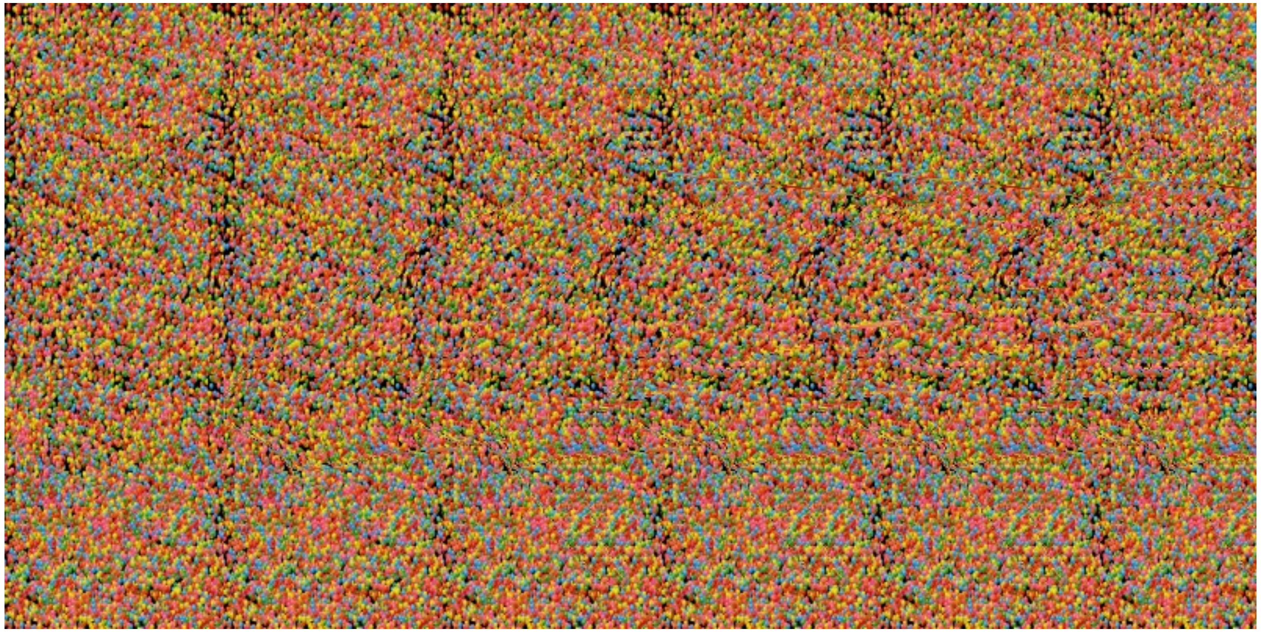
Folien zur Vorlesung am 29.04.2025

3D Computer Vision

DAS PRINZIP DER STEREOREKONSTRUKTION

Binocular Stereo

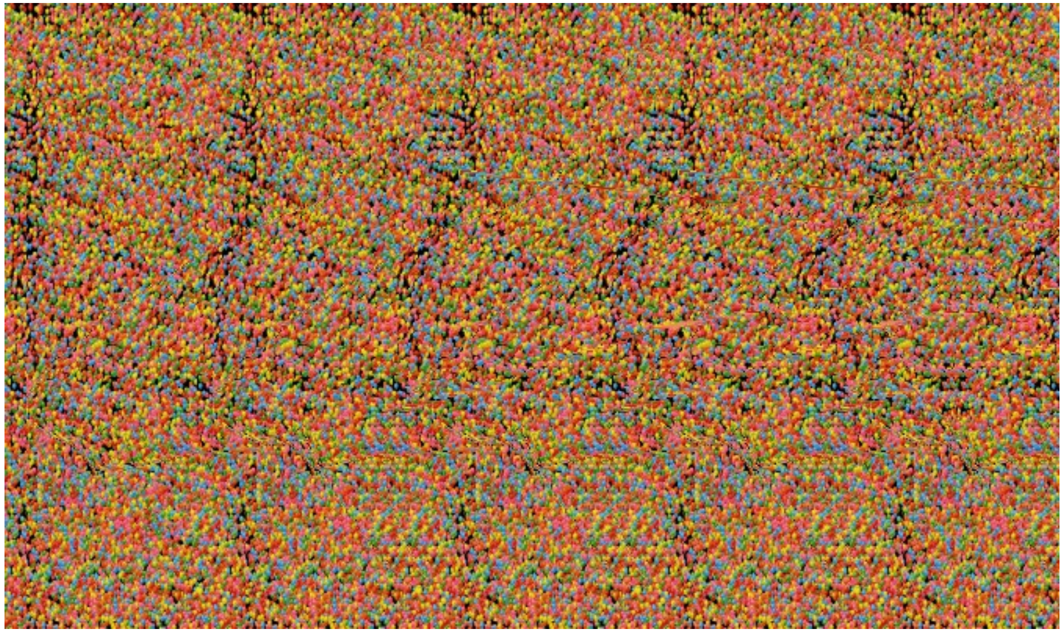
What is this?



Single image stereogram,
https://en.wikipedia.org/wiki/Autostereogram

Binocular Stereo

What is this?



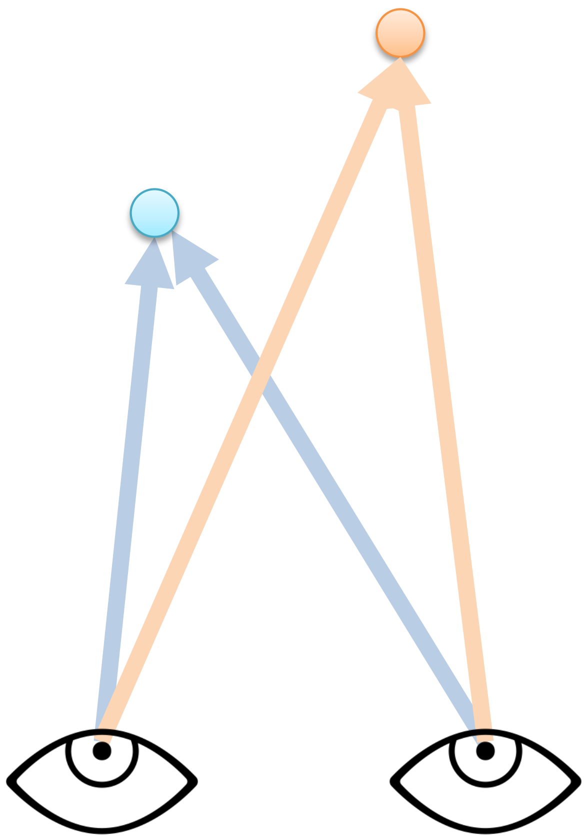
Single image stereogram,
https://en.wikipedia.org/wiki/Autostereogram

