Universitat Autònoma de Barcelona

Adaptive multi-robot control through on-line parameter identification at system level

Pragna Das

Adaptive multi-robot control through on-line parameter identification at system level

Thesis submitted by **Pragna Das** at Universitat Autònoma de Barcelona in partial fulfillment of the degree of **Doctor of Philosophy**.

Bellaterra, April 13, 2018

Director: **Dr. Lluís Ribas-Xirgo**
Universitat Autònoma de Barcelona
Microelectronics and Electronic Systems Department

The research described in this book was carried out at the Microelectronics and Electronic Systems Department, Universitat Autònoma de Barcelona under the PhD Programme in Electrical and Telecommunication Engineering.

This document was typeset by the author using $\LaTeX\,2_\varepsilon$.

Universitat Autònoma de Barcelona

Microelectronics and Electronic Systems Department

DECLARATION

I declare that I am the sole author of this thesis entitled "**Adaptive multi-robot control through on-line parameter identification at system level**" and that the work contained therein is original, where explicitly stated otherwise in the text.

Pragna Das

Bellaterra (Barcelona), April 12, 2018

Universitat Autònoma de Barcelona

Microelectronics and Electronic Systems Department

ADAPTIVE MULTI-ROBOT CONTROL THROUGH ON-LINE PARAMETER IDENTIFICATION AT SYSTEM LEVEL

Thesis presented to obtain the PhD in Electrical and Telecommunication Engineering

Author: Pragna Das

PhD Advisor: Dr. Lluís Ribas Xirgo

CERTIFY:
that the dissertation "**Adaptive multi-robot control through on-line parameter identification at system level**" presented by Pragna Das to obtain the PhD in Electrical and Telecommunication Engineering, has been done under his direction at the Universitat Autònoma de Barcelona.

Dr. Lluís Ribas Xirgo
Bellaterra (Barcelona), April 12, 2018

This thesis is dedicated to all striving doctoral students

Acknowledgement

This work would not have been possible without the financial support of the PIF award of Universitat Autònoma de Barcelona. I am grateful to my advisor Dr. Lluís Ribas Xirgo for his guidance and support. I would like to thank Dr Vincent Hilaire from Laboratoire Systèmes et Transports de l'Institut de Recherche Transport Energie Société, a la Université de Technologie de Belfort-Montbéliard to accept me as a research intern. Also, colleagues like Lu Wang, Natalie, David, Jorge have also helped me a lot in different phases. Last but not the least my friends and family have been a source of constant support and motivation.

Abstract

Industrie 4.0 is the fourth industrial revolution and is related to the application of the generic concept of cyber-physical systems [30]. The outcome of many projects aligned with Industrie 4.0 proposal includes multi-robot systems (MRS) for manufacturing which are more adaptable, decentralised, service oriented and have real-time control and modularity [37]. One of the most crucial aspects of an MRS is the organization of the flow of control decisions. The control and planning have to accomplish different tasks like collaboration and cooperation, order assignment, mitigating collision and maintaining communication among all the robots [39].

The states of the batteries, robots and environment change often in autonomous systems like MRS. The time required to complete tasks or **performance time**s vary according to the change in states of batteries, floor and mechanical parts of the robots.

In most general cases where conditions of the floor, mechanical parts are important, the relation between performance times and these factors are not directly derivable, through, performance time has a direct correlation with discharge of batteries. This work focuses on identifying these performance times and model them so as to reflect the states of the batteries and environment. The inclusion of these performance times in planning and control decision can produce more cost-efficient decisions. They are modelled using non-linear state dependent modelling techniques. Also, they are estimated using the model developed in this work and the values are used in the decision-making process. This is done to study their efficacy in planning in an MRS used for internal transportation.

A prototype MRS for logistics is prepared for the experiments where the floor is described by a topological map with nodes designating ports or junctions and edges connecting the nodes. The robots carry materials from one port to another. The paths between ports are composed of different edges. Traversing an edge is considered as a task. The travel time spent by a robot to traverse an edge is the designated performance time which indicate the cost involved to traverse the edges. Thus, travel times reflect continuous changes in robot and environment and influence to modify decisions which are made without considering them. Travel times for a robot are made available online through estimation. Online estimation demands a suitable formalization and model for these travel times. In this work, a state-dependent bi-linear model from time-series modelling technique is used to model travel times in each mobile robot in order to estimate them online. The efficacy of these travel times is studied by complementing them into a planning algorithm. Routes are computed to find the shortest path from one port to another for each MR. The travel times

are used as weights of edges into route planning instead of weights derived from heuristics or other cost factors. In route planning for a single robot, the travel time for an edge may be required to be estimated for multiple times. The estimated travel times between any two nodes provide the current and close-to-real cost of traversing at different instances. These estimated values form a profile of travel costs for edges through the duration of operation of the robot. These estimated travel times are different than heuristics costs as they depict the real states which are impossible to know from heuristics. This facilitates path planning algorithms to choose the edges with least real travel times or costs to form the path. The experiments show that path obtained through online estimated travel times are of 15% less total cost compared to that obtained by heuristics costs.

Nevertheless, a good estimation is dependent on historical data which are close in time. But, there are situations when all the travel times for one or more edge(s) are not available for the entire duration of operation of the MRS to an individual robot. The proclivity of this occurrence lies in the fact that the edge may not have been travelled even once by the robot, or travel time for that edge have not been recorded in recent past. Then, it is imperative for that robot to gather the necessary travel times from others in the system as a reference observation. But, these observations are from other robots in different battery condition than itself. Still, the bi-linear model for travel time for the robot itself using other robots' observation and its own change or exploration in the travel times till the current instance. The crux of this process is to predict current travel times in the robot using others' travel time for the same edge.

The mechanism of information sharing between one robot to others in the system has been devised in a form of a common ontology-based knowledge. This ontology structure is identical in each robot which contains the travel times of edges with contexts attached to each data about the instances of estimation, the nodes that particular edge connects and other pieces of information. This ontology helps to fetch and share information forming a collective knowledge base facilitating a comprehensive control and planning for the system.

This greatly helps the MR to estimate travel times more accurately and precisely. Also, accurate estimation affects route planning to be more precise with reduced cost. The total cost of paths generated through the travel times estimated through sharing is 40% less on average than that of paths generated through travel times without sharing.

In this work, only a single task is considered whereas in a real industry a robot needs to do a variety of tasks. This work paves the way to consider all the varieties of tasks in an automated system and identify different types of cost coefficients, other than travel times. In that case, estimating and sharing information would be in a bigger domain with more complexity which demands artificial intelligence to be used along with re-enforcement learning. The problem domain can be further enhanced with different kinds of robots in a system like unmanned aerial vehicles, other ground vehicles, and human agents.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Flexible manufacturing plants, warehouses and logistic hubs depend on efficient, robust, adaptable and evolving internal transportation systems [63]. In the recent years, these systems are built upon mobile robots or automatic guided vehicles (AGVs) or mobile robots (MRs) which are driver-less carriers used on the factory floor as mobile production platforms or for material transportation. With the use of AGVs, the production process has become more flexible with a decrease in production time and costs. With the advent of Industry 4.0, the multi-robot system (MRS) came into vogue into industries.

In this chapter, the current requirements in MRS for more cost effective decisions and the problem of this thesis shall be introduced. Also, this chapter elaborates on the necessity and feasibility of the problem in automated factories.

1.1 Context of the problem

Robotics, both as a research field and an application domain, has come a long way from traditional singular robots to collaborative or collective robotics termed as multi-robot systems. In comparison to singular robots, MRS are advantageous for features like collaborative work, decrease in overall cost, more fault tolerant, capability to encompass large area *et cetera* [78]. This kind of autonomous systems in industrial production facilities are one of the facilitators of Industry 4.0 and smart factory. Thus, multi-robot systems are extensively investigated for efficient functioning in automated factories.

Automatic control systems are required to take decisions on real-time based on the available information, without outside interference to cope up with the challenges arising out of the above requirements. For such autonomous operation, awareness about states of robot, charge of batteries and environment play crucial role for real-time decisions. Decisions in an individual robot based on these states can increase operational efficiency, which can eventually increase the efficiency of the team [16]. However, these states are not always available to an mobile robot (MR) due to lack of experience of certain parts of the system or having been experienced quite long back which may be irrelevant at the time of decision making. Nevertheless, all MRs in the

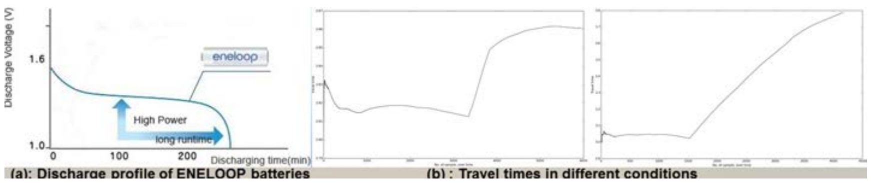


Figure 1.1: Correspondence between discharge of battery and travel time of MR

system can support and help each other to obtain these states through knowledge sharing mechanism. This paves the way for implementing collective intelligence for better optimal decisions.

1.2 Travel time as indicator of state of batteries and environment

The MRS is applied to carry out internal transportation in this work. Generally, the environment in internal transportation comprises of racks to keep materials which are marked as ports. There are pathways in between the racks in the floor through which the MRs traverse to transport the materials. In this context, the time required by any MR to perform some task of carrying a load depends on various environmental as well as internal factors of the robot. These travel time to carry a load will be more if the battery state is exhausting or floor is rough or both. Thus, travel time depicts the cost incurred due to performance of task.

In an experiment conducted in our laboratory, we observe close correlation between state of charge and time taken to traverse distances. At the first part of the experiment, a MR was instructed to travel a particular distance repetitively till the battery is fully exhausted from state of complete charge. The values of the travel time was recorded in secs as a function of iteration or time x_k. In the next step, the floor condition was made rough from smooth and again the MR was instructed to travel that distance repetitively till the battery is fully exhausted from state of complete charge. Part (a) of Figure 1.1 plots the discharging voltage of Lithion-ion batteries over time and Part (b) plots the progressive mean of observed values of x_k, first only with the change of state of charge of batteries (left plot) and second with both the change of state of the charge of batteries and the floor condition (right plot). Analogy is observed in Part (a) and left part of (b) where travel time is in accordance with the battery discharge profile where it increases for the first 100 iterations, then again decrease to a steady value and then increases steadily till full discharge. This behavior is also present when surface conditions are changed from smooth to rough, where the increase of travel time is more after 3500 iterations. On the other hand, travel time increases more in right plot compared to left plot of Part (b) at the same iteration instance. This happens because at equal battery capacity in both cases, more energy is required to traverse the same distance with increased roughness in the floor. These observations shows that travel time taken to cover a particular distance by a MR can provide a quantitative measure for the state of batteries and state of environment.

1.3 Problem definition

Quantitative components like *travel time* can be formalized as cost coefficients or cost functions to obtain and understand the different environmental, physical and mechanical factors of any MRS. Our current work involves identifying these cost coefficients in a MRS specifically used for internal transportation in automated plants. In this work, traffic network topologies are used instead of geometrical maps as the floor plan for automated plants usually do not vary over the time. As vehicle speed depends on friction forces, slope and traffic conditions, state of batteries, the actual cost of traversing a path is provided by travel time. These travel times are time dependant as battery exhaustion, surface condition of shop floor, wear and tear of tyre, *et cetera* vary along time. An optimal path obtained at certain instance may not remain to be optimal at some other future instance as the real cost of traversal is denoted by travel time. Thus, these travel times have the potential to effect the decisions on optimal path. This work focuses on to model and estimate these travel time along the progress of time. Moreover, as they can influence decision making, the problem is to use these travel times as the main parameter for decisions to obtain more cost efficient paths.

Online estimation of travel times in an MR for an edge need observations of the same throughout the history of operation time. But all the observations may not be available to that MR as it may have not traversed the edge before or may have traversed long back. In later case, the current status of the floor for the edge will not be known to the MR. During the run-time of MRS, the values of the estimated travel time will be generated at every instance f control decisions for all MRs, producing a pool of estimated values. One or more of the other MRs may (one must have) traversed this edge in the previous instance or close past whose travel time can be used to estimate the travel time for the MR at current instance. Hence, representation, accumulation, storage and sharing of these travel times become essential [55] in an MRS. More significantly, every estimated value of travel time has inherent context associated with it. Travel time originate from each MR depending on the instance of travelling, zone of the floor, previously traversed edge, neighboring AGVs, state of charge, *et cetera.* All these factors provide context to the estimated values of travel time. The problem addressed in this work is to form a mechanism to share the travel times of each MR efficiently to be used for accurate estimation of travel time for every MR during their decision and control. Also, it is investigated to form a collective intelligence in the MRS so that the knowledge and intelligence of one robot can contribute to the formation of knowledge in other robots as well.

1.4 Summary

The contribution of this dissertation include the following:

- Travel time of edges is identified as a type of cost parameter which reflects the states of charge of battery and floor in case of automated logistics
- Kalman filtering has been identified as a efficient and suitable method for estimation of travel time

- Dynamic parameter identification and estimation method have been developed to estimate travel time online
- Standard route planning algorithm has been modified to use travel time as weights of edges to find minimum cost path
- Collective intelligence have been implemented in MRS with mathematical validity so that each MR can help and support other MRs in their decision process
- Inside collective intelligence, semantic knowledge sharing mechanism is devised in each robot to share estimated values of travel times of one robot to others

1.5 Organization of this dissertation

In this introductory chapter, the context and motivation of the current problem are presented along with the contributions. The second chapter provides insight for the state of the art works in the problem area and provides a comparison of the approach of the current solution with state of the art. Chapter 3 explains the study to find the suitable estimation method for estimation of travel times of an MR. Chapter 4 elaborates on the methods to estimate travel times online and compliment traditional planning algorithms in order to produce better cost-efficient path. The knowledge sharing mechanism and formation of collective intelligence are presented in Chapter 5. Chapter 6 provides the conclusion of this dissertation.

Chapter 2

Related work

The context of the problem, the problem definition and contribution are elaborated in the previous chapter. The state of the art and current trends are discussed in this chapter. The evolution of multi-robot systems for smart factories are discussed in this chapter. This follows with the discussion of state of the art implementtion of collective intelligence in industrial automation.

2.1 Growth of distributed multi-robot system

The research of MRS has grown to a multi-dimensional field from the time of its inception at the late 1980s [7] including biologically inspired multi-robot formations, communication between robots, architectures, task allocation, motion coordination and control, localization, mapping and exploration, object transport and manipulation and reconfiguration. Usually, multiple robots cooperate and coordinate to perform complex tasks which are typically impossible for one single robot powerful robot to accomplish [34]. After the advent of behavior-based control where the desired behavior in system-level is achieved by distributed, interacting modules, called behaviors [53], the control techniques of various biological societies like ants, bees, and birds are deployed to the development of similar behaviors in cooperative robot systems[7]. Through this concept, robots can cooperate with each other to achieve complex behaviors [34], which could be impractical and expensive to carry out by a single powerful robot [79]. There were algorithms being proposed forming a centralized multi-robot control system. However, centralized control in MRS makes commander robot most functional, not only mechanically but computationally which makes the system more vulnerable to crashing, as the leader becomes the single point of contact. Thus, decentralized and distributed control in MRS came into vogue to make it fault tolerance, reliable and robust [6]. A novel flocking algorithm was proposed in [42] based on the leaderless approach where a large scale swarm of robots could navigate autonomously trusting on local interactions in an environment, dense populated with obstacles. Thus a specific robot is not assigned to conduct the group. Coordination and control become utmost importance in any MRS to achieve the former. This marks the genesis of swarm intelligence in robotics or swarm robotics systems (SRS)

[79].

2.2 Advent and propagation of swarm intelligence in multi-robot systems

Swarm intelligence is a bio-inspired approach, where cumulative behaviors of groups of organisms, enables the group to solve problems, typically impossible for a single individual to solve [44]. So, swarm intelligence can be seen as a mechanism which individuals can use to overcome some of their own cognitive limitations. Swarm intelligence results in a global emergent behavior to manage complex system using minimal communication with only local neighbors [79], which do not depend on a single commander. These operational principles from biological systems of animal kingdoms are deployed as swarm intelligence in engineering as SRS for some basic advantages over traditional systems like scalability, flexibility, and robustness [23]. Gradually, the simple and individual behaviors lead to complex and divergent ones in a MRS. This paved the way to achieve the current concept of collective intelligence in MRS. [51].

2.3 Collective intelligence in MRS

A set of intelligent entities like experts, agent systems, or simply a set individuals which are autonomous to make decisions, is termed as a collective [51]. Usually, each member contains its own knowledge but, the whole amount of knowledge in the collective is not a sum of knowledge of each member [61]. One possible solution is proposed in [19] by applying basic data integration techniques in real world application. Thus, integration is a method which can produce knowledge of the collective from knowledge bases of individuals. In the context of swarm robotics, independent robots function as agents and are acting together to accomplish a big task. Each robot or member forms a knowledge base in the system or collective while it functions, guided by reacting to the local situations as they encounter. This mechanism eventually forms a collective intelligence. The whole knowledge base of a collective is represented as a general world utility which is a function of the state of all agents across all time [70]. Moreover, a function of only one agent's state at a single moment is a specific private utility function. The knowledge of a robot as a member of an entity is being acquired by the sensors and similar sources. In a collective or group of individuals, the whole knowledge of the collective or world knowledge is formed in an utility function. For optimized control and coordination in a MRS using collective intelligence, world utility function is maximized through the proper selection of private utility functions. The principles of collective intelligence and crowdsourcing have been investigated in [75] to achieve information fusion or integration in multi-robot search and rescue systems, in cooperation with humans. Although, there are numerous approaches found in the state of the art where the concept of information fusion is being used to process sensor data and data from other similar sources to accomplish search or rescue or navigating in unknown environment, there are no investigation carried out to use the concept of collective intelligence to improve the efficiency of performance of a group of MRs. However, the decisions of planning and control for each robot can be improved when it is based on information about its own state and the environment. Thus, these information of each robot or each member of the collective have to be gathered through sensor or estimated from previous experience of performances. Moreover, similar to [75], these information can be used to maximise the efficiency of the whole MRS using the principles of collective intelligence. Thus, planning decisions in each member entity or robot can be based on the derived information about the environment and other robots, which will eventually make the MRS more cost efficient.

2.4 Information about each robot and environment

In our work we propose to gather on-line information of conditions of robot and the environment in a MRS through travel time, which is conceptualized a type of cost parameter. Typically, these cost parameters are time varying as explained in Section 1.3 in Chapter 1. Although, there are many recent state of the art works related to multi-robot systems in transportation and automated industry, dealing with co-operative path-planning, parameter identification of robot dynamics, adaptive controlling through position control, most related to our work are the followings:

In [54], the authors have proposed a adaptive on-line estimation of system parameters which are time-varying. Here, the model parameters of the dynamics of mobile robot of one robot and then of all the N robots of the system are estimated as system parameters to arrive at a coherent estimate. Although the time varying aspect of the model parameters is similar to our work, we are focused on determining to estimate from performance quality or capability of the mobile robots which are impacted by the different environmental factors and which determine the cost of doing task. We call these as cost parameters and these are time-varying in nature.

The work of Confessore, Fabiano and Liotta in [21] has proposed a minimum cost approach to solve the dispatching problem in an multi-robot system where the network of mobile robots is represented as a graph. The final goal of their work is to determine the minimum cost of performing a task through this graphed network. Like *Confessore, Fabiano and Liotta*, we are also determining the performing ability or quality considered as cost, but our final goal is to use these cost parameters of each AGV in the system to be useful for various decision and controlling purposes.

Though both the above works can be considered most relevant to the current work, these are not directly comparable as they used different time varying parameters differently. In current work we estimate traversal time of particular AGV as an instance of above mentioned cost parameters. There is no research proposal addressing the same problem as ours. This is the first work to address this type of parameters.

2.5 Summary

Numerous works have been done to form a centralized multi-robot control system. But, centralized control is prone to crashing as it is dependent on just a single commander root. This necessitated the evolution of decentralized and distributed control in MRS for fault tolerance, reliable and robust performance. With the advent of swarm intelligence and multi-agent systems, collective intelligence came into vogue in MRS where any intelligent entity like flock of robots, agents *et cetera* are termed as collective. The total knowledge of a collective is maximized than the knowledge of each member through data fusion technique. Usually, knowledge in a collective is the data about the surrounding area acquired through various sensor and similar sources. In most cases, these data are used to enhanced search and rescue operations in MRS. In our proposal, the travel time is proposed as a type of cost parameter which represents the states of batteries of robots and floor in automated logistics. In this work, collective intelligence is used as a method to enhance the knowledge of these travel times for better cost efficient decisions in each MR in the system. This eventually enhances the efficiency of whole MRS.

Chapter 3

Travel time as cost coefficient and its estimation

The main features and demands of MRS are discussed in the introductory chapter. Autonomous systems like MRS need to decide for cost efficient planning online based on the available information about the environment and other robots. In state of the art, this knowledge has been maximized using data fusion techniques based on sensor data through collective intelligence. This method has been implemented mostly in case of search and rescue operations. This work proposes cost parameters which reflect the current states of battery, mechanical parts of robots, environment *et cetera* and finds them suitable to reflect knowledge about robots and environment. They could be used in decisions for planning to generate cost efficient results. Using collective intelligence, these cost parameters can be shared and total knowledge about these can be enhanced in the MRS. This work proposes to increase the cost efficiency of planning decision through collective intelligence in the MRS. These parameters change along time, particularly in battery-operated robots, which are very sensitive to battery level variations. This chapter elaborates on identification and estimation of such a cost parameter of the individual MRs in context of automated logistics.

3.1 Challenges in planning in MRS

The trend in MRS is to use driver-less diesel or electric powered MRs. In [71] *Schmidt et. al.* has investigated the commercial viability of Battery powered robots (B-MRs) and has concluded that the battery mobility is economically beneficial because the charging and maintenance costs of a group of B-MRs are significantly lower [1]. Moreover, the cost of maintaining B-MRs can be instrumental in requiting the higher investment costs of procuring charging infrastructure and spare batteries. Thus, B-MRs have economic, technical and ecological advantages over diesel-electric powered ones in applications of transportation and material handling [71]. B-MRs have ushered in more flexible production process with decrease in production time and costs. So,

[1]This is explained elaborately in [26]

B-MRs are gaining more importance.

On the other hand, the control and planning decisions in an MRS are still based on heuristic computations. But, the states of environment, mechanical parts change during the progress of time. Moreover, in case of B-MRs, state of charge of batteries also change over time. Currently, realistic conditions of robots or environment or both are neither reflected nor instilled in decisions of planning. Hence, the decisions for task allocation, navigation and movement can eventually lead to increase of overall cost of performing the tasks. Also, planning decisions for one MR (for individual task) or for a group (for co-operative tasks) are taken independent of other MRs operating in same environment. However in automated systems, performance of one MR is not only dependent on its own mechanical parts but also state of batteries and environmental factors like floor condition, positions and functioning of other MRs. Thus, planning decisions in MRs, either deployed as an individual worker or in a group, necessitates to estimate performing cost in form of parameter from environmental factors and state of charge of batteries *et cetera* [72].

For a comprehensive idea, we provide the following example. A prototype internal

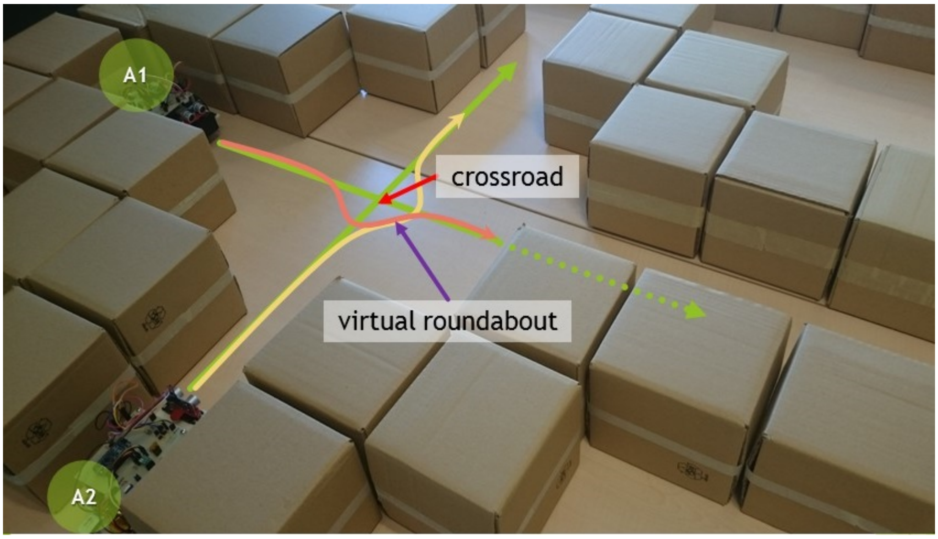


Figure 3.1: Collision at crossroad

transportation system is illustrated in Fig 3.1, where two transportation MRs are about to meet at a crossroad in their respective transportation paths. The illustrated situation arose due to the lack of knowledge about the current capability of the MRs based on their battery level at run-time. Thus, decision for movement was based on incomplete or incorrect prediction about time required by the MRs to reach the crossroad and to cross it.There are two possible solutions to mitigate the collision between them. One way is to make $A1$ wait until $A2$ passes the crossroad and go to its path from the crossroad (marked by solid arrow). After that, $A1$ should pass and go to its own path (marked by dotted arrow). The other one is to coordinate the movements of $A1$ and $A2$ to pass the crossroad without collision using virtual roundabout and go to their respective paths. The first solution is easy and simple, but it decreases the efficiency of $A1$ by forcing it to wait and thus incurs more cost to complete transportation. On the other hand, the second solution is beneficial, in terms of cost and efficiency, as it utilizes the available resources, yet making the task less cost consuming by continuing with the movements of both robot in parallel. The second solution can be implemented when the knowledge of exact time needed to cross the roundabout by each MR can be obtained. This specific time is based on the state of charge of batteries, mechanical status of the robot and the conditions of the environment. Thus continuous real-time knowledge of travel time which reflects state of charge of batteries helps to make better optimal control decisions.

The above example shows that to obtain better path planning, parameters considered in existing decision process is not sufficient. In addition, the controller needs to consider the current speed of individual MR as a parameter in order to correctly take decisions. As these parameter are time varying in nature and directly influence the cost of the current task (consequently the whole system), these can be modelled as cost parameters to the system. As for the above example we model the current speed by travel time.

3.2 Travel time: a type of cost parameter

Aforementioned travel time is a kind of cost coefficient. These cost coefficients can be of individual robots in MRS, irrespective of the kind of task the robot needs to perform. Similar situations can be thought of in other tasks e.g uploading a box or any material, capacity of carrying a particular load or accuracy of fitting a mechanical part. On one hand, they arise locally at each individual AGV due to action of actuators, wheels and other mechanical factors. But they are significantly influenced by environmental factors like battery capacity (in case of battery powered MRs), conditions of the floor, conditions of material, performance and behavior of other AGVs, *et cetera*, as all or most of these factors determine the state of the robot at every instance of time. The experiments show that these travel time vary over several environmental factors as mentioned above (Section 1.2 in Chaper 1).

Current research works on control of MRS estimate these parameters at agent level. The decisions of planning, like the coordinated movements of $A1$ and $A2$ to pass the crossroad in the previous example, can not be taken at lower levels of actuation and control. Influence of factors like friction forces of floor, slope, mechanical part can be corrected by local control on individual MR level (lower levels), but factors like traffic condition, conditions of material, behavior of other MRs are beyond the scope of control by lower levels. Hence, considering cost coefficient at lower levels of actuation and control cannot make better control decisions. On the other hand, there are other research works which consider some heuristic cost for system level decision making. But these cost do not reflect the real conditions as they are not derived from the current and real states. In case of planning, the method of achieving a decided goal in inherent in these cost parameters. But, they are applied in the lower levels. Thus, planning goals are set in high level but the parameters to achieve them are available to lower level. This obscures the path to achieve the high level decisions like computing optimal path or task allotment. We are trying to bridge this gap where the correct estimates of cost parameter like travel time, which evolves over time, can be provided for various purposes of automating the processes and reach the correct goal at current time. Therefore, we want to apply them at higher levels.

