



HAL
open science

Heuristically Driven Front Propagation for Geodesic Paths Extraction

Gabriel Peyré, Laurent D. Cohen

► **To cite this version:**

Gabriel Peyré, Laurent D. Cohen. Heuristically Driven Front Propagation for Geodesic Paths Extraction. Variational, Geometric, and Level Set Methods in Computer Vision (VLSM'05), Oct 2005, Beijing, China. pp.173-185, 10.1007/11567646 . hal-00365623

HAL Id: hal-00365623

<https://hal.science/hal-00365623>

Submitted on 3 Mar 2009

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Heuristically Driven Front Propagation for Geodesic Paths Extraction

Gabriel Peyré¹ and Laurent Cohen²

¹ CMAP, UMR CNRS 7641,
École Polytechnique, 91128 Palaiseau, France
gabriel.peyre@polytechnique.fr

² CEREMADE, UMR CNRS 7534,
Universite Paris Dauphine, 75775 Paris, France
cohen@ceremade.dauphine.fr

Abstract. In this paper we present a simple modification of the Fast Marching algorithm to speed up the computation using a heuristic. This modification leads to an algorithm that is similar in spirit to the A* algorithm used in artificial intelligence. Using a heuristic allows to extract geodesics from a single source to a single goal very quickly and with a low memory requirement. Any application that needs to compute a lot of geodesic paths can gain benefits from our algorithm. The computational saving is even more important for 3D medical images with tubular structures and for higher dimensional data.

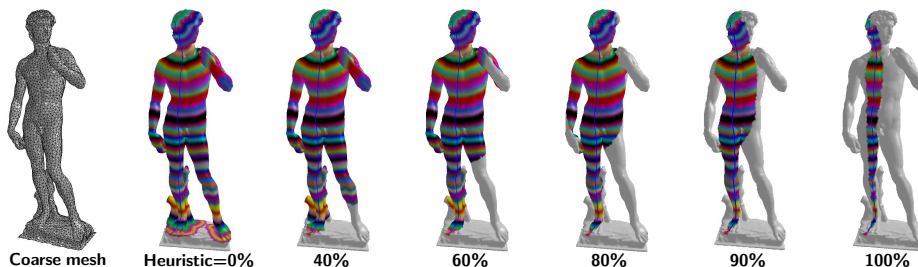


Fig. 1. *Heuristically-driven front propagation on a 3D mesh shown with increasing values of heuristic proportion. The various colors indicate the level sets of the distance function. Only the colored region is explored by our front propagation algorithm.*

1 Shortest Path: Continuous and Discrete Algorithms

A large class of problems can be formulated as the extraction of a shortest paths for a given discrete or continuous metric. The applications of shortest paths thus range from tubular structures extraction in 3D medical images [1] to path finding in video games [2]. The ability to quickly extract a geodesic path is highly desirable. In this paper we propose a modification of front propagation methods. We introduce a heuristic that drives the computations, and greatly reduces the computation time the Fast Marching algorithm.

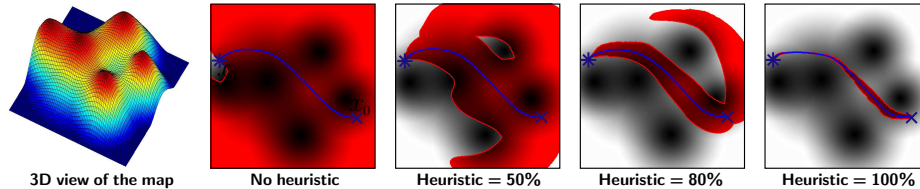


Fig. 2. An example of 2D path planning. The set of alive points according to increasing heuristic is shown in gray.

In this section we briefly review some basic facts about the Fast Marching algorithm and geodesics extraction and we also present a common framework that will allow to introduce our new approach.

