Combined A* and RRT* for indoor navigation in dynamic environments

Akshata Pore
Robotics Engineering
Worcester Polytechnic Institute
aspore@wpi.edu

Kavit Nilesh Shah
Robotics Engineering
Worcester Polytechnic Institute
kshah@wpi.edu

Soumya Srilekha
Robotics Engineering
Worcester Polytechnic Institute
ssbalijepally@wpi.edu

***Abstract*—Path planning in dynamic environment is a challenging task as we have to consider time as the other dimension. Developing computationally efficient motion planners tailored for problems with unknown and dynamic obstacles is an interesting area. Combining strengths of local and global planners can solve this problem. Using A* as a global planner to find the optimal path from start to goal in a static environment would render a path within a given time frame. However considering the fact that this algorithm cannot be used for dynamic obstacles, we can use sampling based RRT* to sample points around the obstacle and connect the nodes before and after the obstacle assuming that the speed of the obstacle is slower as compared to computation time of RRT*. This paper combines A* and RRT* to find a computationally efficient path in an environment with dynamically moving objects in a shorter time.**

Index Terms*—local planner, global planner, A*, RRT

I. Introduction

Path planning for mobile robots is an interesting area to venture into wherein we plan out a trajectory or a path for a mobile robot to follow from the start position to the goal position. While planning a path for the mobile robots, we come across static as well as dynamic environments. We have multitude of algorithms in motion planning, but most of these deal with scenarios where the objects in the environment are static. Path planning in dynamic environment is equally important scenarios with which we deal practically are dynamic in nature. This dynamic nature of obstacles further add to the complexity of path planning. In static environments, once the optimal path is determined using path planning algorithm, the robot can start resort to the computed path to traverse from the starting position to the end position. But in environments where objects are dynamic, the planner needs to take into account the dynamic nature of objects and predict their future trajectories while planning a path [7].

Global planners are generally used for static environments since they require complete information of the environment before planning the path. Since the path given by global planners is rendered after taking environment into consideration, the path is optimal [8]. But these algorithms lack robustness and flexibility as they cannot handle variations in the environment as they are based on information provided by static environments. However local planners on the other hand render a path without prior knowledge of the environment. Local planners take into consideration the dynamic nature of the obstacles. Hence they are mostly employed for obstacle avoidance and finding a path in unknown environment. Though they are flexible and robust but the path rendered by local planners is not optimal [8].

In this paper, we aim to combine the strengths of the local planner and global planner and overcome their weaknesses to find an optimal path in a dynamic environment [8]. In this paper, we have used A* as a global planner and RRT* as a local planner. Using the strength of global planners to find the optimal path in finite time, we will use A* to plan out a trajectory connecting start and end positions. Whenever we encounter an obstacle which are dynamic in nature, we will use local planner RRT* because of its ability to give a quick initial path to the goal.The RRT* will create samples around the obstacle and plan out a path around the obstacle such that the start and end point of the path generated by RRT* lie on the global path.

II. Related Work

While there has been a significant amount of work done in developing novel algorithms and architecture for solving the motion planning problem [7], equal amounts of work is being done in devising strategies which combine the already existing algorithms for robust and effective planning in indoor environments. These strategies have been developed for different robotic systems such as UAVs, indoor robots, industrial manipulators, etc.

One of the strategies involves the use of Artificial Potential Fields (APF) to guide the sampling of new points for RRT* [4]. Given the map of the environment being already known, this algorithm computes the trajectory online and uses the potential fields to sample new points in the vicinity of obstacles. While this method greatly reduces sampling time of RRT* near obstacles by reducing the sample space and thus improving the convergence rate, it is not capable of performing in dynamic environments as the potential fields only include static objects.

Another work was based on using lazy implementation in a discretized work space [3]. It uses a global planner and a local planner where the former restricts the latter on a small region of its search space. It first finds a path on a coarse map and then refines it at higher grid resolutions. However, this algorithm fails in cluttered or dynamic environments.

In [2], a novel implementation is proposed which uses a coarse obstacle-free trajectory and then refines it locally as the UAV traces the path. This novel approach shows the effectiveness of using two-stage planner for solving the planning problem. It proved its significance for static, dynamic and even unknown obstacles.

