Path Planning for Autonomous Multi-Vehicle Team in Dynamic Environment

Mudassir Jan

University of New South Wales at the Australian Defence Force Academy

This thesis paper aims at investigating the dynamic path planning of an autonomous multi-vehicle team to execute a task of traversing a certain terrain with specified start and goal state. The area under consideration also includes a specified number of checkpoints to be visited by at-least one vehicle of the team while avoiding the static and dynamic obstacles. The decision about the traversal to the checkpoints is made by the vehicles themselves. Path-Planning of single vehicle is a well-researched topic; the aim during this work is to extend the concept to multi-vehicle path planning in order to reduce the workload of individual vehicles. The aim is also to reduce the computational time and total distance travelled by all the vehicles. MATLAB was used to program these problems and results have been presented in a graphical manner. These simulations and graphical representation are depicting the real life path planning and obstacle avoidance.

Contents

I. Introduction 2
   A. Background 2
   B. Context and Motivation 2
   C. Aim 2
II. Literature Review 3
III. Problem Definition 3
IV. Constraints and Assumptions 3
V. Methodology and Analysis 4
   A. Working of Algorithms 4
   B. Comparison of Algorithms 5
   C. Dynamic Obstacles 6
   D. Introduction of Checkpoints 6
   E. Multi-Vehicle Path Planning 6
VI. Discussion of Results 8
VII. Conclusions 10
VIII. Future Prospects 10

Acknowledgements 11
References 11

Nomenclature

F (n) = Total cost of traversal function
G (n) = Actual cost of traversal from start node to current node
H (n) = Heuristic estimate of the cost from current node to goal node
K (n) = Key Function – Stores the changes in the value of G (n)
n_s = Start node
n_g = Goal node
n_C = Checkpoint node
x_n = x co-ordinate/position of the current node

1 FLGOFF, School of Engineering & Information Technology. ZEIT4500
\[ y_c \quad x_{\text{goal}} \quad y_{\text{goal}} \quad n \quad m = \]

- y co-ordinate/position of the current node
- x co-ordinate/position of the goal node
- y co-ordinate/position of the goal node
- number of vehicles
- number of checkpoints/secondary goal

## I. Introduction

### A. Background

In real time environment, the vehicles have to operate in dynamic, uncertain and changing circumstances. Therefore, it is very impractical to plan a path prior to the mission [1]. Consequently, our planning system must be capable of re-planning in real time to adapt to the changing environments. This concept transforms the vehicles from passive command followers to active decision makers. The question under consideration is: How can a team of multiple vehicles be programmed to complete a set of tasks with a number of checkpoints? Another important consideration is the dynamic nature of obstacles which are to be avoided. The problems to be addressed are the choice of path, expanding and generating paths and assignment of tasks to each vehicle. The planning system must be able to reallocate the paths of the vehicles if an obstacle is sensed at any time during the mission execution. Each vehicle will have a separate controller and the team as a whole is controlled by decentralized approach where each vehicle can choose its path independently after collaborating with the group. At the same time, the vehicles will also be able to communicate among themselves about the changing environments and the dynamic nature of environments. This aspect of the problem is inspired by multi-robot mapping of an unknown or partially known environment, where the robots need to visit certain points in the environment (the tasks) to explore/map those regions, and also communicate their respective findings with each other in order to build a global map of the environment in a distributed fashion [2], [3].

The first part of this report describes the motivation behind the project and the aim of the work carried out. Some of the literature review is described in section II, followed by the problem description in section III and the assumptions in section IV. Section V describes the methodology of the simulations. Computational results are then analyzed in section VI and some concluding remarks are made at the end.

