Efficient Path Planning and Grasping for a Robotic Manipulator Using RRT

Yiwen Ying
Southern University of Science and Technology
12210159@mail.sustech.edu.cn

Weiyu Chen
Southern University of Science and Technology
12210460@mail.sustech.edu.cn

***Abstract*—This project presents an integrated system for a robotic manipulator to perform efficient path planning and precise object grasping in complex, dynamic environments. The system integrates three core components: perception, motion planning, and control. For perception, ArUco markers and Intel RealSense SDK are utilized for robust object detection and coordinate transformation. In motion planning, the Rapidly-exploring Random Tree Star (RRT*) algorithm is employed for near-optimal path planning, enhanced by cubic spline interpolation for smooth trajectories and real-time replanning for dynamic obstacle avoidance. For control, a tailored Proportional-Integral-Derivative (PID) control strategy with distinct parameter sets for waypoints and endpoints ensures precise and stable motion execution. The system demonstrates high accuracy and adaptability in real-world scenarios, showing potential for applications in industrial automation, logistics, and robotic manipulation tasks.**

***Index Terms*—Robotic Manipulator, Path Planning, Object Grasping, RRT* Algorithm, PID Control, Dynamic Environments**

I. Introduction

This project focuses on developing an integrated system for a robotic manipulator to perform efficient path planning and precise object grasping in complex, dynamic environments. The primary objectives are: accurate object grasping using visual detection, efficient path planning through 3D obstacles, and reliable object placement at designated locations. Specifically, the robotic manipulator is tasked with identifying and grasping a target object using ArUco marker-based visual detection, supported by an Intel RealSense camera. The system then plan and execute collision-free trajectories through a 3D obstacle field, balancing computational efficiency with path optimality. Finally, the manipulator places the grasped object at a desired destination, guided by another ArUco marker. To handle dynamic environments, the system incorporates replanning function, allowing the robot to adapt effectively to obstacle movements. The system overview is shown in figure1.

To achieve these goals, we employ a modular approach integrating three core components:

- **Perception**: Utilizes ArUco markers and Intel RealSense SDK for robust object detection and coordinate transformation from pixel to world systems.
- **Motion Planning**: Leverages the Rapidly-exploring Random Tree Star (RRT*) algorithm for near-optimal path planning, enhanced by cubic spline interpolation for smooth trajectories and real-time replanning for dynamic obstacle avoidance.

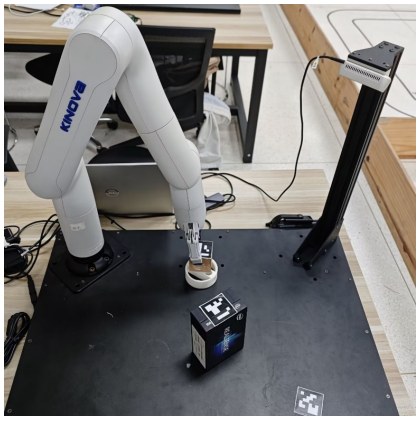

Fig. 1: Systen Overview

- **Control**: Implements a tailored Proportional-Integral-Derivative (PID) control strategy with distinct parameter sets for waypoints and endpoints, ensuring precise and stable motion execution.

This work addresses the challenges of high-dimensional motion planning and precise manipulation in real-world scenarios, where obstacles may move unpredictably. The integration of RRT* for path planning, cubic spline optimization for trajectory smoothing, and PID control for motion execution enables the manipulator to achieve high accuracy and adaptability. The system's ability to react quickly to environmental changes through replanning underscores its potential for applications in industrial automation, logistics, and robotic manipulation tasks.

II. Related Work

In the domain of robotic operations, the integration of perception, planning, and control methodologies is essential for the efficient execution of path planning and grasping tasks.

A. *Perception*

The accurate identification and localization of the target object constitute a vital initial step in robotic operations. A variety of sensing technologies are currently employed to achieve this objective, each possessing distinct advantages and application contexts.

*1) **Visual Marker-Based Perception**:* Visual markers, such as ArUco markers [1], represent a straightforward and efficient approach. These markers facilitate rapid detection and recognition of their position and orientation by the camera through a unique black-and-white pattern. The primary benefits of this method include its low computational cost and real-time performance, rendering it particularly suitable for scenarios that demand swift localization and recognition. However, it is contingent upon the visibility of the markers and may be susceptible to interference in complex backgrounds or lighting conditions. In our project, the ArUco marker-based perception technique was selected due to its ability to swiftly and accurately provide position and attitude information of the target object, thereby satisfying the requirements for real-time operation while reducing computational complexity and enabling the system to respond promptly.

