Path Planning for Autonomous Vehicles in Uneven Terrain

Mushty Sri Sai Kaushik
University of Maryland
College Park, MD, USA
kmushty@terpmail.umd.edu

Divyam Garg
University of Maryland
College Park, MD, USA
dgarg1@umd.edu

John DiNofrio
University of Maryland
College Park, MD, USA
johnjdino@gmail.com

***Abstract*—Finding the shortest feasible path between two locations is a common problem statement in many real-world applications. Previous studies have shown that mobile platforms would consume excessive energy when moving along shortest paths on uneven terrains, which often consist of rapid elevation and terrain changes. However, this poses a problem to many mobile robots that have limited available energy sources. This paper compares and contrasts the performance of a new heuristic search algorithm called Constraints Satisfying A* [4] (CSA*) with the standard A* algorithm. The contrast is also compared with the behaviour of standard Dijkstra algorithm to understand the difference in energy consumption in a predefined terrain. The test simulation for the outputs was done on Robot Operating System using Turtlebot on a standard environment that represents the terrain changes through reaction forces.**

I. Introduction

Path planning for an autonomous vehicle involves the decision making of the robot to move from point A to point B in a given path based on several factors including obstacles, path and many more. However, most autonomous vehicles consider the shortest path and not the terrain of the path.

The terrain changes brings about an additional challenge of consuming extra energy considering the additional frictional constraints that they pose. The approach of this paper is to quantify the energy difference and understand the performance of an energy-based heuristic function in comparison to the behavior of a distance-based heuristic function as defined in a standard A* algorithm.

II. Approach

The Constraint Satisfying A* algorithm [4] in this paper considers heuristic cost as a factor of Energy (kJ) required to traverse a given path. This is obtained from the standard work equation

$$(W = F \times d) \tag{1}$$

Where,
F = force and
d = distance traveled
The equations we have used consider different reaction forces for each terrain. The considered terrains are as follows:

List of Terrain
Inclined up
Inclined down
Rough
Smooth
Inclined up and rough
Inclined up and smooth
Inclined down and rough
Inclined down and smooth

For the purposes of the paper the terrains were arranged to understand the optimal behavior of both algorithms along with the Dijkstra algorithm [8].

The force equations for each terrain was defined based on the behavior of the terrain. The frictional coefficient is estimated based on the type of terrain with the standard value for standard terrain to be μ = 0.1 and μ_0 = 0.08 for smooth surface along with μ_1 = 0.4 for rough surface. The equations were written for the aforementioned parameters as follows:

Terrain	Equation
Inclined up	$F = mg\sin\theta + \mu mg\cos\theta$
Inclined down	$F = -mg\sin\theta + \mu mg\cos\theta$
Rough	$F = \mu_1 mg$
Smooth	$F = \mu_0 mg$
Inclined up and rough	$F = -mg\sin\theta + \mu_0 mg\cos\theta$
Inclined up and smooth	$F = mg\sin\theta + \mu_1 mg\cos\theta$
Inclined down and rough	$F = -mg\sin\theta + \mu_1 mg\cos\theta$
Inclined down and smooth	$F = mg\sin\theta + \mu_0 mg\cos\theta$

A. *Constraints and conditions*

The constraints considered in modeling the equations are based on physical parameters of velocity, time and force.

a) Force: The general force constraints considered for the modeling of the force equations mentioned for each terrain are given as:
There are two main force constraints that are used for the purposes of this paper. They include -
(1) Force due to gravity for inclined surfaces as shown below:

$$F_g = mg\sin\theta \tag{2}$$

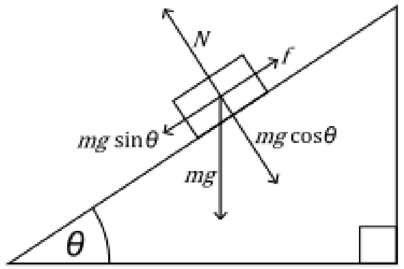


Fig. 1. Forces acting on a body on an inclined surface

(2) Force due to friction of the surface

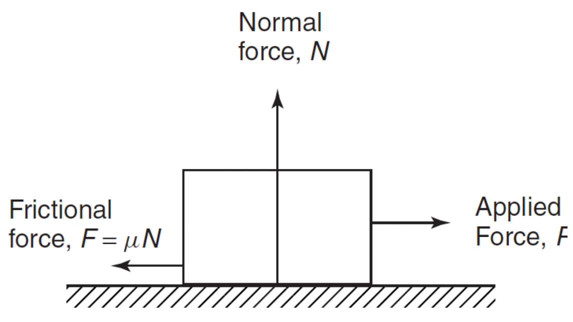


Fig. 2. Forces acting on a body on an inclined surface

$$F_f = mg\mu \tag{3}$$

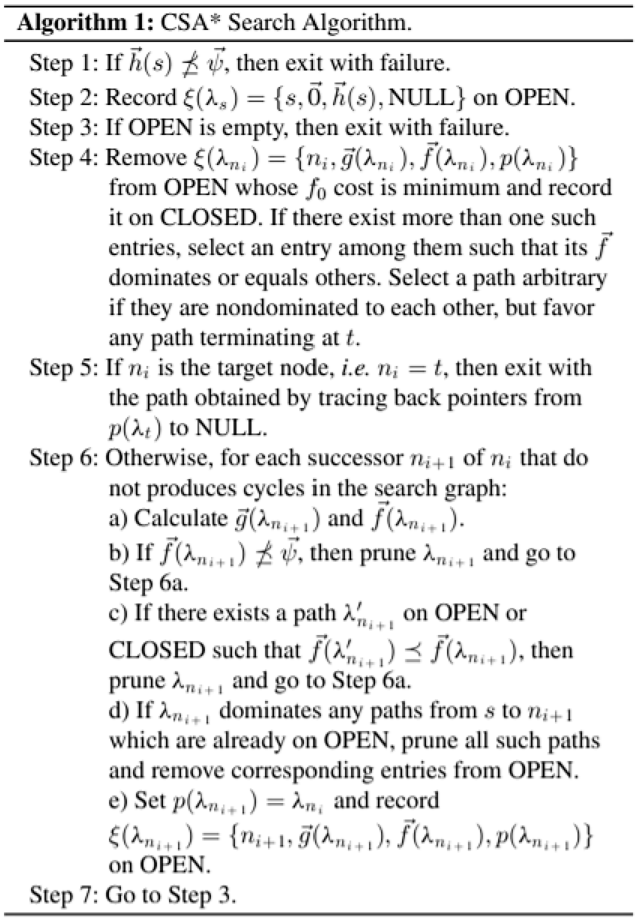
Where, μ = frictional coefficient

Other general constraints imposed are:
1) $F_{Robot} > F_g$
2) $F_{Robot} \leq F_f$
3) $1.75N \leq F_{Robot} \leq 13N$

b) Velocity: The velocity of the robot is considered to be constant in this simulation. This implies that the parameters that vary are Distance and time. In order to reach the constant velocity, the robot has to attain higher RPM values which is directly proportional to the work done by the Robot [2]. Thus, for terrains that are (a) Rough (b) Inclined up, the work done due to the higher RPMs that are required to be reached in order to overcome the surface parameters. Similarly, for terrains that (c) Smooth (d) Inclined down, the work done is less.