3.3 Contribution

The reported works in the field of MRS are mostly concerned with the following two categories:

- Problems with the cooperative and collaborative functions like scheduling [57], task allocation [69], path-planning [2]
- Problems related to individual robots like localization [29], dynamic and physical parameters identification and estimating [35], position and orientation estimation [24] and obstacle estimation [45]

However, none of these proposed any solution for estimation of travel time as a parameters, though the current trend and directions of investigations for smart factories enabling Industry 4.0 are focused on conditioning of available data and states of the robot [13]. This work focuses on identifying the cost parameters for internal logistics to reflect the states of robots, their batteries and the environment. In this work, study is done to form travel time as one kind of cost parameter for traversing tasks in logistics. Also, the suitable method for online estimation of these travel times is investigated in this work. [2].

3.4 Insights of experimentation platform

In order to provide elaborate explanation of the problem, the experimentation platform is briefly described in this section.

3.4.1 Environment for the MRs to work

A scaled down prototype of automated indoor logistics system is built for experimentation. An environment has been developed using uniform sized boxes as shown in Figure 5.7 for the robots to work, first doing single task at a time and then advancing towards two tasks at a time. Thus, a robot doing one task at a time is named as single-task robot. Similarly, a two-task robot can finish two tasks at a time. The boxes create a closed labyrinth path to navigate in a given path. Also designated ports are marked on the boxes.

3.4.2 Description of shop floor

The description of the floor of the prototype factory is provided in Figure 3.3. The floor is described by means of a directed graph $\mathcal{G} = \{\vee, \varepsilon\}$, where the ports or bifurcation points corresponds to a node $n_p \in \vee$ and each link between any two nodes,

[2]These are explained elaborately in [25]

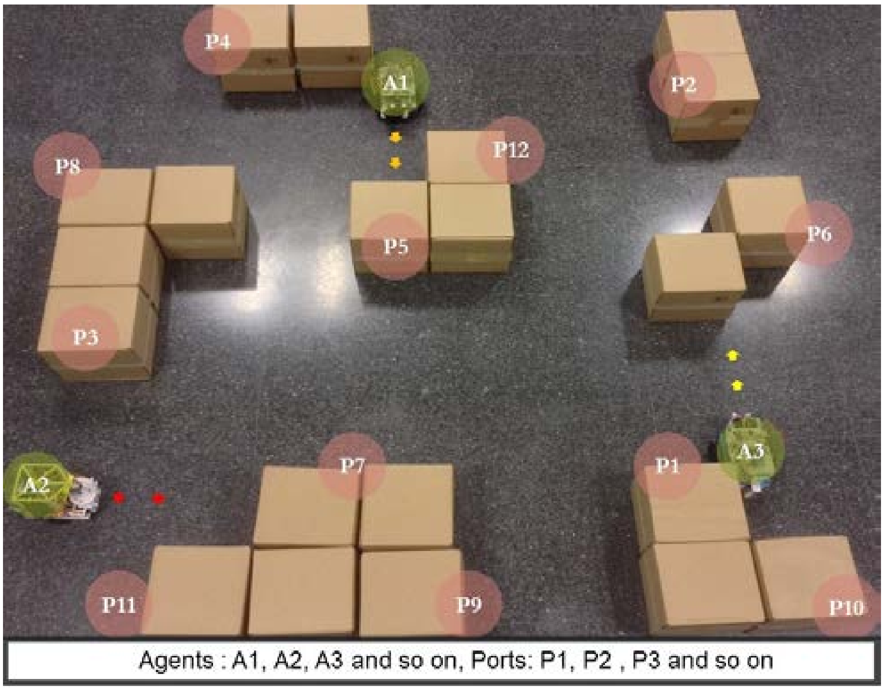


Figure 3.2: Environment of MRS

namely n_p and n_q corresponds to an arc $a_{p,q} \in \varepsilon$. In Figure 3.3, notation like n_a designates a node and edges are marked as $a_{a,b}$. In the map, some nodes are not connected to the neighboring node as there is no connecting edge between them. For example, n_f and n_g seems to be neighbors of each other, but physically a robot cannot go from n_f to n_g as the space between these two nodes is very little for any AGV to move. Also, not all ports are accessible by all its neighboring nodes for the same reason as stated above. This can be observed in case of P_{19} which is impossible to access by n_s although it seems neighboring to P_{19}.

3.5 Problem formulation

In this work, **reaching a particular port** by an MR is considered as a task and route computing is considered as a decision making process. Thus an MR is required to traverse from one node to another, according to the map, which enables the MR to perform single task at a time. The travel time of each arc in a map similar to the floor map given in Figure 3.3 is influenced by energy exhaustion, condition of floors, physical parameters of robot, among others, which incurs cost. Thus time to traverse an arc by an MR can be conceptualized as its cost coefficients. Hence, this coefficient is formalized as $X_{p,q}(k, e, b, f)$ to denote travel cost from n_p to n_q, where k is the instance of time of traversing an edge, e denotes state of charge of batteries, b is for tire condition, f is for frictional force of the floor. $X_{p,q}(k, e, b, f)$ is time-varying from the perspective that at a particular instance of the time, the cost of that particular

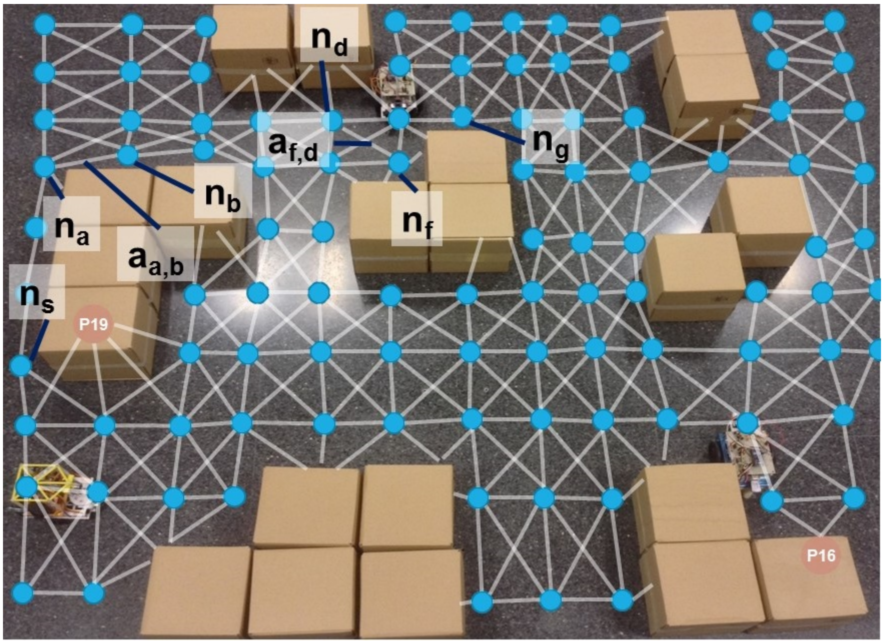


Figure 3.3: Shop-floor described by topological map

arc is dependent on battery discharge, condition of the floor, et cetera which changes over passage of time. Hence, for the arc $a_{a,b}$ in Figure 3.3, travel cost is denoted by $X_{a,b}(k, e, b, f)$. Also, travel cost of the same robot is $X_{f,d}(t, e, b, f)$ for the arc $a_{f,d}$. The $X_{f,d}(t, e, b, f)$ is simplified as $X_{f,d}$ ignoring the terms on which travel time is dependent. Further, $X_{f,d}$ is expressed as X_k in a a general form for any edge at time instance k. This work focuses on finding a suitable method to estimate one such X_k to get the next prediction on the next time instance on-line and recursively update all the predicted values of X_k till full discharge of battery in order to pave the path for exploring its possible utilization in any decision making process.

3.6 Approaches for estimation of parameters

The standard methods usually used for parameter identification and estimation in the state of the art research proposals include least-square estimator approach [33], least-square moving window method (LSMW) [73], recursive least square (RLS) method [73]. Both LSMW and RLS methods are deployed for impedance parameter estimation of state models for robot control in [73].

In LSMW, the data set (X, Y) of length L is such that,

$$Y = X\theta + W \tag{3.1}$$

where, X^T=(x_1,x_2,x_3,......,x_L), Y^T=(y_1,y_2,y_3,......,y_L) and W is the measurement noise.

Now, with a window size $l \in N$ such that $l < L$, the number of estimations of θ

will be $L - l + 1$. The estimation is given by,

$$\hat{\theta}_i = X_i^{\#} Y_i \tag{3.2}$$

where,

$$X_i^{\#} = \left(X_i^T X_i\right)^{-1} X_i^T \tag{3.3}$$

and $Y_i^T = (\ y_i,\ y_{i+1},\ ....,\ y_{i+l-1}),\ X_i^T = (\ x_i,\ x_{i+1},\ ....,\ x_{i+l-1}),\ i = 1,2,......,L - l + 1$ with the estimation error

$$\hat{e_i} = Y_i - X_i \hat{\theta_i} \tag{3.4}$$

. For our application, set Y is our observed data Y_k and set X is the travel cost X_k to be estimated for every k.

Also, [73] has suggested a RLS algorithm based on both constant and variable forgetting factor. After the least square method, the estimate obtained at time t is

$$\hat{\theta}_t = \left(X_t^T X_t\right)^{-1} X_t^T Y_t \tag{3.5}$$

where, $Y_t^T = (\ y_1,\ y_2,\ ....,\ y_t),\ X_t^T = (\ x_1,\ x_2,\ ....,\ x_t)$, the estimation of time $t + 1$ is calculated as

$$\left.\begin{aligned} \hat{\theta_{t+1}} &= \hat{\theta}_t + K_{t+1}\left(y_{t+1} - x_{t+1}^T \hat{\theta}_t\right) \\ P_{k+1} &= \frac{P_t}{\lambda + x_{t+1}^T P_t x_{t+1}} \\ K_{t+1} &= P_{t+1} x_{t+1} \end{aligned}\right\} \tag{3.6}$$

where, λ, According to [73] is the forgetting factor which needs to be carefully set which is a design issue. For time-varying λ a good approach is to set it to a function of estimation error $\hat{e_t}$ as in equation 3.8, which enables more accurate tracking than constant forgetting factor given in equation 3.7.

$$\lambda = 1 - \alpha_1 \left(\frac{1}{\pi} \arctan\left(\alpha_2 \left(|\hat{e_t}| - \alpha_3\right)\right) + \frac{1}{2}\right) \tag{3.7}$$

where, α_1, α_2 and α_3 are all design parameters.

$$\lambda_t = \begin{cases} 1 - \frac{\alpha_3}{\pi} arctan(|R_t - 1|), if\ |R_t - 1| \geq \alpha_2; \\ \alpha_1 + \frac{1}{\pi}(1 - \alpha_1)(arctan(1 - |R_t - 1|))else; \end{cases}$$

$$R_t = \begin{cases} max(\frac{\theta_{t-k}^{ij}}{\theta_t^{ij}}, \frac{\theta_t^{ij}}{\theta_{t-k}^{ij}}), if \theta_{t-k}^{ij} \theta_t^{ij} \neq 0 \\ \infty else. \end{cases} \tag{3.8}$$

$\forall i \in (1, 2, ..., n)$, $\forall j \in (1, 2, .., m)$, with $k, \alpha_1, \alpha_2, \alpha_3$ tunable parameters, $\frac{1}{3} \leq \alpha_1 < 1$, $\alpha_2 \geq 0$, $0 \leq \alpha_3 \leq 2$, $k \in N$
Here also, for our application, set Y is our observed data $Y(k)$ and set X is the travel cost X_k to be estimated.

The KF is often used when the parameter is linearly time-varying. The system state vector was measured and the unknown dynamic parameters are estimated using a Kalman filter (KF) [68]. The equation of the system model which typically uses KF is given as:

$$\begin{aligned} x_{k+1} &= A_k x_k + B_k u_k + G w_k \\ y_k &= C_k x_k + v_k \end{aligned} \tag{3.9}$$

where, x is the parameter to be estimated and y is the observation of x. Also, $x(k) \in \mathbf{R^n}$, $u(k) \in \mathbf{R^n}$, $w(k) \in \mathbf{R^n}$, $v(k) \in \mathbf{R^r}$ and $y(k) \in \mathbf{R^r}$. Moreover, $w(k)$ and $v(k)$ are white, zero mean, Gaussian noise.

In our experiment, x is deployed as our travel cost variable X_k and Y_k as the observation. So, equation 3.9 is transformed to equation 3.10 with X_k and Y_k.

$$\begin{aligned} X_{k+1} &= A_k X_k + B_k u_k + G w_k \\ Y_k &= C_k X_k + v_k \end{aligned} \tag{3.10}$$

The KF results from the recursive application of the prediction and the filtering cycle. Equation 3.11 results from applying the prediction cycle on the model given in equation 3.10.

$$\begin{aligned} \hat{X}_{k+1|k} &= A_k \hat{X}_{k|k} + B_k u_k \\ \hat{P}_{k+1|k} &= A_k P_{k|k} A_k^T \end{aligned} \tag{3.11}$$

Equation 3.12 results from applying the prediction cycle on equation 3.10.

$$\left.\begin{aligned} \hat{X}_{k|k} &= \hat{X}_{k|k-1} + K_k[y_k - C_k \hat{X}_{k|k-1}] \\ K_k &= P_{k|k-1} C_k^T [C_k P_{k|k-1} C_k^T + R]^{-1} \\ P_{k|k} &= [I - K_k C_k P_{k|k-1}] \end{aligned}\right\} \tag{3.12}$$

where, $\hat{X}_{k+1}$ is the new estimation for the variable X_k. K_k in equation 3.12 is the KF gain.

3.7 Experiments to find suitable estimation method

3.7.1 Experiment-I

In our previous experiment, described in Section 1.2, a close correlation between discharge profile of batteries and the travel time is observed. Moreover, dependency of travel time over smoothness or roughness of floor is observed. We present here the same plot of observations for better understanding. As explained before, Part (a) of Figure 3.4 plots the discharging voltage of Li-metal batteries over time and Part (b) plots the progressive mean of observed values of X_k, first only with the change of state of charge of batteries (left plot) and second with both the change of state of the charge of batteries and the floor condition (right plot). The observed data for X_k in Part (b) start from the high magnitude as the discharge of batteries is more at the beginning according to plot in Part (a). Then it follows a steady fall as the discharge decreases and then follows a constant value for a significant span of time as

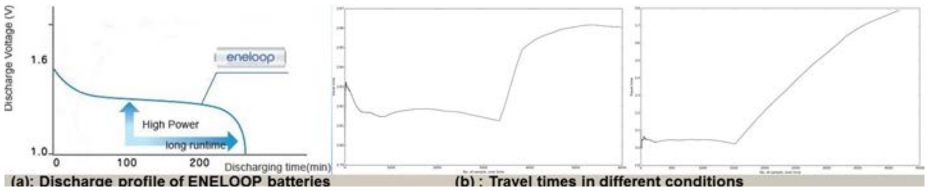


Figure 3.4: Correspondence between discharge of battery and travel time

discharge becomes stagnant. Finally, it again rises up to a big magnitude before the robot finally stops moving as discharge is heavy at the end. Thus travel time varies with state of charge of batteries, according to the variation of discharge, the condition of floor remaining the same.

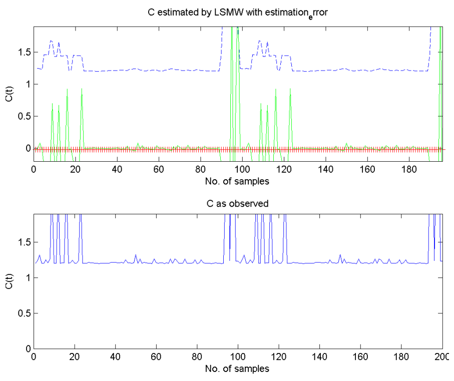
In the right plot of part (b), travel time is initially less as the floor was smooth with the same discharge level as previous. The travel time increases further to great magnitude steadily when the floor is made rough from smooth. Despite the decrease of discharge of batteries, the travel time increases steadily and thus it can be concluded that the influence of roughness plays more on travel time than discharge of batteries. Also, so travel time is greatly influenced by conditions of the floor, in addition to the effect of battery power. Also, it was concluded that travel time needed to traverse a specific distance by an MR reflects a quantitative measure of state of charge of batteries and conditions of floor. The factors like state of charge of batteries, conditions of floor are representative cases of utility based factors which determines cost expended to complete performance. Thus, travel cost serves as a quantitative measure for expended cost to complete traversal.

3.7.2 Experiment-II

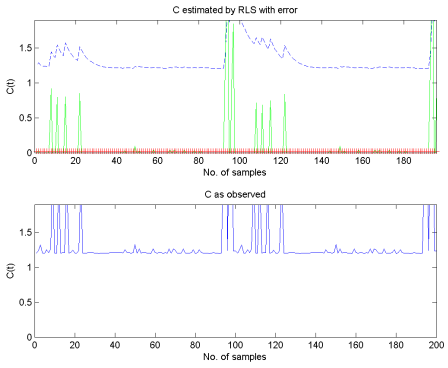
The goal of this experiment is to find a suitable estimating method for time-varying travel time, among the standard estimation methods typically used for estimations of state model parameters of a robot in the state of the art. We are estimating the travel time, which are different from the estimations generally investigated in the state of the art using these standard methods in order to find the most suitable method to estimate these travel times online. We deploy LSMW method [73] first because it is a naive, inexpensive approach to estimate variables on-line. We also deploy the RLS algorithm proposed in [73] with time-varying forgetting factor (equation 3.8) as it helps in accurate tracking. Thereafter, we deploy KF method to further enhance the accuracy.

3.7.3 Results of experiment-II

As discussed in Section 3.7.2, we applied three parameter estimation methods to estimate one type of cost parameter *travel time* designated X_k. The plots in this section shows the estimation result of the methods in upper part and the observation in the lower part. Also, blue line is for the estimated values, the green line stands for the error values and the red thick is the mean error value. The Least square moving window (LSMW) method is implemented first to find the estimation, with The window size as 5.

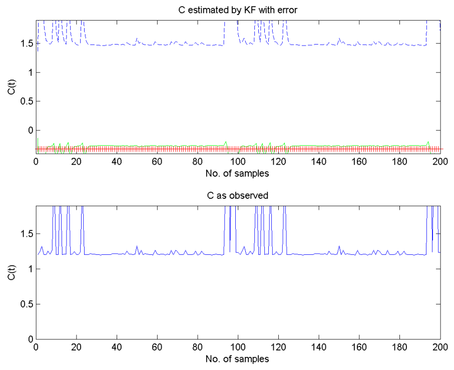


(a) LSMW estimation

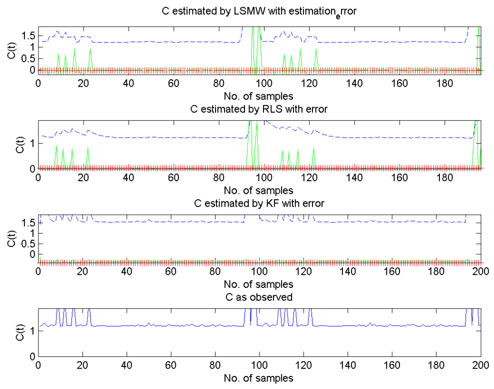


(b) RLS estimation

Figure 3.5



(a) Kalman Filtering estimation



(b) All methods

In the upper part of the plot in Figure 3.5a, the mean of the estimates are observed to be of the order of 10^{-1}. Secondly, RLS method is tested and the mean error is being further reduced to 10^{-2}. This estimation results are shown in Figure 3.5b.

The results of estimation done by the Kalman filtering is given by the Figure 3.6a and the mean of the estimation error is of the order of 10^{-3}. All the computations done are enough short in time to be obtained in real time. Conclusively, we can observe in Figure 3.6b, that in LSMW method the first lap of values are well estimated, but the estimates fall flat to the zone of less variability. Also, the sharp variations are not estimated well by LSMW method. RLS method is able to mitigate the shortcomings of LSMW method where the sharp rise zone in the middle is estimated well. Also, it is evident in Figure 3.6b, that Kalman Filtering provides the best estimates for estimating the time-varying cost parameter where both the sharp rise and less variable zones are estimated well.

Henceforth, we can infer that the Kalman Filtering method provides the most suitable estimation for the cost parameter. These predictions are obtained on-line during the run-time of the multi-robot prototype system [3].

3.7.4 Experiment-III

The shop floor is described through a topological map as given in Figure 3.6 for this experiment. The boxes forms the work cell and creates labyrinth paths. The surfaces marked by single dotted line was made heavily rough to induce variation in travel time. There are several arcs connecting different ports and nodes. Decisions for navigation and coordination are mostly based on heuristics cost. But, in this work travel time is designated as the cost of performing the task of navigating or traversing. Thus, we estimate different travel times of these arcs as realistic costs and compare with fixed heuristics cost like Euclidean distance, which is presented in Section 3.7.3.

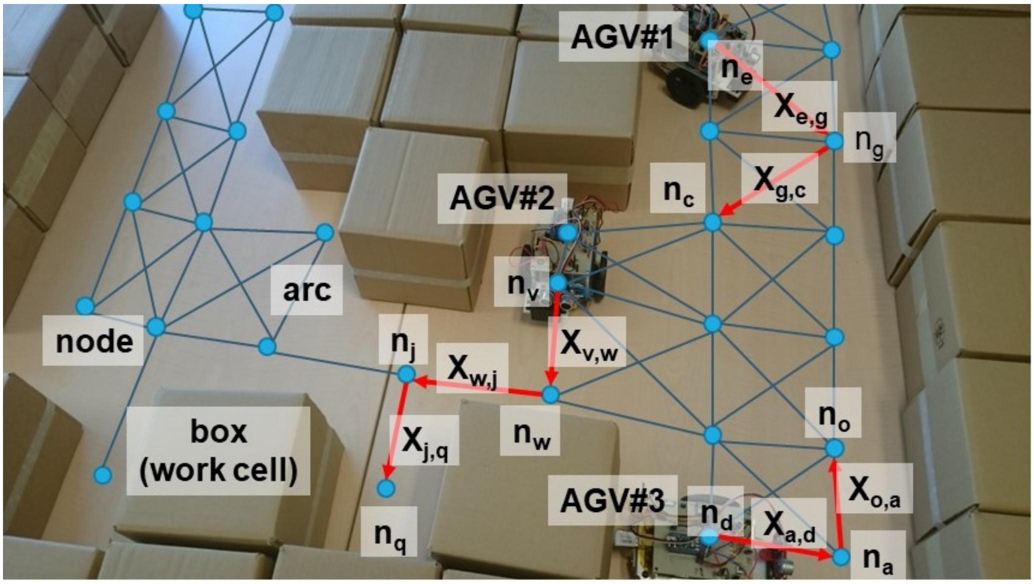


Figure 3.6: Floor described with a map

3.7.5 Results of experiment-III

In Table 3.1, the comparison between these heuristics cost and the real estimated cost are provided. The table shows heuristic costs and real costs of traversing two arcs whose travel costs are designated as $X_{g,c}$ and $X_{j,q}$. The first shows the arc lengths in cm and second column shows the corresponding heuristics cost in sec. The third column shows the real estimated travel time of the same arc in sec. The columns from fourth and on wards shows the real estimated travel time over different state of charge of batteries. The arc whose travel time is $X_{j,q}$ has a part in the rough zones.

[3] [25] depicts the same discussion as it is the same work

Table 3.1: Analysis of real costs

No	Euclidean Cost		100% Battery Level Realistic Static Cost	Costs per Battery Level				
				90%	80%	70%	60%	50%
	[cm]	[s]	[s]	[s]	[s]	[s]	[s]	[s]
$X_{j,q}$	24	3.3379	2.8927	3.1015	4.0270	4.1037	4.1158	4.2425
$X_{g,c}$	17	2.1717	1.9305	1.9608	1.9867	2.0229	2.195	2.4478

Thus their estimated travel time is slightly more than the heuristic euclidean travel times which subsequently increases with discharge of batteries. So, due to continuous performances by the robots in an MRS, performance costs in form of travel time for traversing tasks varies over the time and correct estimates of them needs to be utilized as it is evident from the comparison. Therefore, these real estimated costs can be utilized to make more cost efficient decisions. However, the time-varying nature of costs over different factors is generally not included in heuristics costs. In Section 2.4 we have presented two works which approach a nearly similar problem [54, 21], but their work cannot be directly compared to ours because the approaches are different (also described in Section 2.4). Moreover, to best of our knowledge, there is no other state of the art research proposal which addresses the same problem as we are solving. So we cannot present a comparison of our work with state of the art.

3.8 Summary

In this work, unlike the previous work on parameter estimations, we investigate about cost coefficients to reflect the states of battery and floor in automated logistics. In that process, travel time is identified as a type of cost parameter. Moreover, the necessity of estimating these travel times is shown which can help to generate better control decisions. Kalman filter have been found to be a suitable method to estimate the travel times. The results show that the heuristics travel time differ from the estimated ones. Based on the estimates of the cost parameters, more optimal and cost efficient task allocation decision can be taken at the system level controller.

Chapter 4

Optimal operation in automated logistics

Previous chapters were focused on accurate estimation of travel time to traverse different edges of transportation graphs in an MRS using several estimation methods. It was found that traversal times vary along time due to a variety of factors, including the state of charge of battery, condition of shop floor, dynamic obstacle etc. Moreover, the predicted values of travel time are obtained on-line during the run-time of the MRS, which further predicted the values for the state of costs of tasks in comparison to heuristics costs varies enough to impact high level control decisions like path planning, task allocation, scheduling and navigation for each of the mobile robots and for the system as a whole. From these results, we arrive at a juncture which necessitates estimating these travel time online and incorporating them in decisions for planning. In this chapter, we highlight on making decisions based on travel time, which reflects real-time state of individual parts and also environmental factors. With suitable predictions of these travel times the current state of cost involved in traversing from one node to another can be known. Thus suitable state-space model is formulated to estimate these travel time to use as weights for cost efficient route planning.

