Computer-Aided Design 114 (2019) 73–81

Contents lists available at ScienceDirect

Computer-Aided Design

journal homepage: www.elsevier.com/locate/cad

DE-Path: A Differential-Evolution-Based Method for Computing Energy-Minimizing Paths on Surfaces☆,☆☆

Zipeng Ye [a], Yong-Jin Liu [a,*], Jianmin Zheng [b], Kai Hormann [b,c], Ying He [b,*]

[a] MOE-Key Laboratory of Pervasive Computing, Department of Computer Science and Technology, Tsinghua University, China
[b] School of Computer Science & Engineering, Nanyang Technological University, Singapore
[c] Faculty of Informatics, Università, della Svizzera italiana, Lugano, Switzerland

ARTICLE INFO

Article history:
Received 26 April 2019
Accepted 4 May 2019

Keywords:
Energy-minimizing paths
Differential evolution
Global solver

ABSTRACT

Computing energy-minimizing paths that are general for different energy forms is a common task in science and engineering. Conventional methods adopt numerical solvers, such as conjugate gradient or quasi-Newton. While these are efficient, the results are highly sensitive with respect to the initial paths. In this paper we develop a method based on differential evolution (DE) for computing optimal solutions. We propose a simple strategy to encode paths and define path operations, such as addition and scalar multiplication, so that the discrete paths can fit into the DE framework. We demonstrate the effectiveness of our method on three applications: (1) computing discrete geodesic paths on surfaces with non-uniform density function; (2) finding a smooth path that follows a given vector field as much as possible; and (3) finding a curve on a terrain with (near-) constant slope.

1. Introduction

Computing optimal paths is a common task in many fields of science and engineering [1]. For example, in robotics one needs to compute optimal trajectories [2], and in computational biophysics and biochemistry one needs to find molecular transition paths [3]. A typical way for solving these problems is to formulate an energy functional for the paths and then adopt numerical solvers, such as conjugate gradient or quasi-Newton, to minimize it. By taking advantage of the analytical gradients, these numerical solvers are efficient and converge quickly. However, if the energy function is not convex, the solution is only locally optimal and its quality is highly sensitive to the initialization.

To overcome the limitations of existing methods, this paper develops a new method for computing energy-minimizing paths on surfaces that are general to work with different energy forms. Our main idea is to adopt differential evolution (DE), a popular evolutionary computation method that optimizes a problem by iteratively trying to improve a candidate solution [4,5]. To fit into the DE framework, we propose a simple strategy to encode paths by a density distribution and define path operations, such as addition and scalar multiplication. Our method, called *DE-Path*, maintains a population of agents from random samples of the search space and creates new candidate solutions by combining existing ones according to a simple DE formula. It then keeps the candidate solution with the least energy.

Evaluation on a toy model confirms that with increasing resolution of the discretized domain, DE-path tends to find the globally optimal solution. We demonstrate the effectiveness of our method for three interesting applications: (1) computing discrete geodesic paths on surfaces with non-uniform density function; (2) finding a smooth path that follows a given vector field as much as possible; and (3) finding a curve on a terrain with (near-) constant slope. We also discuss the potential extension of our method for 3D volumes.

1.1. Related work

In computer graphics, a well-known optimal path problem is that of computing locally shortest paths. Computing these geodesics on surfaces has been studied extensively in the last three decades. Popular methods are the wavefront propagation methods [6,7], the PDE method [8], and the graph-based method [9]. Most of these methods focus on computing geodesic distances. To compute geodesic paths, one needs to back-trace the gradient of the distance field. Liu et al. [10] formulated

☆ This paper has been recommended for acceptance by Pierre Alliez, Yong-Jin Liu & Xin Li.

☆☆ No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work. For full disclosure statements refer to https://doi.org/10.1016/j.cad.2019.05.025.

* Corresponding authors.
E-mail addresses: yezp17@mails.tsinghua.edu.cn (Z. Ye), liuyongjin@tsinghua.edu.cn (Y.-J. Liu), asjmzheng@ntu.edu.sg (J. Zheng), kai.hormann@usi.ch (K. Hormann), yhe@ntu.edu.sg (Y. He).

https://doi.org/10.1016/j.cad.2019.05.025
0010-4485/

the geodesic path problem in an optimization framework that supports arbitrary density function, anisotropic metric as well as user-specified constraints. They adopted the quasi-Newton method to solve the optimization problem. Due to the local nature of the solver, their method computes only a locally optimal solution.

Weighted shortest paths on polyhedral surfaces have also been considered in computational geometry [11,12]. A weighted shortest path is a path with a minimum cost among all possible paths on the polyhedral surface and the cost is defined to be the sum of all line segments in the path multiplied by the weights of corresponding faces. A state-of-the-art method [13] can compute an ϵ-approximation of the weighted shortest path – whose distance is not larger than the weighted shortest distance multiplied by $(1+\epsilon)$ – from a fixed point to any other query point in logarithmic time $O(\max(\log^2(1/\epsilon)/\epsilon, q))$, where $\epsilon \in (0, 1)$ and q are the user-specified approximation parameter and query time parameter, respectively.

Another closely related work is energy-minimizing splines [14], in which Hofer and Pottmann proposed an extrinsic definition of spline curves that uses the embedding space and confines the curves to surfaces using nonlinear side condition. This method works for a wide range of surface domains, including parametric surfaces, level sets, triangle meshes, and point samples of surfaces. However, since it adopts a local method to solve the variational problem, the results are often locally optimal.

2. Preliminaries

2.1. Differential evolution

Evolutionary computation is a family of metaheuristics for global optimization inspired by biological evolution. A metaheuristic is a high-level heuristic designed to find, generate, or select a heuristic that may provide a sufficiently good solution for an optimization problem [15]. Popular metaheuristics are simulated annealing [16], genetic programming [17] and differential evolution [4]. All these methods are used for multidimensional real-valued functions without the need of using the gradient of the problem being optimized. Therefore they do not require the optimization problem to be differentiable, unlike classic solvers such as gradient descent and quasi-Newton methods.

Among popular metaheuristics, differential evolution (DE) has advantages of simplicity, robustness, and better convergence rate [18]. For a special class of energy functions (i.e., real-valued second-order continuous functions), the probabilistic convergence and global optimality of DE are proven [19]. For a wide range of optimization problems which do not satisfy the above-mentioned convergence condition, DE also performs highly effectively and outperforms the other metaheuristics [4,5].