### B. Context and Motivation

A lot of research work and human effort has been put into single vehicle path planning, but the purpose of this work is to extend the concept of single vehicle path planning to multi-vehicle case. In this scenario, the total area to be explored is split among multiple vehicles in order to reduce the work load of each vehicle. This saves each vehicle from being over-worked and exhausted. In the practical application of such solutions to the problem, the computational planning and execution time and the distance covered by all the vehicles remains the main constraint. Therefore, this project identifies the factors affecting the path planning and re-planning time of the vehicle team while travelling from one node to another via the shortest possible path. The concept of multi-vehicle team to explore the area can be used by under water and aerial vehicles for both military and civilian purpose. It can be used for information gathering of an unknown terrain and military reconnaissance missions under water, on ground and aerially. The increasing use of the autonomous robots with artificial intelligence makes this kind of problem even better suited for missions where a lot of man power is not available to control the vehicles, so the decisions about the area exploration can still be made by the team itself. To have a fully autonomous vehicle team, the vehicles must be able to successfully plan their path from any given location to a goal point or a safe zone. The solution also incorporates the suitability of the use of such multi-vehicle team in dynamically changing environment.

### C. Aim

The main aim of this project is to devise a path planning package for a multi-vehicle team. The vehicles must be able to successfully plan their path from any start point to the end point while avoiding the static and dynamic obstacles in the path. These vehicles must also be able to explore certain specified checkpoints in the given area grid. The aim is to find the least cost and shortest possible path between the two points and reduce the total computational time for this path planning. The shortest path is found by using the most appropriate path planning algorithm, whereas the computational time will be reduced by optimizing the factors which affect this path planning time. The planning system must be able to reallocate the paths of the vehicles if any obstacle is sensed to be in the path of the vehicles at any time during the mission execution.
II. Literature Review

Several path planning techniques are available for single autonomous vehicle to find the shortest and quickest path in the given area grid. Some of the graph and heuristic path finding techniques include A* [4], D* [5] and D* Lite [6] algorithms. Some other related path planning techniques for multi-vehicle planning include particle swarm optimization, ants’ algorithm and multiple travelling salesman optimization techniques.

In graph based search, the environment is discretized and represented by a graph. If a node $s_i$ is connected to a node $s_j$ by a directed arc from $s_i$ to $s_j$, node $s_j$ is said to be a successor or child of node $s_i$ and node $s_i$ is said to be a predecessor or parent of node $s_j$. When talking about goal oriented vehicle navigation, the Euclidean distance to the goal is the task-dependent information that is the main focus of the search [7].

A*, D* and D* Lite were chosen because of their very basic approach and ease of availability. These algorithms are the modifications of Dijkstra’s algorithm [8] which forms the basis of single vehicle path-planning. When incorporating multiple vehicles for mission execution, the planning problem can be complex and the time available is also limited. In such situations, the vehicles must be provided with the best solutions that can be generated within the time limits. A useful set of algorithms for generating such solutions are known as anytime algorithms.

Path planning for multiple vehicles can be classified into local-planning and global planning. Local planning navigates the vehicles step by step as they proceed towards the secondary goals (i.e., online planning). In global planning, every vehicle plans the entire path before moving towards the goal (i.e., offline planning). Due to the dynamic nature of the environment, our problem will be dealing with online planning.

The exploration of checkpoints by any of the vehicles makes it similar to Multiple Traveling Salesman problem. In this technique there are ‘N’ robots each starting at different locations, ‘M’ goals which must be visited by some robot, and one Base to which all robots must eventually return. The Mission is optimized with respect to getting all robots back to the Base/Goal as quickly as possible [9]. The problem under consideration differs slightly from the multiple travelling salesman approach due to the presence of static as well as dynamic obstacles which change the environment of operation.

We will initially suppose that we have ‘n’ number of vehicles $n_1, \ldots, n_n$ and ‘m’ number of secondary goals/checkpoints $m_1, \ldots, m_m$ to be explored where $m \geq n$. The vehicle team operates in a two dimensional environment populated by static and dynamic obstacles. Each vehicle $n_i$ is meant to cover $m_i$ checkpoints during the execution of the operation. This project will look into the decision making of the vehicles to travel to the checkpoints based on the minimum distance constraints.

III. Problem Definition

Fig. 1 describes the problem. Initially, vehicle 1, 2, 3 and 4 move to the checkpoints 1, 2, 3 and 4 respectively (checkpoint closest to each vehicle) while avoiding the obstacles. This is where the decision about the next path comes into consideration. Checkpoint 6 will be explored by vehicle 3 or 4 depending upon the nature and dynamics of the obstacles 5, 8 and 9. Similarly, checkpoint 5 will be explored by vehicle 1 or 2 depending upon obstacle dynamics. The path will be chosen such that at least one vehicle explores each checkpoint while trying to limit the total time and total distance covered by the team as a whole.

