Planning Navigation Routes in Unknown Environments

Laura Rodriguez(✉), Fernando De la Rosa, and Nicolás Cardozo

Systems and Computing Engineering Department, Universidad de los Andes, Bogotá, Colombia
{la.rodriguez,fde,n.cardozo}@uniandes.edu.co

Abstract. Self-driving robots have to fulfill many different operations, as coordinating the motors' traction, camera movement, or actuator arms mechanics, as well as more high-level operations like driving to different places. Autonomous navigation is of utmost importance for exploration robots, which must drive around exploring areas with unknown terrain conditions, as for example is the case of mars rovers and other space exploration vehicles. Given that the environment is unknown, planning a specific route and driving plan is challenging or even inappropriate due to blocking obstacles in the terrain. To overcome such problems we propose an adaptable plan for driving robots in different situations. Our solutions mixes both global and dynamic planning algorithms to take advantage of available information, if it exist beforehand, and to overcome unknown obstacles if they appear, while still moving towards the goal. In particular, we apply our algorithm to the movement of robots between posts in environments with partial information, as it is the case of space mission competitions. We evaluate our solution in a simulated environment taking into account the effectiveness in fulfilling a mission in the shortest time, using the shortest possible path. Our results show that of the A* algorithm with diagonals in combination with the ABEO algorithm offer the best combination reaching the goal in most cases, in optimal (planning + execution) time.

Keywords: Mobile robotics · Autonomous driving · Dynamic planning

1 Introduction

Autonomous navigation is one of the goals in space missions, as promoted by the University Rover Challenge (URC)[1] or the European Rover Challenge (ERC) competitions.[2] In these missions, a rover robot is required to navigate given GNSS-only waypoints through posts across an easy and moderate terrain. Teams may visit locations in any order, but must declare when they are attempting an objective out of order. Teams are provided with a high-accuracy coordinate at the start gate as a reference. Each post has a marker displaying a black and white ARUCO tag using the 4×4_50 tag library as shown in Fig. 1.

[1] https://urc.marssociety.org.
[2] https://roverchallenge.eu/.

M. Tabares et al. (Eds.): CCC 2023, CCIS 1924, pp. 128–140, 2024.
https://doi.org/10.1007/978-3-031-47372-2_13

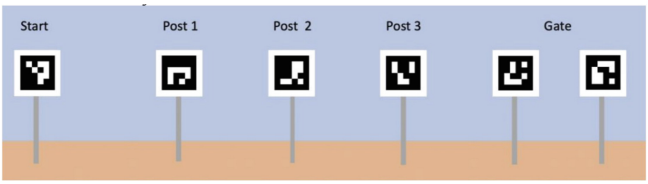


Fig. 1. ARUCOS to visit in a competition.

One problem in navigation competitions is that teams have partial or unknown information about the terrain in which the competition takes place. This means that it is not possible to predefine a route to successfully reach the desired posts, as unknown obstacles (*e.g.,* rocks, steep hills, craters) may be in the way.

The contribution of this paper consists of the exploration of a combination of planning algorithms that allow a rover to autonomously navigate between ARUCO waypoints sorting unknown obstacles. Our solution (Sect. 2) uses a global planning algorithm with the available information to sketch the shortest route between waypoints. Then, at run time, we use ABEO, a bio-inspired algorithm to avoid obstacles and continue towards the goal. Our evaluation of the combination of global and dynamic planning (Sect. 3) proves effective in finding an optimal route with respect to the execution time (*i.e.,* route planning and navigation between the points) using an improvement of the A* algorithm we posit by drawing diagonals between the points in the route. Moreover, our algorithm is also effective in reaching the goal in spite of possible obstacles in the planned route, thanks to the ABEO algorithm.

