Image Scissoring

Computational Photography (CSCI 3240U)

Faisal Z. Qureshi
http://vclab.science.ontariotechu.ca

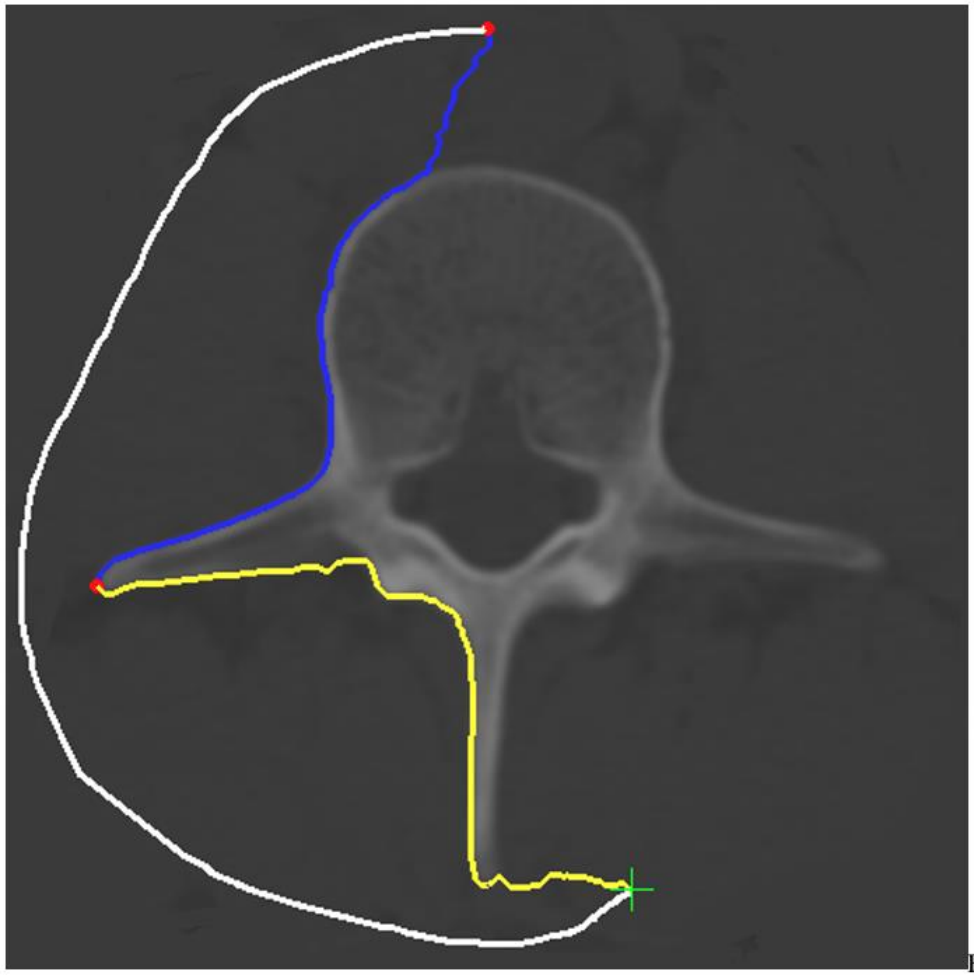
OntarioTech UNIVERSITY

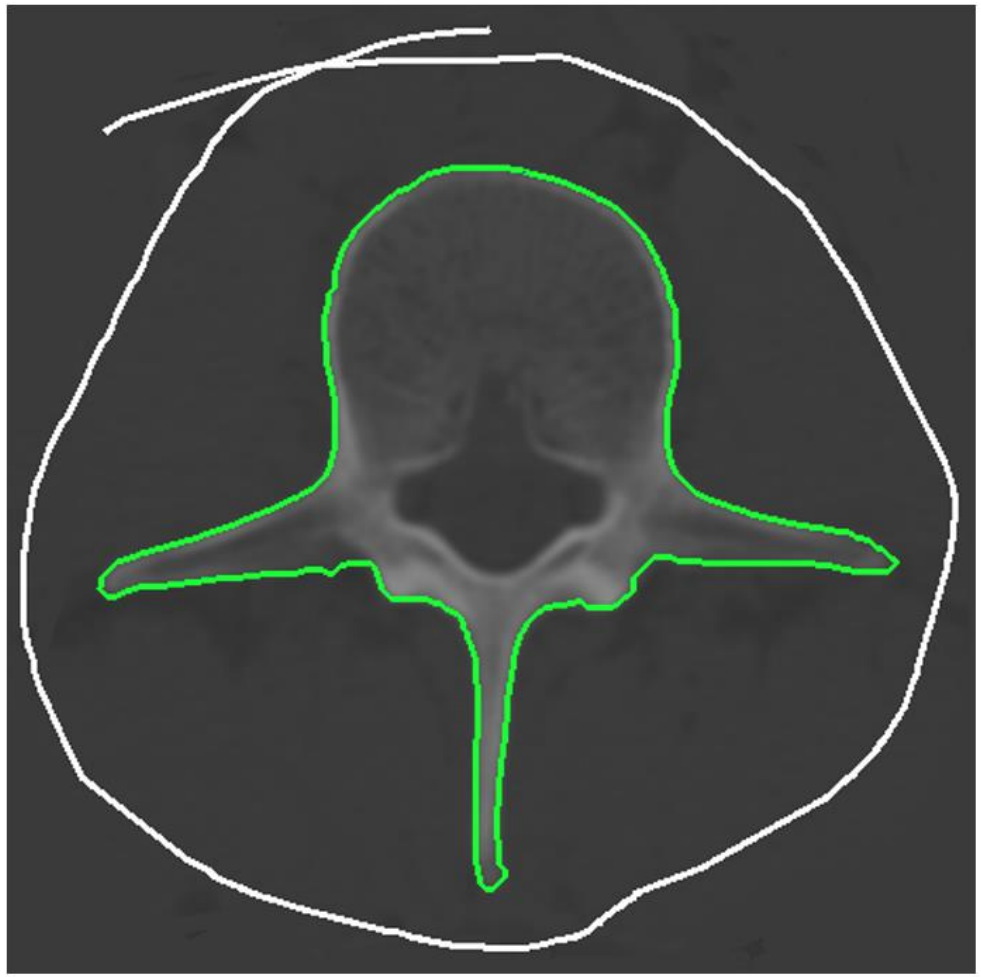
Requirements

- Interactive
  - Fast
- “User” is always right
  - A user is allowed to select arbitrary regions in an image
- Simple and easy to use

Livewire (Barrett and Mortensen, SIGGRAPH 1995)

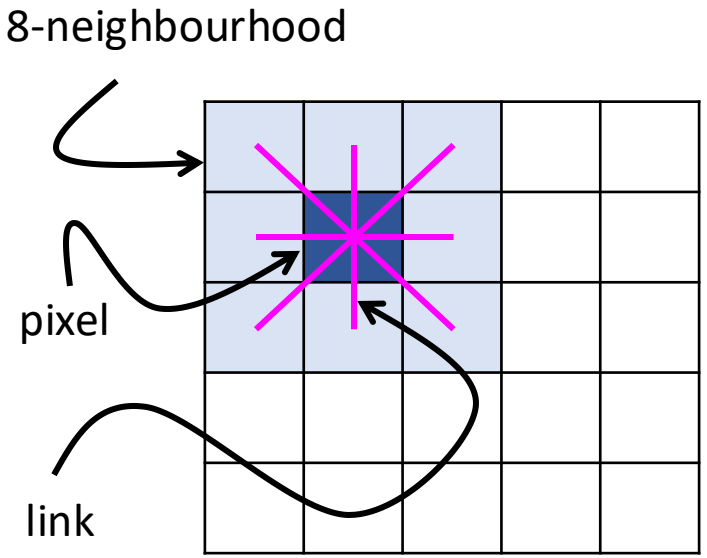
Interactive Boundary Extraction

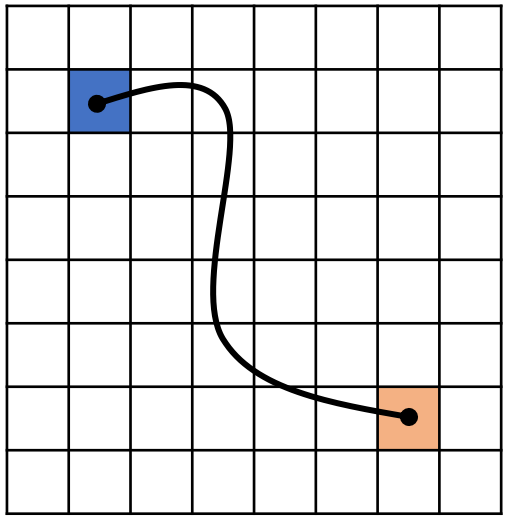




Basic idea

- Every pair of neighboring pixels defines a *link*
  - Each link is assigned a weight
  - Links along object boundaries are assigned lower weights
- User specifies a seed point
- Select a minimum weight path (comprising links) between the seed point and the current mouse location
  - Path is defined as a sequence of adjacent pixels