1.1 Fast Marching Algorithm

The classical Fast Marching algorithm is presented in [3], and a similar algorithm was also proposed in [4]. This algorithm performs a front propagation that can be used to extract geodesic paths. The minimal length properties of geodesic has been applied in computer vision, for example to solve global minimization problems for deformable models [5]. We will illustrate various applications of geodesic paths in section 3.

In the continuous setting, a geodesic curve minimizes the weighted length of the curve for a given metric. In \mathbb{R}^d , we are given a potential function $P(x) > 0$, and the weighted geodesic distance between two points $x_0, x_1 \in \mathbb{R}^d$, is defined as

$$d(x_0, x_1) \stackrel{\text{def.}}{=} \min_{\gamma} \left(\int_0^1 \|\gamma'(t)\| P(\gamma(t)) dt \right), \quad (1)$$

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

To compute the distance function $U(x) \stackrel{\text{def.}}{=} d(x_0, x)$ with an accurate and fast algorithm, this minimization can be reformulated as follows. The level set curve $\mathcal{C}_t \stackrel{\text{def.}}{=} \{x \mid U(x) = t\}$ propagates following the evolution equation $\frac{\partial \mathcal{C}_t}{\partial t}(x) = \frac{1}{P(x)} \vec{n}_x$ and the function U satisfies the nonlinear *Eikonal* equation:

$$\|\nabla U(x)\| = P(x). \quad (2)$$

The function $F = 1/P > 0$ can be interpreted as the propagation speed of the front \mathcal{C}_t .

The Fast Marching algorithm on an orthogonal grid makes use of an upwind finite difference scheme to compute the value u of U at a given point $x_{i,j}$ of a grid (the equation is written in \mathbb{R}^2 for simplicity), solving:

$$\max(u - U(x_{i-1,j}), u - U(x_{i+1,j}), 0)^2 + \max(u - U(x_{i,j-1}), u - U(x_{i,j+1}), 0)^2 = h^2 P(x_{i,j})^2. \quad (3)$$

This is a second order equation that is solved as detailed for example in [6]. An optimal ordering of the grid points is chosen so that the whole computation only takes $O(N \log(N))$, where N is the number of points.

1.2 Front Propagation Methods for Shortest Path

We now give a common framework for front propagation algorithms, which includes the Fast Marching procedure [3], the Dijkstra algorithm [7] to compute shortest paths on graphs, and our heuristical front propagation procedure. These methods can be described using the following formalism:

- *Alive* is the set of grid 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 implementation-dependant sub-functions

- A way to update the value $U(y)$ at a given Trial point y . In the Fast Marching and our Heuristical front propagation, $U(y)$ is computed by solving equation 3.
- 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, $\mathcal{P}(x)$ is chosen to minimize the number of visited points.

We will explain in section 2 how to actually construct a function \mathcal{P} that makes use of an heuristic.

<p><i>Initialization:</i></p> <ul style="list-style-type: none"> • Alive set: the starting point x_0; • Trial set: the neighbors of x_0; • Far: the set of all other grid points. <p><i>Loop:</i></p> <ul style="list-style-type: none"> • 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: <ul style="list-style-type: none"> – 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.

1.3 Geodesic Extraction

For the applications we have in mind, the computation of the geodesic distance U to a point x_0 is only a tool needed before the extraction of a geodesic that links this point to another point x_1 . This curve can be computed by extracting

the parametric curve $C(t)$ that solves the back propagation equation:

$$\frac{dC}{dt} = -\overrightarrow{\nabla U} \quad \text{with} \quad C(0) = x_0.$$

This gradient descent is a very 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.

