UNIVERSITY OF
MARYLAND

ENPM 661 - Group 9

Project-5 Proposal

Exploration and Collision Avoidance in Swarm Robots using ORCA (Optimal Reciprocal Collision Avoidance) and a local planner Algorithm

Submitted by:

Saket Seshadri Gudimetla

Nikhil Mehra

Sayan Brahma

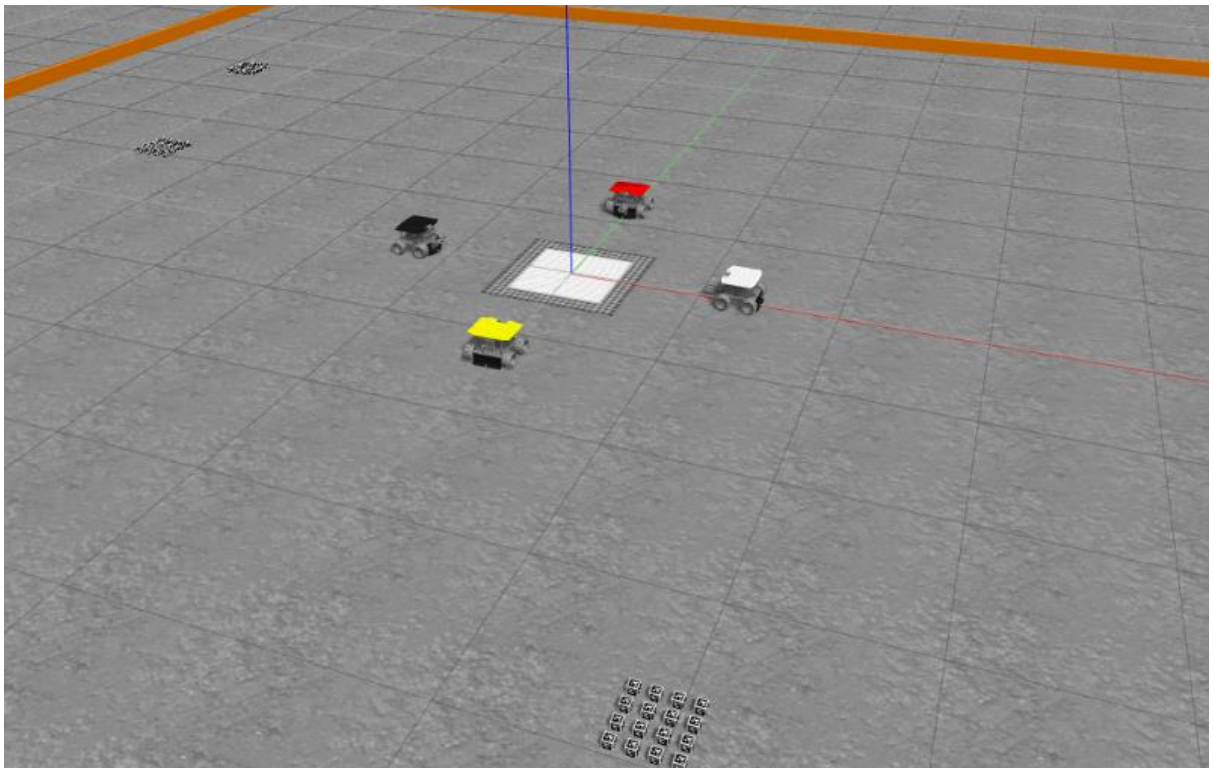
Introduction

Exploration using a robot is a difficult task in itself, whether it may be on a known terrain or an unknown terrain. Known environments can include inside of a building or a warehouse etc. Unknown environments can include a forest area, extra-terrestrial planetary surface etc. Exploration becomes much more difficult when there are unidentified obstacles and resources in the unexplored region. The harder part is not to differentiate between an obstacle and a resource, but to get around an obstacle without losing its path and efficiency. The process becomes more involved when along with static obstacles, dynamic obstacles are also at play.

The only difference between exploring a known space from an unknown space is the availability of a map (a known environment with pre-defined constraints). Hence exploring an unknown environment becomes a complex job with all these intertwined involved processes.

