Corner Detection

CS 543 / ECE 549 – Saurabh Gupta

Slides from S. Lazebnik.

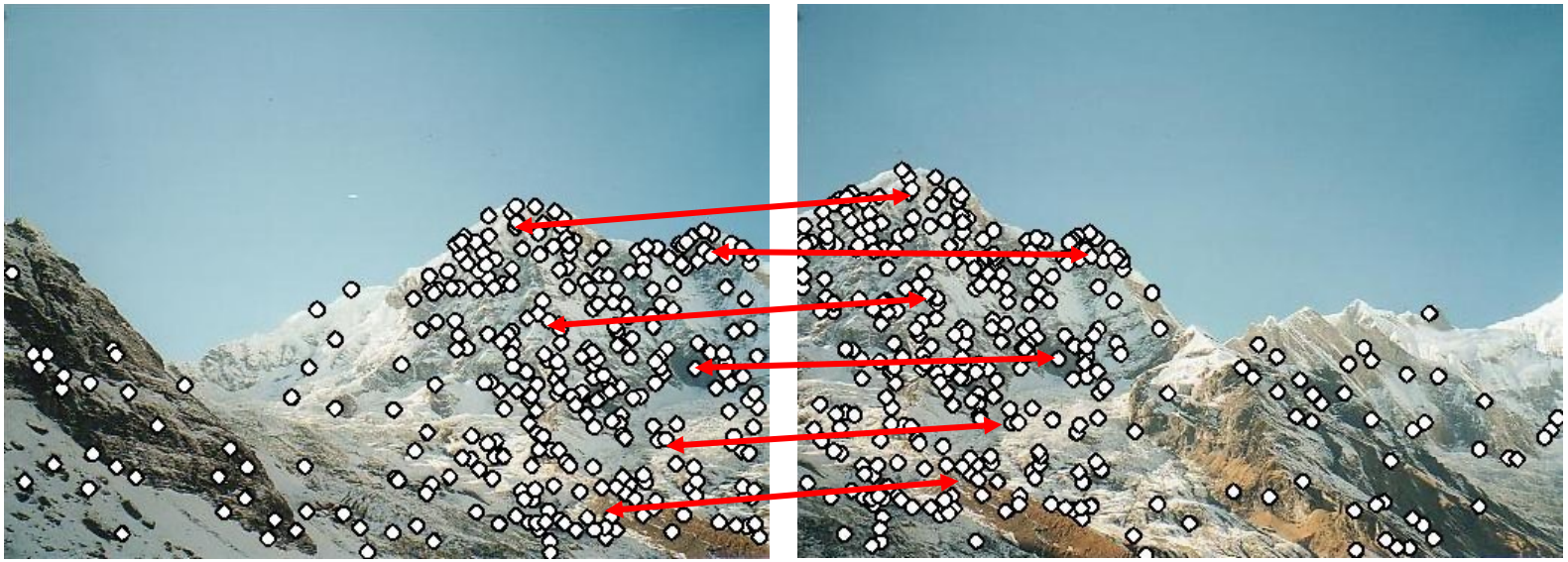
Why extract keypoints?

- Motivation: panorama stitching
  - We have two images – how do we combine them?

Source: L. Lazebnik

Why extract keypoints?

- Motivation: panorama stitching
  - We have two images – how do we combine them?



Step 1: extract keypoints

Step 2: match keypoint features

Source: L. Lazebnik

Why extract keypoints?

- Motivation: panorama stitching
  - We have two images – how do we combine them?

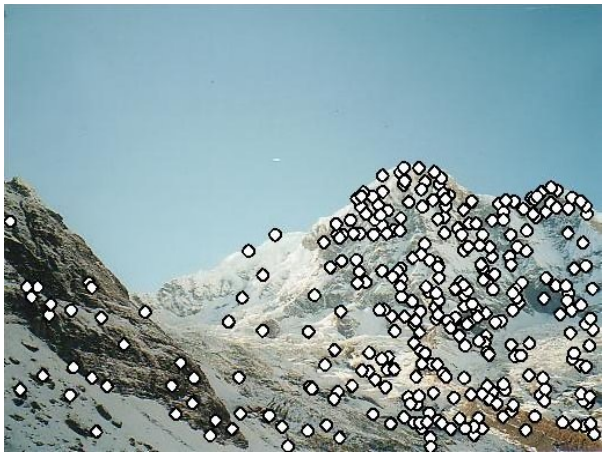
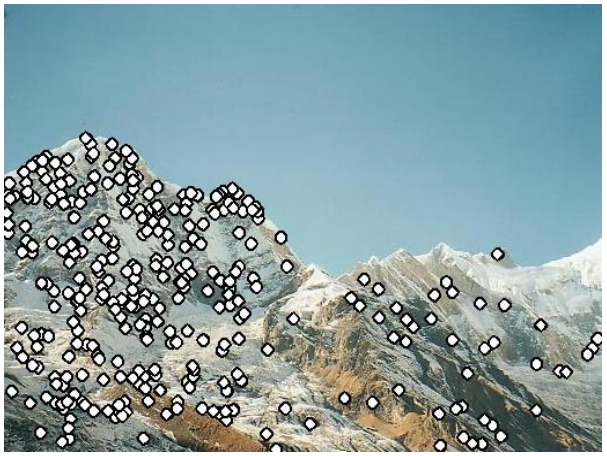
Step 1: extract keypoints

Step 2: match keypoint features

Step 3: align images

Source: L. Lazebnik

Characteristics of good keypoints

- Compactness and efficiency
  - Many fewer keypoints than image pixels
- Saliency
  - Each keypoint is distinctive
- Locality
  - A keypoint occupies a relatively small area of the image; robust to clutter and occlusion
- Repeatability
  - The same keypoint can be found in several images despite geometric and photometric transformations

Source: L. Lazebnik

Applications

Keypoints are used for:

- Image alignment
- 3D reconstruction
- Motion tracking
- Robot navigation
- Database indexing and retrieval
- ~~Object recognition~~

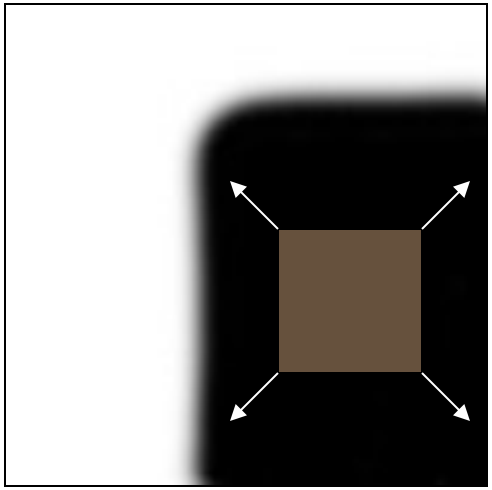
Source: L. Lazebnik

Corner detection: Basic idea

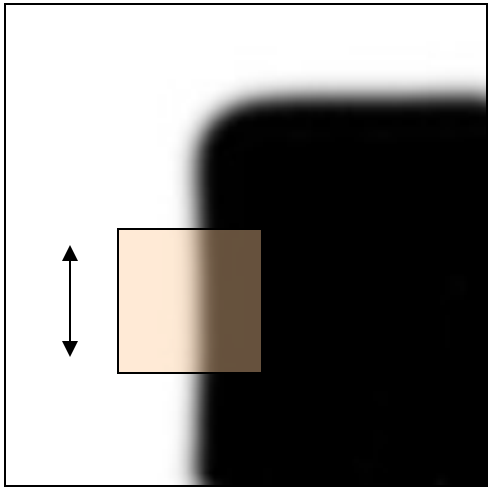
Source: L. Lazebnik

Corner detection: Basic idea

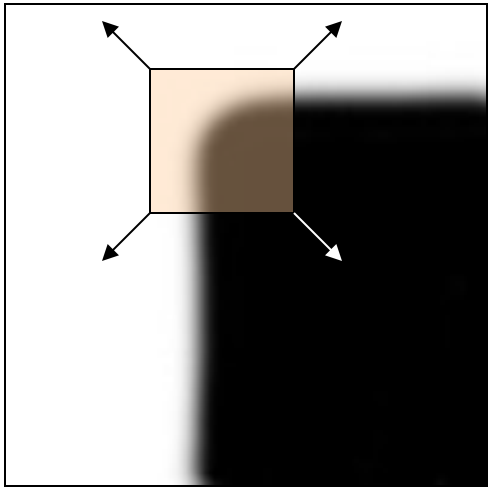
- We should easily recognize the point by looking through a small window
- Shifting a window in *any direction* should give *a large change* in intensity



"flat" region:
no change in all directions



"edge":
no change along the edge direction



"corner":
significant change in all directions

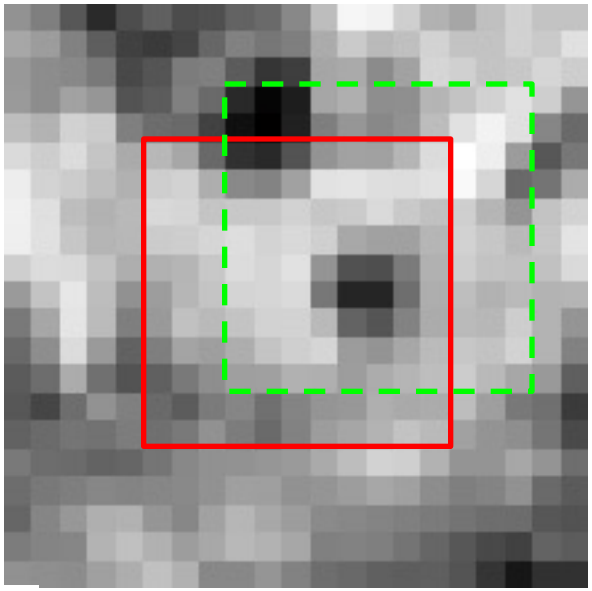
Source: L. Lazebnik

Corner Detection: Derivation

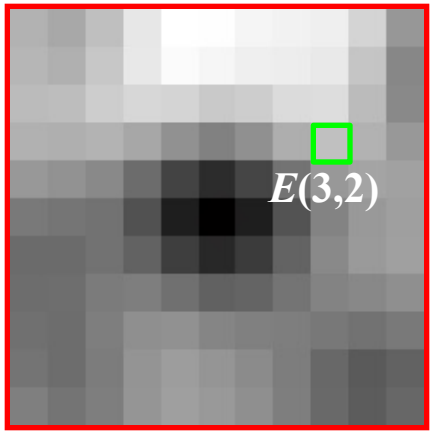
Change in appearance of window *W* for the shift [*u,v*]:

$$E(u,v) = \sum_{(x,y)\in W} [I(x+u, y+v) - I(x,y)]^2$$

$I(x, y)$



$E(u, v)$



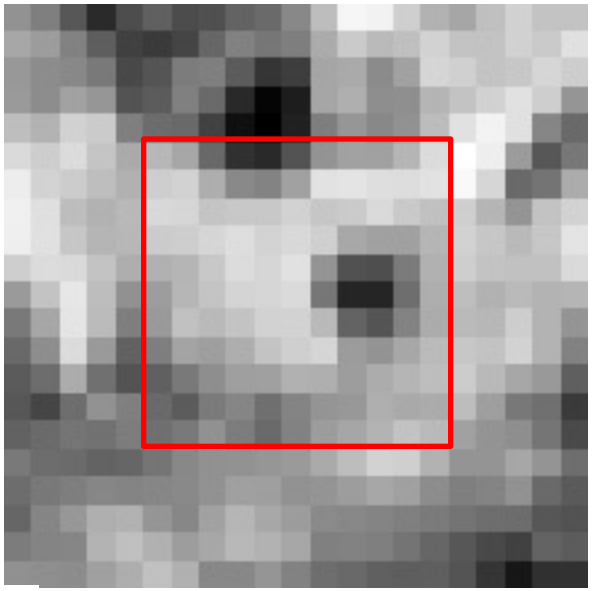
Source: L. Lazebnik

Corner Detection: Derivation

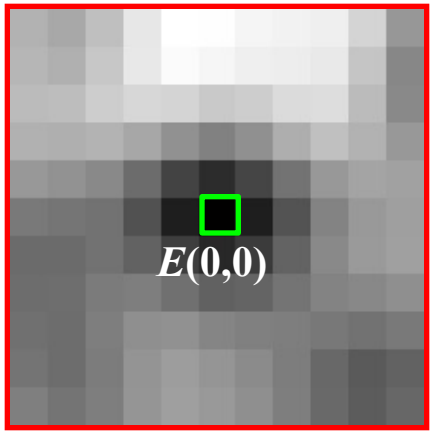
Change in appearance of window *W* for the shift [*u,v*]:

$$E(u,v) = \sum_{(x,y)\in W} [I(x+u, y+v) - I(x,y)]^2$$

$I(x, y)$



$E(u, v)$



Source: L. Lazebnik

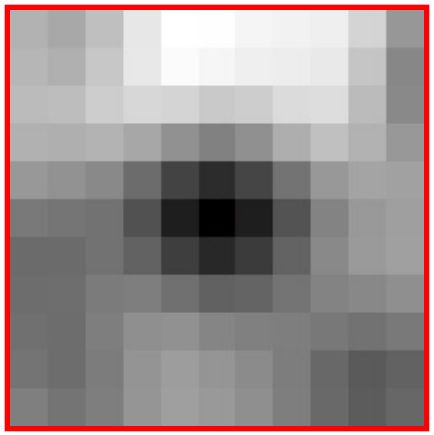
Corner Detection: Derivation

Change in appearance of window *W* for the shift [*u,v*]:

$$E(u,v) = \sum_{(x,y)\in W} [I(x+u, y+v) - I(x,y)]^2$$

We want to find out how this function behaves for small shifts

$E(u, v)$



Source: L. Lazebnik

Corner Detection: Derivation

First-order Taylor approximation for small motions [*u*, *v*]:

$$I(x+u, y+v) \approx I(x,y) + I_x u + I_y v$$

Let's plug this into *E*(*u*,*v*):

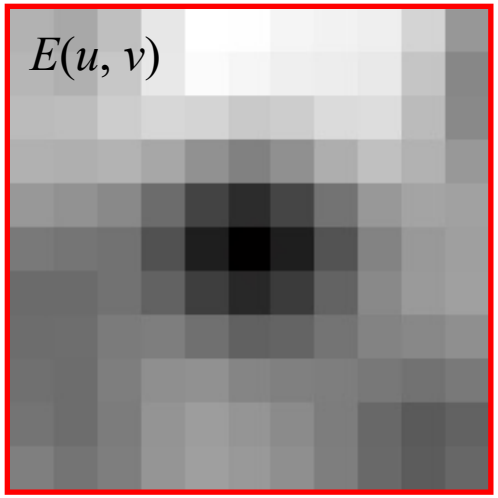
$$E(u,v) = \sum_{(x,y)\in W} [I(x+u, y+v) - I(x,y)]^2$$

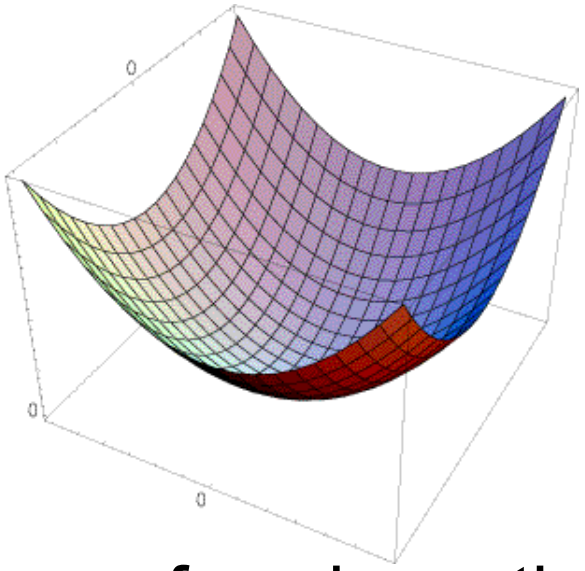
Source: L. Lazebnik

Corner Detection: Derivation

$E(u,v)$ can be locally approximated by a quadratic surface:

$$E(u,v) \approx u^2 \sum_{x,y} I_x^2 + 2uv \sum_{x,y} I_x I_y + v^2 \sum_{x,y} I_y^2$$





In which directions does this surface have the fastest/slowest change?

Source: L. Lazebnik

Corner Detection: Derivation

$E(u,v)$ can be locally approximated by a quadratic surface:

$$E(u,v) \approx u^2 \sum_{x,y} I_x^2 + 2uv \sum_{x,y} I_x I_y + v^2 \sum_{x,y} I_y^2$$

$$= \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} \sum_{x,y} I_x^2 & \sum_{x,y} I_x I_y \\ \sum_{x,y} I_x I_y & \sum_{x,y} I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

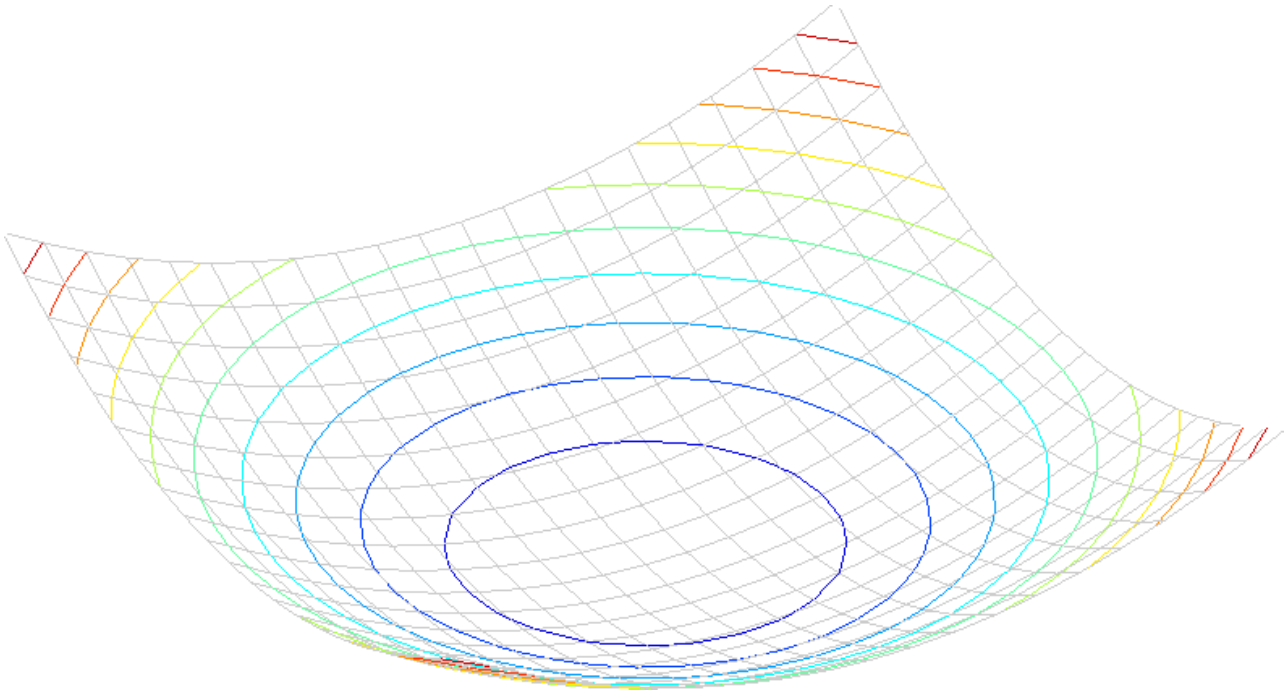
Second moment matrix M

Source: L. Lazebnik

Interpreting the second moment matrix

A horizontal "slice" of $E(u, v)$ is given by the equation of an ellipse:

$$[u \;\; v] \;\; M \begin{bmatrix} u \\ v \end{bmatrix} = \text{const}$$



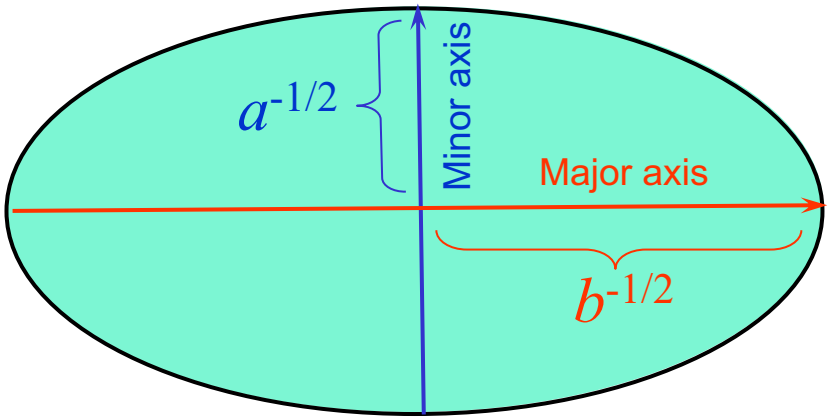
Source: L. Lazebnik

Interpreting the second moment matrix

Consider the axis-aligned case (gradients are either horizontal or vertical):

$$M = \begin{bmatrix} \sum_{x,y} I_x^2 & \sum_{x,y} I_x I_y \\ \sum_{x,y} I_x I_y & \sum_{x,y} I_y^2 \end{bmatrix}$$

$$[u \;\; v] \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = 1$$



Source: L. Lazebnik

Interpreting the second moment matrix

Consider the axis-aligned case (gradients are either horizontal or vertical):

$$M = \begin{bmatrix} \sum_{x,y} I_x^2 & \sum_{x,y} I_x I_y \\ \sum_{x,y} I_x I_y & \sum_{x,y} I_y^2 \end{bmatrix} = \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}$$

If either *a* or *b* is close to 0, then this is **not** a corner, so we want locations where both are large

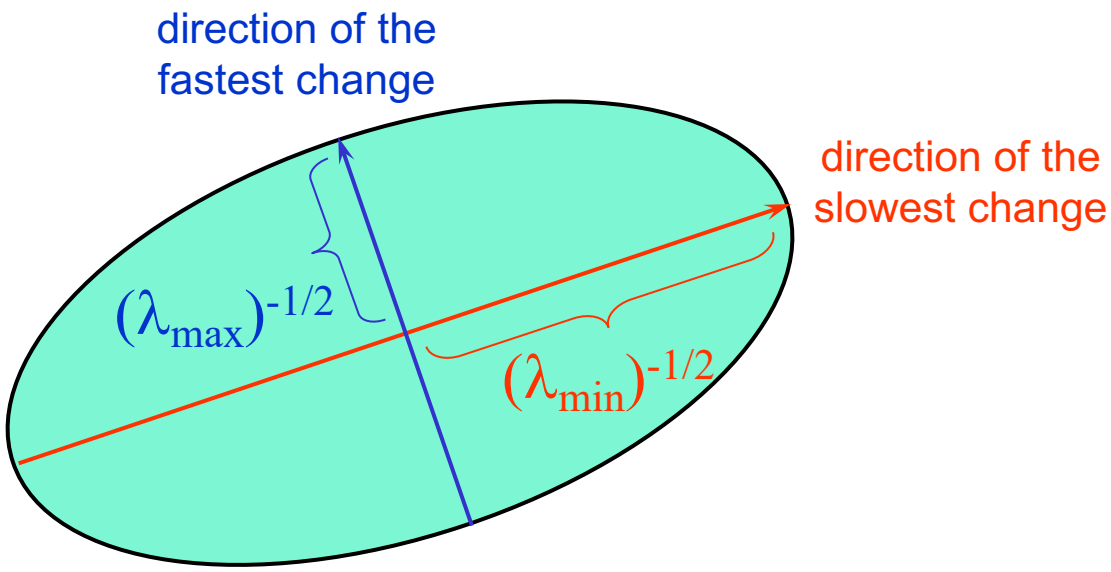
Source: L. Lazebnik

Interpreting the second moment matrix

In the general case, need to *diagonalize* M:

$$M = R^{-1}\begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} R$$

The axis lengths of the ellipse are determined by the eigenvalues and the orientation is determined by *R*:

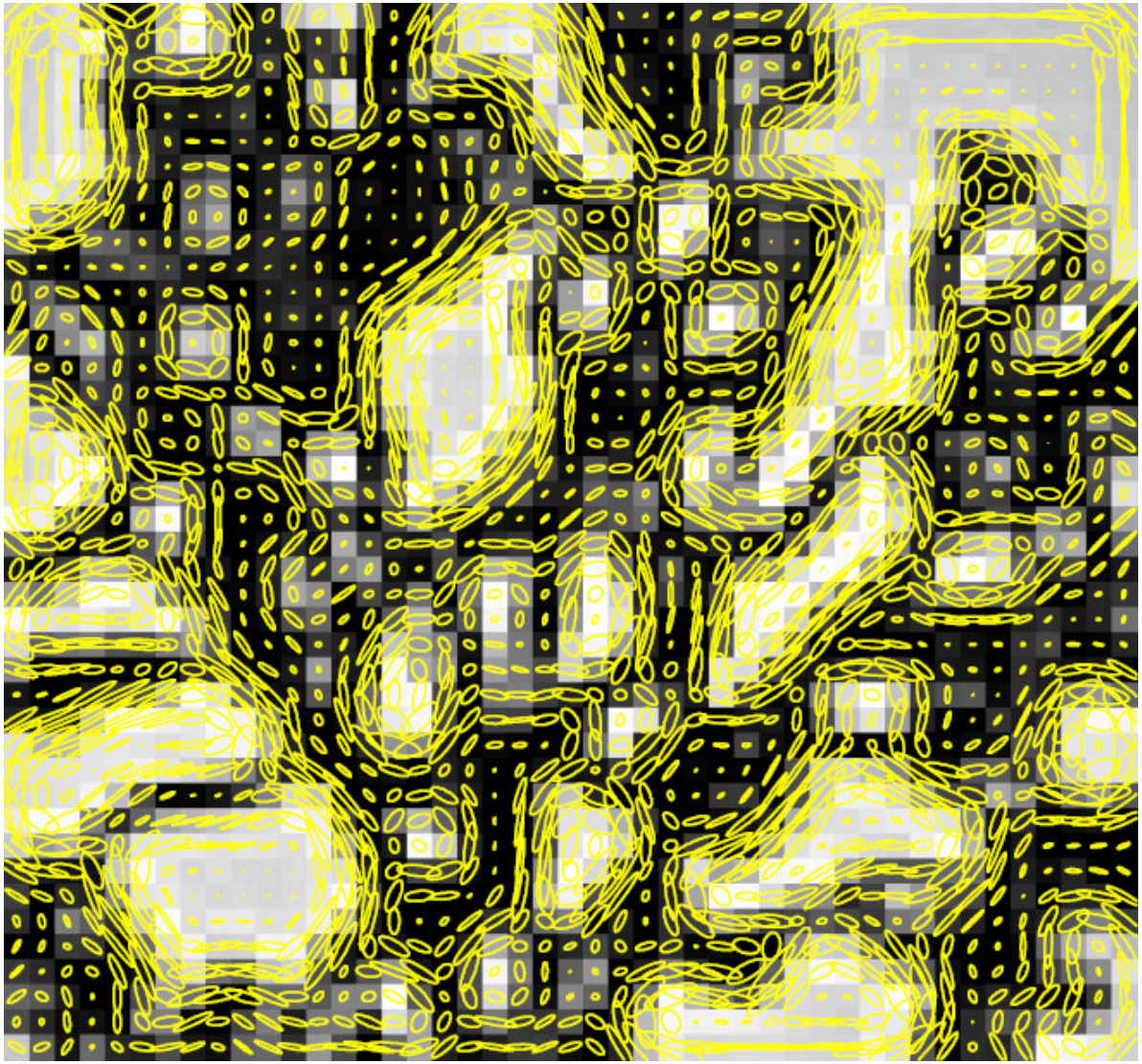


Source: L. Lazebnik

Visualization of second moment matrices

Source: L. Lazebnik

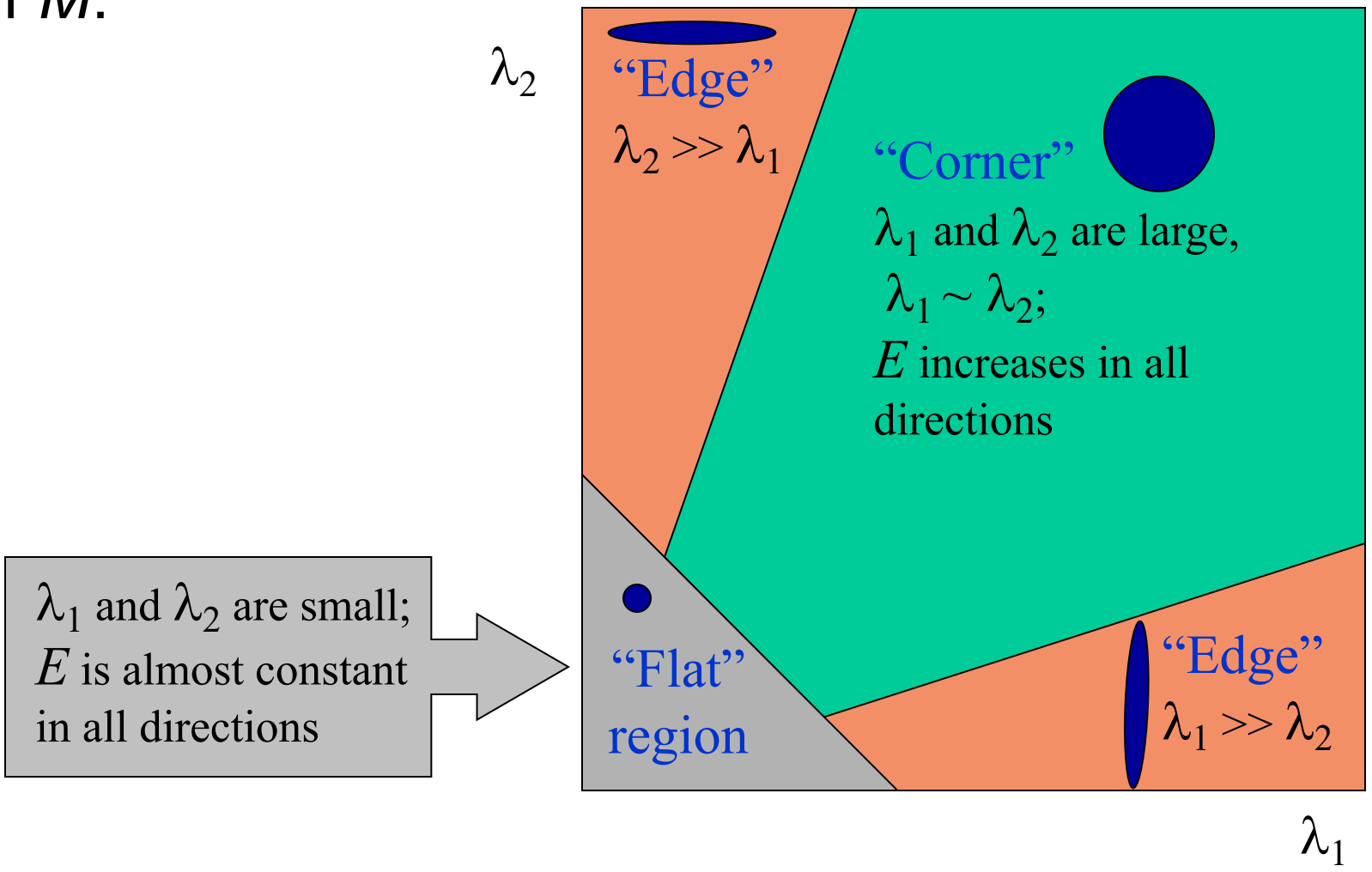
Visualization of second moment matrices



Source: L. Lazebnik

Interpreting the eigenvalues

Classification of image points using eigenvalues of *M*:

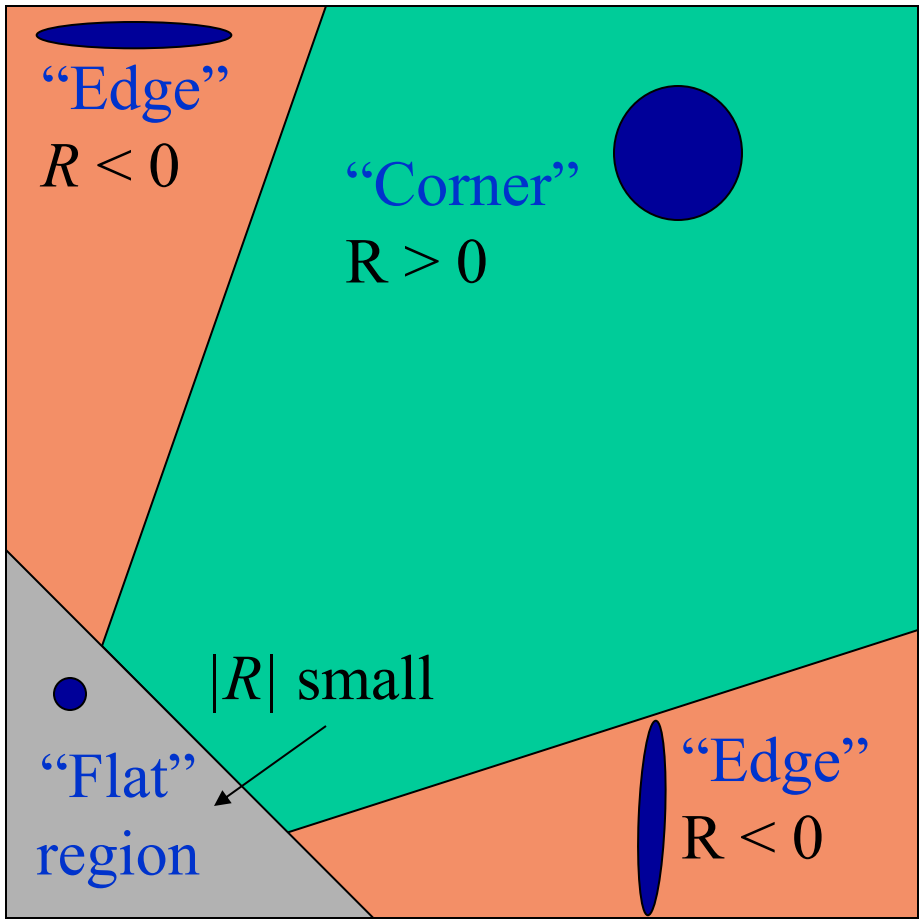


Source: L. Lazebnik

Corner response function

$$R = \det(M) - \alpha \,\mathrm{trace}(M)^2 = \lambda_1\lambda_2 - \alpha(\lambda_1 + \lambda_2)^2$$

α: constant (0.04 to 0.06)



Source: L. Lazebnik

The Harris corner detector

1. Compute partial derivatives at each pixel
2. Compute second moment matrix *M* in a Gaussian window around each pixel:

$$M = \begin{bmatrix} \sum_{x,y} w(x,y) I_x^2 & \sum_{x,y} w(x,y) I_x I_y \\ \sum_{x,y} w(x,y) I_x I_y & \sum_{x,y} w(x,y) I_y^2 \end{bmatrix}$$

C.Harris and M.Stephens, A Combined Corner and Edge Detector, *Proceedings of the 4th Alvey Vision Conference*: pages 147—151, 1988.

Source: L. Lazebnik

The Harris corner detector

1. Compute partial derivatives at each pixel
2. Compute second moment matrix *M* in a Gaussian window around each pixel
3. Compute corner response function *R*

C.Harris and M.Stephens, A Combined Corner and Edge Detector, *Proceedings of the 4th Alvey Vision Conference*: pages 147—151, 1988.

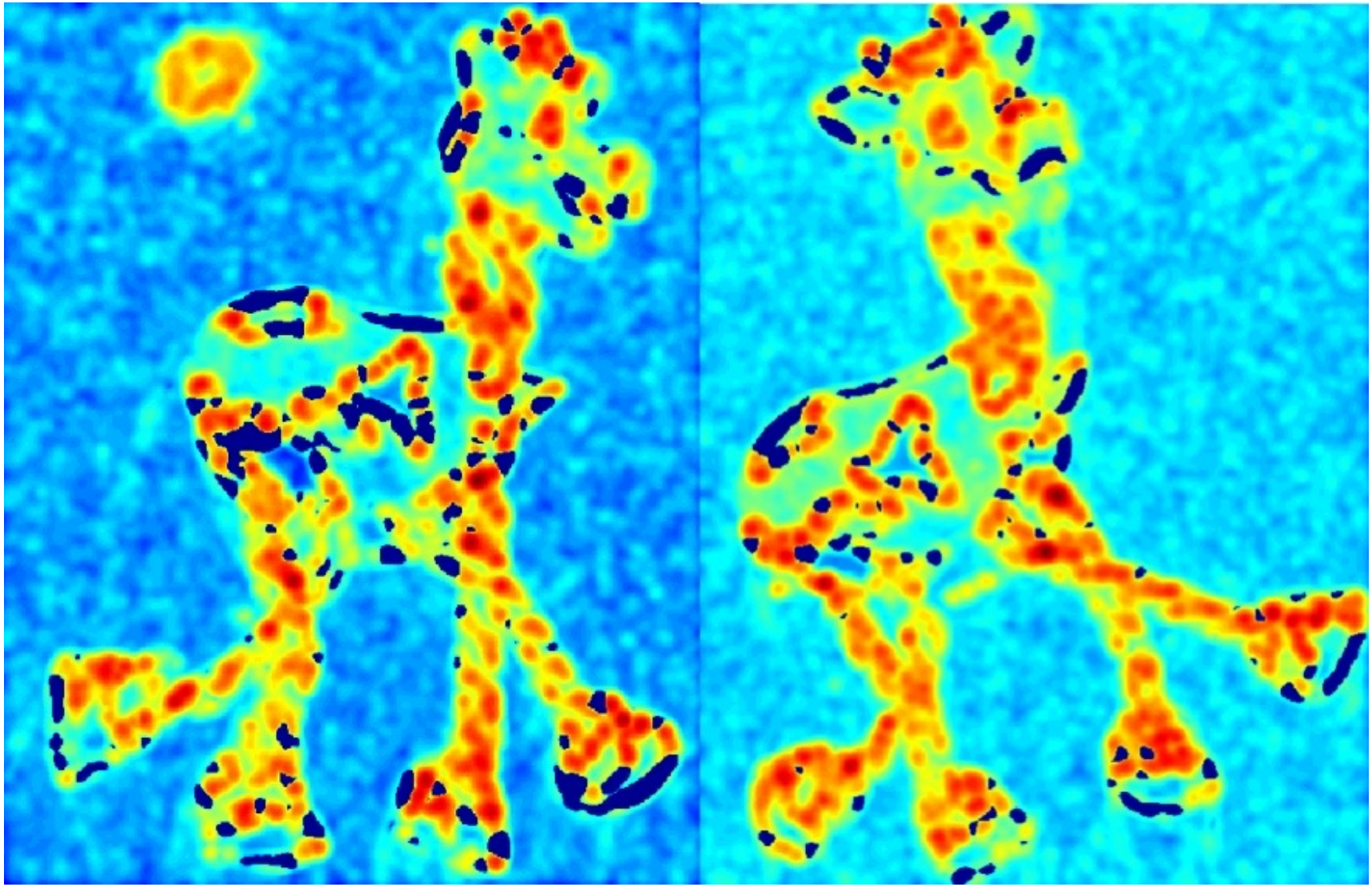
Source: L. Lazebnik

Harris Detector: Steps

Source: L. Lazebnik

Harris Detector: Steps

Compute corner response R



Source: L. Lazebnik

The Harris corner detector

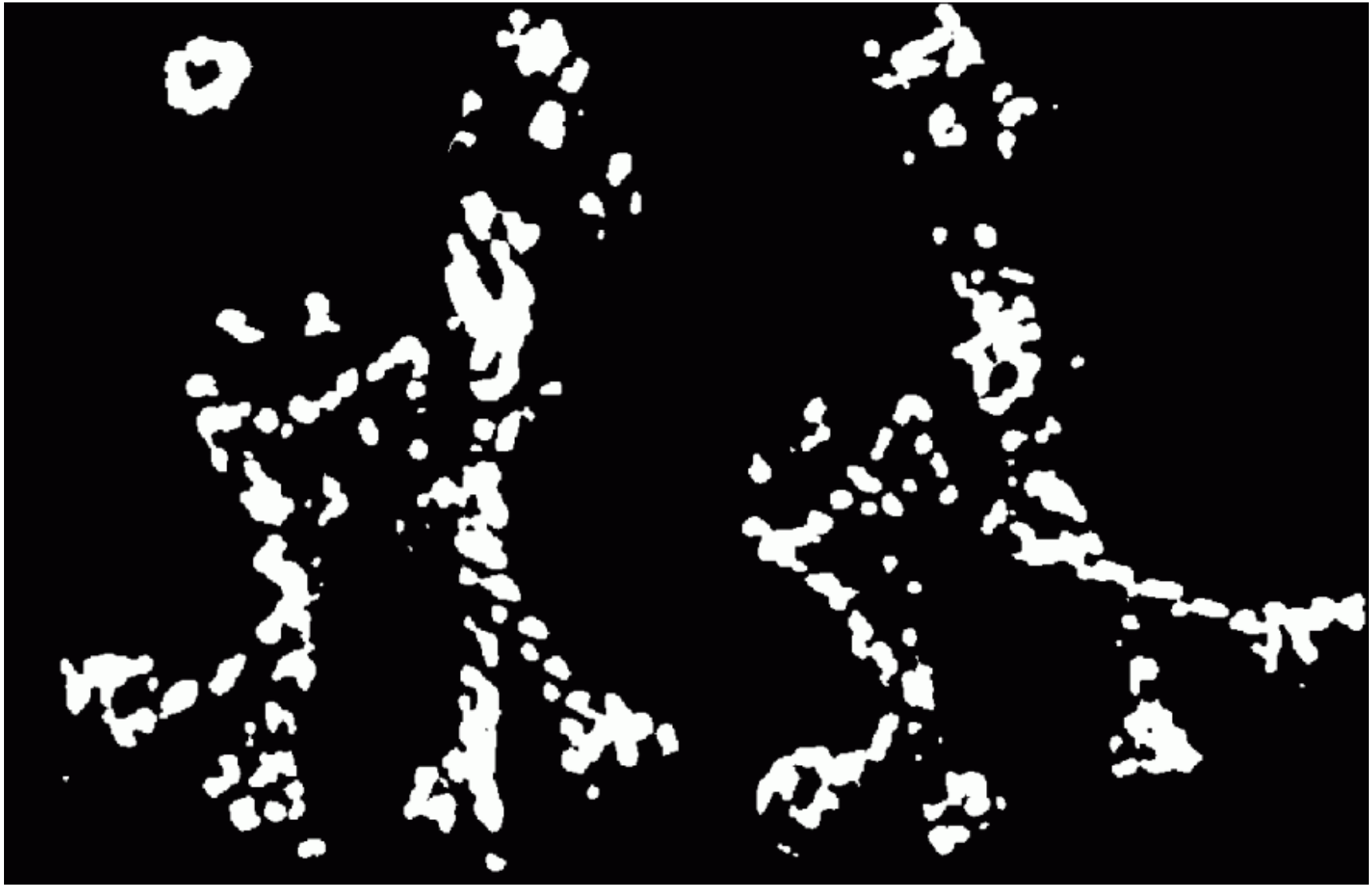
1. Compute partial derivatives at each pixel
2. Compute second moment matrix *M* in a Gaussian window around each pixel
3. Compute corner response function *R*
4. Threshold *R*
5. Find local maxima of response function (nonmaximum suppression)

C.Harris and M.Stephens, A Combined Corner and Edge Detector, *Proceedings of the 4th Alvey Vision Conference*: pages 147—151, 1988.

Source: L. Lazebnik

Harris Detector: Steps

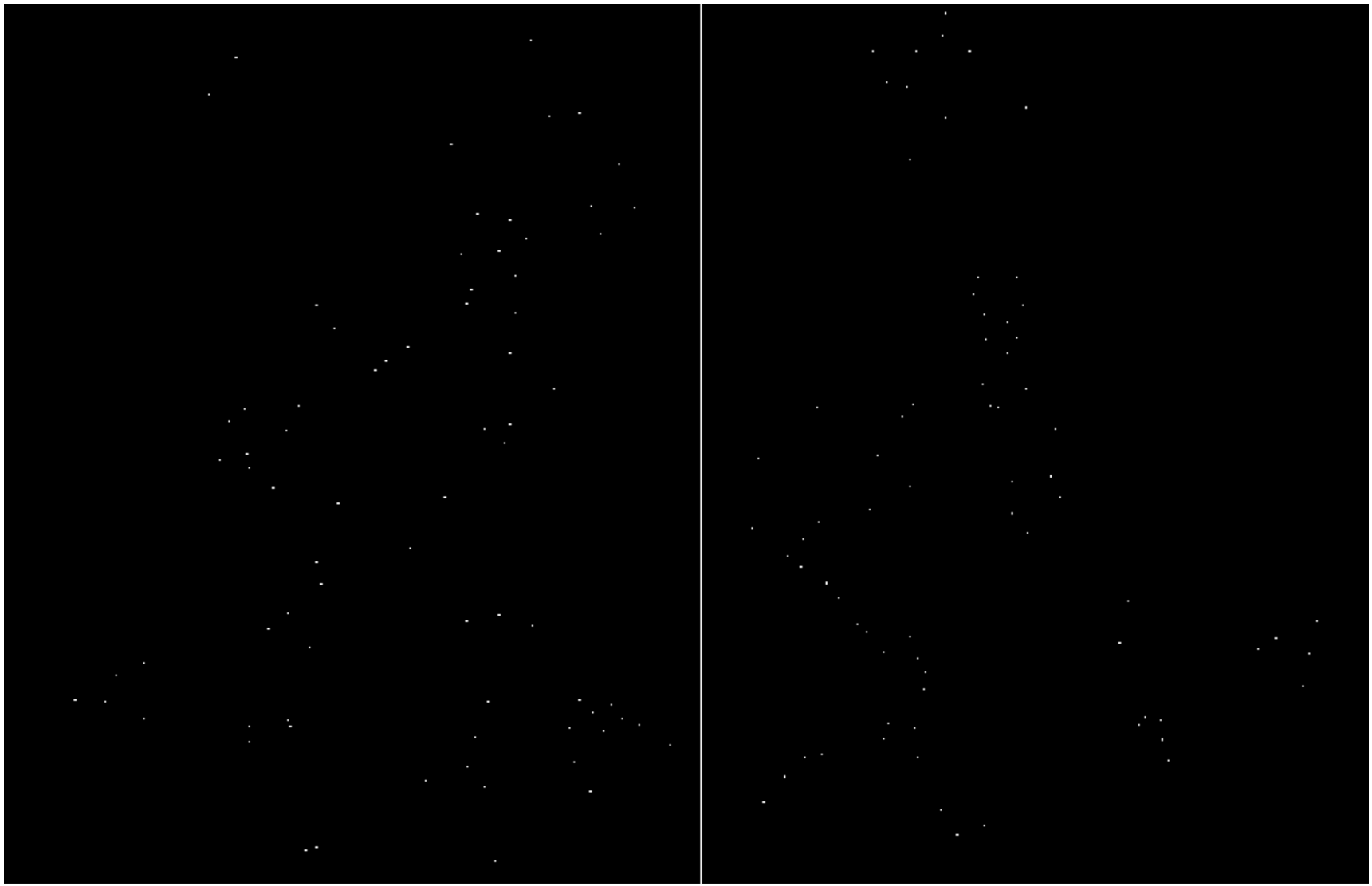
Find points with large corner response: R > threshold



Source: L. Lazebnik

Harris Detector: Steps

Take only the points of local maxima of R



Source: L. Lazebnik

Harris Detector: Steps

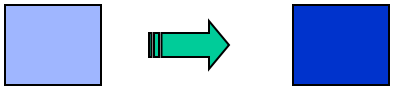
Source: L. Lazebnik

Robustness of corner features

- What happens to corner features when the image undergoes geometric or photometric transformations?

