base station, and a spatially fixed base station, which is also called a sink, collecting all gathered information.

In order to provide the necessary fault tolerant network three different approaches are implemented and tested. The first approach is called "Go In Between", and in this approach each relay robot is asked to move to the middle point of its two closest neighbors using spring forces. The second approach is "Protein imitation", where the robots form a primary chain by connecting to the closest neighbors and a secondary

chain by connecting to some nodes further away, by this way imitating the α-helix structure of proteins. The third approach is based on acute-triangulation of the space forming a Gabriel Graph where robots connect and apply force to their neighbors only if the line connecting the robots is one side of an acute triangle. As will be discussed in the corresponding subsections, each approach has its own advantages and disadvantages. The thesis work in this chapter starts with the implementation and test of the simplest algorithm, which is the Go In Between. Then to remedy its main disadvantage, which is low redundancy, the other two approaches are developed.

In the next section two new performance measures developed for the evaluation of the performance of connectivity maintenance algorithms are discussed. Then in their own sections the three methods of connectivity maintenance are described and related simulations results are given and discussed.

6.1 Developing Novel Performance Measures

Deciding which algorithm works better requires well defined measures. Towards this aim we define two new measures: the *updistance*, which tells us how far a leader can go, and the *resistance*, which tells us how much redundancy the robot network has. Both will be explained in the following subsections.

6.1.1 Updistance

Updistance is a measure proposed in this thesis and inspired by a term called *up-time*, which is used in computer server business, where it means the time the server has been up without any reboot. In our connectivity context, *updistance* means the normalized maximum distance between the leader and the base station in a network of robots. The *normalized distance* is the distance divided by the theoretical maximum coverable distance by the network without loosing connectivity. For example if you have 10 nodes, and each has a communication range of 100 meters, than the maximum coverable distance would be 1000 meters, which could only be achieved by placing the relays side by side along a straight line. If at some time instant, the distance between the base station and the leader has reached at most 680 meters, the

updistance would be 0.68. Although ideally we ask for an updistance value of 1, it is not always achievable for some of our methods, as will be seen in the following sections.

As will be seen specially in the section on Gabriel Virtual Force Graphs, updistance changes depending on the initial configuration. Therefore updistances are given as histogram graphs. However in order to be able to compare updistance of different algorithms using just a single value we will also use the mean value. When the frequency based approach to probability is taken, histograms could be used as a probability density function. And as is seen in figure 6.31 updistance histograms look like probability density functions where the updistance is our random variable. The mean updistance is then calculated using the first moment as

$$\lambda_{updistance} = \int_0^1 xp(x)dx$$

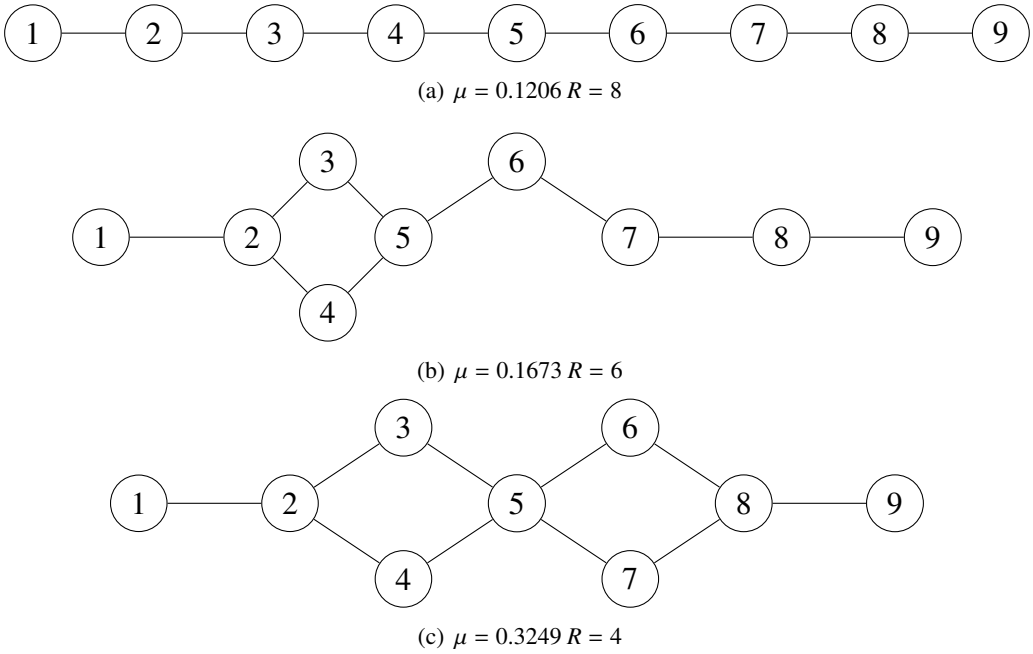
where $p(x)$ is the normalized histogram, which is

$$p(x) = \frac{h(x)}{\int_0^1 h(x)dx}$$

where $h(x)$ is the updistance histogram.

6.1.2 Resistance

Although algebraic graph connectivity (μ) is a good measure by itself, as could be seen from figure 6.1, closeness of the values makes comparison difficult, and also un-intuitive. A better measure can be developed when the graph is considered as a resistive network, where each connection between two relays has a resistance of 1 ohms. As could be seen in figure 6.1 resistance varies a lot for different configurations and it gives us a more intuitive way of looking at the system. In all three configurations of figure 6.1 we have 9 nodes. The diagram in figure 6.1(a) has an equivalent resistance (between the leader, numbered 9, and the base station, numbered 1) of 8 ohms. By looking at this resistance value in a stable network we can conclude that the system is somewhere at the boundary of breaking down. On the other hand the



(a) $\mu = 0.1206$ $R = 8$

(b) $\mu = 0.1673$ $R = 6$

(c) $\mu = 0.3249$ $R = 4$

Figure 6.1: Different distribution of relays, each having a different connectivity and resistance value.

diagram in figure 6.1(c) has a resistance of 4, meaning there is more redundancy, thus the system can tolerate loosing some nodes, and is able to stretch further. Comparing figures 6.1(b) and 6.1(c), we see that the latter having a smaller resistance compared to the previous has more redundant nodes compared to the previous.

In this thesis, the mobile relay networks are studied using three different graphs. The first is connectivity graph, whose edges correspond directly to communication links between any relay, in other words an edge exist between two relays, if they are in the communication range of each other. The second graph is just a neighborhood graph, whose edges are selected according to the connectivity maintenance algorithm used. As will be seen, each algorithm has a different neighborhood selection method. The third graph is called Gabriel Neighborhood Graph, or Acute Triangulation Graph, whose edges are determined according to the rules of acute triangulation. When comparing performance, resistance for both the connectivity graph and the Gabriel Neighborhood graph will be given.

Each algorithm below has different rules for neighbor selection, resulting in a different distribution of the relays. Although on one side we are interested in a strongly connected graph, with a high connectivity, or low resistance, we are also interested

in a sparsely distributed graph. Sparseness of the nodes contradicts the requirement of high connectivity on the one hand, but it increases redundancy and fault tolerance on the other. If for example node 7 in figure 6.1(a) fails for some reason, the network immediately looses connectivity. But such a thing would not happen in 6.1(c), node 6 would keep the system connected. So minimizing redundancy resistance does not necessarily mean minimizing the connectivity resistance, and vice-versa. Minimizing connectivity resistance requires putting all the nodes as close as possible to each other, so that more nodes see each other. However such an approach does decrease the spread of the relays, which means breaking the Gabriel graph and increasing resistance. Similarly, to decrease Gabriel virtual force resistance one has to spread the nodes as much as possible so that more acute triangles are formed, but this causes the relays go away from each other, which means increasing RF equivalent resistance.

Though calculation of resistance diagrams for neighborhood could be done in various different ways, we choose to use neighborhood diagrams created according to Gabriel Graphs. There are two reasons for this, one is this graph structure is already implemented as a connectivity maintenance method in this thesis, and the second reason is its results on sparseness are plausible.

Although similar to the algebraic connectivity we calculate two different equivalent resistance values for a network of relays: the first one is for the RF connected graph, the smaller the resistance, the better connected the system is. The second resistance is defined for the Gabriel Virtual Force Graph, which also favors smaller values for the resistance due to the fact that smaller resistance means a better connected system.

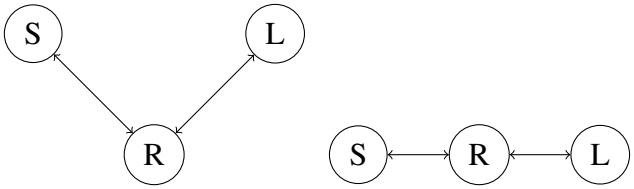
6.2 Go In Between Approach

6.2.1 Method

Although at short distances many different placements of the relays can provide feasible solutions (check fig 6.2 for two examples), at long distances the solution will usually be relays placed on a virtual line/curve connecting the sink and the leader. A straightforward positioning would be for a robot to keep equal distance to its neigh-

(a) (b)

Figure 6.2: Two different possible placement for relays connecting nodes 1 and 6. Arrows show possible communication paths. Their exact number of course depends on the communication range of each node.



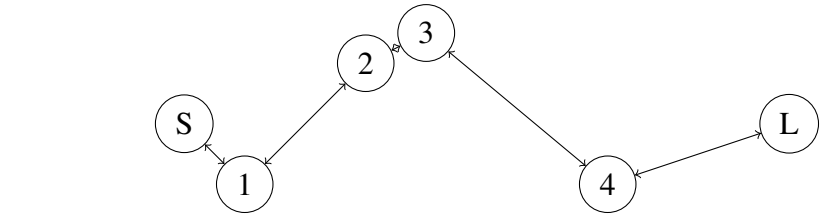
(a) Initial pose (b) Final pose

Figure 6.3: Evolution of robot poses

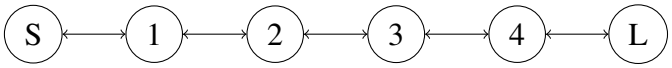
bors on both sides (one to the sink one to the leader). Such a positioning can easily be achieved if we have springs and dampers connecting a node to its neighbors. Due to the force from both sides the relay will eventually move to a location where the net force cancels out. In case of three nodes the final location is directly in the middle of the two neighbors (figure 6.3). In the case of more nodes, the solution would be an extension of this simple case and nodes starting from an arbitrary initial configuration would eventually be placed at equally spaced intervals (figure 6.4). When connected with a virtual spring to its neighbors, the spring force acting on a node i by another node j is given by

$$f_{ij} = -k(d_{ij} - d_0)\frac{\bar{d}_{ij}}{d_{ij}}$$

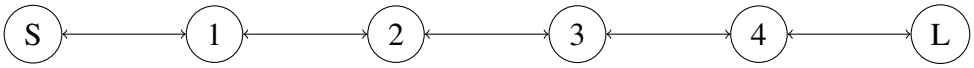
where k is the spring constant between nodes i and j , d_0 is the equilibrium distance where the nodes do not apply any force to each other, d_{ij} is the distance and $\bar{d}_{ij}$ is the distance vector between nodes i and j . The net force on node i when it is not under



(a) Initial pose



(b) Final pose



(c) Final pose

Figure 6.4: Evolution of robot poses

the influence of any obstacle is

$$F_{i_{net}} = \sum_{j \in N_i} f_{ij} - b\dot{x}_i$$

where b is the damping coefficient that will damp the oscillations , N_i is the set of neighbors selected as described below, and x_i is pose robot i.

6.2.1.1 Neighbor Selection

Starting from an arbitrary but connected initial configuration, selection of proper neighbors to bind to is very critical. Choosing the two closest neighbors is not always a good idea because of the following reasons. There may be more than two neighbors at the same distance when taking into account localization uncertainty: this case may happen more often than expected. The same nodes may be selected by different nodes and the result is a set of nodes at steady state far from the expected line. Although the leader or sink are a little further apart then the rest, they may even be completely left out leading to a disconnected network (figure 6.5). For example in figure 6.5 without a predefined conflict resolution both nodes A and B would choose C and D as their neighbors, and as the distance between S and L grows, connectivity of the network will be lost. Therefore a sequence based selection of neighbors is preferred. We assume that each robot has a unique integer identifier. The sink's identifier being the

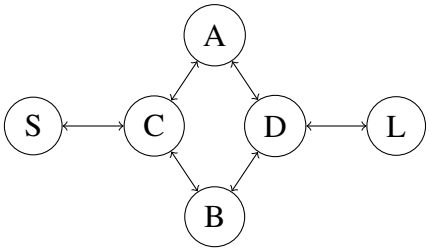


Figure 6.5: A sample initial configuration where relays A and B have the same two closest neighbors. Without a predefined conflict resolution approach, the nodes could choose C and D, and D would choose A and B, and as the leader moves away the system would get disconnected.

smallest and the leader's the largest (the opposite would work too). Then each node selects a neighbor that has the smallest id greater than its own, and another that has the largest id smaller than its own. And then assumes that these nodes are applying a force to that selecting node.

6.2.1.2 Obstacle Avoidance

For obstacle avoidance, we use again virtual springs, where the force applied by an obstacle to a node is given by

$$f_{obst} = \begin{cases} -k_{obst}(d - d_0)\hat{n} & \text{if } d \leq d_0 \\ 0 & \text{if } d > d_0 \end{cases}$$

Here k_{obst} is the spring constant of the node obstacle pair, d_0 is the safety distance, d is the distance between the object and the surface of the obstacle, and $\hat{n}$ is the unit normal vector of the obstacle surface. This force is applied only when $d < d_0$. Obstacle avoidance is not a sufficient measure by itself and requires counter measures to prevent loosing connectivity. The reason is best explained graphically.

In fig 6.6 let node B be moving downwards, and A be connected to it by a virtual spring. As B moves down, the force acting on node A will increase, but since the vertical component of the force is going to increase more, node A will tend to move more in the vertical direction, and less in the horizontal direction. However due to collision avoidance the wall is going to neutralize the vertical component of the force.

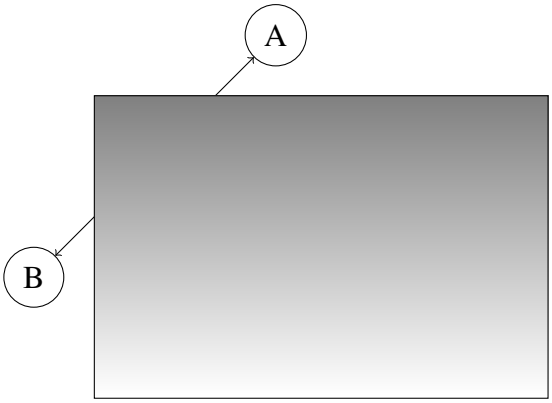


Figure 6.6: Force between two nodes around a corner

So node A will at best move slowly in the horizontal direction. And at the end, connectivity will broken.

To remedy this problem the force link could be seen as a rubber band (figure 6.7), which as B moves would eventually look like two connected straight bands. So the force acting on node A will be redirected towards the corner, removing the problem mentioned in the previous paragraph. However taking this redirection one step further, we assume that the obstacles are able to redirect the force, changing its direction not only till it is non-intersecting with the obstacle but till it is completely parallel to the obstacle surface (Fig. 6.8), so that it can slide over the surface due to the force acting from the other side. Having made a map of the neighboring, the robots can detect that there is an obstacle on the link between its neighbors. Note that when passing over corners due to the low thickness communication would not be immediately blocked but received signal strength would be reduced. And even if the link is lost, a robot can easily locate its neighbor using a state estimator like a Kalman Filter. So a node close to a corner experiences a force equal to

$$f = -k(d_{ij} - d_0)R(\alpha)\hat{d}_{ij}$$

where $R(\alpha)$ is a rotation matrix with

$$\alpha = \pi - \arccos(\hat{n} \,.\, \hat{d}_{ij})$$

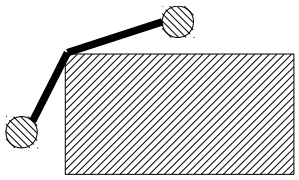


Figure 6.7: Force between two nodes around a corner

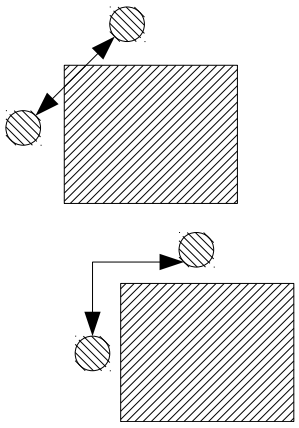


Figure 6.8: Force between two nodes, before and after redirection

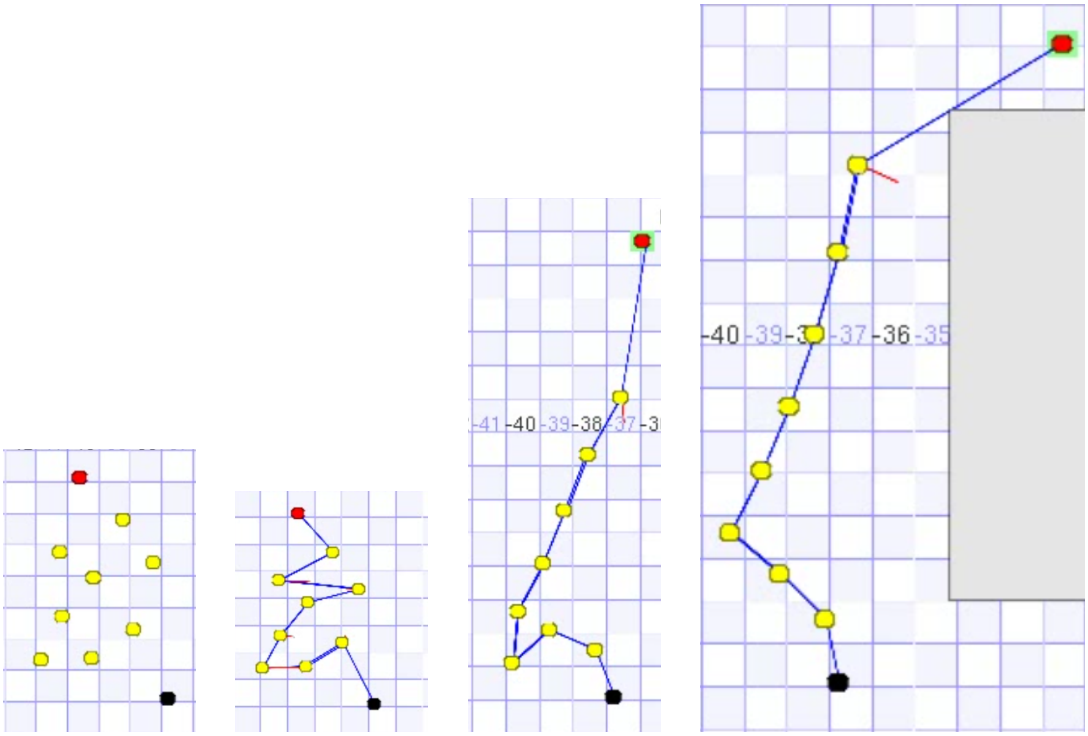
6.2.2 Simulation Results for Go in Between and Discussion

Evolution of the network for a sample scenario where the leader is moved using a mouse around a few corners in player/stage simulator is given in figure 6.9. The network successfully goes around the obstacles without bumping to the walls, and also avoids the corners.

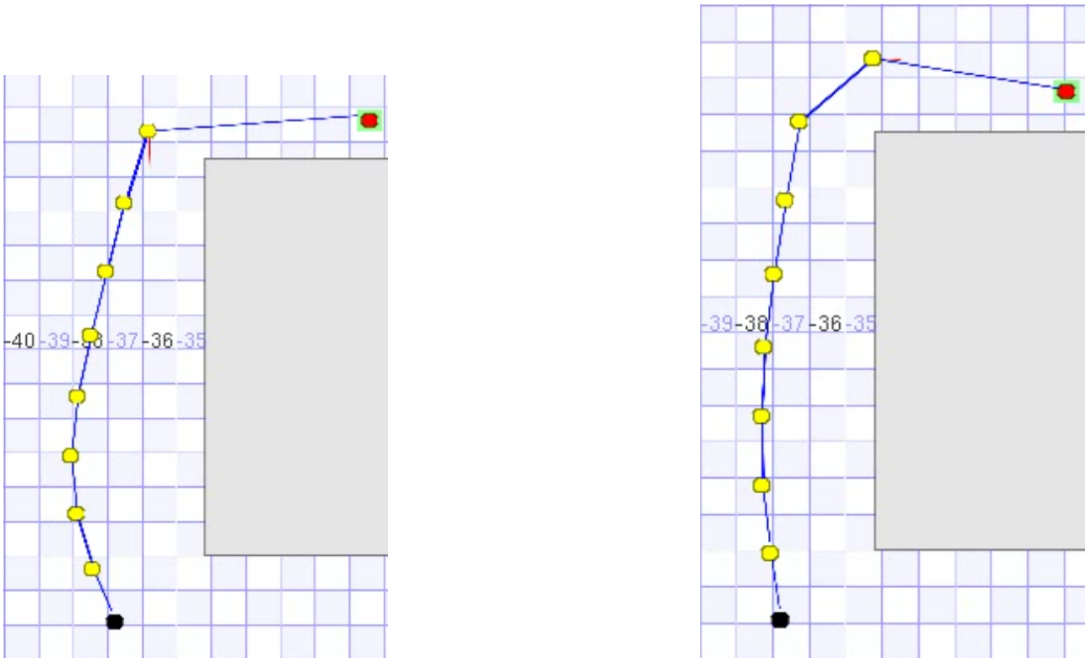
Although the described algorithm is able to properly follow the sycophant, it is not fault tolerant. This is due to the deployment of the relays, all on a sequence of lines. If communication blocking obstacles randomly appear, or some of the nodes suddenly fail for any reason, the network will become disconnected, loosing connectivity. To increase robustness of the relay network alternative methods that can induce redundancy are now developed.