Another line of work and state of art sampling based re-planning algorithm RRTX [5] being probabilistically complete, makes no distinction between local and global planning, yet reacting quickly enough for real-time high-speed navigation through unpredictably changing environments. RRTX achieves a quick reaction time in dynamic environments by maintaining an ϵ-consistent graph, constant neighborhood size at each node, and then using rewiring cascades to transfer information through the graph whenever obstacles change and hence can be applied to an environment where the obstacles unpredictably appear, disappear, or move.

A novel approach has been presented in [1] where authors have used RRT* as a global as well as a local planner for dynamic planning and quick re-planning under presence of a dynamic obstacle. The algorithm first computes a path from start to goal position using RRT* and the robot starts this path. If while following this path, the robot encounters a dynamic obstacle, it re-plans its path avoiding the obstacle and re-joins the global path at some other node closer to the goal. Though it works appropriately in real-life dynamic scenarios, it has one limitation where the robot has to a sub-optimal global path generated by the RRT*.

The remaining of the paper is organized as follows: Section III explains the problem statement, notations and assumptions. Methodology is described in Section IV. Section V gives details about the work done so far and Section VI illustrates the future plan of work.

III. Problem Statement

Let M denote the map in which the robot and the obstacles are present. Here for a 2D map O denotes the obstacles and the robot is a turtlebot denoted as T that can move in the map but must avoid the obstacles, For a 2D map $(M \epsilon R^2)$. The path planning problem can be summarized as: Given the initial placement of the turtlebot in an indoor map compute an optimal path using a global planner and using a local planner to avoid dynamic obstacles and quick re-planning.

A. Input

The placement of the turtlebot, the geometric description of obstacles and the map, location of the desired goal.

B. Output

Computation time of the algorithm, a detailed description of the path from the start to the goal.

C. Assumptions

The following assumptions have been made:

- A discretized map of the space is already obtained.
- The speed of the obstacles is less than the computation speed of RRT*.
- Global path is the best possible path, and thus, even after avoiding the obstacle, the robot again starts following the global path.

D. Objectives

We aim to achieve certain objectives throughout the course of the project which are listed below as follows:

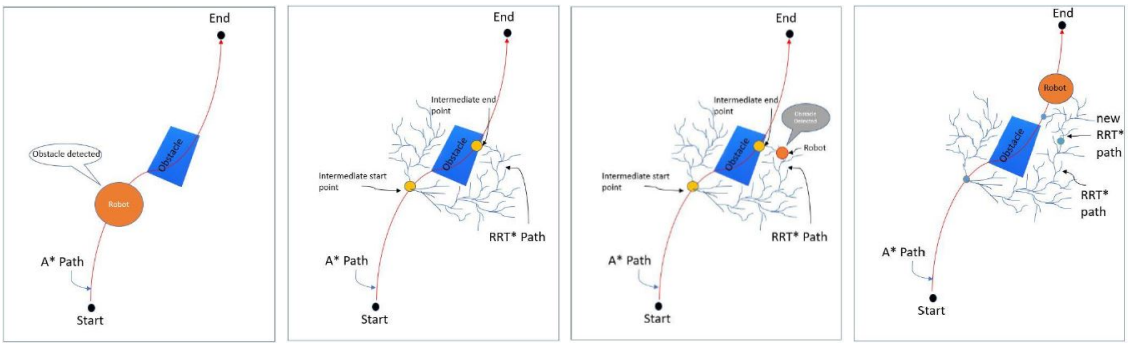
- Find a path in a dynamic environment with fusion of RRT* as local planning algorithm and A* as global planning algorithm.
- Find an optimal global path connecting start and end points using A*
- Sample points around the dynamic obstacle when detected and plan a path around the obstacle using RRT*.
- Connect the path generated by A* and RRT*.
- Create a package in ROS.
- Embed the ROS package in Turtle Bot and simulate on Gazebo.

