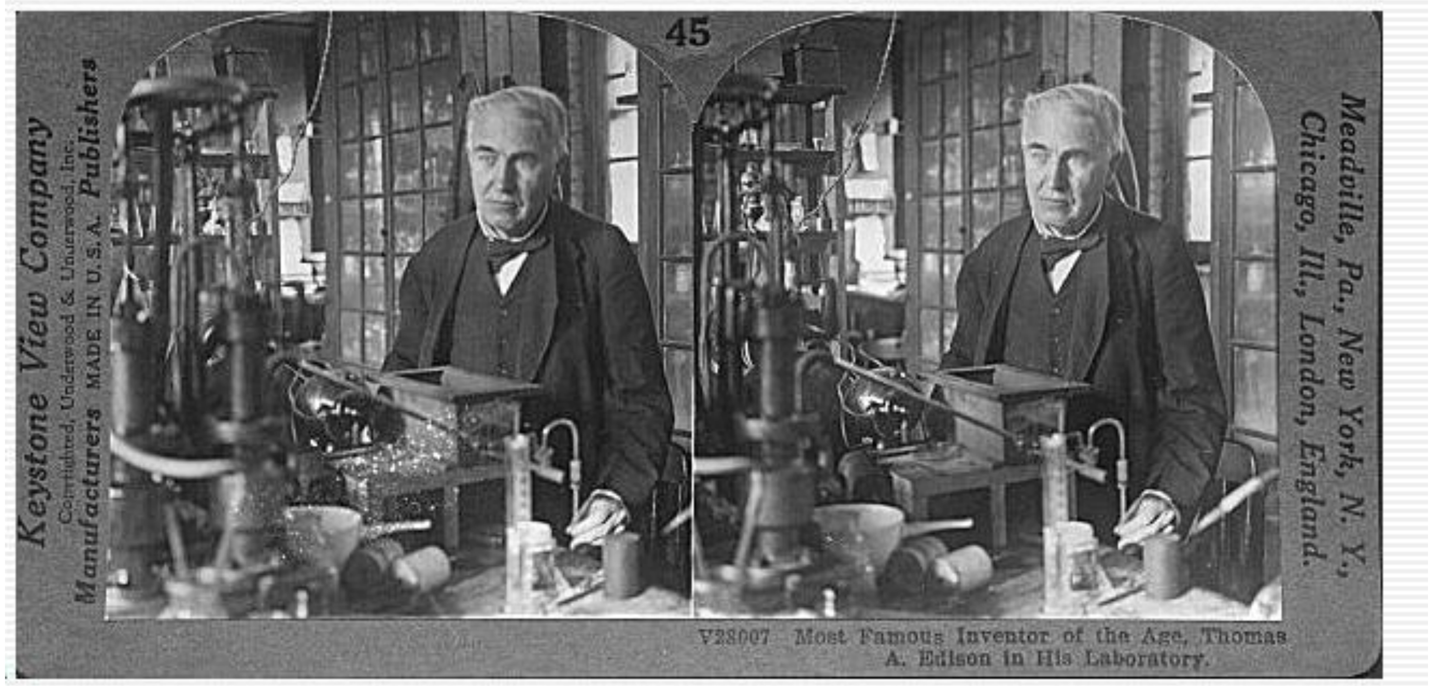
09. Stereo & Optical Flow

Stereo

- Given two calibrated cameras, decide the depth of every pixel in the two images
  - Based on how much each pixel moves between the two images

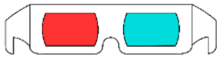


Invented by Sir Charles Wheatstone, 1838

Stereoscopes: a 19th century pastime

Anaglyph 3D

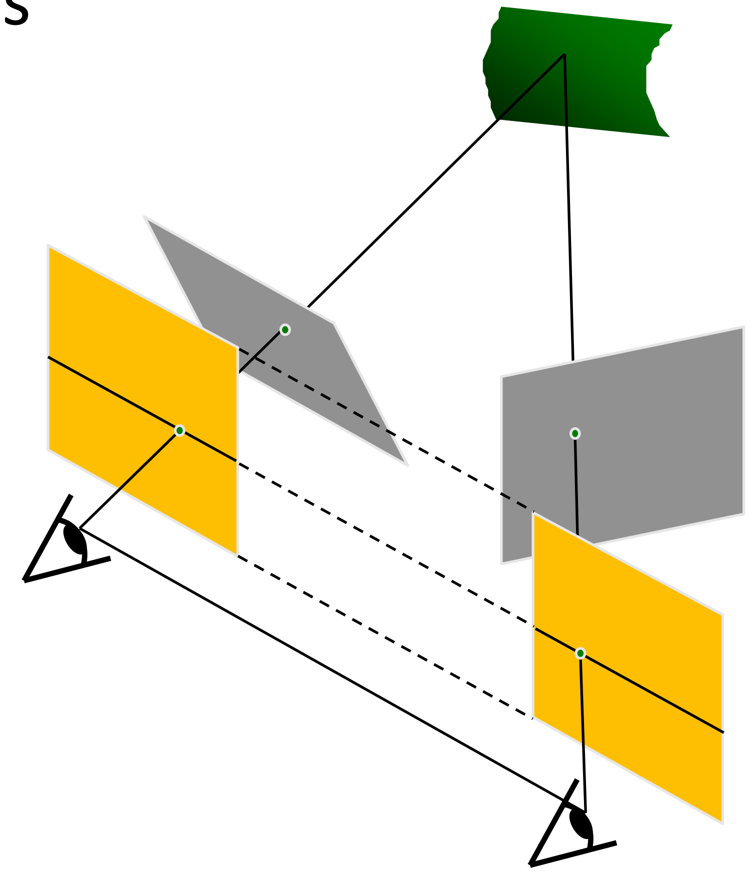
- Encoding each image using different colors (e.g. red and cyan)

Piero della Francesca, Ideal City in an Anaglyph version 

From wikipedia

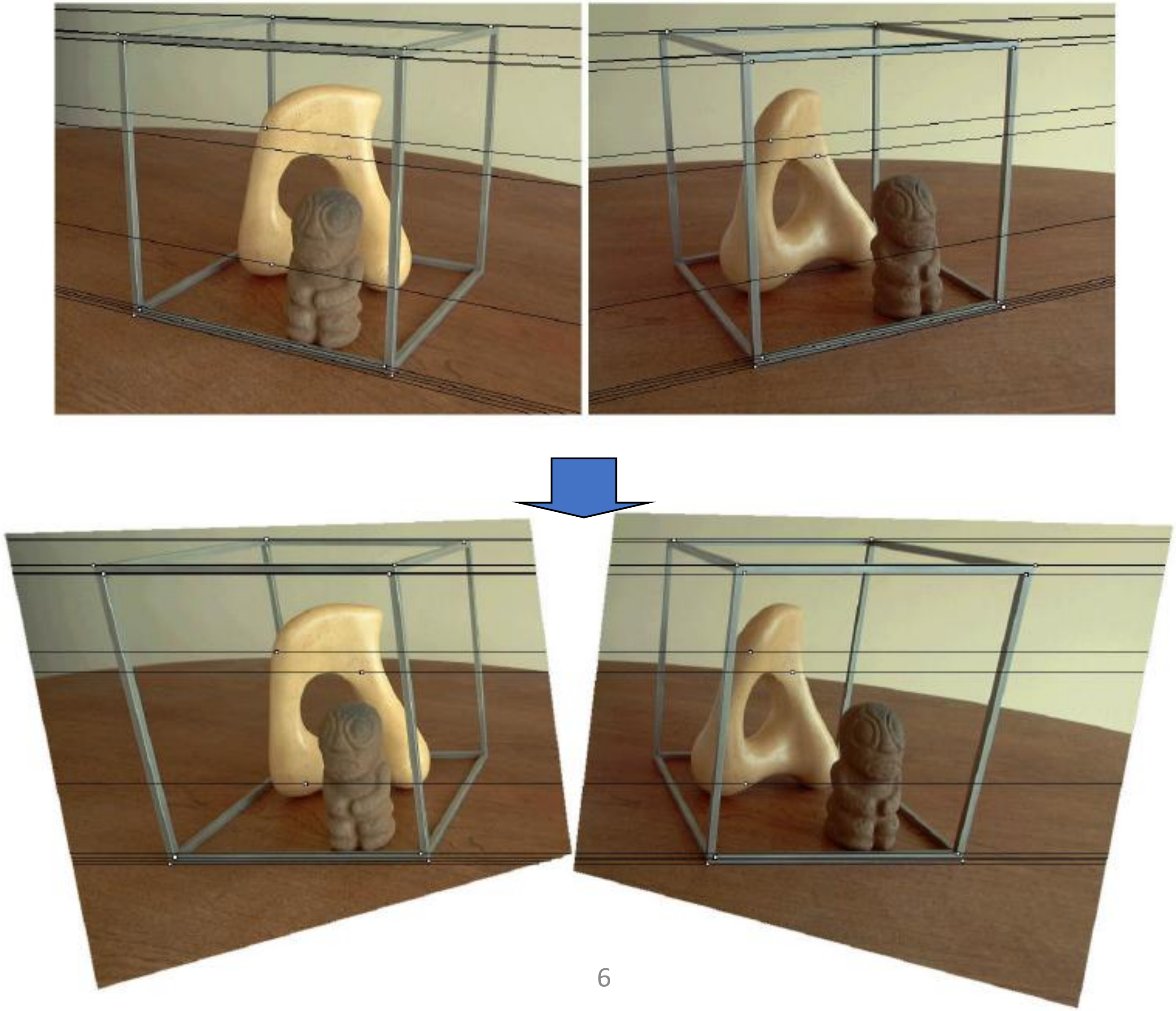
Stereo Image Rectification

- In practice, it is convenient if image scanlines (rows) are the epipolar lines.
- Reproject image planes onto a common plane parallel to the line between optical centers
  - Rotating both cameras without translation
  - Applying a homography to the image
- Pixel motion is horizontal after this transformation



C. Loop and Z. Zhang. Computing Rectifying Homographies for Stereo Vision. CVPR 1999.

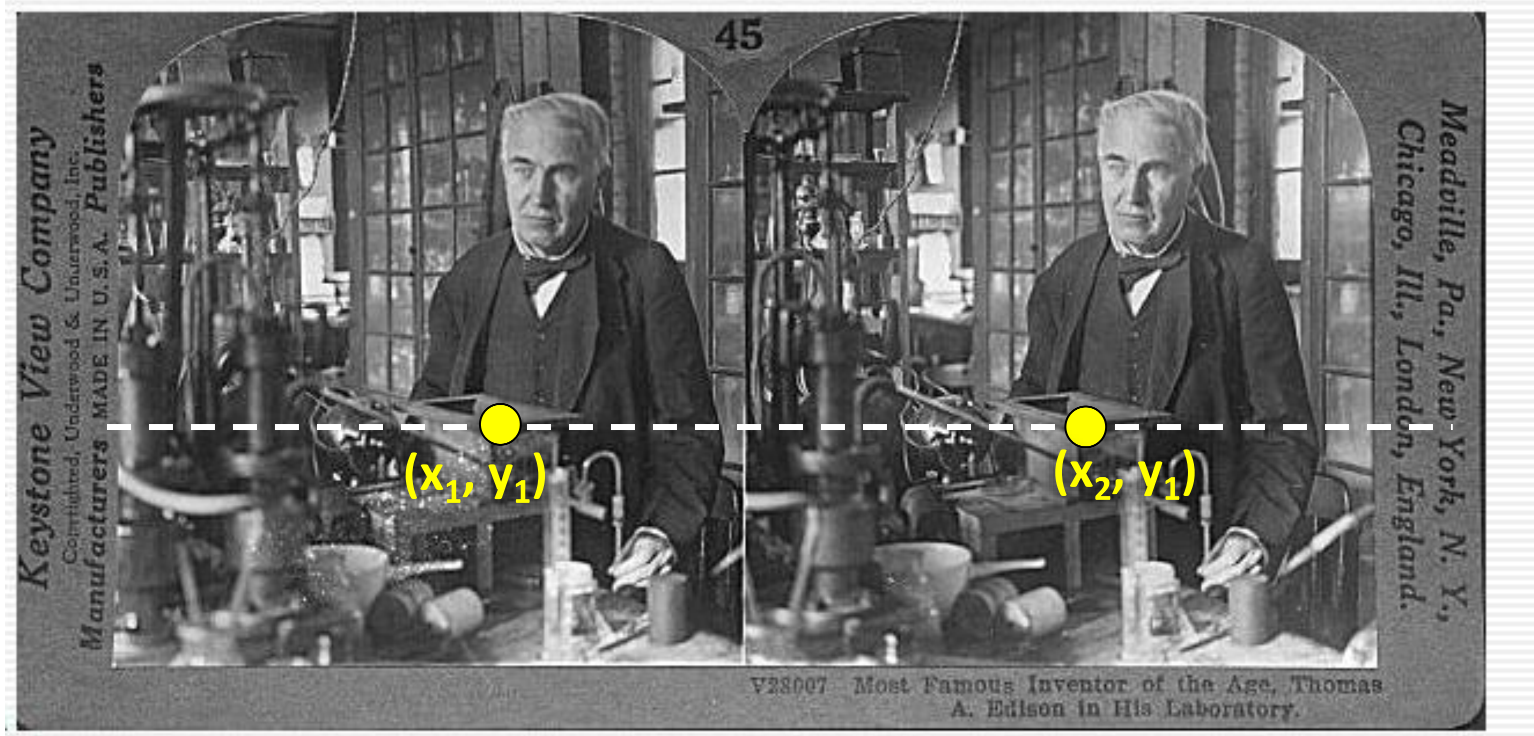
Stereo Image Rectification: example



Epipolar Geometry

The correspondence point lies on the epipolar line

epipolar lines

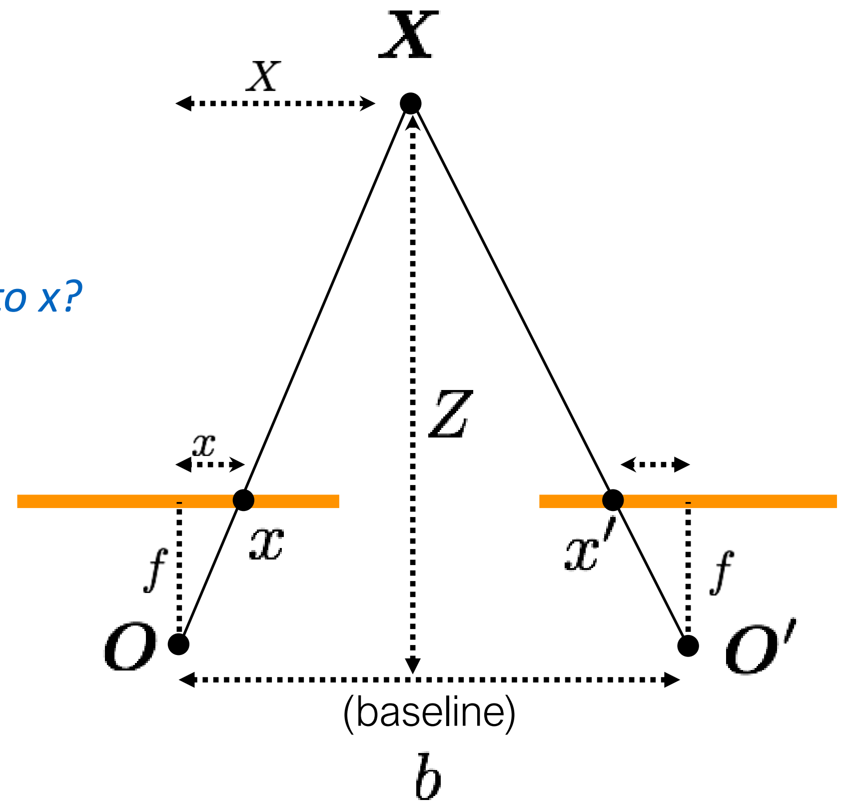


Two images captured by a purely horizontal translating camera

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

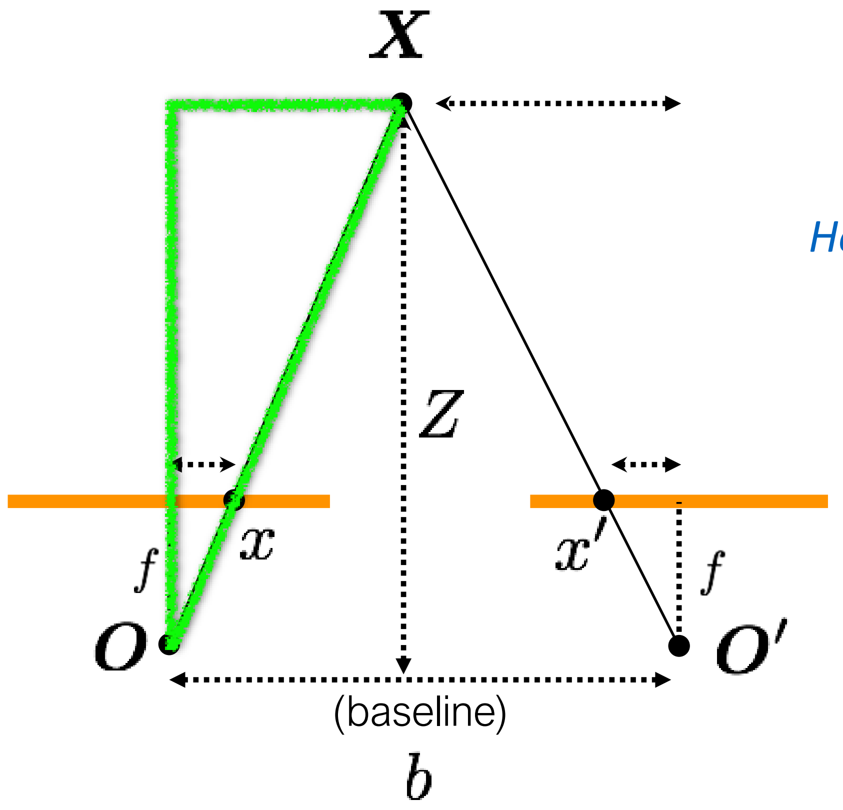
Disparity & Depth

How is X related to x?