4.1 Travel time in decisions for planning

Control of MRS, used in internal logistics for factories, involves planning at many levels to achieve robust, fault tolerant, adaptable and cost efficient co-ordination. Usually, the information about current condition of robot, floor, batteries and other robots play crucial role in decision making [20, 36, 8]. This is explained in the following example. In Figure 4.1 we refer to the same scaled down automated internal

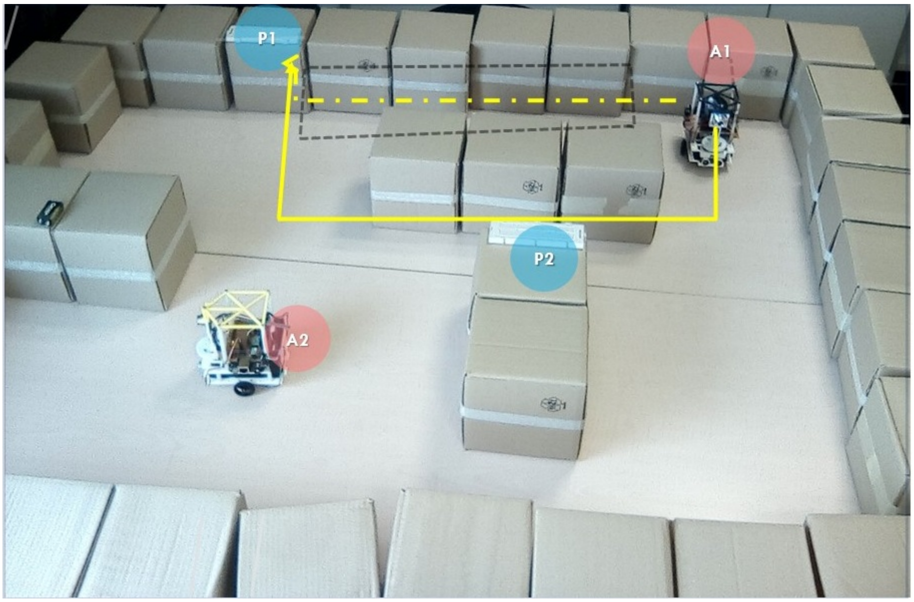


Figure 4.1: An scenario of planning decision

transportation system as explained in Section 3.4 in Chapter 3. Here, all MRs can execute only one task at a time. The dotted line illustrates the path computed for $A1$ to carry some material to $P1$ at t_i. Again, $A1$ will use the same path to carry same material to $P1$ at time t_j $(j > i)$. However, continuous functioning of $A1$ have exhausted the battery of $A1$ along with the deterioration of the condition of the given path (marked by dotted rectangle). Thus, equal amount of time and energy as previous would not be sufficient to reach $P1$ at t_j and more cost in terms of energy and time will be required. The real and current traversal cost to reach $P1$ needs to be estimated to produce the most cost effective decision. A new path, if decided, using this estimated traversal time will enhance optimum resource utilization. For example, the part of the floor in the zone marked by solid line has not deteriorated. Route planning done using estimated travel costs resulted in the path marked by solid line. This path lies in the zone marked by solid line (with no roughness) and so it is more cost effective.

In our current work, traversing a path from one spot to another is considered as a task. Travel time reflects the state of charge of batteries (internal factor of a robot) and environmental factors (Section 3.2 in Chapter 3). Travel time determines the cost of such traversal tasks. Path traversal is a sequence of traversal from one spot to another. So, total travel time of these segments determines the total travelling cost of a path. The dynamic route or task planning can be improved based on the travel time as cost expenditure due to different factors can be directly related to time(s) needed to perform task(s) by individual robots [60] in the system. This can influence not only the efficiency of each MR but also the whole system [36].

4.2 Planning algorithm and cost estimation

The eminent problem in autonomous robots, used for various purposes, is optimum planning. This has been solved in two distinctly different approaches for autonomous robots. General path planning approaches are deployed for implementing path planning in individually controlled robots like Randomized Potential Planner (RPP), Probabilistic Roadmap Method (PRM), Rapidly-exploring Random Trees (RRT). Sampling methods perform reasonably well in solving intricate path planning problems in static and dynamic environment for single robot [31]. In most of these approaches, the vehicular dynamics are considered as state of the robot and the minimum cost path is obtained by spanning the search tree based on the distance between the current state and goal state. Although Suh and Oh in [67] and Achtelik *et. al* in [1] have used Gaussian process as the cost of the path to incorporate environmental parameter, the search mandates to conceive the vehicular dynamics of the robot. Thus, these methods rely on precise information gathered from sensors. Also, these methods are blocked into local minima and uncertainties in the environment hinders successful results using sampling based methods [31]. Further, heuristic approaches like Artificial Neural Network (ANN) [27], Genetic algorithm (GA) [3], Particle swarm optimisation (PSO) [77], Ant colony optimisation (ACO) [17], *et cetera* can adapt to uncertainties and changing environment. But, they are computationally expensive which is a major concern for robotic control units equipped with limited resources [50]. Usually, in state of the art works, either for a single robot or a MRS, the cost functions or reward functions are not from real observation data, to be deployed in various purposes of coordination and planning, both in centralised and distributed control architecture. Our approach takes the benefit of single robot path planning and can be enhanced to multi-robot planning. In our approach, we incorporate the vehicular mechanical factors and environmental factors in travel times to mitigate the identification of the vehicular dynamics. We compute the path in system-level of robotic control and paths are broken down to simple vehicular commands for movements and communicated to the lower-levels of control. Also, we deploy simple, deterministic and computationally inexpensive Dijsktra's algorithm and incorporate travel times with it to decide paths of minimum cost.

On the other hand, motion planning and task planning in MRS are often treated as specific coordination problems. Cost coefficients are usually computed heuristically before hand or being modelled

Like path planning approaches for a single robot, these cost functions are utilized to estimate states of the individual MRs on-line. However, motion planning and task planning problems, as a specific case of coordination problem are typically NP-hard and are addressed to find tractable and good solutions [28] and are mostly treated as individual specific problems in field of MRS [10, 64, 32]. Also, cost functions are heuristically computed for solving both task planning and route planning problems in MRS at the same time in [43].

In this work, we use travel times as cost parameters and utilize them to obtain optimal path in a single robot.

4.3 Using travel time to generate cost efficient planning decisions

This chapter elaborates on investigation conducted on estimation of these travel times on real-time in MRs working for internal transportation. The path planning has been considered as an example of decision making. Moreover, we also design experiments on prototype platform developed mimicking a real automated plant. In this work, traffic network topology maps are used and spots are marked as nodes in the map. Traversing between two nodes is considered as a task by any MR and travel time between any two nodes reflect its cost of traversing. Topological map are used instead of geometrical maps as simple route planning approaches like Dijkstra's algorithm are easy to implement on topological maps (illustrated in Section 4.4.1). The travel time is calculated considering the difference between the departure from one node and reaching the next node and thus travel time is not dependent on the shape of the edge, rather it depends on the time taken to traverse the edge.

The impact of travel time in route planning is shown by calculating total travel cost between two different paths obtained by heuristically gathered cost and real travel time using the same route planning method. In Chapter 3, Kalman filter has been found to be a suitable method to estimate travel times of edges online. The estimated values of travel time of relevant edges is used as weight of edged in simple path planning algorithm to compute the optimal path. The problem proposed here is the one caused by differences of total travelling cost of paths, obtained by real travel time and heuristic weights as costs of edges, while planning routes for MRs, irrespective of planning method.

This problem have been highlighted and explained in Section 4.3.1, where P is defined as an optimal path and P_c as the traveling cost of P. As a path P consists of different edges, total traversing cost of path P_c can be found by summation of travel times of all the edges in P. P_c consists of real and estimated travel times as cost of edges. Dijkstra's algorithm is deployed to compute P. While computing P, Dijkstra's algorithm is complimented with these travel time of edges as cost of traversing that edge. As these travel times reflect the real cost incurred, Dijkstra's algorithm will generate more cost efficient paths using real estimated travel times, similar to Figure 4.1 in Section 4.1.

We have estimated these travelling times of edges statically (variation of travelling time over the time is not considered). Then Dijkstra's algorithm is deployed to compute optimal path using both heuristics cost on distance and edge travelling times separately. Paths obtained by heuristics on distances are compared with paths obtained by directly using edge travelling times. Total path costs using edge travelling times are improved by roughly 5% on average.

On the other hand, in case of battery-operated MRs, observation of the travelling time of any particular edge till full discharged of batteries depict a close correlation between state of charge and speed or time to travel or perform an action.Consequently, travel times of edges vary along time and require to be predicted accordingly during path planning. A good estimation method to accurately predict travel times requires histories of edge costs, which can be collected by random walking the traffic network until it is applicable or progressively built during MR operation. In the latter case, path planning during the learning phase cannot use best estimation methods. When, edge travel costs are estimated taking into account this variation along time and path is computed deploying these estimated costs, total path costs are closer to reality and might have significant differences with paths obtained by other cost estimation methods like theoretical, heuristic and experimental. In fact, computing the costs of paths by using Kalman filtering shows that the former cost estimation methods (using heuristic edge cost) can underestimate total costs and, thus, generate non-minimal paths.

4.3.1 Problem formulation

A path P is formed as a series of arcs between connecting nodes for an MR. The $a_{p,q}$ is defined as a connecting arc between any two nodes, namely n_p and n_q in Section 3.5 in Chapter 3 A path P for a robot is usually defined as,

$$P = \langle (n_a, n_b), (n_b, n_c), (n_c, n_d), (n_d, n_e), ...\rangle \tag{4.1}$$

where n_p is any node. P can be also expressed in terms of connecting edges as

$$P = \langle a_{a,b}, a_{b,c}, a_{c,d}, a_{d,e}, ...\rangle \tag{4.2}$$

where $a_{p,q}$ is any edge.

$X_{p,q}(e, f)$ denotes travel cost from n_p to n_q as previous and e denotes dependency for state of charge of batteries and f denotes dependency for frictional force of the floor. Now, the cost of traversing P can be written in a form P_c of

$$P_c =< X_{a,b}(e, f), X_{b,e}(e, f), X_{e,g}(e, f), ..., ... > \tag{4.3}$$

From now on, $X_{p,q}(e, f)$ will be written as X for simplicity. Path needs to be computed again after reaching a destination and P_c for every ith call of path planning is

In equation 4.3, it is shown that a total path cost is dependent on all edge costs and each edge cost is denoted by its travel time. Thus, the X denotes general travel time of any edge. Also, travel time of any edge X depends on all the previous edges the robot has already traversed. The reason being the discharge of batteries and (or not) possible change of environment. Thus, travel time X becomes a function of k as X_k where, k = number of time a MR has performed the task of traversing any edge.

Hence, X_k is estimated with respect to increase in k for any edge and used as weight of edge to compute path. The estimated value of X_{k+1} depends only on X_k and the observation of X at $(k+1)$. These experiments and results are explained in

Section 3.7.1. Observations of all possible X_k for all possible k needs to be made for this above estimation for a single MR.

However, this is not only cumbersome but also impractical to gather such huge amount of observation. This estimation is static as X is estimated without considering its variation with the total elapse of time from start of system. The static estimation approach is progressed to a different model. Observations of all possible X_k for all possible k is not needed in the latter. A window of previous values of X is decided to form a state vector. This state vector is estimated on every k to find the estimated value of X_{k+1}. Thus, the current value of X is estimated depending on the previous X's i.e.-travel times of edges which are already being found to form the path, along with the variation of exploration of X due to elapse of time. Thus, X values are dynamically estimated considering its variation over elapse of time. Moreover, the model is allowed to gather the possible values of X itself from the beginning of first call of path planning and use these values to estimate current value. This experiment is elaborated in Section 3.7.2.

4.3.2 Contribution

The contribution in this case is twofold. Firstly, we identify edge traveling time as a suitable cost coefficients considering an analogy to real automated and fully functional plant. Secondly, we estimate these identified travel times both statically and dynamically. Further to this, we utilize them in a planning control decision making process to culminate into better optimal results for each MR in order to obtain more cost efficient performances of MRs.

Organization of chapter

The rest of the chapter elaborates on prototype platform to conduct experiments (Section 4.4), gathering real data for static estimation (Section 4.5), static estimation (Section 4.6.1) and dynamic estimation (Section 4.7.1) of travel time and their results (Section 4.6.2 and Section 4.7.2). Section 4.8 draws the conclusion.

4.4 Platform for experiment

The internal transportation system is modeled gathering influences from standard automated logistics and warehouses and adhering to minute details regarding constituting parts like MRs, tasks, controller architecture, the environment. The environment has been developed using uniform sized boxes, while small hobby robots are constructed mimicking the real AGVs used in plants. Figure 4.2 depicts the different parts of a hobby robot, which is constructed in the laboratory for experiments. It has all essential constituents of an AGV like camera, controller board, ultasound sensor, wheels. It is operated by low self-discharge nickel–metal hydride batteries [59], which is similar to the batteries generally used in factories and DC servo motors drive the wheels. The controller architecture of the MRS is decentralized [**chaile2017running**, 47, 4]. Hence each MR has its individual controller with opportunities to communicate information to other MRs in the system. All the robots are single-task and are

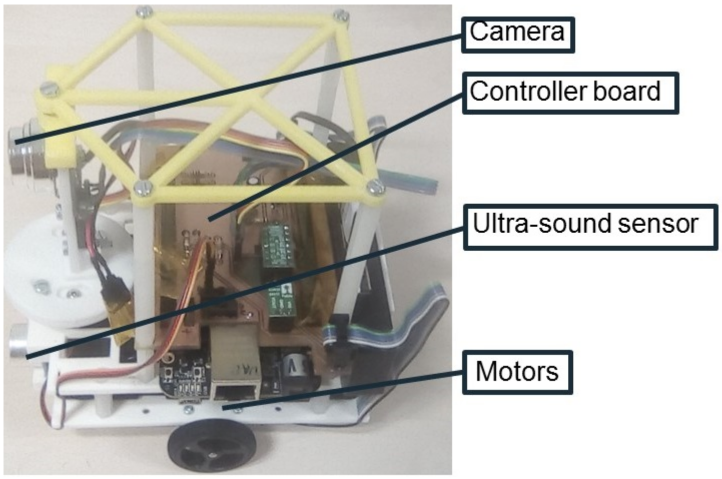


Figure 4.2: Structure of AGV used

equally equipped to perform all the tasks. Also, all the tasks requires only one MR to be completed.

4.4.1 Layout of the factory floor

Topological maps are used to describe the floor plan so that simple approaches like Dijkstra's algorithm are easy to implement. The layout of the factory floor is made in three representative real life industrial scenarios. All the three scenarios are derived from the factory floor layout of the automated factory of Coca-Cola Iberian Partners in Bilbao, Spain as described by the works of Beinschob and Reinke in [9]. The work of [9] is a part of the *PAN ROBOTS* project [**panRobots**] and Coca-Cola Iberian Partners in Bilbao, Spain are their project partners. The production and warehouse facility of of the Coca-Cola Iberian Partners in Bilbao, Spain is fully operational with a fleet of 12 AGVs and the total area covered is 130mX100m which is divided into three different sections. The work of Beinschob and Reinke in [9] deals with Simple Localisation and Mapping (SLAM) on the measured data at the site of Coca-Cola Iberian Partners in Bilbao, Spain. In [9], the SLAM results are presented which immaculately describes the real layout of the production and warehouse facility in a grid map. We use this grid map as a reference layout to generate the floor layout of our scaled down prototype. The three representative layouts are developed from three different sections of the production and warehouse facility of Coca-Cola Iberian Partners in Bilbao, Spain. We use topological map in our prototype, while the warehouse facility is described in a grid map in the works of Beinschob and Reinke in [9]. Conversion from grid map to topological map (CGTM), termed as graph clustering, itself being a deep rooted and eminent research problem, we introduce a naive method to convert the grid map of the warehouse facility given in [9] to a topological map. In general, CGTM is done with a goal to minimise the total number of nodes in the resulting topological map, which makes the method very complex. In our application, the goal is not to minimise the total number of nodes in the resulting topological map as the total number of nodes are not exuberantly high. In our approach, a simple assumption is considered for converting the grid map to a topological map where each free cell in the grid map corresponds to a node in the topological map. The basis for this assumption lies in the fact that total number of nodes generated as a result of this assumption in the topological map will be approximately 6000 and this whole topological map will be again used to extract three sections from three different zones and each section will result in one floor layout, which results in quite less number of nodes in each layout map.

The grid map of the production and warehouse facility of Coca-Cola Iberian Partners in Bilbao, Spain provided in [9] has been converted into a topological map and then three representative maps are extracted from that topological map from three different zones of the whole map area. In the resulting three maps the average of the total number of nodes is 900. The resulting three topological maps are provided in Figure 4.3.

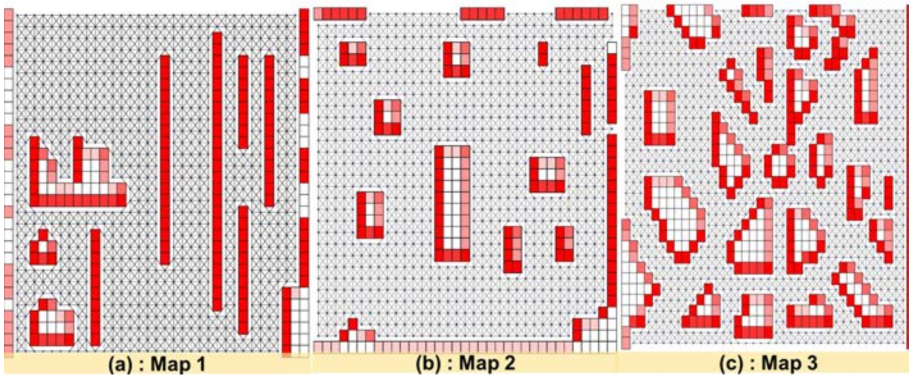


Figure 4.3: Three topology maps

Map 1 in Figure 4.3 is a representative of winding racks in the warehouse facility, while Map 2 in Figure 4.3 represents randomly placed racks and Map 3 in Figure 4.3 represents racks organized in a hub.

4.5 Real data everywhere

Now, we focus on gathering the observed data for all the realistic costs of the edges in the three representative floor layouts (Figure 4.3). To achieve this, we made one MR traverse each edge in each layout repeatedly till the battery exhausts completely. Figure 4.4 demonstrates three different sections in Map 1. Travel time is recorded to traverse edges from maximum battery capacity till complete discharge with dynamic obstacles passing across the edge, during traversal of the MR. For example, travel time of traversing $a_{h,r}$ in Figure 4.5 is recorded with normal condition of floor, being intervened by human being sometimes passing across it. The total cost of path is given by equation 4.4 (Section 4.3.1)

$$P_c =< X_{a,b}(e, f), X_{b,e}(e, f), X_{e,g}(e, f), ..., ... > \tag{4.4}$$

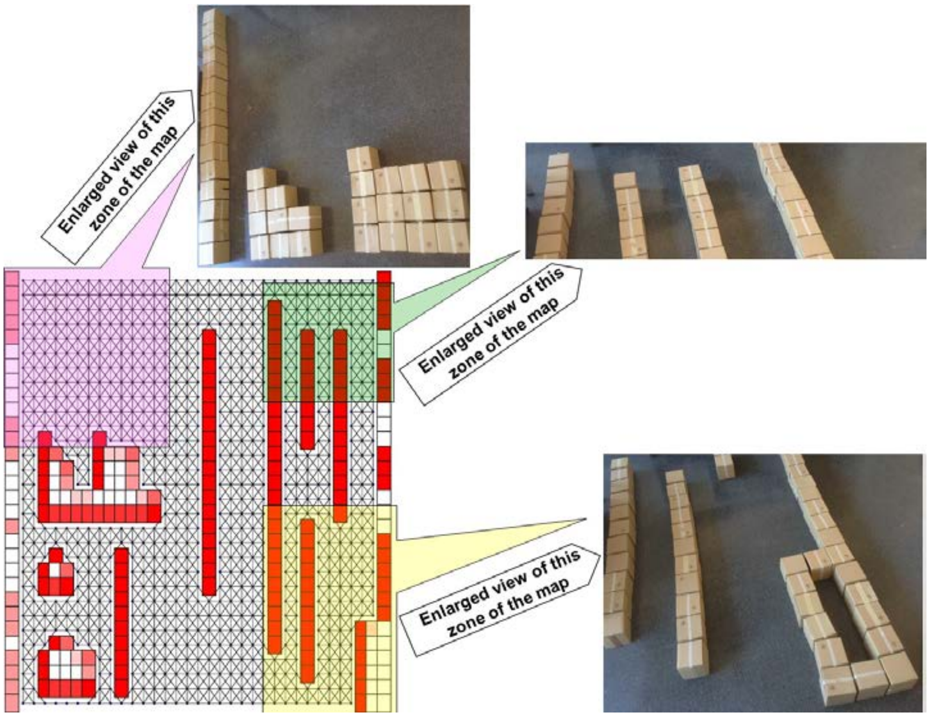

Enlarged view of this zone of the map
Enlarged view of this zone of the map
Enlarged view of this zone of the map

Figure 4.4: Different sections of one topo map

Now, total path cost in ith and jth call of path planning are

$$P_c^i =< ....X_{h,r}^i, ..., X_{e,g}^i, ..., X_{q,r}^i, ... > \tag{4.5}$$

and

$$P_c^j =< X_{d,e}^j, X_{e,g}^i, ..., X_{h,r}^j, ..., X_{w,x}^j > \tag{4.6}$$

where, $X_{c,d}^i$ and $X_{h,r}^j$ are the travel cost of arc between nodes n_h and n_r at i and j calls of path planning respectively. Thus, observation data created through the above process for $a_{h,r}$ produces the observation value of not only $X_{h,r}^i$, but also $X_{h,r}^j$ and travel cost of arc $a_{h,r}$ for more similar path planning calls at different instances till complete exhaustion of batteries.

Also, we made the surface on different sections of the floor differently rough and smooth. For example, the zone marked by solid line in Figure 4.6 is made rough, the zone marked by single dashed line is made moderately rough and the zone marked by double dashed line is made highly smooth. But the surface condition of the floor is changed during the traversal of edge by the MRs. Also, while the edge traversal of MR, we put unknown dynamic obstacle like human walking across the edge to include variation. For example, for edge $a_{c,d}$ in Figure4.7, the surface condition was smooth at the beginning of traversal by an AGV. The AGV was made to start traversing $a_{c,d}$ from the starting node n_c to its ending node n_d repeatedly with maximum capacity of the batteries. After certain interval of time, the surface of the floor is changed to rough for $a_{c,d}$. But the MR is made to continue traversing $a_{c,d}$. Again, after certain interval

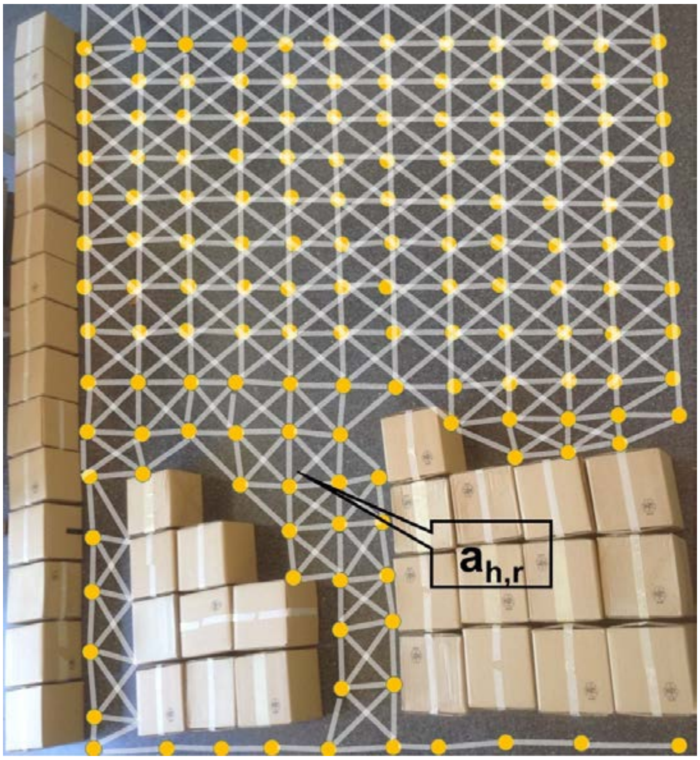


Figure 4.5: An example of an arc

of time the roughness is removed and smoothness of the floor is restored. Still, the MR is made to continue traversing $a_{c,d}$ till the battery exhausts. This observation data consists of travel time of an edge $a_{c,d}$ from maximum battery capacity till draining out of batteries on different floor condition for one MR. We made observation on all the edges similarly like above, in order to complete the whole set of observation data, for all the edge costs on each floor layout for a AGV in the MRS in different floor conditions and state of charge of batteries.

4.6 Using static estimates of travel time in path planning

We generated measurement data of travel time $X_{p,q}$ for all edges in three maps (Figure 4.3) for one MR. It is concluded in Chapter 3 that travel times when estimated on-line produce the real cost of traversing the edge. These estimated travel times for each edge are used during decision making process for computing an optimal route. This can produce a better optimal route and save on cost expenditure.

We have considered **reaching a destination** as a task to produce the efficacy of static and dynamic estimates of travel times in decisions for planning. Dijkstra's algorithm [22] is used for finding optimal path to reach a destination, as it gives

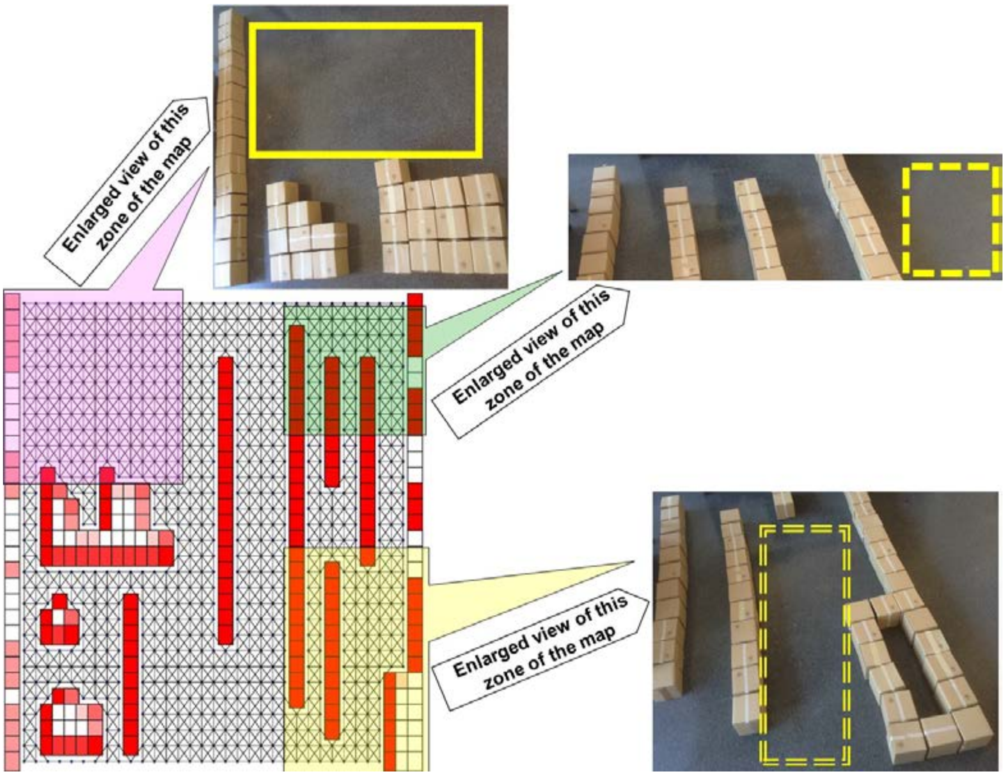

Enlarged view of this zone of the map
Enlarged view of this zone of the map
Enlarged view of this zone of the map

Figure 4.6: Different floor condition in three different sections of one topo map

deterministic solutions.

Usually, in Dijkstra's algorithm, the heuristic cost based on euclidean distance of edge is considered as the wights of each edge to calculate the optimal path. In our current attempt, we will use estimated travel time of an edge in the map as the wight for that edge. Hence, we are using these real estimated cost values to find the optimal path using Dijkstra's Algorithm, instead of taking heuristic cost as the edge weight. Now, the estimation of travel time is made using Kalman filtering method [68], being found as the most suitable method in Chapter 3.

