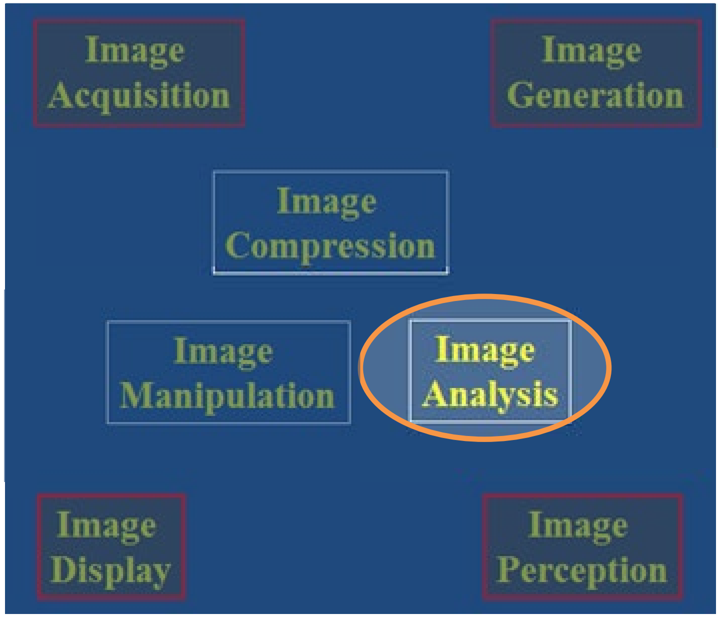
# Digital Image Processing

## Edge Detection: Smart Scissors




DRAFT

Brent M. Dingle, Ph.D.
2015
Game Design and Development Program
Mathematics, Statistics and Computer Science
University of Wisconsin - Stout

Material in this presentation is largely based on/derived from presentations by: Noah Snavely

# Lecture Objectives

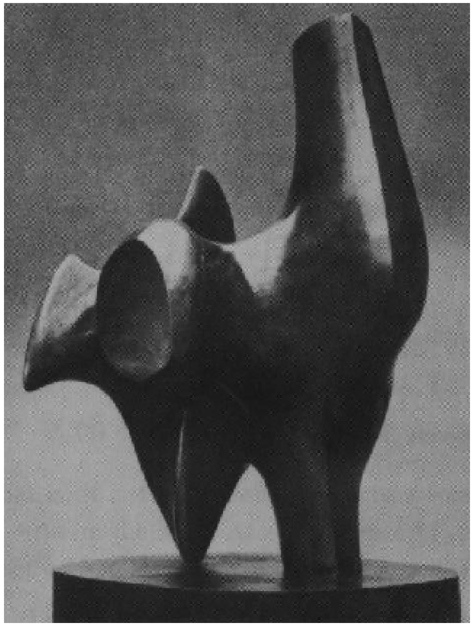
- Previously
  - Filtering
  - Interpolation
  - Warping
  - Morphing
  - Compression
  - Edge Detection
- Today
  - Intelligent Scissors

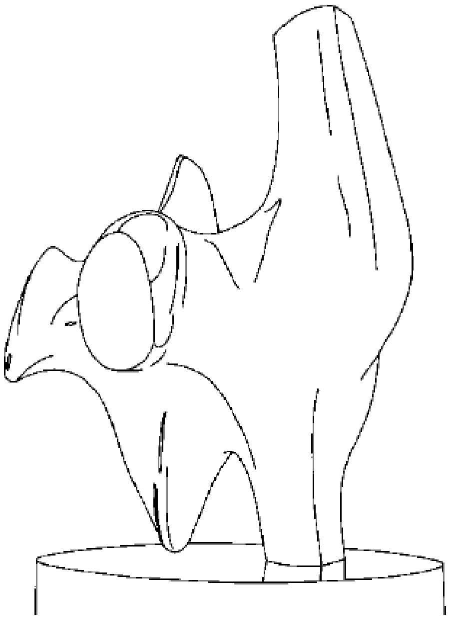
**Image Manipulation and Enhancement** (Filtering, Interpolation, Warping, Morphing)

