Implementation of Multi-robot Heuristic Goods Transportation

*Z. Yan, N. Jouandeau and A. Ali-Chérif, "Multi-robot heuristic goods transportation," 2012 6th IEEE International Conference Intelligent Systems, Sofia, Bulgaria, 2012, pp. 409-414, doi: 10.1109/IS.2012.6335251.

1st Yashas Shetty
University of Maryland, College Park
Maryland, Unites States of America
yshetty@umd.edu

***Abstract*—The aim of the project is to understand the multiple goods transportation problem and implement a unique heuristic approach based strategy to solve the problem. The strategy not only leads to minimum run time but also low cost consumption. As the authors have initially given a background on multi goods transportation problems, it gives a good understanding of the problem and helps to understand why the technique mentioned is novel from the approaches discussed earlier. The algorithm is to be implemented on a real world multi robot system.**

I. Methods to implement

The author states the use of a heuristic function based algorithm. Prior to that using the Greedy algorithm, it has been discussed where the run time based algorithm fails. The methods for both of them are discussed below.

Heuristic value calculation.
The heuristic value h(n) of the production rate is considered in this strategy. It is obtained from the distribution function. But as distribution is uniform, the heuristic value will be the average production rate

The Greedy algorithm
Greedy algorithm is used for calculating the time , when the n+1th goods were carried by a robot

$$X(n+1) = X(n) + min(f(t, h(n), r(n)))$$

Main algorithm is as follows:
Calculate h(n) for all robots for every robot outside depot and without a task, if the robot is close to the source, we allocate it a task
And in case there is a robot in the depot without a task, we wait for a certain time (which is the difference between time required from depot to source and h(n)) and then check for any tasks to assign to the robot.
And this continues until all tasks are done

The reason for using the heuristic algorithm is discussed in a further section.

II. High-Level Motivation

A. Usefulness

A main challenge in the multi robot systems is the coordination of the tasks or task allocation.Achieving this itself makes the system sophisticated at times. The study deals not only with this but also with the transportation of goods problem. This makes the total system more efficient in terms of completion time.

B. Applications

An efficient multi robot system will have applications in Warehouses are a very prime example where multi robot systems can increase efficiency of the unit by a large margin. Hospitals, hotels/restaurants also have a similar use case making multi-robot systems very advantageous there too.
Rescue operations (considering that rescue locations are known) can be benefited a lot if they have multiple agents at hand.

III. Methodology

Firstly I intend to implement the study in a ROS Based Gazebo environment simulation using turtlebots.
Then if successful, I would like to implement the same thing on real turtlebots. The alternative to this could also be implementing it on the Robotarium platform.
If the above is accomplished, I would try to implement it on Kilobots. Implementing on Kilobots can be very hard as we don't have peripherals like GPS on them.
Software wise, Gazebo, Robotarium, KiloGui would be sufficient and the programming languagues would be either C or python. It is considered that Linux would the OS required throughout.

IV. Related Work

The paper focusses on the "Multiple robots transport multiple objects cooperatively" problem. Going through the related works section of the paper gives us a good idea of the backgrounds upon which the formulated algorithm is based.

The basic idea for the autonomous mutli robot system is based on that of the MARTHA project [4] which states that "whenever a robot produces a plan which makes use of some kind of shared resources (cells,·trajectories in areas), it advertises it, and collects from other robots the plans which specify how they plan to use these resources, as , well as the right (i.e., the necessary exclusive tokens) to perform its plan coordination". Hence we can infer that the path taken by a robot takes information based on the activities of other robots and the information gathered from them.

Another strategy the paper is uses is based upon the trail following algorithm of ants. This strategy is validated in [5]. A unique element of the paper is the use of energy consumption as a an additional performance metric in the system. For getting the path, map of environment SLAM would be necessary. There are multiple approaches [6][7] for implementing SLAM. We also need to make sure that robots take other robots into account to avoid collision and congestion [8].

V. Heurestic Planning Strategy

The authors state that a major reason for the low efficiency in the transportation is the unknown production rate. The production rate, denoted by Tn, refers to the elapsed time between the disappearance of the n-th goods (be carried away) and the appearance of the (n + 1)-th goods. Thus, to inculcate this parameter into the strategy, a heuristic function dependant on the production rate is introduced.

$$h(n) = avg(Ti)...i = 0, 1, ..n$$

As mentioned above we can use the greedy algorithm to get the minimum run-time, however it does not guarantee minimum energy consumption. Thus the proposed algorithm in the section I is used for task allocation.

