

Introduction to Level Set Methods and Fast Marching Algorithm

Dmitry Sorokin

Laboratory of Mathematical Methods of Image Processing
Faculty of Computational Mathematics and Cybernetics
Lomonosov Moscow State University

Spring semester 2020

Contents

Level Set Methods

- Basic Idea

- Nice Properties of Implicit Functions

- Evolution Equations

Fast Marching Algorithm

- Theory

- Algorithm

Summary

References

Contents

Level Set Methods

Basic Idea

Nice Properties of Implicit Functions

Evolution Equations

Fast Marching Algorithm

Theory

Algorithm

Summary

References

Contents

Level Set Methods

Basic Idea

Nice Properties of Implicit Functions

Evolution Equations

Fast Marching Algorithm

Theory

Algorithm

Summary

References

Level Sets: Definition

- ▶ Let $f(x, y)$ be an image and c is a constant.
- ▶ A set

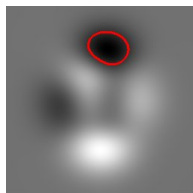
$$\{(x, y) : f(x, y) = c\}$$

is called a c -**level set** of a function f .

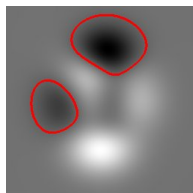
- ▶ When the number of variables is two, this is a **level curve**, if it is three this is a **level surface**.
- ▶ We will use term **level contour** in both cases.

Level contours

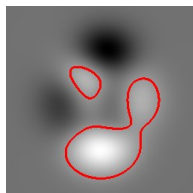
- ▶ Level contours of an image at different levels are shown in red.



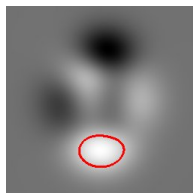
level 50



level 100



level 150



level 200

Parametric Curve and Surface

- ▶ The parametric curve on a plane is a mapping:

$$C(s) : [0, 1] \rightarrow \mathbb{R}^2, \quad s \mapsto C(s) = (x(s), y(s))^T.$$

- ▶ The parametric surface in a volume is a mapping:

$$C(s, r) : \Omega = [0, 1] \times [0, 1] \rightarrow \mathbb{R}^3$$

$$(s, r) \mapsto C(s, r) = (x(s, r), y(s, r), z(s, r))$$

- ▶ If curve $C(s)$ or surface $C(s, r)$ evolves in time it becomes $C(s, t)$ or $C(s, r, t)$
- ▶ To simplify notations we omit parameters (s, r) , so

$C(t)$ is a curve or surface **evolving in time**

$C(t)$ is a **vector** in \mathbb{R}^2 (curve) or \mathbb{R}^3 (surface)

Embedding Curve Evolution in Level Set

- ▶ Consider the curve $C(t)$ evolution in time t in **the most general form**:

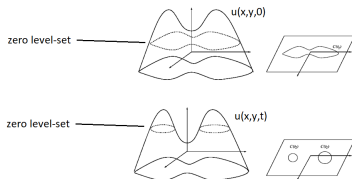
$$\partial_t C = \beta \mathbf{n}$$

- ▶ The evolving curve can be represented in form of k -level set of **some** higher dimension function (i.e. image)

$$u(x, y, t) : \mathbb{R}^2 \times [0, T) \rightarrow \mathbb{R},$$

such that

$$u(x(t), y(t), t) = k \quad \text{for all } (x(t), y(t)) \in C(t)$$



Embedding Curve Evolution in Level Set

- ▶ If we differentiate the equation $u(x(t), y(t), t) = k$ with respect to t

$$0 = \frac{du}{dt} = \frac{\partial u}{\partial x} \frac{\partial x}{\partial t} + \frac{\partial u}{\partial y} \frac{\partial y}{\partial t} + \frac{\partial u}{\partial t} = (\nabla u)^T \partial_t \mathbf{C} + \partial_t u = \dots$$

and plug curve evolution equation $\partial_t \mathbf{C} = \beta \mathbf{n}$

$$\dots = (\nabla u)^T \beta \mathbf{n} + \partial_t u = -(\nabla u)^T \beta \frac{\nabla u}{|\nabla u|} + \partial_t u$$