4.6.1 Experiment I

In Dijkstra's algorithm, at every new node exploration, weights of all the edges arising out of that node needs to be known and to be used in computing optimal path. We propose to use X_k as cost of arcs, instead of using heuristics cost based on distance, Thus, X_k needs to be estimated during optimal path computation at each new exploration for every relevant edge. For example, consider Figure 4.8 where a sample route computation is illustrated. Let n_a be source and the MR needs to reach $P16$. So, path computation starts at n_a with its neighbors n_b, n_c and n_d. The weights of connecting edges arising out of n_a (current explorable node) needs to be computed to progress exploration. So, X_k needs to be estimated for $a_{a,b}$, $a_{a,c}$ and $a_{a,d}$. The

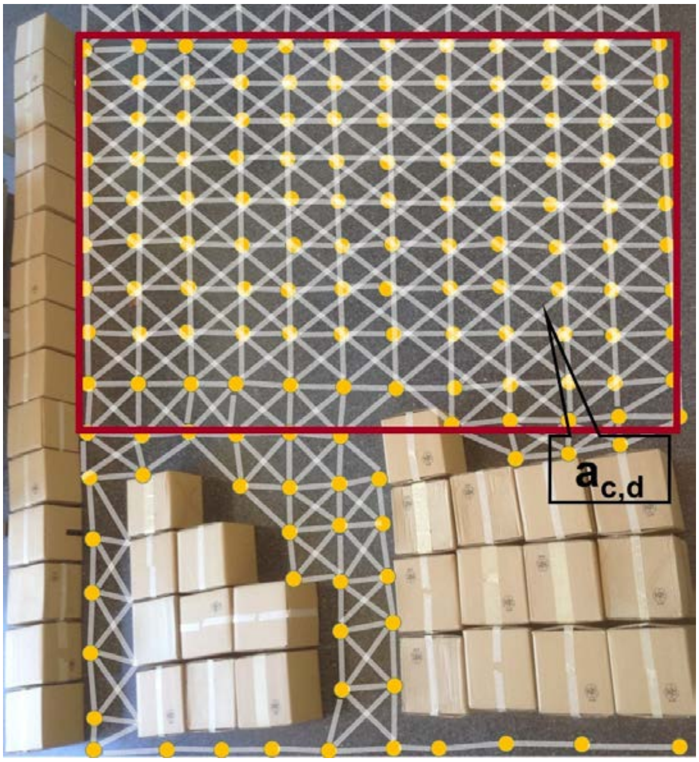


Figure 4.7: Another example of an arc

variable k provides the instance at which the travel cost of the edge is to be obtained. Essentially, value of k is one more than number of predecessors of the current explorable node, the reason being, the number of predecessor provides the number of times the robot has traversed edges. The battery of the robot gets discharged while traversing the edges of the path, so the real edge costs need to be considered while computing optimal path. To account for this factor, X_k has been considered to be changing after finding each new explorable node. Thus a single variable, denoted as X_k, is estimated over and over again whenever edge costs are required to be known to find the optimal route.

Now, k is 1, which is one more than the number of predecessor of n_a (as n_a is source, it has no predecessor). Let, n_c be the next node which forms the successor of n_a in the formation of optimal path. Again, the connecting edges to the neighbors of next explorable node, n_c, are $a_{c,e}$, $a_{c,f}$ and $a_{c,g}$. Now, variable X_k needs to be estimated for all these edges arising out of n_c at current value of k, being 2 (as n_c has 1 predecessor). For this purpose, development of a model is essential. The state-space model provided in equations 4.7 and 4.8 is proposed to estimate edge travel cost in the above manner.

$$X_k = X_{k-1} + \omega_k \tag{4.7}$$

$$Y_k = X_k + \eta_k \tag{4.8}$$

The state vector X_k in equations 4.7 consists of a single variable and thus it is a scalar

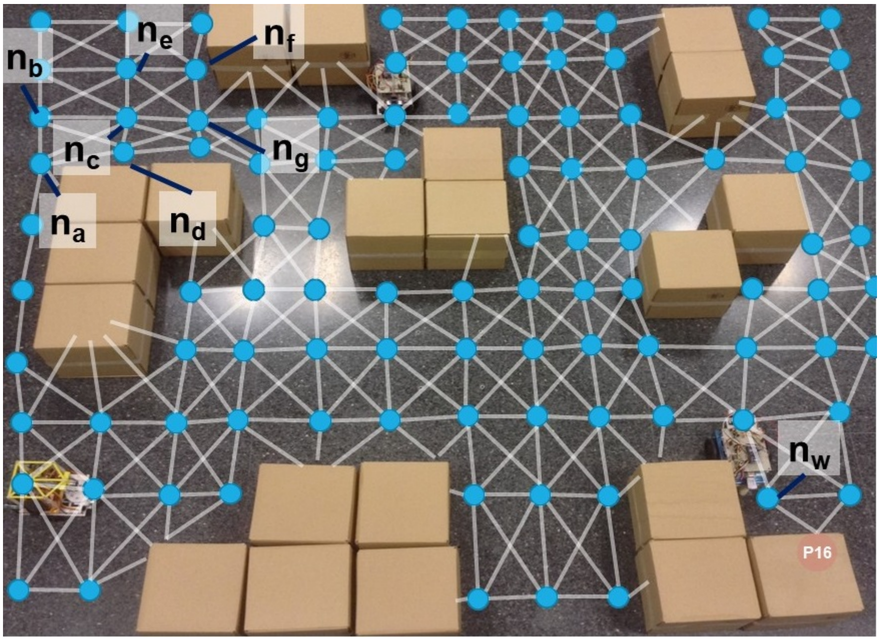


Figure 4.8: Sample run of route computation

quantity, depending on k, k being the number of edges already found in the path or number of nodes found in the path. Hence, X is estimated over and over again for different connecting edges of every new exploring node. Equations 4.7 also involves the error term ω_k, which is independent and normally distributed. Y_k in equation 4.8 is the observation variable for X_k. It has linear relation with the state vector through the observation equation 4.8 of the system. η_k is another term that could be described as measurement error, which is also normally distributed.

According to equations 4.7 and 4.8, the variable X_k depends only on the travel time of the previous edge in the path at the previous k value. For estimating X at every k instance, the value of previous X is given by travel cost of the connecting edge between the current node and its predecessor node. As in the above example, while exploring the node n_c, value of X at (k-1) is given by cost of connecting edge from n_a (predecessor of n_c) to n_c. Then the travel cost of this previous connecting edge can be used as the travel cost at previous iteration to estimate the travel cost at current iteration. So, essentially the travel time is a series over variable k.

Thus Kalman filtering is applied on the above state-space model, given by equations 4.7 and 4.8 is used to compute travel time of edges. We obtain equations 4.9

and 4.10 after applying Kalman filtering method on equations 4.7 and 4.8.

$$\begin{aligned}
\hat{X}_k^- &= \hat{X}_k - 1 \\
P_k^- &= P_{k-1} + \sigma_\omega^2 \qquad (4.9) \\
K_k &= P_k^- / [P_k^- + \sigma_\eta^2] \\
P_k &= P_k^- - [{P_k^-}^2 / [P_k^- + \sigma_\eta^2]] \\
\hat{X}_k &= \hat{X}_k^- + K_k * \omega_k \qquad (4.10) \\
\text{where,} \quad \omega_k &= [Y_k - \hat{X}_k^-]
\end{aligned}$$

$\hat{X^-}_k$ produces the apriori value of X and P^- produces the associated covariance, σ_ω^2 being the covariance of process noise ω_k. $\hat{X}_k$ provides the predicted estimate of X_k, as $\hat{X^-}_k$ is corrected in equation 4.10 with the help of Kalman Gain K_k the observation Y_k. P_k^- provides the associated co-variance matrix, σ_η^2 being the co-variance of the observation noise η_k. Thus, Y_k is required to obtain $\hat{X}_k$ for all k, which is gathered by the process explained in Section 4.5.

At the beginning, there is no previous edges traversed. So, value of k will be 1 at the start.

$$\hat{X}_0 = E[X_0] \qquad (4.11)$$

$$P_0 = E[(X_0 - E[X_0)(X_0 - E[X_0)^T] \qquad (4.12)$$

We use equation 4.9 to obtain $\hat{X^-}_1$ for $X_{a,b}$, $X_{a,c}$, $X_{a,d}$ separately depending on X_0 using equation 4.11. Similarly, we get separate P_1^- using equation 4.12. Next, we obtain $\hat{X}_1$ (estimate) and P_1 for $X_{a,b}$, $X_{a,c}$, $X_{a,d}$ using equation 4.10. Comparison of estimated values of $X_{a,b}$, $X_{a,c}$, $X_{a,d}$ will provide the least cost edge from n_a. Let, the least cost edge be $a_{a,c}$. So n_a will become the predecessor of n_c, i.e.-to reach n_c, the edge should come from n_a. When n_c will be explored, the value for k is 2 as n_c has 1 predecessor. The next least cost edge from n_c in the path is required to be known. Thus, $X_{c,e}$, $X_{c,f}$, $X_{c,g}$ needs to be estimated. Thus, observation Y_k of X at current k is required to estimate X. Thus observation values for travel costs of all possible Xs for all possible ks were collected. Though, X_k varies only on X_{k-1} in this model, in reality, it depends on Xs for all the previous edges in the path and its own variation over the time. Thus, this process of estimation is static. Moreover, as explained in Section 4.3.1, observation of X is required to obtain estimate of X through this static estimation procedure for any k which is not only cumbersome, but also unrealistic. However, this static experiment is conducted to verify that weights of edges can be estimated online during exploration of Dijkstra's algorithm using a state-space model. Also, it is verified that the estimated values of X are correct and real through this experiment, as the values can be compared to real observations. In subsequent sections, we ill show dynamic estimation of X which does not require the offline observation procedure of such mammoth intensity.

The process of estimating static travel times and using them as edge costs to compute optimal path by Dijkstra's algorithm in Algorithm 1

Algorithm 1: Dijkstra's algorithm using static estimation of travel time

```
Initialise_Single_Source ($V, E, s$)
Input  : $V$-list of nodes, $E$-list of edges, $s$-source node
Output: $d[v]$-attribute for each each node, $\pi[v]$-predecessor of each node
for each $x_i \in V$ do
    $\pi[x_i]$ = infinity
    $d[x_i]$ = NIL
end
$d[s] = 0$
Main ($V, E, w, s$)
Input  : $V$-list of nodes, $E$-list of edges, $w$-edge weight matrix, $s$-source node
Output: $\pi[v]$-predecessor of each node
$P :=$ NIL
$Q := V$
$j := 0$
while $Q! = 0$ do
    $j = j+1$ $u :=$ Extract min ($Q$)
    $P := P \bigcup u$
    for each $v \in Adj[u]$ do
        $w = findedgedCost(u, v, j)$
        $relax(u, v, w)$
    end
end
findedgedCost ($u, v, j$)
Input  : $u$-current node, $v$- neighbor node
Output: $w$- estimated travel_time (cost) from $u$ to $v$
$findPredEdge(u)$
$prev_x := x_(prevEdge)$
$w = estimateKF(prev_x, j, Y_j)$
findPredEdge ($u$)
Input  : $u$-current node
Output: $prevEdge$-edge connection $u$ and $predU$
$prevEdge$ = edge between $u$ and $predU$
estimateKF ($prev_x, j, Y$)
Input  : $prev_x$-$x_{j-1}$, $j$-instance for estimation, $Y$- observation variable
Output: $x_j$-travel cost at current j for current edge
Apply KF on state-space model to obtain $x_j$ Relax ($u, v, w$)
Input  : $u$-current node, $v$- neighbor node, $w$- estimated travel_time (cost)
         from $u$ to $v$
Output: $d[v]$-attribute for each each node, $\pi[v]$-predecessor of each node
if $d[v] > d[u] + w(u, v)$ then
    $d[v] = d[u] + w(u, v)$
    $\pi[v] = u$
end
```

4.6.2 Results I

Optimal paths are obtained deploying the original Dijkstra's algorithm in all the three representative floor plans depicted in Figure 4.3, using heuristics on distances for 20, 40, 60 and 80 repetition. The number of repetition is an indicator of passage of time. On the other hand, instead of using heuristic edge weights as costs, Dijkstra's algorithm was modified to find optimal path using the estimated static edge costs according to Algorithm 1 in all three floor plans. As previous, paths were computed for 20, 40, 60 and 80 repetitions. Total path costs of optimal paths produced by Dijkstra's algorithm in both cases are compared. This is summarized in Figure 4.9. The vertical bars of **Eucl** and **SEC** in Figure 4.9 represent the average total path costs for heuristic cost based routes and static estimates based routes respectively. In Figure 4.9, average total path costs of paths obtained using heuristic weights of edges, which are represented by **Eucl** never change with increase in number of repetitive calls, the reason being heuristic edge weights are static over the passage of time. Nevertheless, the other vertical bar **SEC** in Figure 4.9 which represent the average total path costs for paths produced by Dijkstra's algorithm using static estimates of travel time as weights of edges change with increase in number of repetitive calls of path planning in each map. Average total path costs increases with number of repetitions as shown by vertical bar **SEC**, as duration of performance increases with increase of number of repetition. This happens due to the dependency of current edge cost X_k on previous edge cost X_{k-1} (equations 4.7 and 4.8). Also, vertical bar **SEC** shows that average total path costs obtained by Algorithm 1 is 5% less in case of Map 2 and Map 3 and 2% less in Map 1 than that of heuristic cost based Dijkstra's algorithm. Though, average total path costs using Algorithm 1 vary over the time, this variation does not truly reflect the variation of travel time due to time-varying factors, as it is not incorporated in Algorithm 1. This variation is observed due to the dependency of current edge cost on previous edge cost derived from the model (equations 4.7 and 4.8).However, the increase in average total path costs with number of repetitions does not truly reflect the variation of travel time due to time-varying factors as it ignores the inherent increases of travel time with time.

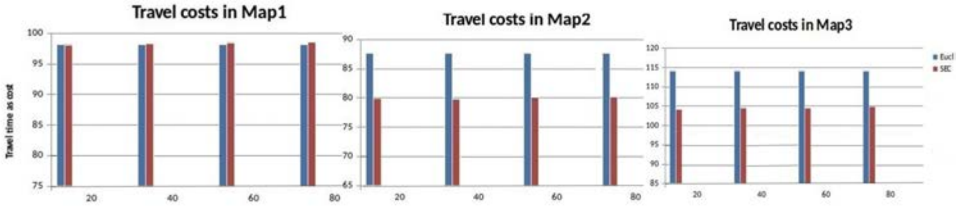


Figure 4.9: Results of static estimation in three maps

4.7 Obtaining optimal path using estimates of dynamic cost

4.7.1 Experiment II

In the experiment explained in Section 4.6.1, travel times of edges is assumed to be dependent only on the previous travel time. But due to discharge of batteries, change of floor conditions, change of condition of mechanical parts (explained in Section 4.1) the travel times of edges vary over all the previous edge travel costs in reality. Hence, for estimation of these varying travel times the model proposed in equations 4.7 and 4.8 is not sufficient, as it does not incorporate the dynamic nature of the travel costs. In order to consider the change of travel costs depending upon all the previous travel costs, a different model is devised. The new factor ξ_k is introduced which accounts for the progressive change in travelling time over the time due to batteries and other factors. Still, X is a time-series on number of explorations, k. The model to estimate dynamic travel costs is devised based on the bi-linear models [62]. The bi-linear model, provided in equation 4.13, is used to model the change of travel costs depending upon all the previous travel costs.

$$\begin{aligned} &X_k + a_1 X_{k-1} + ... + a_j X_{k-j} \\ &= \xi_k + b_1 \xi_{k-1} + ... + b_l \xi_{k-l} \\ &+ \sum \sum c_{rz} \xi_{k-r} X_{k-z} \end{aligned} \tag{4.13}$$

The model described in equation 4.13 is a special case of the general class of non-linear models called state dependent model (SDM) [62]. In equation 4.13, X_k denotes the edge travel cost at k and ξ at k denotes the inherent variation of the edge travel cost. In equation 4.13, X_k depends on all the previous values of X and ξ at previous instances of k, whose number is provided by the variables j and l. However, a fixed number of previous values of X and ξ is used for estimation of current X like an window which we progress over the time. From now on wards, this fixed size of this window will be termed as *regression number* and it is chosen as a design parameter, designated by j and l. The *regression_no* is increased from 2 to 9 and the effects on total edge travel cost of paths is demonstrated in Section 4.7.2. The double summation factor over X and ξ in equation 4.13 provides the nonlinear variation of X due to state of batteries and changes in environment.

The state space form of the bi-linear model is given in equation 4.14.

$$s(k) = F(s(k-1))s(k-1) + V\xi_k + G\omega_{k-1} \tag{4.14}$$

$$Y(k) = Hs(k-1) + \xi_k + \eta_k \tag{4.15}$$

In equation 4.14, the state vector s_j is of the form $(1, \xi_{k-l+1}, ..., \xi_k, X_{k-j+1}, ..., X_k)^T$. The state vector contains the progressive edge costs over time from $X_(k-j+1)$ to X_k. The variable ξ provides values of innovation of edge costs over the time as the exploration proceeds. Here, j denote number of previous edge costs to be included in the state vector among all included edges in the path till kth instance. Also, l denotes the number of previous innovations of these edges. In the example provided in

Figure 4.8 in Section 4.6.1, the computation to find optimal path was done using the method described in Algorithm 1. In this section also, we will use the same example to explain the path computation using the new state-space model of equation 4.14 and 4.15. Thus the same example is provided in Figure 4.10.

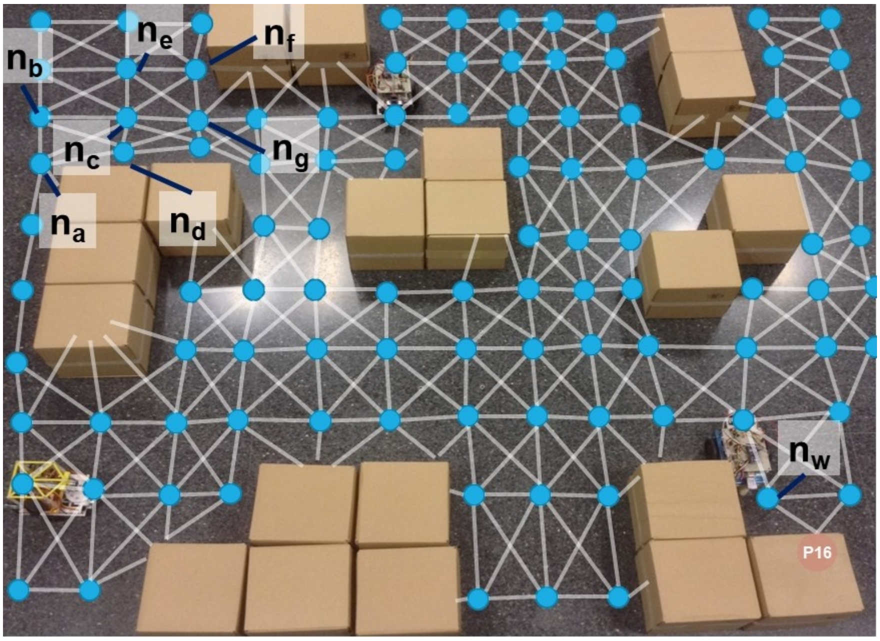


Figure 4.10: Sample run of route computation

The source is considered as n_a. Using Dijkstra's algorithm exploration starts at n_a when k is 1. Let us assume values of j and l are equal which is 2. State vector cannot be formed at k=1 as minimum 2 previous travel costs are needed and thus, exploration proceeds with average travel cost for the edges arising out of n_a. When exploration reaches n_c which has 1 predecessor node, value of k becomes 2. Thus, at this instance, the state vector can be formed with the values of travel cost from n_a (predecessor of n_c) to n_c, denoted by X_1 and travel cost from predecessor of n_a to n_a, denoted by X_0. In this case, n_a has no predecessor as it is the source. Thus X_0 will be 0. As exploration grows, k increases and when n_g needs to be explored, k becomes 3. Hence, X_2 will be travel cost from n_c (predecessor of n_g) to n_g and X_1 will be travel cost from n_a (predecessor of n_c) to n_c. Hence, X_1 and X_2 will be constituting state vector $s(k\text{-}1)$ at k = 3 and $s(k)$ needs to be computed for all the edges arising out of n_g

On the other hand, variable ξ designates the innovation in the travel times of edges over the time as the robot moves from one node to another. As the robot traverses one edge after another to reach the destination, the travel time of the edges innovate due to battery discharge and other factors. Hence, correct estimate of edge costs considering this innovation on the travel time is necessary to culminate into optimal routes. Considering the above example, for k=2, state vector $s(1)$ contains ξ_0 and

ξ_1. Similarly, for k=3, $s(2)$ contains ξ_1 and ξ_2. Moreover, at k =2, $s(1)$ takes the form $(1, \xi_0, \xi_1, X_0, X_1)^T$ in order to estimate $s(2)$ which will produce X_2. Similarly, at k =3 , $s(2)$ is required to be formed as $(1, \xi_1, \xi_2, X_1, X_2)^T$ and $s(3)$ needs to be estimated, which will produce X_3. The values of j and l are thus to be chosen as a design parameter. It is shown in Section 4.7.2 that save on total cost with respect to the paths obtained by the heuristic cost increases with the increase of the value of j and l, but the values of j and l reaches a threshold after which the save on the total cost do not increase further. The matrices in the equation 4.14 are described

$$F = \begin{bmatrix}
1 & 0 & 0 & \dots & 0 & \vdots & 0 & 0 & \dots & 0 & 0 \\
0 & 0 & 1 & \dots & 0 & \vdots & 0 & 0 & \dots & 0 & 0 \\
0 & 0 & 0 & \dots & 1 & \vdots & 0 & 0 & \dots & 0 & 0 \\
0 & 0 & 0 & \dots & 0 & \vdots & 0 & 0 & \dots & 0 & 0 \\
\vdots & \vdots & \vdots & \dots & \vdots & \vdots & \vdots & \vdots & \dots & \vdots & \vdots \\
0 & 0 & 0 & \dots & 0 & \vdots & 0 & 1 & \dots & 0 & 0 \\
0 & 0 & 0 & \dots & 0 & \vdots & 0 & 0 & 1 & \dots & 0 \\
0 & 0 & 0 & \dots & 0 & \vdots & 0 & 0 & 0 & \dots & 1 \\
\mu & \psi_l & \psi_{l-1} & & \dots \psi_1 & \vdots & -\phi_k & -\phi_{k-1} & \cdots & -\phi_1 &
\end{bmatrix}$$

The number of rows of F depends on the number of *regression_no* and given by (2**regression_no* + 1). The matrix F contains many new terms like ψ , ϕ , μ. The ψ terms are denoted as in equation 4.16

$$\psi_l = b_l + \sum_{i=1}^{l} c_{li} X_{k-i} \tag{4.16}$$

All the ϕ terms in F are constants. The term μ is the average value of X till kth instance. Thus, the state transition matrix F depends on the travel times of the previously traversed edges. Also, the matrix V is denoted as

$$V = \begin{bmatrix} 0 & 0 & 0 & \dots & 1 & \vdots & 0 & 0 & \dots & 1 \end{bmatrix}$$

The number of rows of V is again given by (2**regression_no* + 1). The matrix H is denoted as

$$H = \begin{bmatrix} 0 & 0 & 0 & \dots & 0 & \vdots & 0 & 0 & \dots & 1 \end{bmatrix}$$

Contrary to the experiment described in Section 4.6.1, in this experiment, a state vector s is estimated over and over again whenever edge costs are required to be known to find the optimal route. Kalman filtering is applied on the state-space model (equations 4.14 and 4.15) resulting in equations 4.17 and 4.18 to estimate s repeatedly to obtain X for the connecting edges at each node to compute path using Dijkstra's algorithm.

$$\hat{s^-}(k) = F(s(k-1))s(k-1) + V\xi(k) + G\omega(k-1) \tag{4.17}$$

$$\hat{P^-}(k) = F(s(k))P(k-1)F^T(s(k-1)) + Q(k-1)$$

$$K(k) = \hat{P^-}(k)H^T[H\hat{P^-}(k)H^T + R(k)] \tag{4.18}$$

$$\hat{s}(k) = \hat{s^-}(k) + K(k)[Y(k) - H\hat{s^-}(k)]$$

$$P(k) = [I - (K(k))H]\hat{P^-}(k)$$

In equation 4.17, $\hat{s^-}(k)$ provides the apriori estimate of s. $\hat{P^-}$ provides the associated covariance matrix where $Q(k\text{-}1)$ provides the covariance for the process noise $\omega(k\text{-}1)$. In equation 4.18, $K(k)$ is the Kalman gain, $R(k)$ being the covariance of observation noise $\eta(k)$. $\hat{s}(k)$ provides the estimated state vector s at k.

$$\hat{s}(0) = E[s(0)] \tag{4.19}$$

$$P(0) = E[(s(0) - E[s(0))(s(0) - E[s(0))^T] \tag{4.20}$$

Considering the above example, $s(2) = (1, \xi_1, \xi_2, X_1, X_2)^T$ is estimated to find the travel cost of edges arising out of n_c at k =2. Kalman filtering (given in equation 4.17 and 4.18 is used to estimate the travel costs of all 5 edges, one by one. Thus, $s(2)$ is estimated for $a_{c,b}$, $a_{c,d}$, $a_{c,e}$, $a_{c,r}$ and $e_{c,g}$ separately. In this case, $s(1)$ is formed as explained earlier. The innovation variable ξ, thus, also influences the estimated travel cost as observed from the state-space equation of equation 4.14 and 4.15. Hence, X_2 is obtained from $s(2)$ after the estimation for each of the edges arising out of n_c. This process continues with the increase of k until optimal path is computed. After start of computing a path, the real travel time of edges are recorded when the MR actually traverses it. This travel times of edges are used as the observation values for the next call of path planning. Thus observation values of travel times of each edge is grown during run-time. This approach is different than Algorithm 1 in the method of obtaining the estimate of X_k only. In case of Algorithm 1, a linear state space model is used to estimate

4.7.2 Results II

This section comprises of different sets of quantitative verification through our experiments to substantiate our proposal.

Comparison of path costs

Similarly as experiment in Section 4.6.1, original Dijkstra's algorithm is used to compute optimal route with heuristic cost based on the euclidean distance of the connecting edges for multiple repetitions using in the maps (Figure 4.3), obtained from the average travel time of the edges which remains the same with progression of time. Further, we consider change of travel time over time and use estimated values of travel time as weight of edge to find optimal path. In our verification, we gradually increase the *regression_no* in to observe its effect on the total path cost. Along with the repetitions of path computations, value of ϕ is increased from -0.4 to 0.4. Negative values of ϕ produced too high estimates while values greater than 0.2 produced negative estimates. Thus, 0.2 is found as the suitable value of ϕ. Also, values of the mean and covariance of distributions b and c in equation 4.16 -0.2 to 0.2 respectively. Similarly, mean and covariance values less than 0.1 produce high estimates and more than 0.1 produce negative estimates. N(0.1,0.1) suits for both b and c. In Figure 4.11,

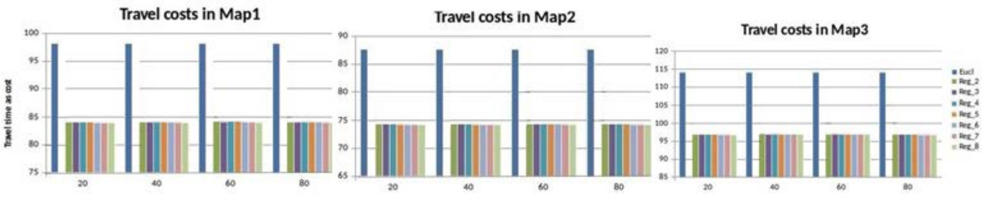


Figure 4.11: Comparison cost

vertical bars marked *Eucl* plot the average total costs of paths obtained using original Dijkstra's algorithm after multiple repetitions like 20, 40, 60 and 80 repetitions of consecutive path planning in three different maps (Figure 4.3). Also, the vertical bars marked from *Reg*_2 to *Reg*_9 plots the average total costs of paths obtained from Dijkstra's algorithm, using dynamic estimates of travel time as weight of edges with the increase of *regression*_*no* from 2 to 9. The number of repetitions denotes the span of time for which optimal route planning is conducted repeatedly.