Link weights

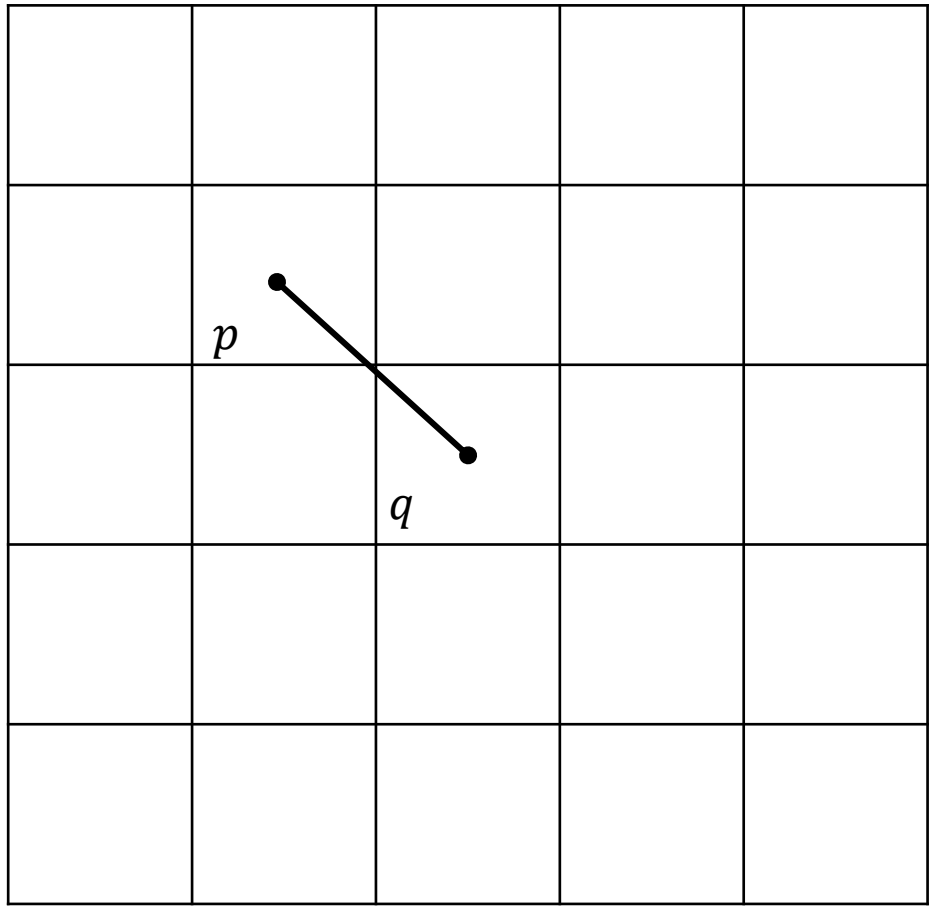
Computing weight of the link between pixels p and q

$$l(p,q) = 0.43 f_Z(q) + 0.43 f_D(p,q) + 0.14 f_G(q)$$

0 if the Laplacian has a zero crossing at q, 1 otherwise

The term that penalizes links not consistent with gradient direction at p and q

largest value over the image



Link weights: f_G

Recall

Edge pixels have high 1st derivative

Gradient: $|\nabla I| = |(I_x, I_y)| = \sqrt{I_x^2 + I_y^2}$

$$f_G(q) = 1 - \frac{|\nabla I|}{\max(|\nabla I|)}$$

- High gradients produce low costs
- Scaled by the largest gradient in the image
  - So lies between 0 and 1

$I_x =$

-1	0	1
-2	0	2
-1	0	1

$I_y =$

-1	-2	-1
0	0	0
1	2	1

Kernels for computing image derivatives

Link weights: f_Z

Recall

Can detect edges where the second derivative is 0 (inflection points)