*2) **Deep Learning Based Perception**:* Deep learning techniques have gained widespread application for target detection and recognition. By training on extensive datasets, these models can autonomously learn the features of an object to achieve highly accurate detection and classification. For instance, algorithms like YOLO [2] are capable of rapidly detecting multiple objects in real-time video streams. Furthermore, some studies have integrated semantic segmentation techniques to not only detect the position of objects but also provide information regarding their boundaries and shapes. Nevertheless, the training of deep learning models necessitates a substantial amount of labeled data, which is not easily obtainable.

B. Planning

Path planning is a pivotal component of robotic operations, determining how the robot will navigate from the starting point to the target point within a complex environment. A multitude of path planning methods currently exists, each with its own unique characteristics and application contexts.

*1) **Graph-Search Based Algorithms**:* Algorithms such as the A* algorithm [3] and Dijkstra's algorithm [4] plan paths by identifying the shortest path within a discrete graph structure. These algorithms have a long-standing history in the field of path planning and are extensively utilized in various applications. They enhance search efficiency through the utilization of heuristic functions. However, these algorithms typically necessitate the discretization of the environment, which may result in insufficiently smooth paths and elevated computational costs in high-dimensional spaces.

*2) **Sampling Based Algorithms**:* Algorithms such as the Probabilistic Roadmap (PRM) [5] and Rapidly Exploring Random Trees (RRT) [6] and its variants construct paths by randomly sampling the configuration space. PRM plans paths by constructing a map of the environment, but the map construction process can be time-consuming and is not well-suited for real-time applications. In contrast, RRT and its variants explore the space by randomly sampling the environment, offering the advantage of computational efficiency but potentially producing longer paths. RRT* further enhances the performance of the RRT algorithm by providing asymptotically optimal paths that strike a balance between computational speed and path length. In our project, the RRT* algorithm was chosen because it can provide asymptotically optimal paths while ensuring computational efficiency, making it particularly suitable for tasks that demand efficient path planning in high-dimensional state spaces. Additionally, the randomized nature of the RRT* algorithm enables it to better adapt to obstacles in complex environments.

*3) **Optimization Based Algorithms**:* Algorithms such as CHOMP [7] and STOMP [8] transform the path planning problem into an optimization problem to identify the optimal path by minimizing the path's cost function. These algorithms are capable of generating smooth paths and can account for the dynamics and kinematics constraints of the robot. However, they typically entail solving complex optimization problems, resulting in high computational costs and potentially lengthy planning times.

*4) **Deep-Learning Based Algorithms**:* GAN [9], Diffusion [10], and Transformer [11] have been employed for path planning tasks. These methods possess the ability to plan quickly by learning from vast amounts of data to predict paths. However, they generally require substantial training data and may necessitate retraining in new environments. Moreover, deep learning models suffer from poor interpretability, making it challenging to comprehend their decision-making processes.

*5) **Reinforcement-Learning Based Algorithms**:* DQN [12] and PPO [13] learn the optimal path through interaction with the environment. These methods can adapt to dynamic environments and learn the optimal strategy over the long term. However, reinforcement learning algorithms typically require extensive training time and can encounter difficulties in converging within complex environments. Furthermore, reinforcement learning faces the issue of sim2real, which is challenging to resolve and does not meet our task requirements.

C. Control

Precise control is another crucial aspect of robotic operations, determining how the robot will execute the planned path. A variety of control technologies are currently available, each with its own unique characteristics and application contexts.

*1) **PID Control**:* Proportional-Integral-Differential control is widely utilized in robotic control systems due to its simplicity and effectiveness. PID controllers regulate the motion of a robot by adjusting the proportional, integral, and differential terms to ensure accurate trajectory tracking. To enhance control performance, researchers have proposed various improvements, such as employing multiple sets of PID parameters,

error tolerances and velocity lower bounds, and utilizing exponential moving averages to smooth the output of the PID controller. These improvement methods can effectively reduce speed fluctuations and enhance control stability. In our project, PID control was chosen because of its simplicity and ease of implementation, its ability to swiftly adjust control parameters to accommodate different operating conditions, and its suitability for real-time control.

