Real-time optimal path planning of non-holonomic robots

Abizer Patanwala
Department of Robotics Engineering
Worcester Polytechnic Institute
Worcester, United States of America
apatanwala@wpi.edu

Keshubh Sharma
Department of Robotics Engineering
Worcester Polytechnic Institute
Worcester, United States of America
kssharma@wpi.edu

Swapneel Wagholikar
Department of Robotics Engineering
Worcester Polytechnic Institute
Worcester, United States of America
swagholikar@wpi.edu

Abstract—**Path planning is the key aspect for performance of a mobile robot in any application. Creating a path around static obstacles is achievable since decades but moving around dynamic obstacles that present themselves on the planned path is a major challenge for modern applications in robotics. In this paper we present a solution combining sampling based algorithms AIT* & BIT* for path planning and Model Predictive Control for obstacle avoidance, and compare them to the already existing solutions in a virtual environment using simulation tools like RViz and Gazebo.**

I. Introduction

Real-time optimal path planning of non-holonomic robots is a complex problem in robotics that involves finding the most efficient path for a robot to navigate from one location to another while considering the physical constraints and limitations of the robot. Non-holonomic robots, which have restrictions on their movement and turning capabilities, require specialized planning algorithms that can optimize their trajectories while ensuring that they avoid obstacles and reach their destination within a specified timeframe. This field of research has applications in many areas, including autonomous vehicles, mobile robots, and industrial automation, and is of great importance for enabling robots to operate in real-world environments with safety and efficiency. For this paper, we are focusing on application impact in warehouse management. In warehouse scenario, there is a complex environment and multiple robots should operate in the same without colliding. Each of these individual robots can be treated as one dynamic obstacle. For these types of problems, we usually need two planners. A global planner to find the optimal path and the local planner to execute the motion in real-time environment with dynamic obstacles. Our robot need to navigate between the free spaces keeping safe distance from the walls or racks and without collision with other dynamics. The complexity of this problem increases exponentially as we increase the number of agents in the environment. To tackle these challenges, the work has been done considering the sampling-based algorithms for global path planning and algorithms like APF etc. for local path planning. We have observed that some state-of-the-art algorithms perform better than existing standard algorithms such as A*, RRT-variants etc. Therefore, we are planning to explore and implement more state-of-the-art algorithms such as AIT*, BIT* for global planning and to combine these with the Model Predictive Control (MPC) algorithm for local planning. MPC overcomes some limitations of APF such as local minima problem and it allows for the vehicles and obstacle constraints to be taken into consideration. The effectiveness of planning algorithms in real-world is directly influenced by computational cost and these state-of-the-art algorithms proved efficient in computations.

A. Adaptively Informed Trees (AIT*)

AIT* is an extension of the traditional Informed RRT* algorithm and uses an adaptive sampling strategy to generate a tree-based roadmap of the environment. This strategy allows AIT* to allocate more sampling effort in areas of the environment where the robot is more likely to find a feasible path, reducing the overall computational cost of the algorithm. Additionally, AIT* uses a novel information-theoretic approach to evaluate the quality of each candidate path. This approach allows the algorithm to not only find the shortest path but also the most informative path that provides the robot with the most knowledge about its environment.

B. Batch Informed Trees (BIT*)

BIT* is also an extension of Informed RRT* algorithm but is uses a batch processing approach to generate a tree-based roadmap of the environment. This approach allows BIT* to process multiple queries simultaneously, reducing the overall computational cost of the algorithm. Additionally, BIT* uses an informed sampling strategy. This strategy helps the algorithm to quickly generate a roadmap of the environment and plan a feasible path for the robot.

C. Model Predictive Control (MPC)

MPC is a control strategy that uses a dynamic model of the system and a cost function to optimize the future trajectory of the robot. The algorithm uses a finite horizon approach to generate a sequence of control actions that minimize the cost function over a certain time. At each time step, the MPC algorithm re-optimizes the trajectory based on the current state of the robot and updates the control actions accordingly. MPC is particularly useful for robots operating in dynamic environments where the optimal trajectory may change over time.

II. Related Work

There have been various approaches to motion planning in the presence of dynamic obstacles, including potential field approach, graph search-based algorithms, sampling-based algorithms, and velocity obstacles. The literature suggests that each approach has some degree of success in certain areas, but also has flaws in other areas. The potential field approach has drawbacks in terms of completeness of the algorithm and the problem of local minima, while graph search algorithms have superior completeness and optimality but have increasing time complexity and space requirements. Sampling-based techniques have weaker completeness and optimality but can perform better in higher-dimensional problems if the number of samples is chosen appropriately. Most of the authors divide the problem into global and local planning. Hence, we will explore the related work in following criteria for the same:

A. Graph-based planning approachess

From the perspective of planning, a path can be established by utilizing graph searching algorithms that traverse the many grid states, providing a solution—which is not always the best one—or not (there is no conceivable solution) to the path planning problem. The Dijkstra Algorithm is a graph-searching technique that identifies the graph's single-source shortest path. The A-Star Algorithm (A*), an extension of Dijkstra's graph search algorithm, is a graph searching algorithm that permits quick node searches thanks to the use of heuristics. It works well for scanning places that the vehicle is familiar with already [1], but it is expensive in terms of memory and speed for large areas. Many mobile robotics applications, such the dynamic A (D) [2], Field D [3], Theta [4], Anytime repairing A (ARA) and Anytime D (AD) [5], have served as the foundation for improvement.

