Multi-Agent Trajectory Planning: For Search and Rescue Operations

1st Sahruday Patti
University of Maryland
College Park
sahruday@umd.edu

2nd Sandeep Adapa
University of Maryland
College Park
sadapa@umd.edu

***Abstract*—This paper addresses trajectory planning of a multi-agent system, where n agents are moving in the same region and need to coordinate so as to maintain a certain pairwise safety distance, while avoiding obstacles. They key features include that at most n iterations are needed to generate pareto optimal trajectories for the agents. Agents use of RRT* algorithms with a prune and graft to generate a trajectory at each iteration.**

***Index Terms*—Multi-agent Trajectory Planning , RRT*, Multi RRT*, Prune and Graft.**

I. Introduction

In a multi-agent system, trajectory planning is determining the trajectories of numerous agents moving in the same region of space, from their origin to their destination positions, while reducing costs such as time, avoiding obstacles and to maintain a certain distance threshold between the objects.

The two main approaches to multi-agent trajectory planning is Centralized and Decentralized. Centralized algorithms requires a priori the locations of the agents position and the information of the agents actuation capabilities, agents states etc for the computation of trajectories. This requires a lot of exchange of information and computation. Where as Decentralized approaches involve less computations and exchange of information. we have chosen to implement one of the Decentralized algorithms in the literature.

Our inspiration for this project comes from a situation that has occurred in the past where 13 football players where stuck in cave in Thailand due to floods for 13 days. Besides knowing the layout of the cave it took 11 days to find the location. we wanted to do trajectory planning for these kind of situations like people stuck in a fire, to stop fire in a forest etc. In a search and rescue operation like these every agents order is equally important and we need to compute the path of the agents as quickly as possible which in turn involves in less exchange of information and being able to be self sufficient for trajectory planning.

In this project we plan to accomplish a Decentralized cooperative approach published to this problem which helps to preserve privacy of the agents parameters and distributes computational load and which integrates all the previous features described.Our chosen Decentralized algorithm does not agree a priori on the order of the agents, involves less exchange of information between the agents and each agent does not have to depend on other information to compute the path to reach the goal. Thus, making it suitable for Search and Rescue operations. One drawback though, is that all the agents are assumed to be having same velocity. This makes for simplicity in calculating collisions between agents when we consider other agents path as a dynamic obstacle.

II. Problem Formulation

Considering 3 agents system where each agent is at some position i.e agents origin say $P_{n,origin}$, and moving to their respective goal positions say $P_{n,goal}$. Here P is the region of space agents are moving and n is the number of agent. Below is the problem formulation for our problem formulation.

Each agents Dynamics is described via a differential equation of the form

$$\dot{S_n}(t) = f_n(S_n(t), u_n(t)); S_n(0) = S_{0,n} \tag{1}$$

where f_i is continuously differentiable as a function state(position) and linear velocity. $S_n(0)$ is the origin of the agent.
An Obstacle free trajectory Z_n is defined as $Z_n = (S_n, u_n, T_n)$ where T_n is the number of time steps in the trajectory. As we have assumed same velocity for all the agents we calculate the path as number of time steps considering the agents control input evolution. we have used turtle bots in non Holonomic constraint formulation and each time step in the trajectory is calculated based on these constraints. This is our f, the differentiable function.

In the formulation, The trajectory Z_n is rated according to the number of time steps as the criteria. Since, we used all the agents as turtle bots with same velocity we know that the cost is same to go from point to point and the number of time steps is a good measure for the cost of the trajectory.

Now, the goal is to compute an Obstacle free trajectory for the three agents that is conflict free and good enough in terms of cost for the agents in a decentralized manner involving less exchange of information and less number of iterations.

III. DECENTRALIZED COOPERATIVE RESOLUTION

The decentralized cooperative resolution is an iterative algorithm. It starts with every agent computing an OPTIMAL obstacle free trajectory path neglecting the presence of all the other agents. Now, the problem to solve is to produce a trajectory while avoiding other agents with minimal performance degradation of this optimal path. Here, to compute a path RRT Star was used. Note that, Although the RRT star is not an optimal path generator for a smaller number of nodes explored the path produced neglecting all the other agents presence is considered an optimal path in the formulation. Thus, the trajectory produced at this phase is considered as the best performance of the agent.