IV. The Algorithm

Most strategies trying to solve the motion planning problem in dynamic environments using a global and a local planner make an assumption that the global path is the most optimal path and the robot, if it encounters a dynamic obstacle,the robot must reconnect to the same global path after avoiding the obstacle on this route. But the issue with these strategies is that the global path sub-optimal, travelling which could be expensive at times [1]. Our proposed method will make an effort to solve this problem by following an optimal path globally instead of a sub-optimal global path. This method would particularly be preferable in applications where the robot is deployed in a medium-sized complex indoor environment and travel time is a critical factor at play as following an optimal path globally would always be quicker than following a sub-optimal one.

The algorithm takes a map of the indoor environment as input obtained as a result of RGB-D or visual SLAM approaches. This map being discretized, makes it convenient to apply a search based algorithm for finding an optimum path.

In this algorithm we give the current position of robot at start as input. However, in practical applications if this input is not available, the robot first localizes itself in the environment to know its current location in the world. Given a goal-position, it computes an optimum path from its initial position to the goal-position by applying the A* algorithm on the discretized map as opposed to using RRT* which would always give a sub-optimal path irrespective of the computation time. A* might take more processing time and give a path slower than the first path given by RRT*, but it would be an optimal one, thus making this algorithm important for time-critical applications.



(a) Detecting obstacle on global path (b) Re-planning using RRT* avoiding the obstacle (c) Re-detection of obstacle while traversing local path (d) Re-planning again using RRT* to avoid the obstacle

Fig. 1: Different stages of operation of the algorithm

Once generated a global path, robot starts executing the path which is void of any obstacles covered in the 2D occupancy grid. If the robot does not detect any 'new' obstacles along this global path, it continues to follow this optimal path to the goal. If however, the robot detects a new obstacle while traversing this route using its range sensor as seen in Fig. 1a, there is a need to re-plan. An important requirement for re-planning is that it must be done quickly in real-time, which could otherwise lead to a collision. On detection of any obstacle by the range sensor, the robot waits for two successive time frames to check if the dynamic obstacle is stationary or moving and if there is any possibility of it colliding with the obstacle by comparing the velocity vectors of obstacle and robot itself. If there is a possibility of collision with this obstacle, then it calls the re-plan module.

From the current location along the global path as an intermediate start point as seen in Fig. 1b, the re-plan module deploys the RRT* algorithm to find an alternate 'local' path to get past this dynamic obstacle. Considering assumption(3), an intermediate goal position is set as a location on the global path a few way-points after the point where the robot detected the obstacle, and the re-plan module will compute a new route from intermediate start position to intermediate goal location. The choice of how many way-points to be skipped from the intermediate start point will depend upon the grid density of the input occupancy map. Having no idea about the dimension and shape of the obstacle, the algorithm assumes the obstacle to be approximately of 60cm in diameter and RRT* samples its points in a local region avoiding the 60cm circular region around where the obstacle was detected. Since this re-route needs to be done between two near-by points, the algorithm rapidly generates a path. Also, since avoiding the dynamic obstacle is the main objective at the spur of the moment, optimality of new path is no longer an immediate concern thus justifying the use of RRT* locally.

Once the local path is obtained, the robot takes a de-tour and starts moving along this new local path avoiding the obstacle. If however, the obstacle is larger than assumed, the robot while traversing along the path would again reach a state of possible collision with the obstacle and would plan a new local route with its current position as intermediate start point and a new position beyond our previous local goal location as our new intermediate goal point. This can be seen happening in Fig. 1c where the robot again detects the obstacle while following the local path.

As seen in Fig. 1d, the robot will avoid any dynamically encountered static or moving obstacles and re-plan its path to reach a new point on the global path from its current position on the global path using successive calls to the re-planning module if needed.

V. Methodology

The following section explains the detailed implementation architecture of our proposed method. The fig. 9 represents the system diagram of the proposed architecture. The architecture is divided into multiple blocks or nodes and each of the node is explained in detail.