Also the above algorithm is a higher level algorithm and does not include algorithms like of those used in the global path planning and local obstacle avoidance. It is mentioned section V of the paper that the wavefront propagation algorithm [15] is used for global path planning and the nearness diagram algorithm [16] is used for goal seeking and local obstacle avoidance.Improvements upon these can be possible and yet to be looked upon.

A. Difference between time efficient and energy efficient algorithm

Intuitively, we think that energy consumption is directly proportional to the time consumed. And hence an algorithm focused on least time will be assumed to be the best. Consider the below path map mentioned in the paper itself for example.

We have 2 robots, robot 0(R0) and robot 1(R1), a depot(D) and a source(S). We can have the sink(K) at any arbitrary point. Now consider that we have received a good pickup message. We will have to send one of the robots to the source then to sink and the other robot will go to the depot. There will be 2 possible outcomes situations.

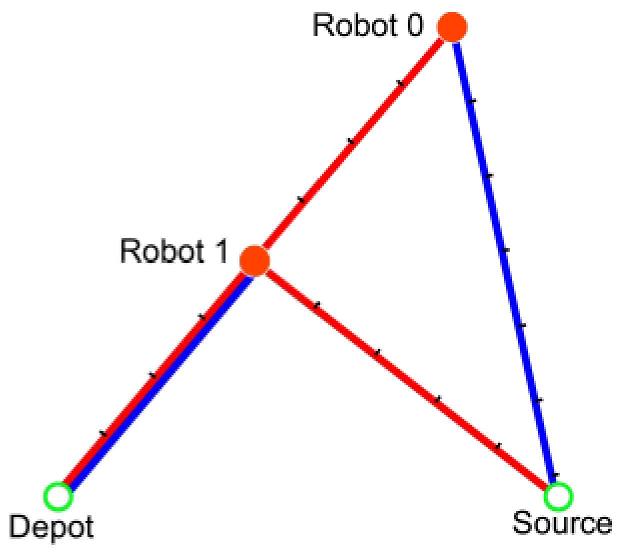

Fig. 1. omparison of two plans for task allocation

1) R0-D, R1-S-K
2) R1-D, R0-S-K

We can see that the "S-K" is common for both paths. Hence we will ignore that, making our new paths

1) R0-D, R1-S (denoted by red)
2) R1-D, R0-S (denoted by blue)

As this is a prolem of goods transportation, the path which will transport goods quickest will be considered to be the fastest algorithm. Thus, the greedy algorithm gives out the red path as the solution. This is because R1 is closer to source and hence it will need lesser time to reach K, i.e. the sink. However our algorithm will give us the blue path. And when we compute the total energy consumed by the system, it will be as follows,

$$Energy\ \alpha\ Pathlength$$
$$LetEnergy = k * Path$$
$$Pathlengthofredpath = R0R1 + R1D + R1S$$
$$Pathlengthofbluepath = R1D + R0S$$
Comparing the 2 paths, we can see that
$$R0R1 + R1S > R0S$$

...(triangle inequality)

Thus as the path length of red path is higher, the energy consumption of this path will be higher than the blue path. And hence the greedy algorithm has given us an energy inefficient algorithm as compared to our algorithm. Thus we can now see that a low time consumption does not necessarily imply energy efficient.

B. How does Heuristic Planning help

A question one might ask is how does this heuristic optimize the energy and time. The reason behind this is that authors have introduced a heuristic, which will takes into account the idle time of the vehicle. In the given formula of heuristic,

$$h(n) = avg(Ti)...i = 0, 1, ..n$$

Where T_i is the time between the disappearance of n-th good and appearance of n+1th good. This means it is the amount of time a robot moves without carrying any good.

Now consider our above example, all the paths shown in the map are of the idle case since both the robots are either going to the depot or to the source meaning they do not have any goods on them. And for these paths, as we have seen earlier, the blue path will need less time to traverse as well. Thus by using the idle time (mentioned as production rate in the paper), we have come up with a stratergy that minimizes the energy of the system too.

VI. Formal Problem Statement

The problem being solved in the paper is getting a task allocation method which is minimum on run time and low on energy consumption. For this, the authors have proposed a heuristic function (based on the productivity rate) for an n-th item instead of the time elapsed when n-th items(goods) are have been carried away.

