Heuristically Driven Front Propagation for Fast Geodesic Extraction

Gabriel Peyré
gabriel.peyre@ ceremade.dauphine.fr
Laurent D. Cohen
cohen@ceremade.dauphine.fr

CEREMADE, UMR CNRS 7534,
Université Paris Dauphine,
Place du Maréchal De Lattre De Tassigny,
75775 Paris, France

May 7, 2007

Abstract

This paper presents a new method to quickly extract geodesic paths on images and 3D meshes. We use a heuristic to drive the front propagation procedure of the classical Fast Marching. This results in a modification of the Fast Marching algorithm that is similar to the A* algorithm used in artificial intelligence. In order to find very quickly geodesic paths between any given pair of points, two methods are proposed to devise an heuristic that restrict the front propagation. The multiresolition heuristic computes the heuristic using a propagation on a coarse map. For applications where pre-computation is acceptable, the landmark-based heuristic pre-computes distance maps to a sparse set of landmark points. We introduce various distortion metrics in order to quantify the errors introduced by the heuristically driven propagations. We show that both heuristic approaches bring a large speed-up for large scale applications that require the extraction of geodesics on images and 3D meshes.

Keywords: geodesics, Fast Marching, shortest paths, heuristic, A* algorithm.

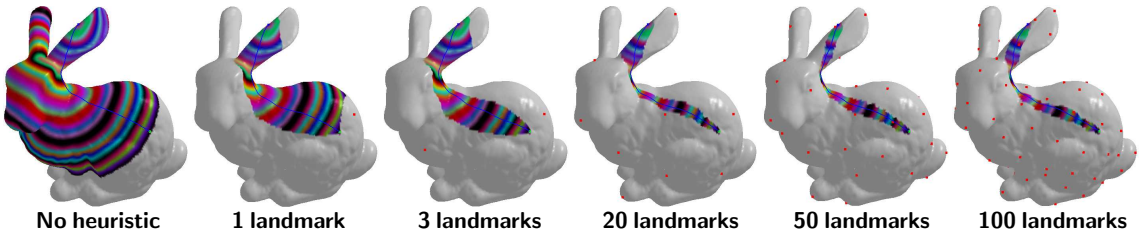

Figure 1: *In order to compute the geodesic path between two points on the surface, computation is needed on a large part of the surface when using classical fast marching (left). When the distance map to a set of landmarks is pre-computed, and the propagation is heuristically driven by these maps, only the colored region is explored, and it becomes smaller and smaller as the number of landmarks increases.*

1 Previous works

The extraction of shortest paths is a building block for a large class of applications ranging from graph theory to computer vision. The computation can be carried in a totally discrete setting (such as a graph) or over a discretized domain (such as an image or a 3D mesh).

Graphs and discrete computation. The canonical method to compute shortest paths on graphs is the Dijkstra algorithm (see for instance [5]). Fast exploration strategies have been used to speed-up the computation. The A^* algorithm [15] makes use of a heuristic to reduce the search space. Other exploration strategies have been proposed in the field of artificial intelligence, such as IDA* [13].

Images and continuous setting. In order to extract geodesics for a continuous metric, the Fast Marching algorithm [19] uses a front propagation to solve non-iteratively a finite difference approximation of the Eikonal equation. A similar algorithm was also proposed in [24]. The minimal length properties of geodesics has been applied in computer vision, for example to solve global minimization problems for deformable models [3]. The continuous nature of this method is particularly attractive for image processing, for instance to extract tubular structures and centerlines in 3D medical data [6].

Path planning: from discrete to continuous. Discrete computation on graphs gives rise to numerous schemes to perform motion planning and the A* algorithm is extensively used for path-finding in video games [21]. For the case of Euclidean metrics, faster algorithms have been devised that exploit specific data structures such as visibility graphs [18]. The Fast Marching algorithm can be used to extract paths with a non-Euclidean metric [19]. The authors of [14] compare the Fast Marching and the A* algorithms to perform motion planning, but they do not propose a heuristic modification of the Fast Marching.

2 Shortest Path: Continuous and Discrete Algorithms

In this section we develop a unifying framework which includes the Fast Marching [19], Dijkstra [5] and A^* [15] algorithms together with our new heuristically driven propagation.

2.1 Front Propagation Methods for Shortest Path

We work on a discrete set of points and for each point x we have access to its neighbors y, defining the relation $y \sim x$. These points can be embedded in discretete grid (for Fast Marching and for our method) or can be the vertices of a graph (for Dijkstra and A*). Our goal is to compute the distance function $U(x) = d(x_0, x)$ to some starting point x_0.

Initialization:
- Alive set: the starting point x_0;
- Trial set: the neighbors of x_0;
- Far: the set of all other grid points.

Loop:
- Let x be the Trial point with the smallest priority $\mathcal{P}(x)$;
- Move it from the Trial to the Alive set;
- For each neighbor y of the current point x:
  - if y is Far, then add it to Alive and compute a new value for $U(y)$,
  - if y is Alive, recompute the value $U(y)$, and update it if the new value is smaller,
  - recompute the priority $\mathcal{P}(y)$.
- If the end point $x = x_1$ is reached, stop the algorithm.

Table 1: Pseudo-code for the common framework for front propagation.

The propagation methods label the points during the computation according to:
- *Alive* is the set of points at which the distance value U has been computed and will not change;
- *Trial* is the set of next grid points to be examined and for which an estimate of U has been computed;
- *Far* is the set of all other grid points, for which there is not yet an estimate for U.

Table 1 shows the main steps of the algorithms. Each algorithm must implement the following subfunctions

- A way to update the value $U(y)$ at a given Trial point y. This computation uses the values of U at the adjacent locations. This computation depends on the metric used, which can be defined on a graph (for discrete methods) or on the whole space (for continuous methods). We explain in sections 2.2 and 2.3 specific instantiations for the Dijkstra and Fast Marching algorithms.
- A priority map $\mathcal{P}$ orders the set of Alive points according to some computational criterion. In the Fast Marching and Dijkstra algorithm, $\mathcal{P}(x) = U(x)$ is the current distance to the starting point. In our heuristical front propagation as in the A* algorithm, $\mathcal{P}(x)$ is chosen to minimize the number of visited points. We explain in sections 3 to 5 how to actually construct a priority function $\mathcal{P}$ that makes use of a heuristic.