6.3 Connectivity Maintenance Inspired by Protein Structures

To increase robustness of a mobile relaying network to node failures as well as to obstacles in the communication path, it is necessary to distribute the nodes in a non-

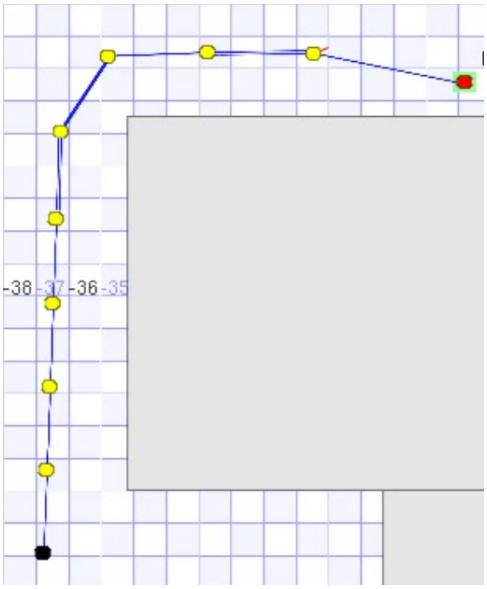


(a) t=0s, initial state (b) (c) (d)

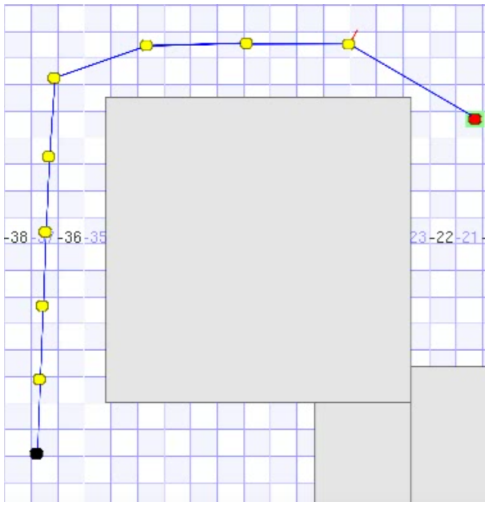


(e) (f)

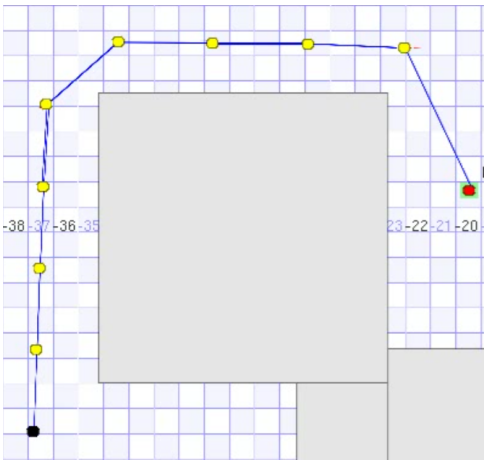
Figure 6.9: Force between two nodes, before and after redirection



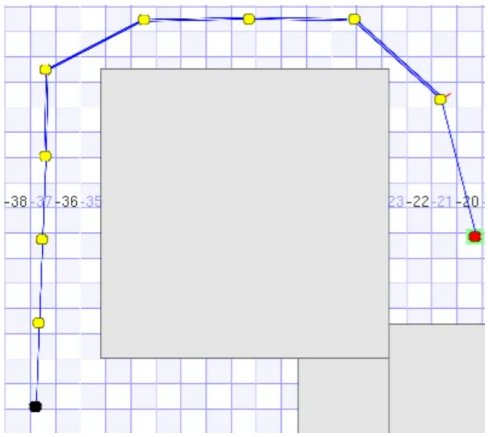
(a) t=20s



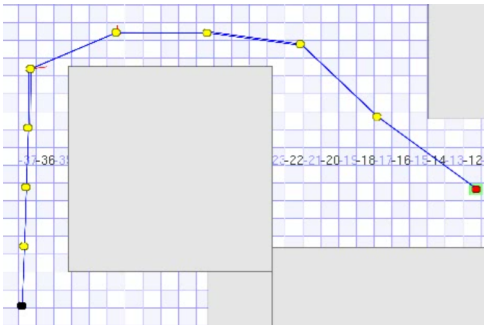
(b)



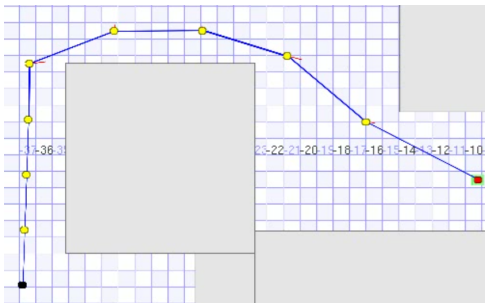
(c)



(d)



(e)



(f) t=100s, final state

Figure 6.9: Force between two nodes, before and after redirection (continuing from previous page)

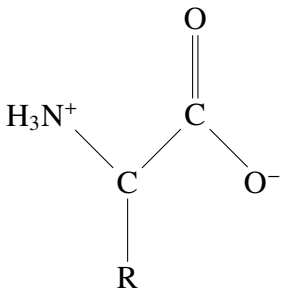


Figure 6.10: An amino acid [15].

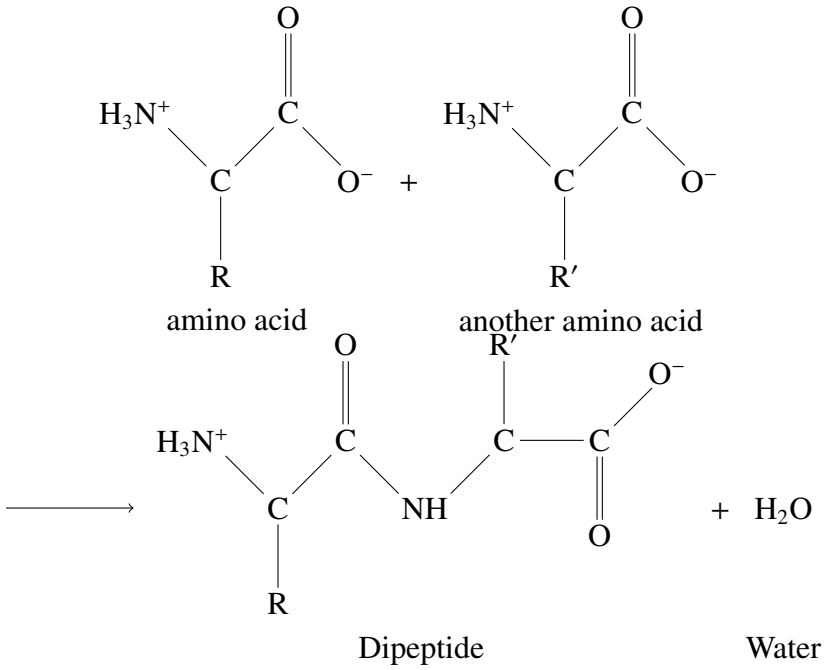
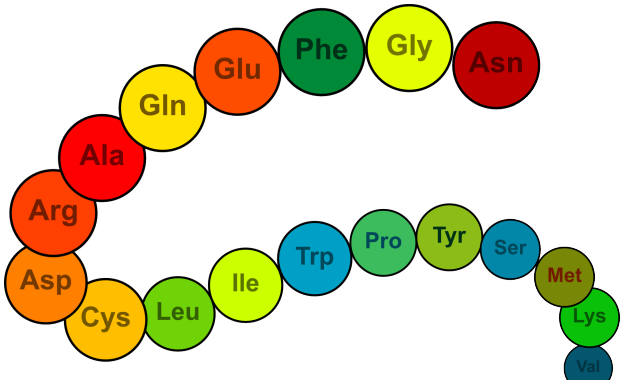


Figure 6.11: Formation of a dipeptide. When thousands of amino acids connect as in here a protein is formed [15].

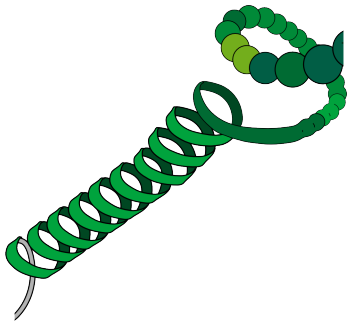
collinear manner. Inspired by the α-helix of protein structures (figure 6.12(b)), the non-collinear distribution of relays is achieved in this section by placing the nodes on a 2D zigzag, the 2D projection of the α-helix. Since proteins do not communicate with each other before applying any force, and they apply force only to nodes that are in their range, the protein approach is an efficient candidate able to provide a decentralized solution to the connectivity maintenance problem.

6.3.1 Protein Background

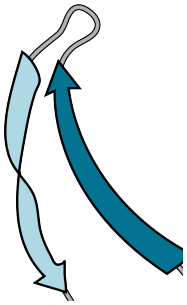
Proteins are made of amino acid residues (figure 6.10) connected by strong peptide bonds (figure 6.11), which form between two amino acids by one denoting its C=O



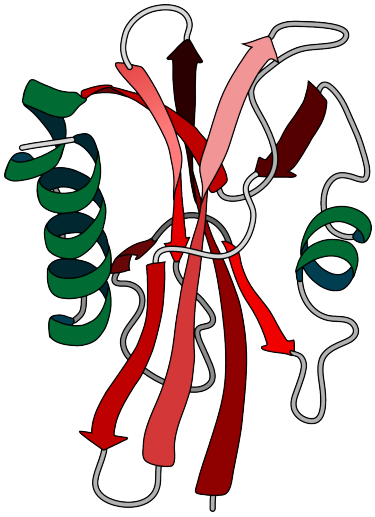
(a) Primary structure



(b) Secondary structure, α-helix



(c) Secondary structure, β-sheet



(d) Tertiary structure, β-sheet

Figure 6.12: Primary, secondary and tertiary structures of a protein molecule [2].

from its carboxyl group and the other its NH from its amino group ([11]). Since each amino acid has both a carboxyl and an amino group it is able to connect to two amino acids. In this way amino acids are able to make very long chains, which are called proteins. The peptide bonds create a structure called the primary structure (6.12(a)). Proteins having initially only the primary structure look like unfolded long chains.

Proteins also have a secondary structure which forms by hydrogen bonding between the amide N-H and carboxyl C=O groups along the backbone (primary structure). Depending on the alignment of the protein segments either helix (figure 6.12(b)) or sheet (figure 6.12(c)) like structures are observed, which are the most common secondary structures. Since only the helix like structure is stretchy and resembles the structure we are after, we are interested only with the helix like structure. The α-helix structure is like a "coiled spring" and forms by hydrogen bonding between the N-H group of one residue and C=O group of another residue 4 residues further along the chain.

In addition to the primary and secondary structures proteins have a tertiary structure (fig 6.12(d))which determines their 3-D shape. This structure is formed by the folding of the secondary structures on themselves. The primary structure determines how the secondary and tertiary structures will be. However the environment of the protein also affects the tertiary structure. For example in an aqueous environment the proteins fold in a way to keep nonpolar side chains to the inside, but polar side chains to the outside, so that polar side chains on the outer surface interact with water.

6.3.2 Method

In this work, the analogy between the protein and our proposed architecture is as follows. We assume that each mobile relay corresponds to an amino acid residue, and is able to make two strong connections to other mobile relays, like peptide bonds formed by the exchange of N-H and C=O groups. The leader and the sink are also assumed to be amino acid residues but they are the beginning and end residues of the poly-peptide chain (protein). We also assume that each mobile relay makes a weak connection to another relay a few relays further in the chain, like the weak hydrogen bonds between the residues that form the secondary structure. Normally in proteins

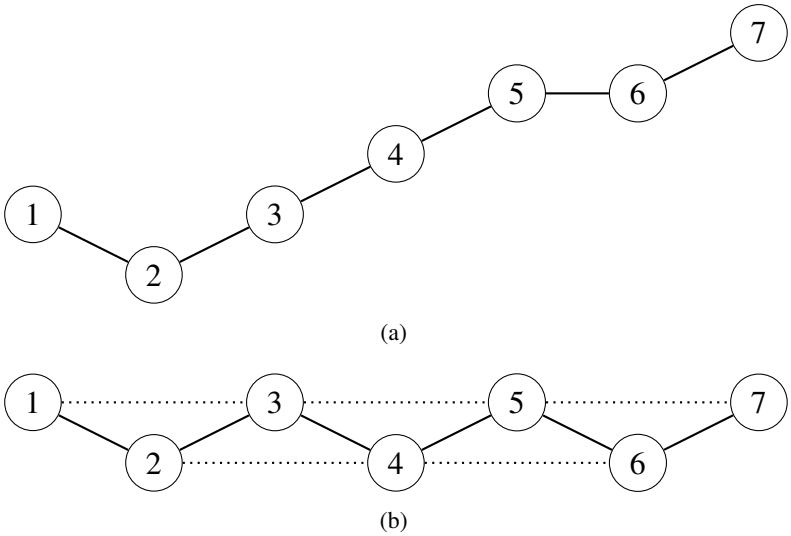


Figure 6.13: Relays under the influence of the primary structure only (a), and under the influence of both the primary and secondary structures of the protein (b). The lines indicate primary connections, whereas dotted lines indicate secondary connections.

the start and end residues are free to move, but in our relay chain we fix the sink end, and let the leader end move under own control commands. The analogy between our relay network and protein structure could be seen in figure 6.13.

To calculate the motion of the above mentioned system we assume that all interacting nodes are connected by a Hookean potential as is done also in the Elastic Network Model used for Normal Mode Analysis of globular protein structures [75]. So the weak forces forming the secondary structures are modeled as loose springs with small spring constants and the strong forces forming the backbone (primary structure) as stiff springs with large spring constants. This allows us to have a redundant network at the beginning. And as the leader moves farther away it allows us to stretch/unfold the chain, resulting in greater coverage,

In summary, the force on a robot j acting under the influence of the protein approach is:

$$F_j = \sum_{i \in S_{P_j}} -k^P(\|x_i - x_j\| - l_0^P)\frac{x_i - x_j}{\|x_i - x_j\|} + \sum_{i \in S_{S_j}} -k^S(\|x_i - x_j\| - l_0^S)\frac{x_i - x_j}{\|x_i - x_j\|} - b\dot{x}$$

S_P: Set of primary neighbors

S_S: Set of secondary neighbors

k^P: Spring constant of primary connections

k^S: Spring constant of secondary connections

l_0^P: Neutral spring length of primary connections

l_0^S: Neutral spring length of secondary connections

b: Friction coefficient x: Pose of the corresponding robot

Algorithm 8 Neighbor Selection for Protein

```
1:  procedure SelectNeighbors
2:      S_p ← 0                                  ▷ Initialize set of primary neighbors
3:      S_s ← 0                                  ▷ Initialize set of secondary neighbors
4:      S ← Query all neighbors                  ▷ Initialize set of all neighbors
5:      for all n ∈ S do
6:          if ||n.id − my.id|| == 1 then
7:              S_P ← S_P + n
8:          else if ||n.id − my.id|| == secondary hop count then
9:              S_S ← S_S + n
10:         end if
11:     end for
12: end procedure
```

Neighbor sets S_{P_j} and S_{S_j} are formed independently by each node according to algorithm 8, before each time the node calculates the virtual forces acting on it. The first step in the construction of the neighborhood sets is to scan the environment for all available relays. Then if a neighbors id is one more or one less than its own, it is added to S_p. If the difference of the ids is equal to the secondary hop count, which is the hop distance to the next residue to which a secondary connection is made, then the node is added to S_s. If neither criteria is satisfied by a node, then it is discarded.

6.3.3 Simulations

The simulations were run both in player/stage and a connectivity simulator developed during this thesis. Player/stage was not stable with a large number of robots, so all performance runs were done on my connectivity simulator. The simulator is a simple physics simulator that is able to move the robots under the influence of forces applied by different algorithms, as well as calculate connectivity, query neighbors and

calculate communication routes in a distributed manner.

The following parameters affect the final shape of a protein, and they should carefully be adjusted for proper shapes:

- Communication range
- Number of robots, or the density of robots in the communication range
- Neutral spring lengths of the springs in the primary structure
- Neutral spring lengths of the secondary structure
- Spring constants of the springs in the primary structure
- Spring constants of the springs in the secondary structure
- Maximum velocity of the nodes
- Dynamic Friction coefficient of nodes

Player/Stage Results

The steady state network configurations that are converged to, as well as the initial network configuration are given as the leader is moved on the horizontal axis using the mouse. We start with the following initial structure for protein like modeling with secondary structures connecting every 2 and every 3 neighbors:

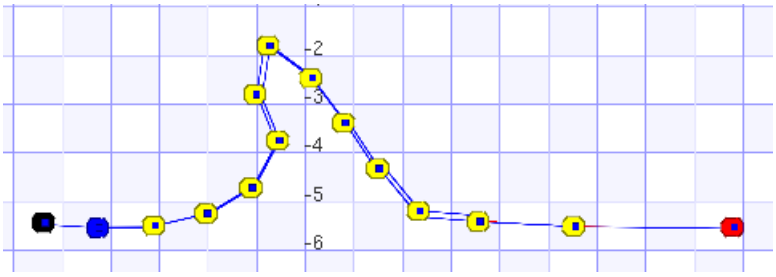


Figure 6.14: Screenshot of the starting configuration of the 2 neighbor hop count variation of the protein approach.

The black(leftmost) and red(rightmost) robots denote the sink and the leader respectively, whereas the other robots denote the relays. The lines between the robots denote connections of the primary structure.

If a mobile relay is let to make secondary connections to relays 2 nodes away we get a very nice zigzag like structure (figure 6.15):

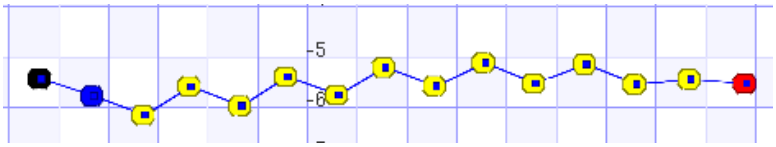


Figure 6.15: A steady state screenshot of the 2 neighbor hop count variation.

This structure could be preserved if the neutral lengths of primary and secondary spring connections are set inversely proportional to the number of robots.

If the leader is moving too fast the following transient shape (figure 6.16)is observed, but at steady state again the above shape (figure 6.15) is recovered.

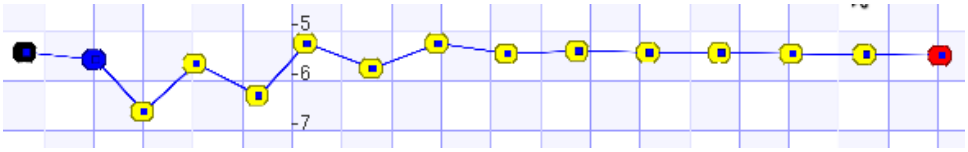


Figure 6.16: A transient state screenshot of the 2 neighbor hop count variation.

If a relay is asked to connect to relays 3 nodes away, depending on final pose of the leader the two different steady state shapes given in figures 6.17 and 6.18 were obtained:

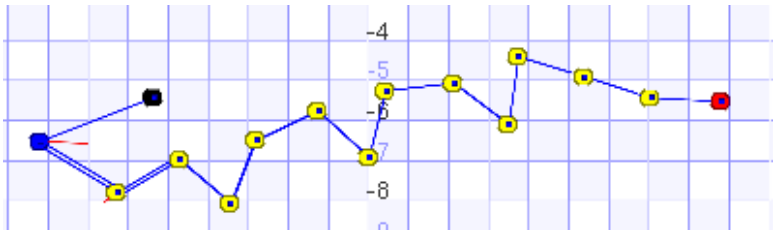


Figure 6.17: A steady state screenshot of the 3 neighbor hop count variation.

As could be seen in the figures, given enough time and properly selected parameters, we get a structure that looks more like a zig zag, which means a more redundant network deployment.

Although some desired shapes are obtained using the protein like approach, the method still requires improvements so that the nodes are uniformly distributed in the region between the sink and the leader, and they are also properly spaced.

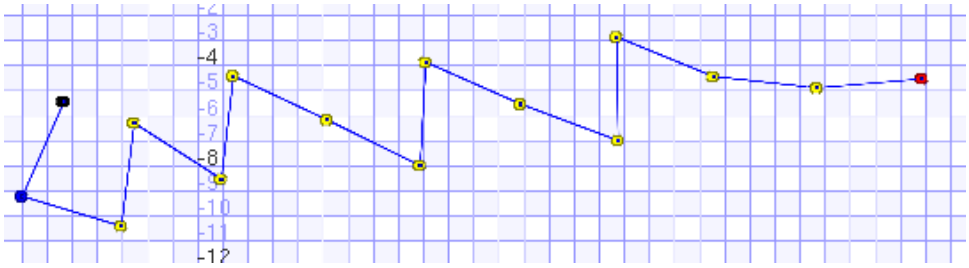


Figure 6.18: A steady state screenshot of the 3 neighbor hop count variation.

Performance runs

Simulations were run for two different speeds of the leader, 0.5m/s (1.8km/h) and 2.5m/s (9km/h). The relays were allowed a maximum speed of 2.78m/s(10km/h).

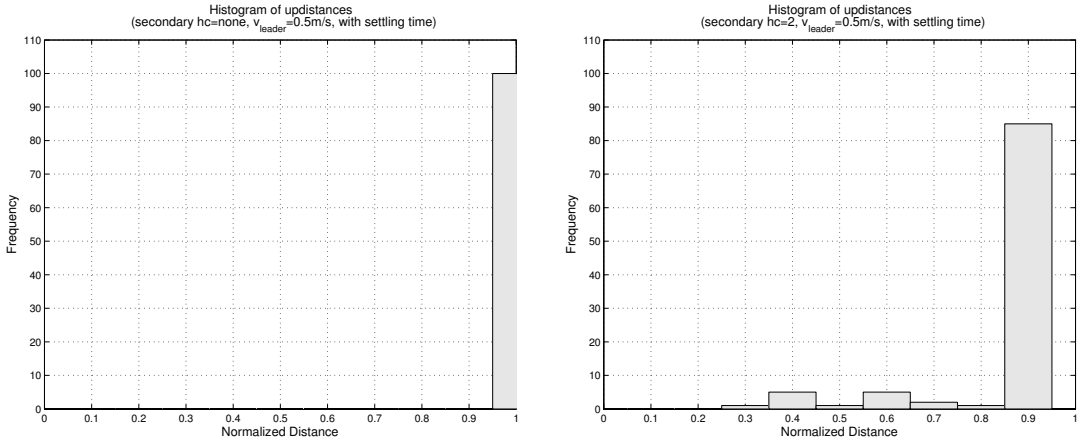
In this section we also give results for Go In Between algorithm in free space. The Go In Between algorithm is a base case, against which improvements could be compared. Also note that the protein approach with secondary spring constants set to zero reduces to Go In Between.

$v_{leader} = 0.5m/s$

As seen in figure 6.19(a), with a 0.5m/s leader velocity the base case is able to cover the whole maximum path achievable, connection is maintained till the last meter . However the protein algorithm does not perform that well in terms of updistance, though it reaches 0.9 (figures 6.19 b,c,d). The 2 neighbors hop count variation as could be seen in figure 6.19(b), has occasional early down-times, at around 0.3, 0.4. However as could be seen in figures 6.21(a), 6.22(a), this is the price paid for the sparseness of the network. As could be observed in 6.22(a) acute resistance for the two hop count variation is considerable smaller compared to the others, which could also be seen in figure 6.29, is due to the spread of the nodes compared to the other cases. The base case has an acute resistance of 18 since the nodes always keep a straight line, so there is no redundancy in the spread of the nodes. Variations with 3 or 4 neighbor hop counts are able to increase redundancy only slightly and only for the first half of the maximum coverable distance.