https://giphy.com/gifs/wigglegram-706pNfSKyaDug

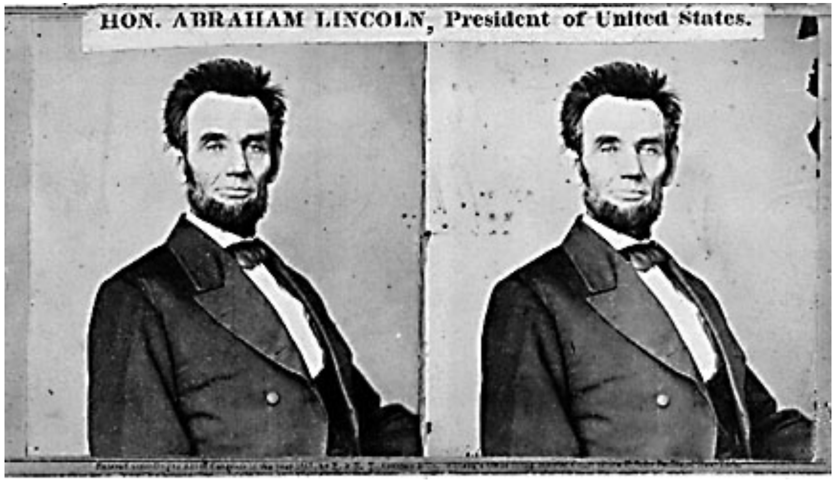
“Mark Twain at Pool Table", no date, UCR Museum of Photography

Stereo Vision as Localizing Points in 3D

- An object point will project to some point in our image
- That image point corresponds to a ray in the world
- Two rays intersect at a single point, so if we want to localize points in 3D we need 2 eyes

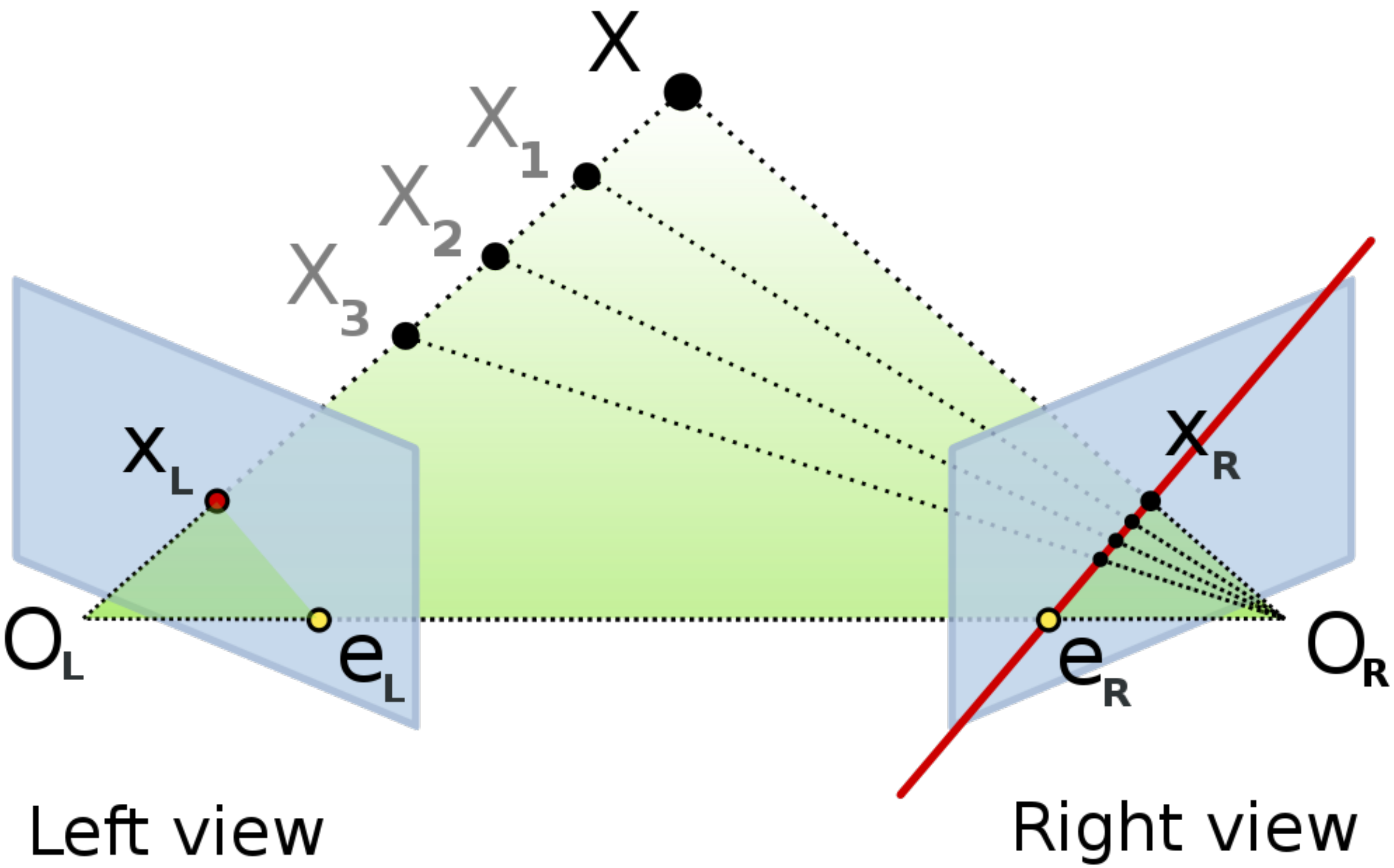


Stereo



- Given two images from different viewpoints
  - How can we compute the depth of each point in the image?
  - Based on *how much each pixel moves* between the two images