Let Ω be the solution space which is a subset of Euclidean space $\mathbb{R}^d$ and $f(X) : \Omega \to \mathbb{R}$ be a loss function where $X = (x_1, x_2, \ldots, x_d)$. DE is a heuristic search algorithm to find a solution with minimum loss. DE searches the solution space by beginning with a randomly initialized population and updating the population iteratively. There are three operations in the updating process, i.e., mutation operation, crossover operation and selection operation. They are combinations of fundamental operations in vector space. The details will be explained in Section 3.2.

Thanks to its simplicity and good performance, DE has been widely used for optimization problems with continuous variables defined in Euclidean spaces. Recent research efforts extended DE to discrete problems. For example, Liu et al. [20] developed manifold differential evolution (MDE) to compute centroidal Voronoi tessellations (CVT) on curved surfaces. Due to the combinatorial nature of the Voronoi diagram, the variables in the CVT energy are orderless. To tackle this challenge, they assigned orders to the CVT generators and proposed an agent matching operator to align two agents. Recently, Yi et al. [21] applied DE to Delaunay mesh simplification by using a 2D Cartesian grid model, in which each grid point corresponds to triangle meshes with a certain number of vertices and a simplification process is a monotonic path on the grid. They developed a DE-based method to compute the optimal path in the discrete solution space.

2.2. Fast marching method

The Fast Marching Method (FMM) [22] is a popular method for computing geodesic distances. It is easy to implement and can work on a wide range of domains, such as regular grids [22], triangle meshes [8,23], implicit surfaces [24], parametric surfaces [25], geometric images [26] and broken meshes [27]. FMM divides the set of all vertices into a visited set and an unvisited set. Vertices in the unvisited set have tentative distance values and vertices in the visited set have determined distance values. FMM uses a priority queue to sort the distances of vertices in the unvisited set and propagates the vertex with minimal distance, which is a Dijkstra-like algorithm. It visits all neighbor faces of the vertex and moves the vertex from the unvisited to the visited set. It propagates the distances from two vertices to another vertex in the same simplex face using finite difference approximations of the gradient. During the propagating process, the distance is updated to the shorter distance. FMM terminates when the unvisited set is empty. FMM runs in $O(n \log n)$ time on a triangle mesh with n vertices.

The weighted shortest path from the source vertex to the target vertex can be obtained when FMM terminates. We obtain it by computing the gradients from the distances and following the gradients from target vertex to source vertex.

3. DE-Path

Given a 2-manifold M and two points s and t on M, there exist many paths between s and t on M. Denote by $P_{s,t}$ the space of all paths from s to t on M. Given an arbitrary energy function

$$\varepsilon: P_{s,t} \to \mathbb{R}$$

which measures a quantity of the paths, the problem is to find a path with minimal energy, that is,

$$\min_{p \in P_{s,t}} \varepsilon(p) \tag{1}$$

To numerically solve the problem, we represent the 2-manifold as a mesh and the path as a polyline. The mesh is represented by $M = (V, E, F)$, where V, E, and F are the sets of vertices, edges, and faces, respectively. The polyline is represented by $\overline{p_0 p_1 \cdots p_k}$, where p_i are vertices of the polyline with $p_0 = s$ and $p_k = t$. We assume that the polyline is a single line segment in a face of M. In other words, each p_i is located on an edge of M for $i = 0, 1, \ldots, k$.

To find a stochastic global optimization solution to the problem defined in (1), we propose a differential evolution (DE) strategy. Since traditional DE only optimizes real-valued functions, the major challenge in our work is how to formulate the discrete-path-based variables in the DE framework. In other words, DE needs to define the operations of subtraction and scalar multiplication of agents, and thus requires the search space to be linear. In Section 3.1 we use the fast marching method to encode a path as a density distribution such that the corresponding search space is linear. In Section 3.2 we present the proposed DE algorithm.

Algorithm 1 DE-Path

Input: 2-manifold M, two points s and t on M, energy function $\varepsilon: P_{st} \to R_{\geq 0}$, differential weight F and crossover probability CR
Output: a path $p \in P_{st}$ with the lowest energy

Initialize all agents $G_0 = \{g_{0,1}, g_{0,2}, \ldots, g_{0,N}\}$
$k \leftarrow 0$ // the k^{th} population
while the termination criterion is not met **do**
  for $i = 0 \to N-1$ **do**
    // Mutation
    Randomly select three agents g_{k,rand_1}, g_{k,rand_2} and g_{k,rand_3} from G_k
    $v_{k,i} = g_{k,\text{rand}_1} \oplus F \otimes (g_{k,\text{rand}_2} \ominus g_{k,\text{rand}_3})$
    // Crossover
    for $x_j \in V$ **do**
      $R_j \leftarrow$ a uniformly distributed random number
      if $R_j > CR$ **then**
        $h_{k,i}(x_j) \leftarrow g_{k,i}(x_j)$
      else
        $h_{k,i}(x_j) \leftarrow v_{k,i}(x_j)$
      end if
    end for
    // Selection
    Compute the path for agent $h_{k,i}$ using FMM and evaluate its energy $\varepsilon(h_{k,i})$
    if $\varepsilon(h_{k,i}) < \varepsilon(g_{k,i})$ **then**
      $g_{k+1,i} \leftarrow h_{k,i}$
    else
      $g_{k+1,i} \leftarrow g_{k,i}$
    end if
  end for
  $k \leftarrow k+1$ // next generation
end while
// find the best agent in G_k
$g_{\text{best}} \leftarrow g_{k,0}$
for $i = 0 \to N-1$ **do**
  if $\varepsilon(g_{k,i}) < \varepsilon(g_{best})$ **then**
    $g_{\text{best}} \leftarrow g_{k,i}$
  end if
end for
$p \leftarrow \text{FMM}(g_{\text{best}})$
(optional) optimize p using a local optimal algorithm
return p

3.1. Encoding paths as density distributions

The variables in the optimization problem defined in (1) are paths on a 2-manifold mesh M. To apply a DE strategy, our key idea is to encode any path by a density distribution on M such that the variables are transformed to density distributions. We further define these density distributions as scalars on mesh vertices. Therefore, the necessary addition, subtraction, and scalar multiplication required by DE can be defined on these scalars. The details are as follows.

