Dijkstra's algorithm, Fast marching & Level sets

FMAN30
Einar Heiberg, einar@heiberg.se

1

Looking back

- Medical image segmentation is (usually) selecting a suitable method from a toolbox of available approaches
- Making problem dependent customizations improvements to include a priori information

2

Tools in the toolbox

- Statistical methods (Markov fields etc)
- Active contours
- Dynamical programming
- Fast marching
- Level set methods
- Variational methods
- …..

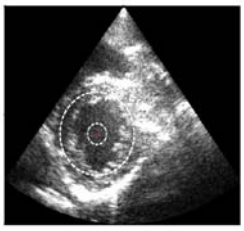
3

Purpose of this lecture(s)

- Dijkstra's shortest path algorithm
- Understanding of fast marching
- Understanding of level set method
- Introducing Assignment 4

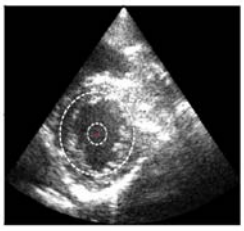
4

Dijkstra's - motivation



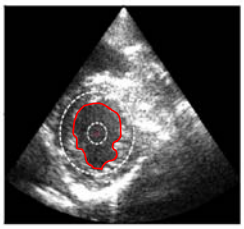
5

Dijkstra's - motivation



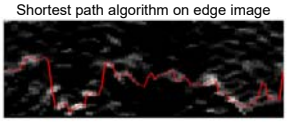
6

Dijkstra's - motivation



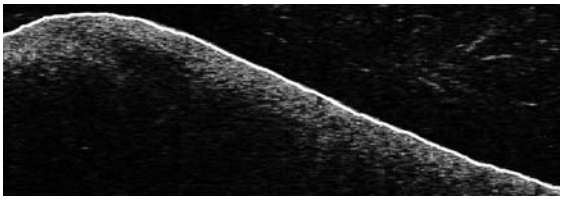
Radial Resample and filtering

Shortest path algorithm on edge image



7

Dijkstra's - motivation



Boundary tracing of ice radar profile on Svalbard glacier

8

Mathematical formulation

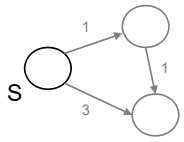
$$\min \sum_{\langle (i,j),(k,l)\rangle \in A} c_{(i,j)(k,l)} x_{(i,j)(k,l)}$$

$$\begin{cases} x_{(i,j)(k,l)} \in \{0,1\} \\ \sum_{(i,j)\in S(i)} x_{(i,j)(k,l)} - \sum_{(i,j)\in T(i)} x_{(k,l)(i,j)} = \begin{cases} -1 \text{ if } (i,j) = S \\ 1 \text{ if } (i,j) = T \\ 0 \text{ otherwise} \end{cases} \end{cases}$$

Where $\langle (i,j),(k,l)\rangle$ is a vertice, $c(i,j)(k,l)>0$ is the cost for the vertice, $x(i,j)(k,l)$ is 1 if the vertice is included in the optimal path, S is the starting point, and T is the end point.

9

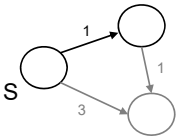
Dijkstra's algorithm



1. Start with node S
2. Keep track of a set of selected nodes *N* (black)
3. Select the shortest vertice *w* out of *N*
4. Keep track of precessor
5. Repeat until all nodes in *N*

10

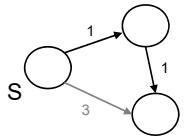
Dijkstra's algorithm



1. Start with node S
2. Keep track of a set of selected nodes *N* (black)
3. Select the shortest vertice *w* out of *N*
4. Keep track of precessor
5. Repeat until all nodes in *N*

11

Dijkstra's algorithm



1. Start with node S
2. Keep track of a set of selected nodes *N* (black)
3. Select the shortest vertice *w* out of *N*
4. Keep track of precessor
5. Repeat until all nodes in *N*

12

Practical implementation

1. Keep a list of selected nodes (*N*)
2. Store a sorted list of outward vertices
3. Take lowest cost vertex from list, add node to *N*
4. Add new vertices to the sorted list
5. Repeat from 3) until all nodes are selected
6. Shortest path is “backtracked”

Suitable data structure for the sorted list is a sorted heap.