Epipolar geometry



Epipolar geometry - Rectification

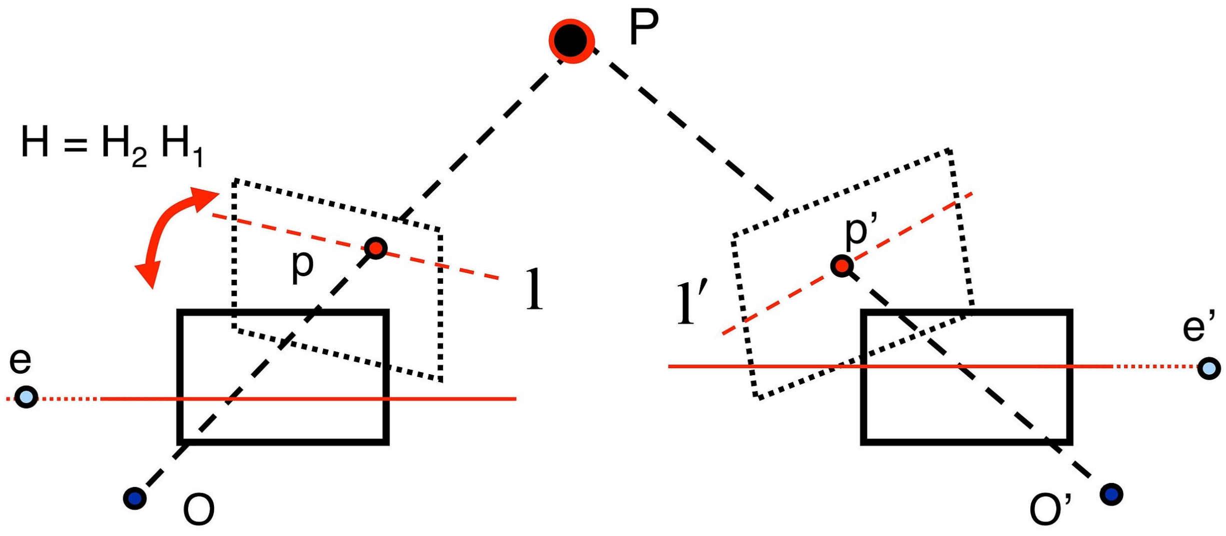
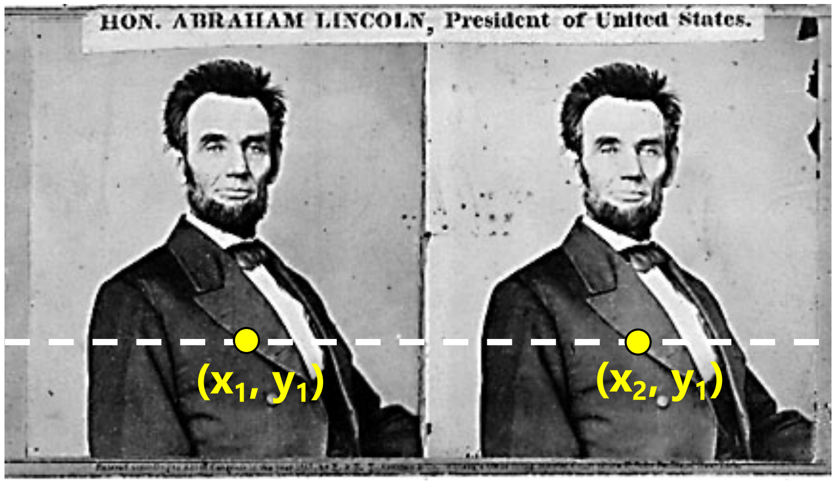


Image from Silvio Savarese, Public domain, via Wikimedia Commons

Epipolar geometry

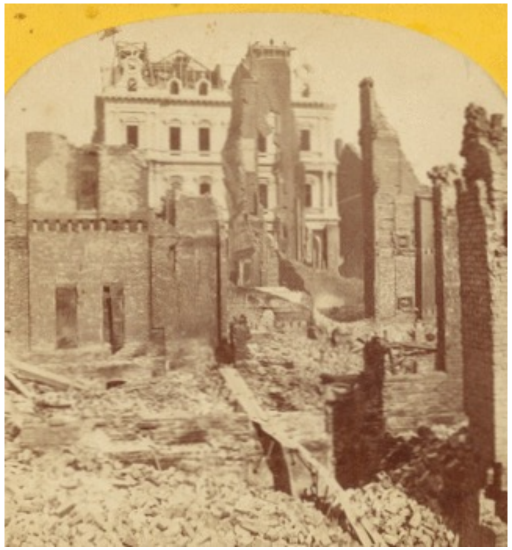
epipolar
lines



Two images captured by a purely horizontal translating camera
(*rectified* stereo pair)

$x_2 - x_1$ = the *disparity* of pixel (x_1, y_1)

Disparity = inverse depth



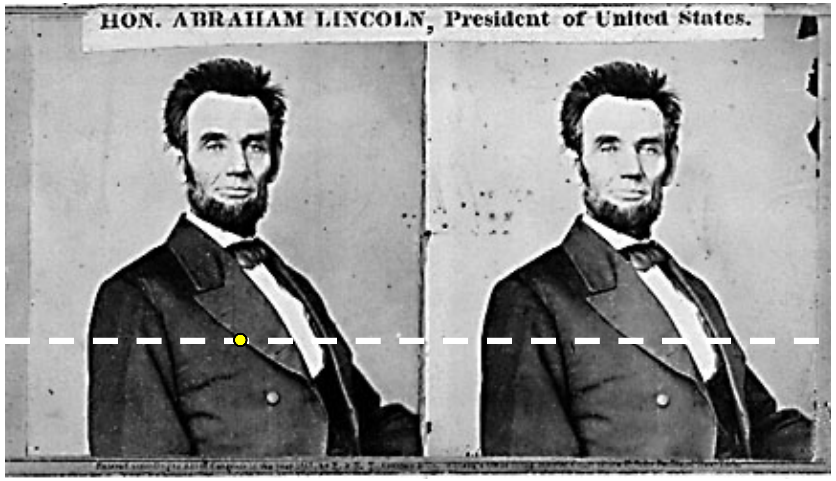
http://stereo.nypl.org/view/41729

(Or, hold a finger in front of your face and wink each eye in succession.)

Your basic stereo matching algorithm

- **Match Pixels in Conjugate Epipolar Lines**
  - Assume brightness constancy
  - This is a challenging problem
  - Hundreds of approaches
    - A good survey and evaluation: vision.middlebury.edu/stereo/
    - A newer evaluation approach: https://www.eth3d.net/low_res_two_view
  - Science community uses fixed data sets for evaluation:
    - https://www.eth3d.net/datasets
    - https://vision.middlebury.edu/stereo/data/