2.2 Discrete Case: Dijkstra

In the discrete setting, a symmetric weight $g(x, y)$ is used to define the distance between two adjacent points $x \sim y$ of the graph. The length of a path $v = [v_1 \sim \ldots \sim v_m]$, of m adjacent points v_i is $L(v) = \sum g(v_i, v_{i+1})$, and we define the distance between two vertices

$$d(x_0, x_1) = \min_v \{L(v) \setminus v_1 = x_0, v_m = x_1\}.$$

The distance $U(x)$ at a vertex x in the alive set is updated during the propagation according to

$$U(x) = \min_{y \sim x} \big(U(y) + g(x, y)\big).$$

The shortest path v from x_0 to x_1 is tracked backward using

$$v_0 = x_1 \qquad \text{and} \qquad v_{i+1} = \operatorname*{argmin}_{y \sim v_i} U(y).$$

2.3 Continuous Case: Fast Marching

In $\mathbb{R}^d$, we are given a potential function $g(x) > 0$, and the weighted geodesic distance between two points $x_0,\, x_1 \in \mathbb{R}^d$, is defined as

$$d(x_0,\, x_1) = \min_\gamma \left(\int_0^1 \|\gamma'(t)\|\, g(\gamma(t)) \mathrm{d}t \right), \tag{1}$$

where γ is a piecewise regular curve with $\gamma(0) = x_0$ and $\gamma(1) = x_1$. When $g = 1$, the integral in (1) corresponds to the length of the curve γ and d is the classical Euclidean distance.

The Fast Marching method uses the fact that the function U satisfies the nonlinear *Eikonal* equation:

$$\|\nabla U(x)\| = g(x). \tag{2}$$

The distance $U(x) = u$ at a point $x = x_{i,j}$ in the trial set is updated during the propagation according to the solution of

$$\begin{aligned} &\max(u - U(x_{i-1,j}),\, u - U(x_{i+1,j}), 0)^2 + \\ &\max(u - U(x_{i,j-1}),\, u - U(x_{i,j+1}), 0)^2 = h^2 g(x_{i,j})^2. \end{aligned}$$

This is a second order equation (the equation is written in $\mathbb{R}^2$ for simplicity) and it can be solved as detailed for example in [4].

The geodesic curve γ from x_0 to x_1 can be computed by extracting the parametric curve $C(t)$ that solves the back propagation equation:

$$\frac{dC}{dt} = -\overrightarrow{\nabla U} \qquad \text{with} \qquad C(0) = x_1.$$

This gradient descent is a local computation, and it only uses the value of U for a small fraction of the visited grid points. Note that these grid points are those located in the Alive set at the end of the front propagation procedure.

3 Heuristically Driven Front Propagation

In this section we explain our algorithm in the 2D setting, and show some numerical results that illustrate the main features of this method.

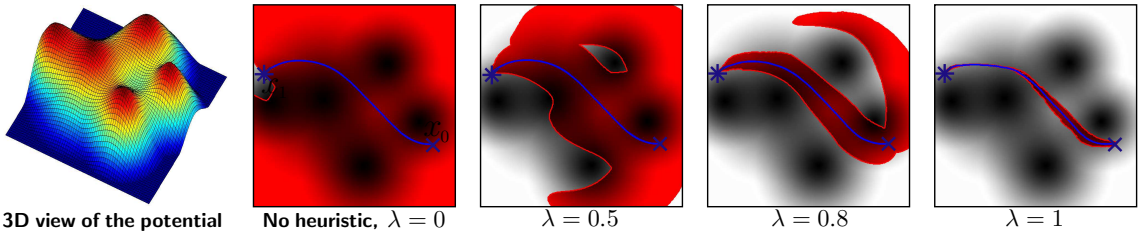

Figure 2: *An example of 2D path planning. The set of alive points according to increasing heuristic is shown in red.*

3.1 Propagation with a Heuristic

In order to minimize the number of Alive points at the end of the front propagation procedure, one should use a priority function $\mathcal{P}$ that tries to advance the front toward the goal point x_1. In order to do so, we assume that, together with the current weighted distance to the start point $U(x) = d(x_0, x)$, we have an estimate of the remaining weighted distance $V(x) \approx d(x_1, x)$. Our heuristical front propagation algorithm follows the implementation of table 1 with a priority map

$$\mathcal{P}(x) = U(x) + V(x). \tag{3}$$

The rationale behind the definition of $\mathcal{P}$ is that $d(x_0, x) + d(x_1, x)$ is minimal and constant along the geodesic path joining x_0 and x_1, see [12].

A* Algorithm. For discrete graphs, it has been shown that if the heuristic satisfies $V(x) \leqslant d(x_1, x)$, then the extracted geodesic is a path of minimum length. This leads to the A* algorithm of [15]. Various strategies have been proposed to devise admissible heuristics, see [10] and the references therein.

Heuristically driven Fast Marching In figure 2, one can see a front propagation, where we have used the oracle heuristic $V(x) = \lambda\, d(x_1, x)$, with a parameter $\lambda \in [0, 1]$. The value $\lambda = 0$ corresponds to the classical Fast Marching propagation, which results in a large region of Alive points (colored in red). However, as we increase the value of λ toward 1, the explored region shrinks around the geodesic path that links x_0 to x_1.

There are however two important issues with this ordering of the Trial set:

- This ordering can break the monotone condition that is required by the Fast Marching algorithm to produce a valid approximation of the continuous underlying distance function. We show in the numerical results presented in sub-section 5.2 that the geometric error on the extracted geodesic remains low.
- We do not have immediate access to the remaining distance $d(x, x_1)$, since it would involve performing another front propagation from x_1. We explain in sections 4 and 5 two methods to overcome this problem.

3.2 Evaluation of a Heuristic

We cast the problem of finding a good heuristic into the problem of approximating the distance function $d(x, y)$ between two points (x, y) by some function $\tilde{d}$. We define the heuristic using

$$V(x) = \tilde{d}(x_1, x) \approx d(x_1, x).$$

This approximated distance $\tilde{d}$ should satisfy $\tilde{d} \leqslant d$ in order not to perturb the propagation from the true shortest path (computed with $V = 0$). It must be fast to evaluate and can use a reasonable amount of pre-computed data.

Section 4 uses a propagation on a coarse map in order to compute such an approximate $\tilde{d}$. This multiresolution heuristic is fast to compute and does not require the storage of additional data. Section 5 uses pre-computed distances to landmark points to compute $\tilde{d}$. This landmark-based heuristic requires additional data but leads to highly accurate estimation of the real distance d while enforcing $\tilde{d} \leqslant d$.

To evaluate the quality of the heuristic, we propose two metrics

- *The approximation metric*

$$E_1(d, \tilde{d}) = \iint |d(x, y) - \tilde{d}(x, y)|^2 \mathrm{d}x\, \mathrm{d}y,$$

is approximated during evaluation using a finite number of precomputed distance maps $d(p_i, x)$ to a set of points $\{p_i\}_{i=1}^m$ chosen at random

$$E_1(d, \tilde{d}) = \frac{1}{m} \sum_{i=1}^{m} \int |d(p_i, y) - \tilde{d}(p_i, y)|^2 \mathrm{d}y.$$

- *The computation gain metric:* We measure the usefulness of the heuristic for extracting geodesics between a set of pairs of points $\{(x_0^i, x_1^i)\}_i$. For each front propagation from x_0^i to x_1^i, we measure the area $\mathcal{A}_i(\tilde{d})$ of the Alive set at the end of the propagation. We also measure the area $\mathcal{A}_i(0)$ of the Alive set for a propagation without the heuristic. We define the computation gain metric to be

$$E_2(d, \tilde{d}) = \frac{1}{m} \sum_{i=1}^{m} \mathcal{A}_i(\tilde{d}) / \mathcal{A}_i(0).$$

When the geodesics queries are random, these two metrics tend to be the same. However, the computation-gain metric is able to better adapt to typical applications where queries can be highly non-uniform (for instance in road or tubular structure extraction).