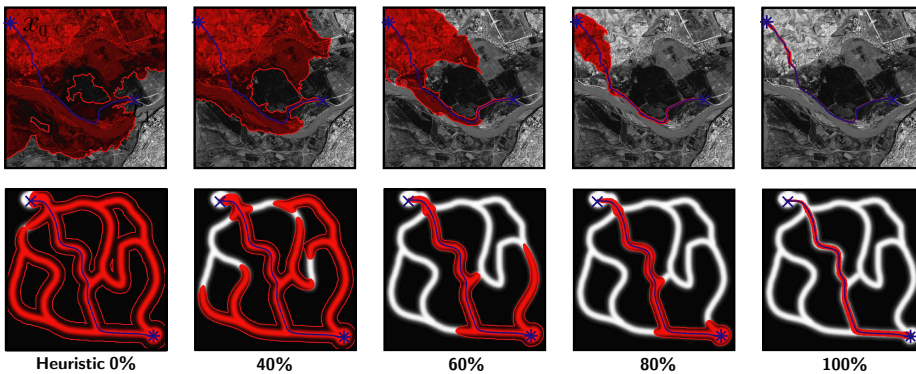


Fig. 3. Path planning using a multiresolution heuristic.

2 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. We also give some insights about how to choose the parameters of our method.

2.1 Fast Marching 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 will try to advance the front toward the goal point x_1 , and not isotropically. In order to do so, assume that together with the the current weighted distance to the start point $U(x) = d(x_0, x)$, we are able to have an estimate of the weighted distance that remains to be marched $V(x) \approx d(x_1, x)$. Then our heuristical front propagation algorithm follows the implementation of the table 1 with a priority map

$$\mathcal{P}(x) = U(x) + \lambda V(x). \quad (4)$$

This introduces the first parameter of our algorithm: the weight λ we use for the heuristic, which typically range from 0% to 100%.

The rational 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 [8]. On figure 2, one can see the effect of various choices for the parameter λ . The value $\lambda = 0$

corresponds to the classical Fast Marching propagation, which result in a very large region of Alive points (shaded in gray). However, as we increase the value of λ toward 100%, the explored region shrinks around the geodesic path that links x_0 to x_1 . There is 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 section 2.4 that although these numerical errors can accumulate during the propagation, the Hausdorff error on the extracted geodesic remains low both in synthetic case studies and in real applications.
- We do not have an immediate access to the remaining distance $d(x, x_1)$, since it would involve performing another front propagation from x_1 . We explain in the next section how to overcome this problem.

Our heuristic ordering strategy is a reminiscence of the A* algorithm [9], which was first introduced to solve problems in artificial intelligence. The use of heuristics and fast exploration strategies (such as IDA* [10]) is crucial in the artificial intelligence field to avoid the complexity of exploring the whole configuration space. These technics are also heavily used for path finding, for example in video games [2]. We show in section 3 that a large class of applications in computer vision can also gain benefit from this paradigm.

2.2 Multiresolution Heuristic

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 P is of size $n \times n$, the query of $\mathcal{P}(y)$ thus requires:

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

There is clearly a tradeoff between choosing a low R to reduce the computation time, and a high R so that $V(x)$ approximates well $d(x, x_1)$. We show in section 2.4 some insight about the correct value for this parameter R .

The new algorithm we propose allows to use multiresolution computation for the extraction of geodesic curve. 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 [11]

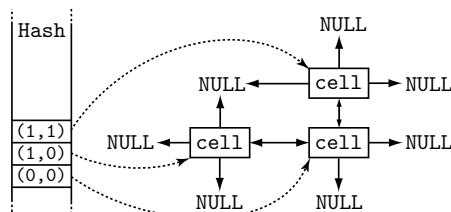
and multigrid [12] have been used in conjunction with geodesic active contours for segmentation purpose. However the problem of segmentation is in some sense dual to the geodesic extraction, in which we are not interested in the level sets of the distance function.

2.3 Reducing Memory Usage

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 [11].