Disparity & Depth

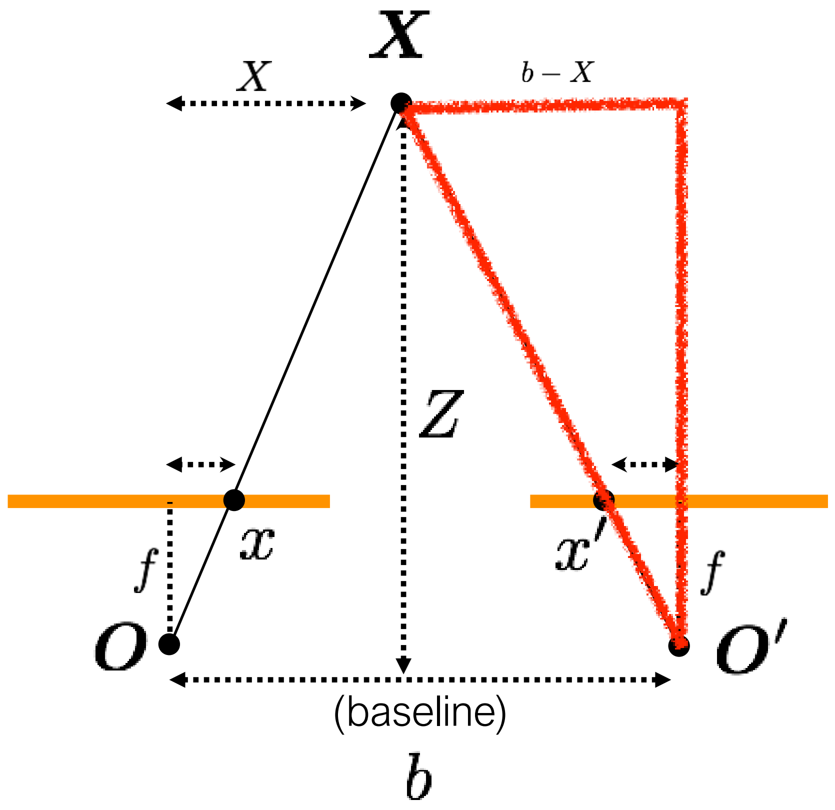
$$\frac{X}{Z} = \frac{x}{f}$$



How is X related to x'?

Disparity & Depth

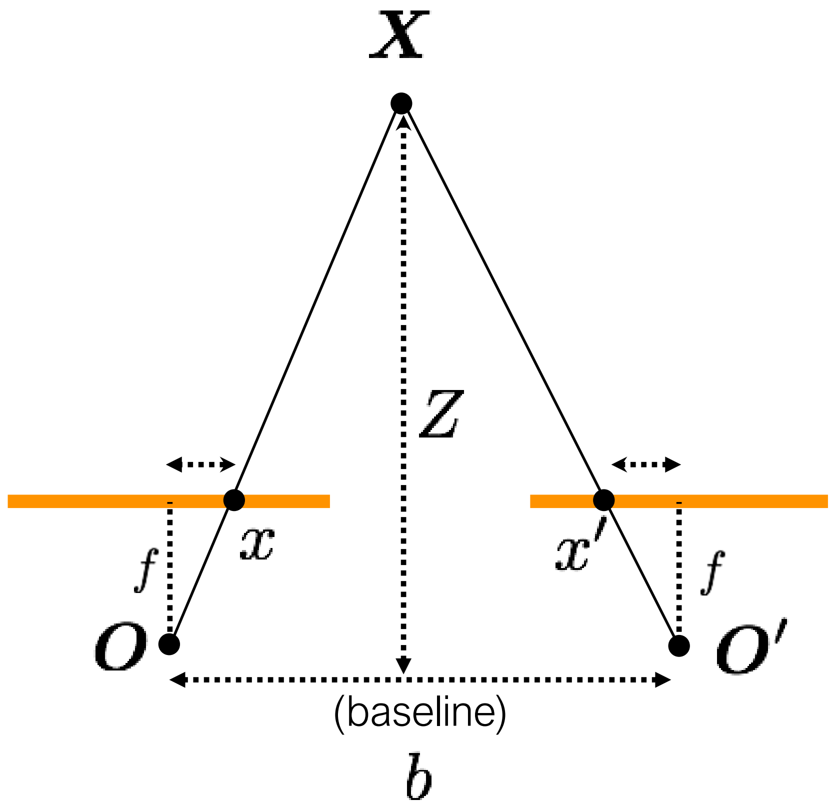
$$\frac{X}{Z} = \frac{x}{f}$$



$$\frac{b - X}{Z} = \frac{-x'}{f}$$

Disparity & Depth

$$\frac{X}{Z} = \frac{x}{f}$$



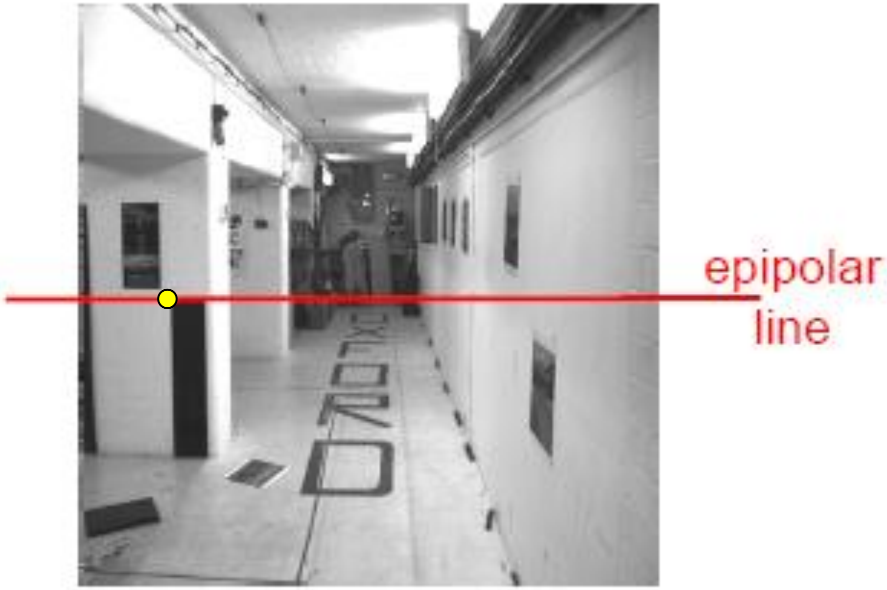
$$\frac{b - X}{Z} = \frac{-x'}{f}$$

Disparity $\quad d = x - x' = \dfrac{bf}{Z}$ inversely proportional to depth

Questions?

Local Methods

f



g

For each epipolar line

For each pixel in the left image

- compare with every pixel on the same epipolar line in right image, e.g. $|f(x, y) - g(x + d, y)|^2$
- pick pixel with minimum match cost

Disparity Space Image (DSI)

$f(x, y)$

$g(x, y)$

- At each pixel (x, y), a cost can be evaluated for each disparity d
- This defines a *cost volume* $C(x, y, d)$, also known as disparity space image (DSI)
- It is often helpful to look at a 2D slice of this cost volume, e.g. by fixing y

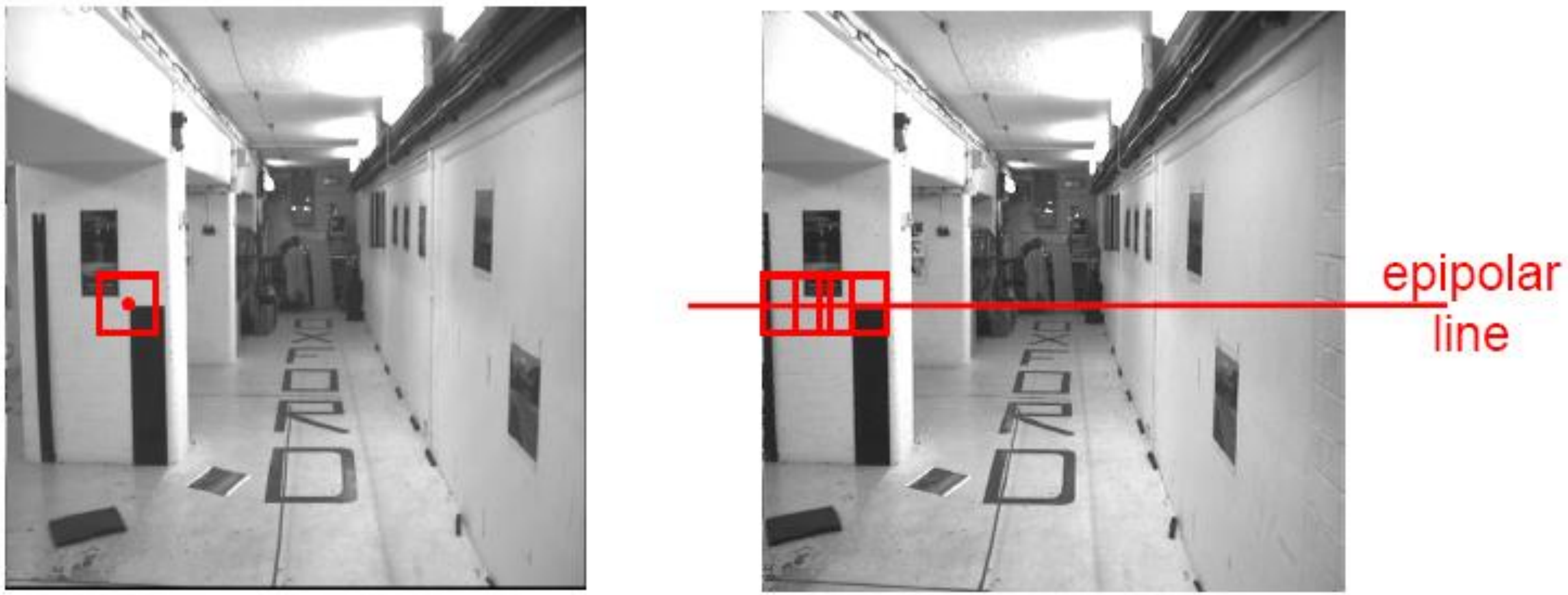
Winner Take All

- Choose the minimum of each column in the DSI independently:

$$d(x, y) = \underset{d'}{\arg\min}\ C(x, y, d')$$

Cost Aggregation

- Some pixels are textureless, matching in the second image is ambiguous
- Evaluate correspondence by comparing a local neighborhood to reduce ambiguity/noise



Cost Aggregation

Questions: Which is the corresponding window in the second image?

f g

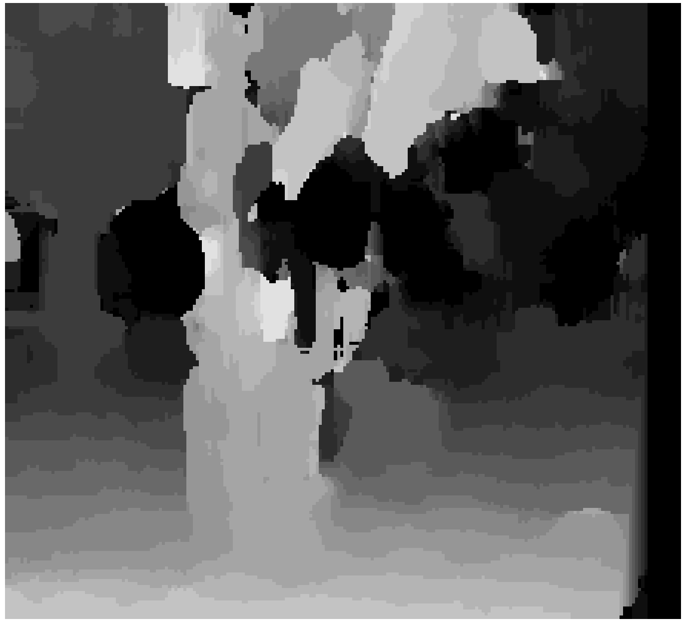
$$SSD = \sum_{(x,y)} | f(x,y) - g(x,y) |^2$$ Sum over costs within a local window

$$\mathrm{ZNCC} = \sum_{(x,y)} \hat{f}(x,y)\hat{g}(x,y)$$ Zero-mean Normalized Cross Correlation

$$\hat{f} = \frac{f - \bar{f}}{\sqrt{\sum | f - \bar{f} |^2}} \qquad \hat{g} = \frac{g - \bar{g}}{\sqrt{\sum | g - \bar{g} |^2}}$$

Drawback of Using a Window

W = 3

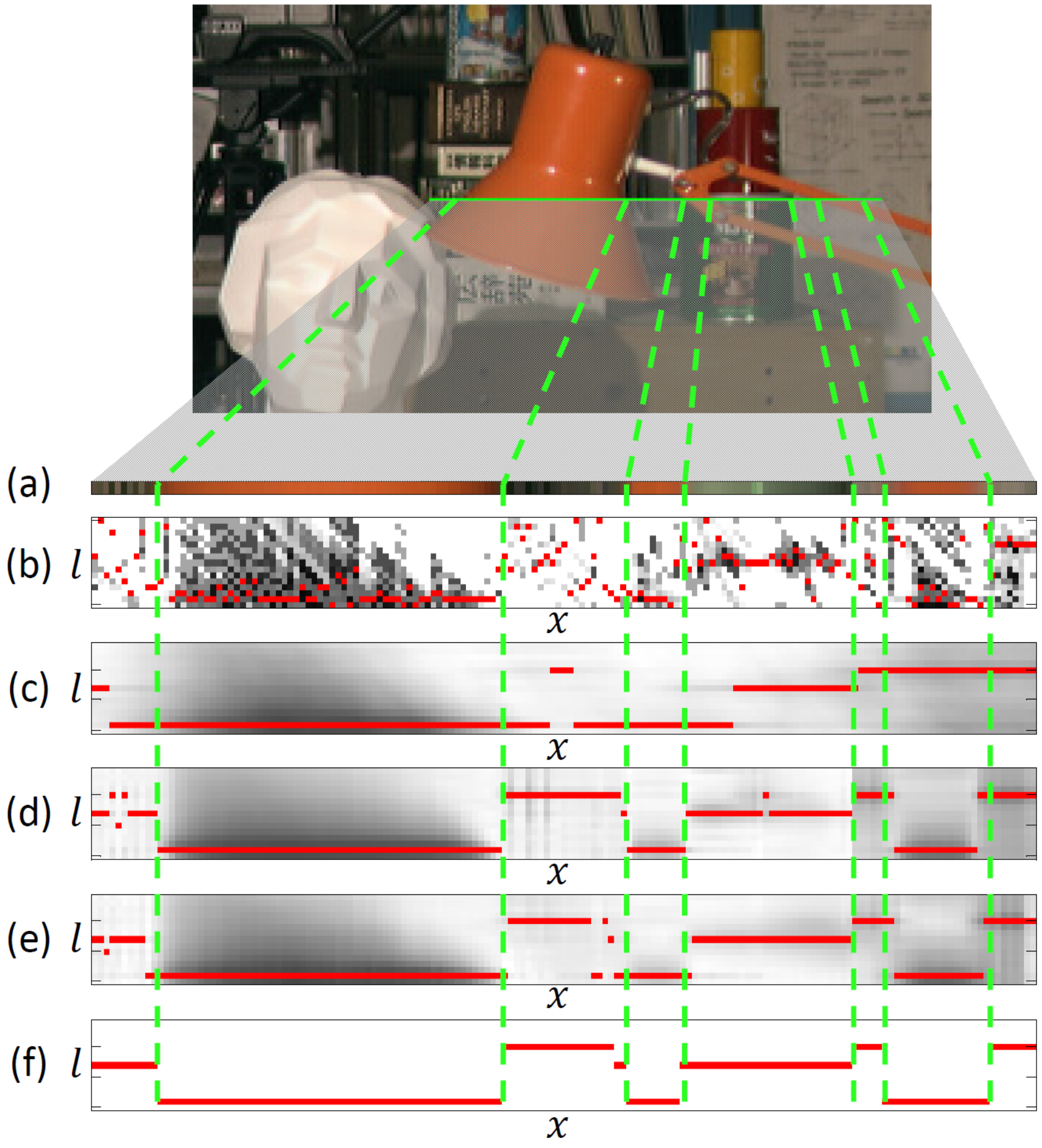


W = 20

- The entire window is assumed to have the same depth (planar and facing the camera)
- Want window large enough to have sufficient intensity variation, yet small enough to contain only pixels with about the same depth

