This is the Pre-Published Version.

Rapid Replanning of Energy-Efficient Paths for Navigation on Uneven Terrains

Nuwan Ganganath†, Chi-Tsun Cheng, and Chi K. Tse
Department of Electronic and Information Engineering
The Hong Kong Polytechnic University
Hung Hom, Kowloon, Hong Kong
†Email: nuwan.marasinghearachchige@connect.polyu.hk

***Abstract*—Mobile robots are often utilized in remote and hostile outdoor environments with uncertainties and unknown dangerous. The energy-efficient paths generated based on prior information can be impracticable due to the changes in the environment. Recently proposed Z* search algorithm is capable of finding physically feasible energy-efficient paths on uneven terrains. It can achieve the same accuracy as any brute force algorithm, but with a low computational complexity. However, neither Z* nor any other energy-efficient path planners can effectively handle path replanning triggered by environment changes such as emergence of obstacles. In order to fill this void, we propose a novel algorithm which can recompute optimal paths efficiently. Simulation results show that the proposed algorithm can find equally energy-efficient paths as Z* does, but at a considerably lower computational cost. Therefore, the proposed algorithm can be very useful in mobile robot navigation on uneven terrains with unknown obstacles.**

***Index Terms*—Mobile robot, path planning, replanning, energy-efficient, uneven terrain.**

I. Introduction

Path planning algorithms have been highly popular in indoor mobile robot applications [1]. In such applications, terrains are assumed to be flat. Recently, mobile robots have been widely used in outdoor applications where they have to often deal with uneven terrains [2]–[4]. In contrast to indoor mobile robot applications, shortest paths on uneven terrains can be physically impracticable due to motion power constraints of the robots and their instability on steep terrains. Moreover, shortest paths can be highly energy inefficient on such terrains. Mobile robots are usually battery powered and utilized in remote and hostile environments. Therefore, energy-efficient path planning is crucial in prolonging their operation durations.

A. Related Work

There were few researches on energy-efficient path planning on uneven terrains in the previous decades. One of the initial attempts on studying characteristics of optimal paths was performed by Rowe and Ross [5]. They proposed an energy-cost model for mobile robots navigating on uneven terrains. The energy-cost of a path between two points on a terrain surface is defined to be the energy loss due to friction and gravity. They added anisotropism to this model by considering impermissible traversal headings due to power limitations and overturn danger. The cost-optimal paths are constructed by appropriately picking path segments from path subspaces. Later, Rowe and Kanayama [6] applied the same energy-cost model to determine near-optimal paths on a surface of a vertical-axis ideal cone. Lanthier et al. [7] assumed a terrain to be composed of triangular faces. The cost of traveling on each face is captured via face weights. Based on the terrain face weight concept, Sun and Reif [8] proposed an approximation algorithm to find energy-efficient paths on terrains. They also computed some upper and lower bounds on the combinatorial size of optimal paths.

Even though the terrain face weight concept helps to reduce the computational complexity of path planning methods, it does not consider the change of potential energy as elevation changes. In order to avoid such drawbacks, Choi et al. [9] proposed an A*-like heuristic search algorithm to find energy-efficient paths on uneven terrains using grid-based elevation maps. However, their path planner fails to find feasible paths on steep terrains due to a deficient heuristic function used. Recently, Ganganath et al. [10] proposed a novel heuristic energy-cost function based on zigzag-like motion patterns on steep terrains. Using this heuristic function, they proposed a heuristic search algorithm (Basic Z*) to find physically feasible energy-efficient paths on uneven terrains. Based on the admissible and consistent nature of the heuristic function, an improved heuristic search algorithm (Z*) was also proposed [11], which can compute equally energy-efficient paths as Basic Z* does while being computationally more efficient.

B. Motivation and Contributions of the Paper

All the algorithms explained above assume complete knowledge of robots' environment in order to find energy-efficient paths. These algorithms generate energy-efficient paths to a given goal location from the current location of the robot based on prior information of the terrain surface and obstacles. Even though high resolution maps are commonly available for many geographical locations, most of the environments are highly dynamic in nature. Therefore, optimal paths generated by traditional path planning algorithms might not be practically realizable with mobile robots. In case of a sudden appearance of obstacles, mobile robots have to recompute their paths from their current location to the goal location. Such replanning can be highly computationally expensive, resulting unacceptable delays in navigation. Despite the vast range of applications in both commercial and military robotics, attempts on replanning

of energy-efficient paths for navigation on uneven terrains have not been reported in literature. Therefore, we believe that it needs to be investigated promptly.