we get

$$\partial_t u = \beta |\nabla u|.$$

Equivalence Between Curve And Level Set Evolution

- ▶ Any curve evolution in the direction of its normal n

$$\partial_t C = \beta \mathbf{n}$$

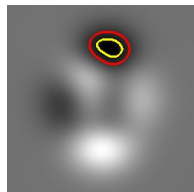
can be **embedded** as a level set into the image evolution

$$\partial_t u = \beta |\nabla u|.$$

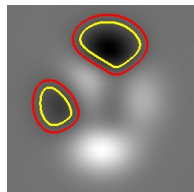
- ▶ Parameter β controls the rate of evolution.
- ▶ Conversely, one can also show that any image evolution of type $\partial_t u = \beta |\nabla u|$ leads to the evolution equation $\partial_t C = \beta \mathbf{n}$ for **all its level curves**.
- ▶ Initial condition: $f(x, y) = u(x, y, 0)$ and $C(0)$ is a level set of f .

Dilation of Level Contours

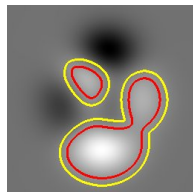
- ▶ Initial contours of an image at different levels are in **red**.
- ▶ Contours of dilated image using disk structuring element with radius 20 ($t = 20, \beta = 1$) are in **yellow**.



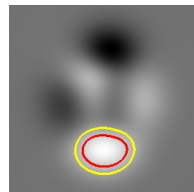
level 50



level 100



level 150



level 200

- ▶ Here by changing **the function u (image)** we change the contour C

Level Set Methods, Idea

- ▶ Contour C is represented as (zero) **level set** of an implicit function u .
- ▶ **Contour evolution** is driven by the following general PDE:

$$\partial_t u = \beta |\nabla u|.$$

i.e. contour points are moving in the normal direction.

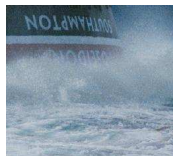
- ▶ Level set methods comprise **numerical algorithms** for the solution of the equation above. The numerical algorithms differ for particular β , which can be a function of u , another image, etc. (the choice of β is application dependent).
- ▶ u is **not the image we are processing**, it's an **embedding function** that represents our evolving contour

Level Set Methods, Applications

- ▶ Computer animation (Shrek, Poseidon, ...), simulation of water, gas, etc.



Source: <http://www.digitalmediafx.com/>



Source: <http://graphics.stanford.edu/~losasso/>

- ▶ Image inpainting



Source: <http://www.iaa.upf.es/~mbertalmio/restoration.html>

- ▶ Image segmentation

Contents

Level Set Methods

Basic Idea

Nice Properties of Implicit Functions

Evolution Equations

Fast Marching Algorithm

Theory

Algorithm

Summary

References

Implicit Functions

- ▶ Let $f(x, y)$ is a function (image) $f : \Omega \rightarrow \mathbb{R}$.
- ▶ **Implicit function** is a function defined by an equation

$$f(x, y) = 0$$

meaning one have to solve an equation $f(x, y) = 0$ to evaluate the function.

- ▶ The solution of the equation is the **zero level set** of f .

Easy inside vs. outside test

- ▶ Function $u : \Omega \rightarrow \mathbb{R}$ divides an image plane Ω into three parts

- ▶ **Inside**

$$\Omega^- = \{x : u(x) < 0\}$$

- ▶ **Outside**

$$\Omega^+ = \{x : u(x) > 0\}$$

- ▶ **Boundary** (contour)

$$\partial\Omega = \mathcal{C} = \{x : u(x) = 0\}$$

Easy set operations

Let $u_1 : \Omega_1 \rightarrow \mathbb{R}$ and $u_2 : \Omega_1 \rightarrow \mathbb{R}$ be two implicit functions. Then

- ▶ Inside part of $u = \max(u_1, u_2)$ is the **intersection**

$$\Omega^- = \Omega_1^- \cap \Omega_2^-$$

of the inside parts.

- ▶ Inside part of $u = \min(u_1, u_2)$ is the **union**

$$\Omega^- = \Omega_1^- \cup \Omega_2^-$$

of the inside parts.

