Optimize Path Planning System In A Given Network Using A* And RRT Algorithm

Naazneen[1],Kotakonda Madhubabu[2],Dr.CRKReddy[3]
P.G Scholar in the DepartmentofComputerScience&Engineering,MahatmaGandhiInstituteofTechnology,Hyderabad,India.
Assistant Professorin the Department of ComputerScience&Engineering,Mahatma Gandhi Institute of Technology,Hyderabad,India.
Professor in the Department of Computer Science & Engineering,Mahatma Gandhi Institute of Technology ,Hyderabad,India.
Corresponding email: *naazneen_pg23csea906@mgit.ac.in,kmadhubabu_cse@mgit.ac.in,crkreddy_cse@mgit.ac.in,

Abstract:Path planning plays a pivotal role in autonomous systems, enabling them to navigate safely and effectively through environments that may be complex, dynamic, or partially unknown. This study introduces a hybrid path planning approach that merges the global search efficiency of the A* algorithm with the dynamic adaptability of the Rapidly exploring Random Tree (RRT) method. The aim is to develop a robust and scalable framework capable of generating smooth, collision-free paths for autonomous agents such as mobile robots or autonomous vehicles while responding in real time to both static and moving obstacles. The proposed system is built upon a graph-based architecture, where nodes represent navigable waypoints and edges indicate feasible transitions between them. The input stage incorporates key constraints including vehicle dynamics, route limitations, environmental obstacles, and predefined start and target locations. Initially, the A* lgorithm is applied to compute a globally optimal path using heuristics like Manhattan distance, ensuring computational efficiency within structured environments.
To enhance flexibility in unstructured or changing conditions, the mework integrates RRT as a refinement tool for local re-planning. Unlike A*, which operates over a grid, RRT performs randomized exploration of the configuration space, making it more suitable for dynamic and continuous domains. It refines the initial path produced by A* by expanding nodes adaptively, allowing for real-time obstacle avoidance and path smoothing. An important feature of the system is its path optimization layer, which evaluates path quality based on smoothness, feasibility, and overall efficiency. Methods such as Bézier curves or spline fitting are utilized to transform rough path segments into smooth, navigable trajectories. Additionally, graph pruning techniques are applied to eliminate unnecessary waypoints, reducing complexity while maintaining the path's validity and safety.

Keywords-PathPlanning,Autonomous Navigation,A* Algorithm,Rapidly-exploring Random Tree (RRT),Hybrid Path Planning

INTRODUCTION

Path planning is a critical function in autonomous driving systems, responsible for ensuring that vehicles can navigate safely, efficiently, and intelligently through complex environments. As these vehicles operate in dynamic and often unpredictable conditions, it becomes essential to employ planning strategies that support real-time decision-making and adaptability to ensure smooth, collision-free travel. Traditional path planning algorithms such as Dijkstra's and A* have long been used to enerate optimal routes based on static maps and heuristic aluations [16]. These algorithms perform well in structured environments, offering fast and reliable solutions. However, their effectiveness diminishes in dynamic settings where obstacles, traffic, or environmental conditions change rapidly. To address this, more advanced planning approaches such as hybrid models, reinforcement learning have been developed to improve robustness and responsiveness [3]. This study focuses on a hybrid path planning framework that combines the deterministic strengths of the A* algorithm with the adaptability of the Rapidly exploring Random Tree (RRT) method [17]. A* is particularly effective for generating globally optimal paths in structured environments, thanks to its heuristic-guided search capabilities. In contrast, RRT is designed for unstructured and dynamic scenarios, using randomized sampling to explore the configuration space quickly and handle unexpected obstacles.

By integrating A* and RRT, the hybrid system leverages both optimality and flexibility. A* provides an initial path through the known environment, while RRT refines this path locally to account for real-time changes such as moving obstacles or altered terrain. This combination enhances path smoothness, improves obstacle avoidance, and increases adaptability without compromising computational efficiency [9]. The proposed framework also incorporates real-time data sources to support intelligent route planning. Additional optimizations like RRT* and cost function optimization are considered to further improve path quality and convergence speed [21]. The overall goal is to develop a path planning system that supports safe, smooth, and efficient autonomous navigation across both structured and unstructured environments.This study evaluates the effectiveness of the proposed hybrid model through simulations and real-world scenarios, aiming to demonstrate its capability to meet the demands of modern autonomous driving systems.**Problem Statement**In the field of autonomous navigation, devising a reliable and collision-free path for vehicles within a defined network remains a significant challenge, especially in environments that are dynamic, complex, or only partially known. Conventional path planning methods such as A* are well-suited for static and structured settings, offering deterministic and globally optimal paths[25]. However, these algorithms often lack the flexibility needed for real-time adjustments and produce rigid paths that may not be suitable for continuous, real-world navigation. In contrast, sampling-based approaches like Rapidly exploring Random Tree (RRT) are more adaptable in high-dimensional and unstructured environments[17]. They allow for quick exploration and are better equipped to handle uncertainty and dynamic changes. Nonetheless, RRT typically results in non-optimal, irregular paths and demands substantial computational power for real-time use.Real-world navigation further complicates path planning with additional factors such as moving obstacles, vehicle kinematic constraints [18]. Relying solely on either A* or RRT in such scenarios can lead to inefficient or impractical trajectories that compromise safety and performance. There is, therefore, a pressing need to develop a hybrid path planning solution that merges the strengths of A* its global structure and optimality with the local adaptability and efficiency of RRT. This system must also integrate real-time input, support trajectory refinement techniques, and optimize computational resources. The ultimate objective is to produce safe, smooth, and dynamically feasible paths that adapt to changing environments and operational constraints.

