Towards efficient path planning of a mobile robot in rough terrain [1]

Diogo Amorim
diogo.amorim@ist.utl.pt
Rodrigo Ventura
rodrigo.ventura@isr.ist.utl.pt

Instituto Superior Técnico - Technical University of Lisbon
Lisbon, Portugal

Abstract

Path planning for autonomous vehicles on rough terrain is a different problem to solve comparing to traditional 2D, flat terrain path planning. We address this problem by first constructing a cost grid map where each cell represents a passage/progress cost for the robot rather than, for example, an occupancy grid map given a known surface representation. The cost value for each (x, y, θ) is based on the pose of the vehicle when on the surface and it results from the pitch angle. One possible way of calculating the robot's pose is to define that as a constrained optimization problem with non- linear constraints.

1 Introduction

This paper shows the steps being taken in order to create a path planning method for a robot presented with a rough terrain. Path planning for 2D maps is usually based on an occupancy grid map with a (x, y, θ) configuration where each entry depicts whether or not that position is free to the robot [7], and the total path cost is usually the distance until the goal position or some simple variation of that value. There already are some path planning methods that solve 2D map path planning problems which are powerful tools [6], and so, part of the intention of this project is to apply these tools to the same sort of problem, and with the same goal, but with a different premise. This premise is the type of map, a map that represents a rough surface has information about the free space and insuperable obstacles but also the elevation of each coordinate. The elevation variations can implicate new obstacles, i.e., if a slope it too steep the vehicle will not be able to climb it.
The purpose of this paper is to show how to create a map in a (x, y, θ) configuration where each coordinate is associated with a cost based on the vehicle's pose if it were to rest on those coordinates. Defining the problem as a constrained optimization problem with non-linear constraints it is possible to determine the pose of the mobile robot as well as the number of contact points with the ground, an important factor to determine whether or not it is possible for it to stand on that position.
After creating a map of the robot's world depicting these degrees of difficulty one could, in principle, apply a number of path planning methods already in existence. The final goal of this project is to develop a controller for the RAPOSA-NG [2], a track wheel robot designed for urban search & rescue operations, that can be equipped with depth sensors. The controller will efficiently drive the robot from point A to point B in any rough terrain.

2 Problem Statement

At this stage we are still at a proof of concept stage where the environment is all simulated. The map is in a (x, y, z) configuration, is defined in the inertial frame of reference (i) and is derived from a sinusoidal surface easily obtainable from the mpl_toolkits package for python. The resulting surface has a bump like shape with a smooth slope and a sharp end as shown in Figure 1. The robot is defined as a 7 point $p_{j,r} = (x_{j,r}, y_{j,r}, z_{j,r})$ structure Figure 1, all fixed to its reference frame, the robot's reference frame (r) where the origin is set at the center of mass of the robot. Three points characterize each contact point of a track, one represents its beginning $(p_{1,r}, p_{2,r})$, another its end $(p_{3,r}, p_{4,r})$ and another its middle point $(p_{5,r}, p_{6,r})$. The last point $(p_{7,r})$ represents the vehicle's center of mass, which, as previously mentioned, was described as the origin of the frame. The robot's pose is defined as :

$$(x_{7,i}, y_{7,i}, z_{7,i}, \theta, \beta, \gamma) \tag{1}$$

where $(x_{7,i}, y_{7,i}, z_{7,i})$ are the coordinates of the r in relation to the i and θ is the angle of rotation of the r around the Z_r axis, β the Y_r axis and γ the X_r axis. To determine the coordinates of the points defining the robot on the i their coordinates in r are multiplied by a rotation matrix:

$$\begin{bmatrix} cos(\beta)\cdot cos(\theta) & cos(\theta)\cdot sin(\gamma)\cdot sin(\beta) - cos(\gamma)\cdot sin(\theta) & cos(\gamma)\cdot cos(\theta)\cdot sin(\beta) + sin(\gamma)\cdot sin(\theta) \\ cos(\beta)\cdot sin(\theta) & cos(\gamma)\cdot cos(\theta) + sin(\gamma)\cdot sin(\beta)\cdot sin(\theta) & -cos(\theta)\cdot sin(\gamma) + cos(\gamma)\cdot sin(\beta)\cdot sin(\theta) \\ -sin(\beta) & cos(\beta)\cdot sin(\gamma) & cos(\gamma)\cdot cos(\beta) \end{bmatrix} \tag{2}$$

and then adding the position of the r relative to the i.