- ▶ etc.

Easy Normal Computation

- ▶ Let $C = \{\mathbf{x} : u(\mathbf{x}) = 0\}$ be a curve ($\mathbf{x} \in \mathbb{R}^2$) or a surface ($\mathbf{x} \in \mathbb{R}^3$).
- ▶ Then the **unit normal vector** at the point \mathbf{x} is defined by

$$\mathbf{n}(\mathbf{x}) = \frac{\nabla u}{|\nabla u|}$$

Easy Curvature Computation

- ▶ The **curvature** is defined by (= mean curvature for \mathbb{R}^3)

$$\kappa(\mathbf{x}) = \nabla \cdot \mathbf{n}(\mathbf{x}) = \nabla \cdot \left(\frac{\nabla u}{|\nabla u|} \right),$$

as the divergence of the direction of the gradient of u .

- ▶ The curvature can be computed in terms of the partial derivatives as:

$$\kappa(\mathbf{x}) = (u_x^2 u_{yy} - 2u_x u_y u_{xy} + u_y^2 u_{xx}) / |\nabla u|^3$$

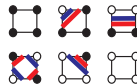
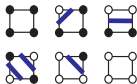
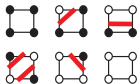
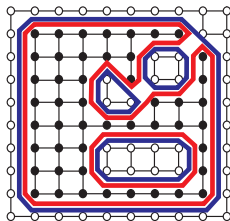
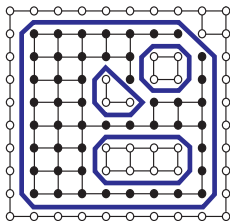
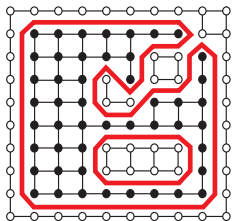
$$\begin{aligned} \kappa(\mathbf{x}) = & (u_x^2 u_{yy} - 2u_x u_y u_{xy} + u_y^2 u_{xx} + u_x^2 u_{zz} - 2u_x u_z u_{xz} \\ & + u_z^2 u_{xx} + u_z^2 u_{zz} - 2u_y u_z u_{yz} + u_z^2 u_{yy}) / |\nabla u|^3 \end{aligned}$$

Topology changes

- ▶ Connectivity is not (and need not be) kept.
- ▶ **Easy topology changes** (e.g., change in the number of components, creation of holes, etc.).

Extraction of Zero Contour Level

- ▶ Marching squares/cubes algorithms approximate **boundary** between positive and negative points.
- ▶ One can also easily detect **border** pixels of inside (outside) regions.



Implicit Functions Can Represent Any Contour

- ▶ We can embed any closed curve $C = \partial\Omega$ into its **signed distance function** d defined by:

$$d(\mathbf{x}) = \begin{cases} -\min_{\mathbf{y} \in C} |\mathbf{x} - \mathbf{y}| & \text{if } \mathbf{x} \in \Omega^- \\ +\min_{\mathbf{y} \in C} |\mathbf{x} - \mathbf{y}| & \text{if } \mathbf{x} \in \Omega^+ \cup C \end{cases}$$

- ▶ Notice, that due to Euclidean distance we have

$$|\nabla d| = 1$$

- ▶ It simplifies and improves numerical computations.

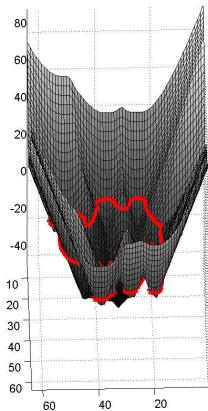
$$n = \nabla d \text{ and } \kappa = \nabla \cdot \nabla d = \nabla^2 d = \Delta d,$$

where Δd is the Laplacian of d , i.e. $\Delta d = d_{xx} + d_{yy} + d_{zz}$.