Core Challenges Include: Balancing Optimality and Real-Time Responsiveness

A* delivers optimal routes but may become computationally expensive in dense or large-scale environments [25].

RRT handles complex spaces more efficiently but typically generates irregular, non-optimal paths.

Combining both requires strategic tuning to ensure speed without sacrificing quality.Ensuring Path Smoothness and Physical

Feasibility
Paths from A* may result in abrupt angles due to grid-based layouts.
RRT-generated paths can be uneven or erratic due to random sampling.
Smoothing techniques such as spline interpolation or Bézier curves are essential for generating realistic, drivable trajectories [9].

Objectives
The main objective of this study is to develop and implement an optimized, adaptive path planning framework that integrates the deterministic accuracy of the A* algorithm with the exploratory strength of the RRT method. The system aims to achieve the following goals:

a. Efficient Path Generation
Develop algorithms that compute safe, cost-effective paths for autonomous agents across both static and dynamically changing environments with varying levels of complexity [6].

b. Scalability Across Large Networks
Design the framework to operate effectively in expansive networks with diverse topologies, ensuring performance does not degrade as network size increases [23].

c. Real-Time Adaptability
Enable the system to respond promptly to dynamic elements such as moving obstacles, unexpected blockages, or sudden environmental changes [18].

d. Hybrid Strategy Integration
Combine the structured, goal-oriented search capabilities of A* with the exploratory and randomized search nature of RRT to optimize path planning in both known and uncertain regions [13].

e. Smooth and Feasible Trajectories
Ensure that the generated paths are continuous, smooth, and kinematically feasible, making them suitable for real-world robotic or vehicular motion constraints [15].

f. Support for Real-Time Execution
Incorporate techniques such as parallel processing, waypoint reduction to facilitate rapid planning and re-planning in real-time operational scenarios.

Existing System and Limitations

i. A*-Based Systems
Deterministic Graph Search: A* is widely applied in structured, well-defined environments such as grid maps, where nodes represent discrete positions [20].
Heuristic-Driven Planning: It employs admissible heuristics like Manhattan or Euclidean distance to efficiently compute the shortest path.
Best for Static Maps: Commonly used in 2D navigation tasks for robots or autonomous vehicles operating in known, static environments.

ii. RRT-Based Systems
Sampling-Based Strategy: Rapidly exploring Random Tree (RRT) is designed for exploration in high-dimensional or continuous spaces, making it suitable for complex terrain.
Probabilistically Complete: Although it doesn't guarantee optimal paths, RRT is effective at covering large or unknown areas rapidly [23].
Algorithm Variants: RRT-Connect simultaneously grows trees from both start and goal positions to improve the speed of convergence and path discovery.

iii. Hybrid Planning Approaches
Global-Local Division: In semi-structured environments, RRT is often used to generate a broad, initial path, while A* is applied locally for fine-tuning and optimization [11].
Heuristic-Driven RRT: A* can influence or constrain RRT's random expansion, effectively guiding it toward the goal.
Hierarchical Planning Frameworks: High-level paths are generated using RRT, and refined with grid-based algorithms like A* to ensure better handling of obstacles and tighter areas.

Limitations of Existing Systems
A* Algorithm

i. Limited Scalability: Suffers performance degradation in large-scale or high-dimensional spaces due to exhaustive exploration.

ii. High Memory Usage: Needs to maintain open and closed lists, leading to memory constraints in dense or complex environments.

iii. Lacks Real-Time Flexibility: Not naturally suited for dynamic scenarios; replanning is time-consuming and computationally expensive[4].

iv. Dependent on Grid Resolution: Path quality and speed vary significantly with grid size—finer grids yield better paths but increase computational overhead.

RRT Algorithm

i. Sub-Optimal Path Quality: Basic RRT does not ensure the most efficient or smooth path; often results in jagged, indirect routes[9].

ii. Inconsistent Output: The inherent randomness can produce different results each time, reducing predictability and reliability.

iii. Struggles in Tight Spaces: Has difficulty navigating through narrow corridors or tight constraints due to sparse sampling.

iv. Needs Additional Processing: Typically requires further steps like smoothing, pruning, or re-evaluation before the path can be executed.