In this paper, we consider the problem of replanning energy-efficient paths for mobile robot navigation on uneven terrains. Uneven terrains are represented by using gird-based elevation maps. Inspired by the Z* search algorithm, we propose a novel algorithm, namely Dynamic Z*, which can perform fast planning and replanning of physically feasible energy-efficient paths for navigation on uneven terrains. Similar to Z* search algorithm, it uses prior information for the initial planning of routes. Z* algorithm has to recompute paths from scratch if obstacles suddenly appear on the initially planned paths. In contrast, Dynamic Z* efficiently recompute optimal paths from the current robot location to its goal location by using its previous search results and updating them locally.

The rest of the paper is organized as follows. Section II explains the problem formulations of path planning and replanning on weighted graphs transformed from elevation maps of uneven terrains. It also briefly discusses the energy-cost model used in this paper. The proposed Dynamic Z* algorithm is explained in details in Section III, including a brief review on heuristic search algorithms for energy-efficient path planning. Results of Dynamic Z* algorithm are presented and performances of the proposed path planner are analyzed in Section IV. Concluding remarks are given in Section V.

II. Problem Formulation

Here, the robot is represented as a point on the terrain surface and assumed to be rigid and holonomic. In order to facilitate the path planning and replanning tasks, we transform a grid-based elevation map of a terrain surface into a weighted king's graph $\mathcal{G}$ which consists of 8-connected neighborhoods. Let n be a node in $\mathcal{G}$. Terrain surface coordinates corresponding to n can be represented as $(n.x, n.y, n.z)$. Let n_{s} and n_{r} be the nodes corresponding to the initial location and the current location of the robot, respectively. Note that $n_{\mathrm{s}} \equiv n_{\mathrm{r}}$ initially. Here, the path planning task is to find physically feasible energy-efficient paths from n_{s} to a given goal location n_{g} and the path replanning task is to recompute such paths from n_{r} to n_{g}. Hence, we need to define the edge costs of $\mathcal{G}$ in terms of energy-costs.

A. Energy-Cost Model

In order to calculate the edge-costs of $\mathcal{G}$, the energy-cost model proposed by Rowe and Ross [5] is adopted. Their model comprises all previously published criteria for traversing across uneven terrain with anisotropic friction and gravity effects. Due to its versatility, Rowe and Ross energy-cost model has been adopted by many others for path planning on terrains [6]–[8]. This energy-cost model assumes that a robot travels at a constant velocity v and consumes negligible energy for making turns.

Let n_{c} be the current node of search in $\mathcal{G}$ and n_{n} be a neighboring node. The energy-cost of traversing $n_{\mathrm{c}}n_{\mathrm{n}}$ is defined as the energy loss due to work against external forces along $n_{\mathrm{c}}n_{\mathrm{n}}$. According to Rowe and Ross's model [5], the resultant of two major external forces applying on the robot, gravity and friction, can be given as $mg(\mu\cos\phi(n_{\mathrm{c}}, n_{\mathrm{n}}) + \sin\phi(n_{\mathrm{c}}, n_{\mathrm{n}}))$. Here, m is the mass of the robot, μ is the friction coefficient, g is the gravitational field strength, and $\phi(n_{\mathrm{c}}, n_{\mathrm{n}})$ is the inclination angle between n_{c} and n_{n}.

Due to the motion power limitations in uphilling on steep terrains, the maximum slope that the robot can overcome is defined as $\phi_{\mathrm{f}} = \sin^{-1}(\frac{P_{\max}}{mgv\sqrt{\mu^2+1}}) - \tan^{-1}(\mu)$, where $P_{\max}$ is the maximum motion power of the robot. Also, anisotropic traction-loss phenomena will occur if the slope is greater than $\phi_{\mathrm{s}} = \tan^{-1}(\mu_{\mathrm{s}} - \mu)$, where μ_{s} is the static friction coefficient. Thus, the critical impermissible angle for uphilling is defined as the minimum of ϕ_{f} and ϕ_{s}. For downhilling, the robot has to spend negligible energy to travel at a constant speed if the slope angle is less than critical breaking angle $\phi_{\mathrm{b}} = -\tan^{-1}(\mu)$. Thus, the energy-cost for traversing $n_{\mathrm{c}}n_{\mathrm{n}}$ can be summarized as

$$k(n_{\mathrm{c}}, n_{\mathrm{n}}) = \begin{cases} \infty, & \text{if } \phi(n_{\mathrm{c}}, n_{\mathrm{n}}) > \phi_{\mathrm{m}} \\ mgs(n_{\mathrm{c}}, n_{\mathrm{n}})(\mu\cos\phi(n_{\mathrm{c}}, n_{\mathrm{n}}) + \sin\phi(n_{\mathrm{c}}, n_{\mathrm{n}})), & \\ & \text{if } \phi_{\mathrm{m}} \geq \phi(n_{\mathrm{c}}, n_{\mathrm{n}}) > \phi_{\mathrm{b}} \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

Here, $s(n_{\mathrm{c}}, n_{\mathrm{n}})$ is the Euclidean distance between n_{c} and n_{n} in a 3D space. Please refer to [5], [10] for further details on the energy-cost model.