Signed Distance Function (SDF): Example



digital shape



distance $d(\mathbf{x})$ from its boundary

Properties of Implicit Functions, Summary

- ▶ Easy computation of **geometric properties** (normal, curvature)
- ▶ Easy **inside vs. outside** test
- ▶ Easy set operations (e.g. intersection, union, ...)
- ▶ Discretization on a regular grid leads to a **digital image**
- ▶ Boundary is defined implicitly
 - ▶ It must be computed, (e.g. by marching squares or marching cubes algorithm)
 - ▶ Connectivity is not (and need not be) kept, easy **topology changes** during its evolution
- ▶ Formulation does not depend on the number of dimensions (**easy generalization** from 2D to 3D images - or even more).

Contents

Level Set Methods

Basic Idea

Nice Properties of Implicit Functions

Evolution Equations

Fast Marching Algorithm

Theory

Algorithm

Summary

References

Curve Evolution using Level Sets

- ▶ The evolution of the contour

$$\partial_t C(t) = \beta \mathbf{n}$$

is equivalent to PDE

$$u_t = \beta |\nabla u| \tag{1}$$

where β is the rate of evolution in the normal direction to the contour.

- ▶ We can evolve the contour using the following steps:
 1. Determine the embedding function (image) $u(x, y, t)$ e.g. by SDF
 2. Evolve it locally according to the level set flow (1)
 3. Recover the zero-level set iso-surface (curve position)
 $C = (x, y) : u(x, y, t) = 0$
 4. Re-initialize the implicit function $u(x, y, t)$ as SDF and Go to step 1

Level Set Evolution Equations: Basic Motions Types

- ▶ **Contour motion** is driven by PDE

$$u_t = \beta |\nabla u|$$

where β is the rate of evolution in the normal direction to the contour.

- ▶ There are three basic types of motion:
 - ▶ motion under **velocity field** $V(\mathbf{x}, t)$

$$\beta = F_{ext} = V(\mathbf{x}, t) \cdot \mathbf{n}$$

- ▶ motion in the **normal direction** (balloon force, dilation)

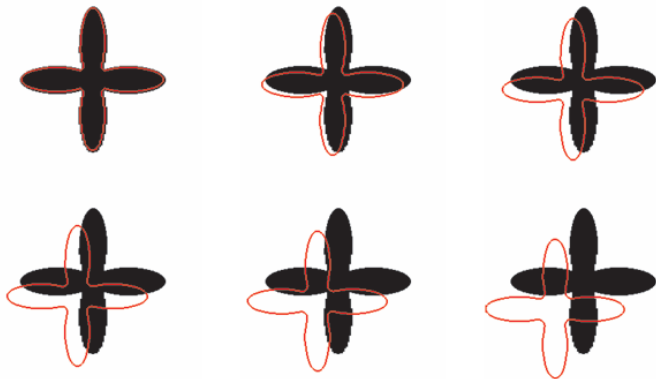
$$\beta = a$$

- ▶ motion under mean **curvature** (internal force)

$$\beta = F_{curv} = -\epsilon \kappa$$

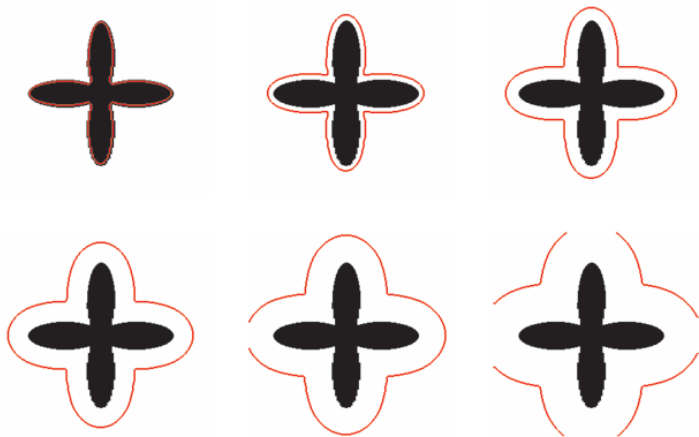
- ▶ Different motions require different numerical solutions (See the next Lecture).

Example 1



Motion under external velocity field. The velocity vector is equal to $(-1, -1)$ for every grid point in this example.