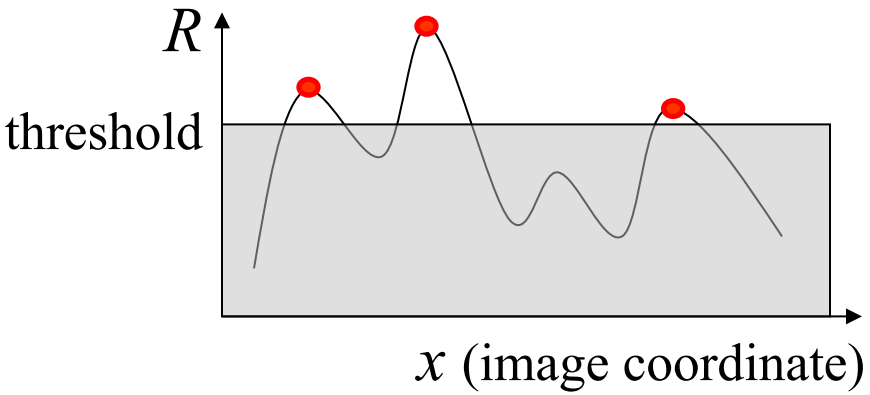
Source: L. Lazebnik

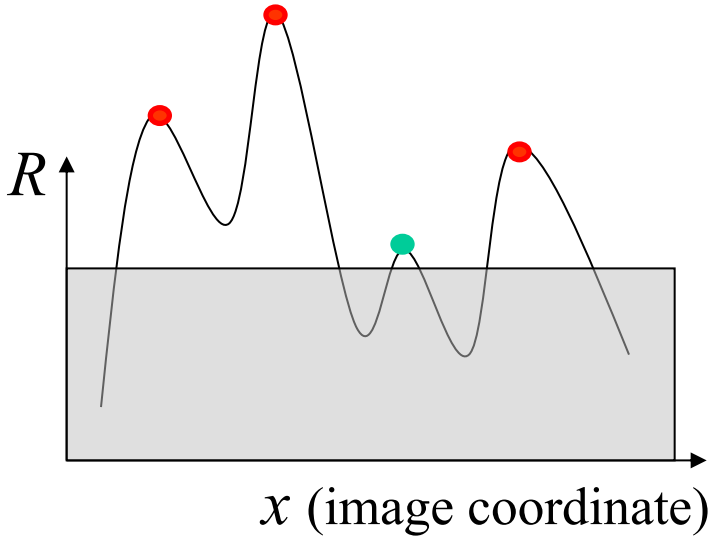
Affine intensity change



$$I \rightarrow a\,I + b$$

- Only derivatives are used, so invariant to intensity shift $I \rightarrow I + b$
- Intensity scaling: $I \rightarrow a\,I$

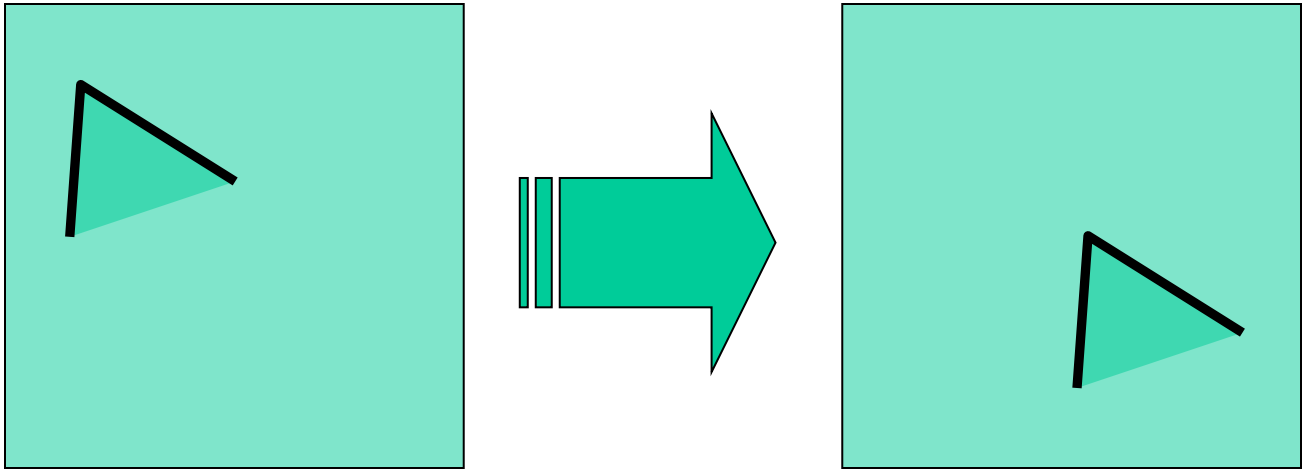




Partially invariant to affine intensity change

Source: L. Lazebnik

Image translation

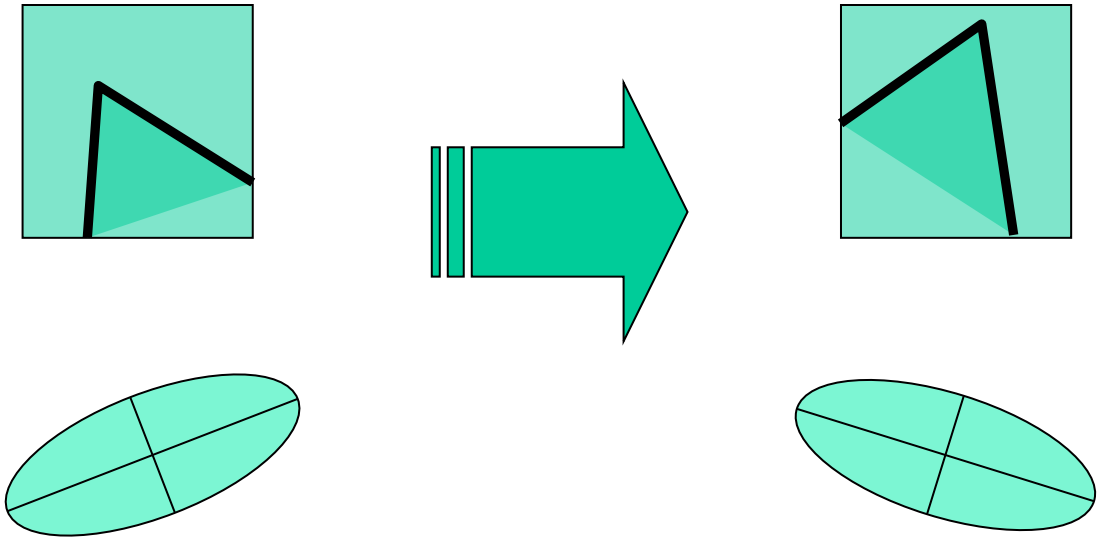


- Derivatives and window function are shift-invariant

Corner location is *covariant* w.r.t. translation

Source: L. Lazebnik

Image rotation

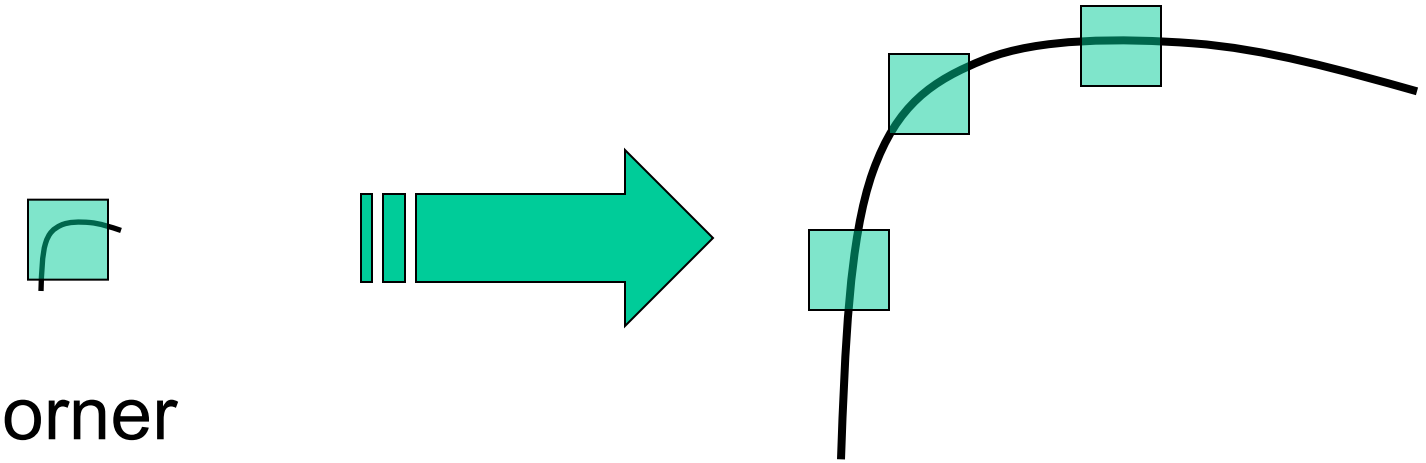


Second moment ellipse rotates but its shape (i.e. eigenvalues) remains the same

Corner location is covariant w.r.t. rotation

Source: L. Lazebnik

Scaling



Corner

All points will be classified as edges

Corner location is not covariant w.r.t. scaling!

Source: L. Lazebnik

Optical flow

Many slides adapted from S. Seitz, R. Szeliski, M. Pollefeys

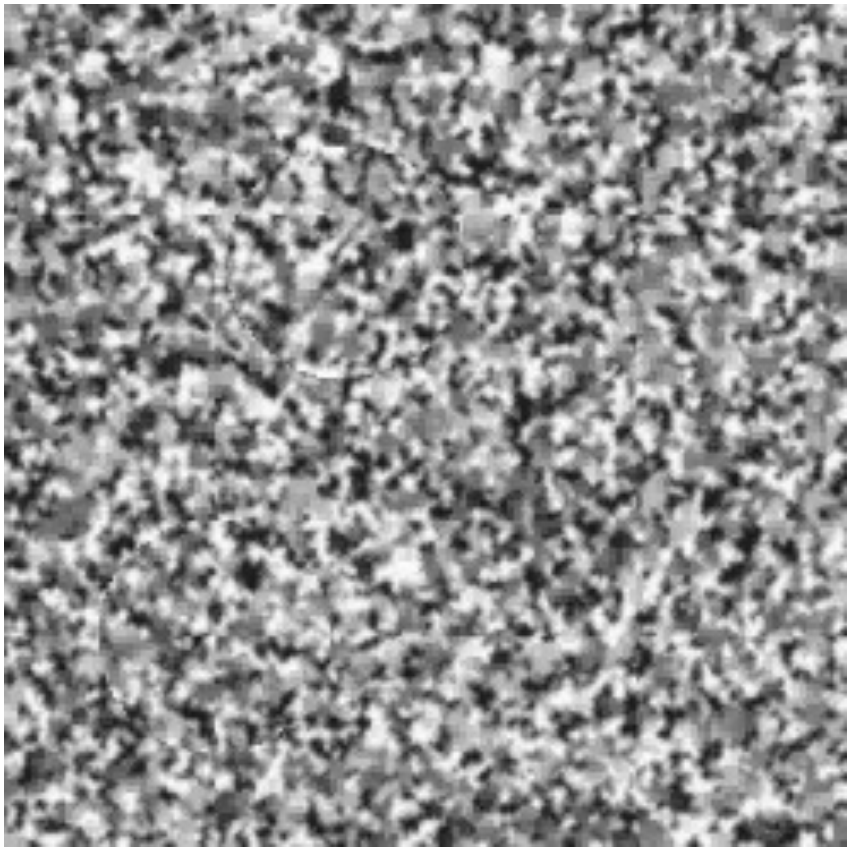
Slides from S. Lazebnik.

What direction is the object moving?

A. Left right
B. Up down
C. Top-left to bottom-right
D. Bottom left to top-right

Motion is a powerful perceptual cue

- Sometimes, it is the only cue



Motion is a powerful perceptual cue

- Even “impoverished” motion data can evoke a strong percept

G. Johansson, “Visual Perception of Biological Motion and a Model For Its Analysis", *Perception and Psychophysics 14, 201-211, 1973.*

Source: L. Lazebnik

Motion is a powerful perceptual cue

- Even “impoverished” motion data can evoke a strong percept

G. Johansson, “Visual Perception of Biological Motion and a Model For Its Analysis", *Perception and Psychophysics 14, 201-211, 1973.*

Source: L. Lazebnik

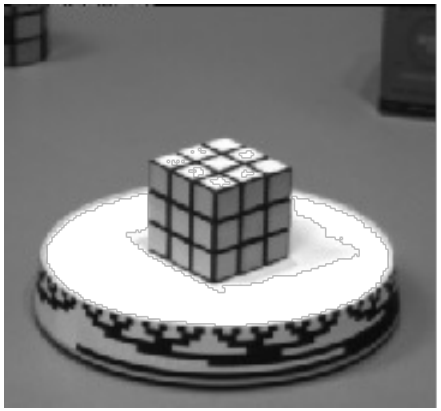
Uses of motion in computer vision

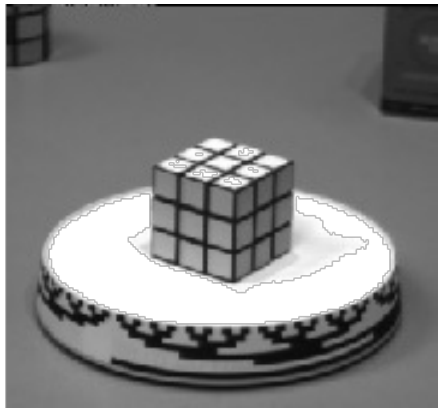
- 3D shape reconstruction
- Object segmentation
- Learning and tracking of dynamical models
- Event and activity recognition
- Self-supervised and predictive learning

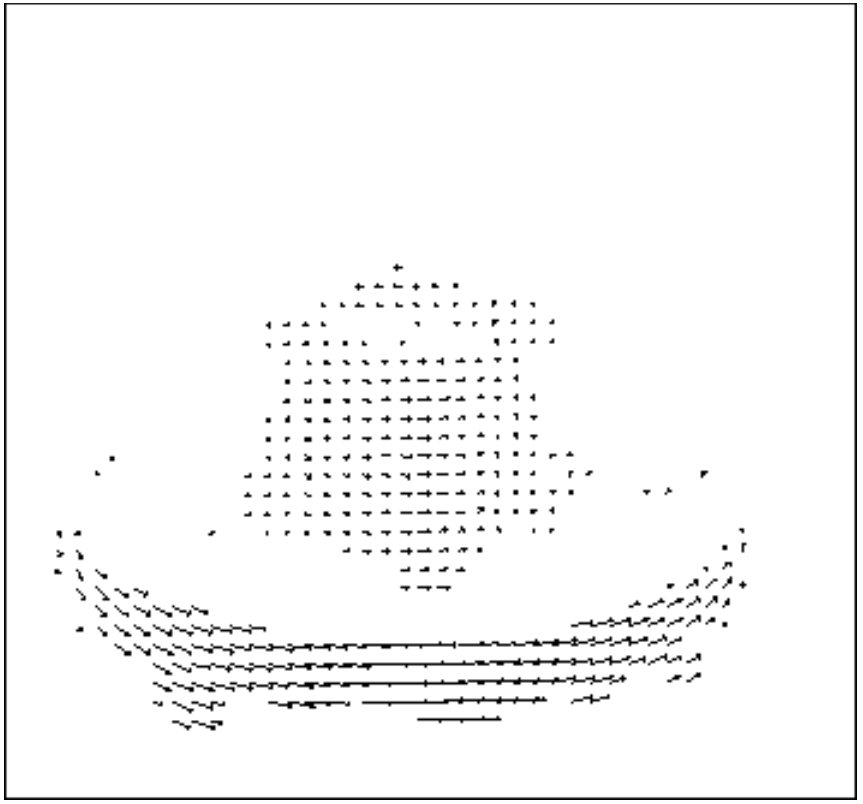
Source: L. Lazebnik

Motion field

- The motion field is the projection of the 3D scene motion into the image







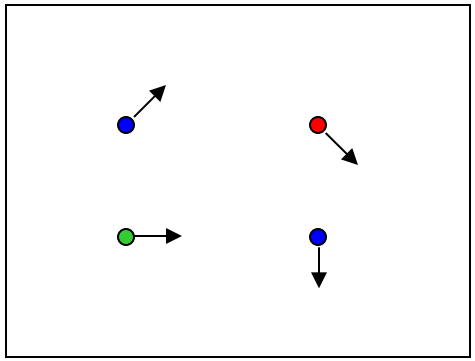
Source: L. Lazebnik

Optical flow

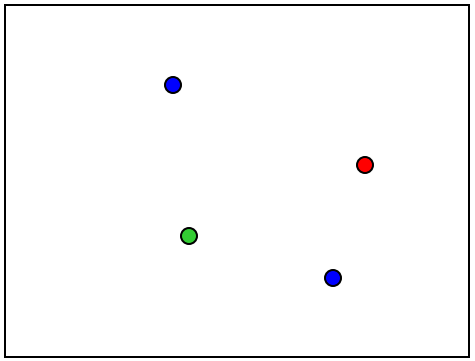
- **Definition**: optical flow is the *apparent* motion of brightness patterns in the image
- Ideally, optical flow would be the same as the motion field
- Have to be careful: apparent motion can be caused by lighting changes without any actual motion
  - Think of a uniform rotating sphere under fixed lighting vs. a stationary sphere under moving illumination

Source: L. Lazebnik

Estimating optical flow



$I(x,y,t-1)$

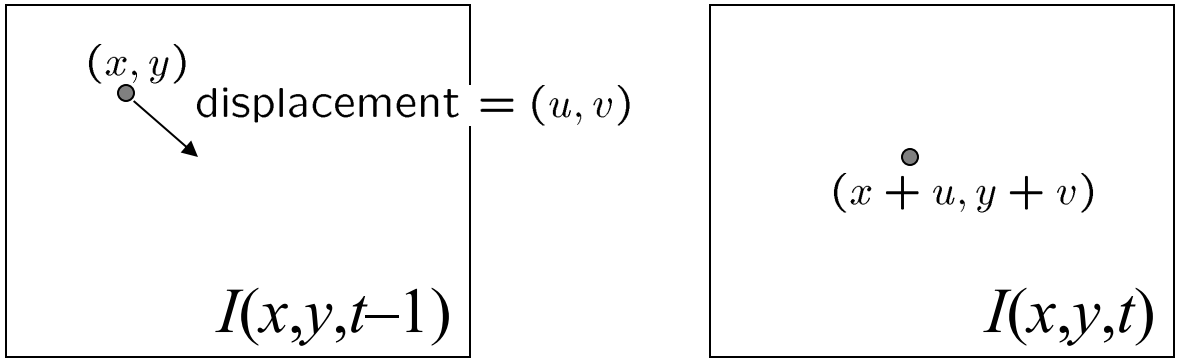


$I(x,y,t)$

- Given two subsequent frames, estimate the apparent motion field $u(x,y)$ and $v(x,y)$ between them
- Key assumptions
  - **Brightness constancy:** projection of the same point looks the same in every frame
  - **Small motion:** points do not move very far
  - **Spatial coherence:** points move like their neighbors

Source: L. Lazebnik

The brightness constancy constraint



Brightness Constancy Equation:

$$I(x, y, t-1) = I(x + u(x, y), y + v(x, y), t)$$

Linearizing the right side using Taylor expansion:

$$I(x, y, t-1) \approx I(x, y, t) + I_x\, u(x, y) + I_y\, v(x, y)$$

Hence, $I_x\, u + I_y\, v + I_t \approx 0$

Source: L. Lazebnik

The brightness constancy constraint

$$I_x u + I_y v + I_t = 0$$

- How many equations and unknowns per pixel?
  - One equation, two unknowns
- What does this constraint mean?

$$\nabla I \cdot (u, v) + I_t = 0$$

- The component of the flow perpendicular to the gradient (i.e., parallel to the edge) is unknown!

Source: L. Lazebnik

The brightness constancy constraint

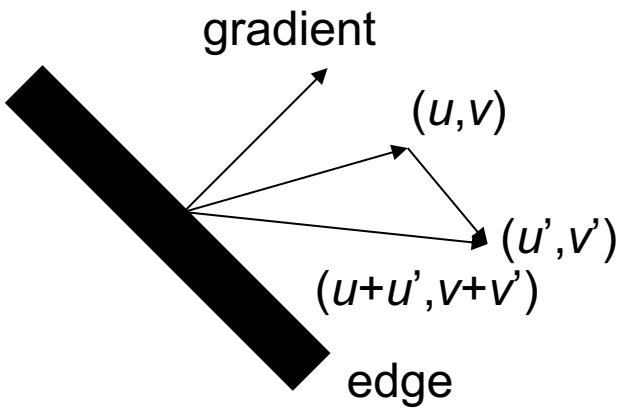
$$I_x u + I_y v + I_t = 0$$

- How many equations and unknowns per pixel?
  - One equation, two unknowns
- What does this constraint mean?

$$\nabla I \cdot (u, v) + I_t = 0$$

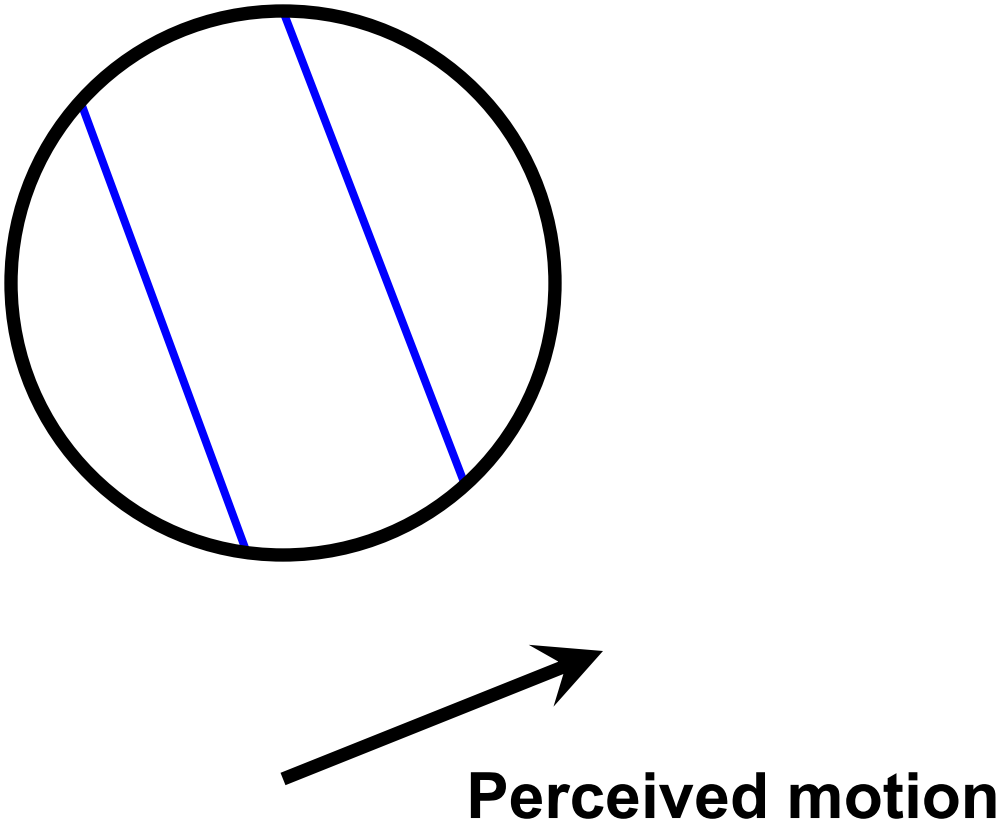
- The component of the flow perpendicular to the gradient (i.e., parallel to the edge) is unknown!

If (u, v) satisfies the equation,
so does (u+u', v+v') if $\nabla I \cdot (u', v') = 0$



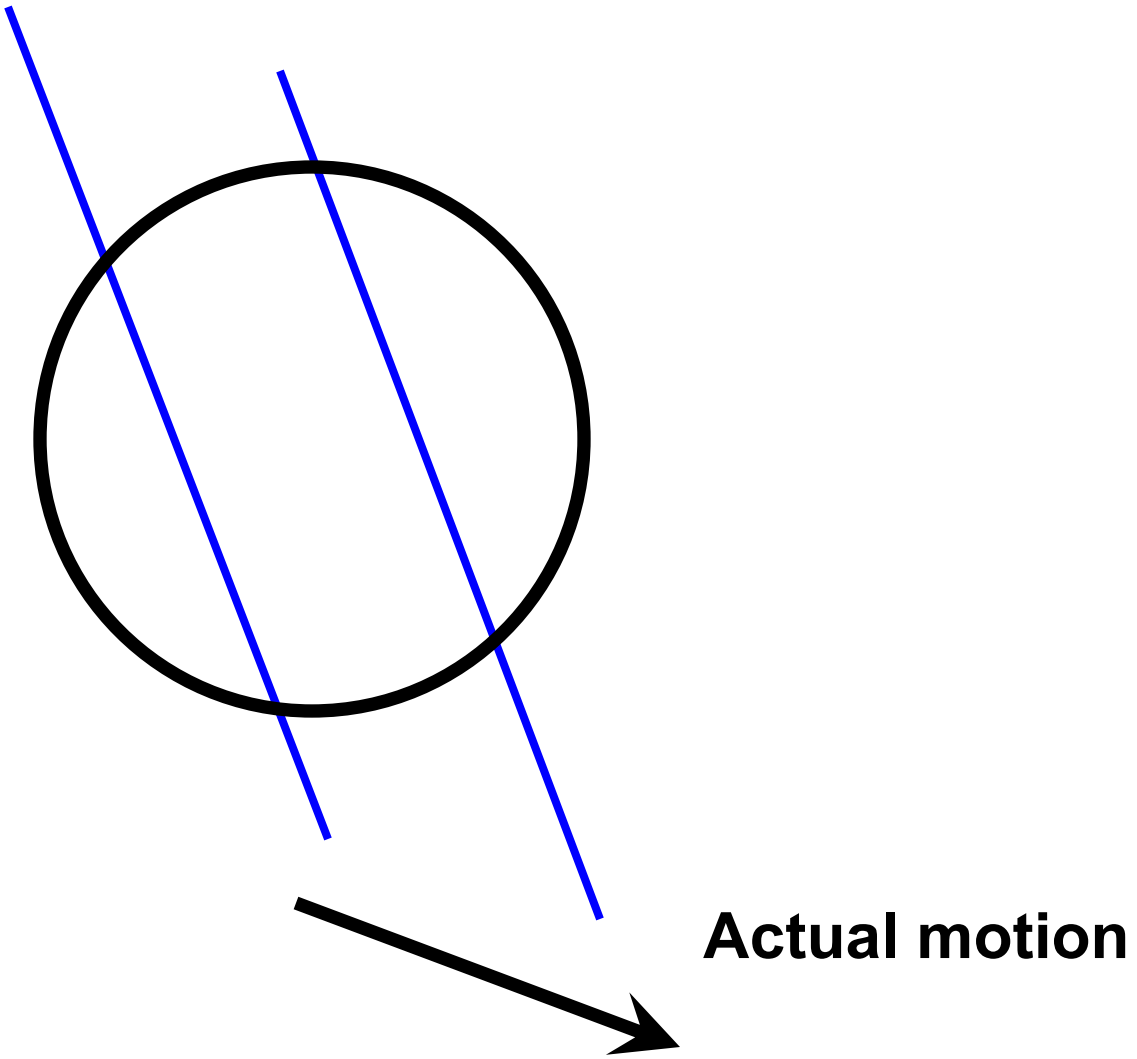
Source: L. Lazebnik

The aperture problem



Source: L. Lazebnik

The aperture problem



Source: L. Lazebnik

The barber pole illusion

http://en.wikipedia.org/wiki/Barberpole_illusion

Source: L. Lazebnik

The barber pole illusion

http://en.wikipedia.org/wiki/Barberpole_illusion

Source: L. Lazebnik

Solving the aperture problem

- How to get more equations for a pixel?
- **Spatial coherence constraint:** assume the pixel's neighbors have the same (u,v)
  - E.g., if we use a 5x5 window, that gives us 25 equations per pixel

$$\nabla I(\mathbf{x}_i)\cdot[u,v]+I_t(\mathbf{x}_i)=0$$

$$\begin{bmatrix} I_x(\mathbf{x}_1) & I_y(\mathbf{x}_1) \\ I_x(\mathbf{x}_2) & I_y(\mathbf{x}_2) \\ \vdots & \vdots \\ I_x(\mathbf{x}_n) & I_y(\mathbf{x}_n) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = -\begin{bmatrix} I_t(\mathbf{x}_1) \\ I_t(\mathbf{x}_2) \\ \vdots \\ I_t(\mathbf{x}_n) \end{bmatrix}$$

B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 674–679, 1981.

Source: L. Lazebnik

Lucas-Kanade flow

- Linear least squares problem:

$$\begin{bmatrix} I_x(\mathbf{x}_1) & I_y(\mathbf{x}_1) \\ I_x(\mathbf{x}_2) & I_y(\mathbf{x}_2) \\ \vdots & \vdots \\ I_x(\mathbf{x}_n) & I_y(\mathbf{x}_n) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = -\begin{bmatrix} I_t(\mathbf{x}_1) \\ I_t(\mathbf{x}_2) \\ \vdots \\ I_t(\mathbf{x}_n) \end{bmatrix}$$

- When is this system solvable?

B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 674–679, 1981.

Source: L. Lazebnik

Lucas-Kanade flow

- Linear least squares problem:

$$\begin{bmatrix} I_x(\mathbf{x}_1) & I_y(\mathbf{x}_1) \\ I_x(\mathbf{x}_2) & I_y(\mathbf{x}_2) \\ \vdots & \vdots \\ I_x(\mathbf{x}_n) & I_y(\mathbf{x}_n) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = -\begin{bmatrix} I_t(\mathbf{x}_1) \\ I_t(\mathbf{x}_2) \\ \vdots \\ I_t(\mathbf{x}_n) \end{bmatrix}$$

$$\underset{n\times 2}{\mathbf{A}}\ \underset{2\times 1}{\mathbf{d}} = \underset{n\times 1}{\mathbf{b}}$$

- Solution given by $(\mathbf{A}^T\mathbf{A})\mathbf{d} = \mathbf{A}^T\mathbf{b}$

$$\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = -\begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$$

(summations are over all pixels in the window)

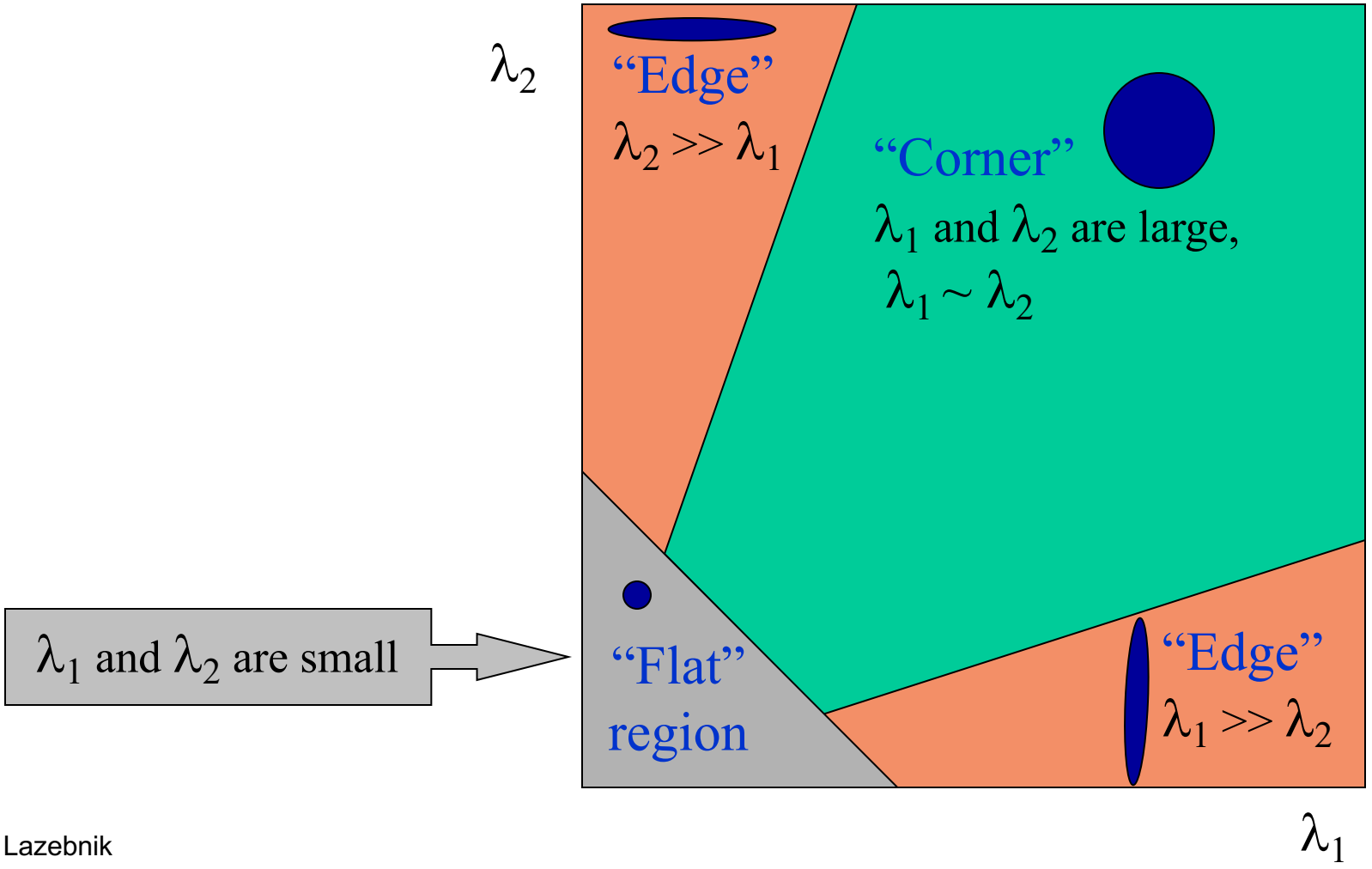
M = $\mathbf{A}^T\mathbf{A}$ is the second moment matrix!

B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 674–679, 1981.

Source: L. Lazebnik

Recall: second moment matrix

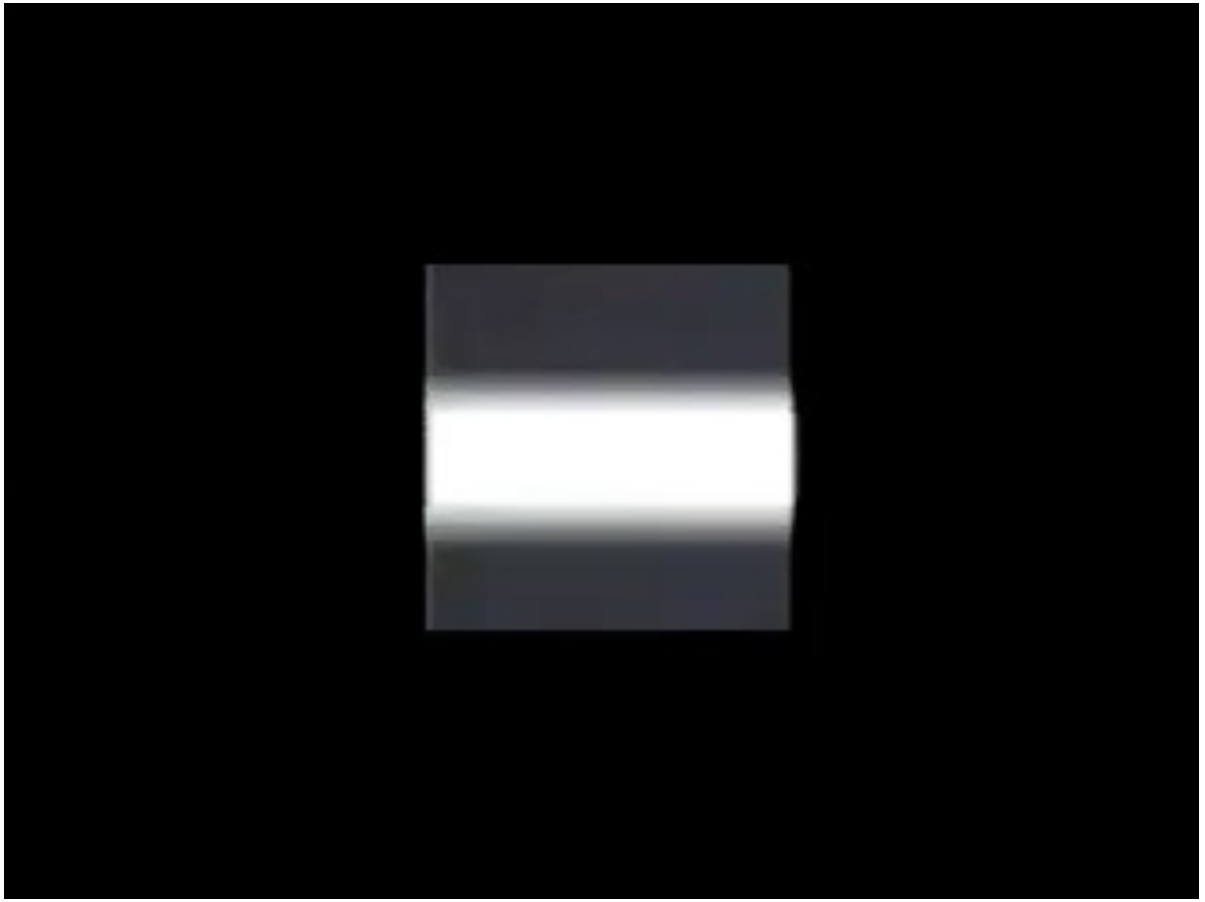
- Estimation of optical flow is well-conditioned precisely for regions with high “cornerness”:



Source: L. Lazebnik

Conditions for solvability

- “Bad” case: single straight edge



Source: L. Lazebnik

Conditions for solvability

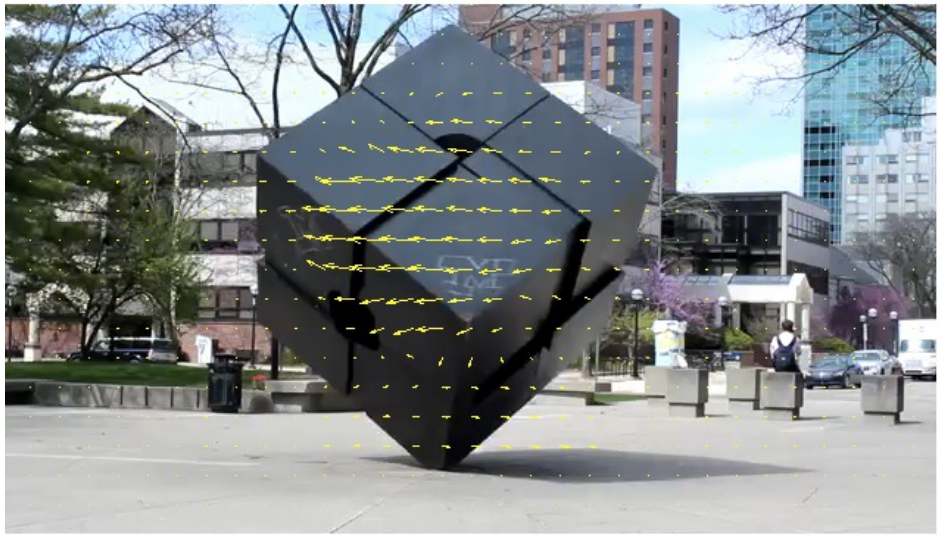
- “Good” case

Source: L. Lazebnik

Lucas-Kanade flow example

Input frames

Output



Source: L. Lazebnik

Source: MATLAB Central File Exchange

Errors in Lucas-Kanade

- The motion is large (larger than a pixel)
- A point does not move like its neighbors
- Brightness constancy does not hold

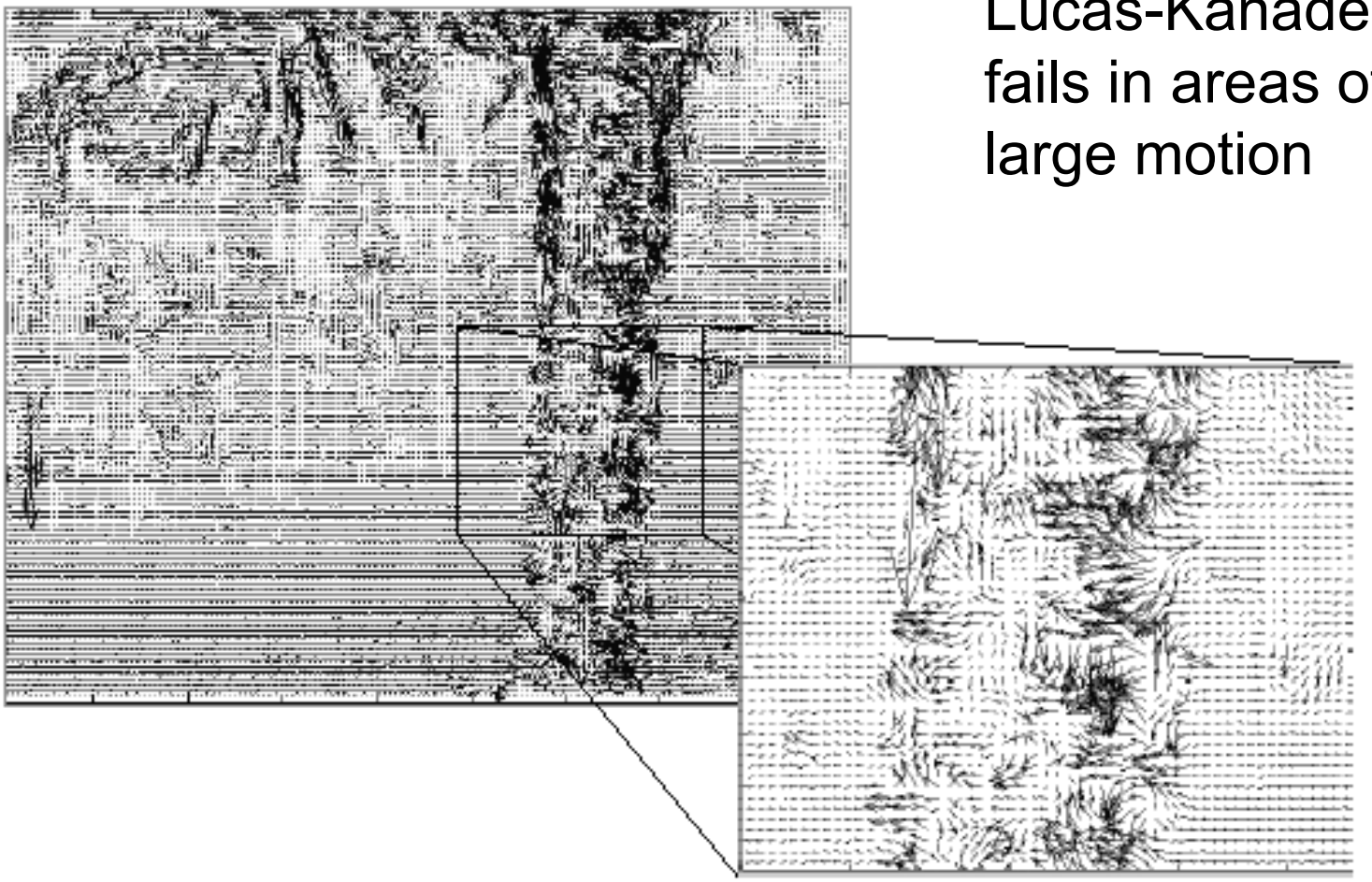
Source: L. Lazebnik

“Flower garden” example

Source: L. Lazebnik

*** From Khurram Hassan-Shafique CAP5415 Computer Vision 2003**

“Flower garden” example

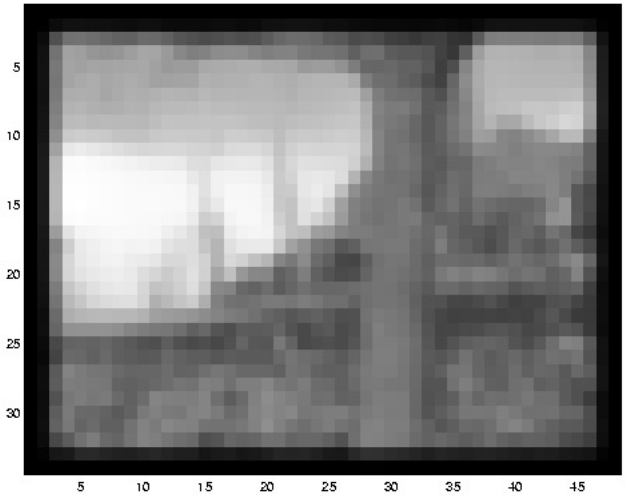


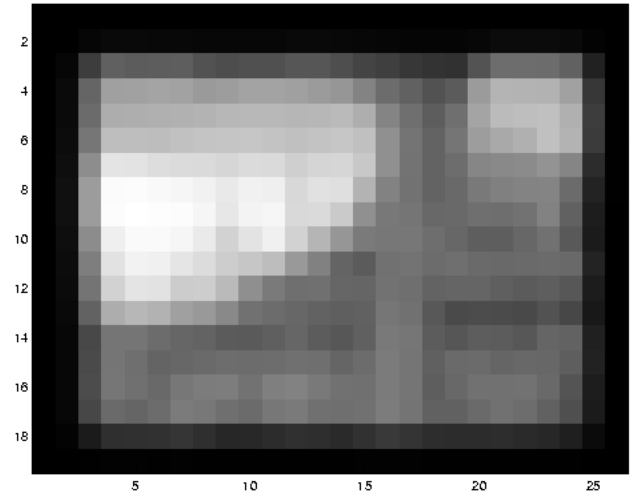
Lucas-Kanade fails in areas of large motion

Source: L. Lazebnik

*** From Khurram Hassan-Shafique CAP5415 Computer Vision 2003**

Multi-resolution estimation

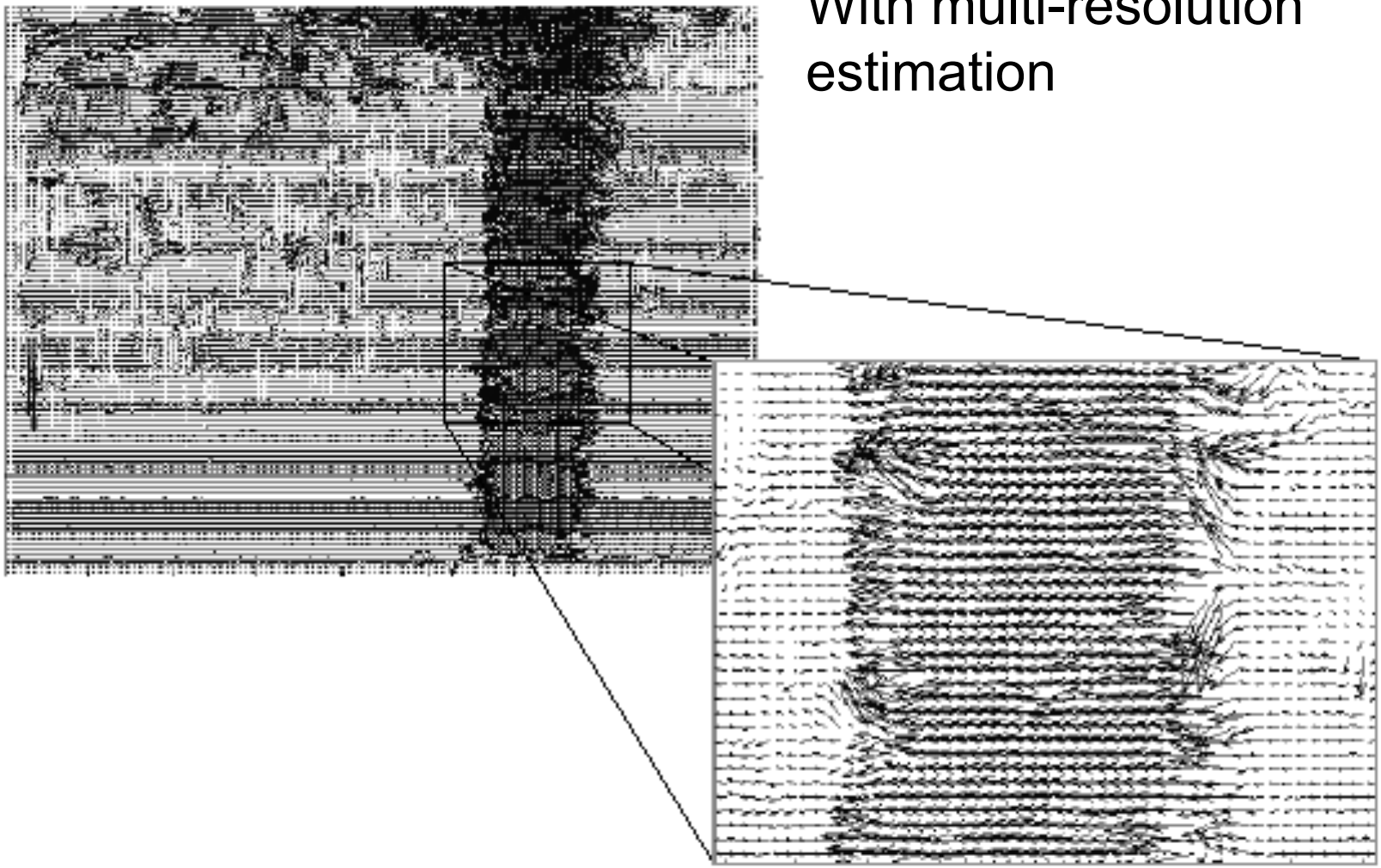




Source: L. Lazebnik

*** From Khurram Hassan-Shafique CAP5415 Computer Vision 2003**

Multi-resolution estimation



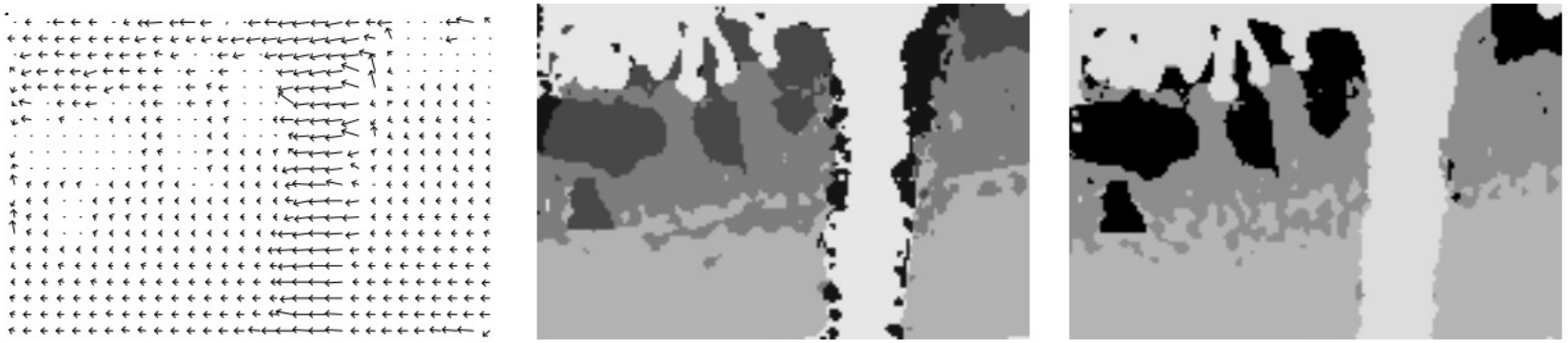
With multi-resolution estimation

Source: L. Lazebnik

*** From Khurram Hassan-Shafique CAP5415 Computer Vision 2003**

Fixing the errors in Lucas-Kanade

- The motion is large (larger than a pixel)
  - Multi-resolution estimation, iterative refinement
  - Feature matching
- A point does not move like its neighbors
  - Motion segmentation



(a) (b) (c)

Figure 11: (a) The optic flow from multi-scale gradient method. (b) Segmentation obtained by clustering optic flow into affine motion regions. (c) Segmentation from consistency checking by image warping. Representing moving images with layers.