13

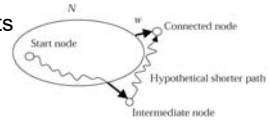
Optimality

- We keep track on selected nodes *N*
- In each iteration find the shortest vertex *w* that connects to a node outside of *N*
- Need to prove that there can not be a shorter path from start to the node connected by the shortest vertex *w*

14

Optimality

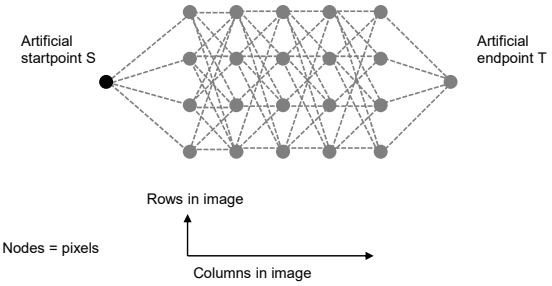
Suppose a such path exists



1. w was selected as the shortest vertex in *N* to the node
2. Cost between nodes are >=0.
3. Hypothetical shorter path must go via an intermediate node as *w* was chosen as the smallest vertex from the set *N*.
4. The hypothetical path can not be shorter than the path to *w* as costs >= 0. Thus the calculated path is the optimal shortest path.

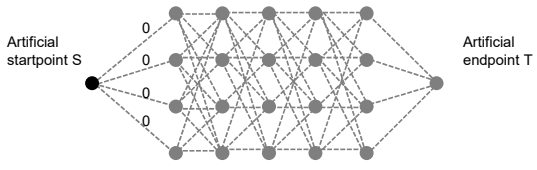
15

Dijkstra’s algorithm



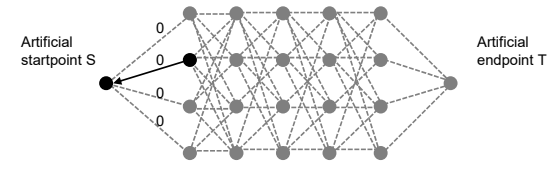
16

Dijkstra’s algorithm



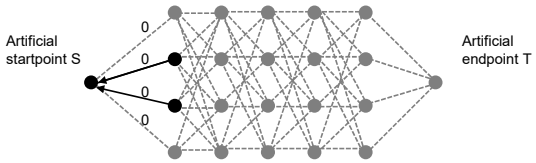
17

Dijkstra’s algorithm



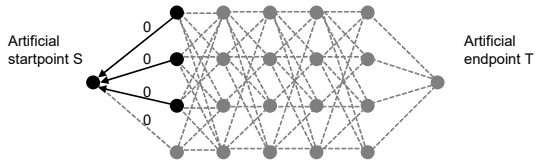
18

Dijkstra's algorithm



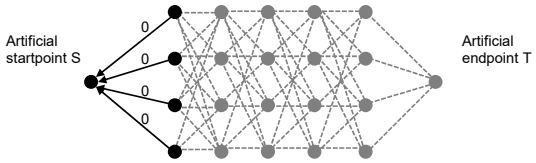
19

Dijkstra's algorithm



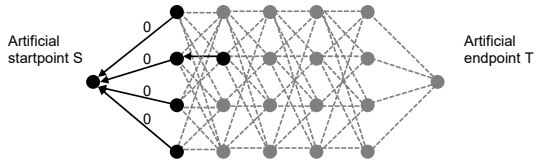
20

Dijkstra's algorithm



21

Dijkstra's algorithm

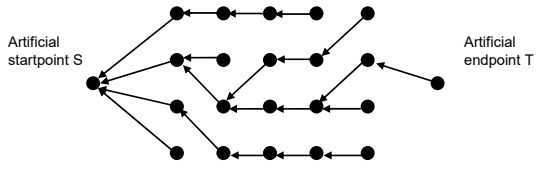


22

Continue….

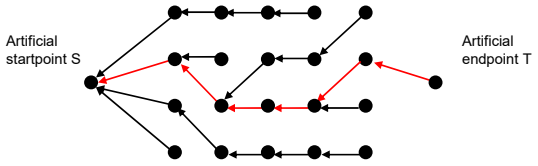
23

Dijkstra's algorithm



24

Dijkstra's algorithm



Final optimal path

25

Setting the costs

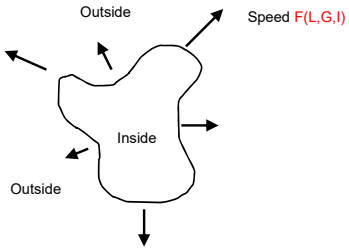
- Problem dependent
- Low cost for favorable areas?
- Low cost for edges?
- Penalize large jumps in image (derivative)
- Penalize curvature (how....?)
- Force a closed line?