Example 2



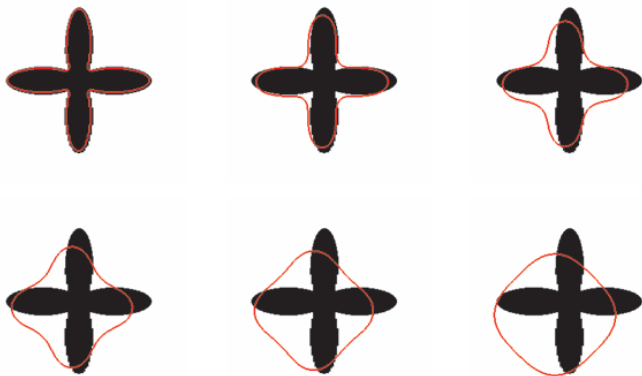
Motion in the normal direction. $a = 1.0$.

Example 3



Motion under mean curvature ($\epsilon = 1.0$)

Example 4



All three types of motion together. All vectors in V are equal to $(-1, -1)$, $\epsilon = 0.25$ and $a = 1.0$.

Contents

Level Set Methods

Basic Idea

Nice Properties of Implicit Functions

Evolution Equations

Fast Marching Algorithm

Theory

Algorithm

Summary

References

Contents

Level Set Methods

Basic Idea

Nice Properties of Implicit Functions

Evolution Equations

Fast Marching Algorithm

Theory

Algorithm

Summary

References

Fast Marching Algorithm – Idea

- ▶ Assume, that we have an equation

$$\begin{aligned}u(\mathbf{x}, t)_t &= a(\mathbf{x})|\nabla u(\mathbf{x}, t)|, \\u(\mathbf{x}, 0) &= u_0\end{aligned}$$

where always $a(\mathbf{x}) > 0$. Then the contour is moving in one direction only and every point is visited only once.

- ▶ The value of $a(\mathbf{x})$ describes **how fast the contour may propagate** through the given point.
- ▶ The evolution equation is equivalent to the following equation

$$\begin{aligned}a(\mathbf{x})|\nabla T(\mathbf{x})| &= 1, \\T(C_0) &= 0,\end{aligned}$$

where $C_0 = \{(x, y) | u(x, y, 0) = 0\}$ is the initial position of the boundary and $T(x, y)$ is the crossing times of zero level set in all points.

Equivalence to Eikonal equation

- ▶ The equations are equivalent (see Slide 9)

$$u(\mathbf{x}, t)_t = a(\mathbf{x})|\nabla u(\mathbf{x}, t)|, \quad \partial_t C(t) = a(\mathbf{x})\mathbf{n},$$

$$u(\mathbf{x}, 0) = u_0 \quad C(0) = C_0$$

- ▶ Let $T(x, y)$ be the time when the curve $C(t)$ reaches the point \mathbf{x} :

$$T(C(t)) = t \implies \left\{ \frac{\partial}{\partial t} \right\} \implies \nabla T(\mathbf{x}) \cdot C_t = 1 \implies$$

$$\nabla T(\mathbf{x}) \cdot \left(a(\mathbf{x}) \frac{\nabla T(\mathbf{x})}{|\nabla T(\mathbf{x})|} \right) = 1 \implies$$

thus we get the **Eikonal equation**

$$a(\mathbf{x}) \cdot |\nabla T(\mathbf{x})| = 1,$$

$$T(C_0) = 0$$

where $C_0 = \{(x, y) | u(x, y, 0) = 0\}$ is the initial position of the boundary.

Important Speed Functions

- ▶ If $a = 1$ for every pixel, we obtain (in T) the shortest **Euclidean distance** to the initial contour in every pixel.
- ▶ If $a = 1$ inside a mask and $a \rightarrow 0$ outside the mask, we obtain **geodesic distance** within the mask to the contour.
- ▶ If $a = g(|\nabla f(x, y)|)$, where g is a decreasing function, e.g.
 - ▶ $g = \frac{1}{1 + \lambda |\nabla G_{\sigma} * f(x, y)|}$ or $g = e^{-\lambda |\nabla G_{\sigma} * f(x, y)|}$,which goes to 0 for points with the high gradient and is close to 1 for points with almost zero gradient, we obtain **fast marching segmentation algorithm**.

