A Roughness-based RRT for Mobile Robot Navigation Planning

Adnan Tahirovic ∗ Gianantonio Magnani ∗∗

∗ Politecnico di Milano, 20133 Milan, Italy (e-mail: tahirovic@elet.polimi.it).
∗∗ Politecnico di Milano, 20133 Milan, Italy (e-mail: magnani@elet.polimi.it).

Abstract: This paper proposes a novel Rapidly exploring random tree algorithm on rough terrains (RRT-RT) for the purpose of outdoor mobile robot navigation. Differently from other RRTs adopted for rough terrains where finding a nearest neighbor from a new random state within the tree is based on Euclidian distance, the proposed algorithm uses a roughness based metric. The metric is defined by the help of the roughness based navigation function, RbNF, that represents an estimate of the roughness-to-go value from each terrain location to the goal position. Simulation study shows that the RRT-RT provides an effective way to explore more promising terrain regions in order to decrease the total roughness along the resulting path.

1. INTRODUCTION

This work considers developing a new motion and navigation planner that helps a vehicle to move from an initial position to a goal position in rough terrains. Planetary exploration using an unmanned vehicle allows for understanding the planet surface geology, its present and past climate conditions, and for discovering potential signs of other lives. Rescue missions in dangerous environments and surveillance with reduced human operations and interventions have also become interesting areas for unmanned vehicles. With the special interest within the military industry, these areas are growing rapidly and motivate new research in autonomous vehicle motion planning in difficult environments. The use of the autonomous vehicles in a de-mining process decreases the danger and the cost of manual mine detection. Both humanitarian and economic motivations to use such vehicles to this purpose are obvious. Finally, agriculture applications have recently recognized the potential to use the fully autonomous vehicles in agricultural operations reducing the total cost of the final product.

The sample-based technique for robot motion planning was introduced in Barraquand and Latombe [1991]. The first sample-based motion planners were not computationally efficient for certain environments. In Kavraki et al. [1996], Kavraki [1995], Overmars and Švestka [1995] the probabilistic roadmap method (PRM) was developed for path planning in configuration spaces with many degrees of freedom. A comprehensive overview and discussion about PRM is given in Choset et al. [2005] and Hsu et al. [2006]. PRM method has proved to work well in static well-known environments and is considered computationally efficient for car-like vehicles Song and Amato [2001]. However, PRM may not be suitable for planning in a dynamic environment, especially because it does not take into account the vehicle dynamics and might result in very sharp turning points. In Branicky et al. [2001] the authors introduced quasi-PRM and lattice roadmap (LRM) algorithms. LRM was extended in Pivtoraiko et al. [2009] to allow the state lattice to represent the differential constraints of the mobile vehicle. Rapidly-exploring random trees (RRT) is a type of probabilistic planners originally developed to cope with differential constraints Kuffner and Lavalle [1999], Lavalle and Kuffner [2000, 2001]. A significant feature of the RRT-like algorithms is that the resulting trajectories are executable by the underlying dynamical system. The RRT algorithm has been proven probabilistically complete Lavalle and Kuffner [2000], meaning that the probability of finding a solution feasible path converges to one if such a path exists. An improvement of the RRT algorithm was proposes in Hsu et al. [2001], where the obtained exponential convergence speed yielded a good performance.

The RRT algorithm was also successfully used for path planning in rough terrains in Ettlin and Bleuler [2006], Kobilarov and Sukhatme [2004], Spero and Jarvis [2002]. Differently than the path planning on flat terrains, the cost function used to find a resulting path on rough terrains may differ from the minimum path or minimum time required to reach the goal position. On rough terrains, it may be desirable to find such terrain regions to decrease side effects caused by the vehicle-terrain interaction. Hence, finding less rough terrain regions may increase maneuverability of the vehicle during its task execution. In Spero and Jarvis [2002], the authors have implemented an RRT-based planner for a non-flat terrain working only with sensor range data and without focusing on a globally optimal solution. The RRT expansion was based on the Euclidian distance. An approximately time-optimal RRT path planner in rough terrains is presented in Kobilarov and Sukhatme [2004]. The dynamic constraints of the robot have been considered, and the RRT expansion has been based on the path length metric. In Ettlin and Bleuler [2006], the authors have used the idea of local trees extension, introduced in Strandberg [2004], where the root of each new local tree and its expansion is guided by the terrain roughness. The RRT algorithm is not considered as an optimal planner but it may include a heuristic which estimates a cost-to-go from a given node on the map tree to the goal position. Such heuristics can help to grow the exploring tree toward the area which consists of states with lower cost-to-go values. The better the estimation of the cost-to-go value the more promising areas are explored yielding better solutions. According to
our knowledge, none of these approaches considers roughness based heuristics to guide the RRT expansion.