III. The Proposed Algorithm

In this section, procedures of Basic Z* and Z* heuristic search algorithms are revisited. Afterward, Dynamic Z* search algorithm will be introduced for rapid replanning of energy-efficient paths on uneven terrains.

A. Background

One of the most popular solutions for path planning on weight graphs was proposed by Hart et al. [12]. They showed that A*-like heuristic search algorithms can find exactly the same solution as brute force algorithms at a considerably lower computational cost if the heuristics are selected appropriately. Inspired by the A* search algorithm, Basic Z* [10] optimizes the expected energy-cost of traversing to n_{g} through n_{c}, which can be defined as $f(n_{\mathrm{c}}) = g(n_{\mathrm{s}}, n_{\mathrm{c}}) + h(n_{\mathrm{c}}, n_{\mathrm{g}})$. The energy-cost of traversing from n_{s} to n_{c} is defined as $g(n_{\mathrm{s}}, n_{\mathrm{c}})$. It can be calculated by using (1) for each intermediate step between n_{s} and n_{c}. The heuristic energy-cost estimate of $n_{\mathrm{c}}n_{\mathrm{g}}$ traversal is defined as [10]

$$h(n_{\mathrm{c}}, n_{\mathrm{g}}) = \begin{cases} \frac{mg\Delta(n_{\mathrm{c}}, n_{\mathrm{g}})}{\sin\phi_{\mathrm{m}}}(\mu\cos\phi_{\mathrm{m}} + \sin\phi_{\mathrm{m}}), & \\ & \text{if } \phi(n_{\mathrm{c}}, n_{\mathrm{g}}) > \phi_{\mathrm{m}} \\ mgs(n_{\mathrm{c}}, n_{\mathrm{g}})(\mu\cos\phi(n_{\mathrm{c}}, n_{\mathrm{g}}) + \sin\phi(n_{\mathrm{c}}, n_{\mathrm{g}})), & \\ & \text{if } \phi_{\mathrm{m}} \geq \phi(n_{\mathrm{c}}, n_{\mathrm{g}}) > \phi_{\mathrm{b}} \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

Here, $\Delta(n_{\mathrm{c}}, n_{\mathrm{g}})$ is the elevation of n_{g} in reference to the elevation of n_{c}.

Basic Z* uses best-first search to find the energy-efficient paths. It starts by calculating the energy-cost of n_s using (2), where $f(n_s) = h(n_s)$. A node is added to an OPEN set once it is visited. In each iteration, a node with minimum expected energy-cost n_c, is taken out from the OPEN set and all its neighbors are visited. Then, all those visited neighbors are added to OPEN set. Yet any visited nodes can be revisited and their cost may be improved. If the revisiting nodes have already been removed from the OPEN set, they will be added again to the OPEN set and their energy-cost and parents will be updated. If they are still in the OPEN set, just their energy-cost and parents will be updated. When $n_c = n_g$, the algorithm has reached the goal and the iterative procedure will be terminated. The energy-efficient path can be obtained by traversing back from n_g to n_s by tracing their parent connections.

Even though Basic Z* search algorithm is capable of finding an optimal solution, its computational efficiency may be degraded due to node revisits. Therefore, Z* heuristic search algorithm [11] is proposed to improve the computational efficiency of Basic Z*. It uses another set called CLOSED set in addition to the OPEN set used in Basic Z* algorithm. Similar to Basic Z* algorithm, in each iteration, a node with the minimum energy-cost, n_c is taken out from the OPEN set and all neighbors are added to OPEN set. Here, n_c is added to the CLOSED set. The nodes in the CLOSED set will not be revisited.

B. Dynamic Z Search Algorithm*

While navigating on terrains, the robot is capable of observing the traversability of neighboring nodes. Initially, it plans its route from the start node to the goal node assuming that nodes with unknown traversability status are traversable. Once the energy-efficient path is obtained, the robot follows the path until the goal is reached. However, if the path is obstructed by a previously unknown obstacle, it has to recompute its path from its current location to the goal. Since Z* algorithm (or any other energy-efficient path planner) does not have effective mechanisms to recompute its paths in such a scenario, it has to perform a search from scratch. Therefore, we introduce Dynamic Z* algorithm which is capable of planning and rapid replanning of energy-efficient routes. The complete routine of the proposed Dynamic Z* search algorithm is given in Algorithm 1.