Aggregation by Filtering

- Aggregating with SSD is equivalent to filter the DSI with a box filter
  - (b) the original DSI (red marks the disparity with minimum cost)
  - (c) the DSI filtered by a box filter
- Bilateral filters can generate better results
  - (d) the DSI filtered by a bilateral filter
- Guided filters are faster when large windows are used
  - (e) the DSI filtered by a Guided filter



Spatially Variant Kernels

- Bilateral filter and guided filters adaptively change kernel weights according to image content
  - Assuming pixels on the same object to have the same depth

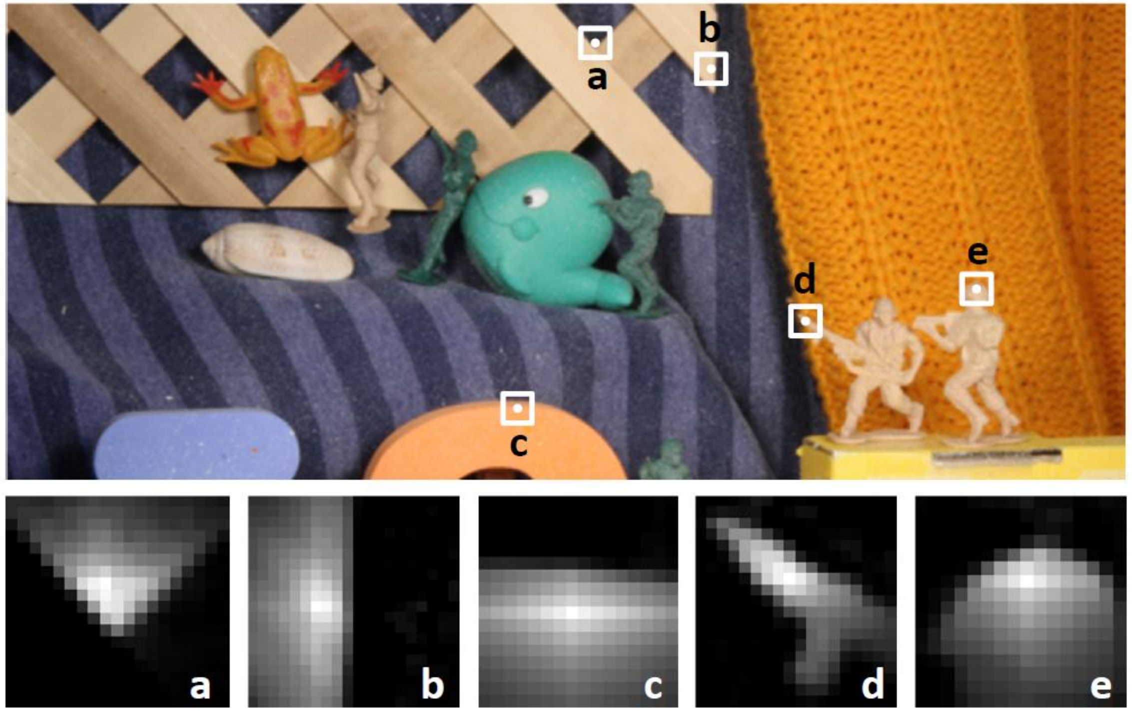


Figure 3. **Filter kernels.** We show kernels of the guided filter with $r = 9$ and $\epsilon = 0.01^2$, at different locations in an image of [1].

Local Methods

Adaptive Support-Weight Approach for Correspondence Search

Kuk-Jin Yoon, *Student Member, IEEE*, and
In So Kweon, *Member, IEEE*

[PAMI 2006]

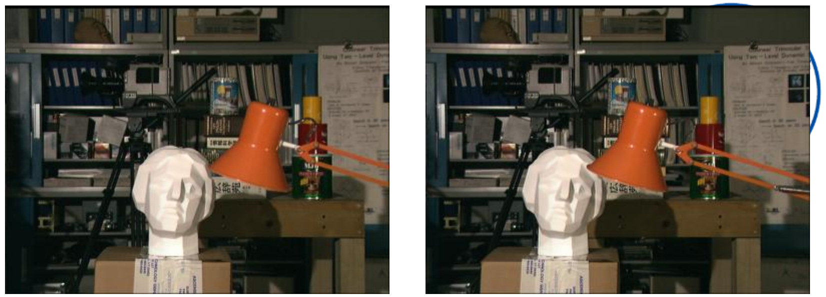
Fast Cost-Volume Filtering for Visual Correspondence and Beyond

Christoph Rhemann[1], Asmaa Hosni[1], Michael Bleyer[1], Carsten Rother[2], Margrit Gelautz[1]
[1]Vienna University of Technology, Vienna, Austria [2]Microsoft Research Cambridge, Cambridge, UK

[CVPR 2011]

Questions?

Global Methods



result by global methods

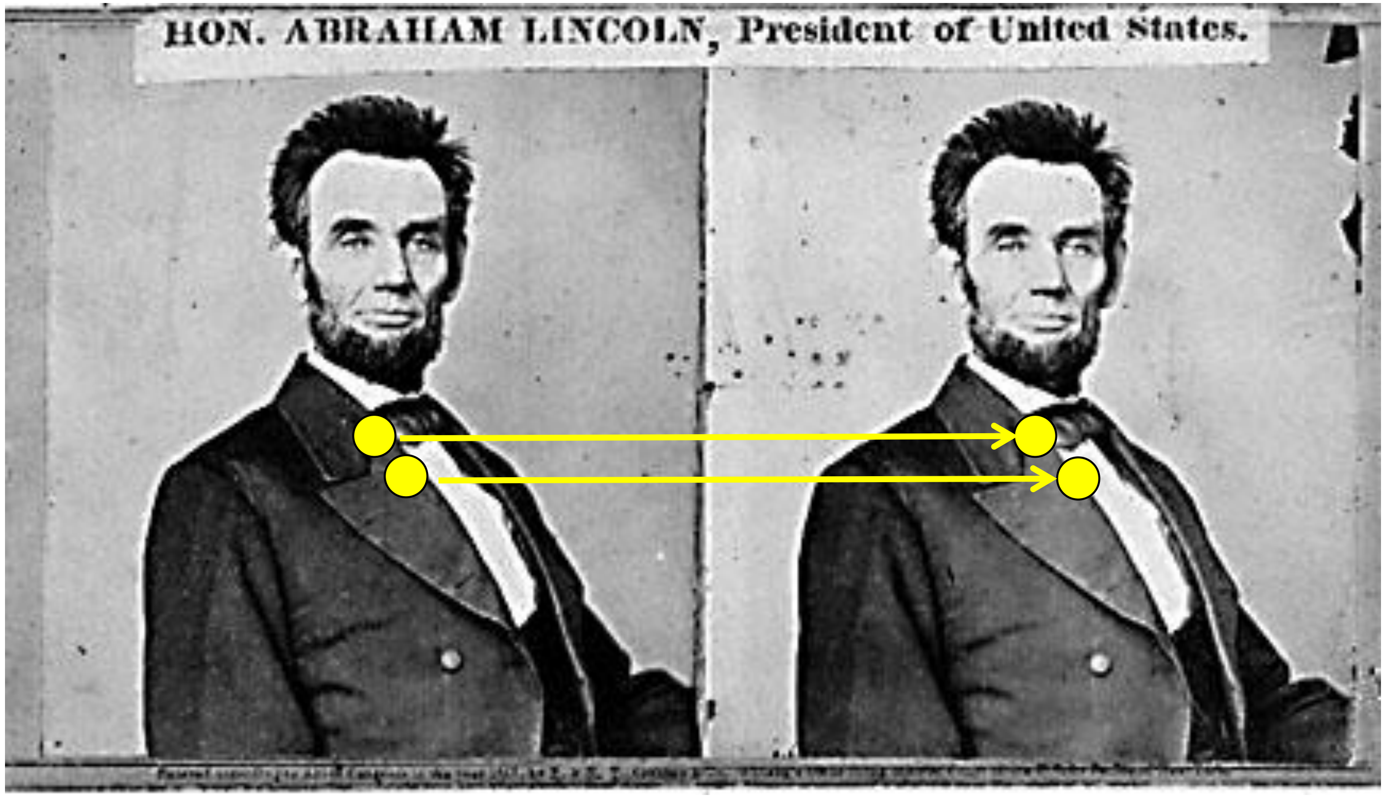
Boykov et al., [Fast Approximate Energy Minimization via Graph Cuts](), International Conference on Computer Vision, September 1999.

ground truth

the latest and greatest results at: http://www.middlebury.edu/stereo

Smoothness Assumption

- Textureless regions are ambiguous to match
- If two pixels are adjacent, their disparities should be similar



Stereo by Energy Minimization

- An objective function with data/match cost and smoothness cost

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

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

Adjacent pixels should (usually) have the same depth

Stereo by Energy Minimization

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

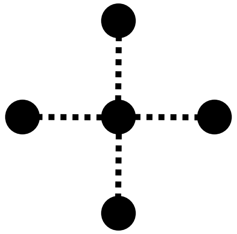
match cost: $$E_d(d) = \sum_{p \in I} D(p, d_p)$$

$D(p, d_p)$ the matching cost of the pixel p with disparity d_p

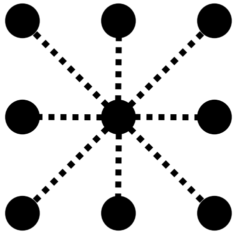
smoothness cost: $$E_s(d) = \sum_{(p,q) \in E} S(p, q, d_p, d_q)$$

$S(p, q, d_p, d_q)$ the smoothness cost when p, q have disparities of d_p, d_q respectively

E : set of neighboring pixels



4-connected neighborhood



8-connected neighborhood

Smoothness Cost

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

How do we choose S?

1. Pixels of similar color have stronger smoothness

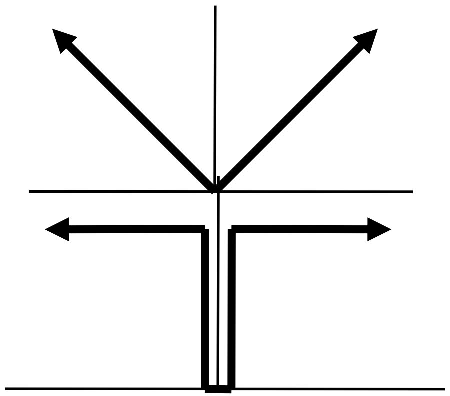
$$S \propto |color(p) - color(q)|^{-1}$$

2. Neighboring pixels have similar depth

$$S \propto |d_p - d_q|$$

or

$$S \propto \begin{cases} 0 & if\ d_p = d_q \\ 1 & if\ d_p \neq d_q \end{cases}$$



"Potts model"
The most common choice

$$\boxed{S = |color(p) - color(q)|^{-1}\delta(d_p - d_q)}$$

Global Methods

Results by Graph-Cut Optimization

result by global methods

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

ground truth

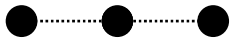
the latest and greatest results at: http://www.middlebury.edu/stereo

Questions?

Stereo by Dynamic Programming

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

- The optimization problem is simplified by considering one row at a time



- Global optimal can be achieved by the *Dynamic Programming* algorithm (for each row)

Stereo by Dynamic Programming

- For each row of pixels, we need to find a "smooth" path through DSI from left to right that minimizes the cost

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

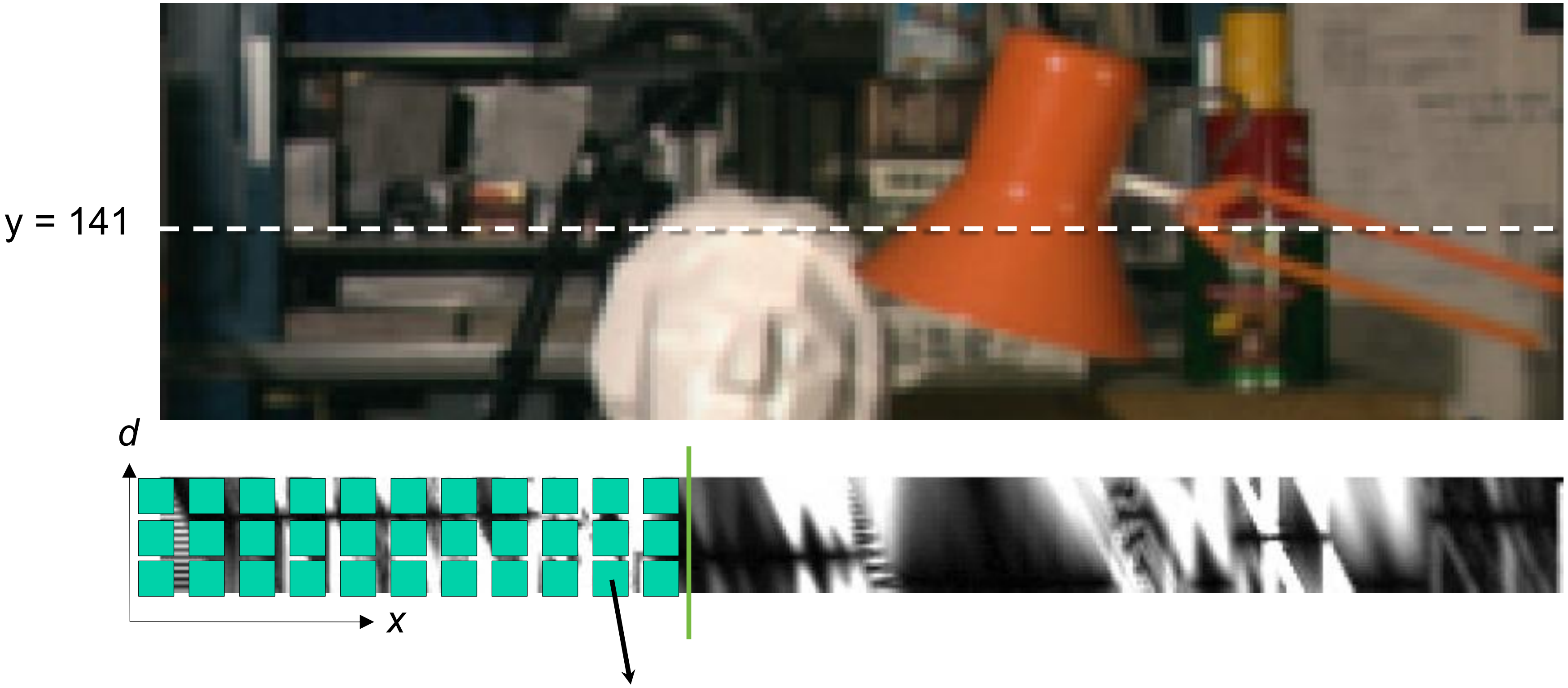
Dynamic Programming

- We define $D(x, y_0, d)$ as the minimum cost among all paths that
  - Start from $(0, y_0)$, i.e. the left border of the image
  - End at (x, y_0) with disparity d
- Dynamic programming computes D efficiently by:

$$D(x, y_0, d) = \underbrace{C(x, y_0, d)}_{\text{Data cost at } (x, y_0)} + \min_{d'}\{D(x-1, y_0, d') + \underbrace{\lambda|d - d'|}_{\text{smooth cost between } (x-1, y_0) \text{ and } (x, y_0)}\}$$