Recall that a density distribution on a smooth 2-manifold $\tilde{M}$ is a real value function

$$\rho: \tilde{M} \to \mathbb{R}.$$

On a 2-manifold mesh $M = (V, E, F)$, a density distribution on M can be defined as a real-value function on V, i.e.,

$$\rho: V \to \mathbb{R}.$$

Given any density distribution ρ, we apply the fast marching method (FMM) – which is a numerical method for solving the boundary value problem of the Eikonal equation – to find the weighted shortest path between s and t on M,

$$\begin{cases} |\nabla u(x)| = \rho(x), \\ u(s) = 0, \end{cases} \tag{2}$$

where s is the source point and $u(x)$ is the minimum distance from s to x in M under the density distribution ρ. Therefore, FMM can be regarded as a function which maps any density distribution ρ to a path $p \in P_{s,t}$,

$$f_{\text{FMM}}: D_M \to P_{s,t},$$

where D_M is the space of all density distributions in M. Given that D_M is a linear space, the addition, subtraction, and scalar multiplication of real-valued functions can then be easily designed for D_M naturally.

FMM requires the density distribution to be non-negative everywhere, but the difference of two density distributions could be negative. To solve this conflict, one way is to trivially set all negative values to zero. In our algorithm we use the following definitions of addition, subtraction, and scalar multiplication, which enlarges the search space more than the trivial clipping scheme,

$$\begin{cases} a \oplus b = ab, \\ a \ominus b = a/b, \\ k \otimes a = a^k. \end{cases}$$

The multiplication, division, and exponential operation defined in (3.1) can be regarded as addition, subtraction, and scalar multiplication in an exponential domain. Note that positive real numbers are closed under these operations. At the implementation level, positive real numbers could be too large or too close to zero, which is out of the range of precision. Let DBL_{MAX} and DBL_{MIN} be the maximum and minimum positive numbers that can be represented. We set a real number to DBL_{MAX} if it exceeds DBL_{MAX} and to DBL_{MIN} if it is less than DBL_{MIN}, which is called the maximum principle. It can make the difference of two big (or small) numbers disappear. We use a trick to keep the differences by storing its exponential domain $log(x)$. We apply no maximum principle to it in the DE part. As Eq. (2) needs the density, we apply the maximum principle to it only in the fast marching part.

The density with new operations is the same as the traditional density in FMM. Different operations lead to different search behaviors, which only make DE different. The new operations avoid negative numbers and keep the difference so it can make DE more effective.

By transforming each density distribution $\rho \in D_M$ to a path in $p \in P_{s,t}$ using f_{FMM}, we can represent the energy function $\varepsilon(p)$ as $\varepsilon(f_{\text{FMM}}(\rho))$, denoted by $\varepsilon(\rho)$ for short. Hereafter, we will focus on D_M with the energy defined on density distributions.

3.2. Algorithmic details

The pipeline of our algorithm follows the traditional differential evolution algorithm, as shown in Fig. 1. First, we generate a set of density distributions in V randomly as initial agents. We iteratively update the agents until the termination condition is met. We call the agents after the kth iteration as the kth population. Denote by $G_k = \{g_{k,i}\}_{i=1}^N$ the kth population, where N is the size of populations. The scale of populations is the same in different iterations, which is a parameter in the differential evolution algorithm. There are three operations in an iteration for each agent, which are mutation, crossover, and selection. The pseudo-code of our algorithm is shown in Algorithm 1.

The initial agents, which are a set of density distributions in V, are generated randomly. The path will not change if the

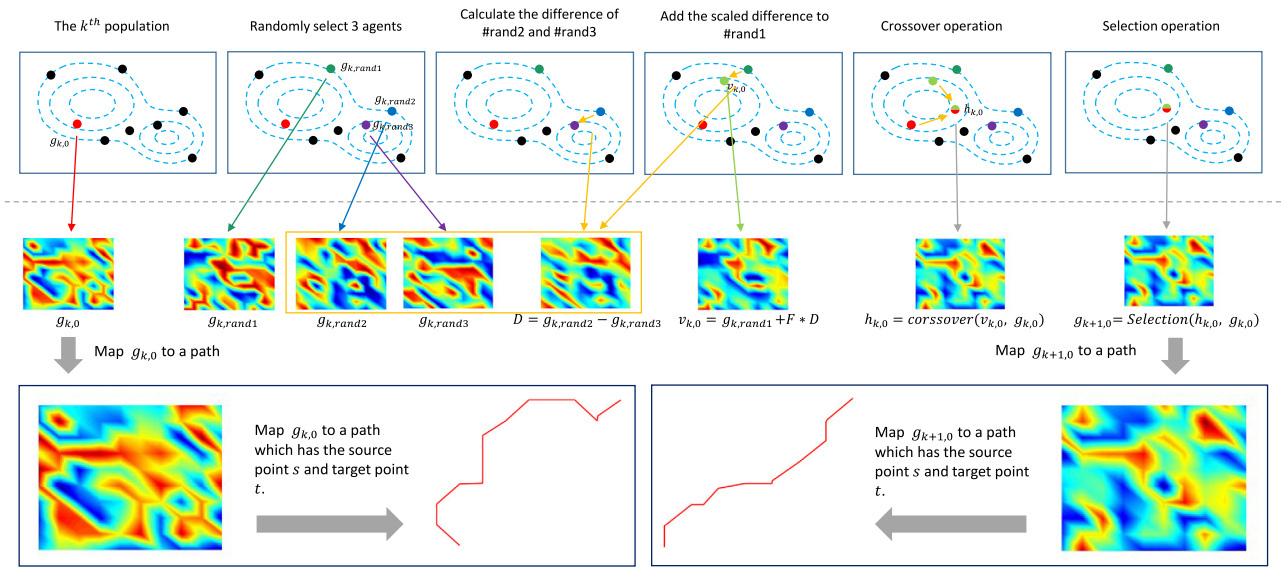


Fig. 1. Illustration of one iteration in differential evolution. The agents are a set of density distributions in V and the fast marching method maps a density distribution to a path from s to t. The kth is $G_k = \{g_{k,i}\}_{i=1}^N$. For each agent $g_{k,i}$ in G_k, mutation, crossover, and selection are applied. We select three agents from G_k randomly and add the scaled difference of #rand2 and #rand3 to #rand1 during the mutation operation. After crossover and selection, we update $g_{k,i}$ to $g_{k+1,i}$.

densities in all vertices are multiplied by some number together. For any $c > 0$, uniform samples in $(0, c]$ are the same as in $(0, 1]$. Therefore, we take a uniform sample in $(0, 1]$. Uniform sampling in the density domain leads to non-uniform sampling in the path domain. The probability of a path being the minimizing path is non-uniform in the path domain and to take a uniform sampling in the path domain is not easy, so we take a uniform sampling in the density domain.