Find zero crossings (sign change w.r.t. the 8-meighbouts)

$$L = I_{xx} + I_{yy}$$

$f_Z(q) = 0$ if the Laplacian has a zero crossing at q

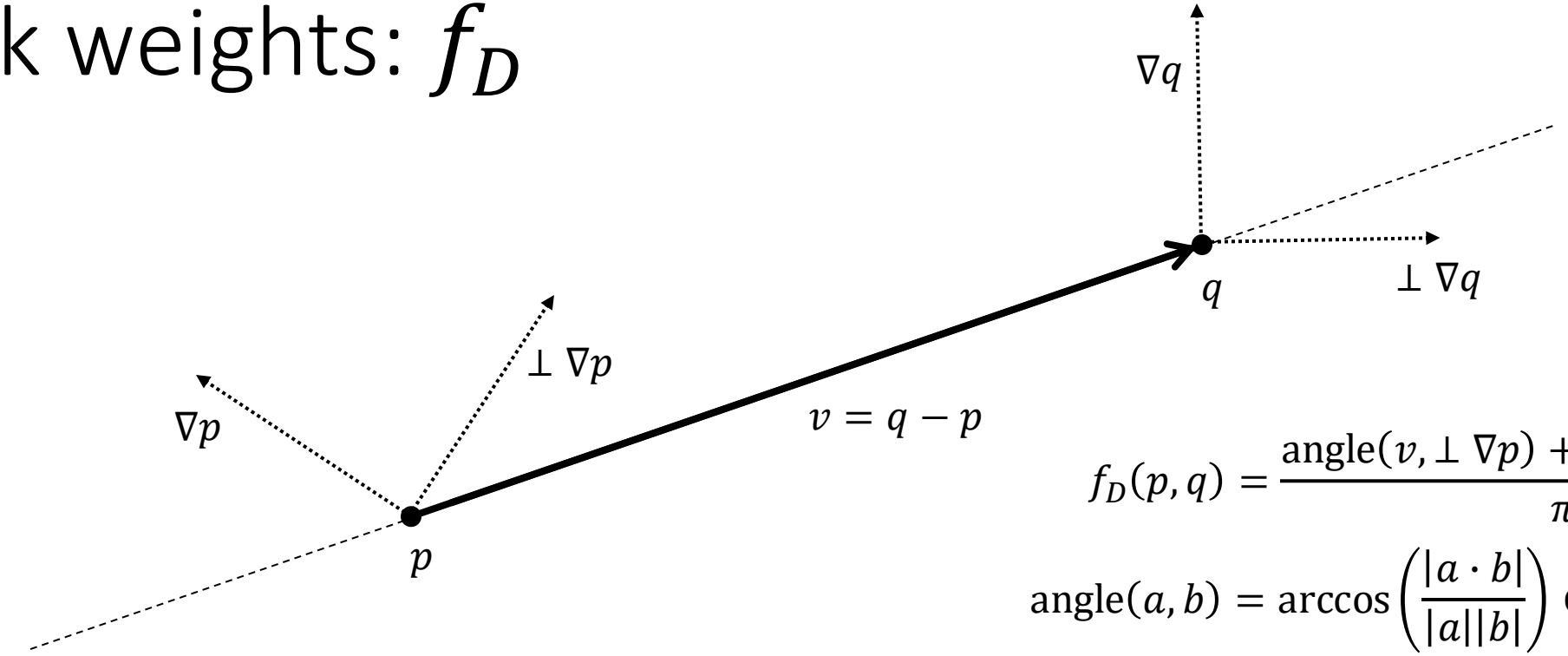
$f_Z(q) = 1$ otherwise

- Zero crossings produce low weights
- Keep in mind that many zero crossings are due to noise

0	1	0
1	-4	1
0	1	0

Kernel that approximates the Laplacian

Link weights: f_D



$$f_D(p,q) = \frac{\text{angle}(v, \perp \nabla p) + \text{angle}(v, \perp \nabla q)}{\pi}$$

$$\text{angle}(a,b) = \arccos\left(\frac{|a \cdot b|}{|a||b|}\right) \in [0, \pi/2]$$

- Penalizes sharp change in path direction (creases)
- Penalizes paths that do not follow edges in the image
- Normalized to lie between 0 and 1

Path optimization

- Formulated as a graph search problem that computes the minimum-cost path from seed to all other image pixels
  - Use Dijkstra's Algorithm

Path optimization

Input:

seed (start pixel)

graph (edge-weighted graph corresponding to the local cost $l(p, q)$)

Output:

A tree on graph, where each node is pointing to its successor along the minimum cost path from that node to the seed (root)

Comments:

Each node can have one of three states: *initial*, *active*, *expanded*

The algorithm ends when all nodes have been *expanded*

Active nodes are kept in PQ, ordered by the total path cost from the node to the seed

Path optimization

Initialization:

add the seed node s to the Priority Queue (PQ), and set it to *active*
set cost$(s) = 0$
set all other nodes to *initial*

