Autonomous Navigation and Manipulation for a Mobile Base with a 6-DOF Robotic Arm

1st Girish Krishnan
Dept. of Electrical and Computer Engineering
University of California, San Diego
La Jolla, United States
gikrishnan@ucsd.edu

***Abstract*—This project investigates autonomous navigation and manipulation for a robotic system consisting of a car-like mobile base with an attached 6-DOF robotic arm. The primary objective is to enable the robot to navigate to a specified goal pose and configuration while avoiding obstacles in a predefined environment. The problem is framed within a 20x20 workspace filled with obstacles, where the robot must exit the area safely while adhering to motion and control constraints. As a baseline, we implemented a combination of Rapidly-Exploring Random Tree Star (RRT*) for global path planning and Proportional-Integral-Derivative (PID) control for local trajectory following. Although effective, this approach showed limitations in handling dynamic environments and complex constraints. To address these shortcomings, we introduced two key improvements. First, we employed Model Predictive Control (MPC) to optimize local trajectory tracking, incorporating obstacle avoidance and smooth motion generation directly into the optimization framework. This significantly enhanced the system's ability to navigate complex environments while adhering to kinematic and dynamic constraints. Second, we integrated a vision-based deep reinforcement learning (DRL) approach, enabling the robot to leverage depth images for direct policy learning. This allowed the robot to handle unforeseen scenarios and adapt to dynamic obstacles in real time.**

Index Terms*—Autonomous Navigation, Robotic Manipulation, Model Predictive Control, Vision Transformer, RRT

I. Introduction and Background

Mobile robots with manipulators, such as a car with a 6-DOF robotic arm, present unique challenges in planning and control, especially in cluttered environments with dynamic or static obstacles. The objective of this work is to achieve autonomous navigation to a goal pose and configuration while ensuring collision-free operation. Specifically, the robot must navigate out of a predefined 20x20 area filled with obstacles in the MuJoCo simulator, reaching a specified goal pose while avoiding collisions.

Traditional navigation techniques, such as Rapidly Exploring Random Trees (RRT*), provide a global path planning framework by iteratively sampling configurations and connecting feasible trajectories. While RRT* is effective for finding paths in complex spaces, it relies on a low-level controller, such as Proportional-Integral-Derivative (PID) control, to execute the planned trajectories. However, PID controllers are often insufficient in dynamic or high-dimensional environments, as they lack the capability to handle state and control constraints or optimize trajectories for smoothness and efficiency.

To address these limitations, we propose two improvements to the baseline RRT* + PID approach:

- **Model Predictive Control (MPC):** MPC optimizes the control inputs over a finite horizon by solving a constrained optimization problem at each timestep. It incorporates dynamic models of the robot and enforces state and control constraints, leading to smoother trajectories and better obstacle avoidance.
- **Vision-based Deep Reinforcement Learning (DRL):** DRL leverages high-dimensional visual inputs for policy learning, enabling the robot to learn complex navigation behaviors directly from raw images. By combining DRL with RRT* for global planning, we exploit the strengths of both approaches to handle high-dimensional state spaces and improve adaptability.

This project evaluates the baseline method and the two proposed improvements in terms of navigation success rate, obstacle avoidance, trajectory smoothness, and computational efficiency. The results demonstrate the advantages of MPC and vision-based DRL in achieving safe and efficient navigation for a robot with a mounted manipulator in a cluttered environment.

Note: The MPC and the Deep RL are implemented as two separate improvements to the baseline.

II. Methods

A. Baseline Method

The baseline approach integrates Rapidly Exploring Random Trees (RRT*) for global path planning and a proportional-integral-derivative (PID) controller for executing the planned trajectory. This method assumes a map of the environment populated with obstacles, and the robot's task is to navigate from an initial pose to a goal pose while avoiding obstacles.