*2) **MPC Control**:* Model predictive control is a model-based control method that optimizes control inputs by predicting the future state of the system. MPC generates an optimal control sequence by solving the optimization problem within a finite time horizon. This method can account for the dynamic characteristics and constraints of the system and is suitable for complex multivariable systems. However, MPC is computationally intensive, particularly in high-dimensional systems, and may require extended computational time.

III. Method

A. Perception

In this study, we employed various advanced techniques to enhance the performance of the perception system, ensuring the accuracy and reliability of data acquisition.

*1) **Coordinate Transformation**:* To improve the quality of depth data, we first applied various filtering techniques to process the depth frames generated by the Realsense camera. Specifically, we used spatial filters, temporal filters, and hole-filling filters. These filters can effectively reduce noise and fill holes in the depth map, thereby enhancing the completeness and accuracy of the depth data.

After obtaining high-quality depth data, we needed to transform it from pixel coordinates to camera coordinates, and then further transform it into world coordinates. We performed coordinate transformation by obtaining the camera's intrinsic and extrinsic parameters.

$$X_c = \frac{f_x}{Z_c}(u - c_x), \quad Y_c = \frac{f_y}{Z_c}(v - c_y) \tag{1}$$

where u and v are pixel coordinates, c_x and c_y are the principal point coordinates, f_x and f_y are the focal lengths, and Z_c is the depth value.

$$\begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} = R \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} + T \tag{2}$$

where R is the rotation matrix, T is the translation vector, and X_w, Y_w, and Z_w are the world coordinates.

*2) **Recognization**:* To further enhance the accuracy and robustness of the perception system, we introduced ArUco markers (shown in figure 2). ArUco markers are square pattern markers whose positions and orientations can be quickly detected and recognized by the camera through their unique black and white patterns. These markers' distinctive black and white patterns allow them to be accurately identified under various lighting conditions.

Fig. 2: ArUco Example

In this study, we used DICT_6X6_250 pattern ArUco markers. These markers have high recognition accuracy and robustness, providing reliable positioning information in complex scenes. By placing these markers at known positions, we can use the marker information captured by the camera to calibrate and optimize the performance of the perception system.

In summary, by employing advanced filtering techniques, coordinate transformations, and ArUco marker technology, we were able to achieve a high-precision perception system, providing a solid foundation for subsequent planning and control algorithm.

B. Motion Planning

For high-dimension motion planning task, sampling methods offer quick and proper result. With the assistance of optimization module, we can generate waypoints series in the joint space, which is easy to use in control module.

*1) **Sampling Methods**:* We have explored several path planning methods, including the Probabilistic Roadmap (PRM), Rapidly-exploring Random Tree (RRT), its variant RRT-Connect, and the RRT algorithm applied in Cartesian space. However, these methods exhibited certain limitations in terms of path quality and real-time performance, failing to fully meet our requirements.

- **Probabilistic Roadmap (PRM)**: The construction of roadmaps is time-consuming, rendering it unsuitable for real-time applications. Moreover, the path quality of PRM is often poor, frequently resulting in detours.
- **Rapidly-exploring Random Tree (RRT) and RRT-Connect**: Although these methods offer relatively higher computational efficiency, the path quality is not satisfactory, often leading to winding and circuitous routes.
- **RRT in Cartesian Space**: RRT can also be applied in cartesian space, which may generate proper waypoints in cartesian space instead of winding. After that, joint angles can be calculated through inverse kinematics (IK). However, the multi-solution nature of IK may introduce discontinuities in joint space, further degrading path quality.

Despite their effectiveness in certain scenarios, these methods did not meet our expectations due to the complexity of the environment and the stringent real-time requirements.