26

Level set theory

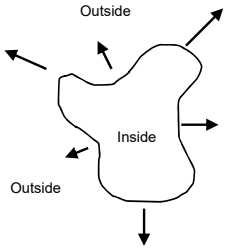
27

Interface propagation



28

Interface propagation



Speed F(L,G,I)

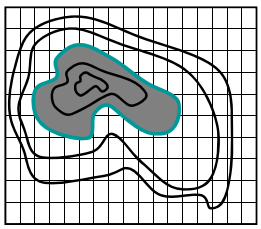
L = Local properties, such as curvature & normal direction

G = Global properties

I = Independent properties, for instance underlying vector field

29

Representing moving interfaces as "height curves"

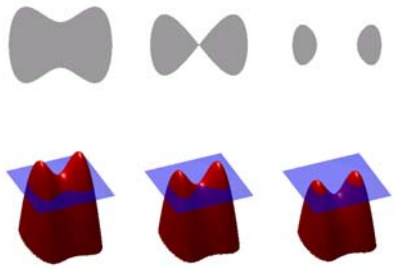


Advantages:
- Variable topology
- Generic

Disadvantages:
- Difficult to introduce *a priori* to constrain topology

30

Object representation



31

Advantages

- Expansion to higher dimensions trivial
- Topological changes naturally
- Simple to calculate curvature
- Efficient computational schemes can be developed

32

Different interface formulations

- Boundary value formulation
- Intitial value formulation

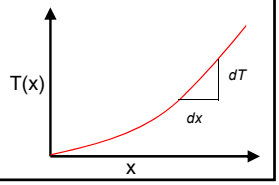
33

Boundary value formulation

- Assume F>0 (i.e always outward motion)
- Describe interface by computing arrival time T(x,y)
- Use distance = rate*time

$$dx = FdT$$

$$1 = F\frac{dT}{dx}$$



34

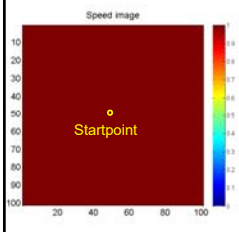
Boundary value formulation

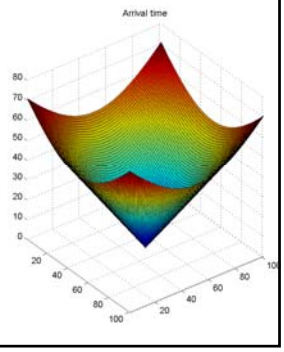
- In higher dimensions dT becomes ∇T which gives the Eikonal equation:

$$|\nabla T|F = 1$$

35

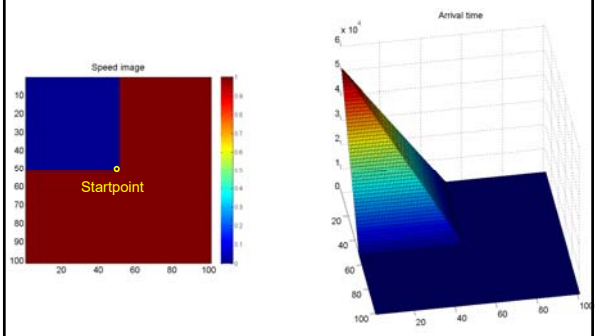
Examples





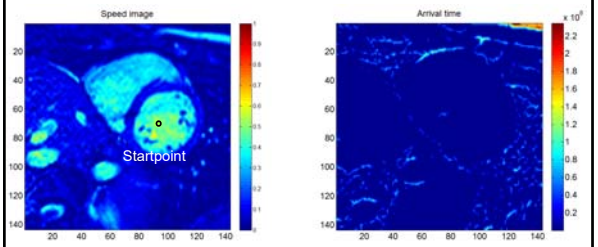
36

Examples



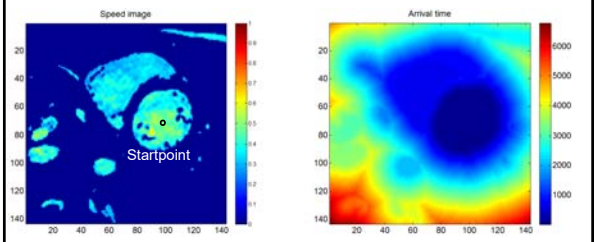
37

Examples



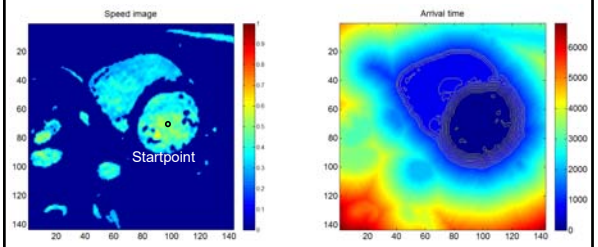
38

Examples



39

Examples



40

Key issues

- Finding good starting point(s) where T=0
- Finding suitable speed map (translating intensity and edges)
- Selecting the object based on arrival time

41

Initial value formulation

Assume that F can be both positive and negative.