In this paper, an RRT navigation planning approach guided by roughness based navigation function, RbNF, has been proposed. The RbNF is a numerical function that provides the cost-to-go values (roughness-to-go) for each terrain location. The roughness-to-go value of a terrain location represents an approximate value of the remaining cost of the terrain traversability toward the goal position. Simulation results show that the RRT-RT planner explores the terrain in an efficient manner, and even generates final paths that slightly deviate from paths obtained by the Dijkstra’s algorithm.

How to compute a roughness-based navigation function is recalled in Section II. Section III explains the proposed RRT-RT algorithm and it is followed by the simulation study presented in Section IV. Conclusion is outlined in Section V.

2. ROUGHNESS-BASED NAVIGATION FUNCTION, RbNF

The RbNF has been proposed for the purpose of navigation planning based on the model predictive (MPC) paradigm in Tahirovic and Magnani [2011],Tahirovic and Magnani [2010]. An MPC-based navigation planner repeatedly performs the optimization within a region around the current vehicle position to generate an appropriate path toward the goal. As in case of each MPC approach, the knowledge on the optimal cost-to-go term required at the end of an MPC optimization cycle yields the optimal control. In Tahirovic and Magnani [2011], the RbNF is used as an estimate of the optimal value of the cost-to-go term (roughness-to-go), significantly improving an MPC navigation planning paradigm in rough terrains. For the purpose of clarity, the RbNF algorithm is recalled here.

2.1 Traversability measure of a terrain patch

Given elevation and rock maps on a 2D terrain, the traversability measure (the roughness value is equally used within the paper) of a terrain patch in the map grid \( M \) is computed by taking the standard deviation of the terrain elevations associated to each cell over that patch, as proposed in Iagnemma and Dubowsky [2004].

Let us assume that the terrain map \( M \) is presented as a rectangle grid that consists of the regular squared patches of dimension \( L' \times L' \), where \( L' \) is taken such that the patch can approximately cover a vehicle-size field. Formally, a patch \( p_{ij} \) is defined as

\[
 p_{ij} = \{ (x,y) \in M | (x_i - L'/2 \leq x \leq x_i + L'/2), \quad (y_j - L'/2 \leq y \leq y_j + L'/2) \},
\]

where \( i, j = -N, ... , N; (2N + 1)^2 \) being the number of patches, and \( p_{00} \) the patch around the goal position \((x_{go}, y_{go}) = (x_{goal}, y_{goal})\). The roughness of a terrain patch can be defined in different ways Guo et al. [2003], Howard and Seraji [2001], Iagnemma and Dubowsky [2004], Seraji [2000], Singh et al. [2000], Ye and Borenstein [2004]. In this work, the roughness of a patch will be a function of the terrain parameters such as the elevation and the friction coefficient.

It is assumed that the terrain map is obtained in the form of elevation map in \( z : X \to R \) associated with each cell over the patch. In this work, the standard deviation of the terrain elevations over a patch is used as proposed in Iagnemma and Dubowsky [2004] with a slight modification to include the friction coefficient. If \( R_{ij} \) is the set of terrain elevation points inside the patch \( p_{ij} \), then the traversability hardness of this patch can be defined as

\[
 \hat{\gamma}(p_{ij}) = \alpha(p_{ij}) \sqrt{\text{var}(z(R_{ij}))},
\]

where \( d \) is the vehicle wheel diameter scaling the selected roughness measure to vehicle size and \( \text{var} \) stands for the variance function. The scaling is used to make a distinction between the difficulties to traverse a patch for the vehicles with different wheel diameters, i.e., the larger diameter, the higher mobility.

\[
 \alpha(p_{ij}) : X \to \{1,M\} \text{ satisfies } \alpha \to 1 \text{ when the friction coefficient is large enough making no additional problem to the terrain traversability, and } \alpha \to M \text{ when the friction coefficient has a small value imposing additional traversability hardness over the patch.} \]

2.2 Roughness-to-go Algorithm

The main purpose of the roughness-to-go algorithm is to compute a numerical function RbNF: \( X \to [0, \infty) \), which provides estimated roughness-to-go values for each patch of the given terrain. Such function can be used either as a navigation function or as a cost-to-go term within the MPC optimization guiding the vehicle toward the more traversable areas while approaching the goal position. \( X \) space is restricted to the vehicle position, \( X = \{ \forall (x_{cg}, y_{cg}) | (x_{cg}, y_{cg}) \in M \} \), and \( r = (x_{cg}, y_{cg}) \) being the vehicle coordinates in the terrain map \( M \).