III. Algorithm

The CSA* algorithm is quite similar to the A* except that it takes on multiple heuristic costs while satisfying all constraints. A* typically works by calculating the Euclidean distance from the start to the current node (cost to come) and the current node to the goal node (cost to go) [1]. When it revisits a node that its already found, it checks if the cost to come is cheaper than the previous path. CSA* performs the same iterations but has to keep track of several cost to comes. At the same time, it must satisfy any constraints is has on it, hence why it is called the Constraints Satisfying A* algorithm. The heuristic value is normalised according to the configuration space for a more accurate value. The configuration space of 100*100 is considered for 10*10 meters.



```
Algorithm 1: CSA* Search Algorithm.
Step 1: If $\vec{h}(s) \not\preceq \vec{\psi}$, then exit with failure.
Step 2: Record $\xi(\lambda_s) = \{s, \vec{0}, \vec{h}(s), \text{NULL}\}$ on OPEN.
Step 3: If OPEN is empty, then exit with failure.
Step 4: Remove $\xi(\lambda_{n_i}) = \{n_i, \vec{g}(\lambda_{n_i}), \vec{f}(\lambda_{n_i}), p(\lambda_{n_i})\}$
        from OPEN whose $f_0$ cost is minimum and record
        it on CLOSED. If there exist more than one such
        entries, select an entry among them such that its $\vec{f}$
        dominates or equals others. Select a path arbitrary
        if they are nondominated to each other, but favor
        any path terminating at $t$.
Step 5: If $n_i$ is the target node, i.e. $n_i = t$, then exit with
        the path obtained by tracing back pointers from
        $p(\lambda_t)$ to NULL.
Step 6: Otherwise, for each successor $n_{i+1}$ of $n_i$ that do
        not produces cycles in the search graph:
        a) Calculate $\vec{g}(\lambda_{n_{i+1}})$ and $\vec{f}(\lambda_{n_{i+1}})$.
        b) If $\vec{f}(\lambda_{n_{i+1}}) \not\preceq \vec{\psi}$, then prune $\lambda_{n_{i+1}}$ and go to
        Step 6a.
        c) If there exists a path $\lambda'_{n_{i+1}}$ on OPEN or
        CLOSED such that $\vec{f}(\lambda'_{n_{i+1}}) \preceq \vec{f}(\lambda_{n_{i+1}})$, then
        prune $\lambda_{n_{i+1}}$ and go to Step 6a.
        d) If $\lambda_{n_{i+1}}$ dominates any paths from $s$ to $n_{i+1}$
        which are already on OPEN, prune all such paths
        and remove corresponding entries from OPEN.
        e) Set $p(\lambda_{n_{i+1}}) = \lambda_{n_i}$ and record
        $\xi(\lambda_{n_{i+1}}) = \{n_{i+1}, \vec{g}(\lambda_{n_{i+1}}), \vec{f}(\lambda_{n_{i+1}}), p(\lambda_{n_{i+1}})\}$
        on OPEN.
Step 7: Go to Step 3.
```

Fig. 3. Psuedocode for CSA* algorithm

The pseudocode for CSA* [4]is shown in Fig. 3. The symbol f(λ) is the expected cost for reaching a point in the search graph and g(λ) is its corresponding cost vector. Even though a path might have a low g(λ), it might violate one of the constraints making the path invalid. Because of this, undesirable paths may become more desirable as the search goes on. This also means that paths with higher g() might be the most optimized route even though shorted paths beat them. If even one constraints is broken, the pass is considered impassable.

IV. Map and Terrain

In order to create the terrain for testing the robot, multiple planar equations were used to identify different types of terrain. Fig. 4 shows what the final output for the map should look like. Each colored region represents a different kind of terrain [7]. Anytime the robot enters one of these areas, the new constraints take affect immediately. The brown smooth incline is also too steep and slippery for the robot to climb, so the robot should not be able to pass through it if it follows the constraints. Fig. 4 shows the terrain, but this is just for visual affect because the vector path graphs do not display terrain. For future installments of this project, more sophisticated displays of terrain may add easier understanding of the map as well as appeal to readers [6]. The map in the Gazebo simulation is also not decorated with a 3D landscape. All environment features are computed while the algorithm runs but not displayed in the outputs.