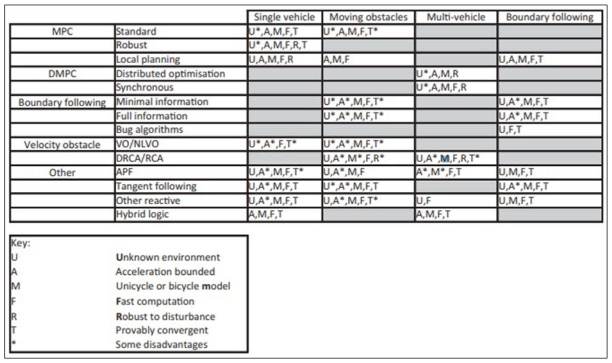
B. Sampling-based planning approaches

By planning in high-dimensional spaces, these planners attempt to address time constraints that deterministic techniques cannot. The method entails randomly selecting the configuration space or state space and searching within it for connectedness [6]. The problem is that the solution is not ideal. The Probabilistic Roadmap Method (PRM) and the Rapidly Exploring Random Tree (RRT) are the two techniques most frequently employed in robotics [7, 8]. By doing a random search through the navigation area, it enables quick planning in semi-structured settings [8]. It can also take non-holonomic constraints into account. In sampling-based planning, heuristics have been employed to direct the search and concentrate the approximation. Rapidly-exploring Random Trees (RRT) [8] is built upon by RRTConnect [9] by progressively constructing two trees, one rooted in the start state and the other in the destination state. Although this method can produce very quick initial solution times, it is not virtually certainly asymptotically optimal and does not become better with further computational time. The almost certainly asymptotically optimal RRT* [10] includes an ellipsoidal heuristic into informed RRT* [11]. This does not direct the search but increases the convergence rate by directing the incremental approximation to the pertinent area of the state space.

An growing number of states are sampled in batches by Batch Informed Trees (BIT*) [12, 13], which perceives these sampled states as an edge-implicit random geometric graph (RGG) [14]. As a result, BIT* can process the states in order of potential solution quality using a series of informed graph searches. By employing incremental search methods, BIT* effectively reuses data from earlier searches and approximations, but it does not alter its heuristic during the search.

C. Local path planning approaches

There are various approaches for local path planning. Some of the standards are listed below which are used in non-holonomic robots. Other algorithms can be found in the fig.1



		Single vehicle	Moving obstacles	Multi-vehicle	Boundary following
MPC	Standard	U*,A,M,F,T	U*,A,M,F,T*		
	Robust	U*,A,M,F,R,T			
	Local planning	U,A,M,F,R	A,M,F		U,A,M,F,T
DMPC	Distributed optimisation			U*,A,M,R	
	Synchronous			U*,A,M,F,R	
Boundary following	Minimal information		U*,A*,M,F,T*		U,A*,M,F,T
	Full information		U*,A*,M,F,T*		U,A*,M,F,T
	Bug algorithms				U,F,T
Velocity obstacle	VO/NLVO	U*,A*,F,T*	U*,A*,M,F,T*		
	DRCA/RCA		U,A*,M*,F,R*	U,A*,M,F,R,T*	
Other	APF	U,A*,M,F,T*	U,A*,M,F	A*,M*,F,T	U,M,F,T
	Tangent following	U,A*,M,F,T	U*,A*,M,F,T		U,A*,M,F,T
	Other reactive	U,A*,M,F,T	U,A*,M,F,T*	U,F	U,M,F,T
	Hybrid logic	A,M,F,T		A,M,F,T	

Key:	
U	Unknown environment
A	Acceleration bounded
M	Unicycle or bicycle model
F	Fast computation
R	Robust to disturbance
T	Provably convergent
*	Some disadvantages

Fig. 1. Comparison of local path planning algorithms

The concept of artificial potential field (APF) is derived from the potential field concept in physics, which views object movement as the outcome of two different types of forces. The target point's gravitational pull pulls on the robot in the planning space, and the obstacle repels it. The robot moves in the direction of the target point under the influence of the two forces, and during movement it can avoid obstacles in the planning space and arrive at the target safely. For real-time robot path planning, Vadakkepat et al. [15] presented a novel method dubbed evolutionary APF (EAPF). Through simulating, the robustness and effectiveness are confirmed. A virtual obstacle idea built on the APF was introduced by Min GP et al. [16] to examine the path planning of mobile robots. The outcomes demonstrate the method's viability and ease of use. A modified APF technique was put forth by Cao et al. [17] for the path planning of mobile robots in dynamic environments. The dynamic path-planning scheme's usefulness was shown by computer simulation and testing. Zhang et al. [18] introduced the evolving APF approach as a solution to the issues that path planning caught in local minimum.

Model Predictive Control (MPC), or alternatively Receding Horizon Control (RHC), is a feedback control scheme that generates the control action based on a finite horizon open loop optimal control with the measured state as the initial state and

is capable of directly handling the state/input constraints. Due to the strong results in industrial process control applications, the MPC approach has grown highly popular [19].
Model predictive control (MPC) architectures have recently been used to tackle collision avoidance issues, and this method appears to have a lot of promise for delivering effective navigation that is easily extended to robust and nonlinear issues. They have several advantages over commonly employed velocity obstacle-based and artificial potential field (APF) approaches, which may be more conservative when applied to higher order vehicle models. MPC is still being developed and has many advantageous characteristics for sensor-based navigation, such as solutions to boundary following issues, the ability to avoid moving impediments, and the ability to coordinate many vehicles.

III. Proposed Method

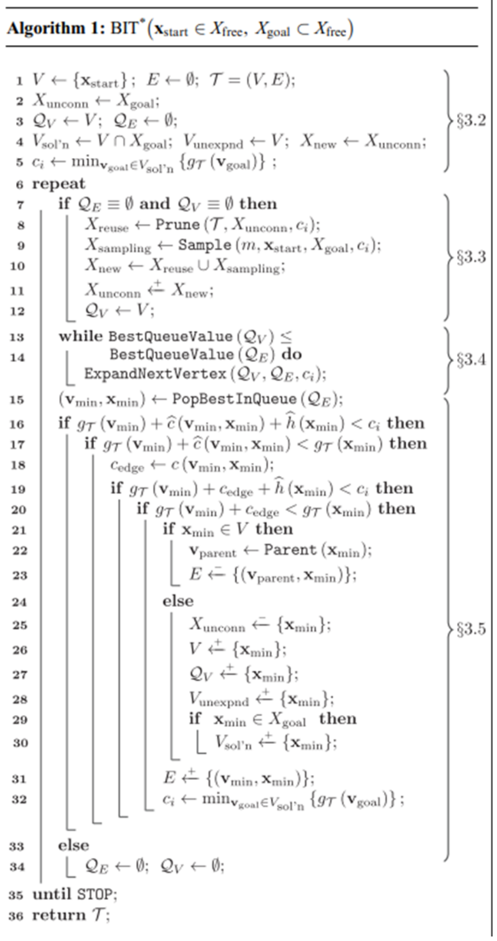
We are implementing state-of-the-art algorithms like AIT*, BIT* for global path planning along with MPC local path planner. For baseline, we are comparing it with standard APF + RRT planner. We believe that this extension from the standard baseline algorithm will result into better computational efficiency, less time-complexity and more smoothness of path traversed. More details about these algorithms are described below:

A. Batch Informed Trees (BIT*) Algorithm Overview

The key idea behind BIT* is to reduce the computational cost of RRT* by processing batches of nodes instead of one node at a time. This allows BIT* to explore the configuration space more efficiently and converge to a solution faster than RRT*. Additionally, the cost updates and pruning steps help to maintain an efficient search tree that does not waste computational resources exploring regions of the configuration space that are unlikely to lead to a solution.
Here is a high-level overview of the BIT* algorithm:

1) Initialize the search tree with the start node.
2) Sample a batch of nodes from the configuration space. These nodes are called ”waypoints”.
3) For each waypoint, attempt to connect it to the existing tree by finding the nearest node in the tree and creating a new edge between the waypoint and the nearest node.
4) Once all the waypoints have been connected to the tree, perform a cost update for each node in the tree. The cost of a node is the sum of the costs of the edges that connect it to the root node.
5) Prune the tree by removing any nodes whose cost is greater than a certain threshold.
6) Repeat steps 2-5 until the goal is reached or a certain time limit is exceeded.
7) If a path to the goal is found, return it. Otherwise, return failure.

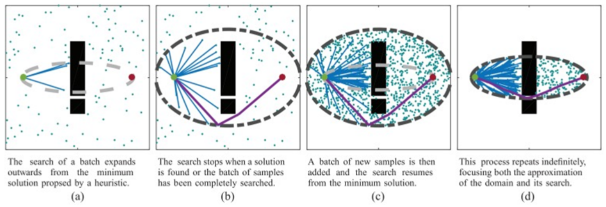
Detailed Algorithm can be seen in fig. 2.



Algorithm 1: BIT*$(\mathbf{x}_{\text{start}} \in X_{\text{free}},\ X_{\text{goal}} \subset X_{\text{free}})$

```
$V \leftarrow \{\mathbf{x}_{\text{start}}\};\ E \leftarrow \emptyset;\ \mathcal{T} = (V, E);$                                  } §3.2
$X_{\text{unconn}} \leftarrow X_{\text{goal}};$
$\mathcal{Q}_V \leftarrow V;\ \mathcal{Q}_E \leftarrow \emptyset;$
$V_{\text{sol'n}} \leftarrow V \cap X_{\text{goal}};\ V_{\text{unexpnd}} \leftarrow V;\ X_{\text{new}} \leftarrow X_{\text{unconn}};$
$c_i \leftarrow \min_{\mathbf{v}_{\text{goal}} \in V_{\text{sol'n}}} \{g_{\mathcal{T}}(\mathbf{v}_{\text{goal}})\};$
repeat
    if $\mathcal{Q}_E \equiv \emptyset$ and $\mathcal{Q}_V \equiv \emptyset$ then                                                    } §3.3
        $X_{\text{reuse}} \leftarrow$ Prune$(\mathcal{T}, X_{\text{unconn}}, c_i);$
        $X_{\text{sampling}} \leftarrow$ Sample$(m, \mathbf{x}_{\text{start}}, X_{\text{goal}}, c_i);$
        $X_{\text{new}} \leftarrow X_{\text{reuse}} \cup X_{\text{sampling}};$
        $X_{\text{unconn}} \overset{+}{\leftarrow} X_{\text{new}};$
        $\mathcal{Q}_V \leftarrow V;$
    while BestQueueValue$(\mathcal{Q}_V) \leq$                                                                    } §3.4
            BestQueueValue$(\mathcal{Q}_E)$ do
        ExpandNextVertex$(\mathcal{Q}_V, \mathcal{Q}_E, c_i);$
    $(\mathbf{v}_{\min}, \mathbf{x}_{\min}) \leftarrow$ PopBestInQueue$(\mathcal{Q}_E);$                                       } §3.5
    if $g_{\mathcal{T}}(\mathbf{v}_{\min}) + \widehat{c}(\mathbf{v}_{\min}, \mathbf{x}_{\min}) + \widehat{h}(\mathbf{x}_{\min}) < c_i$ then
        if $g_{\mathcal{T}}(\mathbf{v}_{\min}) + \widehat{c}(\mathbf{v}_{\min}, \mathbf{x}_{\min}) < g_{\mathcal{T}}(\mathbf{x}_{\min})$ then
            $c_{\text{edge}} \leftarrow c(\mathbf{v}_{\min}, \mathbf{x}_{\min});$
            if $g_{\mathcal{T}}(\mathbf{v}_{\min}) + c_{\text{edge}} + \widehat{h}(\mathbf{x}_{\min}) < c_i$ then
                if $g_{\mathcal{T}}(\mathbf{v}_{\min}) + c_{\text{edge}} < g_{\mathcal{T}}(\mathbf{x}_{\min})$ then
                    if $\mathbf{x}_{\min} \in V$ then
                        $\mathbf{v}_{\text{parent}} \leftarrow$ Parent$(\mathbf{x}_{\min});$
                        $E \overset{-}{\leftarrow} \{(\mathbf{v}_{\text{parent}}, \mathbf{x}_{\min})\};$
                    else
                        $X_{\text{unconn}} \overset{-}{\leftarrow} \{\mathbf{x}_{\min}\};$
                        $V \overset{+}{\leftarrow} \{\mathbf{x}_{\min}\};$
                        $\mathcal{Q}_V \overset{+}{\leftarrow} \{\mathbf{x}_{\min}\};$
                        $V_{\text{unexpnd}} \overset{+}{\leftarrow} \{\mathbf{x}_{\min}\};$
                        if $\mathbf{x}_{\min} \in X_{\text{goal}}$ then
                            $V_{\text{sol'n}} \overset{+}{\leftarrow} \{\mathbf{x}_{\min}\};$
                    $E \overset{+}{\leftarrow} \{(\mathbf{v}_{\min}, \mathbf{x}_{\min})\};$
                    $c_i \leftarrow \min_{\mathbf{v}_{\text{goal}} \in V_{\text{sol'n}}} \{g_{\mathcal{T}}(\mathbf{v}_{\text{goal}})\};$
    else
        $\mathcal{Q}_E \leftarrow \emptyset;\ \mathcal{Q}_V \leftarrow \emptyset;$
until STOP;
return $\mathcal{T}$;
```