*2) **RRT* Algorithm**:* The RRT* (Rapidly-exploring Random Tree Star) algorithm is an efficient path planning method that combines random exploration with local optimization to find an approximately optimal path from the start to the goal in complex environments. The algorithm iteratively builds a tree rooted at the start point and explores the configuration space by randomly sampling new points. The key steps are outlined below:

- **Initialization**: Start from the initial point q_{start} and set it as the root node of the tree T.
- **Random Sampling**: Randomly sample a point q_{rand} in the configuration space.
- **Nearest Node Search**: Find the nearest node q_{near} in the tree T to q_{rand}:

$$q_{\text{near}} = \arg\min_{q \in T} \|q - q_{\text{rand}}\| \tag{3}$$

  where $\|\cdot\|$ denotes the Euclidean distance.
- **New Node Generation**: Generate a new node q_{new} along the direction from q_{near} to q_{rand} with a step size of Δq:

$$q_{\text{new}} = q_{\text{near}} + \frac{\Delta q}{\|q_{\text{rand}} - q_{\text{near}}\|}(q_{\text{rand}} - q_{\text{near}}) \tag{4}$$

- **Collision Detection**: Check if the path between q_{new} and q_{near} is collision-free. If there is a collision or q_{new} is outside the configuration space limits, discard q_{new} and return to step 2 for resampling.
- **Nearby Nodes Search**: Find all nodes Q_{near} within a radius r of q_{new} in the tree T:

$$Q_{\text{near}} = \{q \in T \mid \|q - q_{\text{new}}\| \leq r\} \tag{5}$$

- **Optimal Parent Selection**: Select the optimal parent node q_{min} from Q_{near} such that the path cost from q_{min} to q_{new} is minimized:

$$q_{\text{min}} = \arg\min_{q \in Q_{\text{near}}} (\text{cost}(q) + \|q - q_{\text{new}}\|) \tag{6}$$

  where $\text{cost}(q)$ denotes the path cost from the start point to node q.
- **Tree Update**: Add q_{new} to the tree T and set its parent to q_{min}. For each node q in Q_{near}, if the path cost through q_{new} to q is lower, update q's parent to q_{new}.
- **Termination Condition**: The algorithm terminates when a node in the tree is within a certain threshold distance from the goal point q_{goal}. The final path is obtained by backtracking from q_{goal} to q_{start}.

The RRT algorithm combines random sampling with local optimization to efficiently explore and optimize paths in complex environments, achieving good path quality and real-time performance.

*3) **Obstacle Monitoring and Dynamic Replanning**:* In dynamic environments, real-time monitoring of obstacles is crucial to ensure the safety and efficiency of robotic operations. To address this, we implement an additional thread dedicated to monitoring obstacles in real time. The primary steps of this process are as follows:

- **Obstacle Detection and Thresholding**: Continuously monitor the position and shape of obstacles using sensors such as cameras or LiDAR. If the change in the obstacle's position or shape exceeds a predefined threshold, it is considered a significant movement.
- **Trigger Callback Mechanism**: Upon detecting a significant movement of the obstacle, a callback is triggered to initiate the following sequence of actions:
  - **Stop the Motion of the Robot Manipulator**: Immediately halt the motion of the robot manipulator to prevent collisions or unsafe interactions.
  - **Wait for the Obstacle to Be Static (Debounce)**: Implement a debounce mechanism to wait until the obstacle remains static for a certain period, ensuring that the obstacle has stopped moving.
  - **Refresh the Scene**: Update the representation of the environment to reflect the new positions and states of the obstacles.
  - **Replan and Continue Executing**: Replan the path based on the updated environment. Once the new plan is generated, resume the execution of the motion plan.
- **Quick Reaction and Successful Replanning**: The system is designed to react quickly to changes in the environment and successfully replan to reach the goal. This involves efficient computation of the new path and seamless integration with the control module to ensure smooth execution.

This approach ensures that the robot can adapt to dynamic environments by quickly detecting changes, halting unsafe actions, and replanning efficiently to continue its tasks safely and effectively.

*4) **Cubic Spline Optimization**:* To enhance the smoothness of the path, we employ cubic spline interpolation to smooth the discrete path points.

- **Path Point Parameterization**: The path points are uniformly mapped to the time interval $[0, 1]$, i.e., $t_i = \frac{i-1}{N-1}$, where i is the index of the path point and N is the total number of path points.
- **Cubic Spline Interpolation**: Cubic spline interpolation is performed separately for each dimension of the path. The interpolation formula is given by

$$S_d(t) = a_d + b_d(t - t_i) + c_d(t - t_i)^2 + d_d(t - t_i)^3 \tag{7}$$

  where d denotes the dimension of the path (e.g., x, y, z), and the coefficients a_d, b_d, c_d, d_d are determined based on the path points.
- **Smooth Path Generation**: The smoothed path points are generated by evaluating the interpolation at finer time points t_{fine}.