In mathematical form, we can write as follows

Use a heuristic function instead of using the run-time of a n-th item to allocate tasks in a multi robot goods transportation system in order to achieve the minimize the energy consumption and run-time of the system

Problem : To find the appropriate task allocation so as to get the minimum of both, energy consumption and time consumption

Given :

n - number of robots

h(n) - heuristic function

t - time set, this set contains the time required by various tasks, like $T_{DepotToSource}, T_{toSource}, T_{SourceToSink}$, etc. and also the T_{Wait}.

We define

$$\Sigma_E = Total\ energy\ consumed$$

$$\Sigma_E = f(n, h(n), t)$$

$$\Sigma_T = Total\ time\ passed$$

,

$$\Sigma_T = g(n, h(n), t)$$

To find: Our aim is to minimize the sum of the above two , i.e.

$$min(\Sigma_T + \Sigma_E)$$

Simulation

For simulating, I am trying to implement the algorithm on the Robotarium simulator. But creating desired maps and doing a real life implementation can be difficult. Hence in parallel, I am simulating multiple turtlebots in Gazebo.

As given in the experiments section of the paper, I will be first implementing the planning algorithm for each robot. The wavefront propagation algorithm[11] is being used and will be replicated by me. Also the global path planning and the nearness diagram algorithm [12] is to be implemented for for goal seeking and local obstacle avoidance.

After this is done, the main task allocation algorithm, as stated by the author will be implemented. This algorithm will be tested for various locations of Depot, Source and Sink.

VII. Attempts in simulation

For simulation, ROS simulations using 2 different planning algorithms were tried out unsuccessfully. The purpose behind implementing a planning algorithm was that to get optimum energy and time consumption, it is elementary to have a the optimum path known beforehand. And as the aim was to try out the heuristic approach in a map with obstacles, optimum path would needed to be calculated first. However these 2 algorithms failed and the details about them are described below.

A. ROS Simulations

The initial aim as stated above was to simulate the algorithm in the Gazebo environment using ROS (Robot Operating System) packages on the turtlebot3 .

Below is the map design I had planned for impelementing in Gazebo.

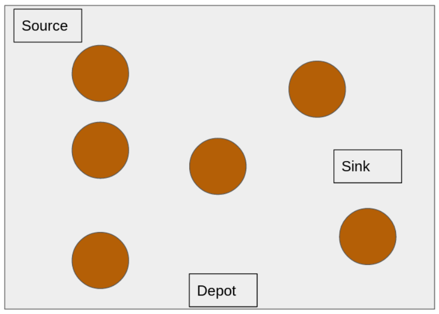

Fig. 2. Map drawing

I also designed a map in Gazebo world to simulate the above diagram.

B. Trial 1 - Real Time Planning

At first I tried to go with a real-time planning algorithm for each robot. The idea behind this approach was that a particular robot may have to change its path while traversing in a certain direction. In our case it is when a robot is returning from the

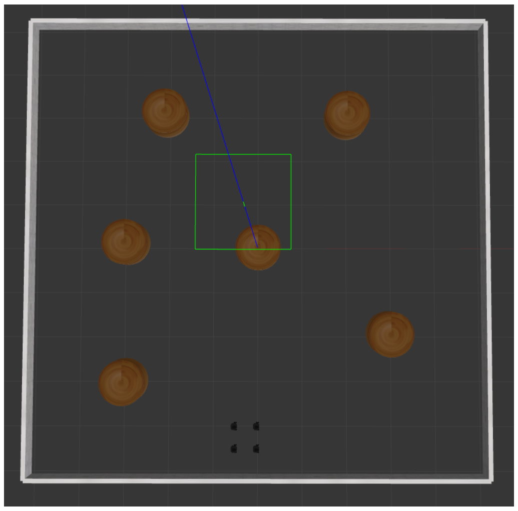

Fig. 3. Map in Gazebo

sink to the depot and it has been assigned with delivery of a new good and hence it has to change its path to go towards the source now. Hence I decided to implement a real time path planning algorithm. The RT RRT* algorithm introduced by Naderi et al[12] was tried out. Below is an example of the trees generated by the algorithm.