Fig. 2. BIT* algorithm



The search of a batch expands outwards from the minimum solution propsed by a heuristic.
(a)

The search stops when a solution is found or the batch of samples has been completely searched.
(b)

A batch of new samples is then added and the search resumes from the minimum solution.
(c)

This process repeats indefinitely, focusing both the approximation of the domain and its search.
(d)

Fig. 3. visualization of BIT* algorithm

B. Adaptively Informed Trees (AIT*) Algorithm Overview

Adaptively Informed Trees (AIT*) is a sampling-based motion planning algorithm that is designed to efficiently solve high-dimensional motion planning problems with multiple objectives. AIT* builds on the A* search algorithm and the RRT* algorithm, combining their strengths and adding a few novel techniques to achieve better performance. One key feature of AIT* is its ability to adaptively choose the sampling bias based on the proximity to the goal configuration. AIT* uses a heuristic estimate of the remaining cost to the goal, similar to A*, to bias the sampling towards areas closer to the goal. This allows AIT* to efficiently explore the state

space and find a solution faster. Another feature of AIT* is its ability to handle multiple objectives. AIT* uses a weighted sum approach to combine the different costs, allowing the user to specify the relative importance of each objective. Additionally, AIT* uses a novel technique called ”cost-to-go estimation” to estimate the cost of extending the tree towards a sampled configuration.
Here is a high-level overview of the AIT* algorithm:

1) Initialize a tree with the starting configuration as the root node.
2) Sample a random configuration in the state space and find the closest node in the tree to it.
3) Extend the tree towards the sampled configuration by generating a path from the closest node to the sampled configuration.
4) Calculate the cost of the path based on the multiple objectives, such as minimizing distance, maximizing clearance, or minimizing time.
5) Update the cost of all nodes along the path and rewire the tree to ensure optimality.
6) Repeat steps 2 to 5 until the goal configuration is reached or a certain termination condition is met.
7) If the goal configuration is reached, return the path from the root node to the goal configuration. Otherwise, return failure.

Detailed Algorithm can be seen in fig. 4.

Algorithm 2. Adaptively Informed Trees (AIT*)

```
$c_{\text{current}} \leftarrow \infty$
$X_{\text{sampled}} \leftarrow X_{\text{goal}} \cup \{\mathbf{x}_{\text{start}}\}$
$V_{\mathcal{F}} \leftarrow \mathbf{x}_{\text{start}}$; $E_{\mathcal{F}} \leftarrow \phi$; $\mathcal{Q}_{\mathcal{F}} \leftarrow$ expand($\mathbf{x}_{\text{start}}$)
$V_{\mathcal{R}} \leftarrow X_{\text{goal}}$; $E_{\mathcal{R}} \leftarrow \phi$; $\mathcal{Q}_{\mathcal{R}} \leftarrow X_{\text{goal}}$
repeat
    if $\min_{\mathbf{x} \in \mathcal{Q}_{\mathcal{R}}} \{\text{key}_{\mathcal{R}}^{\text{AIT}^*}(\mathbf{x})\} < \min_{(\mathbf{x}_s, \mathbf{x}_t) \in \mathcal{Q}_{\mathcal{F}}} \{\text{key}_{\mathcal{F}}^{\text{AIT}^*}(\mathbf{x}_s, \mathbf{x}_t)\}$
    or target of best edge in forward queue is inconsistent
        $\mathbf{x} \leftarrow \text{argmin}_{\mathbf{x} \in \mathcal{Q}_{\mathcal{R}}} \{\text{key}_{\mathcal{R}}^{\text{AIT}^*}(\mathbf{x})\}$
        $\mathcal{Q}_{\mathcal{R}} \xleftarrow{-} \mathbf{x}$
        if $\widehat{h}_{\text{con}}[\mathbf{x}] < \widehat{h}_{\text{exp}}[\mathbf{x}]$
            $\widehat{h}_{\text{exp}}[\mathbf{x}] \leftarrow \widehat{h}_{\text{con}}[\mathbf{x}]$
        else
            $\widehat{h}_{\text{exp}}[\mathbf{x}] \leftarrow \infty$
            update_state($\mathbf{x}$)
        for all $\mathbf{x}_i \in$ neighbors($\mathbf{x}$) do
            update_state($\mathbf{x}_i$)
    else if $\min_{(\mathbf{x}_s, \mathbf{x}_t) \in \mathcal{Q}_{\mathcal{F}}} \{g_{\mathcal{F}}(\mathbf{x}_s) + \widehat{c}(\mathbf{x}_s, \mathbf{x}_t) + \widehat{h}_{\text{con}}[\mathbf{x}_t]\} < c_{\text{current}}$
        $(\mathbf{x}_s, \mathbf{x}_t) \leftarrow \text{argmin}_{(\mathbf{x}_s, \mathbf{x}_t) \in \mathcal{Q}_{\mathcal{F}}} \{\text{key}_{\mathcal{F}}^{\text{AIT}^*}(\mathbf{x}_s, \mathbf{x}_t)\}$
        $\mathcal{Q}_{\mathcal{F}} \xleftarrow{-} (\mathbf{x}_s, \mathbf{x}_t)$
        if $(\mathbf{x}_s, \mathbf{x}_t) \in E_{\mathcal{F}}$
            $\mathcal{Q}_{\mathcal{F}} \xleftarrow{+}$ expand($\mathbf{x}_t$)
        else if $g_{\mathcal{F}}(\mathbf{x}_s) + \widehat{c}(\mathbf{x}_s, \mathbf{x}_t) < g_{\mathcal{F}}(\mathbf{x}_t)$
            if collision_free($\mathbf{x}_s, \mathbf{x}_t$)
                if $g_{\mathcal{F}}(\mathbf{x}_s) + c(\mathbf{x}_s, \mathbf{x}_t) + \widehat{h}_{\text{con}}[\mathbf{x}_t] < c_{\text{current}}$
                    if $g_{\mathcal{F}}(\mathbf{x}_s) + c(\mathbf{x}_s, \mathbf{x}_t) < g_{\mathcal{F}}(\mathbf{x}_t)$
                        if $\mathbf{x}_t \notin V_{\mathcal{F}}$
                            $V_{\mathcal{F}} \xleftarrow{+} \mathbf{x}_t$
                        else
                            $E_{\mathcal{F}} \xleftarrow{-}$ (parent$_{\mathcal{F}}$($\mathbf{x}_t$), $\mathbf{x}_t$)
                        $E_{\mathcal{F}} \xleftarrow{+} (\mathbf{x}_s, \mathbf{x}_t)$
                        $\mathcal{Q}_{\mathcal{F}} \xleftarrow{+}$ expand($\mathbf{x}_t$)
                        $c_{\text{current}} \leftarrow \min_{\mathbf{x}_{\text{goal}} \in X_{\text{goal}}} \{g_{\mathcal{F}}(\mathbf{x}_{\text{goal}})\}$
            else
                $E_{\text{invalid}} \xleftarrow{+} \{(\mathbf{x}_s, \mathbf{x}_t), (\mathbf{x}_t, \mathbf{x}_s)\}$
                if $(\mathbf{x}_s, \mathbf{x}_t) \in E_{\mathcal{R}}$
                    invalidate_rev_branch($\mathbf{x}_s$)
    else
        prune($X_{\text{sampled}}$)
        $X_{\text{sampled}} \xleftarrow{+}$ sample($m$, $c_{\text{current}}$)
        $V_{\mathcal{R}} \leftarrow X_{\text{goal}}$ ; $E_{\mathcal{R}} \leftarrow \phi$ ; $\mathcal{Q}_{\mathcal{R}} \leftarrow X_{\text{goal}}$
        $\forall \mathbf{x} \in X_{\text{sampled}} \setminus X_{\text{goal}}$, $\widehat{h}_{\text{con}}[\mathbf{x}] = \widehat{h}_{\text{exp}}[\mathbf{x}] = \infty$
        $\mathcal{Q}_{\mathcal{F}} \leftarrow$ expand($\mathbf{x}_{\text{start}}$)
until stopped
```