J. Wang and E. Adelson, Representing Moving Images with Layers, IEEE Transactions on Image Processing, 1994

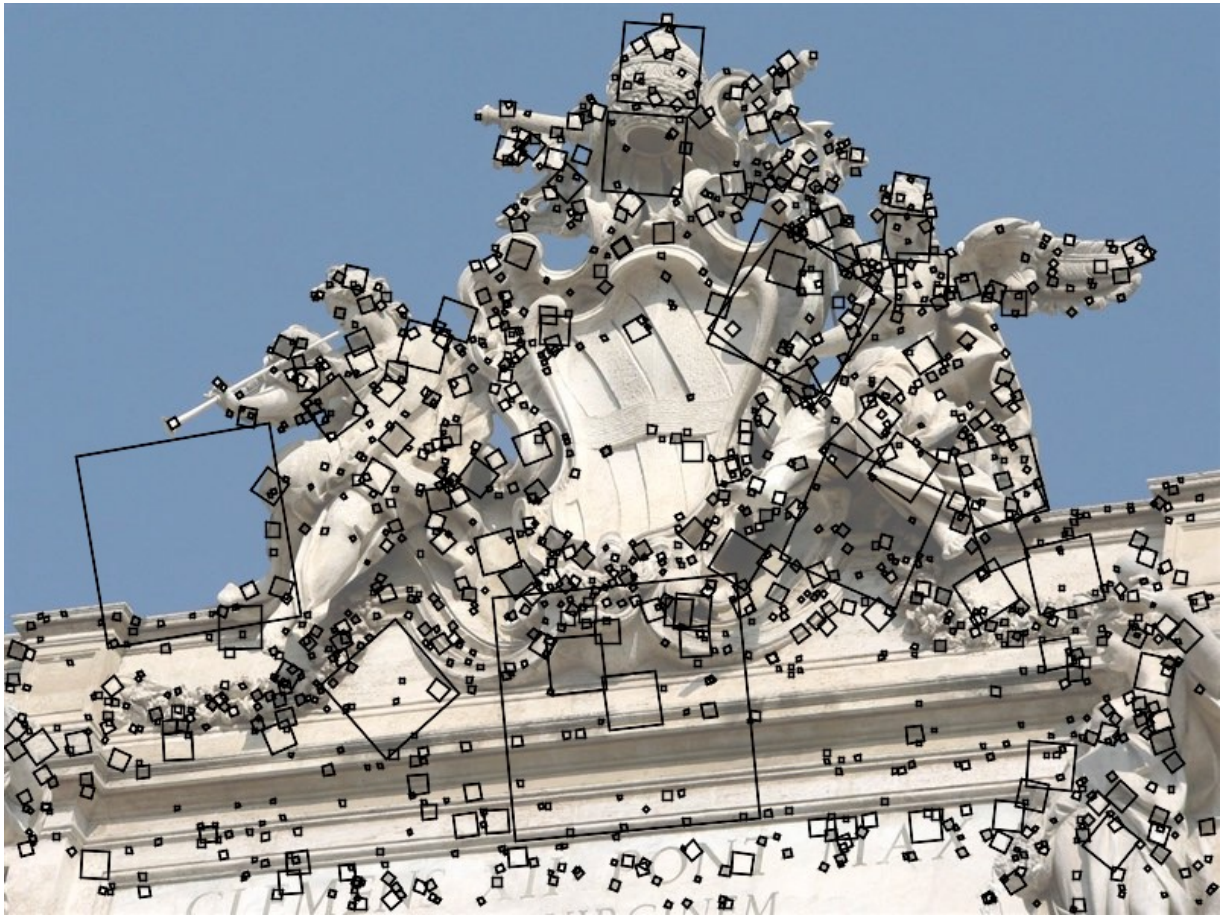
Source: L. Lazebnik

Fixing the errors in Lucas-Kanade

- The motion is large (larger than a pixel)
  - Multi-resolution estimation, iterative refinement
  - Feature matching
- A point does not move like its neighbors
  - Motion segmentation
- Brightness constancy does not hold
  - Feature matching

Source: L. Lazebnik

SIFT keypoint detection

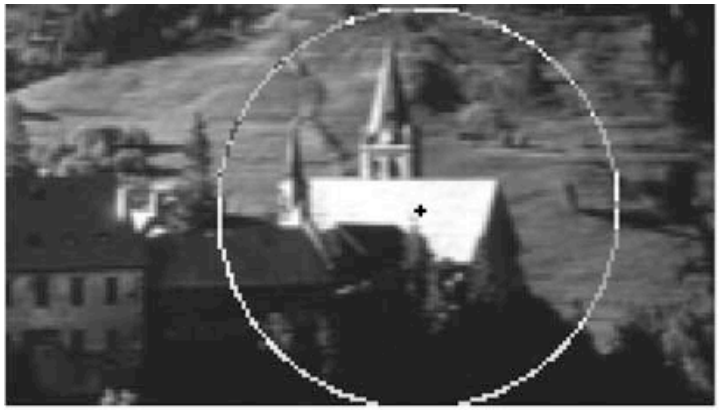


D. Lowe, Distinctive image features from scale-invariant keypoints, *IJCV* 60 (2), pp. 91-110, 2004

Slides from S. Lazebnik.

Keypoint detection with scale selection

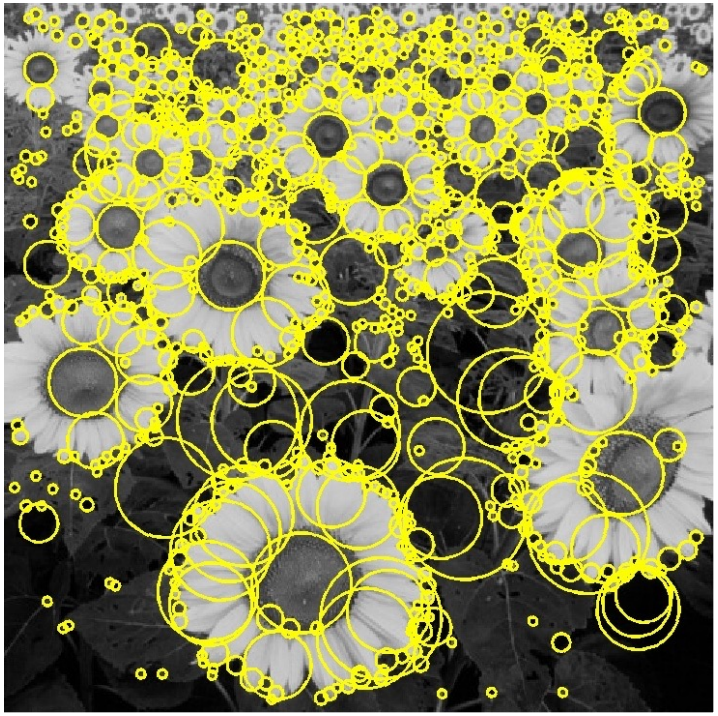
- We want to extract keypoints with *characteristic scales* that are *covariant* w.r.t. the image transformation

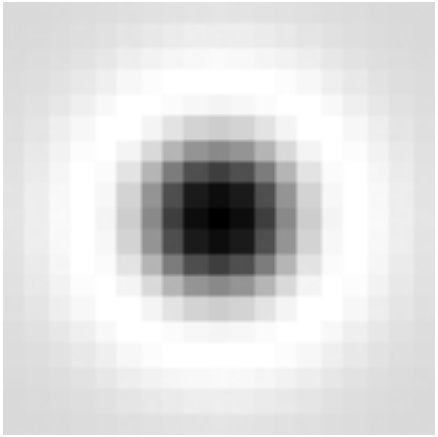


Source: L. Lazebnik

Basic idea

- Convolve the image with a “blob filter” at multiple scales and look for extrema of filter response in the resulting *scale space*

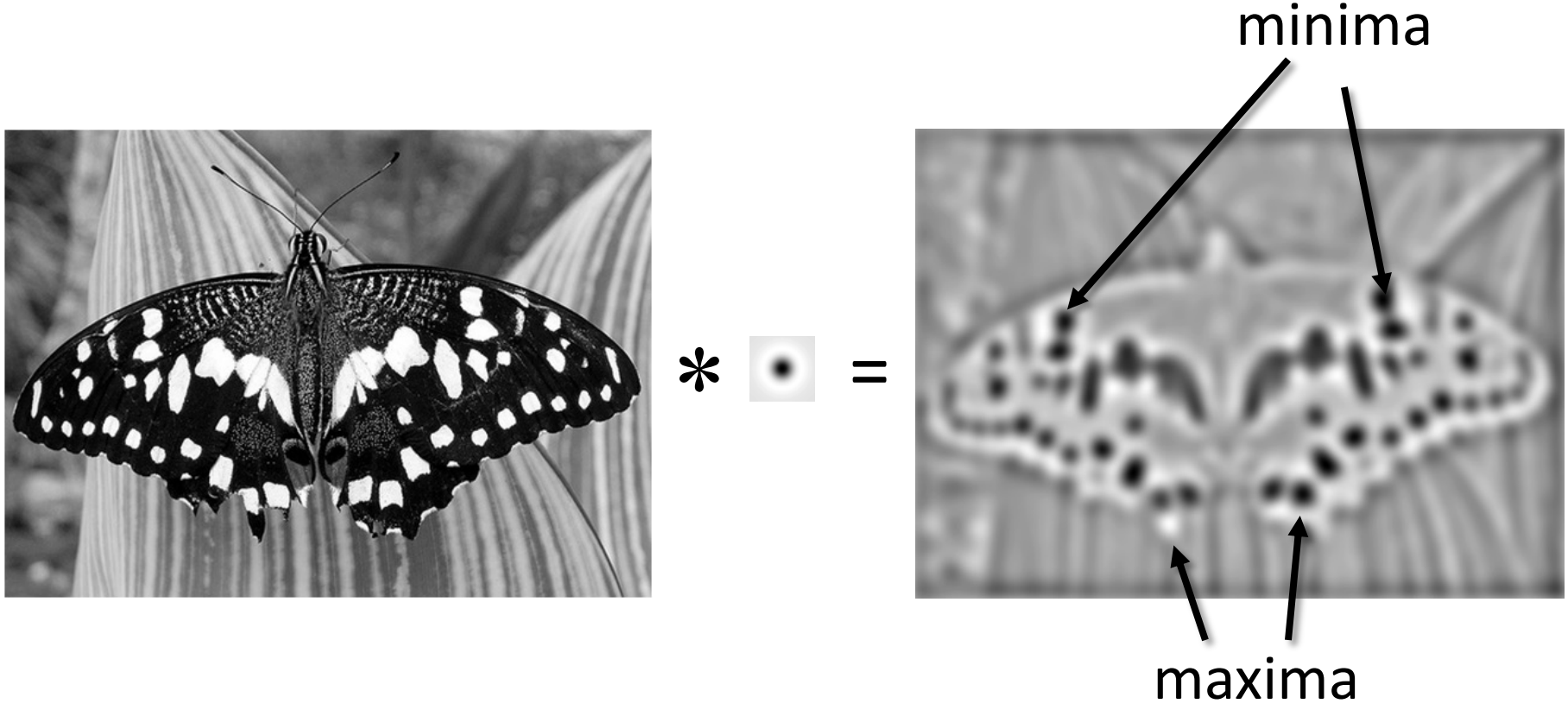




T. Lindeberg, Feature detection with automatic scale selection, *IJCV* 30(2), pp 77-116, 1998

Source: L. Lazebnik

Blob detection



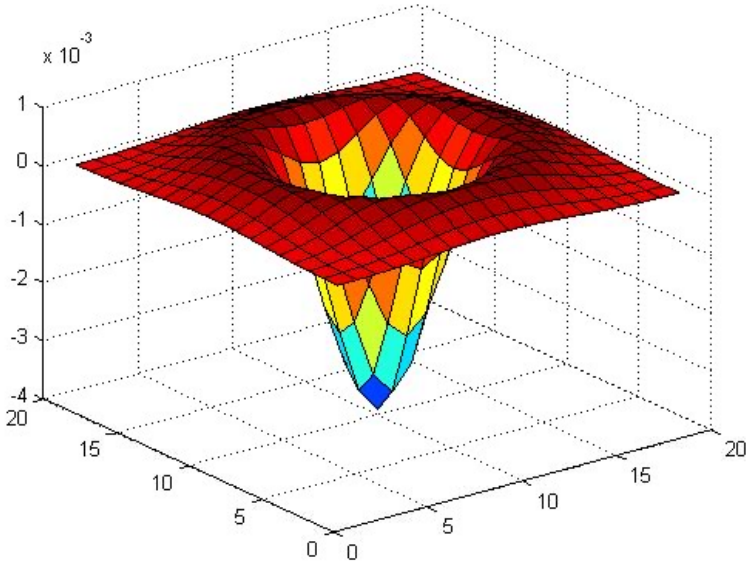
Find maxima *and minima* of blob filter response in space *and scale*

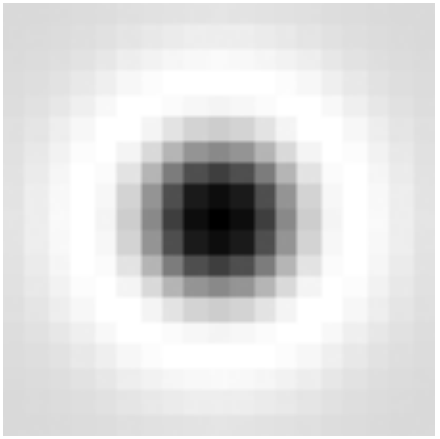
Source: L. Lazebnik

Source: N. Snavely

Blob filter

Laplacian of Gaussian: Circularly symmetric operator for blob detection in 2D



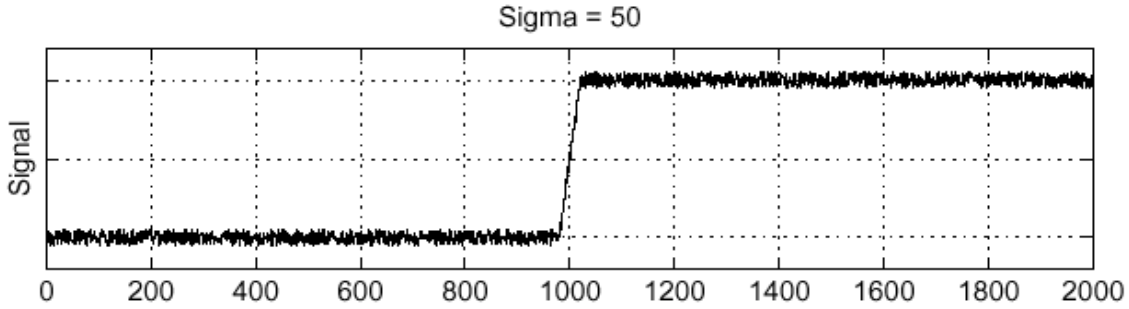


$$\nabla^2 g = \frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2}$$

Source: L. Lazebnik

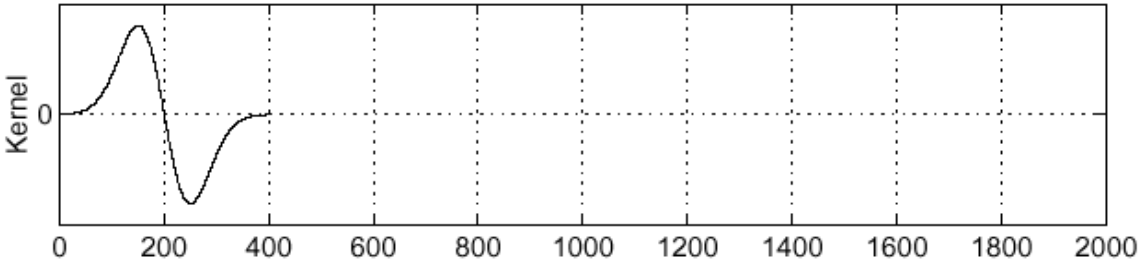
Recall: Edge detection

f



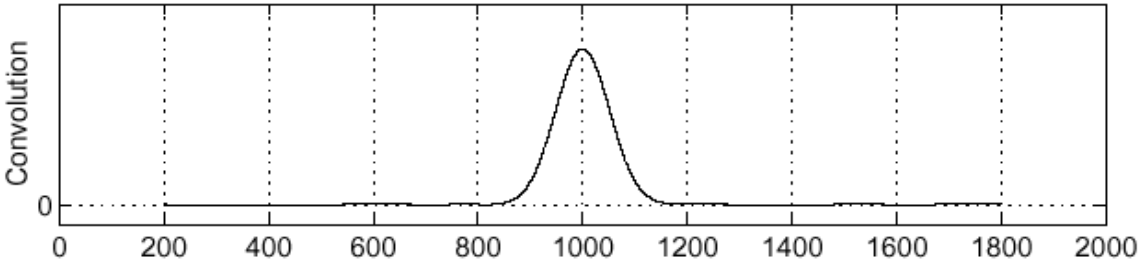
Edge

$\frac{d}{dx} g$



Derivative of Gaussian

$f * \frac{d}{dx} g$



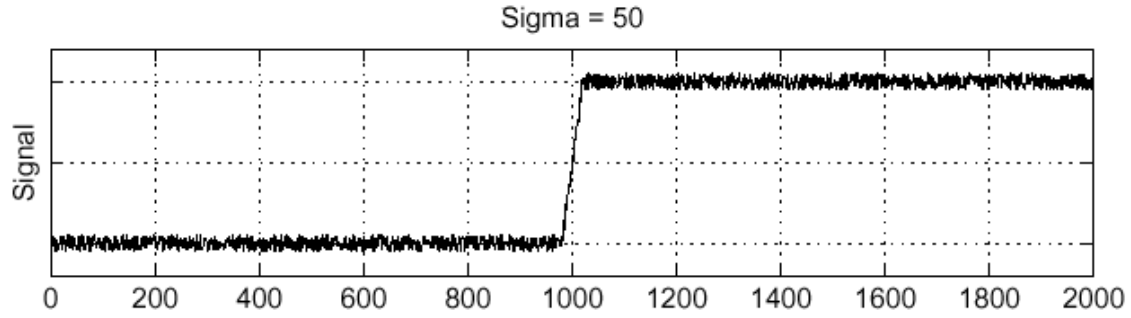
Edge = maximum of derivative

Source: L. Lazebnik

Source: S. Seitz

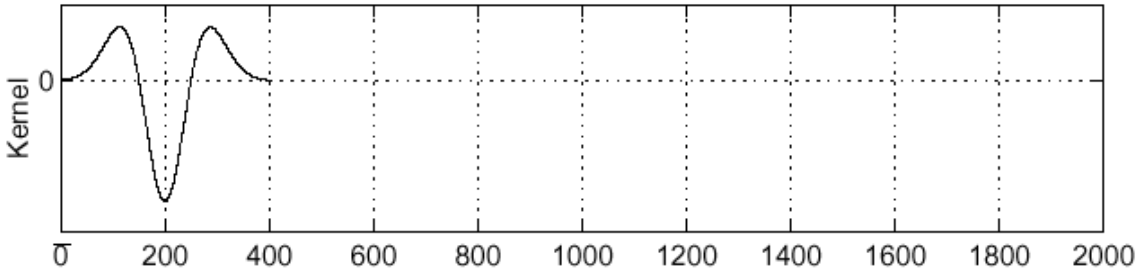
Edge detection, Take 2

f



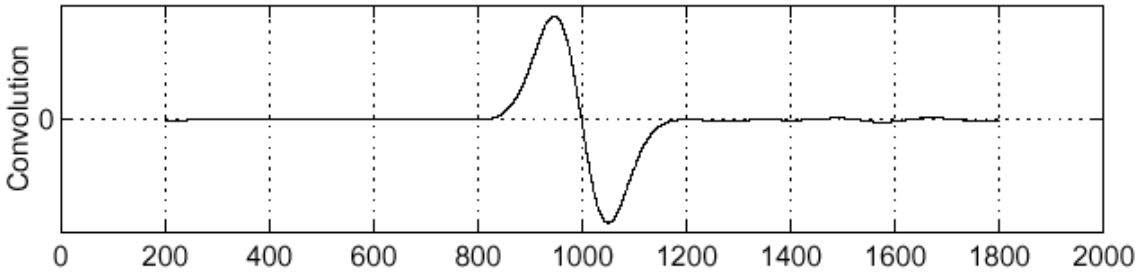
Edge

$$\frac{d^2}{dx^2} g$$



Second derivative of Gaussian (Laplacian)

$$f * \frac{d^2}{dx^2} g$$



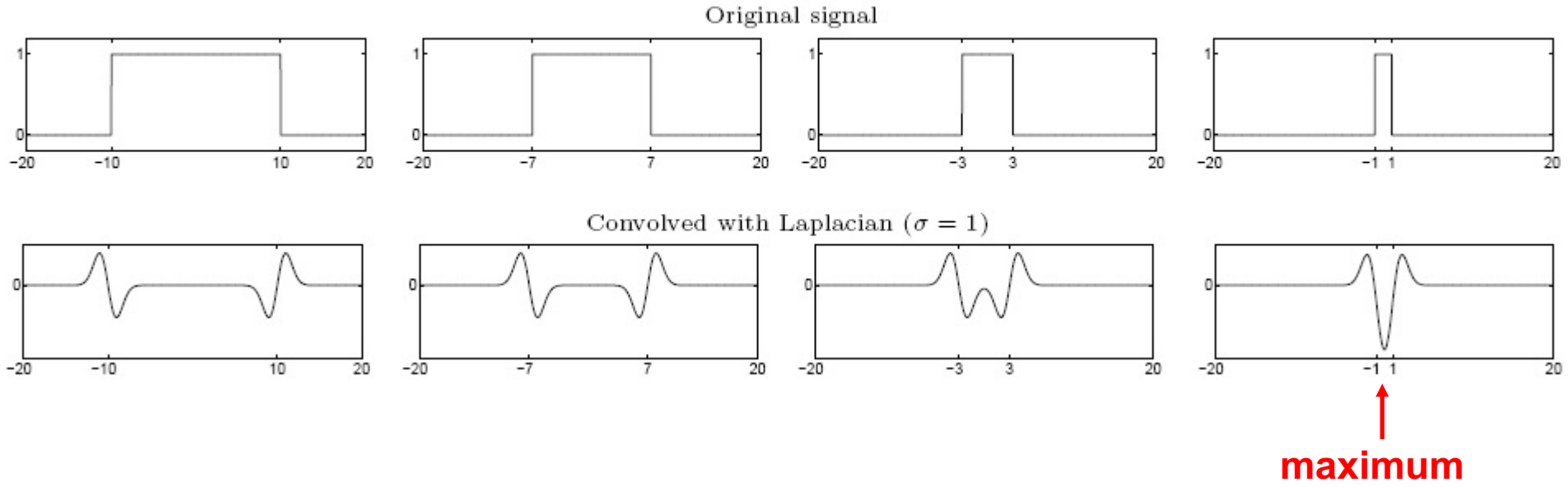
Edge = zero crossing of second derivative

Source: L. Lazebnik

Source: S. Seitz

From edges to blobs

- Edge = ripple
- Blob = superposition of two ripples

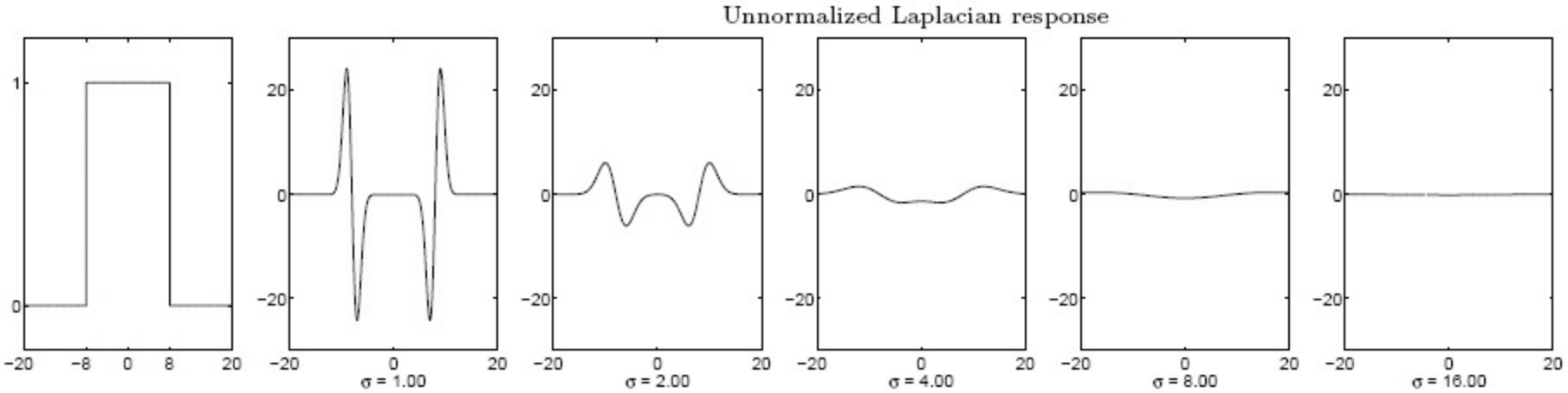


Spatial selection: the magnitude of the Laplacian response will achieve a maximum at the center of the blob, provided the scale of the Laplacian is "matched" to the scale of the blob

Source: L. Lazebnik

Scale selection

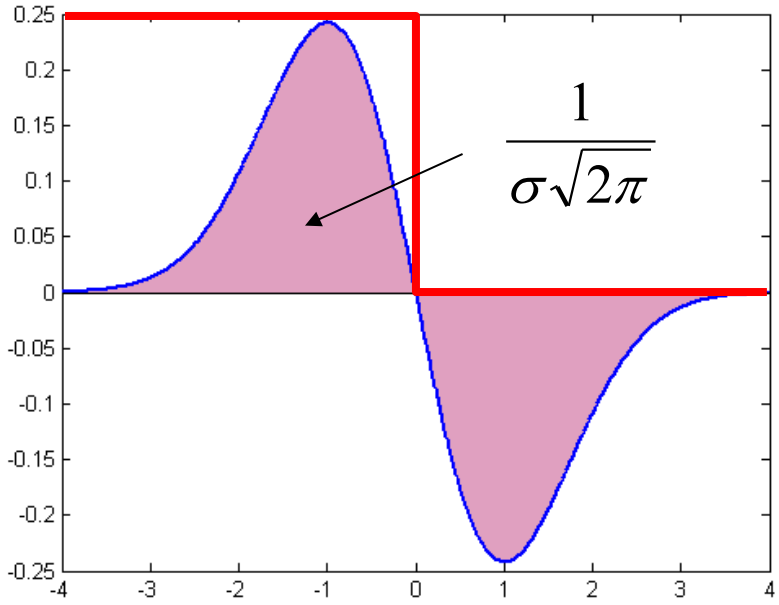
- We want to find the characteristic scale of the blob by convolving it with Laplacians at several scales and looking for the maximum response
- However, Laplacian response decays as scale increases:



Source: L. Lazebnik

Scale normalization

- The response of a derivative of Gaussian filter to a perfect step edge decreases as σ increases:



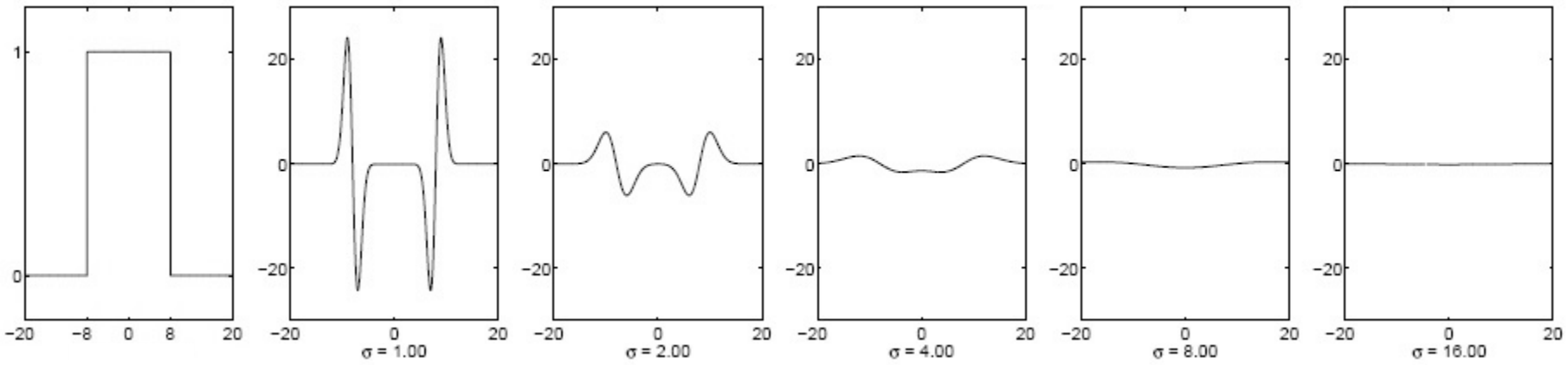
- To keep response the same (scale-invariant), must multiply Gaussian derivative by σ
- Laplacian is the second Gaussian derivative, so it must be multiplied by σ^2

Source: L. Lazebnik

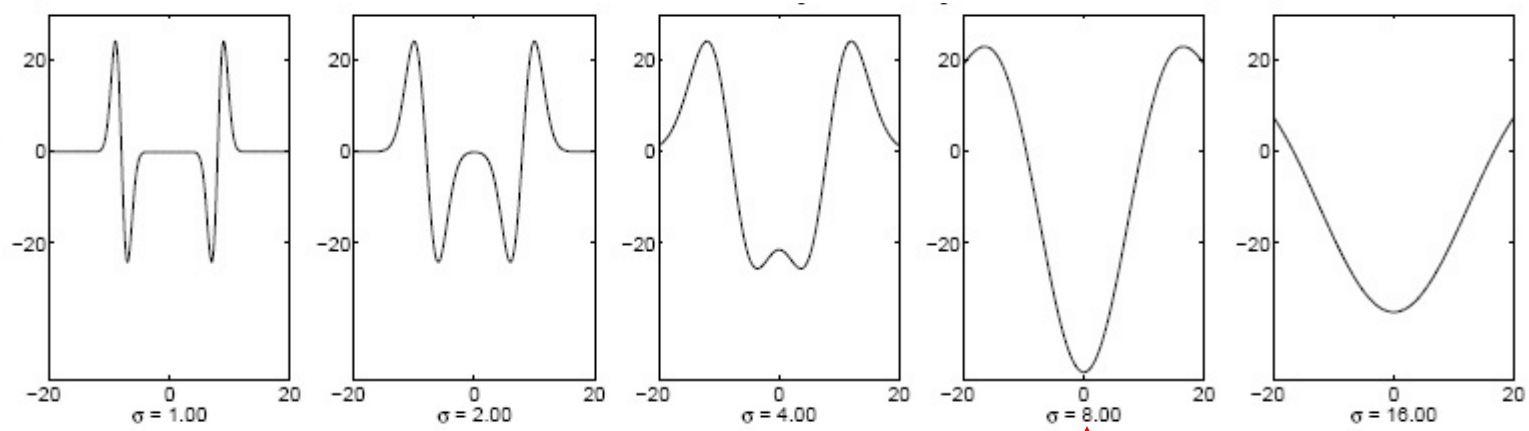
Effect of scale normalization

Original signal

Unnormalized Laplacian response



Scale-normalized Laplacian response



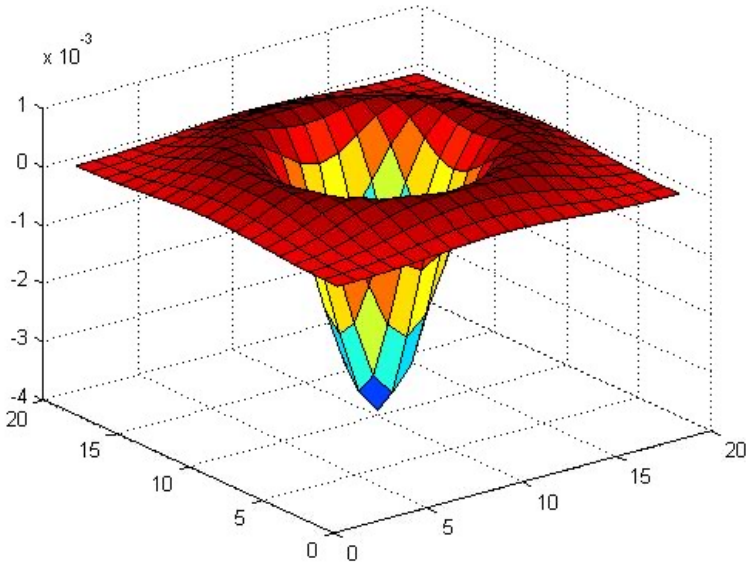
maximum

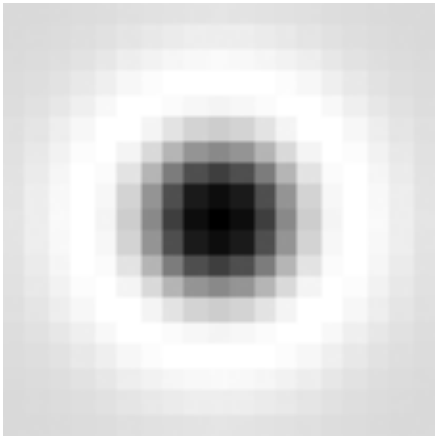
Source: L. Lazebnik

Blob detection in 2D

- *Scale-normalized* Laplacian of Gaussian:

$$\nabla^2_{\text{norm}} g = \sigma^2 \left(\frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2} \right)$$

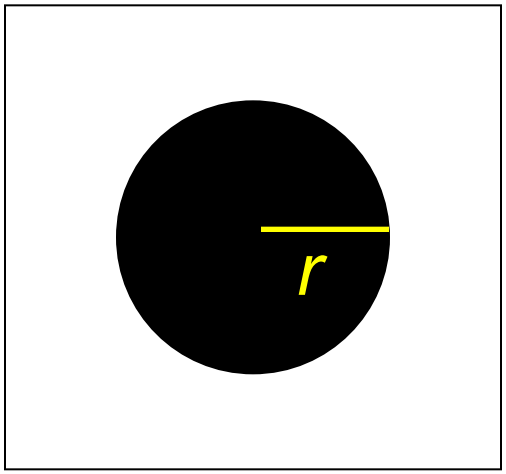




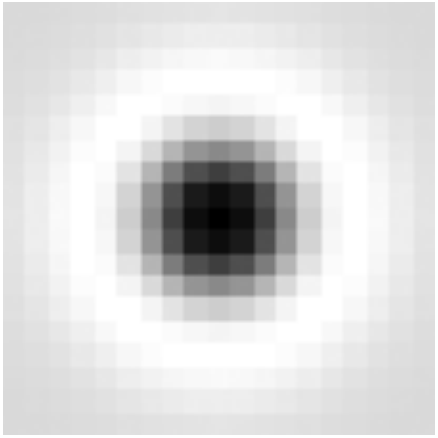
Source: L. Lazebnik

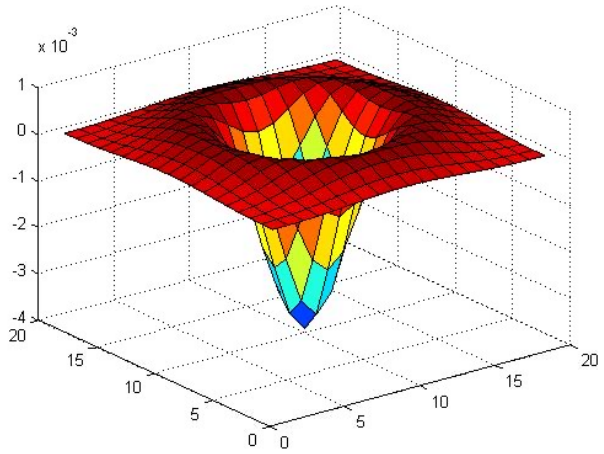
Blob detection in 2D

- At what scale does the Laplacian achieve a maximum response to a binary circle of radius r?



image





Laplacian

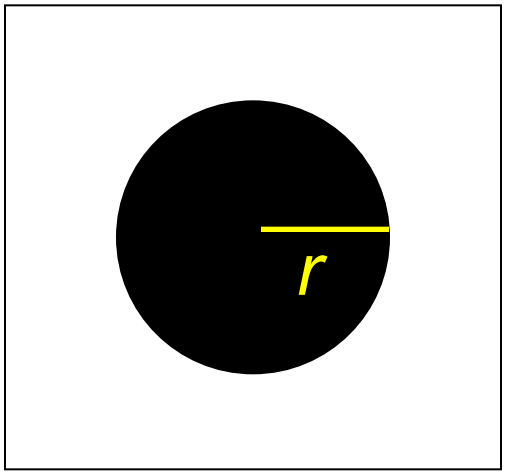
Source: L. Lazebnik

Blob detection in 2D

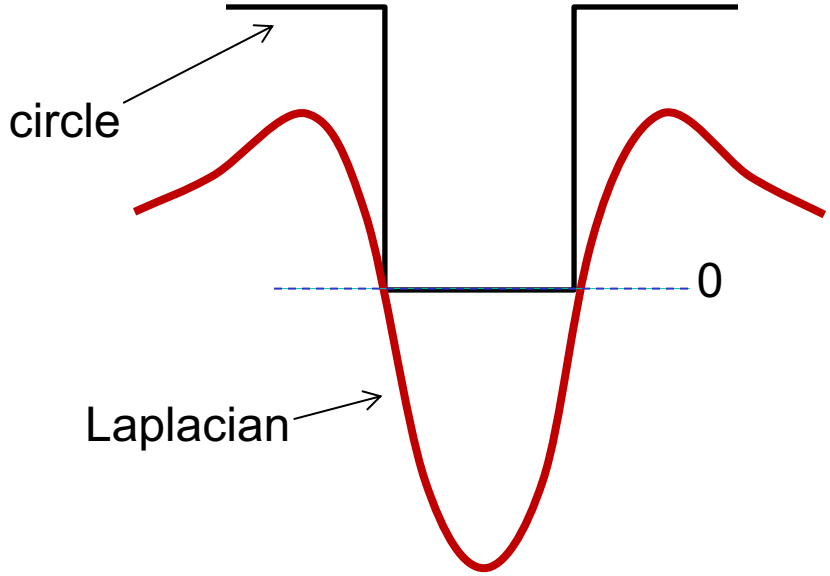
- At what scale does the Laplacian achieve a maximum response to a binary circle of radius r?
- To get maximum response, the zeros of the Laplacian have to be aligned with the circle
- The Laplacian is given by (up to scale):

$$(x^2 + y^2 - 2\sigma^2)\, e^{-(x^2+y^2)/2\sigma^2}$$

- Therefore, the maximum response occurs at $\sigma = r/\sqrt{2}$.



image



Source: L. Lazebnik

Scale-space blob detector

1. Convolve image with scale-normalized Laplacian at several scales

Source: L. Lazebnik

Scale-space blob detector: Example

Source: L. Lazebnik

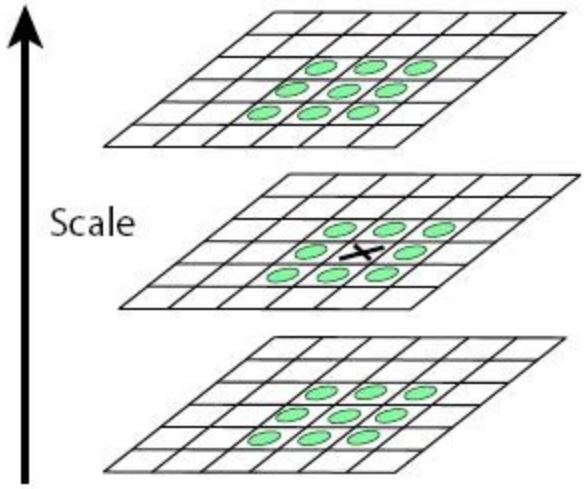
Scale-space blob detector: Example

sigma = 11.9912

Source: L. Lazebnik

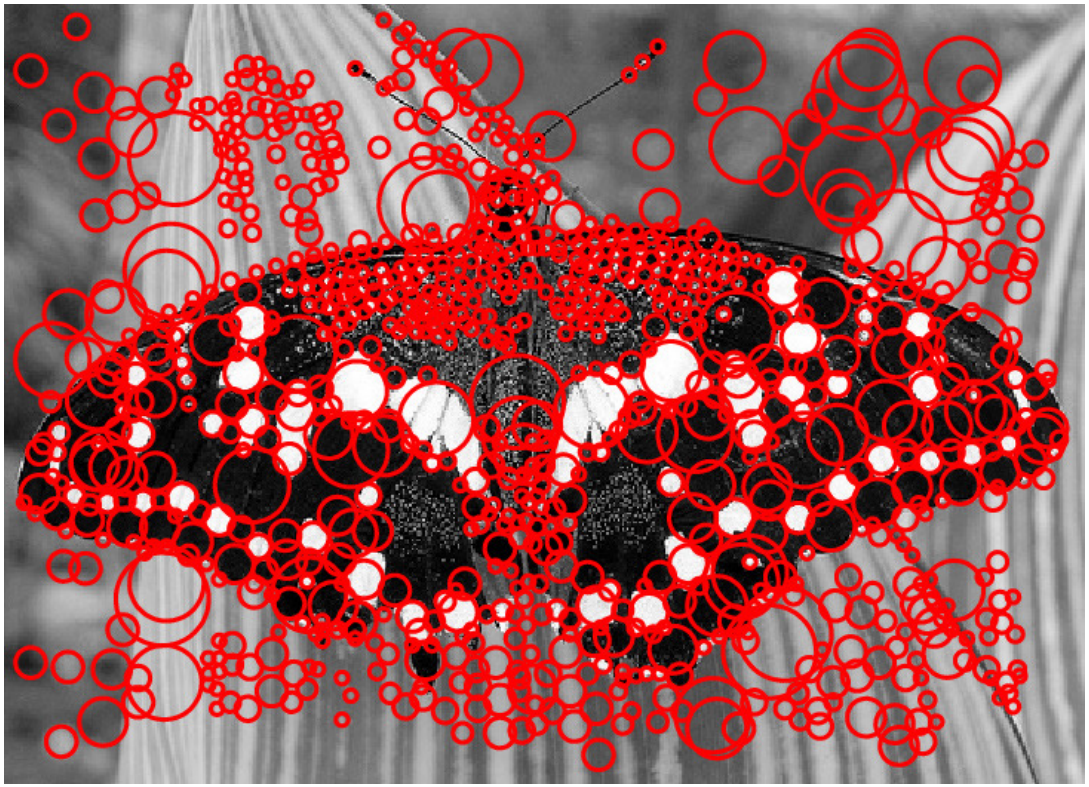
Scale-space blob detector

1. Convolve image with scale-normalized Laplacian at several scales
2. Find maxima of squared Laplacian response in scale-space



Source: L. Lazebnik

Scale-space blob detector: Example



Source: L. Lazebnik

Efficient implementation

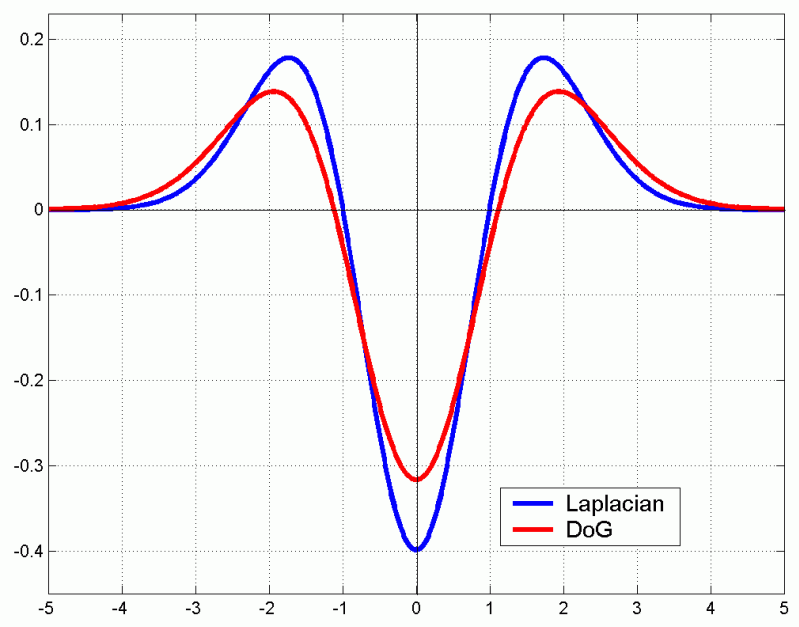
- Approximating the Laplacian with a difference of Gaussians:

$$L = \sigma^2 \left(G_{xx}(x, y, \sigma) + G_{yy}(x, y, \sigma) \right)$$

(Laplacian)

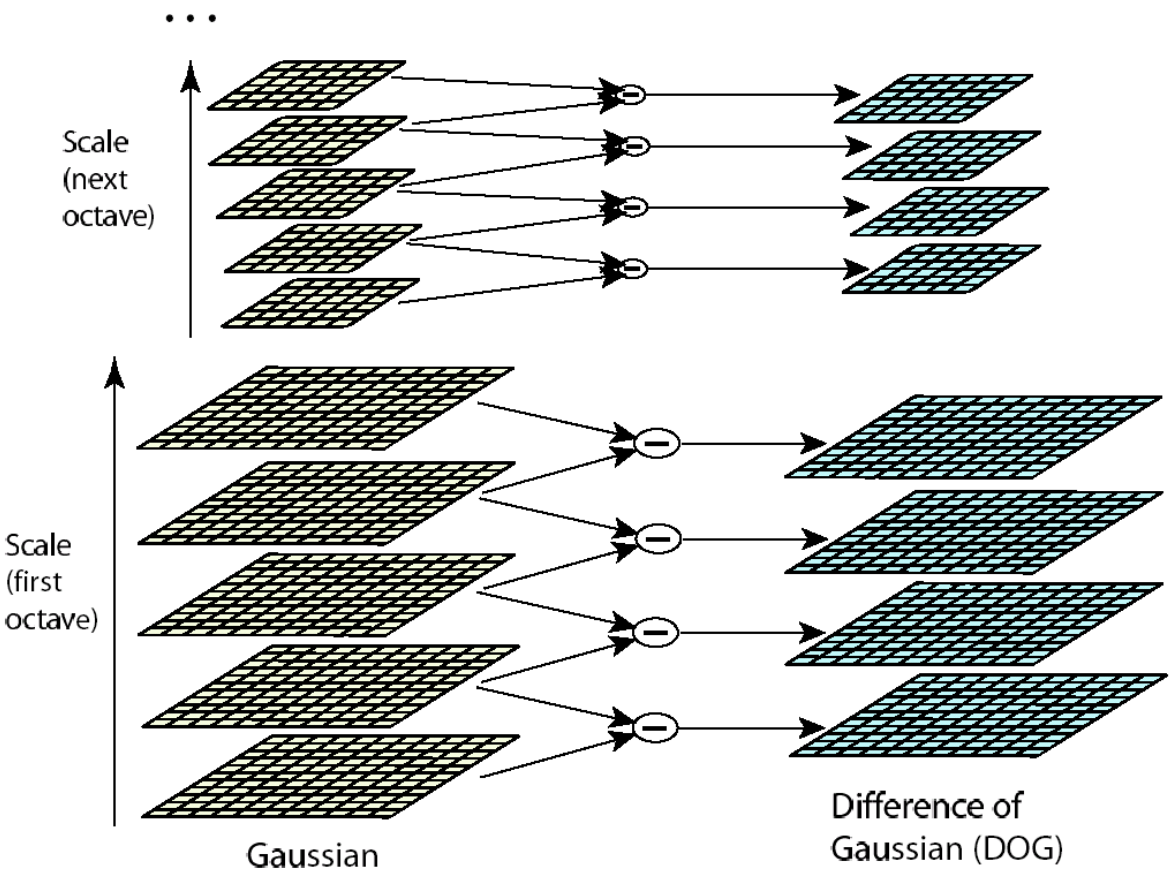
$$DoG = G(x, y, k\sigma) - G(x, y, \sigma)$$

(Difference of Gaussians)



Source: L. Lazebnik

Efficient implementation

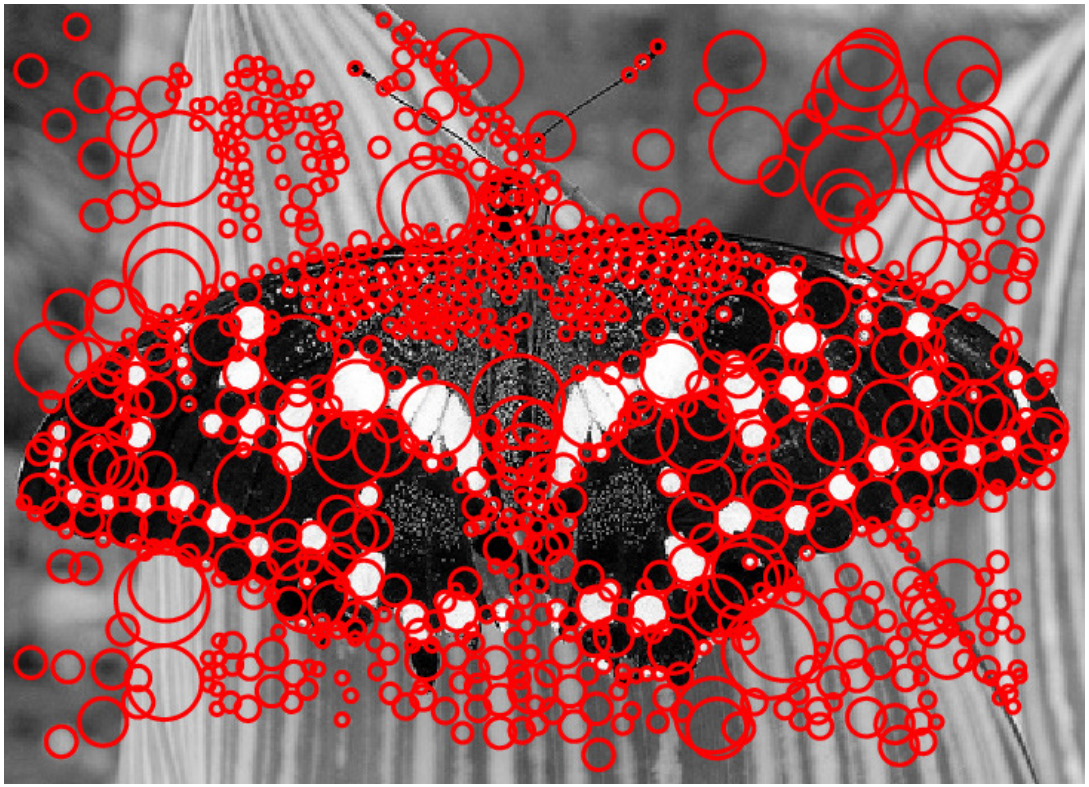


David G. Lowe. **"Distinctive image features from scale-invariant keypoints."** *IJCV* 60 (2), pp. 91-110, 2004.

