Energy-Efficient Coverage Path Planning for General Terrain Surfaces

Chenming Wu[1], Chengkai Dai[2], Xiaoxi Gong[3],
Yong-Jin Liu[1], Jun Wang[3], Xianfeng David Gu[4] and Charlie C.L. Wang[5†]

***Abstract*— This paper tackles the problem of energy-efficient coverage path planning for exploring general surfaces by an autonomous vehicle. An efficient algorithms are developed to generate paths on freeform 3D surfaces according to a special design pattern as height-extremity-aware Fermat spiral for this purpose. By using the exact boundary-sourced geodesic distances, the method for generating Fermat spiral paths is first introduced to cover a general surface. Then, heuristics for energy-efficiency are incorporated to add peak points of a height-field as sources for geodesic computation. The paths generated by our method can significantly reduce the cost caused by gravity. Physical experiments have been taken on different terrain surfaces to demonstrate the effectiveness of our approach.**

I. Introduction

Coverage Path Planning (CPP) is a classic problem in robotics that aims at finding a path to cover all points in a specified space Ω. This fundamental problem has a variety of applications such as cleanning, lawn mowing, forest surveillance, underwater inspections and UAVs monitoring. In general, CPP demands a path satisfying three criteria [1]:

1) full coverage – every point has been visited;
2) no repetition – a point is only visited once;
3) quality – application-based requirements such as simplicity of motion or energy consumption.

By partitioning the space Ω and building connectivities between neighboring regions [2], a CPP problem can be approximately mapped to *traveling salesman problem* (TSP), where the requirement of full coverage is converted to control the maximal distance between edges and nodes on a graph. TSP (therefore also CPP) are NP-hard, and the demand of no repetition is tackled in a *weak form* by minimizing the overlap of a path. For quality, we target on generating energy-efficient paths.

[1]C. Wu and Y-J Liu are with the TNList, Department of Computer Science and Technology, Tsinghua University, Beijing, China.

[2]C. Dai is with the Department of Design Engineering, Delft University of Technology, The Netherlands.

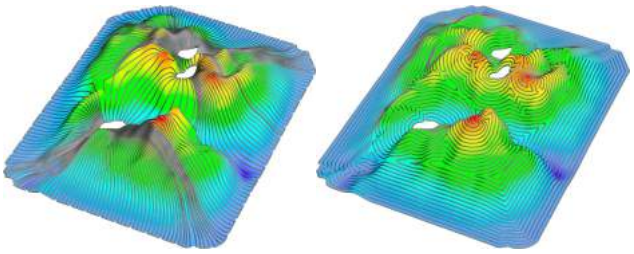
[3]X. Gong and J. Wang are with Nanjing University of Aeronautics and Astronautics, Nanjing, China.

[4]X.D. Gu is with the Department of Computer Science, Stony Brook University, Stony Brook, NY, USA.

[5]C.C.L. Wang is with the Department of Mechanical and Automation Engineering, The Chinese University of Hong Kong; part of this work was completed when he worked at Delft University of Technology.

This work was partially supported by the Natural Science Foundation of China (61432003, 61521002, 61661130156, 61628211, 61725204), the National Key Research and Development Plan of China (2016YFB1001202), the Royal Society-Newton Advanced Fellowship and CUHK Direct Research Grant (4055094).

†Corresponding Author (C.C.L. Wang): cwang@mae.cuhk.edu.hk



(a) $E = 13490.0KJ$ (b) $E = 7556.1KJ$

Fig. 1. Different from the prior method for CPP only considering the geometry – e.g., (a) the result from Lin et al. [3], the spiral path generated by our method leads to less energy consumption (b). Here the energy cost E is evaluated by the widely used model in [4], the weight of mobile robot is 50kg and the friction coefficient between the mobile robot and the terrain surface is assigned as $\mu = 0.2$.

Most mobile robots are equipped with portable energy storage devices. Limited by the power sources, the area that can be covered highly depends on the energy consumption during the exploration.

- In many cases, the widely used zig-zag pattern is not an energy-efficient choice. The back-and-forth coverage would bring more unnecessary energy consumption when encounters mountains or valleys on a terrain surface. A key observation is that the energy required in a fixed 2D slice of the input surface remains fixed regardless of the traversal, but an increased energy consumption occurs when the coverage path traverses in the z-direction in a redundant manner.
- Another desired feature for mobile robots in CPP is that the start position and end position prefer to be close to each other since charging devices and refuelling depots are usually not portable. Although the widely adopted 2D zig-zag pattern can also be modified to satisfy this demand of 'nearby' starting and ending points, the distance between paths is difficult to control when mapping 2D zig-zag paths onto a 3D terrain surface.

Here we propose an efficient heuristic method to tackle the CPP problem considering these factors simultaneously.

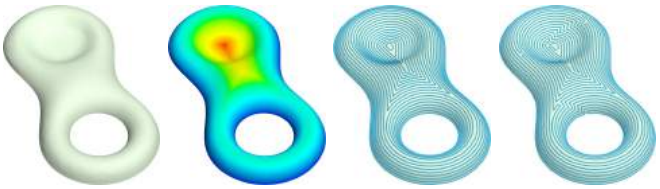
Specifically, the energy cost caused by gravity is effectively reduced by a well designed pattern – the height-extremity-aware Fermat spiral. According to a recent survey of CPP [5], spiral-based paths have been used in prior research. Different from the existing works such as [6], [7], our coverage path is globally connected and generated upon an energy-aware 3D field with a heuristic that iso-contours are aligned with similar height-values and can avoid

revisiting. Iso-contours of a field is the set of points that have the same field value. In this approach, we can control the distance between paths in a conservative way to ensure the full coverage – i.e., for a robot with coverage radius r, the geodesic distance (i.e., distance on curved surface) between iso-contours can be selected as $r/2$ to generate the final spiral path. Or we can also choose the geodesic distance in an aggressive way as $r/2$ to general a shorter path but with less coverage. Note that, exploring along a shorter path usually consumes less energy. For example, as shown in Fig.1, the energy consumption of a path generated by using an aggressive strategy in our method can be reduced by more than 40% compared to a recent 3D-CPP method also developed for general surfaces [3].