We chose 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 {
    double distance;      // current geodesic distance
    char state;          // either far, open or close
    cell* neighbors[4];  // pointers to the 4 neighbors
};
```



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. This pointer-based representation of the neighboring relation is very convenient to extract the geodesic with a gradient descent. There is some memory overhead due to the fact that we explicitly store pointer to neighbors, but the fact that our scheme can explore significantly less cells than the classical fast marching allows to save much more memory, as shown in next section. The computing time overhead due to the use of a hash table is about 40% in all our tests.

2.4 Numerical Validation

A Matlab implementation of our algorithm, together with the scripts needed to reproduce the figures of this paper, is freely available on Matlab Central [13].

In order to estimate the precision of the results, we use the Hausdorff error between the found paths and the paths obtained by fast marching without heuristic. On 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%.

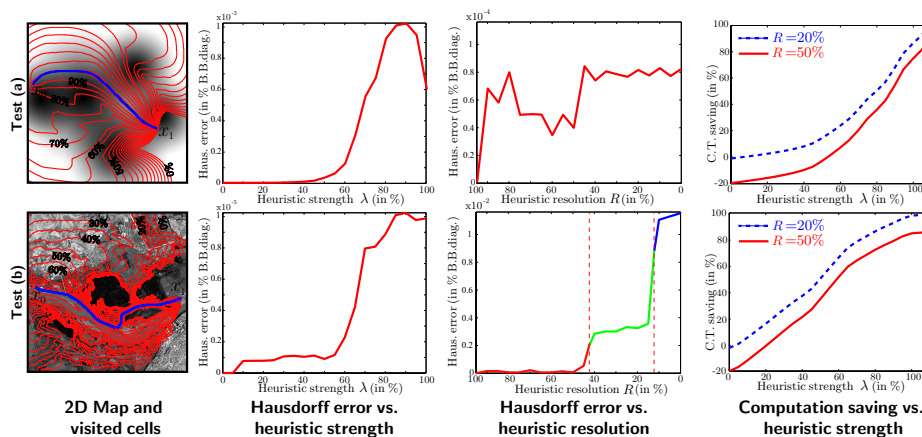


Fig. 4. Influence of heuristic strength and resolution on number of visited cells, Hausdorff error and computation time reduction.

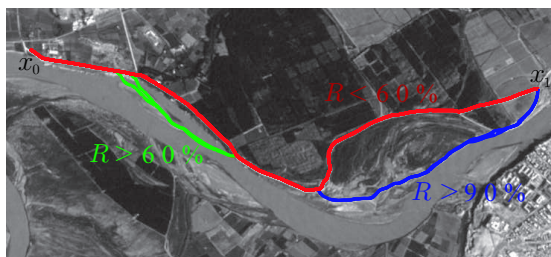


Fig. 5. Influence of the resolution of the heuristic on the shape of the geodesic.

- 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 a lot the resolution without too much impact on the accuracy of the Heuristic. In contrast, one can see that the satellite image suffers from too huge variation when the resolution parameter R becomes smaller than 60% and then again for 90%. This is due to strong topological change in the path, as depicted in figure 5.
- Computation time saving vs. heuristic strength λ : the saving is computed relatively to the time spent by the classical Fast Marching. In 2D the computation times decrease roughly linearly with the strength of the heuristic.
- Computation time saving vs. heuristic resolution R (not shown): there is a constant overhead due to the coarse resolution computation (which results in

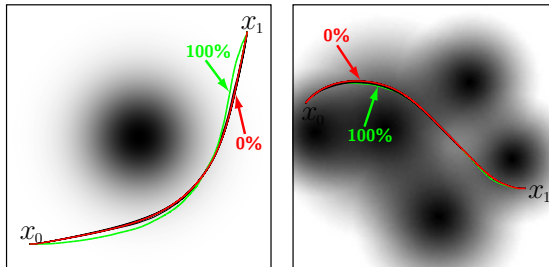


Fig. 6. Graphical display of extracted geodesics for various heuristic strengths.

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 $\lambda \geq 5\%$.

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.

3 Applications

In this section, we show the versatility of our method by proposing various applications where the extraction of geodesics is a central issue. We explain why our algorithm can bring a speed improvement and allows us to use simpler methods.

3.1 Volumetric Geodesics Extraction

