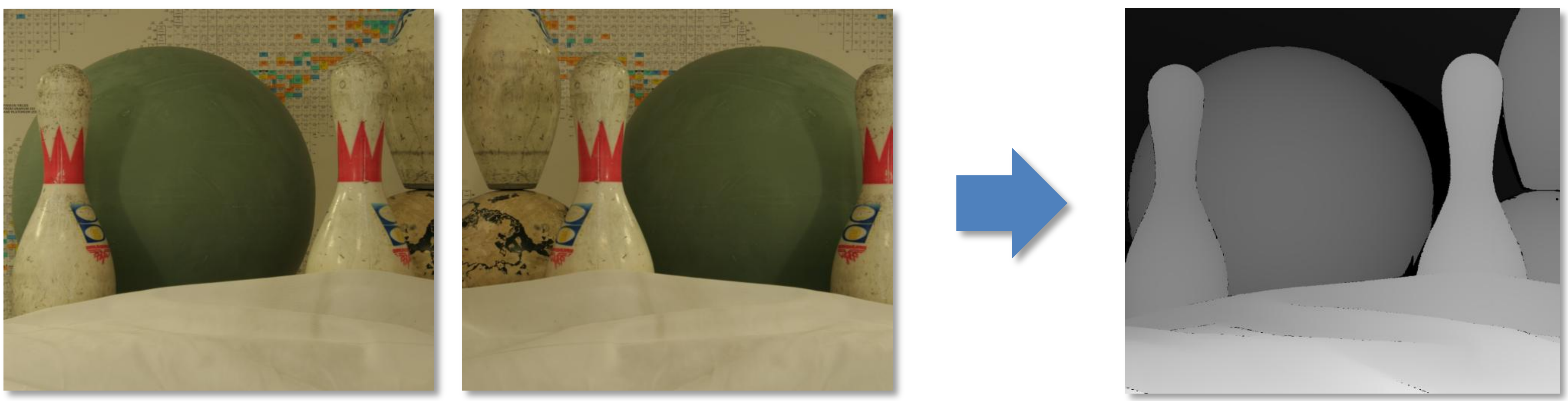
CS6670: Computer Vision

Noah Snavely

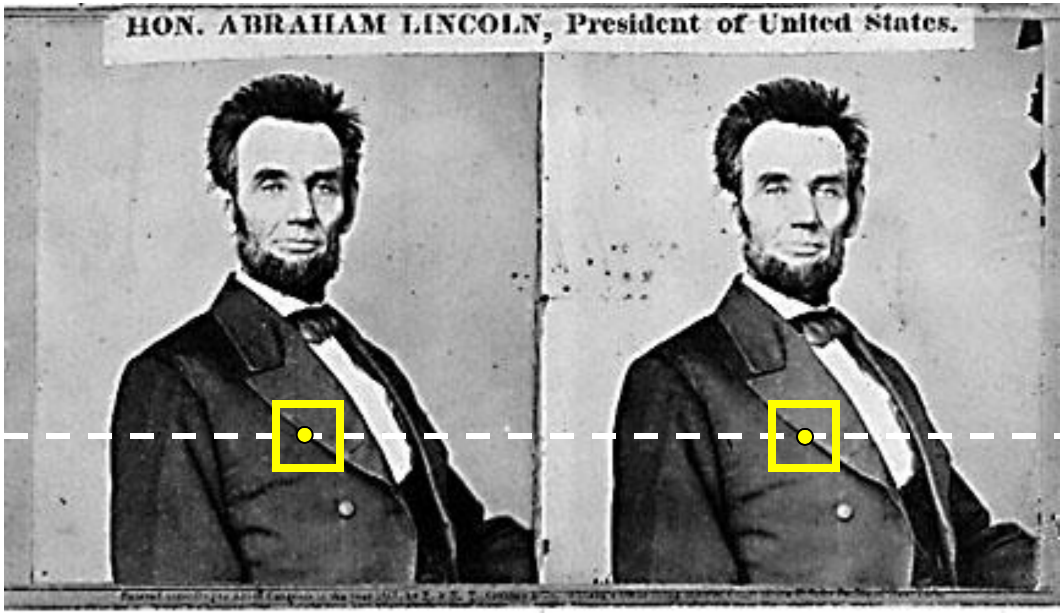
Lecture 11: Stereo and optical flow



Readings

- Szeliski, Chapter 11.2 – 11.5

Your basic stereo algorithm



For each epipolar line

For each pixel in the left image

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

Stereo as energy minimization

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

- Simple pixel / window matching

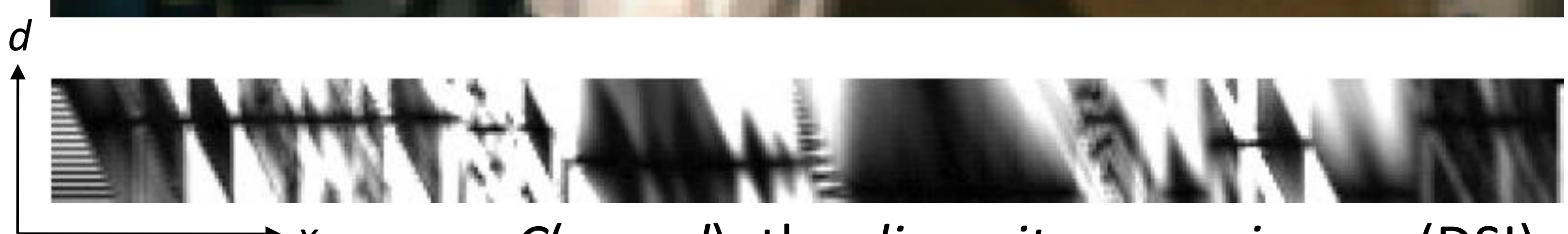
$$E(d) = \sum_{(x,y) \in I} C(x, y, d(x, y))$$

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

Stereo as energy minimization

$I(x, y)$

$J(x, y)$



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

Stereo as energy minimization

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

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

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 → match cost