A. Related Work

The CPP problem has been studied for a long time [1], [5]. For 2D fields, existing methods can be roughly summarized into two categories: decomposition-based and miscellaneous. Decomposition-based methods plan paths on each region and then finish the global coverage. According to the type of decomposition, these methods can be further classified into trapezoid decomposition [8], [9], boustrophedon decomposition [10]–[12], Morse-based cellular decomposition [13], [14], and exact cell decomposition [15]. Some other methods are based on grid representation of the space to be covered such as wavefront algorithm [16], spanning trees coverage [17], neural network based coverage [18], [19]. These methods are suitable for indoor mobile robot operation since the area to be covered is typically small. In the real world, the scenarios of many application are three-dimensional (3D). The rapid development of sensors and actuators drives the motivation that robots can take over jobs in a complicated 3D environment. Many algorithms for CPP in 3D space are also based on decomposition. For instance, Hert et al. [20] propose an algorithm which applies a 2D coverage algorithm on successive horizontal spaces and entirely covers the 3D vollume. Atkar et al. [21] extend the idea of Morse decomposition to 3D surfaces for spray-painting robots. Recently, Lin et al. [3] propose a natural, intrinsic and global parametrization, called *holomorphic quadratic differentials*, to induce non-intersecting trajectories for CPP. However, energy consumption has not been considered in their approach.

In the thread of energy-efficient CPP, Jin and Tang [22] redesign zig-zag patterns from straight line to seed curve to cover 3D surfaces with reduced headland turning cost, soil erosion cost and skipped area. Hameed [23] incorporates the factors of field operational time, fuel consumption, non-productive traveled distance and maneuvering over the terrain surfaces and propose a method that first partitions the terrain according to obstacles and then applies a genetic algorithm to find the optimal driving direction of the paths. Hameed et al. [24] further propose a side-to-side algorithm to improve the performance of coverage in [23], in which only a single track is generated. Dogru and Marques [25] decompose the terrain with the considerations of not only the obstacles but also the natural curvature of the environment. Wei and Isler [26] consider the reality of battery limitations and propose an algorithm to plan multiple closed paths to cover the whole region. These methods have been successfully used in uncomplicated terrain surfaces, such as agricultural fields. However, unlike our approach, they cannot be applied to general terrain surfaces with large height variation and many obstacles due to the difficulty of general terrain decomposition.



(a) Input $\mathcal{M}$ (b) Field $\mathcal{D}_{\mathcal{M}}$ (c) Iso-contours (d) Spiral Path

Fig. 2. An illustration of the steps for generating a spiral path to cover a given terrain surface with the help of an exact geodesic field, where the color map shows a height-extremity-aware geodesic distance field $\mathcal{D}_{\mathcal{M}}$.

B. Contribution

By tackling the CPP problem with concern on energy consumption, we make the following contributions in this paper.

- Based on the exact boundary-sourced geodesic distance field, we can generate geodesic Fermat spiral paths on a 3D surface to solve the CPP problem.
- By incoporting the height-extremity into the computation of geodesic distance field, we propose a heuristic algorithm to plan the Fermat spiral paths for CPP in an energy-efficient way.

The coverage of our CPP is commonly guaranteed by the exact computation of geodesic distance, the conservative strategy of choosing iso-contours for generating the spiral path, and the circular coverage pattern of a robot. The demand of no overlap is weakly preserved by the pattern of Fermat spiral (see Fig.2 for the steps of path generation).

It has been verified by experiments that our method can significantly reduce the energy consumption while simultaneously preserving other requirements for a satisfactory CPP solution. Note that, we do not consider the energy consumption on sharp turns as it has no influence on some robots – e.g., an omnidirectional robot uses almost no extra energy at sharp corner when running on its path.

II. Preliminaries and Problem Definition

A. Terrain representation

Terrain modeling is an important step before analyzing and planning. According to [27], a terrain surface is always represented as a digital elevation model and can be converted to a triangular mesh surface with boundaries to be used in a computer system. In short, a triangle mesh $\mathcal{M}$ consists of geometric components and topological components, which is represented by a set of vertices $\mathcal{V}$ and a set of triangular

facets $\mathcal{F}$. It is called a 2-manifold (possibly with boundaries and holes) if every *interior* point on the mesh surface is locally homeomorphic to a disk and every boundary point is locally homeomorphic to a half-disk. This is an important criterion to characterize the topological quality of a triangle mesh. The geodesic metric used in our approach relies on the correctness of 2-manifold presented on a terrain surface. After converting obstacles into holes on a mesh surface, the paths generated by our method will be intrinsically collision-free.

B. Computation of exact geodesic distance-field

There are two classes of techniques for computing discrete geodesics on triangle meshes. One is the computational geometry approaches and the other is the *partial differential equation* (PDE) approaches (e.g., *fast marching method* (FMM) [28]). The classical computational geometry approaches include *Mitchell-Mount-Papadimitriou* (MMP) algorithm [29] and *Chen-Han* (CH) algorithm [30]. Each class of approaches has its own merits and limitations. The MMP algorithm can obtain the exact geodesics on arbitrary triangle meshes if numerical operations are exact. PDE approaches are usually faster than the MMP algorithm, but can only provide approximate solutions (e.g., the first-order approximation by the FMM), which may be poor on meshes with highly irregular tessellation. In our work, we apply the *Fast Wavefront Propagation MMP* (FWP-MMP) algorithm [31] which is fast and provides exact geodesics.

The FWP-MMP method computes the exact geodesic distance field over a surface by propagating *windows*, which are a set of partitioned intervals on mesh edges. All points in a *window* defined on an edge sharing the same geodesic path to the source point. The window pruning strategy is important for the efficiency of distance computation. Specifically, we use polyline-source geodesic methods [32], [33], which are able to construct a geodesic distance field with respect to polyline sources on meshes. There are two types of windows in this case: line-source windows and point-source windows (as shown in the left of Fig.3). As a result, the window intersections and trimming operations can have three different combinations as point-source/point-source, line-source/line-source and point-source/line-source. More details of FWP-MMP can be found in [33].