4 Multiresolution Heuristic

4.1 Coarse-to-fine Geodesic Computations

In order to compute the remaining distance $V(x) \approx d(x, x_1)$ with a fast algorithm, we perform a Fast Marching front propagation starting from the point x_1, but on a coarser grid. We thus have introduced a second parameter for our heuristical front propagation: the resolution $R \in (0, 1)$ we use for the coarse grid. If the original potential map g is of size $n \times n$, the query of $\mathcal{P}(y)$ thus requires:

- The pre-computation of a coarse potential map g_R of size $(Rn) \times (Rn)$. This is done by first pre-filtering g (to avoid aliasing of high frequencies) and then applying a cubic spline re-interpolation on a coarser grid.
- The pre-computation of the approximate distance map V of size $(Rn) \times (Rn)$ is done by performing a full Fast Marching on a coarse grid, using potential g_R, and starting from point x_1.
- During the heuristical front propagation starting from point x_0, when $\mathcal{P}(y)$ is queried, the value of V is interpolated with cubic splines on the coarse grid to retrieve a value on the original grid.

In figure 3 one can see the coarse map g_R, and high frequency details such as small roads disappear as the resolution gets too low.

There is clearly a tradeoff between choosing a low R to reduce the computation time, and a high R so that $V(x)$ approximates $d(x, x_1)$ well.

Figure 3: *Resulting potential map g_R for various values of R.*

The new algorithm we propose allows us to use multiresolution computation for the extraction of geodesic curves. Using a multiresolution framework for solving the point-to-point geodesic problem is not so easy because it is a boundary problem, and for instance, multigrid methods are not suitable. Adaptive mesh [7] and multigrid methods [16] have been used in conjunction with geodesic active contours for segmentation.

4.2 Numerical Validation of Multiresolution Heuristic

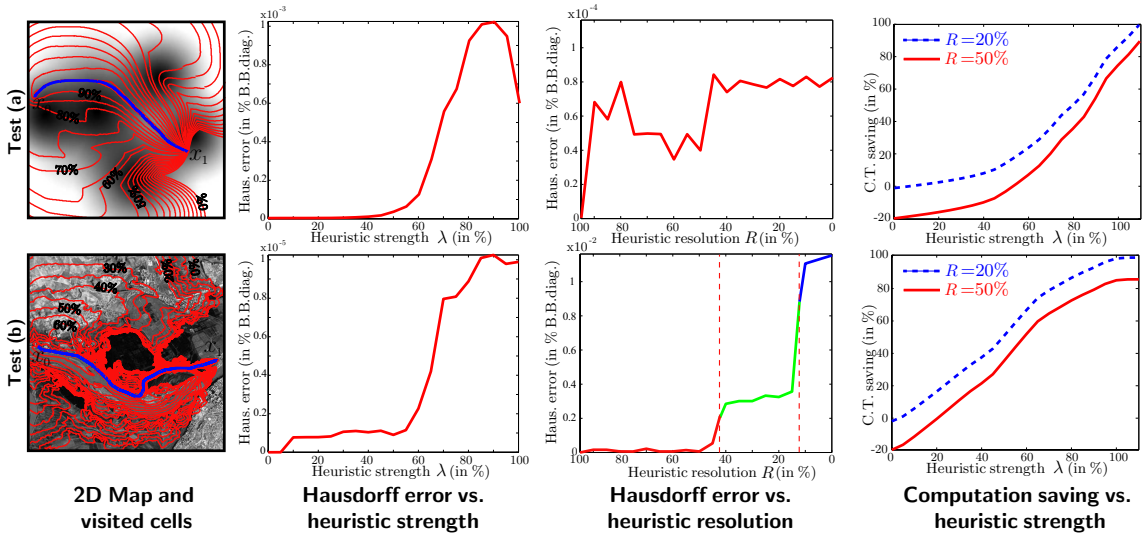

Figure 4: *Influence of heuristic strength and resolution on number of visited cells, Hausdorff error and computation time reduction.*

Figure 5: *Influence of the resolution of the heuristic on the shape of the geodesic.*

In order to estimate the precision of the results, we use the Hausdorff error between the paths obtained by fast marching with and without the multiresolution heuristic. In figure 6 one can see the geodesics extracted for different values of λ. Figure 4 shows the result of our algorithm for various settings on (a) a synthetic map and (b) a satellite image. We have depicted:

- *The 2D map*: The red curves indicate the boundary of the visited region. One can see that these curves shrink toward the geodesic (central blue curve) as one increase the strength of the heuristic from 0% to 100%.
- *Hausdorff error vs. heuristic strength* λ: We have set the heuristic resolution R to 50%. One can see that the error is higher for the synthetic map (a). This is due to the fact that this map contains large flat areas, where a small error in the computed geodesic distance leads to deviation of the extracted geodesic. In contrast, the geodesic in the satellite image (b) contains very anisotropic areas, which stabilize the extracted geodesic.
- *Hausdorff error vs. heuristic resolution* R: We have set the heuristic strength λ to 50%. One can see that the synthetic map (a) is nearly insensitive to the resolution of the coarse map used to compute the heuristic. This is because the underlying function is very smooth, so one can reduce the resolution a lot without too much impact on the accuracy of the heuristic. In contrast, one can see that the satellite image suffers from excessive variation when the resolution parameter R becomes smaller than 60% and then again for 90%. This is due to strong topological changes in the path, as depicted in figure 5.

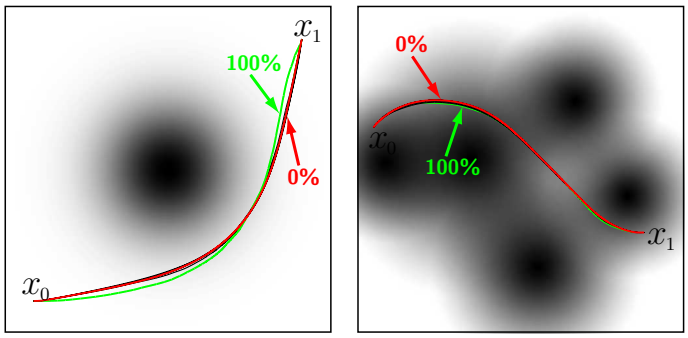

Figure 6: *Graphical display of extracted geodesics for various heuristic strengths.*

- *Computational time saving vs. heuristic strength* λ: The saving is computed relative to the time spent by classical Fast Marching. In 2D the computation times decrease roughly linearly with the strength of the heuristic.
- *Computational time saving vs. heuristic resolution* R *(not shown)*: There is a constant overhead due to the coarse resolution computation (which results in an offset between the curves for $R = 50\%$ and $R = 20\%$). For $R = 20\%$, this overhead is balanced by the heuristic saving as soon as $\lambda \geqslant 5\%$.