In this project, we are basically working on the local planning part required for the ‘NASA Swarmathon’ competition organized by NASA every year. We mainly want to emulate the exploratory behavior of a swarm, preferably three to four, in an extra-terrestrial environment. Exploration obviously includes obstacle avoidance from static obstacles and collision avoidance between the robots. It is assumed that all robots communicate with the server and not with each other, therefore each robot can be assumed to act independently.

A simple exploratory algorithm such as ray algorithm or spiral algorithm will be used. As robots can’t directly communicate with each other we are using ORCA (Optimal Reciprocal Collision Avoidance) algorithm, which is specially designed for such systems. ORCA is uses the concept of velocity obstacles (velocity space), where required conditions for the collision-free motion are derived using linear programming. Basically the problem is reduced into a low-dimensional linear program.



Example of an Exploration Arena

Applications

The above combination of exploratory and collision avoidance algorithms can be used in the following situations -

1. Search and Rescue
   Search and rescue using robots is the search and aid of provisions to the people who are in distress or are in an imminent danger/life threating situations. Search and rescue is mostly carried out in an unknown environment. These environments may include mountain, forest, and urban areas.

2. Extra-terrestrial Exploration
   Resource exploration on the moon or on another planet such as Mars. Such explorations are always carried out in an unknown environment.

3. Pursuit and Evasion Planning
   In pursuit-evasion planning, a pursuer tries to catch/capture an evader while the evader tries to avoid the capture from its pursuer. In robotics, pursuit-evasion scenarios are used to obtain worst-case guarantees to collision-

avoidance problems. In general to the motion planning in a highly dynamic environments (unknown environment).

Goals/Objectives

The proposed objectives of this project are as follows:-

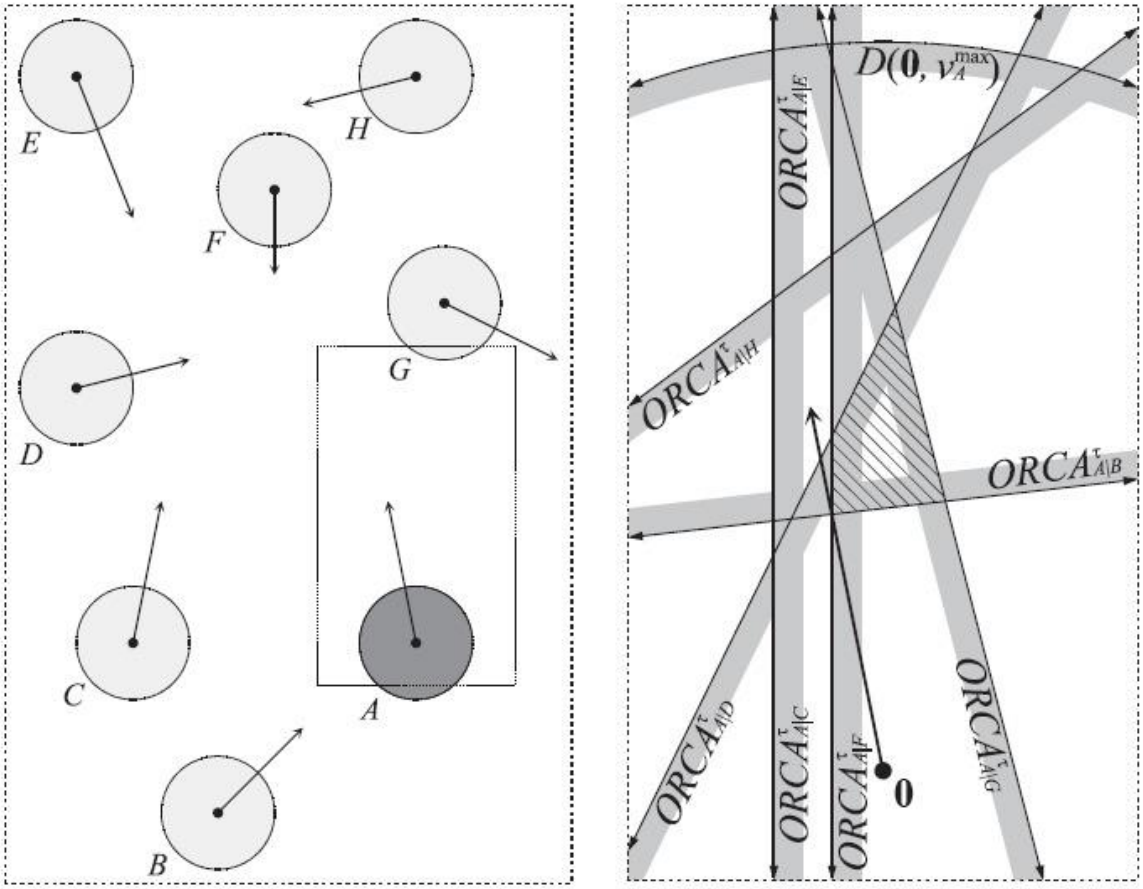
1. Exploration of the space using a simple algorithm such as ray algorithm.
2. Path planning would be done using A* or weighted A*.
3. Generating a grid for the global planner as well as velocity space for the local planner (ORCA).
4. Using ORCA as a collision avoidance algorithm implemented to avoid both other robots (dynamic obstacles) and random objects in the configuration space (static obstacles) of the robot.
5. Completion of the task viz. exploring the whole space without any collisions and reach the designated goal location.