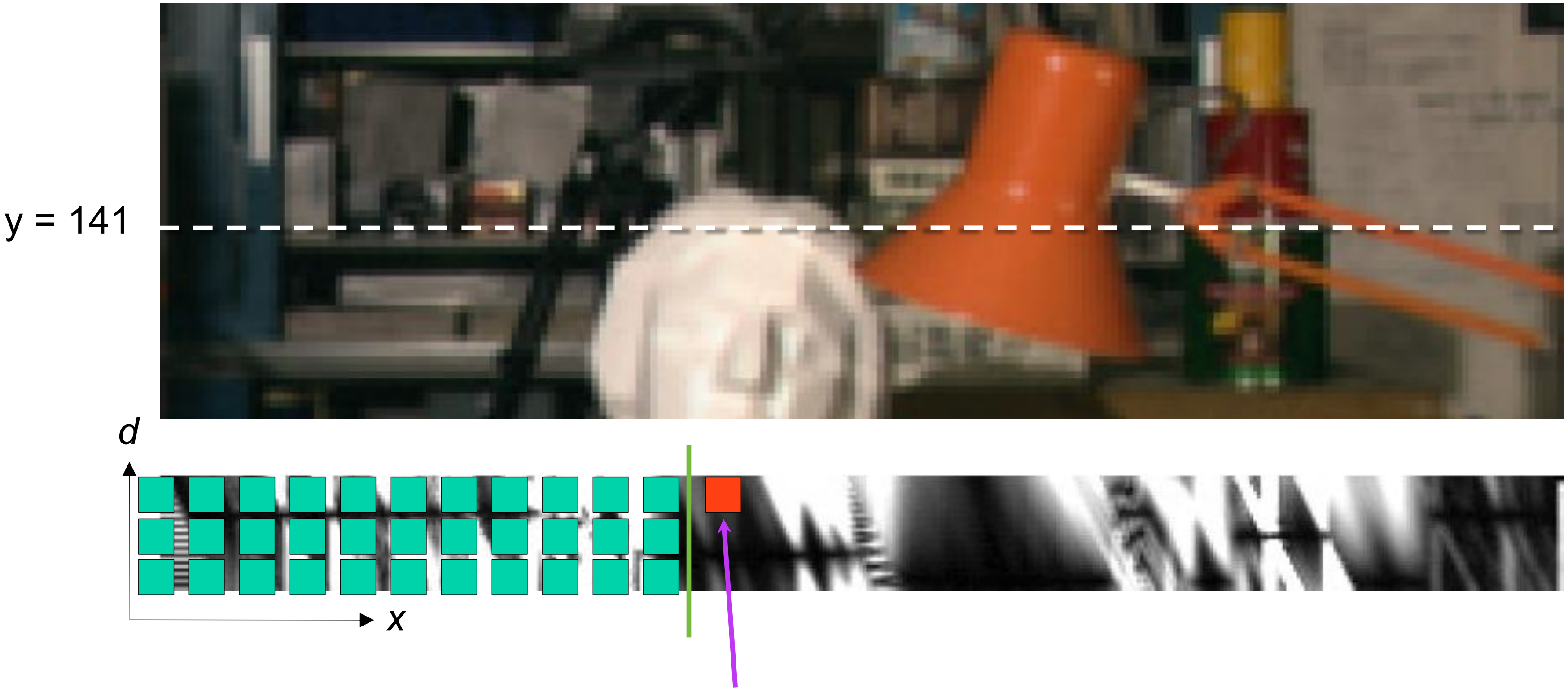
- This strategy allow us to reuse computation at previous pixels

Dynamic Programming



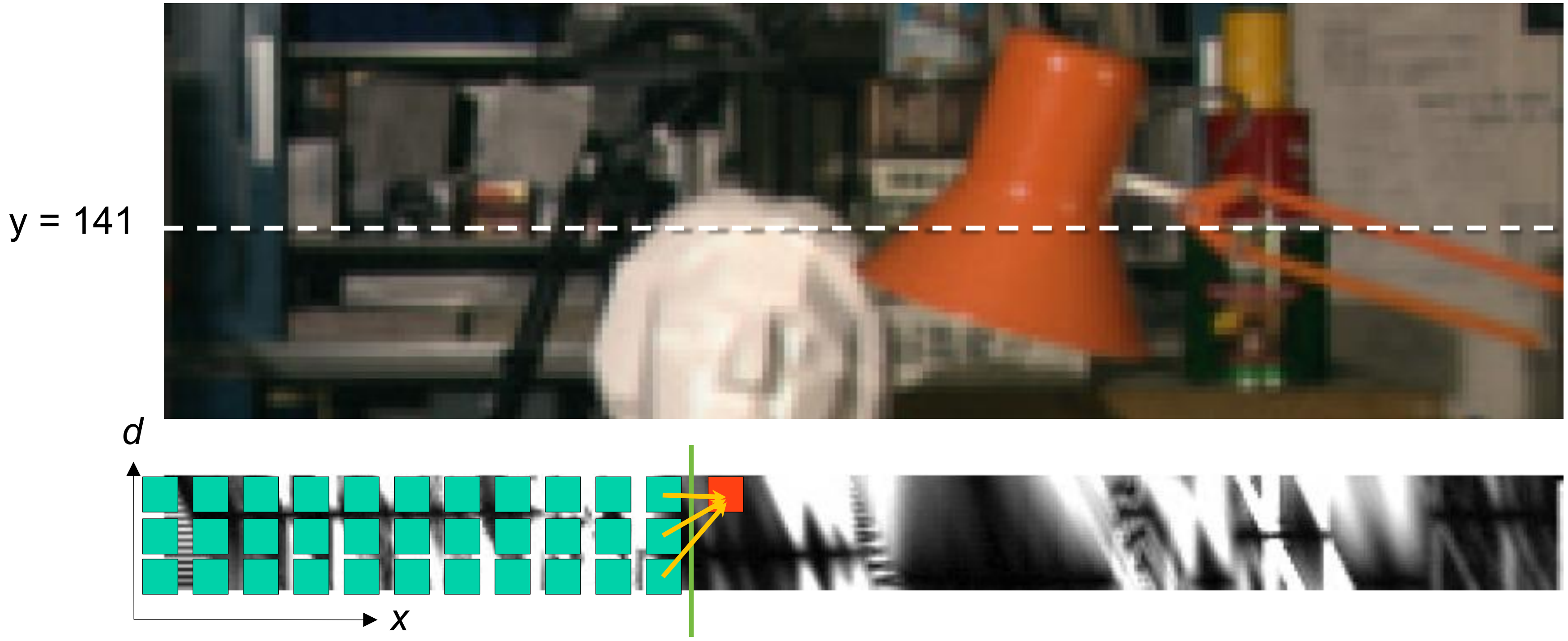
Cost of optimal path from the left

Dynamic Programming



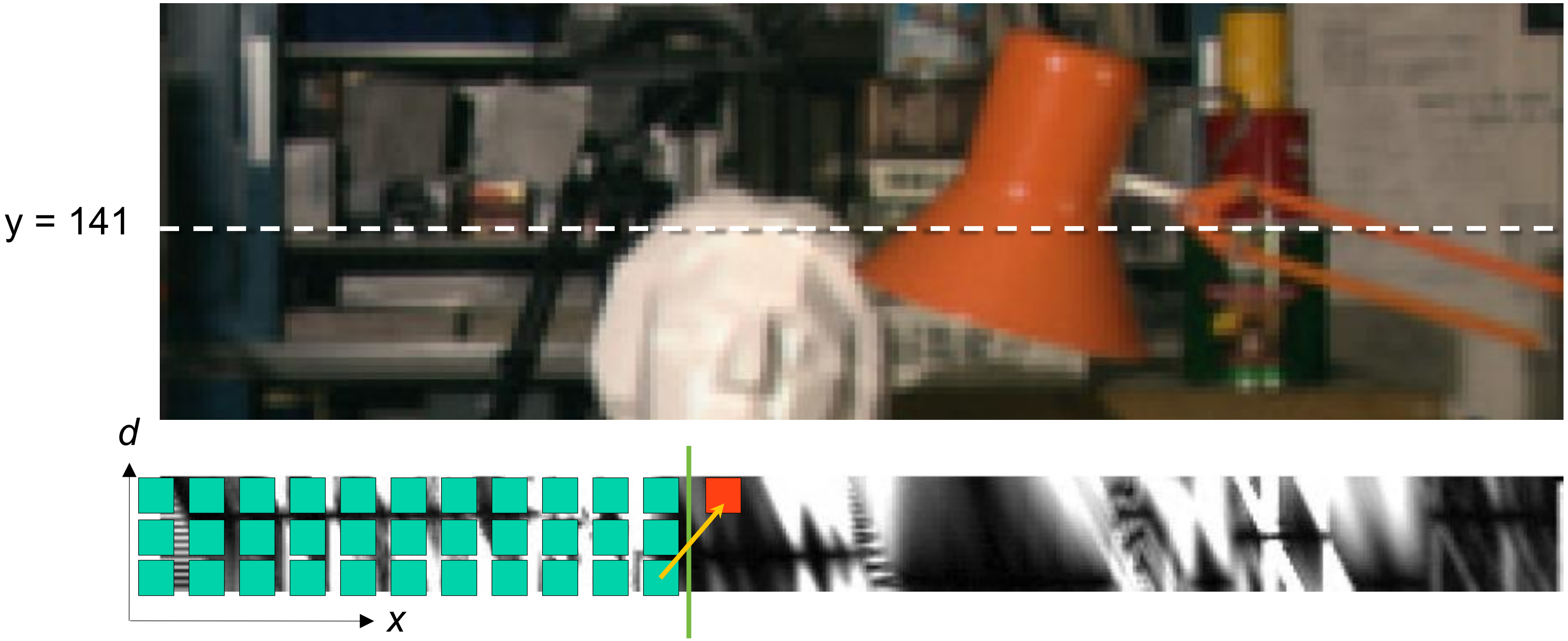
How to compute for this pixel?

Dynamic Programming



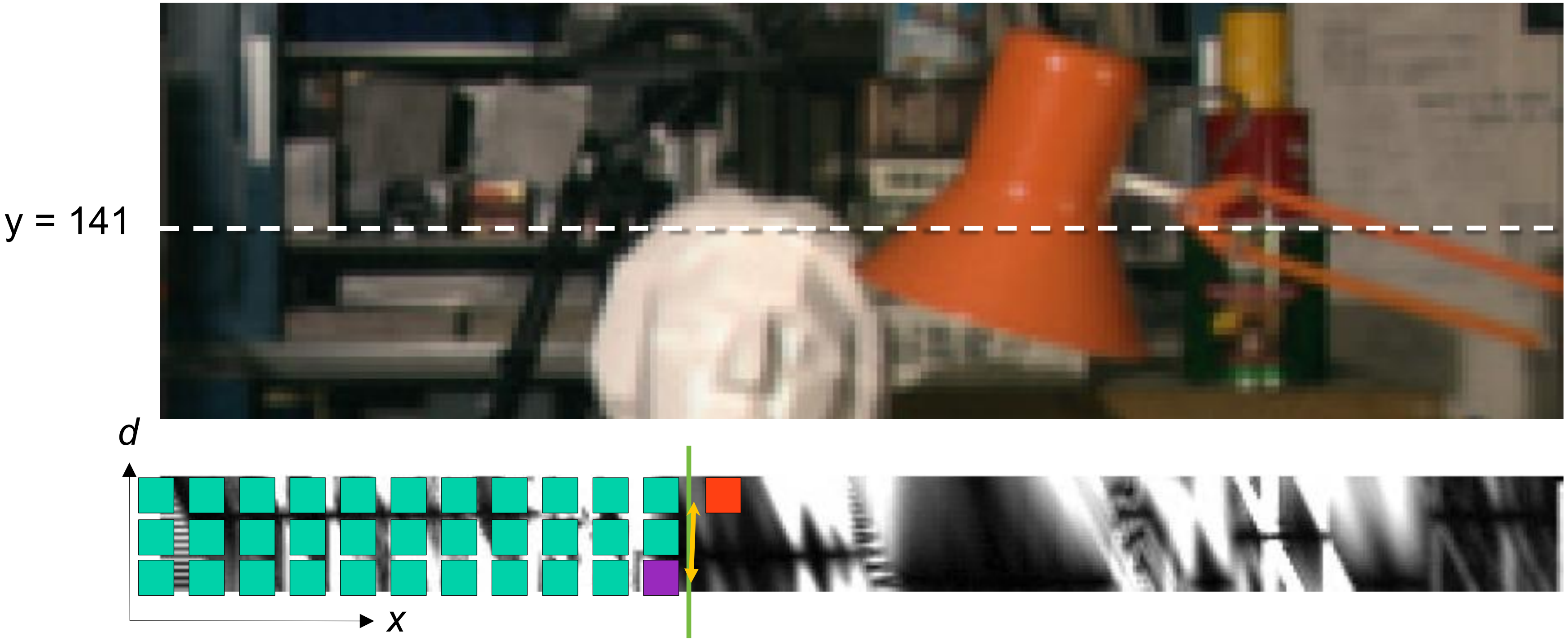
How to compute for this pixel?
Only 3 ways to get here...

Dynamic Programming



This cost is ...

Dynamic Programming

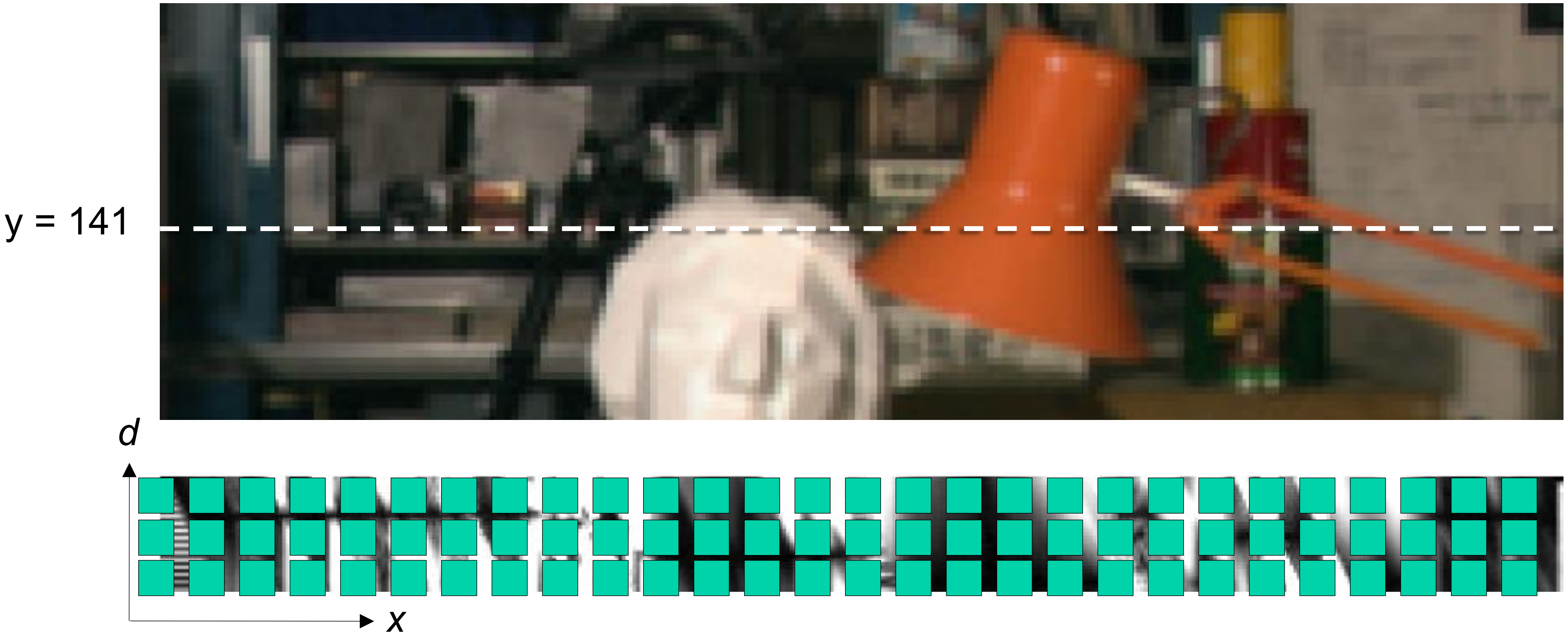


This cost is ...