In Figure 4.11, vertical bars of *Eucl* does not change with the number of repetition for any map as average total cost of paths does not change with the number of repetitions as euclidean cost does not vary time. Hence, it does not reflect the actual cost in reality, as actual travelling cost is based on different time-varying factors like state of charge of batteries and other environmental factors. Nevertheless, vertical bars of *Reg*_2 to *Reg*_9 in Figure 4.11, increase with the increase of repetitions. Hence, average total path cost increases with number of repetitions for all the maps. Moreover, as the number of repetition increases, the time-span for which path planning is conducted also increases. As the battery gets discharged over the time, the real cost or real travel time increases for any edge. So, as the time span of path planning increases with number of repetitions, travel cost of each edge also increases and thus the average total path cost increases accordingly. Thus, heuristic edge cost does not reflect the true cost of traversal. On the other hand, vertical bars of *Reg*_2 to *Reg*_9 are less than vertical bars of *Eucl* by 15% in average in value in all three maps. Thus, the average total path costs of paths obtained using travel time is 15% less in all three maps than that of heuristic cost based Dijkstra's algorithm. Moreover, this difference is increased with the increase of *regression*_*no*, though the rate of increase is low as the data itself is not broadly spread with standard deviation of 0.219 on average. The observation Y_k developed during run-time is considered as signal and the values of ω are modified to increased the Signal-to-Noise Ratio (SNR) from 10dB to 50 dB along with the repetitions of path planning. The vertical bars marked 10dB, 25dB and 50dB in Figure 4.12 plots the average path costs obtained by changing the SNR for each regression no. which shows that with the increase of SNR, the average travel

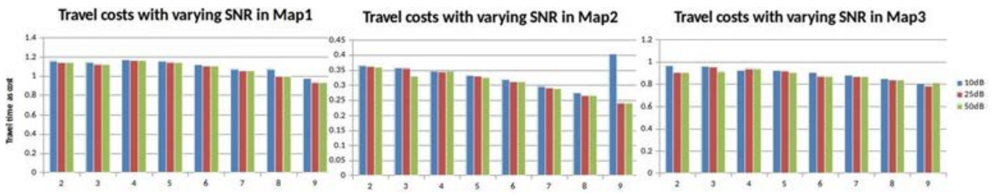


Figure 4.12: The effect of changing Signal-to-noise ratio

cost decreases.

Path comparison

Part (a) of Figure 4.13 plots three single paths Path$A$, Path$B$ and Path$C$ obtained using from Dijsktra's algorithm based on heuristic costs, statically estimated and dynamically estimated edge travel costs respectively for the same pair of source and destination nodes in Map 2 including only the variation induced by discharge of batteries.

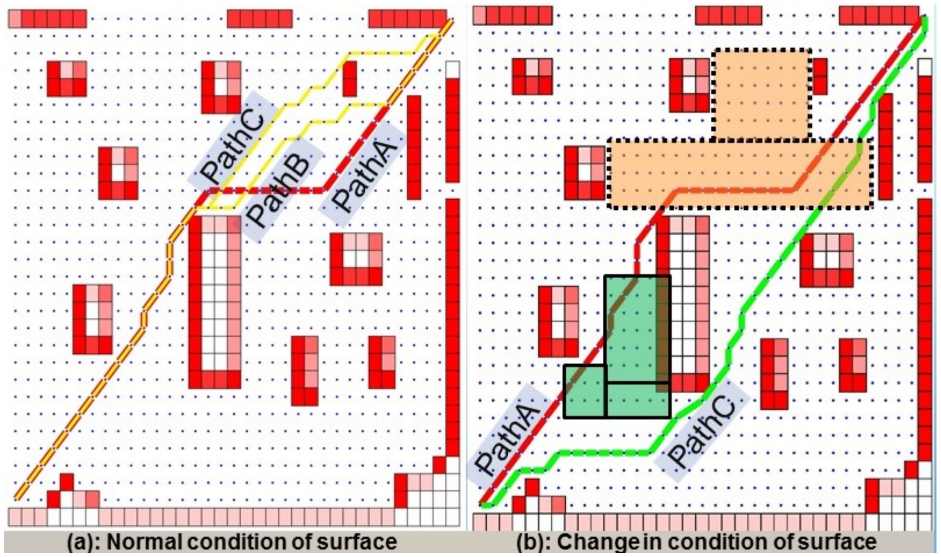

PathC
PathB
PathA
PathA
PathC
(a): Normal condition of surface
(b): Change in condition of surface

Figure 4.13: Paths in Map2

Here, path costs for Path$A$, Path$B$ and Path$C$ are denoted by P_{cA}, P_{cB} and P_{cC} stands for the general P_c vector explained in Section 4.3 for Path$A$, Path$B$ and Path$C$ respectively. Though all the paths contain the same number of edges and intermediate nodes, the edges and intermediate nodes are different in three cases. P_{cA}, P_{cB} and P_{cC} have many common elements, despite having different elements. This is because of the fact that Dijkstra's algorithm used three different kinds of weight of edges to compute the optimal path. When path planning is done using real and estimated travel costs for each edge, the edge costs of many edges becomes either more or less than the assumed heuristic cost and thus the path planner compute path with different cost values and finds a different minimal cost path than that of heuristic cost based path planner. Thus the resultant optimal paths are different in three cases. Also, optimal paths obtained using estimated travel time as weights of edges produce different and more cost efficient path.

Moreover, the total travel cost in these 3 paths are different. In Figure 4.11 in Section4.7.2, the average path cost of paths obtained usingtravel times is 15% less than that of heuristic cost based path planning using Dijkstra. After obtaining the total travel costs of P_{cA}, P_{cB} and P_{cC}, it can be stated that,

$$\sum P_{cB} < \sum P_{cA} \; by \; 5\% \; and \; \sum P_{cC} < \sum P_{cA} \; by \; 15\%$$

This establishes the proposal that heuristics based path planning can underestimate real edge travelling costs and lead to expensive paths.

On the other hand, Part (b) Figure 4.13 plots the two different paths, Path$C$ obtained by estimated travel time and Path$A$, obtained by heuristic weights. Zone marked with dotted line has moderately rough surface and zone marked with solid line has little rough surface. As heuristic weights do not reflect the increased travel cost for the changed status of surface, Path$A$ consists of edges which are inside in both the two rough zones. As the estimated travel times are used to get Path$C$, it avoids the little rough zone. The number of edges in moderately rough zone is 4 Path$C$ and 12 in Path$A$. Thus, paths obtained using estimated travel time as weights show more realistic path than that obtained by heuristic costs.

Nevertheless, Path$C$ finds a part of path in the zone with moderate roughness, because the cost incurred by moderate roughness is still less than the cost incurred in avoiding that zone and adding more nodes in the path. Thus, the deterministic robustness of computation by Dijkstra's algorithm is unchanged by using estimated travel time as weights of edges.

Similar results are obtained in case of path planning done in Map1, which are plotted in Part (a) of Figure 4.14. The zone in Map1 marked by dotted line is made moderately rough and zone marked by solid line is made little rough. Path$A$ is the path obtained using heuristic cost and Path$C$ is the path obtained by travel times. The Path$C$ clearly avoids the zone with moderate roughness, but it finds a part of path in the zone with little roughness, because the cost incurred by little roughness is still less than the cost incurred in avoiding that zone and adding more nodes in the path. When this zone is changed to heavily rough floor then Path$C$ clearly avoids this zone and finds a minimum cost path deviating to other direction and adding more nodes (Path$C$ in Part(b)). Dijkstra's algorithm takes nodes in these two zones to compute Path$A$, not being aware of the changed floor condition through heuristic cost and thus incurs more cost than apprehended. But, Path$C$ in both Part (a) and (b) is realistic it considers the real cost of traversing edges in the form of travel time to compute path.

Real cost saving for paths

According to Part (a) of Figure 4.13, there are equal number of edges in all the two paths Path$A$ and Path$C$ in Map 2, but the edges are different. In Table 4.1, under column Map 2 and Case (a), 19 edges are shown to be different between Path$A$ and

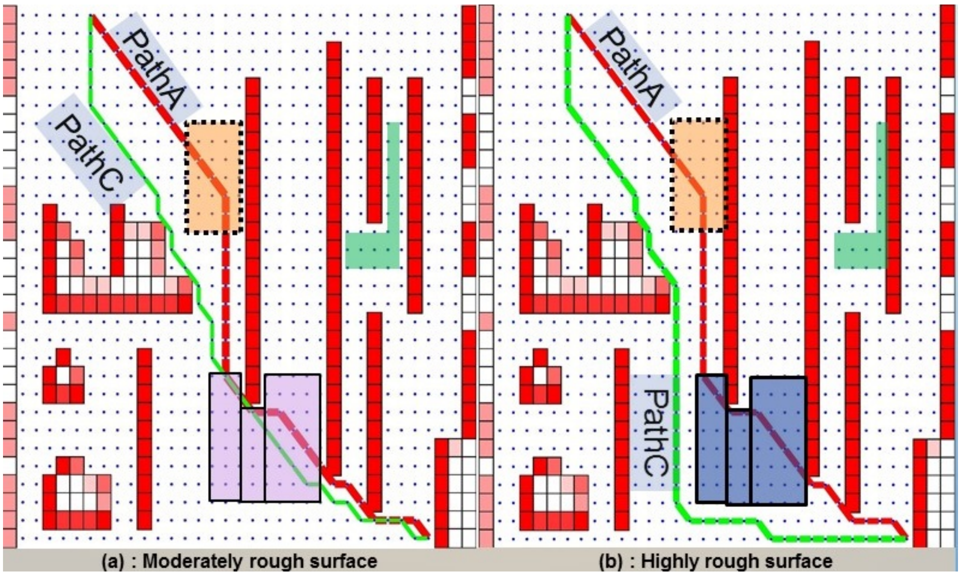


Figure 4.14: Paths in Map1

Path$C$ of Part (a) in Figure 4.13. The total path cost of Path$A$ is more than Path$C$

Table 4.1: Calculating real total path cost of PathA

	Map2		Map1		
	Case(a)	Case(b)	Case(a)	Case(b)	Case(c)
No. of different edges	19	18	12	12	12
P_{cA}	87.5	87.5	98.125	98.125	98.125
P_{cC}	74.06	74.24	84.52	82	82
Actual cost of Path$A$	89.4	90.85	100.175	100.975	101.725
Save on path cost	17.15%	18.22%	15.62%	19%	19.39%

due to these 19 edges which are different in these two paths. Moreover, real edge costs of these 19 edges however is not considered in calculating the total path cost of Path$A$ as cost of each edge are more in reality than the assumed heuristic cost. P_{cA} (path cost of Path$A$ in Part (a) of Figure 4.13) is shown as 87.5, according to the result under column Map 2 and Case (a) in Table 4.1.

We attempt to empirically find the real cost of traversing these edges and total path cost for Path$A$ to find the real save on costs. Let, a variable δ accounts for the additional edge costs in each edge. When path cost of Path$A$ is computed considering the extra cost of these 19 edges in Path$A$, it becomes $(87.5 + 19*\delta)$. Now, The value of δ can never be zero as changes in environment and batteries will always be present. Let, value of δ be 0.1 and hence the real P_{cA} becomes $(87.5 + 19*0.1) = 89.4$ (provided in third row under Case (a) under Map 2 in Table 4.1). Thus,

$$P_{cC} < P_{cA} \; by \; 17.15\%$$

Hence, save on total path cost is more than that is tangible enough to calculate. Also, we conclude that paths computed using heuristic cost may have similar edges with paths computed using real estimated travel times, but the latter is more cost efficient.

A different aspect is encountered in Part (b) of Figure 4.13, where the edges in Path$A$ and Path$C$ are completely different. This is depicted in Case (b) under Map 2 in Table 4.1. As, heuristic cost does not reflect the true traversing cost of edges, path planner has computed an optimal path consisting of edges lying in rough zones. Thus, Path$A$ consists of 13 edges from moderately rough zone and 5 edges from lightly rough zone. P_{cA} is 87.5 from the results. Here, Path$A$ and Path$C$ have different edges, still the actual cost of Path$A$ is not comprehended through heuristic edge weights as it does not reflect the changed surface conditions. We use two variables δ_1 for the moderately rough zone and δ_2 for the lightly rough zone to find the actual P_{cA}. So, actual P_{cA} = (87.5 + 13*δ_1 + 5*δ_2). Let, δ_1 be 0.2 as it is the coefficient for edges in moderately rough zone and δ_2 be 0.15 as it is the coefficient for edges in lightly rough zone. So, now, P_{cA} = (87.5 + 13*0.2 + 5*0.15) = 90.85 (provided in third row under Case (b) under Map 2 in Table 4.1). Thus,

$$P_{cC} < P_{cA} \; by \; 18.22\%$$

Thus, we conclude that heuristic cost not only is incapable of producing the real time and effort or energy to traverse an edge, but also actual cost savings are not reflected through it. Moreover, Path$C$ is more cost efficient than Path$A$, though they consist of completely different edges.

Similarly, Path$A$ of Part (a) in Figure 4.14. has 12 edges in the two rough zones in Map 1, where 5 edges in moderately rough zone and 7 edges in lightly rough zone. P_{cA} is 98.125 from the results. Similar to previous computation, we use two variables δ_1 for the moderately rough zone and δ_2 for the lightly rough zone to find the actual P_{cA}. So, actual P_{cA} = (98.125 + 5*δ_1 + 7*δ_2). Let, δ_1 be 0.2 as it is the coefficient for edges in moderately rough zone and δ_2 be 0.15 as it is the coefficient for edges in lightly rough zone. So, now, P_{cA} = (98.125 + 5*0.2 + 7*0.15) = 100.175 (provided in third row under Case (a) under Map 1 in Table 4.1). Thus,

$$P_{cC} < P_{cA} \; by \; 15.62\%$$

In Part (b) in in Figure 4.14, the lightly rough zone is changed to heavily rough zone. So, now the value of δ_2 is changed to 0.3. So, now, P_{cA} = (98.125 + 5*0.2 + 7*0.3) = 101.225 (provided in third row under Case (b) under Map 1 in Table 4.1). Thus,

$$P_{cC} < P_{cA} \; by \; 19\%$$

Moreover, we change the moderately rough zone to highly rough zone. So, now the value of δ_1 and δ_2 both have value as 0.3. So, now, P_{cA} = (98.125 + 5*0.3 + 7*0.3) = 101.725 (provided in third row under Case (c) under Map 1 in Table 4.1). Thus,

$$P_{cC} < P_{cA} \; by \; 19.39\%$$

Thus, we conclude that the difference between travel costs of paths obtained by heuristic cost and estimated travel time will always increase with the increase of hostility in the environment, which will eventually lead to more save on costs of traversing.

4.8 Summary

The travel times are investigated as a cost coefficients for edges for the task of traversing a path. They are modeled first using a linear state-space model and are estimated online using Kalman filter. These estimated values are used as weights of edges in route planning through Dijkstra's algorithm. This method is designated as static estimation in this work. It was evident from the results that total path costs of path obtained using travel time is 5% less than that obtained using Euclidean cost. This shows that the total cost of reaching the particular destination is less by using the real estimated cost. But this model do not encompass the variation of travel times with the progress of time. In order to address this, travel times are modeled using the bi-linear state dependent one and are estimated online using the Kalman filter again. This method is termed as dynamic estimation here. The results in this case showed that total path costs of path obtained using travel time is 15% less than that obtained using Euclidean cost. Hence, it is proved that travel times are better than heuristic euclidean cost as for deciding optimal path in internal transportation as they reflect the true and real-time states of battery and floor. Our approach is implemented in single robot path planning which has the opportunity to be extended to multi-robot planning. Moreover, we present a scaled platform prototyped and developed in our laboratory with analogy to real industry scenarios with a purpose of identifying the travel times and record their suitable observations with an example of real tasks similar to automated industries. We develop representative scenarios to demonstrate arrangements and operation conducted in industry into this prototype platform which will generate accurate industry like observations for these identified traveling time as costs. These observations are instrumental for feeding into estimation algorithms to estimate traveling time to be used in decision making. This approach is a representative case in the MRS with distributed control architecture. This approach can be replicated in all robots of the system with the facility of sharing the information about travel times. Nevertheless, the quantity of observed data essential for generating on-line estimates in case of static estimation and updated values of the estimates in both methods of estimation motivates to an obvious concern of data storage space in the memory devices, efficient data storage and data access mechanisms. In our work, we focus on a single MR and data for individual MR needs to be stored in the individual memory devices of the MRs in the decentralized control architecture. The estimated travel times in each MR can be accessed by other MRs through data sharing mechanism in the highest level of the individual controller, which enhances further investigation towards implementing collaborative or collective intelligence in MRS. The next chapter elaborates in this direction.

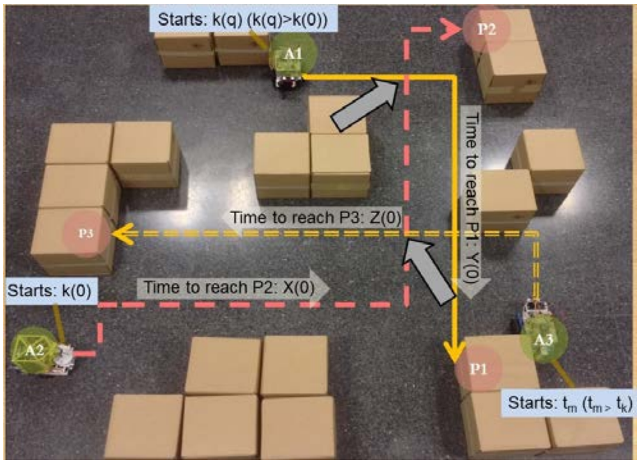
Chapter 5

Collective intelligence based on travel time in multi-robot system

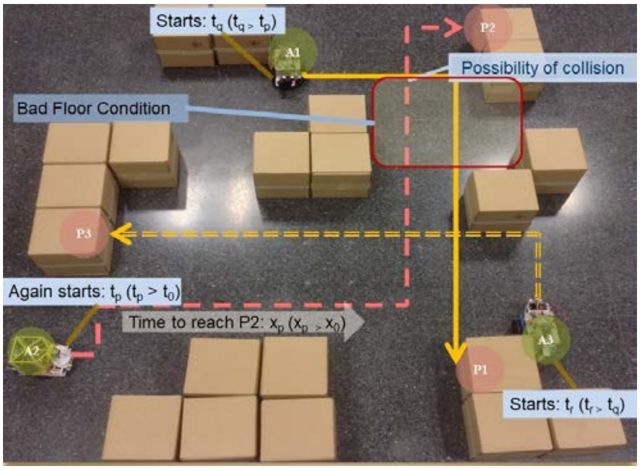
In the previous chapter, travel times of edges complemented decisions on planning for a single MR in automated logistic systems. These are influenced by state of charge of batteries (in case of battery powered MRs), condition of the mechanical parts like wheel, tyre, *et cetera*, condition of floor, weight of the load being carried, *et cetera*. So, they vary over time and they signify the cost of traversing edges. They were estimated on for the time instance when they are considered in decision making to find optimal path using Dijkstra's algorithm. These online estimated produced better cost efficient paths, than that of heuristic costs. However, online estimation requires closely packed historical data over time. In automated system, there are possibilities for a robot not to have traversed an edge or a zone in the floor at least once or travel time for one or more edge have been recorded not in recent past by a robot. This brings us to utilize these estimated travel times for sharing to other MRs in the system which will eventually enable collective intelligence based on the pool of knowledge gathered from these estimated values. In this chapter, the knowledge about floor conditions, battery status, obstacles *et cetera* are shared among all MRs through a framework of collective intelligence. This paves the way for the improved decisions for planning in the MRS which increases the efficiency and utilizes less cost.

5.1 Necessity of sharing travel time in multi-robot system

An MRS necessitates robust and cost efficient co-ordination and planning procedures. For dynamic co-ordination and planning, when all the robots are performing together, each of them can generate better and accurate estimation of travel time using the estimated travel times of others. This concept is explained in the following example



(a) Case I:



Starts: t_q ($t_q > t_p$)
P2
A1
Possibility of collision
Bad Floor Condition
P3
Again starts: t_p ($t_p > t_0$)
Time to reach P2: x_p ($x_p > x_0$)
A2
A3
P1
Starts: t_r ($t_r > t_q$)

(b) Case II:

Figure 5.1: Example Scenario for the problem.

The Figures 5.1a and 5.1b illustrates a scaled prototype internal transportation system for automated factories. Similar to previous cases, route or path traversal by any MR is again considered as a task. In this example, *A*2 carries some material to port $P2$ starting at $k(0)$ in $X(0)$ seconds, the computed path being marked by the single dotted line. Other MRs have crossing points in their paths with that of $A2$ (marked by arrows). Thus, movements of other two MRs are synchronized according to the time taken by $A2$ to reach $P2$. Starting at $k(p)$, let, $A2$ is again assigned to carry some material to port $P2$. But, the condition of floor in parts of the computed path has deteriorated (as marked in Figure 5.1b) at $k(p)$. Moreover, the battery capability of $A2$ has decreased due to execution of previous tasks. Hence, the equal amount of time ($X(0)$ seconds) and energy as previous would not be sufficient to reach port $P1$. Thus, travel time of edges needs to be estimated correctly to compute the path of minimum cost. For correct estimation, the information of the changed floor condition should be known to $A2$. As explained in Chapter 3, travel time represents the battery as well as environmental conditions. So, an MR utilizes travel time as an quantitative factor to get information about the batteries and environment in route planning (Chapter 4). But, $A2$ has not traversed this zone in previous instance or close past. Nevertheless, the other MRs may (one must have) traversed this zone in the previous instance or close past whose travel time can be used to estimate the travel time at $k(p)$. The online estimated travel times for edges will give rise to a new route or a modified route for $A2$ to reach $P2$. The resultant path will avoid the rough zone and have minimum total path traversal cost.

So, it is beneficial to communicate the travel times of one AGV to another to generate better estimations which will lead to optimal decisions. In our example, the estimated travel times for $A2$ must be computed, stored and communicated to other AGVs when required by an intelligent mechanism. These shared knowledge is to be used for accurate estimation of travel time during decision and control for each robot. Thus, the knowledge and intelligence of one robot can contribute to the formation of knowledge in other robots as well. This will generate better estimation of travel time for each robot. This essentially forms the foundation of collective intelligence.

5.2 Collective intelligence based on travel time

There are numerous rich and diverse definition of collective intelligence in literature. Nevertheless, the definition by P. Levy [48] is cited here where collective intelligence has been defined as a constantly evolving and coordinated system of distributed intelligence in real time. This definition is in accordance to the method of application of collective intelligence in MRS in this work. An MRS is dynamic as its statuses change over time. Also, it is evolving as it gathers more and more knowledge about its statuses through the course of its operation. Moreover, the source of knowledge of an MRS is distributed to each of its constituent robots. Thus, an MRS has the ingredients suitable for application of collective intelligence. We propose a framework to apply concepts of collective intelligence into the MRS involving passive crowd sourcing. The concept of passive crowd sourcing is applied in a conglomeration of entities or individuals where the individuals do not meet but still their experience can be mined to extract information in order to implement collective behaviour or intelligence. In the MRS, the individual robots cannot meet with each other, but the estimated travel times can be shared through the control architecture in each controller of the member MRs. Thus, passive crowd sourcing can fetch relevant travel times to use them in estimation to enable more accurate decisions. This framework essentially exploits collective intelligence for better optimal decisions. In subsequent sections, we explain the MRS in light of a collective and its main features which substantiates its capability to enable collective intelligence.

5.2.1 Multi-robot system: A collective

A collective is a conglomeration of autonomous, intelligent and continuously performing members.

The MRS implemented in this work is for logistics and manufacturing system which constitute robots for performing tasks. So these robots are continuously performing. The MRS has a decentralized flow of control, suitable to logistics. The flow of control is based on the concept of sub-sumption, where each robot has the same sub-sumption model. This model involves the control structure to be organized in layers one above the other with increasing level of competencies and each level can interact with all other levels with messages. This technique of flow of control is described in Figure 5.2, which consists of two major control layers. The top most

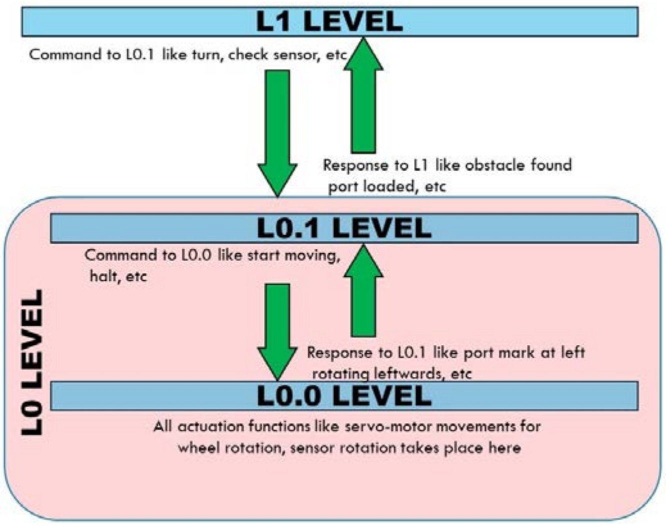

L1 LEVEL
Command to L0.1 like turn, check sensor, etc
Response to L1 like obstacle found port loaded, etc
L0.1 LEVEL
Command to L0.0 like start moving, halt, etc
Response to L0.1 like port mark at left rotating leftwards, etc
L0.0 LEVEL
All actuation functions like servo-motor movements for wheel rotation, sensor rotation takes place here
L0 LEVEL

Figure 5.2: Controller architecture

layer is $L1$ level and the $L0$ level is below it. The $L0$ level is divided into two sub levels $L0.1$ and $L0.0$ levels respectively. The $L1$ level is the agent level control layer where it functions on all the agents in the transportation system and is engaged in controlling more complex functions like finding path, organizing task, finding destination poses, *et cetera* for each of the robots. The $L0$ level functions on each of the agents individually and controls the movements. Each robot has its own $L0.0$ and $L0.1$ levels respectively which controls the movements in each of them. Here, the $L0.0$ level communicates with the $L0.1$ level and have no communication with the $L1$ level.

The $L0.1$ level is the intermediate level which communicates both with $L0.0$ level and $L1$ level. The control levels functions in co-ordination with each other to control the movements of the robots in the environment [56]. Moreover, the top most $L1$ level is responsible for intelligent decision making for task assignments and path traversal, based on the available travel time, which represents knowledge about the individual robot and the environment. Hence, each robot is capable of taking the decision itself, with the capability of gathering information about the environment from other MRs. This sub-sumption model makes each MR autonomous. All these three levels are for controlling one robot. This three-layered control structure is repeated for each MR. Thus, the control flow in the MRS is decentralized.

During the run-time of MRS, the estimation of travel time or X_k is conducted for all necessary edges while finding the optimal path. Thus, estimated values of X_k will be generated at every instance of control decisions, producing a pool of estimated values. In previous section (Section 5.1), it was explained that X_k values of one MR when shared contribute in the estimation of X_ks in other MRs. The L level conducts the planning and it need the X_ks values to be used for decisions. So, the knowledge sharing process is realized in L.

In summary, each member robot in MRS are task followers who are continuously performing. Each MR has individual layered controllers who make them autonomous. The top most $L1$ level controller in each MR is responsible for intelligent decision making for path traversal, based on the available travel time. Thus, all the basic features of a collective can be found in an MRS and we can conclude that an MRS is a collective.

5.2.2 Travel time: knowledge for collective intelligence

The essential feature of a collective which makes it feasible for collective intelligence is the inconsistency in knowledge of each member [51].