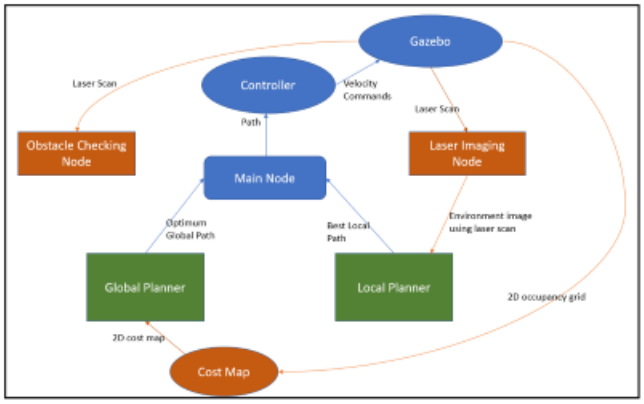


Fig. 2: Flow diagram representation of the proposed method.

A. Cost Map

Cost map takes in sensor data from the world, builds a 2D or 3D occupancy grid of the data (depending on whether a voxel based implementation is used), and inflates costs in a 2D costmap based on the occupancy grid and a user

specified inflation radius. The costmap_2d package provides a configurable structure that uses sensor data to store and update information about obstacles in the world. Each cell in this structure can be either free, occupied, or unknown. The costmap performs map update cycles at the rate specified by the update_frequency parameter. Each cycle, sensor data comes in, marking and clearing operations are performed in the underlying occupancy structure of the costmap, and this structure is projected into the costmap where the appropriate cost values are assigned. So in our project, we have the basic feature of costmap to detect obstacle. Here we check the value of the occupancy grid map provided by the costmap and decide whether the particular cell belongs to obstacle region or free space. We basically look out for free regions while checking for the valid node node points to be considered while planning the path.

B. Global Planner

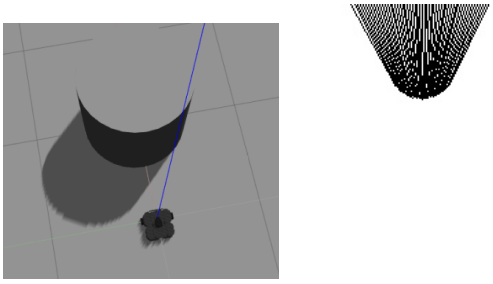
As the name suggests, global planner gives the path from the specified start position and the end position. Here we have used A* as our global planner which is known to give the optimal path in finite time. In this node, we calculate the optimal path between start and end positions using A*. The operation begins by taking the information about the grids from the costmap that which of the nodes in our map of the environment belong to free, obstacle and unknown regions. Also we get the information about the various other parameters like the grid resolution, grid height and width. Once we get this information, we start with the specified starting point and check whether it belongs to free space or not and start computing its neighbours. We maintain two lists namely open and closed list. Open list contains valid nodes. Valid nodes are the ones which do not lie in the obstacle region and also which are never checked before. Closed list contains those nodes which have been analysed. We calculate the total cost by adding the heuristic and cost. Heuristic is calculated as the euclidean distance between the current and the goal node while cost is calculated as the euclidean distance between the current node and its parent. The node with the least total cost is considered for further evaluation. Thus this procedure continues until the all the nodes in open list are checked. For formulating the path we backtrack the nodes obtained from goal to start nodes. Like we find the parent of goal node and then find the parent of parent until we reach the start node.

C. Obstacle Checking node

The node performs the crucial task of constantly checking if the future path of the robot is obstacle free or contains an obstacle. The node works on laser scan data and sets a flag in Ros params conveying other nodes about presence of a dynamic obstacle lying along the path. For this purpose, the algorithm checks if the point 1 meter away from current point along the path is obstacle free and sets the flag accordingly.

D. Laser Imaging node

The laser imaging node works in conjunction with the local planner node which executes RRT*. Once the obstacle checking node conveys the presence of an obstacle along the path, the laser imaging node is deployed. This node takes the laser scan of the environment and converts it into a 2D occupancy image. The laser sends a 360° scan of the immediate environment of the robot via ros_msgs. The node takes this ros_msgs at every instance and converts it into an image since the RRT* i.e the local planner node requires environment information as an image. The fig. 3a shows the robot detecting an obstacle while the fig. 3b represents the output of the laser imaging node creating local image of the environment around the robot. The node gets a 360° scan of the environment and along each of the scan line if the range value along the scan lies within the dimensions of the image, black lines are constructed from the point of detection of the obstacle to end of image along the scan direction. The image file created at every scan iteration and edited using OpenCV. The white specs found in the laser scan in fig. 3b is update using image processing with OpenCV libraries to make the algorithm more robust and less prone to bugs. The final updated image is then sent to other node using a ros_bridge.