3D geodesic extraction is very useful in medical volumetric data analysis. It can be applied to perform tubular structures extraction, and it is extended to virtual endoscopy in [1]. On figure 7 one can see the extraction 3D geodesics on synthetic data (top and middle rows) and on real medical data (bottom row) for $R = 20\%$. The red surface shows the boundary of the explored regions of alive cells. The computation time gain (C.t. gain) is also indicated.

3.2 2D Path Planning

Direct application of AI techniques has been intensively used for path finding, and A* is the method of choice for video games [2]. For the Euclidean case, faster and more complex graph-based methods have been proposed, such as visibility graphs [14].

The Fast Marching method can be used to produce continuous paths with a sub pixel precision [3], even when a non Euclidean metric is used. This is very powerful since it allows the modeling of smooth obstacles (such as forest, enemy repulsion, etc). In [15], the authors compare the Fast Marching and the

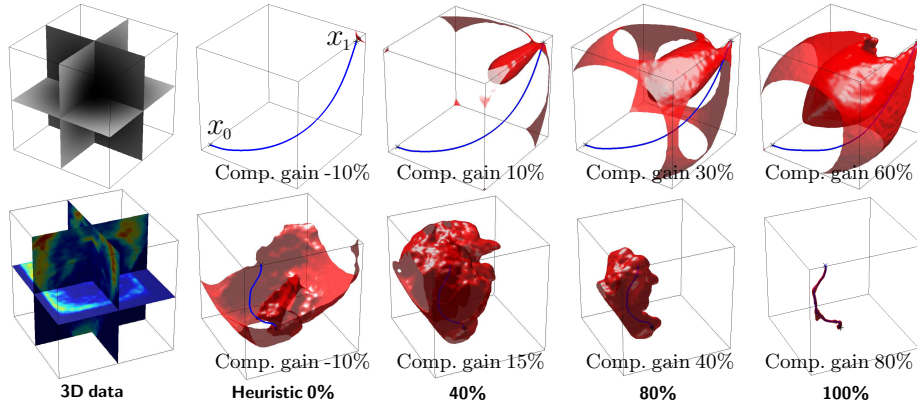


Fig. 7. Extraction of geodesics in 3D.

A* algorithms for path planning. However they do not go one step further and connect these two powerful methods. On figure 3 one can see various paths extracted from a 2D map for a resolution $R = 20\%$ and a varying λ . Although the computed path can deviate from the real geodesic, our algorithm always produces locally acceptable path (smooth and avoiding obstacles).

3.3 Constrained Path Planning

Geodesics can be used to compute the path of a robot with various shape and motion constraints [16]. Basically, each additional degree of freedom add 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 2.3 is crucial to scale to complex problems. The resolution R of the heuristic should be chosen carefully as a function of the typical width of the corridors and rooms the robot must pass through. In our tests we set $R = 20\%$ and we use a varying heuristic strength λ . Note however that although our algorithm can produce a wrong path for aggressive heuristic usage, it never produces non-admissible moves.

On figure 8, 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 9 shows the influence of the heuristic strength λ on the cells explored by the front propagation.

3.4 Globally Optimal Geodesic Active Contours

The concept of circular geodesics was first introduced in [17]. The authors of [18] proposed a simple way to compute circular geodesics about 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 and then the

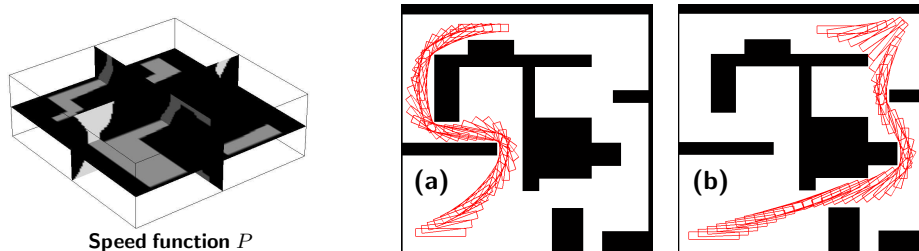


Fig. 8. Examples of constrained path planning.

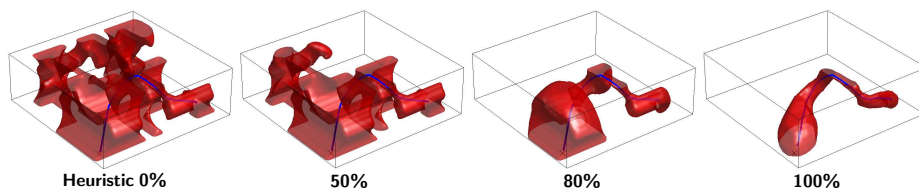


Fig. 9. Explored area for constrained path planning.