![Figure 1. Problem Definition](image)

IV. Constraints and Assumptions

It is assumed that all the vehicles will be able to successfully communicate with other vehicles and update the changes in the position of the obstacles. All the sensors work properly and convey the position of the obstacles successfully and accurately. None of the hardware restrictions have been considered in this analysis.
To allow different analysis configurations and have scenarios that are close to real environment, the start position of each vehicle, the goal position, the checkpoints to be traversed and the position of static and dynamic obstacles can be selected as desired. Also, the grid size can be changed and set according to the requirement. The implications of a bigger and higher resolution area grid will be explored along with the increase in number of dynamic obstacles. The velocity of the dynamic obstacles remains the same during the path planning. These obstacles move at a constant velocity throughout the runtime of the simulation. Same is true for the direction of the dynamic obstacle. The velocity of all the vehicles also remains constant during the runtime.

The team is controlled by decentralised approach where each vehicle can choose its path independently. There is no particular leader and the decision making authority is distributed within the group. These decisions are based on the least cost calculations. In this decentralized approach, each vehicle runs its own online algorithm to find the least cost path. At the same time, the vehicles will also be able to collaborate among themselves about the changing environments.

V. Methodology and analysis

A. Working of Algorithms

A* algorithm uses a mechanism in which the vehicle can explore alternate routes once it reaches a dead end in order to avoid traversing over the obstacles. This is done by maintaining two lists “OPEN” and “CLOSED”. The list “OPEN” stores all successive paths that are yet to be explored while list “CLOSED” stores all paths that have been explored. The list “OPEN” also stores the parent node of each node. This is used at the end to trace the path from the Goal to the Start position, thus generating the optimal route [10]. All the obstacles are in the “CLOSED” list. The start node nS is the first item on the “OPEN” list, the algorithm adds all the traversable adjacent nodes to this list. Each of the adjacent nodes will have the start node as their parent node. The node ‘n’ to be traversed next is selected using Eq. (1) which gives the node with the lowest cost:

\[ F(n) = G(n) + H(n) \]  

where \( G(n) \) = the actual movement cost from the starting node ‘nS’ to the given node ‘n’. 
and \( H(n) \) = the estimated heuristic cost to traverse from the given node ‘n’ to the goal node ‘ng’. 

The node with the lowest \( F(n) \) score is chosen. Now we add this square to the closed list and check the score of the adjacent squares. If the adjacent square is already on the open list, check the score to see if traversing to this next node costs less if we go through our present node. If this new path costs less, we change the parent node of this adjacent node to our current node [4]. This process is carried out till the goal node is reached.

D* produces an initial plan based on known and assumed information, and then incrementally repairs the plan as new information is discovered about the world [11]. For the dynamic environments, the information has to be updated through sensors. This update in the information about the dynamic obstacles and environment increases the need for a re-plan. Dynamic A*[5], continually improves its solution while deliberation time allows, and corrects its solution when updated information is received [12]. The execution of the D* algorithm can be divided into initial planning and re-planning phases. Initial planning is performed if the robot is standstill at the start position and re-planning is performed if the robot detects nodes with changed occupancy values during its motion. The number of expanded nodes is kept to a minimum and consequently the time of execution [6], [13].

D* Lite is an adaptation of Koenig's Lifelong Planning A* (LPA*) [14], which in turn is an incremental version of A* search. LPA* is based on the assumption that there are only small changes.

It maintains two estimates of the start distance of each node:

- \( G*(n) \) which is the same as \( G(n) \) value in A* i.e., the actual cost of moving from a start node ‘s’ to a given node ‘n’.
- \( rhs \) values, which are one step look-ahead values. These values are based on the \( G*(n) \) and are estimates of minimal distance from the given node ‘n’. These values are better informed than the \( G*(n) \) values. Their name comes from Dynamic SWSF-FP where they are the values of the right-hand side (rhs) of grammar rules [14].