**Image Compression** (Compression)

**Image Analysis** (Edge Detection, Intelligent Scissors)

# Recall: Edge Detection





- **Goal**: Identify sudden changes (discontinuities) in an image
  - Much of the 'shape' information of the image can be realized in the edges
- **Ideal Result**: an artist's line drawing
  - *aside: who can use knowledge beyond a single image*

contour line drawing image from: https://www.youtube.com/watch?v=Eu9sUWLfGng

# Recall: Canny Edge Detector

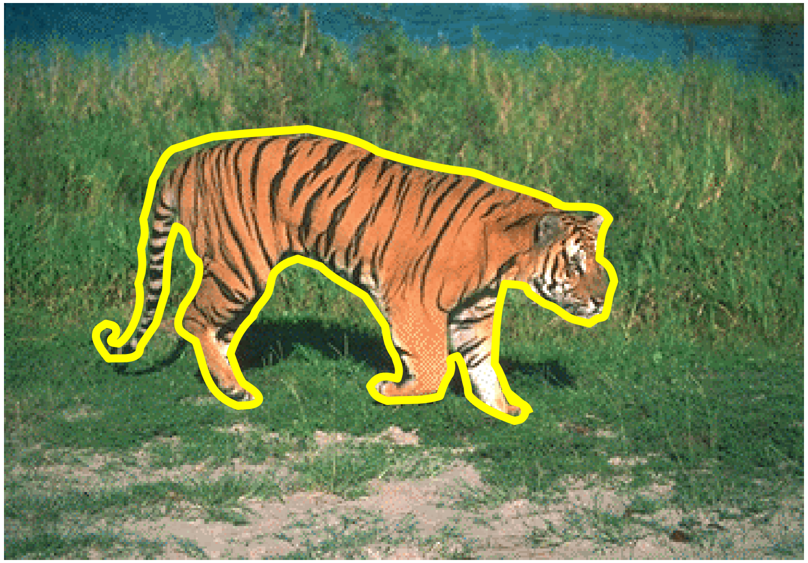
- Filter image with derivative of Gaussian

- Find magnitude and orientation of gradient

- Apply non-maximum suppression
- Linking and thresholding (hysteresis)
  - Define 2 thresholds: low and high
  - Use the high threshold to start edge curves and the low threshold to continue them

source: L. Fei-Fei, D. Lowe

# Extracting Objects



- How can the tiger be extracted from the image?
  - Difficult to do manually
  - Difficult to do automatically (but possible)
  - Easier to do semi-automatically