1) Robot State and Control Model: The robot consists of a car with Ackermann steering and a robotic arm mounted on top. The state of the robot is defined as:

$$\mathbf{x} = [x, y, \theta, q_1, q_2, q_3, q_4, q_5, q_6]^\top,$$

where (x, y, θ) represents the 2D position and orientation of the car in the world frame, and $q_i, i = 1, \ldots, 6$ denote the joint angles of the robotic arm.

The control inputs are defined as:

$$\mathbf{u} = [\delta, v, \dot{q}_1, \dot{q}_2, \dot{q}_3, \dot{q}_4, \dot{q}_5, \dot{q}_6]^\top,$$

where δ is the steering angle, v is the throttle or longitudinal velocity of the car, and $\dot{q}_i$ are the joint velocities of the robotic arm.

The Ackermann motion model for the car is given by:

$$\dot{x} = v\cos(\theta), \quad \dot{y} = v\sin(\theta), \quad \dot{\theta} = \frac{v}{L}\tan(\delta),$$

where L is the wheelbase of the car.

2) RRT Path Planning:* The RRT* algorithm generates a collision-free trajectory from the robot's initial pose to the goal pose. The key steps in the algorithm are as follows:

Algorithm 1 RRT* Algorithm

```
1: Initialize tree $\mathcal{T}$ with root node as the initial pose $\mathbf{x}_{\text{init}}$.
2: for each iteration do
3:     Sample a random state $\mathbf{x}_{\text{rand}}$ within the map.
4:     Find the nearest node $\mathbf{x}_{\text{nearest}}$ in $\mathcal{T}$ to $\mathbf{x}_{\text{rand}}$.
5:     Steer from $\mathbf{x}_{\text{nearest}}$ to $\mathbf{x}_{\text{rand}}$ to generate $\mathbf{x}_{\text{new}}$.
6:     if $\mathbf{x}_{\text{new}}$ is collision-free then
7:         Add $\mathbf{x}_{\text{new}}$ to $\mathcal{T}$.
8:         Rewire the tree to ensure path cost optimality.
9:     end if
10: end for
11: Return the optimal path from $\mathbf{x}_{\text{init}}$ to $\mathbf{x}_{\text{goal}}$.
```

3) PID Control: To track the trajectory generated by RRT*, we employ a PID controller for the car's Ackermann steering model. The controller computes the desired control inputs (δ, v) based on the error between the current state and the target state on the path:

$$\delta = K_p e_\theta + K_i \int e_\theta \, dt + K_d \frac{de_\theta}{dt},$$

$$v = K_p^v e_d + K_i^v \int e_d \, dt + K_d^v \frac{de_d}{dt},$$

where $e_\theta = \theta_{\text{target}} - \theta$ is the heading error, $e_d = d_{\text{target}} - d$ is the distance error, and K_p, K_i, K_d are the PID gains for steering and velocity control.

4) Limitations of the Baseline: The RRT* and PID-based approach performs well in simple environments but has several limitations:

- **Suboptimal Trajectories:** RRT* trajectories can be jerky and suboptimal, leading to inefficient navigation.
- **Local Control Limitations:** The PID controller struggles with sharp turns and dynamically changing environments.
- **Lack of Arm Coordination:** The baseline method does not explicitly optimize the joint configurations of the robotic arm to avoid collisions during navigation.
- **Static Map Assumption:** The approach assumes a known, static map, limiting its applicability in dynamic or partially observed environments.

III. Proposed Method

This section introduces the proposed methods for autonomous navigation and manipulation. The methods include Model Predictive Control (MPC) for local trajectory optimization and vision-based Deep Reinforcement Learning (DRL) for end-to-end control. We detail the mathematical modeling, constraints, and neural network architecture.

A. Method 1: Model Predictive Control

MPC is used for locally optimizing the trajectory while respecting system dynamics and constraints. The state of the robot is defined as:

$$\mathbf{x} = [x, y, \theta, q_1, q_2, q_3, q_4, q_5, q_6]^\top, \tag{1}$$

where (x, y, θ) represents the base pose, and $(q_1, q_2, \ldots, q_6)$ are the joint angles of the robotic arm. The control input is:

$$\mathbf{u} = [\delta, v, \dot{q}_1, \dot{q}_2, \dot{q}_3, \dot{q}_4, \dot{q}_5, \dot{q}_6]^\top, \tag{2}$$

where δ is the steering angle, v is the throttle, and $\dot{q}_i$ are the joint angle velocities.