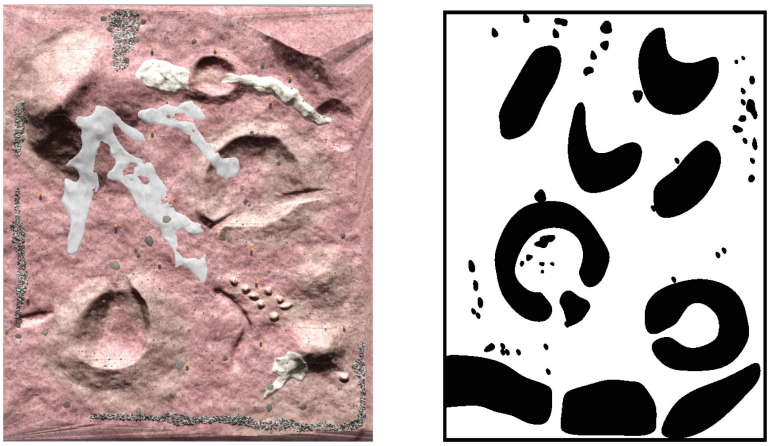
2 Route Planning for Unknown Environments

We posit a solution to the problem of autonomous navigation in unknown environments, in two parts: (1) a global or static planning, to define a potential route using available map information, and (2) a dynamic planning to sort out obstacles (*e.g.,* rocks, steep hills, craters) unknown during the planning phase.

2.1 Global Planning

In known environments it is possible to plan a route between two points avoiding all obstacles, as explained in the following. In our case, the robot only has partial information about the environment. Nonetheless, it is possible to plan an initial route. Take for example the environment in Fig. 2a. The rover may produce an approximated map as the one shown in Fig. 2b, where the most visible obstacles on the map are marked in black. From this image it is possible to plan a route, globally, given the initial and destination waypoints.

We explore three different algorithms to define a route globally, based on classic shortest-path and computational geometry. We present the reasoning behind each of the algorithms now, presenting the results later, in Sect. 3.



(a) Original Terrain (b) Image Transformation

Fig. 2. View of the terrain and the identification of obstacles for the robot

A*. The A* algorithm [4] is a heuristic-based search algorithm to find the shortest path between two points in a graph. If we represent the map as a graph, where each pixel corresponds to a node and the edges between nodes correspond to the robot's movements (*e.g.,* up, down, left, and right), we can use A* to plan a route between the given initial and end waypoints. The objective used for the heuristics is to find a path between waypoints that takes the shortest time, and/or the shortest distance.

A* with Diagonals. Starting from the A* algorithm, we implement an extension using diagonals to cut corners between the points used to surround obstacles. The diagonals are drawn by taking rays from the points in the route and projecting them in the direction of the destination. If we intersect the route, then this is updated with the diagonal. This extension of A*, should reduce the distance between waypoints.

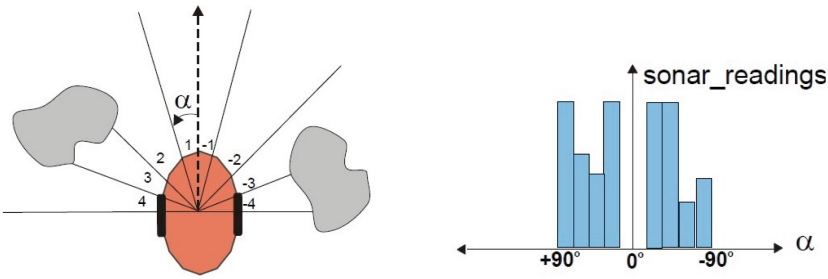
Convex Hull. The Convex hull [6] is an algorithm to calculate the smallest convex polygon that encloses all the points in a set. We use the convex hull to enclose each of the obstacles in the map in its convex hull. Then we use the A* algorithm to better avoid obstacles. This extension should give a shorter distance, as we are stepping away from obstacles' irregularities.

2.2 Dynamic Planning

As mentioned before, having an initial route may not be sufficient in unknown environments as obstacles may appear in the route. To overcome this problem we evaluate three algorithms designed for obstacle avoidance.

The Bubble Rebound Algorithm. This algorithm detects obstacles within the robot's *sensitivity bubble*. The area covered by the sensitivity bubble depends on the robot's sensors, speed, and geometry, among other characteristics. Upon detecting an obstacle, the robot *bounces* in a direction that has the least obstacle density and continues its movement in this direction until the target becomes visible or the robot encounters a new obstacle [8].

Figure 3a shows the sensitivity bubble at work, as a semi-ring (only 180°) equidistant from the robot's center. The bubble defines a protection field, where, if a sensor detects an obstacle in a direction (*e.g.,* ray −3), it bounces the robot to the area with the lowest obstacle density (*e.g.,* rays −2, −1, and 1), as represented in the distribution in Fig. 3b, where higher bars mean less density.