The target is to minimize the z coordinate of the robot's center of mass, in relation to the i but with the restriction that none of the points that define the robot can pass through the surface. The vehicle also has pose limitations, it cannot be upside down and cannot climb hills steeper than 45 degrees (value stipulated for the simulation) so its roll and pitch angle absolute maximums were set at that same value. These limitations are translated as constrictions when inserted in an optimization problem, and so, in order to solve this specific problem one can resort to a constrained (multivariate) problem solving routine already available and developed.
In order to introduce constraints in the optimization function it is necessary to formulate them as inequalities, functions whose values are always positive which in this case means, for example, that the z coordinate of each point defining the robot minus the z coordinate of the point of the map directly below must be positive, and this condition is respected by the algorithm. There are 9 constraints to this simple problem, one per each point that defines the robot, one for the roll and another for pitch angle. More constrictions can and will be added in order to simulate the hull of the vehicle more accurately simulating it. The task of determining the robot's pose can then be defined as a constrained optimization problem in the following way:

$$\begin{array}{ll} \text{Minimize:} & z_{7,i} \\ \text{Variables:} & z_{7,i},\ \beta,\ \gamma \\ \text{Subject to:} & z_{j,i} - map_{x_{j,i}, y_{j,i}} \geq 0 \\ & \frac{\pi}{4} - |\beta| \geq 0 \\ & \frac{\pi}{4} - |\gamma| \geq 0 \end{array} \tag{3}$$

The optimization function minimizes the value of $z_{7,i}$ by manipulation of the three variables it has access to: ($z_{7,i}, \beta$ and γ). The constrictions determine that the robot's pitch (γ) and roll (β) angles don't reach values greater than 45° ($\frac{\pi}{4}$ *rad*) or smaller than -45° ($-\frac{\pi}{4}$ *rad*). Another limitation is that none of the z coordinates of the points defining the robot ($z_{j,i}$) can be lower than the elevation of the map directly bellow ($map_{x_{j,i}, y_{j,i}}$) thus $z_{j,i} - map_{x_{j,i}, y_{j,i}}$ must be grater than 0. The results are as expected, the robot touches the ground with three or more of the six points defined as its tracks depending on the surface roughness. All those are valid positions, but if the function returns that only two or less points are touching the surface that means that the autonomous system can't be on that position because of pose limitations introduced as constraints and that same position on the map is considered an obstacle. It is now possible to build a new map where instead of z coordinates we use a combination of the pose angles (pitch and roll) provided by the previous routine. For every cell of the map, i.e., every time the optimization function returns a pose, the value of γ is stored in the equivalent cell of the cost map. If we were to keep the absolute value of γ , we would be admitting the cost of moving uphill or downhill on a slope with equal inclination is the same. It is intuitive to say the robot will struggle more going uphill then going downhill and this is why we kept the information about the sign of γ. We now have the information about where the map, at a certain θ, is up or downhill. At the end of the process of calculating the pose of the robot for every map cell, the lowest cost value is found and added to every cost map cell, this action prevents the existence of negative cost values. For each possible θ angle of the robot a different cost map will be produced by the algorithm, and so there is not just one 2D matrix depicting the new map, but as many

[1] This work was supported by the FCT project [$PEst - OE/EEI/LA0009/2013$]
[2] http://mediawiki.isr.ist.utl.pt/wiki/RAPOSA_robot#RAPOSA_NG

as the possible θ angles the robot can assume. All these matrices can form a 3D matrix where each layer is a the map for a specific θ angle.

3 Results

The process explained above was applied to the already mentioned bump like surface. This map of 1225 cells takes about 0.013 seconds/cell to be processed but there is still room for speed improvement as we are still developing concepts.

As we can see from the contour map on Figure 2 (a) of the bump the cross sections are almost tear shaped and this explains the behavior of Figure 2(b). The cost map obtained for $\theta = 0$ from the algorithm is shown in Figure 2(b), it shows cost values ranging from dark blue to dark red as shown on the side bar. Higher cost values coincide with steeper uphill slopes and lower cost figures translate steeper downhill slopes. This cost map is only applicable to a robot travelling bearing $\theta = 0$, so one could imagine a robot starting from $(0,-30)$ (bottom of the graph) and travelling to $(0,20)$ or in lines parallel to that. The light green/yellow line demonstrates the beginning of the slope, until then the cost was constant, but now the robot is climbing uphill the cost values will increase. The values go from green/yellow up to red (if the color does not change it means the map has a constant inclination). Because the surface is rounded in all edges the transition from uphill to downhill is translated by the green/light blue line, and as the surfaces becomes steeper (downhill) again the blue color darkens. At the end of the descent the color goes from light blue to green and the cost represented by the green color is the same up to the upper edge of the map indicating, again, a constant slope. The Scipy package for python includes a function (*fmin_cobyla*) that solves Constrained Optimization problems BY Linear Approximation (COBYLA) [4] and it is one of the possibilities that was used to obtain the results here shown. The fmin_cobyla respects the constrictions to a user defined error margin which here was defined as a millimetre (although the authors do not guarantee 100% accuracy in every constraint, the results for the tested surfaces were always within the defined error).

4 Conclusions and Future work

Although being a different path planning problem, the presented conundrum can be solved with adaptations of technologies already in existence. Simulating the structure of the robot and using an elevation map it is possible to create a cost map, by computing the robot's pose in every map cell. This cost map is a crucial tool for the task ahead, path planning. We are still in an early stage of the solution and because of that, some questions that arise when solving the main problem are not completely answered yet. The future work will be about the use of the cost map as an input for an already existing path planning method. It is theoretically possible to apply a Fast Marching Method (FMM) [2, 5],rapidly exploring random trees (RRT) [1, 3] or other path planning methods and obtain an efficient path for the robot.