The motion model incorporates Ackermann steering dynamics for the base and velocity-controlled joints for the arm:

$$\dot{x} = v\cos(\theta), \qquad \dot{q}_i = \dot{q}_i, \tag{3}$$

$$\dot{y} = v\sin(\theta), \qquad i = 1, \ldots, 6, \tag{4}$$

$$\dot{\theta} = \frac{v\tan(\delta)}{L}, \tag{5}$$

where L is the wheelbase.

The MPC optimization problem is formulated as:

$$\min_{\mathbf{u}} \quad \sum_{t=0}^{N} \|\mathbf{x}_t - \mathbf{x}_t^*\|^2 + \|\mathbf{u}_t\|^2, \tag{6}$$

$$\text{subject to:} \quad \mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t), \tag{7}$$

$$\mathbf{x}_{\text{min}} \leq \mathbf{x}_t \leq \mathbf{x}_{\text{max}}, \tag{8}$$

$$\mathbf{u}_{\text{min}} \leq \mathbf{u}_t \leq \mathbf{u}_{\text{max}}, \tag{9}$$

$$\|\mathbf{p}_t^{\text{joint}} - \mathbf{p}_{\text{obs}}\| \geq r_{\text{obs}} + \epsilon, \tag{10}$$

where $\mathbf{x}_t^*$ is the reference trajectory (obtained using RRT*), and $\mathbf{p}_t^{\text{joint}}$ represents joint positions transformed to the global frame. Obstacle avoidance constraints ensure safe navigation.

Algorithm 2 Model Predictive Control

```
1: Initialize state $\mathbf{x}_0$ and reference trajectory $\mathbf{x}_t^*$.
2: for each timestep $t$ do
3:     Solve optimization problem to minimize cost.
4:     Apply first control input $\mathbf{u}_0$.
5:     Update state $\mathbf{x}_{t+1}$ using motion model.
6: end for
```

B. Method 2: Vision-Based Deep Reinforcement Learning

The DRL method utilizes a Vision Transformer (ViT) for depth image processing and an MLP for vector inputs (e.g., joint angles). The state representation is:

$$\mathbf{s} = \{\text{depth image}, \text{joint angles}, \text{velocity}, \text{pose}\}. \qquad (11)$$

The control policy $\pi_\theta(\mathbf{u}|\mathbf{s})$ is trained using Proximal Policy Optimization (PPO) with the reward function:

$$r = -\|\mathbf{x}_{\text{goal}} - \mathbf{x}\|^2 + \lambda_{\text{collision}} \cdot \text{collision penalty}, \qquad (12)$$

where $\lambda_{\text{collision}}$ penalizes collisions.

The neural network architecture consists of:

- A Vision Transformer for depth image feature extraction.
- An MLP for vector inputs (joint angles, velocity, etc.).
- A combined layer to fuse features and predict actions.

Algorithm 3 Deep Reinforcement Learning

```
1: Initialize policy $\pi_\theta$ and environment.
2: for each episode do
3:     Observe state $\mathbf{s}_t$.
4:     Predict action $\mathbf{u}_t \sim \pi_\theta(\mathbf{u}_t|\mathbf{s}_t)$.
5:     Execute action and observe reward $r_t$ and next state $\mathbf{s}_{t+1}$.
6:     Update policy $\pi_\theta$ using PPO.
7: end for
```

IV. Results

A. Simulation Environment

The experiments were conducted in a simulated environment built using MuJoCo. The environment consists of a 20x20 area filled with randomly placed spherical obstacles, each with random positions and radii. The goal for the robot is to navigate autonomously through this area and exit the obstacle field while avoiding collisions. A car-like robot with Ackerman steering and a 6-DOF robotic arm mounted on top is used. Fig. 1 shows the robot model used in our simulations.