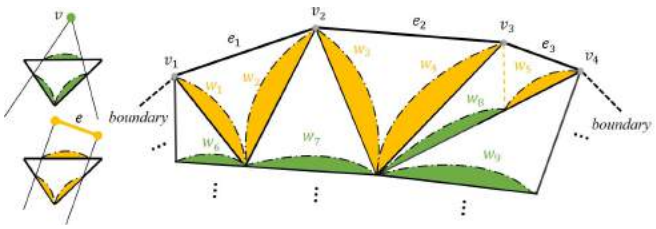


Fig. 3. The propagation based computation of geodesic distances on surfaces encoded with point-source and edge-source: (left) the illustration of windows from point-source (top) and edge-source (bottom), and (right) an example of window propagation from edge-sources located on the surface boundary.

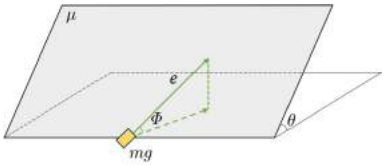


Fig. 4. The energy model of [4] is used in our work to verify the energy-efficiency of CPP.

The boundary of a terrain surface may consist of multiple disjoint components and each component is a closed polyline. We set every line segments and points on the boundary as sources such that an exact geodesic distance field $\mathcal{D}_{\mathcal{M}}$ for generating iso-contours can be computed by following the strategy of *windows* propagation presented in [31]. A propagation example with edge-sources on the boundary is shown in the right of Fig.3.

C. Energy consumption

The energy model used in our validation for evaluating the energy-efficiency of a path is based on the analysis of anisotropic friction and gravity [4], which has been successfully adopted in energy-minimizing path planning [34]. For a mobile robot travelling along an arbitrary short path e with slope θ on a terrain surface $\mathcal{M}$, the energy consumption is defined as

$$E = \max(mg(\mu\cos\phi + \sin\theta)\cdot||e||, 0) \tag{1}$$

where ϕ is the inclination angle of the gradient and mg is the weight of robot. The friction coefficient μ between the mobile and all terrain surfaces in our evaluations is set as 0.2 (see Fig. 4). In practice, their model has been confirmed within 1% of error for wheeled vehicles traveling on shallow slopes.

After introducing the background and all necessary preliminaries, we formally define our problem as follows.

Problem statement: Given a terrain surface Ω, we generate an energy-efficient coverage path Π for a robot having circular coverage pattern (with radius r) that minimizes

$$J = \int_{\Pi} \mathrm{U}(\Delta h)|\Delta h| ds \tag{2}$$

where Δh indicates the height variation and $\mathrm{U}(\cdot)$ is the Heaviside step function.

The energy consumption of a mobile robot to cover a terrain surface Ω heavily relies on the climbing part when ensuring the coverage. This is because that braking in downhills would convert a portion of accumulated gravitational potential energy to the wasted heat energy [4]. Therefore, we need to plan a coverage path minimizing J. However, J is difficult to be optimized explicitly, so we minimize its alternative form as

$$\tilde{J} = \int_{\Pi} |\Delta h| ds \tag{3}$$

according to the following analysis with Π_s and Π_e defining the starting and the ending points of the path Π.

Lemma: A minimized $\tilde{J}$ (i.e., Eq.(3)) leads to a minimized value of J (i.e., Eq.(2)) in CPP problem when $\Pi_s = \Pi_e$.
Proof: As we are tackling the CPP problem, the traversal of coverage path Π is invertible as $\Pi_s = \Pi_e$. We have

$$\int_\Pi |\Delta h| ds = \int_\Pi \mathrm{U}(\Delta h)|\Delta h| ds + \int_\Pi \mathrm{U}(-\Delta h)|\Delta h| ds,$$

where $\int_\Pi \mathrm{U}(-\Delta h)|\Delta h| ds = \int_\Pi \mathrm{U}(-\Delta h)|(-\Delta h)| ds$ can be considered as J defined on an inverse path of Π. According to Eq.(2), the value of J is non-negative. Therefore, while reducing the value of $\int_\Pi |\Delta h| ds$, one of $\int_\Pi \mathrm{U}(\Delta h)|\Delta h| ds$ and $\int_\Pi \mathrm{U}(-\Delta h)|\Delta h| ds$ will also be reduced. Based on the invertibility of a path in CPP, we can always pick the one with reduced energy as an optimized path.

III. Spiral Path Planning

Fermat spirals are defined by $r = \pm\sqrt{\theta}$ in polar coordinates [35]. Similar to other patterns such as zig-zag, contour parallel or Hilbert, a Fermat spiral is an interesting pattern that can be used in filling or covering a space. The space filled by this pattern shows less sharp turns [36]. The construction of the Fermat spiral pattern is based on distance fields, which provides the freedom for users to adjust by placing different sources or alternating the fields.

The spiral path generation consists of three major steps (see Fig.2). First, an exact geodesic distance field $\mathcal{D}_\mathcal{M}$ is constructed on the input surface $\mathcal{M}$ with sources defined on (1) the boundaries and (2) the peaks of height-field. Then, iso-contours(i.e., contour curves with the same field values) are extracted from the field. In the third step, a connected spiral path is generated from the iso-contours.

A. Iso-contours construction

Based on the exact geodesic, we use the following method to extract iso-contours from the geodesic distance field $\mathcal{D}_\mathcal{M}$.

As it is possible to have multiple extrema on an edge $e \subset \mathcal{M}$, we follow the pre-processing step in [37] to partition edges and triangles of $\mathcal{M}$ so that each resultant edge only contains a monotonic portion of the field $\mathcal{D}_\mathcal{M}$. This pre-processing step runs in a recursive manner and has the complexity $O(m^2)$ in the worst case with m being the number of triangles on $\mathcal{M}$. As a result, each edge e on the resultant mesh $\mathcal{M}^*$ has either a line-source window or a monotonic point-source window.