Similar to Basic Z* and Z* algorithms, Dynamic Z* is also based on the best-first search. Nevertheless, the search direction of Dynamic Z* is the reverse of its counterparts, i.e. Dynamic Z* always plans its paths from goal node n_g to current robot location n_r. In contrast to path distance which remains the same despite of the travel direction, the energy-cost of the robot is obviously depends on its travel direction. Therefore, the energy-cost should be calculated in the same direction as Z* does, despite the search is performed in reverse direction. The same condition applies to the heuristic cost estimation since the heuristics used here are not backward consistent [13].

Algorithm 1: Pseudocode of Dynamic Z* search algorithm

```
1:  function INITIALIZE()
2:      CLOSED ← ∅
3:      OPEN ← {n_g}
4:      g[n_g] ← 0
5:      f[n_g] ← h(n_r, n_g)
6:      previous[n_g] ← n_g
7:  end function
8:  function COMPUTE_PATH()
9:      while OPEN ≠ ∅ do
10:         n_c ← argmin_{n∈OPEN} f[n]
11:         if f[n_c] == ∞ then
12:             return failure
13:         else if n_c == n_r then
14:             break
15:         end if
16:         OPEN ← OPEN \ {n_c}
17:         CLOSED ← CLOSED ∪ {n_c}
18:         for ∀n_n ∈ {neighbor[n_c]} \ CLOSED do
19:             g_temp ← g[n_c] + k(n_n, n_c)
20:             f_temp ← g_temp + h(n_r, n_n)
21:             if n_n ∉ OPEN or f_temp < f[n_n] then
22:                 previous[n_n] ← n_c
23:                 g[n_n] ← g_temp
24:                 f[n_n] ← f_temp
25:                 if n_n ∉ OPEN then
26:                     OPEN ← OPEN ∪ {n_n}
27:                 end if
28:             end if
29:         end for
30:     end while
31: end function
32: function REFRESH_SETS()
33:     for ∀n ∈ CLOSED do
34:         if f[n] ≥ f[previous[n_r]] then
35:             CLOSED ← CLOSED \ {n}
36:         end if
37:     end for
38:     TEMP ← ∅
39:     for ∀n ∈ CLOSED do
40:         if {neighbor[n]} ∩ CLOSED ≠ {neighbor[n]} then
41:             TEMP ← TEMP ∪ {n}
42:         end if
43:     end for
44:     CLOSED ← CLOSED \ TEMP
45:     if TEMP ≠ ∅ then
46:         OPEN ← TEMP
47:     else
48:         OPEN ← {n_g}
49:     end if
50:     for ∀n ∈ OPEN do
51:         f[n] ← g[n] + h(n_r, n)
52:     end for
53: end function
54: function MAIN()
55:     INITIALIZE()
56:     COMPUTE_PATH()
57:     while n_r ≠ n_g do
58:         observe neighboring nodes and update map
59:         if previous[n_r] is unoccupied then
60:             n_r ← previous[n_r]
61:             move robot to n_r
62:         else
63:             REFRESH_SETS()
64:             COMPUTE_PATH()
65:         end if
66:     end while
67: end function
```

Dynamic Z* algorithm starts from the function MAIN() in Algorithm 1 {54}. (Numbers in curly braces refer to line numbers in the pseudocode.) It first initializes the search problem {55}. In the function INITIALIZE(), the CLOSED set is initialized to an empty set {2}. The OPEN set is initialized to the goal node where Dynamic Z* begins its search {3}. Since $k(n_{\rm g}, n_{\rm g}) = 0$, thus $g(n_{\rm g}, n_{\rm g}) = 0$ and $f(n_{\rm g}) = h(n_{\rm r}, n_{\rm g})$. Those values are stored as properties of $n_{\rm g}$ {4-5}. Also, since $n_{\rm g}$ is the first node in the search, it does not have a parent other than itself {6}. After initialization, Dynamic Z* can compute the initial path based on the available information {56}.