source: Noah Snavely

# Paper to Read/Reference

- Intelligent Scissors,
  Mortensen et. al, SIGGRAPH 1995

source: Noah Snavely

# Intelligent Scissors

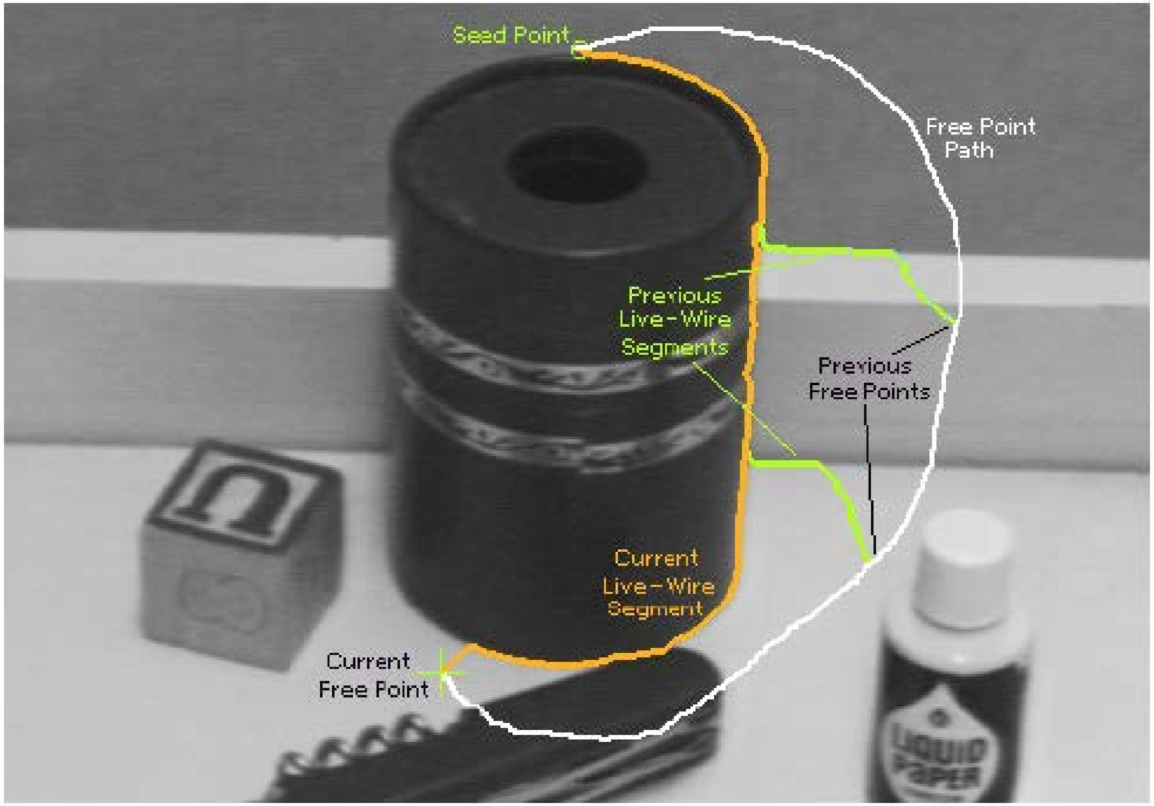


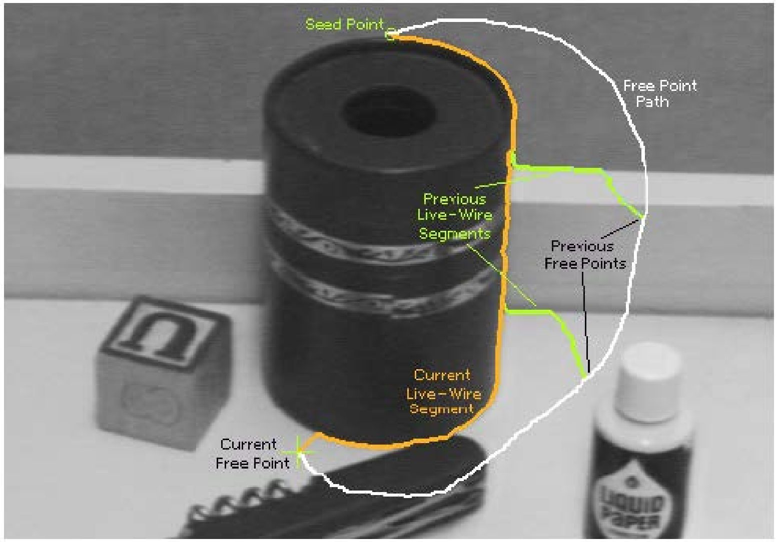

**Figure 2:** *Image demonstrating how the live-wire segment adapts and snaps to an object boundary as the free point moves (via cursor movement). The path of the free point is shown in white. Live-wire segments from previous free point positions ($t_0$, $t_1$, and $t_2$) are shown in green.*

source: Noah Snavely

# Intelligent Scissors

- **Question**
  - How do we find a path from seed to mouse that follows an object boundary as closely as possible?