During the mutation operation, we select three agents, denoted by g_{k,rand_1}, g_{k,rand_2} and g_{k,rand_3}, from G_k randomly. Let

$$D = g_{k,\text{rand}_2} \ominus g_{k,\text{rand}_3}$$

and

$$v_{k,i} = g_{k,\text{rand}_1} \oplus F \otimes D,$$

where F is the *differential weight* parameter.

In the crossover operation, we generate a new density distribution $h_{k,i}$ whose value is either $g_{k,i}$ or $v_{k,i}$. For each vertex x_j in V, we pick a uniformly distributed random number R_j. We have

$$h_{k,i}(x_j) = \begin{cases} g_{k,i}(x_j), & \text{if } R_j > CR \\ v_{k,i}(x_j), & \text{otherwise,} \end{cases}$$

where CR is the *crossover probability* parameter.

The selection operation computes the energy function values of $h_{k,i}$ and $g_{k,i}$ using FMM to decode a density distribution to a path. We select the better one and discard the other, that is,

$$g_{k+1,i} = \begin{cases} g_{k,i}, & \text{if } \varepsilon(g_{k,i}) < \varepsilon(h_{k,i}), \\ h_{k,i}, & \text{otherwise,} \end{cases}$$

where $\varepsilon: D_M \to R$ as mentioned before.

Applying the three operations to each agent, we update the population to the next population. The loop terminates if the iteration number exceeds the maximal iteration specified by the user or if the energy function does not change in n_c consecutive iterations, where n_c is a number specified by the user. Finally, we find the best agent in the final population and compute a path from s to t using FMM. Optionally, a local optimal algorithm could be applied. The path is the result of our method.

Table 1
Computation times of our method.

Application	Category	Vertices	Agents	Iterations	Time (s)
Vector field	Simple converge	1 681	1000	300	354.2
Vector field	Simple converge	6 561	1000	50	480.1
Vector field	Simple converge	14 641	2000	400	9526.6
Vector field	Simple Rotation	6 561	1000	300	2468.8
Vector field	Complex 1	6 561	1000	300	1493.3
Vector field	Complex 2	6 561	1000	300	1582.5
CSC ($k = 0.2$)	Cone	1 681	1000	200	542.6
CSC ($k = 0.2$)	Peaks	6 561	1000	300	2520.3
Uniform geodesic	Kitten	1 370	500	300	438.5
Uniform geodesic	Dragon	5 098	1000	200	1680.9
Non-uniform geodesic	Beethoven	10 406	1000	200	3080.8
Non-uniform geodesic	Bima	15 653	500	300	3469.5

3.3. Time complexity

DE-Path is a heuristic search algorithm, hence the computation time is highly related to the specific application and the manifold. It is hard to provide a supremum of the time complexity. We provide an upper bound of the time complexity and show the computation time for our experiments in Table 1.

The computation time depends on the mesh complexity, the population size, and the number of iterations. Given a triangle mesh with n vertices, FMM runs in $O(n \log n)$ time. The DE-Path runs in $O(I_{max} N n \log n)$ time, where N is the population size and I_{max} is the maximal number of iterations, which are both specified by the user.

4. Applications

This section demonstrates DE-Path on three applications: computing geodesic paths on triangle meshes with non-uniform density function, finding a smooth path that follows a given vector field as much as possible, and computing a curve on a terrain with constant slope.

4.1. Application 1: Geodesics

Given a manifold M and a density function $\rho: M \to \mathbb{R}$, a geodesic is a path with weighted shortest length, which is also

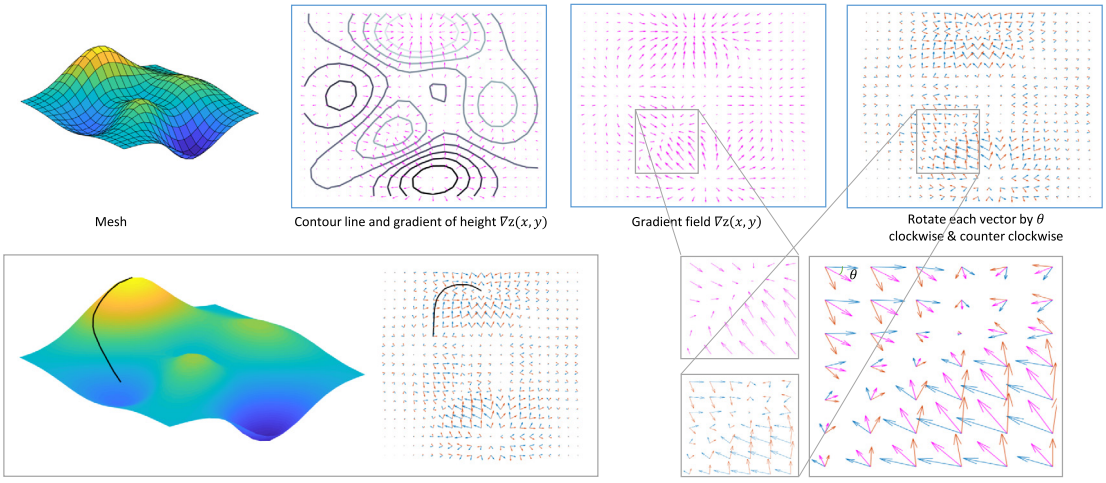


Fig. 2. Constant-slope curve. Given a terrain model and a desired slope k, we first compute the gradient of the terrain model and then rotate the vector field by θ clockwise and counterclockwise to obtain a 2-vector field. A constant slope curve follows one of the two vectors at each point.

the minimum of the energy function

$$\varepsilon_G(p) = \int_{x \in p} \rho(x)\, dx.$$

It is called a uniform geodesic if function $\rho(x)$ is constant, and a non-uniform geodesic otherwise.