The purpose of the function COMPUTE_PATH() is to calculate an energy-efficient path from $n_{\rm g}$ to $n_{\rm r}$. Even though the search starts from $n_{\rm g}$ and continues until it reaches $n_{\rm r}$ {13-15} or the OPEN set is empty {9}, one should note that the energy-cost of the path is calculated in the opposite direction, i.e. always directing towards $n_{\rm g}$ in the tree structure. The function COMPUTE_PATH() starts by selecting a minimum cost node $n_{\rm c}$ from the OPEN set {10}. If $f(n_{\rm c}) = \infty$, it indicates that no feasible paths exists, thus, the algorithm terminates {11-12}. Otherwise, it removes $n_{\rm c}$ from OPEN set {16} and adds it to CLOSED set {17}. Then, all the neighbors of $n_{\rm c}$ which are not already in CLOSED set, are considered for cost update {18-24}. The neighboring nodes which are not in the OPEN set are added to the OPEN set {25-27}. Once the search process reaches $n_{\rm r}$, it returns back to the function MAIN().

After the initial route is planned, robot starts following the path until the goal is reached {57}. Every time that the robot moves to a new node, it observes its neighboring nodes and update the map accordingly {58}. If its parent node is unoccupied, it moves to the parent node and update its location information {59-61}. Otherwise, it needs to recompute another path {62-64}. Since the nodes in the CLOSED set create a search tree spanning from $n_{\rm g}$, we can reuse this tree structure even if one of its branches fails. The challenge is to identify a cut-off level for the tree so that rest of the tree can be linked again with $n_{\rm r}$. That is the purpose of the function REFRESH_SETS(). It first removes all the nodes from the CLOSED set, whose cost is not less than the previous cost of reaching the occupied node {34-36}. Then, it checks the remaining nodes in the CLOSED set to identify the new leaf nodes on the search tree {38-43}. These leaf nodes are then removed from the CLOSED set {44} and defined as a new OPEN set {46}. If such leaf nodes are not available, the function defines a new OPEN set with the sole node $n_{\rm g}$ {48}. Once the OPEN set is properly redefined, Dynamic Z* updates the heuristic energy-cost of all the nodes in the OPEN set {50-52}. Since heuristics change as the robot moves, they need to be calculated with respect to the current location of the robot.

The main advantage of this procedure is that it can resume its search process from the most appropriate place. Therefore, it is not necessary to go through the whole set of nodes repeatedly to find an optimal path. Furthermore, it only needs to update the heuristics of the nodes that are in the OPEN set. Since the nodes in the CLOSED set are locally consistent, their parent connections do not change.

IV. Simulations

In this section, we evaluate the proposed Dynamic Z* algorithm against Z* algorithm for energy-efficient path planning and replanning on uneven terrains. Z* algorithm is selected because it guarantees to provide a physically feasible energy-optimal path between two given points by visiting a minimum possible number of nodes, if such a path exists. Simulations were conducted in MATLAB using two terrain models.

A. Terrain Models

The terrain models used in this paper are 3D landscapes imitating uneven terrains which can be expressed using the following formulas:

$$\text{Model 1: } z(x,y) = 4.81\Bigg[1.5\cos\left(\frac{x}{4\pi}\right) + 0.5\sin\left(\frac{y}{4\pi}\right) - 0.5\sin\left(2.5\sqrt{\left(\frac{x}{4\pi}\right)^2 + \left(\frac{y}{4\pi}\right)^2}\right)\Bigg]^2, \quad (3)$$

$$\text{Model 2: } z(x,y) = 3.79\Bigg[\sin\left(\frac{y}{3\pi} + 0.5\right) - 2\sin\left(\frac{y}{3\pi}\right) + 1.3\cos\left(\frac{x}{3\pi}\right) - 0.3\sin\left(3\sqrt{\left(\frac{x}{2\pi}\right)^2 + \left(\frac{y}{2\pi}\right)^2}\right)\Bigg]^2. \quad (4)$$

The elevation of a grid centered at (x, y) is given by $z(x, y)$, i.e. $n.z = z(n.x, n.y)$. Illustrations of Model 1 and Model 2 are shown in Figs. 2 (a) and (b), respectively. Similar terrain representations have been used previously in [10], [14], [15].