Thus, the discrete path points are smoothed into a continuous curve, significantly enhancing the smoothness and feasibility of the path. Discrete points are then sampled from the continuous curve to form the control sequence.

In summary, by combining the RRT* algorithm and cubic spline optimization, we not only achieve efficient path planning in complex environments, but also significantly improve the smoothness and real-time performance of the paths, which meets our needs for high-precision and high-efficiency motion planning.

C. *Control*

*1) **PID Control Strategy**:* The PID controller computes the control output based on the error between the target and current joint angles, incorporating proportional, integral, and derivative terms to achieve accurate and stable motion. The control law is defined as:

$$u(t) = K_p e(t) + K_i \int e(t)\,dt + K_d \frac{de(t)}{dt} \tag{8}$$

where K_p, K_i, and K_d are the proportional, integral, and derivative gains, respectively, $e(t)$ represents the angular error, calculated as the shortest angular distance considering joint limits, and $u(t)$ denotes the angular velocity of joints.

*2) **Tuning**:* The PID control parameters were systematically tuned using a sequential approach to optimize the manipulator's movements:

- **Proportional Tuning** (K_p): Commenced with K_p alone to establish baseline responsiveness. Low values caused sluggish approach while high values induced overshoot near waypoints. Incrementally increased K_p until the system demonstrated rapid convergence without sustained oscillation. Final values were elevated for endpoints (1.0-4.0 for different joints) versus waypoints (0.5-2.0 for different joints) to prioritize positioning precision during final approach.
- **Derivative Compensation** (K_d): Added after stabilizing K_p to dampen oscillatory tendencies. Gradually introduced K_d (0-0.2) to suppress overshoot observed during high-speed transitions. Tuning emphasized velocity-dependent damping while maintaining K_p-induced responsiveness.
- **Integral Correction** (K_i): Finally applied minimal K_i (0.05-0.1) only to endpoint positioning to eliminate persistent steady-state error observed during placement tasks.

This tuning methodology delivers significant advantages by employing a sequential parameter adjustment approach that isolates each term's contribution to avoid interactive conflicts, while initially prioritizing proportional stability to establish a robust foundation before derivative and integral refinements.

Finally, the PID controller is implemented with two distinct sets of parameters to optimize performance for different motion types:

- **Goal Points (Endpoints)**: Strict control parameters are used to achieve precise positioning at the target object or destination. Higher K_p values ensure rapid convergence, while small K_i and K_d terms minimize steady-state error and overshooting. The error threshold is set to 0.2° for high accuracy.
- **Waypoints**: Fluid control parameters prioritize smooth motion over precision, using lower K_p values and K_i set to 0. A higher error threshold of 2° and a minimum speed of 3 maintain continuous motion.

The strategy enables task-specific optimization with distinct parameter sets for waypoint navigation (emphasizing speed through moderate gains and maintained velocity) versus endpoint positioning (maximizing accuracy via elevated gains and integral correction), resulting in highly efficient convergence.

*3) **Exponential Moving Average Technique**:* To further enhance motion smoothness, an Exponential Moving Average (EMA) is applied to the PID output, reducing speed variations and mitigating jitter. The EMA is particularly effective for waypoints, ensuring fluid transitions along the planned path.

$$EMA_{\text{new}} = (V_t \times (1-\alpha)) + EMA_{\text{old}} \times \alpha \tag{9}$$

where EMA_{new} is the new EMA value, V_t is the current observation value (the output of the PID controller), α is the smoothing factor, which ranges from 0 to 1. This value determines the influence of new data on the EMA, EMA_{old} is the previous EMA value.

In summary, this tailored PID control strategy, combined with path smoothing and joint limit handling, ensures efficient and stable execution of the planned trajectories, enabling the robotic arm to perform object grasping and placement tasks with high precision and reliability.