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 .

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

$$P(x) = \frac{1}{\|C - x\|} \frac{1}{1 + \|\nabla I(x)\|^2} + \varepsilon, \quad (5)$$

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

The authors of [18] proposed a powerful algorithm based on the branch-and-bound paradigm, which is a dichotomic search that avoids computing the closed geodesic for each point S on the segment CD . However, with our our heuristically driven front propagation, we have tested a simpler algorithm that works well in practice. We simply compute the circular geodesics that pass through 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 through the cut segment.

In figure 10, we have shown a globally optimal circular geodesic, computed with various heuristic strength λ .

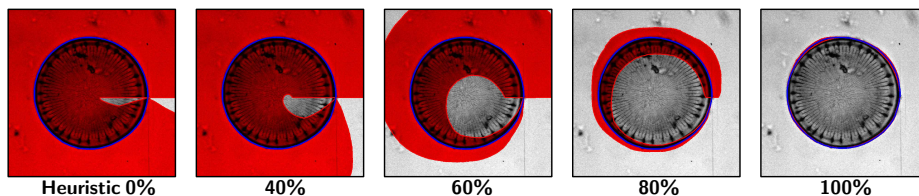


Fig. 10. Globally optimal circular path extraction with increasing heuristic.

3.5 Geodesic Extraction on 3D Meshes

The Fast Marching algorithm has been extended to 3D meshes in [19]. 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 [20]. In our tests, we use the farthest point strategy proposed in [21] for remeshing, since it uses the 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 use a method derived from harmonic mesh parameterization [22]. This involves the resolution of a sparse linear system that search 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 once for all 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 covers the region of alive vertices.

On figure 1 and 11, one can see the algorithm in action on various meshes, and for various values of the parameter λ .

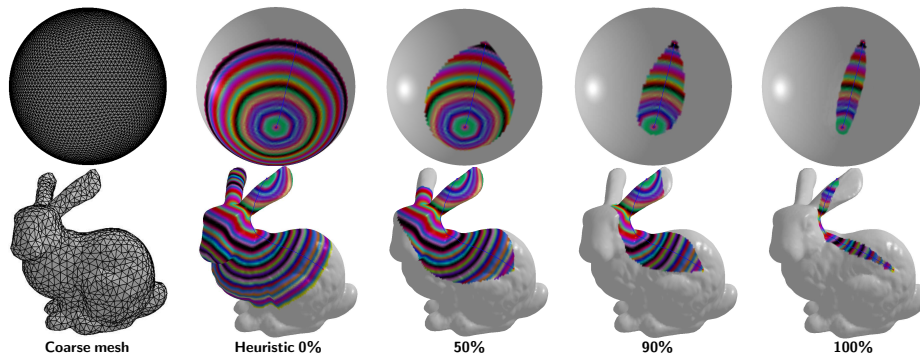


Fig. 11. *Heuristically driven front propagation on 3D meshes.*

4 Conclusion

In this paper we have presented a simple modification of the Fast Marching to speed up the computation time and the memory requirement of the algorithm. This modification is straightforward to implement, since it only involves the computation of a multiresolution heuristic to propagate the front toward the correct direction. Numerical tests on synthetic and real data show that this modification does not result in large distortion of the extracted curve. We examine some

potential applications of this algorithm and show that it fit nicely into various existing computational frameworks.

References

1. Deschamps, T., Cohen, L.: Fast Extraction of Minimal Paths in 3D Images and Applications to Virtual Endoscopy. *Medical Image Analysis* **5** (2001)
2. Stout, W.B.: Smart moves: Intelligent path-finding. *Game Developer* (1996)
3. Sethian, J.: *Level Sets Methods and Fast Marching Methods*. 2nd edn. Cambridge University Press (1999)