D* Lite produces an initial plan based on known and assumed information, and then incrementally repairs the plan as new information is discovered about the world. D* Lite has shown to be up to two orders of magnitude more efficient than planning from scratch with A*[15]. D* Lite expands fewer nodes than regular A* and hence is more efficient [16]. When working with D* lite, role of the \( G*(n) \) function must be clarified. It lets one choose which node to expand next based not only on how good the node itself looks but also on how good the path is thus far [17].
Firstly, all the above mentioned algorithms were compared to select the most time efficient algorithm. The most efficient algorithm i.e., D* Lite was used to run a simulation for single vehicle path planning having only static obstacles in the environment. The next step was the addition of dynamic obstacles. Some checkpoints were added to the environment and the vehicle had to travel to all these checkpoints/secondary goals before ending the path at the final goal. The next step was the addition of more vehicles. Addition of vehicles initiated the need for a decision to be made about which vehicle will explore which of the checkpoints in order to reduce the total planning time and the total distance. Having multiple vehicles in the scenario also calls for the need of collision avoidance among the vehicles themselves. Fig. 2 shows the sequence followed during the problem solution.

![Figure 2. Methodology Flow Chart](image)

**B. Comparison of Algorithms**

Initially, all the above mentioned algorithms were compared to check the most time efficient algorithm. A simulation was run for the path planning of single vehicle having several static obstacles in between the start and goal node. The position of the obstacles, start and goal node is kept same for the both algorithms. The simulation was run three times and the computational path planning time shown is the average of the results. The results of this comparison are represented in Fig. 3. It can also be noticed in the figure that A* makes its way from the start to goal while the D* Lite algorithm starts its path planning from the goal node and makes its way back to the start node. D* Lite took considerably less time to reach the target node successfully while avoiding all obstacles Therefore, D* Lite algorithm was used for all the further cost and distance calculations.

![Figure 3. Comparison of A* and D* Lite Algorithm](image)
C. Dynamic Obstacles

The initial simulation was run in an environment with static obstacles only. The dynamic obstacles were later added to show a realistic and dynamic environment. The dynamic obstacles have a constant velocity and direction which can be changed to achieve different configurations. This shows a realistic environment and adds another challenge for the vehicles to adapt to the environment. The program also warns about possible collisions that might occur with the dynamic obstacle if the path is not re-planned. It then replans the path to go around the obstacle.

The coding was initially done with one dynamic vehicle. The increase in the number of dynamic obstacles will have the largest influence on the runtime and path replanning time of the vehicles. This is because the code checks for possibility of the intersection of the vehicle’s path with the dynamic obstacle’s path. This means that every node occupied by the path of the vehicle is checked against every node occupied by the path of the dynamic obstacle. Increasing the number of dynamic obstacles increases the total number of nodes occupied by the paths of the obstacles. This increases the computational time exponentially. The position of the dynamic obstacles relative to the position of the vehicles and the static obstacles also influences the planned path and the replan time.

D. Introduction of Checkpoints

The next step was the addition of checkpoints during the vehicle’s traversal from start node to the goal node. The single vehicle was programmed to go through all the checkpoints before finally reaching the goal. The planned path is shown in Fig. 4. The number and position of these checkpoints can be altered to have different configurations. Increasing the number of checkpoints increases the computational effort because the vehicle has to redo all its calculations once it reaches a checkpoint and decides to go to the next checkpoint. This is because the secondary goal node changes after traversal to each checkpoint. The computational path planning time is also affected by the number and position of the checkpoints relative to the position of obstacles.

![Figure 4. Planned Path for Single Vehicle with some Checkpoints](image)

E. Multi-Vehicle Path Planning

In order to achieve the goal of multi-vehicle path planning, the number of vehicles was increased. When the second vehicle (with a different start node but the same goal node) was added to the simulation, the initial path planning technique from start to the goal remains same. After the addition of a second vehicle, a decision needs to be made about which of these two vehicles will explore which of the checkpoints in order to reduce the total time of the task execution while travelling the least possible distance. This decision was made based on the cost calculation (the cost of traversal being calculated by the D* Lite algorithm).