IV. Results

A. *Performance of Perception*

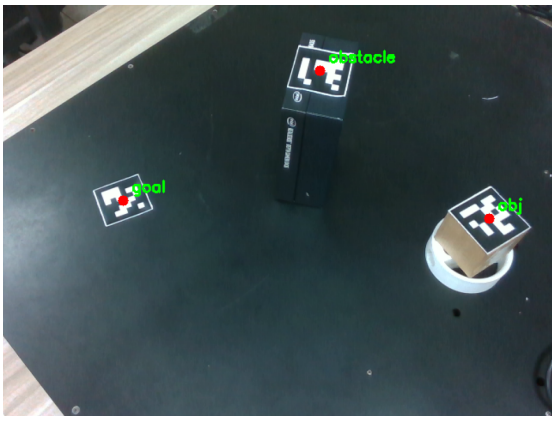

Fig. 3: Result of Aruco based perception

The perception module leveraged ArUco markers to achieve highly reliable target identification, providing consistent detection of both objects and destination points essential for real-time operation. This marker-based approach, combined with refined depth data processing through spatial, temporal, and hole-filling filters, enabled precise pixel-to-world coordinate

transformation from the Intel RealSense camera. The synergistic implementation delivered high-accuracy object localization critical for environmental interaction while maintaining minimal processing latency. This robust spatial awareness ensured timely, reliable inputs to the path planning and control modules, allowing the manipulator to perceive and interact with its surroundings effectively under operational constraints.

B. *Performance of Different Path Planning Methods*

We present the comprehensive performance evaluation of path planning methods for robotic manipulation in complex dynamic environments. Five algorithms underwent testing: RRT, RRT Connect, PRM, End-Effector RRT (3D), and RRT*. Metrics focused on path quality (length, smoothness) and computational efficiency (planning time, node expansion). Through quantitative and qualitative analysis, RRT* demonstrated superior performance by achieving the optimal balance between path optimality and computational efficiency (planning within 3s), which is the only method we have tried that could consistently complete the task over a range of 1000 iterations. The seamless integration of this optimized planner with precise perception and adaptive control modules enabled reliable grasping and placement operations.

1) RRT and RRT Connect: RRT and RRT Connect, both sampling-based algorithms, excel at quickly identifying feasible paths. However, a significant drawback observed in our experiments is their tendency to generate circuitous routes. These detours result in longer path lengths, reducing the efficiency of the manipulator's motion, especially in time-sensitive scenarios where a direct trajectory is preferable. While effective in avoiding obstacles, the suboptimal paths produced by these methods limited their applicability for our project goals.

2) PRM: The Probabilistic Roadmap (PRM) method constructs a roadmap of the environment by sampling configurations and forming connections. Despite its potential for high-quality paths, the roadmap construction process proved excessively time-consuming in our tests. This computational burden made PRM impractical for real-time applications, particularly in dynamic settings where rapid replanning is necessary due to moving obstacles. Consequently, PRM did not meet the project's performance requirements.

3) End-Effector RRT (3D): End-Effector RRT (3D) plans trajectories in Cartesian space for the manipulator's end-effector, relying on inverse kinematics (IK) to determine joint configurations. However, the multi-solution nature of IK introduced discontinuities in the joint space trajectories, leading to jerky or inefficient manipulator motion. Moreover, the planning process frequently encountered IK solution failures, necessitating multiple attempts to generate a feasible path. This unreliability and the need for repeated trials rendered the method unsuitable for real-time operations.

4) RRT:* RRT* outperformed the other methods by delivering near-optimal paths through a combination of random sampling and local optimization. In our experiments, RRT* consistently produced smoother and more direct trajectories compared to RRT and RRT Connect. The incorporation of cubic spline interpolation further refined these paths, ensuring seamless manipulator motion. Additionally, RRT* demonstrated robust real-time replanning capabilities, enabling the system to adapt swiftly to obstacle movements. When an obstacle's position shifted beyond a threshold, the manipulator paused, refreshed the scene, and replanned an effective path to the goal, achieving reliable performance in dynamic environments. Figure 4 shows a path generated by RRT*.