Euclidean Distance Map: Example



The inside of
a contour

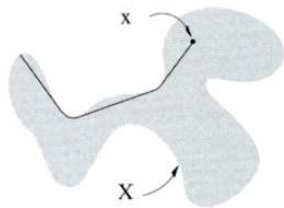


Distance from outside



Distance from inside

Geodesic Distance Map: Example

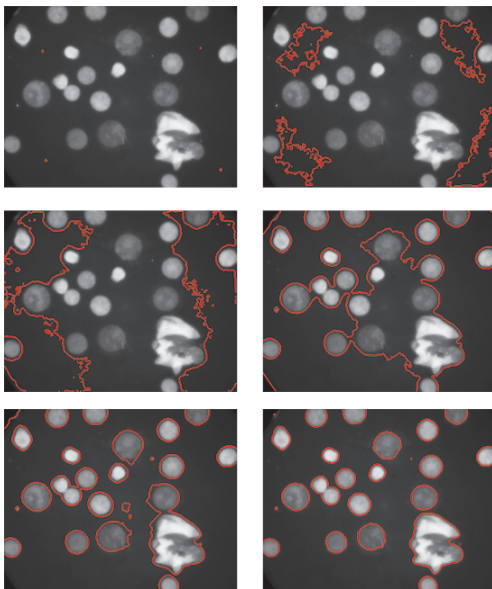


Longest geodesic from x in X



Geodesic distance function

Segmentation using fast marching method, Example 1



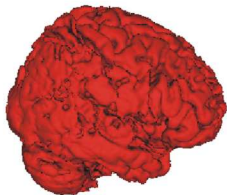
3D Example - Interphase Chromosome Reconstruction

- ▶ 5-inputdata3d.avi
- ▶ 7-fmmcontours.avi

3D Example - Brain Surface Reconstruction

- ▶ 10-brainslides.avi
- ▶ 11-braincontours.avi

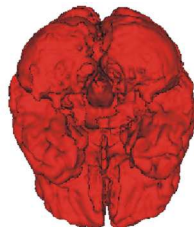
3D Example - Brain Surface Reconstruction, Result



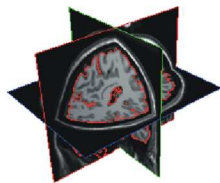
a) $T = 105$, pohled z boku



b) $T = 105$, pohled zezadu



c) $T = 105$, pohled zespodu



d) $T = 105$, 2D hranice - řezy x, y, z



e) $T = 105$, 2D hranice - řezy y, z



f) $T = 105$, 2D hranice - řez z

Contents

Level Set Methods

Basic Idea

Nice Properties of Implicit Functions

Evolution Equations

Fast Marching Algorithm

Theory

Algorithm

Summary

References

Fast Marching Algorithm

- ▶ We want to solve

$$\begin{aligned}a(x, y)|\nabla T(x, y)| &= 1, \\ T(C_0) &= 0,\end{aligned}$$

where $C_0 = \{(x, y) | u(x, y, 0) = 0\}$ is the initial position of the boundary

- ▶ We assume grid with distance h_1 and h_2 between grid lines.

$$x_i := ih_1, \quad i = 0, 1, \dots, N - 1$$

$$y_j := jh_2, \quad j = 0, 1, \dots, M - 1$$

$$a(x_i, y_j) \approx a_{i,j}$$

$$T(x_i, y_j) \approx T_{i,j}$$

Fast Marching Algorithm

- ▶ Finite differences

$$D_{ij}^{-x} T = \frac{T_{i,j} - T_{i-1,j}}{h}$$

$$D_{ij}^{+x} T = \frac{T_{i+1,j} - T_{i,j}}{h}$$

$$D_{ij}^{-y} T = \frac{T_{i,j} - T_{i,j-1}}{h}$$

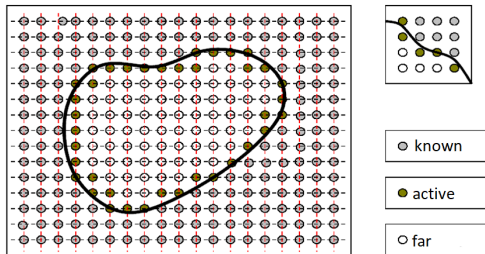
$$D_{ij}^{+y} T = \frac{T_{i,j+1} - T_{i,j}}{h}$$