Source: L. Lazebnik

Eliminating edge responses

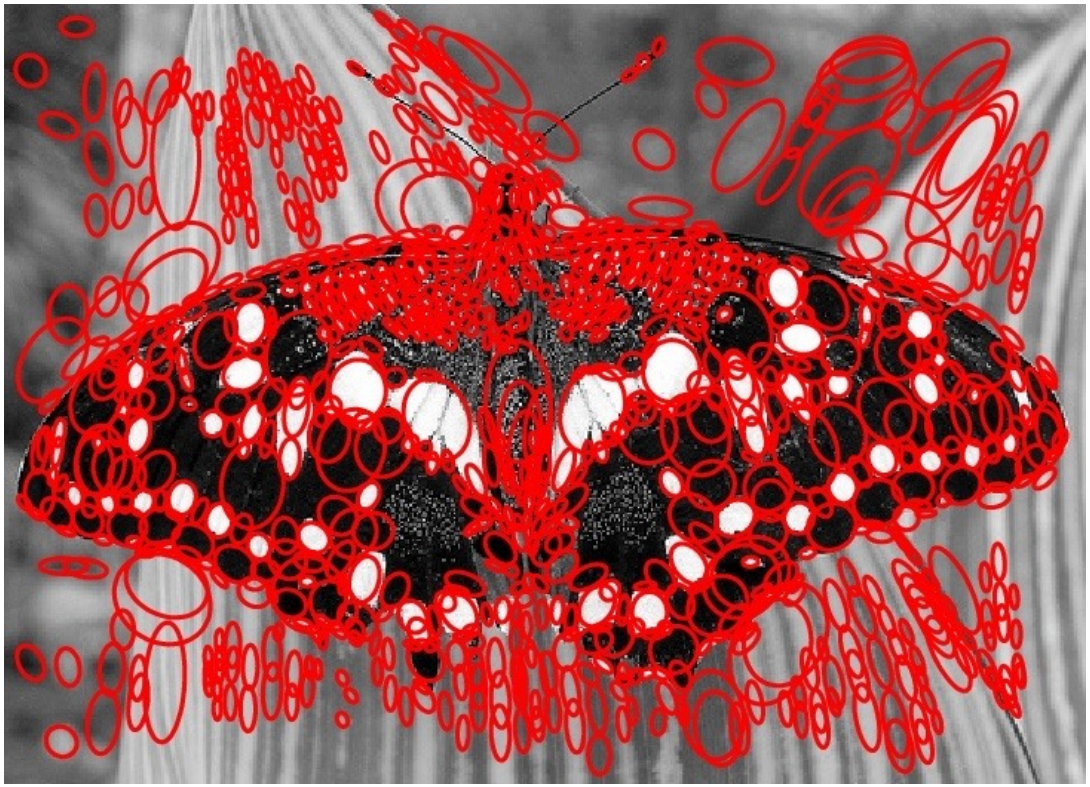
- Laplacian has strong response along edges



Source: L. Lazebnik

Eliminating edge responses

- Laplacian has strong response along edges

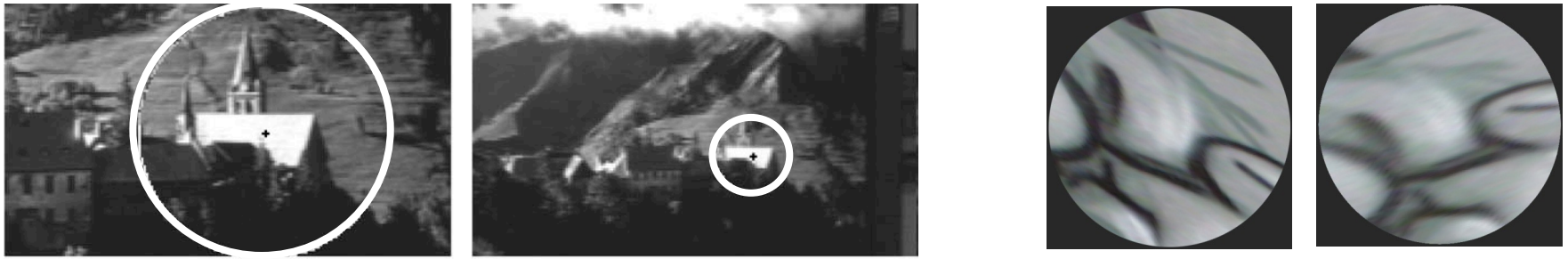


- Solution: filter based on Harris response function over neighborhoods containing the “blobs”

Source: L. Lazebnik

From feature detection to feature description

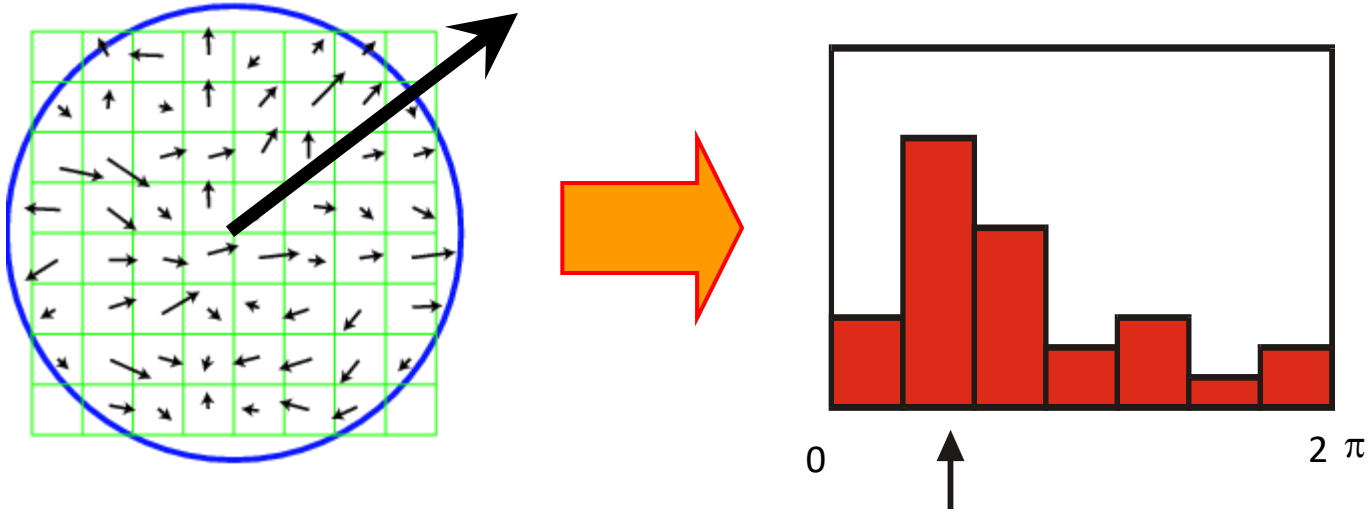
- To recognize the same pattern in multiple images, we need to match appearance "signatures" in the neighborhoods of extracted keypoints
  - But corresponding neighborhoods can be related by a scale change or rotation
  - We want to *normalize* neighborhoods to make signatures invariant to these transformations



Source: L. Lazebnik

Finding a reference orientation

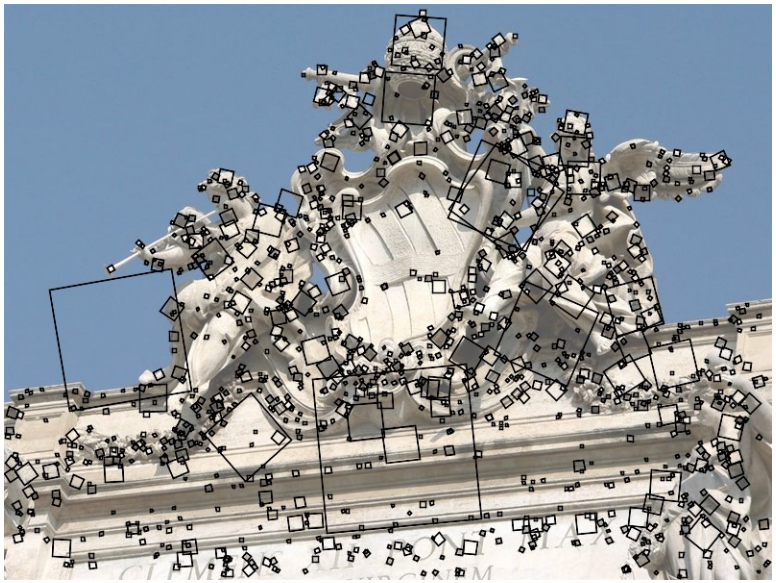
- Create histogram of local gradient directions in the patch
- Assign reference orientation at peak of smoothed histogram



Source: L. Lazebnik

SIFT features

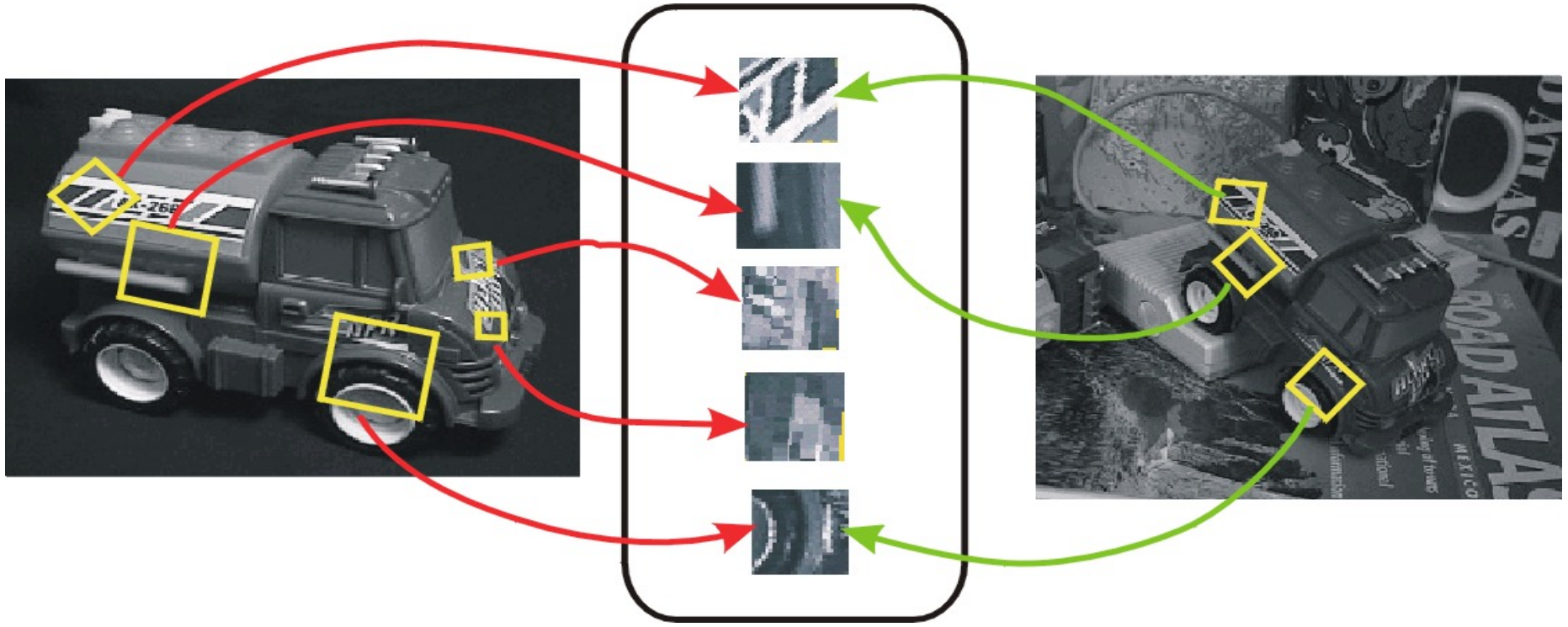
- Detected features with characteristic scales and orientations:



David G. Lowe. **"Distinctive image features from scale-invariant keypoints."** *IJCV* 60 (2), pp. 91-110, 2004.

Source: L. Lazebnik

From keypoint detection to feature description



Detection is *covariant*:

features(transform(image)) = transform(features(image))

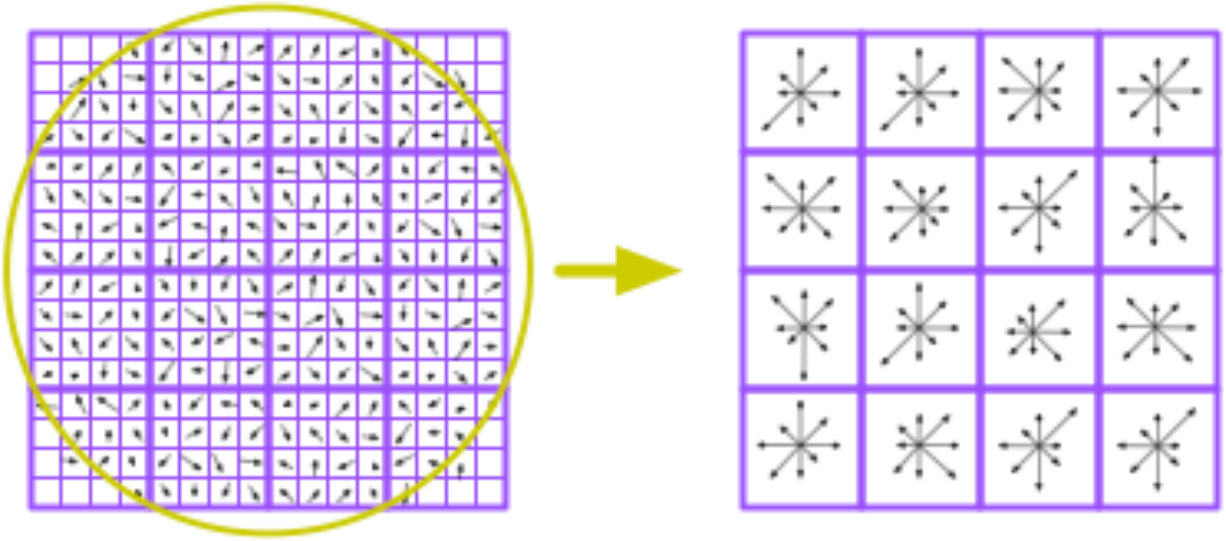
Description is *invariant*:

features(transform(image)) = features(image)

Source: L. Lazebnik

SIFT descriptors

- Inspiration: complex neurons in the primary visual cortex



D. Lowe, Distinctive image features from scale-invariant keypoints, *IJCV* 60 (2), pp. 91-110, 2004

Source: L. Lazebnik

Properties of SIFT

Extraordinarily robust detection and description technique

- Can handle changes in viewpoint
  - Up to about 60 degree out-of-plane rotation
- Can handle significant changes in illumination
  - Sometimes even day vs. night
- Fast and efficient—can run in real time
- Lots of code available

Fitting

Computer Vision

CS 543 / ECE 549

University of Illinois

Slides from Derek Hoiem, Svetlana Lazebnik

Fitting lines to point sets

In general, fit a function from a given function class to samples from the function

Fitting: Overview

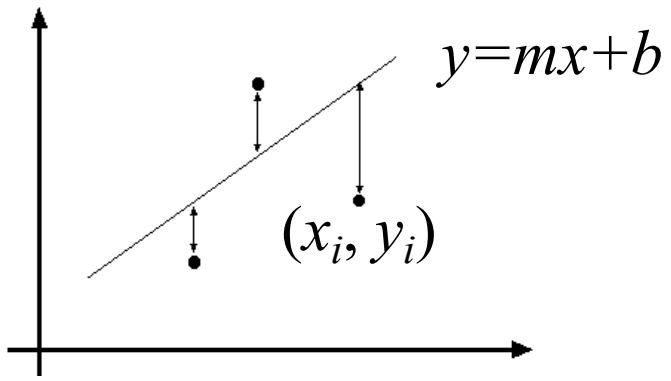
- If we know which points belong to the line, how do we find the “optimal” line parameters?
  - Least squares
- What if there are outliers?
  - Robust fitting, RANSAC
- What if there are many lines?
  - Voting methods: RANSAC, Hough transform
- What if we’re not even sure it’s a line?
  - Model selection (not covered)

Simple example: Fitting a line

Slide from D. Hoiem

Least squares line fitting

- Data: $(x_1, y_1), \ldots, (x_n, y_n)$
- Line equation: $y_i = m x_i + b$
- Find (m, b) to minimize



$$E = \sum_{i=1}^{n} (y_i - mx_i - b)^2$$

$$E = \sum_{i=1}^{n} \left(\begin{bmatrix} x_i & 1 \end{bmatrix} \begin{bmatrix} m \\ b \end{bmatrix} - y_i \right)^2 = \left\| \begin{bmatrix} x_1 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{bmatrix} \begin{bmatrix} m \\ b \end{bmatrix} - \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \right\|^2 = \|\mathbf{Ap} - \mathbf{y}\|^2$$

$$= \mathbf{y}^T\mathbf{y} - 2(\mathbf{Ap})^T\mathbf{y} + (\mathbf{Ap})^T(\mathbf{Ap})$$

$$\frac{dE}{dp} = 2\mathbf{A}^T\mathbf{Ap} - 2\mathbf{A}^T\mathbf{y} = 0$$

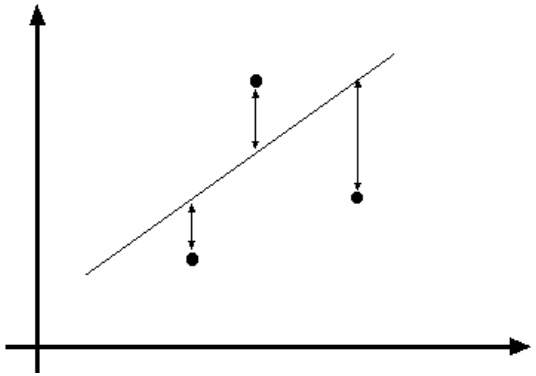
Matlab: `p = A \ y;`

$$\mathbf{A}^T\mathbf{Ap} = \mathbf{A}^T\mathbf{y} \Rightarrow \mathbf{p} = \left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T\mathbf{y}$$

Modified from S. Lazebnik

Problem with “vertical” least squares

- Not rotation-invariant
- Fails completely for vertical lines



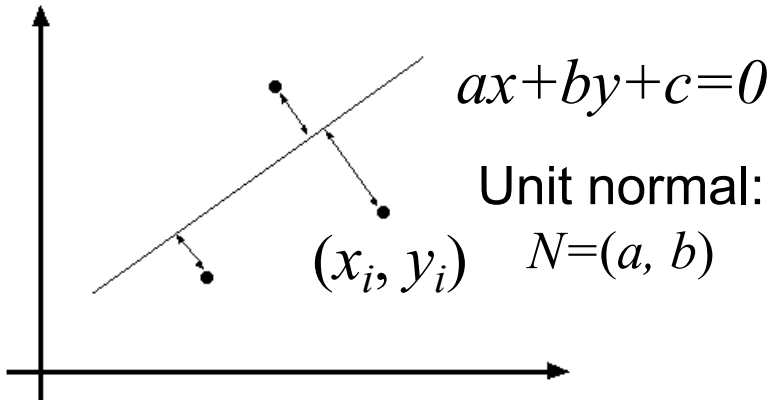
Slide from S. Lazebnik

Total least squares

If $(a^2+b^2=1)$ then

Distance between point (x_i, y_i) and line $ax+by+c=0$ is $|ax_i + by_i + c|$

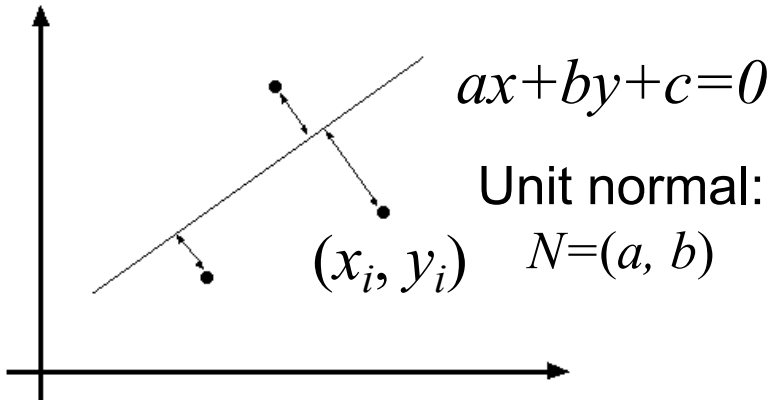
proof:
http://mathworld.wolfram.com/Point-LineDistance2-Dimensional.html



Total least squares

If ($a^2+b^2=1$) then

Distance between point (x_i, y_i) and line $ax+by+c=0$ is $|ax_i + by_i + c|$



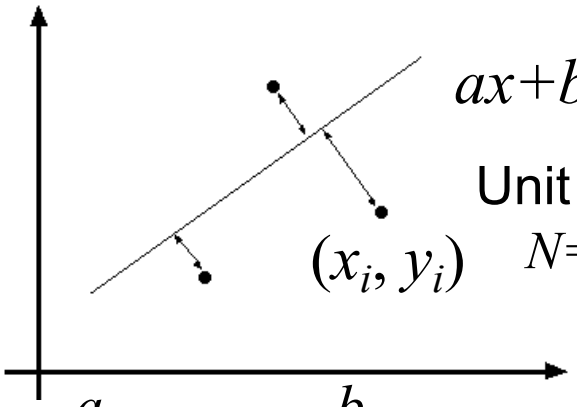
Find (a, b, c) to minimize the sum of squared perpendicular distances

$$E = \sum_{i=1}^{n} (ax_i + by_i + c)^2$$

Total least squares

Find (a, b, c) to minimize the sum of squared perpendicular distances

$$E = \sum_{i=1}^{n} (ax_i + by_i + c)^2$$



$$\frac{\partial E}{\partial c} = \sum_{i=1}^{n} 2(ax_i + by_i + c) = 0$$

$$c = -\frac{a}{n}\sum_{i=1}^{n} x_i - \frac{b}{n}\sum_{i=1}^{n} y_i = -a\bar{x} - b\bar{y}$$

$$E = \sum_{i=1}^{n} (a(x_i - \bar{x}) + b(y_i - \bar{y}))^2 = \left\| \begin{bmatrix} x_1 - \bar{x} & y_1 - \bar{y} \\ \vdots & \vdots \\ x_n - \bar{x} & y_n - \bar{y} \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} \right\|^2 = \mathbf{p}^T \mathbf{A}^T \mathbf{A} \mathbf{p}$$

$$\text{minimize} \mathbf{p}^T \mathbf{A}^T \mathbf{A} \mathbf{p} \quad \text{s.t. } \mathbf{p}^T \mathbf{p} = 1 \quad \Rightarrow \quad \text{minimize} \frac{\mathbf{p}^T \mathbf{A}^T \mathbf{A} \mathbf{p}}{\mathbf{p}^T \mathbf{p}}$$

Solution is eigenvector corresponding to smallest eigenvalue of A^TA

See details on Raleigh Quotient: http://en.wikipedia.org/wiki/Rayleigh_quotient

Recap: Two Common Optimization Problems

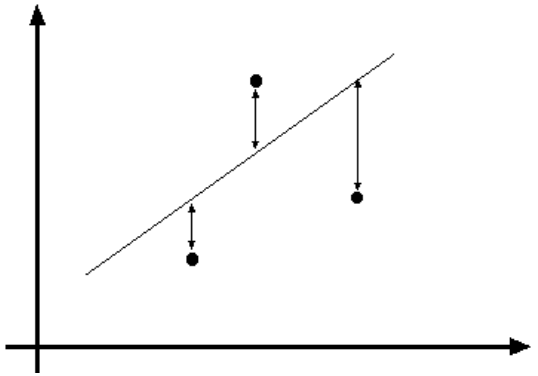
Problem statement	Solution
$\text{minimize } \|\mathbf{Ax}-\mathbf{b}\|^2$	$\mathbf{x}=\left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T\mathbf{b}$
least squares solution to $\mathbf{Ax}=\mathbf{b}$	$\mathbf{x}=\mathbf{A}\setminus\mathbf{b}$ (matlab)

Problem statement	Solution
$\text{minimize } \mathbf{x}^T\mathbf{A}^T\mathbf{Ax} \text{ s.t. } \mathbf{x}^T\mathbf{x}=1$	$[\mathbf{v},\lambda]=\text{eig}(\mathbf{A}^T\mathbf{A})$
$\text{minimize}\dfrac{\mathbf{x}^T\mathbf{A}^T\mathbf{Ax}}{\mathbf{x}^T\mathbf{x}}$	$\lambda_1<\lambda_{2..n}:\mathbf{x}=\mathbf{v}_1$
non - trivial lsq solution to $\mathbf{Ax}=0$	

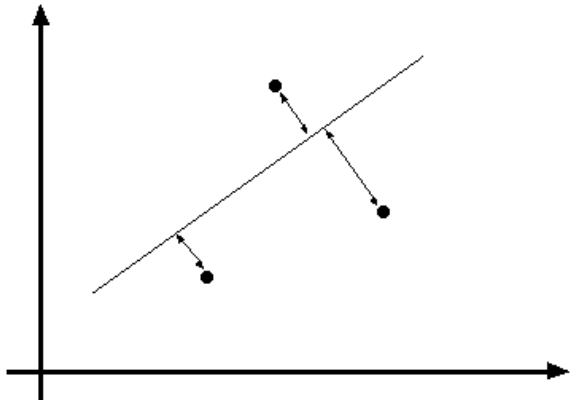
Slide from D. Hoiem

Recap: Fitting Lines

- a. fit y = mx + b



- b. fit ax + by + c = 0



Solution involves:

- 1. Eigen vector
- 2. Pseudo-inverse

- A. a -> 1, b-> 2
- B. a -> 2, b -> 1
- C. a -> 1, b -> 1
- D. a -> 2, b -> 2

Slide from D. Hoiem

Least squares (global) optimization

Good

- Clearly specified objective
- Optimization is easy

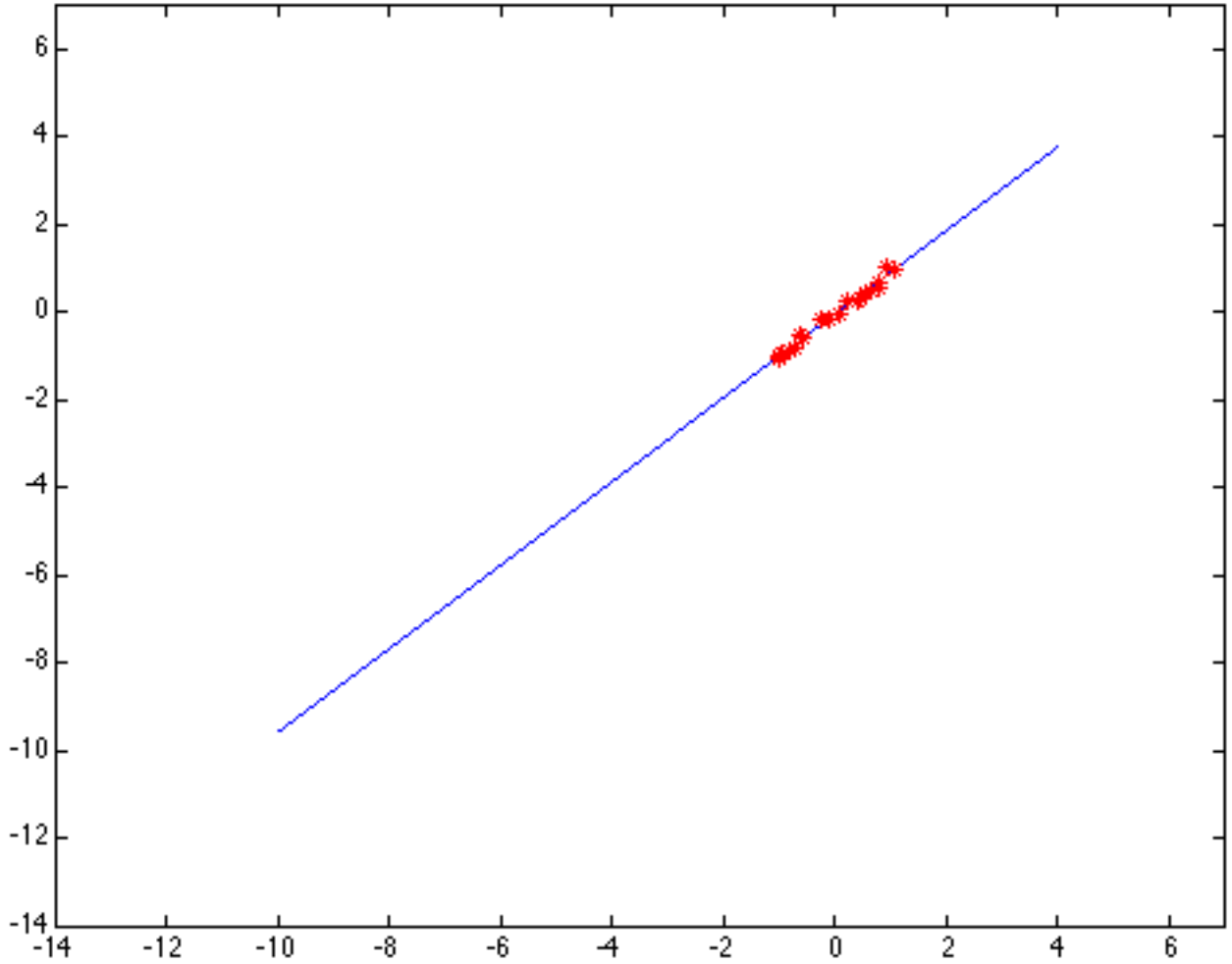
Bad

- May not be what you want to optimize
- Sensitive to outliers
  - Bad matches, extra points
- Doesn’t allow you to get multiple good fits
  - Detecting multiple objects, lines, etc.

Slide from D. Hoiem

Least squares: Robustness to noise

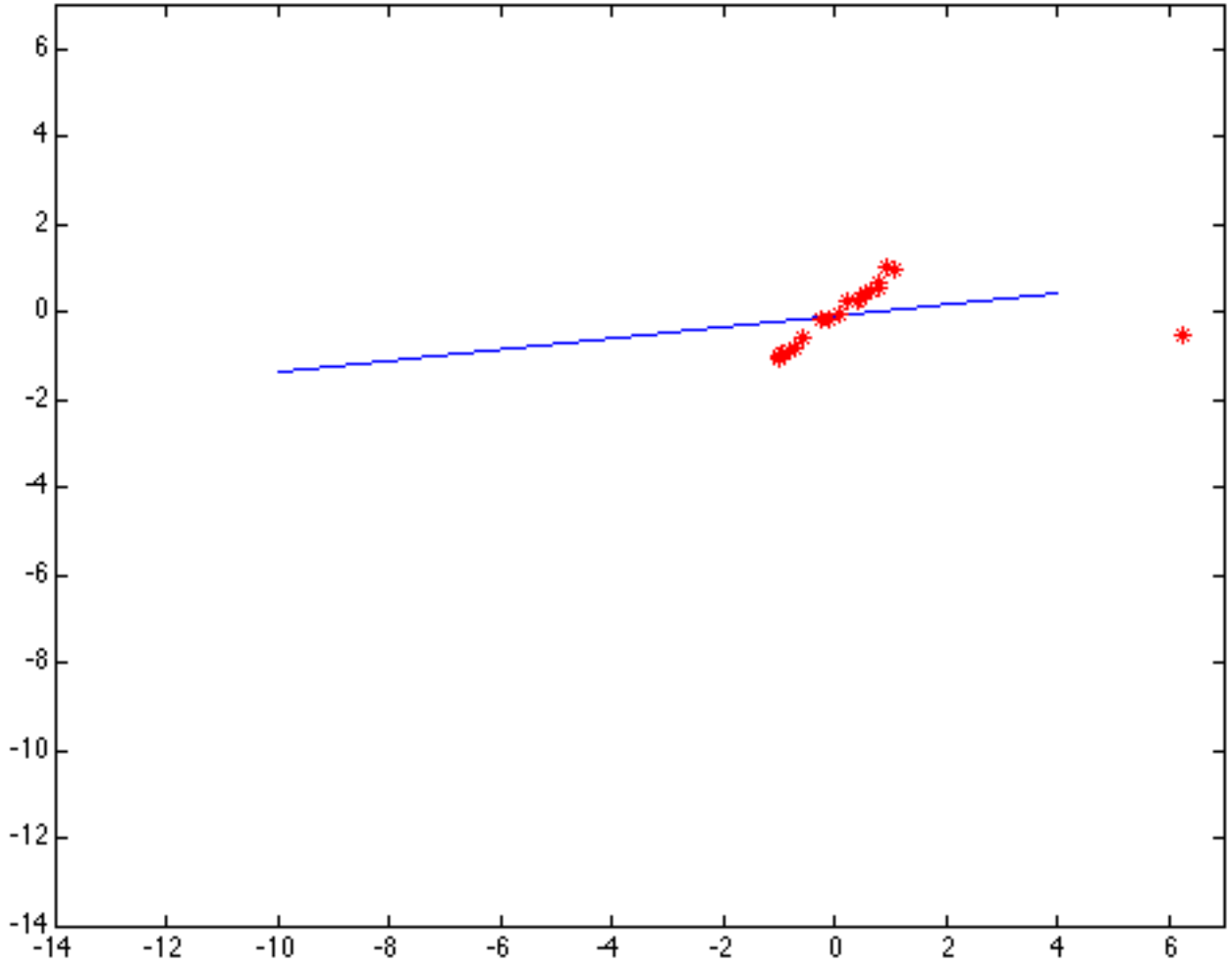
- Least squares fit to the red points:



Slide from L. Lazebnik

Least squares: Robustness to noise

- Least squares fit with an outlier:



Problem: squared error heavily penalizes outliers

Slide from L. Lazebnik

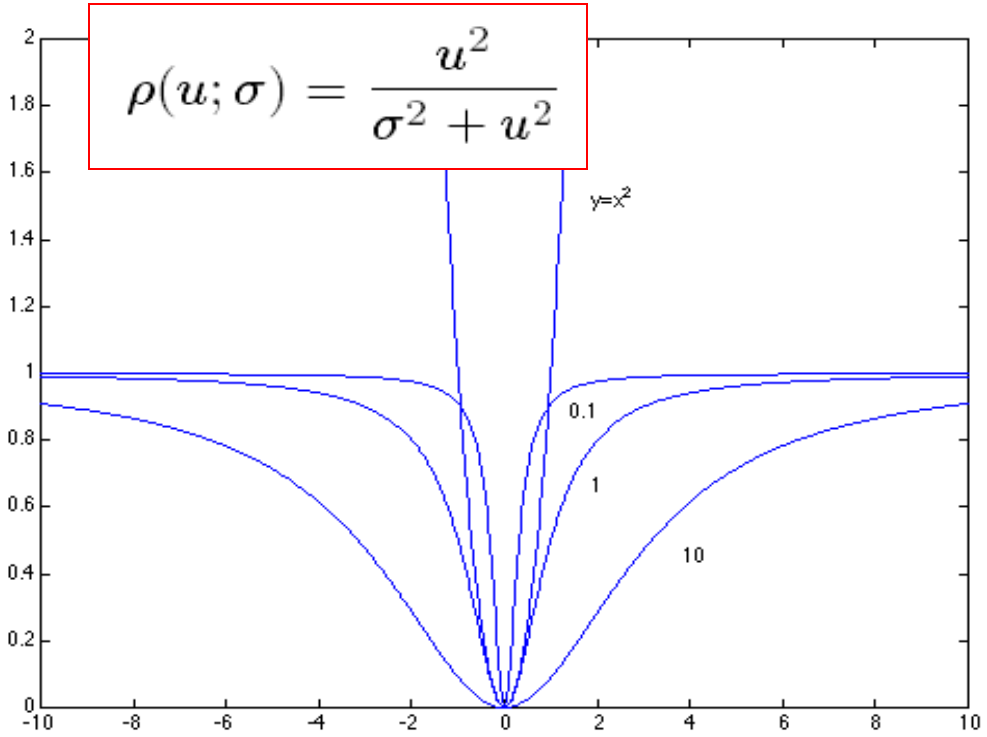
Robust least squares (to deal with outliers)

General approach:

minimize $\sum_i \rho(u_i(x_i, \theta); \sigma)$ $\quad u^2 = \sum_{i=1}^{n} (y_i - mx_i - b)^2$

$u_i(x_i, \theta)$ – residual of i[th] point w.r.t. model parameters ϑ

ρ – robust function with scale parameter σ



The robust function ρ

- Favors a configuration with small residuals
- Constant penalty for large residuals

Slide from S. Savarese

Robust Estimator

1. Initialize: e.g., choose θ by least squares fit and $\sigma = 1.5 \cdot \text{median}(error)$

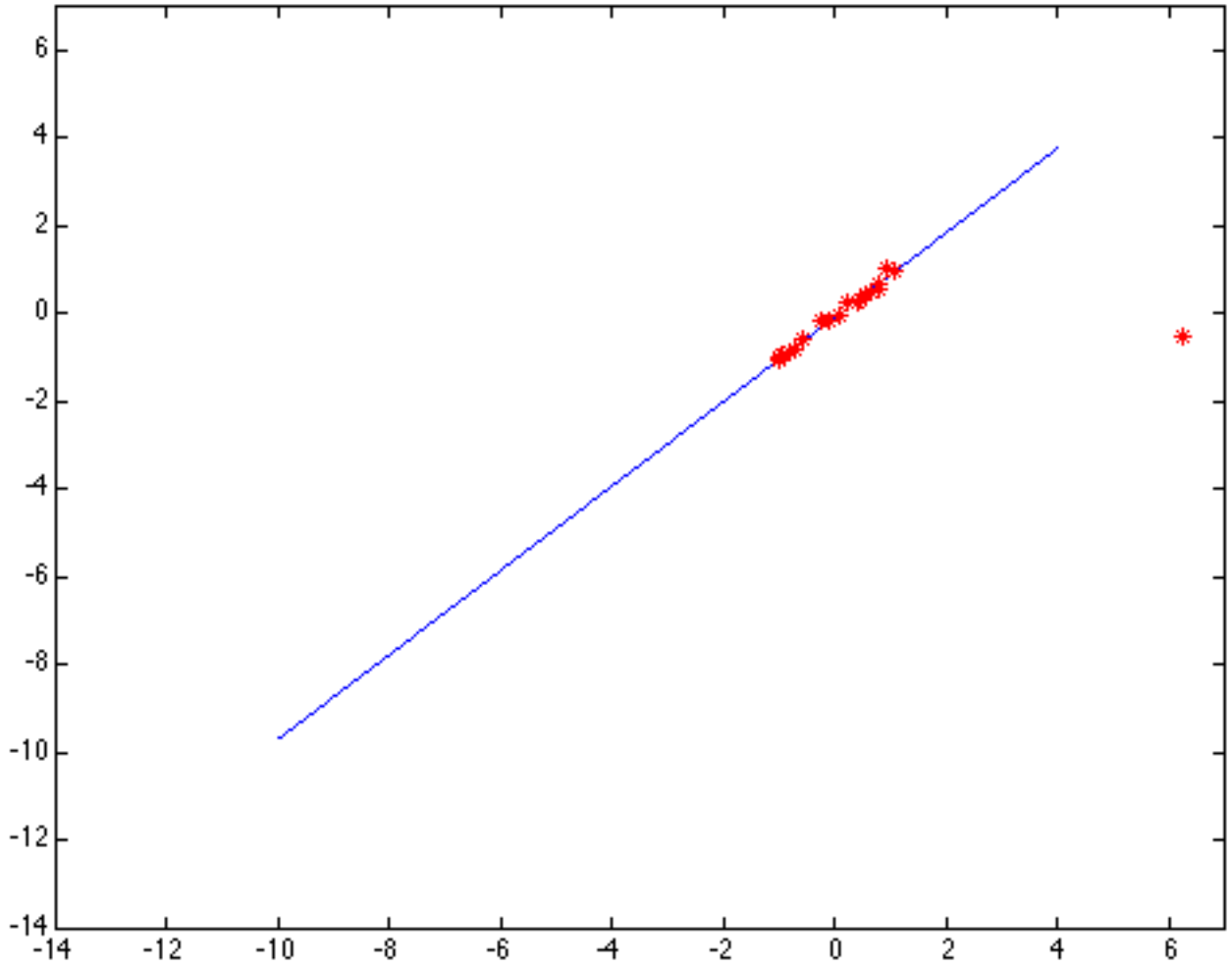
2. Choose params to minimize: $\sum_i \frac{error(\theta, data_i)^2}{\sigma^2 + error(\theta, data_i)^2}$
   - E.g., numerical optimization

3. Compute new $\sigma = 1.5 \cdot \text{median}(error)$

4. Repeat (2) and (3) until convergence

Slide from D. Hoiem

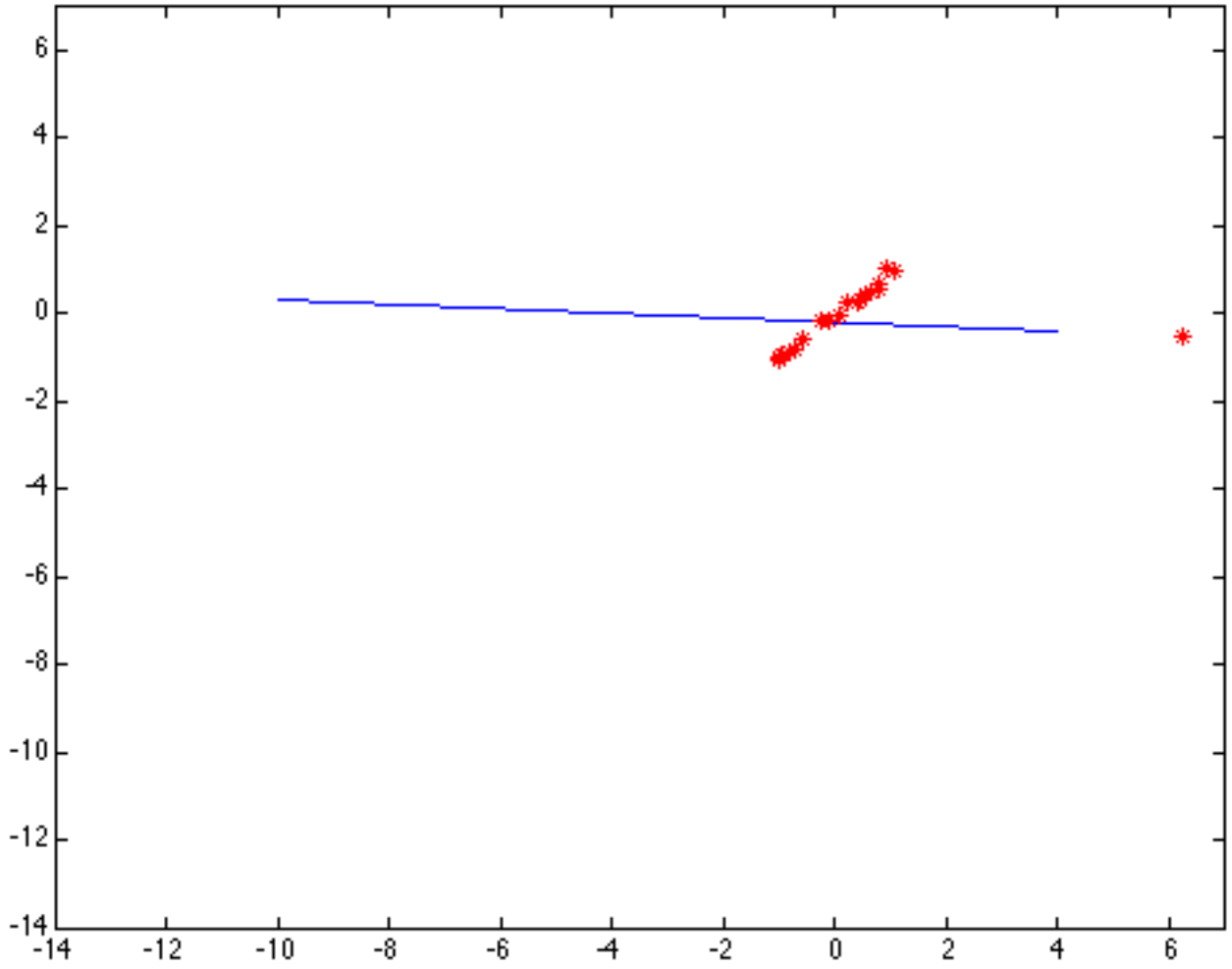
Choosing the scale: Just right



The effect of the outlier is minimized

Slide from L. Lazebnik

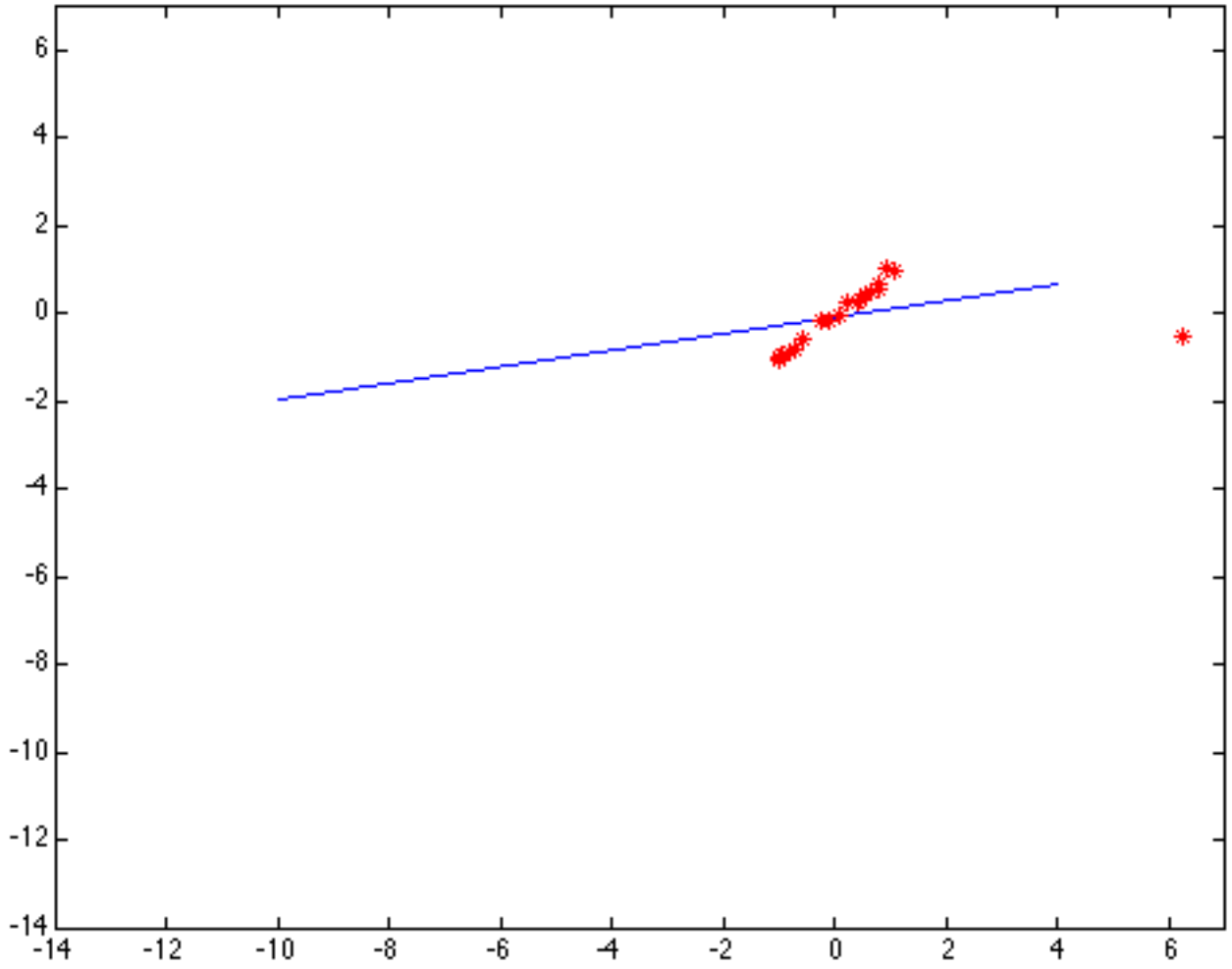
Choosing the scale: Too small



The error value is almost the same for every point and the fit is very poor

Slide from L. Lazebnik

Choosing the scale: Too large



Behaves much the same as least squares

Slide from L. Lazebnik

Fitting: Overview

- If we know which points belong to the line, how do we find the “optimal” line parameters?
  - Least squares
- What if there are outliers?
  - Robust fitting, RANSAC
- What if there are many lines?
  - Voting methods: RANSAC, Hough transform
- What if we’re not even sure it’s a line?
  - Model selection (not covered)