Knowledge in an MRS can be attributed to the status of the MRs regarding battery and mechanical parts, traffic condition, conditions of floor, behavior of other MRs at every instance of time. In Chapter 3, it was elaborated that X_k of an edge can represent one or all of these knowledge of MRS. In our work, X_k represents the battery status of individual MR and the floor condition. Each estimated X_k of an edge has inherent contextual information about the floor condition of that edge while being traversed by an MR at a certain battery level. Hence, these travel times or X_k values contribute to form the knowledge of the MRS. On the other hand, X_k for an edge is obtained at non-consecutive times by an MR. Thus, their are inconsistencies in this knowledge.

Chapter 4 describes the process of estimating X_k to complement the decisions on path planning for one MR. At the first call of path planning, an MR A_r acquires estimated X_k for only those edges which are linked to the explored nodes using the average observation data from the legacy system. This legacy system is built from the experiences of traversing paths by the robots before implementation of travel times. The estimated X_k values of first call are subsequently used for further estimation from second call on-wards. However, estimates and observation data were available for those edges which the robot has traversed. There are edges whose travel times are not estimated in the history of that particular MR's performance. Hence, there is no available value of X_k for these edges either in legacy or through estimation. This is explained in the example in Figure 5.3. The computation starts at different source

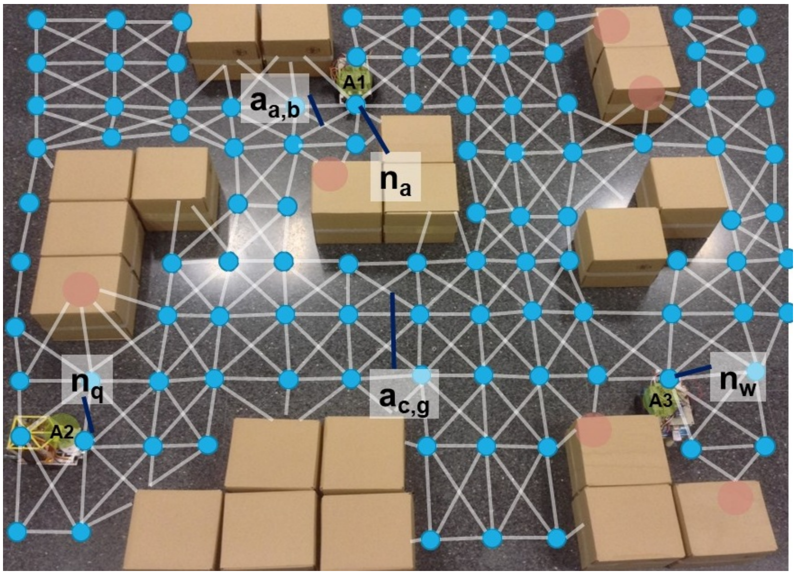


Figure 5.3: An example of path computation

nodes at the same time. In Figure 5.3, source node for $A1$ is n_a, source node for $A2$ is n_q and source node for $A3$ is n_w. So, at first call for path planning, all robots estimate travel times of edges from observation data of legacy system. Let, after few iterations of path planning, $A3$ needs to estimate $X_{c,g}$, but it has no previous history of traversing edge $a_{c,g}$. Similarly, $A2$ needs to estimate $X_{a,c}$ without having any previous experience of traversing it. For a single MR scenario, similar problem is mitigated by creating observation data from past values. Nevertheless, in case of MRS, we propose to solve it utilizing integration or fusion of data, which in essence implements collective intelligence.

5.2.3 Contribution

In the available literature, there are contributions towards using the concepts of data fusion and integration for multi-robot surveillance and rescue operations. However, there are no contributions for utilizing data integration enabling collective intelligence for multi-robot operation in automated factories and warehouse. This work is focused on developing a framework of collective intelligence for an MRS, typically used for automated logistics. Moreover, an ontology based knowledge sharing mechanism is developed to update the estimated values of the travel times during run-time and store progressively and intelligently. Also, the stored data needs to be shared to be available for usage by other robots. In other domain like traffic management system, there are instances where information about the environment and traffic are obtained in real-time and shared to form a collaborative real-time traffic information gathering and sharing framework [46]. In [46], collective intelligence is implemented through a wiki-like information sharing platform. Similarly, in our work we use an ontology model to share information of travel time, data fusion methods to resolve the inconsistency in travel time and produce estimate of travel time whenever required. All these components in essence form a collective intelligence framework in MRS.

Organization of chapter

The rest of the chapter elaborates on the different aspects of the proposed problem, where Section 5.3 explains data fusion to enable collective intelligence, feasibility of travel time for data fusion and achieving four layers of data fusion in MRS. Section 5.4 formulates the integration process mathematically which evidences that the new knowledge obtained using data fusion is more that the knowledge of the MRS. Section 5.5 describes the process of storing and sharing the data of travel times in each MR. Section 5.6 explains the process of predicting travel time in a MR using the travel times of other MRs through behavior forecasting technique. The comparison of estimates obtained using data integration and ontology with that of without using them are analyzed and tabulated in Section 5.7.3 with discussion and conclusions.

5.3 Data fusion to enable collective intelligence

Recently, data fusion or integration has spread its wings into many fields which possess similar problems, though its genesis lies in the field of military operations like tactical intelligence *et cetera.* Data integration has been defined in literature [40] as a bunch of interrelated problems of estimation and prediction of state of entities, both inside and outside of an operational system like multi-robot systems working in automated logistics [5], autonomous vehicular traffic network [80], urban search and rescue (USAR) [76], human-robot interaction [49], *et cetera.* Data fusion encompasses the interrelations among functional entities of a system.

In a collective, data originates from different sources, which makes the data inconsistent with each other. Hence, a process or mechanism is required which can dissolve the inconsistency and could provide useful data for further use. This is essentially an data integration method [51]. This data integration process is fundamentally a technique of multi-sensor data fusion. This invariably directs towards forming a paradigm of collective intelligence.

The concepts of multi-sensor data fusion is not new as it is naturally performed in animal kingdom. However, advancement of different techniques to acquire data has provided a mileage to this field of study. Multi-sensor data fusion involves transformation of observed parameters, obtained through multiple sources or sensors, into a inferential stage through hierarchical levels [41]. The sources from which data in obtained refers to the members of a collective, in the context of collective intelligence. Moreover, the hierarchical transformation results into some estimation or inference process, which is the outcome of fusion [41]. This transformation process enables the collective to dissolve the inconsistency in data from different members and provides with an inference or estimation, which forms the new knowledge in the collective.

Thus, the fundamental condition of collective intelligence, being handling inconsistency in data and formation of new knowledge, is carried out using the data fusion or integration process.

5.3.1 Information fusion using travel time

In this work, we propose to solve the inconsistency in the travel time records in history for each robot in the MRS, using the process of collective intelligence which will also enable the MRS to obtain more accurate estimation of travel times for each robot. In this attempt, data integration is ingrained which is essential to solve the problem of inconsistency in data of travel time to form the new estimation of travel time. Thus, data integration has been used to form the knowledge of the whole MRS. Further, when travel times are estimated, there are cases where one robot has traversed different arcs, but the list is not exhaustive. But, other mobile robots have traversed different arcs other than the previous one. In this case, data fusion and integration provides the mechanism to gather knowledge about the estimated travel time whose knowledge are not available to any particular robot. In this way, the gap of experience about the environmental conditions among the MRs can be bridged. For example in Figure 5.3, when $A3$ needs to estimate $X_{c,g}$, the estimate or observation can be made available to $A3$ from $A2$ at that instance of estimation to generate the new estimate for $A3$. Thus, we propose to utilize data integration techniques to generate new knowledge in MRS [51]. In this context, the different robot's record of travel times serve as different sources of data. Through data fusion, travel time records for any edge over the passage of time can be formed which is useful to obtain the estimate of travel time at current time. Thus, as a result of data fusion, the inference as estimation is formed. Also, every new estimation contributes new knowledge to the MRS. Thus, collective intelligence is fulfilled through data fusion in the MRS.

5.3.2 Four levels of data fusion in MRS

Data integration is described as a four-level process in literature [], which is termed as JDL Data fusion model. In this section, analysis and justification is provided for implementing the same four levels of data integration in the MRS.

- Level-0 involves the hypothesis of the presence of a signal from the sources and that this signal can be estimated. Moreover, the source of the signal can be a local sensor associated with the fusion system or a distributed sensor electronically linked with the fusion system [41]. In the process of implementing data fusion in the MRS, it is hypothesized that these travel time of edges are observable through the sensors of each robot. Moreover, it is conceived that each robot's sensor can observe and record the travel time for each edge while the robot traverses it. In this case, the sources of data are the robot's sensors and they are distributed in the MRS but are connected to the data fusion process through the controller architecture. In the controller, the data can flow from the sensor in the lower level (L0.0 in Section 5.2.1) to the system level controller (L1 in Section 5.2.1) where data fusion system is implemented. On the other hand, it is hypothesized that estimation of these travel times are possible during the operation of MRS, dynamically using previous records of travel times. This estimation process can start with the average values of data from the legacy system.

- Level-1 involves selecting the data from the observations in specific tracks. This process can be thought of as organizing the observations through a specific rule in groups or clusters or tracks. In the MRS, a set of observations of travel times are associated together for a particular edge for a particular robot. This set signifies the travel times of an edge over the operational time of the system. Thus Level 1 clustering is conducted during estimation of travel time for an edge, when all the relevant travel time for that edge in previous instances are associated and grouped. Thus, this clustering of observation reports is trivial in this implementation of data fusion in MRS. On the other hand, level 1 includes a hypothesis that a certain set of data from the observations or tracks is to be considered as the total set of data available to that system regarding a certain individual entity to which the data is referred. In the MRS, after the relevant information for an edge is clustered together, it appears to the system level controller that this set is the total set of values or track in history for the relevant edge for that robot.

- After Level-1 grouping, a set of data for each robot regarding the travel times of all the edges it has traversed til the current time is created in level-2 as a result of accumulating the tracks into assemblage. These tracks are hypothesized as complete set of reports available to the system in reference to any individual edge in level-1. Further in level-2, these hypothesized tracks or set of records of travel time of the edges for are associated into aggregations based on relations. These relations can be of any variety adhering to the goal of the system. In context of this work, the goal of the MRS is to decide optimal paths based on the real and estimated travel times in each robot. As the floor of the plant is described using topological map, the relations are developed based on the topological connections. Moreover, the relations between sets of travel time for any two different edges are built using topological relations. These sets of travel times serve as the knowledge and are essentially based on contextual information that can influence the decision making regarding the optimal path at individual and MRS level. The context of a travel time is developed while a MR traverses an edge. The travel time depends on various conditions like battery state of charge, floor condition, movements of other robots, physical condition of the robot, as explained in details in Chapter 3. Therefore, when a travel time is obtained it carries within itself several contextual information regarding the status of the environment and the robot. In Chapter 4, we have demonstrated that using these travel time as cost of edges can result in less cost consuming path when compared to heuristic cost. In this chapter, we implement data fusion levels to incorporate these knowledge of travel time in the whole MRS. In level-2 of the data fusion model, we develop a semantic relation to convey these contextual information inherent in travel time so as to communicate enough meaning to other MRs in the system which eventually directs towards procuring more cost efficient paths during their decision making. Level-2 is a general framework of building the relations among the sets of different travel times for different edges in each robot. This framework is a representative one which can be replicated in all the MRs in the system From the system level controller, these relations are general where only the sets are different in each robot. After the integration process in the collective or the MRS, a new knowledge is formed contributing to the pool of knowledge of the MRS. At that time, the relations of the new knowledge is estimated and given subsequent connections to be a part of the existing framework. A subset of these relations are considered in level-3

- Level-3 is involved with estimation to infer from the Level-2 association. It is a method to project the current situation into the future. Level-2 associations in MRS build the relations among the travel times and form the aggregations. In level-3, the future is predicted relation through estimating the future travel time. Thus level-3 is a special case of level-2, where it involves estimating the travel time for the required future instance of time and building the future relation. As a result of building the relation of the newly estimated travel time, the relations of new knowledge of the MRS is built. In this level, filtering is done using the methods described in Chapter 4 to obtain the new estimated travel times in each robot.

- Level-4 is solely involved with planning and control based on the newly estimated travel times.

5.4 Mathematical proof

This section is devoted to establish that a collective intelligence is formed with a integration function in the MRS and that as a result of integration process some new knowledge (new travel time) is formed in the collective (MRS) which suffices to be termed as collective intelligence. Maleszka and Nguyen in [51] have provided the mathematical description of the integration function to be applied to a collective. Also, elaborate method has been described to calculate the resulting new knowledge in the collective in [51]. In this section we will utilize these mathematical formulation to prove the process of integration and new knowledge creation in the collective intelligence process formed in the MRS.

A set C represents the knowledge of a collective where each member robot has knowledge $c \in C$. As stated earlier the knowledge of each robot is comprised of the estimated travel times which it has gathered. Now, we denote the process of integration using a integration function $I(C)$ which results to produce the new knowledge in the collective. We denote the new estimate produced from Kalman filtering based on the available estimates as the new knowledge. Hence, essentially the integration process is the Kalman filtering in our implementation of collective intelligence.

Maleszka and Nguyen in [51] has defined the postulates which $I(C)$ satisfies. We will show the proof of each postulate in respect to the integration function for our application.

1. Unanimity:
$$I(n.c) = c \tag{5.1}$$
for each $n \in N$ and $c \in U$, U being the universal set of knowledge.

   The equation 5.1 depicts that the knowledge of each member of the robot remains same after repeating the available knowledge by n. For our application, the integration of knowledge is executed through Kalman filtering, which produces new estimate at the current time for the necessary edge. The state-model for estimation is explained in previous chapter as
$$s(k) = F(s(k-1))s(k-1) + V\xi_k + G\omega_{k-1} \tag{5.2}$$
$$Y(k) = Hs(k-1) + \xi_k + \eta_k \tag{5.3}$$
   The state vector s is composed of a definite number of travel times, which are estimated at previous instances. Repeating the same travel time means inserting same elements repeatedly in s. After Kalman filtering is applied on the state-space model (equations 5.2 and 5.3) using equations 5.4 and 5.5 to estimate s at current instance, a new estimate of travel time of current edge will be produced.
$$\hat{s^-}(k) = F(s(k-1))s(k-1) + V\xi_k + G\omega_{k-1} \tag{5.4}$$
$$\begin{aligned} \hat{P^-}(k) &= F(s(k))P(k-1)F^T(s(k-1)) + Q(k-1) \\ K(k) &= \hat{P^-}(k)H^T[H\hat{P^-}(k)H^T + R(k)] \\ \hat{s}(k) &= \hat{s^-}(k) + K(k)[Y(k) - H\hat{s^-}(k)] \\ P(k) &= [I - (K(k))H]\hat{P^-}(k) \end{aligned} \tag{5.5}$$
   This process of Kalman filtering is explained in Section 4.7.1 in Chapter 4. However, when the s vector has repeated elements, the Kalman filter will produce the same new estimate and no extra estimate is produced. Thus, the knowledge of the MR remains the same after repeating the available knowledge.

2. Simplification:
$$I(C) = I(D) \tag{5.6}$$
where, collective knowledge C is for MRS_C, collective knowledge D is for MRS_D and MRS_C is a multiple of MRS_D.

   When one MRS is scaled up to another MRS, keeping the constituents same and only increasing the magnitude or number, the sources and elements of knowledge does not change. For example, MRS_C has same travel time as the source of knowledge about the environment and battery status as MRS_D, MRS_C being a scaled-up version of MRS_D. Thus, Kalman filtering, when applied to estimate the same state s with the same number of elements, will produce travel time and no new extra knowledge in MRS_C, given MRS_C is a multiple of MRS_D

3. Quasi-unanimity:
$$c \notin I(C) \Rightarrow (\exists n \in N : c \in I(E)) \tag{5.7}$$

for each $c \in U$
where $E =$ C $\uplus$ n.c and $\uplus$ symbol denotes union of multi-sets

This postulate denotes that when some knowledge c from the universal set U is repeated by a $n \in$ N, summed with the collective knowledge C and the resultant union is integrated, then c which initially did not belong to integration set of the collective C, becomes a member of the integrated set.

Now, after integrating the collective knowledge C, the resultant set may not contain the travel time of few edges. This holds true because of the fact that there can be some edges whose travel times are not estimated by any MR till current time. There are possibilities that an edge is never traversed even after repeated operation by all MRs. When the number of MRs are increased to a certain n, the travel time of these ignored edges can be obtained through these MRs. The knowledge of the current collective incorporates the travel times of these newly incorporated MRs, termed as E (let). Hence, the new collective knowledge essentially becomes a union of previous collective knowledge and new estimated travel times. Therefore, after integration operation on this collective E incorporates the estimated travel times by the additional MRs.

Above three postulates are used to define the Integration function I.

On the other hand, Maleszka and Nguyen in [51] have also provided the definition of $Aug(C)$ which is used to calculate the new and evolved knowledge in the collective C. Several conditions are defined for function Aug in [51] which we explain here according to our application.

1. For any $C \subseteq U$:
$$Aug(C) \cap C = \emptyset \tag{5.8}$$
In the MRS, the new estimated travel times are formed as a result of integration through filtering. Thus, these estimates are not a part the collective C and is only formed after filtering. So, when collective knowledge C is intersected with the new formed set of travel times, the resultant is nothing.

2. For any $C \subseteq U$:
$$Aug(C) \subseteq I(C) \tag{5.9}$$
The integration process is executed through Kalman filtering which provides the required new estimated travel times, which forms the set arising out of integration on C. On the other hand, $Aug(C)$ denotes these new estimated travel times. Thus, the resultant set $I(C)$ consisting of all the new travel times and existing ones, invariably contain $Aug(C)$.

3. If C is inconsistent then
$$Aug(C) = I(C) \setminus C \tag{5.10}$$
The new elements of knowledge form the set $Aug(C)$ and these were not present in C before estimation process. The elements of $Aug(C)$ has been inferred through estimation using the elements of C. Hence, these new estimated travel times are not a member of C, but member of the resultant set $I(C)$ arising out of integration process through filtering.

4. Aug(c) $= \emptyset$ for any $c \in U$

   If $C = c$, then there is no new element.

   Integration process leading to infer data from existing ones is no more required when there is a single MR in the system. Estimation of travel time for a single MR needs to be done from its own experience of traversing different edges, as discussed in Chapter 4. Hence $Aug(c)$ will result to nothing as no data integration is performed to infer new estimated travel times from existing knowledge elements as a collective.

5. If $I(C) = \emptyset$ then $Aug(C) = \emptyset$

   The estimated travel times of each MR together for set C. The integration of data needs to be performed to obtain new travel times. This forms the set $Aug(C)$. If somehow, this integration cannot be performed on the available knowledge, then the resultant estimated travel times will either be incorrect or not available.

6. If $Aug(C) = \emptyset$ then there exists such $Y \subseteq U$ that $Aug(\ X \cup Y) \neq \emptyset$.

   In an MRS, at the beginning of system operation, the estimation process is done using average data from the legacy. On this occasion, there are cases when integrating the knowledge of travel times for relevant edges did not produce new and accurate estimates of travel times. This occurs due to the fact that these estimates are not enough to provide accurate future estimates. The accuracy increases after few iterations as the real observation gathers and it is utilized through integration. Thus, addition of useful estimates can be enhanced by incorporating more data from existing MRs or adding more MRs in the system. When the system can start gathering observations and correct estimates, the integration process produces more useful estimates, eventually increasing the accuracy.

All the above postulates are defined as conditions for Aug which holds true in our application.

It is to be noted here that the new knowledge of travel time formed as a result of collective intelligence contributes for higher accurate future estimates. These estimates eventually reflects more realistic cost of edges. Thus, Dijkstra's algorithm decides paths which incur less total cost.

5.5 Mechanism for storing and sharing knowledge

5.5.1 Semantics in travel time

An MRS is dynamic as its states change over time. Also, it is evolving as it gathers more and more knowledge about its states through the course of its operation. Moreover, the source of knowledge of an MRS is distributed to each of its constituent robots. The behavior in $L1$ has the role of finding paths using Dijkstra's algorithm. Dijkstra's algorithm needs to know the estimated X_k for the concerned edges to decide the path as X_k designates the cost of traveling the edge. Now, there are possibilities when an MR has not yet traversed many edges. The estimation of X_k for these edges depends on the obtained travel cost of them from other MRs. Thus, knowledge sharing mechanism improves the estimation of X_k for accuracy. This will be instrumental to Dijkstra's algorithm to produce better optimal paths with minimum path costs.

The X_ks originate from each MR depending on the instance of travelling, zone of the floor, previously traversed edge, neighboring AGVs, state of charge, *et cetera.* All these factors provide context to the estimated values of X_k.

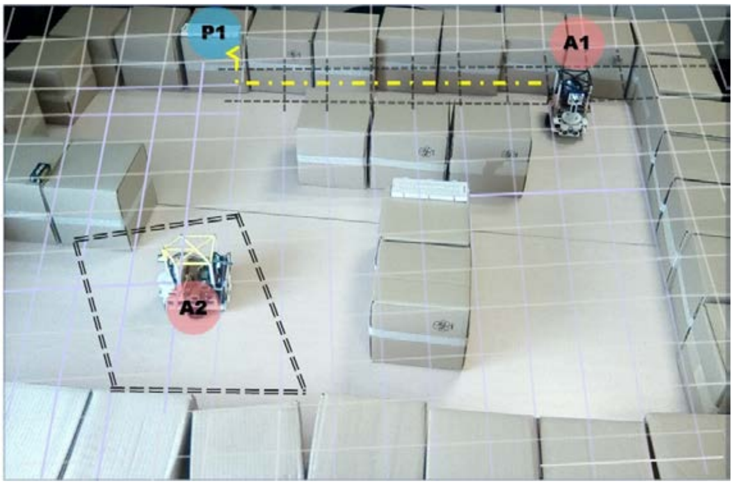


Figure 5.4: An example of semantics

For example, in Figure 5.4, the X_k of an edge by $A1$ at t_m will be different than that at t_0 due to discharging of batteries as explained in Chapter 3. On the other hand, X_k for nth edge ($n \neq m$) by $A2$ in a different zone (marked by double dotted line) will be different than that of mth edge, though both mth and nth edge can have same length. This happens due to different states of floor. Moreover, X_k for nth edge by $A1$ will be different than that by $A2$ at any t_i because of differently discharged batteries for different previous tasks. Thus, estimated travel time provides contextual information representing state of charge, condition of floor, instance of travelling.

These values of X_k at a particular instance for a particular edge of one MR provide contextual information about cost for that edge to other MRs when communicated. Hence, semantics can be built from these knowledge of travel time as they have inherent contextual information. They convey information about the costs of traversing through different edges in the topological map, which describes the factory floor.

5.5.2 Using semantics for knowledge sharing

Semantic relationships are built in this work to form semantic sentences in order to fetch the values of travel times with the inherent contextual information. This concept is illustrated in Figure 5.5.

From the above example, a semantic sentence can be derived as

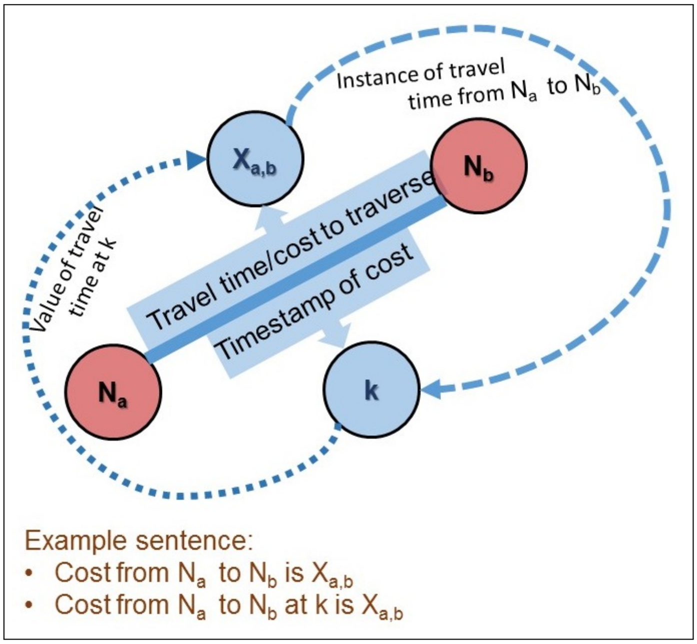

Instance of travel time from N_a to N_b
$X_{a,b}$
N_b
Value of travel time at k
Travel time/cost to traverse
Timestamp of cost
N_a
k
Example sentence:
• Cost from N_a to N_b is $X_{a,b}$
• Cost from N_a to N_b at k is $X_{a,b}$

Figure 5.5: An example of semantic relationship in MRS

- Cost from node N_a to node N_b is $X_{a,b}$ at time instance k

where, k is the instance of estimation. N_a and N_b refer to specific nodes, travel time $X_{a,b}$ refer to specific kind of cost. Cost refer to specific kind of utility expenditure while performing the task. Thus, **cost** establishes the relationship between nodes N_a and N_b and **travel time** $X_{a,b}$. When the system knows the meaning of **nodes**, **cost**, **travel time**, then the above sentence will convey some meaning to the system. This is precisely the method of developing semantics in the MRS in order to convey the contextual meaning instilled in travel time to the L level controller.

5.5.3 Ontology to represent semantics

The most traditional, flexible and useful method of representing knowledge using semantics is expressions based on subject, predicate and object logic [65]. Positioning the obtained knowledge is the next progressive step which is defined in philosophical terms as ontology. Ontology helps to create order and define relationships among things useful to an application. A domain specific ontology is developed in this work to efficiently store, access and communicate meaningful semantics across all the MRs in the system regarding the real-time travel costs of edges.

There are significant advantages of implementing ontology for the already mentioned application of this work.

- **Conceptualization of information**: An ontology is defined explicitly to form a specification for a shared conceptualization of a pool of knowledge [74], [38], [66]. Ontologies define the concepts of the domain formally and explicitly making further modifications or reversals less cumbersome.

- **Data representation**: Ontology is based on dynamic data representation where a new instance definition is not constrained to a definite rule. Thus adding new elements is easy and fast as and when required. This virtue of ontology is essentially beneficial to share the knowledge of travel time in MRS. The number of travel time grows with the increase of operation time. Moreover, reasoners in ontology solve the problem of data parity, integrity and adhering to constraints. When a new element is added to an ontology, the reasoner performs to check the integrity of the information. This capability of ontology makes the knowledge sharing method in the MRS flexible yet robust. Data addition in MRS is not required to be done on all instances and when it is added the reasoner checks for data integrity and new information can be added smoothly without adhering to rules, previously defined.

- **Modeling technique**: Ontology possesses the capability to express semantic concepts. In case of MRS, conveying the contextual information inherent to any cost parameter like travel time requires this semantic expressiveness than just defining or extracting data. Moreover, the pool of knowledge gathered in the MRS through travel time or similar parameters need to be reused which is only possible through the descriptive logic models of ontology.

In nutshell, ontology provide an unrestricted framework to represent a machine readable reality, which assumes that information can be explicitly defined, shared, reused or distributed. Moreover, information can also be interchanged and used to make deductions or queries. Such representation is imperative for representing the travel time for reasons described above.

5.5.4 Application of ontology

Semantics is an efficient way to communicate enough meaning which can actuate some action. The focus on representing semantic data is through entities. Semantic models are property oriented and semantic entities are members of a class. Semantic classes are defined on properties, it is also possible to define classes in terms of value of a property. A property type is *object property* when it signifies some abstract property like character, contribution, virtue *et cetera*. A property type is a *data property* when it signifies some literal value. On the other hand, classes can have any of the type of properties. The subclasses are defined which can avail all the properties of the superclass. The properties have range and domain. Range is the source type of a property, while domain is the destination type of the property.