Fig. 4. AIT* algorithm

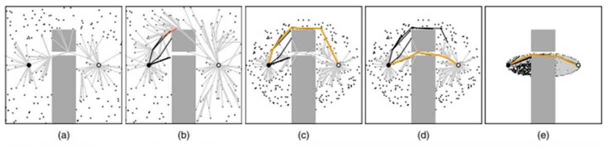


Fig. 5. visualization of AIT* algorithm

C. Model Predictive Control Algorithm Overview

The following is a high-level algorithm overview of Nonlinear Model Predictive Control (NMPC):
System modeling: Develop a mathematical model that describes the behavior of the system being controlled. This model should be nonlinear and time-varying, accounting for any nonlinearities and dynamics present in the system.
Define control objectives: Determine the control objectives, including the desired output and any constraints on the inputs and outputs.
Time horizon selection: Select a time horizon over which to optimize the control actions. This time horizon is typically divided into a finite number of time intervals or ”prediction horizons”.
Predict future system behavior: Using the mathematical model, predict the future behavior of the system over the prediction horizons.
Optimization: Determine the optimal control actions that achieve the desired output while satisfying the input and output constraints, based on the predicted behavior of the system.
Implementation: Implement the optimal control actions for the current time interval, and update the model and control strategy for the next time interval.
Repeat: Repeat steps 4-6 for each time interval in the prediction horizon, continuously updating the model and control strategy as needed.
Overall, NMPC is a complex and computationally intensive control strategy that involves predicting the future behavior of the system, optimizing control actions based on this prediction, and implementing the optimal actions over a finite prediction horizon. NMPC is an effective approach for controlling nonlinear systems with time-varying dynamics and constraints, and it is widely used in process control, robotics, and other advanced control applications.

IV. Experiment

In this section, we are planning to evaluate the performance of the proposed planner (AIT*, BIT* + MPC planner) in comparison to baseline based on our quantifying metrics. In the experiments we are conducting, the robot will have start point and end point given by the user in the dynamic obstacle environment. Here, dynamic obstacles represent the other robots and some randomly placed obstacles in the warehouse

which are not present in the map while global planning.
To begin with, we are experimenting on a 2D toy problem and the baseline we are considering is RRT + APF planner as considered state-of-the-art algorithms are the extensions of informed RRT*. After successful implementation of this 2D toy problem, we are planning to simulate the proposed planner in Gazebo warehouse environment and compare it with the baseline planner in this environment. We will be using Rviz for visualization purposes.

A. Performance Criteria

We are planning to evaluate these state-of-the-art algorithms with the standard RRT + APF planner with below performance metrics:

- Percentage of dynamic obstacles detected
- Computational efficiency
- Energy consumption
- Time taken to traverse the path
- Smoothness of path traversal
- Accuracy of optimal path traversal

B. Simulation Setup

We have conducted a thorough literature review of the latest algorithms used for both local and global path planning of non-holonomic robots and gain understanding of how these algorithms work. The implementation is planned in following 2 scenarios.

- **2D toy problem Scenario:** 2D baseline environment is implemented in python which uses RRT + APF path planner. The results are shown below. We are working on algorithm implementations and will be demonstrated in the similar fashion as baseline implementation. Therefore, it will be better for visualizing the differences between standard baseline and proposed state-of-the-art planner implementations. Fig. 6 shows the path explored and proposed by RRT. Fig. 7 shows the path explored and proposed by BIT*. Fig. 8 shows the path explored and proposed by AIT*. This implementation is done in python and visualized using matplotlib.
- **Warehouse Scenario:** As one of our application impact is warehouse management, we have started with creating an environment in Gazebo and planning to do the further simulations in the same.
    1) Smaller version of a warehouse environment
    2) A maze type warehouse environment

C. Simulation Results

For the algorithms stated before we've used the described simulation setup to run their implementation and compare them according to the performance criteria.