Your basic stereo matching algorithm

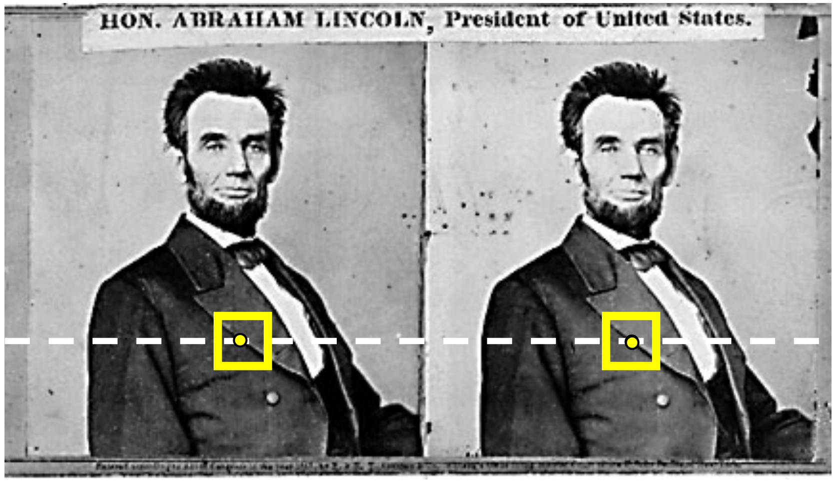


For each epipolar line

For each pixel in the left image

- compare with every pixel on same epipolar line in right image

Your basic stereo matching algorithm



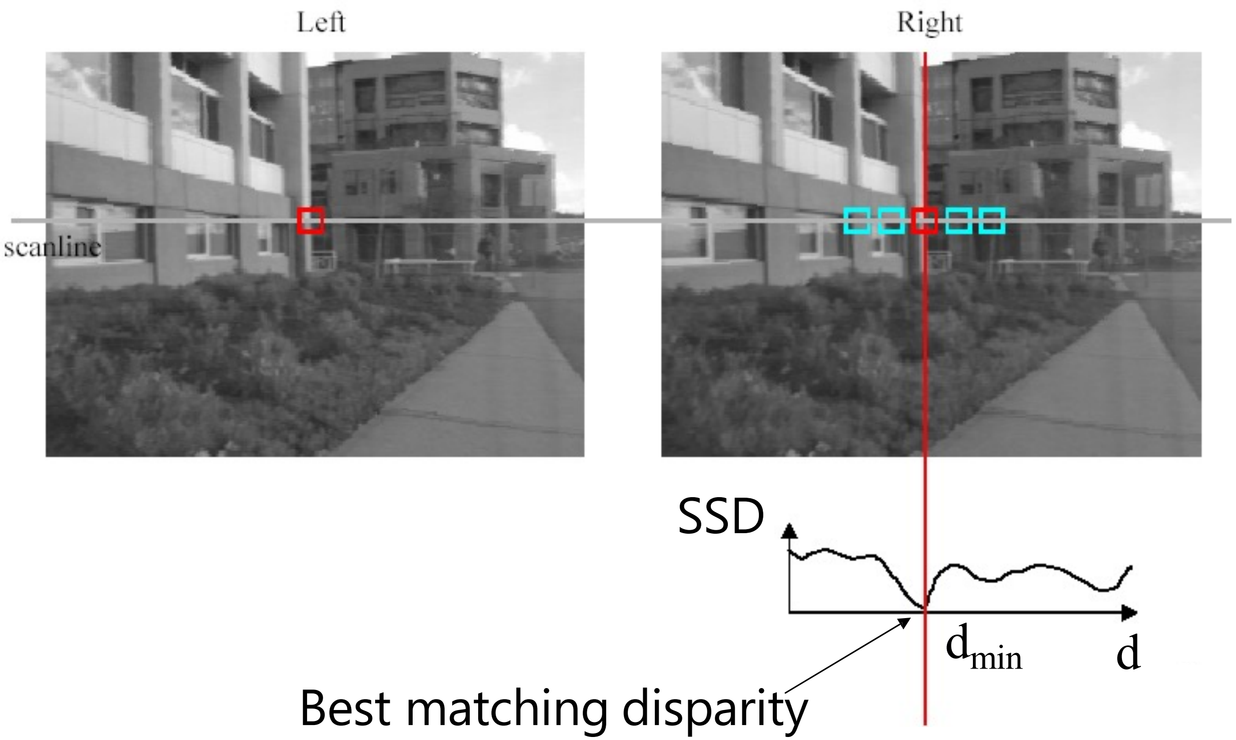
For each epipolar line

For each pixel in the left image

- compare with every pixel on same epipolar line in right image
- pick pixel with minimum match cost

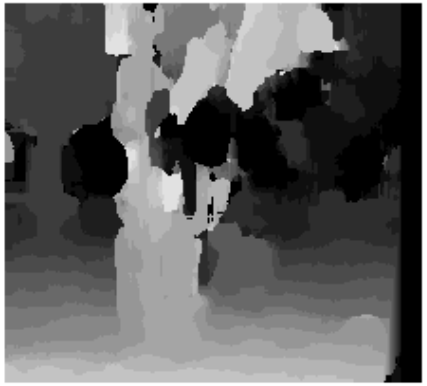
Improvement: match ***windows***

Stereo matching based on SSD



Window size

W = 3



W = 20

Effect of window size

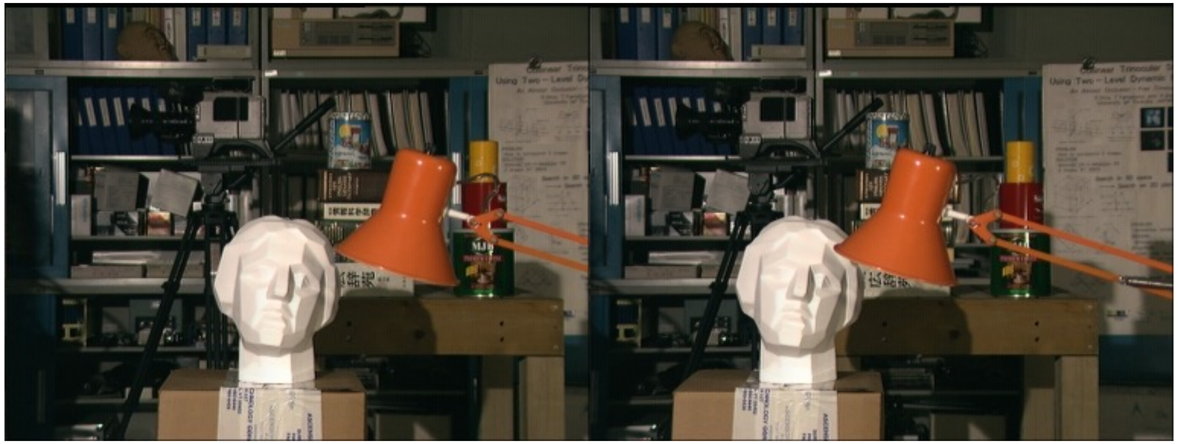
– Smaller window
  + more detail
  - more noise
– Larger window
  + less noise
  - less detail

Better results with *adaptive window*

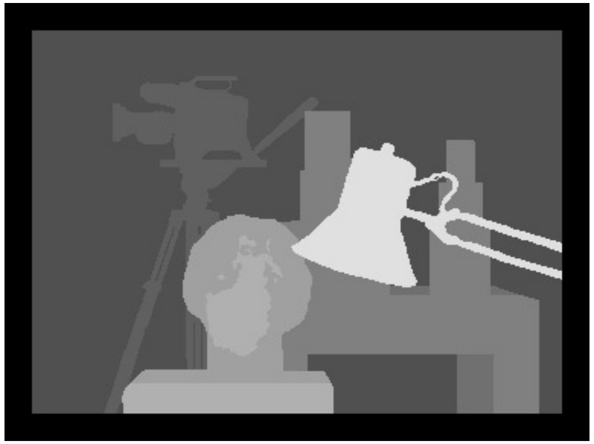
- T. Kanade and M. Okutomi, *A Stereo Matching Algorithm with an Adaptive Window: Theory and Experiment*, ICRA 1991.
- D. Scharstein and R. Szeliski. Stereo matching with nonlinear diffusion. IJCV, July 1998