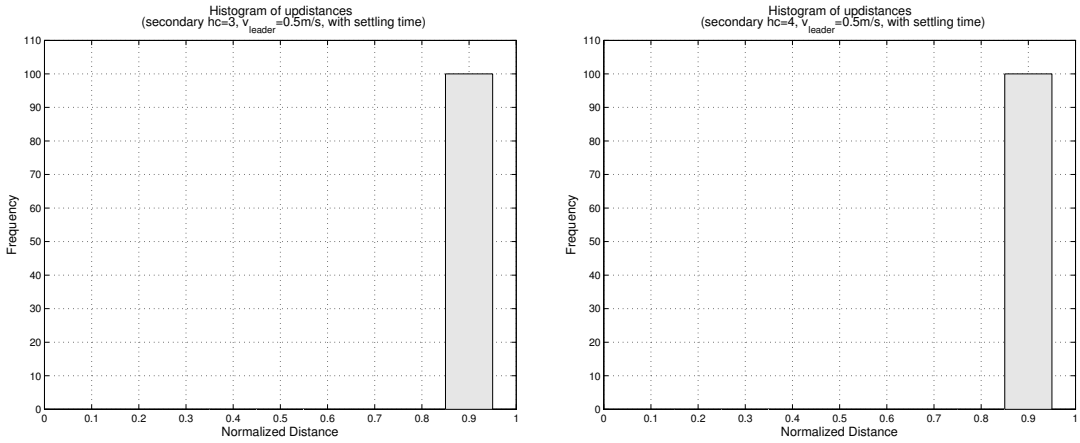
$v_{leader} = 2.5m/s$

Updistance performance of even the base case decreases (figure 6.23(a)) as the leader



(a) Base case, Go In Between

(b) Secondary structure's hop count is 2



(c) Secondary structure's hop count is 3

(d) Secondary structure's hop count is 4

Figure 6.19: Updistance histograms for the base case where there are no secondary connections and protein cases where each relay makes secondary connections to relays 2, 3 and 4 nodes away given in a, b, c, and d respectively. Leader robot's velocity is 0.5m/s.

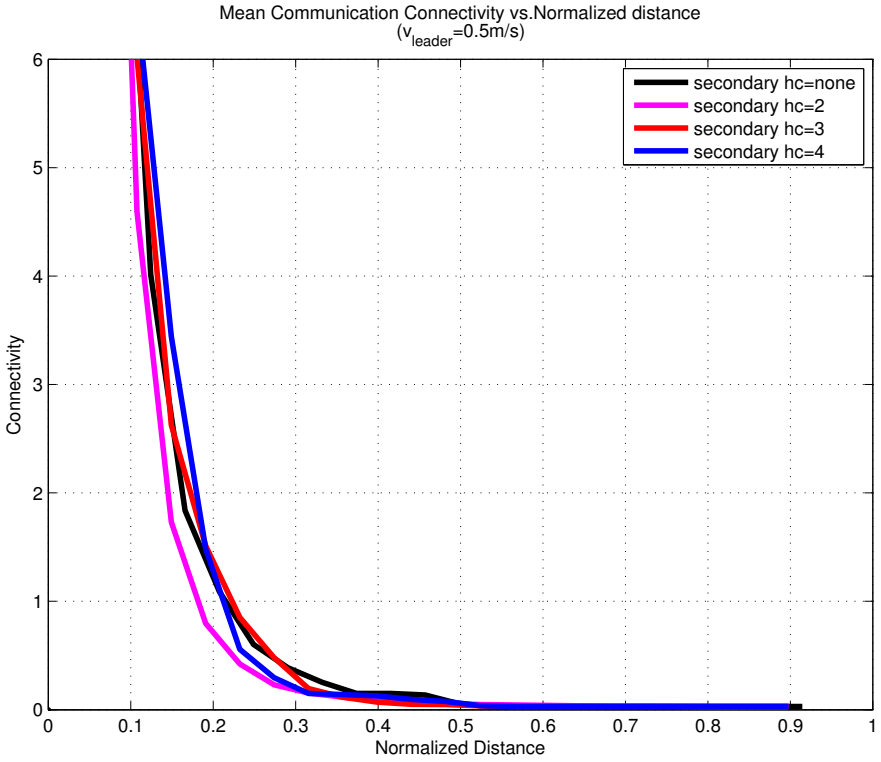
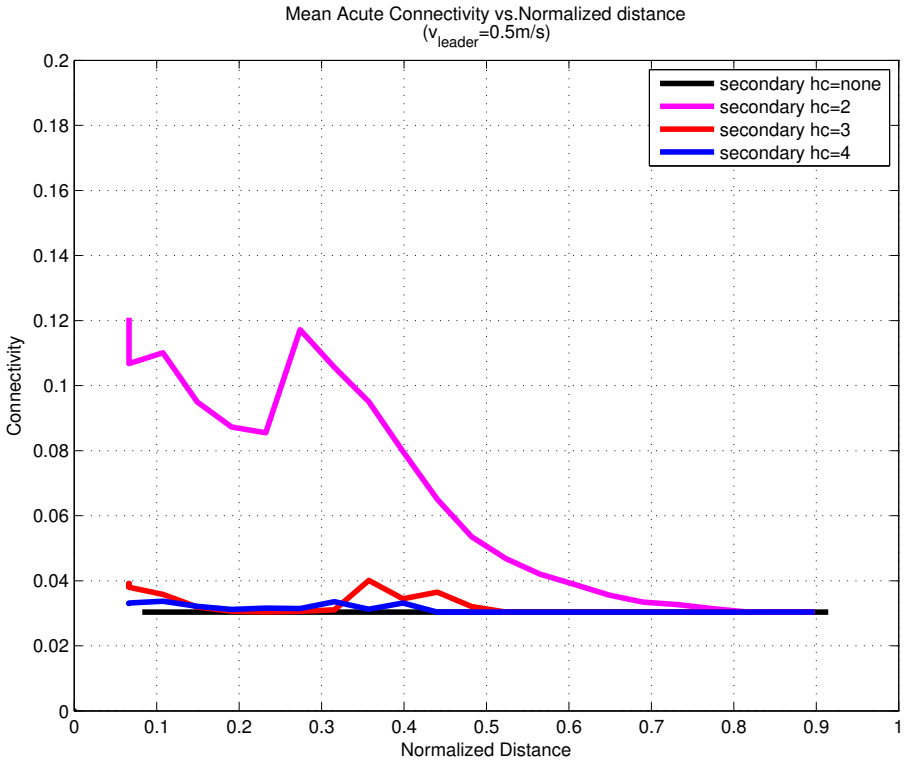
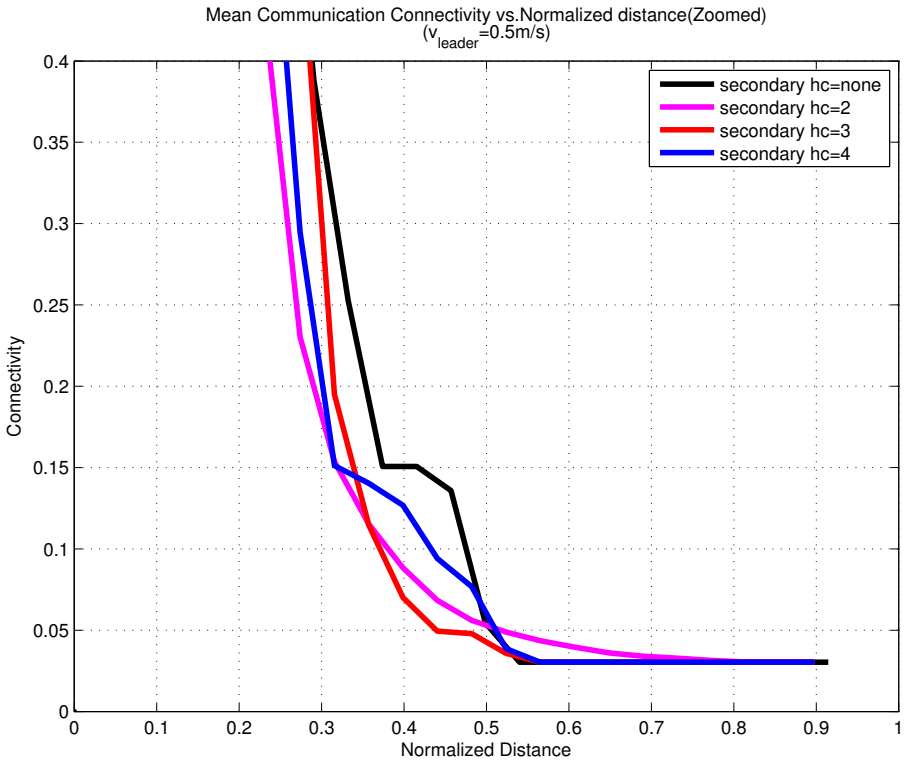


Figure 6.20: μ_{mean}, mean algebraic connectivity of the base and protein structures for being in RF communication range of each other, when the leader as traveling at 0.5m/s.

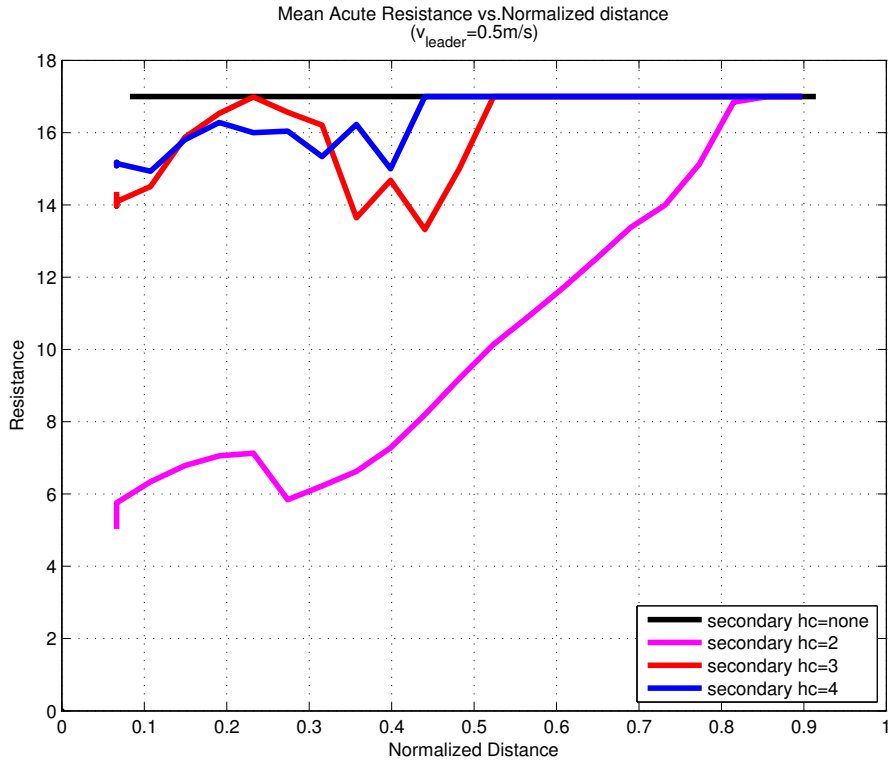


(a)

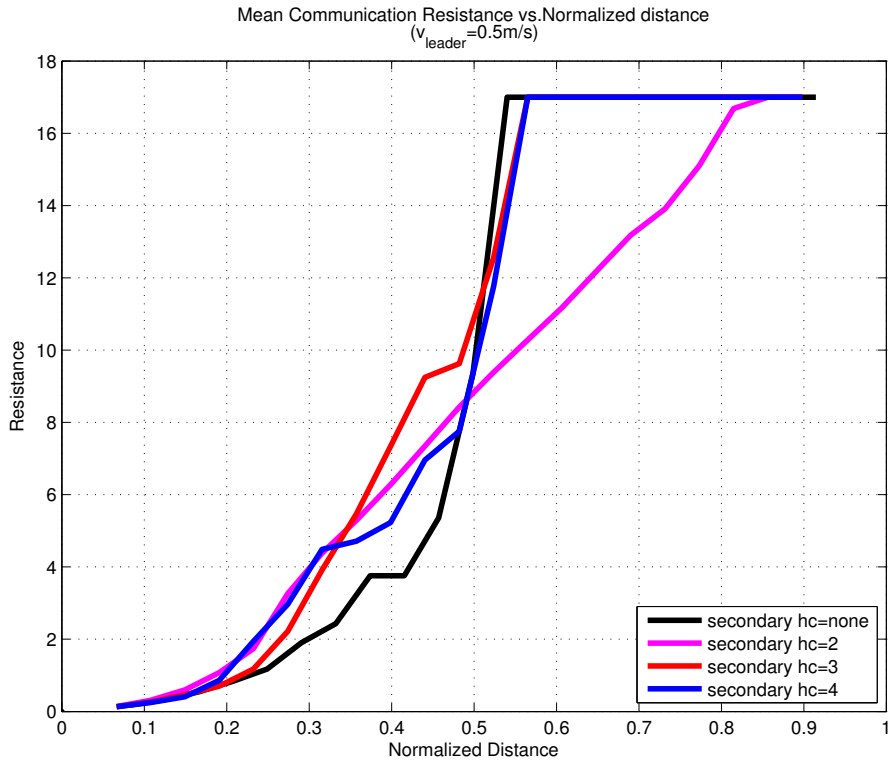


(b)

Figure 6.21: μ_{mean}, mean algebraic connectivity of the base and protein structures for the graph of virtual forces (a), and zoomed version of being in RF communication range of each other for the base and protein structures(b), when the leader as traveling at 0.5m/s.



(a)



(b)

Figure 6.22: $R_{eq_{mean}}$, mean equivalent resistance for the Protein Graph of virtual forces (a), and of being in RF communication range of each other (b), when the leader as traveling at 0.5m/s.

is set to travel at a speed of 2.5m/s. Now the maximum updistance achievable is 0.7 units for the base case. This drop in performance is due to the limited speed of the relays, 2.78m/s, which do not have the chance to detect and react to the changes sufficiently fast. Note that nodes away from the leader do not know its speed and the only way for a node to detect it is through its neighbor, which notices it through another neighbor. In short, it takes time till the velocity propagates through the whole network, and since the nodes are already not fast enough to overcompensate the time delay the connection is broken sooner. And again the 2 neighbors hop count for secondary connections has better deployment performance(figures 6.25(a), and 6.26(a)), but this time at a higher price (figure 6.23(b)), the leader is not allowed to cover more than 0.5 units of distance, where other variations as well as the base case can cover upto 0.7 units. Note that RF connectivity is adversely affected too (figures 6.24, 6.25(b) and 6.26(b))

Time Evolution of Protein Structures

To understand better how the algorithm works and as guidance for improvements few screenshots of the algorithm are given in figures 6.27, 6.28, 6.29, 6.30. In figures 6.27, 6.28 time evolution of the base case is given for the two different leader velocities studied. In both figures the starting states are close to a line, and as the leader moves away the shape turns to a line. Due to the mechanism in effect changes at the leader need some time to arrive at the base station. Therefore a node that is closest to the leader will be pulled first, then it will pull its neighbor, and so on, till the sink is reached. Due to this time delay spring lengths vary from the leader to the sink, being longest at the leader. The difference between the lengths becomes larger as the leader moves faster, compare 6.27 (d) and 6.28 (d) for example. This goes on till the distance between the leader and its neighbor relay becomes large enough to make them out of range, breaking connectivity of the network.

In figures 6.29 and 6.30 we see how the 2 hop count protein approach works. The graphs are acute triangle graphs and are intended to show the distribution of nodes as time goes on. As seen in the figures the 2 hop count protein algoritm is able to keep the network redundant, and when the leader is slow, it is able to transfer itself to the base case, utilizing full potential of the network for connectivity. However, its

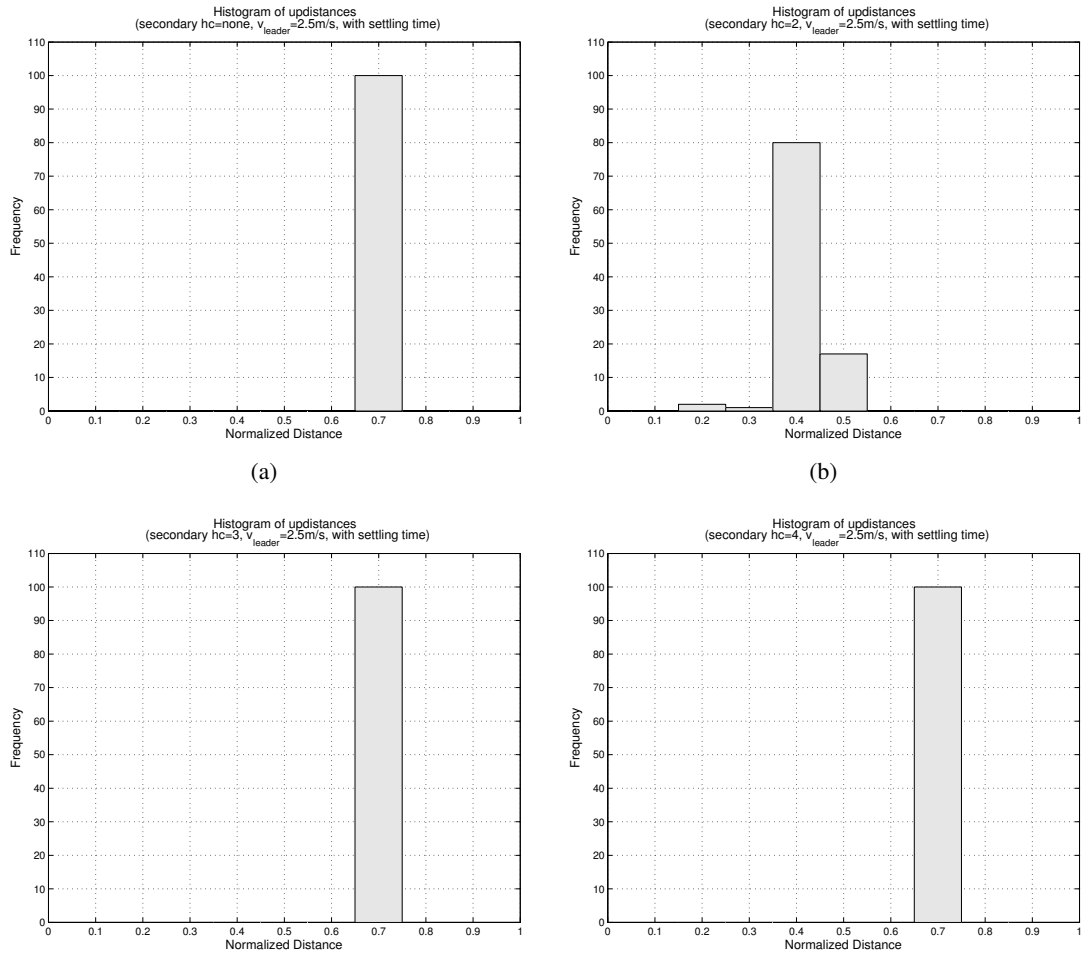
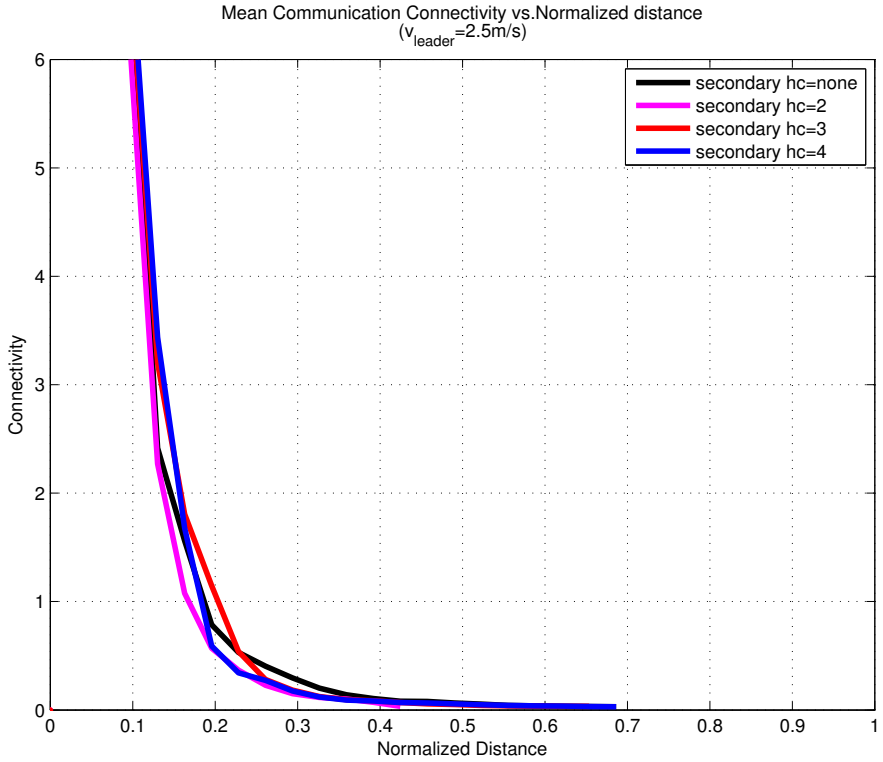


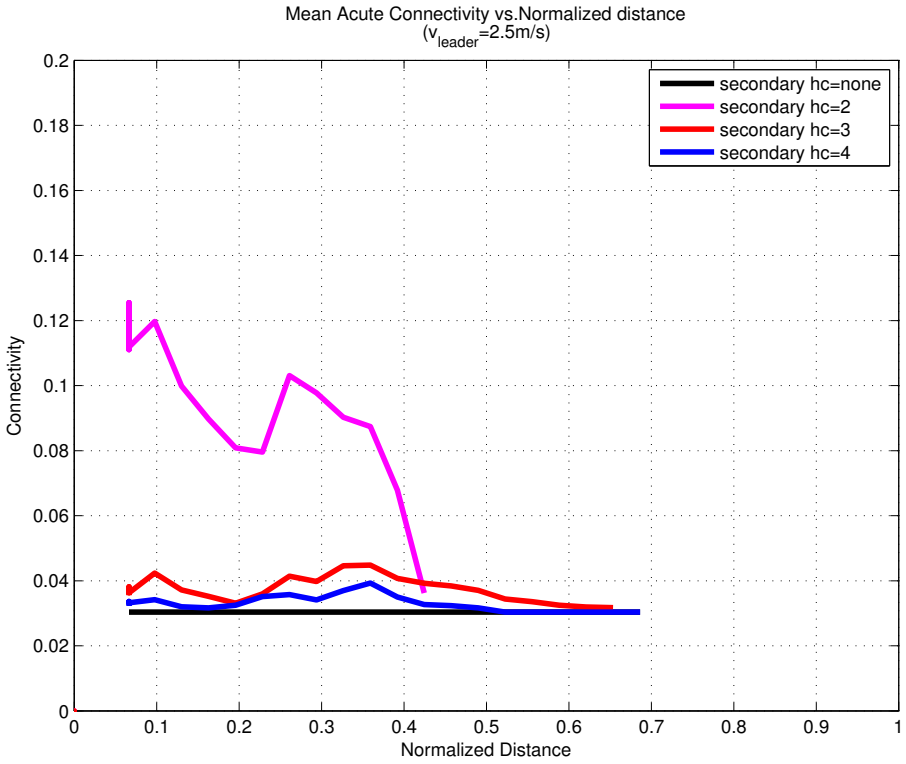
Figure 6.23: Updistance histograms for the base case where there are no secondary connections and protein cases where each relay makes secondary connections to relays 2, 3 and 4 nodes away given in a, b, c, and d respectively. Leader robot's velocity is 2.5m/s.

performance is not that good when it comes to a fast moving leader, the network is disconnected at a quite early stage.