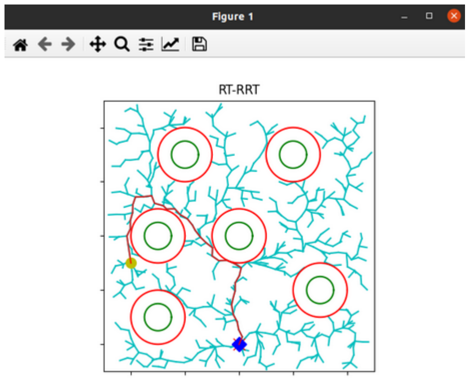

Fig. 4. RT-RRT* Plot

As we can see in the above figure, the blue lines are the explored points in the tree and the red path is the path given by the algorithm. We can clearly see that the path given by the algorithm is not optical. And the reason behind this lies in the working of the algorithm.

The RT-RRT*, being a real-time algorithm, tries to find a ”feasible” path to the goal in a certain time step. After that time step elapses, if it has found a path, it will move in that direction, or else keep searching for a path. Even the searching is limited to a certain time period. If the algorithm does not find a path , it just gives up. And even when it does find a path, it is one of the ”feasible” paths rather than the optimum path.

Thus even though real time planning was implemented, as it is not optimal, I could not go ahead with approach.

C. Trial 2 - Simple goto method

As the previous method did not work, I decided to implement a simple go to point function which will direct the robot to a certain point. The idea behind this to implement the simplest version of the heuristic algorithm. That is, we will ask a robot to go from one certain point to another point in a straight line. An empty Gazebo world with 4 turtlebots was going to be used for the simulation.

But the implementation of this approach had a fundamental problem. The goto functions implemented ran in a while loop till the robot reached the goal point. Hence once this goto function is called, the robot would not stop till it reached the required point. And this is not acceptable as we have already stated the robot might need to change its path real time.

D. Trial 3 - Present method

If we take he approach of the previous method, and give the robot velocity in the direction of the path instead of commanding it to go to the goal, we can solve the previous problem. Giving velocity ensures that the robot moves in along a certain path. And changing the velocity direction ensures that the robot changes its path too. Thus, in the current approach, I have given velocity on a time step basis.For each time step, the required velocity will be checked and changed if required. As this update happens at every time step, it becomes a real time process.

This approach was tried out on python plot. Scatter points were used to show the robots and the source, sink points. A ”unique” Depot point was not defined as we cannot have multiple robots at a single point at the same time. instead, every robot’s initial position was considered to be its Depot point.

Algorithm 1 Heuristic task allocation based on an empirical model

```
1:  if the $n$-th goods has been carried away by a robot then
2:      Calculate $h(n)$ by using (2).
3:      for all robot $i$ which is outside of the depot and not in the task do
4:          if $h(n) < T_{toSource}(i)$ and $h(n) > \frac{1}{2}T_{toSource}(i)$ then
5:              Robot $i$ accepts the task and go to the source.
6:              break
7:          end if
8:      end for
9:      if there is no robot which has accepted the task and there is a robot in the depot then
10:         Calculate the waiting time for the robot in depot by using $T_{wait} = \max(0, h(n) - T_{DepotToSource})$.
11:         After $T_{wait}$ time, wake up the robot in depot and assign the task to it.
12:     end if
13: end if
```

VIII. Implementation of the algorithm

The authors have given their algorithm of heuristic based task allocation, as shown above . The above discussed planning was integrated with this algorithm to get the required results. As seen from the algorithm, for a robot not carrying any goods and not in depot as well, we compare the heuristic with the time to source i.e., the time needed by the robot to move to the source. Hence when doing this comparison, we are actually evaluating what is more wasteful to this robot, going to the source or going back to the depot and we send a different robot instead. Thus we have estimated the minimum loss option for our system and allocated tasks accordingly. If the robot does not find any robot outside the depot which can do its task, it will check if there is any robot in the depot area and if there is, it will wait for some specific wait time which again derived by the values of the heuristic $(h(n)T_{DepotToSource})$. As the heuristic is updated in every iteration, its value remains accurate over time as well.

Implementation in python

Below is a photo of the implementation in the python plot

Below is the plot when robots are moving

Below is the gitHub repository for the same https://github.com/YashasShetty/Multigoofs

Below is a google Drive link for the videos of the simulation https://drive.google.com/drive/folders/1$_l 90TxUbsy - DJv2jy9_m MWebgGJ7IgGI?usp = sharing$

Different methodologies

I think that the below studies mentioned in the references, can have techniques useful to improve upon the study of the paper.