Hough Transform: Outline

1. Create a grid of parameter values

2. Each point votes for a set of parameters, incrementing those values in grid

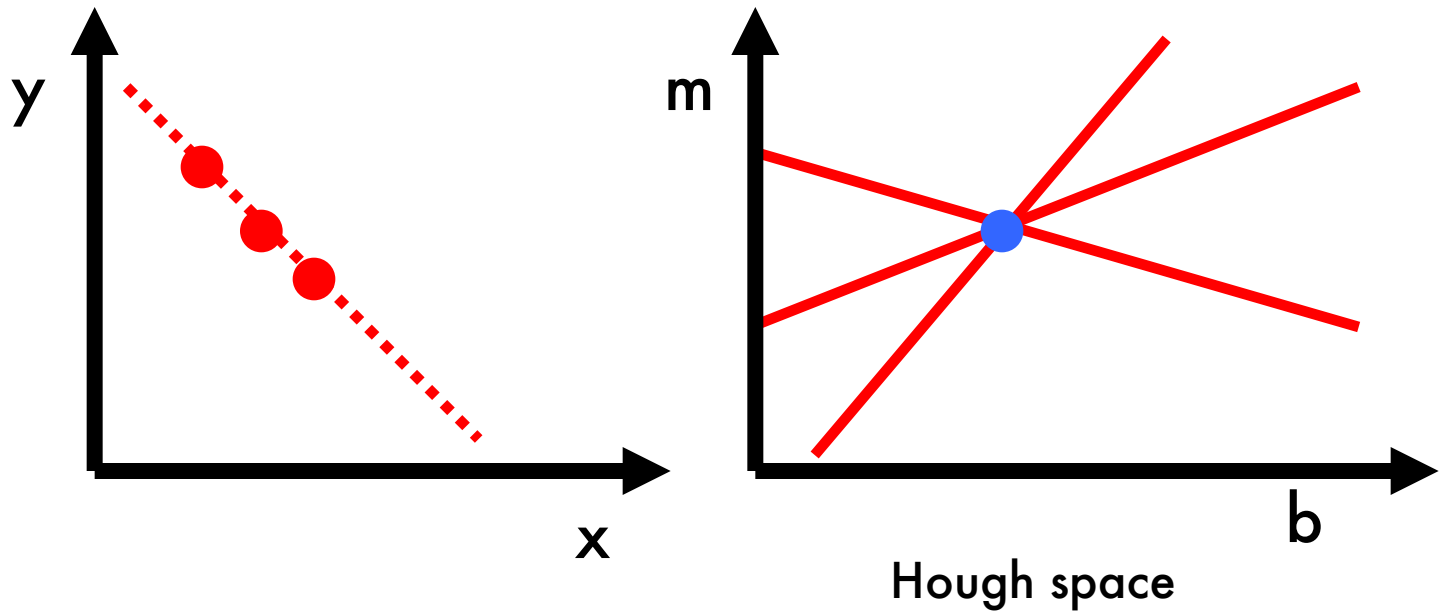
3. Find maximum or local maxima in grid

Slide from D. Hoiem

Hough transform

P.V.C. Hough, *Machine Analysis of Bubble Chamber Pictures,* Proc. Int. Conf. High Energy Accelerators and Instrumentation, 1959

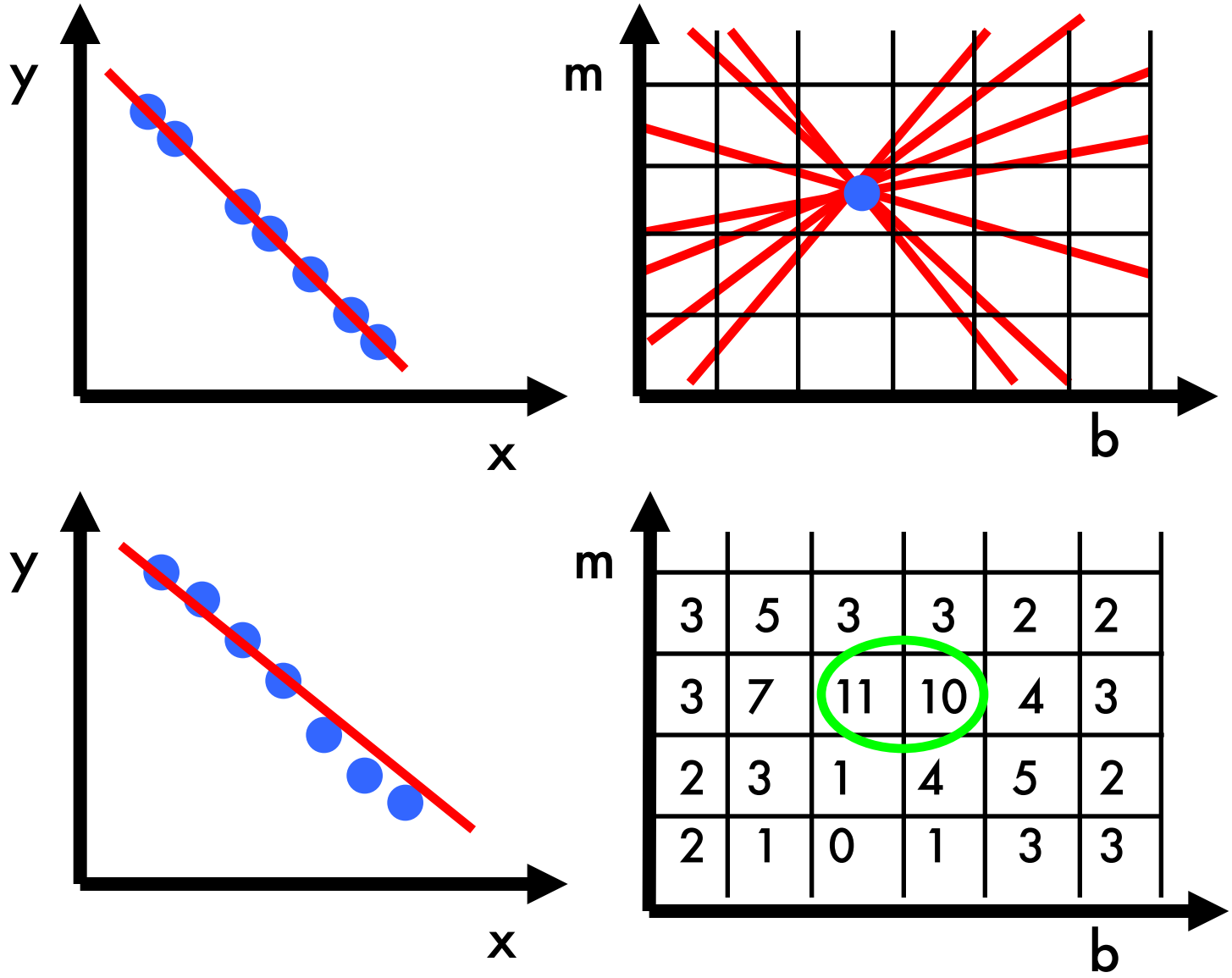
Given a set of points, find the curve or line that explains the data points best



$$y = m\,x + b$$

Slide from S. Savarese

Hough transform



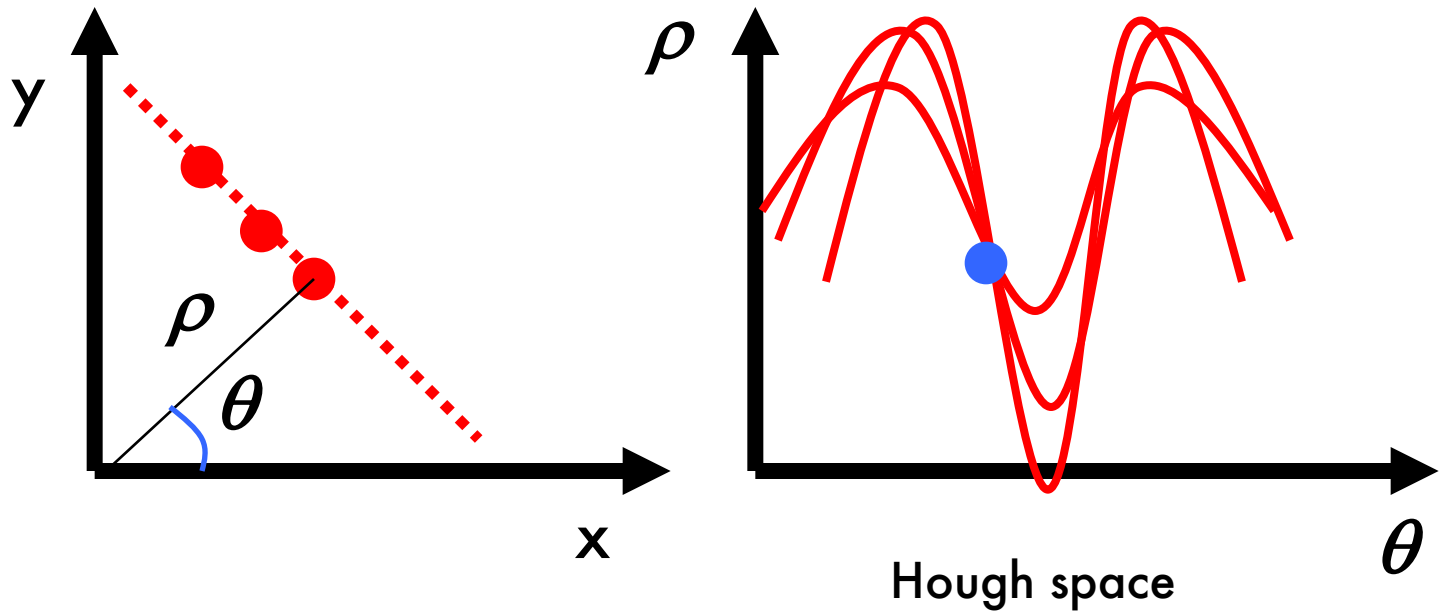
Slide from S. Savarese

Hough transform

P.V.C. Hough, *Machine Analysis of Bubble Chamber Pictures,* Proc. Int. Conf. High Energy Accelerators and Instrumentation, 1959

Issue : parameter space [m,b] is unbounded…

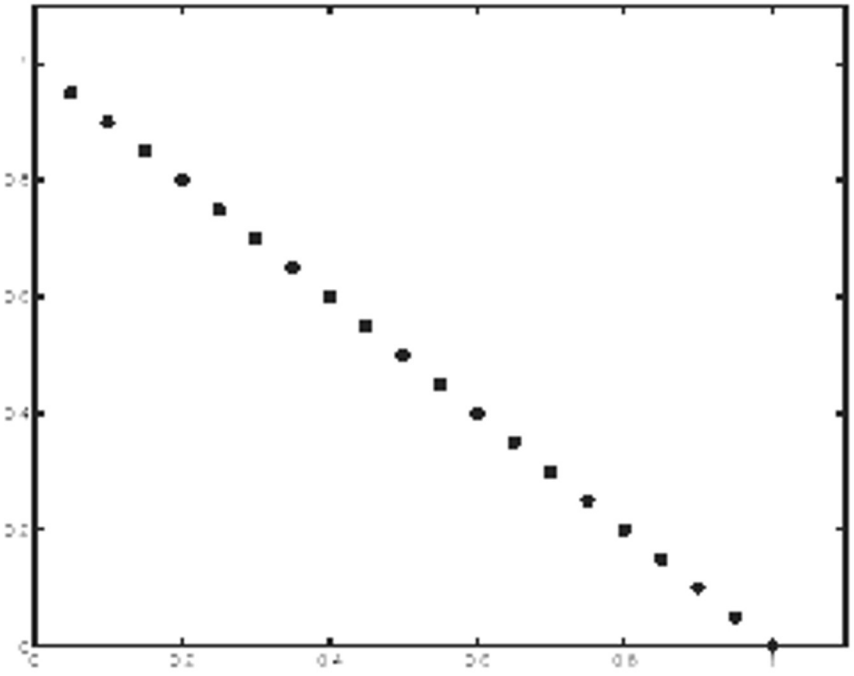
Use a polar representation for the parameter space



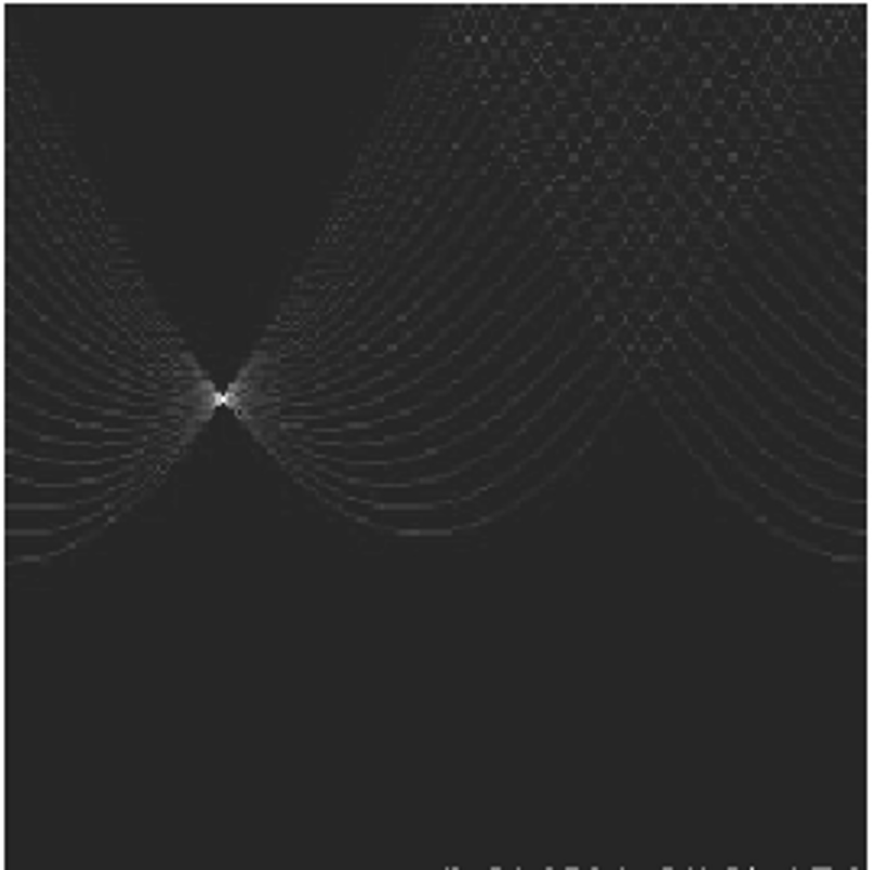
$$x\cos\theta + y\sin\theta = \rho$$

Slide from S. Savarese

Hough transform - experiments



features

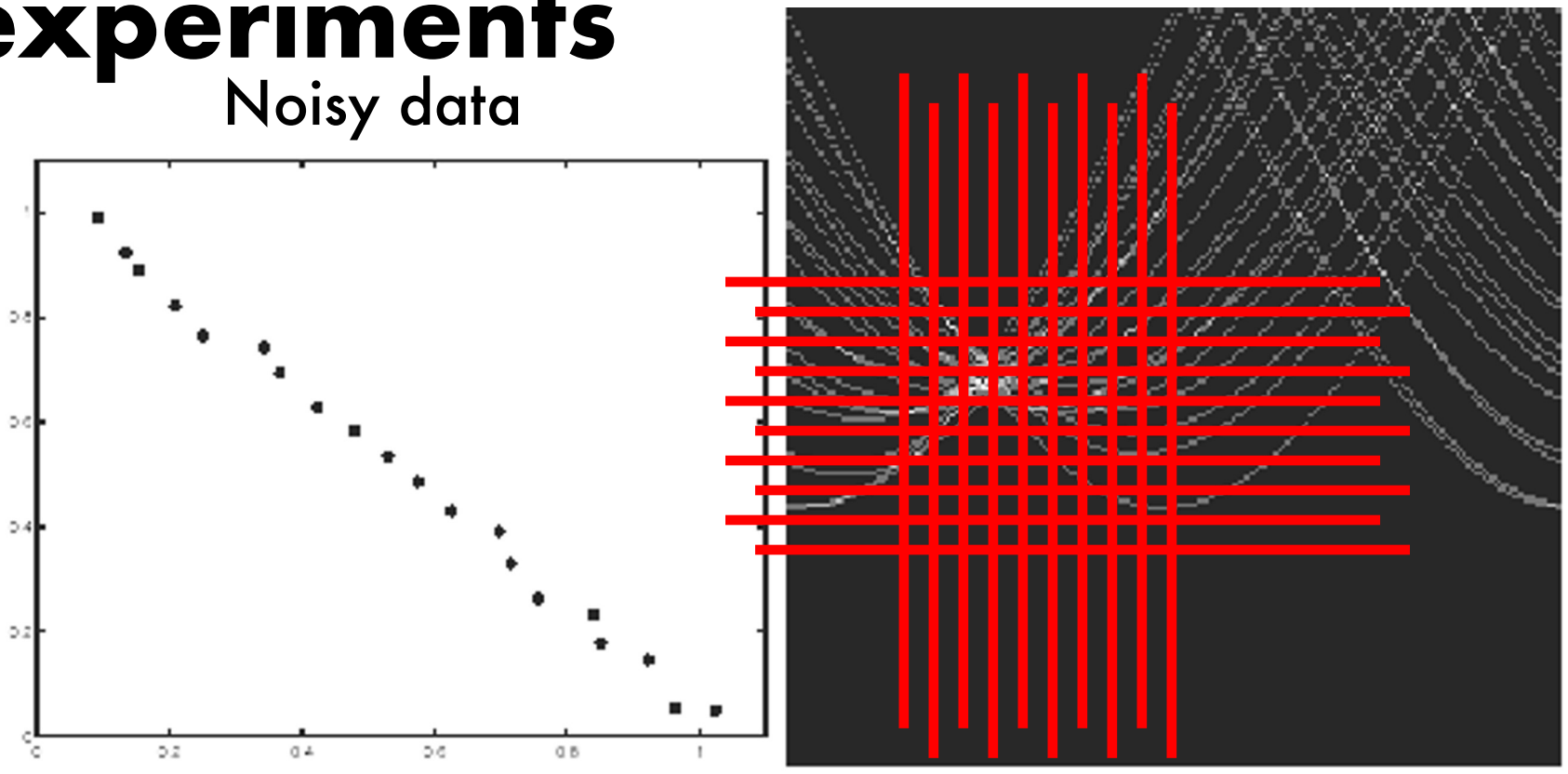


votes

Slide from S. Savarese

Hough transform - experiments

Noisy data

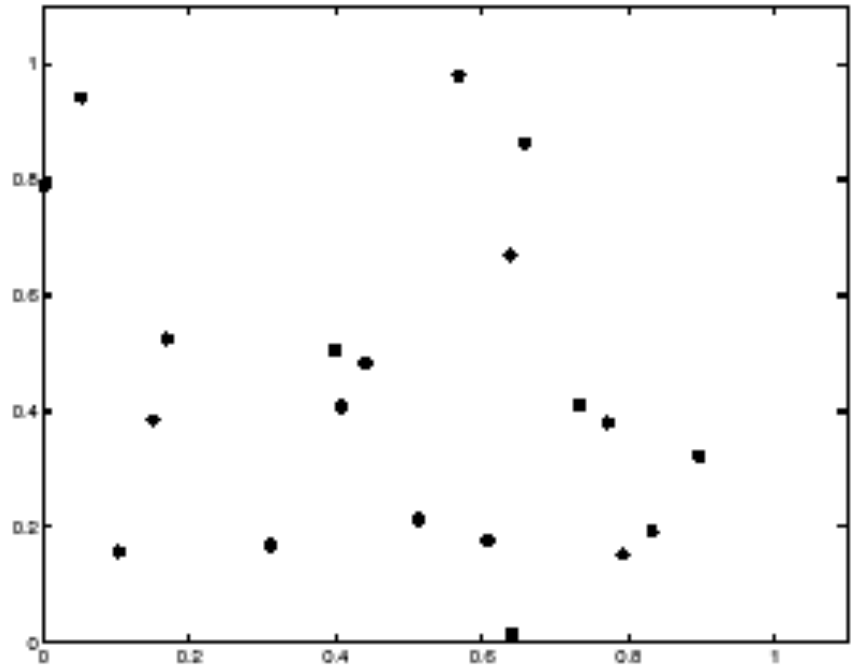


features votes

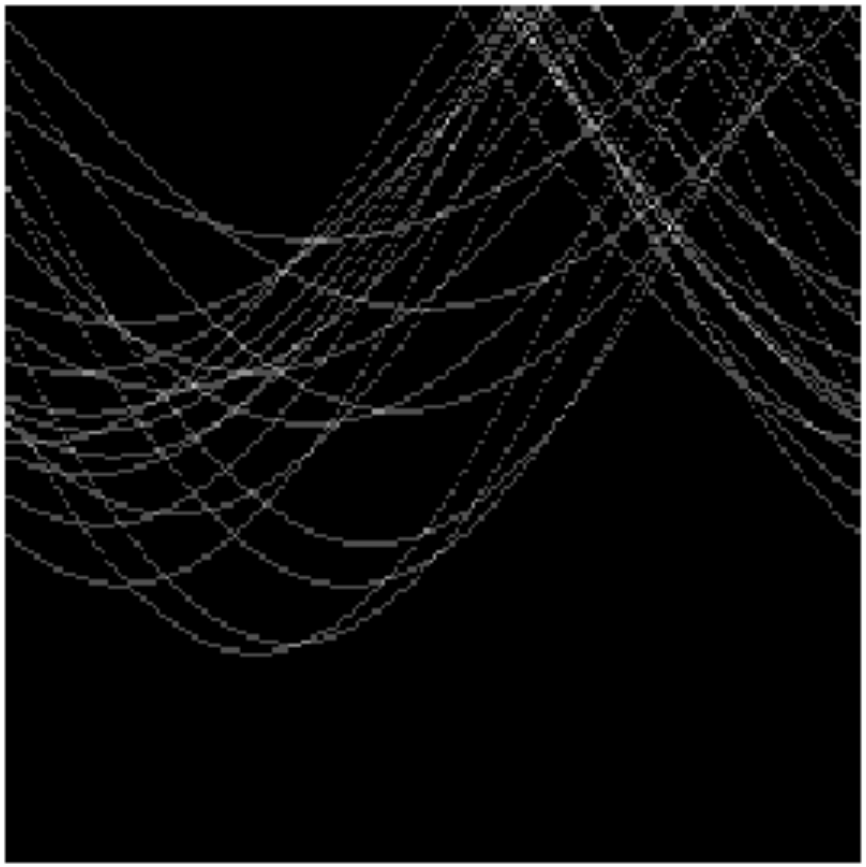
Need to adjust grid size or smooth

Slide from S. Savarese

Hough transform - experiments



features



votes

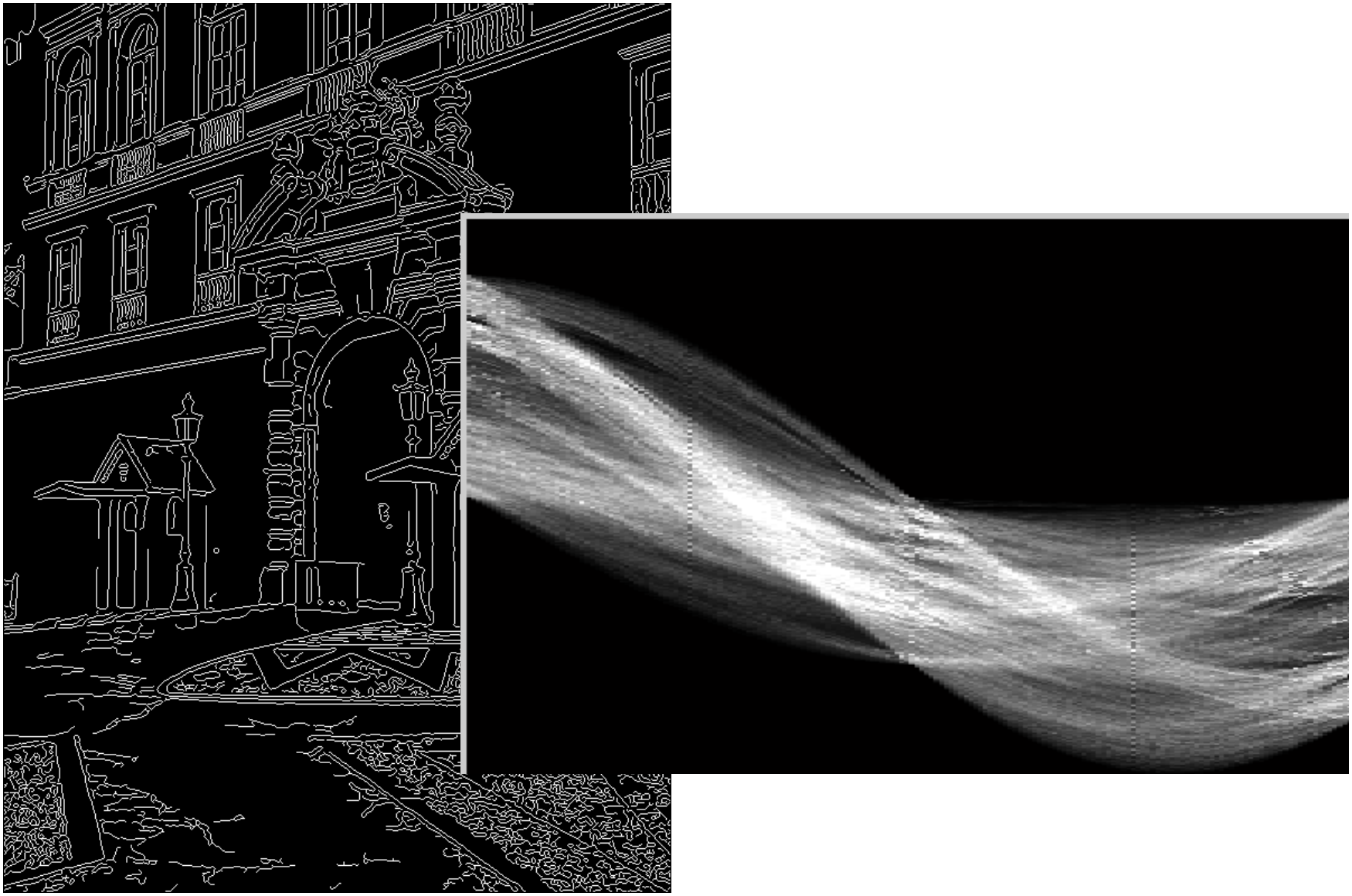
Issue: spurious peaks due to uniform noise

Slide from S. Savarese

1. Image → Canny

Slide from D. Hoiem

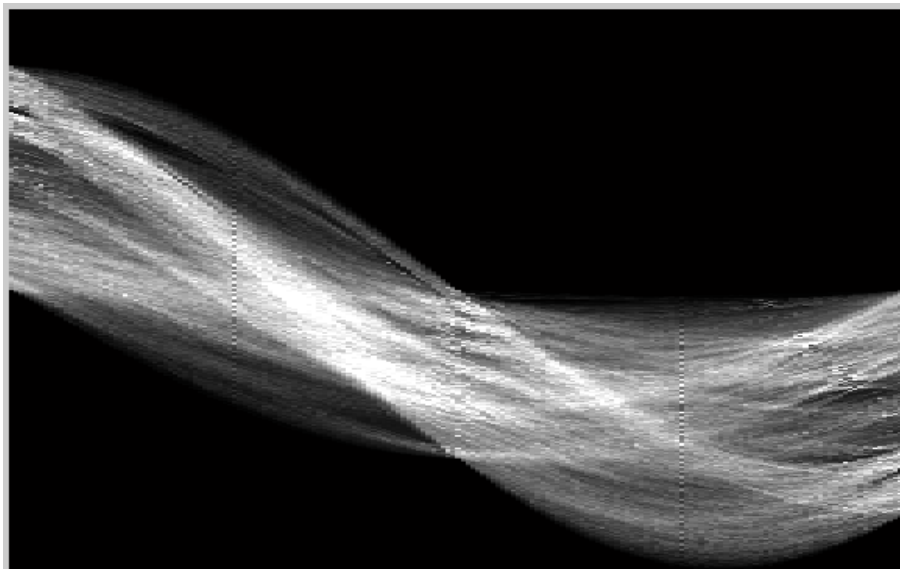
2. Canny → Hough votes



Slide from D. Hoiem

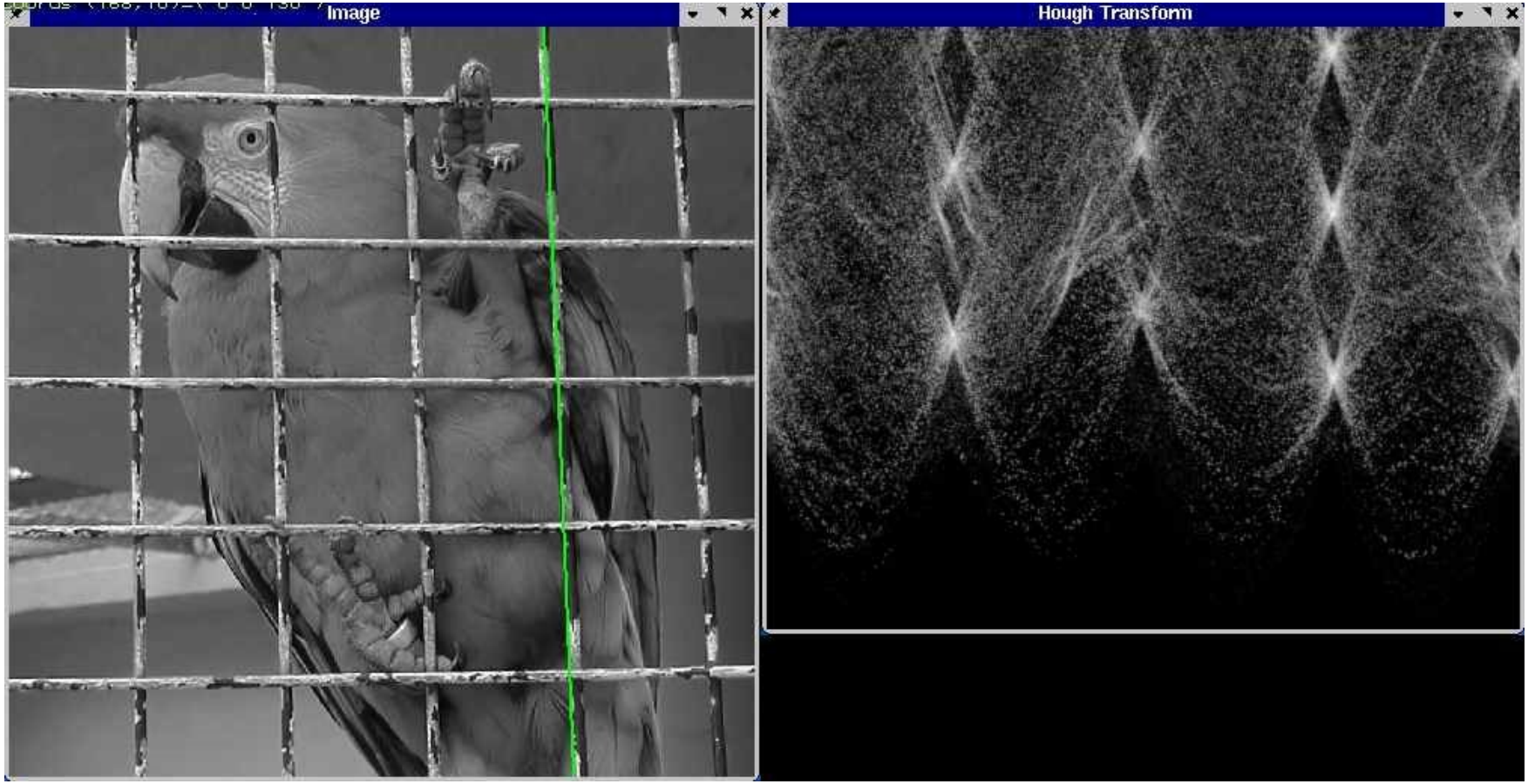
3. Hough votes → Edges

Find peaks and post-process



Slide from D. Hoiem

Hough transform example



Slide from D. Hoiem

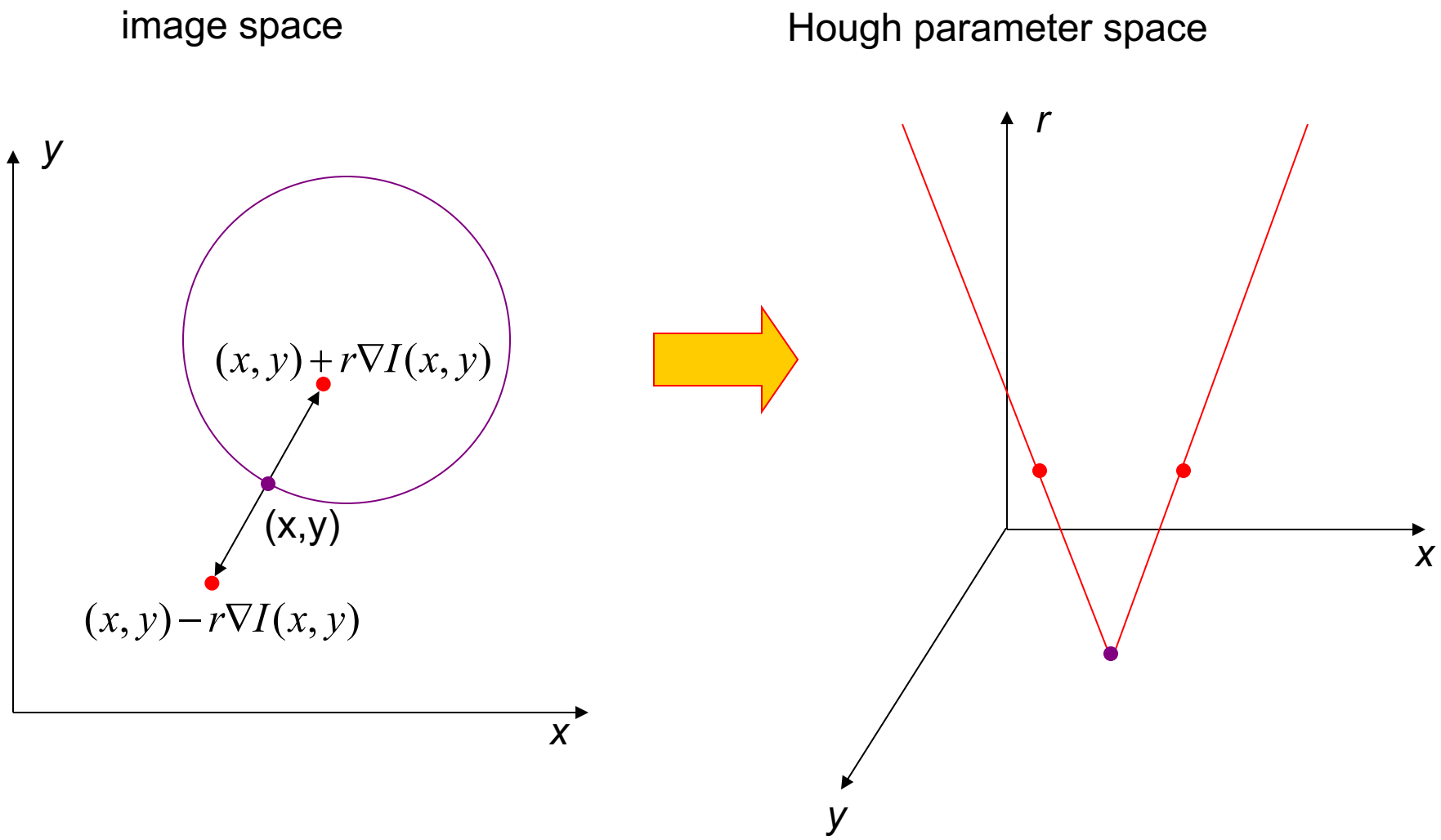
http://ostatic.com/files/images/ss_hough.jpg

Finding circles (x_0, y_0, r) using Hough transform

- Fixed r
- Variable r

Slide from D. Hoiem

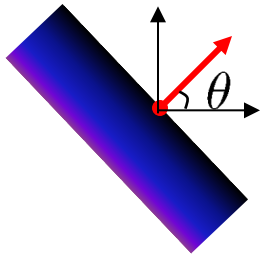
Hough transform for circles



Slide from L. Lazebnik

Incorporating image gradients

- When we detect an edge point, we also know its gradient orientation
- How does this constrain possible lines passing through the point?



$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$$

- Modified Hough transform:

- For each edge point (x,y)
  θ = gradient orientation at (x,y)
  ρ = x cos θ + y sin θ
  H(θ, ρ) = H(θ, ρ) + 1

end

Slide from L. Lazebnik

Hough transform conclusions

Good

- Robust to outliers: each point votes separately
- Fairly efficient (much faster than trying all sets of parameters)
- Provides multiple good fits

Bad

- Some sensitivity to noise
- Bin size trades off between noise tolerance, precision, and speed/memory
  - Can be hard to find sweet spot
- Not suitable for more than a few parameters
  - grid size grows exponentially

Common applications

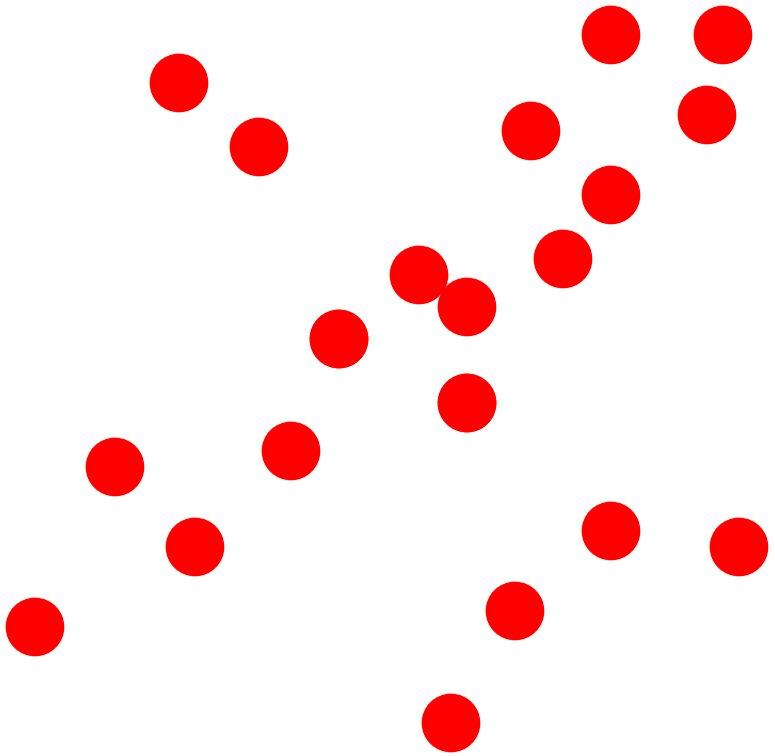
- Line fitting (also circles, ellipses, etc.)
- Object instance recognition (parameters are position/scale/orientation)
- Object category recognition (parameters are position/scale)

Slide from D. Hoiem

RANSAC

(RANdom SAmple Consensus) :

Fischler & Bolles in ‘81.



Algorithm:

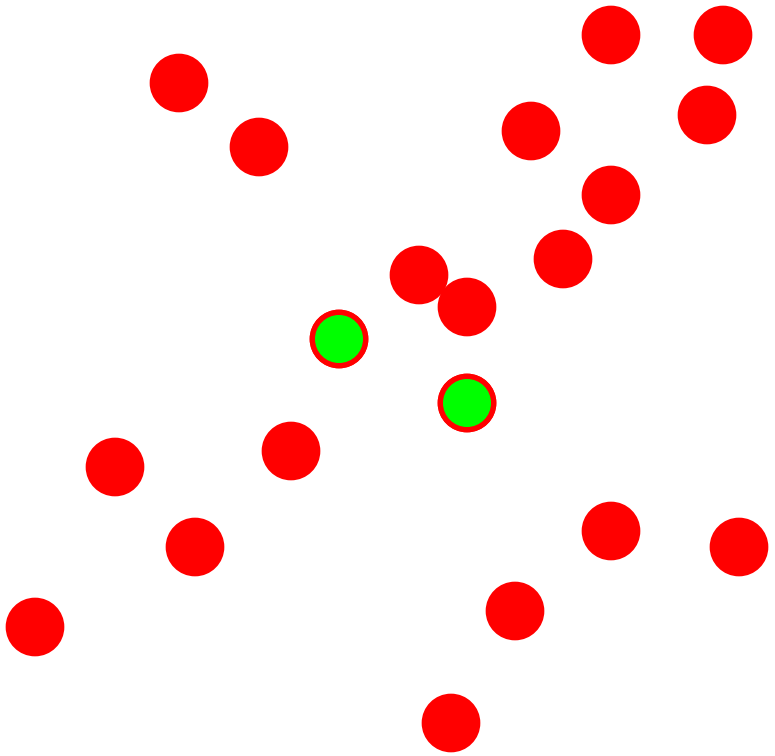
1. **Sample** (randomly) the number of points required to fit the model
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

Slide from D. Hoiem

RANSAC

Line fitting example



Algorithm:

1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

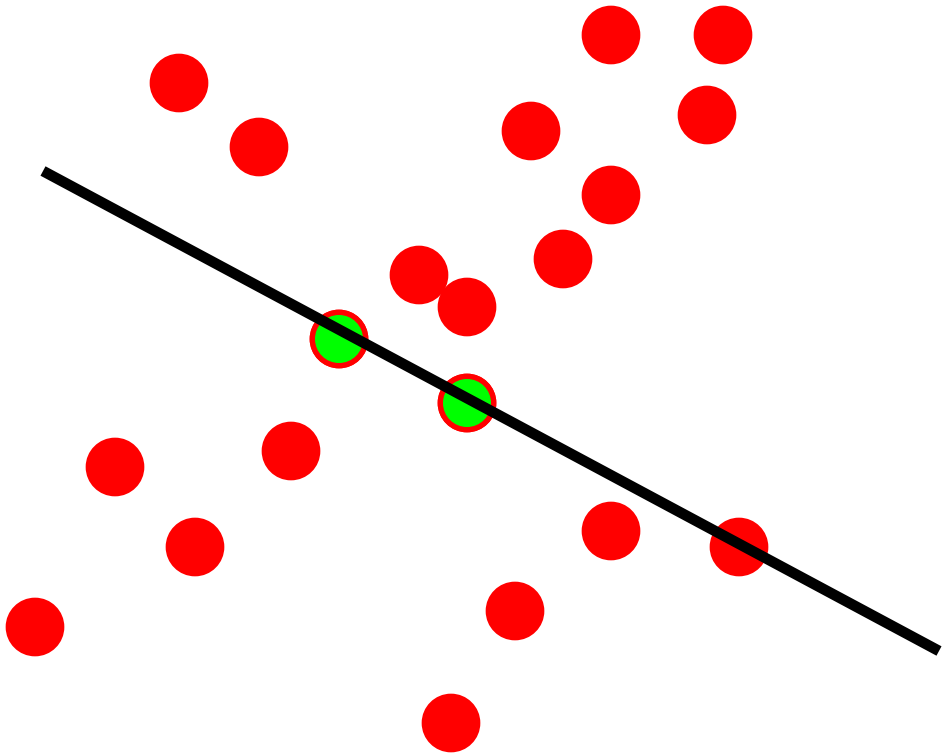
Repeat 1-3 until the best model is found with high confidence

Slide from D. Hoiem

Illustration by Savarese

RANSAC

Line fitting example



Algorithm:

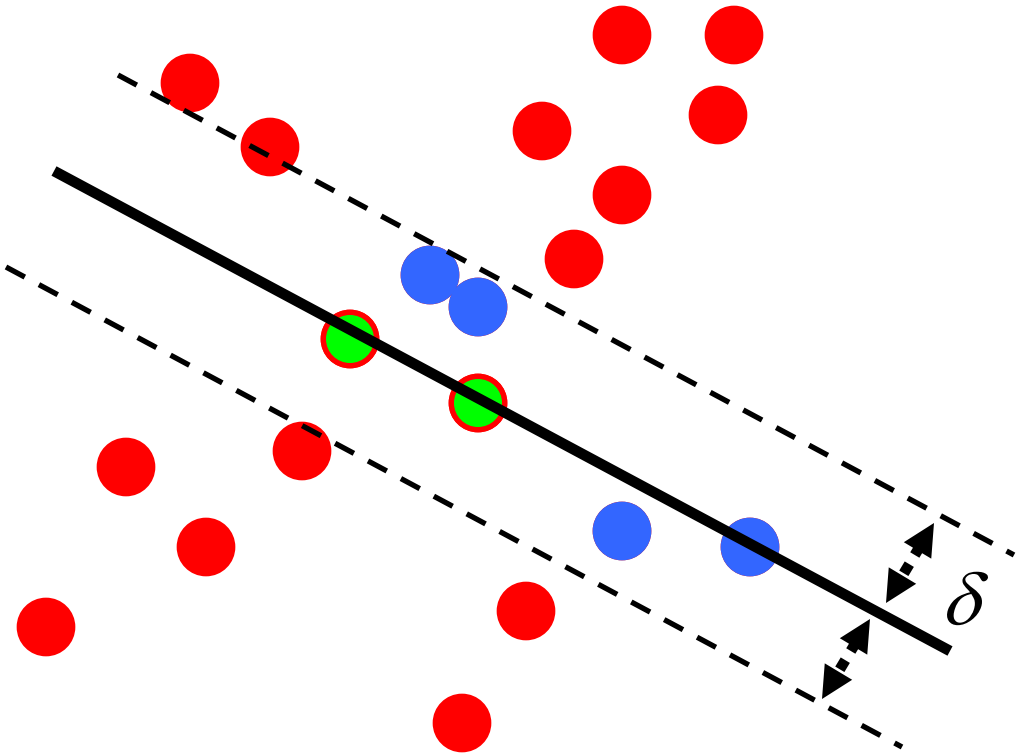
1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

Slide from D. Hoiem

RANSAC

Line fitting example



$$N_I = 6$$

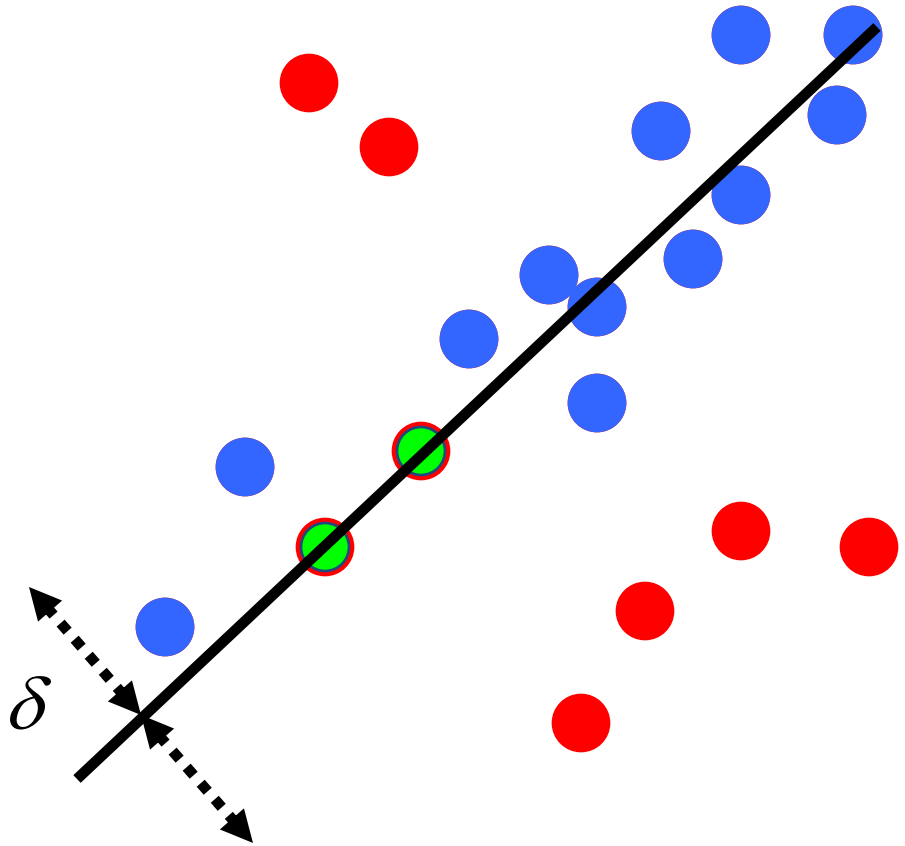
Algorithm:

1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

Slide from D. Hoiem

RANSAC



$N_I = 14$

Algorithm:

1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

Slide from D. Hoiem

How to choose parameters?