This Optimal path is broadcasted to all the agents and will be used to check for conflicts. If a conflict between the trajectories is detected , then the iterative procedure is used for conflict resolution.

In the first iteration, agents that are involved in a conflict re plan their trajectory considering other agents as time varying obstacles that are moving according to the trajectory that they broadcasted.agents then broadcast the performance degradation associated with its new trajectory. The agent with minimal performance degradation is assigned the task of contributing to conflict resolution by updating its trajectory and replacing it with the newly planned one. Then this newly planned trajectory is broadcasted to other agents and the trajectory of this agent will not be changed in the further iterations.The iterations are continued till all trajectories are conflict free.

With this approach,it takes at most n-1 iterations to compute a conflict free path for n agents. Therefore, an agent has to compute its trajectory at most n times. note that, the trajectories produced are Pareto Optimal.

As mentioned in the formulation, the performance degradation is evaluated based on the number of time steps. we compute the Performance degradation as the change in number of time steps between the optimal trajectory and current trajectory divided by the number of time steps in optimal trajectory.

$$D_n^{(k)} = ((J_n\left(z_n^{(k)}\right) - J_n^*) \div J_n^*).100 \tag{2}$$

where :

$\mathrm{D}_n^{(k)}$ *is the degradation index of nth agent in the kth iteration*

J_n *is the cost criteria which is based on number of time steps*

J_n^* *is the optimal cost function*

$\mathrm{z}_n^{(k)}$ *is the trajectory of nth agent in the kth iteration*

IV. MULTI RRT*

In order to generate a trajectory for each agent, we implement a RRT* algorithm. We implement RRT* on each agent(multi rrt*) iteratively considering other paths as obstacles and the re planning is performed according to an order of minimal performance degradation. This iterative generation of RRT* from ground up can be computationally expensive. In order to alleviate this computational load a solution was proposed called the Prune and Graft.

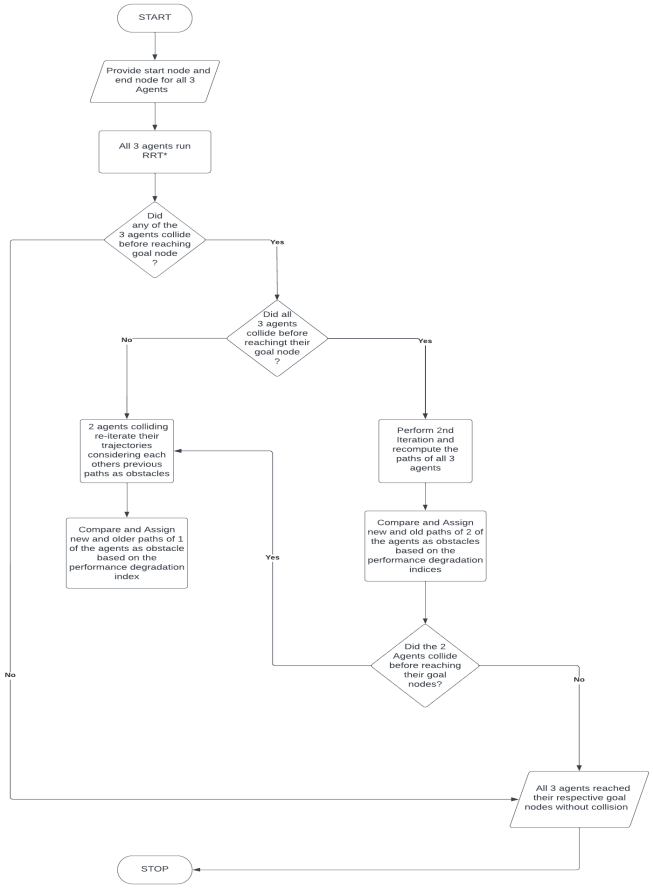

Fig. 1. Implementation Flow Chart of Multi-RRT* Algorithm based on Performance Degradation Index

We produce a complete tree at the start for each agent. Then at each iteration after that we use the nodes generated before the following iteration i.e after the initial iteration if we collide at the time step t, we prune our trajectory till that

time step t and delete all the nodes after that time step t and we graft the tree from that time step.

In Summary, the agents run a complete iteration of RRT* considering only the obstacles in the region P neglecting the other agents. Then the Prune and graft is utilized to regrow the tree considering other agents paths as Dynamic Obstacles.