Uniform geodesics are a well-studied application in computational geometry and there exist many methods to compute them. The Mitchell–Mount–Papadimitriou (MMP) algorithm [6] is a fast method to compute exact geodesics, so its result can be regarded as the ground truth. In our method, FMM can find an approximate geodesic and the DE framework can improve the quality of the geodesic. We compare the results of our method, FMM, and the MMP algorithm to validate our method in Section 5.

The classic window propagation methods, such as the MMP [6] and CH algorithms [7], do not work for triangle meshes with non-uniform density function. FMM can be extended for handling non-uniform density. However, the path computed by FMM is different from the exact solution because of the back-tracing process. In this paper, we adopt FMM in our DE framework to compute geodesic paths with non-uniform density.

4.2. Application 2: Vector fields

A vector field X on a manifold M is an assignment of a tangent vector to each point in M [28]. It can be considered as a map

$$X : M \to TM, \tag{3}$$

such that $proj \circ X$ is the identity mapping, where TM is the tangent bundle of M and $proj$ denotes the projection from TM to M. In the discrete case of a 2-manifold embedded in $\mathbb{R}^3$, the manifold is represented by a mesh $M = (V, E, F)$ and the vector field X is defined on M.

The simplest vector field is the gradient field which is the gradient direction field of a real value function defined on M. In the gradient field, curl is zero, but convergence and divergence are usually not zero. We can rotate the gradient field and obtain a non-zero-curl vector field. Using this way, we can obtain a complex vector field with non-zero curl, convergence and divergence from a real value function.

Given two points s and t on M, the problem is to find a path along the vector field from s to t, where the vector field is bidirectional. In most cases, no path can exactly follow the vector field. This is because given a starting point t, following the vector field X will lead to a path, on which usually t may not be.

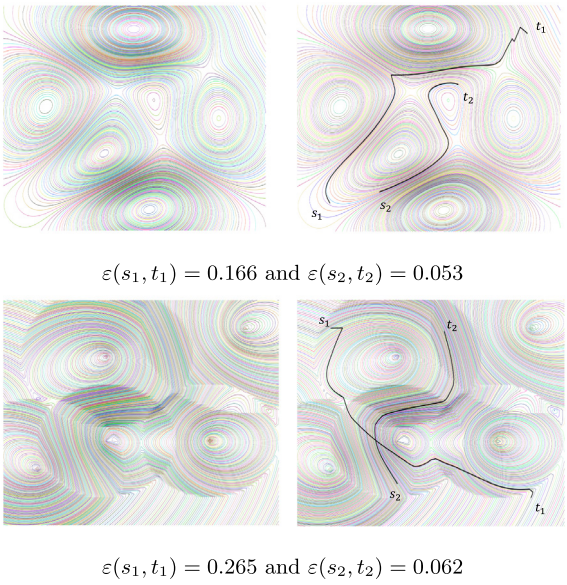


Fig. 3. Results of DE-Path on two complex vector fields with curls. We pick two pairs of endpoints on each vector field and set the smoothness factor 0.1. The endpoints s_2 and t_2 are on the *same* integral curve so that the optimal path has a zero energy. Our computed paths (s_2, t_2) for both vector fields have low normalized energies, meaning that they are close to the ground truth.

An energy function for measuring how well a path p follows the vector field X is defined as

$$\varepsilon_1(p) = \int_{x \in p} \measuredangle\langle \dot{p}(x), X(x) \rangle\, dx,$$

which is, the integral of the acute angle between the lines of $\dot{p}$ and X at each point. Here notation $\measuredangle\langle a, b \rangle$ represents the acute angle between the lines of a and b. The path along the vector field has the lowest energy. Therefore, we can solve the problem by minimizing the energy using our proposed DE solution.

4.3. Application 3: Constant-slope curves

Given a 2-manifold $S(x, y) = (x, y, z(x, y))$, a constant-slope curve (CSC) is a curve $c(t) = (x(t), y(t), z(x(t), y(t)))$ satisfying

$$\frac{dz}{dt} \Big/ \frac{dl}{dt} = k$$

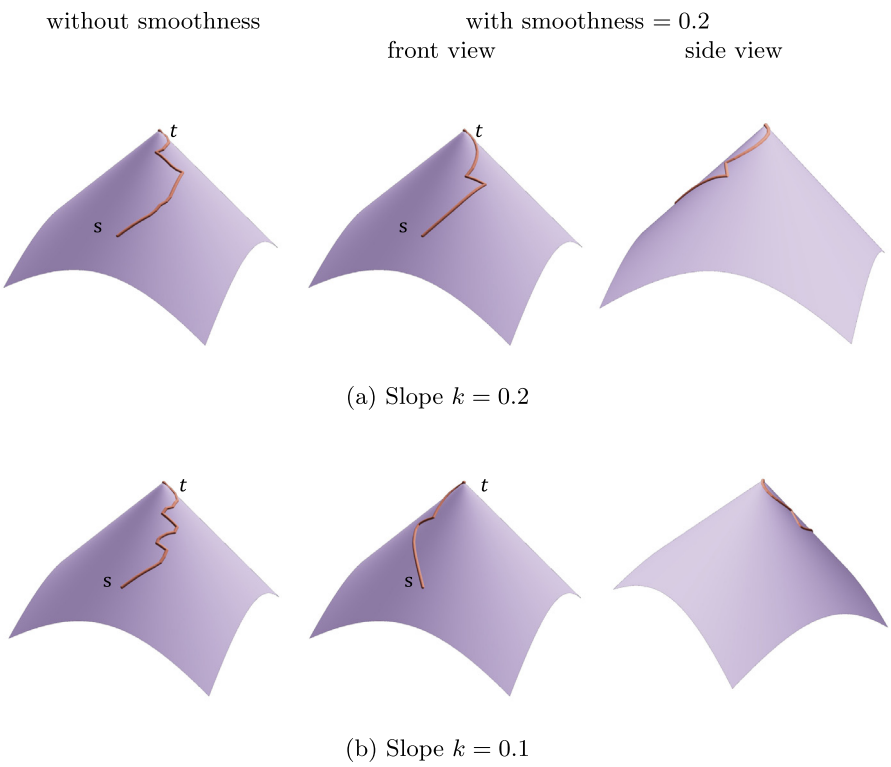
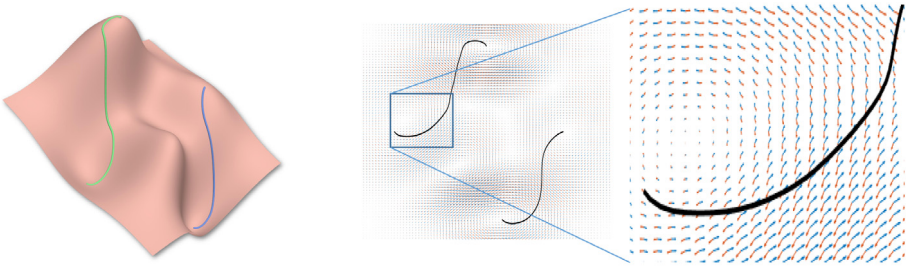


Fig. 4. Constant slope curves on a cone with (a) $k = 0.2$ and (b) $k = 0.1$. We show the results without (left) and with (middle and right) the smoothness term.