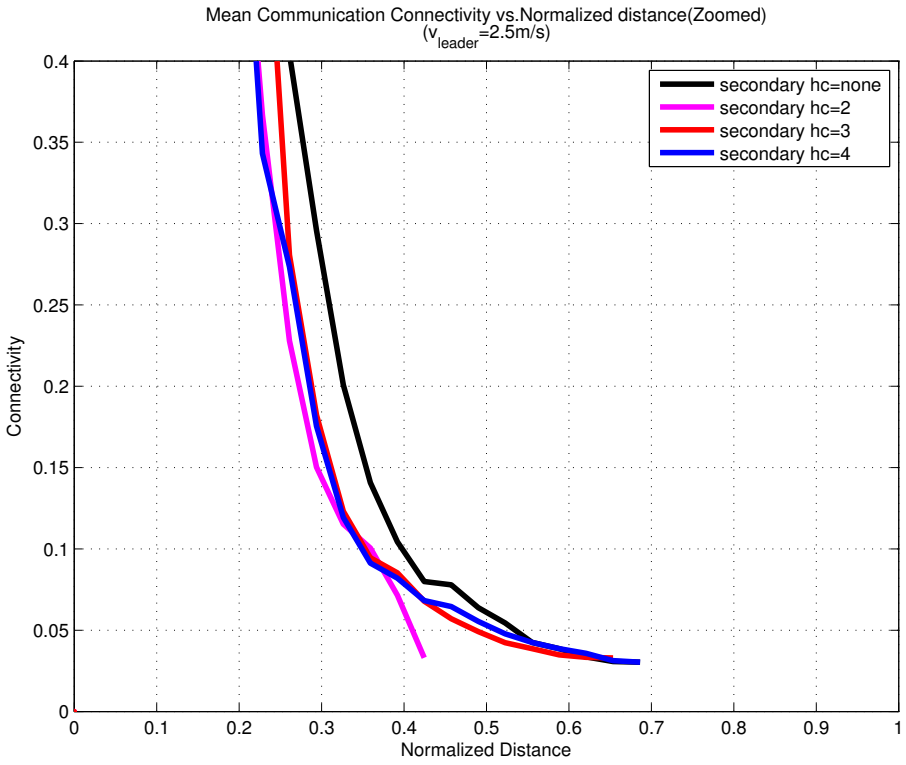


(a)

Figure 6.24: μ_{mean}, mean algebraic connectivity for being in RF communication range of each other, when the leader as traveling at 2.5m/s.

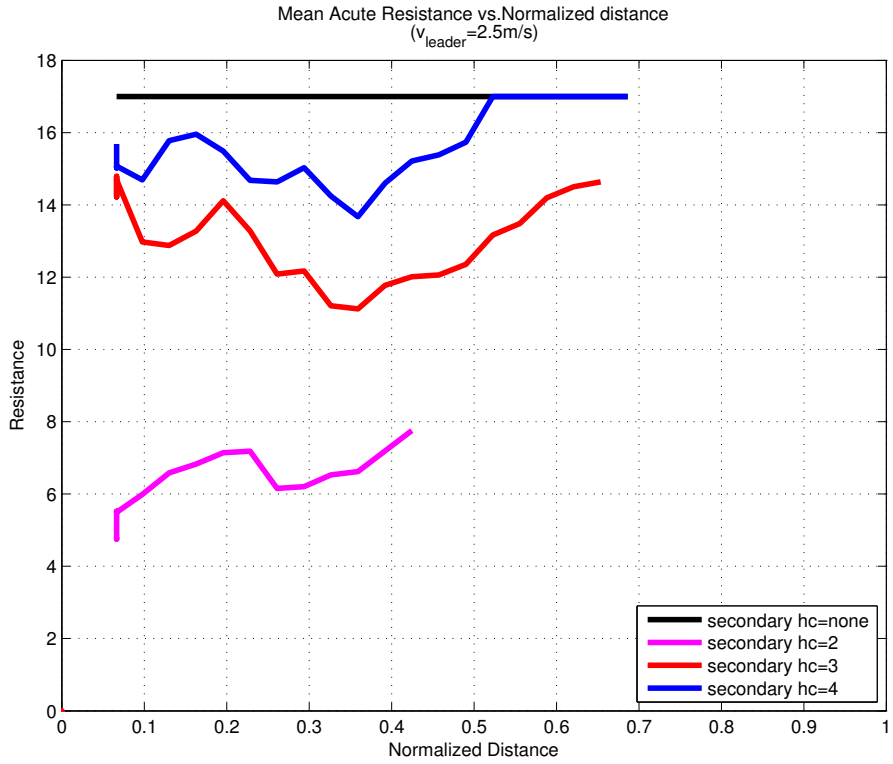


(a)

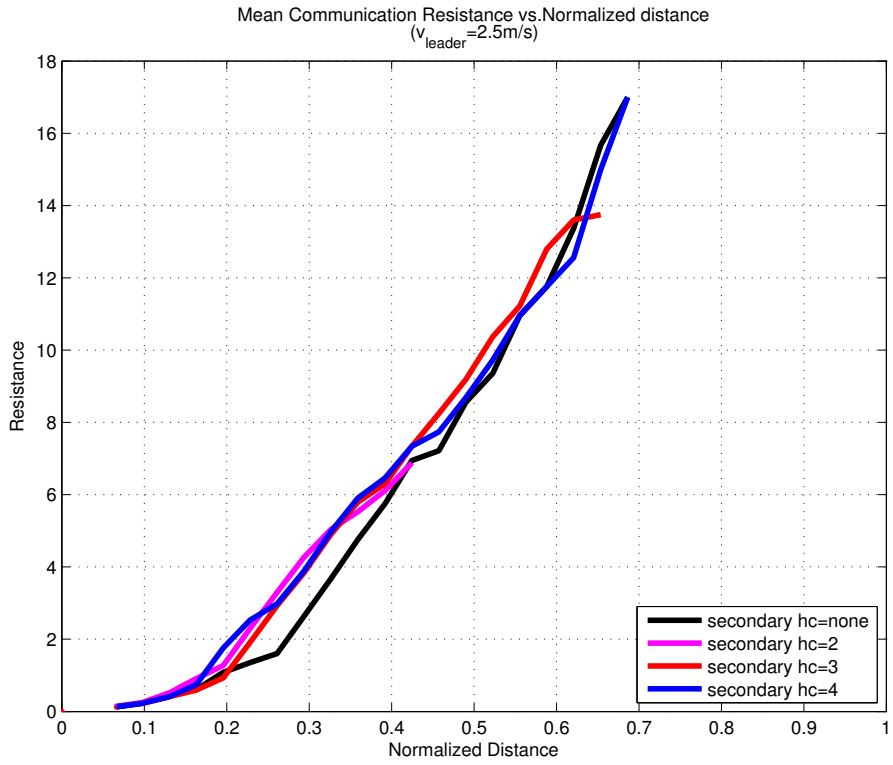


(b)

Figure 6.25: μ_{mean}, mean algebraic connectivity for the Protein Graph of virtual forces (a), and zoomed version of being in RF communication range of each other (b), when the leader as traveling at 2.5m/s.

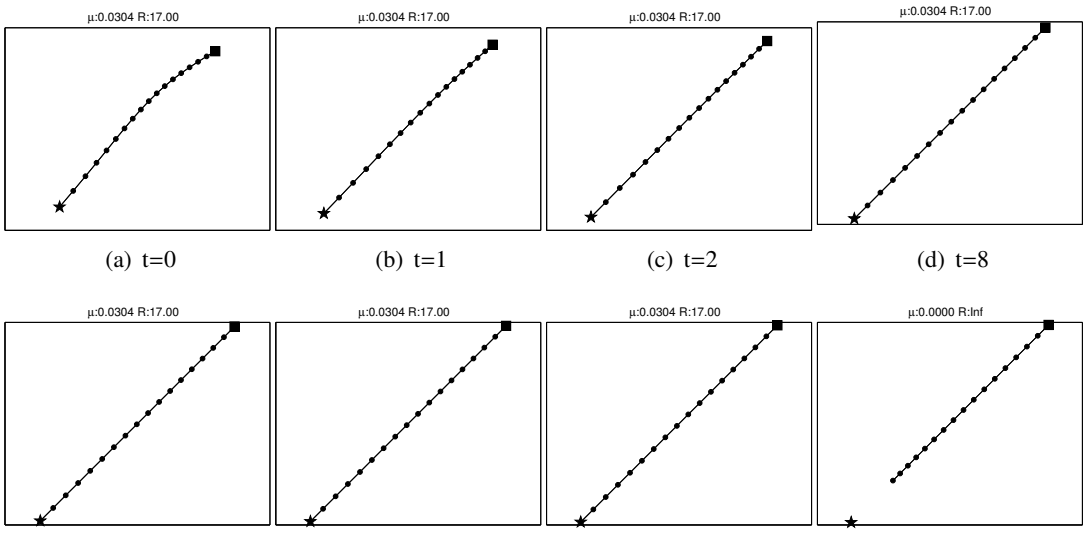


(a)



(b)

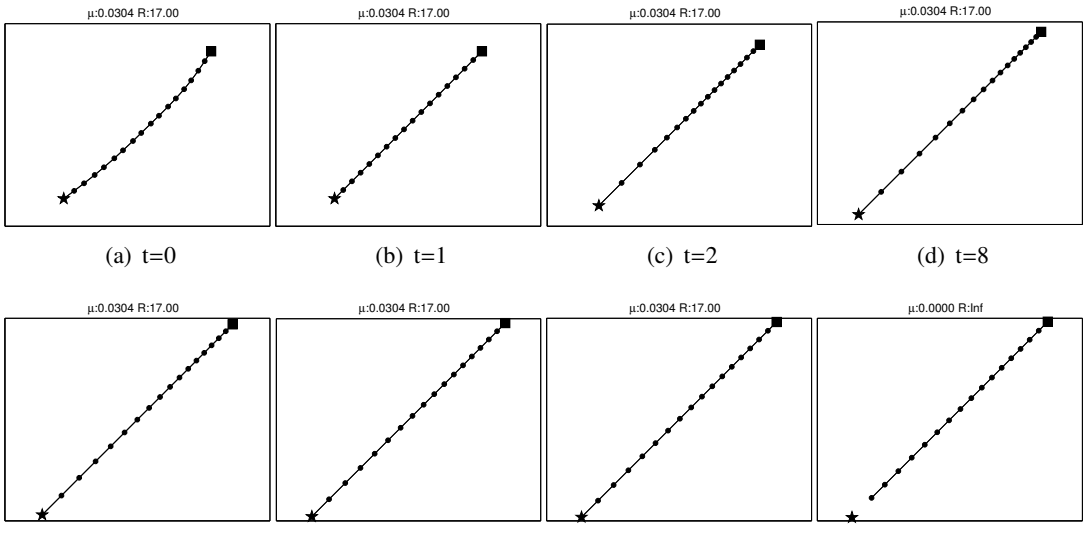
Figure 6.26: $R_{eq_{mean}}$, mean equivalent resistance for the Protein Graph of virtual forces (a), and of being in RF communication range of each other (b), when the leader as traveling at 2.5m/s.



(a) t=0 (b) t=1 (c) t=2 (d) t=8

(e) t=12 (f) t=15 (g) t=18 (h) t=21

Figure 6.27: Screenshots taken at different times of the base protein algorithm as the leader moves at $v_{leader} = 0.5m/s$, The robots start with the initial configuration given in (a), and connection is lost in (h). The square indicates the sink, the star the leader, the black dots other mobile relays. The lines are the connections between the robots.



(a) t=0 (b) t=1 (c) t=2 (d) t=8

(e) t=12 (f) t=15 (g) t=18 (h) t=21

Figure 6.28: Screenshots taken at different times of the base protein algorithm as the leader moves at $v_{leader} = 2.5m/s$, The robots start with the initial configuration given in (a), and connection is lost in (h). It can be observed that as the leader moves away, uniformity of the link lengths is lost (compare to 6.27). This happens because the leader is moving too fast for the network to stabilize. The square indicates the sink, the star the leader, the black dots other mobile relays. The lines are the connections between the robots.

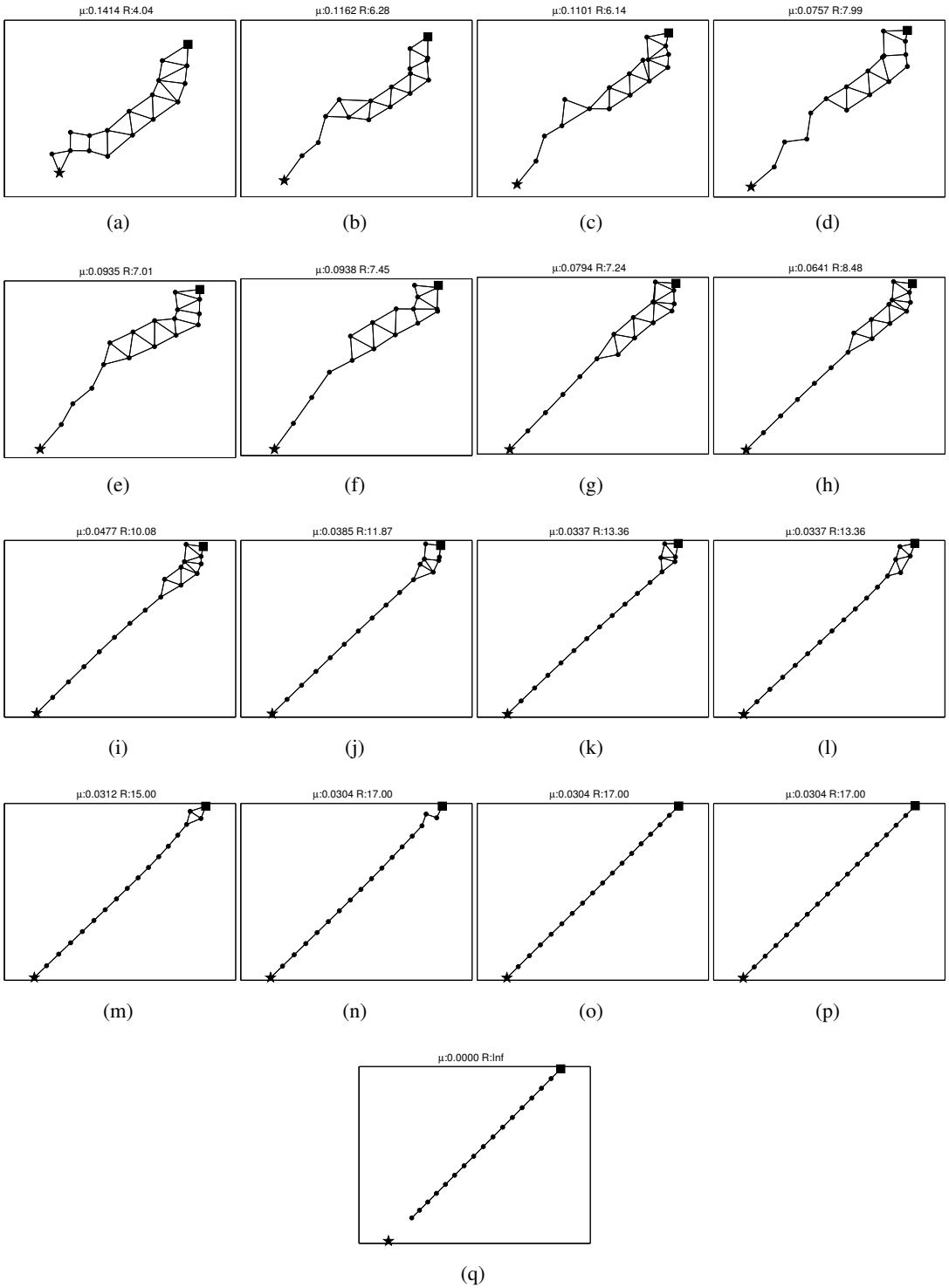


Figure 6.29: Screenshots taken at different times of the 2 hop count protein algorithm as the leader moves at $v_{leader} = 0.5m/s$, The robots start with the initial configuration given in (a), and connection is lost in (q). The square indicates the sink, the star the leader, the black dots other mobile relays. The lines are the connections between the robots.

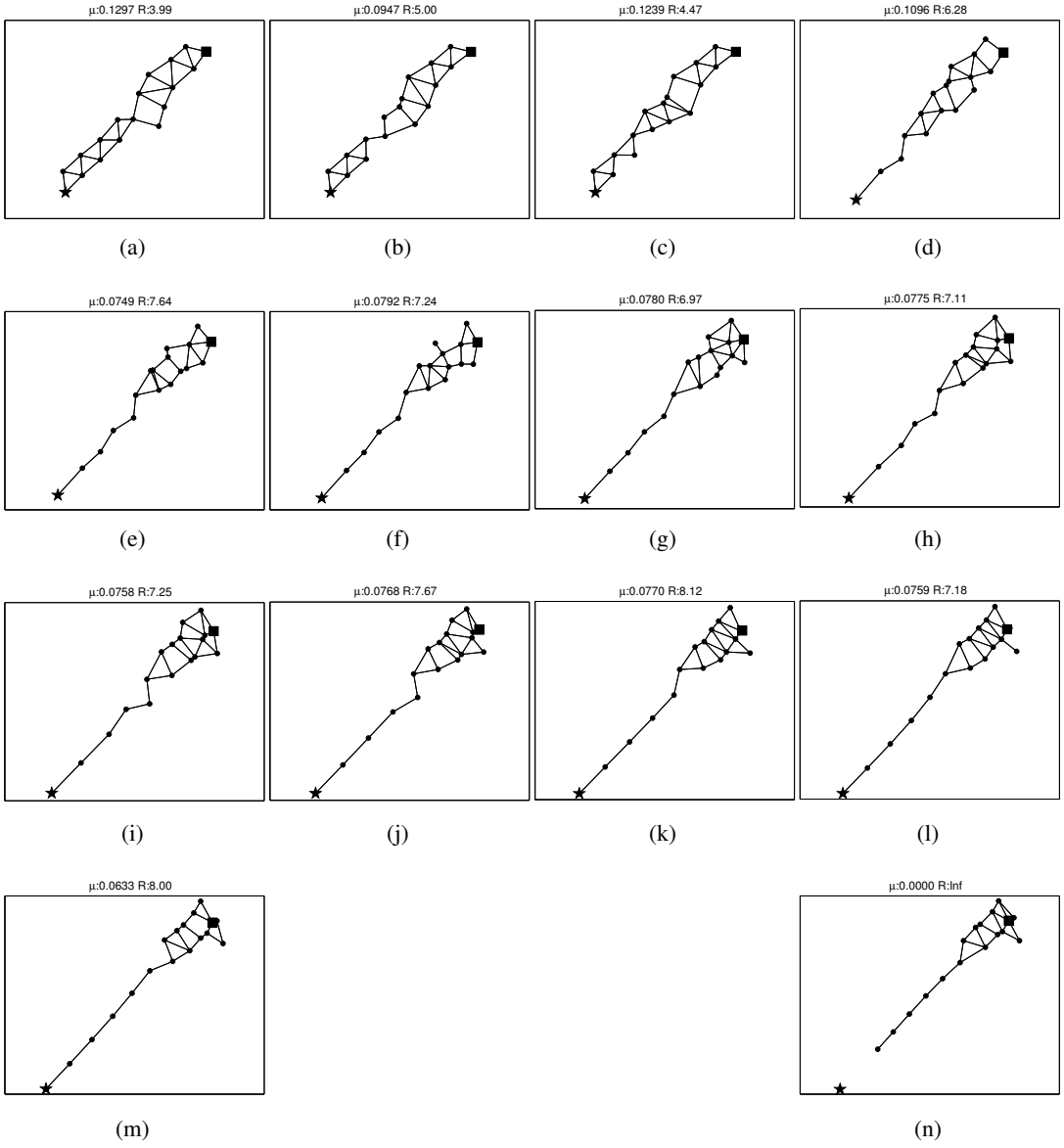


Figure 6.30: Screenshots taken at different times of the 2 hop count protein algorithm as the leader moves at $v_{leader} = 2.5m/s$, The robots start with the initial configuration given in (a), and connection is lost in (n). The network does not have enough time to adept to the changes induced by the high speed of the leader (compare to 6.29). The square indicates the sink, the star the leader, the black dots other mobile relays. The lines are the connections between the robots.

6.4 Gabriel Virtual Force Graphs

Another way to improve robustness and redundancy of a mobile relay network is to cover the available space uniformly with the available relays, without loosing connectivity. For this the relays should be somehow pushed or directed to proper places, and this should be done in a decentralized manner. If the nodes are connected to each other using virtual springs where the neighbors to which they connect are chosen in such a way that the resulting global graph is a Gabriel Graph, the network eventually becomes uniformly distributed. And depending on the parameters used it can become sparsely distributed. This graph is also referred as Gabriel Virtual Force Graph (GVFG) in the text.

6.4.1 Method

Algorithm 9 Neighbor selection to create a Gabriel Graph

```
1:  procedure SELECT NEIGHBORS
2:      S ← Query all neighbors                    ▷ Initialize set of all neighbors
3:      S_2 ← S                                    ▷ Copy of all neighbors
4:      S_a ← 0                                    ▷ Initialize set of acute neighbors
5:      for all n ∈ S do
6:          x ← (n.pose + my.pose)/2
7:          r ← my.distanceTo(n)/2
8:          for all nn ∈ S_2 do
9:              d ← x.distanceTo(nn)
10:             if d < r then
11:                 flag ← true
12:             end if
13:         end for
14:         if flag then
15:             S_a ← S_a + n
16:         end if
17:     end for
18: end procedure
```

A mobile robot j in a GVFG moves under the influence of the following force, which is the total spring force acting on it through its neighbors, selected according to algorithm 9, minus the damping coefficient.

$$F_j = \sum_{i \in S_j} -k(\|x_i - x_j\| - l_0)\frac{x_i - x_j}{\|x_i - x_j\|} \; - \; b\dot{x}$$

S_j: Set of neighbors forming an acute triangle with the current node

The neighborhood algorithm (algorithm 9) is straightforward and is run periodically and independently by each node, just before calculating the force acting on it. The first step in the neighborhood algorithm is to find all the neighbors that are in range. After these neighbors are found, it is checked whether a link between the robot and its neighbor could be one side of an acute triangle. This check is easily done by finding if there is any other node in the circle whose diameter is the link in question. If there is any, then this link could not be one side of an acute triangle in the graph and therefore it is discarded, else it is added to the set S.

Assuming that all the nodes are close enough to form a connected graph, acute triangulation of the workspace will create a connected Graph. This graph is not uniform at the beginning. However due to the virtual springs connecting the nodes, the graph converges soon to a steady state with a uniform distribution of the nodes. This uniform distribution is going to both help equi-distribute the load of communication and add redundancy to the system. Of course this happens when the nodes are initially dense enough. If they are not dense enough, we may get a graph that does not cover the space uniformly. The connected crowded distribution of nodes is usually what we can expect at the beginning of a mission, where all the relays and the sycophant are around the base station. However as the leader (sycophant) moves away from the base station, due to restricted communication range, the robots ability to keep a highly connected network will degrade and we will end with a chain, and if the range and number of relays are not sufficient we will loose connectivity between the leader and the base station. Aim of a connectivity maintenance algorithm should be to keep the system connected and redundant as much as (in terms of redundant paths, and in terms of algebraic connectivity) and as long as (in terms of distance) possible.