These tests clearly show that our algorithm can bring a large computational speed-up, but the parameters should be finely tuned to adapt to the characteristic of each map. For instance, these experiments show that the user must have some prior knowledge about the typical width of the tubular structures he wants to extract, and set the resolution R so that the coarse map V_R still contains these structures.

4.3 Applications of a Multiresolution Heuristic

Volumetric Geodesic Extraction 3D geodesic extraction is very useful in medical volumetric data analysis. It can be applied to perform tubular structure extraction, and it is extended to virtual endoscopy in [6]. In figure 7 one can see the extraction of 3D geodesics from synthetic data (top and middle rows) and from real medical data (bottom row) for $R = 20\%$. The red surface shows the boundary of the explored regions of alive cells. The computational time gain (Comp. gain) is also indicated.

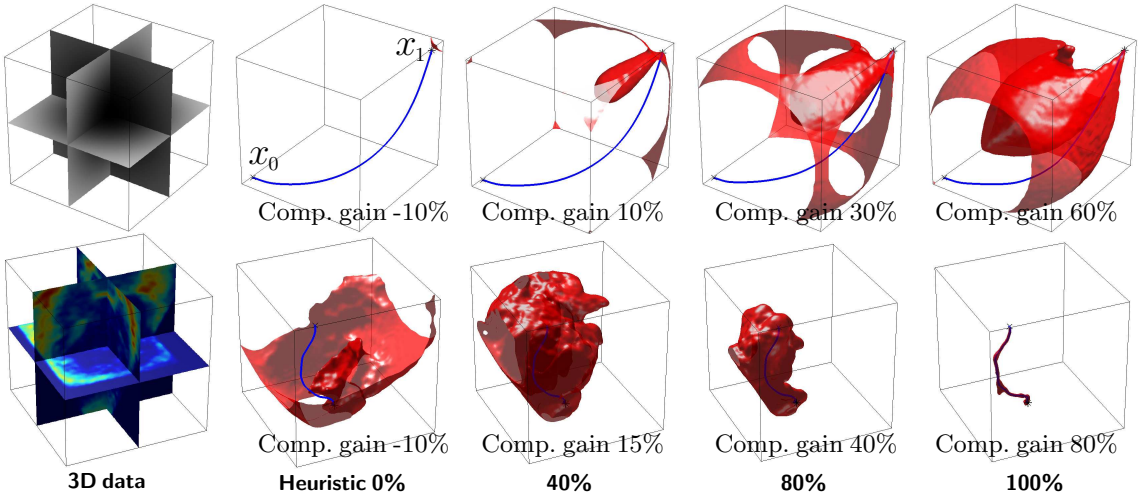

Figure 7: *Extraction of geodesics in 3D.*

Globally Optimal Geodesic Active Contours The concept of circular geodesics was first introduced in [22]. The authors of [1] proposed a simple way to compute circular geodesics around a point, in order to compute a globally optimal geodesic, with an application to object segmentation. The user simply select a point C inside the object to segment (see figure 8), and then the algorithm virtually "cuts" the image along a horizontal line that links C to the boundary of the image. This way, one can force a geodesic path to go

around C by running a classical Fast Marching from a point S to itself, but forbidding the front to pass through the segment CD.

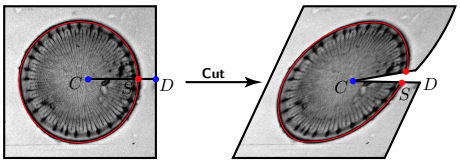

Figure 8: *Cutting the square domain to compute circular paths.*

For an underlying image I, the globally optimal geodesic around C is defined as the closed geodesic curve with minimum length, where the metric is defined as

$$g(x) = \frac{1}{\|C - x\|} \frac{1}{1 + \|\nabla I(x)\|^2} + \varepsilon, \tag{4}$$

where $\|C - x\|$ is the distance from the curve point x to the center C.

The authors of [1] proposed a powerful algorithm based on the branch-and-bound paradigm, which is a binary search that avoids computing the closed geodesic for each point S on the segment CD. However, with our heuristic front propagation, we have tested a simpler algorithm that works well in practice. We simply compute the circular geodesics that pass though a given fixed number of points along the cut segment CD. These extractions can be performed quickly using our heuristically driven front propagation, with the restriction that the front should not pass though the cut segment.

In figure 9, we have shown a globally optimal circular geodesic, computed with various heuristic strengths λ.

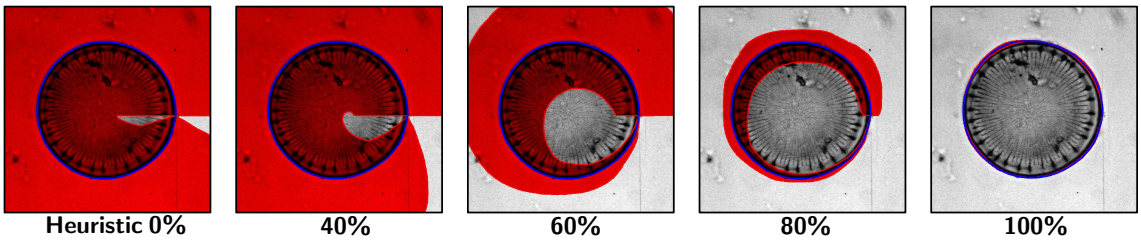

Figure 9: *Globally optimal circular path extraction with increasing heuristic.*

Geodesic Extraction on 3D Meshes The Fast Marching algorithm has been extended to 3D meshes in [20]. Our heuristic algorithm also extends to 3D meshes, with the following modifications with respect to the Euclidean setting:

- We must construct a coarse mesh approximation of the original 3D mesh. Mesh simplification is a large topic, and several greedy methods exist, see for example [11]. In our tests, we use the farthest point strategy proposed in [17] for remeshing, since it uses Fast Marching as a building block.
- Once the heuristic function has been computed on the coarse mesh, it must be interpolated on the original dense mesh. Several methods for data interpolation on 3D meshes exist, and we have used a method derived from harmonic mesh parameterization [8]. This involves the solution of a sparse linear system that describes a harmonic function that fits the values computed on the coarse mesh.

These two steps are quite computationally intensive, but note that:

- The coarse mesh can be pre-computed, and can be re-used for multiple geodesic extraction.
- To avoid the computational overhead of computing the interpolation on the whole mesh, we use the local parameterization strategy of [23]. We compute the interpolation only on a small set of overlapping disk-like charts that cover the region of alive vertices.

In figure 1 and 10, one can see the algorithm in action on various meshes, and for various values of the parameter λ.