Fig. 1 shows the flow of the main function that calls PATCH-ROUGHNESS() and EXPAND-SQUARE() functions. PATCH-ROUGHNESS(obst, Z, \( \Lambda \)) computes the roughness value for each terrain patch \( p_{ij} \), \( i, j = -N, ... , N \) and assigns these values to the function \( \Gamma \). Matrix obst contains the information on the obstacles, where \( \text{obst}(i,j) = 1 \) if the patch \( p_{ij} \) contains an obstacle and \( \text{obst}(i,j) = 0 \) if the patch \( p_{ij} \) is obstacle free. Matrix \( Z \) represents the terrain elevations of each cell over the patches in the map \( M \), while matrix \( \Lambda \) contains information on the corresponding friction coefficients.

Let \( \text{SQ}(k) \) be the \( k^{th} \) square matrix of dimensionality \((2k + 1) \times (2k + 1)\), \( k = 0, ... , N \), centered at the goal position (see Fig. 4-left for \( \text{SQ}(0), \text{SQ}(1) \) and \( \text{SQ}(2) \)), whose elements \( \text{sq}_{mn}^k \) carry the currently computed roughness-to-go values \( \text{sq}_{mn}^k = \Gamma(p_{mn}) \), where \( m, n = -k, ... , k \). EXPAND-SQUARE(obst, SQ, \( \Gamma(\text{SQ}) \)) presented in Fig. 3 is a function that expands \( \text{SQ}(k) \) into \( \text{SQ}(k+1) \) by computing roughness-to-go values of the patches at the border of the square \( \text{SQ}(k+1) \) based on the roughness-to-go propagation from the patches at the border of the \( \text{SQ}(k) \). This is done by the function RADIAL-ROUGHNESS-TO-GO(), where the roughness-to-go value of a new patch \( \text{sq}_{mn}^{k+1} \) located at the \( \text{SQ}(k+1) \) border is given as a sum of its roughness \( \Gamma(p_{mn}) \) and the minimum propagated roughness-to-go value from one of its connecting patches located at the \( \text{SQ}(k) \) border. Fig. 4-left shows an example of the patches located at the \( \text{SQ}(2) \) border noted as 1-4.
function MAIN
  \( \Gamma \) ← PATCH-ROUGHNESS(obst, Z, \( \Lambda \))
  SQ ← \( \Gamma(p_{0,0}) \)
  for \( k=1 \)
    \( \triangleright k \) is the indicator of the current square
    SQ ← EXPAND-SQUARE(SQ, \( \Gamma \))
  end for
  RbNF ← SQ
  return RbNF
end function

Fig. 1. Main Algorithm RbNF

function Patches-ROUGHNESS(obst, Z, \( \Lambda \))
  for \( i=-N:N \)
    for \( j=-N:N \)
      \( \gamma_{i,j} \) ← \( \alpha_{i,j} \sqrt{\text{var}(z(R_{i,j}))} \)
      \( \Gamma \) ← \( \gamma_{i,j} \)
    end for
  end for
  return \( \Gamma \)
end function

Fig. 2. Procedure to obtain the traversability index (roughness) of each patch

function EXPAND-SQUARE(obs, SQ, \( \Gamma(SQ) \))
  SQ ← RADIAL-ROUGHNESS-TO-GO()
  SQ ← BORDER-ROUGHNESS-TO-GO()
  SQ ← BACKWARD-ROUGHNESS-TO-GO()
  return SQ
end function