Based on these concepts, the ontology stores and shares the knowledge of travel time (Figure 5.6). An ontology is developed to be implemented in each MR for their storing and sharing. The ontology is developed using Protégé, available from Stanford University, USA [58]. The ontology has two types of classes (**owl:Class**), **NS:Edge** and **NS:Node**, as shown in Figure 5.6. Thus, **NS:Edge** and **NS:Node** are subclasses of **owl:Class**. There are two properties a class can possess, **owl:ObjectProperty** and **owl:DatatypeProperty**. **NS:Origin** and **NS:Destination** are of types of **owl:ObjectProperty**, while **NS:tt**, **timeStamped** are of types of **owl:DatatypeProperty**.

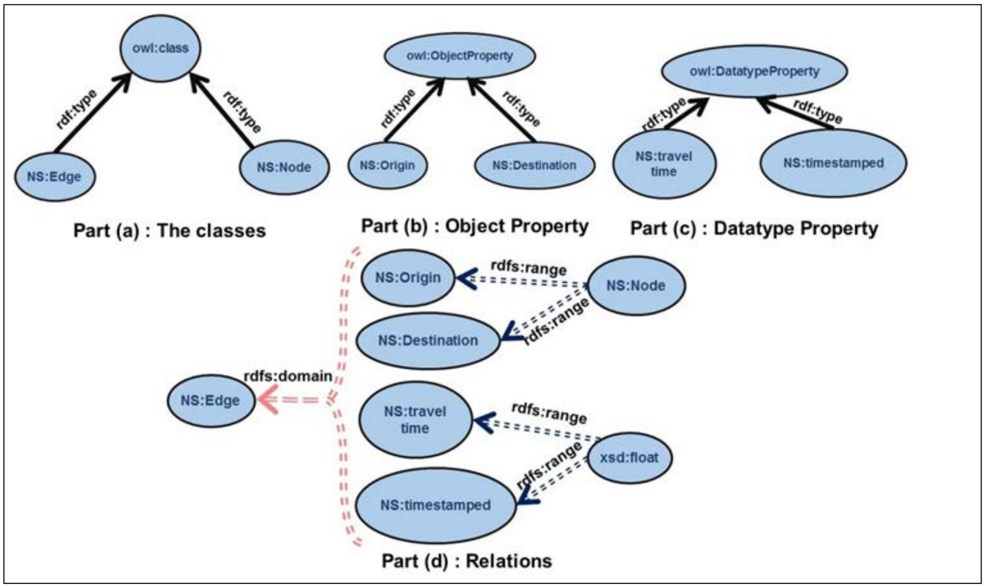


Figure 5.6: Ontology

The range of **NS:Origin** is subclass **NS:Node**, being the source type of a property, while domain is **NS:Edge** being the destination type of the property. Similarly, the range of **NS:Destination** is subclass **NS:Node**, being the source type of a property, while domain is **NS:Edge** being the destination type of the property. On the other hand, the range of **NS:tt** is a float, being the source type of a property, while domain is **NS:Edge** being the destination type of the property. Similar is the case for **timeStamped**. The tupled relationships are formed by using these domain and range connections. For example, let mth edge be between nodes n_g and n_h. $X(k)$ for mth edge at k can be formed as **NS:tt** value at **timeStamped** value k for the mth individual of subclass **NS:Edge** whose **NS:Origin** is individual n_g of subclass **NS:Node** and **NS:Destination** is individual n_h of subclass **NS:Node**. This semantic sentence can be disintegrated into several subject, predicate and object logic to derive the necessary $X(k)$. For example,

- individual mth edge is of type **NS:Edge**
- individual n_g is of type **NS:Node**
- individual n_h is of type **NS:Node**
- mth edge has **NS:Origin** n_g
- mth edge has **NS:Destination** n_h
- mth edge has **NS:tt** $X(k)$
- mth edge has **timeStamped** k

This way the **owl:ObjectProperty** and **owl:DatatypeProperty** of the subclass **NS:Edge** provides the $X(k)$ for the mth edge. Also, the $X(k)$ gets a context about its edge (between a pair of nodes) and time stamp. The advantage of this ontology lies in this formation, as discussed in previous Section 5.5.3, where any new element can be inserted through these property formations without being restrained semantically.

The structure illustrated in Figure 5.6 shows the formation of ontology which is replicated in each robot in the MRS. After developing the ontology in Protégé, it is manipulated in python to accessed for storing and sharing using RdfLib, a python library for ontology. With the use of ontology, travel time X_k can be efficiently stored annotated with a pair of nodes demarcating the edge and the time stamp of traversing it. This process can be accomplished using the modules of RdfLib. Also, when the information of travel cost for any edge for any time instance is required by any MR, X_k for that edge at the required time stamp can be retrieved from ontology of other MRs. The SPARQL query generator can be used to access relevant travel times using the modules of RdfLib.

The shared information from other MRs can provide as observation or historical data for those edges which either have not been yet travelled or have been travelled long back. This helps in achieving accurate estimates of X_k of these edges.

For example, in Figure 5.4, when $A2$ requires to estimate X_k for edges through the marked zone (marked by dotted line), the historical observation data of X_k in that zone can be obtained from the ontology of $A1$ whi h as traversed those edges in previous or nearly previous instance. The estimated values at current instance become more accurate using X_k of the same edges by $A1$ at previous instances.

This information can be sought by the $L1$ level behaviors in any MR to other $L1$ level behaviors in other MRs. Thus, this ontology fulfills the mechanism of knowledge sharing inside the L level behaviors. A co-operative approach in achieved through this knowledge sharing for better cost efficient decisions in each MR, which in turn enhances the cost efficiency of the MRS.

5.6 Retrieval of travel time and using in estimation

The sharing of travel time to all MRs is implemented through ontology in each of them to generate better estimate of travel time among all (Section 5.1). This section describes the methodology of using travel time of others in the estimation process of an MR.

The travel time of an MR is modelled using bi-linear state dependent time series [62], which is described in Section 4.7.1 in Chapter 4. This is again produced here for convenience. The bi-linear model, provided in equation 4.13, is used to model the change of travel costs depending upon all the previous travel costs.

$$\begin{aligned} & X_k + a_1 X_{k-1} + ... + a_j X_{k-j} \\ & = \xi_k + b_1 \xi_{k-1} + ... + b_l \xi_{k-l} \\ & + \sum \sum c_{rz} \xi(k-r) X_{k-z} \end{aligned} \tag{5.11}$$

The model described in equation 5.11 is a special case of the general class of non-linear models called state dependent model (SDM) [62]. In equation 5.11, X_k denotes the edge travel cost at k and ξ at k denotes the inherent variation of the edge travel cost. In equation 5.11, X_k depends on all the previous values of X and ξ, whose number is provided by the variables j and l. However, a fixed number of previous values of X and ξ is used for estimation of current X like an window which moves with increase of time. This fixed size of this window is termed as *regression number* and it is chosen as a design parameter, designated by j and l. The double summation factor over X and ξ in equation 5.11 provides the nonlinear variation of X due to state of batteries and changes in environment.

The state space form of the bi-linear model is given in equation 5.12 and equation 5.13.

$$s(k) = F(s(k-1))s(k-1) + V\xi_k + G\omega_{k-1} \tag{5.12}$$

$$Y(k) = Hs(k) + \xi_k + \eta_k \tag{5.13}$$

The equation 5.12 is the state equation which provides the next state from the current state. In equation 5.12, the state vector $s(k)$ is of the form $(1, \xi_{k-l+1}, ..., \xi_k,$ $X_{k-j+1}, ..., X_k)^T$. The state vector contains the edge costs obtained progressively over time from X_{k-j+1} to X_k. The variable ξ provides values of innovation or evolution of edge costs over the time as the exploration proceeds. Here, j denote number of previous edge costs to be included in the state vector among all edges included in the path till kth instance. Also, l denotes the number of previous evolution values of these edges. The ξ values are specific for each MR and originate from the changes in travel time of the particular MR. The values of ξ are obtained by sampling using the observation data of travel time. This observation data is obtained for the static online estimation of travel time (Section 4.6.1 in Chapter 4). The ξ values obtained through this method represents the projection of change of travel time. Though, these sampling method does not produce the perfect data to represent the change of travel time, this is suitable to this simple case where cost factor of one task is considered. This method should be improved for the case where cost factors of two or more tasks are to be considered.

The matrices of equation 5.12 are F, V and G which are explained in the following.

$$F = \begin{bmatrix}
1 & 0 & 0 & \dots & 0 & \vdots & 0 & 0 & \dots & 0 & 0 \\
0 & 0 & 1 & \dots & 0 & \vdots & 0 & 0 & \dots & 0 & 0 \\
0 & 0 & 0 & \dots & 1 & \vdots & 0 & 0 & \dots & 0 & 0 \\
0 & 0 & 0 & \dots & 0 & \vdots & 0 & 0 & \dots & 0 & 0 \\
\vdots & \vdots & \vdots & \dots & \vdots & \vdots & \vdots & \vdots & \dots & \vdots & \vdots \\
0 & 0 & 0 & \dots & 0 & \vdots & 0 & 1 & \dots & 0 & 0 \\
0 & 0 & 0 & \dots & 0 & \vdots & 0 & 0 & 1 & \dots & 0 \\
0 & 0 & 0 & \dots & 0 & \vdots & 0 & 0 & 0 & \dots & 1 \\
\mu & \psi_l & \psi_{l-1} & \dots & \psi_1 & \vdots & -\phi_k & -\phi_{k-1} & \cdots & -\phi_1 &
\end{bmatrix}$$

The number of rows of F depends on the number of *regression_no* and given by

(2*$regression\_no$ + 1). The matrix F contains many new terms like ψ , ϕ , μ. The ψ terms are denoted as in equation 5.14

$$\psi_l = b_l + \sum_{i=1}^{l} c_{li} X(k-i) \tag{5.14}$$

All the ϕ terms in F are constants. The term μ is the average value of X till kth instance. Thus, the state transition matrix F depends on the travel times of the previously traversed edges. Also, the matrix V is denoted as

$$V = \begin{bmatrix} 0 & 0 & 0 & \dots & 1 & \vdots & 0 & 0 & \dots & 1 \end{bmatrix}$$

The number of rows of V is again given by (2*$regression\_no$ + 1). The equation 5.13 is the observation equation which forms the observation for the current instance. The matrix in equation 5.13 is H which is described as

$$H = \begin{bmatrix} 0 & 0 & 0 & \dots & 0 & \vdots & 0 & 0 & \dots & 1 \end{bmatrix}$$

The observation is formed by multiplying the H matrix with the state vector s and added with the innovation at the current instance. In equation 5.13, $s(k\text{-}1)$ denotes the state vector at current instance. Here, $s(k)$ is of the form $(1, \xi_{k-l+1}, ..., \xi_k,$ $X_{k-j+1}, ..., X_k)^T$. The X values in this vector are the travel times obtained for the edges which are already explored and included in the path. But, the observation for travel time of the current edge is not available. Thus, the travel times of the required edge at same instance are gathered from other MRs. In order to gather this data, the relevant edge costs are queried in the ontology of other MRs. Then after retrieval of the data they are filled in the position of X_k in $s(k)$ in the equation 5.13. The equation 5.13 have ξ which corresponds to the innovation or change of travel time. Thus, this factor plays the role of projecting the travel time of the particular MR at some particular instance. In equation 5.12, this factor contributes not only to the formation of state but also forms the state equation to predict the next state. In equation 5.13, this ξ_k is added to the product of H and $s(k)$ to form observation. The product of H and $s(k)$ is X_k. The addition of X_k, ξ_k and noise term η_k produces the observation $Y(k)$. In this way, the travel time of other MRs are used in the model for estimation of travel time. The estimation is done by Kalman filtering. The equations obtained after applying Kalman filtering this bi-linear model are explained in Section 4.7.1 in Chapter 4. The same process is continued to obtain the travel time of relevant edges.

This travel times are used as the edge weights to decide the path using Dijkstra's algorithm. This whole process is summarized in Algorithm 2.

5.7 Experiments and results

The experimentation platform is briefly described in this section to provide elaborate explanation of the experiment. A scaled down prototype MRS for automated logistics is developed based on the decentralized controller explained in Section 5.2.1.

Algorithm 2: Dijkstra's algorithm using dynamic estimation of travel time

```
Initialise_Single_Source ($V, E, s$)
Input  : $V$-list of nodes, $E$-list of edges, $s$-source node
Output: $d[v]$-attribute for each each node, $\pi[v]$-predecessor of each node
for each $x_i \in V$ do
    $\pi[x_i]$ = infinity
    $d[x_i]$ = NIL
end
$d[s] = 0$
Main ($V, E, w, s$)
Input  : $V$-list of nodes, $E$-list of edges, $w$-edge weight matrix, $s$-source node
Output: $\pi[v]$-predecessor of each node
$P :=$ NIL
$Q := V$
$j := 0$
while $Q! = 0$ do
    $j = j+1$ $u :=$ Extract min ($Q$)
    $P := P \bigcup u$
    for each $v \in Adj[u]$ do
        $w = findTravelTime(u,v, j)$
        $relax(u,v,w)$
    end
end
findTravelTime ($u, v, j$)
Input  : $u$-current node, $v$- neighbor node, $j$-instance
Output: $w$- estimated travel_time (cost) from $u$ to $v$
$prev_x := findTTinOtherMR(u, v, j)$
$w = estimateKF(prev_x, j, X)$
findTTinOtherMR ($u, v, j$)
Input  : $u$-current node, $v$- neighbor node, $j$-instance
Output: $obsTT$-list of travel time between $u$ and $v$ by other MRs
Search for travel time at $j$ in ontology of other MRs
estimateKF ($prev_x, s, j$)
Input  : $prev_X$-previous travel time, $s$-state vector, $j$-instance for estimation
Output: $X_j$-travel cost at current j for current edge
Apply Kalman filtering to find $s_j$ and return $X_j$
Relax ($u, v, w$)
Input  : $u$-current node, $v$- neighbor node, $w$- estimated travel_time (cost)
         from $u$ to $v$
Output: $d[v]$-attribute for each each node, $\pi[v]$-predecessor of each node
if $d[v] > d[u] + w(u, v)$ then
    $d[v] = d[u] + w(u, v)$
    $\pi[v] = u$
end
```

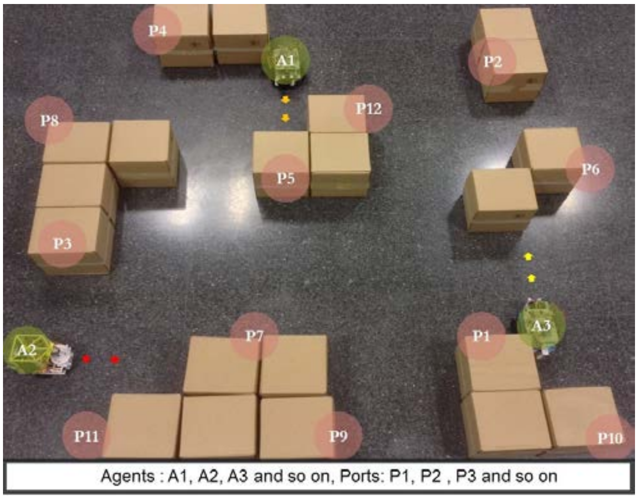


Figure 5.7: Environment of MRS

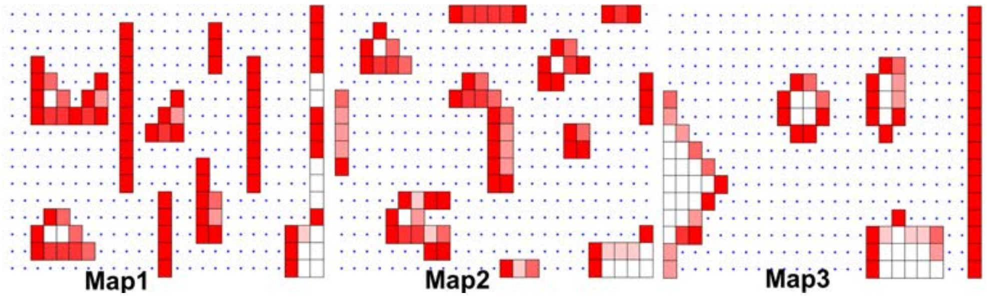


Figure 5.8: Three representative maps

An environment has been developed using uniform sized boxes as shown in Figure 5.7 for the robots to work, doing single task at a time and is named as single-task robot. The boxes create a closed labyrinth path to navigate. Also designated ports are marked on the boxes. The floor is described in three different topological maps. These maps are provided in Figure 5.8.

The control structure is same in each MR which consists of two layers of subsumption structure (Section 5.2.1). The lowest $L0.0$ level is implemented in the body of each MR inside the beagle board which forms the main processor of each robot. The middle $L0.1$ level and $L1$ are implemented in desktop PCs where each level is separated for each MR. There are four MRs functioning simultaneously in the MRS using their respective controllers. The MRs carry out the task of pick-up or drop and carrying materials between different pair of ports. The $L1$ level controller in each MR is responsible for planning decisions to make them reach designated ports. The optimal path between different pair of ports is found out using Dijsktra's algorithm.

5.7.1 Experiment -I

The MRs are made to traverse repeatedly between different pairs of nodes. The pairs are designated previously from a list in order to suit the carriage necessity. This experiment conducts path planning simultaneously in four MRs for 100 repetitions using Dijkstra's algorithm. Dijkstra's algorithm X_k as weight of an edge at every step of forming the path (Section 5.6). X_k is estimated on real-time by Kalman Filtering at required k in each MR using the bi-linear state space model. It was stated in Section 5.6 that online estimation of X_k requires observation of the same at k-1. In this experiment, these observations are gathered from the beginning of first decision making and are used in subsequent calls for estimation. But, an MR may need to estimate the travel time of one or more edge which it did not traverse previously or has traversed long back. In this case, the observation of X_k for the concerned edge is not available. In this experiment, observation of X_k for the concerned edge at k-1 could not be obtained always and thus the available observation for the concerned edge is used. Thus, X_ks are estimated solely based on the historical observation of the concerned MR. Thus, no ontology and sharing of travel time is involved to obtain observation. Dijkstra's algorithm uses the estimated X_ks for each edge. Then it chooses the predecessor node of the current node, from which arrival to current node becomes least cost consuming. This way the optimal path is formed using X_k. This experiment is designated as finding paths without sharing travel times. The paths and their total path costs obtained through this experiment are shown in Section 5.7.3.

5.7.2 Experiment -II

In this experiment, ontological data sharing is incorporated. The route computation between different pairs of node are done similarly as in Experiment I in Section 5.7.1, using online estimated values of X_k as weight of edge.

Online estimation of X_k at k requires observation of the same at (k-1). In both Experiment I and Experiment II, these observations are gathered from the beginning of first decision making and are used in subsequent calls for path planning. However, when an MR needs to estimate the travel time of one or more edge(s) which it did not traverse previously, the available observation of X for the concerned edge is at some previous instance which may not be (k-1) or close to that in many cases. These observations of distant past are used in experiment I for estimating X_k at k. Thus, this will generate less accurate estimates.

This is mitigated in this experiment II by sharing the observation value from other MR who has travelled that edge in nearly previous instance. This way the observation of X for the concerned edge at (k-1) or close will be available during estimation at k. This observation retrieval procedure is explained in Section 5.6. The knowledge sharing contributes to estimate the travel cost for an unexplored edge at current instance in an MR. The $L1$ layer in the controller of each robot can ask the $L1$ level of other neighboring robots for observation values of X_k whenever required. X_ks are estimated for the necessary edges using observations either from the own MR or from neighbors.

On the other hand, estimated X_k values of the relevant edges are used by Dijsktra's algorithm as weights of edges. These estimates are the main instrument at every step of deciding the predecessor to the current node. Dijsktra's algorithm makes a node predecessor to current, when weight or cost from the former to later becomes minimum. Thus, accurate estimated value of X_k plays a vital role in deciding the predecessor to current node, in turn deciding the path. More accurate estimates contributes to generate paths with less total cost. Optimal paths are obtained with (experiment II) and without (experiment I) sharing the X_k values. These paths are compared in Section 5.7.3.

5.7.3 Results

5.7.4 Analysis of path costs

This section illustrates the comparison of paths and their costs obtained with and without sharing the travel times in the MRS. The path planning is done for 100 repetitions while increasing the $regression_no$ from 4 to 7. Figure 5.9 illustrates

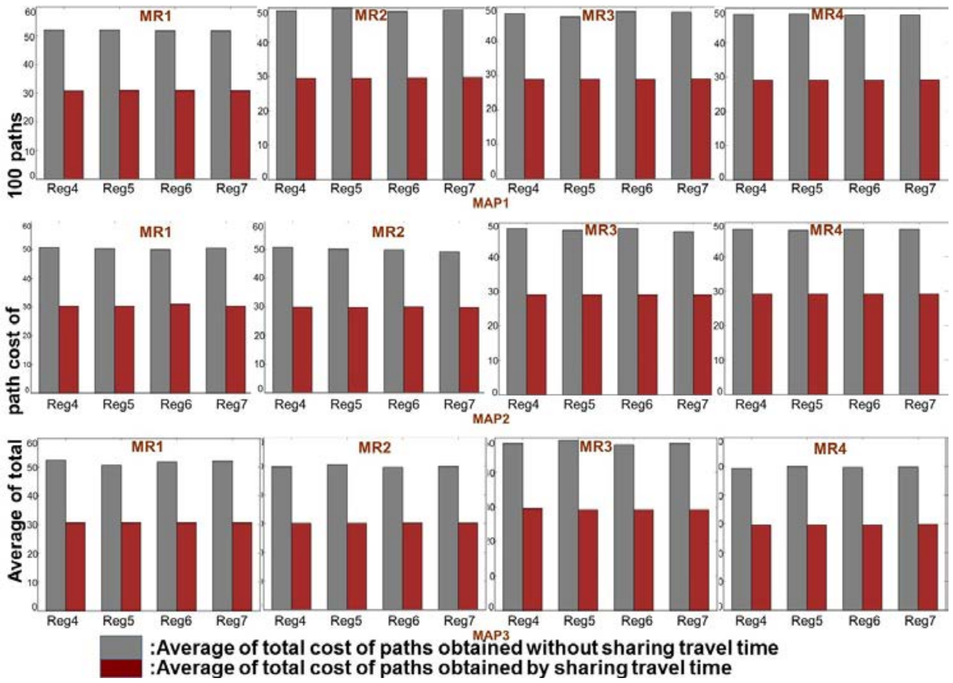


Figure 5.9: Average of total path costs

average total path costs of 100 paths obtained in both Experiment I and Experiment II in four MRs operating in all three maps (Figure 5.8). The average path costs of 100 paths obtained by sharing (Experiment II) and not sharing (Experiment I) travel times are plotted for each regression, namely $Reg4$, $Reg5$, $Reg6$ and $Reg7$.

For each regression in Map 1, the average of path costs obtained through collective intelligence are 40% less than the average of path costs obtained without it. For each MR, the average of total path costs is almost same or vary in small margin with the increase of regression number. The reason of this is the lack of variation in environmental conditions. The travel times are varying on battery condition and floor. No other factor for affecting travel time could be incorporated in the laboratory set-up.

On the other hand, the save of total path costs are same for all MRs in a single map. This signifies that paths found through collective intelligence in each MR is 40% more cost efficient than the paths obtained without it. Thus, collective intelligence using travel time can affect to find more cost efficient paths in MRS. The average path costs decrease in case of Experiment II as through collective intelligence more relevant observation of travel times are obtained in each MR. These values are instrumental for obtaining more accurate estimates of travel time. As a matter of fact, more accurate estimated values result in more optimal path with less cost than in that of obtained in Experiment I. Few examples of these paths are discussed in the next section.

Moreover, the save on total path costs is consistent in all the maps. Thus, the travel time is estimated better due to sharing of travel times from other MRs and this is true for all the representative structures of the floor. This signifies that more accurate estimation is possible through collective intelligence and this is independent of the structure of the floor.

5.7.5 Analysis of obtained paths

This section illustrates few paths obtained in Experiment I and Experiment II under the same condition of regression no and MR. Figure 5.10 plots two paths, P_A and P_B

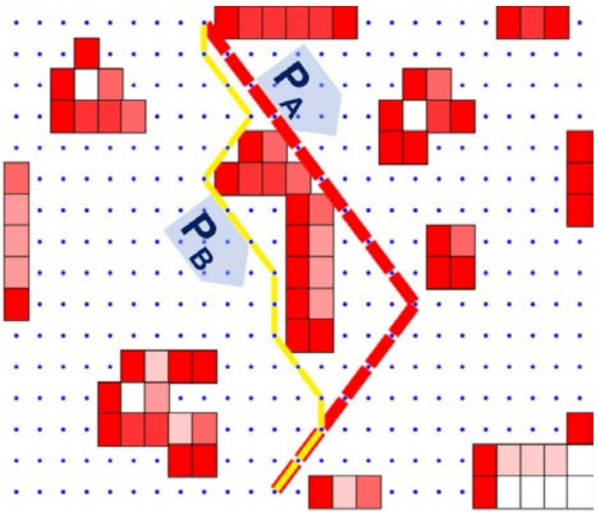


Figure 5.10: Paths found by MR 2 in Map 2

obtained in Map 1 for MR 1. P_A and P_B both have same source and destination. P_A is obtained in Experiment I in the third iteration of path planning, while P_B is obtained in Experiment II at the same iteration. Thus, they are both obtained at the same battery level and in the same map. Still, both the paths are different and have different total path cost. Similar to Section4.3.1 in Chapter4, P_c denotes the cost of a path. P_{cA} and P_{cB} denote cost of P_A and P_B of Figure 5.10 respectively. The results show $P_{cA} = 66.5326$ and $P_{cB} = 39.5385$. Thus,

$$P_{cB} < P_{cA} \text{ by } 40\%$$

Figure 5.11 plots two paths, P_C and P_D obtained in Map 2 for MR 2. P_C and

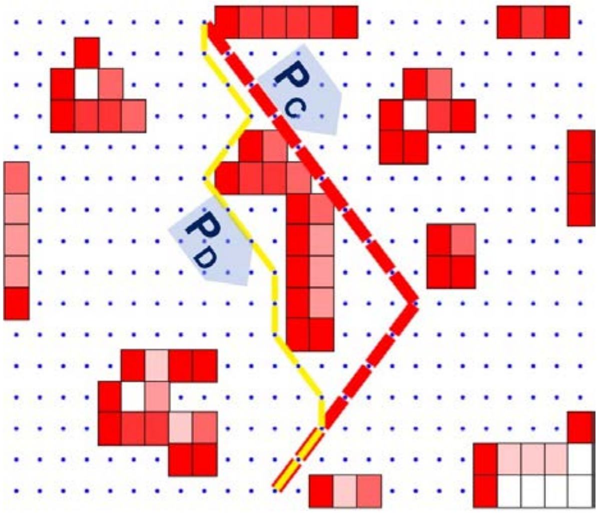


Figure 5.11: Paths found by MR 2 in Map 2

P_D both have same source and destination. P_C is obtained in Experiment I in the third iteration of path planning, while P_D is obtained in Experiment II at the same iteration. Thus, they are both obtained at the same battery level and in the same map. Still, both the paths are different and have different total path cost. Similar to Section4.3.1 in Chapter4, P_c denotes the cost of a path. P_{cC} and P_{cD} denote cost of P_C and P_D of Figure 5.10 respectively. The results show $P_{cC} = 58.0729$ and $P_{cD} = 33.5707$. Thus,

$$P_{cD} < P_{cC} \text{ by } 42\%$$

From these two comparisons, it is evident that after sharing the travel times among the MRs, the path obtained in each MR have improved and are of less cost than that obtained without the sharing.