(a) constant slope curve ($k = 0.2$, smoothness $= 0.1$) on a terrain

(b) 2-vector field

Fig. 5. Two constant slope curves with $k = 0.1$. (a) The mean errors of the green and the blue curves are 0.157 and 0.227, respectively. (b) To visually check the quality, we embed the projected curve in the corresponding 2-vector field. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

everywhere, where l is length of the curve and k is a given constant. Rewriting the equation, we have

$$\frac{dz}{ds} = k,$$

where s is the arc-length parameter. Using the chain rule, we have

$$\frac{\partial z}{\partial x}\frac{\partial x}{\partial s} + \frac{\partial z}{\partial y}\frac{\partial y}{\partial s} = k.$$

The left hand side can be regarded as the inner product of two vectors, so that

$$\left\langle \left(\frac{\partial z}{\partial x}, \frac{\partial z}{\partial y}\right), \left(\frac{\partial x}{\partial s}, \frac{\partial y}{\partial s}\right) \right\rangle = k$$

and further

$$\langle \nabla z(x, y), \dot{p}(s) \rangle = k, \tag{4}$$

where $\nabla z(x, y)$ is the gradient of $z(x, y)$ and $\dot{p}(s)$ is the derivative of $p(s)$. We first compute the gradient field $GF(S)$ of $S(x, y)$ and note that there are at most two directions satisfying the condition at each point. By rotating $GF(S)$ at each point by θ clockwise and counterclockwise we obtain two vector fields, VF_l and VF_r, where θ is the angle satisfying Eq. (4) and

$$\theta = \arccos\left(\frac{k}{|\nabla z(x, y)|\,|\dot{p}(s)|}\right).$$

The constant-slope curve is then along either VF_l or VF_r. The problem can be solved using a 2-vector field model, as shown in

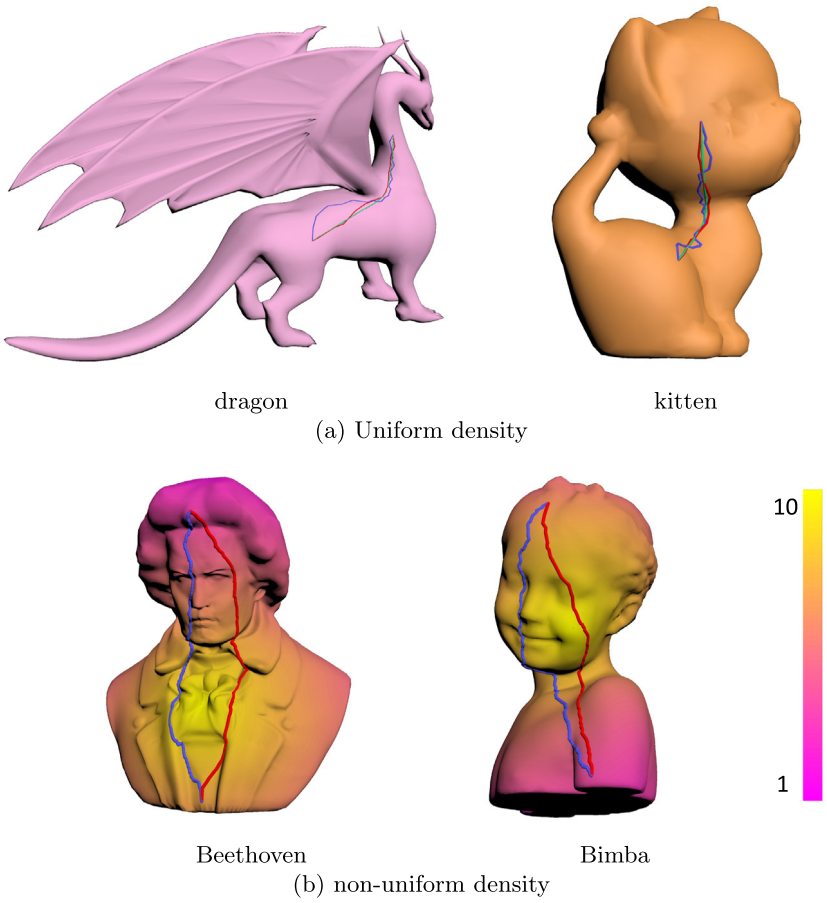


Fig. 6. Computing geodesic paths. (a) The models have uniform density. The green path is the ground truth computed by the MMP algorithm. The red is our result without local refinement, and the blue is the result of FMM. The lengths of the geodesic paths are: Dragon: 70.46 (MMP), 71.68 (ours) and 81.44 (FMM); Kitten: 49.52 (MMP), 50.11 (ours), and 57.38 (FMM). (b) The models have non-uniform densities and the MMP algorithm does not work. The energies are: Beethoven 20.47 (ours) and 22.48 (FMM); Bimba: 22.41 (ours) and 25.32 (FMM). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