(a) Sensitive bubble (b) Sonar readings

Fig. 3. Sensitive bubble (taken from [8])

The flow chart in Fig. 4 shows the algorithm's process. The algorithm continuously tries to move towards the objective, once an obstacle is found, the movement direction is adjusted until the robot surrounds the obstacle.

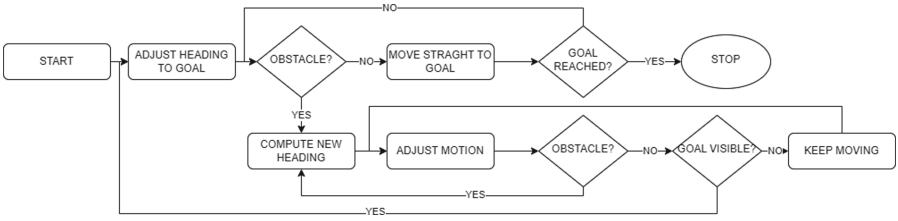


Fig. 4. Flow chart Bubble rebound algorithm [8]

Obstacle-Dependent Gaussian Potential Field (ODG-PF). The idea behind this method is that, after receiving distance data from the range sensors, the robot considers the objects within a given threshold as obstacles. To avoid obstacles, these are made larger with respect to the size of the vehicle,

building a Gaussian (repulsive) potential field. Next, we calculate the attractive field from the yaw angle information of an Inertial Measurement Unit (IMU). The total field is a combination of the two fields. We choose the angle with the minimum value of the total field [1]. Figure 5 describes the process to define the Gaussian field around the robot and choose the minimum angle for the robot's direction.

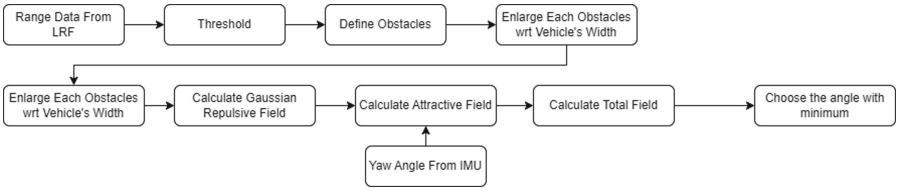


Fig. 5. Algorithm flowchart for the ODG-PF [1]

Bioinspired Algorithm for Obstacle Avoidance (ABEO). The ABEO algorithm [7] is a bio-inspired algorithm mimicking the behavior from nature combining two approaches: a force vector system, and a contour following. The force vector system builds a system of forces, which is determined by the repulsive forces F_r (in red) produced by the obstacles, and the attractive force F_a (in green) received by the target, as shown in Fig. 6.

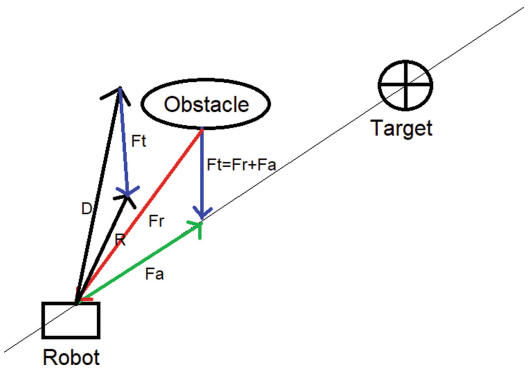


Fig. 6. ABEO force vector system approximation [7]

One of the most significant advantages of this approach, is that the robot gradually moves away from obstacles, and deviates its trajectory in short steps, so its movements are smooth. Additionally, the low complexity in computing and implementing the F_r and F_a vectors is an advantage of the algorithm. This computation entails basic linear algebra vector operations.

2.3 Implementation

The development of the route planing in unknown environments is built on top of: (1) the Robot Operating System (ROS), to communicate with the robot's sensors and actuators, and (2) Gazebo, used as a simulation environment to test the different algorithms.

We first use Jupyter notebook to implement and test algorithms, to then deploy them into the robot.

ROS. ROS consists of a set of software libraries and tools to build robot applications. The main functionality of ROS is to communicate robot systems. ROS defines the components, interfaces, and tools to build robots. The connection between tools and components is done through topics and messages [2].