Method

→ Obstacle avoidance using ORCA (Optimal reciprocal collision avoidance) [1]:-

1. For making it simple, we assume that the mobility of the robot is in plane R^2. Each robot has a current position P_a and a current velocity V_a. These parameters are part of the robot's external state, i.e., we assume that they can be observed by other robots. Furthermore, each robot has a maximum speed of V_A^{max} and a preferred velocity V_A^{pref}, which is the velocity with which the robot will traverse assuming that there is no other robot in its way. We consider these parameters as a part of the internal state of the robot, and can therefore not be observed by other robots.
2. The task for each robot is to independently select a new velocity V_A^{new} for itself such that all robots are guaranteed to be collision-free for at least present amount of time(τ) when they would continue to move with their new velocity.
3. Thus we would like to choose sets of permitted velocities V_a for A and V_b for B such that they are reciprocally collision-avoiding for (τ) time and maximal. There are infinitely many pairs of sets that obey these requirements but among those, we choose the pairs that maximizes the amount of permitted velocities "close" to optimization velocities V_A^{opt} for A V_B^{opt} for B.
4. In the below figure(a)and(b)[1] "Reciprocal n-body Collision Avoidance"]: We plot each velocity of A induced by B and B induced by A as half planes, and the area formed by the intersection of those planes are our point of interest and the solution set.

5. The overall approach is as follows: - Each robot performs a continuous cycle of seeing and acting with a time step Δt. In each iteration, the robot requires



(a) (b)

the radius, the current position and the current optimized velocity of the other robots (as mentioned above these are the external parameters). Based on this information, the robots infer the permitted half-plane of velocities with respect to each other robots.

6. After analyzing all these data the robot selects a new velocity V_A^{new} for itself that is closest to its preferred velocity V_A^{preff} amongst all other velocities inside the region of permitted velocities. After calculating the velocities efficiently the robot reaches its new position, which is calculated using the following equation:

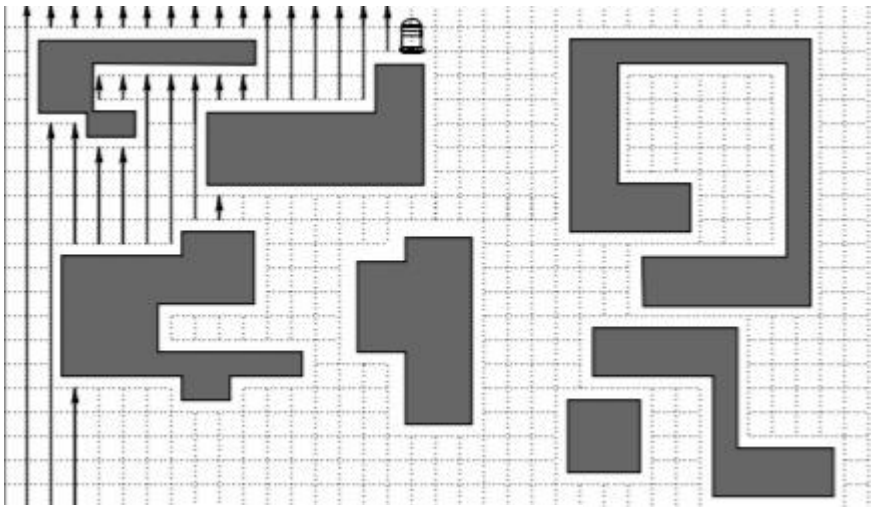
$$P_A^{new} = P_a + V_A^{new} \Delta t$$

and the sensing-acting cycle repeats.