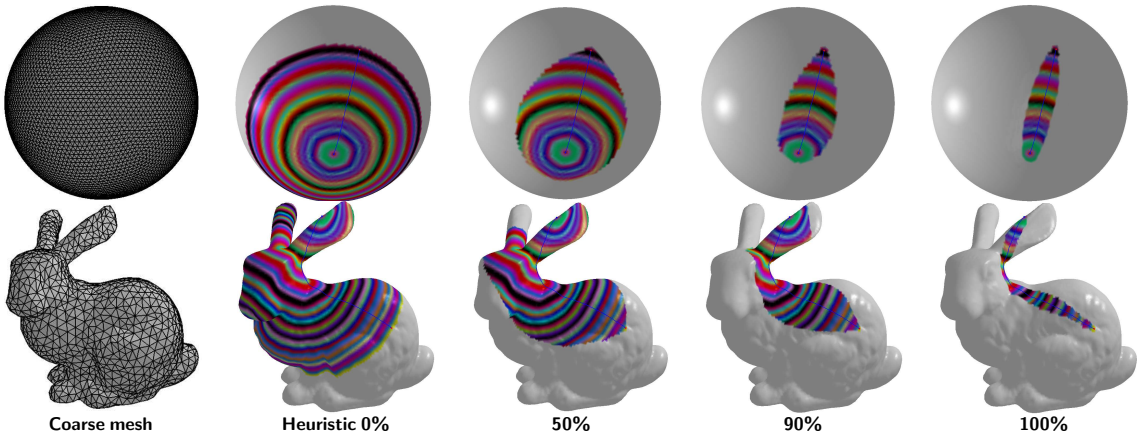

Figure 10: *Heuristically driven front propagation on 3D meshes with a multiresolution heuristic.*

5 Landmark-based Front Propagation

In this section, we describe an alternate way to implement the heuristic. This heuristic is computed using an approximated distance $\tilde{d}$ evaluated with a set of pre-computed distances to landmark points.

5.1 Landmark-based heuristic

A new method for distance evaluation on a graph has been introduced in [10] as an admissible heuristic for the A* algorithm. We explain why this method can be useful for our heuristically driven Fast Marching and give a quantitative numerical study.

This method exploits the triangle inequality, since we have, for every pair of points x and y,

$$d(x,y) = \sup_z \Big(|d(x,z) - d(z,y)| \Big).$$

This equality is still valid in the continuous framework of Fast Marching, and in order to derive a useful heuristic, one can chose a small set $z_1, \ldots, z_n$ of Landmark points. The set of distance maps $d_k(x) = d(z_k, x)$ is pre-computed, and we define the approximation

$$\tilde{d}_{z_1 \ldots z_n}(x,y) = \sup_{k=1\ldots n} \Big(|d_k(x) - d_k(y)| \Big). \tag{5}$$

In the following, we drop the $z_1, \ldots, z_n$ dependencies and call the approximated distance $\tilde{d}$. We note that this approximation always satisfies the condition $\tilde{d} \leqslant d$.

Figure 11 gives an intuitive explanation of the efficiency of this approximation. In the ideal case, the geodesic γ_k joining a landmark z_k to x also passes through y. In this case, we have $d(x,y) = \tilde{d}(x,y)$ and there is no approximation. But in most cases, this is not true, but a geodesic γ_k passes close to y and $\tilde{d}$ is indeed a good approximation of the real distance.

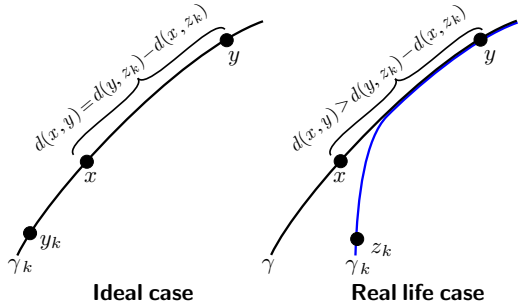

Figure 11: *Justification of the approximation properties of $\tilde{d}$.*

5.2 Landmark-based Path Extraction

This landmark-based propagation can be used to speed-up the computation of most path planing applications, including path finding in video games or robotic path planing.

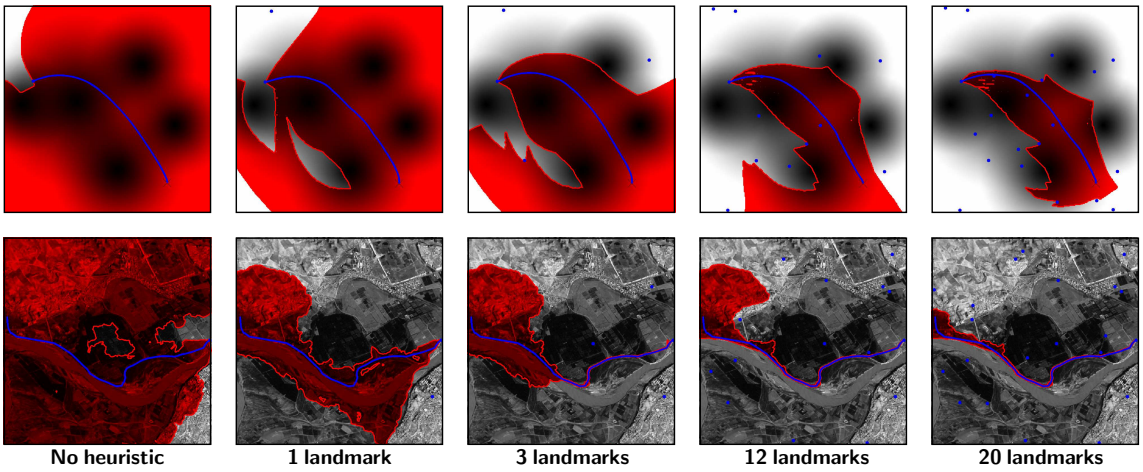

Figure 12: *Example of propagation using a landmark-based heuristic with random seeding of base points.*

In figure 12, one can see the active region explored by the propagation algorithm for an increasing number of landmark points. These points are chosen at random on the 2D image. The explored region progressively shrinks toward the central extracted curve when we add more points, since the heuristic is becoming more accurate.

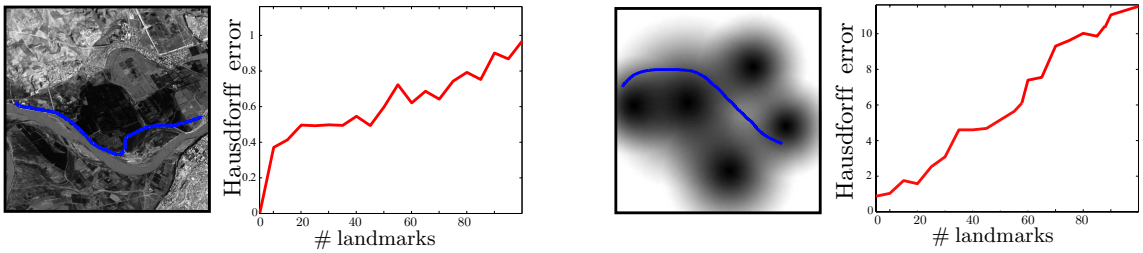

Figure 13: *Distortion caused by our heuristic propagation on the extracted geodesics. The error is reported using the Hausdorff distance expressed in pixels (the size of the image is* 256×256*).*

In figure 13, one can see a numerical evaluation of the precision of the extracted geodesic. It reports the distortion $\|\gamma - \tilde{\gamma}\|_{\mathcal{H}}$ between the original geodesic γ (computed without the heuristic) and $\tilde{\gamma}$ (computed with the heuristic). We use the symmetric mean-square Hausdorff metric

$$\|\gamma - \tilde{\gamma}\|_{\mathcal{H}}^2 = \frac{1}{2}\Big(\int_{\gamma} \min_{y \in \tilde{\gamma}} \|x - y\|_2^2 \mathrm{d}x + \int_{\tilde{\gamma}} \min_{x \in \gamma} \|x - y\|_2^2 \mathrm{d}x \Big),$$

since this captures the geometric distortion caused by our partial propagation.