Structure of iso-contours Due to the propagation of two different types of windows during the process of constructing $\mathcal{D}_\mathcal{M}$, the iso-contours on $\mathcal{M}$ have the following analytical structure: (1) an iso-contour may consist of one or more disjoint components, (2) each component is a closed curve and (3) each closed curve consists of circular arc and line segment.

- An **arc segment** is determined by two windows that have the same nearest point-source, and any point on the circular arc has the same distance (i.e., iso-value) to the point-source.
- A **line segment** is determined by two windows that have the same nearest line-source. For any pair of linear windows satisfying this criterion, the line segment and its corresponding line-source should be parallel and therefore have a constant distance (i.e., with the same iso-value on the line segment).

Algorithm 1: Generate Iso-Contours

Input: An iso-value d and a pre-processed mesh surface $\mathcal{M}^*$ with $\mathcal{D}_\mathcal{M}$.
Output: A set of closed curves $\mathcal{C}$.

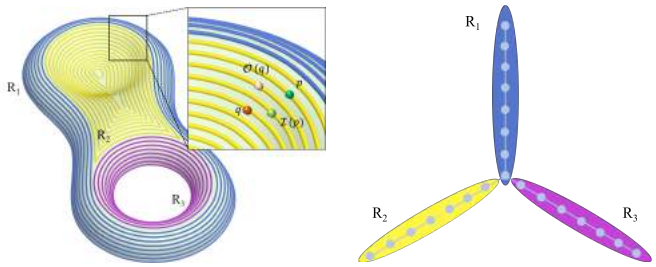
```
Sort all edges of M* to construct an interval tree T;
Stabbing query the iso-value d in T and export to a set
  of intersected edges in L;
while L ≠ ∅ do
    Remove the first edge e_top of L;
    Marching edges from e_top by half-edge flip while
      connecting the points with the iso-value d on them
      by line segments;
    Add the newly created line segments to C;
    Remove marched edges from L;
end
```

Construction A practical algorithm is presented here to construct a valid iso-contour $\mathcal{D}_\mathcal{M}(\mathbf{p}) = d$ for a given iso-value d. After the pre-processing step of refining $\mathcal{M}$ into a mesh $\mathcal{M}^*$ with simple configurations of field-values on all edges, the source information of circular window and linear window have already been well-stored to help the construction of iso-contours. An interval tree that sort all edges of $\mathcal{M}^*$ by the distance intervals covered by them is first built to accelerate queries. Given an iso-value d, our algorithm employs the stabbing query to find all edges whose intervals across d and defines them as *intersected edges*. Then, the algorithm runs in a marching manner, where the process starts from an intersected edge and then flips to its neighboring triangle to find the next intersected edge. A portion of the iso-contour between these two intersected edges can be generated by connecting the points with the iso-value d on them by a line segment. The algorithm runs iteratively until all intersected edges have been visited. When degenerate case happens (e.g., iso-contour passes through a vertex of $\mathcal{M}^*$), the flip step will check all triangles around this vertex to find a correct marching direction. A pseudo-code of this algorithm can be seen in Algorithm 1.

Denote $\partial\mathcal{M}$ as the boundary of $\mathcal{M}$, the distance transformation $\mathcal{D}_\mathcal{M}(d)$ defines the iso-contour(s) having the geodesic distance d with respect to $\partial\mathcal{M}$. In our CPP with spiral paths, the iso-contours are generated by computing a set of $\mathcal{D}_\mathcal{M}(d_i)$ over $\mathcal{M}$. The difference between neighboring iso-values is a parameter depending on the local area that can be covered by a field exploration robot.

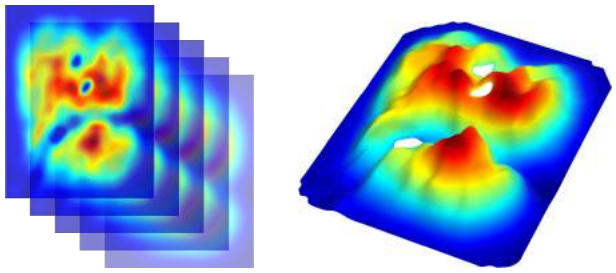
B. Generation of connected spirals

Iso-contour is a special geometric structure that may have multiple disjoint components corresponding to a distance transformation $\mathcal{D}_\mathcal{M}(\mathbf{p})$ ($\forall \mathbf{p} \in \mathcal{M}$). If every transformation in a set of consecutive distance transformation $\{\mathcal{D}_\mathcal{M}(\mathbf{p}) = d_i\}$ has a single connected component, these transformations



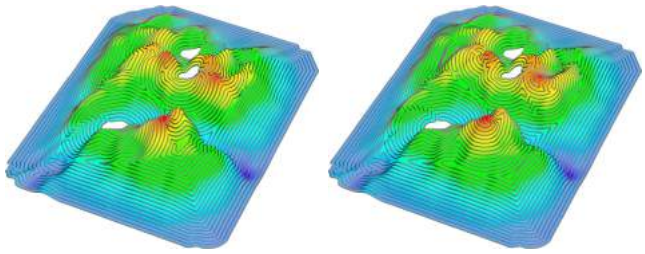
(a) Grouped contours (b) Spanning tree

Fig. 5. Spiral paths can be generated from iso-contours of a field $\mathcal{D}_{\mathcal{M}}$, and a minimal spanning tree helps to connect the simple set of spirals to cover regions with a complicated topology where each node represents an iso-contour.



(a) Filtered height-fields (b) Blended height-field

Fig. 6. Peak points can be obtained by nonlinearly blending the height-fields filtered in different scales.



(a) $E = 8134.5KJ$ (b) $E = 7556.1KJ$

Fig. 7. (a) Without considering the peak and valley points leads to a path with much higher energy consumption than (b), in which they are added as sources for generating a geodesic field. Again, the energy cost E is evaluated by the widely used method in [4].