Smoothing the Path : The path generated by the RRT* is not optimal and has so many curves. In order to compensate for the curves, we fit a total least square fit on the obtained trajectory and obtain a smooth curve.

V. Conclusions and Future work

In the 2D simulation, we were able to generate Pareto optimal trajectories for each agent while also utilizing the prune and graft .

As we are working in a very small region, the agents collide in a very short time step even at the time step 1 mostly, which made it difficult for us to show the pruning and grafting. We tried many different origin and goal positions to visualize the output.

We have simulated turtle bots in the gazebo world and final trajectories were generated. The bots do not follow the path generated and hence agents do not reach the goal locations. We tried to write a PID control, but did not implement it successfully.

We intend to extend this approach to drones, in turn achieving the solution for our inspired problem.

VI. Simulations and Results

The following plots were obtained during the trajectory generation of the 3 agents and significance of each of the plot is explained below

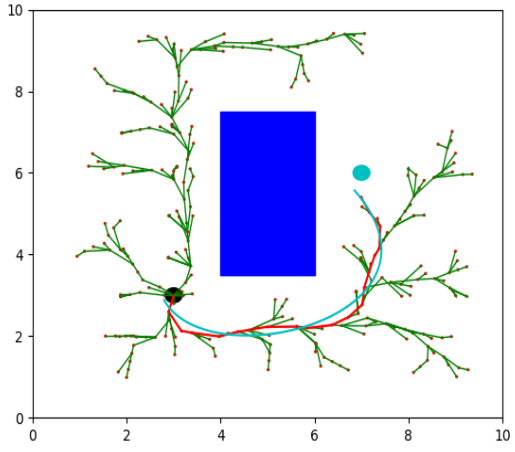

Fig. 2. Optimal Path of Agent 1

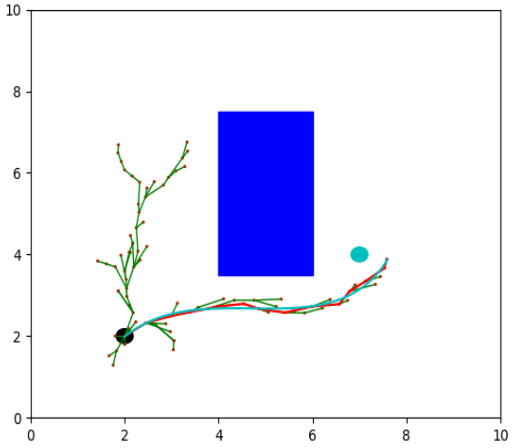

Fig. 3. Optimal Path of Agent 2

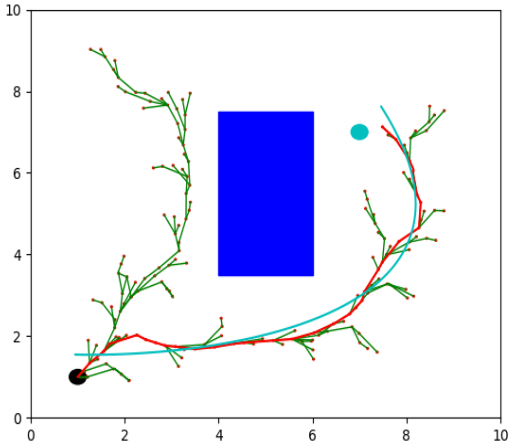