In order to travel to a checkpoint at a node $n_c$, the decision has to be made about which one of the two vehicles will go to this checkpoint. The nearest checkpoint is traversed by using Eq. (2) which gives the least cost path to travel from the start node to this checkpoint:

$$ C(d) = C_c(d) + C_d(d) $$

where $C_i(d) = \text{cost of travelling from the start node } n_s \text{ to the goal node } n_g \text{ on a path } P_i$

$C_c(d) = \text{cost of travelling from the start node } n_s \text{ to the checkpoint } n_c \text{ under consideration}$
\[ C_d(d) = \text{cost of deviation from the path } P_i \text{, i.e., the extra cost incurred if the vehicle deviates to the} \]
\[ \text{checkpoint before reaching the goal node.} \]

Both vehicles calculate this cost \( C(d) \) and the vehicle having the lowest value travels to this checkpoint while the other vehicle moves to the second nearest checkpoint. This process is repeated after the traversal to every checkpoint till both the vehicles reach the final goal node. This means that a vehicle might not go to the checkpoint nearest to it, but this ensures that the total distance travelled by all the vehicles is minimized. Once the checkpoint is explored by either of the vehicles, it is then put on the CLOSED list in order to avoid other vehicles to travel to this checkpoint again. The planned path with two vehicles is shown in Fig. 6.

![Figure 6. Planned Path for two vehicles with one dynamic obstacle](image)

The number of vehicles was increased from two to three and then to four. Most of the time-analysis was performed with four vehicles. The number of vehicles can be increased even further using same methodology but the increase in the path planning time was observed to be affected by the same variable parameters (e.g., number of obstacles and area grid). Thus, no further increase in the number of vehicles was deemed necessary. The planned path for four vehicles with one dynamic obstacle in the environment is shown in Fig. 7.

When there is more than one vehicle in the given area, there is also a possibility of vehicles colliding with other vehicles. To avoid these collisions, the path of each vehicle is considered to be a dynamic obstacle by all the other vehicles. Therefore, every vehicle also checks the path of all other vehicles for the possibility of collision. The velocity of all the vehicles also remains constant throughout the simulation but it can be changed as required.

![Figure 7. Planned Path for four vehicles with one dynamic obstacle](image)
VI. Discussion of Results

The numerical simulations were run for different configurations. The variable parameters during the path planning/re-planning analysis were the number of vehicles, number of dynamic obstacles and area grid size. The results show that the computational path planning time depends significantly on the area grid size. All the simulations were run three times for each configuration. The computational times shown in the results are averaged over these three runs. These simulations were run on the UNSW Lab PC with 8GB memory, 8 MB cache and clock speed of 2.7 GHz.

The total distance travelled by all the vehicles and the path planning time is also affected by the number of vehicles, the number and position of checkpoints, the allocation of checkpoints, the number and position of static obstacles and the start and goal state of the vehicles. This is because the start and end point and the position and number of static vehicles initially dictate the total distance to be travelled by all the vehicles. But these variables do not affect the time as much as the position and number of the dynamic obstacles and the area grid size. When analyzing the results for the below mentioned configurations, the position and number of the static obstacles, the start and goal node were kept the same for each run of each configuration in order to have consistency in results. The effects of change in the number and position of static obstacles for each configuration were studied separately.

The computational Path Planning and re-planning time for different number of vehicles and one dynamic obstacle in a 50*50 area grid are shown in Table 1.

<table>
<thead>
<tr>
<th>TABLE 1: 50*50 Area Grid – 1 Dynamic Obstacle</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of Vehicles</td>
</tr>
<tr>
<td>-----------------</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
</tbody>
</table>

The computational Path Planning and re-planning time for different number of vehicles and two dynamic obstacles in a 50*50 area grid are shown in Table 2.

<table>
<thead>
<tr>
<th>TABLE 2: 50*50 Area Grid – 2 Dynamic Obstacles</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of Vehicles</td>
</tr>
<tr>
<td>-----------------</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
</tbody>
</table>

The computational Path Planning and re-planning time for different number of vehicles and one dynamic obstacle in a 500*500 area grid are shown in Table 3.

<table>
<thead>
<tr>
<th>TABLE 3: 500*500 Area Grid – 1 Dynamic Obstacle</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of Vehicles</td>
</tr>
<tr>
<td>-----------------</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
</tbody>
</table>
The computational Path Planning and re-planning time for different number of vehicles and two dynamic obstacles in a 500*500 area grid are shown in Table 4.