(i.e., contours) can be spiraled to a connected path using a method similar to [36]. The difference $\Delta d = d_{i+1} - d_i$ is a parameter that can be adjusted according to the coverage area of different mobile robots. We select its value according to a conservative strategy to ensure the coverage of CPP. Specifically, considering the usual circular covering pattern (with radius r) provided by a robot, we use $\Delta d = r/2$ as the distance value between neighboring iso-contours. This is because the distance between neighboring iso-contours near the medial-axis region can vary in the range of $[\Delta d, 2\Delta d)$. Therefore, using $r/2$ as the sampling interval between iso-contours can guarantee the coverage of our spiral path even at the medial-axis region.

For the rest of this section, we simply denote the contour with iso-value d_i by $\mathcal{D}_i$. Given a starting point $\mathbf{q} \in \mathcal{D}_i$ with $i = 0$, we firstly precede $\mathbf{q}$ along $\mathcal{D}_i$ and then connect it with its inward point $\mathcal{I}(\mathbf{q}) \in \mathcal{D}_{i+2}$. The process executes iteratively until reaching the innermost contour. After that, we start from the innermost contour and execute the inversed process, which succeeds $\mathbf{q}$ along $\mathcal{D}_k$ and then connects with its outward point $\mathcal{O}(\mathbf{q}) \in \mathcal{D}_{k-2}$ until reaching the outer most contour $\mathcal{D}_0$. An illustration can be found in Fig.5. More algorithm details can be found in [36]. The fundamental difference here is that the distance metric of inward and outward operations is based on an exact geodesic field as described above.

For a more complicated case in which an iso-contour has multiple disjoint components, we construct a graph $\mathcal{G}$ to represent the neighboring information between the simply connected contours (as illustrated in Fig.5). Then, a minimal spanning tree is applied to generate the sequence for connecting these contours by the depth-first-search order. As a result, a continuous path can be generated to cover a given surface by multiple spiral regions, where each region is covered by a local spiral (see Fig.2(d) for example).

IV. Energy-Oriented Optimization

With the help of the algorithm presented above, we are able to generate a spiral path Π to cover a general terrain surface $\mathcal{M}$. To accomplish the goal of energy-efficient exploration of a 3D terrain surface, energy cost needs to be analyzed and considered during the path planning. It is obvious that a path with frequent uphill and downhill travels on terrain may result in unnecessary consumption of energy [4]. In this section, we propose a heuristic strategy for energy-aware optimization, which can be naturally integrated into our algorithm of spiral path generation.

Due to the constraint of gravity, faces on a terrain surface should be all facing-up; otherwise, the surface cannot be visited by mobile robots. In general, it would be energy efficient if mountains and valleys are visited with minimal total height variation on a path. Based on this observation, we introduce a heuristic to decrease the usage of energy when conducting CPP on a terrain surface. The regions that dominate the energy consumption will be first extracted from the height-field $H_{\mathcal{M}}(\mathbf{p})$ of $\mathcal{M}$; then, the center of these regions will be added as sources for generating a height-extremity-aware geodesic field. With these changes, the finally generated spiral paths will significantly reduce the total height variations as

$$\bar{J} = \int_{\Pi} |H(\mathbf{q}) - H(\mathbf{q} + \delta\mathbf{q})| d\mathbf{q}. \tag{4}$$

The total energy consumption therefore will also be significantly reduced. Similar to [4], we consider brake when descending hills.

The peaks and the valleys of $H_{\mathcal{M}}(\mathbf{p})$ are considered separately on a shifted height-field $H_+(\mathbf{p})$ and an inverse height-field $H_-(\mathbf{p})$ respectively.

$$H_+(\mathbf{p}) = H_{\mathcal{M}}(\mathbf{p}) - \min(H_{\mathcal{M}}(\mathbf{p}))$$
$$H_-(\mathbf{p}) = -H_{\mathcal{M}}(\mathbf{p}) + \max(H_{\mathcal{M}}(\mathbf{p}))$$

To robustly extract the peak points, we apply the Gaussian filter in different scales to $H_{\mathcal{M}}(\mathbf{p})$ – i.e., $\sigma_i = i\sigma$ with $i = 1, \cdots, 5$ and $\sigma = 5\Delta d$ are used in our examples. The filtered height fields are denoted as $G(H_+, \sigma_i)$. For blending these height-fields in different scales, the nonlinear suppression method proposed in [38] is applied in three steps:

- First, normalize $G(H_{\mathcal{M}}, \sigma_i)$ to a fixed range $[0, 1]$ as $\tilde{G}(H_{\mathcal{M}}, \sigma_i)$;
- Second, find its global maximal $g_i^{\max}$ and compute the average of all local maximums except the global maximal as $\bar{g}_i$;
- Third, form the nonlinear combination of multiple height fields as

$$\tilde{H}_+(\mathbf{p}) = \sum_{i=1}^{5} (g_i^{\max} - \bar{g}_i)^2 \tilde{G}(H_+, \sigma_i). \qquad (5)$$

Finally, each local maximal point on $\tilde{H}_{\mathcal{M}}(\mathbf{p})$ is a peak point of $H_{\mathcal{M}}(\mathbf{p})$ (see Fig.6 for an illustration). When applying the same method to $H_-(\mathbf{p})$, the valley points can be found from $\tilde{H}_-(\mathbf{p})$. Note that local maximums located on the boundary of $\mathcal{M}$ are excluded from the detection of height-extremity.

After inserting the peak and valley points as sources to generate a new geodesic field, we can compute an energy-aware spiral path by the same algorithm presented in Section III. As shown in Fig.7, the energy consumption can be reduced by 7.1% after incorporating the peak and valley points as sources.

V. Experimental Results

We have implemented the algorithms presented in this paper in a C++ program and tested its performance on a variety of surface patches. All examples are tested on a laptop with Intel i5-8300H Quad Core CPU and 16GB RAM. To verify the advantages of our approach, we compare our results with a state-of-the-art CPP method [3] that can also work on a complex terrain surface. The performance of our method is analyzed in terms of energy consumption, coverage rate and overlap rate. Note that, although the full coverage can be ensured by using a conservative strategy (i.e., $\Delta d = r/2$), it is also interesting to study the effectiveness of coverage when an aggressive strategy is adopted (i.e., $\Delta d = r$). The aggressive strategy is obviously more energy efficient as it results in much shorter paths.