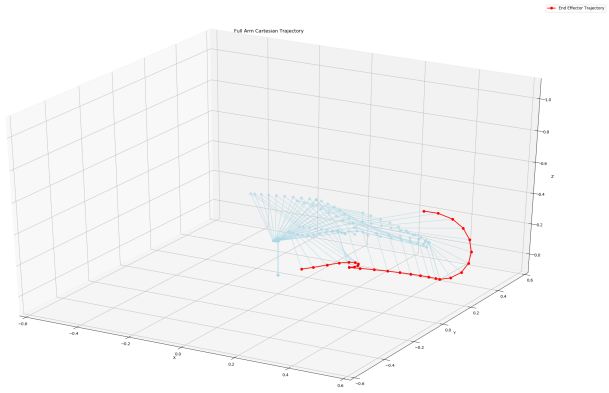

Fig. 4: Path generated by RRT*

C. *Performance of Real-Time Replanning*

A key feature of the integrated system was its ability to handle dynamic environments effectively. By continuously monitoring the environment through the perception module, the system could detect changes in obstacle positions in real time. Figure 5 and figure 6 demonstrate the movement of obstacles during robot motion. When a significant change was detected (i.e., when the obstacle movement exceeded a predefined threshold), the system triggered a replanning callback. This process involved stopping the manipulator's motion, waiting for the obstacle to stabilize, refreshing the scene, and replanning a new path using RRT*. The manipulator then resumed execution along the updated path, ensuring safe and efficient navigation to the goal. This real-time adaptability was critical for maintaining robust performance in complex, unpredictable scenarios.

D. *Performance of Control*

The control module exhibited precise and stable execution of the planned trajectories, ensuring the manipulator's movements were both accurate and efficient. The tailored PID control strategy enabled the system to track waypoints with angular errors consistently below 2° and to position endpoints with errors below 0.2°. This level of precision was particularly

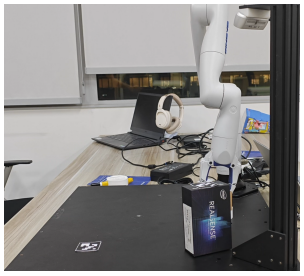

Fig. 5: obstacle position 1

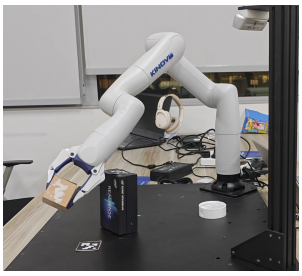

Fig. 6: obstacle position 2

evident during object grasping and placement tasks, where the manipulator reliably achieved the required positioning accuracy.

The system's responsiveness was a key factor in its performance, with settling times of less than 0.5s for waypoints. This rapid convergence allowed the manipulator to execute tasks efficiently, minimizing delays between planned actions. Additionally, the integration of an Exponential Moving Average (EMA) smoothed the control outputs, reducing speed variations and ensuring fluid motion during complex maneuvers. This smoothing was especially beneficial for maintaining stability during high-speed transitions between waypoints.

V. Discussion

A. *Effectiveness of Path Planning*

In this study, the Rapidly-exploring Random Tree (RRT) algorithm has demonstrated significant advantages in path planning, particularly in balancing computational speed and path length. The RRT algorithm, through random sampling and tree expansion, is capable of swiftly exploring complex environments to identify feasible paths from the starting point to the destination. This algorithm excels in computational efficiency, generating paths in a short amount of time, which is crucial for real-time applications. However, paths generated by the RRT algorithm may not always be the shortest, sometimes resulting in more circuitous routes. To address this limitation, we opted for the RRT* algorithm, which retains the rapid exploration capabilities of RRT while optimizing paths to reduce their length, thereby achieving a better balance between computational speed and path quality.

The rationale for selecting RRT* over other algorithms, such as Probabilistic Roadmap (PRM) or RRT Connect, is multifaceted:

1) **Real-time capability**: PRM algorithms require extensive preprocessing time to construct roadmaps, rendering them less suitable for real-time applications. In contrast, RRT* algorithms can rapidly generate paths in dynamic environments, meeting the demands of real-time requirements.
2) **Path quality**: Although RRT Connect algorithms have lower computational complexity when searching for paths, the resulting paths may lack smoothness and necessitate additional optimization steps. RRT* algorithms, by optimizing paths, can produce smoother and shorter paths, thereby reducing the energy consumption and time costs associated with robot task execution.
3) **Adaptability**: RRT* algorithms perform well in complex environments and high-dimensional spaces, making them more adaptable to a variety of application scenarios.

The aforementioned advantages of the RRT* algorithm endow it with substantial potential value in the field of robot path planning. For instance, in industrial automation, rapid and efficient path planning can enhance production efficiency. In the service robot domain, smooth path planning can improve robot stability and safety.