Fig. 3. Procedure to propagate roughness-to-go values

The presented propagation of the roughness-to-go value is the result of the assumption that the vehicle moves by decreasing the square index \( k \) at each step, from the border of the square \( \text{SQ}(k+1) \) to the border of the square \( \text{SQ}(k) \). This means that the vehicle monotonically approaches the goal position in the sense of the Chebyshev distance. The Chebyshev distance can be easily explained using a chess board and it represents the minimum number of moves a king requires to move between two positions. Formally, the Chebyshev metric is the infinity norm, \( L_{\infty} \), and the distance between two terrain patches \( p_{ij} \) and \( p_{mn} \) is given by

\[
d_{\text{chebyshev}}(p_{ij}, p_{mn}) := \max(|i-m|, |j-n|). \tag{2}
\]

The restriction that considers only the patches located at the border of the \( \text{SQ}(k) \) from which the roughness-to-go value can be propagated to the patches of the \( \text{SQ}(k+1) \) border, instead of using 8-connected neighborhood patches, loses the optimality but significantly simplifies the algorithm. For the path planning purposes, the MPC planning paradigm can compensate for this restriction allowing the vehicle to move in any direction, since the vehicle performs the optimization within the sensor range where the RbNF is used as a cost-to-go approximation.

Functions given in line 3 and 4 of Fig. 3 are used in the presence of obstacles. Fig. 4-right illustrates an example of using BORDER-ROUGHNESS-TO-GO(\( \hat{\Gamma} \)), where the obstacle is depicted with the black color. When the minimum roughness-to-go values of the patches located at the border of the \( \text{SQ}(2) \) square are computed using line 2 of Fig. 3, then the patches depicted with the gray color are those hidden by the obstacle carrying the wrong roughness-to-go values. These patches that contain roughness-to-go values and no obstacles are marked as shadowed and require additional recalculation steps.

Def. A patch \( p_{ij} \) is called a shadowed patch if \( \text{obs}(p_{ij}) = 0 \) and \( \hat{\Gamma}(p_{ij}) \geq M \), where \( M \) is a number used to indicate an obstacle in the function \( \hat{\Gamma} \).

In order to deal with shadowed patches, it is possible to choose a patch from the current square \( \text{SQ}(2) \) and to propagate roughness-to-go values in both clockwise and counterclockwise directions. The roughness-to-go value of a shadowed patch can be recomputed only by propagation from an obstacle-free and not shadowed neighborhood patch which is located at the same square border. This means that the patch assigned as 1 in Fig. 4-right can propagate its roughness-to-go value to its neighbor when the algorithm moves in clockwise direction. Here, with domino effect, all roughness-to-go values of shadowed patches can be recomputed using this direction. This also worth for the patch assigned with 2 in Fig. 4-right if the propagation is performed in counterclockwise direction. At the end, it is possible to take a minimum roughness-to-go value obtained from both directions for each shadowed patch.

Fig. 5-left shows the case when the BORDER-ROUGHNESS-TO-GO(\( \Gamma \)) function is not sufficient to recompute the roughness-to-go values for each shadowed patch. After the \( \text{SQ}(3) \) square is expanded and the function BORDER-ROUGHNESS-TO-GO() is executed, as it is shown in Fig. 5-right, all patches marked with an interrogative sign will still remain shadowed. In such case, the algorithm records all shadowed patches and the function BACKWARD-ROUGHNESS-TO-GO(), called at line 4 of Fig. 3, tends to recompute their roughness-to-go values backward, starting from the last recorded shadowed patch. In this case, the roughness-to-go value of a shadowed patch is recomputed as a minimal propagated roughness-to-go value from each of its neighborhood 8-connected patches. A shadowed patch becomes a non-shadowed one if at least one non-shadowed patch in its 8-connected neighborhood exists from which the roughness-to-go value can be propagated.

A rough terrain example and the shape of corresponding RbNF computed by the proposed algorithm are shown in Fig. 6 and 7.

2.3 Runtime Analysis

The RbNF and the optimal cost-to-go map are compared regarding the execution time needed to obtain them. The compar-
Fig. 5. An example where the BACKWARD-ROUGHNESS-TO-GO() function is needed in order to propagate roughness-to-go values.

Fig. 6. Rough terrain without obstacles.

Fig. 7. Roughness based navigation function, RbNF.

In addition, the machine was out of memory while performing the algorithm to obtain the optimal cost-to-go map for a case of 8km by 8km terrain. Such difficulty is due to the fact that the optimal algorithm requires a connectivity graph unlike the RbNF. If sparse matrices are used for an 8-connectivity graph the memory required by the Dijkstra’s algorithm is 8 times larger than the memory required by RbNF.