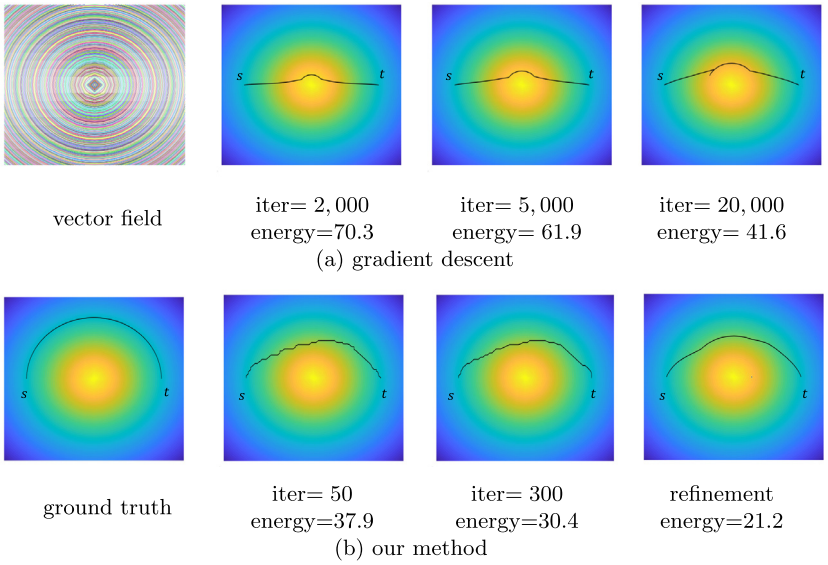


Fig. 7. Validation on a toy model. We show results of our method (without penalty term) after 300 iterations (bottom) and the gradient descent method after 20 000 iterations (top). The energy of our method is lower than that of the gradient descent method.

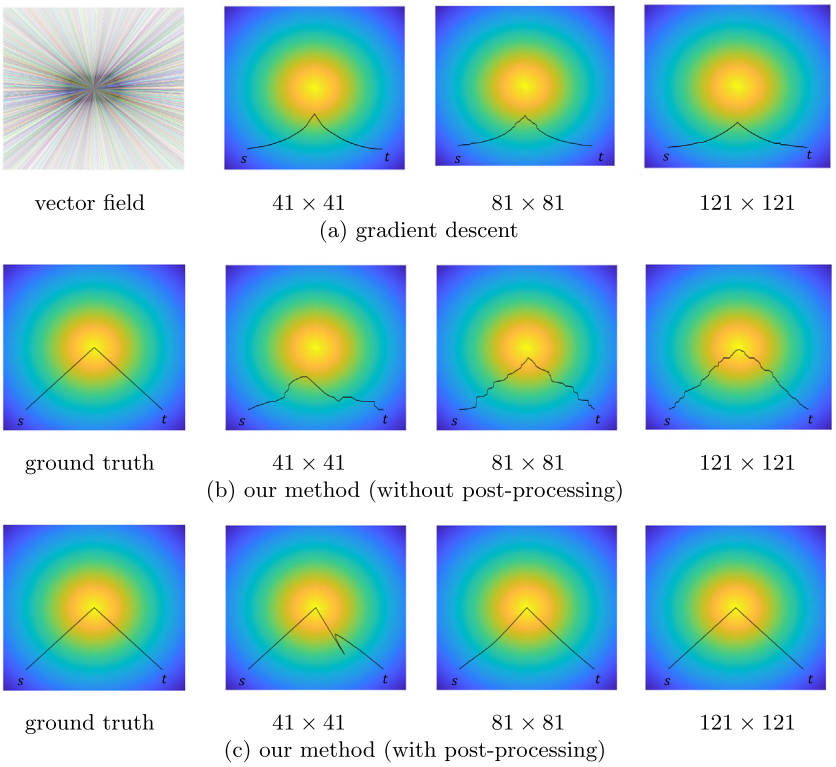


Fig. 8. Validation on a toy model. With increasing resolution of the discretized domain, DE-Path is able to find the solution with higher quality and shows the positive trend of finding the global optimal. In contrast, the classic gradient descent solver gets stuck in a local optimum.

Fig. 2. In the case $k = 0$, the problem reduces into a vector field problem, as discussed in Section 4.2.

To measure how well a path p keeps a constant slope, we define the energy function

$$\varepsilon_2(p) = \int_{x \in p} \min\big(\langle \dot{p}(x), VF_l(x)\rangle, \langle \dot{p}(x), VF_r(x)\rangle\big)\, dx.$$

4.4. Penalty term in applications 2 and 3

In some applications, we hope to find a path satisfying a certain property. A straightforward way is to add a penalty term into the energy function. With this penalty term, the energy function no longer has nice properties and some methods fail, but ours still work.

A penalty term for smoothness in the discrete case is defined as

$$\varepsilon_s(p) = \sum_{i=0}^{k-2} \langle \overline{p_i p_{i+1}}, \overline{p_{i+1} p_{i+2}} \rangle,$$

where $p = \overline{p_0 p_1 \dots p_k}$. Two possible directions at each point make a constant slope curve to bend everywhere. The penalty term can alleviate it. A comparison of a constant slope curve with and without the penalty term for smoothness is shown in Section 5.

5. Results & discussions

We implemented our method in C++ and tested it on a PC with an Intel Xeon E5-2620 CPU (2.00 GHz) and 32 GB RAM. By adopting a DE framework, our method can effectively jump out local minimums and achieve a stochastic global solution. To demonstrate this nice property, we compare our method with a local optimization method, i.e., a gradient descent method, in the same environment. The running time for all results of our method is listed in Table 1, which also reports the number of vertices, the scale of agents and the number of iterations.

To illustrate the two applications presented in Section 4, some results are shown in Figs. 3–5. Two paths following two complex vector fields are shown in Fig. 3(a) and (b). We obtain the two complex rotation vector fields by rotating two gradient vector fields of two terrains by 90 degrees. In many practical scenarios, adding a penalty term to the energy function can make the energy function quite different. To demonstrate that our method can fit diverse energy functions, four constant slope curves ($k = 0.2$ and $k = 0.1$) with and without a smoothness penalty term on a cone are shown in Fig. 4. Front view and side view are both shown because the constant slope curves could round to off side of the cone. Two constant slope curves in a complex terrain are shown in Fig. 5. We also put them in the corresponding 2-vector field mentioned in Section 4.3, where we can check whether they satisfy Eq. (4).

We validate our method on two toy models (Figs. 7 and 8), and compare with a gradient decent method. A simple rotation vector field with the ground truth and two results of our method after 300 iterations and a gradient decent method after 20 000 iterations are shown in Fig. 7. We apply a local refinement to our result after 300 iterations. The result of our method, which has lower energy, is similar to the ground truth. The gradient decent method falls into a local optimum and its result has overlaps. A simple converge vector field with the ground truth and six results in different resolutions are shown in Fig. 8. With increasing resolution of the discretized domain, DE-Path is able to find the solution with higher quality and shows the positive trend of finding the global optimal. In contrast, the classic gradient descent solver gets stuck in a local optimum.