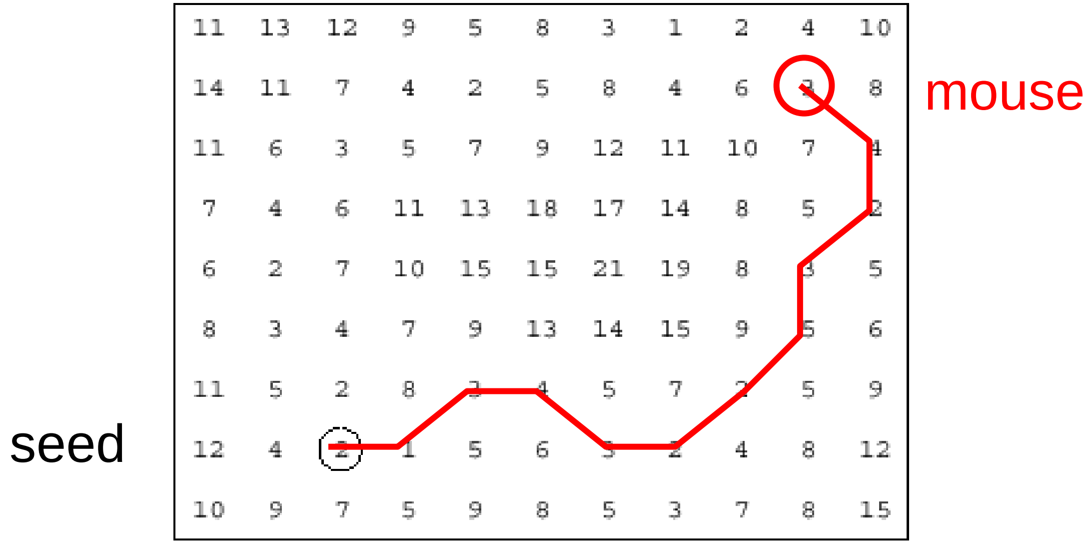
- **Answer**
  - Define a path that stays as close as possible to edges




source: Noah Snavely

# Intelligent Scissors

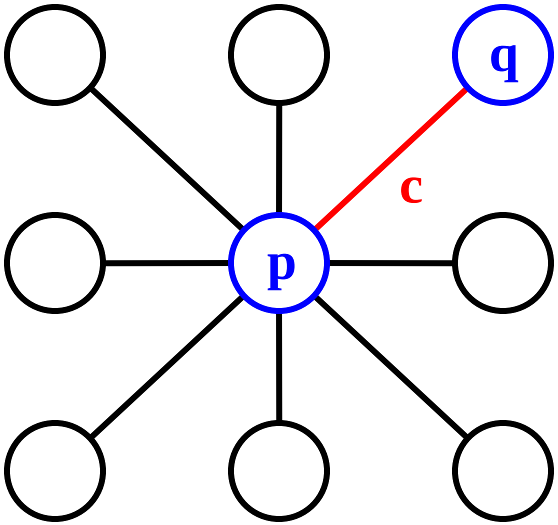
- Basic Idea
  - Define edge score for each pixel
    - assign edge pixels a low score
  - Find lowest cost path from seed to mouse




source: Noah Snavely

# Let’s look at this more closely

- Treat the image as a graph




Graph

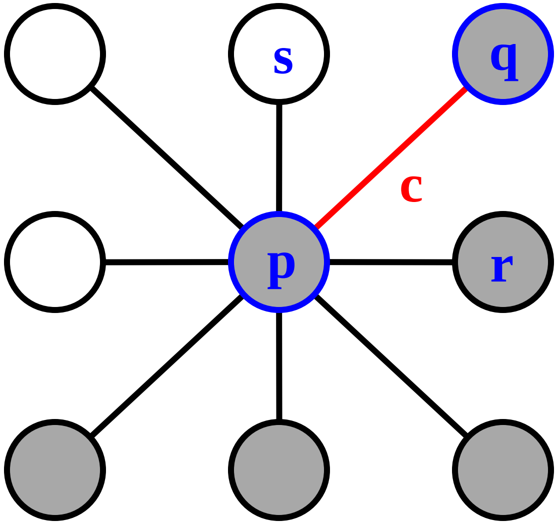
- node for every pixel **p**
- link between every adjacent pair of pixels, **p,q**
- cost **c** for each link