Gabriel Graphs have the advantage that in a naive implementation they do not require communication between the nodes, so the network does not have to reach a consensus, can act quickly and allocate the available network bandwidth to relay information from the sycophant as well as themselves instead of wasting it for formation control. Neighborhood selection, and the forces nodes apply to each other are all symmetric. The spring constants, the damping coefficients as well as the neutral lengths of the connecting springs determine how fast the system reacts, how stable it is and for how long robustness and redundancy are kept. Yet another parameter not evident in the equations but still in effect is communication interval. Although having continuous communication is the best, it is a power-hungry process and may be unwanted in some mission since it would reveal locations of the nodes.

In the Gabriel Graph algorithm, spring constants, damping coefficients as well as the neutral lengths of the springs do not have to be the same over all the graph or over all times, and they have been varied in the simulations. Some variations degraded the performance of the network, while some improved. We got different results for different initial distribution of the relays and therefore when judging the performance I had to run the algorithms many times. Performance was presented in terms of mean connectivity, mean resistance and updistance.

The spring constant parameters determines how fast the system reaches a steady state or how fast it responds to change in the configuration of the system. A high k may even result in an oscillatory system, whereas a very small one means a sluggish system.

The neutral spring length of the springs determines how spread the relays are. A small l_0 means a dense system whereas a large one means a sparse system. And although a network that is too sparse is not preferred, a sparse system is able to keep communication longer compared to a dense one. l_0 was varied geographically, i.e. depending on the neighborhood of the relays to critical nodes, namely the sycophant, the sink and nodes that have a single connection. In such critical places if nodes are loosely connected the connection may easily break, so by reducing l_0 in these areas we are able to both make those links stronger and increase the chance that new links are formed here, avoiding a disconnected graph. l_0 in these areas is called *critical* l_0

critical l_0	$0.5m/s$	$2.5m/s$
$l_0/1$	0.6564	0.3432
$l_0/2$	0.7292	0.4191
$l_0/4$	0.7920	0.5643
$l_0/8$	0.8128	0.5761
$l_0/16$	0.8244	0.6219
l_0/∞	0.8340	0.6003

Table 6.1: Mean values of updistance for different speeds of the leader

in this text.

6.4.2 Simulations

In order to have a more realistic simulation, the parameters were set with examples from the real world in mind. The communication range between the robots was taken to be 100m, which is the default maximum range of wireless devices (access points, modems etc.). The sycophant's speed was taken to be between $0.5m/s$ and $2.5m/s$, which is in range of a walking or running human leader. The relays' maximum velocity was fixed to $2.78m/s$ ($10km/h$).

$v_{leader} = 0.5m/s$

As could easily be seen in the updistance histograms given in figure 6.31, algorithms ability to keep up with a leader that is moving away gets better as critical l_0 decreases. When critical l_0 is the same as l_0, the algorithm fails sometimes even at a very close distance of 0.3 units (figure 6.31(a)), which is approximately the initial distance between the leader and the base station. As critical l_0 decreases the algorithm's early failure rate decreases and it's ability to cover longer distances improves as could be seen by comparing figures 6.31(a) and the others. In figure 6.31(a) maximum covered distance is 0.7 whereas it reaches 0.9 in figure 6.31(c) and afterwards. Comparing 6.31 c d e it is also observed that the smaller critical l_0 is the more items are accumulated towards 0.9.

Mean algebraic connectivity of the RF communication range could be seen in 6.32, and 6.33(b) where it is zoomed both to see more detail and easily compare with the algebraic connectivity for the Gabriel Virtual Force Graph, which is given in 6.33(a).

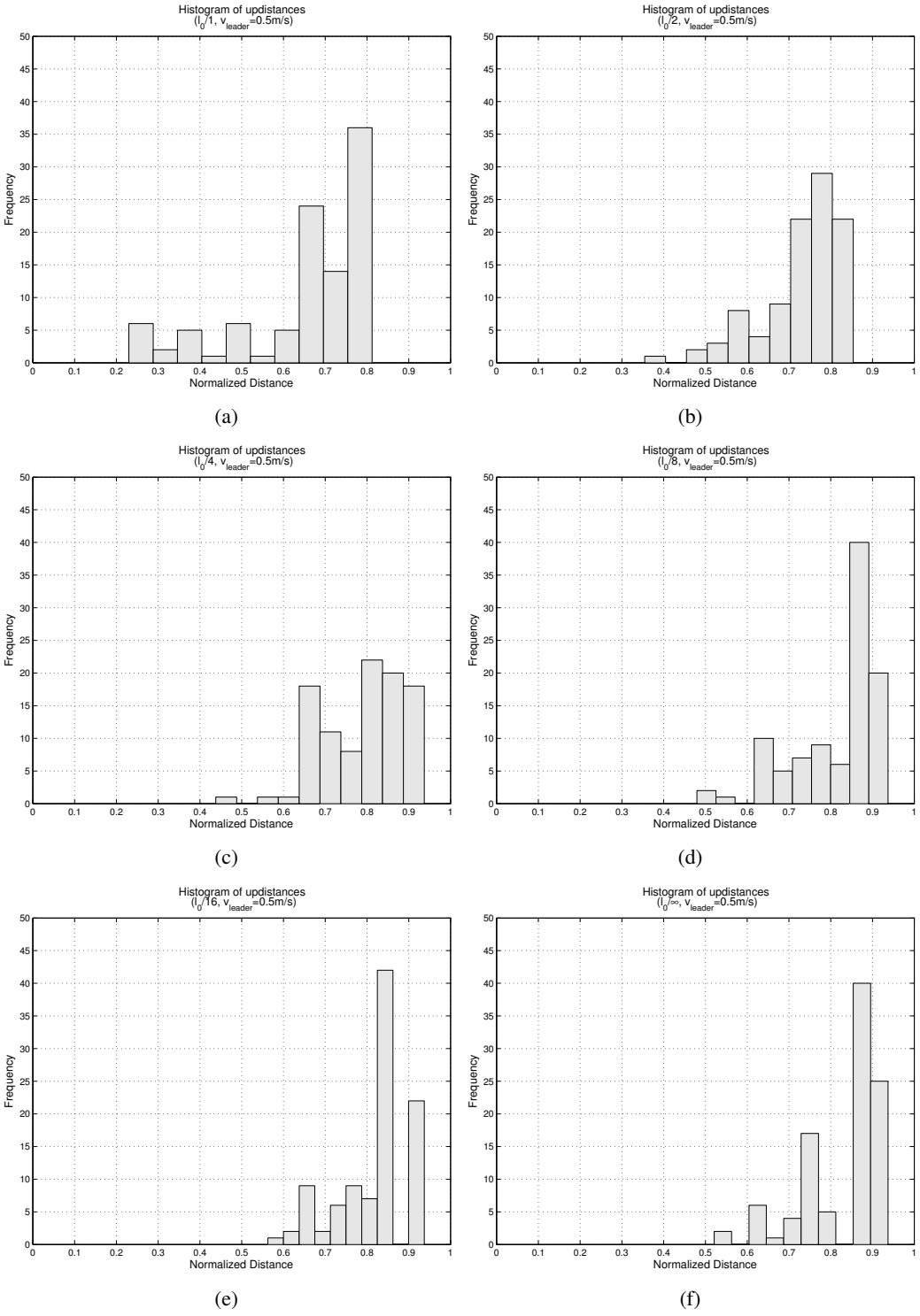


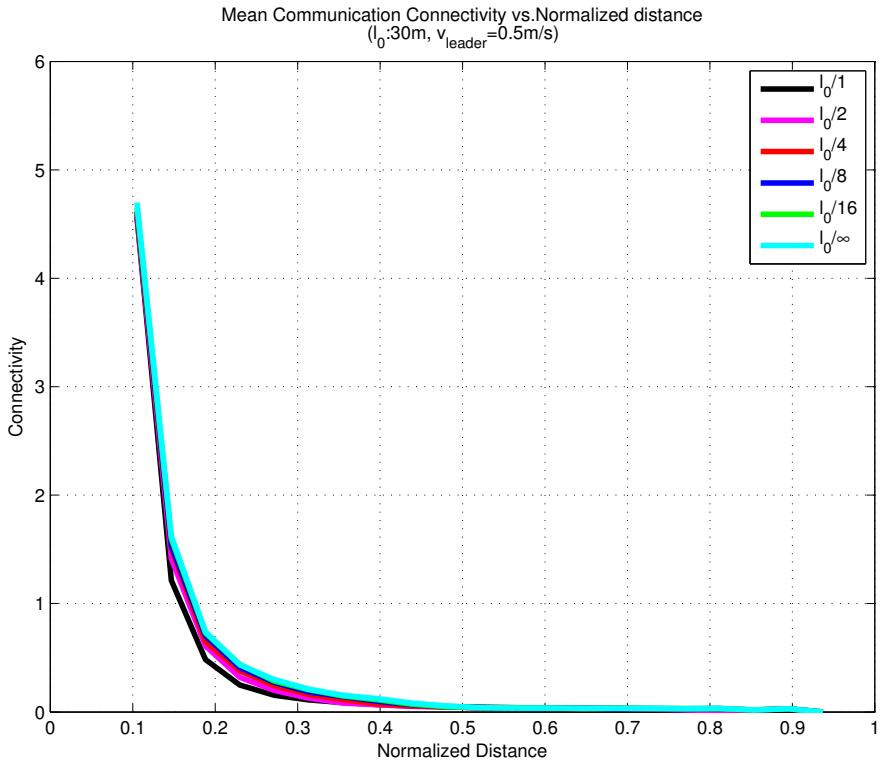
Figure 6.31: Updistance histograms for various critical l_0 values at critical junctions and fixed leader velocity of 0.5m/s.

Mean algebraic connectivities of the communication range for different critical l_0 values drop quickly as the leader moves away from the sink. For example the difference of connectivity values at 0.2 updistance for $l_0/1$ and l_0/∞ is 0.25 and this difference is almost zero at 0.5 updistance. Comparing mean communication and acute algebraic connectivities to each other (comparing figures 6.33(a) and 6.33(b)), although initially quite different, after a distance of approximately 0.6, they become quite alike. The initial difference is due to the fact that μ for the communication range is calculated by checking all neighbors in the RF communication range. However μ for the Gabriel graph is calculated only by taking into account neighbors to whom/by who force is applied, which is relatively fewer since many of the links do not satisfy the acute triangle condition. However as the distance between the leader and the sink gets larger, both the RF connectivity graph and the Gabriel virtual force graph become almost identical. This is because much fewer nodes are seeing each other (being in range of each other), and when they see each other they have to somehow connect, either forming an acute triangle or just two links one to the neighbor on the right, and the other to the one on the left, which still lets them be part of a Gabriel graph. The few that stay together, cause the slight difference between the algebraic connectivities.

As had been mentioned in the section on performance measures, there is a contradiction between the need for a better connected system and a sparse system, which could easily be seen in figure 6.34. Order of curves of acute resistance for normalized distance less than 0.4 units is the reverse of the order of curves of the RF resistance. In the acute resistance the curve corresponding to $l_0/1$ is on the top, whereas it is on the bottom for RF equivalent resistance, and the acute resistance at the bottom of the graph which corresponds to l_0/∞ goes to the top as RF resistance. However this changes after $\mu = 0.45$, because after a certain distance the relays have to get closer to the line connecting the leader and the sink to keep connected, increasing equivalent acute resistance. This could also be seen as convergence of the acute resistance to the RF resistance as the distance between the sink and the leader increases.

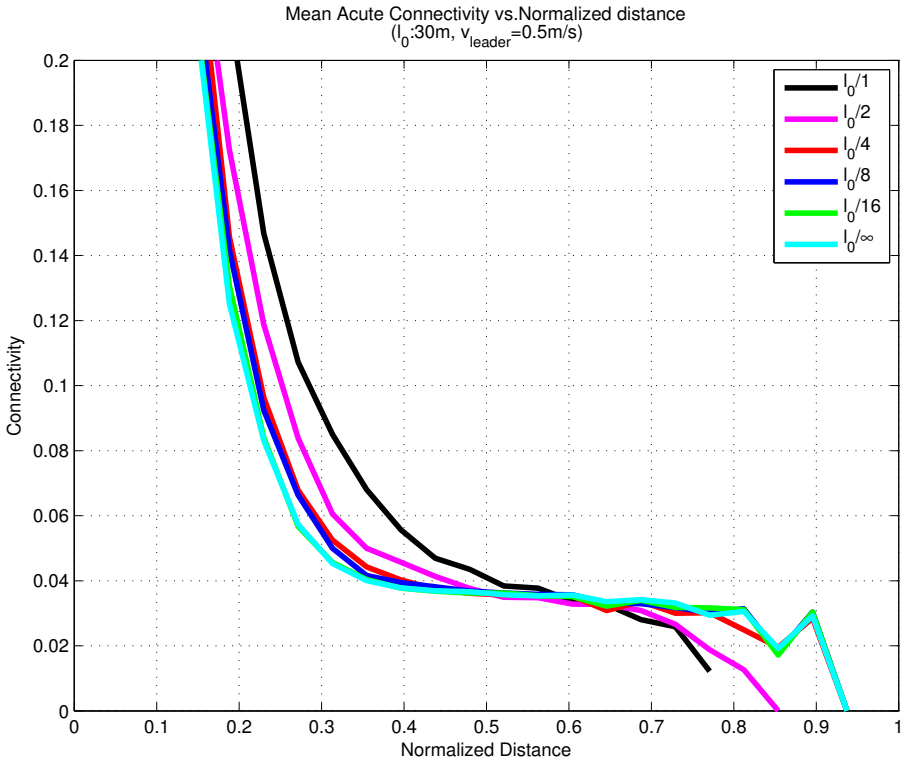
$v_{leader} = 2.5m/s$

Our observations for a different leader velocity ($2.5m/s$) are similar: the smaller the critical l_0 the better updistance we have, although the difference is not as obvious as

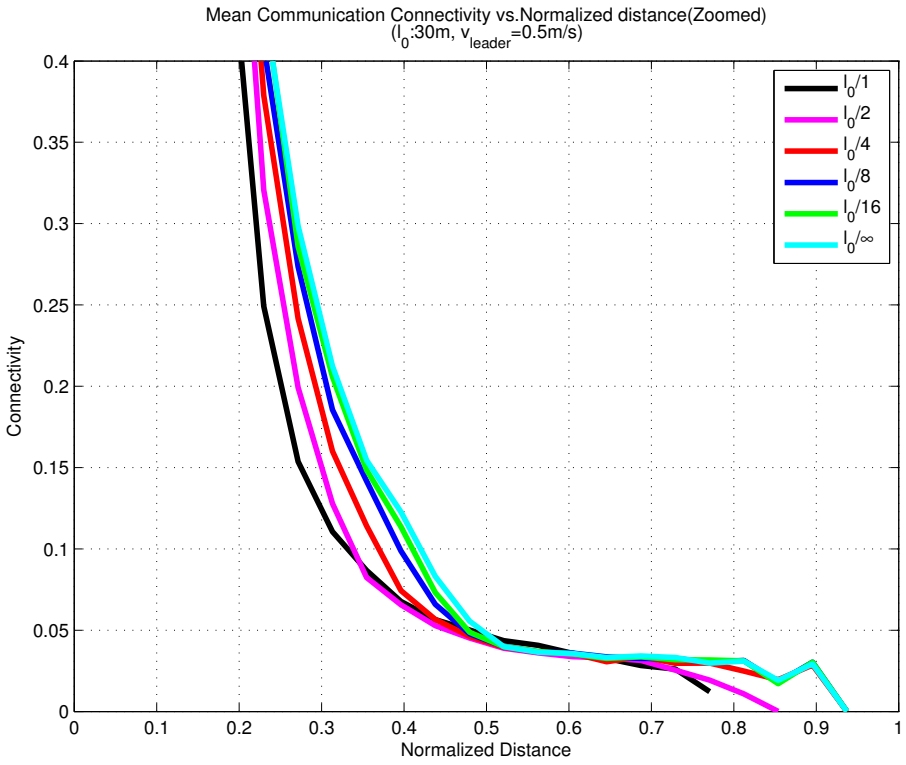


(a)

Figure 6.32: μ_{mean}, mean algebraic connectivity for being in RF communication range of each other, when the leader as traveling at 0.5m/s.

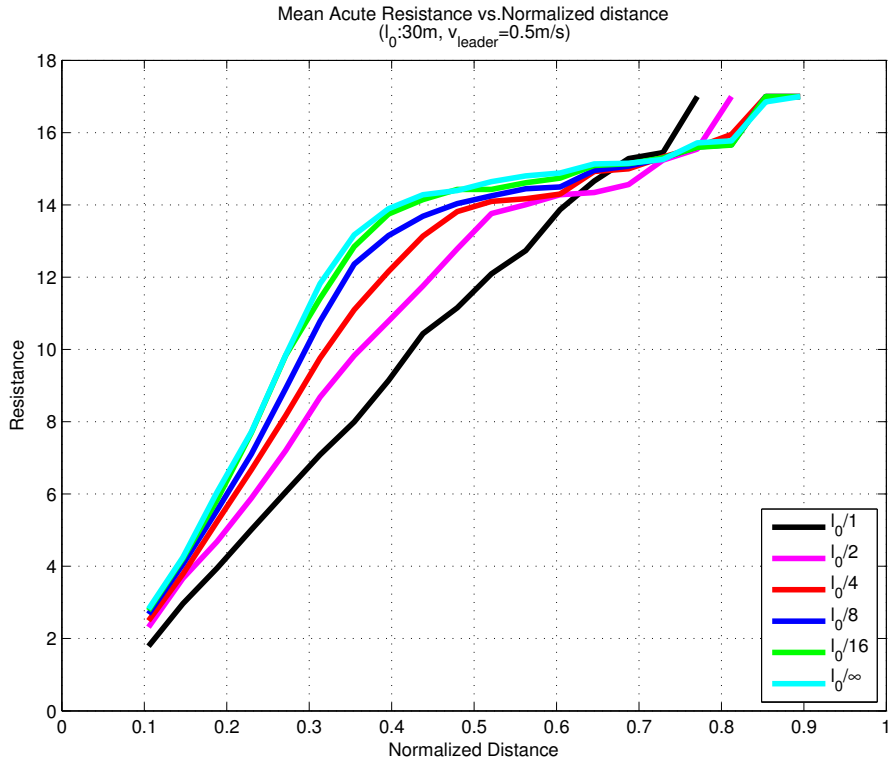


(a)

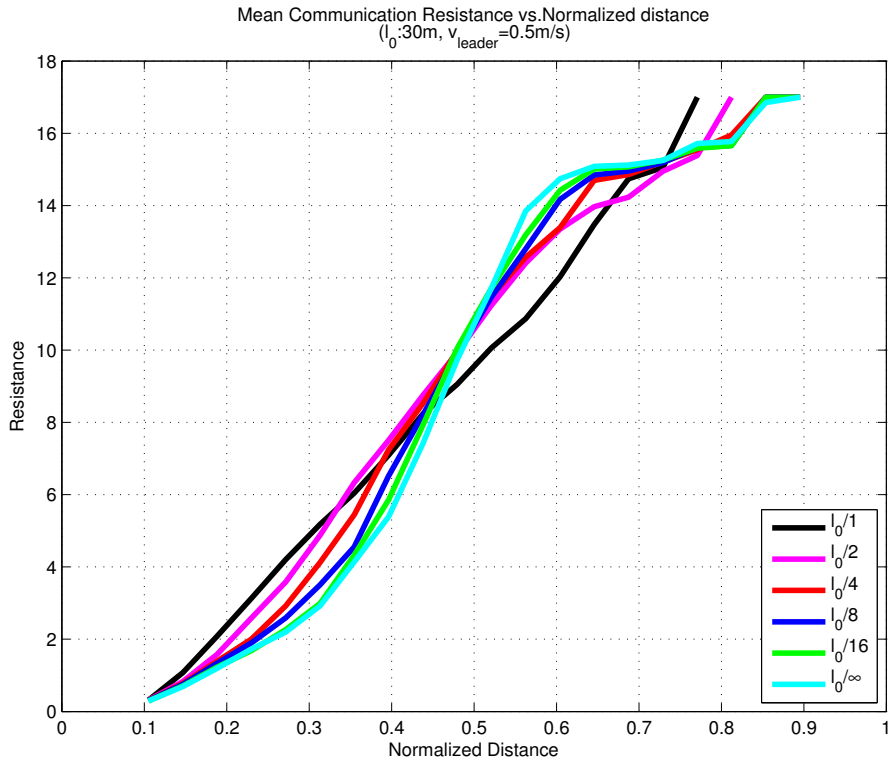


(b)

Figure 6.33: μ_{mean}, mean algebraic connectivity for the Gabriel Graph of virtual forces (a), and zoomed version of being in RF communication range of each other (b), when the leader as traveling at 0.5m/s.



(a)



(b)

Figure 6.34: $R_{eq_{mean}}$, mean equivalent resistance for the Gabriel Graph of virtual forces (a), and of being in RF communication range of each other (b), when the leader as traveling at 0.5m/s.

the previous case. This time the network cannot keep to high updistance as was in the previous case. This is mainly caused by the high velocity of the leader compared to the previous case and relays maximum allowed velocity of $2.78m/s$. A leader moving too fast does not give the relays enough time to follow it while also trying to establish a stable network. Note that this time the algorithm is not even able to keep the system connected for a distance greater than 0.7 units.

Time Evolution Of The Gabriel Graph Time evolution of the mobile relay network for different speeds and different critical l_0 is given in figures 6.39, 6.41, 6.40, 6.42, 6.43, 6.46, 6.47. Note that evolution of the network depends not only on the parameters but also the initial state. Evolutions put here are randomly selected with the only criteria to display what is happening when an algorithm reaches maximum updistance, what is happening when it does not. When updistance performance of a variation of the algorithm increases, it means that the good cases start occuring more frequently, and the bad cases less.

As has already been mentioned, updistance depends on the initial state, and this could be seen in figures 6.39 and 6.40 which are both for the same critical l_0 of l_0 and the same leader velocity of $v_{leader} = 0.5m/s$. In figure 6.39 although link lengths get non-uniform, thanks to the slow moving leader, the network has enough time and suitable initial state to correct this problem. However note the existance of the quadrupole nodes close to the sink, which due to symmetry do not let any of its links disappear. However with the same leader speed and critical l_0, the network in figure 6.40 cannot reach its full extend, mainly due to distribution of the nodes. This is similar to what happens in figure 6.40 where the leader is moving at $2.5m/s$.

In figures 6.42 and 6.43 critical l_0 is reduced to $l_0/2$. In figure 6.42 the network stretches till all nodes are on a straight line, achieving full potential. But in figure 6.43 we have again formation of symmetric node quadruples, fixing two extra nodes to a region where they are not needed any more, and as a result preventing full stretch of the network.