### TABLE 4: 500*500 Area Grid – 2 Dynamic Obstacles

<table>
<thead>
<tr>
<th>No. of Vehicles</th>
<th>Initial Planning Time (Without dynamic obstacle)</th>
<th>Total Time after Dynamic Obstacle Avoidance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.2123 s</td>
<td>14.6895 s</td>
</tr>
<tr>
<td>2</td>
<td>0.3643 s</td>
<td>32.1030 s</td>
</tr>
<tr>
<td>3</td>
<td>0.5252 s</td>
<td>51.5066 s</td>
</tr>
<tr>
<td>4</td>
<td>0.8453 s</td>
<td>129.2705 s</td>
</tr>
</tbody>
</table>

The second column shows the initial path planning time from the start point to the goal node while going through all the checkpoints. The third column shows the time after the path has been checked for the possibility of intersection with the dynamic obstacles or other vehicles. The path has been re-planned in order to avoid any collisions that are expected. The time for this re-plan is significantly larger than the initial planning time because during this time the vehicles check each and every node on their path against every node on the path of dynamic obstacles to check for the possibility of any collisions. The planned path for four vehicles with two dynamic obstacles is shown in Fig. 9.

It was observed that increasing the number of vehicles, number of dynamic obstacles or the area grid size increases the total computational re-plan time. This is because the increase in any of the above mentioned factors increases the number of nodes to be checked for the possibility of collision which consequently affects the total computational time.

The increase in the computational time due to the increase in the number of obstacles and increase in the area grid size can be visualized in Fig. 8. The top two histograms show the comparison of path planning time with one and two obstacles. Bottom two histograms show the comparison of path planning time in a 50*50 grid area and 500*500 grid area resolutions.

![Figure 8. Computational Path Planning Time for different Configurations](image-url)
VII. Conclusions

The algorithm plans the shortest path from start to goal node while exploring all the checkpoints in the way. The static and dynamic obstacles in the environment are also avoided. D* lite algorithm is used to plan the path for each vehicle individually. The code compares the path of each vehicle to travel to certain checkpoints and then decision is made based on the traversal cost calculated by D* Lite to travel to this checkpoint.

The path planning and re-planning time is mainly influenced by the grid area and the number and position of dynamic obstacles. More number of dynamic obstacles and a bigger area will take more time to find out the possibility of collision and hence the re-planning time will be more as compared to the case of less number of dynamic obstacles and a smaller area. Path planning time is also affected slightly by the number and position of the static obstacles. But this difference in time is not as significant as that of the position and number of the dynamic obstacles. The number and position of the checkpoints to be travelled dictates the total distance travelled by all the vehicles.

VIII. Future Prospects

This computational time for path re-plan can be decreased in further studies by limiting the grid points to be scanned for the possibility of collision. In this particular case, the program checks for the possibility of collision at each and every node of the planned vehicle path. This takes a lot of time. This computational time can be reduced by limiting the number of nodes to be checked by using the distance sensors and checking for the possibility of collision only when the dynamic obstacle is within a specified area range. There is some work being done on optic flow to calculate this distance by the sensors.

The vehicles in these simulations move with a constant velocity. Vehicles can be given variable velocity. This will save the time of traversing around the dynamic obstacles and other vehicles to avoid collision. The vehicles having variable velocity can slow down or speed up as required to avoid the collision. This will also save traversal cost of moving around the obstacle.

During this work all the static obstacles were considered to be present at just one node. Some percentage of tolerance can be built into this by putting some specific area around the obstacle to be out of bounds for the vehicles. This will make the vehicles re-plan their path slightly before they reach the obstacle itself.

Optimization of the number of vehicles for a certain grid size needs to be carried out as well. The number of vehicles can be decided based on the selected area and number of checkpoints in order to maximize the efficiency of all the vehicles and reduce the total distance travelled and fuel consumed by all the vehicles.
Acknowledgements
I would like to acknowledge and thank my supervisor Dr. Sreenatha Anavatti for his guidance and support throughout this project. I would also like to acknowledge Sumana Biswas and Sober Francis for their help during the initial project phases.

References