On 2D maps containing important curvilinear features (such as the road in the example on the left), the distortion caused by the heuristic is nearly unnoticeable. On the contrary, for relatively flat maps (such as the one depicted on the right) the distortion can be relatively high (about few pixels for a map of 256×256 pixels). This is because the salient features of this map catch the geodesic and avoid large lateral moves when the front propagation is modified.

In order to improve the quality of the heuristic and thus the resulting speed-up, one has to carefully seed the landmarks across the image. In the next section we set up a complete framework for the evaluation of this sampling, together with strategies to get a high-quality seeding of the base points.

5.3 Seeding strategies for the landmarks

The quality of a sampling can be measured using the previously introduced metrics

$$E_i(z_1, \dots, z_n) = E_i(d, \tilde{d}_{z_1 \dots z_n}) \quad \text{for} \quad i = 1, 2.$$

In order to find sampling locations $\{z_1, \dots, z_n\}$ for the landmarks, we use various strategies, among which:

(a) A manual sampling that exploits specific knowledge of the 2D map g. This semi-automatic method is not studied in this paper.

(b) A random sampling in the image.

(c) A uniform sampling according to the distance d. This can be accomplished using the farthest point sampling procedure proposed in [17], where one chooses

$$z_{n+1} = \operatorname*{argmax}_z (\operatorname*{argmin}_{1 \leqslant k \leqslant n} d(z_k, z)).$$

(d) A uniform seeding on the boundary of the domain.

(e) A greedy strategy minimizing the E_1 metric. We define the locations recursively

$$z_{n+1} = \operatorname*{argmin}_{z \in R} E_1(z_1, \dots, z_n, z),$$

where R is some small set of random candidates.

(f) Same as (e) but using E_2 instead of E_1.

Distance approximation evaluation. A good seeding strategy should be able to put a landmark before any couple of points in the domain, near the geodesic path connecting these points. For a constant (i.e. Euclidean) metric, (d) is thus the optimal sampling scheme. However, for a complex metric, this is no more the case.

In figure 14, one can see the approximated distance $\tilde{d}(x_1, x)$ to a target point x_1, computed for $n = 32$ landmarks using various seeding strategies.

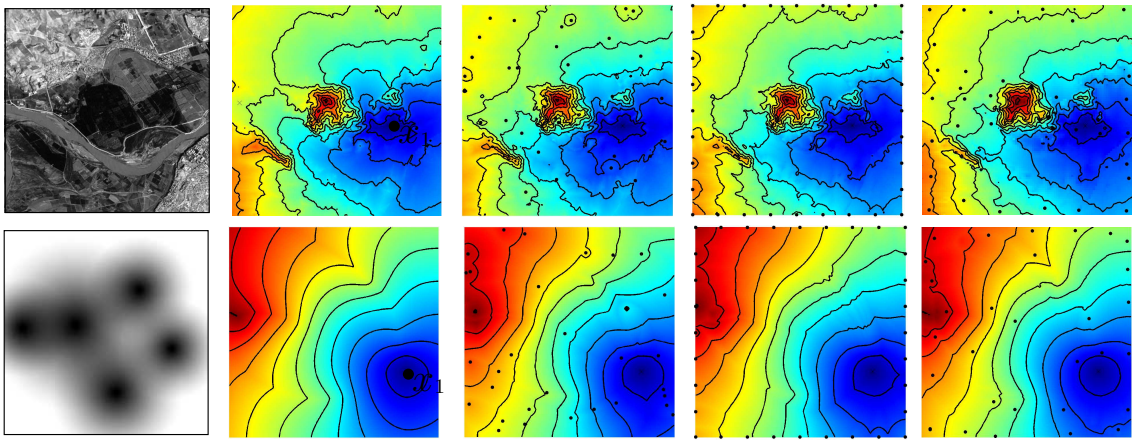

Figure 14: *Graphical display of the approximated distance $\tilde{d}(x_1, x)$ for various seeding strategies.*

In figure 15, one can see a plot of the approximation error $E_1(x_1, \dots, x_n)$. The error-driven seeding strategy (e) clearly performs the best as expected by its definition. For a smooth potential g (left) the seeding strategy on the boundary (d) performs well, but it fails to capture the topology of complex 2D maps (right).

Computational saving evaluation. In order to evaluate our algorithm in a real setting, we have set-up a complete framework using 500 typical queries $\{(x_0^i, x_1^i)\}_i$ for various 2D maps. Some resulting shortest paths queries are shown on the left of figure 16, they are relevant for many standard applications such as path finding in video games or robot path planing.

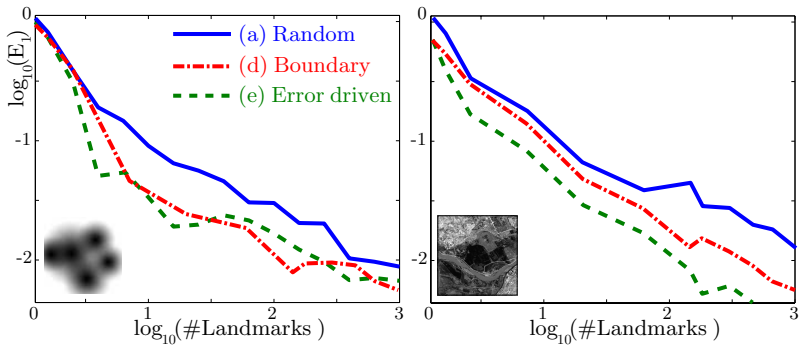

Figure 15: *Decreasing of the approximation error $E_1(z_1, \ldots, z_n)$ for various seeding strategies.*

The right of figure 16 shows how the computational metric $E_2(x_1, \ldots, x_n)$ decreases for several sampling strategies. For the greedy sampling strategy (f), we have used another set of 100 typical queries, in order not to bias the computation of the error E_2.

One can see that the computation-driven sampling strategy (f) clearly outperforms the other strategies. In particular, the approximation-based strategy (e) does not give good results, mainly because typical paths are not correlated to areas where the approximation error $\|d - \tilde{d}\|$ is large.