Reducing critical l_0 to $l_0/4$ in 6.44 and 6.45 does not change the global outlook: the network may reach its full potential, or some node qudrupoles may prevent the

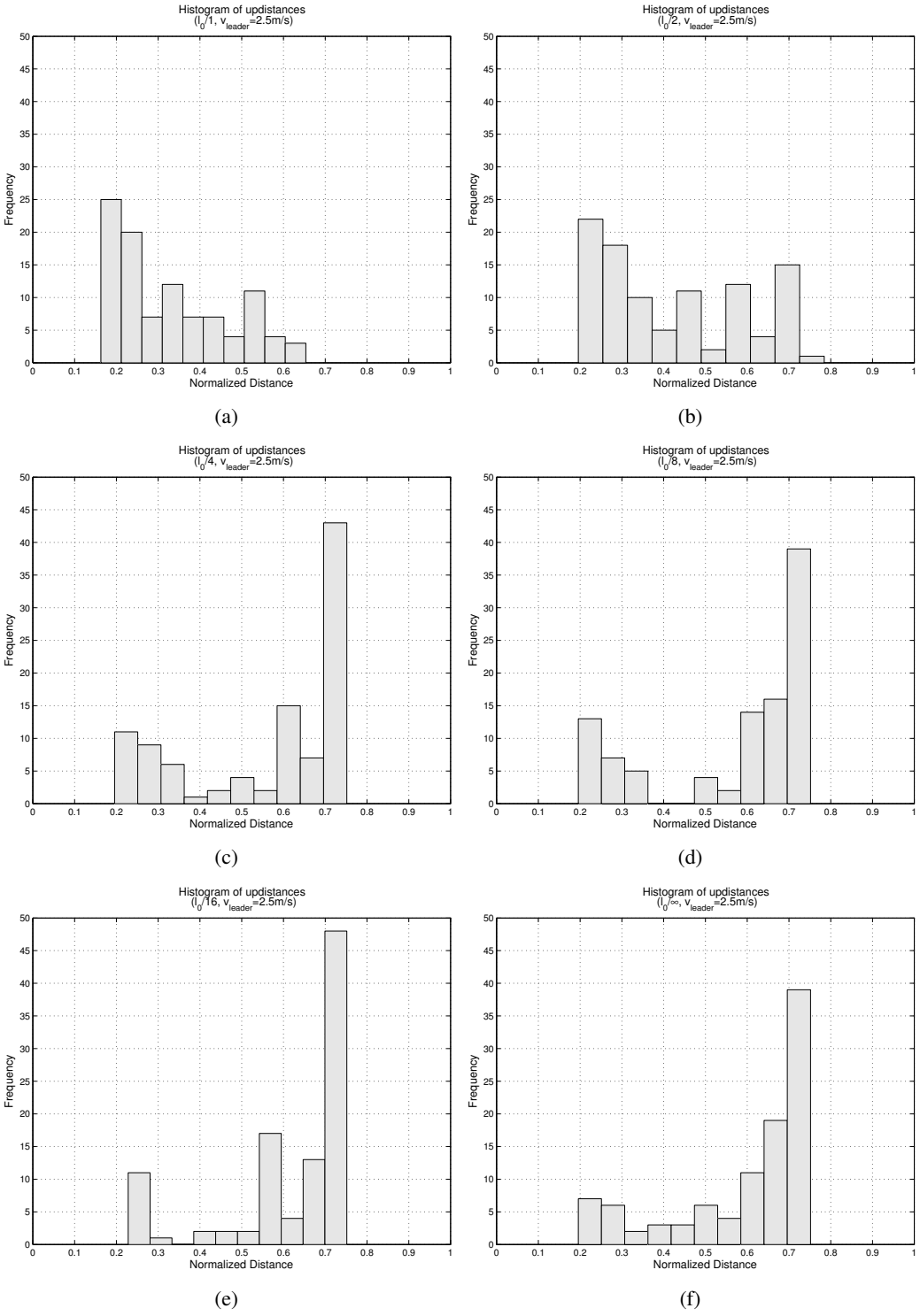
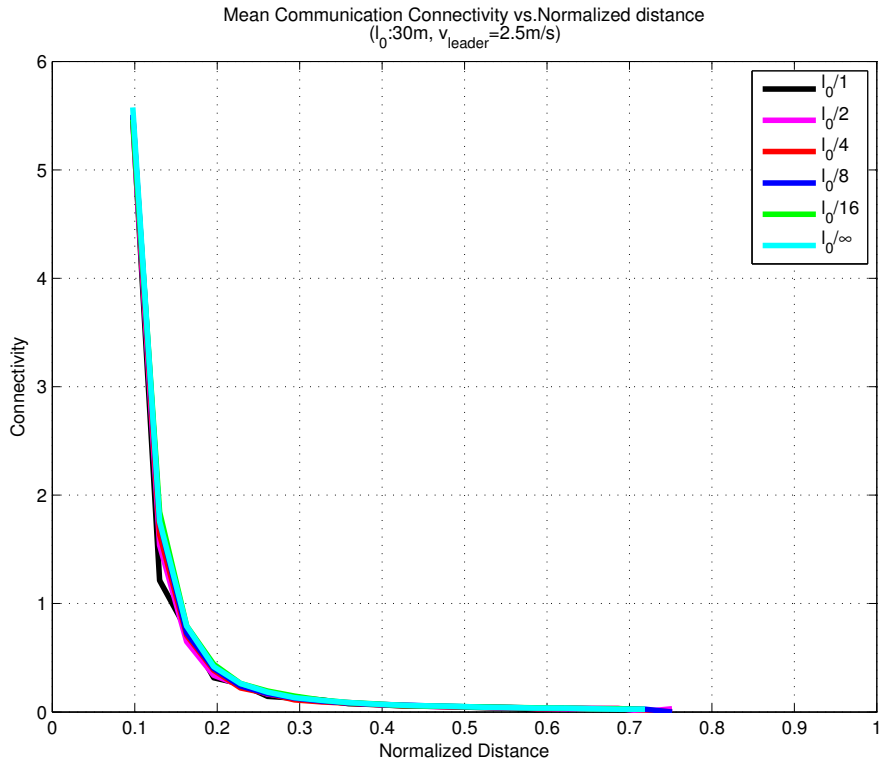
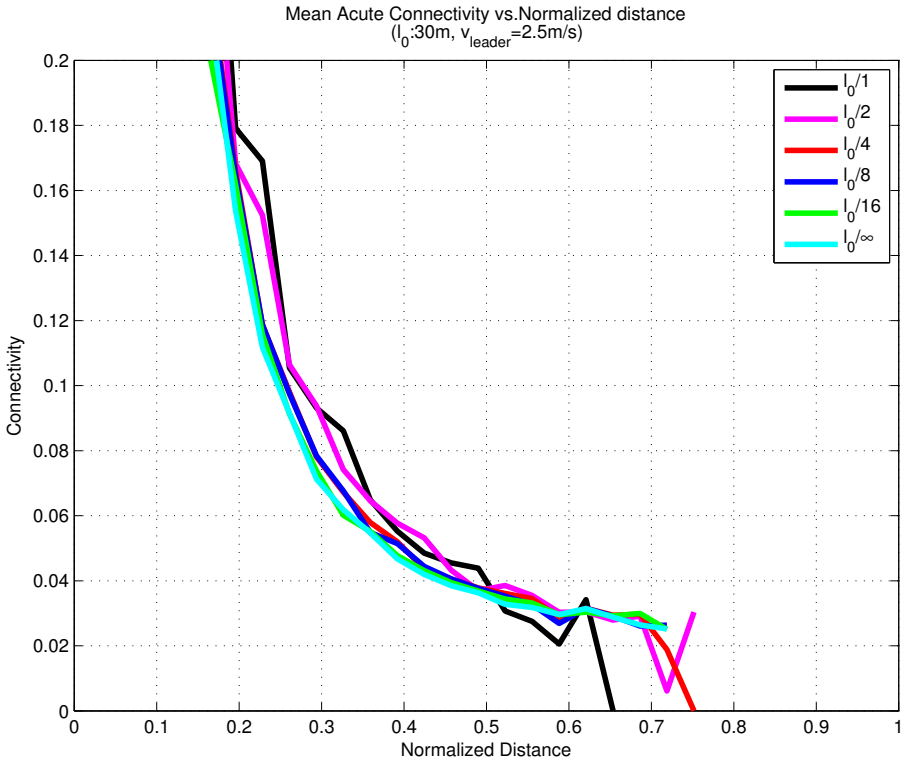


Figure 6.35: Updistance histograms for various critical l_0 values at critical junctions and fixed leader velocity of 2.5m/s.

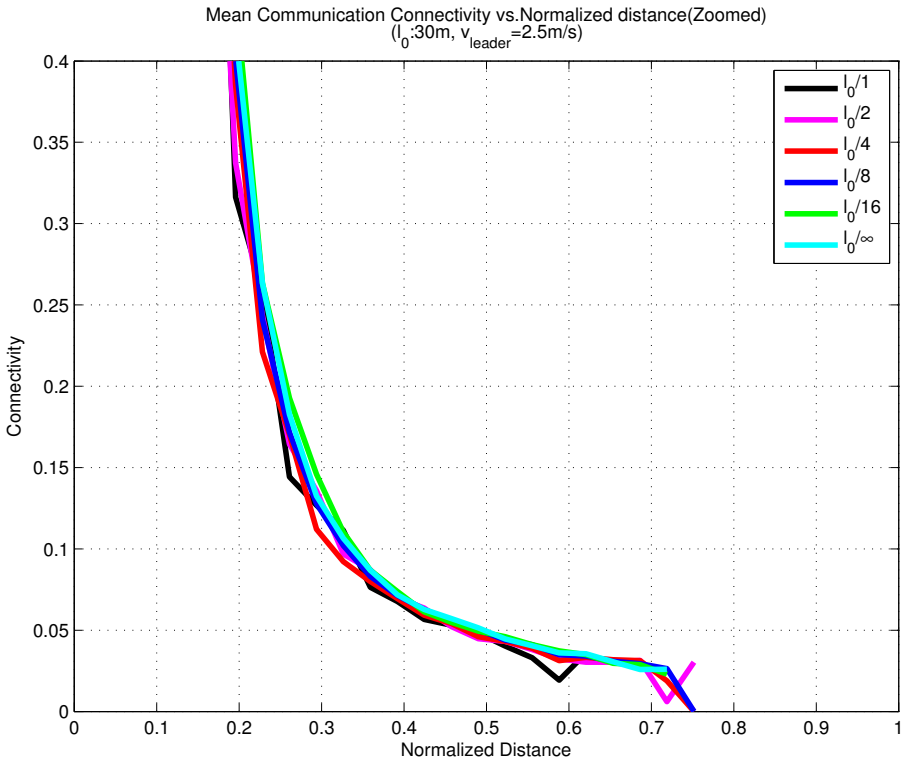


(a)

Figure 6.36: μ_{mean}, mean algebraic connectivity for being in RF communication range of each other, when the leader as traveling at 2.5m/s.

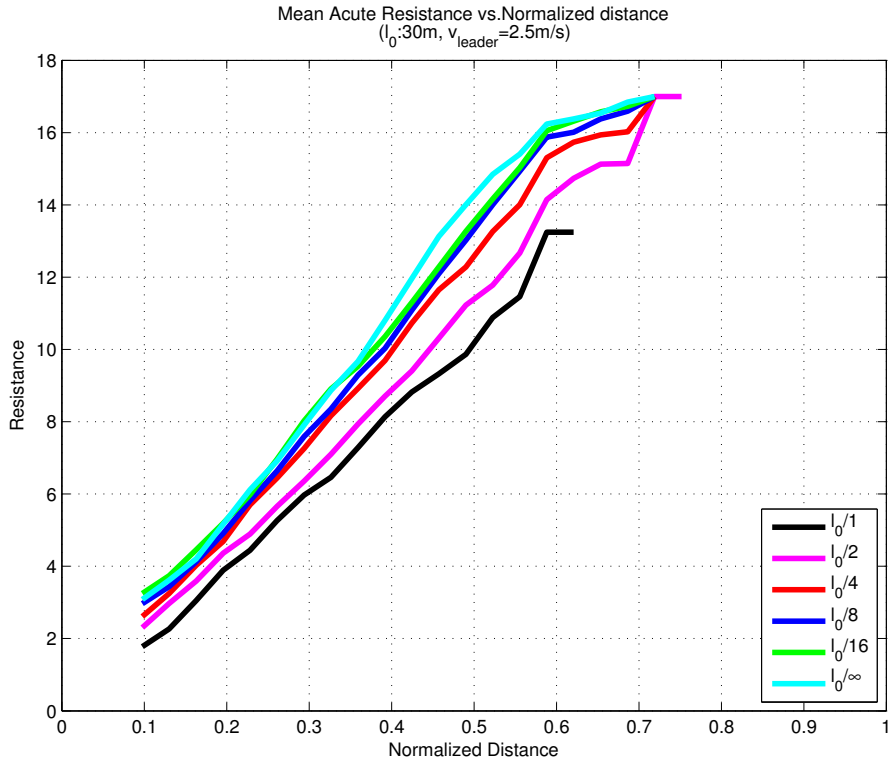


(a)

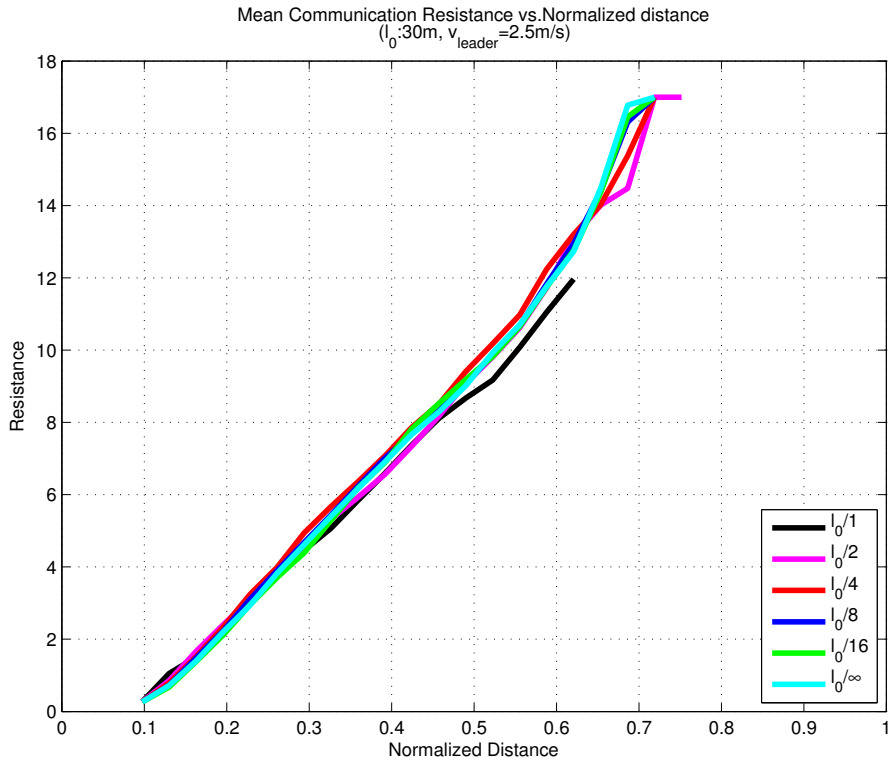


(b)

Figure 6.37: μ_{mean}, mean algebraic connectivity for the Gabriel Graph of virtual forces (a), and zoomed version of being in RF communication range of each other (b), when the leader as traveling at 2.5m/s.



(a)



(b)

Figure 6.38: $R_{eq_{mean}}$, mean equivalent resistance for the Gabriel Graph of virtual forces (a), and of being in RF communication range of each other (b), when the leader as traveling at 2.5m/s.

network from achieving its full potential.

In figures 6.46 and 6.47 critical l_0 is reduced to $l_0/8$ and the leader is set to move at $v_{leader} = 2.5m/s$. In figure 6.46 the network is not able to adapt, connectivity is broken quite at an early stage, however in figure 6.47 the network (the relays not the individual links) stretches fully. Note that although the relays are fully stretched, due to the high speed of the leader links closer to the sink do not have enough time to stretch to full length whereas the link to the leader stretches fully, so connection is broken at an earlier time step.

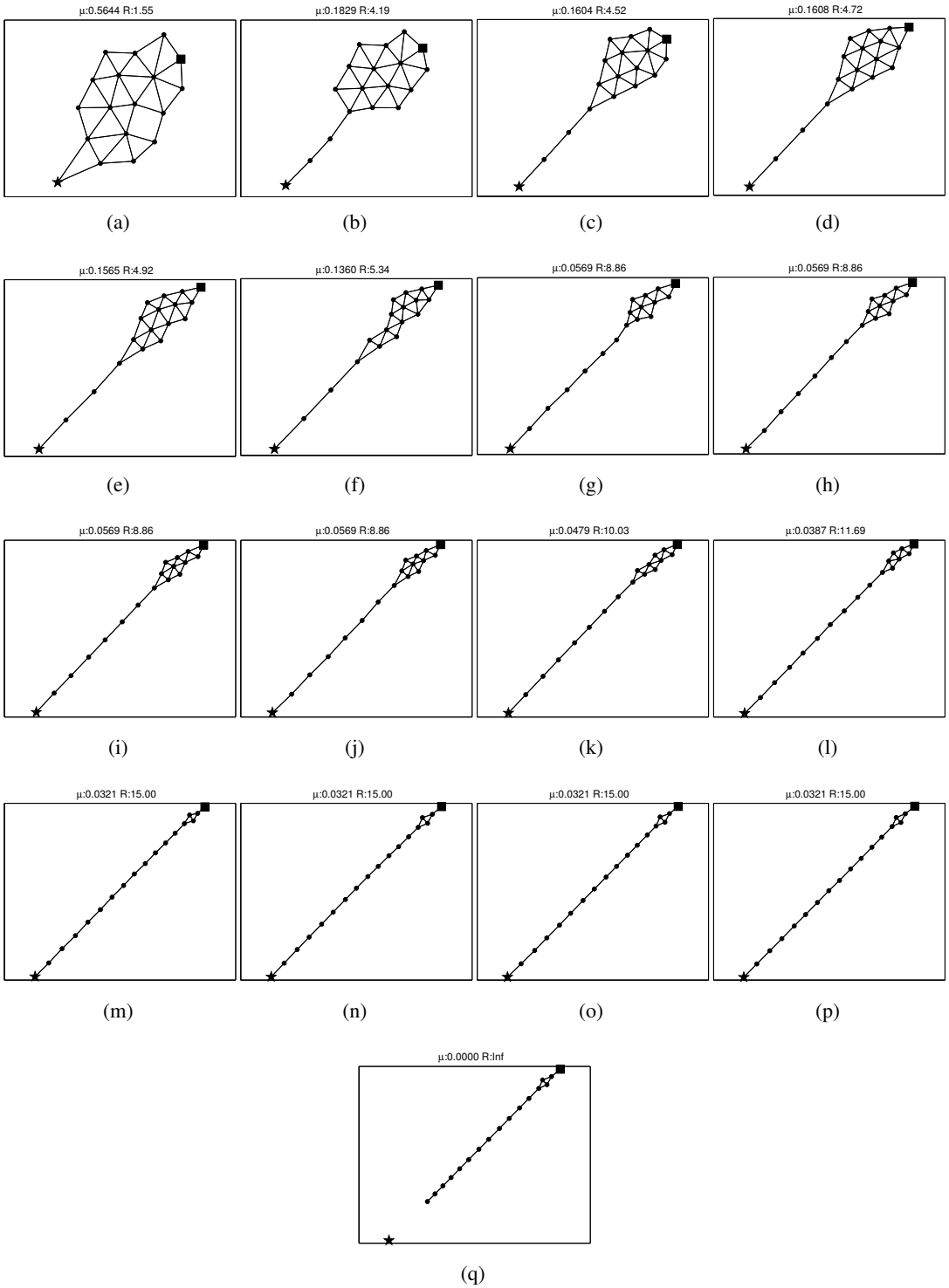


Figure 6.39: Screenshots taken at different times of the acute triangulation algorithm as the leader moves at $v_{leader} = 0.5m/s$ and critical l_0 is l_0. The robots start with the initial configuration given (a), and connection is lost in (q). See in the figure that although link lengths get non-uniform, thanks to the slow moving leader, the network has enough time to correct this problem. The square indicates the sink, the star the leader, the black dots other mobile relays. The lines are the connections between the robots.

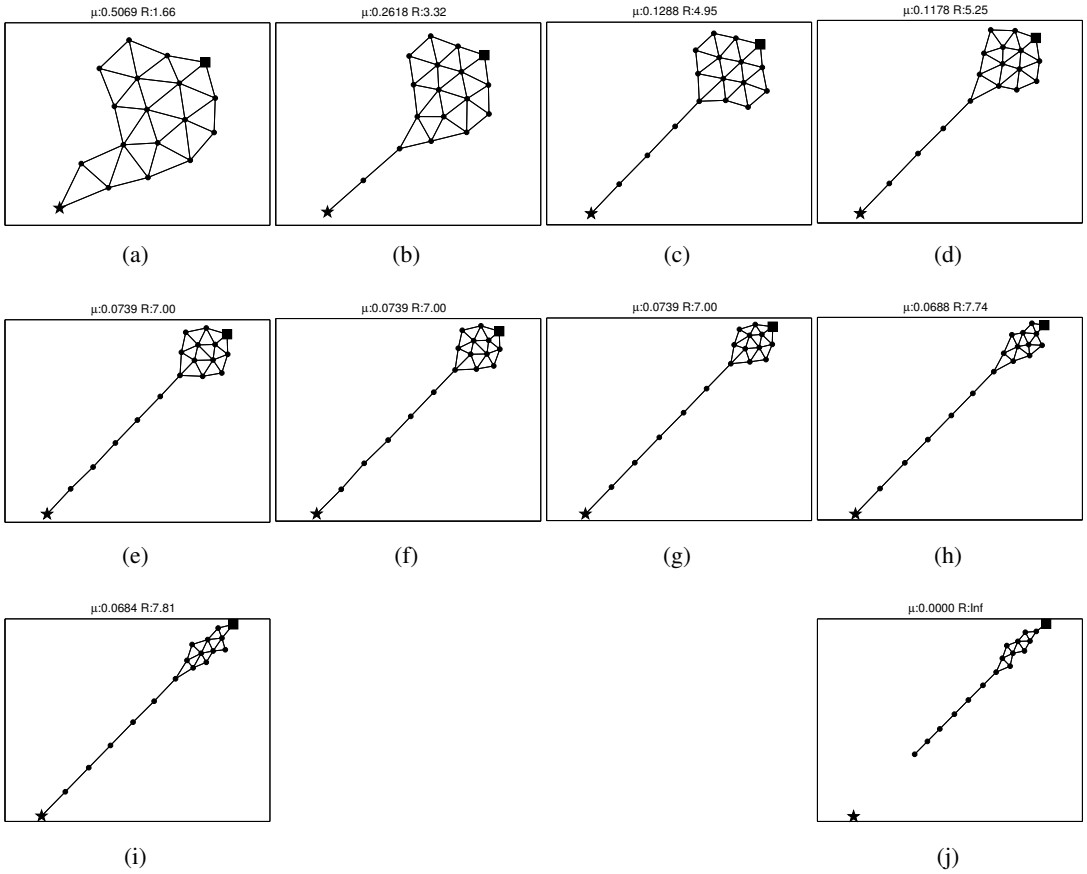


Figure 6.40: Screenshots taken at different times of the acute triangulation algorithm as the leader moves at $v_{leader} = 0.5m/s$ and critical l_0 is l_0. The robots start with the initial configuration given (a), and connection is lost in (j). See in the figure that link lengths get non-uniform, but this time the network cannot adept despite the slow moving leader. The square indicates the sink, the star the leader, the black dots other mobile relays. The lines are the connections between the robots.