We use two main topics in the project. One topic is used to publish the initial and final coordinates for the robot. The second topic is used to publish the route to follow. The architecture of our solution using ROS is shown in Fig. 7.

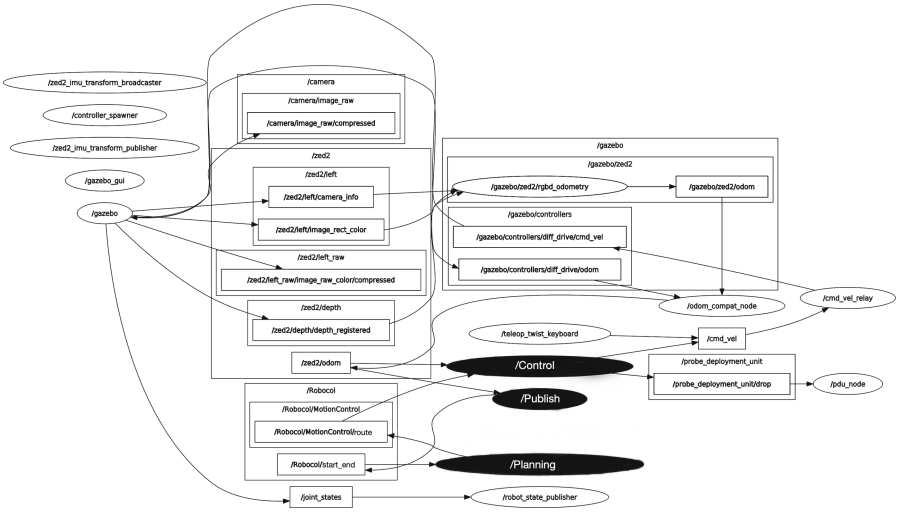
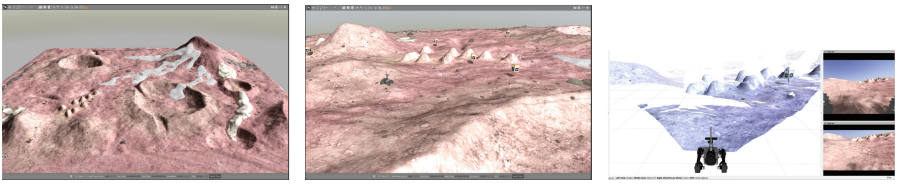


Fig. 7. ROS nodes' graph and associated topics

In Fig. 7 three main nodes (in black) publish information: the planning node (`Planning` is in charge of planning the route given its coordinates), the publish coordinates node (`/Publish`) is in charge of publishing the initial and final coordinates, and the control node (`/Control`) is in charge of sending signals to the robot's wheels, so that it moves. These nodes, in turn, subscribe and publish different information to different topics. For example, the `/zed2/odom` topic is in charge of receiving and publishing the information about the odometry (position) of the robot; the `/Robocol/start_end` topic is in charge of receiving and

publishing the information of the initial and final waypoints for the planning, and the `/Robocol/MotionControl/route` topic is in charge of connecting the planning nodes and control of the rover.

Gazebo. Gazebo is an open source robotic simulator, which provides an opportunity to simulate different environments with a complete toolbox of development libraries and cloud services. In this project, we use the Gazebo environment from the ERC competition as the platform to run our experiments on autonomous navigation. This simulation environment loads the information of the rover and the ARUCOS to mark each of the way-points, and the information about the terrain conditions. Figure 8 captures images from the simulation environment used in this project.



(a) (b) (c)

Fig. 8. Simulation Environment in Gazebo

Jupyter Notebook. We use Jupyter Notebook as the implementation platform for the route planning algorithms. The algorithms' implementation, results and simulation examples are available at our online repository.[3]

3 Route Planning Evaluation

This section presents the evaluation of the planning algorithms described in the previous section, taking into account the global and dynamic planning algorithms individually. Our experiments are run in a ASUS - ROG Zephyrus machine with 16GB of RAM and a AMD Ryzen 9 processor, using Python version 3.8.10 with the roslib, rospy, cv2, numpy, cv_bridge, matplotlib, rospkg, pynput libraries. Additionally, video recordings of the simulation for the different algorithms are available online.[4]

[3] Repository: https://anonymous.4open.science/r/Navigation_Planning-C3D9.
[4] Simulation videos: https://tinyurl.com/Simvid52.