(a) The local environment around the robot (b) Laser image of the environment

Fig. 3: Laser Imaging node in action

E. Local Planner

The local planner node is the RRT* implementation over the laser scan image to find a quick route from current location of the robot to the 4th next point on the global path. The grid size of global map on which A* computes its path is 0.5m x 0.5m while the laser imaging node creates an image of 4m x 4m environment around the robot with robot at its center. This means that, along any direction the minimum distance from position of robot to end of image is 2 m, which would mean the presence of minimum of 4 way-points from current robot location to end of image in any direction. The local planning node reads the processed local environment image created by the laser imaging node and then uses it to sample points around the robot to reach to the desired intermediate goal point from the current position. An example of the input laser scan image can be seen in fig. 4. The image on the right is the one on which RRT* works upon. There are a few unwanted white specs in the image, but using gaussian blur and image processing blocks those unwanted white regions from the area of interest. As a result, even though the RRT* might sample some points in the white region beyond region of interest, these

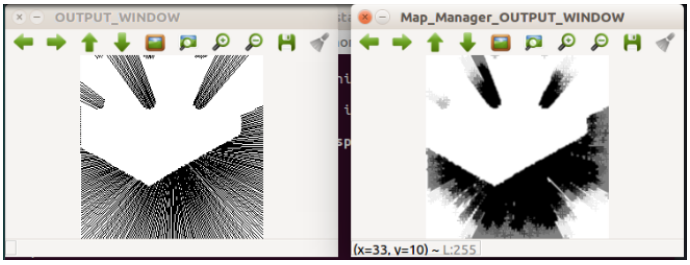


Fig. 4: Laser scan images.

points would get rejected due to lack of possible connections with the existing tree. This helps to further eliminate unwanted bugs.

F. Controller Node

The global and local planner nodes give the desired path to be followed, but along with it, a controller is required to complete the navigation stack. For locomotion of robot, a PID controller was implemented to achieve point-to-point drive for the robot.

G. Main Node

This node performs the crucial task of integrating all the above mentioned nodes of the robot. As seen in the fig. 9, the main node links all other nodes together. It first creates an instance of global planner and gets a global path from the global_planning node. This path is fed into the controller which moves the robot along the path until the obstacle_checker node detects an obstacle and changes the flag status. Once detecting the obstacle, the controller returns back the non-executed path to the runner node. The runner node then initiates the local planner node with the local environment laser image and gets back a path from the local node. The main node sends this local-path to the controller and the controller moves the robot along this new path. If the new path is free of obstacle, the robot will reach the desired goal position and continue on the previous global path from the newly reached node, otherwise, the robot, on the path, will again call the RRT* module and repeat this process till it safely reaches a point on the global path.

Just like any other other project, this project too did leave us in many challenges. But as they say, each challenge brings new opportunities to learn along with it. Initially being totally new to ROS, it was quite a task to get the initial scratch into the problem. With gradual efforts and perseverance we somehow managed to get past the initial mole-hill. Next we encountered a problem about how to use the laser scan as an input to the local planner. The solution idea was obtained while searching through papers and online forums. The final challenge which we faced was integrating the two algorithms for effective navigation indoor spaces. While we haven't been able to solve this issue yet, we hope to eventually solve it.