Proposed System And Advantages

The proposed path planning framework integrates both structured and flexible navigation strategies to enable efficient, adaptive, and collision-free movement of autonomous agents in diverse environments. It combines the strengths of A* and RRT algorithms for global and local planning, respectively, allowing for real-time responsiveness and smooth trajectory generation.

1. Environment Modeling

A* operates on a discrete grid-based map, ideal for structured spaces where waypoints and paths are predefined.

RRT utilizes a continuous configuration space, making it suitable for unstructured or dynamically changing regions.

Obstacles are defined as non-traversable zones or moving objects, depending on whether the environment is static or dynamic[12].

2. Global Path Planning using A*

A heuristic-guided search algorithm (e.g., Manhattan distance) is employed to compute an initial optimal path[16].

The algorithm avoids known obstacles and incorporates dynamic cost maps to reflect real-time changes in the environment.

This phase is responsible for ensuring a globally optimal and safe trajectory under predictable conditions.

3.Local Path Refinement using RRT

RRT enhances the initial path by refining segments located in complex or dynamic areas[14].

A goal-biased sampling approach is used to increase efficiency, guiding the tree toward the target location more rapidly.

Features such as collision checking and adaptive step sizing are incorporated to ensure smooth and feasible paths even in tight spaces.

Advantages of the Proposed System

i. Optimal Pathfinding in Structured Areas

A* efficiently generates the shortest possible path in well-defined, grid-based environments, reducing total traversal cost[2].

ii. Robust Navigation in Cluttered or Dynamic Regions

RRT effectively explores high-dimensional or unstructured spaces where deterministic planners often fail[1].

iii. Improved Convergence via Heuristic Guidance

The use of A*-derived information to bias RRT sampling enhances convergence speed and minimizes random deviations[9].

iv. Efficient Resource Utilization

Strategies like guided sampling, selective A* expansion, and obstacle-aware pruning minimize unnecessary exploration and reduce computational overhead—crucial for large or complex environments[4].

v. Smooth and Executable Paths

Final trajectories are refined using methods such as B-spline or curve interpolation, ensuring they are physically feasible and ready for real-world deployment in autonomous systems[7].

1. Adaptability Across Diverse Terrain

The hybrid system seamlessly adjusts to both urban, grid-like environments and irregular, unstructured terrains without the need for major structural changes.**LITERATURE SURVEY**

i. Mohamed Reda,Ahmed Onsy ,Amira Y. Haikal,Ali Ghanbaria, “Path planning algorithms in the autonomous driving system: A comprehensive review” Robotics and Autonomous Systems (2024)

This comprehensive review focuses on the Autonomous Driving System (ADS), which aims to reduce human errors that are the reason for about 95% of car accidents. The ADS consists of six stages: sensors, perception, localization, assessment, path planning, and control. We explain the main state-of-the-art techniques used in each stage, analyzing 275 papers, with 162 specifically on path planning due to its complexity, NP-hard optimization nature, and pivotal role in ADS. This paper categorizes path planning techniques into three primary groups: traditional (graph-based, sampling-based, gradient-based, optimization-based, interpolation curve algorithms), machine and deep learning, and meta-heuristic optimization, detailing their advantages and drawbacks.

ii. Zhixian.Li,Nianfeng Shi,Liguo zhao,Mengxia zhang, “Deep Reinforcement learning path planning and task allocation for multi-robot collaboration” Alexandria Engineering Journal(2024)

In the current technological landscape, Multi-Robot Systems (MRS) have become crucial for complex tasks, with applications in industrial automation, search and rescue, and intelligent transportation. However, existing techniques face challenges in path planning and task allocation, particularly regarding adaptability, real-time decision-making, and efficiency. Deep Reinforcement Learning (DRL) has emerged as a promising solution due to its robust learning capabilities. To address these challenges, we propose an innovative DRL-MPC-GNNs model that integrates Deep Reinforcement Learning, Model Predictive Control (MPC), and Graph Neural Networks (GNNs). Our model aims to optimize path planning and task allocation in multi-robot systems.

iii. Rohan Inamdar,S.Kavin Sundarr,Deepan Khandelwa,Varun dev sahu,nitish katal, “ A comprehensive review on safe- reinforcement learning for autonomous vehicle control in dynamic environments” e-prime-advances in electrical engineering, electronics and energy(2024)

Autonomous vehicles (AVs), commonly known as self-driving cars, constitute a radical advancement in modern transportation systems. These vehicles have sophisticated sensor arrays, cameras, and state-of-the-art information processing systems that enable them with the capability to perceive their surroundings, make real-time decisions, and navigate independently. The deployment of autonomous vehicles holds the promise of enhancing road safety, reducing traffic congestion, facilitating more effortless mobility for individuals with disabilities, and promoting energy efficiency.