Fig. 4. Optimal Path of Agent 3

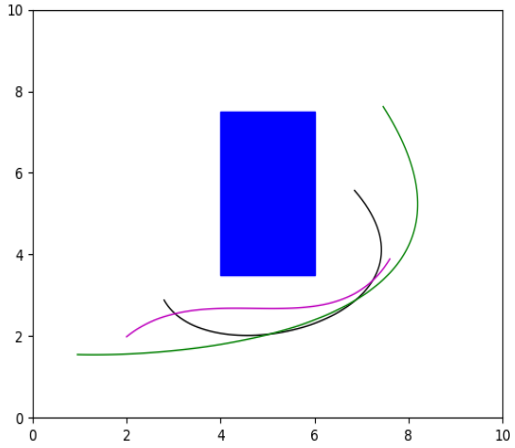

Fig. 5. Combined plot of optimal paths generated by all 3 agents

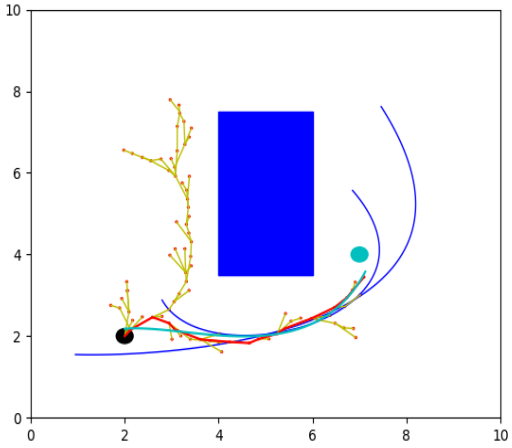

Fig. 7. Agent 2 : 1st iteration

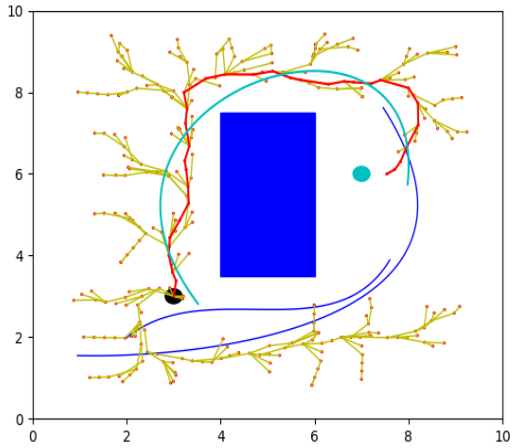

Fig. 6. Agent 1 : 1st iteration

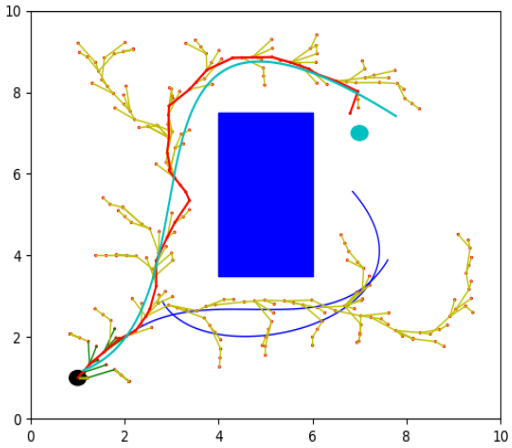

Fig. 8. Agent 3 : 1st iteration

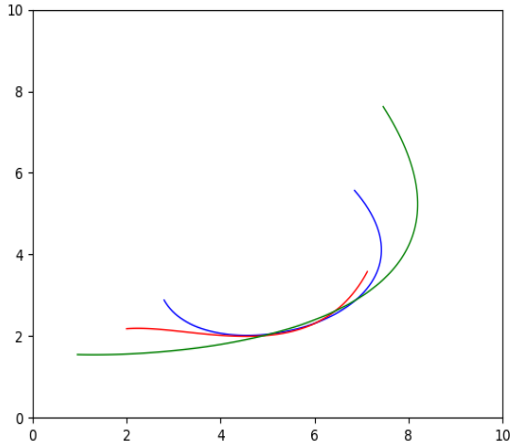

Fig. 9. Paths generated after iteration 1

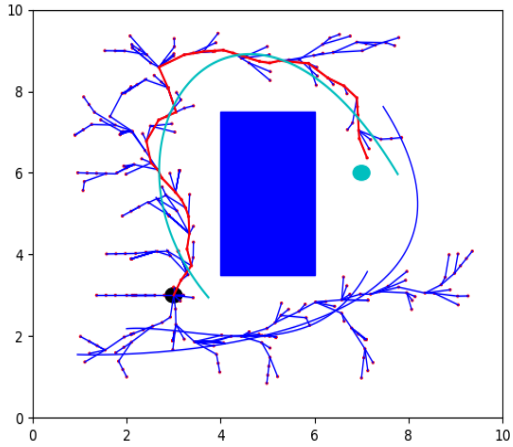

Fig. 10. Agent 1 : 2nd iteration

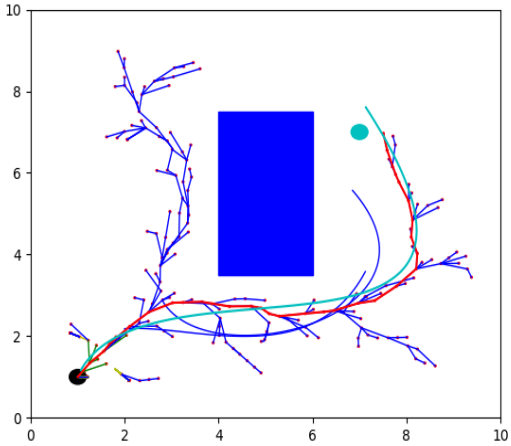

Fig. 11. Agent 3 : 2nd iteration

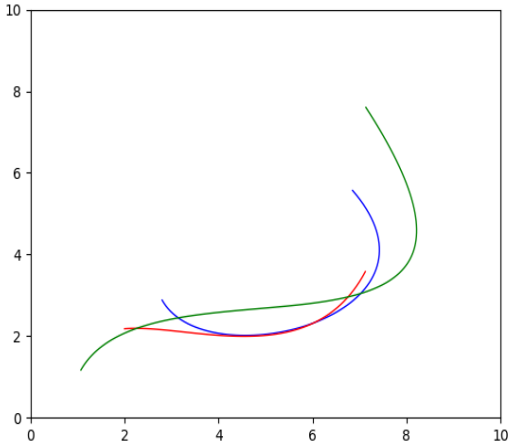

Fig. 12. Paths generated after iteration 2