Stereo results

– Data from University of Tsukuba
– Similar results on other images without ground truth

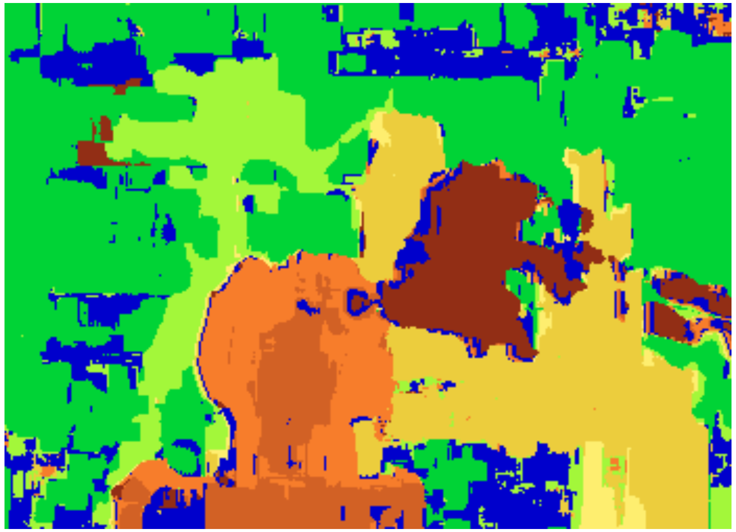


Scene left and right



Ground truth

Results with window search



Window-based matching
(best window size)

Ground truth

Better methods exist...

Graph cuts-based method

Boykov et al., Fast Approximate Energy Minimization via Graph Cuts, International Conference on Computer Vision 1999.

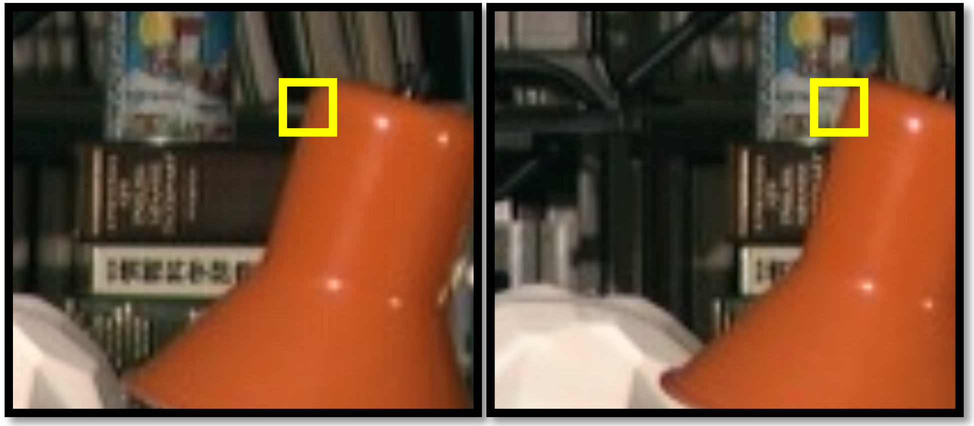
Ground truth

For the latest and greatest see http://www.middlebury.edu/stereo/ and https://www.eth3d.net/low_res_two_view

Stereo results

- Why are the wrong values where they are?

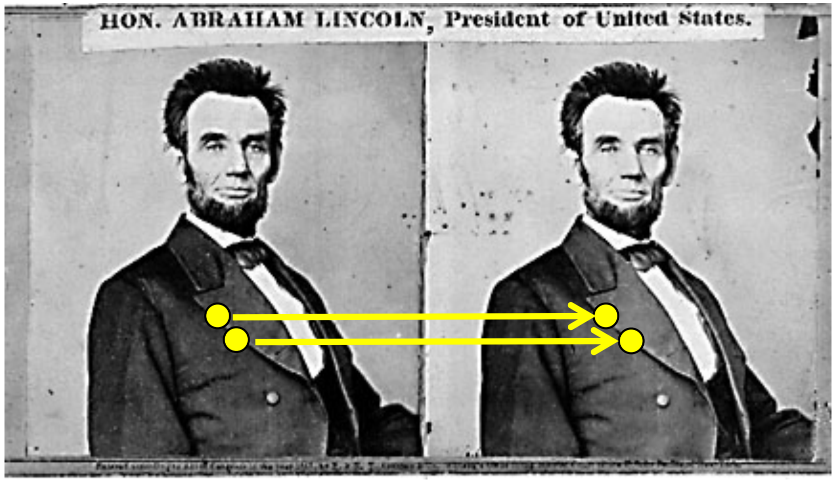
Bad pixels (absolute disparity error > 1.0)



Graph cuts-based method

Boykov et al., Fast Approximate Energy Minimization via Graph Cuts, International Conference on Computer Vision 1999.

Stereo as energy minimization



- What defines a good stereo correspondence?
  1. Match quality
     - Want each pixel to find a good match in the other image
  2. Smoothness
     - If two pixels are adjacent, they should (usually) move about the same amount