iv. Ali Enes Dingil,Ondrej Pribyl, “Understanding state of the art situation of transport planning strategies in earthquake prone area by using AI-Supported literature review methodology” Heliyon(2024)

This review aims to explore earthquake-based transport strategies in seismic areas, providing state-of-the-art insights into the components necessary to guide urban planners and policymakers in their decision-making processes.The review provides a variety of

methodologies and approaches employed for the reinforcement planning and emergency demand management to analyze and evaluate the impact of seismic events on transportation systems, in turn to develop strategies for preparedness, mitigation, response, and recovery phases.

v. S Alagumuthu krishnan,s.deepajothi ,rajasekar vani,s velliangiri, " Reliable and Efficient Lane Changing Behaviour for Connected Autonomous Vehicle through Deep Reinforcement Learning" Procedia computer science(2023)

The establishment of future intelligent transport systems is dependable on the reliable and seamless function of Connected and Autonomous Vehicles (CAV). Reinforcement learning (RL), which allows autonomous vehicles (AVs) to learn an ideal driving strategy through constant contact with the environment, plays a significant part in the decision-making process of autonomous driving (AD). The networking of CAV is advantageous since it allows for the transmission of traffic-related data to vehicles via Vehicle-to-External (V2X) communication. Recognition and anticipation of driving behaviour are critical for avoiding collisions because they can provide useful information to other drivers and vehicles.

3.DESIGN METHODOLOGY OF OPTIMIZE PATH PLANNING SYSTEM IN A GIVEN NETWORK

3.1 DESIGN METHODOLOGY

To develop a hybrid A*–RRT path planning system for autonomous navigation, a structured methodology is followed, consisting of problem definition, environment modeling, system architecture, module design, testing, and iterative improvements.

i. Problem Definition

Goal: To generate an efficient, smooth, and collision-free path from a given start point to a destination within a partially known or dynamically changing environment.

ii. Environment Modeling

The environment is represented using occupancy grids or point clouds for accurate spatial planning.

The space is categorized as:

Structured areas: Well-defined zones where grid-based algorithms like A* are effective.Unstructured or dense areas: Complex regions where RRT is better suited due to its exploratory nature

iii. Environment Design

The system follows a modular and adaptive architecture integrating A* and RRT based on environmental complexity.It starts with inputs such as map data (grid) and start-goal coordinates.An environment classification module analyzes the map to determine static or dynamic zones.In structured, static regions, A* is used with heuristic functions to compute optimal paths.

In dynamic zones, RRT is employed to explore feasible paths using randomized sampling.

iv. Module Implementation

A* Module: Computes deterministic paths over grid maps using cost-based heuristics.

Heuristic Biasing Module: Guides RRT expansion using cost-to-go values from A*, enhancing convergence speed and direction.

RRT Module: Handles complex, high-dimensional path planning using a probabilistic tree-building approach.

Post-Processing Module: Replanning Module, Continuously monitors the environment and triggers local re-planning in response to new obstacles or changes.

v. Validation and Testing Strategy

Simulation Tools: Systems like ROS(Robot Operating System) with Gazebo, or Webots, are used to simulate autonomous navigation in predefined maps.Performance Indicators:Success rate of path completion, Path smoothness and length, Responsiveness to environmental changes.

vi. Iterative Development and Optimization

Phase 1: Independently test A* and RRT in controlled environments to evaluate their baseline performance.

Phase 2: Integrate both algorithms into a hybrid framework; fine-tune the heuristic biasing mechanism for optimal coordination.

Phase 3: Conduct experiments in complex and dynamic environments to analyze trade-offs between speed, optimality, and smoothness.

Phase 4: Enhance performance by applying parallel processing techniques or hierarchical planning structures for large-scale deployment.

3.2 SYSTEM ARCHITECTURE

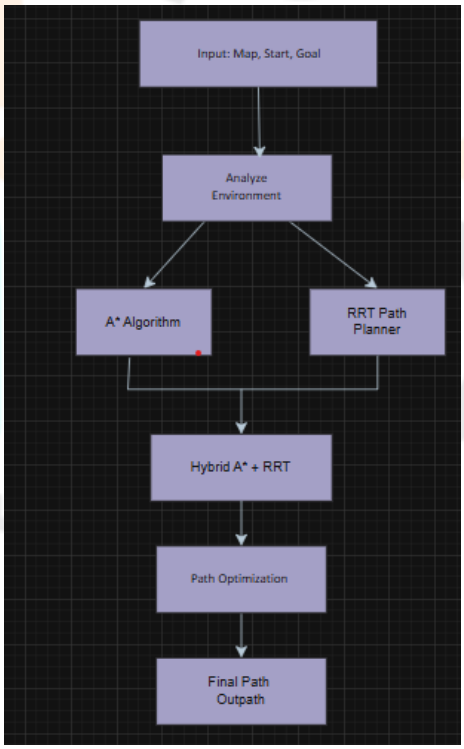

Figure 3.1: System Architecture Of Optimize Path Planing System In A Given Network

The Above Figure 3.1 Explain about System Architecture Of Optimize Path Planing System In A Given Network:

The proposed hybrid path planning system operates through a structured pipeline that intelligently selects between A* and RRT algorithms based on the nature of the environment. The system consists of six major components, each responsible for a critical step in generating a smooth and feasible path.

i. Input Acquisition: Map, Start, and Goal

The system begins by accepting the following inputs:

A representation of the environment, which is a grid map.