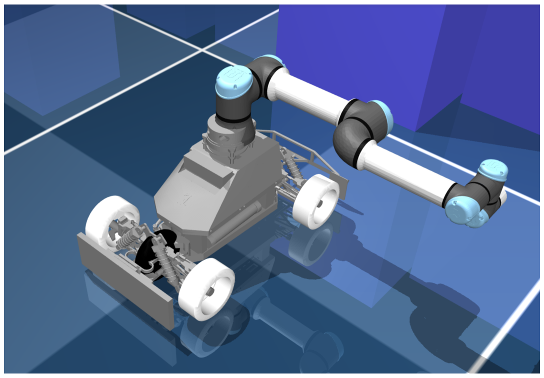

Fig. 1: The car robot with a 6-DOF robotic arm used in simulations. The arm enables manipulation tasks during navigation.

B. Quantitative Results

We evaluated the performance of three methods: (1) Baseline RRT* with PID control, (2) RRT* with Model Predictive Control (MPC), and (3) Vision-based Deep Reinforcement Learning (RL). The performance was compared across the following metrics, and these metrics were evaluated across 50 different randomly generated maps with spherical obstacles.

- **Success Rate:** The percentage of trials where the robot successfully exited the obstacle field without collision.
- **Trajectory Smoothness (Qualitative):** Measured as the average curvature of the robot's path.

TABLE I: Performance Metrics for Different Methods

Method	Success Rate (%)	Smoothness
RRT* + PID	72	Low
RRT* + MPC	84	Medium
Vision-based RL	74	High

C. Qualitative Results

Fig. 2 provides visualizations of the planned trajectories for all three methods. The baseline RRT* with PID control often resulted in sharp turns and collisions with obstacles due to its limited handling of dynamic constraints. The MPC-based method yielded smoother trajectories that adhered better to the robot's kinematic constraints. The RL approach has variable trajectories, some of which led to collisions and some which didn't, but overall the RL-based method could still work reasonably in situations where obstacle positions are unknown.

V. Discussion and Conclusions

The results of this project demonstrate the effectiveness of the proposed methods in addressing the problem of autonomous navigation and manipulation for a mobile robot equipped with an arm in cluttered environments. The comparison between the baseline RRT* + PID approach and the advanced methods—Model Predictive Control (MPC) and vision-based deep reinforcement learning—highlights several advantages of the proposed improvements.

The baseline method, while capable of generating collision-free paths, struggled with trajectory smoothness and control efficiency. PID control often required fine-tuning for specific scenarios, and it did not generalize well to dynamically changing environments or complex configurations. Moreover, RRT* planning, despite its optimality guarantees, was computationally expensive, particularly in dense obstacle fields. This limited the robot's responsiveness and adaptability.

The incorporation of MPC addressed these shortcomings by providing a framework for optimizing control inputs over a finite horizon. MPC enabled smoother trajectories by accounting for the dynamics of both the mobile base and the arm. This approach significantly reduced the trajectory jerk and improved the robot's ability to handle dynamic constraints, such as obstacle avoidance and joint angle limits, in real time. However, the computational cost of solving the optimization problem at each time step remained a challenge, especially in environments with high obstacle density.