Stereo as energy minimization

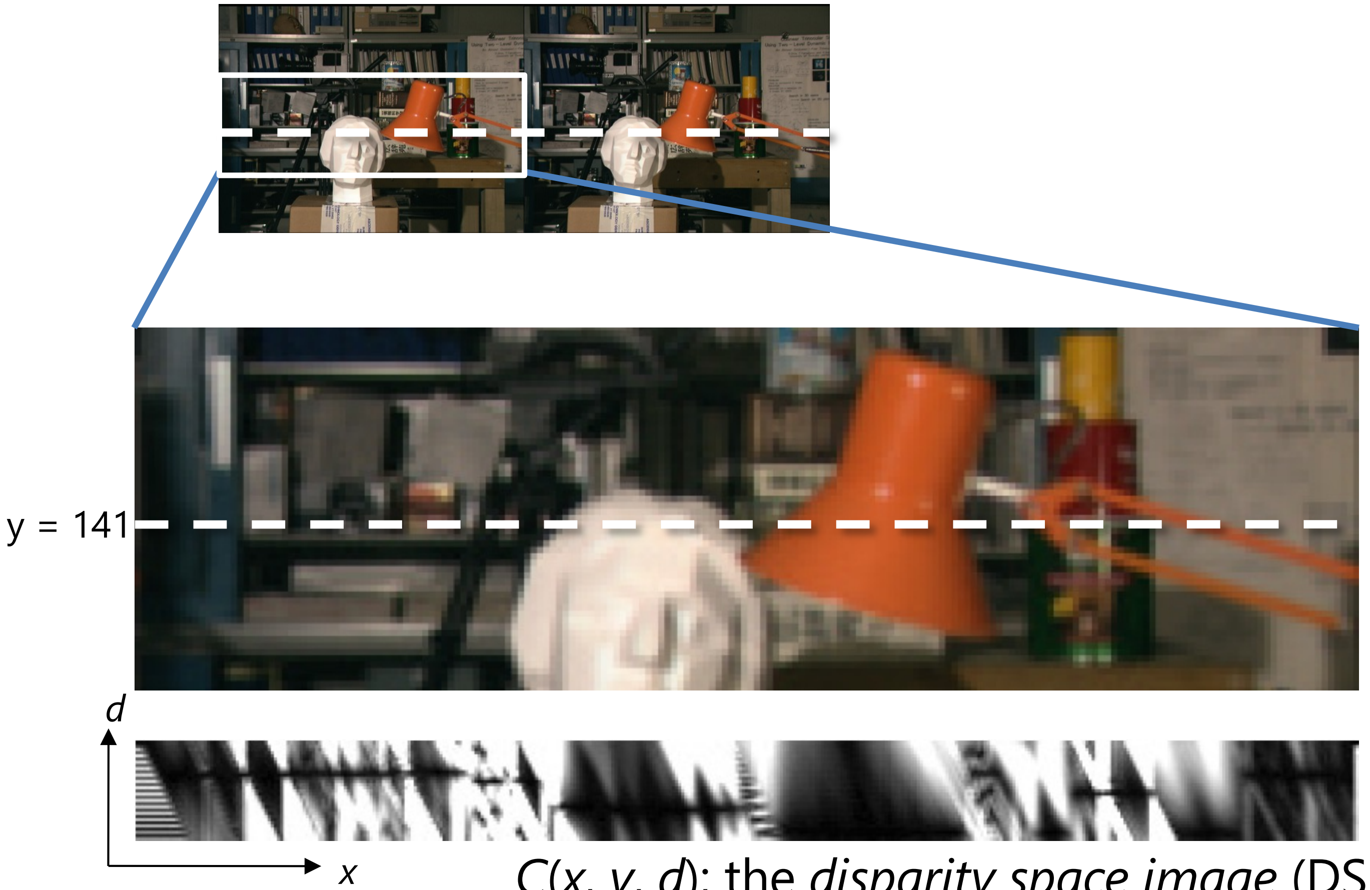
- Find disparity map *d* that minimizes an *energy function* $E(d)$

- Simple pixel / window matching

$$E(d) = \sum_{(x,y)\in I} C(x, y, d(x, y))$$

$C(x, y, d(x, y))$ = SSD distance between windows *I*(*x*, *y*) and *J*(*x* + *d*(*x*,*y*), *y*)

Stereo as energy minimization



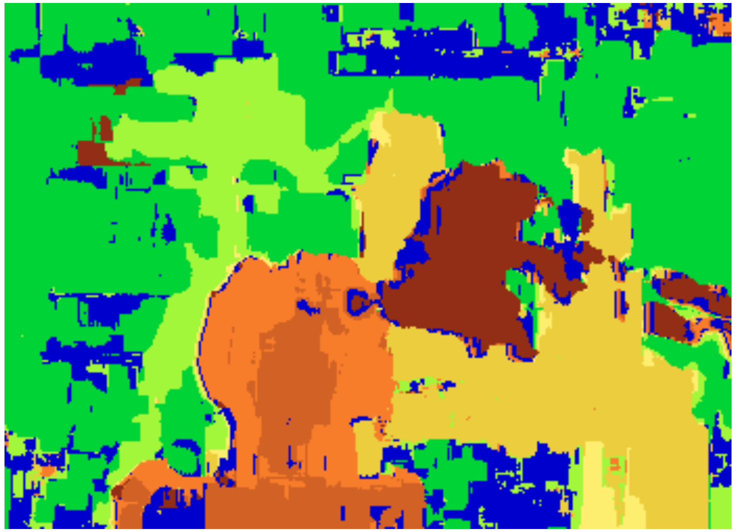
$C(x, y, d)$; the *disparity space image* (DSI)

Stereo as energy minimization

Simple pixel / window matching: choose the minimum of each column in the DSI independently:

$$d(x, y) = \arg\min_{d'} \; C(x, y, d')$$

Greedy selection of best match



Stereo as energy minimization

- Better objective function

$$E(d) = \underbrace{E_d(d)}_{\text{match cost}} + \lambda \underbrace{E_s(d)}_{\text{smoothness cost}}$$

Want each pixel to find a good match in the other image

Adjacent pixels should (usually) move about the same amount

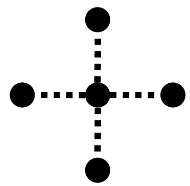
Stereo as energy minimization

$$E(d) = E_d(d) + \lambda E_s(d)$$

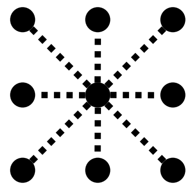
match cost: $$E_d(d) = \sum_{(x,y)\in I} C(x, y, d(x, y))$$

smoothness cost: $$E_s(d) = \sum_{(p,q)\in \mathcal{E}} V(d_p, d_q)$$

$\mathcal{E}$: set of neighboring pixels



4-connected neighborhood



8-connected neighborhood

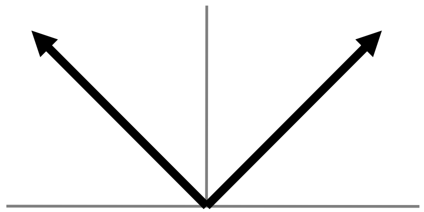
Smoothness cost

$$E_s(d) = \sum_{(p,q)\in\mathcal{E}} V(d_p, d_q)$$

How do we choose *V*?

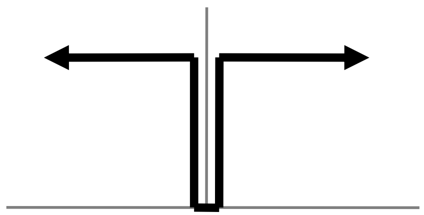
$$V(d_p, d_q) = |d_p - d_q|$$

L_1 distance



$$V(d_p, d_q) = \begin{cases} 0 & \text{if } d_p = d_q \\ 1 & \text{if } d_p \neq d_q \end{cases}$$

"Potts model"



Smoothness cost

$$E(d) = E_d(d) + \lambda E_s(d)$$

- If λ = infinity, then we only consider smoothness
- Optimal solution is a surface of constant depth/disparity
  – *Fronto-parallel* surface