- Cost to goal
  This criteria shows us how the cost to goal changes per algorithm as the execution ensues. Fig. 12 shows that cost

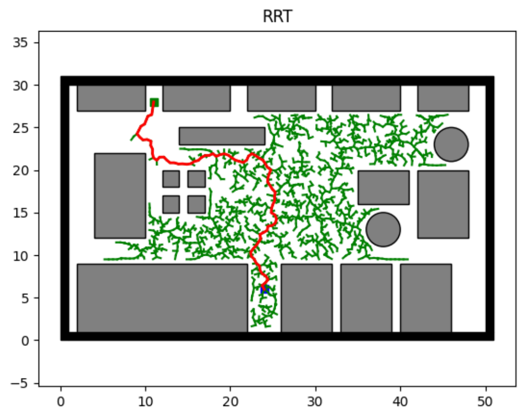


Fig. 6. Path planning using RRT

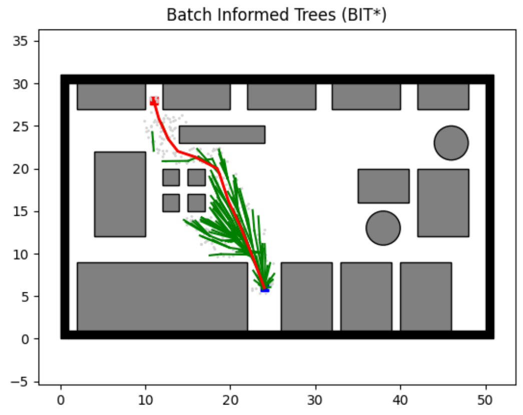


Fig. 7. Path planning using BIT*

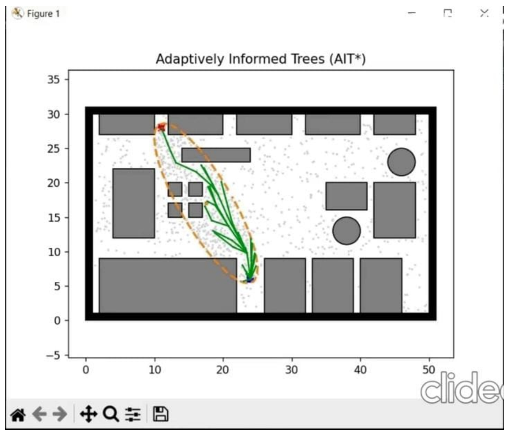


Fig. 8. Path planning using AIT*

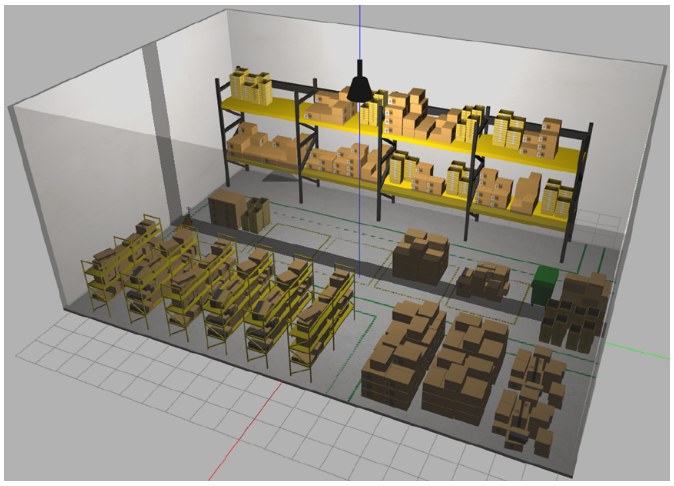


Fig. 9. Warehouse environment (orthogonal view)

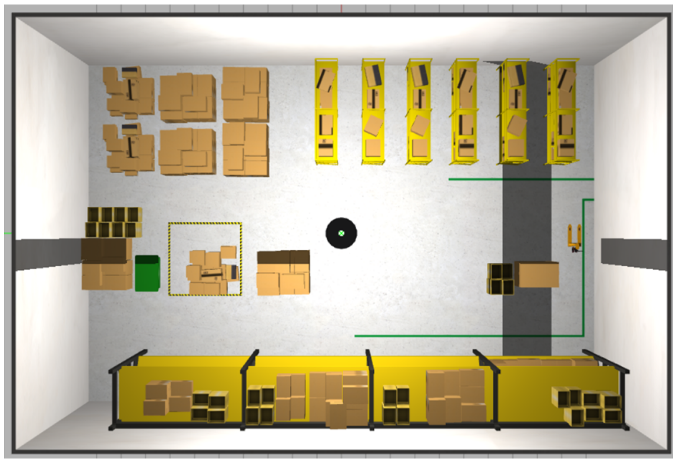


Fig. 10. Warehouse environment (top view)

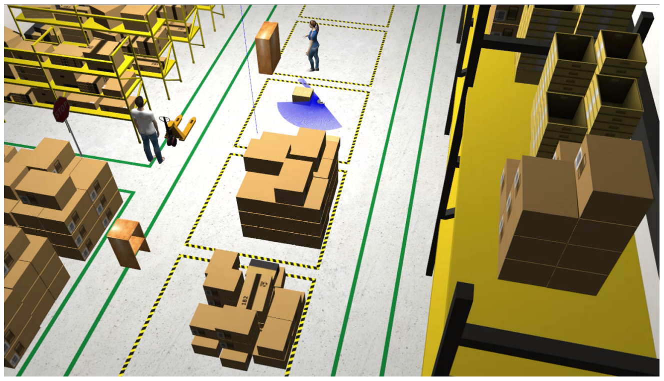


Fig. 11. TurtleBot traversing Warehouse environment

to goal for RRT stays the same until the goal node is find. Fig. 13 shows that cost to goal for BIT* reduces once the goal is found and is further dropped when the path is optimized. Fig. 14 shows the cost for AIT* reduces once the goal is found and then the algorithm tries to optimise it.