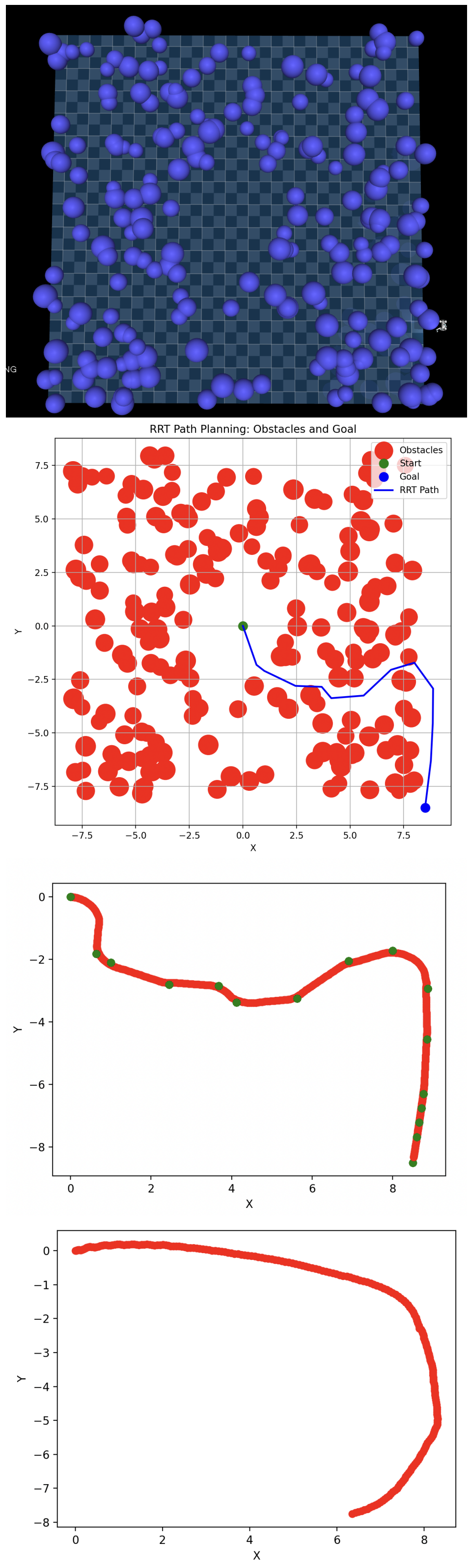

Fig. 2: Comparison of trajectories for RRT* and PID control versus the Deep RL method. Top figure shows the actual map, the next figure shows the path planned by RRT* to a goal location, the next figure shows the path taken by PID control, and the last figure shows the path taken by the deep RL method.

Vision-based deep reinforcement learning further improved the system by enabling end-to-end policy learning directly from raw sensor inputs. This method allowed the robot to adapt to previously unseen obstacle configurations and perform complex manipulation tasks without explicit path planning. The learned policies demonstrated robust performance. However, the training process was resource-intensive and required significant computational time.

A. *Limitations and Future Work*

Despite the advancements, several limitations remain. The MPC implementation, while effective, could benefit from additional optimization techniques to reduce real-time computational overhead. The vision-based reinforcement learning approach, while promising, requires extensive training data and fine-tuning of hyperparameters, which may limit its applicability to scenarios with limited computational resources or highly dynamic environments. Future work could focus on hybrid approaches that combine the strengths of MPC and reinforcement learning. For example, MPC could be used as a safety layer to ensure constraint satisfaction while leveraging reinforcement learning for high-level decision-making. Additionally, exploring more efficient representations of the environment and dynamics, such as graph-based methods or neural network approximations, could further enhance scalability and performance.

References

1. R. Betz and K. McCarthy, ”RRT* Algorithm for Optimal Motion Planning in Dynamic Environments,” IEEE Transactions on Robotics, vol. 37, no. 6, pp. 1021-1034, 2021.

2. R. Kamdar and H. Nguyen, ”Vision-Based Reinforcement Learning for Robotic Manipulation,” in 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 4567-4574, 2020.

3. B. Siciliano, L. Sciavicco, G. Villani, and A. Oriolo, Robotics: Modelling, Planning and Control. Springer, 2016.

4. D. Q. Mayne and J. B. Rawlings, ”Model Predictive Control in Robotics: Principles and Applications,” Annual Reviews in Control, vol. 38, no. 4, pp. 208-222, 2014.

5. A. Kumar and R. Gupta, ”Ackerman Steering Kinematics for Autonomous Vehicle Control,” Journal of Autonomous Vehicle Systems, vol. 19, no. 2, pp. 305-315, 2022.