B. Simulation Setup

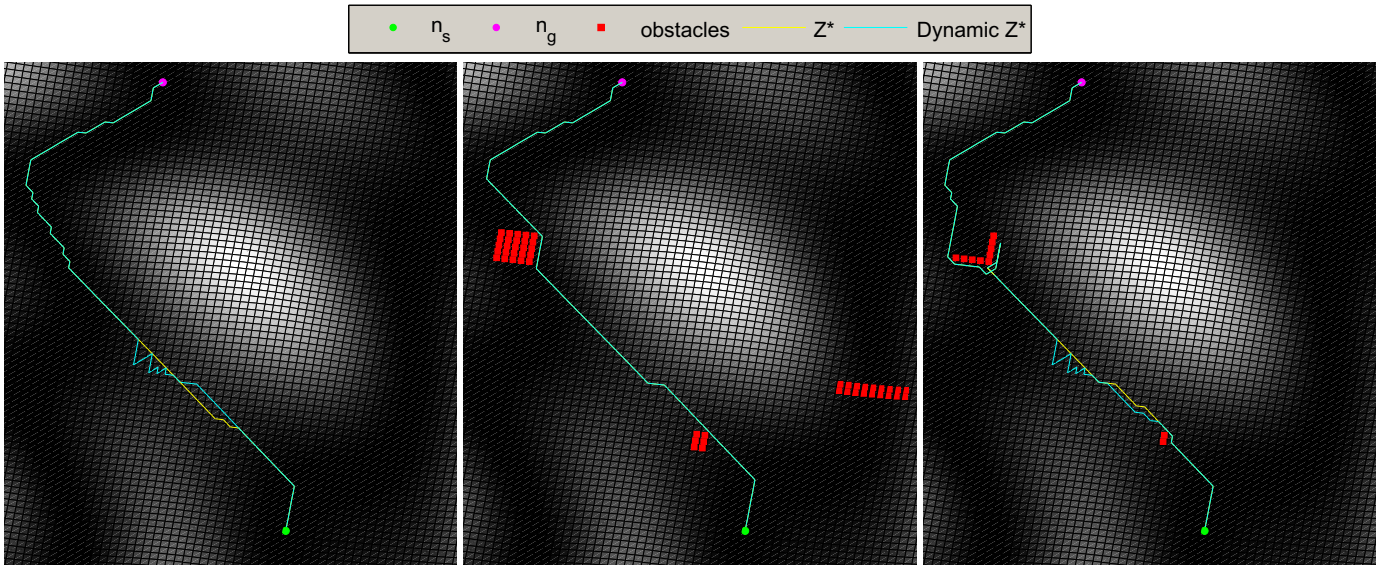
We conducted 3 sets of simulations (I-III) using Z* and Dynamic Z* algorithms. In all the simulations, the following parameters remained unchanged: $m = 22$ kg, $v = 0.35$ ms^{-1}, $P_{\max} = 72$ W, $\mu = 0.01$, $\mu_s = 1.0$, and 9.81 ms^{-2}.

Simulation I was conducted on a part of Model 1 with user defined obstacles. Starting point and goal location of the robot were set to $n_{\rm s} \equiv (52, 18)$ m and $n_{\rm g} \equiv (27, 85)$ m, respectively. Three different simulation setups were arranged to evaluate the performances of the algorithms under test, (a) with no obstacles, (b) with known obstacles, and (c) with unknown obstacles on the terrain surface. Simulation parameters and results are given in Table I and paths generated are shown in Fig. 1. Since the robot had complete information about all the obstacles in setup (b), all the obstacles are illustrated in Fig. 2 (b). Fig. 2 (c) illustrates only the obstacles detected by the robot during navigation.

Simulations II and III were carried out on Models 1 and 2 with randomly generated obstacles with a uniform distribution. The bases of the terrain models considered were 100×100 m^2 squares and the spacial density of obstacles was set to 0.1 for both the simulations. Simulation parameters and results are given in Table II and paths generated are shown in Fig. 2. In these simulations, all the obstacles were unknown at the beginning of the simulations and some of them were detected by the robot during the navigation. Nevertheless, in Fig. 2 both detected and undetected obstacles are illustrated to give a clear idea about the distribution of the obstacles on the terrains.

TABLE I: PARAMETERS AND RESULTS OF SIMULATION I.

Setup	Total energy-cost (J)		Number of nodes visited during navigation		Total number of nodes visited	
	Z*	Dynamic Z*	Z*	Dynamic Z*	Z*	Dynamic Z*
(a) No obstacles	1119.8873	1119.8873	0	0	3706	2001
(b) Known obstacles	1380.0689	1380.0689	0	0	4080	2774
(c) Unknown obstacles	1713.8518	1713.8518	9075	2667	12781	4668



(a) (b) (c)

Fig. 1: Energy-efficient paths generated by Z* and Dynamic Z* algorithms with (a) no obstacles, (b) known obstacles, and (c) unknown obstacles, while rest of the parameters remain same.