3. RRT-RT

Figs. 9 and 10 illustrates a pseudocode for constructing an RRT planner.

Fig. 8. Execution time for an optimal cost-to-go map (blue) and RbNF (red).

The fundamental difference comparing to the state of the art RRTs is comprised in the function NEAREST-NEIGHBOR(x, T). Namely, a classical RRT algorithm uses a metric based on Euclidian distance to find the nearest vertex of the tree T to a new random state x_{random}, while the algorithm proposed in this paper uses a measure based on the terrain roughness. The proposed metric is given by

\[ d := g(x) + h(x), \]  

where the term \( g(x) \) represents the roughness traversed from the initial position \( x_{init} \) to the state \( x_{new} \), while the term \( h(x) \) represents a heuristic which estimates cost-to-go value from \( x_{new} \) to the goal position. The heuristic \( h(x) \) is based on the RbNF function, where \( h(x) = RbNF(x) \).

In addition, only those vertices that have roughness-to-go values larger than the roughness-to-go value at \( x_{rand} \), estimated by the RbNF, are considered for potential tree extension. Such selective policy guarantee a decrease of the RbNF providing an efficient way of exploration of the terrain locations to extend the tree toward less rough terrain regions.

For the most general case, one must use the vehicle model to extend the tree inside the EXTENSION() function to be sure the vehicle will be able to follow the suggested routes. The EXTENSION-FIXED() function extends the tree for a fixed time toward \( x_{rand} \), generating a new tree vertex \( x_{new} \).
If this newly extended edge is feasible, which is verified by the function NEW-STATE(), then $x_{new}$ is a new tree state. Each new state has a recorded preceding state as well as corresponding control action, $u$, that guides the vehicle toward the state $x_{new}$ from its preceding tree state.

4. SIMULATION

In the presented simulations the pointwise robot was used in order to compare the RRT-RT with the optimal planner obtained by the Dijkstra’s algorithm. The cost function used to find an optimal path by the Dijkstra’s algorithm is a line integral of the function given in (1) along the traversed path.

In the first experiment, the paths generated by the RRT-RT algorithm are shown to illustrate its potential for navigation in rough terrains. Fig.11 shows small deviations between the paths obtained by the RRT-RT (red color) and by the Dijkstra (black color) algorithms for some different vehicle initial positions, where the goal position is fixed in the middle of the terrain, at (50m,50m).

Fig.12 illustrate the second experiment, where the explored terrain locations by the RRT-RT planner are depicted for different initial positions. If these figures are compared to the terrain presented in Fig. 11, one can see that the RRT explorations are performed within less rough terrain regions toward the goal position.

In the third experiment presented in Fig.13, several different runs of the RRT-RT are illustrated using a fixed initial and goal position, given at (10m, 10m), (50m,50m), respectively. Despite the fact that the obtained solutions are different in all given runs, it can be seen that the generated paths avoid high rough terrain regions (see also Fig.11).

In the forth experiment, twenty randomly vehicle initial positions and terrain roughnesses are selected in order to compare the total costs obtained by the RRT-RT and Dijkstra’s algorithms. The comparison is given in Fig. 14, where the total roughness traversed by the vehicle along these twenty RRT’s paths is compared to that obtained along the optimal ones. The value of this coefficient, $r = 0.8082$, shows the ability of the RRT-RT planner to follow the optimal paths. Since finding an optimal path within a large-scale rough terrain using Dijkstra’s algorithm can be computationally too expensive, the obtained suboptimality of the RRT-RT algorithm makes it a preferred tool to be used in such environments.

5. CONCLUSION

The information on the roughness-to-go values provided by the RbNF is used to guide an RRT planner for mobile vehicle navigation. For this purpose, the selection of the nearest neighbor within the tree from a new randomly generated state was based on the estimated roughness to be traversed. Such an RRT policy extends the exploring tree toward less rough terrain regions in order to minimize the total roughness along the final path as much as possible. Simulation results have shown that the paths generated by the RRT-RT did not significantly deviate from the optimal ones.

REFERENCES


---

**Fig. 13.** The RRT-RT different runs for the fixed vehicle initial position for the terrain presented in Fig. 11

**Fig. 14.** The comparison between the roughness costs of the RRT-RT and Dijkstra planners for randomly generated initial positions and terrains