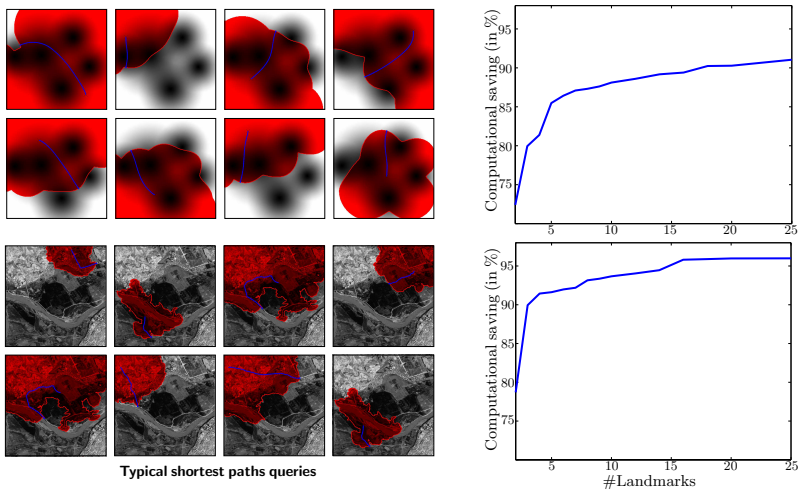

Figure 16: *Computation speed-up $E_2(z_1, \ldots, z_n)$ obtained with a various number of landmarks. The areas colored in red correspond to the explored region without heuristic.*

5.4 Reducing Memory Usage

Cells Representation Classical methods, such as using an octree data structure, can be used to reduce the memory usage of level set algorithms, for example in order to perform image segmentation [7].

We choose to implement a simple data structure to reduce the memory usage by allocating the grid cell on the fly during the propagation. A typical cell data structure, for 2D computation, is:

```
struct cell {
  // current geodesic distance
  double distance;
  // either alive, trial or far
  char state;
  // pointers to the 4 neighbors
  cell* neighbors[4]; };
```

To be able to retrieve a given cell in constant time, we also store the list of allocated cells in a hash table.

This is important because when a new cell is allocated, we need to connect it to the existing cells. Figure 17 shows a graphical display of the overall data structures.

This pointer-based representation of the neighboring relation is convenient to extract the geodesic with a gradient descent. There is some memory overhead due to the explicit storage of pointers to neighbors, but the fact that our scheme explores significantly fewer cells than classical Fast Marching allows to save much more memory, as shown in the next section. The computing time overhead due to the use of a hash table is about 40% in all our tests.

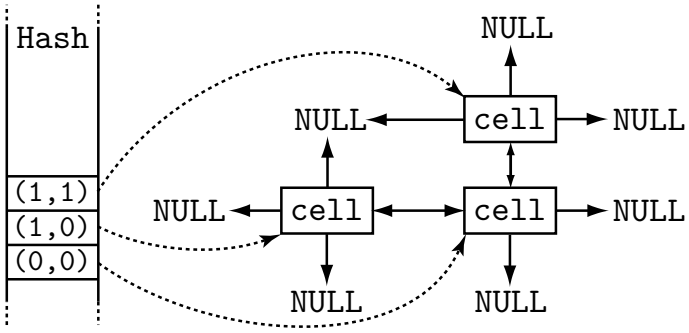

Figure 17: *Data structures used for the propagation.*

Computation from Compressed Data To reduce the storage requirement of the distances to landmark $d_k(x)$, we implement a compression procedure. A typical implementation should

- Allow random access of the value $d_k(x)$, without decompressing the whole data.
- Be asymetric, since we do not care about the compression time, but we need fast access to distance values.
- Give low decompression error, since we need an accurate heuristic and we need to satisfy the condition $\tilde{d} \leqslant d$ as much as possible.

We use a two-step procedure:

- Differential representation: $A_1(x) = d(x_1, x)$ and

$$A_{k+1}(x) = d(x_{k+1}, x) - d_{x_1 \ldots x_k}(x_{k+1}, x),$$

  where we use the approximated distance $\tilde{d}_{x_1 \ldots x_k}$ to give an estimate of $d(x_{k+1}, x)$ using the first available k landmarks. This new representation removes the redundancy that exists between the different distance maps.
- We code each map A_i using a vector quantization scheme [9]. In our tests, we use codeblocks of size 3×3, and we quantize the resulting vectors of $\mathbb{R}^9$ using a codebook of size 256. This results in a memory gain of $36 : 1$ with respect to storing single precision floats.

The constraints we have on the computation from compressed data are similar to the process of rendering from compressed textures [2]. We use the differential coding strategy to cope with the particular redundancy of distances functions. The compression produces neither noticeable artifacts nor distortion of the extracted geodesics. The computational time overhead due to decompression is about 15% in CPU time.

5.5 Applications of Landmark-based Heuristics

Our landmark-based propagation can be applied verbatim to higher dimensional propagation and to triangulated surfaces in order to speed-up shortest path queries.

Constrained Path Planning Geodesics can be used to compute the path of a robot with various shape and motion constraints [19]. Basically, each additional degree of freedom adds a new dimension to the domain in which the front propagation should be performed. Solving such high dimensional problems is time and memory consuming, so the use of a heuristic is highly desirable. In our experiment, the most important issue is the memory used by the full-grid classical Fast Marching, and the memory management strategy exposed in subsection 5.4 is crucial to scale to complex problems.

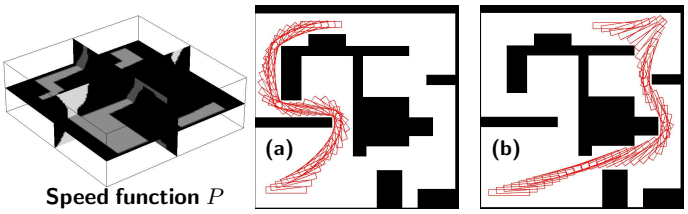

Figure 18: *Examples of constrained path planning.*

In figure 18, one can see two examples of path extractions in 2D with one rotational additional degree of freedom. This results in 3D front propagation, and the corresponding speed function is depicted on the left. Figure 19 shows the influence of the heuristic strength λ on the cells explored by the front propagation.

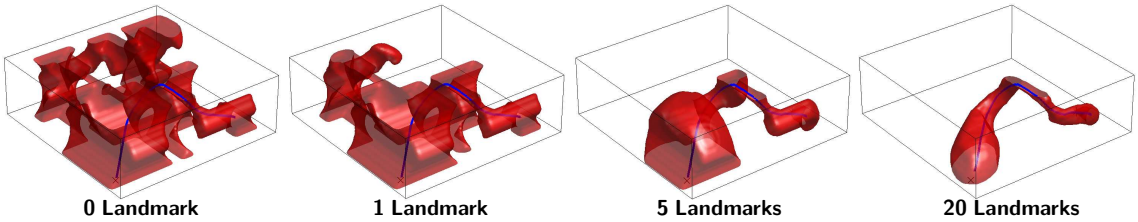

Figure 19: *Explored area for constrained path planning.*

Geodesic Extraction on 3D Meshes The Fast Marching algorithm has been extended to 3D meshes in [20]. Our heuristically driven front propagation extends to 3D surfaces in a straightforward manner. We use a constant metric $g = 1$ and as the surfaces considered do not have boundaries, the uniform seeding strategy (c) gives the best results for the E_1 metric. In figure 20, one can see the approximated distance $\tilde{d}(x_1, x)$ for various numbers of landmarks. In figure 1, one can see the front for various numbers of landmark points.