Defined start and target positions for the autonomous agent.

These inputs are used to construct the planning workspace where path computation will occur.

ii. Environment Assessment

In this stage, the system analyzes the characteristics of the environment to determine the appropriate planning strategy:

If the environment is structured , it is best handled using deterministic planning.

If the environment is unstructured, highly dynamic, or unknown a sampling-based method is more effective.

This classification guides the system's choice between A* and RRT.

grid = np.zeros((30, 30), dtype=int)

This creates a 30 × 30 grid map where each cell has a value:

0 → free cell (no obstacle)

1 → occupied cell (an obstacle or wall)

Each cell behaves like a pixel on a map, but for path planning purposes. The robot can only move through cells where the value is 0.

X and Y Axes (Detailed Orientation)

grid[y][x]: NumPy arrays use row (Y) first, then column (X).

Y-axis ↑ is the vertical axis (rows)

X-axis → is the horizontal axis (columns), So:

(0,0) = top-left corner of the grid

(29,29) = bottom-right corner

This structure means:

Y-axis = 0 to 29 from top to bottom in matrix terms

X-axis = 0 to 29 from left to right

iii. Path Planning Using A* (Structured Regions)

When dealing with predictable and well-defined areas:

The A* algorithm is employed to perform a grid-based search using a manhattan distance such as distance or risk.It calculates the shortest and most efficient path by evaluating nodes based on a heuristic.This method yields precise and optimal routes suitable for environments with low uncertainty.

iv. Path Planning Using RRT (Unstructured or Dynamic Areas)

For regions with complex, high-dimensional layouts:

The RRT algorithm explores the space by randomly sampling points and expanding a tree structure toward the goal.

While not guaranteeing an optimal path, RRT is effective at rapidly discovering feasible routes in challenging conditions.It is particularly useful for navigating narrow passages, cluttered environments, or areas with dynamic elements.

v. Hybrid A* + RRT

Purpose: Combines results from both A* and RRT.

Details:

Merges global planning from A* with local adaptability from RRT.

Enhances overall path feasibility, especially in mixed or dynamic environments.

vi. Path Fusion and Optimization

Once individual segments are generated:

The system merges the outputs from A* and RRT based on region boundaries or planning context.

The combined path is passed through an Optimization Module that: Eliminates redundant waypoints or unnecessary detours.

Applies smoothing algorithms such as Bézier curves or B-splines.

Verifies that the trajectory meets the kinematic constraints of the vehicle or robot, ensuring safe and executable motion.

vii. Final Path Generation and Execution

The final output of the system is:

A set of optimized waypoints or a continuous trajectory.

A visual representation suitable for simulation platforms like RViz or Gazebo.

A path ready to be executed by the robot's motion controller, enabling smooth and responsive navigation in the real world.

4.IMPLEMENTATION

This code uses a modular structure with two main planning classes:

i. Path Planning Framework

A*: Handles deterministic path planning on a grid

RRT: Locally refines A* segments using randomized sampling

These modules operate on a shared grid map and return path waypoints to be visualized at the end.

ii. Grid Environment Setup

python

grid = np.zeros((30, 30), dtype=int)grid[10:25, 15] = 1grid[5:15, 10] = 1

A 30 × 30 NumPy grid represents the environment.

0 denotes free space; 1 marks obstacles.

Obstacle layout:

Vertical wall along column 15 from row 10 to 24

Horizontal wall along row 10 from column 5 to 14

This creates a partial "L"-shaped barrier for path testing.

iii. Start and Goal Configuration

python

start = (0, 0)goal = (29, 29)

Start = top-left corner

Goal = bottom-right corner

Coordinates follow NumPy format: (row, column) → (y, x)

iv. A* Class: Deterministic Planning

Key Components:

heuristic(a, b): Uses Manhattan distance for cost estimation

get_neighbors(node): Expands four directions (up/down/left/right)

run(): Core loop using a min-heap via heapq to find lowest-cost path

Behavior:

Explores space using cost function f(n) = g(n) + h(n)

Backtracks through came_from to generate the path

v. RRT Class: Local Refinement

Key Components:

get_random_node(): Uniform sampling over grid

nearest(node): Finds closest tree node to connect

steer(from, to): Takes one step toward target node

run(): Builds a tree incrementally; if it connects to goal, path is returned

Behavior:

Runs for a max of 200 iterations per segment

Avoids obstacles via collision checks (grid[new_node] == 0)

Constructs sub-paths for A* segments

vi. Hybrid Execution Logic

Iterates over each A* segment

Refines each (A* segment) using local RRT planning

If RRT fails (can't find connection), falls back to using A* segment directly

The final path is a combination of A*-defined structure + locally improved steps

vii. Visualization with Matplotlib

Visual elements:

Obstacles → grayscale (cmap='Greys')

Start → green circle ('go')

Goal → red circle ('ro')

A* Path → blue dashed line ('b--')

Optimized (RRT-refined) Path → green solid line ('g-')

The plot clearly shows:

How A* navigates around the obstacles

How RRT potentially smooths or re-routes jagged A* segments.