Note:  each *link* has a cost

- this is a little different than the figure before where each pixel had a cost

source: Noah Snavely

# Defining the costs



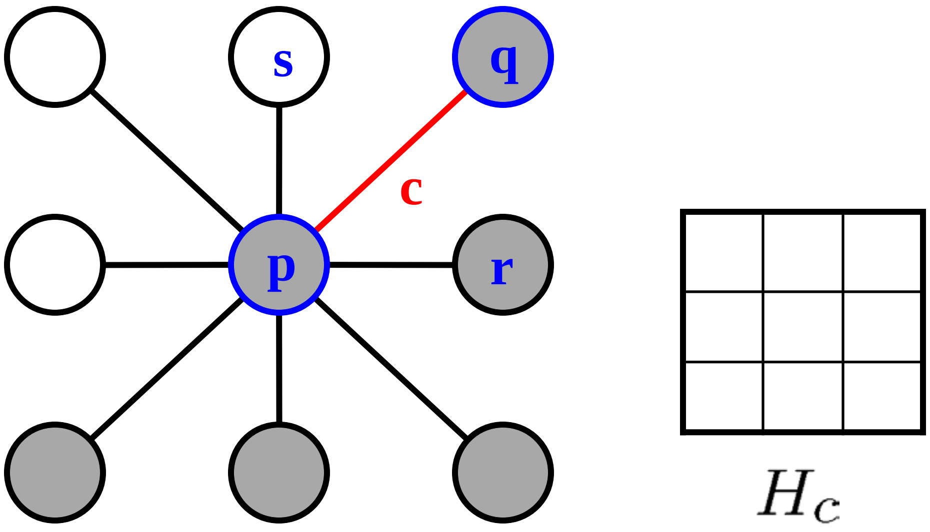

Want to hug image edges: how to define cost of a link?

- good (low-cost) links follow the intensity edge
– want intensity to change rapidly ⊥ to the link

- $c \approx -\frac{1}{\sqrt{2}}$ |intensity of r – intensity of s|

source: Noah Snavely

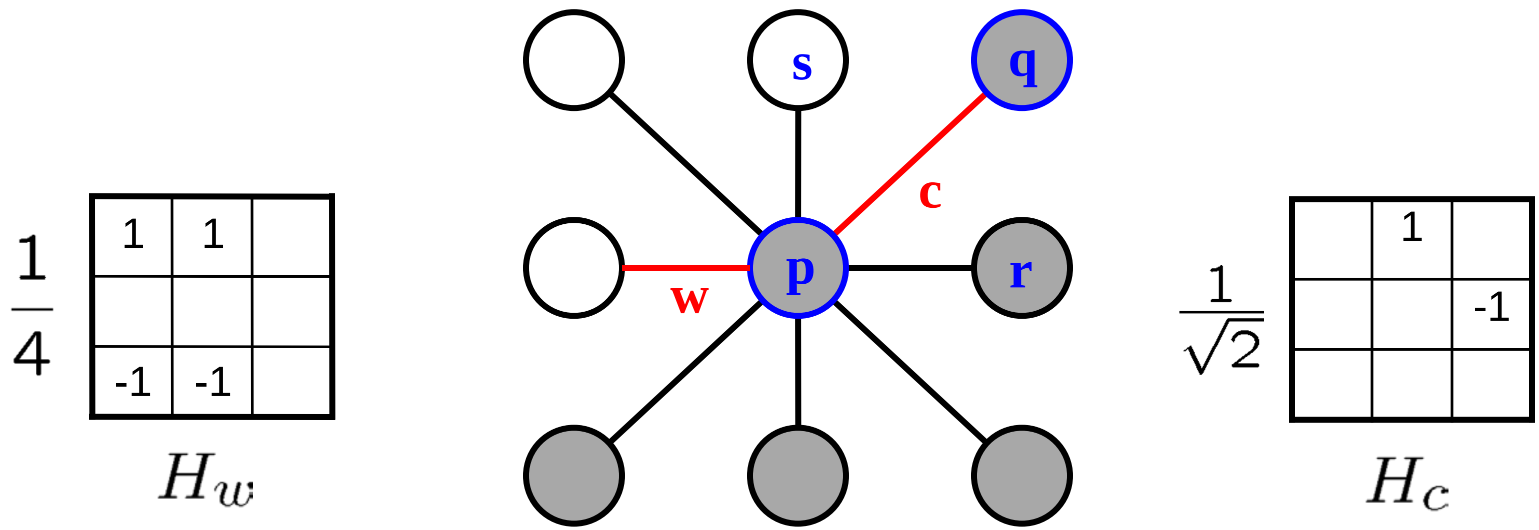
# Defining the costs




- c can be computed using a cross-correlation filter
  - assume it is centered at p