3.1 Global Planning Results

To evaluate the effectiveness of global planning algorithms we measure three variables: (1) the time that the algorithm takes to finish the planning, (2) the number of coordinates that it sends to the control node to start moving the rover, and (3) the total distance in the planned route. The evaluation scenario for the three algorithms consists of 6 waypoints visited in order.

Table 1. A* Results.

Waypoint	Time (ms)	Number of Coordinates	Total Distance (m)
W1	**3.1178**	9	12.68857754
W2	**6.0641**	26	21.65640783
W3	**9.9446**	15	16.94107435
W4	**3.5518**	20	**20.29778313**
W5	**4.0237**	25	21.9544984
W6	**10.0512**	24	29.84962311

Table 1 shows the results for the global planning using the A* algorithm, which we use as a baseline for the comparison with the other two implementations. A* shows the best planning time performance. However, we observe that A* generates multiple coordinates between waypoints, as these are the points in which the robot turns to avoid obstacles. Finally, the last column presents the total distance traveled by the robot in between waypoints.

Table 2. A* with Diagonals Results.

Waypoint	Time (ms)	Number of Coordinates	Total Distance (m)
W1	4.0906	**7**	**9.38083152**
W2	11.9102	**17**	**12.32882801**
W3	21.9023	24	**15.23154621**
W4	4.2370	**6**	21.72556098
W5	8.2526	**18**	**19.89974874**
W6	18.8678	**20**	**14.83239697**

Table 2 shows the results for our extension of the A* algorithm using diagonals. In the table we see that this algorithm has a better performance with respect to the distance traveled and the number of points visited than the baseline A* algorithm. The gained performance in the distance (0.10× to 0.76× shorter routes) comes at the cost of having to plan the route for longer, as we now need to project the diagonals in the route, and therefore take longer in

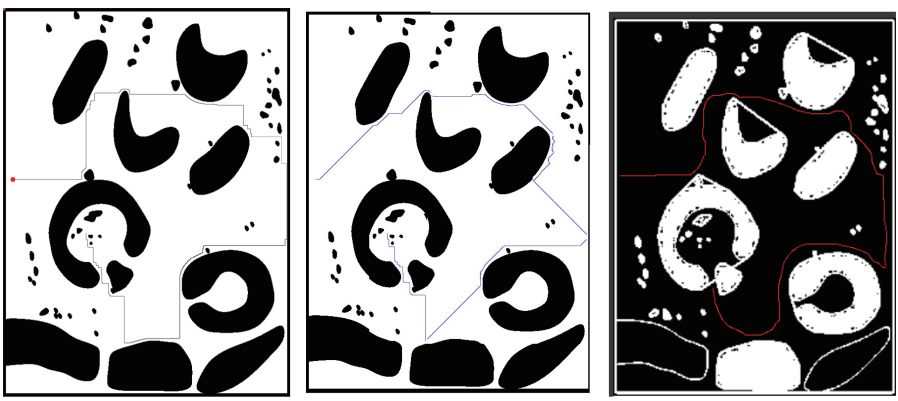
planning the route; a slowdown factor of 0.16× to 0.55×. The advantage of this algorithm is that in an unknown environment, a shorter planning route is an advantage because the rover may encounter fewer unknown obstacles.

Table 3. Convex Hull Results.

Waypoint	Time (ms)	Number of Coordinates	Total Distance (m)
W1	5.3812	11	13.6898965
W2	11.9906	30	21.98456783
W3	17.0895	27	18.5643461
W4	5.0672	25	25.2873613
W5	6.3678	30	23.95444345
W6	25.9430	27	34.1131986

The convex hull algorithm has the worst performance results across the three metrics, shown in Table 3. This can be explained as to calculate the route, we first calculate the convex hull, and then use A* (or A* with diagonals) to plan the route. Moreover, as we use either A* or A* with diagonals to plan the route, the distance covered would be at best that of the original algorithm, disproving our hypothesis that we generate less intermediate points traveling a shorter distance.

Figure 9 shows the route planned for the 6 way-points using each of the aforementioned global algorithms.