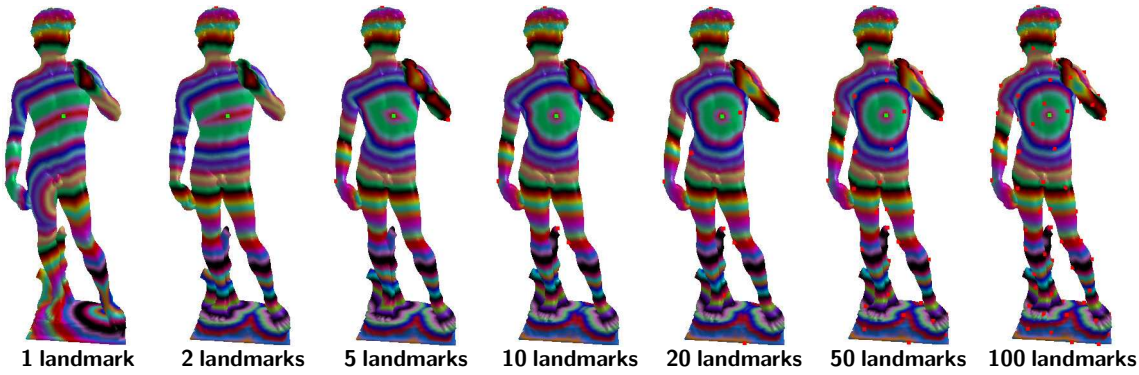

Figure 20: *Approximation of the distance function $d(x_1, x)$ using $\tilde{d}(x_1, x)$ on a 3D mesh, with an increasing number of landmarks.*

6 Conclusion

In this paper we have presented a simple modification of Fast Marching to speed-up the extraction of geodesics on images, higher dimensional data and triangulated surfaces. This modification uses a heuristic that drives the propagation and reduces the number of visited cells. This heuristic takes into account the remaining distance to the end point and two approaches are proposed to estimate this geodesic distance. The first approach performs a backward propagation on a coarse grid and uses an interpolation scheme to retrieve the remaining distance on the full resolution grid. This method provides an important speed-up and

numerical evidences show its efficiently as long as the coarse grid can resolve fine scale details of the metric. The other approach uses pre-computed distances to a set of landmark points that are combined during the propagation according to the triangle inequality. This method provides estimations with arbitrary accuracy when the number of landmarks is increased.

Acknowledgment. We would like to thank Anthony Yezzi for his comments and corrections on the manuscript.

References

[1] B. Appleton and H. Talbot. Globally optimal geodesic active contours. *J. Math. Imaging Vis.*, 23(1):67–86, 2005.

[2] A. C. Beers, M. Agrawala, and N. Chaddha. Rendering from compressed textures. In *Proc. of SIGGRAPH 1996*, pages 373–378, 1996.

[3] L. D. Cohen and R. Kimmel. Global Minimum for Active Contour models: A Minimal Path Approach. *International Journal of Computer Vision*, 24(1):57–78, Aug. 1997.

[4] L.D. Cohen. Multiple Contour Finding and Perceptual Grouping Using Minimal Paths. *Journal of Mathematical Imaging and Vision*, 14(3):225–236, May 2001.

[5] T. H. Cormen, C. E. Leiserson, and R. R. Rivest. *Introduction to Algorithms*. MIT Press, Cambridge, Massachusetts, 1990.

[6] T. Deschamps and L.D. Cohen. Fast Extraction of Minimal Paths in 3D Images and Applications to Virtual Endoscopy. *Medical Image Analysis*, 5(4), December 2001.

[7] M. Droske, M. Meyer, M. Rumpf, and C. Schaller. An adaptive level set method for interactive segmentation of intracranial tumors. *Neurosurgical Research*, 27, 2005.

[8] M. S. Floater, K. Hormann, and M. Reimers. Parameterization of Manifold Triangulations. *Approximation Theory X: Abstract and Classical Analysis*, pages 197–209, 2002.

[9] A. Gersho. *Vector Quantization and Signal Compression*. Kluwer Academic Publishers, Boston, 1992.

[10] A. V. Goldberg and C. Harrelson. Computing the shortest path: A* search meets graph theory. *Technical Report MSR-TR-2004-24*, 2004.

[11] H. Hoppe. Progressive meshes. *Proc. of SIGGRAPH 1996*, pages 99–108, 1996.

[12] R. Kimmel, A. Amir, and A. M. Bruckstein. Finding shortest paths on surfaces using level sets propagation. *IEEE Trans. on PAMI*, 17(6):635–640, 1995.

[13] R. E. Korf. Depth-first iterative-deepening: an optimal admissible tree search. *Artif. Intell.*, 27(1):97–109, 1985.

[14] P. Melchior, B. Orsoni, O. Lavialle, A. Poty, and A. Oustaloup. Consideration of obstacle danger level in path planning using A* and fast-marching optimisation: comparative study. *Signal Processing*, 11(11):2387–2396, November 2003.

[15] N.J. Nilsson. *Problem-solving Methods in Artificial Intelligence*. McGraw-Hill, New York, 1971.

[16] G. Papandreou and P. Maragos. A fast multigrid implicit algorithm for the evolution of geodesic active contours. In *CVPR04*, pages II: 689–694, 2004.

[17] G. Peyré and L. D. Cohen. Geodesic remeshing using front propagation. *Int. J. Comput. Vision*, 69(1):145–156, 2006.

[18] H. Rohnert. Shortest paths in the plane with convex polygonal obstacles. *Inf. Process. Lett.*, 23(2):71–76, 1986.

[19] J.A. Sethian. *Level Sets Methods and Fast Marching Methods*. Cambridge University Press, 2nd edition, 1999.

[20] J.A. Sethian and R. Kimmel. Computing Geodesic Paths on Manifolds. *Proc. Natl. Acad. Sci.*, 95(15):8431–8435, 1998.

[21] W. B. Stout. Smart moves: Intelligent path-finding. *Game Developer*, oct. 1996.

[22] C. Sun and S. Pallottino. Circular shortest path in images. *Pattern Recognition*, 36(3):711–721, March 2003.

[23] V. Surazhsky, P. Alliez, and C. Gotsman. Isotropic remeshing of surfaces: a local parameterization approach. In *Proceedings of 12th International Meshing Roundtable*, 2003.

[24] J. Tsitsiklis. Efficient Algorithms for Globally Optimal Trajectories. *IEEE Trans. on Automatic Control*, 40(9):1528–1538, Step. 1995.