VI. Results

Dynamic obstacle avoidance is an interesting and important research domain in motion planning as most of the real

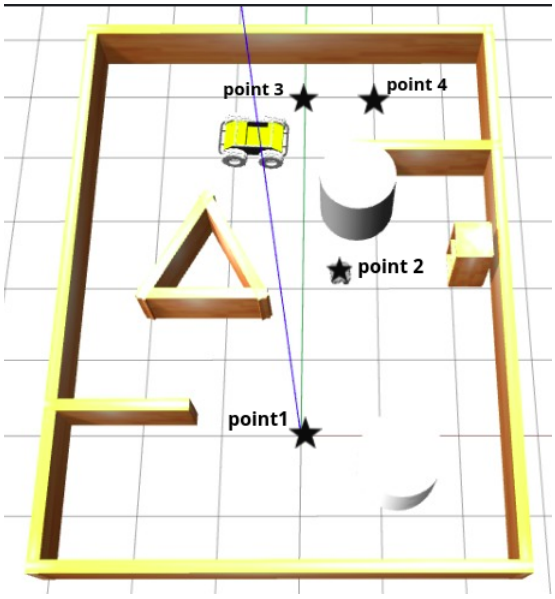


Fig. 5: Robot Environment in gazebo.

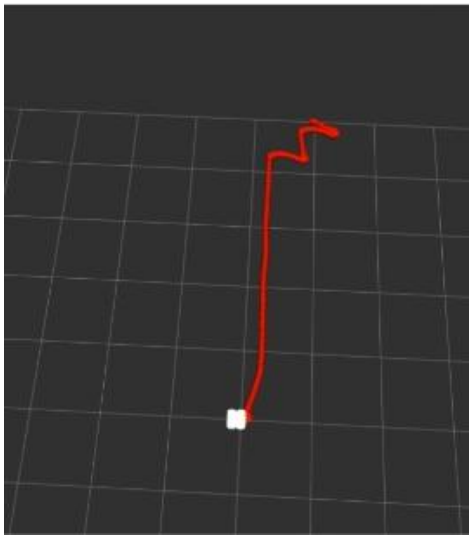


Fig. 6: A*path execution from rviz window.

life scenarios encountered are dynamic in nature. For the simplified problem statement to this dynamic motion planning, we offer the results of our approach.

The individual algorithms of A* and RRT* have been tested for providing the optimal path in different types of environments. Each of the two algorithm along with the different nodes have proved to provide satisfactory results.

The fig 5 is a screenshot from one of the environments which the algorithm was tested in. The A* algorithm was executed on the robot between Point 1 and Point 4 while the RRT* was executed between Point 2 and Point3. In the absence of the cylindrical obstacle, the robot successfully traverses between Points 1 and 4. This can be seen in fig. 6 which shows the rviz plot of the path executed by the robot in

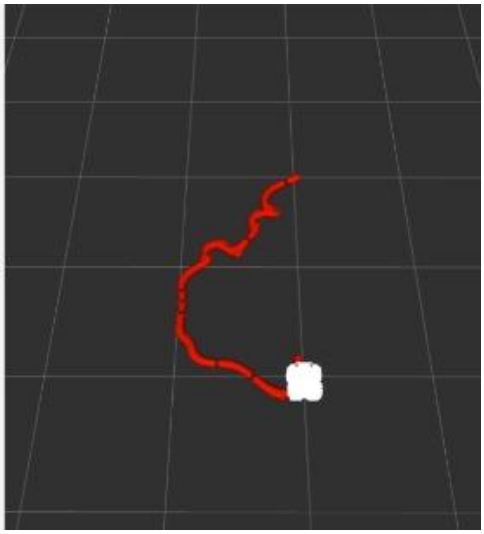


Fig. 7: RRT* path execution from rviz window.

gazebo. On encountering a dynamic obstacle, the robot would switch onto the RRT* and go past the obstacle avoiding it. This could be seen in fig. 7 in which shows the actual path followed by the robot in gazebo.

Although the algorithms execute smoothly when ran individually, we are facing certain issue while integrating the two due to which we haven't been able to provide an exhaustive evaluation of our algorithm.

VII. Conclusion and Future Work

The area of dynamic obstacle detection and avoidance is one of the niche fields in robotics which is considered as an unsolved problem. Though a solution already exists for the above described problem, there can always be multiple approaches to existing issues which open up new research areas in the field of robotics.

The open-ended problem of dynamic obstacle avoidance can be in future extended to dynamically moving obstacles. Also, with developments in sensor technology provide an opportunity for sensor fusion which is another emerging area in robotics. By sensor fusion, cheap and reliable sensors could be used to eliminate the potential drawbacks of individual sensors. The prospective future application of this kind of dynamic obstacle avoidance systems could be for autonomous robots to work in hospital like settings in collaboration with nursing robots like Baxter to take care of humans in quarantine rooms.