+ +

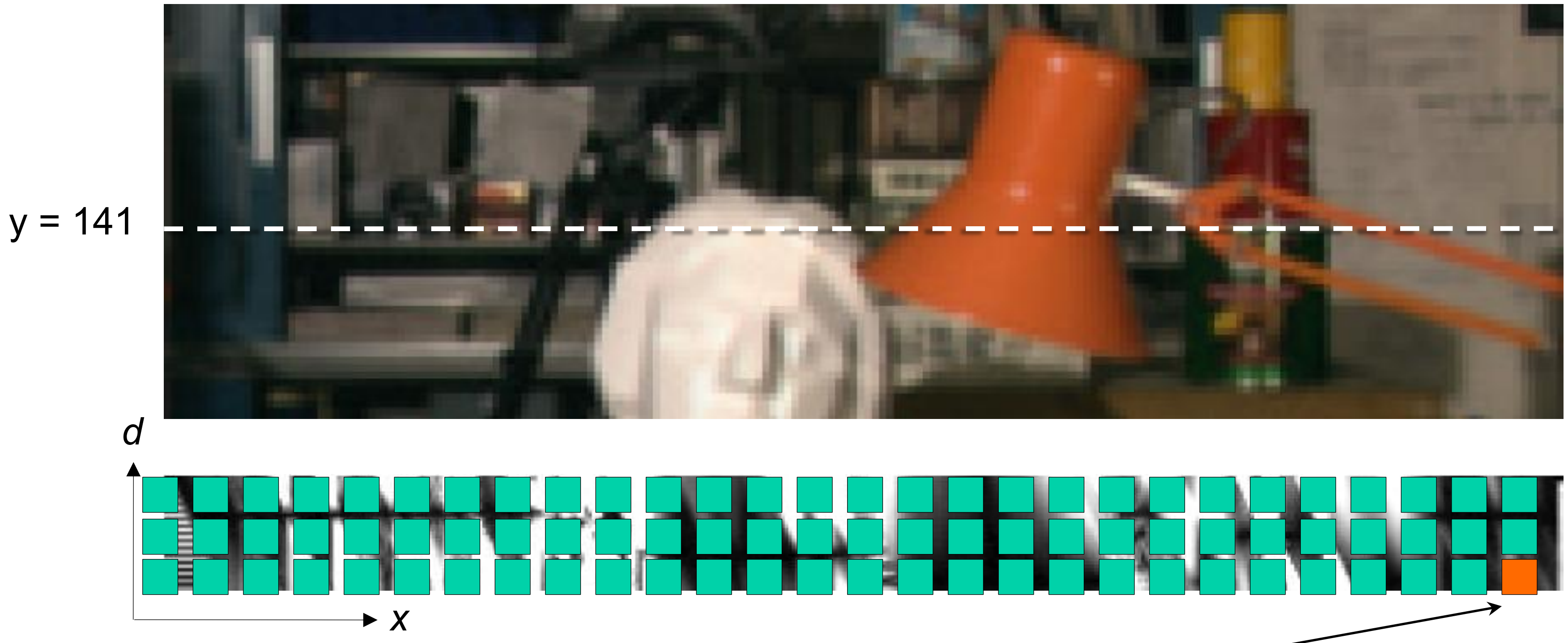
D Ed Es

Dynamic Programming



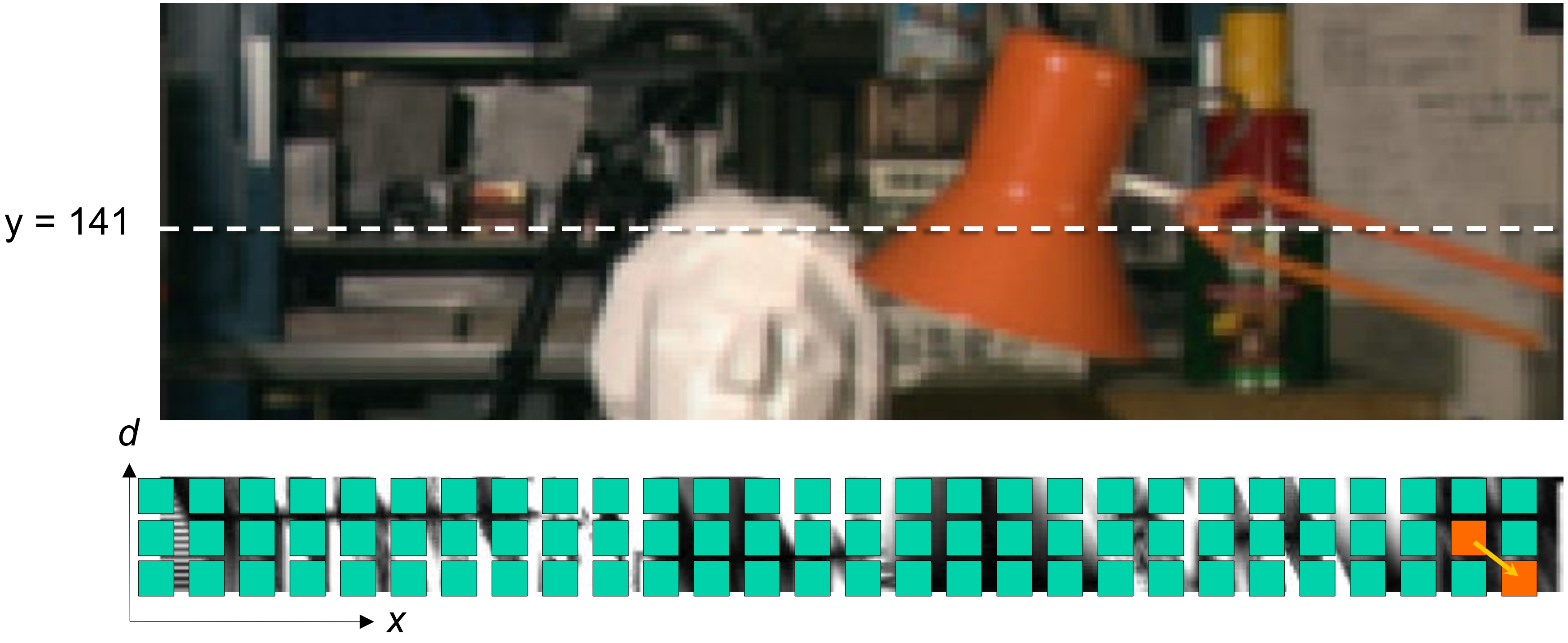
Fill D all the way to the right

Dynamic Programming



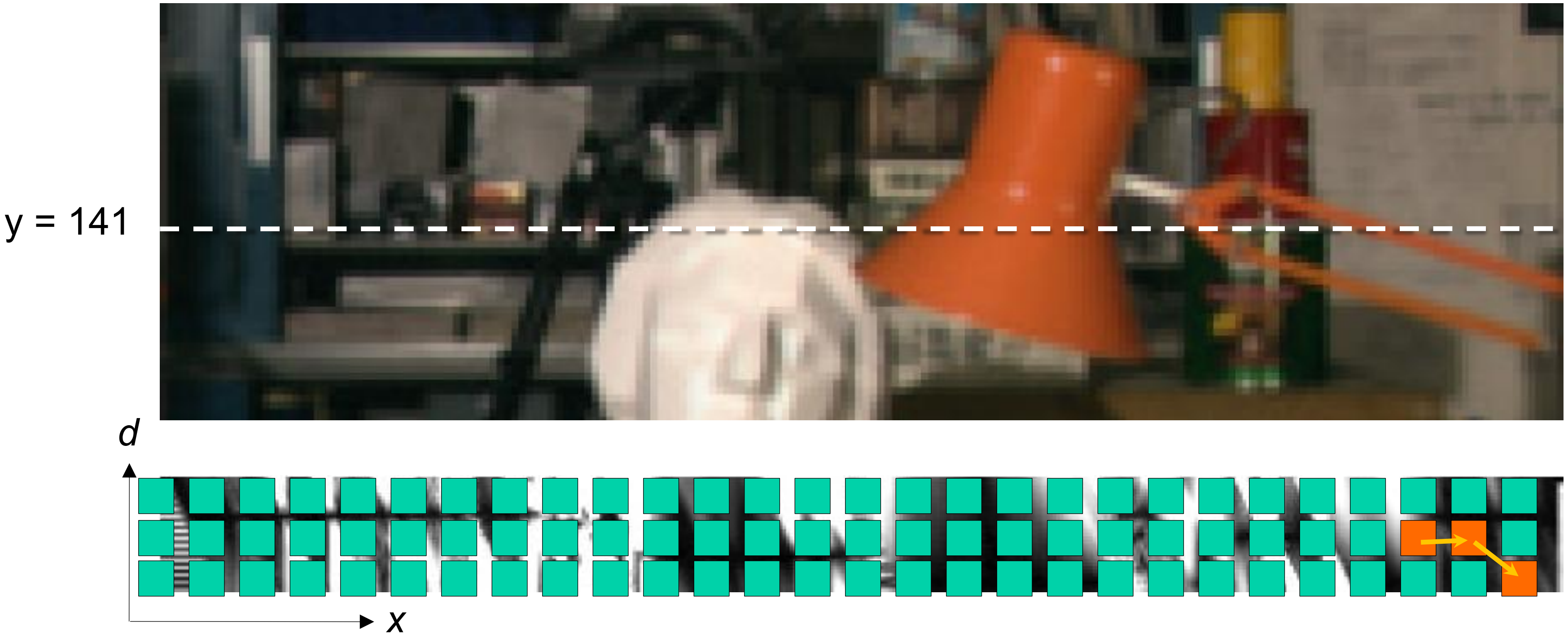
Pick the best one from the right column

Dynamic Programming



Back track....

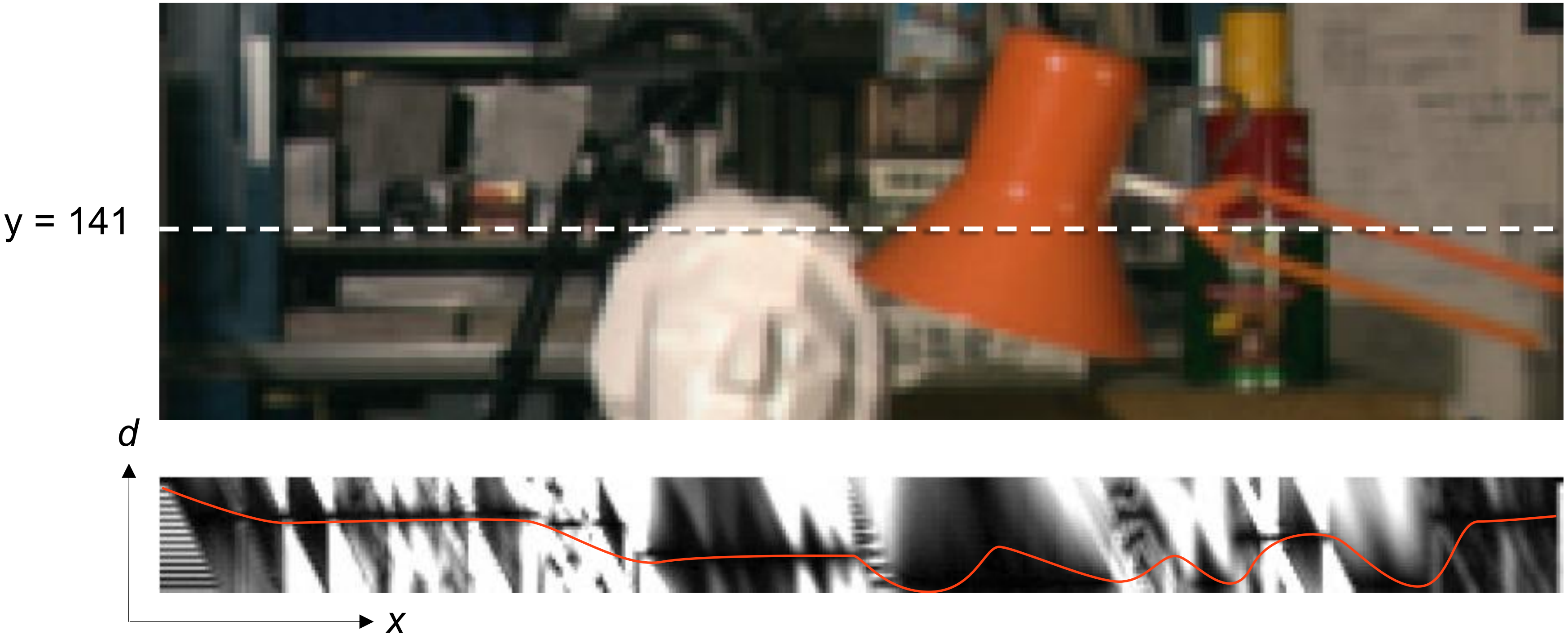
Dynamic Programming



Back track....

Dynamic Programming

- Final result at one row



Dynamic Programming Results

- Suffers from streak errors

Semi-Global Method

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

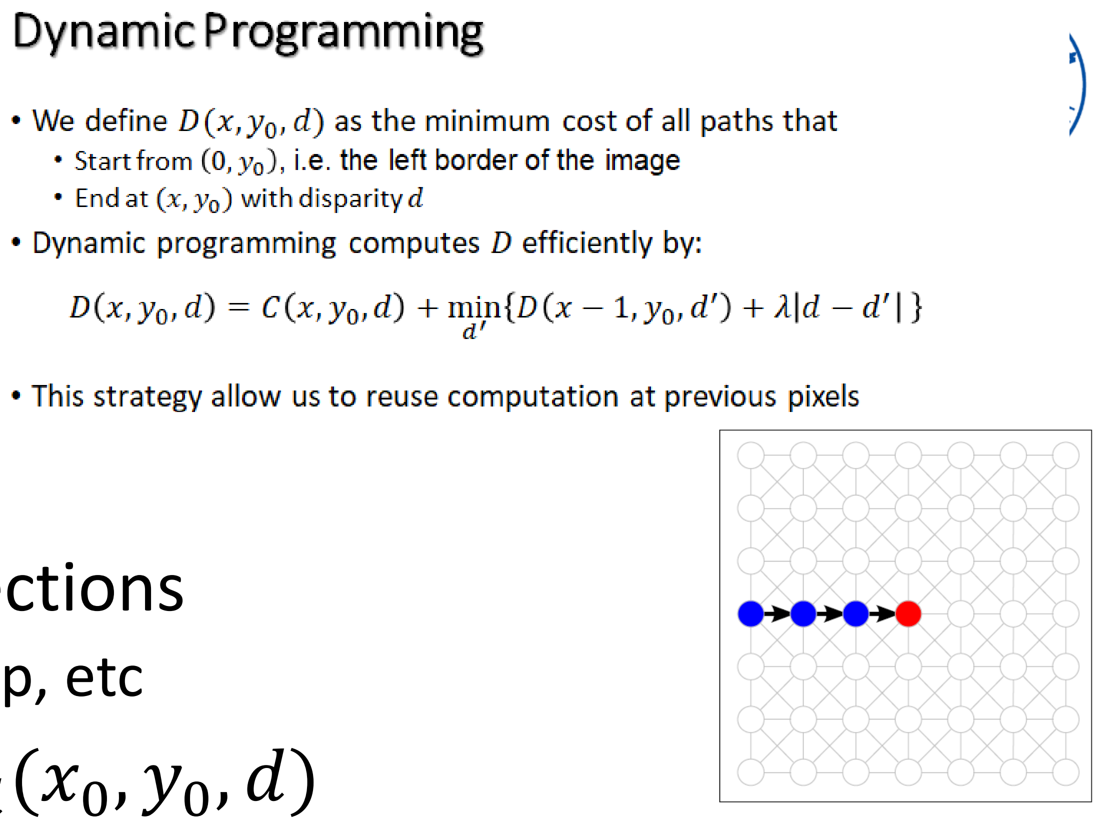
- Aim to minimize this global energy function
- Between local and global optimization
- Utilize dynamic programming for *cost aggregation*
- Faster than global method, with similar accuracy

Stereo Processing by Semiglobal Matching and Mutual Information

Heiko Hirschmüller

[PAMI 2008]

Semi-Global Method



Dynamic Programming

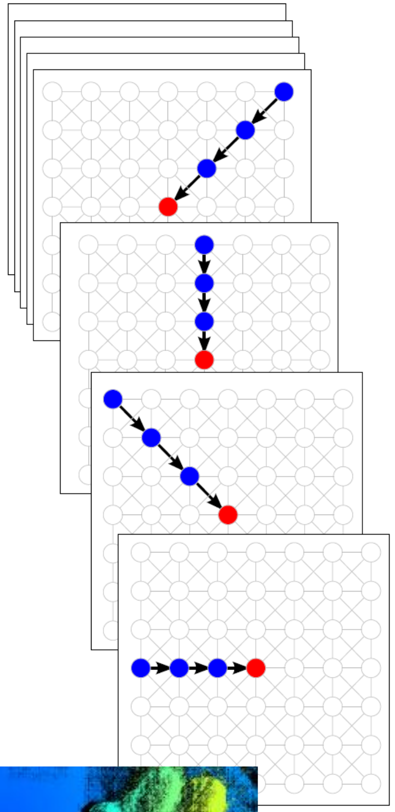
- We define $D(x, y_0, d)$ as the minimum cost of all paths that
  - Start from $(0, y_0)$, i.e. the left border of the image
  - End at (x, y_0) with disparity d
- Dynamic programming computes D efficiently by:

$$D(x, y_0, d) = C(x, y_0, d) + \min_{d'}\{D(x-1, y_0, d') + \lambda|d - d'|\}$$

- This strategy allow us to reuse computation at previous pixels

- Recall the function $D(x, y_0, d)$ defined in Page 34
- It aggregates cost from left to right
- We could aggregate cost from other directions
  - From right to left, top -> down, bottom -> up, etc
- For each direction i, define a function $D_i(x_0, y_0, d)$
  - That is the minimum cost among all paths that along the i-th direction
  - Start from the image border
  - End at the pixel (x_0, y_0) with disparity d
- D_i can be evaluated by DP as well