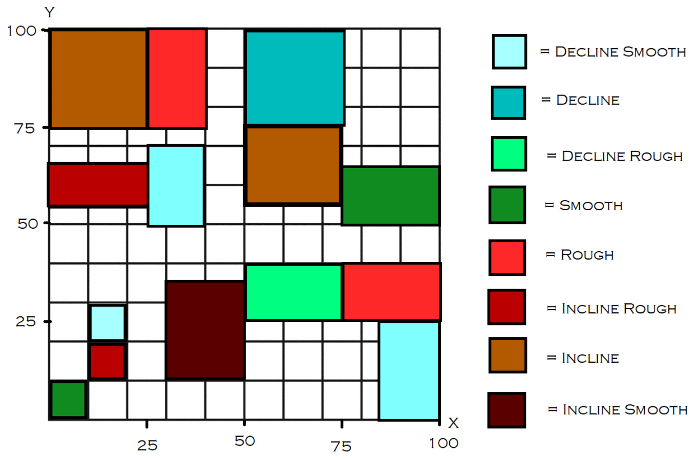


Fig. 4. Colored coded visual of the terrain environment

V. Results

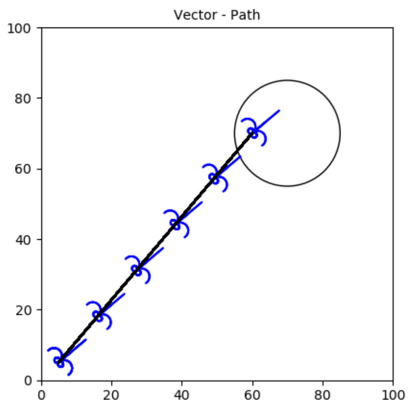


Fig. 5. Path for A*

CSA* performed extremely well in the experiment. Fig. 5 and Fig. 6 show the two paths that the robots took. Though they do not seem much different, CSA* was able to reduce energy use by an alarming amount without sacrificing practically any time or distance. As shown in Fig. 7, the energy consumed for the given simulation space is found to be a value of 74.6 kJ for the robot using A*, in comparison to the 17.78 kJ used by the robot using CSA* in Fig. 8. CSA* was only 1.5 seconds

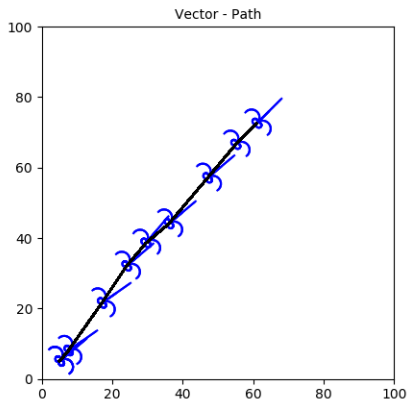
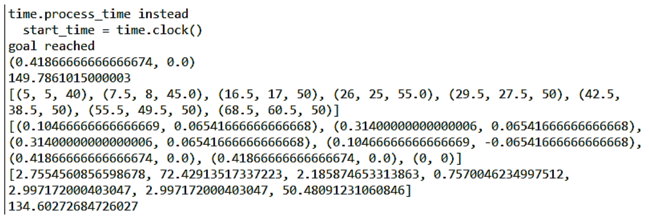


Fig. 6. Path for CSA*



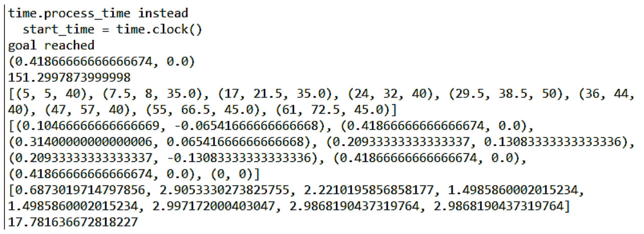
```
time.process_time instead
  start_time = time.clock()
goal reached
(0.41866666666666674, 0.0)
149.7861015000003
[(5, 5, 40), (7.5, 8, 45.0), (16.5, 17, 50), (26, 25, 55.0), (29.5, 27.5, 50), (42.5,
38.5, 50), (55.5, 49.5, 50), (68.5, 60.5, 50)]
[(0.10466666666666669, 0.06541666666666668), (0.31400000000000006, 0.06541666666666668),
(0.31400000000000006, 0.06541666666666668), (0.10466666666666669, -0.06541666666666668),
(0.41866666666666674, 0.0), (0.41866666666666674, 0.0), (0, 0)]
[2.7554560856598678, 72.42913517337223, 2.185874653313863, 0.7570046234997512,
2.997172000403047, 2.997172000403047, 50.48091231060846]
134.60272684726027
```

Fig. 7. Output for A*

slower than A* in finding the goal which equates to one percent slower solve speed [5]. This is practically negligible. The same goes for the increased distance that CSA*. With only 1.9 m longer in its path, or two percent increase in distance, the algorithm used a quarter of the energy consumption that A* used. This dramatic decrease in energy consumption while maintaining exceptionally optimized distance and time heuristics is proof that CSA* is more efficient than A* when it comes to multi-cost scenarios.

The second area which CSA* outperformed A* is in the constraints. CSA* did not violate a single constraint. A* on the other hand traveled through a steep incline with smooth ground, an area that our robot cannot pass through. While satisfying the constraints, the CSA* even managed to reduced its energy use by finding downhill slopes. In no way did A* outperform CSA*.