- Number of samples *N*
  - Choose *N* so that, with probability *p*, at least one random sample is free from outliers (e.g. *p*=0.99) (outlier ratio: *e*)
- Number of sampled points *s*
  - Minimum number needed to fit the model
- Distance threshold δ
  - Choose δ so that a good point with noise is likely (e.g., prob=0.95) within threshold
  - Zero-mean Gaussian noise with std. dev. σ: $t^2=3.84\sigma^2$

$$N = \log(1-p)/\log\left(1-(1-e)^s\right)$$

	proportion of outliers *e*						
s	5%	10%	20%	25%	30%	40%	50%
2	2	3	5	6	7	11	17
3	3	4	7	9	11	19	35
4	3	5	9	13	17	34	72
5	4	6	12	17	26	57	146
6	4	7	16	24	37	97	293
7	4	8	20	33	54	163	588
8	5	9	26	44	78	272	1177

Slide from D. Hoiem

modified from M. Pollefeys

RANSAC conclusions

Good

- Robust to outliers
- Applicable for larger number of objective function parameters than Hough transform
- Optimization parameters are easier to choose than Hough transform

Bad

- Computational time grows quickly with fraction of outliers and number of parameters
- Not as good for getting multiple fits (though one solution is to remove inliers after each fit and repeat)

Common applications

- Computing a homography (e.g., image stitching)
- Estimating fundamental matrix (relating two views)

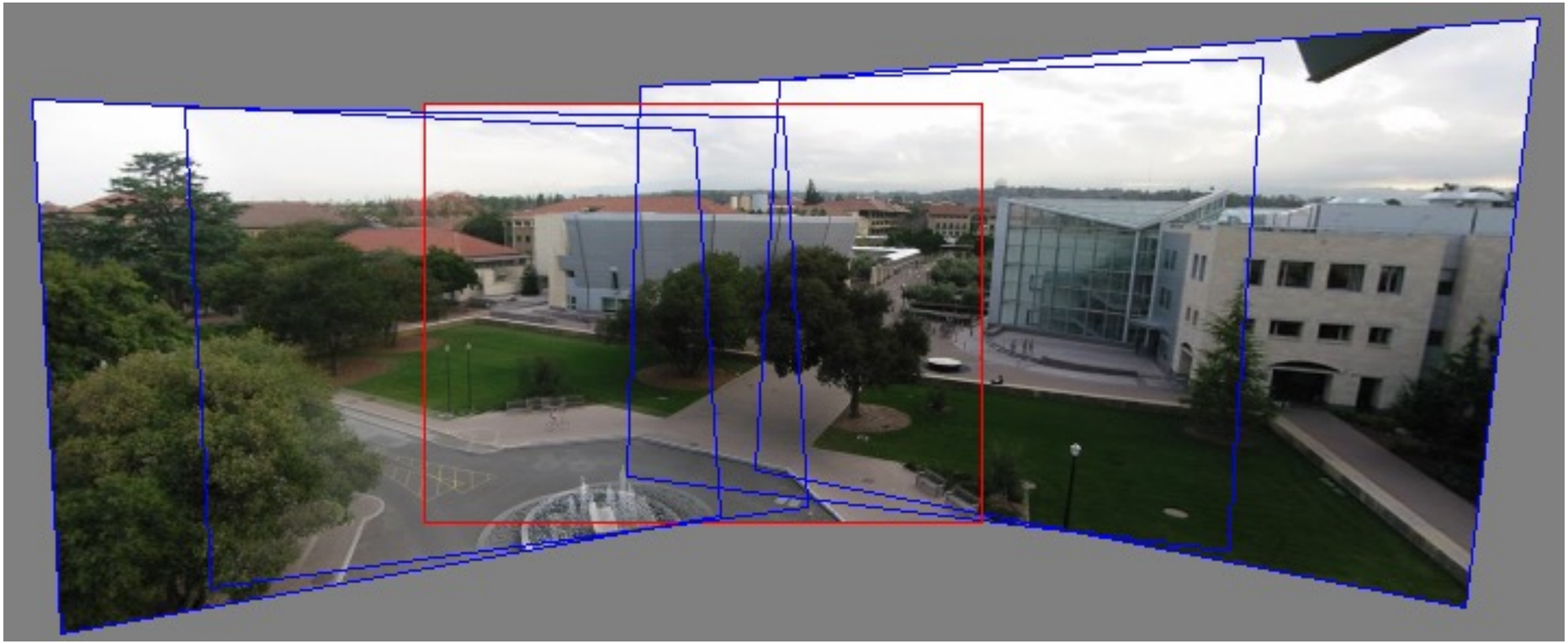
Slide from D. Hoiem

Fitting Summary

- Least Squares Fit
  - closed form solution
  - robust to noise
  - not robust to outliers
- Robust Least Squares
  - improves robustness to noise
  - requires iterative optimization
- Hough transform
  - robust to noise and outliers
  - can fit multiple models
  - only works for a few parameters (1-4 typically)
- RANSAC
  - robust to noise and outliers
  - works with a moderate number of parameters (e.g, 1-8)

Slide from D. Hoiem

Image alignment



Slides from Derek Hoiem, Svetlana Lazebnik

Image source

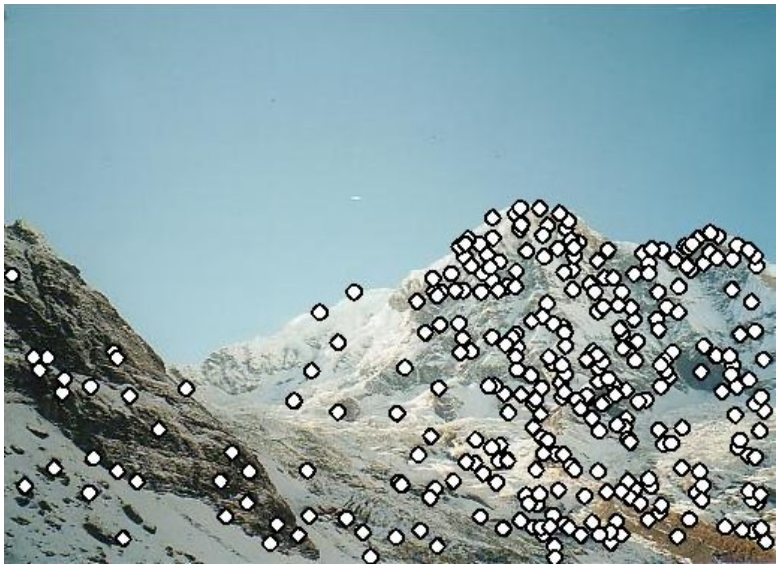
Image Alignment

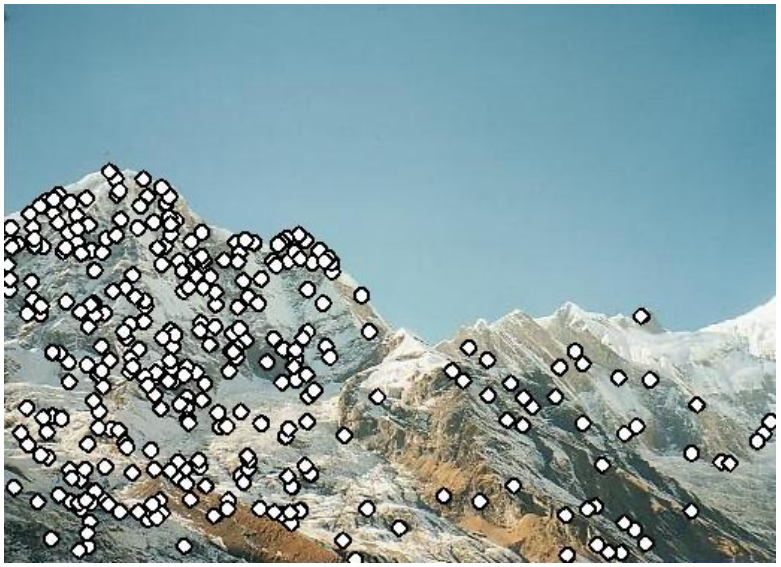
Slide from L. Lazebnik.

Robust feature-based alignment

Slide from L. Lazebnik.

Robust feature-based alignment

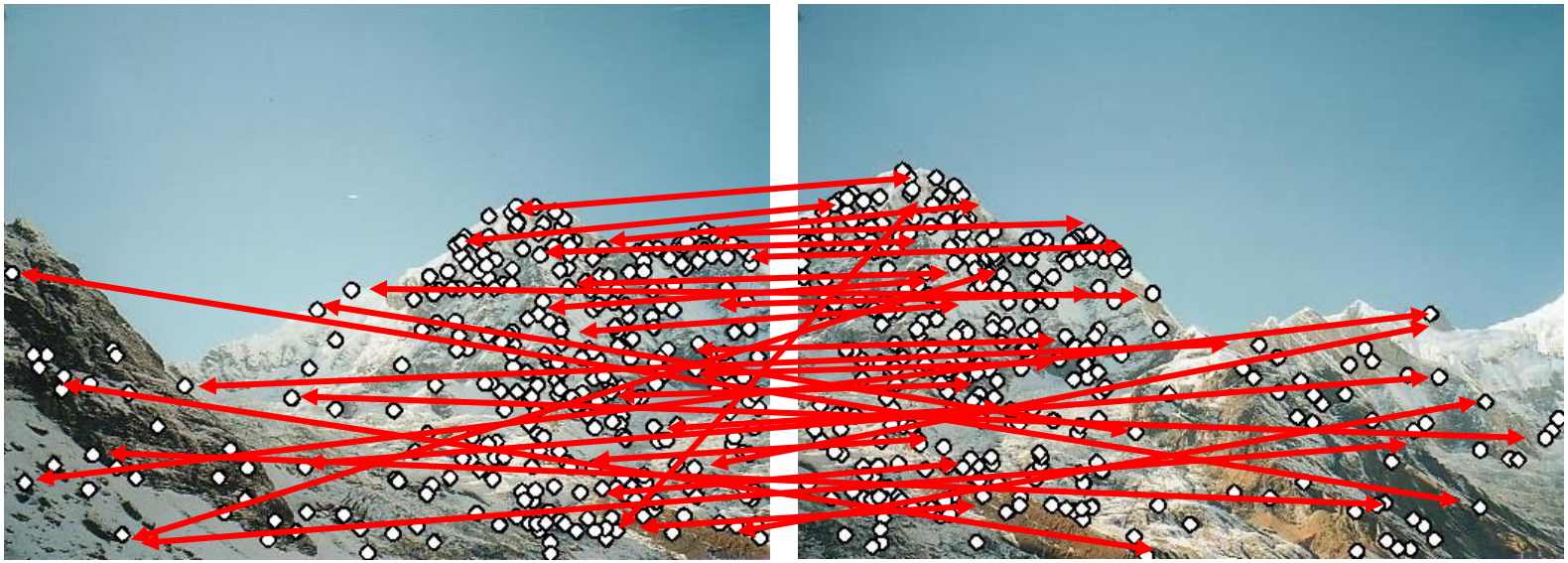




- Extract features

Slide from L. Lazebnik.

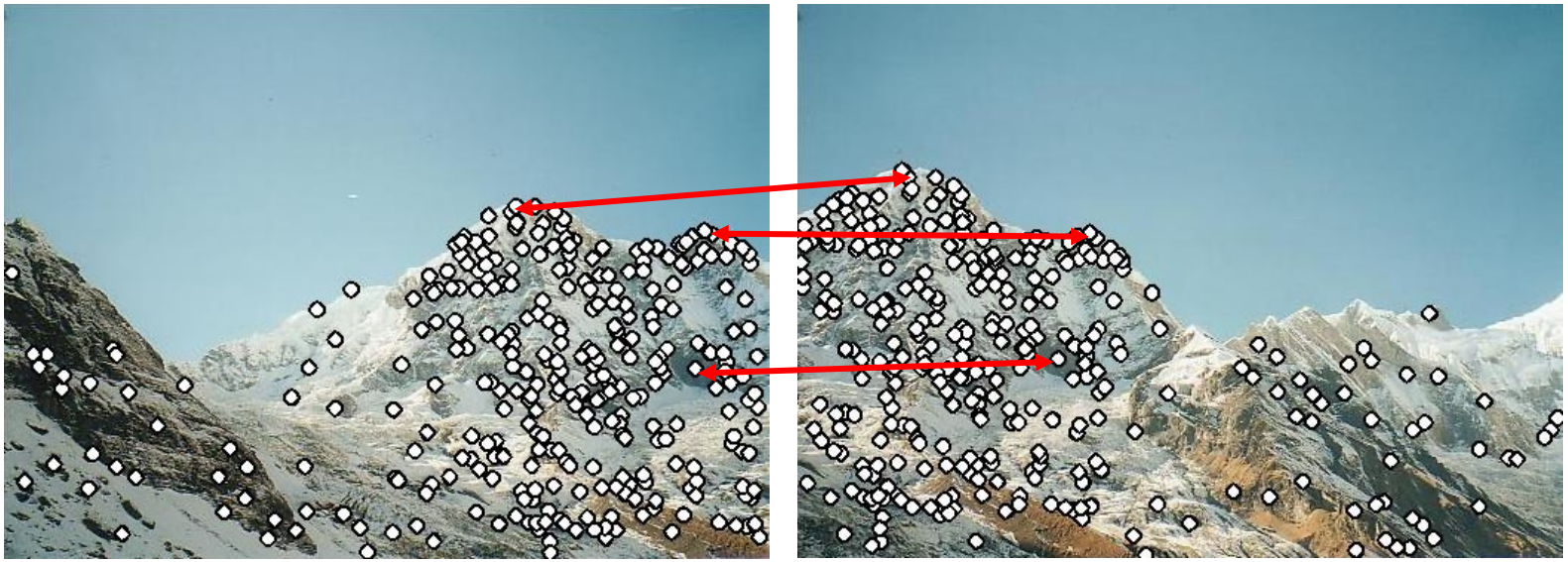
Robust feature-based alignment



- Extract features
- Compute *putative matches*

Slide from L. Lazebnik.

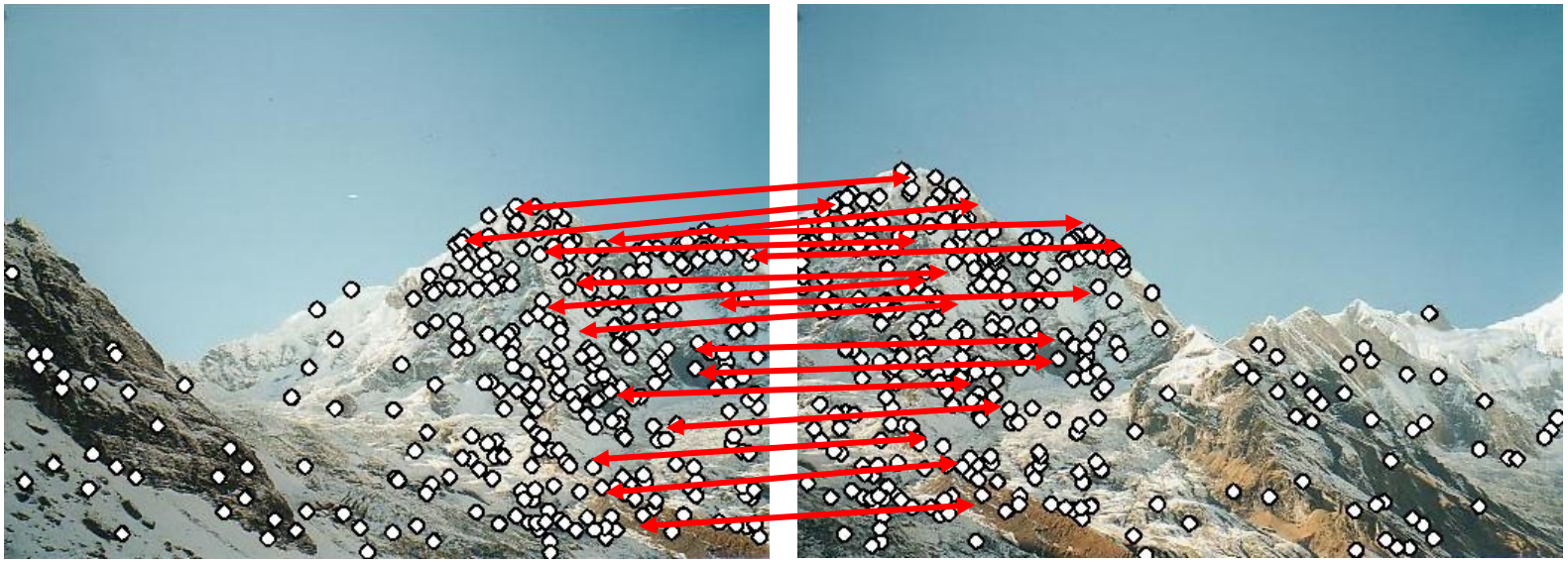
Robust feature-based alignment



- Extract features
- Compute *putative matches*
- Loop:
  - *Hypothesize* transformation *T*

Slide from L. Lazebnik.

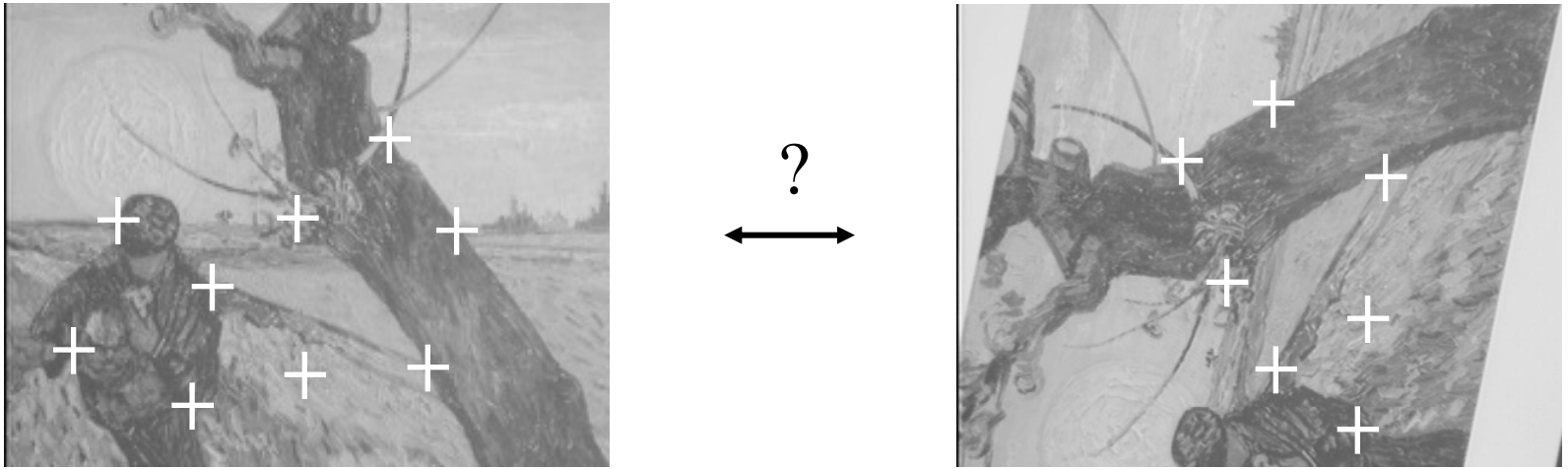
Robust feature-based alignment



- Extract features
- Compute *putative matches*
- Loop:
  - *Hypothesize* transformation *T*
  - *Verify* transformation (search for other matches consistent with *T*)

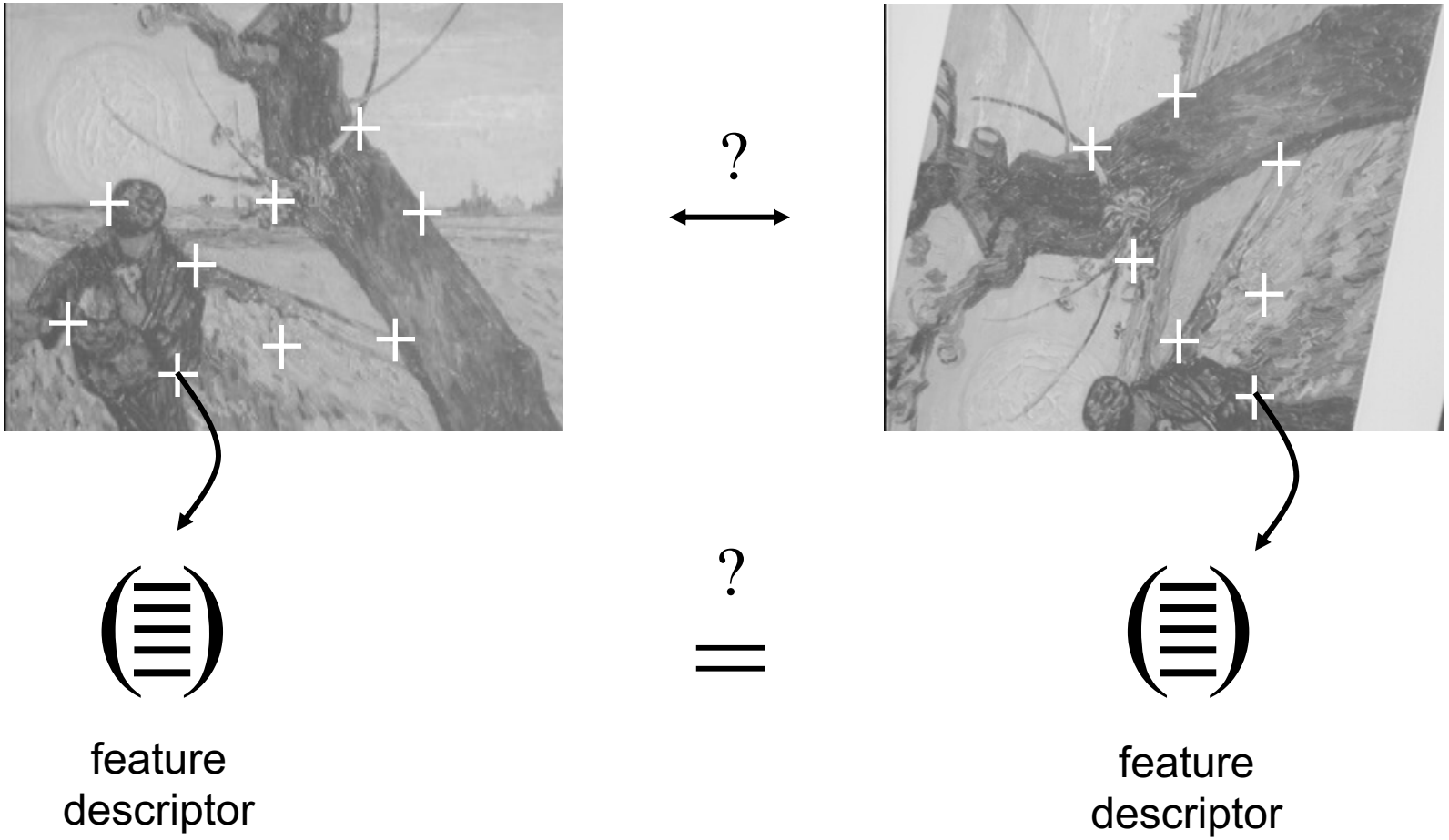
Slide from L. Lazebnik.

Generating putative correspondences



Slide from L. Lazebnik.

Generating putative correspondences

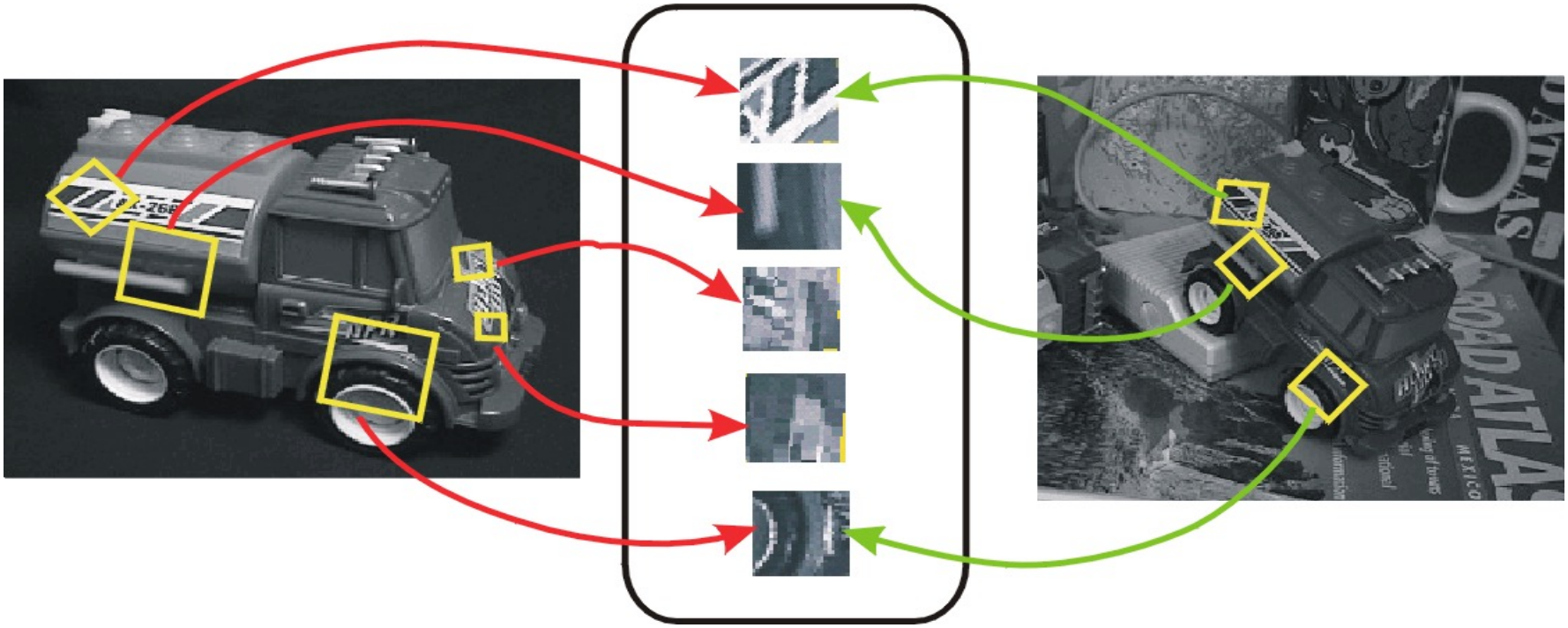


- Need to compare *feature descriptors* of local patches surrounding interest points

Slide from L. Lazebnik.

Feature descriptors

- Recall: feature detection and description



Slide from L. Lazebnik.

Feature descriptors

- Simplest descriptor: vector of raw intensity values
- How to compare two such vectors?
  - Sum of squared differences (SSD)

$$\mathrm{SSD}(\mathbf{u},\mathbf{v}) = \sum_i \left(u_i - v_i\right)^2$$

    - Not invariant to intensity change

  - Normalized correlation

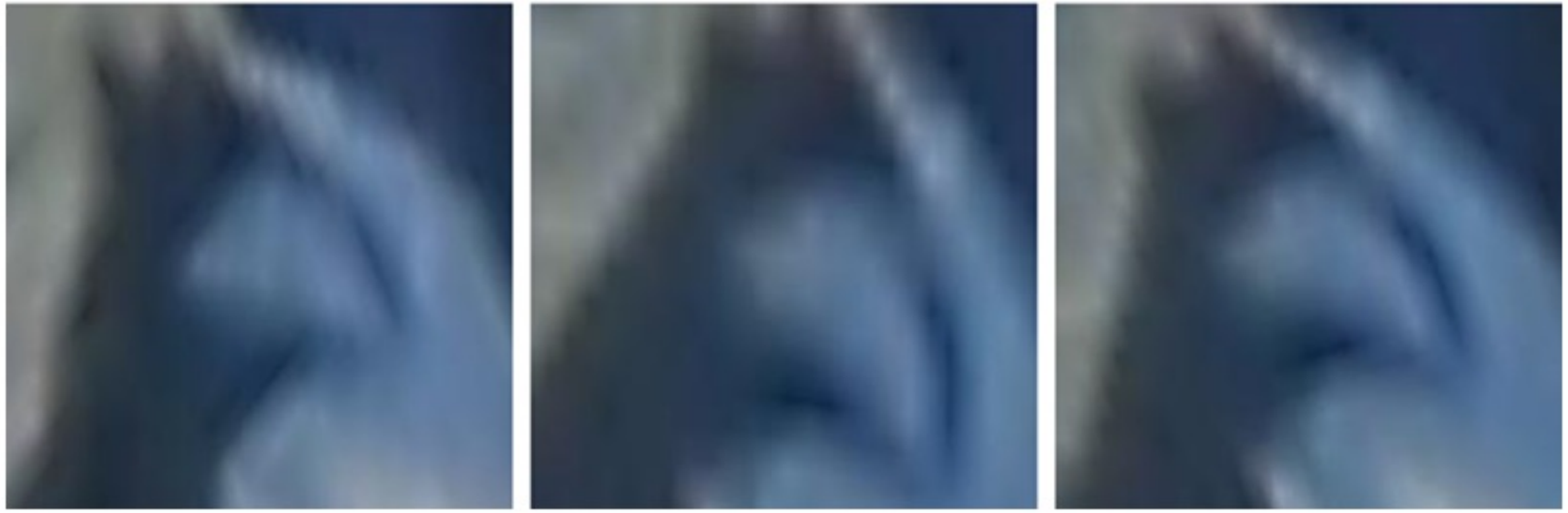
$$\rho(\mathbf{u},\mathbf{v}) = \frac{(\mathbf{u}-\bar{\mathbf{u}})}{\|\mathbf{u}-\bar{\mathbf{u}}\|} \cdot \frac{(\mathbf{v}-\bar{\mathbf{v}})}{\|\mathbf{v}-\bar{\mathbf{v}}\|} = \frac{\sum_i (u_i - \bar{\mathbf{u}})(v_i - \bar{\mathbf{v}})}{\sqrt{\left(\sum_j (u_j - \bar{\mathbf{u}})^2\right)\left(\sum_j (v_j - \bar{\mathbf{v}})^2\right)}}$$

    - Invariant to affine intensity change

Slide from L. Lazebnik.

Disadvantage of intensity vectors as descriptors

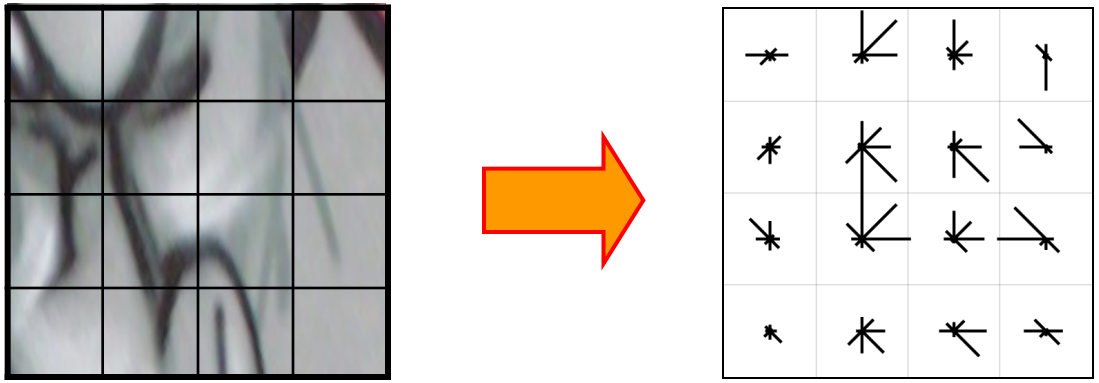
- Small deformations can affect the matching score a lot



Slide from L. Lazebnik.

Feature descriptors: SIFT

- Descriptor computation:
  - Divide patch into 4x4 sub-patches
  - Compute histogram of gradient orientations (8 reference angles) inside each sub-patch
  - Resulting descriptor: 4x4x8 = 128 dimensions



David G. Lowe. "Distinctive image features from scale-invariant keypoints." *IJCV* 60 (2), pp. 91-110, 2004.

Slide from L. Lazebnik.

Feature descriptors: SIFT

- Descriptor computation:
  - Divide patch into 4x4 sub-patches
  - Compute histogram of gradient orientations (8 reference angles) inside each sub-patch
  - Resulting descriptor: 4x4x8 = 128 dimensions

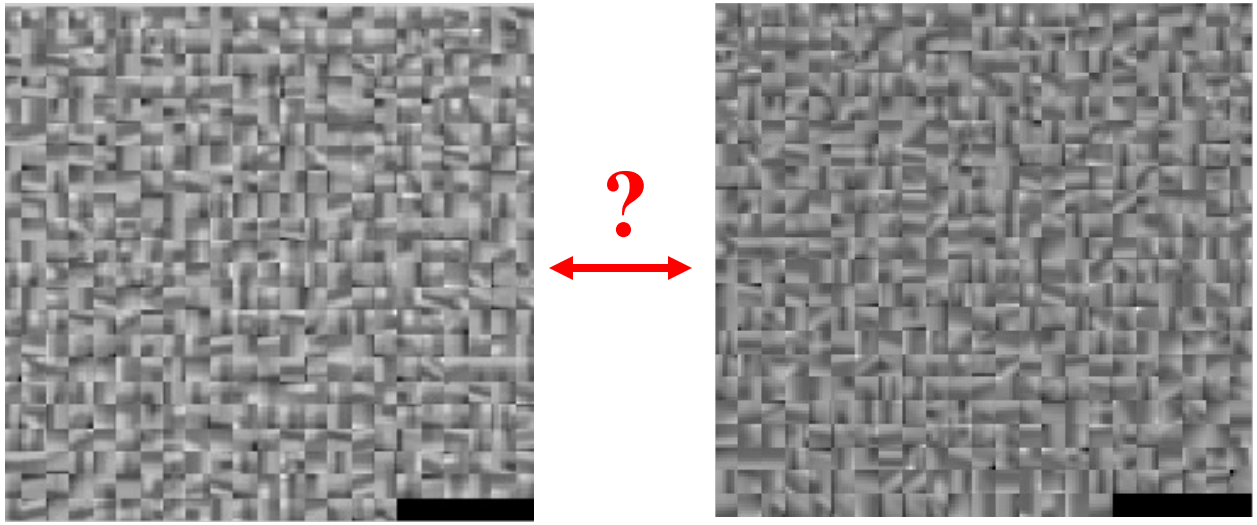
- Advantage over raw vectors of pixel values
  - Gradients less sensitive to illumination change
  - Pooling of gradients over the sub-patches achieves robustness to small shifts, but still preserves some spatial information

David G. Lowe. "Distinctive image features from scale-invariant keypoints." *IJCV* 60 (2), pp. 91-110, 2004.

Slide from L. Lazebnik.

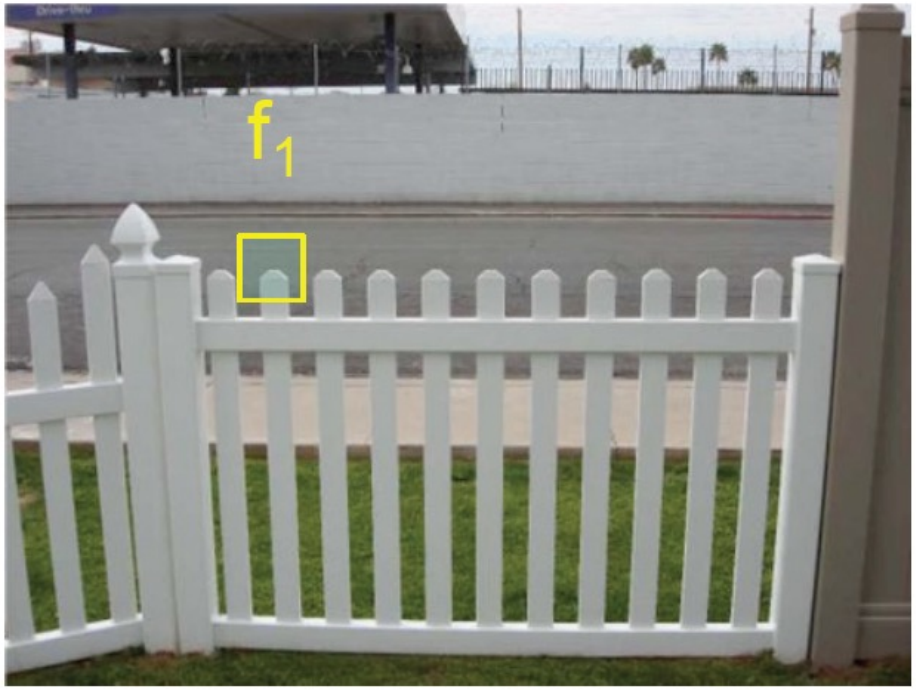
Feature matching

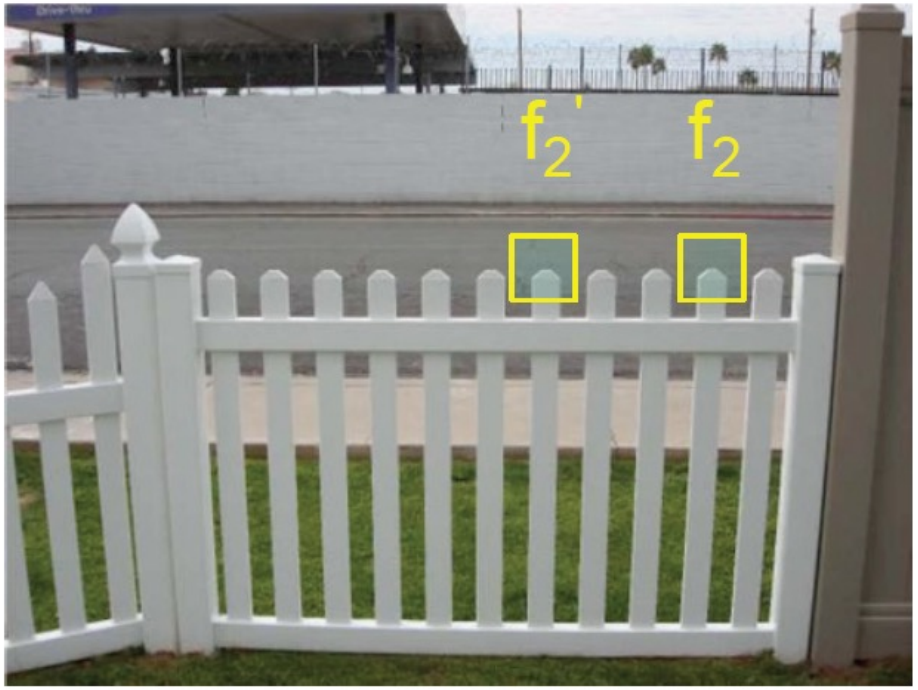
- Generating *putative matches*: for each patch in one image, find a short list of patches in the other image that could match it based solely on appearance



Slide from L. Lazebnik.

Problem: Ambiguous putative matches

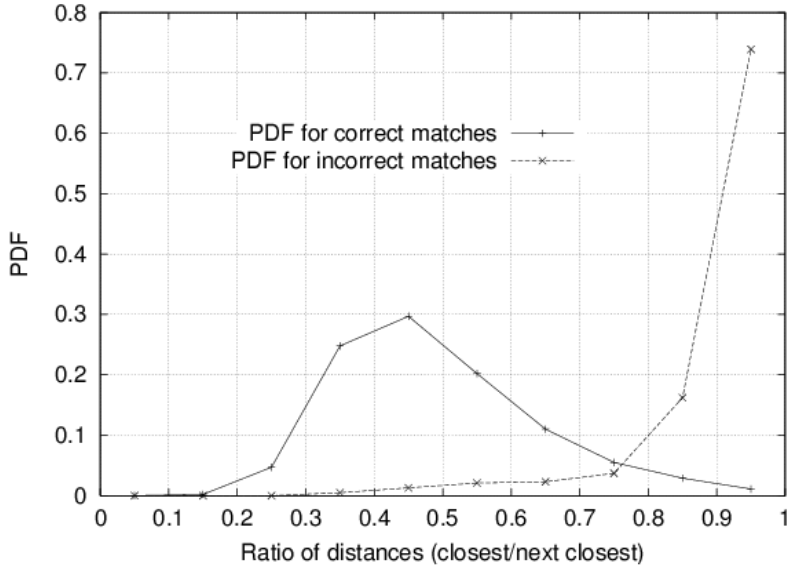




Source: Y. Furukawa

Rejection of unreliable matches

- How can we tell which putative matches are more reliable?
- Heuristic: compare distance of **nearest** neighbor to that of **second** nearest neighbor
  - Ratio of closest distance to second-closest distance will be *high* for features that are *not* distinctive



Threshold of 0.8 provides good separation

David G. Lowe. "Distinctive image features from scale-invariant keypoints." *IJCV* 60 (2), pp. 91-110, 2004.

Slide from L. Lazebnik.

RANSAC

- The set of putative matches contains a very high percentage of outliers

RANSAC loop:

1. Randomly select a *seed group* of matches
2. Compute transformation from seed group
3. Find *inliers* to this transformation
4. If the number of inliers is sufficiently large, re-compute least-squares estimate of transformation on all of the inliers

Keep the transformation with the largest number of inliers

Slide from L. Lazebnik.

Robust feature-based alignment

- Extract features
- Compute *putative matches*
- Loop:
  - *Hypothesize* transformation *T*
  - *Verify* transformation (search for other matches consistent with *T*)

Slide from L. Lazebnik.

Common transformations

original

Transformed

translation

rotation

aspect

affine

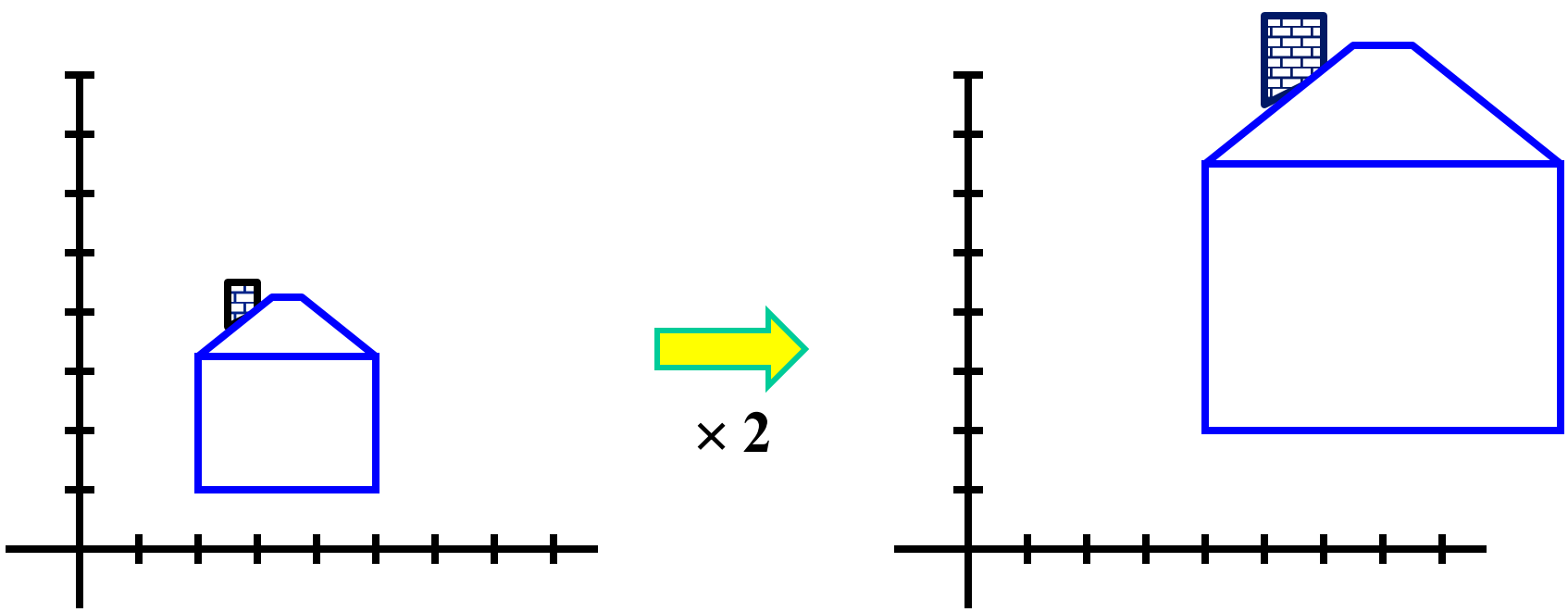
perspective

Slide from A. Efros, S. Seitz, D. Hoiem

Scaling

Scaling a coordinate means multiplying each of its components by a scalar

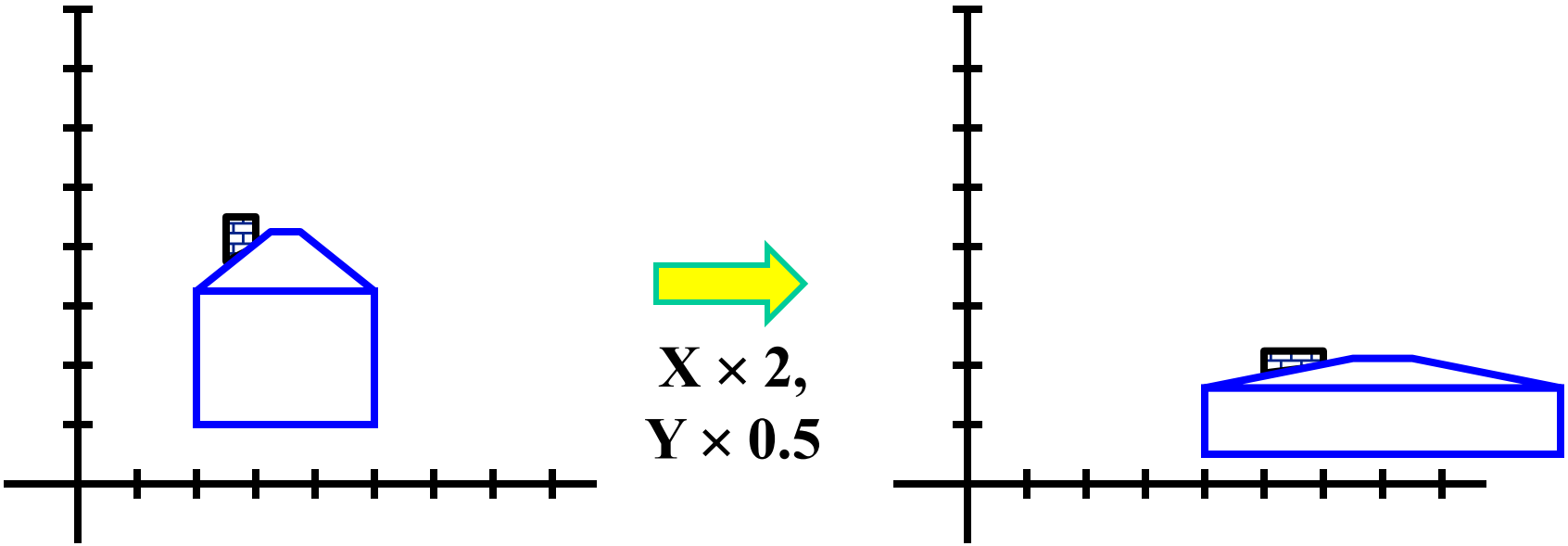
Uniform scaling means this scalar is the same for all components:



Slide from A. Efros, S. Seitz, D. Hoiem

Scaling

Non-uniform scaling: different scalars per component:



Slide from A. Efros, S. Seitz, D. Hoiem

Scaling

Scaling operation:

$$x' = ax$$

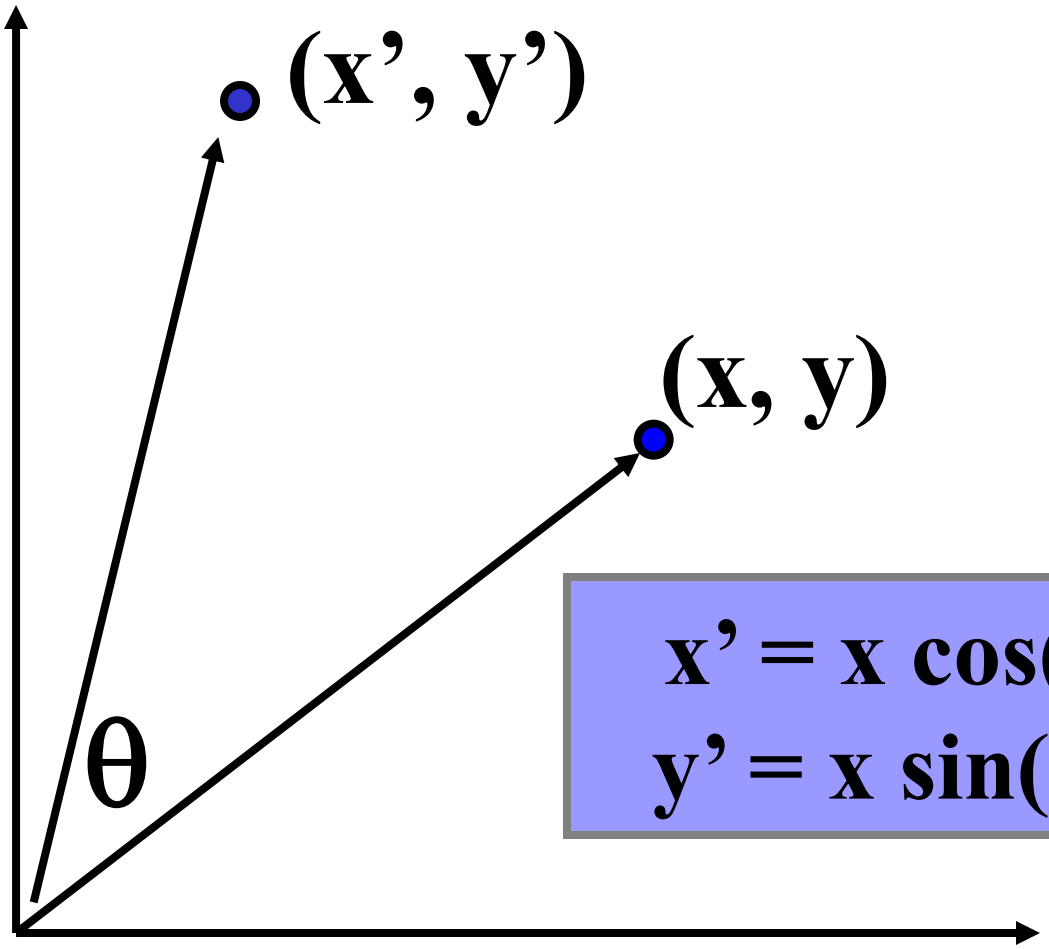
$$y' = by$$

Or, in matrix form:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \underbrace{\begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}}_{\textbf{scaling matrix S}} \begin{bmatrix} x \\ y \end{bmatrix}$$

Slide from A. Efros, S. Seitz, D. Hoiem

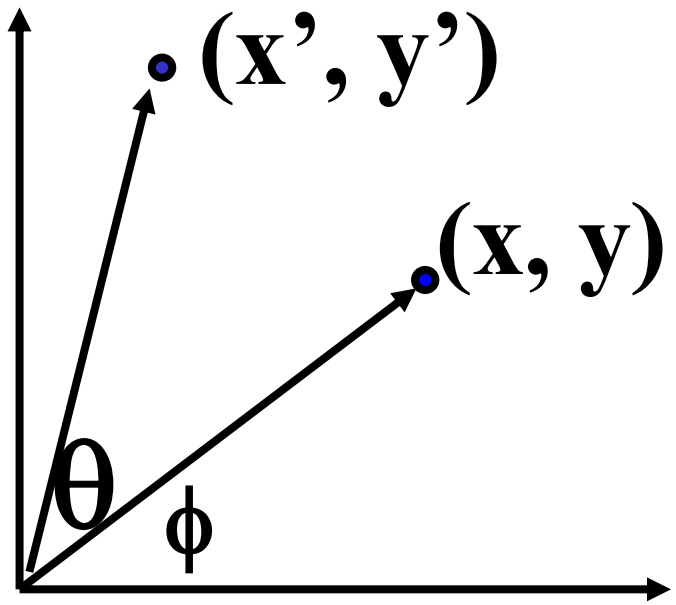
2-D Rotation



$$x' = x \cos(\theta) - y \sin(\theta)$$
$$y' = x \sin(\theta) + y \cos(\theta)$$

Slide from A. Efros, S. Seitz, D. Hoiem

2-D Rotation



Polar coordinates…

$x = r \cos(\phi)$

$y = r \sin(\phi)$

$x' = r \cos(\phi + \theta)$

$y' = r \sin(\phi + \theta)$

Trig Identity…

$x' = r \cos(\phi) \cos(\theta) - r \sin(\phi) \sin(\theta)$

$y' = r \sin(\phi) \cos(\theta) + r \cos(\phi) \sin(\theta)$

Substitute…

$x' = x \cos(\theta) - y \sin(\theta)$

$y' = x \sin(\theta) + y \cos(\theta)$

Slide from A. Efros, S. Seitz, D. Hoiem

2-D Rotation

This is easy to capture in matrix form:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \underbrace{\begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}}_{\mathbf{R}} \begin{bmatrix} x \\ y \end{bmatrix}$$

Even though sin(θ) and cos(θ) are nonlinear functions of θ,

- ***x' is a linear combination of x and y***
- ***y' is a linear combination of x and y***

What is the inverse transformation?