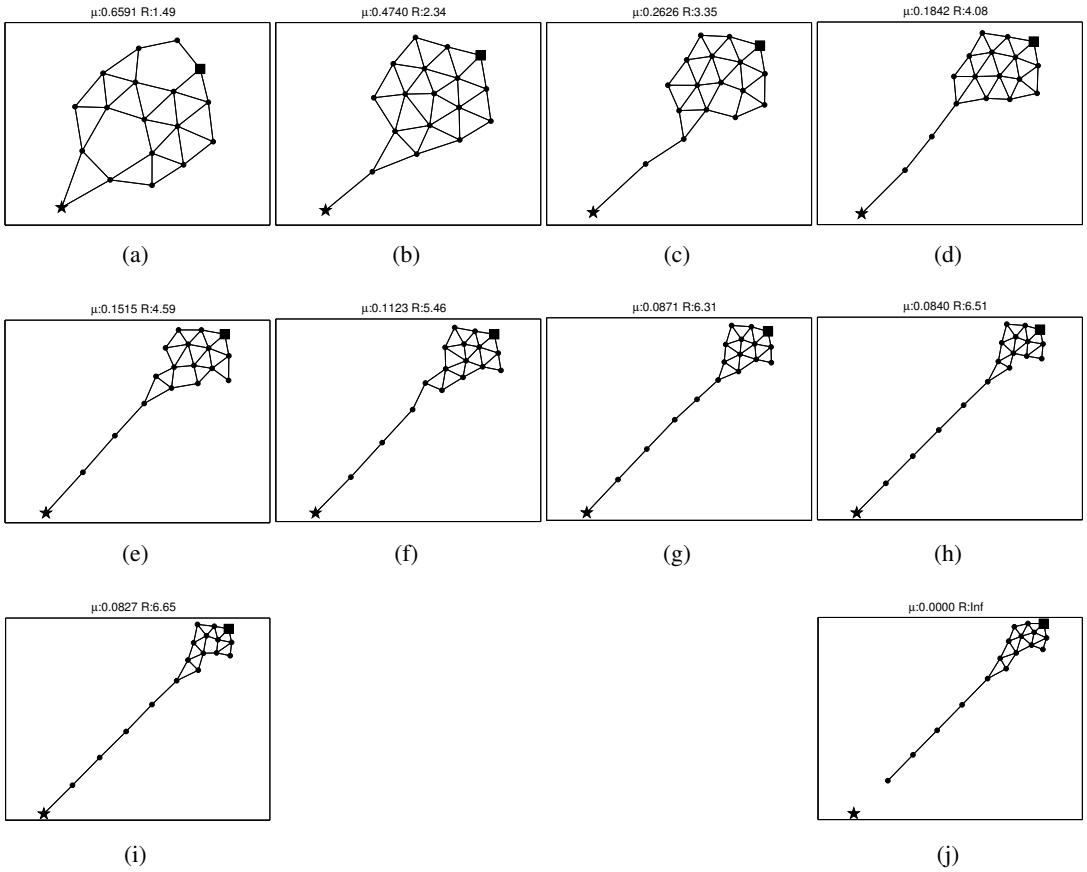


Figure 6.41: Screenshots taken at different times of the acute triangulation algorithm as the leader moves at $v_{leader} = 2.5m/s$ and critical l_0 is l_0. The robots start with the initial configuration given (a), and connection is lost in (j). See in the figure that link lengths are not uniform: part of the network that is crowded has small links whereas leaders neighbors have longer ones, which soon disappear, disconnecting the network. Since the leader is too fast, the network does not have enough time to adept (compare to figure 6.39). The square indicates the sink, the star the leader, the black dots other mobile relays. The lines are the connections between the robots.

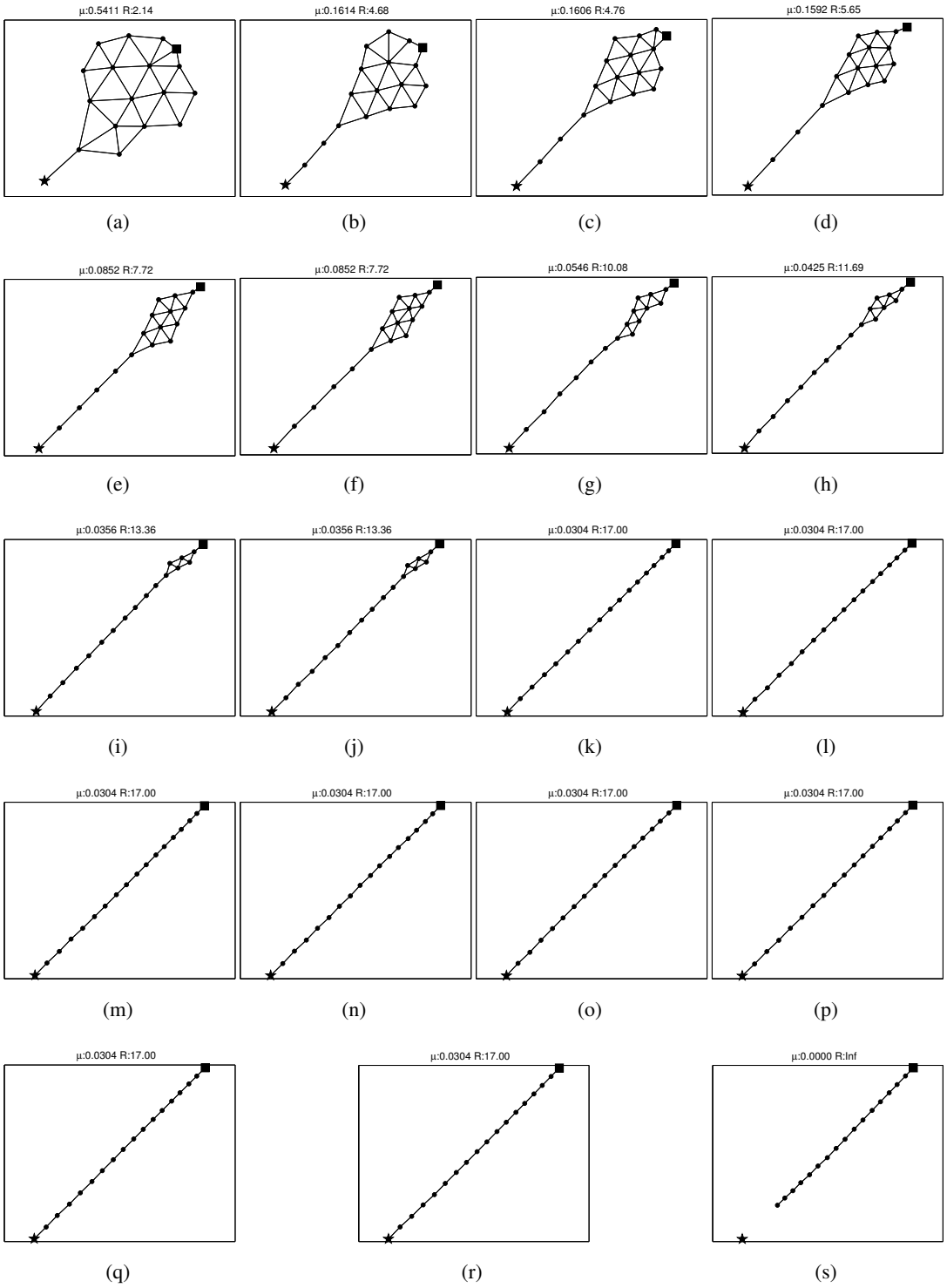


Figure 6.42: Screenshots taken at different times of the acute triangulation algorithm as the leader moves at $v_{leader} = 0.5m/s$ and critical l_0 is $l_0/2$. The robots start with the initial configuration given (a), and connection is lost in (s). See in the figure that link lengths get non-uniform, but this time not as much as the cases for critical $l_0 = l_0$. The system stretches till all nodes are on a straight line. The square indicates the sink, the star the leader, the black dots other mobile relays. The lines are the connections between the robots.

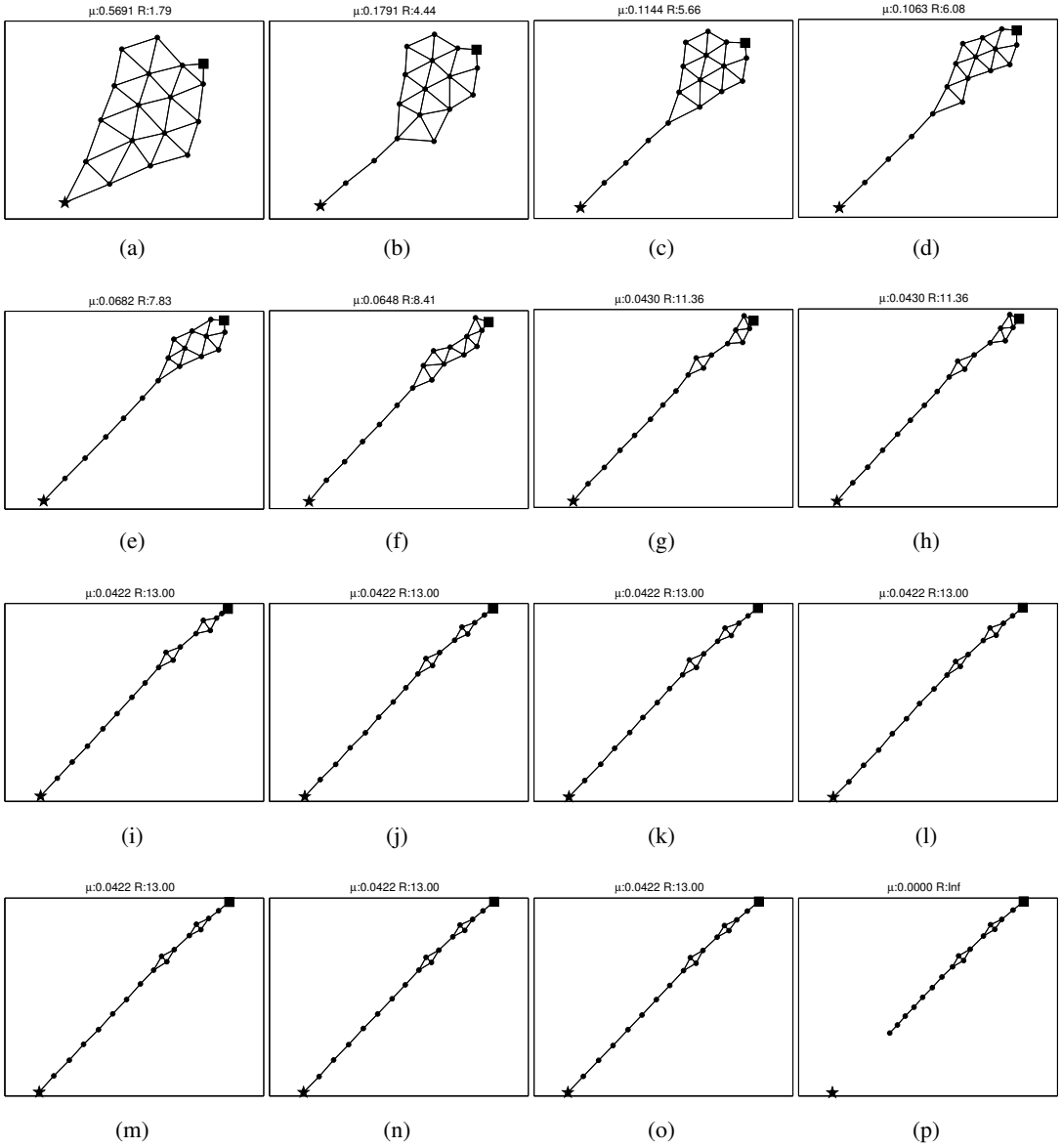


Figure 6.43: Screenshots taken at different times of the acute triangulation algorithm as the leader moves at $v_{leader} = 0.5m/s$ and critical l_0 is $l_0/2$. The robots start with the initial configuration given (a), and connection is lost in (p). See in the figure that link lengths get non-uniform, but this time not as much as the cases for critical $l_0 = l_0$. However here we see new problem, symmetric node quadruples are formed, which resist forces to break any one of the links, so some other link is broken and the system is not able to utilize its full potential. The square indicates the sink, the star the leader, the black dots other mobile relays. The lines are the connections between the robots.

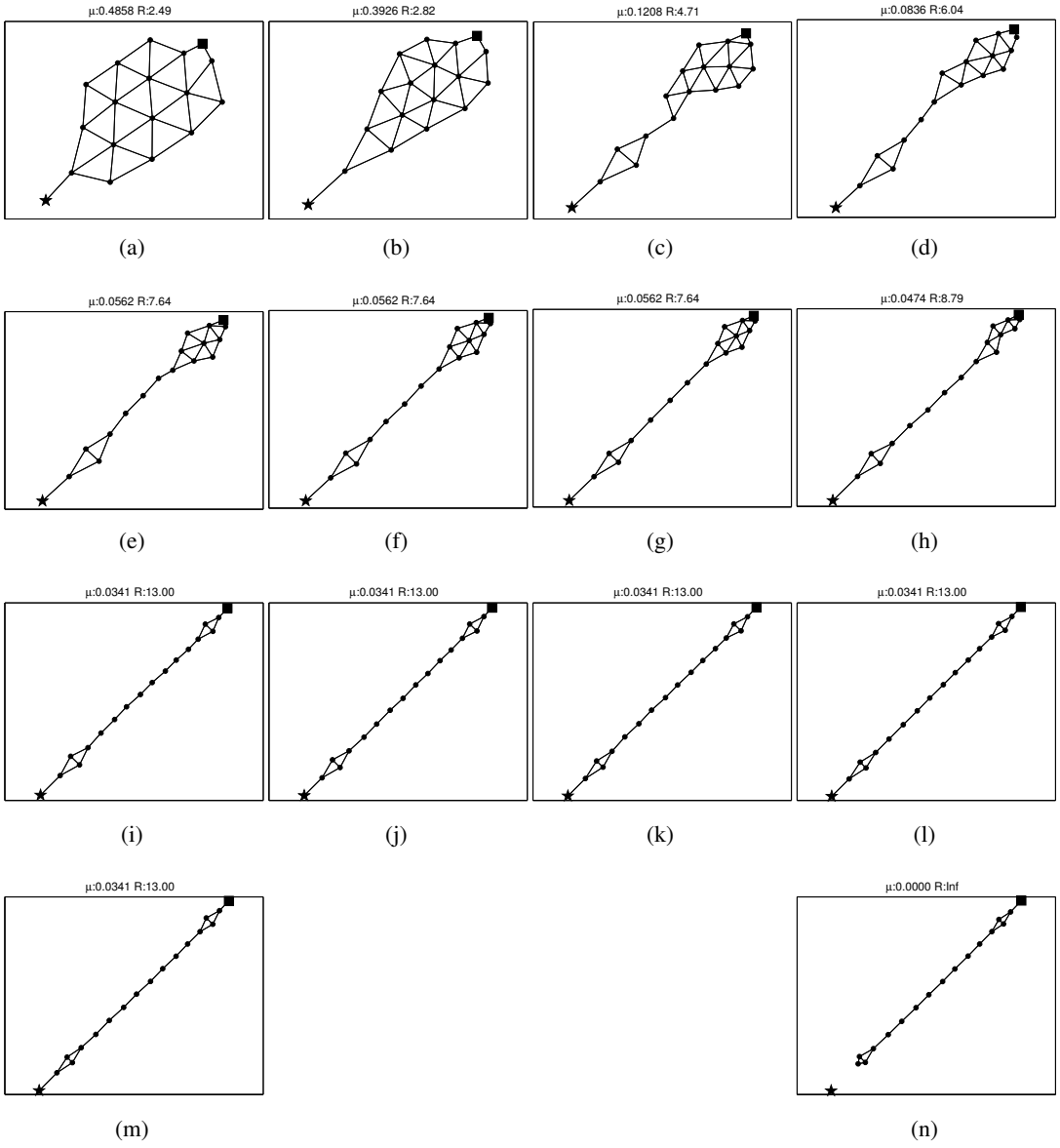


Figure 6.44: Screenshots taken at different times of the acute triangulation algorithm as the leader moves at $v_{leader} = 0.5m/s$ and critical l_0 is $l_0/4$. The robots start with the initial configuration given (a), and connection is lost in (n). See in the figure that link lengths get non-uniform, but not as much as the cases for critical $l_0 = l_0$. However here we see the same problem as in figure 6.43, symmetric node quadruples are formed, which resist forces to break any one of the links, so some other link is broken and the system is not able to utilize its full potential. The square indicates the sink, the star the leader, the black dots other mobile relays. The lines are the connections between the robots.

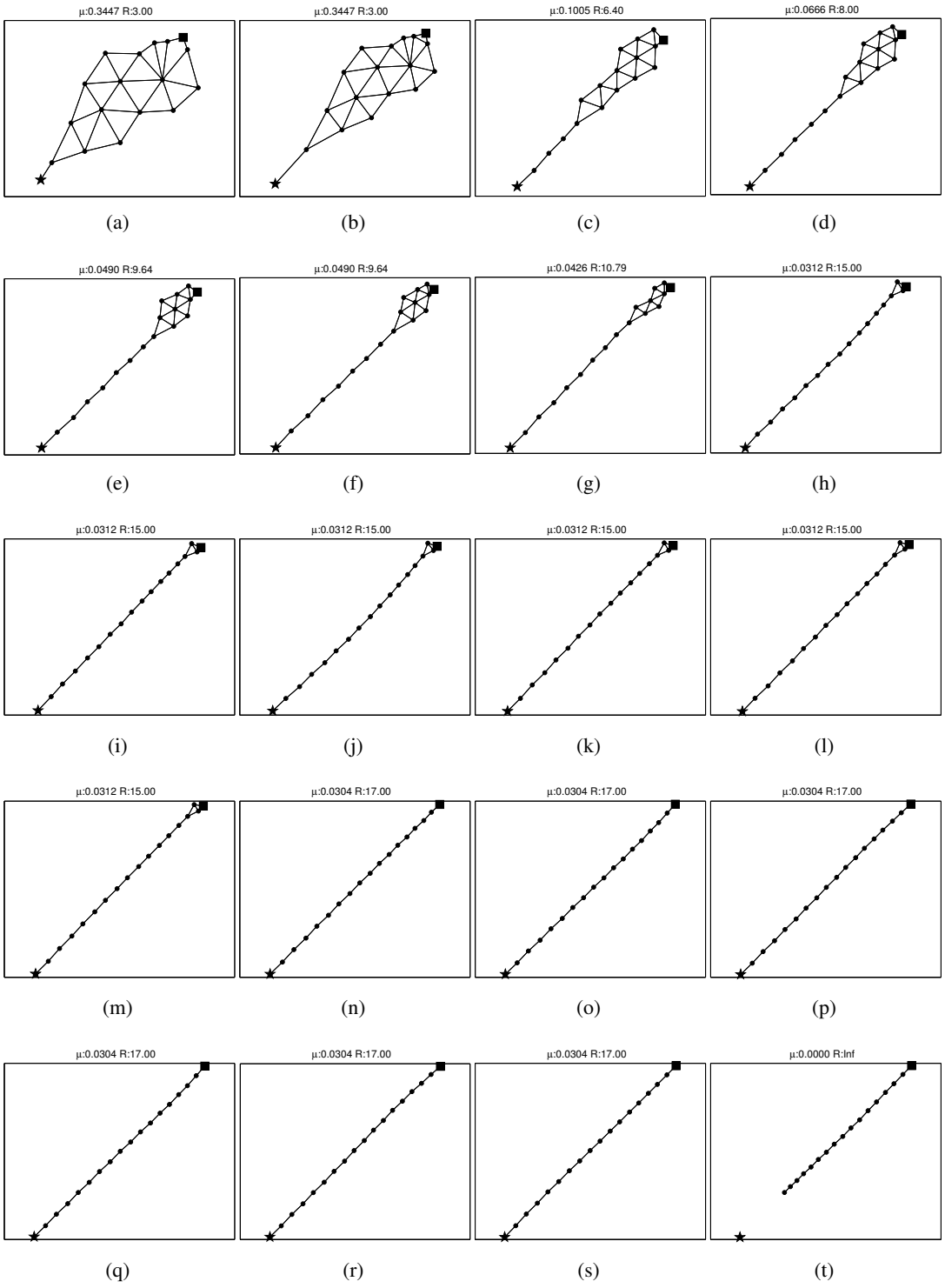


Figure 6.45: Screenshots taken at different times of the acute triangulation algorithm as the leader moves at $v_{leader} = 0.5m/s$ and critical l_0 is $l_0/4$. The robots start with the initial configuration given (a), and connection is lost in (t). See in the figure that link lengths get non-uniform, but not as much as the cases for critical $l_0 = l_0$. And there are no symmetric node quadruples as in figures 6.43 and 6.44. So the network achieves its full potential. The square indicates the sink, the star the leader, the black dots other mobile relays. The lines are the connections between the robots.

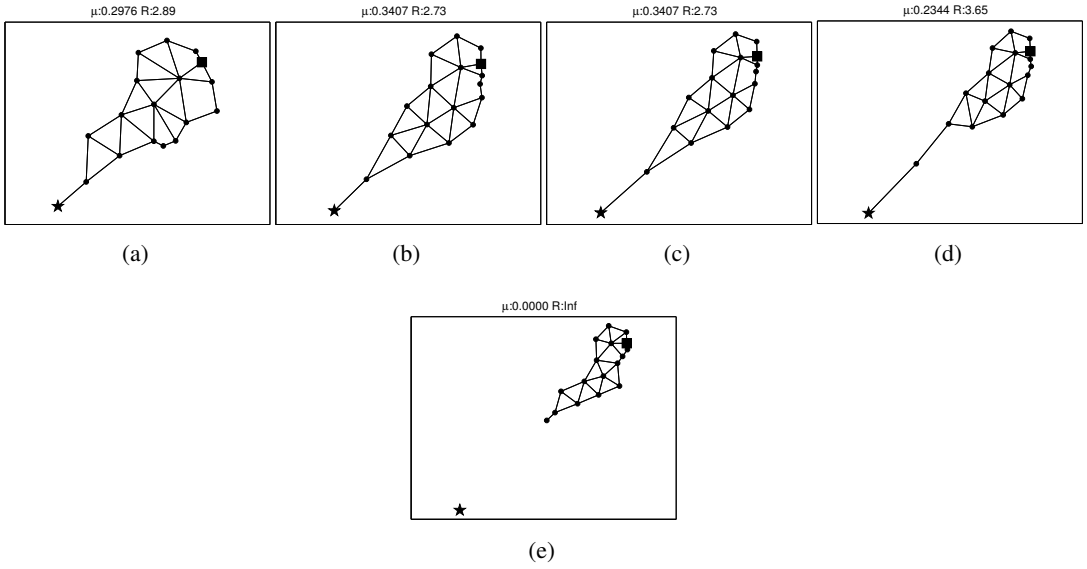


Figure 6.46: Screenshots taken at different times of the acute triangulation algorithm as the leader moves at $v_{leader} = 2.5m/s$ and critical l_0 is $l_0/8$. The robots start with the initial configuration given (a), and connection is lost in (e). See in the figure that the leader fast, stretching its links far but on the other hand due to the critical link around the sink some nodes are collected around the sink. Connection is soon broken. The square indicates the sink, the star the leader, the black dots other mobile relays. The lines are the connections between the robots.