4. Tsitsiklis, J.: Efficient Algorithms for Globally Optimal Trajectories. *IEEE Trans. on Automatic Control* **40** (1995) 1528–1538
5. Cohen, L.D., Kimmel, R.: Global Minimum for Active Contour models: A Minimal Path Approach. *International Journal of Computer Vision* **24** (1997) 57–78
6. Cohen, L.: Multiple Contour Finding and Perceptual Grouping Using Minimal Paths. *Journal of Mathematical Imaging and Vision* **14** (2001) 225–236
7. Cormen, T.H., Leiserson, C.E., Rivest, R.R.: *Introduction to Algorithms*. MIT Press, Cambridge, Massachusetts (1990)
8. Kimmel, R., Amir, A., Bruckstein, A.M.: Finding shortest paths on surfaces using level sets propagation. *IEEE Trans. on PAMI* **17** (1995) 635–640
9. Nilsson, N.: *Problem-solving Methods in Artificial Intelligence*. McGraw-Hill, New York (1971)
10. Korf, R.E.: Depth-first iterative-deepening: an optimal admissible tree search. *Artif. Intell.* **27** (1985) 97–109
11. Droske, M., Meyer, M., Rumpf, M., Schaller, C.: An adaptive level set method for interactive segmentation of intracranial tumors. *Neurosurgical Research* **27** (2005)
12. Papandreou, G., Maragos, P.: A fast multigrid implicit algorithm for the evolution of geodesic active contours. In: *CVPR04. (2004) II*: 689–694
13. Peyré, G.: Fast marching toolbox, available on Matlab Central. <http://www.mathworks.com/matlabcentral/> (2005)
14. Rohnert, H.: Shortest paths in the plane with convex polygonal obstacles. *Inf. Process. Lett.* **23** (1986) 71–76
15. Melchior, P., Orsoni, B., Laviaille, O., Poty, A., Oustaloup, A.: Consideration of obstacle danger level in path planning using a* and fast-marching optimisation: comparative study. *Signal Processing* **11** (2003) 2387–2396
16. Kimmel, R., Sethian, J.A.: Optimal algorithm for shape from shading and path planning. *Journal of Mathematical Imaging and Vision* **14** (2001) 237–244
17. Sun, C., Palottino, S.: Circular shortest path in images. *Pattern Recognition* **36** (2003) 711–721
18. Appleton, B., Talbot, H.: Globally optimal geodesic active contours. *Journal of Mathematical Imaging and Vision*, To Appear (2005)
19. Sethian, J., Kimmel, R.: Computing Geodesic Paths on Manifolds. *Proc. Natl. Acad. Sci.* **95** (1998) 8431–8435
20. Hoppe, H.: Progressive meshes. *Proc. of SIGGRAPH 1996* (1996) 99–108
21. Peyré, G., Cohen, L.D.: Geodesic Remeshing Using Front Propagation. *Proc. IEEE Variational, Geometric and Level Set Methods 2003* (2003) 33–40
22. Floater, M.S., Hormann, K., Reimers, M.: Parameterization of Manifold Triangulations. *Approximation Theory X: Abstract and Classical Analysis* (2002) 197–209
23. Surazhsky, V., Alliez, P., Gotsman, C.: Isotropic remeshing of surfaces: a local parameterization approach. In: *Proceedings of 12th International Meshing Roundtable*. (2003)