- Rotation by –θ
- For rotation matrices $\mathbf{R}^{-1} = \mathbf{R}^{T}$

Slide from A. Efros, S. Seitz, D. Hoiem

Basic 2D transformations

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Scale

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & \alpha_x \\ \alpha_y & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Shear

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\Theta & -\sin\Theta \\ \sin\Theta & \cos\Theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Rotate

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Translate

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Affine

Affine is any combination of translation, scale, rotation, shear

Slide from A. Efros, S. Seitz, D. Hoiem

Affine Transformations

Affine transformations are combinations of

- Linear transformations, and
- Translations

Properties of affine transformations:

- Lines map to lines
- Parallel lines remain parallel
- Ratios are preserved
- Closed under composition

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

or

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Slide from L. Lazebnik.

Projective Transformations

Projective transformations are combos of

- Affine transformations, and
- Projective warps

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

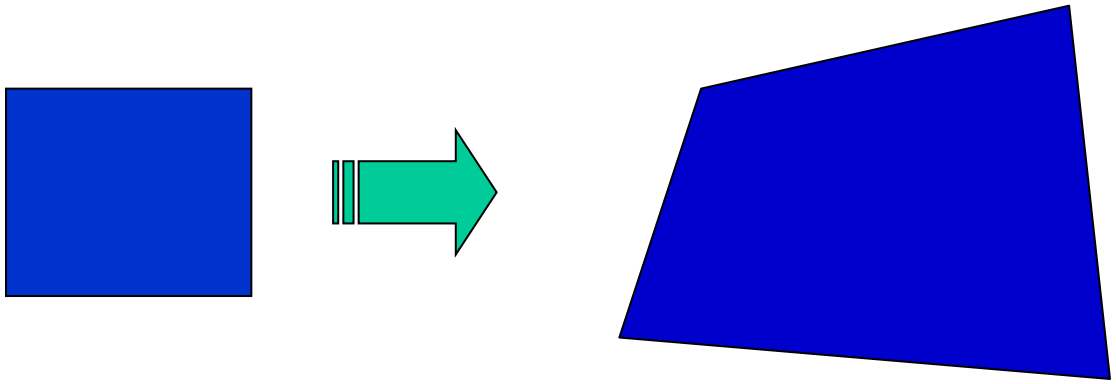
Properties of projective transformations:

- Lines map to lines
- Parallel lines do not necessarily remain parallel
- Ratios are not preserved
- Closed under composition
- **Projective matrix is defined up to a scale (8 DOF)**

Slide from L. Lazebnik.

Fitting a plane projective transformation

- **Homography:** plane projective transformation (transformation taking a quad to another arbitrary quad)

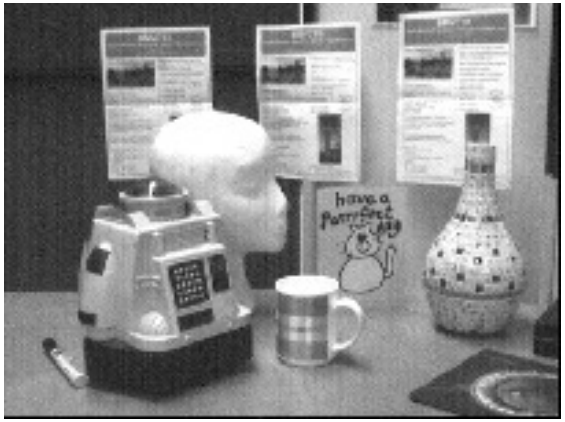


Slide from L. Lazebnik.

Homography

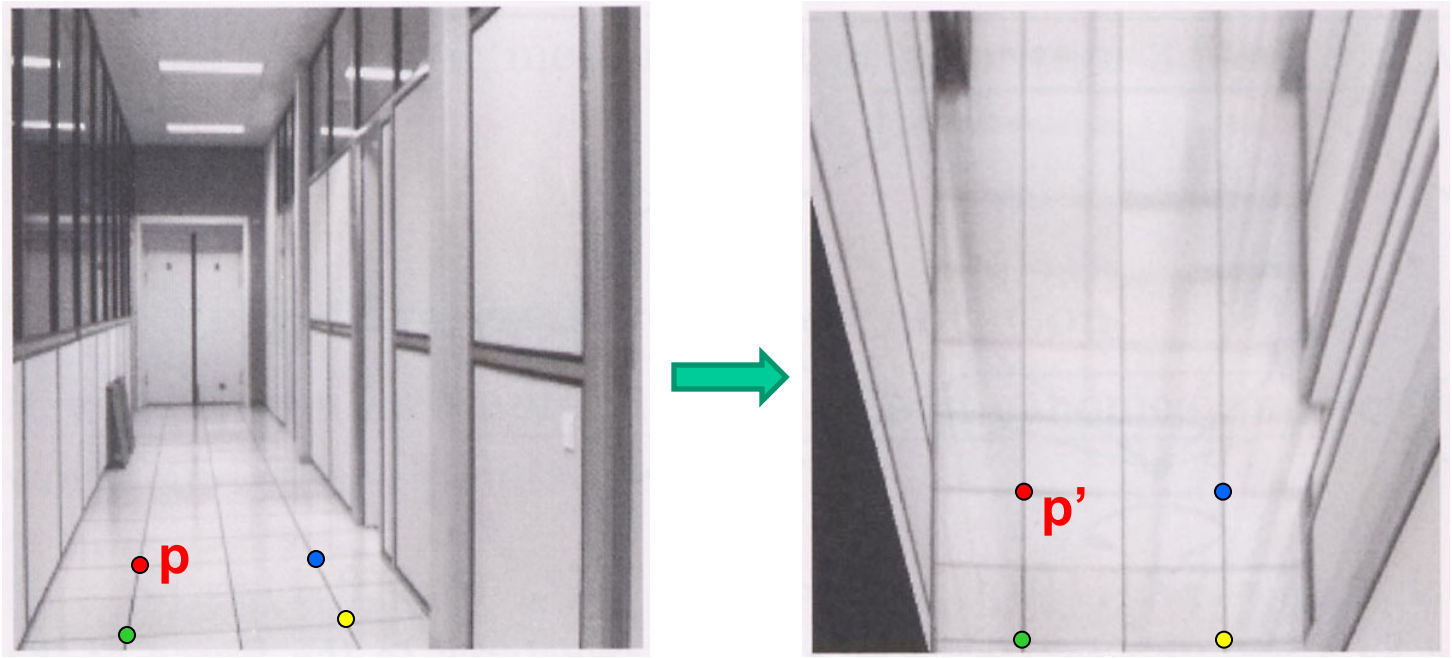
- The transformation between two views of a planar surface


- The transformation between images from two cameras that share the same center

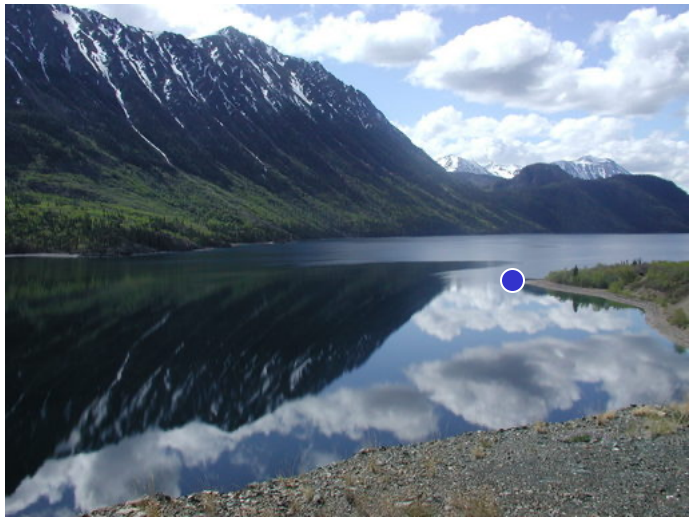
Slide from L. Lazebnik.

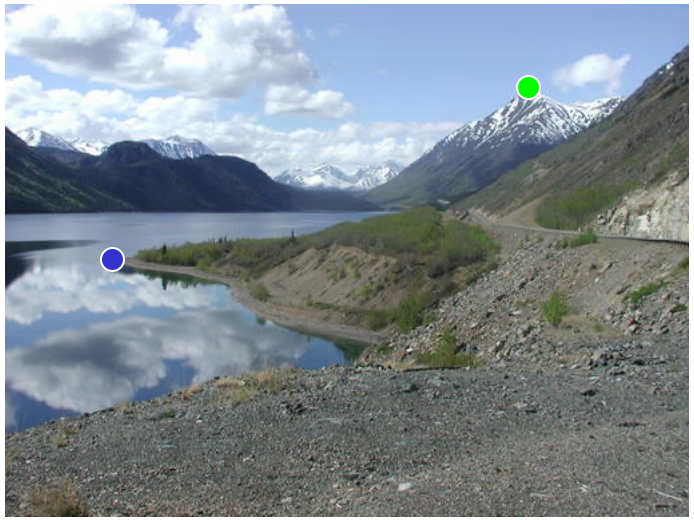
Homography example: Image rectification

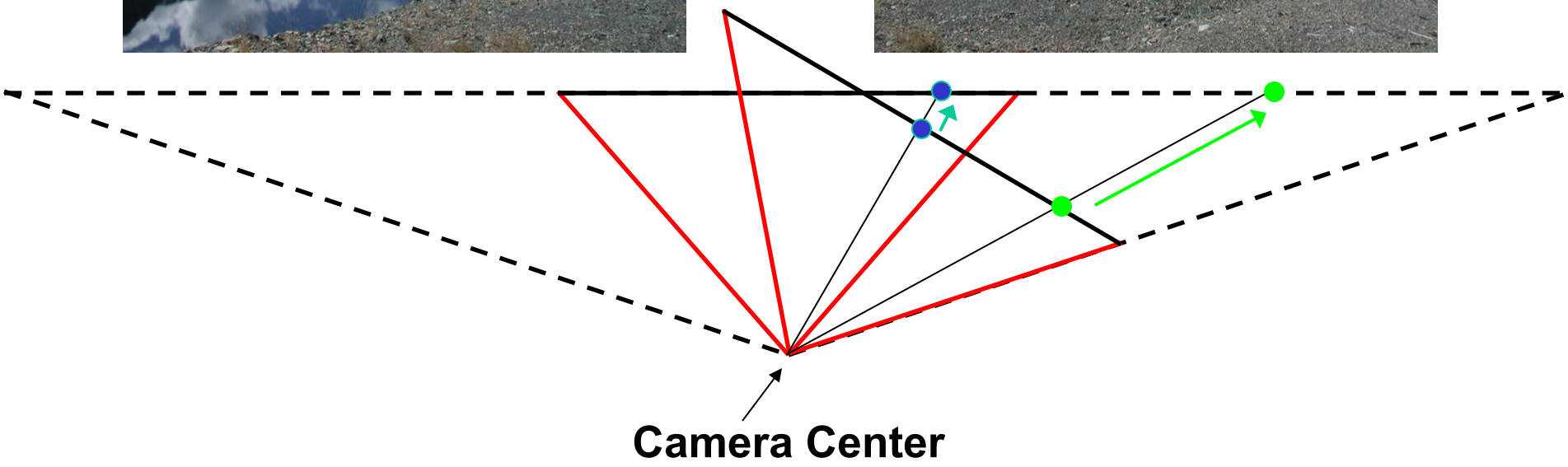


Slide from A. Efros, S. Seitz, D. Hoiem

Homography Example

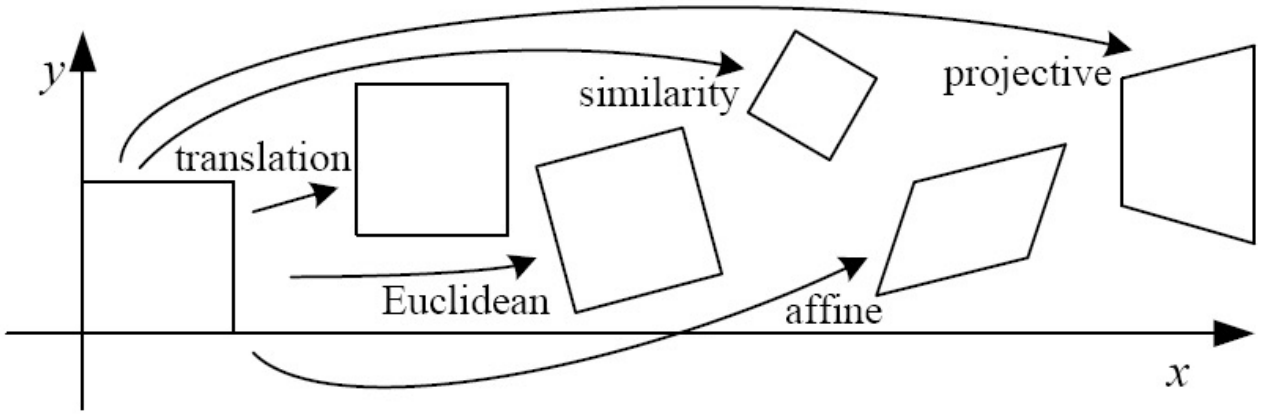






Slide from A. Efros, S. Seitz, D. Hoiem

2D image transformations

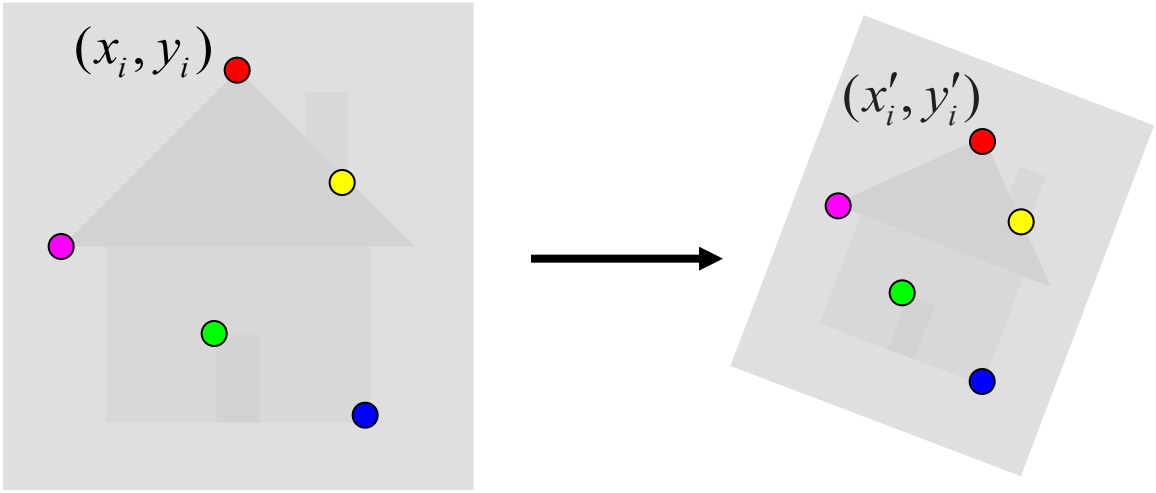


Name	Matrix	# D.O.F.	Preserves:	Icon
translation	$\left[\begin{array}{c\|c} \boldsymbol{I} & \boldsymbol{t} \end{array}\right]_{2\times 3}$	2	orientation $+\cdots$	
rigid (Euclidean)	$\left[\begin{array}{c\|c} \boldsymbol{R} & \boldsymbol{t} \end{array}\right]_{2\times 3}$	3	lengths $+\cdots$	
similarity	$\left[\begin{array}{c\|c} s\boldsymbol{R} & \boldsymbol{t} \end{array}\right]_{2\times 3}$	4	angles $+\cdots$	
affine	$\left[\begin{array}{c} \boldsymbol{A} \end{array}\right]_{2\times 3}$	6	parallelism $+\cdots$	
projective	$\left[\begin{array}{c} \tilde{\boldsymbol{H}} \end{array}\right]_{3\times 3}$	8	straight lines	

Slide from R. Szeliski

Fitting an affine transformation

- Assume we know the correspondences, how do we get the transformation?



$$\begin{bmatrix} x_i' \\ y_i' \end{bmatrix} = \begin{bmatrix} m_1 & m_2 \\ m_3 & m_4 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} + \begin{bmatrix} t_1 \\ t_2 \end{bmatrix}$$

$$\mathbf{x}_i' = \mathbf{M}\mathbf{x}_i + \mathbf{t}$$

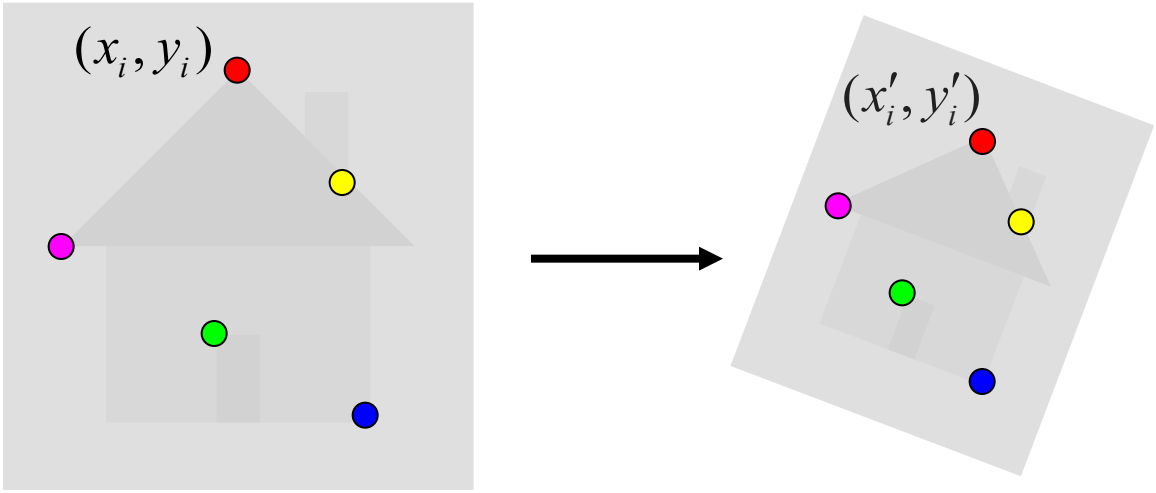
Want to find **M**, **t** to minimize

$$\sum_{i=1}^{n} || \mathbf{x}_i' - \mathbf{M}\mathbf{x}_i - \mathbf{t} ||^2$$

Slide from L. Lazebnik.

Fitting an affine transformation

- Assume we know the correspondences, how do we get the transformation?



$$\begin{bmatrix} x_i' \\ y_i' \end{bmatrix} = \begin{bmatrix} m_1 & m_2 \\ m_3 & m_4 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} + \begin{bmatrix} t_1 \\ t_2 \end{bmatrix}$$

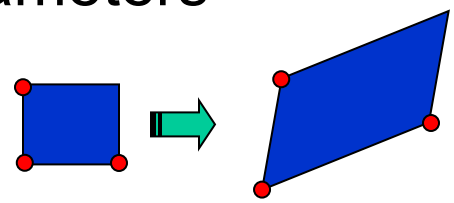
$$\begin{bmatrix} & & & & & \\ & & & & & \\ & & & & & \\ & & & & & \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ t_1 \\ t_2 \end{bmatrix} = \begin{bmatrix} \cdots \\ x_i' \\ y_i' \\ \cdots \end{bmatrix}$$

Slide from L. Lazebnik.

Fitting an affine transformation

$$\begin{bmatrix} & & \cdots & & & \\ x_i & y_i & 0 & 0 & 1 & 0 \\ 0 & 0 & x_i & y_i & 0 & 1 \\ & & \cdots & & & \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ t_1 \\ t_2 \end{bmatrix} = \begin{bmatrix} \cdots \\ x_i' \\ y_i' \\ \cdots \end{bmatrix}$$

- Linear system with six unknowns
- Each match gives us two linearly independent equations: need at least three to solve for the transformation parameters



Slide from L. Lazebnik.

Fitting a homography

- Recall: homogeneous coordinates

$$(x, y) \Rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Converting *to* homogeneous image coordinates

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} \Rightarrow (x/w, y/w)$$

Converting *from* homogeneous image coordinates

Slide from L. Lazebnik.

Fitting a homography

- Recall: homogeneous coordinates

$$(x, y) \Rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Converting *to* homogeneous image coordinates

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} \Rightarrow (x/w, y/w)$$

Converting *from* homogeneous image coordinates

- Equation for homography:

$$\lambda \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Slide from L. Lazebnik.

Fitting a homography

- Equation for homography:

$$\lambda \begin{bmatrix} x_i' \\ y_i' \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \qquad \lambda \mathbf{x}_i' = \mathbf{H}\mathbf{x}_i$$

$$\mathbf{x}_i' \times \mathbf{H}\mathbf{x}_i = 0$$

$$\begin{bmatrix} x_i' \\ y_i' \\ 1 \end{bmatrix} \times \begin{bmatrix} \mathbf{h_1}^T \mathbf{x}_i \\ \mathbf{h_2}^T \mathbf{x}_i \\ \mathbf{h_3}^T \mathbf{x}_i \end{bmatrix} = \begin{bmatrix} y_i' \mathbf{h}_3^T \mathbf{x}_i - \mathbf{h}_2^T \mathbf{x}_i \\ \mathbf{h}_1^T \mathbf{x}_i - x_i' \mathbf{h}_3^T \mathbf{x}_i \\ x_i' \mathbf{h}_2^T \mathbf{x}_i - y_i' \mathbf{h}_1^T \mathbf{x}_i \end{bmatrix}$$

$$\begin{bmatrix} 0^T & -\mathbf{x}_i^T & y_i' \mathbf{x}_i^T \\ \mathbf{x}_i^T & 0^T & -x_i' \mathbf{x}_i^T \\ -y_i' \mathbf{x}_i^T & x_i' \mathbf{x}_i^T & 0^T \end{bmatrix} \begin{pmatrix} \mathbf{h}_1 \\ \mathbf{h}_2 \\ \mathbf{h}_3 \end{pmatrix} = 0$$

3 equations, only 2 linearly independent

Slide from L. Lazebnik.

Fitting a homography

$$\begin{bmatrix} 0^T & \mathbf{x}_1^T & -y_1' \mathbf{x}_1^T \\ \mathbf{x}_1^T & 0^T & -x_1' \mathbf{x}_1^T \\ \cdots & \cdots & \cdots \\ 0^T & \mathbf{x}_n^T & -y_n' \mathbf{x}_n^T \\ \mathbf{x}_n^T & 0^T & -x_n' \mathbf{x}_n^T \end{bmatrix} \begin{pmatrix} \mathbf{h}_1 \\ \mathbf{h}_2 \\ \mathbf{h}_3 \end{pmatrix} = 0 \qquad \mathbf{A}\,\mathbf{h} = 0$$

- H has 8 degrees of freedom (9 parameters, but scale is arbitrary)
- One match gives us two linearly independent equations
- Homogeneous least squares: find $\mathbf{h}$ minimizing $||\mathbf{A}\mathbf{h}||^2$
  - Eigenvector of $\mathbf{A}^\mathbf{T}\mathbf{A}$ corresponding to smallest eigenvalue
  - Four matches needed for a minimal solution

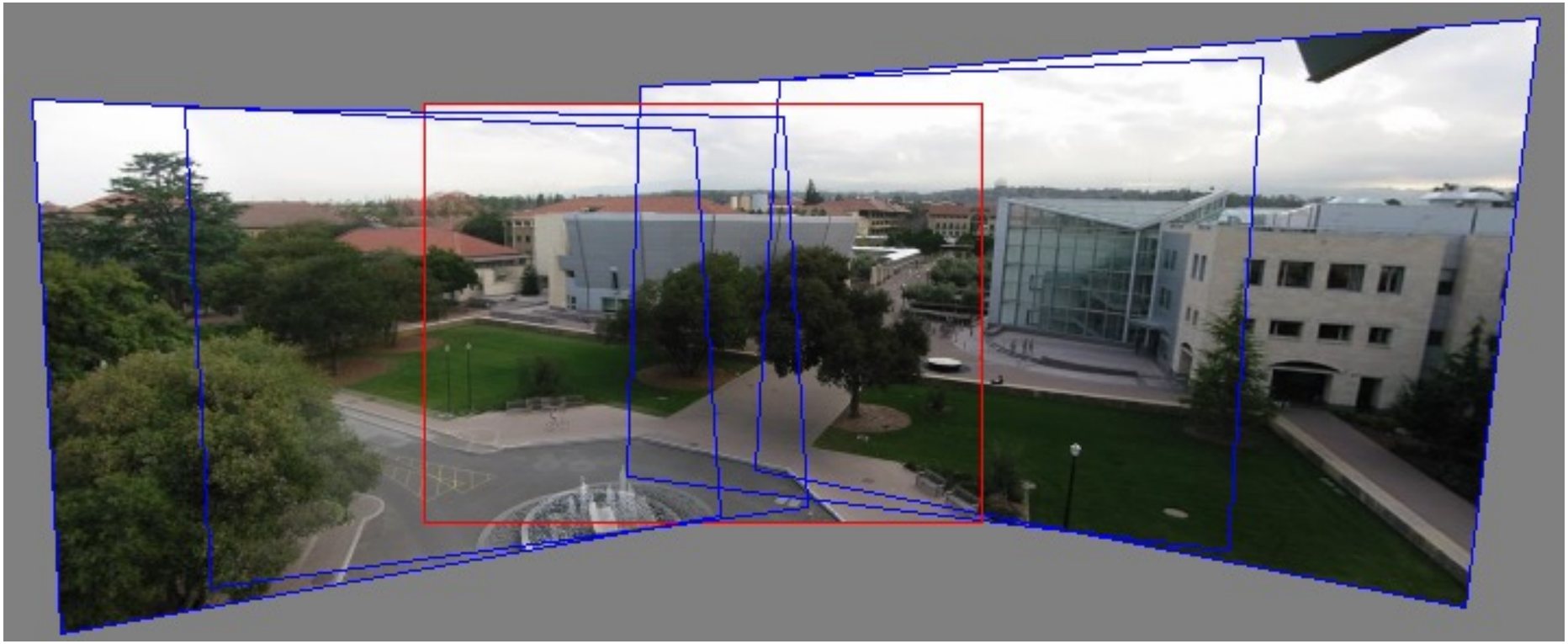
Slide from L. Lazebnik.

Feature-based alignment: Overview

- Alignment as fitting
  - Affine transformations
  - Homographies
- Robust alignment
  - Descriptor-based feature matching
  - RANSAC
- Applications

Slide from L. Lazebnik

Image alignment



Slides from Derek Hoiem, Svetlana Lazebnik

Image source

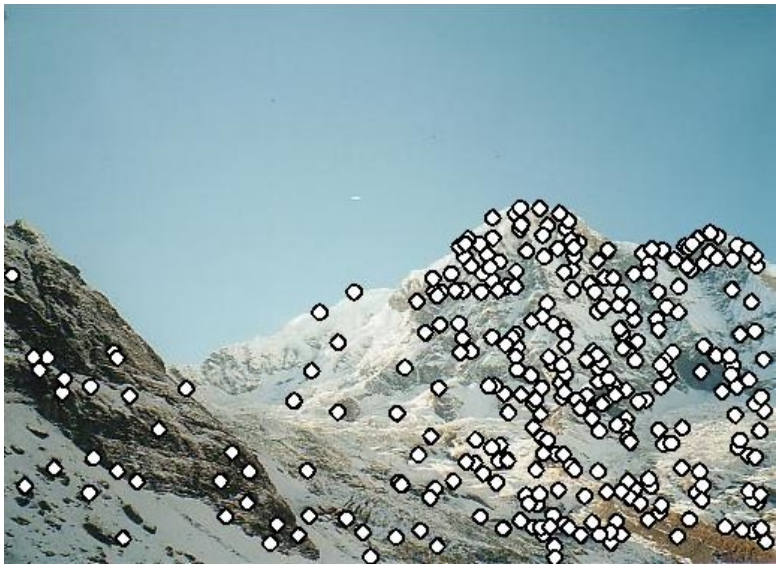
Image Alignment

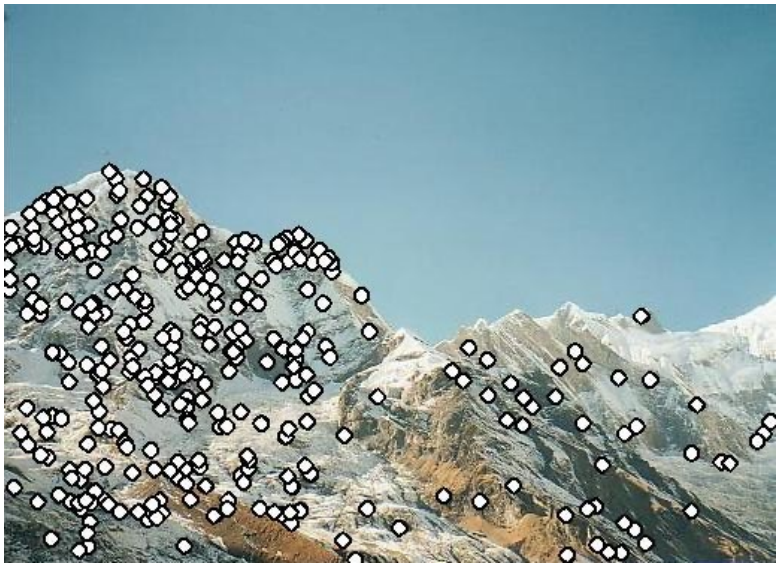
Slide from L. Lazebnik.

Robust feature-based alignment

Slide from L. Lazebnik.

Robust feature-based alignment

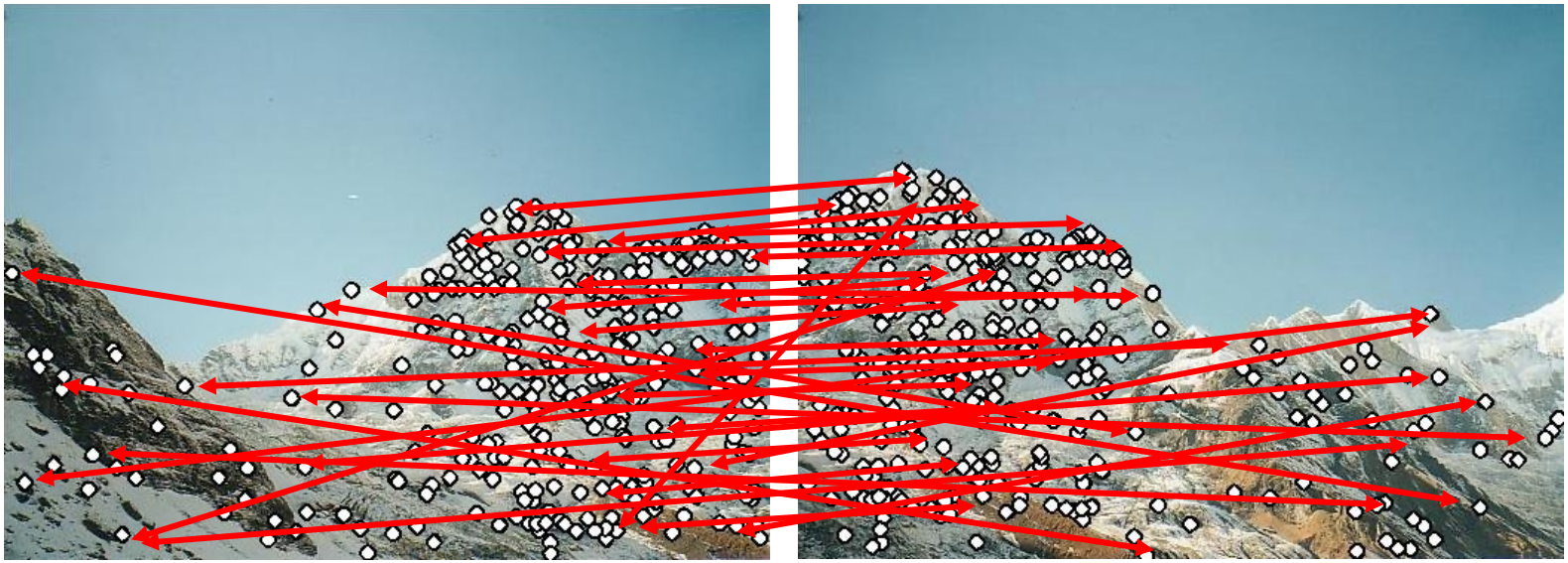




- Extract features

Slide from L. Lazebnik.

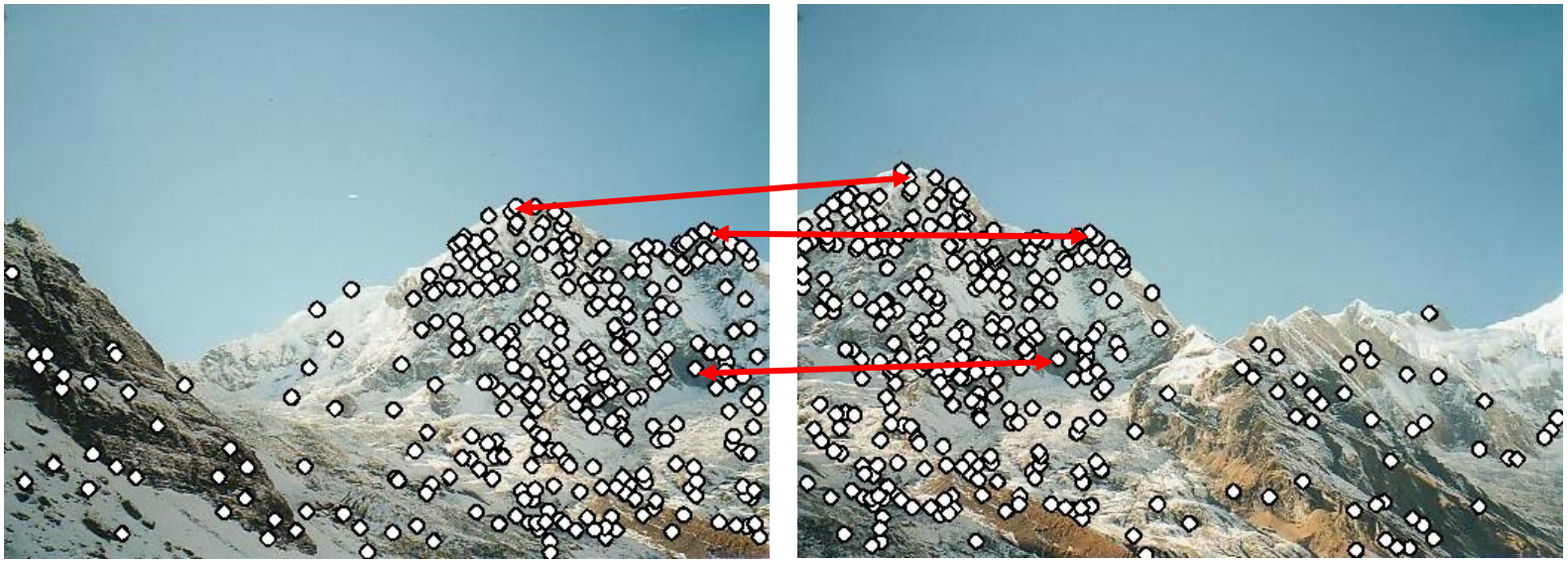
Robust feature-based alignment



- Extract features
- Compute *putative matches*

Slide from L. Lazebnik.

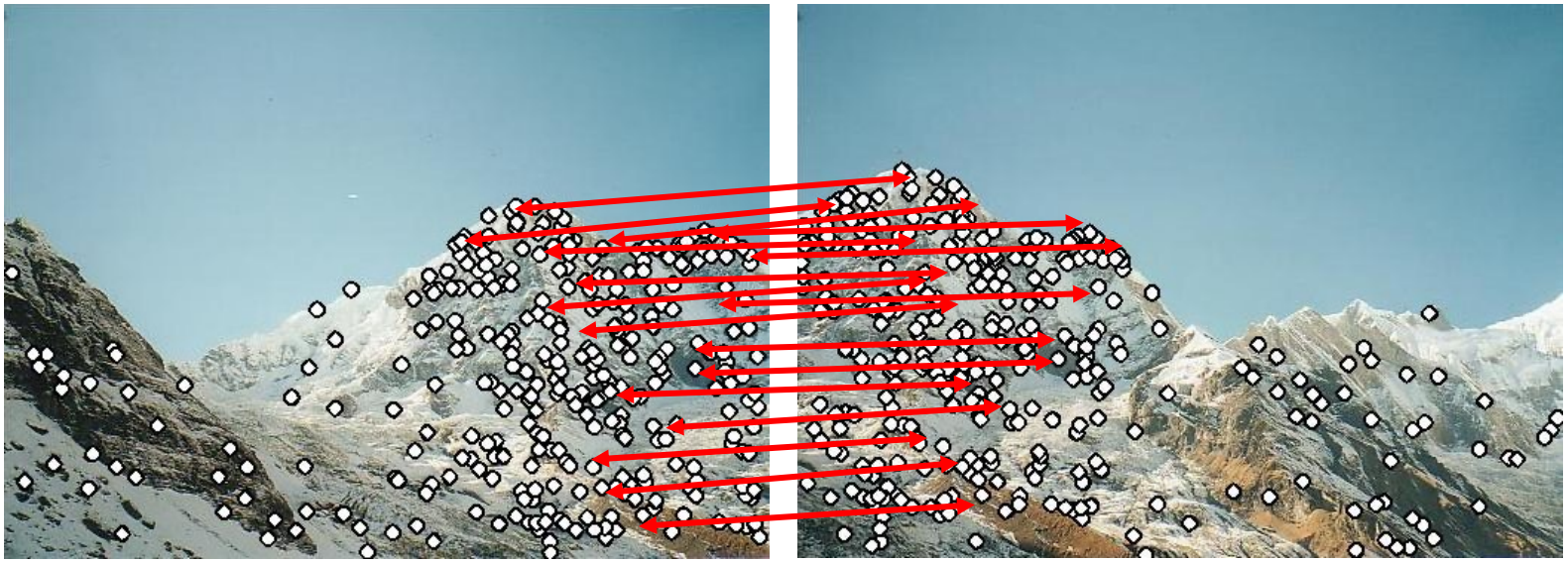
Robust feature-based alignment



- Extract features
- Compute *putative matches*
- Loop:
  - *Hypothesize* transformation *T*

Slide from L. Lazebnik.

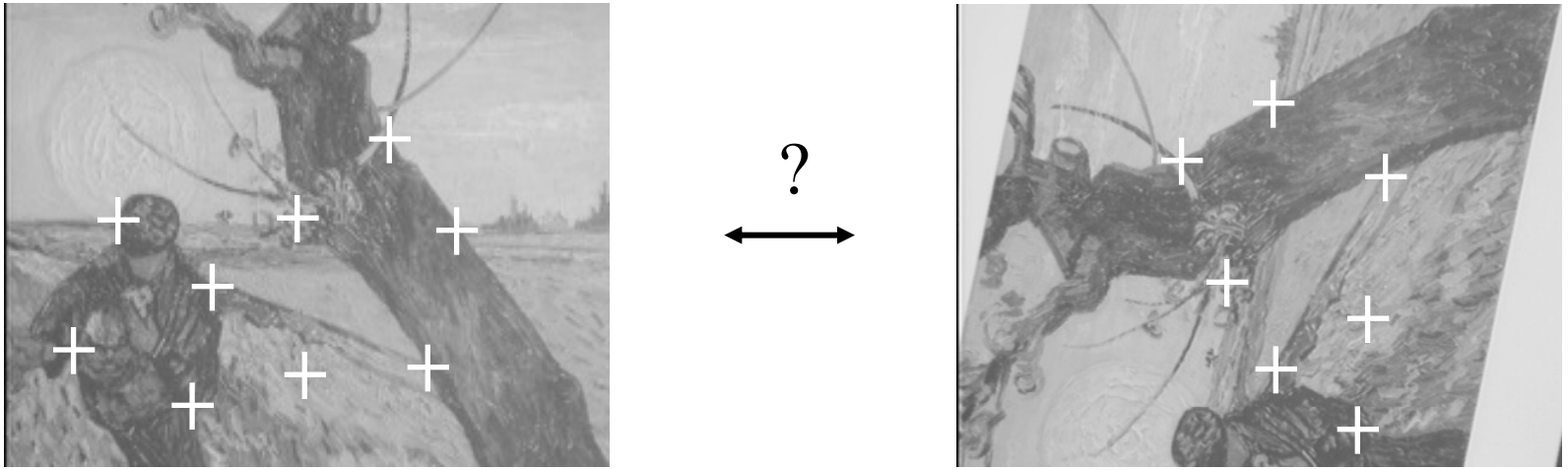
Robust feature-based alignment



- Extract features
- Compute *putative matches*
- Loop:
  - *Hypothesize* transformation *T*
  - *Verify* transformation (search for other matches consistent with *T*)

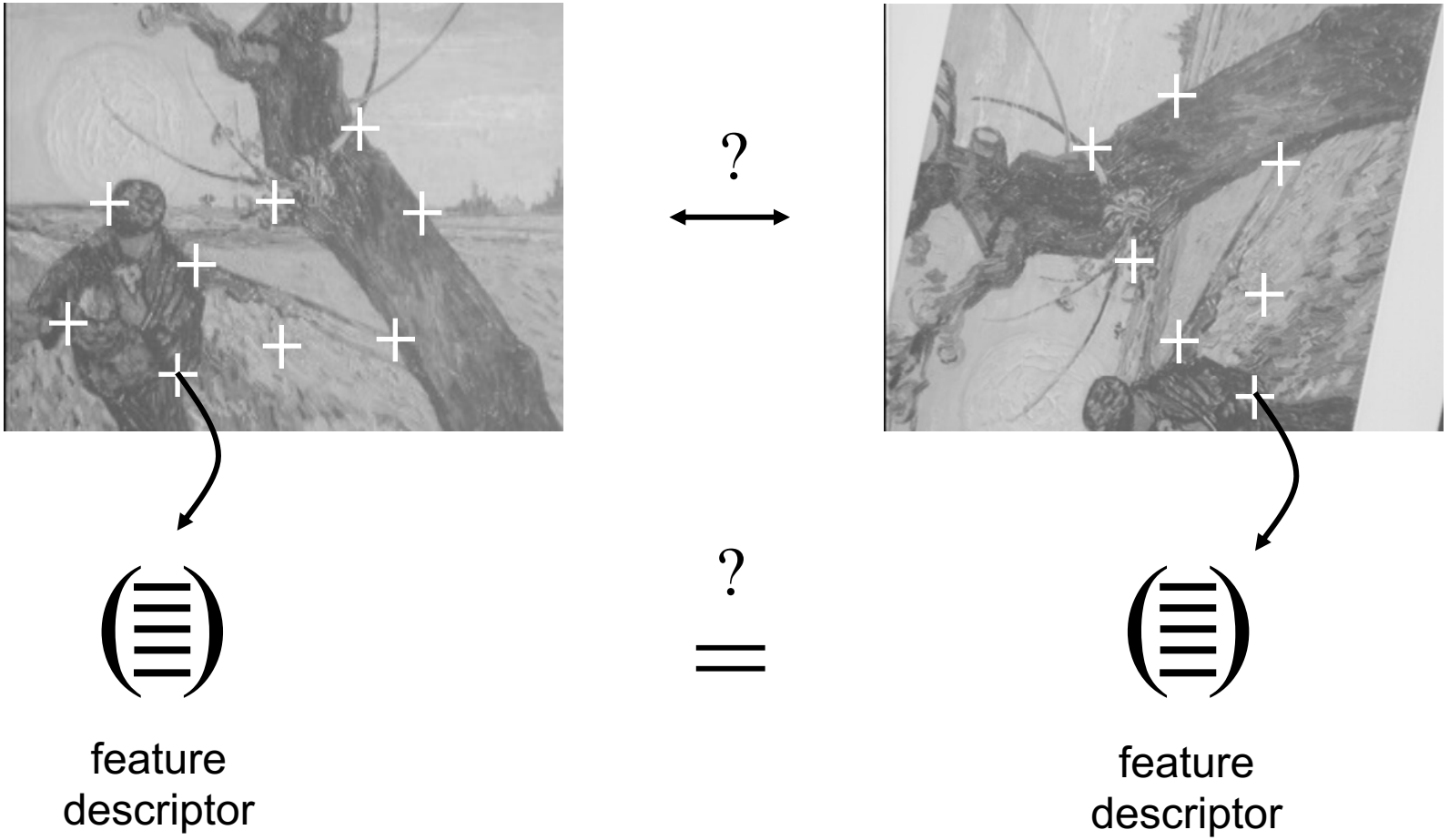
Slide from L. Lazebnik.