To validate our method, we compare our method and FMM for computing geodesics with uniform and non-uniform densities (see Fig. 6). To make the comparison fair, we run our method without local refinement. The result of our method has lower

energy than that of FMM, which verifies that the DE framework finds a better density than the original density.

6. Conclusion & future work

We presented a simple yet effective method to solve a general minimal-energy-path problem on 2-manifold meshes M, in which the energy can be in any user-specific form. By proposing a novel mapping from discrete paths to density distributions on M, we successfully introduce the powerful differential evolution strategy into our solution, which can effectively jump out the local minimums and achieve a good result. Experimental results and two novel applications demonstrate the merits of the proposed DE-path solution.

Since the fast marching method can be used in 3D volumes, it is possible to extend DE-path to 3D volumes as well. A possible way is to define a density distribution on vertices of 3D volumes and use FMM to obtain a path from s to t. We will study the implementation details of this extension in future work.

Acknowledgments

This work is supported by the National Science Foundation of China (61725204, U1736220), the Royal Society-Newton Advanced Fellowship, China (NA150431) and Singapore Ministry of Education Grant (MoE 2017-T2-1-076 & RG26/17).

References

[1] Souissi O, Benatitallah R, Duvivier D, Artiba A, Belanger N, Feyzeau P. Path planning: A 2013 survey. In: Proceedings of the 2013 IEEE international conference on industrial engineering and systems management, Rabat; 2013, p. 849–56.
[2] Khatib O. Real-time obstacle avoidance for manipulators and mobile robots. Int J Robot Res 1986;5(1):90–8.
[3] Chen C, Xiao Y. Accurate free energy calculation along optimized paths. J Comput Chem 2010;31(7):1368–75.
[4] Das S, Suganthan PN. Differential evolution: A survey of the state-of-the-art. IEEE Trans Evol Comput 2011;15(1):4–31.
[5] Das S, Mullick SS, Suganthan PN. Recent advances in differential evolution — An updated survey. Swarm Evol Comput 2016;27:1–30.
[6] Mitchell JSB, Mount DM, Papadimitriou CH. The discrete geodesic problem. SIAM J Comput 1987;16(4):647–68.
[7] Chen J, Han Y. Shortest paths on a polyhedron. In: Proceedings of the sixth annual symposium on computational geometry, Berkley, CA; 1990, p. 360–9.
[8] Kimmel R, Sethian JA. Computing geodesic paths on manifolds. Proc Natl Acad Sci USA 1998;95(15):8431–5.
[9] Ying X, Wang X, He Y. Saddle vertex graph (SVG): A novel solution to the discrete geodesic problem. ACM Trans Graph 2013;32(6):12, Article 170.
[10] Liu B, Chen S, Xin S-Q, He Y, Liu Z, Zhao J. An optimization-driven approach for computing geodesic paths on triangle meshes. Comput Aided Des 2017;90:105–12.
[11] Lanthier M, Maheshwari A, Sack J-R. Approximating weighted shortest paths on polyhedral surfaces. In: Proceedings of the thirteenth annual symposium on computational geometry, Nice; 1997, p. 485–6.
[12] Aleksandrov L, Maheshwari A, Sack J-R. Determining approximate shortest paths on weighted polyhedral surfaces. J ACM 2005;52(1):25–53.
[13] Aleksandrov L, Djidjev H, Guo H, Maheshwari A, Nussbaum D, Sack J. Algorithms for approximate shortest path queries on weighted polyhedral surfaces. Discrete Comput Geom 2010;44(4):762–801.
[14] Hofer M, Pottmann H. Energy-minimizing splines in manifolds. ACM Trans Graph 2004;23(3):284–93.
[15] Bianchi L, Dorigo M, Gambardella LM, Gutjahr WJ. A survey on metaheuristics for stochastic combinatorial optimization. Nat Comput 2009;8(2):239–87.
[16] Kirkpatrick S, Gelatt Jr CD, Vecchi MP. Optimization by simulated annealing. Science 1983;220(4598):671–80.
[17] Smith SF. A learning system based on genetic adaptive algorithms [Ph.D. thesis], Computer Science Department, University of Pittsburgh; 1980.
[18] Vesterstrøm J, Thomsen R. A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems. In: Proceedings of the 2004 congress on evolutionary computation, Portland, OR; 2004, p. 1980–7.
[19] Ghosh S, Das S, Vasilakos AV, Suresh K. On convergence of differential evolution over a class of continuous functions with unique global optimum. IEEE Trans Syst Man Cybern B 2012;42(1):107–24.
[20] Liu Y-J, Xu C-X, Yi R, Fan D, He Y. Manifold differential evolution (MDE): a global optimization method for geodesic centroidal Voronoi tessellations on meshes. ACM Trans Graph 2016;35(6):10, Article 243.
[21] Yi R, Liu Y-J, He Y. Delaunay mesh simplification with differential evolution. ACM Trans Graph 2018;37(6):12, Article 263.
[22] Sethian JA. A fast marching level set method for monotonically advancing fronts. Proc Natl Acad Sci USA 1996;93(4):1591–5.
[23] Sethian JA, Vladimirsky A. Fast methods for the Eikonal and related Hamilton–Jacobi equations on unstructured meshes. Proc Natl Acad Sci USA 2000;97(11):5699–703.
[24] Mémoli F, Sapiro G. Fast computation of weighted distance functions and geodesics on implicit hyper-surfaces. J Comput Phys 2001;173(2):730–64.
[25] Spira A, Kimmel R. An efficient solution to the eikonal equation on parametric manifolds. Interfaces Free Bound 2004;6(3):315–27.
[26] Weber O, Devir YS, Bronstein AM, Bronstein MM, Kimmel R. Parallel algorithms for approximation of distance maps on parametric surfaces. ACM Trans Graph 2008;27(4):16, Article 104.
[27] Campen M, Kobbelt L. Walking on broken mesh: Defect-tolerant geodesic distances and parameterizations. Comput Graph Forum 2011;30(2):623–32.
[28] Tu LW. An introduction to manifolds. 2nd ed. New York: Springer; 2011.