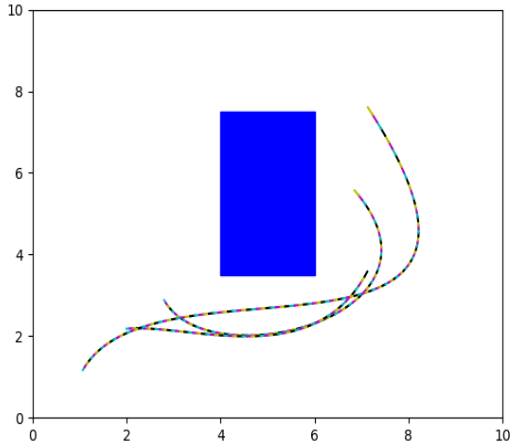

Fig. 13. Final Trajectories

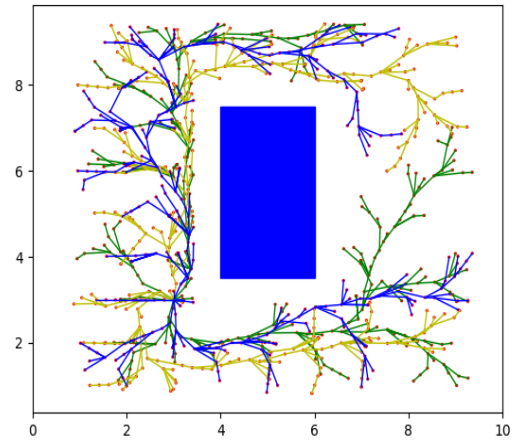

Fig. 14. All nodes generated by Agent 1 in 3 iterations

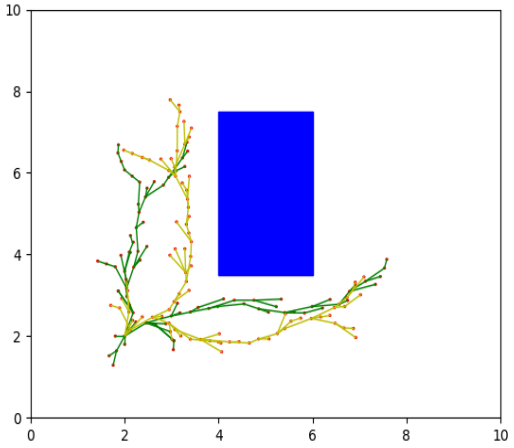

Fig. 15. All nodes generated by Agent 2 in 3 iterations

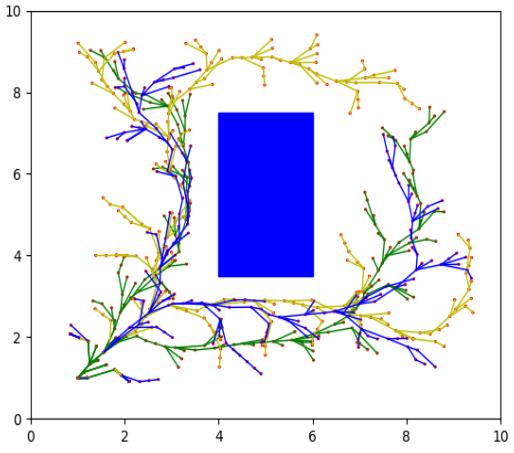

Fig. 16. All nodes generated by Agent 3 in 3 iterations

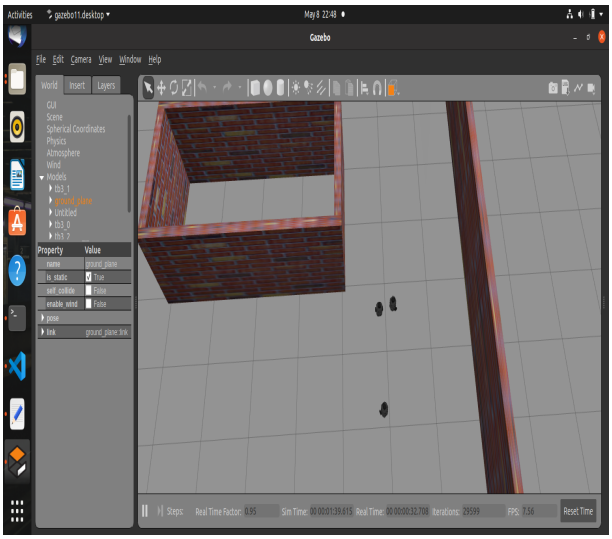

Fig. 17. ROS simulation of Multi RRT* on 3 turtule bots : Phase 1

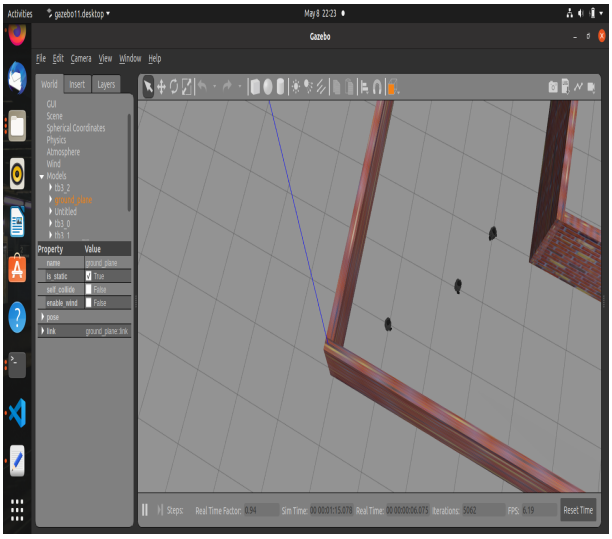

Fig. 18. ROS simulation of Multi RRT* on 3 turtule bots : Phase 2

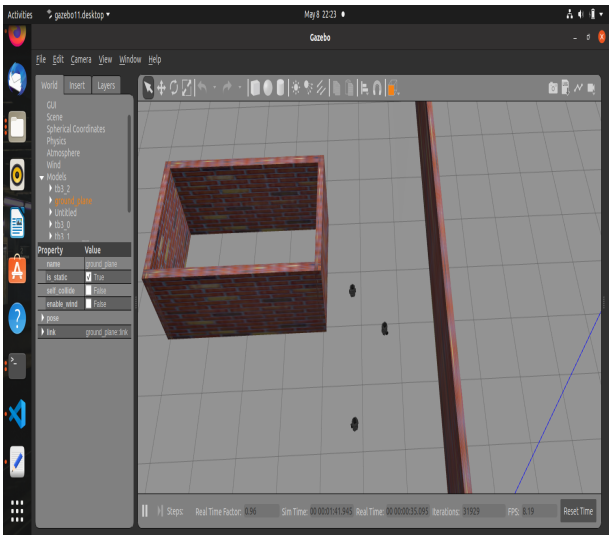

Fig. 19. ROS simulation of Multi RRT* on 3 turtule bots : Phase 3