Fig. 9 shows how the CSA* program avoided the smooth uphill area as well as minimizing its time in the uphill rough



```
time.process_time instead
  start_time = time.clock()
goal reached
(0.41866666666666674, 0.0)
151.2997873999998
[(5, 5, 40), (7.5, 8, 35.0), (17, 21.5, 35.0), (24, 32, 40), (29.5, 38.5, 50), (36, 44,
40), (47, 57, 40), (55, 66.5, 45.0), (61, 72.5, 45.0)]
[(0.10466666666666669, -0.06541666666666668), (0.41866666666666674, 0.0),
(0.31400000000000006, 0.06541666666666668), (0.20933333333333337, 0.13083333333333336),
(0.20933333333333337, -0.13083333333333336), (0.41866666666666674, 0.0),
(0.41866666666666674, 0.0), (0, 0)]
[0.6873019714797856, 2.9053330273825755, 2.2210195856858177, 1.4985860002015234,
1.4985860002015234, 2.997172000403047, 2.9868190437319764, 2.9868190437319764]
17.781636672818227
```

Fig. 8. Output for CSA*

region. By doing this, it also was able to maximize its time in the downhill smooth area which reduces energy reduce. Fig. 10 shows A* just finding the most direct and shortest route without avoiding high energy cost areas.

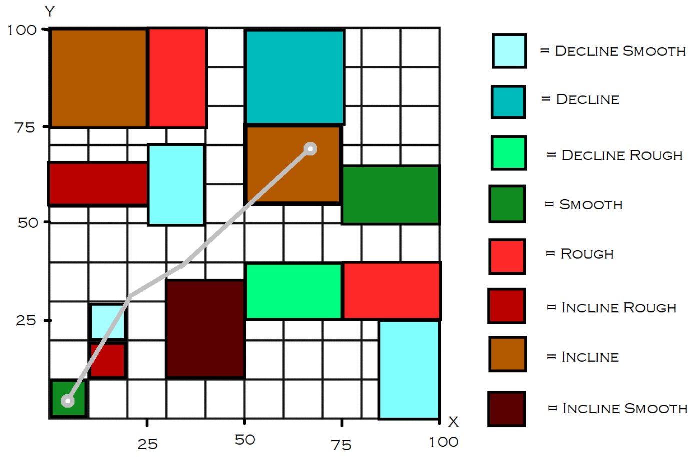


Fig. 9. Visual output for CSA* with terrain

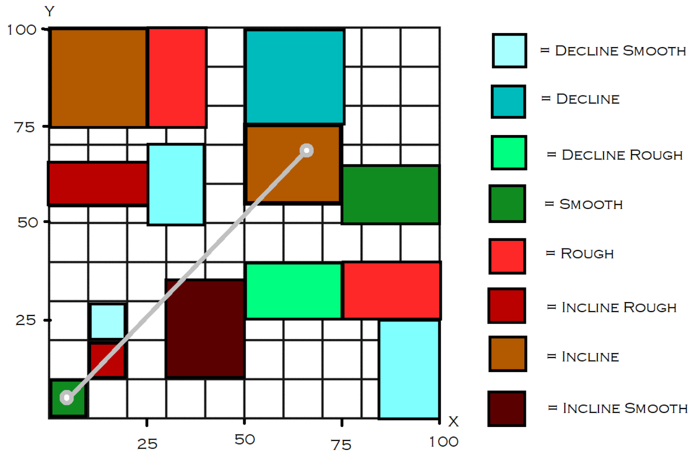


Fig. 10. Visual output for A* with terrain

VI. Simulation

The simulation is done using the values of mass, maximum force exerted and RPMs are considered according to the specifications of TurtleBot3. The simulation is done with differential drive constraints with 8 action set for different RPM value namely R1 = 50 and R2 = 100. The action-set are as follows: - [(0,R1) (0,R2) (R1,0) (R2,0) (R1,R1) (R2,R2) (R1,R2) (R2,R1)] For representation purposes, configuration space of 10*10 meters is considered. The simulation is done in ROS - gazebo environment where, TurtleBot3 - burger is used as the acting robot. The final outputs published to the ROS nodes are linear and angular velocities. Since gazebo environment doesn't allow different terrain, to conclude the difference between A* and CSA* both were simulated with same start and goal point.