References

[1] S. Garrido, L. Moreno, and D. Blanco. Voronoi diagram and fast marching applied to path planning. *IEEE International Conference on Robotics and Automation*, 2006.

[2] S. Garrido, L. Moreno, D. Blanco, and F. Martin. Smooth path planning for non-holonomic robots using fast marching. *IEEE International Conference on Mechatronics*, 2009.

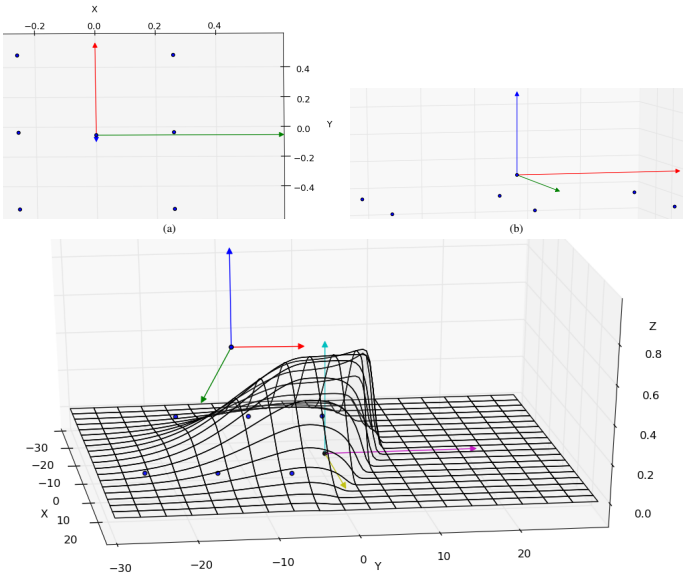
[3] S. M. LaValle and Jr. J. J. Kuffner. Randomized kinodynamic planning. *The International Journal of Robotics Research*, pages 378–400, 2001.

[4] M. J. D. Powell. A view of algorithms for optimization without derivatives. Technical report, Cambridge University, 2007.

[5] J. A. Sethian. Fast marching methods. *SIAM Review*, 41(2):199–235, 1999.

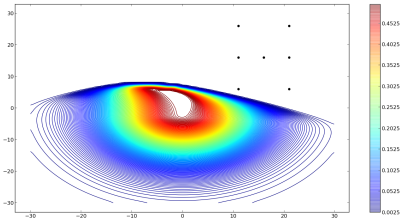
[6] Roland Siegwart, Illah R. Nourbakhsh, and Davide Scaramuzza. *Introduction to Autonomous Mobile Robots*. The MIT Press, ISBN-13: 978-0262015356, 2011.

[7] Sebastian Thrun, Wolfram Bugard, and Dieter Fox. *Probilistic Robotics*. The MIT Press, ISBN-13: 978-0262201629, 2005.

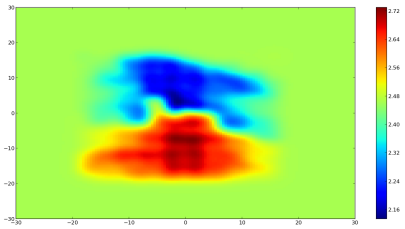


(c)

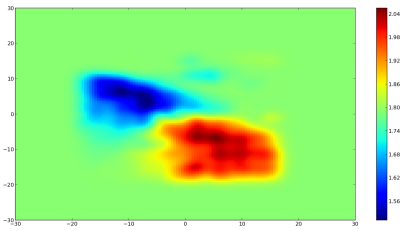
Figure 1: Robot representation in (a) by 7 points as seen from the top. Figure 1 (b) shows the positions of all 7 points in relation to the r, the blue vector displays the origin and direction of the Z_r, Y_r and X_r are represented by the red and green vectors, respectively, (c) show the relation between the two frames, the robot has a pose defined by $(10,-13,0.659,-0.523,1.188,0.213)$ (note that the Z axis scale is not the same as the X's and Y's to better visualize the relation between frames)



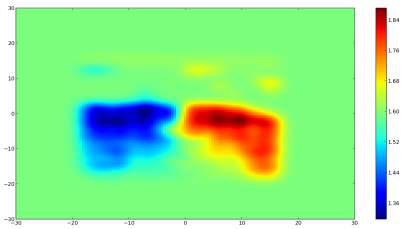
(a)



(b)



(c)



(d)

Figure 2: (a) depicts an elevation map of the test surface, each contour line defines a constant elevation value, dark blue for low values until dark red for the higher figures. The 7 black dots represent the robot and its scale. (b) shows a representation of the difficulty the vehicle bearing $\theta = 0$ would have at each point of the test map if it were to pass by it. The darker the red the harder, the darker the blue, the easier. Figure 2(c) and (d) represent the same as Figure 2(b) bau for $\theta = \frac{\pi}{4}$ and $\theta = \frac{\pi}{2}$. Note the graphic representation is qualitative within each θ map.