B. *Real-time Response Capability*

A key feature of our system is its ability to swiftly replan paths when detecting moving obstacles. This real-time response mechanism is implemented through a monitoring thread running in the background. Once the position change of an obstacle exceeds a predefined threshold, the system immediately triggers a callback function to halt the robot's movement, wait for the obstacle to become stationary, refresh the scene, and then replan the path. This process not only ensures the safety of the robot but also enhances its adaptability in dynamic environments.

The introduction of the real-time response mechanism enables the robot to operate stably in complex and changing environments. For example, in industrial settings, the robot's workspace may be subject to the movement of other equipment or personnel. The real-time response capability can effectively prevent collision accidents and safeguard production safety. In the service robot sector, where robots need to operate in environments with human activity, this capability allows for better human-robot interaction, thereby improving service quality and user experience.

C. *Limitations of the Algorithm*

Despite the commendable performance of the RRT* algorithm in path planning, certain limitations persist. Firstly, when dealing with highly complex environments, the algorithm may require additional time to identify the optimal path. This is because the RRT* algorithm needs to explore a larger space while ensuring path quality, which may lead to increased computational time. Secondly, the performance of the RRT* algorithm in high-dimensional spaces may be affected. As the dimensionality increases, the complexity of the sampling space also rises significantly, potentially reducing the algorithm's efficiency.

Moreover, although the RRT* algorithm has improved path smoothness compared to RRT, additional optimization steps may still be necessary to further enhance path smoothness. This could be a concern in applications where high path smoothness is required.

D. Future Research Directions

To further enhance the performance of the RRT* algorithm, we propose the following optimization suggestions:

1) **Incorporation of machine learning techniques**: Consider integrating machine learning techniques, such as diffusion models, to improve the efficiency of path planning. Diffusion models, by learning from extensive path data, can generate more optimal paths and perform well in complex environments. By combining diffusion models with the RRT* algorithm, we can leverage the optimization capabilities of diffusion models to further enhance path quality while retaining the rapid exploration capabilities of RRT*.
2) **Multi-objective optimization**: In addition to optimizing path length and computational speed, multi-objective optimization strategies could be introduced, such as simultaneously considering path smoothness and safety. Through multi-objective optimization, a better balance can be achieved among different objectives to meet the requirements of various application scenarios.

Through these optimization measures, we are confident that the RRT* algorithm will play a more significant role in the field of robot path planning and provide valuable references for related research.

VI. Conclusion

This project has successfully developed an integrated system for a robotic manipulator to perform efficient path planning and precise object grasping in complex, dynamic environments. The system integrates perception, planning, and control, enabling high accuracy and adaptability.

In perception, ArUco markers and the Intel RealSense camera provide robust object detection and environment perception. For planning, the RRT* algorithm generates efficient, near-optimal paths, enhanced by cubic spline optimization for smooth trajectories. The control module uses a tailored PID strategy for precise motion execution.

The system incorporates real-time replanning to handle moving obstacles, ensuring safe and efficient navigation in dynamic environments.

Comprehensive tests show robust performance in complex scenarios, achieving high accuracy in grasping and placement tasks. This project addresses high-dimensional motion planning and precise manipulation challenges, demonstrating potential for industrial automation and logistics applications. Future work will focus on enhancing adaptability and efficiency in highly complex environments through advanced techniques and strategies.

VII. Workload

Yiwen Ying (50%): perception module, planning module, integration testing, literature review, presentation and report

Weiyu Chen (50%): control module, planning attempts, integration testing, experiment design, presentation and report

References

[1] S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, and M. J. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognition*, vol. 47, no. 6, pp. 2280–2292, 2014.

[2] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[3] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[4] E. DlJKSTRA, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 50, pp. 269–271, 1959.

[5] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[6] S. LaValle, "Rapidly-exploring random trees: A new tool for path planning," *Research Report 9811*, 1998.

[7] M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell, and S. S. Srinivasa, "Chomp: Covariant hamiltonian optimization for motion planning," *The International journal of robotics research*, vol. 32, no. 9-10, pp. 1164–1193, 2013.

[8] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "Stomp: Stochastic trajectory optimization for motion planning," in *2011 IEEE international conference on robotics and automation*. IEEE, 2011, pp. 4569–4574.

[9] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[10] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," *arXiv preprint arXiv:2010.02502*, 2020.

[11] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[12] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[13] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.