Adjacent pixels should (usually) move about the same amount → smoothness cost

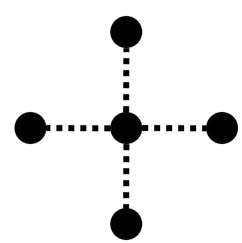
Stereo as energy minimization

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

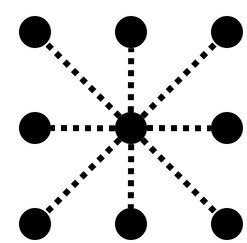
match cost: $$E_d(d) = \sum_{(x,y) \in I} C(x, y, d(x, y))$$

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

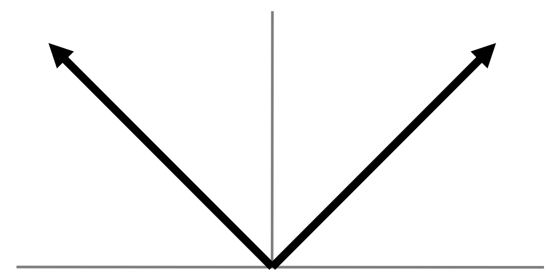
Smoothness cost

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

How do we choose *V*?

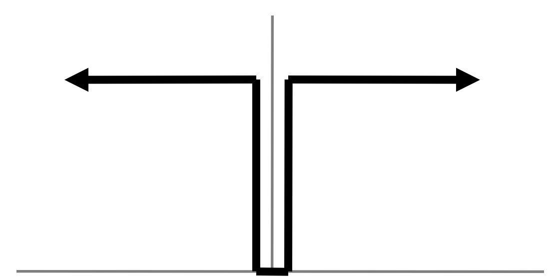
$$V(d_p, d_q) = |d_p - d_q|$$

L_1 distance



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

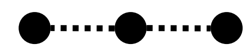
"Potts model"



Dynamic programming

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

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



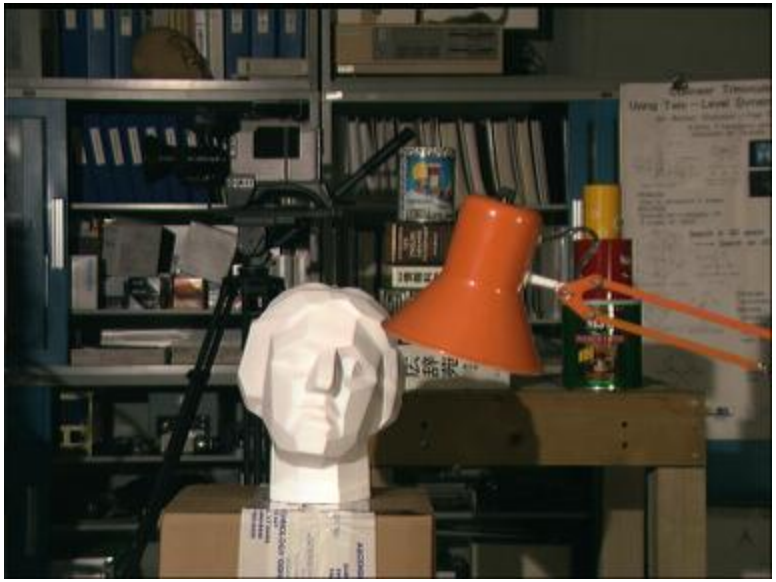
$D(x, y, d)$: minimum cost of solution such that $d(x,y) = d$