→ **Exploration Strategy [4]:-**

In the first step, this Ray algorithm explores the exterior boundary of a scene. Starting from start(s), it travels in clockwise direction along the boundary until s is

reached again. Knowing the exterior boundary will be convenient because the robot can then distinguish it from the unexplored boundary of obstacles. During the exploration of the exterior boundary, whenever dist = R at some node x, the robot executes a procedure Refuel. In this procedure the robot moves back to s and then relocates again to x to resume exploration.
An obstacle is new if none of the nodes and edges on the boundary are explored. An obstacle is mapped fully if all nodes and edges on the boundary are explored.



Ray Algorithm [4]

The actual exploration of the scene proceeds in rays. Let S be the bottom segment of the exterior boundary and let $s = x_1; : : : ; x_n$ be the vertices of S.
For every i, 1<=i<=n, the robot starts at node x_i and explores a vertical ray of edges in northern direction until an obstacle or the exterior boundary is hit. Then the robot backtracks to x_i and moves to the neighboring node x_{i+1} if it exists. In general, we say that the robot hits an obstacle if it reaches a node on the boundary with degree less than 4. We refine this definition when rays are traversed. Here the robot hits an obstacle if the obstacle blocks the ray, i.e. there is no outgoing edge in northern direction. An important feature of the Ray algorithm is that it applies a depth-first strategy.

Assumptions

1. It is assumed that all robots communicate with the server and not with each other, therefore each robot can be assumed to act independently.
2. We assume that the robots move in plane R^2.
3. The Robots are non-holonomic, i.e., they cannot move in lateral direction.
4. Preferable velocity for all the robots is $V_A^{pref} = V_B^{pref}$ = 0.5 m/s.
5. The sensor data is assumed to be robust and noiseless.

6. Robots are able to observe each other's current position and velocity (which are external parameters).

Software

We will use ROS/Gazebo for modeling and simulation of the project. C++ or Python will be our programming language of choice.

Conclusion

To summarize, we will be implementing a local planning algorithm called ORCA and in order to perform obstacle avoidance and exploration for a swarm of robots (3-4) that would be released in an unexplored region. The proposed work can be used for solving real-world problems such as search and rescue, extraterrestrial exploration missions etc. We hope to learn and implement the fundamental aspects of motion planning with this project, and if possible improve upon the existing literature.

References

1. Jur van den Berg, Stephen J. Guy, Ming Lin, and Dinesh Manocha Cédric Pradalier, Roland Siegwart, and Gerhard Hirzinger (eds.), Robotics Research: The 14th International Symposium ISRR, Springer Tracts in Advanced Robotics, vol. 70, Springer-Verlag, May 2011, pp. 3-19.
2. Jamie Snape, Jur van den Berg, Stephen J. Guy, and Dinesh Manocha Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), October 2010, pp. 4584-4589.
3. M. Betke, R. Rivest and M. Singh. Piecemeal learning of an unknown environment. Proc. 5th Conference on Computational Learning Theory, pages 277{286, 1993.
4. S. Albers and M. Henzinger. Exploring unknown environments. Proc. 29th Annual ACM Symposium on Theory of Computing, pages 416{425, 1997.
5. Borenstein, J., Koren, Y.: The vector field histogram - fast obstacle avoidance for mobile robots. IEEE Trans. Robot. Autom. 7(3), 278–288 (1991).
6. Ulrich, I., Borenstein, J.: VFH+: reliable obstacle avoidance for fast mobile robots. In: Proceedings of IEEE International Conference on Robotics and Automation, vol. 2, pp. 1572–1577, May 1998.