Loop: *dynamic programming algorithm O(n)*
while PQ is not empty

remove from PQ, node q with minimum path cost; recall that cost of q is cost(q)

mark q as *expanded*

for each node r that is 8-neighbour of q

if r is *initial*

insert r in PQ
set cost$(r) = q + l(q, r)$
mark r as *active*

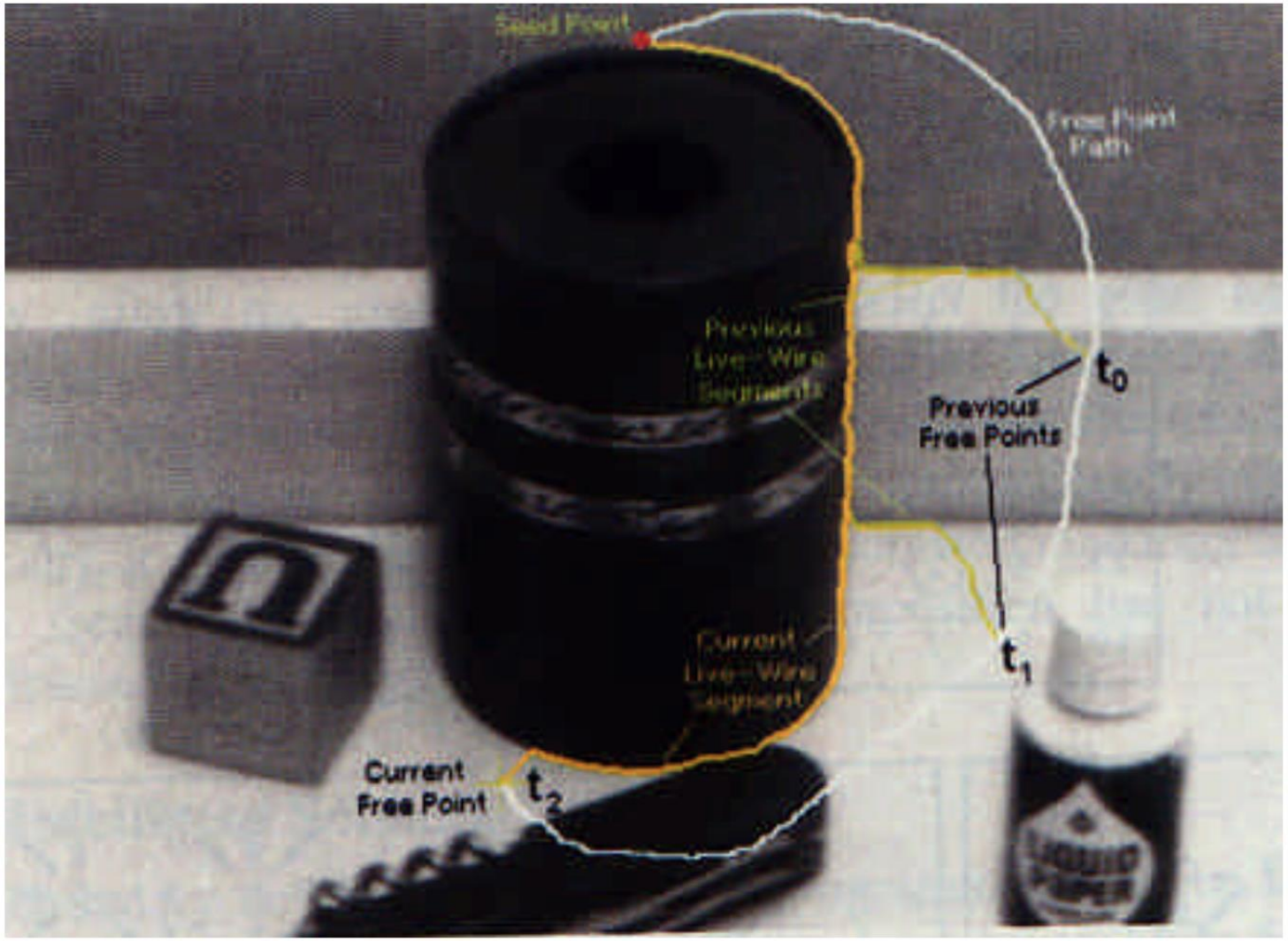
else if r is *active* (i.e., it is in PQ)

set cost$(r) = \min(\text{cost}(r), q + l(q, r))$

else

pass

Intelligent scissors



Summary

- Intelligent scissors
  - An example of interactive boundary discovery
- Images as graphs
- Boundary detection as path finding in weighted graphs
- Another example where gradient and Laplacian are used to identify edge pixels