We have performed physical experiments by using a Pioneer 3AT robot. A DC power analyzer that has build-in sensors for voltage and current is used for collecting real-time data of energy consumption. The robot uses a client-server structure, where an onboard PC acts as the decision-making unit and sends movement commands to the micro-controller via a serial port. The experimental results and more details about the power analyzer can also be found in the supplementary video.

A. Climbing energy verification

To verify that frequent uphill climbing will consume more energy, we set up five different experiments, including 1) go straight, 2) 90° left turn, 3) 90° right turn, 4) climb a slope with 10° and 5) go downhill the same slop. The left of Fig.8 shows the slope used in our tests. The results of our experiments given in the right of Fig.8 show that the energy consumption during climbing is very intensive, but the consumption of going downhill is similar to the one of going straight on the flat ground. This is mainly because that the accumulated gravitational potential energy cannot be fully converted to kinetic energy due to friction and brakes.

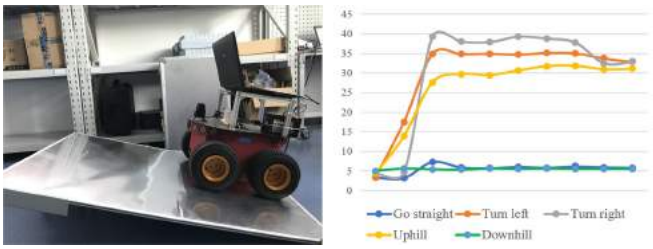


Fig. 8. We build a testing environment to verify that additional energy consumption is needed when the mobile robot goes uphill (left) while the gain of accumulated energy cannot be fully used when the robot goes downhill. (Right) The energy consumption under different testing scenarios.

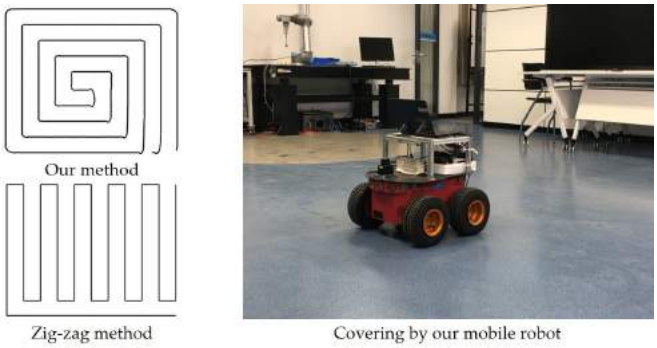


Fig. 9. Turning energy verification by an indoor experiment.

B. Turning energy verification

As a byproduct, the spiral path generated by our method naturally contains less sharp turns comparing to the conventional zig-zag pattern. We then compare our CPP method with the conventional zig-zag method at an indoor environment – with a dimension of 5×4.4m as shown in Fig.9. A robot with differential drive (i.e., Pioneer 3AT) is used in this test to reflect the additional energy consumption at sharp turns. The length of the path planned by our method is $52.9m$, and the length of a zig-zag path is $53.4m$. We set the tracking speed of straight lines as a constant value $v = 200mm/s$. The overall energy consumption of the zig-zag path is $9.288KJ$ while our method consumes $7.944KJ$ – leading to 14.5% saving of energy. Both are measured by the DC power analyzer. This is because the zigzag path consists of 180-degree turns while ours only have 90-degree turns. The energy consumption of differential drive mobile robot is sensitive to the 'sharpness' of turning.

Fig. 10. Experiments taken in an outdoor environment for verifying the energy consumption by following a zig-zag path vs. a spiral path generated by our method (see the path shown on the reconstructed surface).

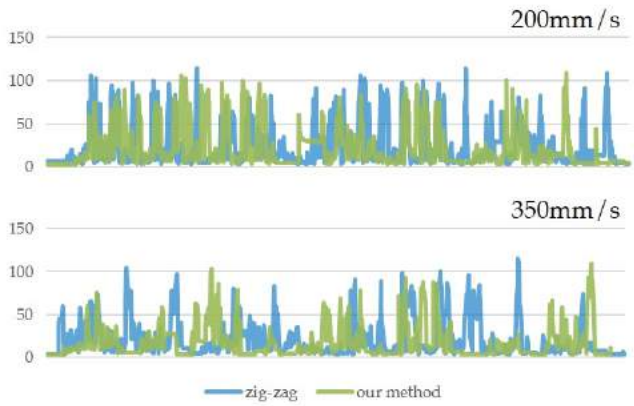


Fig. 11. Comparison between the zig-zag method and our method in the outdoor environment shown in Fig.10 with the robot running in two different speeds. The vertical axis gives the instant energy consumption (unit: Watt), and the horizontal axis is time (sec.).

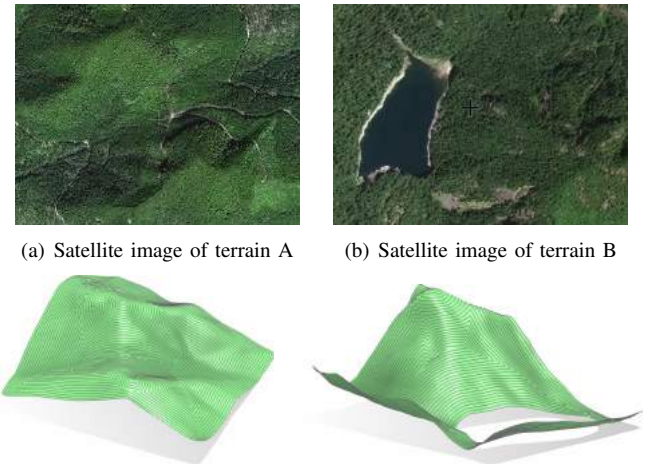
C. Outdoor terrain surveillance

We verify the performance of our energy-efficient CPP algorithm on an outdoor terrain located at [N32°02′10.2″ E118°49′08.3″]. As shown in Fig.10, we first use a high-precision Lidar device to collect the 3D point cloud of this outdoor terrain. Since the surface of terrain is covered with grasses, we filter the noisy point cloud by keeping best points evenly spaced. After that, we upsample the filtered point cloud and construct a smooth surface to represent the terrain.