C. Results and Performance Analysis

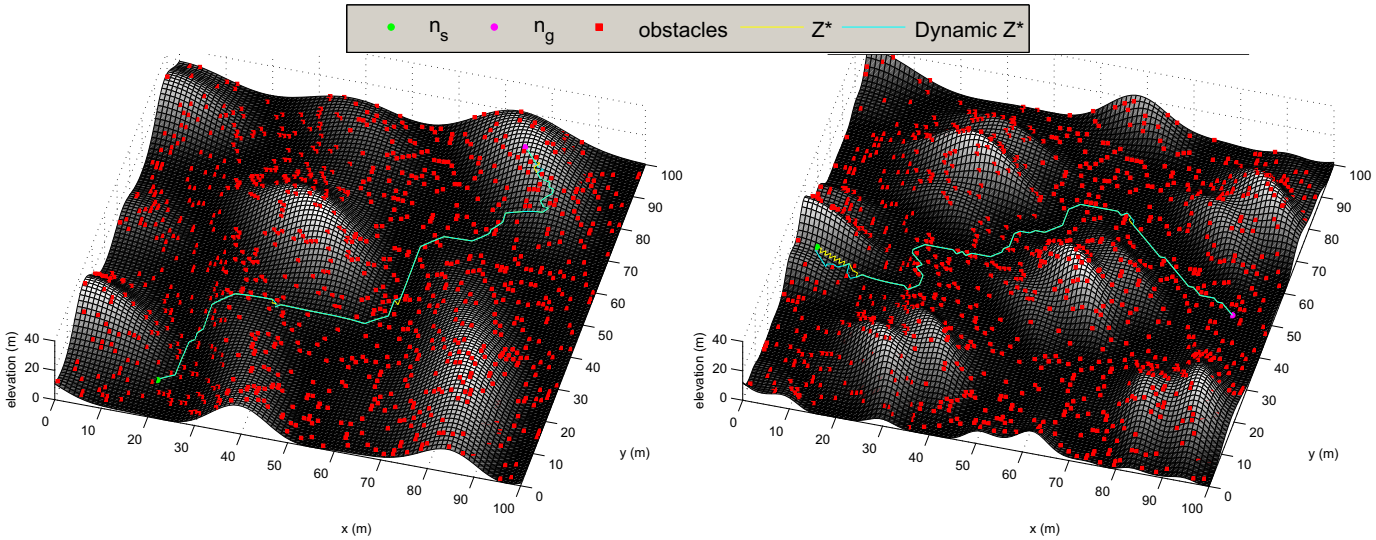
According to the results of Simulation I, Dynamic Z* is capable of finding paths which has equal energy-costs as what Z* produces, with or without having prior knowledge of the environment. In case (a), since there are no obstacles present in the environment, both path planners do not do any replanning. Hence, they do not visit any additional nodes during navigation to find optimal paths because the robot can traverse the initially computed path without encountering any obstruction. Even though the energy-cost of the paths computed by the two algorithms are equal, those paths are slightly different from each other. This is mainly due to the difference in search direction and availability of multiple routes that associate with the same energy-cost. Also in case (b), none of the path planners has visited any nodes to find the optimal paths during the navigation because the prior information about the obstacles was available. In this case, both path planners have generated identical routes as illustrated in Fig. 1 (b).

In case (c), the path planners under test are not aware of the obstacles in the environment during their initial planning. Hence, they assume that all the nodes are traversable and their initial paths generated are equivalent to the paths generated in case (a). However, as the robot discovers obstacles in the environment during its navigation, it has to recompute its path to the goal. Such replanning often results in more energy consuming paths compared to the paths generated with prior information about the obstacles. Nevertheless, Dynamic Z* is capable of providing a equally energy-efficient path as what is obtained by repeatedly applying Z*. The robot is only obstructed by the obstacles which are coincide with its previously planned path. This can be verified by comparing the obstacles shown in Figs. 1 (b) and (c). Interestingly, Dynamic Z* has visited 8113 less number of nodes than Z* to achieve the same results and Dynamic Z* is over 3 times faster than Z* in replanning.

Simulations II and III were conducted in much obstacle dense environments compared to previous cases. Since none of the path planners had prior information about the obstacle distribution in the environment, they had to plan and replan several times to drive the robot to the final goal. Dynamic Z* has found energy-efficient paths considerably faster than Z* does. In Simulation II, Dynamic Z* has visited 13387 less number of nodes compared to Z* for overall planning and Dynamic Z* is over 32 times faster than Z* in replanning. In Simulation III, Dynamic Z* visited has 1086 less number of nodes compared to Z* for overall planning and Dynamic Z* is nearly 5 times faster than Z* in replanning. Similar to the paths generated by Z* algorithm, Dynamic Z* also generates energy-efficient paths which are physically feasible under the motion power constraints of the mobile robot.

TABLE II: SIMULATION PARAMETERS AND RESULTS.