Standard Algorithm Structure:

Algorithm: Hybrid A* + RRT Path Planning

Step 1: Start

Step 2: Initialize grid, start node, and goal node

Step 3: Mark obstacles in the grid

Step 4: Run A* algorithm to compute initial path from start to goal

Step 5: If A* finds a path, proceed to refinement

Step 6: For each segment in the A* path:

a. Initialize RRT with segment start and goal

b. For a fixed number of iterations:

i. Sample a random node

ii. Find nearest node in the tree

iii. Steer toward the random node

iv. If new node is valid and goal is reached → reconstruct path

c. If RRT segment succeeds → append refined segment

d. Else → append original A* segment

Step 7: Combine all segments into a refined path

Step 8: Visualize the grid, A* path, and refined path

Step 9: Return the final optimized path

Step 10: Stop

5. RESULTS AND DISCUSSION

Step 1:Comparing Hybrid A *+ RRT and Basic A* algorithms in autonomous path planning:

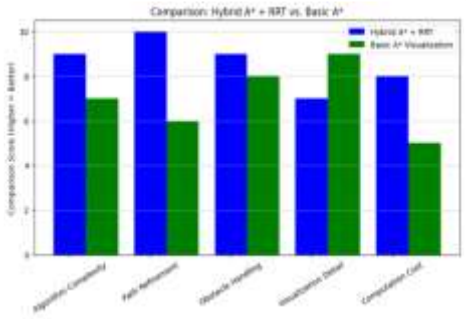

Figure 5.1 comparing Hybrid A *+ RRT and Basic A* algorithms in autonomous path planning

The Above Figure 5.1 explain about the bar graph comparing Hybrid A* + RRT and Basic A* algorithms in autonomous path planning:

This graph provides a side-by-side evaluation of the Hybrid A*+ RRT algorithm and the traditional A* algorithm across six performance categories. X-Axis Categories (Performance Metrics).The Y-axis indicates a score ranging from 0 to 10, with higher values representing better outcomes.

Category Breakdown:

i. Algorithm Complexity

Hybrid A* + RRT: Receives a higher score (~9) due to its combined use of deterministic (A*) and sampling-based (RRT) planning methods.

Basic A*: Scores lower (~7) because it relies on a simpler heuristic-driven approach.

ii. Path Refinement

Hybrid A* + RRT: Achieves top marks (~10) as it produces refined, smooth trajectories using techniques like Bézier curves or spline interpolation.

Basic A*: Scores (~6), often generating sharp, grid-aligned turns that may require further processing.

iii. Obstacle Handling

Hybrid A* + RRT: Performs strongly (~9), especially in unpredictable or obstacle-rich environments thanks to RRT's flexible sampling.

Basic A*: Slightly less adaptable (~8), functioning well in static setups but not ideal for dynamic spaces.

iv. Visualization Clarity

Hybrid A* + RRT: Moderate score (~7), as the complexity of paths may reduce visual clarity.

Basic A*: Scores highly (~9) due to its structured, easy-to-understand output paths.

v. Computation Cost

Hybrid A* + RRT: Higher (~8), since combining two algorithms increases the demand on system resources.

Basic A*: Lower (~5), offering faster processing with minimal computational load.

Step 2:Comparison: Hybrid A* + RRT vs. Standalone RRT

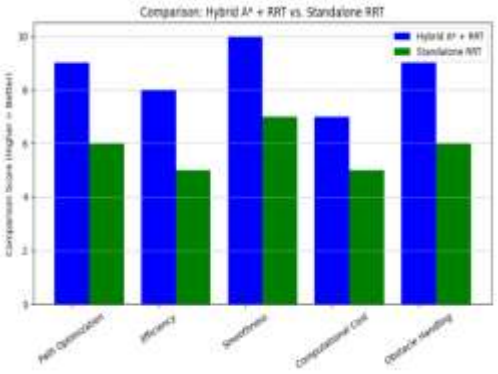

Figure 5.2 Comparison: Hybrid A* + RRT vs. Standalone RRT

The Above Figure 5.2 Comparison: Hybrid A* + RRT vs. Standalone RRT

This graph illustrates a side-by-side evaluation of two path planning approaches — the Hybrid A* + RRT algorithm (shown in blue) and the Standalone RRT algorithm (shown in green).