5.8 Summary

The collective intelligence have been implemented in MRS with the goal to increase the cost efficiency of planning decisions in each MR. In previous chapter, it was shown that the decisions in route planning result in less cost path when estimated travel times are considered as cost or weights of edges. However, this estimation lacks observation data from history for some instances of estimation. This occurs when the robot has no past experience of an edge or the experience is done quite early which seems irrelevant in the current instance of estimation. In order to mitigate this, four levels of collective intelligence is implemented in MRS through which this gap in observation data is filled and new knowledge for the MRS is created. The observation values of one MR is shared to others to help in the estimation of their travel times. The travel times of one MR reflects the state of the battery and the floor condition of the edge. These information produces the time a robot takes to traverse the edge. In this context, traversing the edge becomes the behavior of the robot. The travel time quantitatively produce the behavior of the robot to traverse the edge. This behavior of one robot is used to predict or estimate the behavior of other robots in the same condition. This technique is called behavior forecasting and is commonly used in econometric for behavior forecasting for human.

The travel times from other MRs contribute to produce more accurate estimate for one MR. Thus, estimated values are improved when collective intelligence is implemented. The mathematical proof of implementing collective intelligence is also derived to show that collective intelligence is functional and fruitful in the MRS. As a result of collective intelligence, the estimates are improved in accuracy and paths with less cost are found as optimal ones.

Chapter 6

Conclusions

6.1 Importance of travel time

The new method to compute cost parameter to be used in transportation and automation industry is proposed. With this new method, parameters now reflect the states of individual robots, their batteries and their environment. They usually arise locally at the robots as a result of performances of task. It is interesting to mention here that route planning is done in Tesla's new X 75D model cars according to battery need. The route planner proposes breaks of variable times to enable recharging while travellers enjoy recess in driving. Thus, states of battery and environment are considered for optimal route planning in these models.

In context of automated logistics, the floor is designed in the form of a topology map. The robot carry out the task of traversing paths in the floor to different spots or ports. The travel time taken to traverse an edge in the map is identified as the key parameter. The experiments show that travel time varies with state of batteries and floor. Thus, travel time can reflect the condition of floor and batteries.

In case of planning, the current state of robots and environment plays crucial role. The usual practice is to decide path using Euclidean distance and a path is considered optimal with optimal length or distance. Many industries (like BlueBotics [11]) use topology maps to describe the floor and employs a depth-first search to generates a length-optimal path. However, the true cost of traversing a path is not accounted in this case. The cost involved in traversing the path is generated from condition of floor, state of batteries, mechanical parts of robots. It is intuitive that an edge of same length will incur more cost in a rough floor than in smooth one. Thus, travel time is a better tool to decide a path than heuristics based on Euclidean distance.

6.2 Online estimation of travel time

The necessity of estimating the travel times originates from the fact that they reflect realistic conditions of robots or environment or both and as the conditions in the MRS are varying over time these factors should be instilled in decisions of planning.

Hence, the decisions for task allocation, navigation and movement can eventually lead to decrease of overall cost of performing the tasks. We have conducted the experiments to evaluates different estimating methods and Kalman filtering proves to be the best method for estimating the proposed cost parameters. Also, comparison of travel times with heuristics ones shows that the former differ enough from the latter, which is based on euclidean distance to affect in the path planning.

6.3 Using travel time to generate paths

We have modified the Dijkstra's algorithm to incorporate travel time as edge costs in route planning to generate better optimized paths with minimal cost. We presented a scaled platform prototyped and developed in our laboratory with analogy to real industry scenarios for identifying travel times. We develop representative topology map to describe the floor and arrange similar situations as real industry. The proposed travel time as cost parameters are timed linked to each arc of the map. In this way environmental factors are instilled so as to determine true cost of traversing edges. In state of the art, environmental factors based costs are modeled as Gaussian process regression from already obtained finite measured data in [67], but it does not include time-varying changes in batteries and environment. On the other hand, sampling based heuristic path planning [18, 52] requires to explore a significant portion of the graph to find a suitable path, which is computationally expensive. Nevertheless, in this work, the cost of traversing every edge is estimated, which facilitates to apply deterministic path planning algorithms like Dijkstra's algorithm, Bellmont-Ford algorithm *et cetera*. The focus of our work is not proposing a new path planning method for autonomous vehicles, rather we use path planning as an example to show the efficacy of cost parameters on decisions required for planning and coordination.

In our work, travel times are estimated online in two ways, once statically and second dynamically. In static estimation, the change or exploration of travel time through the progress of time is not considered and travel time is estimated using a linear state space model. In this case, the observation data are collected offline. This estimation is done to test whether the online estimation is possible to take place along with the planning algorithm, though this process involves the cumbersome and impractical process of collecting data. The dynamic estimation is done to mitigate this problem. The travel time is modeled using bi-linear state dependent modeling techniques to include the variation in travel time due to progress of time. In this process, the observations for travel time are collected during the path planning. During the first few iterations of path planning, the travel time of an edge is estimated using available observation for it from few legacy data and when the robot actually traverses the path, the observation is gathered. Thus, the model gradually learns the observation to be used in future.

These online estimated values, both static and dynamic, are used as weights of edges to compute path using Dijkstra's algorithm. The paths in the latter cases are better and realistic as they use real estimated costs.

There are works in state of the art which consider environmental factor dependent real costs to obtain optimal path. The results of our work cannot be compared with the results of these works as these state of the art works propose a new method of path planning. We consider one kind of cost parameter like travel time to be used to decision making like path planning. In our results, we show that incorporating real and estimated travel time as edge weights in Dijkstra's algorithm does not disrupt it's basic functionality. We do not generate a new path planning method based on environmental costs to produce optimal path. Hence, we do not use sampling based or multi-agent system based path planners on elaborate and computationally expensive grid maps and cost functions. Rather, we introduce a general purpose concept of cost parameters to be used for automatic decision making. The type of cost parameters depends on the application. The cost parameters depend on *time*, *energy* and *reliability or quality* of the autonomous vehicles and these are derived from different environmental factors. This can be complimentary to the usual navigation method done in industry (like BlueBotics) where toplogy map is used and paths are computed using deterministic algorithms based on euclidean costs. This method can be improved just by deviating the costs from heuristics to travel time, keeping the original arrangement same. This will not only help to generate better minimal paths, but also reduce the overhead cost of implementing a new system.

6.4 Support of all MRs to everyone through collective intelligence

In this work, the decision making of each robot is based solely on the travel costs of its own. In the dynamic estimation process, there are possibilities of not being able to learn observation of few edges due to lack of experience. Also, the observation gathered for a particular edge is too old to be relevant at the current instance of estimation. To address this, the collective intelligence is incorporated to be able to share data of travel time from one MR to others. This enables the MR to generate more accurate estimation for travel times.

6.5 Future directions

Also, cost parameters can be found out in any process and its kind depends on the task to be performed. Thus, it can implemented in any decision-making process, other than path planning, for coordination and control, not only in automated logistics but also other automatic processes.

In most cases, path planning for autonomous robots are solved considering both static and dynamic environment, separately. Also, there are approaches solving path planning considering unknown dynamic environment. But, environment for internal transportation in automated logistics and warehouse facilities are static. Moreover, we propose cost parameters as a general concept to be used for planning to achieve increased cost efficiency in case of autonomous robots and we consider path planning as a specific planning and coordination problem. So, we do not consider unknown and dynamic environment in this work. Nevertheless, there are scope for further research to investigate the behavior of cost parameters in case of planning and coordination problems in unknown and dynamic environment.

In our current work, a single task is considered for each MR. Here, ’single task’ means traversing edges over and over again. However, in real industrial scenario, each AGV is capable of performing multiple tasks. Hence, in the future investigations, travel times are to be considered in two-task robots where two different tasks will be interleaved with one another. The nature of these cost parameters will change for each MR.

Bibliography

[1] Markus W Achtelik et al. "Path planning for motion dependent state estimation on micro aerial vehicles". In: *Robotics and Automation (ICRA), 2013 IEEE International Conference on.* IEEE. 2013, pp. 3926–3932.

[2] Faez Ahmed and Kalyanmoy Deb. "Multi-objective optimal path planning using elitist non-dominated sorting genetic algorithms". In: *Soft Computing* 17.7 (2013), pp. 1283–1299.

[3] Maram Alajlan et al. "Global path planning for mobile robots in large-scale grid environments using genetic algorithms". In: *Individual and Collective Behaviors in Robotics (ICBR), 2013 International Conference on.* IEEE. 2013, pp. 1–8.

[4] I. F. Chaile Alfaro and L. Ribas-Xirgo. "MASYM, a Framework to Deploy Synchronized Industrial Systems Based on Any ABM Simulator". In: *IEEE Latin America Transactions* 13.10 (2015), pp. 3244–3252.

[5] I. F. Chaile Alfaro and L. Ribas Xirgo. "MASYM, a Framework to Deploy Synchronized Industrial Systems Based on Any ABM Simulator". In: *IEEE Latin America Transactions* 13.10 (2015), pp. 3244–3252.

[6] Gianluca Antonelli. "Interconnected dynamic systems: An overview on distributed control". In: *IEEE Control Systems* 33.1 (2013), pp. 76–88.

[7] Tamio Arai, Enrico Pagello, and Lynne E Parker. "Advances in multi-robot systems". In: *IEEE Transactions on robotics and automation* 18.5 (2002), pp. 655–661.

[8] Haluk Bayram and H Işıl Bozma. "Coalition formation games for dynamic multirobot tasks". In: *The International Journal of Robotics Research* 35.5 (2016), pp. 514–527.

[9] Patric Beinschob and Christoph Reinke. "Graph SLAM based mapping for AGV localization in large-scale warehouses". In: *Intelligent Computer Communication and Processing (ICCP), 2015 IEEE International Conference on.* IEEE. 2015, pp. 245–248.

[10] Subhrajit Bhattacharya, Maxim Likhachev, and Vijay Kumar. "Multi-agent path planning with multiple tasks and distance constraints". In: *Robotics and Automation (ICRA), 2010 IEEE International Conference on.* IEEE. 2010, pp. 953–959.

[11] *BlueBotics.* 2001 (accessed April 12, 2018). URL: `http://www.bluebotics.com/products/`.

[12] R. Brooks. “A robust layered control system for a mobile robot”. In: *IEEE Journal on Robotics and Automation* 2.1 (1986), pp. 14–23.

[13] Jan Carstensen et al. “Condition Monitoring and Cloud-based Energy Analysis for Autonomous Mobile Manipulation - Smart Factory Concept with LUHbots”. In: *Procedia Technology* 26 (2016), pp. 560–569.

[14] I. F. Chaile and L.R. Xirgo. “Agent simulator-based control architecture for rapid development of multi-robot systems”. In: *International Conference on Systems, Control, Signal Processing and Informatics (SCSI 2015), INASE Joint ConferencesBarcelona, Spain.* 2015.

[15] I. F. Chaile and L.R. Xirgo. “Running Agent-based-models Simulations Synchronized with Reality to Control Transport Systems”. In: *Automatika* 57 (2017), pp. 452–465.

[16] Rohit Chandra and Rui P Rocha. “Knowledge-based Framework for Human-Robots Collaborative Context Awareness in USAR Missions”. In: *Autonomous Robot Systems and Competitions (ICARSC), 2016 International Conference on.* IEEE. 2016, pp. 335–340.

[17] Xiong Chen et al. “A fast two-stage ACO algorithm for robotic path planning”. In: *Neural Computing and Applications* 22.2 (2013), pp. 313–319.

[18] Marcello Cirillo, Tansel Uras, and Sven Koenig. “A lattice-based approach to multi-robot motion planning for non-holonomic vehicles”. In: *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on.* IEEE. 2014, pp. 232–239.

[19] Jan Claes and Geert Poels. “Merging event logs for process mining: A rule based merging method and rule suggestion algorithm”. In: *Expert Systems with Applications* 41.16 (2014), pp. 7291–7306.

[20] Mitchell Colby, Jen Jen Chung, and Kagan Tumer. “Implicit adaptive multi-robot coordination in dynamic environments”. In: *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on.* IEEE. 2015, pp. 5168–5173.

[21] Giuseppe Confessore, Marcello Fabiano, and Giacomo Liotta. “A network flow based heuristic approach for optimising AGV movements”. In: *Journal of Intelligent Manufacturing* 24.2 (2011), pp. 405–419.

[22] Thomas H Cormen. *Introduction to algorithms.* MIT press, 2009.

[23] Vasco Costa et al. “Design and development of an inexpensive aquatic swarm robotics system”. In: *OCEANS 2016-Shanghai.* IEEE. 2016, pp. 1–7.

[24] Giuseppe Cotugno et al. “Extended and Unscented Kalman Filters for mobile robot localization and environment reconstruction”. In: *Control & Automation (MED), 2013 21st Mediterranean Conference on.* IEEE. 2013, pp. 19–26.

[25] Pragna Das and Lluís Ribas-Xirgo. "A study of time-varying cost parameter estimation methods in automated transportation systems based on mobile robots". In: *Emerging Technologies and Factory Automation (ETFA), 2016 IEEE 21st International Conference on*. IEEE. 2016, pp. 1–4.

[26] Pragna Das and Lluís Ribas-Xirgo. "Predicting Battery Level Analysing the Behaviour of Mobile Robot". In: *Physical Agents, XVII Workshop of*. 2016, pp. 91–98.

[27] S Hamid Dezfoulian, Dan Wu, and Imran Shafiq Ahmad. "A generalized neural network approach to mobile robot navigation and obstacle avoidance". In: *Intelligent Autonomous Systems 12*. Springer, 2013, pp. 25–42.

[28] M. B. Dias et al. "Market-Based Multirobot Coordination: A Survey and Analysis". In: *Proceedings of the IEEE* 94.7 (2006), pp. 1257–1270.

[29] E. DiGiampaolo and F. Martinelli. "Mobile Robot Localization Using the Phase of Passive UHF RFID Signals". In: *IEEE Transactions on Industrial Electronics* 61.1 (2014), pp. 365–376.

[30] R. Drath and A. Horch. "Industrie 4.0: Hit or Hype? [Industry Forum]". In: *IEEE Industrial Electronics Magazine* 8.2 (2014), pp. 56–58.

[31] Mohamed Elbanhawi and Milan Simic. "Sampling-based robot motion planning: A review". In: *IEEE Access* 2 (2014), pp. 56–77.

[32] Alessandro Farinelli, Luca Iocchi, and Daniele Nardi. "Distributed on-line dynamic task assignment for multi-robot patrolling". In: *Autonomous Robots* (2016), pp. 1–25.

[33] Fabrizio Flacco et al. "On-line estimation of variable stiffness in flexible robot joints". In: *The International Journal of Robotics Research* 31.13 (2012), pp. 1556–1577.

[34] Avinash Gautam and Sudeept Mohan. "A review of research in multi-robot systems". In: *Industrial and Information Systems (ICIIS), 2012 7th IEEE International Conference on*. IEEE. 2012, pp. 1–5.

[35] Maxime Gautier, Alexandre Janot, and Pierre-Olivier Vandanjon. "A new closed-loop output error method for parameter identification of robot dynamics". In: *Control Systems Technology, IEEE Transactions on* 21.2 (2013), pp. 428–444.

[36] B Gerkey and Maja J Mataric. "Are (explicit) multi-robot coordination and multi-agent coordination really so different". In: *Proceedings of the AAAI spring symposium on bridging the multi-agent and multi-robotic research gap*. 2004, pp. 1–3.

[37] A. Grau et al. "Industrial robotics in factory automation: From the early stage to the Internet of Things". In: *IECON 2017 - 43rd Annual Conference of the IEEE Industrial Electronics Society*. IEEE, 2017, pp. 6159–6164.

[38] Thomas R. Gruber. "A translation approach to portable ontology specifications". In: *Knowledge Acquisition* 5.2 (1993), pp. 199–220.

[39] M. Guo, M. Egerstedt, and D. V. Dimarogonas. “Hybrid control of multi-robot systems using embedded graph grammars”. In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 5242–5247.

[40] D. L. Hall and J. Llinas. “An introduction to multisensor data fusion”. In: *Proceedings of the IEEE* 85.1 (1997), pp. 6–23.

[41] D. L. Hall and J. Llinas. “An introduction to multisensor data fusion”. In: *Proceedings of the IEEE* 85.1 (1997), pp. 6–23.

[42] Y. Hanada, G. Lee, and N. Y. Chong. “Adaptive Flocking of a Swarm of Robots Based on Local Interactions”. In: *2007 IEEE Swarm Intelligence Symposium*. IEEE. 2007, pp. 340–347.

[43] Kelin Jose and Dilip Kumar Pratihar. “Task allocation and collision-free path planning of centralized multi-robots system for industrial plant inspection using heuristic methods”. In: *Robotics and Autonomous Systems* 80 (2016), pp. 34–42.

[44] Jens Krause, Graeme D Ruxton, and Stefan Krause. “Swarm intelligence in animals and humans”. In: *Trends in ecology & evolution* 25.1 (2010), pp. 28–34.

[45] Tae-Seok Lee et al. “Mobile robot navigation with reactive free space estimation”. In: *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*. IEEE. 2010, pp. 1799–1804.

[46] Wei-Hsun Lee, Shian-Shyong Tseng, and Wern-Yarng Shieh. “Collaborative real-time traffic information generation and sharing framework for the intelligent transportation system”. In: *Information Sciences* 180.1 (2010). Special Issue on Collective Intelligence, pp. 62–70.

[47] Charles Lesire et al. “A distributed architecture for supervision of autonomous multi-robot missions”. In: *Autonomous Robots* 40.7 (2016), pp. 1343–1362.

[48] P. Levy. “Collective Intelligence: Mankind’s Emerging World in Cyberspace”. In: Plenum Press, 1994.

[49] R. C. Luo, Y. C. Wu, and P. H. Lin. “Multimodal information fusion for human-robot interaction”. In: *2015 IEEE 10th Jubilee International Symposium on Applied Computational Intelligence and Informatics*. IEEE, 2015, pp. 535–540.

[50] Thi Thoa Mac et al. “Heuristic approaches in robot path planning: A survey”. In: *Robotics and Autonomous Systems* 86 (2016), pp. 13–28.

[51] Marcin Maleszka and Ngoc Thanh Nguyen. “Integration computing and collective intelligence”. In: *Expert Systems with Applications* 42.1 (2015), pp. 332–340.

[52] Neil Mathew, Stephen L Smith, and Steven L Waslander. “Optimal path planning in cooperative heterogeneous multi-robot delivery systems”. In: *Algorithmic Foundations of Robotics XI*. Springer, 2015, pp. 407–423.

[53] François Michaud and Monica Nicolescu. “Behavior-Based Systems”. In: *Springer Handbook of Robotics*. Ed. by Bruno Siciliano and Oussama Khatib. Springer International Publishing, 2016, pp. 307–328.

[54] Stephen S Nestinger and Michael A Demetriou. “Adaptive collaborative estimation of multi-agent mobile robotic systems”. In: *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE. 2012, pp. 1856–1861.

[55] Tim Niemueller et al. *Cyber-Physical System Intelligence*. Ed. by Sabina Jeschke et al. Springer International Publishing, 2017, pp. 447–472.

[56] A. Norouzi and C. A. Acosta. “An approach to design a robust software architecture and an intelligent model for multi-agent systems”. In: *2013 3rd Joint Conference of AI Robotics and 5th RoboCup Iran Open International Symposium*. IEEE, 2013, pp. 1–7.

[57] Reza Nourjou et al. “Dynamic assignment of geospatial-temporal macro tasks to agents under human strategic decisions for centralized scheduling in multi-agent systems”. In: *International Journal of Machine Learning and Computing* 4.1 (2014), p. 39.

[58] M. Obitko and V. Marik. “Ontologies for multi-agent systems in manufacturing domain”. In: *Proceedings. 13th International Workshop on Database and Expert Systems Applications*. IEEE, 2002, pp. 597–602.

[59] Panasonic. *eneloop*. 2005. URL: `https://www.panasonic.com/global/consumer/battery/eneloop.html`.

[60] R. Parasuraman et al. “Model Based On-Line Energy Prediction System for Semi-autonomous Mobile Robots”. In: *2014 5th International Conference on Intelligent Systems, Modelling and Simulation*. 2014, pp. 411–416.

[61] Lynne E. Parker, Daniela Rus, and Gaurav S. Sukhatme. *Multiple Mobile Robot Systems*. Ed. by Bruno Siciliano and Oussama Khatib. Springer International Publishing, 2016, pp. 1335–1384.

[62] MB Priestley. “Current developments in time series modelling”. In: *Journal of Econometrics* 37.1 (1988), pp. 67–86.

[63] L. Ribas-Xirgo, J. M. Moreno-Villafranca, and I. F. Chaile. “On using automated guided vehicles instead of conveyors”. In: *2013 IEEE 18th Conference on Emerging Technologies Factory Automation (ETFA)*. Sept. 2013, pp. 1–4.

[64] Indranil Saha et al. “Implan: scalable incremental motion planning for multi-robot systems”. In: *Cyber-Physical Systems (ICCPS), 2016 ACM/IEEE 7th International Conference on*. IEEE. 2016, pp. 1–10.

[65] Toby Segaran et al. *Programming the Semantic Web*. 1st. O’Reilly Media, Inc., 2009.

[66] Rudi Studer, V.Richard Benjamins, and Dieter Fensel. “Knowledge engineering: Principles and methods”. In: *Data & Knowledge Engineering* 25.1 (1998), pp. 161–197.

[67] J. Suh and S. Oh. “A cost-aware path planning algorithm for mobile robots”. In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2012, pp. 4724–4729.

[68] Vivek A Sujan and Steven Dubowsky. “An optimal information method for mobile manipulator dynamic parameter identification”. In: *Mechatronics, IEEE/ASME Transactions on* 8.2 (2003), pp. 215–225.

[69] Cheng-Kuo Sung, Nora Ayanian, and Daniela Rus. “Improving the performance of multi-robot systems by task switching”. In: *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE. 2013, pp. 2999–3006.

[70] Kagan Tumer and David Wolpert. “Collective intelligence and Braess’ paradox”. In: *Aaai/iaai*. 2000, pp. 104–109.

[71] “Using battery-electric {AGVs} in container terminals — Assessing the potential and optimizing the economic viability”. In: *Research in Transportation Business & Management* 17 (2015), pp. 99–111.

[72] Kelen Vivaldini et al. “Integrated tasks assignment and routing for the estimation of the optimal number of AGVS”. In: *The International Journal of Advanced Manufacturing Technology* 82.1 (2015), pp. 719–736.

[73] Zheng Wang, Angelika Peer, and Martin Buss. “Fast online impedance estimation for robot control”. In: *IEEE International Conference on Mechatronics, 2009. ICM*. IEEE. 2009, pp. 1–6.

[74] Daya C. Wimalasuriya and Dejing Dou. “Ontology-based information extraction: An introduction and a survey of current approaches”. In: *Journal of Information Science* 36.3 (2010), 306–$_3$23.

[75] V. Zadorozhny and M. Lewis. “Information Fusion Based on Collective Intelligence for Multi-robot Search and Rescue Missions”. In: *2013 IEEE 14th International Conference on Mobile Data Management*. Vol. 1. IEEE, 2013, pp. 275–278.

[76] V. Zadorozhny and M. Lewis. “Information fusion for USAR operations based on crowdsourcing”. In: *Proceedings of the 16th International Conference on Information Fusion*. IEEE. 2013, pp. 1450–1457.

[77] Yong Zhang, Dun-wei Gong, and Jian-hua Zhang. “Robot path planning in uncertain environment using multi-objective particle swarm optimization”. In: *Neurocomputing* 103 (2013), pp. 172–185.

[78] Anmin Zhu and Simon X Yang. “A survey on intelligent interaction and cooperative control of multi-robot systems”. In: *Control and Automation (ICCA), 2010 8th IEEE International Conference on*. IEEE. 2010, pp. 1812–1817.

[79] Yan-fei Zhu and Xiong-min Tang. “Overview of swarm intelligence”. In: *Computer Application and System Modeling (ICCASM), 2010 International Conference on*. Vol. 9. IEEE. 2010, pp. V9–400.

[80] Adam Ziebinski et al. “A Survey of ADAS Technologies for the Future Perspective of Sensor Fusion”. In: *Computational Collective Intelligence: 8th International Conference, ICCCI 2016, Halkidiki, Greece, September 28-30, 2016. Proceedings, Part II*. Springer, 2016, pp. 135–146.

Appendices

Appendix A

Implementation of decentralized control

A.1 Implementation

The control technique [14],[15], described in previous section, is achieved using behaviors as the building block of both decision-making level ($L1$ level) and action execution level ($L0$ level). Separate sets of behaviors are designed for two layers as illustrated in Figure A.1, which is based on the control framework proposed by R. Brooks in [12]. The hierarchical control framework has three behavioral levels and each level has an objective and a corresponding output, formed as commands or replies. Moreover, the process of execution of all levels start simultaneously. However, the output in the form of commands from the highest level ($L1$) need to pass on to the next priority level ($L0.1$) for it to start execution, and similar process is followed in $L0.0$ level. This happens because of the hierarchical framework and the command from the highest-priority behavioral level is required as input to process the low-priority behavioral level. On the other hand, the replies from the lower level act as a feed back to the control rules of the higher level which determines the final decision and output of the control framework.

The general design of an MR is considered in the prototype system which consists of servo-motors to rotate wheels and camera. The sensing is conducted with infra-red sensors and camera. The beagle-bone forms the processor for the robot. Each MR in the system is autonomous with its own three level of behavioral control framework. The following are the behaviors developed in each level.

- In $L0.0$ level, actuation behaviors are developed. This behavior conducts the starting of motors for wheel rotation, camera movements and infra-red sensor movements. The commands which refer to target poses are obtained from $L0.1$ level in this behavior using extended finite state machines to conduct the movement of the individual robots. The sensor readings are transferred to $L0.1$ for processing as feedback of commands.
- In $L0.1$ level, three behaviors are developed. They are generating target poses

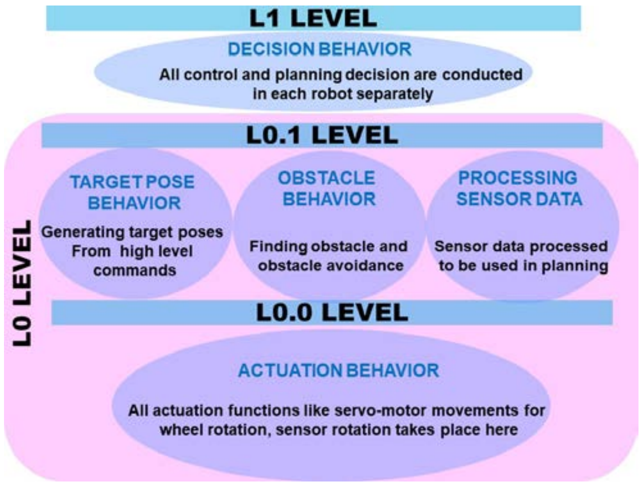

L1 LEVEL
DECISION BEHAVIOR
All control and planning decision are conducted in each robot separately
L0 LEVEL
L0.1 LEVEL
TARGET POSE BEHAVIOR
Generating target poses From high level commands
OBSTACLE BEHAVIOR
Finding obstacle and obstacle avoidance
PROCESSING SENSOR DATA
Sensor data processed to be used in planning
L0.0 LEVEL
ACTUATION BEHAVIOR
All actuation functions like servo-motor movements for wheel rotation, sensor rotation takes place here

Figure A.1: Behaviors for the layers of control

from high level commands like destination port, finding obstacle and obstacle avoidance, processing sensor data to be used in planning and decision in $L1$. All these behaviors are developed using extended finite state machine.

- In $L1$ level, decision making behaviors are developed which are finding paths and assigning tasks. These behaviors are developed using extended finite stacked-state machine. Also, behavior of maintaining and sharing the knowledge of travel time is developed. More details about the sharing mechanism is provided in next sections.