Sim.	Terrain model	n_s (m)	n_g (m)	Total energy-cost (J)		Number of nodes visited during navigation		Total number of nodes visited	
				Z*	Dynamic Z*	Z*	Dynamic Z*	Z*	Dynamic Z*
II	Model 1	(20,10)	(78,88)	6697.1873	6697.1873	13035	405	19652	6265
III	Model 2	(5,43)	(92,51)	302.0689	302.0689	7339	1500	8630	7544



(a) (b)

Fig. 2: Energy-efficient paths generated by Z* and Dynamic Z* algorithms in (a) Simulation II and (b) Simulation III.

V. Conclusion

We present Dynamic Z* algorithm for energy-efficient path planning and rapid replanning on uneven terrains. Traditional path planning methods fail to recompute energy-efficient paths promptly if the robot is unable to traverse originally planned paths due to environmental changes. Dynamic Z* algorithm uses its previous search results in replanning, thus, it is much faster than ordinary energy-efficient path planners. The simulation results show that Dynamic Z* can generate energy-efficient paths which associate with the same energy-cost as the optimal paths obtained by applying Z* repeatedly. Therefore, Dynamic Z* can generate physically feasible energy-efficient paths quickly on terrains with unknown obstacles.

Acknowledgment

This work is supported by the Dept. of Electronic and Information Eng., the Hong Kong Polytechnic University (Project G-UB45) and the Hong Kong PhD Fellowship Scheme.

References

[1] P. Raja and S. Pugazhenthi, "Optimal path planning of mobile robots: A review," *International Journal of Physical Sciences*, vol. 7, no. 9, pp. 1314–1320, 2012.

[2] C. A. Brooks and K. Iagnemma, "Vibration-based terrain classification for planetary exploration rovers," *Robotics, IEEE Transactions on*, vol. 21, no. 6, pp. 1185–1191, 2005.

[3] F. Naderi, D. McCleese, and J. Jordan, "Mars exploration," *IEEE Robotics Automation Magazine*, vol. 13, no. 2, pp. 72–82, June 2006.

[4] C. C. Ward and K. Iagnemma, "A dynamic-model-based wheel slip detector for mobile robots on outdoor terrain," *Robotics, IEEE Transactions on*, vol. 24, no. 4, pp. 821–831, 2008.

[5] N. C. Rowe and R. S. Ross, "Optimal grid-free path planning across arbitrarily contoured terrain with anisotropic friction and gravity effects," *Robotics and Automation, IEEE Transactions on*, vol. 6, no. 5, pp. 540–553, 1990.

[6] N. C. Rowe and Y. Kanayama, "Near-minimum-energy paths on a vertical-axis cone with anisotropic friction and gravity effects," *The International journal of robotics research*, vol. 13, no. 5, pp. 408–433, 1994.

[7] M. Lanthier, A. Maheshwari, and J.-R. Sack, "Shortest anisotropic paths on terrains," in *Automata, Languages and Programming*. Springer, 1999, pp. 524–533.

[8] Z. Sun and J. H. Reif, "On finding energy-minimizing paths on terrains," *Robotics, IEEE Transactions on*, vol. 21, no. 1, pp. 102–114, 2005.

[9] S. Choi, J. Park, E. Lim, and W. Yu, "Global path planning on uneven elevation maps," in *Ubiquitous Robots and Ambient Intelligence (URAI), 2012 International Conference on*. IEEE, 2012, pp. 49–54.

[10] N. Ganganath, C.-T. Cheng, and C. K. Tse, "Finding energy-efficient paths on uneven terrains," in *Mechatronics (MECATRONICS), 2014 10th France-Japan/8th Europe-Asia Congress on*. IEEE, 2014, pp. 383–388.

[11] N. Ganganath, C.-T. Cheng, and C. K. Tse, "A constraint-aware heuristic path planner for finding energy-efficient paths on uneven terrains," *Industrial Informatics, IEEE Transactions on*, 2015.

[12] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *Systems Science and Cybernetics, IEEE Transactions on*, vol. 4, no. 2, pp. 100–107, 1968.

[13] S. Koenig and M. Likhachev, "Fast replanning for navigation in unknown terrain," *Robotics, IEEE Transactions on*, vol. 21, no. 3, pp. 354–363, 2005.

[14] N. Ganganath and C.-T. Cheng, "A 2-dimensional ACO-based path planner for off-line robot path planning," in *Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), 2013 International Conference on*. IEEE, 2013, pp. 302–307.

[15] N. Ganganath, C.-T. Cheng, and C. K. Tse, "An ACO-based off-line path planner for nonholonomic mobile robots," in *Circuits and Systems (ISCAS), 2014 International Symposium on*. IEEE, 2014, pp. 1038–1041.