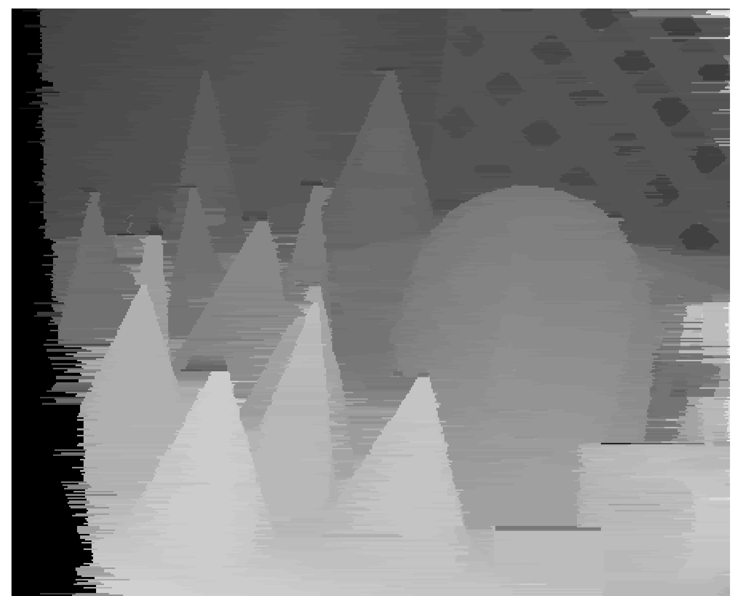
$$D(x, y, d) = C(x, y, d) + \min_{d'} \left\{ D(x-1, y, d') + \lambda \left| d - d' \right| \right\}$$

Dynamic programming

- Finds “smooth” path through DPI from left to right

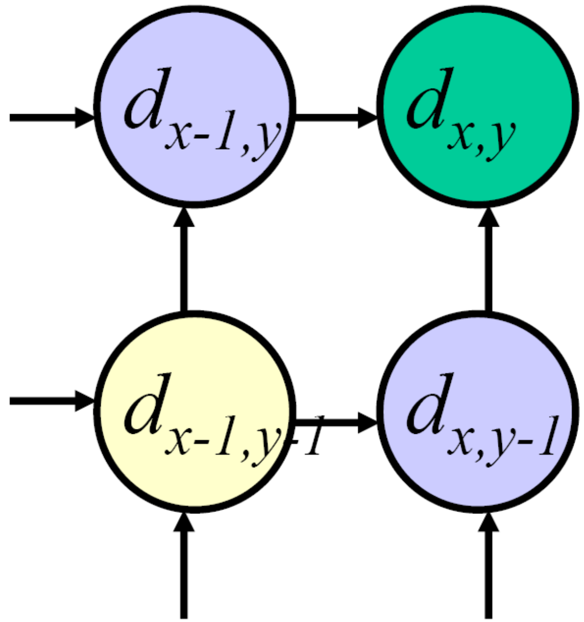
Dynamic Programming





Dynamic programming

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



- No: $d_{x,y-1}$ and $d_{x-1,y}$ may depend on different values of $d_{x-1,y-1}$

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

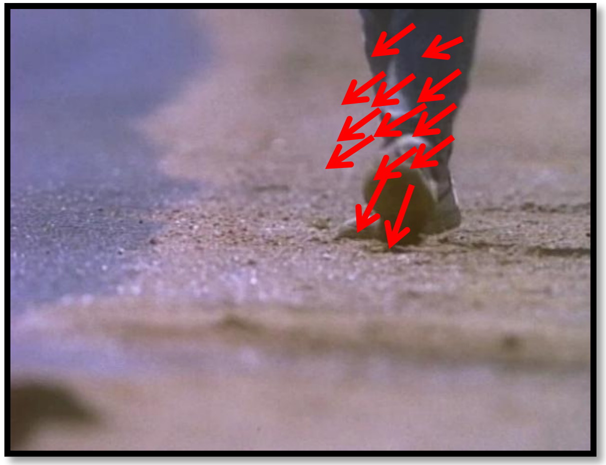
- Good approximations exist… we'll see this soon

Questions?

What if the scene is moving?

- And the camera is fixed (or moving)

Optical flow



- Why would we want to do this?