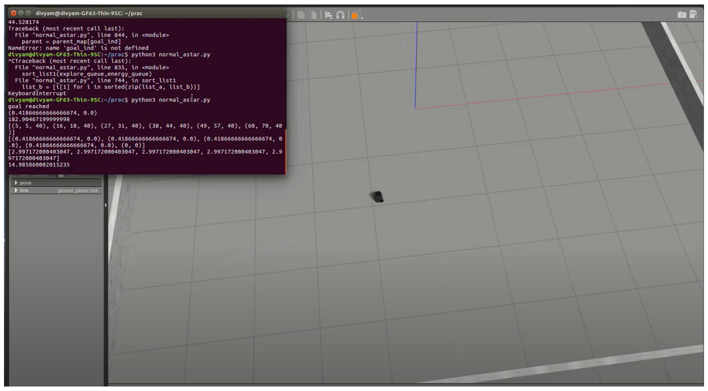

```
divyam@divyam-GF63-Thin-9SC: ~/prac
44.528174
Traceback (most recent call last):
  File "normal_astar.py", line 844, in <module>
    parent = parent_map[goal_ind]
NameError: name 'goal_ind' is not defined
divyam@divyam-GF63-Thin-9SC:~/prac$ python3 normal_astar.py
^CTraceback (most recent call last):
  File "normal_astar.py", line 835, in <module>
    sort_list1(explore_queue,energy_queue)
  File "normal_astar.py", line 744, in sort_list1
    list_b = [i[1] for i in sorted(zip(list_a, list_b))]
KeyboardInterrupt
divyam@divyam-GF63-Thin-9SC:~/prac$ python3 normal_astar.py
goal reached
(0.41866666666666674, 0.0)
182.90467199999998
[(5, 5, 40), (16, 18, 40), (27, 31, 40), (38, 44, 40), (49, 57, 40), (60, 70, 40
)]
[(0.41866666666666674, 0.0), (0.41866666666666674, 0.0), (0.41866666666666674, 0
.0), (0.41866666666666674, 0.0), (0, 0)]
[2.997172000403047, 2.997172000403047, 2.997172000403047, 2.997172000403047, 2.9
97172000403047]
14.985860002015235
```

Fig. 11. Simulation of the output in ROS

VII. Future Work

As a part of the possibilities that can be built on this project, there are several improvements that can be made to our algorithm in terms of considering various other real world terrains. Another important upgrade would be giving the robot the ability to identify the type of terrain using LiDAR in order to give it more intelligence. the basis of the code is with the idea that it already knows the type of terrain, however, this is not the case in the real-world scenario. Finally, after updating all the changes mentioned above, we can create our environment using PyBullet - A real-time simulation Software for Python - to create a more accurate representation of the conditions.

References

[1] Mohamed Saad, Ahmed I. Salameh, and Saeed Abdallah. *Energy-Efficient Shortest Path Planning on Uneven Terrains: A Composite Routing Metric Approach*.
2019 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT), Ajman, United Arab Emirates, 2019, pp. 1-6

[2] Sebastian Thrun, Michael Montemerlo, James Diebel *Path Planning for Autonomous Vehicles in Unknown Semi-structured Environments*.
The International Journal of Robotics Research, 29(5), 485–501.

[3] Rekleitis, Ioannis Bedwani, Jean-Luc Dupuis, Erick Allard, Pierre. *Path Planning for Planetary Exploration*.
Proceedings of the 5th Canadian Conference on Computer and Robot Vision, CRV 2008. 61-68. 10.1109/CRV.2008.46.

[4] N. Ganganath, C. Cheng, T. Fernando, H. H. C. Iu and C. K. Tse *Shortest Path Planning for Energy-Constrained Mobile Platforms Navigating on Uneven Terrains*.
IEEE Transactions on Industrial Informatics, vol. 14, no. 9, pp. 4264-4272, Sept. 2018, .

[5] A. S. Santos, H. Ignacio Perez Azpúrua, G. Pessin and G. M. Freitas *Path Planning for Mobile Robots on Rough Terrain*.
Latin American Robotic Symposium, 2018 Brazilian Symposium on Robotics (SBR) and 2018 Workshop on Robotics in Education (WRE), Joao Pessoa, 2018, pp. 265-270,

[6] Belaidi, H., Bentarzi, H., Belaidi, A. et al. *Terrain Traversability and Optimal Path Planning in 3D Uneven Environment for an Autonomous Mobile Robot*.
Arab J Sci Eng 39, 8371–8381 (2014).

[7] Dorian J. Spero, Ray A. Jarvis. *Path Planning for a Mobile Robot in a Rough Terrain Environment*.

[8] Nicholas Melchior, Jun-young Kwak and Reid Simmons. *Particle RRT for Path Planning in Very Rough Terrain*.
Conference Paper, Proceedings of the NASA Science Technology Conference 2007 (NSTC-07), May, 2007