Conclusion

Thus we have used the heuristic based on production rate to allocate tasks and achieve an optimal solution. Although the intuitive thought process is of going for best time based functions, it might not be the same for fuel consumption problems. A python plot using the algorithm has been implemented and we can see how robots move divert from their original trajectories to get the optimal result. ROS implementation of the algorithm was not successful, it is in the future scope of the project and would be interesting to view the results in a simulation. The same goes for hardware deployment. Another future scope of this implementation would be using a proper path planning algorithm as every path a robot works on is not a straight line

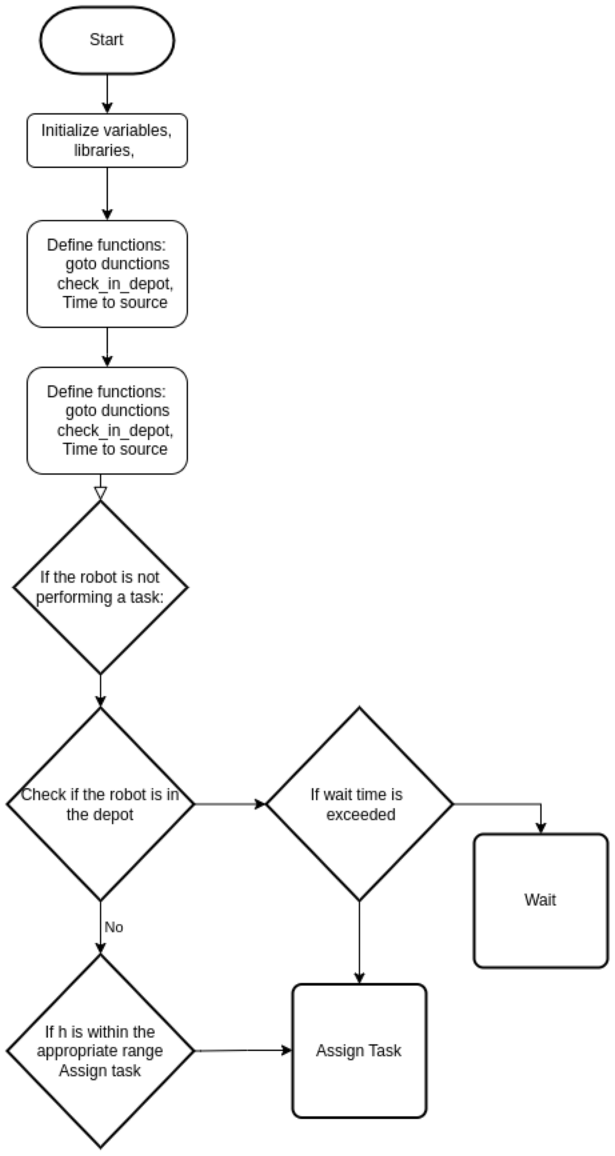
Start
Initialize variables, libraries,
Define functions: goto dunctions check_in_depot, Time to source
Define functions: goto dunctions check_in_depot, Time to source
If the robot is not performing a task:
Check if the robot is in the depot
If wait time is exceeded
Wait
No
If h is within the appropriate range Assign task
Assign Task

Fig. 5. Fowchart of implemeneted algorithm

References

[1] P. Pirjanian and M. Mataric, ”Multi-robot target acquisition using multiple objective behavior coordination,” Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065), San Francisco, CA, USA, 2000, pp. 2696-2702 vol.3, doi: 10.1109/ROBOT.2000.846435.

[2] Z. Xing, W. Wu, X. Wang and R. Hu, ”A Collision Avoidance Algorithm for Idle Robots in Multi-robot System,” 2021 IEEE International Conference on Networking, Sensing and Control (ICNSC), Xiamen, China, 2021, pp. 1-5, doi: 10.1109/ICNSC52481.2021.9702220.

[3] Chia-How Lin, Kai-Tai Song and G. T. Anderson, ”Agent-based robot control design for multi-robot cooperation,” 2005 IEEE International Conference on Systems, Man and Cybernetics, Waikoloa, HI, USA, 2005, pp. 542-547 Vol. 1, doi: 10.1109/ICSMC.2005.1571202.

[4] R. Alami, S. Fleury, M. Herrb, F. Ingrand, and F. Robert, “Multi-robot cooperation in the MARTHA project,” IEEE Robotics and Automation Magazine, vol. 5, no. 1, pp. 36–47, March 1998.