- In practice, want to balance data term with smoothness term

Dynamic Programming

$$E(d) = E_d(d) + \lambda E_s(d)$$

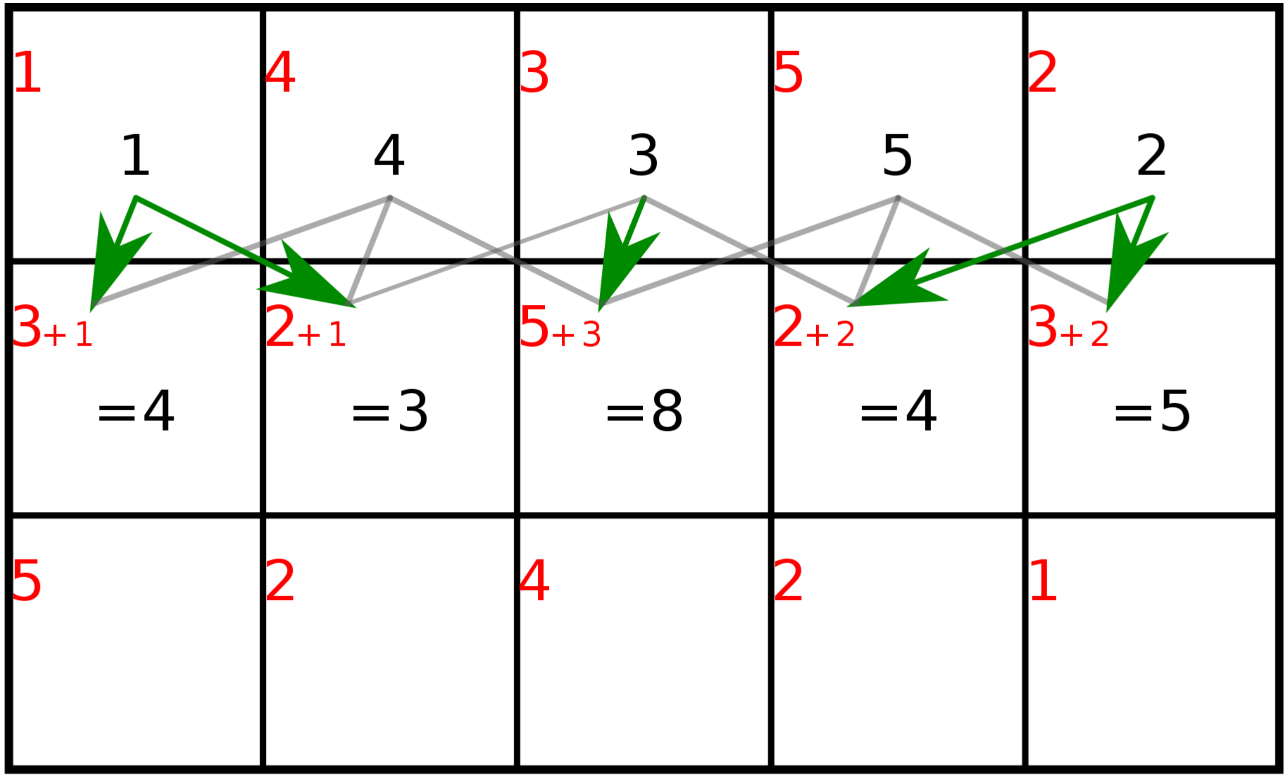
- Can minimize this independently per scanline using dynamic programming (DP)

Dynamic Programming

- Finds "smooth", low-cost path through DSI from left to right
- Visiting a node incurs its data cost, switching disparities from one column to the next also incurs a (smoothness) cost