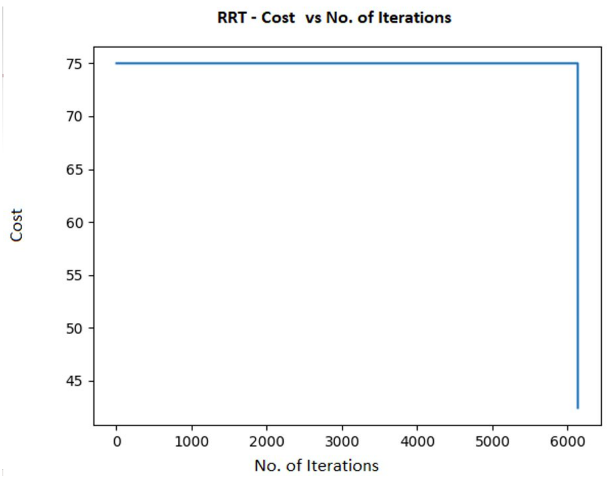


Fig. 12. Change in cost to goal in RRT

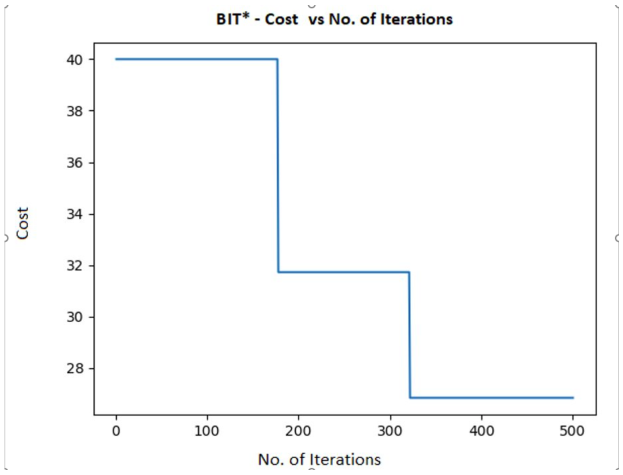


Fig. 13. Change in cost to goal in BIT*

- Nodes explored
This criteria shows us the computation needed per algorithm to reach to the goal. Fig. 15 shows that the total number of nodes explored increases as the algorithm executes. We can also see that the number of nodes explored is multiple times that of BIT* or AIT*. Fig. 16 shows that the total number of nodes explored in BIT* increases and once the goal is found the unnecessary nodes are pruned and the value drops. As the optimization occurs the number of nodes increases again. Fig. 17 shows that the total number of nodes explored in AIT* is continuously increasing till the goal is reached.
- Optimization time
This criteria shows us how long the algorithm takes

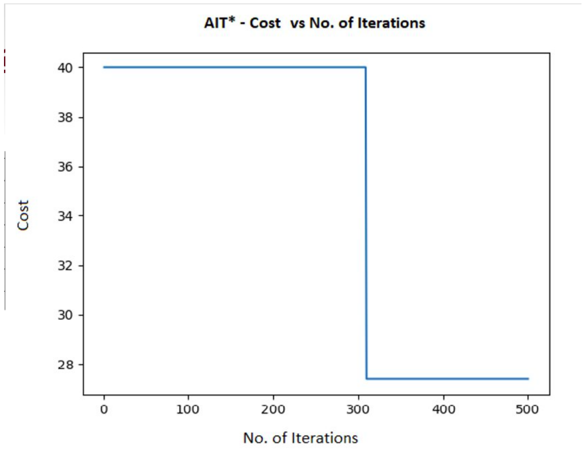


Fig. 14. Change in cost to goal in AIT*

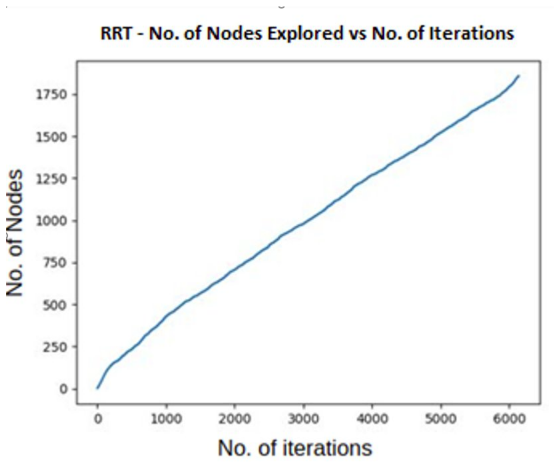


Fig. 15. Nodes explored in RRT

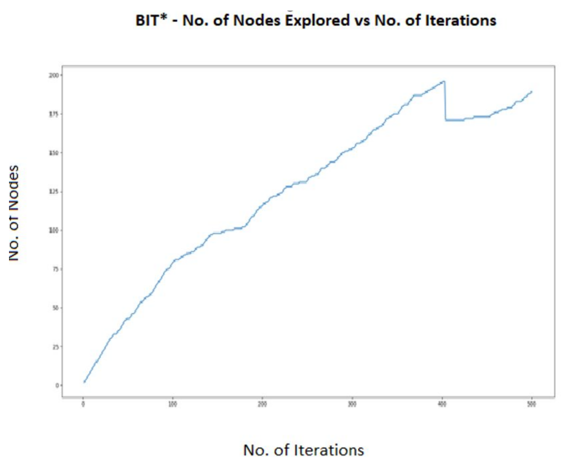


Fig. 16. Nodes explored in BIT*

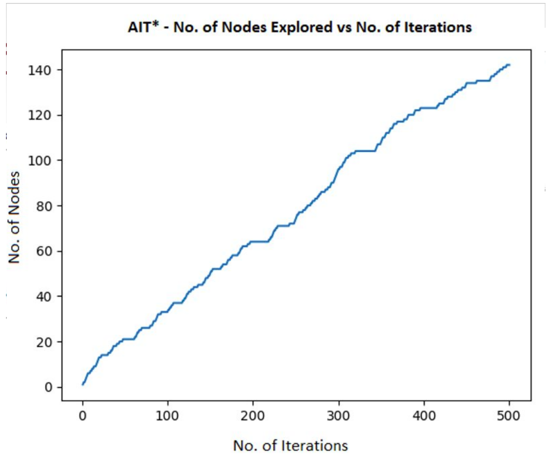


Fig. 17. Nodes explored in AIT*

to optimize the given trajectory. Fig. 18 shows us the optimization time for each path point provided by the global path planner.