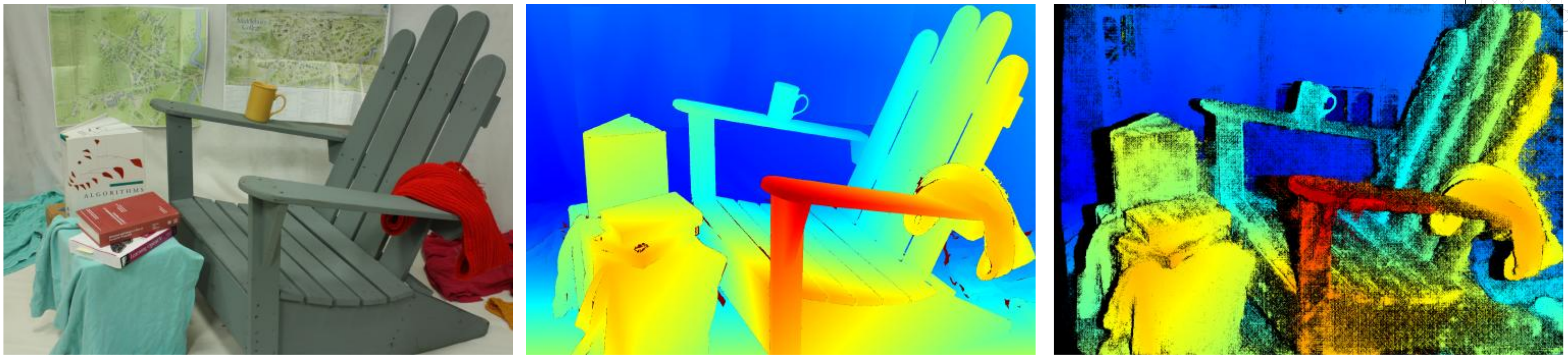
Semi-Global Method



- With D_i computed for all directions, the final cost is

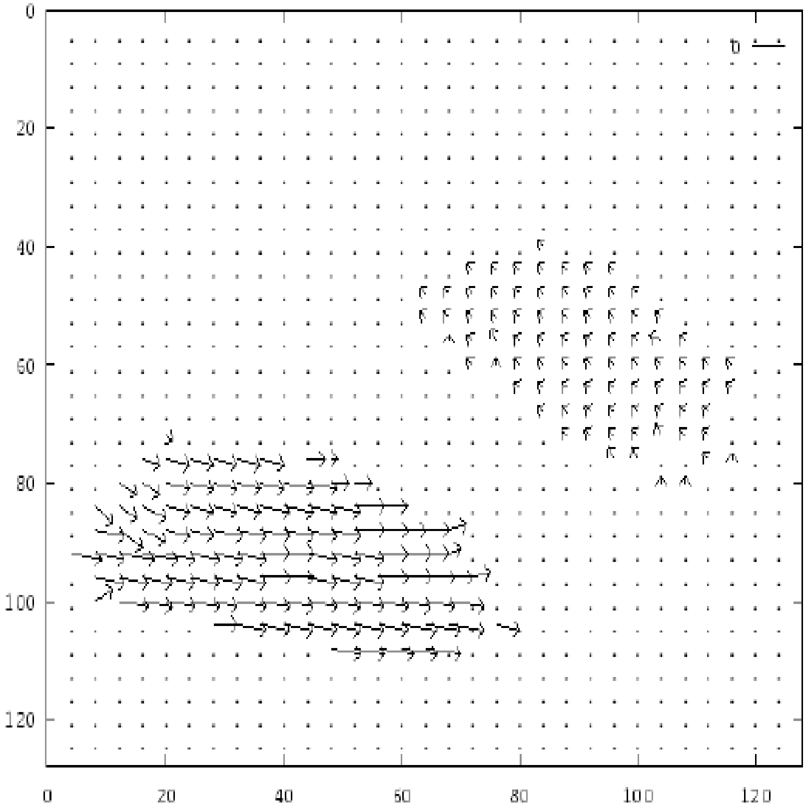
$$S(x_0, y_0, d) = \sum_i D_i(x_0, y_0, d)$$

- Finally, apply Winter-Take-All at each pixel to choose the optimal d

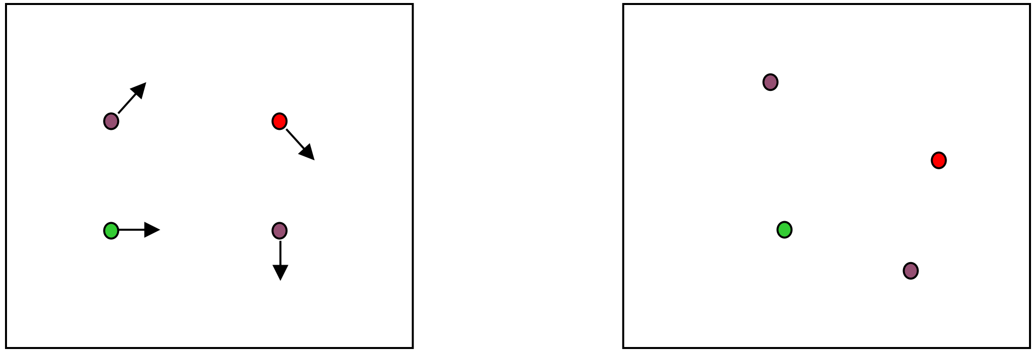


Questions?

Optical Flow



Problem Definition: optical flow

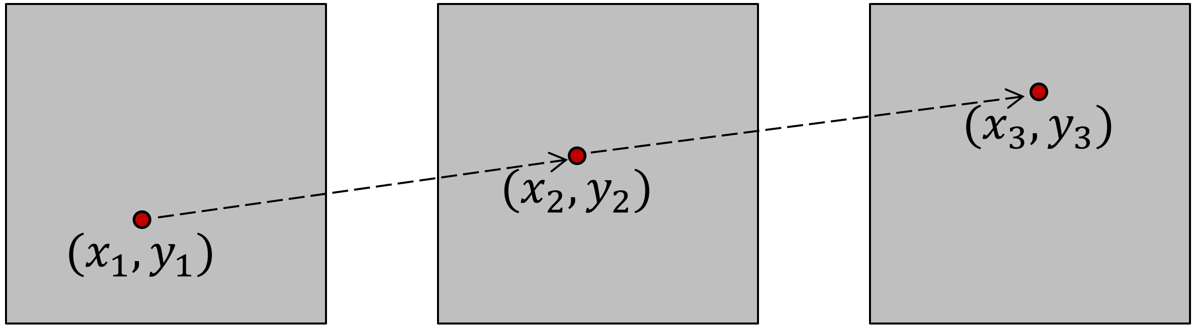


$H(x,y)$ $I(x,y)$

- How to estimate pixel motion from image H to image I?
  - Solve pixel correspondence problem
    - given a pixel in H, look for nearby pixels of the same color in I
  - Difference from stereo: no epipolar lines, dynamic scenes
  - Key assumptions
    - color constancy: a point in H looks the same in I
      - For grayscale images, this is brightness constancy
    - small motion: points do not move very far

Brightness Constancy

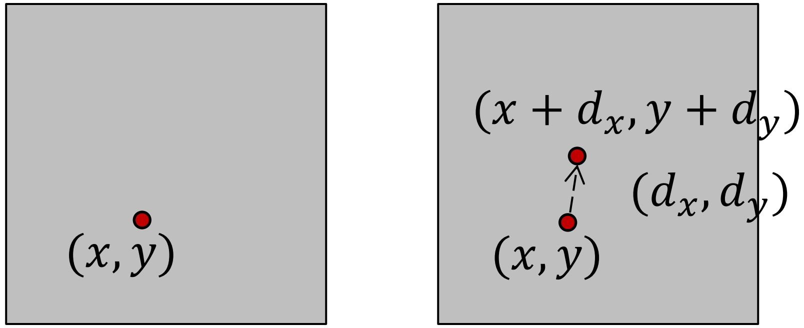
- A scene point moves through multiple frames
- The brightness of the point remains the same



$$I(x(t), y(t), t) = \underset{\text{constant}}{c}$$

Small Motion

$$I(x, y, t) = I(x + d_x, y + d_y, t + d_t)$$



- Assume the motion is small
  - Taylor expansion:

$$I(x, y, t) = I(x, y, t) + I_x d_x + I_y d_y + I_t d_t$$

$$\rightarrow \qquad I_x d_x + I_y d_y + I_t d_t = 0$$

The Optical Flow Equation

$$I_x d_x + I_y d_y + I_t d_t = 0$$

- Divide by d_t

$$I_x \underbrace{\frac{d_x}{d_t}}_{u} + I_y \underbrace{\frac{d_y}{d_t}}_{v} + I_t = 0$$

(u, v) is the velocity

- The optical flow equation:

$$\boxed{I_x u + I_y v + I_t = 0}$$

  - Derived from brightness constancy and small motion

The Optical Flow Equation

$$\boxed{I_x u + I_y v + I_t = 0}$$

- I_x, I_y are the image gradients, known
  - Computed by difference, e.g., Sobel filter, Scharr filter, etc.

- I_t is the temporal gradient, known
  - Computed by frame difference

- u, v are the velocity of a pixel, unknown
  - Solve them by optical flow algorithm

Aperture Problem

The optical flow equation:

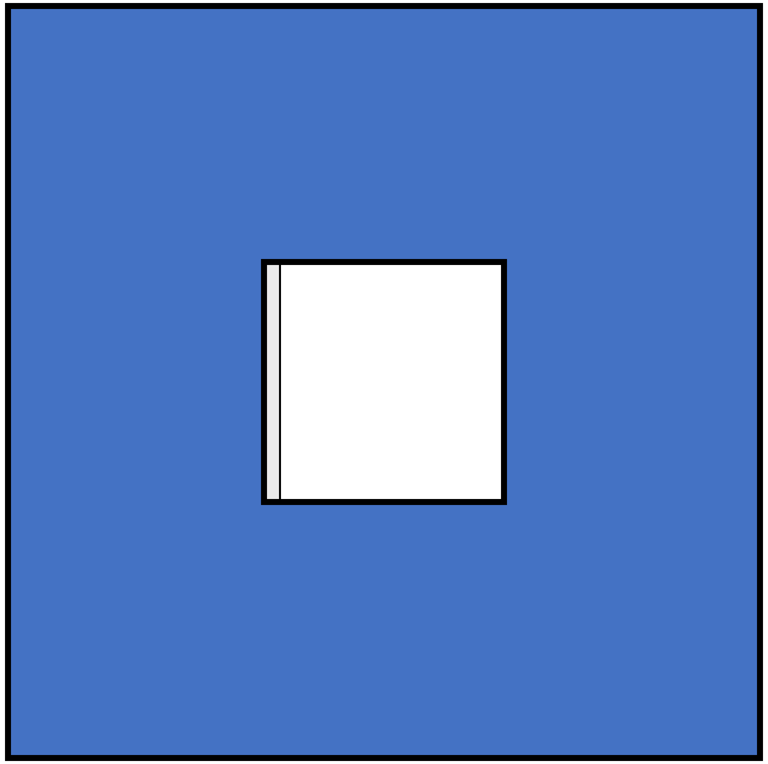
$$\boxed{I_x u + I_y v + I_t = 0}$$

At each pixel, we have:

- One equation
- Two unknowns: u, v
- Multiple solutions
  (because of less equation than unknowns)

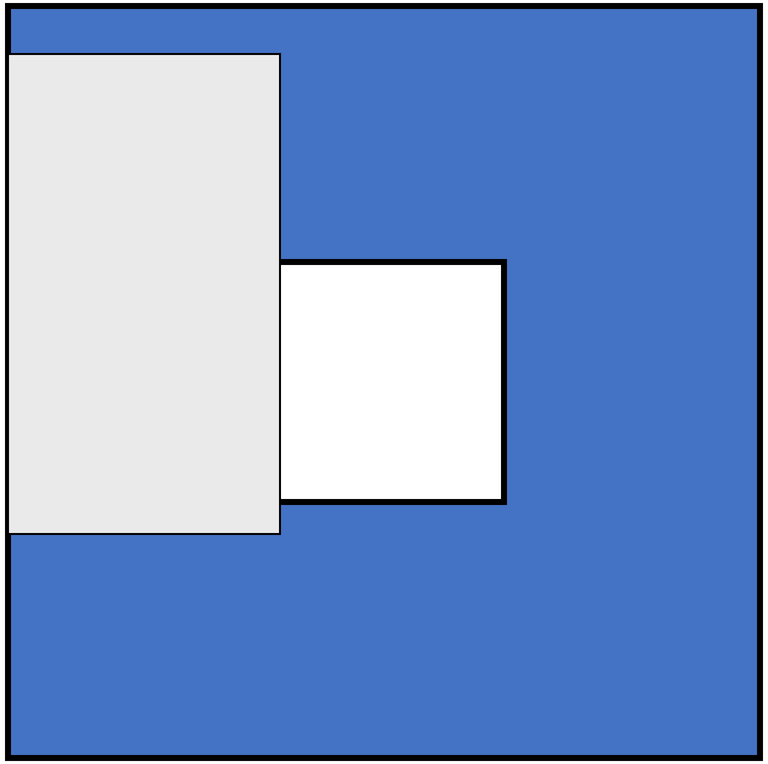
Aperture Problem

Example I



Aperture Problem

Example I



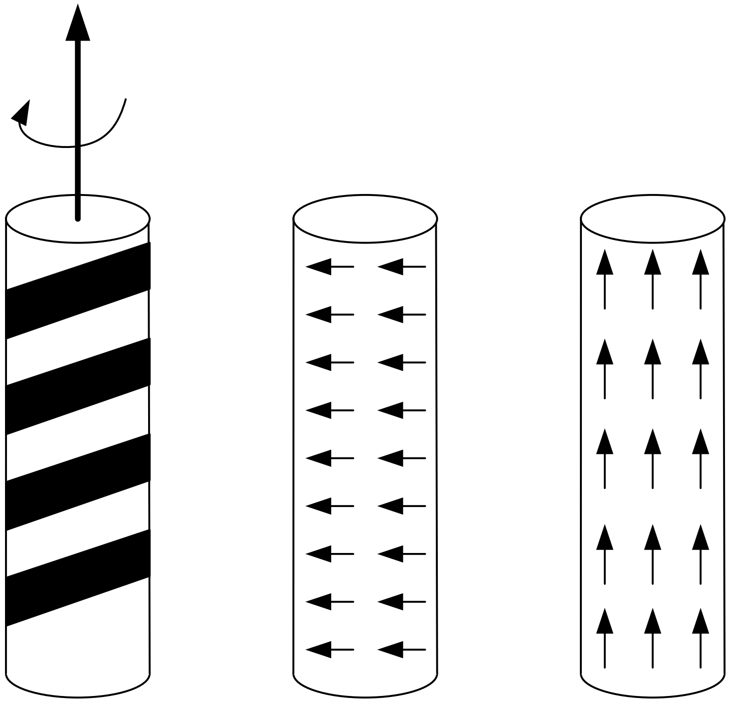
Aperture Problem

Example II

barber's pole

Aperture Problem

Example II



barber's pole **motion field** **optical flow**

Questions?