VIII. Timeline and Division of Tasks

We have proposed the following timeline which we have designed to complete our project as well as to achieve the desired objectives.

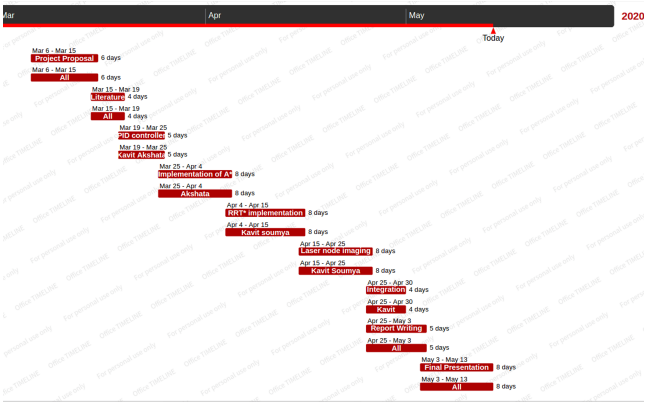


Fig. 8: Devised timeline for our project.

IX. Different aspects of Project

Table listed below describes the different of aspects of project to be covered as mentioned in the class.

Project Components	Aspects Covered	Details
Intellectual Merits	✓	There are papers which present combination of various algorithms proposed. But idea of combination of RRT* and A* was novel
Application Impacts	✓	The scope of project application extends to dynamic as well as static environment indoor as well outdoor environments. A simple example could be a robot vacuum cleaner or an autonomous shopping cart.
Related Work Review	✓	We went through good amount of literature review to cover most of the aspects of combination of algorithms.
Autonomy Design	✓	The algorithm inhibits the capability to detect dynamic obstacles and replans a path just enough to avoid the obstacle and follow the optimal path
Evaluation	✗	We have done extra group literature review as a substitute
Platform	✓	Simulation(Gazebo) .

Fig. 9: Project Aspects

X. Acknowledgement

For this project we would like to thank Prof. Jane Li and TA Alexandra Valiton as well as Mr. Sapan Agrawal and Mr. Sanjeev Kannan for helping us in different sections of our project.

References

[1] Connell, Devin, and Hung Manh La. "Dynamic path planning and replanning for mobile robots using RRT*." 2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC). IEEE, 2017.

[2] Hwangbo, Myung, James Kuffner, and Takeo Kanade. "Efficient two-phase 3D motion planning for small fixed-wing UAVs." Proceedings 2007 IEEE International Conference on Robotics and Automation. IEEE, 2007.

[3] Bohlin, Robert. "Path planning in practice; lazy evaluation on a multi-resolution grid." Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No. 01CH37180). Vol. 1. IEEE, 2001.

[4] Qureshi, Ahmed Hussain, et al. "Potential guided directional-RRT* for accelerated motion planning in cluttered environments." 2013 IEEE International Conference on Mechatronics and Automation. IEEE, 2013.

[5] Otte, Michael, and Emilio Frazzoli. "RRTX: Asymptotically optimal single-query sampling-based motion planning with quick replanning." The International Journal of Robotics Research 35.7 (2016): 797-822.

[6] Ferrer, Gonzalo, et al. "Robot social-aware navigation framework to accompany people walking side-by-side." Autonomous robots 41.4 (2017): 775-793.
[7] Vemula, Anirudh, Katharina Muelling, and Jean Oh. "Path planning in dynamic environments with adaptive dimensionality." Ninth Annual Symposium on Combinatorial Search. 2016.
[8] M. Imran and F. Kunwar, "A hybrid path planning technique developed by integrating global and local path planner," 2016 International Conference on Intelligent Systems Engineering (ICISE), Islamabad, 2016, pp. 118-122.

XI. Appendix

The entire code of out project can be viewed from our github repository which can be accessed via the following link: https://github.com/KavitShah1998/Combined_Astar_with_RRTstar_in_ROS