- ▶ Numerical scheme (upwind)

$$\max\left(D_{ij}^{-x} T, D_{ij}^{+x} T, 0\right)^2 + \max\left(D_{ij}^{-y} T, D_{ij}^{+y} T, 0\right)^2 = \frac{1}{a_{i,j}^2} \quad (2)$$

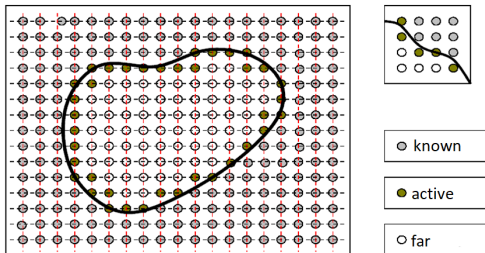
Fast Marching: Input and Initialization

- ▶ Input
 - ▶ Speed function $a_{ij} > 0$ defined on discrete grid G .
 - ▶ Initial contour $C_0 = \{(x_i, y_j) \in G : T(x_i, y_j) = 0\}$
- ▶ Output: Arrival times T_{ij}
- ▶ Initialization:
 - ▶ Set $Trial = C_0$, i.e. $T_{ij} = 0$ for $(x_i, y_j) \in Trial$.
 - ▶ Set $Far = G \setminus C_0$, $T_{ij} = \infty$ for $(x_i, y_j) \in Far$.
 - ▶ Set $Known = \emptyset$.



Fast Marching: Main loop

- ▶ Find $p \in Trial$ having the smallest arrival time $T(p)$,
- ▶ Exclude p from $Trial$ and include it into $Known$,
- ▶ For each neighbour q of p do
 1. Compute the new arrival time $T_{new}(q)$ using (2).
 2. If $T_{new}(q) < T(q)$ then $T(q) = T_{new}(q)$.
 3. If $q \in Far$ then include it into $Trial$.
- ▶ Break main loop and stop the algorithm as soon as $Trial$ set is empty.



Trial set

- ▶ We can implement *Trial* set as a min-heap data structure.
- ▶ It is an ordered balanced binary tree.
- ▶ Operation *PushElement* costs $\mathcal{O}(\log n)$
- ▶ Operation *PopMinElement* costs $\mathcal{O}(\log n)$
- ▶ One can implement it using an array.

Fast Marching Algorithm – Discussion

- ▶ It considers **one directional motion only**.
- ▶ a is always positive.
- ▶ **Fast computation** (Non iterative process in time!) – time complexity $\mathcal{O}(n \log n)$, where n is the number of grid points.
- ▶ Easy to implement (uses min-heap structure).
- ▶ It is a general algorithm (can compute Euclidean as well as geodesic distance maps, perform segmentation, ...)
- ▶ In the case of segmentation, the problem of the ideal crossing-time arises.

Contents

Level Set Methods

Basic Idea

Nice Properties of Implicit Functions

Evolution Equations

Fast Marching Algorithm

Theory

Algorithm

Summary

References

Summary

- ▶ Contour is a **zero level set** of an implicit function u
- ▶ Evolution is driven by a special PDE in u .
- ▶ Evolution of u leads to the evolution of all its contours.
- ▶ Implicit representation of contours has many advantages
 - ▶ Easy computation of geometric properties (normal, curvature)
 - ▶ Discretization leads to a digital image
 - ▶ Easy topology changes
- ▶ One directional motion can be computed by means of **fast marching algorithm**.

Contents

Level Set Methods

Basic Idea

Nice Properties of Implicit Functions

Evolution Equations

Fast Marching Algorithm

Theory

Algorithm

Summary

References

References

- ▶ S. Osher, R. Fedkiw, Level Set Methods and Dynamic Implicit Surfaces, Springer 2003
- ▶ S. Osher, N. Paragios, Geometric Level Set Methods in Imaging, Vision, and Graphics, Springer 2003
- ▶ R. Kimmel, Numerical Geometry of Images: Theory, Algorithms, and Applications, Springer 2004