source: Noah Snavely

# Defining the costs




c can be computed using a cross-correlation filter

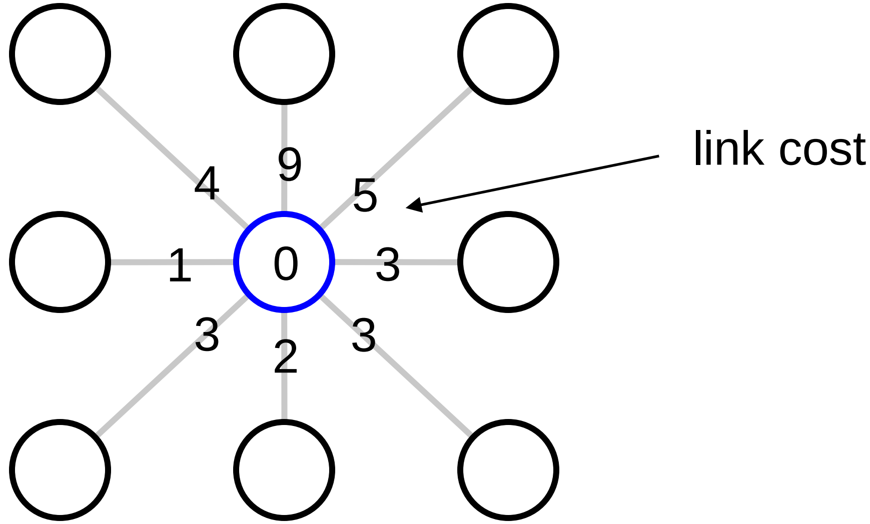
- assume it is centered at p

A couple more modifications

- Scale the filter response by length of link c. Why?
- Make c positive
  - Set c = (max-|filter response|*length)
  - where max = maximum |filter response|*length over all pixels in the image

source: Noah Snavely

# Dijkstra's shortest path algorithm




Algorithm

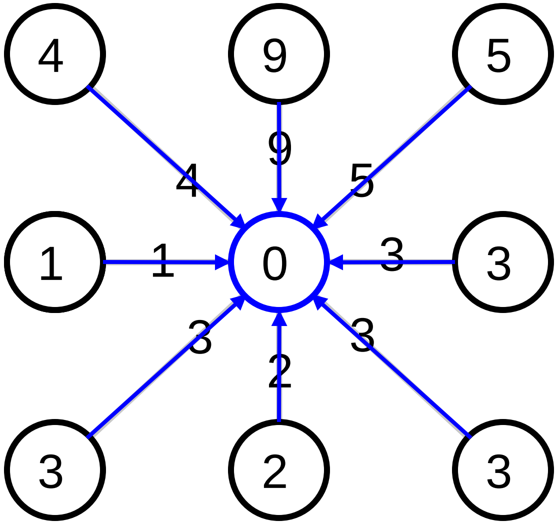
1. init node costs to $\infty$, set p = seed point, cost(p) = 0
2. expand p as follows:

   for each of p's neighbors q that are not expanded

   set cost(q) = min( cost(p) + $c_{pq}$, cost(q) )

source: Noah Snavely

# Dijkstra's shortest path algorithm




Algorithm

1. init node costs to $\infty$, set p = seed point, cost(p) = 0
2. expand p as follows:

   for each of p's neighbors q that are not expanded

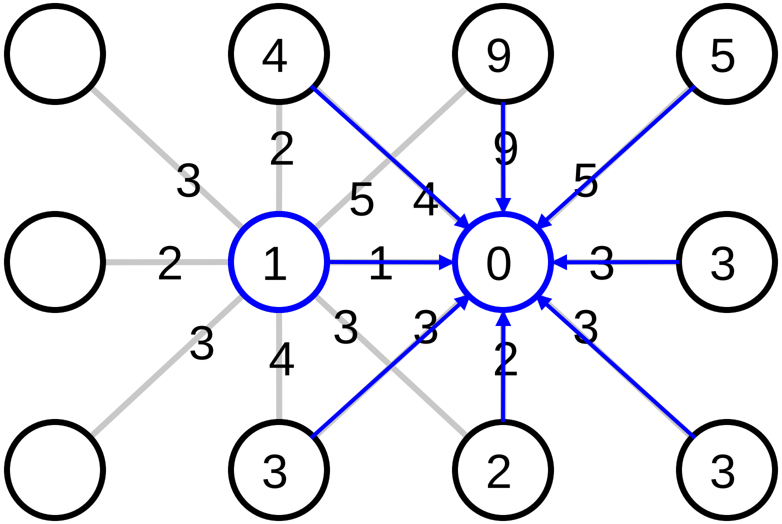
   set cost(q) = min( cost(p) + $c_{pq}$, cost(q) )