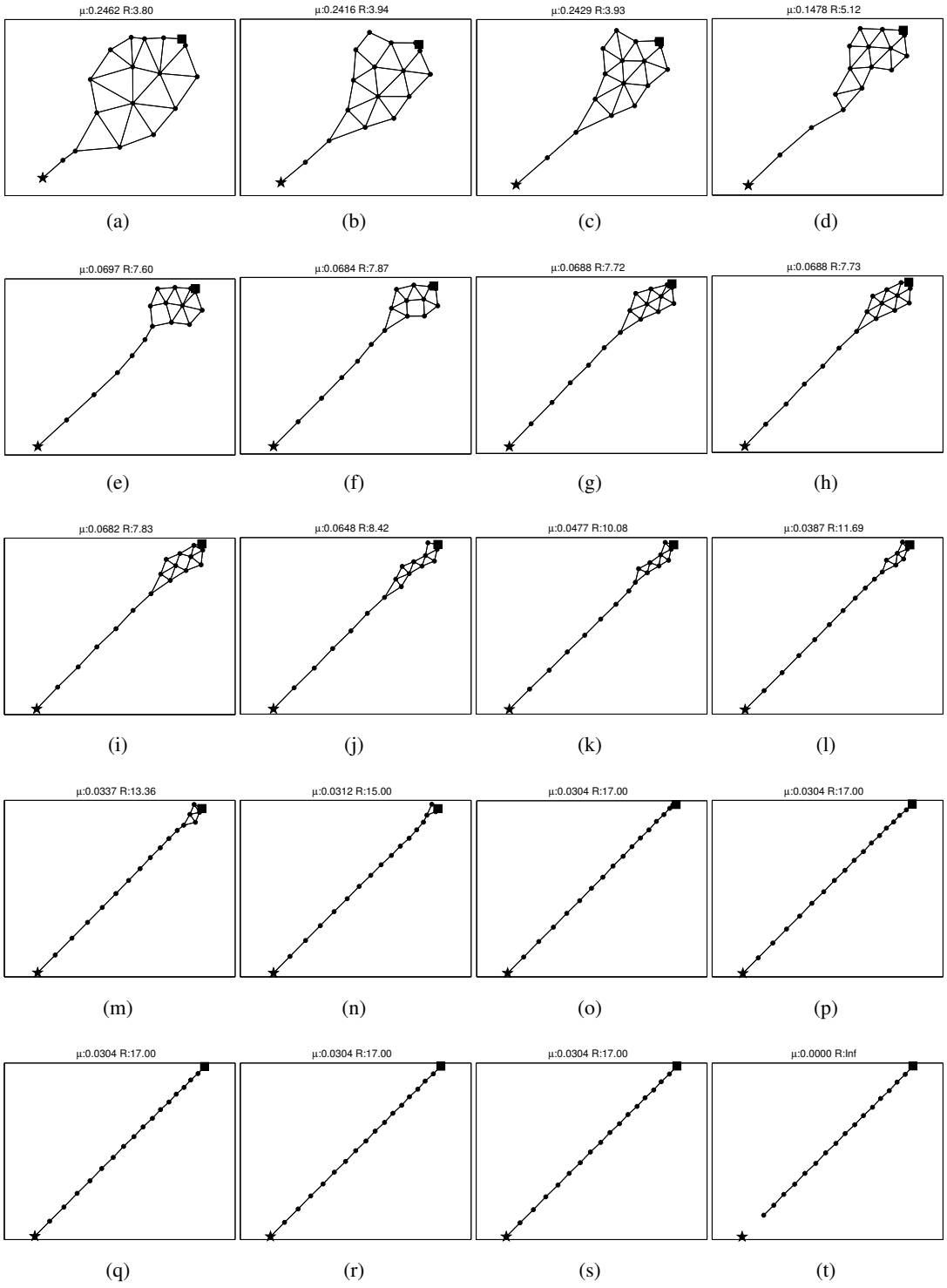


Figure 6.47: Screenshots taken at different times of the acute triangulation algorithm as the leader moves at $v_{leader} = 2.5m/s$ and critical l_0 is $l_0/8$. The robots start with the initial configuration given (a), and connection is lost in (t). Contrary to the previous figure (figure 6.46), though the same parameters are used (but a different initial state), the network utilizes its full potential. The square indicates the sink, the star the leader, the black dots other mobile relays. The lines are the connections between the robots.

CHAPTER 7

Conclusion & Future Work

7.1 Conclusion

In this thesis we have proposed a new Wireless Sensor Network deployment architecture, and called it Sycophant Wireless Sensor Networks. In order to improve the usability of SWS networks we proposed using them in combination with sparse Mobile Sensor Networks consisting of mainly mobile robots, although in addition to mobile robots some stationary sensors or other types of mobile agents are welcome. This hybrid architecture has led us to create hybrid 2D maps. Then as a further application of the hybrid network, novel 3D maps using horizontal 2D map slices were created. Simulation as well as hardware results for these maps were given. Then full 3D maps were created fusing vertical 2D scans of the sycophant and 2D map as well as the tracking performance of the follower robot. Afterwards three connectivity maintenance methods, two of which novel, were implemented and related simulation results were presented and discussed. In order to judge the performance, two new performance metrics were presented. The first one was updistance and it tells us how far a leader can go without breaking connectivity of the network. The second one was resistance and it tells us how sparse a relay network is.

7.2 Future Work

Theoretical bounds on tracking performance should be found, and simulations be run in more complicated, more crowded, totally unstructured larger areas. Although in

this work jerky paths have been chosen to make the system observable, finding an energy efficient smooth path planner that still makes the system observable is a possible way to go. Another way to go is utilizing more data from the SWS in order to relax the constraint of motion for observability. Even if this information is available partially or for certain time slots, it is for sure going to increase tracking performance and contribute more to an energy efficient tracking. Using hybrid SLAM, as is mentioned in this thesis would be a nice starting point to improve tracking too, since extra sensor data would transform the problem from tracking to multi-agent SLAM where poses of the sycophants are corrected through the measurement update of the Particle/Kalman Filter. Experimenting with different sensors and deciding on optimal ones for the SWS network is a critical step that deserves attention in a hybrid SLAM architecture. Although not mentioned in this thesis, there is considerable research on Personal Dead Reckoning (PDR) systems. Integration of PDR to the SWS network would be still another possible way to increase tracking performance considerably and improve abilities and usability of the SWS network. Large scale hardware implementation is a necessary but big step to take, requiring solutions to some of the problems mentioned.

2D horizontal maps based 3D mapping should be improved to be more real-time. This would also help improve map of the follower robot. Inclusion of more sycophants or followers at different levels can improve the created 3D maps.

The restriction of a predefined angle for the full 3D maps should be relaxed. That will allow the carrying agent of the sycophant move more freely and naturally in the environment. But the scanning angle of the SWS will need to be estimated at every step.

The two new connectivity maintenance methods should be improved to take into account obstacles present in the environment. The theoretical bounds for sparseness should be found and compared to simulations results.

Bibliography

[1] Crossbow technology : MICAz 2.4GHz - wireless module. http://www.xbow.com/Products/productdetails.aspx?sid=164.

[2] Main protein structure levels. http://en.wikipedia.org/.

[3] TinyOS community forum || an open-source OS for the networked sensor regime. http://www.tinyos.net/.

[4] N. Adluru, L. J Latecki, M. Sobel, and R. Lakaemper. Merging maps of multiple robots. In *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, pages 1 –4, 2008.

[5] N. Aizawa and N. Kubota. Intelligent formation control based on directionality of multi-agent system. In *Robotic Intelligence in Informationally Structured Space, 2009. RIISS '09. IEEE Workshop on*, pages 87–92, April 2009.

[6] L. A.A Andersson and J. Nygards. On multi-robot map fusion by inter-robot observations. In *Information Fusion, 2009. FUSION '09. 12th International Conference on*, pages 1712 –1721, 2009.

[7] R. Aylward, S. D Lovell, and J. A Paradiso. A compact, wireless, wearable sensor network for interactive dance ensembles. *Wearable and Implantable Body Sensor Networks, 2006. BSN 2006. International Workshop on*, pages 4 pp.–70, April 2006.

[8] R. Aylward and J. A Paradiso. A compact, High-Speed, wearable sensor network for biomotion capture and interactive media. *Information Processing in Sensor Networks, 2007. IPSN 2007. 6th International Symposium on*, pages 380–389, April 2007.

[9] R. B. Bapat. *Graphs and Matrices*. Springer, 1st edition. edition, March 2011.

[10] Yaakov Bar-Shalom and Thomas E Fortmann. *Tracking and data association*, volume 179 of *Mathematics in Science and Engineering*. Academic Press Professional, Inc., San Diego, CA, USA, 1987.

[11] Frederick A. Bettelheim, William H. Brown, and Jerry March. *Introduction to General, Organic & Biochemistry*. Harcourt College Publishers, 2001.

[12] A. Birk and S. Carpin. Merging occupancy grid maps from multiple robots. *Proceedings of the IEEE*, 94(7):1384 –1397, July 2006.

[13] Andreas Birk, Kaustubh Pathak, Narunas Vaskevicius, Max Pfingsthorn, Jann Poppinga, and Soren Schwertfeger. Surface representations for 3d mapping. *KI - Kunstliche Intelligenz*, 24:249–254, 2010. 10.1007/s13218-010-0035-1.

[14] J. A. Bondy and U. S. R. Murty. *Graph Theory*. Springer, 2008.

[15] Engelbert Buxbaum. *Fundamentals of protein structure and function*. Springer, 2007.

[16] M. Cardei, Yinying Yang, and Jie Wu. Non-uniform sensor deployment in mobile wireless sensor networks. *World of Wireless, Mobile and Multimedia Networks, 2008. WoWMoM 2008. 2008 International Symposium on a*, pages 1–8, June 2008.

[17] S. Carpin. Merging maps via hough transform. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 1878 – 1883, 2008.

[18] J. Chen, Karric Kwong, D. Chang, J. Luk, and R. Bajcsy. Wearable sensors for reliable fall detection. *Engineering in Medicine and Biology Society, 2005. IEEE-EMBS 2005. 27th Annual International Conference of the*, pages 3551–3554, January 2005.

[19] Xi Chen, Wei Zhuang, and Jindong Tan. The maintaining of communication links quality in unknown environment. In *Information and Automation, 2009. ICIA '09. International Conference on*, pages 218–223, June 2009.

[20] N. Correll, J. Bachrach, D. Vickery, and D. Rus. Ad-hoc wireless network coverage with networked robots that cannot localize. In *Robotics and Automation,*

2009. ICRA '09. IEEE International Conference on, pages 3878–3885, May 2009.

[21] J. Cortes, S. Martinez, and F. Bullo. Robust rendezvous for mobile autonomous agents via proximity graphs in arbitrary dimensions. *Automatic Control, IEEE Transactions on*, 51(8):1289–1298, August 2006.

[22] P. de la Puente, D. Rodriguez-Losada, A. Valero, and F. Matia. 3d feature based mapping towards mobile robots' enhanced performance in rescue missions. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 1138 –1143, oct. 2009.

[23] Goksel Dedeoglu and Gaurav Sukhatme. Landmark-based matching algorithm for cooperative mapping by autonomous robots. In *Distributed Autonomous Robotics Systems*, pages 251–260. Springer-Verlag, 2000.

[24] D. V Dimarogonas and K. H Johansson. Decentralized connectivity maintenance in mobile networks with bounded inputs. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 1507–1512, May 2008.

[25] D. V Dimarogonas and K. J Kyriakopoulos. On the rendezvous problem for multiple nonholonomic agents. *Automatic Control, IEEE Transactions on*, 52(5):916–922, May 2007.

[26] Cory Dixon and Eric W. Frew. Maintaining optimal communication chains in robotic sensor networks using mobility control. In *Proceedings of the 1st international conference on Robot communication and coordination*, pages 1–8, Athens, Greece, 2007. IEEE Press.

[27] I. Dryanovski, W. Morris, and Jizhong Xiao. Multi-volume occupancy grids: An efficient probabilistic 3d mapping model for micro aerial vehicles. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 1553 –1559, oct. 2010.

[28] Miroslaw Dynia, Jaroslaw Kutylowski, Friedhelm Meyer auf der Heide, and Jonas Schrieb. Local strategies for maintaining a chain of relay stations between an explorer and a base station. In *Proceedings of the nineteenth annual*

ACM symposium on Parallel algorithms and architectures, pages 260–269, San Diego, California, USA, 2007. ACM.

[29] Miroslaw Dynia, Jaroslaw Kutylowski, Pawel Lorek, and Friedhelm auf der Heide. Maintaining communication between an explorer and a base station. In *Biologically Inspired Cooperative Computing*. 2006.

[30] Yesim H Esin, Mustafa Unel, and Mehmet Yildiz. Formation control of multiple robots using parametric and implicit representations. In *ICIC '08: Proceedings of the 4th international conference on Intelligent Computing*, pages 558–565, Berlin, Heidelberg, 2008. Springer-Verlag.

[31] J. M Esposito and T. W Dunbar. Maintaining wireless connectivity constraints for swarms in the presence of obstacles. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 946–951, May 2006.

[32] E.H.L. Fong, W. Adams, F.L. Crabbe, and A.C. Schultz. Representing a 3-d environment with a 2 1/2 -d map structure. In *Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, volume 3, pages 2986 – 2991 vol.3, oct. 2003.

[33] N. Heo and P. K Varshney. A distributed self spreading algorithm for mobile wireless sensor networks. *Wireless Communications and Networking, 2003. WCNC 2003. 2003 IEEE*, 3:1597–1602 vol.3, March 2003.

[34] Nojeong Heo and P. K Varshney. An intelligent deployment and clustering algorithm for a distributed mobile sensor network. *Systems, Man and Cybernetics, 2003. IEEE International Conference on*, 5:4576–4581 vol.5, October 2003.

[35] A. Howard. Multi-robot simultaneous localization and mapping using particle filters. *International Journal of Robotics Research*, 25(12):1243–1256, 2006.

[36] Meng Ji and M. Egerstedt. Distributed coordination control of multiagent systems while preserving connectedness. *Robotics, IEEE Transactions on*, 23(4):693–703, August 2007.

[37] Holger Karl and Andreas Willig. *Protocols and Architectures for Wireless Sensor Networks*. John Wiley & Sons, 2005.

[38] Yoonsoo Kim and M. Mesbahi. On maximizing the second smallest eigenvalue of a state-dependent graph laplacian. *Automatic Control, IEEE Transactions on*, 51(1):116–120, January 2006.

[39] K. Konolige. Centibots: Very large scale distributed robotic teams. *Springer Tracts in Advanced Robotics*, 21:131–140, 2006. cited By (since 1996) 0.

[40] A. Krause, A. Smailagic, and D. P Siewiorek. Context-aware mobile computing: learning context- dependent personal preferences from a wearable sensor array. *Mobile Computing, IEEE Transactions on*, 5(2):113–127, February 2006.

[41] S. Kroc and V. Delic. Personal wireless sensor network for mobile health care monitoring. *Telecommunications in Modern Satellite, Cable and Broadcasting Service, 2003. TELSIKS 2003. 6th International Conference on*, 2:471–474 vol.2, October 2003.

[42] M. Langerwisch. Registration of indoor 3D range images using virtual 2D scans. In *ICINCO 2010 - Proceedings of the 7th International Conference on Informatics in Control, Automation and Robotics*, volume 2, pages 327–332, 2010. cited By (since 1996) 0.

[43] Steven M. Lavalle. Rapidly-exploring random trees: A new tool for path planning. Technical report, 1998.

[44] Steven M Lavalle. *Planning Algorithms*. Cambridge University Press, May 2006. Published: Hardcover.

[45] A. Leon. SLAM and map merging. *Journal of Physical Agents*, 3(1):13–23, 2009. cited By (since 1996) 0.

[46] Kian Hsiang Low, W. K Leow, and M. H Ang. Autonomic mobile sensor network with self-coordinated task allocation and execution. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 36(3):315–327, May 2006.

[47] E. Z MacArthur and C. D Crane. Compliant formation control of a Multi-Vehicle system. In *Computational Intelligence in Robotics and Automation, 2007. CIRA 2007. International Symposium on*, pages 479–484, June 2007.

[48] S. Mahlknecht and S. A Madani. On architecture of low power wireless sensor networks for container tracking and monitoring applications. *Industrial Informatics, 2007 5th IEEE International Conference on*, 1:353–358, June 2007.

[49] E. Marder-Eppstein, E. Berger, T. Foote, B. Gerkey, and K. Konolige. The office marathon: Robust navigation in an indoor office environment. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 300 –307, may 2010.

[50] O. Michel. Webots: Professional mobile robot simulation. *Journal of Advanced Robotics Systems*, 1(1):39–42, 2004.

[51] H. Mizuno, H. Nagai, K. Sasaki, H. Hosaka, C. Sugimoto, K. Khalil, and S. Tatsuta. Wearable sensor system for human behavior recognition (First report: Basic architecture and behavior prediction method). *Solid-State Sensors, Actuators and Microsystems Conference, 2007. TRANSDUCERS 2007. International*, pages 435–438, June 2007.

[52] Keiji Nagatani, Yoshito Okada, Naoki Tokunaga, Seiga Kiribayashi, Kazuya Yoshida, Kazunori Ohno, Eijiro Takeuchi, Satoshi Tadokoro, Hidehisa Akiyama, Itsuki Noda, Tomoaki Yoshida, and Eiji Koyanagi. Multirobot exploration for search and rescue missions: A report on map building in robocuprescue 2009. *Journal of Field Robotics*, 28(3):373–387, 2011.

[53] Hoa G. Nguyen, H. R. Everett, Narek Manouk, and Ambrish Verma. Autonomous mobile communication relays. Orlando, FL, April 2002.

[54] Hoa G. Nguyen, Nathan Farrington, and Narek Pezeshkian. Maintaining communication link for tactical ground robots. Anaheim, CA,, August 2004.

[55] Narek Pezeshkian, Hoa G. Nguyen, and Aaron Burmeister. Unmanned ground vehicle non-line-of-sight operations using relaying radios. In *Proceedings of the IASTED International Conference on Robotics and Applications*, pages 1–6, 2006. cited By (since 1996) 3.

[56] Nissanka Bodhi Priyantha, Anit Chakraborty, and Hari Balakrishnan. The cricket Location-Support system. In *6th ACM MOBICOM*, Boston, MA, August 2000.

[57] John H Reif and Hongyan Wang. Social potential fields: a distributed behavioral control for autonomous robots. In *WAFR: Proceedings of the workshop on Algorithmic foundations of robotics*, pages 331–345, Natick, MA, USA, 1995. A. K. Peters, Ltd.

[58] Julian Ryde and Huosheng Hu. 3d mapping with multi-resolution occupied voxel lists. *Autonomous Robots*, 28:169–185, 2010. 10.1007/s10514-009-9158-3.

[59] Andreas Savvides, Chih-Chieh Han, and Mani B Strivastava. Dynamic fine-grained localization in Ad-Hoc networks of sensors. In *MobiCom '01: Proceedings of the 7th annual international conference on Mobile computing and networking*, pages 166–179, New York, NY, USA, 2001. ACM Press.

[60] M. Schuresko and J. Cortes. Distributed motion constraints for algebraic connectivity of robotic networks. In *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*, pages 5482–5487, December 2008.

[61] B. Shucker and J. K Bennett. Target tracking with distributed robotic macrosensors. In *Military Communications Conference, 2005. MILCOM 2005. IEEE*, pages 2617–2623 Vol. 4, October 2005.

[62] B. Shucker, T. Murphey, and J.K. Bennett. Switching rules for decentralized control with simple control laws. In *American Control Conference, 2007. ACC '07*, pages 1485–1492, 2007.

[63] Sung-Woo Song and Kang-Hyun Jo. 3d mapping and estimation from moving direction of indoor mobile robot using vanishing points. In *ICCAS-SICE, 2009*, pages 3504 –3508, aug. 2009.

[64] Taek Lyul Song. Observability of target tracking with range-only measurements. *Oceanic Engineering, IEEE Journal of*, 24(3):383–387, July 1999.

[65] Michael R. Souryal, Johannes Geissbuehler, Leonard E. Miller, and Nader Moayeri. Real-time deployment of multihop relays for range extension. In *Proceedings of the 5th international conference on Mobile systems, applications and services*, pages 85–98, San Juan, Puerto Rico, 2007. ACM.

[66] E. Stump, A. Jadbabaie, and V. Kumar. Connectivity management in mobile robot teams. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 1525–1530, May 2008.

[67] O. Tekdas and V. Isler. Robotic routers. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 1513–1518, May 2008.

[68] S. Thrun, W. Burgard, and D. Fox. A real-time algorithm for mobile robot mapping with applications to multi-robot and 3D mapping. In *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, volume 1, pages 321 –328 vol.1, 2000.

[69] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, September 2005. Published: Hardcover.

[70] P. Urcola, L. Riazuelo, M. T Lazaro, and L. Montano. Cooperative navigation using environment compliant robot formations. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 2789–2794, September 2008.

[71] Volgyesi, Balogh, Nadas, Nash, and Ledeczi. Shooter localization and weapon classification with soldier-wearable networked sensors. In *MobiSys '07: Proceedings of the 5th international conference on Mobile systems, applications and services*, pages 113–126, New York, NY, USA, 2007. ACM.

[72] You-Chiun Wang and Yu-Chee Tseng. Distributed deployment schemes for mobile wireless sensor networks to ensure multilevel coverage. *Parallel and Distributed Systems, IEEE Transactions on*, 19(9):1280–1294, September 2008.

[73] J. Welle, D. Schulz, T. Bachran, and A.B. Cremers. Optimization techniques for laser-based 3d particle filter slam. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 3525 –3530, may 2010.

[74] S. B. Williams. Towards multi-vehicle simultaneous localisation and mapping. In *Proceedings - IEEE International Conference on Robotics and Automation*, volume 3, pages 2743–2748, 2002. cited By (since 1996) 13.

[75] Lei Yang, Guang Song, and Robert L. Jernigan. Protein elastic network models and the ranges of cooperativity. *Proceedings of the National Academy of Sciences*, 106(30):12347–12352, July 2009.

[76] Weizhong Ye, Yangsheng Xu, and Ka Keung Lee. Shoe-Mouse: an integrated intelligent shoe. *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 1163–1167, August 2005.

[77] M. M Zavlanos and G. J Pappas. Distributed connectivity control of mobile networks. *Robotics, IEEE Transactions on*, 24(6):1416–1428, December 2008.

[78] Zhe Zhang, Goldie Nejat, Hong Guo, and Peisen Huang. A novel 3d sensory system for robot-assisted mapping of cluttered urban search and rescue environments. *Intelligent Service Robotics*, 4:119–134, 2011. 10.1007/s11370-010-0082-3.

[79] X. S Zhou and S. I Roumeliotis. Multi-robot SLAM with unknown initial correspondence: The robot rendezvous case. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 1785 –1792, 2006.

Vita

Sedat Dogru received from Middle East Technical University Turkey his B.Sc. degree in Electrical and Electronics Engineering and B.Sc. degree in Physics (double major) in 2000 and 2001 respectively. He has served in the industry as software developer and project leader for almost 4 years. He is currently a Ph.D. student at the Department of Electrical and Electronics Engineering of Middle East Technical University Turkey. His research interests are mainly Wireless Sensor Networks, SLAM, path and motion planning, and connectivity maintenance.