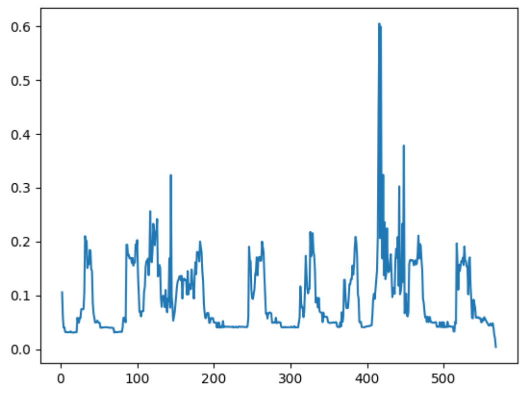


Fig. 18. Optimization time for MPC

- Path Deviation
  Path deviation refers to the alteration of provided path to avoid the unplanned obstacle in path. Fig. 19 shows us how much the robot deviates from the planned trajectory over time.

D. Limitations experienced

- AIT* is more computationally expensive than search algorithms such as RRT* and BIT*.
- BIT* requires large memory to store the search tree.
- BIT* is sensitive to the parameters such as batch size and threshold for node expansion.
- Careful Tuning of MPC is required which can be time consuming.
- Accurate model of the system is required in MPC for good performance.

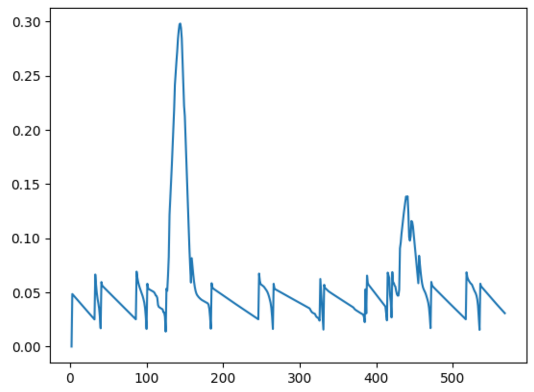


Fig. 19. Path deviation of robot during trajectory tracking

- If obstacles are placed such that the optimization provides 2 possible path for consecutive time steps, then the robot may collide.

V. Task division

- Literature Review: Collaborative Effort
- Environment Setup: Keshubh, Abizer
- Global Planner Implementation: Keshubh, Swapneel
- Local Planner Implementation: Swapneel, Abizer
- Output Evaluation: Collaborative Effort
- Documentation: Collaborative Effort

References

[1] M. Likhachev and D. Ferguson, "Planning long dynamically feasible maneuvers for autonomous vehicles," Int. J. Robot. Res., vol. 28, no. 8, pp. 933–945, Aug. 2009

[2] A. Stentz, "Optimal and efficient path planning for partially-known environments," in Proc. IEEE Int. Conf. Robot. Autom., 1994, pp. 3310–3317.

[3] D. Ferguson and A. Stentz, "Using interpolation to improve path planning: The field d* algorithm," J. Field Robot., vol. 23, no. 2, pp. 79–101, Feb. 2006.

[4] A. Nash, K. Daniel, S. Koenig, and A. Felner, "Theta*: Any-angle path planning on grids," in Proc. Nat. Conf. Artif. Intell., 2007, pp. 1177–1183

[5] M. Likhachev, D. Ferguson, G. Gordon, A. Stentz, and S. Thrun, "Anytime search in dynamic graphs," Artif. Intell., vol. 172, no. 14, pp. 1613–1643, Sep. 2008.

[6] M. Elbanhawi and M. Simic, "Sampling-based robot motion planning: A review," IEEE Access, vol. 2, pp. 56–77, 2014.

[7] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," IEEE Trans. Robot. Autom., vol. 12, no. 4, pp. 566–580, Aug. 1996.

[8] S. M. LaValle and J. J. Kuffner Jr., "Randomized kinodynamic planning," The International Journal of Robotics Research (IJRR), vol. 20, no. 5, pp. 378–400, 2001.

[9] J. J. Kuffner Jr. and S. M. LaValle, "RRT-Connect: An efficient approach to single-query path planning," in Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), 2000, pp. 995–1001.

[10] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," The International Journal of Robotics Research (IJRR), vol. 30, no. 7, pp. 846–894, 2011.

[11] J. D. Gammell, T. D. Barfoot, and S. S. Srinivasa, "Informed sampling for asymptotically optimal path planning," IEEE Transactions on Robotics, vol. 34, no. 4, pp. 966–984, 2018.

[12] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Batch Informed Trees (BIT*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs," in Proceedings of the IEEE International Conference of Robotics and Automation (ICRA). IEEE, 2015, pp. 3067–3074.

[13] J. D. Gammell, T. D. Barfoot, and S. S. Srinivasa, "Batch Informed Trees (BIT*): Informed asymptotically optimal anytime search," The International Journal of Robotics Research (IJRR), 2020.

[14] M. Penrose, Random Geometric Graphs. Oxford University Press, 2003, vol. 5.

[15] Vadakkepat, P.; Tan, K.C.; Wang, M.L. "Evolutionary Artificial Potential Fields and their application in real time robot path planning". In Proceedings of the 2000 Congress on Evolutionary Computation, LA Jolla, CA, USA, 16–19 July 2000; pp. 256–263.

[16] Park, M.G.; Lee, M.C. "Artificial potential field based path planning for mobile robots using a virtual obstacle concept". In Proceedings of the 2003 IEEE/ASME International Conference on Advanced Intelligent Mechatronics, Kobe, Japan, 20–24 July 2003; pp. 735–740.

[17] Cao, Q.X.; Huang, Y.W.; Zhou, J.L. "An evolutionary artificial potential field algorithm for dynamic path planning of mobile robot". In Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, Beijing, China, 9–15 October 2006; pp. 3331–3336.

[18] Zhang, Q.S.; Chen, D.D.; Chen, T. "An obstacle avoidance method of soccer robot based on evolutionary artificial potential field". Energy Procedia 2012, 16, 1792–1798

[19] Qin, Joe Badgwell, Thomas. (2001). Review of nonlinear model predictive control applications. 3-32.