   if q's cost changed, make q point back to p

   put q on the ACTIVE list (if not already there)

source: Noah Snavely

# Dijkstra's shortest path algorithm




Algorithm

1. init node costs to $\infty$, set p = seed point, cost(p) = 0
2. expand p as follows:

   for each of p's neighbors q that are not expanded

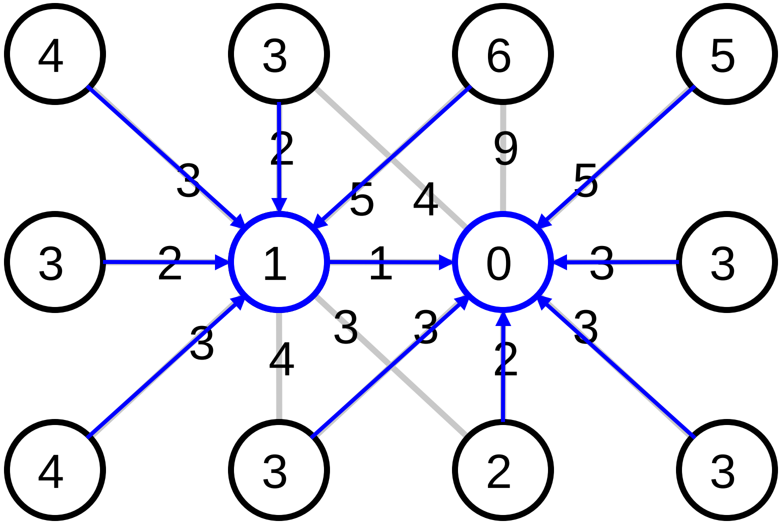
   set cost(q) = min( cost(p) + $c_{pq}$, cost(q) )

   if q's cost changed, make q point back to p

   put q on the ACTIVE list (if not already there)
3. set r = node with minimum cost on the ACTIVE list
4. repeat Step 2 for p = r

source: Noah Snavely

# Dijkstra's shortest path algorithm




Algorithm

1. init node costs to $\infty$, set p = seed point, cost(p) = 0
2. expand p as follows:

   for each of p's neighbors q that are not expanded

      set cost(q) = min( cost(p) + $c_{pq}$, cost(q) )

      if q's cost changed, make q point back to p

      put q on the ACTIVE list (if not already there)

3. set r = node with minimum cost on the ACTIVE list
4. repeat Step 2 for p = r

source: Noah Snavely

# Dijkstra’s shortest path algorithm

- Properties
  - It computes the minimum cost path from the seed to every node in the graph. This set of minimum paths is represented as a *tree*
  - Running time, with N pixels:
    - $O(N^2)$ time if you use an active list
    - O(N log N) if you use an active priority queue (heap)
    - takes fraction of a second for a typical (640x480) image
  - Once this tree is computed once, we can extract the optimal path from any point to the seed in O(N) time.
    - it runs in real time as the mouse moves
  - What happens when the user specifies a new seed?

source: Noah Snavely

# Questions?

- Beyond D2L
  - Examples and information can be found online at:
    - *http://docdingle.com/teaching/cs.html*

    - *Continue to more stuff as needed*

# Extra Reference Stuff Follows

# Credits

- Much of the content derived/based on slides for use with the book:
  - *Digital Image Processing,* Gonzalez and Woods
- Some layout and presentation style derived/based on presentations by
  - Donald House, Texas A&M University, 1999
  - Sventlana Lazebnik, UNC, 2010
  - Noah Snavely, Cornell University, 2012
  - Xin Li, WVU, 2014
  - George Wolberg, City College of New York, 2015
  - Yao Wang and Zhu Liu, NYU-Poly, 2015