Lucas & Kanade Method

Local continuous constraint –

- Each pixel gives a optical flow equation

$$I_x(i,j)u(i,j) + I_y(i,j)v(i,j) + I_t(i,j) = 0$$

  - Two unknowns for each equation (aperture problem)
- Assume the motion is the constant locally to get more equations
- How many equations do we get from a 5x5 block?

$$\begin{gathered} I_x(p_1)u + I_y(p_1)v + I_t(p_1) = 0; \\ I_x(p_2)u + I_y(p_2)v + I_t(p_2) = 0; \\ \vdots \\ I_x(p_{25})u + I_y(p_{25})v + I_t(p_{25}) = 0; \end{gathered}$$

Lucas & Kanade Method

Matrix form

- Using a 5x5 patch, gives us 25 equations

$$\begin{bmatrix} I_x(p_1) & I_y(p_1) \\ I_x(p_2) & I_y(p_2) \\ \vdots & \vdots \\ I_x(p_{25}) & I_y(p_{25}) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} I_t(p_1) \\ I_t(p_2) \\ \vdots \\ I_t(p_{25}) \end{bmatrix}$$

$$A_{25\times 2} \qquad x_{2\times 1} \qquad b_{25\times 1}$$

Lucas & Kanade Method

To obtain the solution, we need to solve:

$$\begin{array}{ccc} A^T A & x & A^T b \\ \begin{bmatrix} \sum_p I_x I_x & \sum_p I_x I_y \\ \sum_p I_x I_y & \sum_p I_y I_y \end{bmatrix} & \begin{bmatrix} u \\ v \end{bmatrix} = & \begin{bmatrix} \sum_p I_x I_t \\ \sum_p I_y I_t \end{bmatrix} \end{array}$$

which gives:

$$x = (A^T A)^{-1} A^T b$$

Lucas & Kanade Method

$$A^T A = \begin{bmatrix} \sum_p I_x I_x & \sum_p I_x I_y \\ \sum_p I_x I_y & \sum_p I_y I_y \end{bmatrix}$$

- When is $A^T A$ invertible?
  - λ_1, λ_2 should be large
- Where have you seen $A^T A$ before?
  - Harris corner detector
  - Corners are when λ_1, λ_2 are big; this is also when Lucas-Kanade optical flow works best
  - Corners are regions with two different directions of gradient (at least)
  - Corners are good places to compute flow!

Horn-Schunck vs Lucas-Kanade

- Lucas-Kanade assume locally constant flow
- Horn-Schunk assume globally smooth flow
  - The flow vectors at neighboring pixels should be similar
  - Just like the smoothness term on disparity at neighboring pixels

Horn-Schunck Optical Flow (1981)	**Lucas-Kanade Optical Flow (1981)**
brightness constancy small motion	method of differences
'smooth' flow (flow can vary from pixel to pixel)	**'constant' flow** (flow is constant for all pixels)
global method (dense)	local method (sparse)

Horn & Schunck Method

- Two key constraints:
  - Optical flow equation (from brightness constancy)
  - Global smoothness
- Optical flow equation: $I_x u + I_y v + I_t = 0$
- Solving these equations at each pixel equivalents to minimize:

$$E_d = \sum_{i,j} \left[I_x(i,j)u(i,j) + I_y(i,j)v(i,j) + I_t(i,j) \right]^2$$

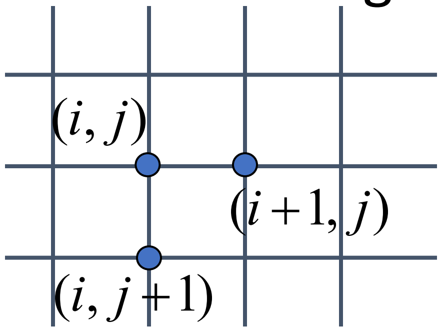
  - Sometimes, written in integration form:

$$E_d = \iint \left[I_x(x,y)u(x,y) + I_y(x,y)v(x,y) + I_t(x,y) \right]^2 dxdy$$

Horn & Schunck Method

- Global smoothness constraint –

neighboring pixels have similar motion



$$u(i,j) \approx u(i+1,j); \quad v(i,j) \approx v(i+1,j);$$
$$u(i,j) \approx u(i,j+1); \quad v(i,j) \approx v(i,j+1);$$

Or, equivalently, in continuous form:

$$u_x \approx u_y \approx 0; \quad v_x \approx v_y \approx 0$$

- Enforcing smoothness at each pixel equivalents to minimize:

$$E_s = \sum_{i,j} \left[\left(u_{ij} - u_{i+1,j}\right)^2 + \left(u_{ij} - u_{i,j+1}\right)^2 + \left(v_{ij} - v_{i+1,j}\right)^2 + \left(v_{ij} - v_{i,j+1}\right)^2 \right]$$

  – Sometimes, written in integration form:

$$E_s = \iint [|\nabla u|^2 + |\nabla v|^2] dxdy = \iint \left[u_x^2 + u_y^2 + v_x^2 + v_y^2\right] \; dxdy$$

Horn & Schunck Method

Minimize the combined energy

$$\{u, v\} = \arg\min(\lambda E_d + E_s)$$

- In discrete form:

$$E_d = \sum_{i,j} \left[I_x(i,j)u(i,j) + I_y(i,j)v(i,j) + I_t(i,j)\right]^2$$

$$E_s = \sum_{i,j} \left[\left(u_{ij} - u_{i+1,j}\right)^2 + \left(u_{ij} - u_{i,j+1}\right)^2 + \left(v_{ij} - v_{i+1,j}\right)^2 + \left(v_{ij} - v_{i,j+1}\right)^2\right]$$

- In continuous form:

$$\lambda E_d + E_s = \iint \lambda\left(I_x u + I_y v + I_t\right)^2 + (|\nabla u|^2 + |\nabla v|^2)dxdy$$

Horn & Schunck Method

Minimize by set partial derivative to 0:

- In discrete form: $\frac{\partial(E_d+E_s)}{\partial u_{ij}} = 0$

$$\frac{\partial(E_d+E_s)}{\partial u_{ij}} = 2(u_{ij}-\bar{u}_{ij}) + 2\lambda(I_x u_{ij} + I_y v_{ij} + I_t)I_x = 0$$

$$\frac{\partial(E_d+E_s)}{\partial v_{ij}} = 2(v_{ij}-\bar{v}_{ij}) + 2\lambda(I_x u_{ij} + I_y v_{ij} + I_t)I_y = 0$$

Local average: $\bar{u}_{ij} = \frac{1}{4}\{u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1}\}$

After simple manipulations,

$$(1+\lambda I_x^2)u_{ij} + \lambda I_x I_y v_{ij} = \bar{u}_{ij} - \lambda I_x I_t$$
$$\lambda I_x I_y u_{ij} + (1+\lambda I_y^2)v_{ij} = \bar{v}_{ij} - \lambda I_y I_t$$

This becomes a large linear system

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$$

$\boldsymbol{A}$ is a large sparse matrix!

Horn & Schunck Method

A simple iterative solver:

$$(1+\lambda I_x^2)u_{ij}^{n+1} + \lambda I_x I_y v_{ij}^{n+1} = \bar{u}_{ij}^n - \lambda I_x I_t$$
$$\lambda I_x I_y u_{ij}^{n+1} + \left(1+\lambda I_y^2\right) v_{ij}^{n+1} = \bar{v}_{ij}^n - \lambda I_y I_t$$

- Solve a 2×2 inverse matrix at each pixel

$$u_{ij}^{n+1} = \bar{u}_{ij}^n - \frac{I_x \bar{u}_{ij}^n + I_y \bar{v}_{ij}^n + I_t}{\lambda^{-1} + \left(I_x^2 + I_y^2\right)} I_x$$
$$v_{ij}^{n+1} = \bar{v}_{ij}^n - \frac{I_x \bar{u}_{ij}^n + I_y \bar{v}_{ij}^n + I_t}{\lambda^{-1} + \left(I_x^2 + I_y^2\right)} I_y$$

Summary

1. Precompute image gradient I_x, I_y
2. Precompute temporal gradient I_t
3. Initialize flow $u = v = 0$
4. While not converge

    Compute flow field updates for each pixel:

$$u_{ij}^{n+1} = \bar{u}_{ij}^{n} - \frac{I_x \bar{u}_{ij}^{n} + I_y \bar{v}_{ij}^{n} + I_t}{1 + \lambda\left(I_x^2 + I_y^2\right)} I_x$$

$$v_{ij}^{n+1} = \bar{v}_{ij}^{n} - \frac{I_x \bar{u}_{ij}^{n} + I_y \bar{v}_{ij}^{n} + I_t}{1 + \lambda\left(I_x^2 + I_y^2\right)} I_y$$

Horn & Schunck Method

- The continuous version can be minimized similarly
  - $$E = \lambda E_d + E_s = \iint \lambda\left(I_x u + I_y v + I_t\right)^2 + (|\nabla u|^2 + |\nabla v|^2)dxdy$$
- Want to evaluate $\frac{\partial E}{\partial u}$, but u itself is a function
  - E is a function of a function, called functional
  - Minimize a functional = solve its Euler equation $\frac{\partial E}{\partial u} = 0$
  - In our case, the Euler equations are:

$$\Delta u - \lambda\left(I_x u + I_y v + I_t\right) = 0,$$
$$\Delta v - \lambda\left(I_x u + I_y v + I_t\right) = 0.$$
$$\Delta u = \bar{u} - u$$

Questions?

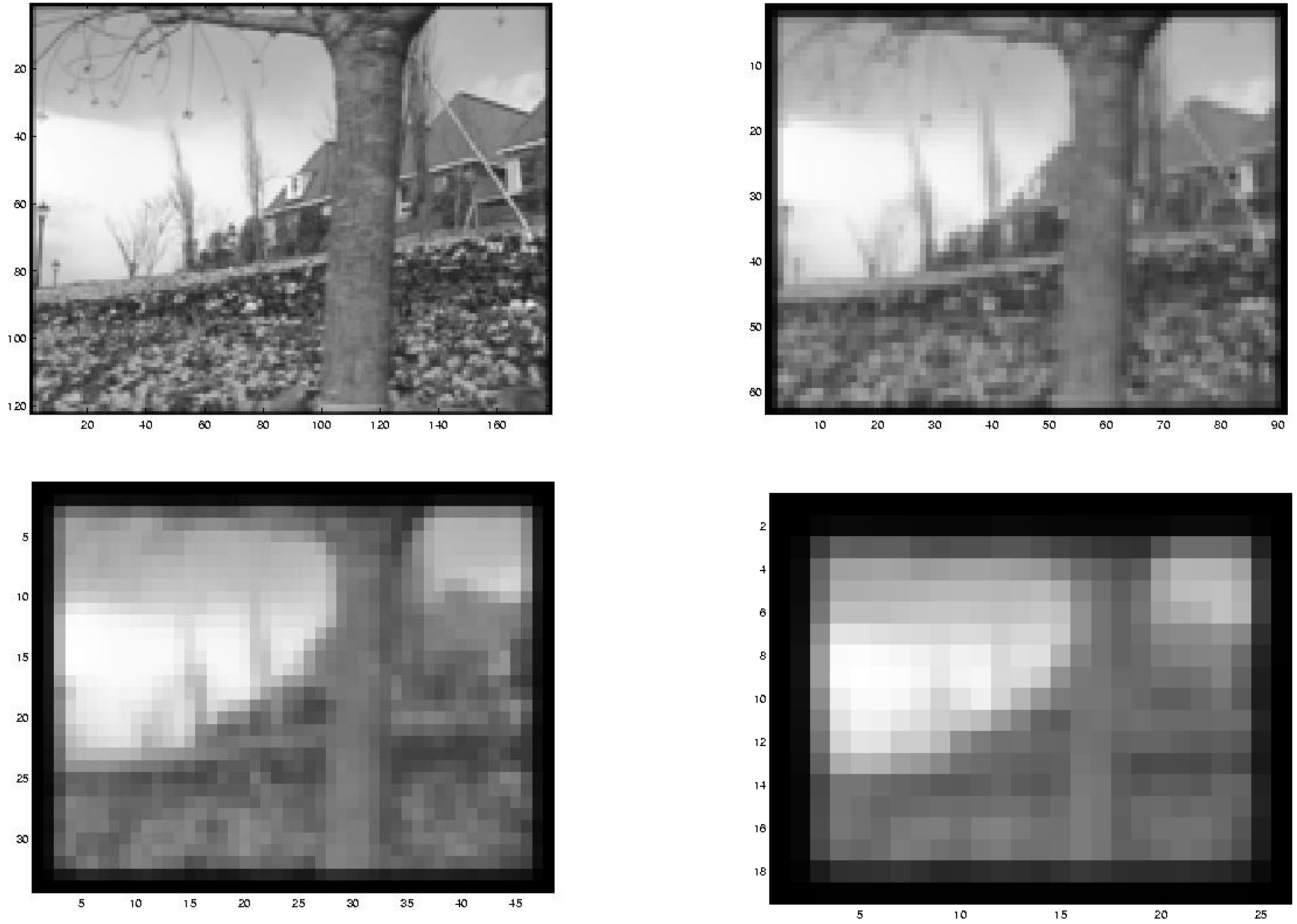
Modern Techniques for Optical Flow

- Warping
- Gradient constancy
- Robust cost function
- Median / Bilateral filter

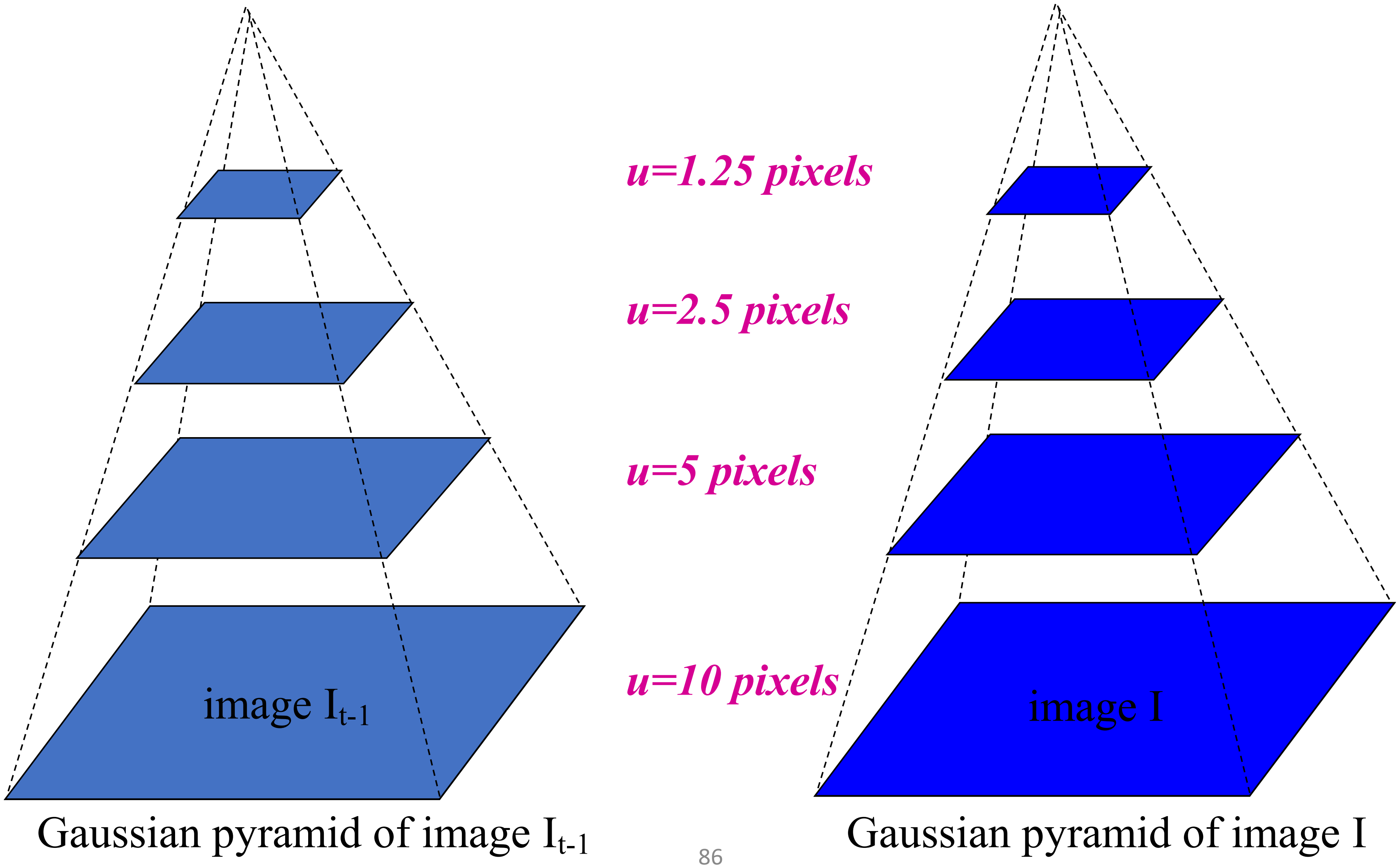
Warping

- Revisiting the small motion assumption
- Is this motion small enough?
  - Probably not—it’s much larger than one pixel
  - How might we solve this problem?