Y-Axis (Performance Score):

Ranges from 0 to 10, where higher scores reflect better effectiveness.

Each score is based on performance in a specific planning criterion.

X-Axis (Performance Criteria):

i. Path Optimization

Hybrid A* + RRT: Achieves a higher score (~9) by combining the precision of A* with RRT's exploratory capability to produce efficient routes.

Standalone RRT: Scores lower (~6) due to its tendency to create longer, less efficient paths from random sampling.

ii. Efficiency

Hybrid Approach: Performs well (~8) by guiding RRT using A*'s heuristics, reducing exploration overhead.

Standalone RRT: Less efficient (~5), as it may repeatedly sample irrelevant regions in the environment.

iii. Smoothness

Hybrid A*+ RRT: Excels (~10), as the raw paths are refined using methods like Bézier curves or splines.

Standalone RRT: Less smooth (~5), since paths are often jagged unless specifically refined.

iv. Computational Cost

Hybrid Method: Slightly higher (~7) due to the combined algorithm structure and additional smoothing steps.

Standalone RRT: Lower (~5), offering quicker results but at the expense of path quality.

v. Obstacle Handling

Hybrid A* + RRT: Strong performance (~9), especially in cluttered or changing environments.

Standalone RRT: Moderate (~6), as it handles obstacles but with less precision or consistency.

Step 3:Comparison: Efficiency-Enhanced Hybrid A* + RRT vs. A* vs. Standalone RRT

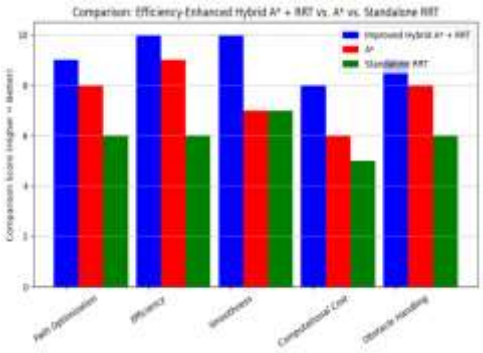

Figure 5.3 Comparison: Efficiency-Enhanced Hybrid A*+ RRT vs. A* vs. Standalone RRT

The Above Figure 5.3 Comparison: Efficiency-Enhanced Hybrid A* + RRT vs. A* vs. Standalone RRT

This bar chart offers a detailed performance comparison of three path planning strategies used in autonomous systems:

Improved Hybrid A* + RRT (Blue)

Basic A* Algorithm (Red)

Standalone RRT (Green)

Y-Axis (Performance Score):

The vertical axis shows scores from 0 to 10, indicating how well each algorithm performs.

A higher score reflects better performance in each criterion.

X-Axis (Evaluation Criteria):

i. Path Optimization

Hybrid A* + RRT scores the highest (~9), benefiting from A*'s precise search and RRT's ability to explore complex spaces.

Basic A* follows (~8), using a deterministic search that generally finds optimal paths in structured maps.

Standalone RRT ranks lower (~6), as it may generate longer or less efficient routes due to random sampling.

ii. Efficiency

Hybrid A* + RRT is rated (~9) for effectively balancing structured search and exploratory planning.

A* performs well (~8), especially in environments with known obstacles.

RRT shows lower efficiency (~6) because of its unguided, random path generation.

iii. Smoothness

Hybrid A* + RRT excels (~10), thanks to smoothing techniques like B-spline or Bézier curves applied to raw paths.

A* is moderately smooth (~7), but its grid-based outputs may include sharp turns.

RRT matches A* (~7) when smoothing is applied; otherwise, its outputs are uneven.

iv. Computational Cost

Hybrid A* + RRT has a slightly higher cost (~8) due to algorithm integration and refinement steps.

A* remains efficient (~7) with lower overhead in simple environments.

RRT shows the lowest cost (~6), though this comes with trade-offs in performance quality.

v. Obstacle Handling

Hybrid A* + RRT performs best (~9), as it adapts well to dynamic changes and complex obstacles.

Standalone RRT also does well (~8), particularly in unfamiliar or unstructured terrain.

A* scores lower (~6), effective mostly in static environments.

Comparative Analysis of Path Planning Algorithms

Performance Metrics Overview

Parameter	Hybrid A* + RRT*	A*	RRT	Unit / Interpretation
Path Optimization	9	7	6	Score (0–10): Higher = shorter, more efficient path
Efficiency	10	8	6	Score (0–10): Combines speed and resource usage
Smoothness	9	7	6	Score (0–10): Fewer sharp turns, more drivable trajectory
Computational Cost	8	8	4	Score (0–10): Higher = more efficient (lower actual cost)
Obstacle Handling	9	7	6	Score (0–10): Ability to avoid static/dynamic obstacles

Table 5.1 Result Comparison Table A* ,RRT and Hybrid A*+RRT

The above Table 5.1 Explain about the Result Comparison Table A* ,RRT and Hybrid A*+RRT

Scores are on a scale from 0 to 10, with higher values indicating better performance

Metric Definitions

Path Efficiency: Evaluates the optimality of the path in terms of distance or cost.