(a) Simple A* (b) A* with diagonals (c) Convex Hull

Fig. 9. Comparison of the planed routes

3.2 Dynamic Planning Results

For dynamic planning we concentrate on the evaluation of the ABEO algorithm by means of a simulation placing unknown obstacles in the path between waypoints, as shown in Fig. 10. The initial simulation environment tests the algorithm in 10×10 and 20×20 grids, before testing them on the Gazebo. We focus on the ABEO algorithm, as this is the algorithm that presents the best opportunities to sort obstacles with a lower computational cost. In our evaluation we focus on the success of the robot to avoid obstacles placed in the route between waypoints. To do this, we run ABEO 60 times starting randomly from any of the yellow numbered points in the Fig. 10, to reach the green square in the middle. All blue squares are obstacles unknown to the robot. The result of this simulation is that all executions successfully reach the objective without crashing into any of the obstacles.

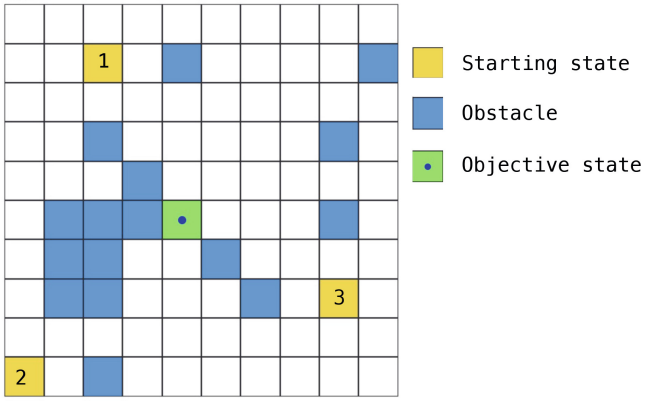


Fig. 10. Jupyter Notebook simulation environment

Once tested, we used ABEO on the Gazebo simulator (Fig. 11), where we observed that the robot can indeed reach the destination in most cases. Inaccuracies in reaching the goal are due to terrain conditions that cause the robot to tip over, or get stuck in a hill because the motor has insufficient power to climb, these factors are not related to the algorithm's performance but to the robot's specification and the environment conditions.

4 Related Work

There exist different route planning algorithms for autonomous navigation, as the ones already discussed in the previous sections. In this section we present further algorithms relevant to route planning in unknown environments, and put them in perspective of our work.

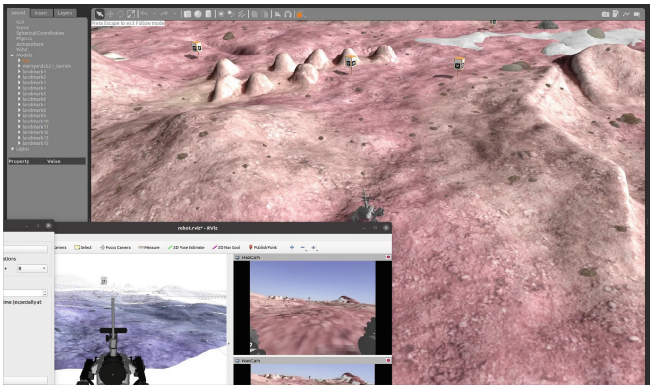


Fig. 11. Video of the ABEO simulation

Planning algorithms are split into two categories, global planning and local planning [1], matching the division we use in this paper. Global planning refers to plans that require geographic information, gathered from sensors, and a global localization of vehicles/robots. This allows us to set a defined route and see the robot's position in every step of said route. Local planning is based on relative positioning (of the robot and obstacles) to avoid obstacles effectively. In our work, this division is used to define routes between waypoints globally, and sorting obstacles locally (to return to the planed route afterwards). Global planning algorithms are off-the-shelf standards with the Dijkstra or A* algorithm as a base, optimized with an heuristic. This is similar to our algorithm, where we use A* with diagonals as an optimization heuristic. Therefore, the results using other variants, should be comparable with the ones we obtained.

Local planing algorithms have a larger variability, as we now discuss. A bio-inspired algorithm for local planning and emergence is that of ant colony behavior [5]. Route planning using ant colony algorithms is based on the probabilistic states of transition and a strategy based on the pheromones of the ants, which are used to follow a functional route planning algorithm. The behavior of ant colony algorithms is similar to the approach taken by us. However, the implementation of ABEO is simpler as we do not require to encode pheromones, which would require a sensor signaling in the robot.