Dynamic Programming

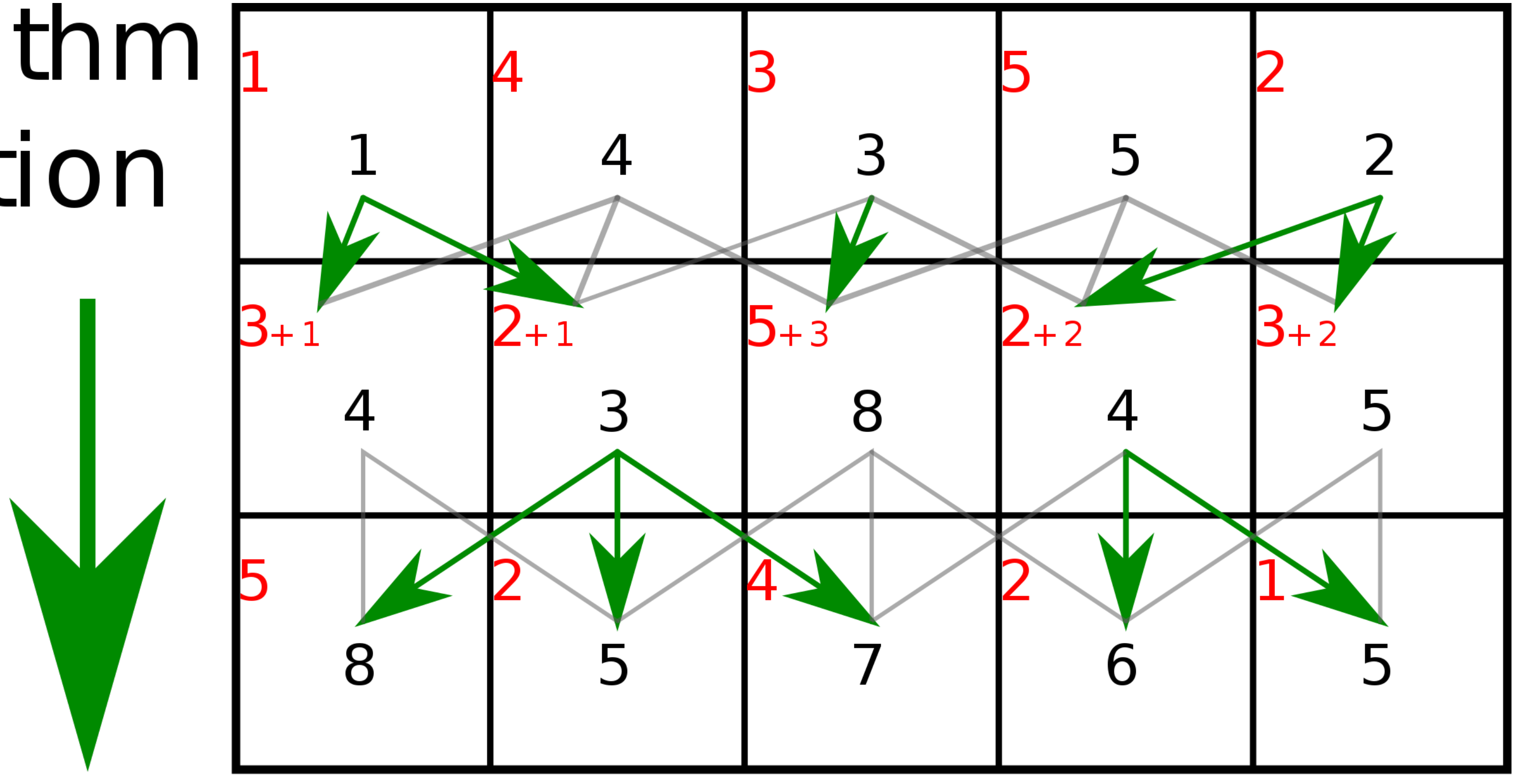
Algorithm
Direction



https://en.wikipedia.org/wiki/Seam_carving

Dynamic Programming

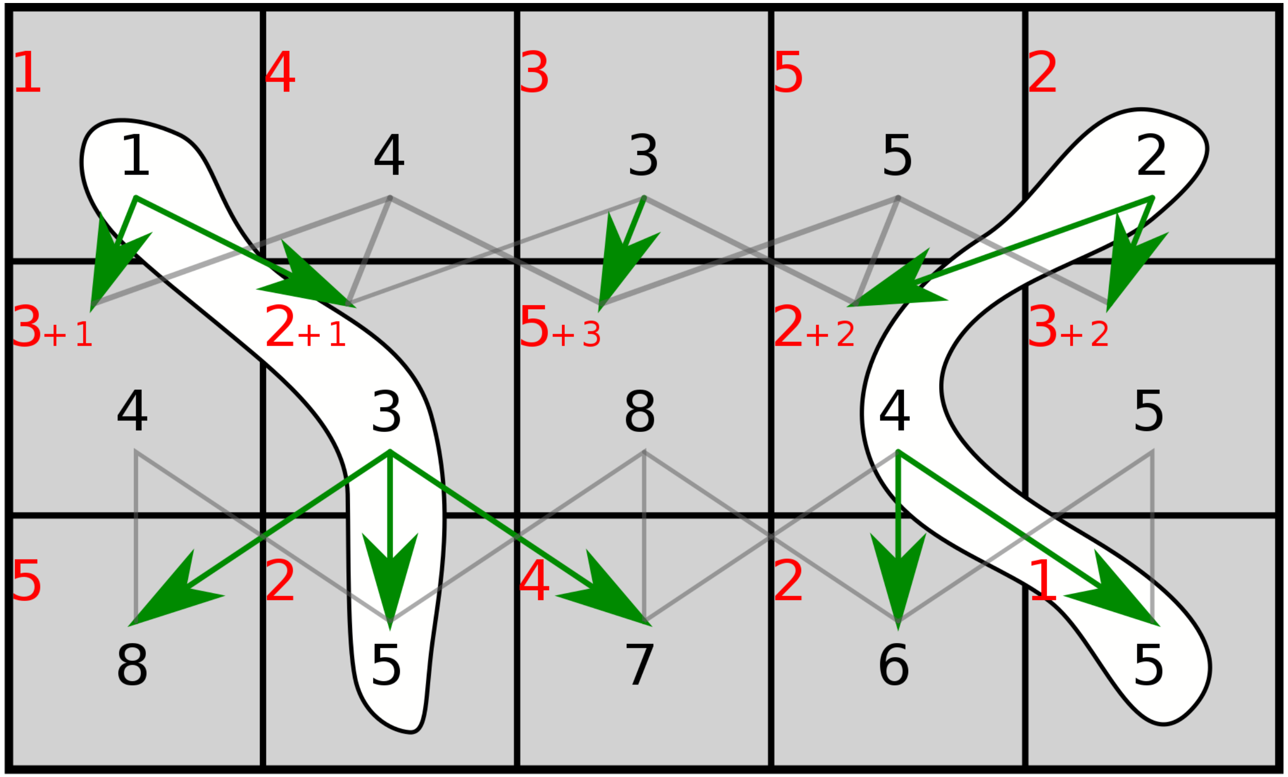
Algorithm Direction



https://en.wikipedia.org/wiki/Seam_carving

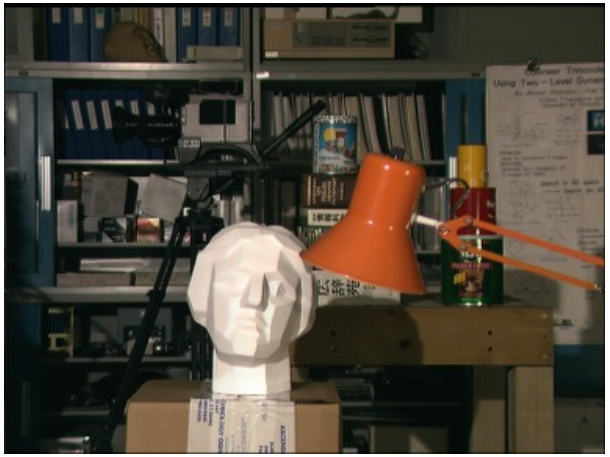
Dynamic Programming

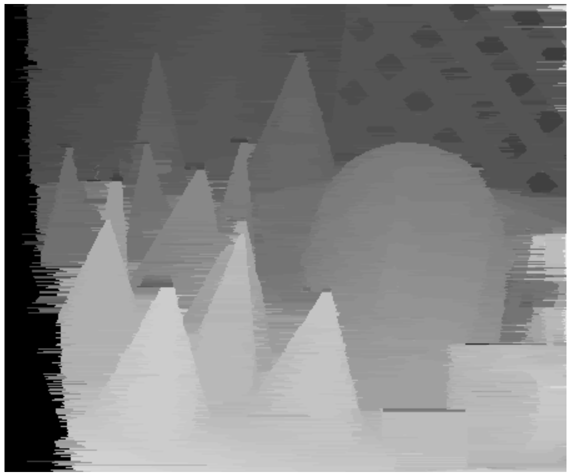
Algorithm Direction



https://en.wikipedia.org/wiki/Seam_carving

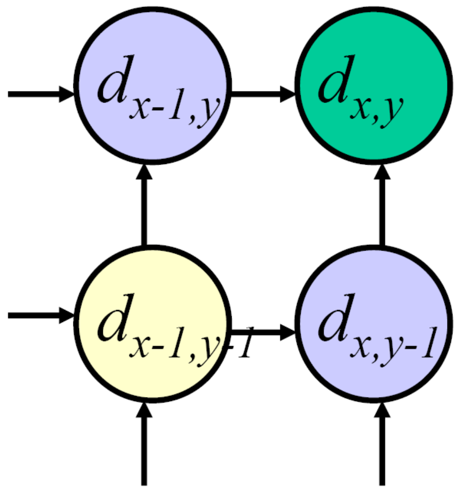
Dynamic Programming





Dynamic Programming

- Can we apply this trick in 2D as well?



- No: the shortest path trick only works to find a 1D path

Slide credit: D. Huttenlocher

Stereo as a minimization problem

$$E(d) = E_d(d) + \lambda E_s(d)$$

- The 2D problem has many local minima
  - Gradient descent doesn't work well
- And a large search space
  - n x m image w/ k disparities has k^{nm} possible solutions
  - Finding the global minimum is NP-hard in general
- Good approximations exist (e.g., graph cuts algorithms)