[5] R. T. Vaughan, K. Stoy, G. S. Sukhatme, and M. J. Mataric, “LOST: Localization-space trails for robot teams,” IEEE Transactions on Robotics and Automation, vol. 18, no. 5, pp. 796–812, October 2002.

[6] A. Huletski, D. Kartashov and K. Krinkin, ”Evaluation of the modern visual SLAM methods,” 2015 Artificial Intelligence and Natural Language and Information Extraction, Social Media and Web Search FRUCT Conference (AINL-ISMW FRUCT), St. Petersburg, Russia, 2015, pp. 19-25, doi: 10.1109/AINL-ISMW-FRUCT.2015.7382963.

[7] A. R. Khairuddin, M. S. Talib and H. Haron, ”Review on simultaneous localization and mapping (SLAM),” 2015 IEEE International Conference on Control System, Computing and Engineering (ICCSCE), Penang, Malaysia, 2015, pp. 85-90, doi: 10.1109/ICCSCE.2015.7482163.

[8] Z. Yan, N. Jouandeau, and A. Ali Cherif. “Sampling-based multi-robot

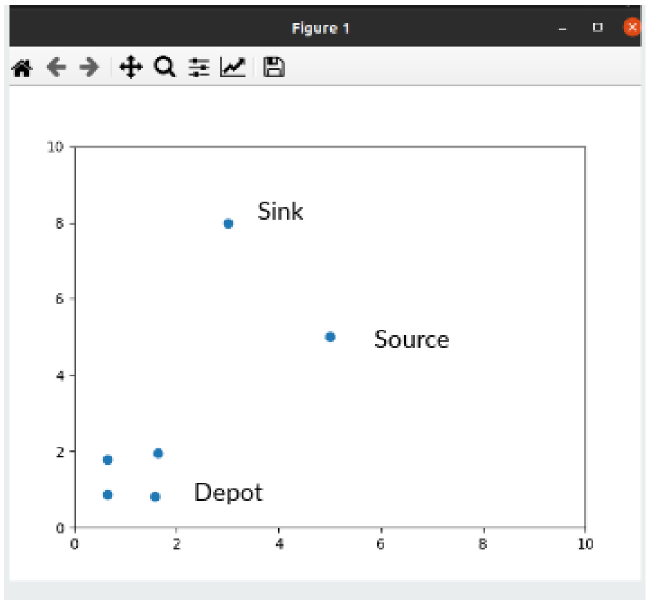

Fig. 6. Implementation in python

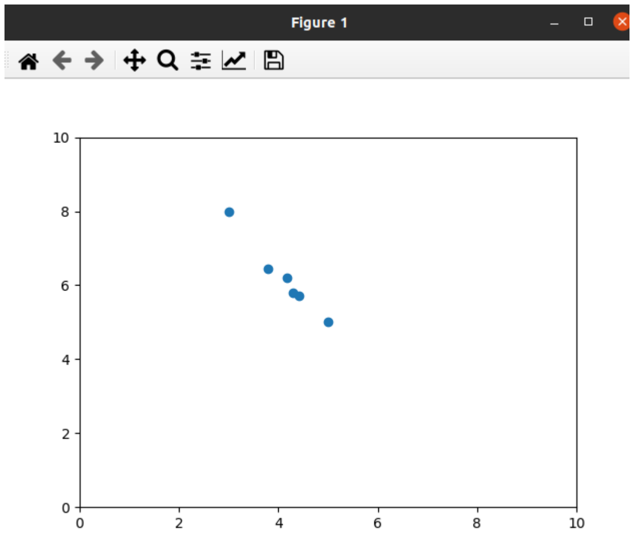

Fig. 7. Implementation in python moving bots

exploration," in Proceedings of the Joint 41st international Symposium on Robotics and 6th German Conference on Robotics (ISR/ROBOTIK 2010), Munich, Germany, June 2010, pp. 44–49.

[9] J.-C. Latombe, Robot Motion Planning. Kluwer Academic Publishers, 1991.

[10] J. Minguez and L. Montano, "Nearness diagram (ND) navigation: Collision avoidance in troublesome scenarios," IEEE Transactions on Robotics and Automation, vol. 20, no. 1, pp. 45–49, February 2004.

[11] J.-C. Latombe, Robot Motion Planning. Kluwer Academic Publishers, 1991.

[12] Naderi, Kourosh Rajamäki, Joose Hämäläinen, Perttu. (2015). RT-RRT*: a real-time path planning algorithm based on RRT*. 113-118. 10.1145/2822013.2822036.