The size of the reconstructed terrain surface is around $10m \times 5m$, the height variation is around $0.58m$. The grasses and wet soil make the outdoor environment more complicated than the indoor environment for our mobile robot to cover. We run our robot in two different velocities $200mm/s$ and $350mm/s$ in our experiments. Both experimental results are encouraging – our method consumes 30.952KJ and 13.766KJ energy while the zig-zag path uses 36.131KJ and 17.052KJ energy respectively. The comparison of energy consumption during exploration is shown in Fig.11.

D. Tests on large scale terrains

We apply our method to large scale terrains of forests to further demonstrate the effectiveness of our method. Fig.12(a) shows the terrain A located at [N47°44′09.9″ W122°55′27.2″] and Fig.12(b) shows the terrain B located at [N47°37′00.5″ W123°09′26.0″]. The terrain B has a lake obstacle represented by an empty hole which is difficult to handle for many existing CPP approaches. We generate height-extremity-aware Fermat spiral paths on these fields and the results are shown in Fig.12(c) and (d) respectively.



(a) Satellite image of terrain A (b) Satellite image of terrain B

(c) Coverage path on terrain A (d) Coverage path on terrain B

Fig. 12. More results of paths generated by our algorithm to cover different terrain surfaces.

1) Coverage and overlap: It is interesting to study the percentage of coverage and overlap (i.e., *redundancy*) when an aggressive strategy is applied to generate paths for exploration. We evaluate our algorithm on two different terrain surfaces as shown in Fig.12.

To efficiently estimate the approximate coverage and overlap rates on 3D terrain surfaces, we compute the swept volume of the generated path by a sphere with radius Δd, which results in a solid $\mathcal{R}$. Here Δd is considered as the support radius of the area that can be locally covered by a robot. The area that covered by the mobile robot's travel can then be defined as $\mathcal{R} \cap \mathcal{M}$. The coverage rate is evaluated as $Area(\mathcal{R} \cap \mathcal{M})/Area(\mathcal{M})$. For evaluating the overlapped region, we evaluate the self-intersected region by using a robust 3D inside-outside segmentation algorithm [39]. The overlap rate is defined as the ratio between the area of the self-intersected region and the area of coverage (i.e., $Area(\mathcal{R} \cap \mathcal{M})$).

As the distance metric we used is exact geodesic on 3D surfaces, the coverage rate and overlap rate outperforms significantly than previous methods. As reported in Table I, our paths can achieve a very high coverage rate while keeping a very low overlap rate at the same time.

2) Energy consumption: The climbing energy defined in [4] is employed here to verify the effectiveness of our approach. With the help of the advantage of height-extremity-aware smooth Fermat spiral, the generated paths outperform the state-of-art method [3] in all aspects. Comparison of results has been listed in Table I.

TABLE I

STATISTIC OF GENERATED SPIRAL PATHS

Surface	Terrain A by ours	Terrain B by ours	Mountains by ours	Mountains by [3]
Coverage Rate	92.2 %	92.4 %	94.1 %	82.3 %
Overlap Rate	0.8 %	0.6 %	1.2 %	71.4 %
Climb Energy	15.42 MJ	11.29 MJ	7.56 MJ	13.49 MJ
Path Length	126.86km	85.12km	62.50km	87.80km
Planning Time	7.6 sec.	7.0 sec.	7.4 sec.	\
Figure	12(c)	12(d)	1(b)	1(a)

VI. CONCLUSIONS AND FUTURE WORKS

We have developed a new method to generate an energy-efficient coverage path planning for exploring a given terrain surface, which can have a complicated topology with holes and irregular boundaries. In summary, height-extremity-aware paths are generated in the pattern of smooth Fermat spiral, and such paths can significantly reduce the energy consumption caused by gravity changing. The effectiveness of our method has been verified on a few examples.

In our current implementation, the nonlinear blending of Gaussian filtered height-fields is used to extract the peak and valley points to incorporate height information into the generation of spiral paths. A more advanced method will be investigated in our future research to extract other types of extremities on the height-field of a terrain surface (e.g., crest lines).

REFERENCES

[1] H. Choset, "Coverage for robotics–a survey of recent results," *Ann. Math. Artif. Intell.*, vol. 31, no. 1, pp. 113–126, 2001.

[2] E. Arkin, S. Fekete, and J. Mitchell, "Approximation algorithms for lawn mowing and milling," *Comput. Geom.*, vol. 17, no. 1-2, pp. 25–50, 2000.

[3] Y. Y. Lin, C. C. Ni, N. Lei, X. D. Gu, and J. Gao, "Robot coverage path planning for general surfaces using quadratic differentials," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2017, pp. 5005–5011.

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

[5] E. Galceran and M. Carreras, "A survey on coverage path planning for robotics," *Robot. Auton. Syst.*, vol. 61, no. 12, pp. 1258–1276, 2013.

[6] E. González, O. Alvarez, Y. Diaz, C. Parra, and C. Bustacara, "BSA: A complete coverage algorithm," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2005, pp. 2040–2044.

[7] M. Bosse, N. Nourani-Vatani, and J. Roberts, "Coverage algorithms for an under-actuated car-like vehicle in an uncertain environment," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2007, pp. 698–703.

[8] R. N. D. Carvalho, H. A. Vidal, P. Vieira, and M. I. Ribeiro, "Complete coverage path planning and guidance for cleaning robots," in *Proc. IEEE Int. Symp. Ind. Electron.*, vol. 2, 1997, pp. 677–682.

[9] T. Oksanen and A. Visala, "Coverage path planning algorithms for agricultural field machines," *J. Field Robot.*, vol. 26, no. 8, pp. 651–668, 2009.

