AMR Path Planning Optimization

Layout Comparison & Energy Analysis

Rahul Yadav (2022MEB1334)   Sumer Bassi (2022MCB1351)

Department of Mechanical Engineering
Indian Institute of Technology Ropar

Supervisor: Dr. Ekta Singla

Mid-Semester Progress Report

Project Overview

Research Objectives

- Develop path planning optimization strategies for AMRs
- Compare three layouts: Grid, Fishbone, Serpentine
- Analyze energy consumption and travel distance
- Applications in warehousing and vertical farming

Platform

- Robot: Novus Carry AMR
- Payload: 100-1500kg
- Navigation: LIDAR-based
- Framework: ROS Jazzy

Tools

- Ubuntu 20.04 LTS
- Gazebo 11 Simulator
- RViz Visualization
- Python Analytics

Phase 1: Manual AMR Operation

COMPLETED

Controller Testing

- Forward/Backward validated
- Left/Right turn confirmed
- Emergency stop functional

Achievements

- Hardware validation complete
- Safety systems verified
- Platform ready

Phase 1: Autonomous Operation Setup

COMPLETED

Network Configuration

Connection:

- Direct LAN connection
- Laptop: 192.168.100.120
- AMR: 192.168.100.104
- Connectivity verified

Portal:

- NHRSL portal accessed
- Factory BPT created
- Web interface operational

Navigation Workflow

Factory Info → SLAM Mapping → Waypoints → Mission Execute

First Autonomous Mission Success

COMPLETED

Mission Details

- Location: Corridor outside the lab
- Path: Loop (back and forth)
- Mode: Fully autonomous
- Status: Success

Challenges Resolved

- IP configuration issues
- Interface connectivity
- Waypoint accuracy
- AMR behavior control

Documentation: Video + procedure document

Video Demonstration

AMR Moving Autonomously (Back and Forth):
[Google Drive - Video Link]

ME1 First Floor Mapping

COMPLETED

Completed

- Full floor map created
- SLAM-based scanning
- Obstacle detection
- Map refinement done

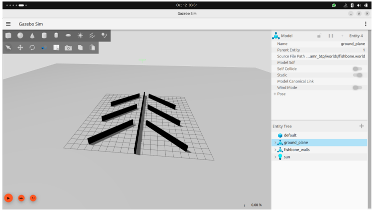
Pending

- Waypoint generation
- Awaiting permission
- Safety approval
- Mission ready

Gazebo Simulation Layouts

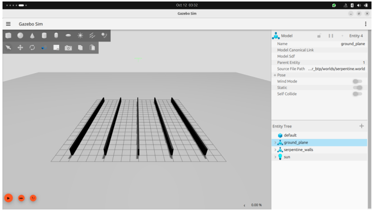
IN PROGRESS

Fishbone Layout



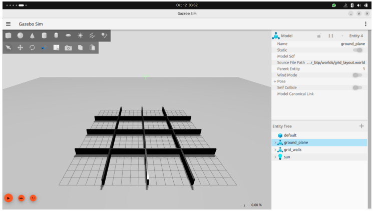
- Central spine
- Diagonal aisles
- Reduced distance

Serpentine Layout



- Continuous path
- Minimal turns
- Sequential

Grid Layout



- Traditional design
- Parallel aisles
- Baseline

URDF files created and tested

AMR Model Integration Strategy

Plan A (Primary)

Official Model Files

- Source: Manufacturer
- Files: URDF + STL
- Accurate specs
- Pre-verified

Status: Awaiting delivery

Plan B (Backup)

Custom Creation

- In-house URDF/STL
- SolidWorks
- Full control

Status: Prep initiated

Parallel preparation for both plans

Path Planning Algorithms

Selection Criteria

- Completeness: Guaranteed path finding
- Optimality: Shortest/optimal path
- Efficiency: Real-time performance

A*
Heuristic-based

RRT
Random Tree

D*
Dynamic

A* Algorithm

Overview

Best-first search using heuristic function:

$$f(n) = g(n) + h(n)$$

- g(n): Actual cost from start
- h(n): Heuristic to goal
- f(n): Total estimated cost

Data Structures

- Open List (Priority Queue)
- Closed List (Hash Set)
- Parent Map

Properties

- Complete
- Optimal (if admissible)
- Time: $O(b^d)$

A* Algorithm

Advantages

- Optimal path
- Good for static environments
- Efficient with good heuristic
- Grid layout perfect
- Easy implementation

Limitations

- High memory usage
- Not for dynamic obstacles
- Needs complete map
- Re-plan from scratch

Best for: Static warehouse layouts

RRT Algorithm

Overview

Sampling-based for high-dimensional spaces

- Random sampling
- Tree growth toward unexplored areas
- Goal bias (0.05-0.1)

Parameters

- Step size
- Goal bias
- Max iterations
- Threshold

Variants

- RRT* (optimal)
- RRT-Connect
- Informed RRT*

RRT Algorithm

Advantages

- Complex obstacles
- Probabilistically complete
- Fast in high-D spaces
- Non-holonomic constraints
- Good for Serpentine

Limitations

- Not optimal (basic)
- Jagged paths
- Random behavior
- Quality varies
- Slow in narrow passages

Best for: Complex obstacles, RRT* recommended

D* Algorithm

Overview

Dynamic replanning for changing environments

- Backward search (goal to start)
- Only recalculates affected portions
- Extremely efficient

D* Lite

- g-value: Cost from start
- rhs-value: Lookahead value
- Consistent when g = rhs
- Simpler than original D*

D* Algorithm

Advantages

- Dynamic environments
- 10-100x faster replanning
- Real-time avoidance
- Temporary obstacles
- Perfect for workers
- Seamless updates

Limitations

- Complex implementation
- Initial $=$ A* complexity
- Needs sensor integration
- Higher memory
- Only benefits in dynamic

Best for: Dynamic warehouses with workers

Algorithm Comparison

Criteria	A*	RRT	D* Lite
Completeness	Complete	Probabilistic	Complete
Optimality	Optimal	Sub-optimal	Optimal
Replanning	From scratch	Fast new	Very fast
Memory	High	Moderate	High
Static	Excellent	Good	Excellent
Dynamic	Poor	Good	Excellent
Implementation	Simple	Moderate	Complex
Grid Layouts	Perfect	Good	Perfect

Hybrid Strategy

Phased Implementation

Phase 1

A*

Baseline testing

Grid/Fishbone layouts

Phase 2

RRT*

Complex scenarios

Serpentine layout

Phase 3

D* Lite

Dynamic testing

Real-world scenarios

Software Environment

COMPLETED

Core Stack

- Ubuntu 20.04 LTS
- ROS Jazzy
- Gazebo 11
- RViz

Tools

- Python environment
- SLAM packages
- Path planning
- Network tools

Progress Summary

Completed Milestones

- Hardware validation (manual + autonomous)
- ME1 floor mapping complete
- Three Gazebo layouts created
- Software environment ready
- Documentation and videos

Upcoming Work

Model Integration

- Plan A: Await manufacturer files
- Plan B: Custom URDF/STL creation

Simulation & Analysis

- A* implementation and testing
- All layouts simulation
- Energy data collection
- RRT* comparison

Conclusion

Achievements

- Phase 1 completed successfully
- Clear methodology established
- On schedule for remaining work

Key Takeaways

- Project feasibility demonstrated
- Technical readiness confirmed
- Detailed path forward established

Status: On Schedule