Problem: The boundary can pass a certain position multiple times.

Solution: Represent the curve as the zero level of a time dependent level set function φ instead of arrival time

42

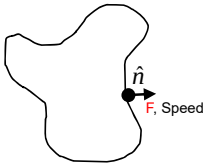
Initial value formulation

The level set value for a point on the boundary must always be zero:

$\phi(x(t),t)=0$

chain rule gives:

$\phi_t+\nabla\phi(x(t),t)\cdot\frac{dx(t)}{dt}=0$



We have $\frac{dx(t)}{dt}\cdot\hat{n}=F,\quad \hat{n}=\nabla\phi/|\nabla\phi|$ gives:

$$\phi_t+F|\nabla\phi|=0$$

43

Boundary value formulation	Initial value formulation
$\lvert\nabla T\rvert F=1$	$\phi_t+F\lvert\nabla\phi\rvert=0$
Front representation: $\Gamma(t)=\{(x,y)T(x,y)=t\}$	Front representation: $\Gamma(t)=\{(x,y)\phi(x,y,t)=0\}$
Requires F>0	Arbitrary F

44

Boundary value formulation	Initial value formulation
$\lvert\nabla T\rvert F=1$	$\phi_t+F\lvert\nabla\phi\rvert=0$
Front representation: $\Gamma(t)=\{(x,y)T(x,y)=t\}$	Front representation: $\Gamma(t)=\{(x,y)\phi(x,y,t)=0\}$
Requires F>0	Arbitrary F
Advantage: Fast	Advantage: Can use shape dependent speed function (curvature, smoothing)

45

Fast numerical solution to Eikonal equation - Fastmarching

Solution similar to Dijkstra's:

1. For each neighbor to the starting node(s) compute arrival time. Store arrival times in a sorted list of arrival times.
2. Choose the next unselected node with the shortest arrival time.
3. For each unselected neighbor compute arrival time. Store arrival times in the sorted list of arrival times.
4. Repeat from 2 until all nodes selected (or another stop criteria).

46

Calculating arrival time

- Uses arrival time of selected nodes (upwind scheme)
- Use hyperbolic conservation laws =>

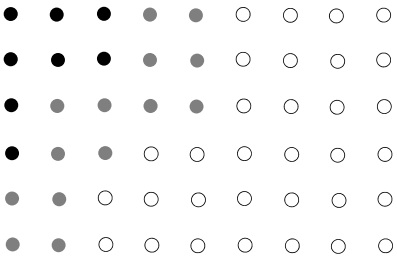
$$|\nabla T|^2=\begin{cases}\max(V_A-V_B,V_A-V_C,0)^2+\\ \max(V_A-V_D,V_A-V_E,0)^2+\\ \max(V_A-V_F,V_A-V_G,0)^2\end{cases}$$

where V is distance for nodes, and V_A is sought.

- Solve by 1) ignoring unselected nodes, 2) reduce by looking at $V_B<V_C$?, $V_D<V_E$?, $V_F<V_G$?
- Remains a second order equation to solve where largest value is taken (if two solutions exist). Ex $(V_A-V_B)^2+(V_A-V_E)^2=F^2$
- Information flows from selected nodes ("upwind") to unselected nodes ("downwind", and towards higher arrival times).

47

Fast numerical solution to initial value formulation – Narrow band



Solve equation in a narrow band (grey pixels) close to the zero level set. Fastmarching can be used for initialization

48

Time complexity

- Dijkstra's $O(E+N \log N)$ where E is the number of edges in the graph and N is the number of nodes.
- Fast marching $O(N \log N)$ where N is the the number of nodes.
- This assumes heap implementation for managing the sorted list.

49

Take home messages

- Dijkstra's solves shortest path in an optimal sense.
- Boundary formulation / initial value formulation.
- Fast numeric schemes using sorted heap.
- Trick is selecting good speed maps and finding stop criteria.

50

Short about hand-in assignment

- DICOM reader
- Display medical images
- Fastmarching in 2D and 3D

51