Operational Speed: Combines the time taken and computational resources used during path planning.

Trajectory Smoothness: Assesses the path's continuity, with smoother paths having fewer abrupt changes in direction.

Computational Load: Measures the algorithm's demand on processing power and memory; lower actual resource usage corresponds to higher scores.

Obstacle Navigation: Determines the algorithm's proficiency in identifying and circumventing obstacles in the environment.

Insights

Hybrid A* + RRT: This algorithm synergizes the global path optimality of A* with the local adaptability of RRT. It excels in generating efficient, smooth paths while effectively navigating complex environments.

A*: Renowned for its ability to find optimal paths in well-structured settings, A* may struggle with adaptability in dynamic or unstructured environments.

RRT: Offers rapid path generation and flexibility, making it suitable for dynamic scenarios. However, without post-processing, the paths may be suboptimal and less smooth.

6. CONCLUSION

This project successfully demonstrates the complete application of software engineering principles, encompassing all phases from initial planning and requirements analysis to design, development, testing, and deployment. The proposed hybrid path planning framework, which combines the A* and RRT algorithms, effectively merges deterministic accuracy with adaptive exploration to enable autonomous systems to navigate complex and dynamic environments with high reliability.

The integrated A* + RRT approach offers real-time adaptability, computational efficiency, and practical viability for operating in unpredictable conditions. Although the current implementation meets its primary objectives, future improvements such as enhanced trajectory smoothing, parallel processing for faster computations, and advanced dynamic obstacle management are expected to significantly optimize its performance. Overall, this work not only achieves its intended goals but also lays a strong foundation for future innovations, real-world applications, and research contributions in autonomous navigation.

6.1 Future Scopes

RRT Integration for A*symptotic Optimality Replace standard RRT with RRT* to continuously improve the solution quality toward optimality over time ideal for high-fidelity path requirements.

Dynamic Replanning with Real-Time Sensor Feedback Fuse LiDAR or camera data to detect moving obstacles and trigger partial replanning without starting from scratch. Could leverage Elastic Band or D* Lite for local repair.

Machine Learning–Guided Planner Switching Use reinforcement learning or a classifier to automatically decide when to use A*, RRT, or hybrid planning based on environmental patterns.

Multi-Agent Coordination Extend to support swarm navigation or collaborative robots using techniques like prioritized planning, reciprocal velocity obstacles, or decentralized RRT.

Integration with SLAM Couple with SLAM algorithms (e.g., GMapping or Cartographer) so the robot builds its map on-the-fly while planning paths in partially known environments.

3D Environment Support Expand the system to 3D space for drones or warehouse automation including voxel-based occupancy grids and 3D motion constraints.

Energy or Time-Aware Cost Functions Refine your path scoring to account for power consumption, elevation, or traffic conditions critical for long-range robots or delivery drones.

GUI-Based Interactive Planning Build a user-friendly interface that allows users to click-start and goal positions, view path progress in real time, and override replanning triggers.

Reference

1.MDN Web Docs. (2024). HTML, CSS, and JavaScriptDocumentation. https://developer.mozilla.org

2.W3Schools. (2024). Web Development Tutorials. https://www.w3schools.com

3.Stack Overflow. (2024). Programming Q&A Platform. https://stackoverflow.com

4.GitHub Docs. (2024). Version Control and Git Usage. https://docs.github.com

5.React Documentation. (2024). https://reactjs.org/docs

6.Node.jsDocumentation.(2024).https://nodejs.org/en/docs

7.Express.js Guide. (2024). https://expressjs.com

8.MongoDBDocumentation.(2024). https://www.mongodb.com/docs

9.Firebase Documentation. (2024). https://firebase.google.com/docs

10.Jest Testing Framework. (2024). https://jestjs.io/docs

11.Postman Learning Center. (2024). https://learning.postman.com

12.Docker Documentation. (2024). https://docs.docker.com

13.Visual Studio Code Docs. (2024). https://code.visualstudio.com/docs

14.GitLab CI/CD Documentation. (2024). https://docs.gitlab.com/ee/ci/

15.JUnit 5 User Guide. (2024). https://junit.org/junit5/docs/current/user-guide/

16.AgileAlliance.(2024).AgileManifesto&Principles.https://www.agilealliance.org/agile101/

17.Jira Software Guide. (2024). https://support.atlassian.com/jira-software-cloud/

18.OWASP Foundation. (2024). Top 10 Web Security Risks. https://owasp.org/www-project-top-ten/

19.REST API Tutorial. (2024). https://restfulapi.net

20.SQLBolt. (2024). Interactive SQL Lessons. https://sqlbolt.com

21.Python Documentation. (2024). https://docs.python.org/3/