A second category of algorithms consists of field-boundary definition algorithms, in which a boundary field is build around the robot/obstacle to avoid collisions. Dynamically, the robot is pushed away from high density boundary detection (*i.e.,* obstacles). Exemplars of these algorithms are the bubble rebound algorithm [8], and ODG-PF [1], discussed previously.

Finally, genetic algorithms are also used for obstacle avoidance. Here, a genetic search technique is used for a faster execution to find a way of avoiding an obstacle [3]. This algorithm presents a strategy called *survival of fitness* to determine the best solution over a set of competing potential solutions. The alternative of genetic algorithms adds the complexity of defining the fitness function

and population to generate the potential solutions, but is an interesting possibility to study further.

5 Conclusion and Future Work

Route planning in unknown environments is a challenging task, as autonomous vehicles/robots may find a variety of obstacles blocking their path, and possibly get stuck. Unfortunately, these situations are common in rover space missions or disaster area exploration, where no defined satellite images are available to plan a route avoiding all obstacles. To tackle the complexity of moving through unknown environments, in this work, we use a combination of global and dynamic planning to route robots. We use global planning optimizing the route that the robot should follow, taking into account the trade off between the routing algorithm's execution time and total distance traveled. Our experiments show that while the A* algorithm with diagonals is between 0.16× and 0.55× slower than the A* baseline in calculating the route, the distance traveled by the robot is between 0.10× to 0.76× shorter (in most cases). Therefore, we conclude that it is better to use the A* with diagonals algorithm for the global planning, as this is the algorithm that optimizes time overall. Now, since we know obstacles may appear at run time, the robot uses the ABEO dynamic planning algorithm to sort obstacles in the way. Our evaluation shows that in a small simplified simulation environment, the robot is always able to avoid unknown obstacles and reach its destination. However, we note that in the simulated environment, the robot does not always reach its goal, due to terrain conditions and the physical capabilities of the robot, which are not included in the simulation.

As future work, we foresee the further evaluation of dynamic planning algorithms to be able to identify the optimal algorithm for the fastest route, with the highest success rate. Finally, we require a real-world experimental environment to work with our rover and make tests to fine tune the algorithms.

Acknowledgment. Laura Rodriguez acknowledges the support of a UniAndes-DeepMind Scholarship 2023

References

1. Cho, J.H., Pae, D.S., Lim, M.T., Kang, T.K.: A real-time obstacle avoidance method for autonomous vehicles using an obstacle-dependent gaussian potential field. J. Adv. Transp. 1–16 (1: school of Electrical Engineering. Korea University, Seoul, Republic of Korea (2018)
2. Garage, W.: Ros robot operating system. https://www.ros.org/ (12 2007). Stanford Artificial Intelligence Laboratory Open Robotics
3. Han, W.G., Baek, S.M., Kuc, T.Y.: Genetic algorithm based path planning and dynamic obstacle avoidance of mobile robots. In: International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation, vol. 3, pp. 2747–2751 (1997). Intelligent Control and Dynamic Simulation Lab

4. Hart, P.E., Nilsson, N.J., Raphael, B.: A formal basis for the heuristic determination of minimum cost paths. IEEE Trans. Syst. Sci. Cybern. **4**(2), 100–107 (1968)
5. Meng, X., Zhu, X.: Autonomous obstacle avoidance path planning for grasping manipulator based on elite smoothing ant colony algorithm. Symmetry **14**(9), 1–20 (2022)
6. Sklansky, J.: Finding the convex hull of a simple polygon. Pattern Recogn. Lett. **1**(2), 79–83 (1982)
7. Susa Rincon, J.L., Ramos, D.: (ABEO) - Algoritmo Bioinspirado de Evasión de Obstáculos. Tecnura **13**(25), 36–47 (2009)
8. Susnea, I., Minzu, V., Vasiliu, G.: Simple, real-time obstacle avoidance algorithm for mobile robots. In: International Conference on Computational Intelligence, Man-machine Systems and Cybernetics. World Scientific, Engineering Academy e Society, pp. 24–29. WESEAS 2009 (2009)