[10] H. Choset and P. Pignon, "Coverage path planning: The boustrophedon cellular decomposition," in *Proc. Field Serv. Robot.*, A. Zelinsky, Ed., 1998, pp. 203–209.

[11] E. Garcia and P. G. de Santos, "Mobile-robot navigation with complete coverage of unstructured environments," *Robot. Auton. Syst.*, vol. 46, no. 4, pp. 195–204, 2004.

[12] A. Xu, C. Viriyasuthee, and I. Rekleitis, "Optimal complete terrain coverage using an unmanned aerial vehicle," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2011, pp. 2513–2519.

[13] E. U. Acar, H. Choset, A. A. Rizzi, P. N. Atkar, and D. Hull, "Morse decompositions for coverage tasks," *Int. J. Robot. Res.*, vol. 21, no. 4, pp. 331–344, 2002.

[14] E. Galceran and M. Carreras, "Efficient seabed coverage path planning for asvs and auvs," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2012, pp. 88–93.

[15] S. C. Wong and B. A. MacDonald, "A topological coverage algorithm for mobile robots," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, vol. 2, 2003, pp. 1685–1690.

[16] A. Zelinsky, R. A. Jarvis, J. C. Byrne, and S. Yuta, "Planning paths of complete coverage of an unstructured environment by a mobile robot," in *Proc. Int. Conf. Adv. Robot.*, vol. 13, 1993, pp. 533–538.

[17] Y. Gabriely and E. Rimon, "Spiral-stc: An on-line coverage algorithm of grid environments by a mobile robot," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2002, pp. 954–960.

[18] S. X. Yang and C. Luo, "A neural network approach to complete coverage path planning," *IEEE Trans. Syst., Man, Cybern. B*, vol. 34, no. 1, pp. 718–724, Jan. 2004.

[19] Y. Guo and M. Balakrishnan, "Complete coverage control for non-holonomic mobile robots in dynamic environments," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2006, pp. 1704–1709.

[20] S. Hert, S. Tiwari, and V. Lumelsky, "A terrain-covering algorithm for an AUV," *Auton. Robot.*, vol. 3, no. 2, pp. 91–119, 1996.

[21] P. N. Atkar, H. Choset, A. A. Rizzi, and E. U. Acar, "Exact cellular decomposition of closed orientable surfaces embedded in $\Re^3$," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2001, pp. 699–704 vol.1.

[22] J. Jin and L. Tang, "Coverage path planning on three-dimensional terrain for arable farming," *J. Field Robot.*, vol. 28, no. 3, pp. 424–440, 2011.

[23] I. A. Hameed, "Intelligent coverage path planning for agricultural robots and autonomous machines on three-dimensional terrain," *J. Intell. Robot. Syst.*, vol. 74, no. 3, pp. 965–983, Jun. 2014.

[24] I. Hameed, A. la Cour-Harbo, and O. Osen, "Side-to-side 3D coverage path planning approach for agricultural robots to minimize skip/overlap areas between swaths," *Robot. Auton. Syst.*, vol. 76, pp. 36 – 45, 2016.

[25] S. Dogru and L. Marques, "Energy efficient coverage path planning for autonomous mobile robots on 3D terrain," in *Proc. IEEE Int. Conf. Auton. Robot Syst. Competitions*, 2015, pp. 118–123.

[26] M. Wei and V. Isler, "Coverage path planning under the energy constraint," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2018, pp. 368–373.

[27] P. A. Longley, *Geographical Information Systems: Principles, Techniques, Management and Applications*. New York, NY, USA: John Wiley & Sons, Inc., 2005.

[28] R. Kimmel and J. A. Sethian, "Computing geodesic paths on manifolds," *Proc. Natl. Acad. Sci.*, vol. 95, no. 15, pp. 8431–8435, 1998.

[29] J. Mitchell, D. Mount, and C. Papadimitriou, "The discrete geodesic problem," *SIAM J. Comput.*, vol. 16, no. 4, pp. 647–668, 1987.

[30] J. Chen and Y. Han, "Shortest paths on a polyhedron," in *Proc. 6th Ann. Symp. Comput. Geom.* ACM, 1990, pp. 360–369.

[31] C. Xu, T. Y. Wang, Y. J. Liu, L. Liu, and Y. He, "Fast wavefront propagation (FWP) for computing exact geodesic distances on meshes," *IEEE Trans. Vis. Comput. Graph.*, vol. 21, no. 7, pp. 822–834, Feb. 2015.

[32] C. Xu, Y. Liu, Q. Sun, J. Li, and Y. He, "Polyline-sourced geodesic voronoi diagrams on triangle meshes," in *Comput. Graph. Forum*, vol. 33, no. 7, 2014, pp. 161–170.

[33] D. Bommes and L. Kobbelt, "Accurate computation of geodesic distance fields for polygonal curves on triangle meshes." in *VMV*, vol. 7, 2007, pp. 151–160.

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

[35] "Fermat's spiral - wikipedia," (Date last accessed 02-Sept-2018). [Online]. Available: https://en.wikipedia.org/wiki/Fermat's_spiral

[36] H. Zhao, F. Gu, Q. Huang, J. Garcia, Y. Chen, C. Tu, B. Benes, H. Zhang, D. Cohen-Or, and B. Chen, "Connected Fermat spirals for layered fabrication," *ACM Trans. Graph.*, vol. 35, no. 4, p. 100, 2016.

[37] Y. Liu, Z. Chen, and K. Tang, "Construction of iso-contours, bisectors, and voronoi diagrams on triangulated surfaces," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 8, pp. 1502–1517, Dec. 2011.

[38] L. Itti, C. Koch, and E. Niebur, "A model of saliency-based visual attention for rapid scene analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 11, pp. 1254–1259, Nov. 1998.

[39] A. Jacobson, L. Kavan, and O. Sorkine-Hornung, "Robust inside-outside segmentation using generalized winding numbers," *ACM Trans. Graph.*, vol. 32, no. 4, pp. 33:1–33:12, 2013.