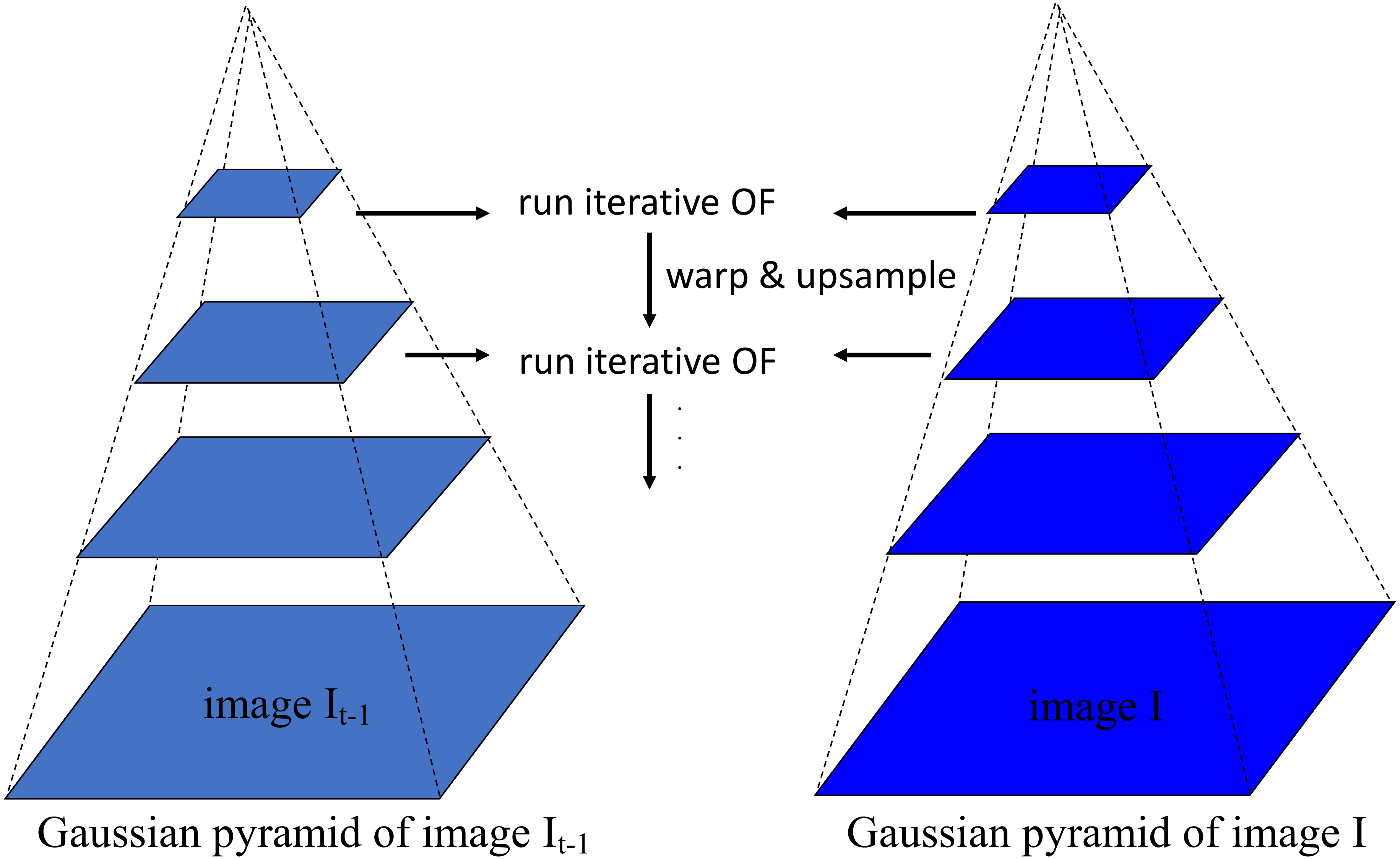
reduce the resolution!



coarse-to-fine optical flow estimation



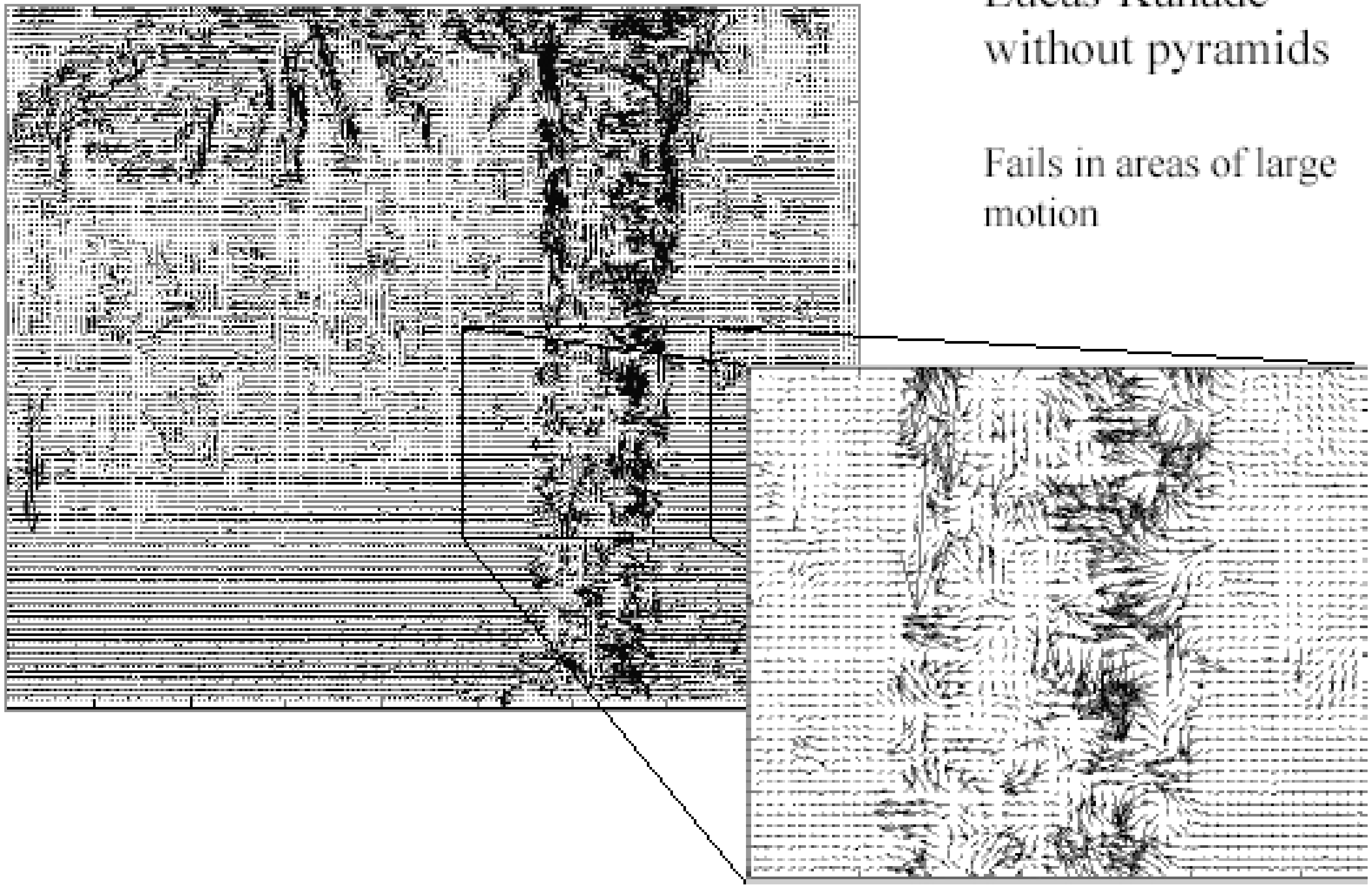
coarse-to-fine optical flow estimation



Warping based Optical Flow

1. Compute optical flow at the highest pyramid level
2. At level i
    1. Take flow u_{i-1}, v_{i-1} from level $i-1$
    2. Interpolate it to create u_i^*, v_i^* of twice resolution for level i
    3. Multiply u_i^*, v_i^* by 2
    4. Warp $I(t-1)$ by u_i^*, v_i^*, denote the warped image as $I^*(t-1)$
    5. Compute correction flow u_i', v_i' between $I^*(t-1)$ and $I(t)$
    6. Add corrections, i.e. $u_i = u_i^* + u_i'; \;\; v_i = v_i^* + v_i'$.

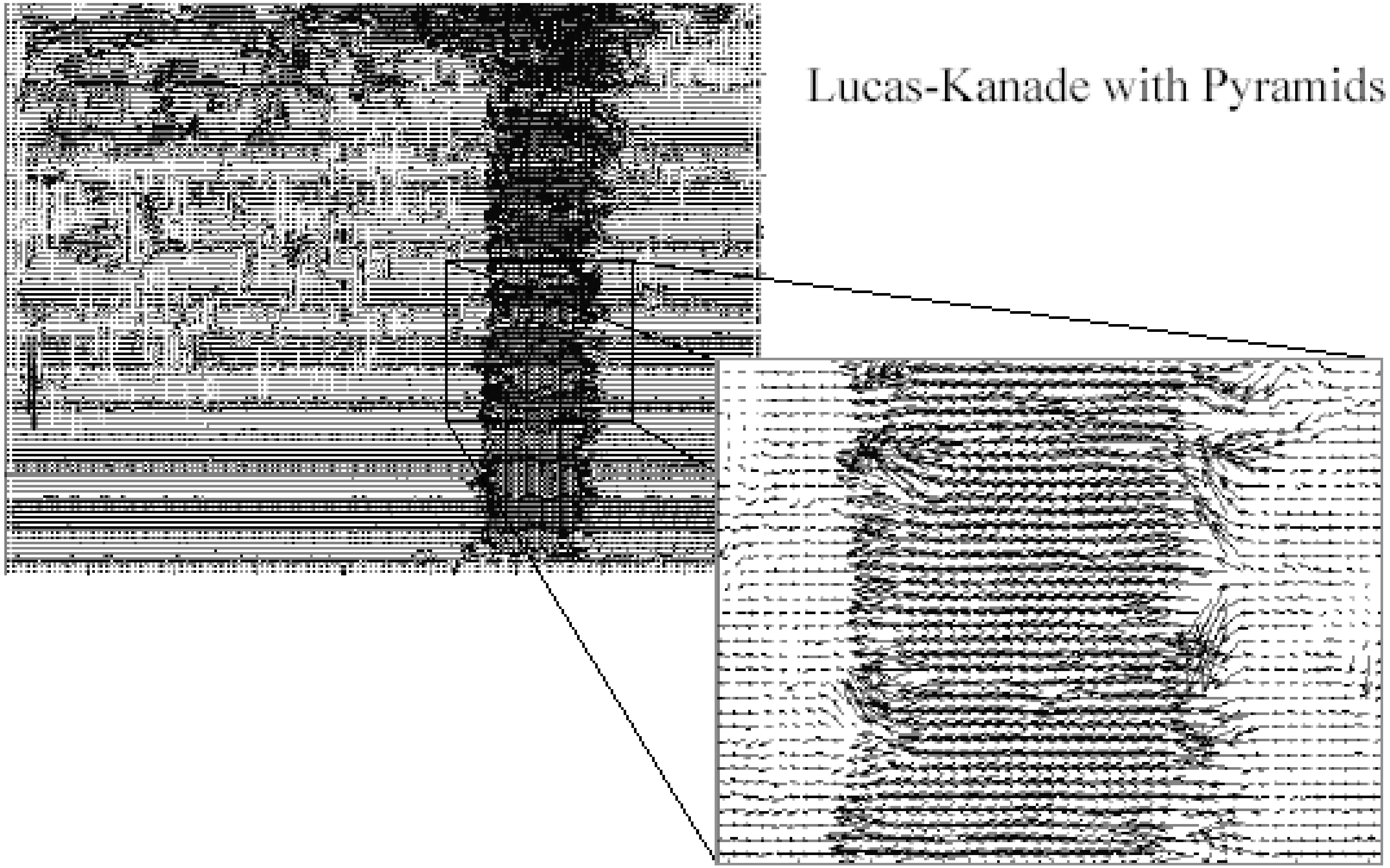
examples



Lucas-Kanade
without pyramids

Fails in areas of large motion

examples



Lucas-Kanade with Pyramids

Gradient Constancy

- The brightness constancy assumption might be wrong
  - Due to illumination/exposure/white balance changes
- Add gradient constancy

$$\nabla I_t(x, y) = \nabla I_{t+1}(x + d_x, y + d_y)$$

  - Robust to illumination changes
  - But violated by rotations
- It might be better to switch between color and gradient constancy instead of directly combine them

Motion Detail Preserving Optical Flow Estimation*

Li Xu Jiaya Jia
The Chinese University of Hong Kong
{xuli,leojia}@cse.cuhk.edu.hk

Yasuyuki Matsushita
Microsoft Research Asia
yasumat@microsoft.com

[CVPR 2010]

Robust Cost Function

- Instead of enforcing L2 penalty on color constancy, M-estimator can be used here
  - Recall the Horn-Schunk formulation

$$\iint \left(I_x u + I_y v + I_t\right)^2 + (|\nabla u|^2 + |\nabla v|^2) dxdy$$

  - A commonly used loss is L1 penalty, or the Charbonnier penalty

$$\rho(x) = \sqrt{x^2 + \epsilon^2}$$

  - Can be used for both data and smoothness terms

Median/Bilateral Filter

- Top secrets of optical flow estimation: applying a median filter to the flow after the flow update (step 6)
  - It removes outliers and preserves edges
  - A bilateral filter gives even better result
- Effectively, this changes the objective function to:

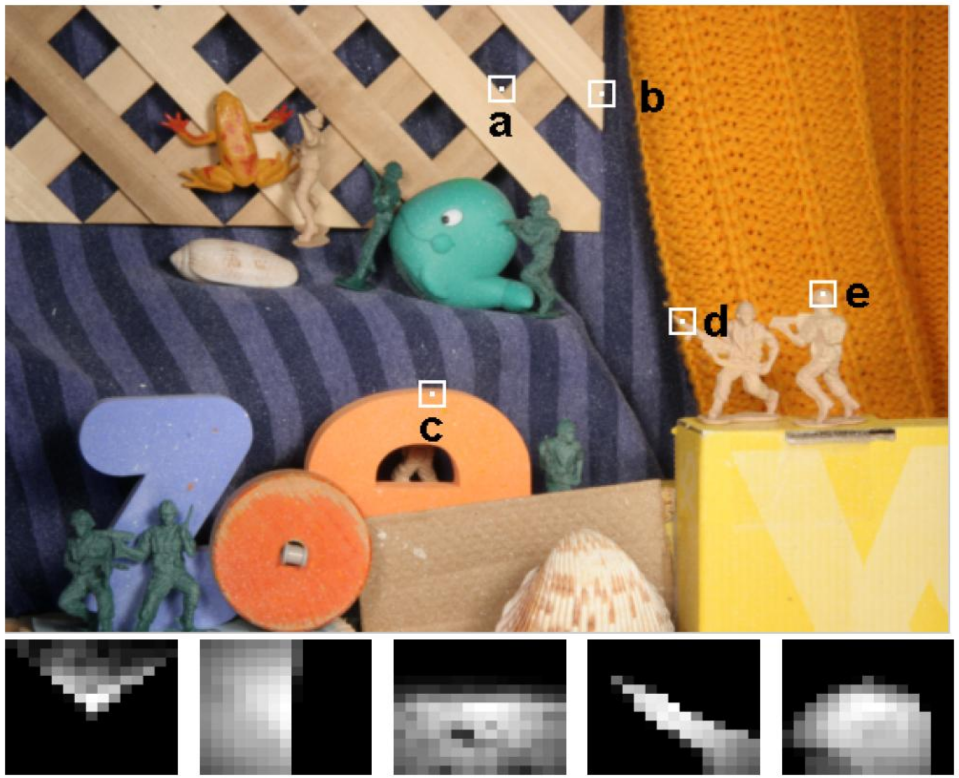
$$E_d(u,v) + \lambda_1 E_s(u,v) + \lambda_2 \underbrace{(|u-\hat{u}| + |v-\hat{v}|)}_{\text{Coupling term}}$$

$$+ \lambda_3 \underbrace{\sum_i \sum_{j \in N_i} \omega_{ij}\left(\left|\hat{u}_i - \hat{u}_j\right| + \left|\hat{v}_i - \hat{v}_j\right|\right)}_{\substack{\text{Energy formulation of the filtering} \\ \text{A better smoothness term}}}$$

Median/Bilateral Filter

- The bilateral weights

$$\omega_{ij} = \underbrace{g_1(|i-j|)}_{\text{Spatial Gaussian}}\underbrace{g_2(I_i - I_j)}_{\text{Range Gaussian}}$$



(a) (b) (c) (d) (e)

Secrets of Optical Flow Estimation and Their Principles

Deqing Sun
Brown University

Stefan Roth
TU Darmstadt

Michael J. Black
Brown University

[CVPR 2010]

Questions?