Generating putative correspondences



Slide from L. Lazebnik.

Generating putative correspondences

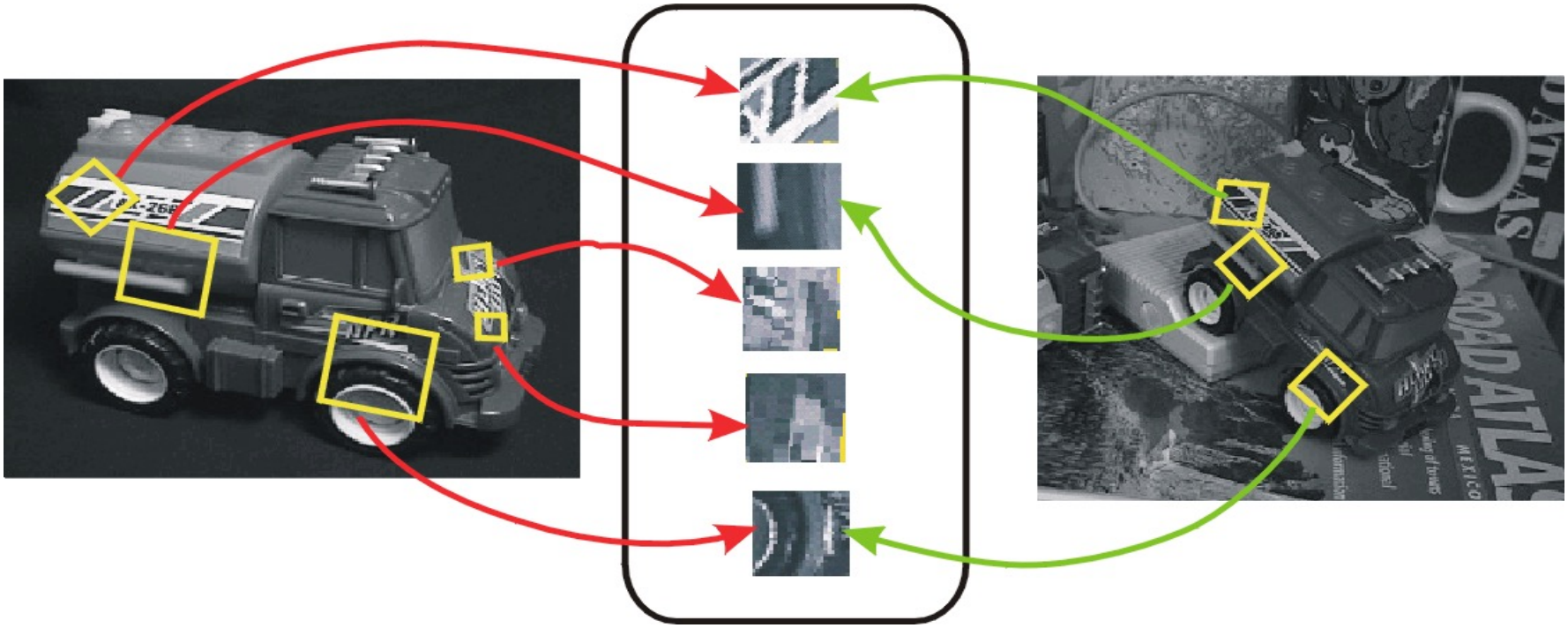


- Need to compare *feature descriptors* of local patches surrounding interest points

Slide from L. Lazebnik.

Feature descriptors

- Recall: feature detection and description



Slide from L. Lazebnik.

Feature descriptors

- Simplest descriptor: vector of raw intensity values
- How to compare two such vectors?
  - Sum of squared differences (SSD)

$$\mathrm{SSD}(\mathbf{u},\mathbf{v}) = \sum_i \left(u_i - v_i\right)^2$$

    - Not invariant to intensity change

  - Normalized correlation

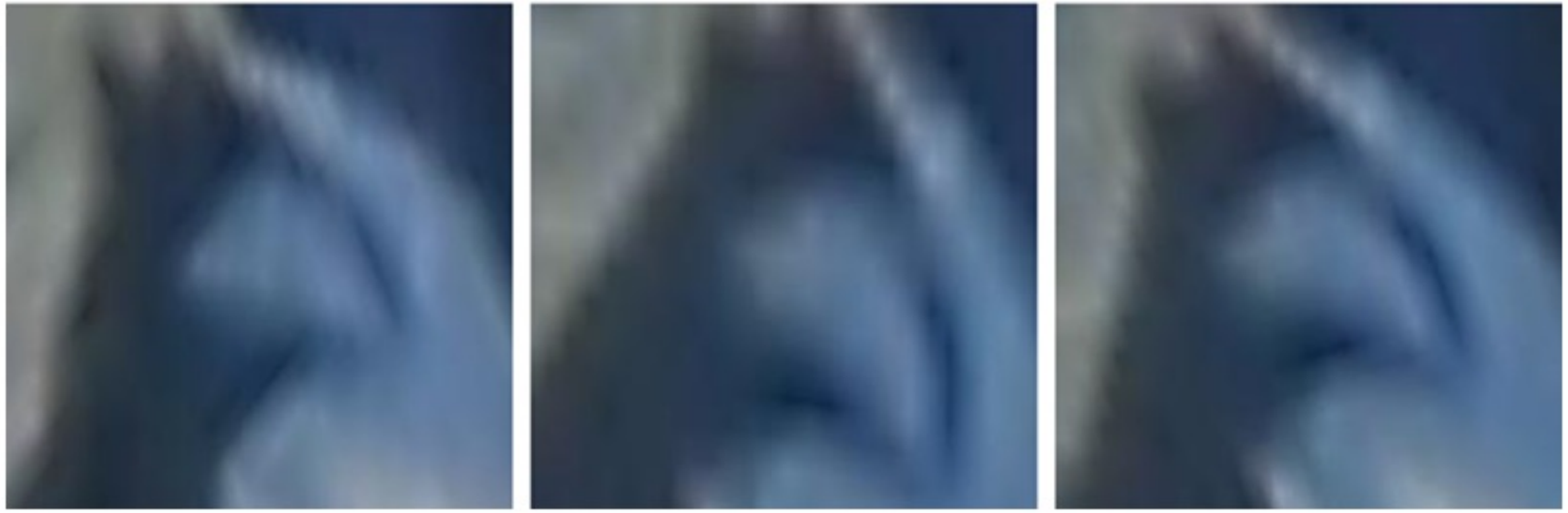
$$\rho(\mathbf{u},\mathbf{v}) = \frac{(\mathbf{u}-\bar{\mathbf{u}})}{\|\mathbf{u}-\bar{\mathbf{u}}\|} \cdot \frac{(\mathbf{v}-\bar{\mathbf{v}})}{\|\mathbf{v}-\bar{\mathbf{v}}\|} = \frac{\sum_i (u_i - \bar{\mathbf{u}})(v_i - \bar{\mathbf{v}})}{\sqrt{\left(\sum_j (u_j - \bar{\mathbf{u}})^2\right)\left(\sum_j (v_j - \bar{\mathbf{v}})^2\right)}}$$

    - Invariant to affine intensity change

Slide from L. Lazebnik.

Disadvantage of intensity vectors as descriptors

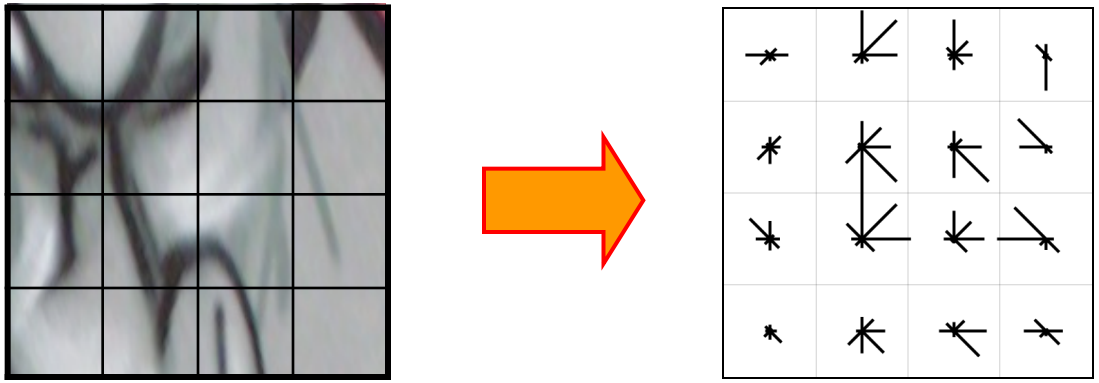
- Small deformations can affect the matching score a lot



Slide from L. Lazebnik.

Feature descriptors: SIFT

- Descriptor computation:
  - Divide patch into 4x4 sub-patches
  - Compute histogram of gradient orientations (8 reference angles) inside each sub-patch
  - Resulting descriptor: 4x4x8 = 128 dimensions



David G. Lowe. "Distinctive image features from scale-invariant keypoints." *IJCV* 60 (2), pp. 91-110, 2004.

Slide from L. Lazebnik.

Feature descriptors: SIFT

- Descriptor computation:
  - Divide patch into 4x4 sub-patches
  - Compute histogram of gradient orientations (8 reference angles) inside each sub-patch
  - Resulting descriptor: 4x4x8 = 128 dimensions

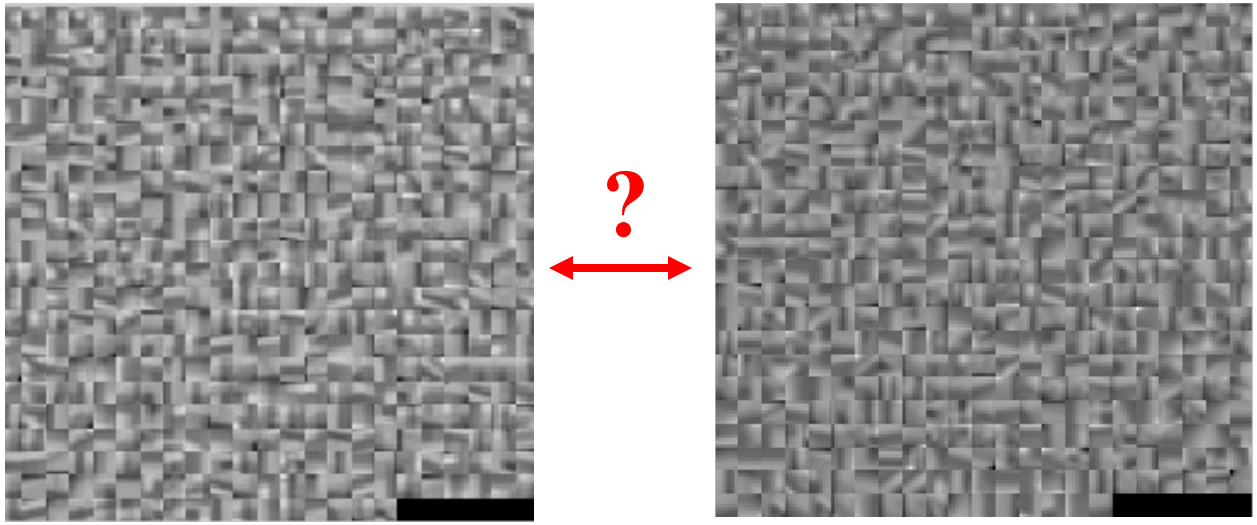
- Advantage over raw vectors of pixel values
  - Gradients less sensitive to illumination change
  - Pooling of gradients over the sub-patches achieves robustness to small shifts, but still preserves some spatial information

David G. Lowe. "Distinctive image features from scale-invariant keypoints." *IJCV* 60 (2), pp. 91-110, 2004.

Slide from L. Lazebnik.

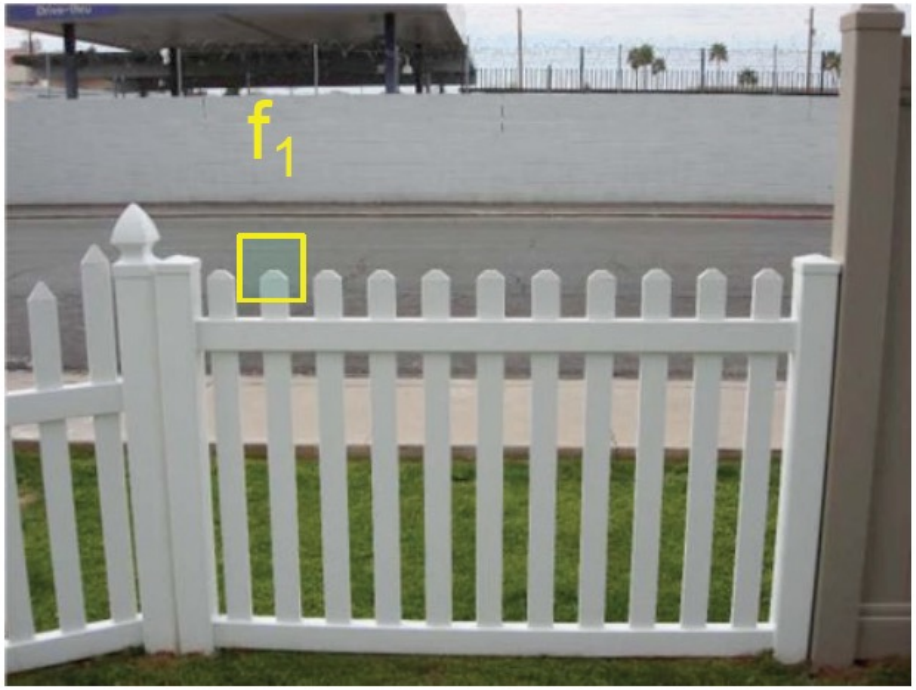
Feature matching

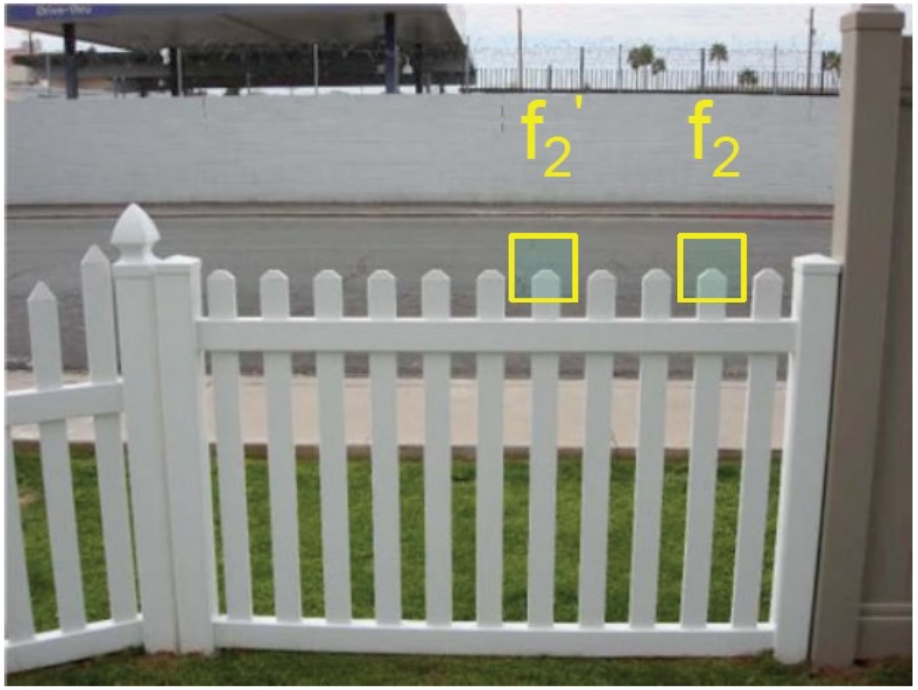
- Generating *putative matches*: for each patch in one image, find a short list of patches in the other image that could match it based solely on appearance



Slide from L. Lazebnik.

Problem: Ambiguous putative matches

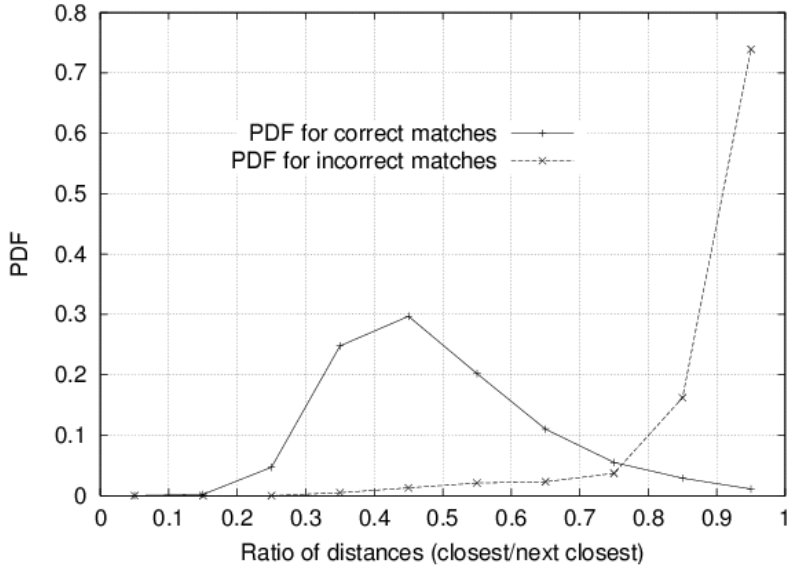




Source: Y. Furukawa

Rejection of unreliable matches

- How can we tell which putative matches are more reliable?
- Heuristic: compare distance of **nearest** neighbor to that of **second** nearest neighbor
  - Ratio of closest distance to second-closest distance will be *high* for features that are *not* distinctive



Threshold of 0.8 provides good separation

David G. Lowe. "Distinctive image features from scale-invariant keypoints." *IJCV* 60 (2), pp. 91-110, 2004.

Slide from L. Lazebnik.

RANSAC

- The set of putative matches contains a very high percentage of outliers

RANSAC loop:

1. Randomly select a *seed group* of matches
2. Compute transformation from seed group
3. Find *inliers* to this transformation
4. If the number of inliers is sufficiently large, re-compute least-squares estimate of transformation on all of the inliers

Keep the transformation with the largest number of inliers

Slide from L. Lazebnik.

Robust feature-based alignment

- Extract features
- Compute *putative matches*
- Loop:
  - *Hypothesize* transformation *T*
  - *Verify* transformation (search for other matches consistent with *T*)

Slide from L. Lazebnik.

Common transformations

original

Transformed

translation

rotation

aspect

affine

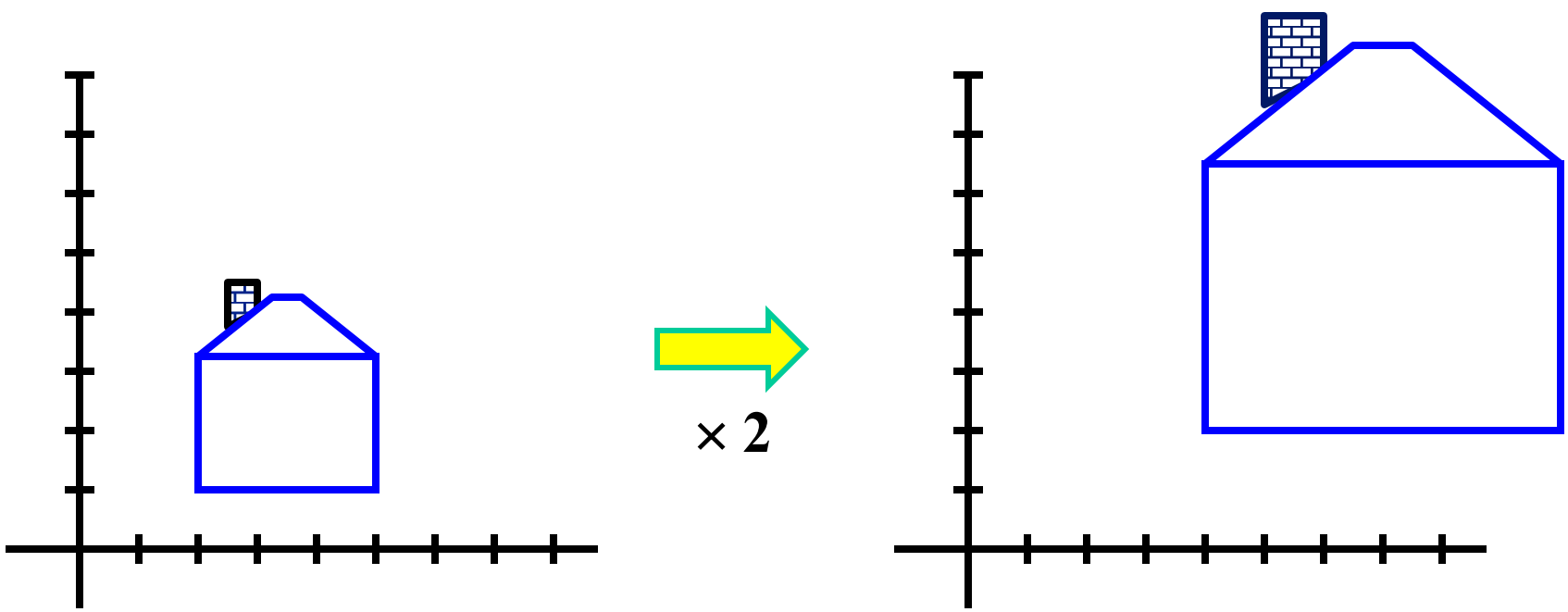
perspective

Slide from A. Efros, S. Seitz, D. Hoiem

Scaling

Scaling a coordinate means multiplying each of its components by a scalar

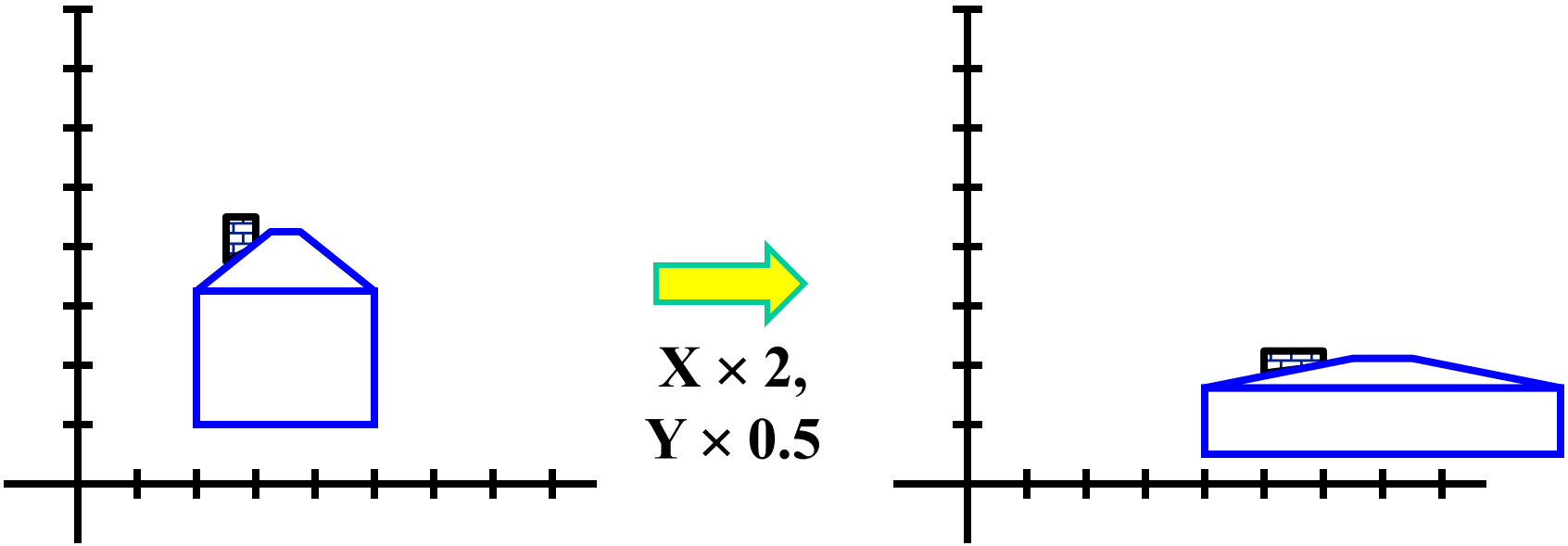
Uniform scaling means this scalar is the same for all components:



Slide from A. Efros, S. Seitz, D. Hoiem

Scaling

Non-uniform scaling: different scalars per component:



Slide from A. Efros, S. Seitz, D. Hoiem

Scaling

Scaling operation:

$$x' = ax$$

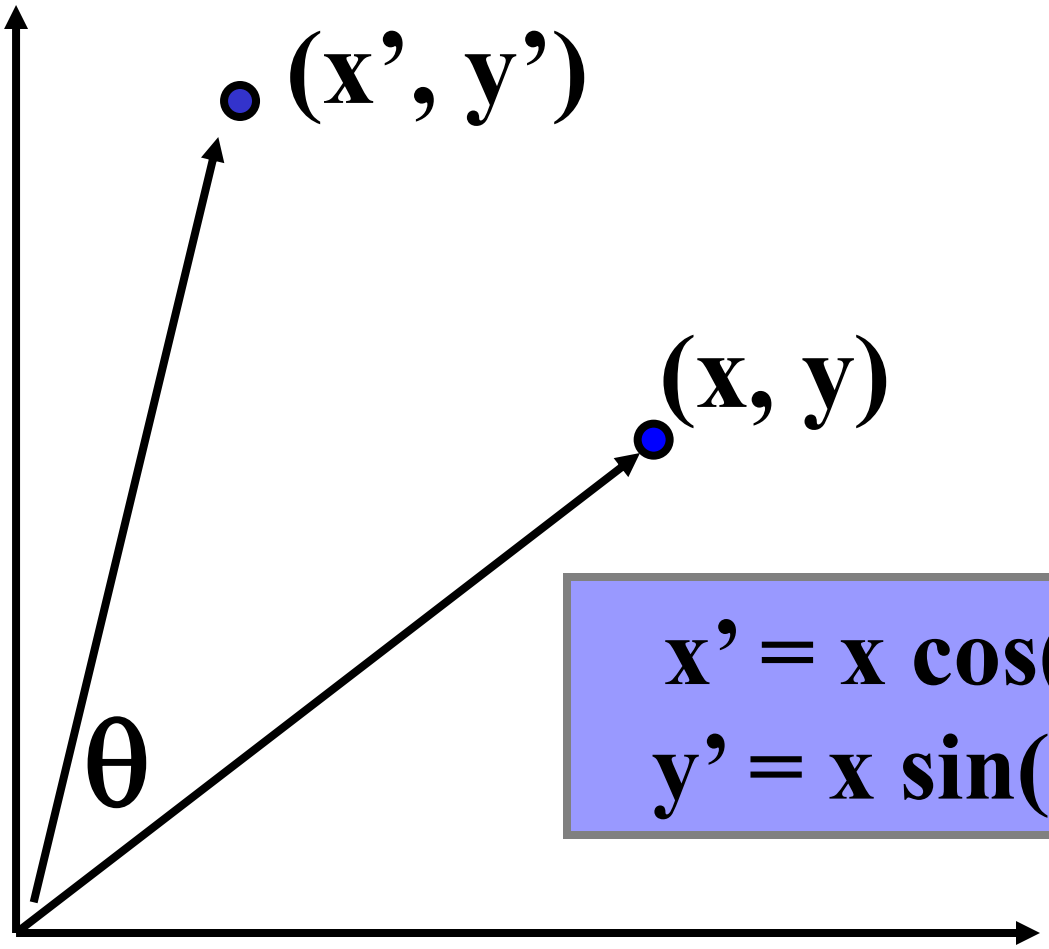
$$y' = by$$

Or, in matrix form:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \underbrace{\begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}}_{\textbf{scaling matrix S}} \begin{bmatrix} x \\ y \end{bmatrix}$$

Slide from A. Efros, S. Seitz, D. Hoiem

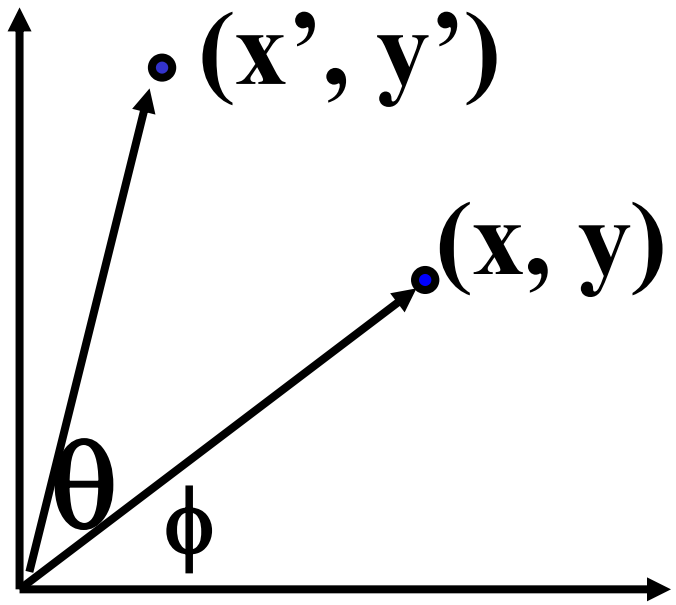
2-D Rotation



$$x' = x \cos(\theta) - y \sin(\theta)$$
$$y' = x \sin(\theta) + y \cos(\theta)$$

Slide from A. Efros, S. Seitz, D. Hoiem

2-D Rotation



Polar coordinates…

$x = r \cos(\phi)$

$y = r \sin(\phi)$

$x' = r \cos(\phi + \theta)$

$y' = r \sin(\phi + \theta)$

Trig Identity…

$x' = r \cos(\phi) \cos(\theta) - r \sin(\phi) \sin(\theta)$

$y' = r \sin(\phi) \cos(\theta) + r \cos(\phi) \sin(\theta)$

Substitute…

$x' = x \cos(\theta) - y \sin(\theta)$

$y' = x \sin(\theta) + y \cos(\theta)$

Slide from A. Efros, S. Seitz, D. Hoiem

2-D Rotation

This is easy to capture in matrix form:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \underbrace{\begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}}_{\mathbf{R}} \begin{bmatrix} x \\ y \end{bmatrix}$$

Even though sin(θ) and cos(θ) are nonlinear functions of θ,

- ***x' is a linear combination of x and y***
- ***y' is a linear combination of x and y***

What is the inverse transformation?

- Rotation by –θ
- For rotation matrices $\mathbf{R}^{-1} = \mathbf{R}^{T}$

Slide from A. Efros, S. Seitz, D. Hoiem

Basic 2D transformations

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Scale

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & \alpha_x \\ \alpha_y & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Shear

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\Theta & -\sin\Theta \\ \sin\Theta & \cos\Theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Rotate

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Translate

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Affine

Affine is any combination of translation, scale, rotation, shear

Slide from A. Efros, S. Seitz, D. Hoiem

Affine Transformations

Affine transformations are combinations of

- Linear transformations, and
- Translations

Properties of affine transformations:

- Lines map to lines
- Parallel lines remain parallel
- Ratios are preserved
- Closed under composition

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

or

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Slide from L. Lazebnik.

Projective Transformations

Projective transformations are combos of

- Affine transformations, and
- Projective warps

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

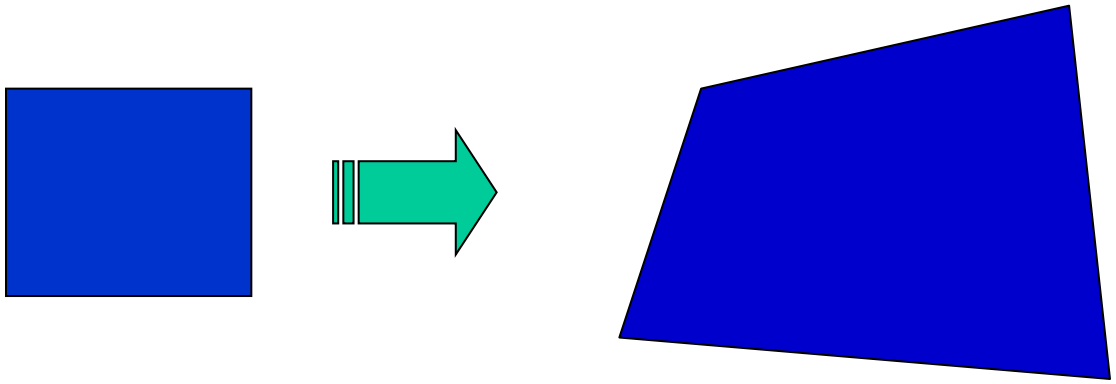
Properties of projective transformations:

- Lines map to lines
- Parallel lines do not necessarily remain parallel
- Ratios are not preserved
- Closed under composition
- **Projective matrix is defined up to a scale (8 DOF)**

Slide from L. Lazebnik.

Fitting a plane projective transformation

- **Homography:** plane projective transformation (transformation taking a quad to another arbitrary quad)

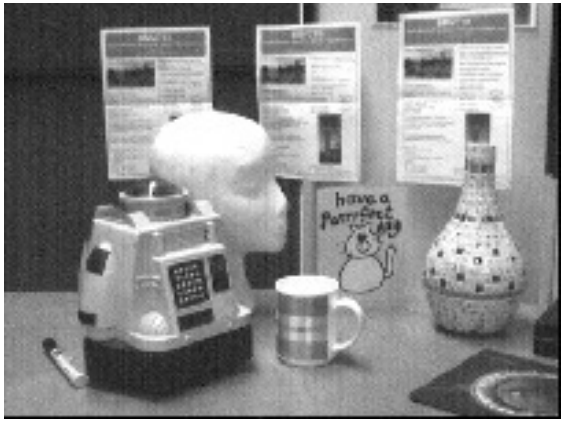


Slide from L. Lazebnik.

Homography

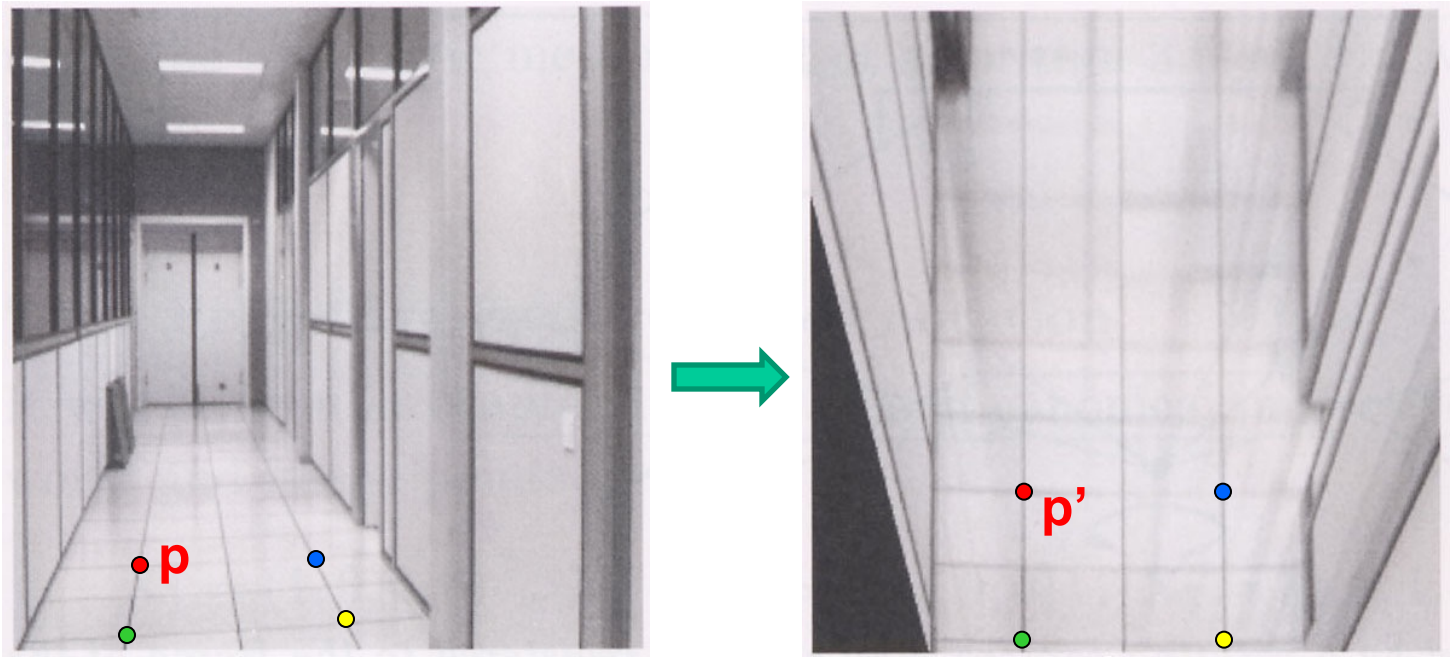
- The transformation between two views of a planar surface


- The transformation between images from two cameras that share the same center

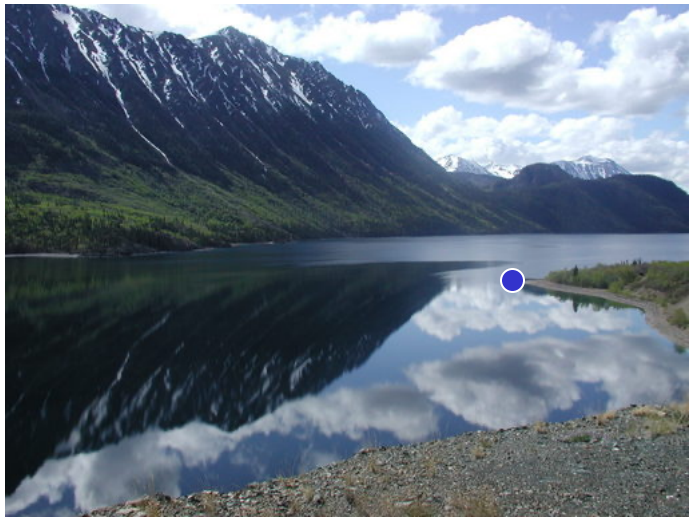
Slide from L. Lazebnik.

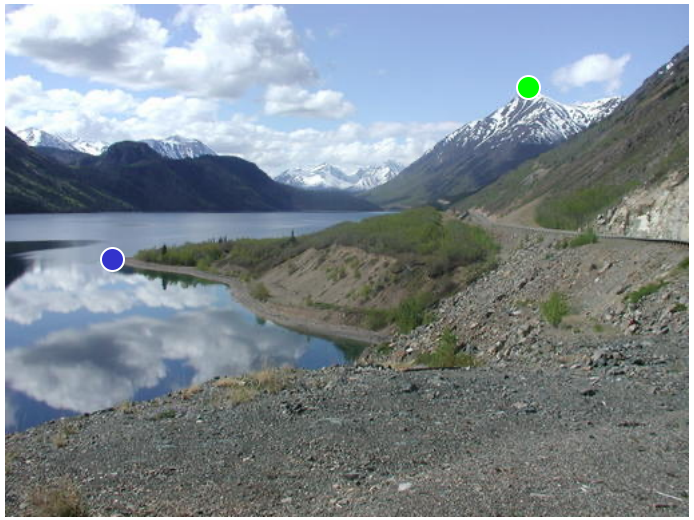
Homography example: Image rectification

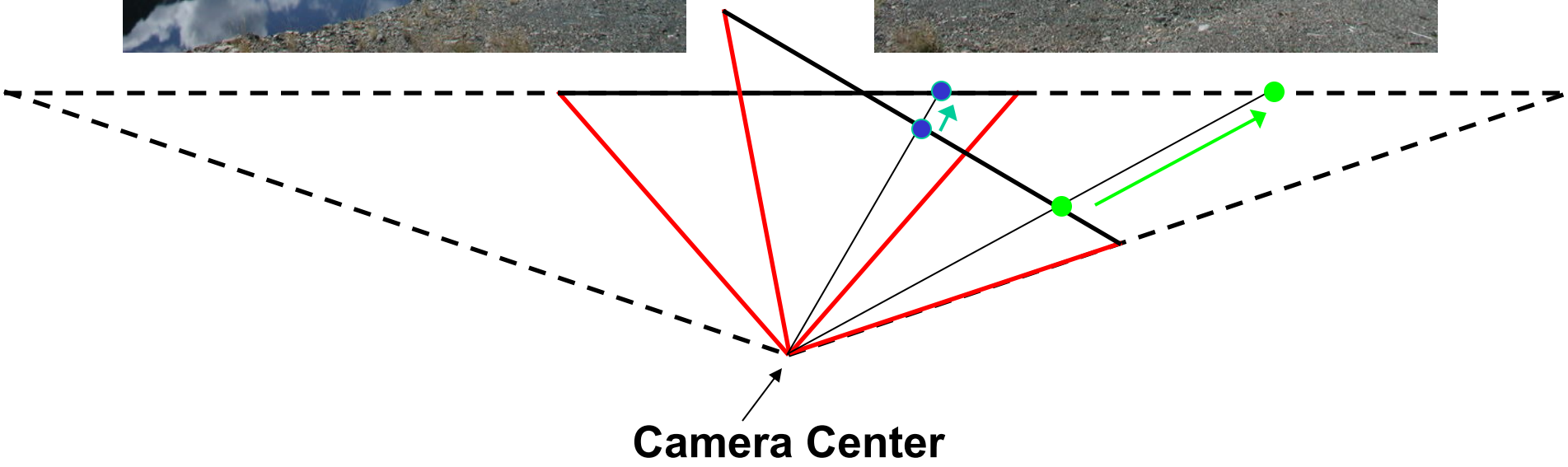


Slide from A. Efros, S. Seitz, D. Hoiem

Homography Example

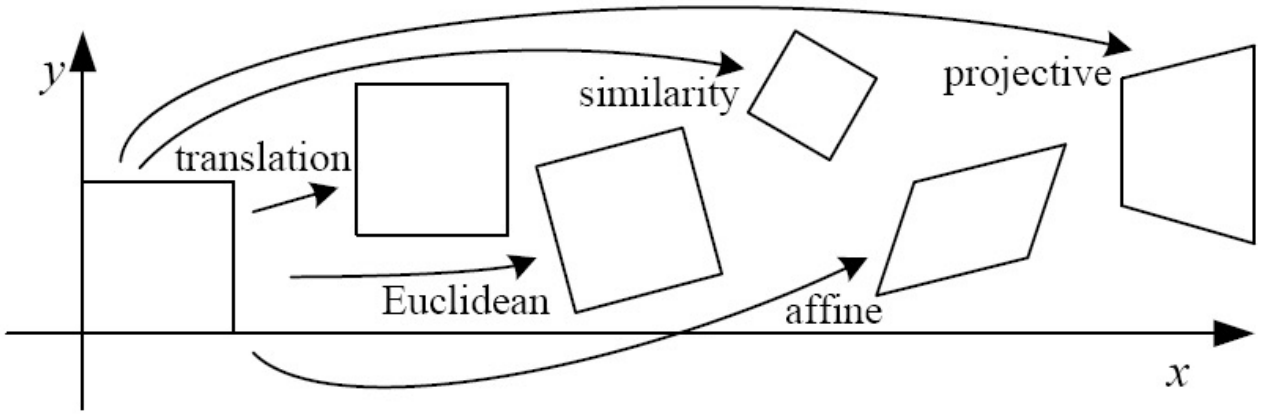






Slide from A. Efros, S. Seitz, D. Hoiem

2D image transformations

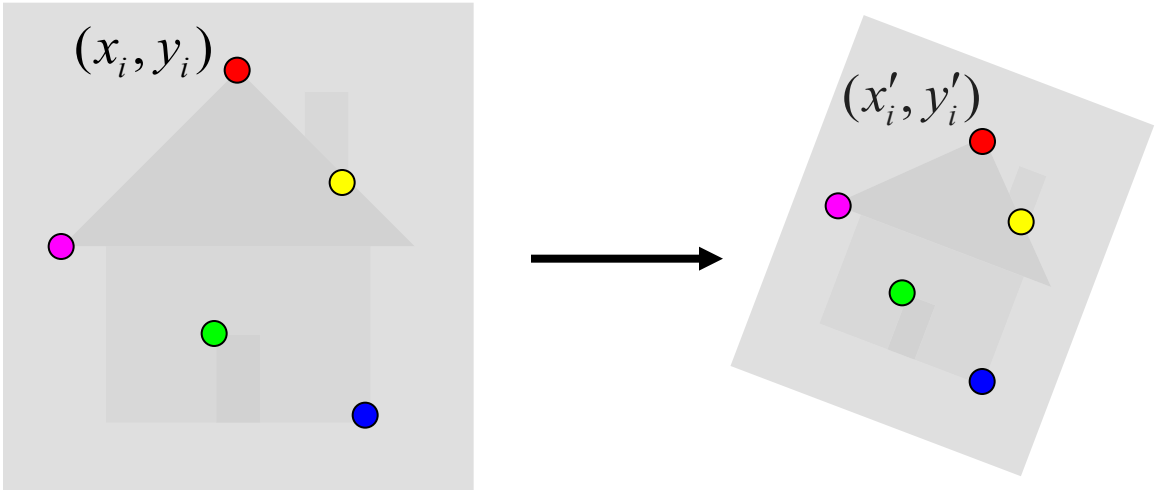


Name	Matrix	# D.O.F.	Preserves:	Icon
translation	$\left[\begin{array}{c\|c} \boldsymbol{I} & \boldsymbol{t} \end{array}\right]_{2\times 3}$	2	orientation $+\cdots$	
rigid (Euclidean)	$\left[\begin{array}{c\|c} \boldsymbol{R} & \boldsymbol{t} \end{array}\right]_{2\times 3}$	3	lengths $+\cdots$	
similarity	$\left[\begin{array}{c\|c} s\boldsymbol{R} & \boldsymbol{t} \end{array}\right]_{2\times 3}$	4	angles $+\cdots$	
affine	$\left[\begin{array}{c} \boldsymbol{A} \end{array}\right]_{2\times 3}$	6	parallelism $+\cdots$	
projective	$\left[\begin{array}{c} \tilde{\boldsymbol{H}} \end{array}\right]_{3\times 3}$	8	straight lines	

Slide from R. Szeliski

Fitting an affine transformation

- Assume we know the correspondences, how do we get the transformation?



$$\begin{bmatrix} x_i' \\ y_i' \end{bmatrix} = \begin{bmatrix} m_1 & m_2 \\ m_3 & m_4 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} + \begin{bmatrix} t_1 \\ t_2 \end{bmatrix}$$

$$\mathbf{x}_i' = \mathbf{M}\mathbf{x}_i + \mathbf{t}$$

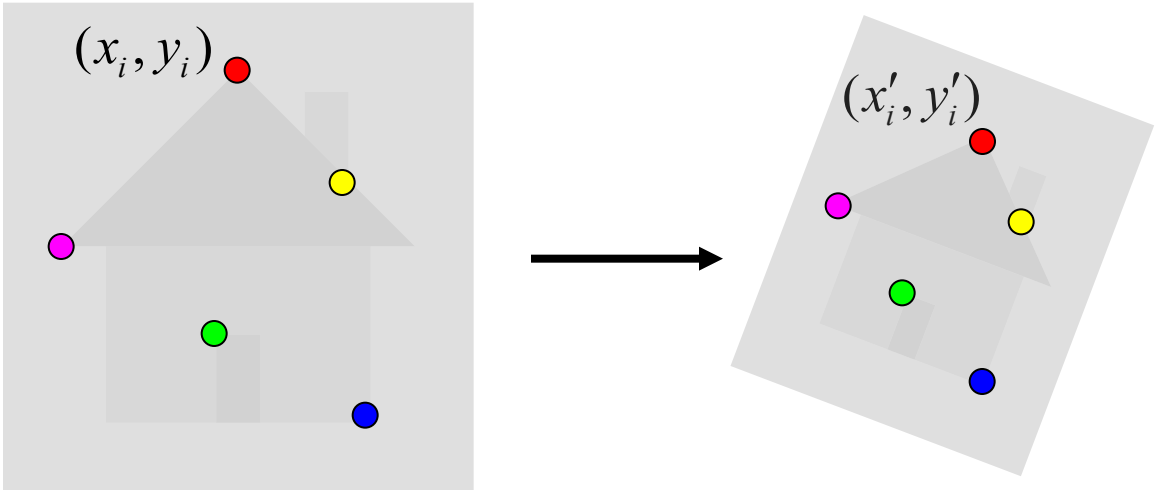
Want to find **M**, **t** to minimize

$$\sum_{i=1}^{n} || \mathbf{x}_i' - \mathbf{M}\mathbf{x}_i - \mathbf{t} ||^2$$

Slide from L. Lazebnik.

Fitting an affine transformation

- Assume we know the correspondences, how do we get the transformation?



$$\begin{bmatrix} x_i' \\ y_i' \end{bmatrix} = \begin{bmatrix} m_1 & m_2 \\ m_3 & m_4 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} + \begin{bmatrix} t_1 \\ t_2 \end{bmatrix}$$

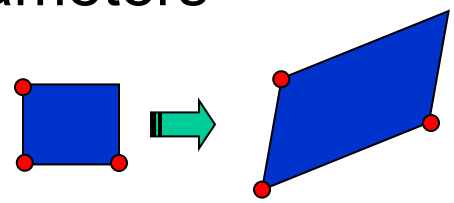
$$\begin{bmatrix} & & & & & \\ & & & & & \\ & & & & & \\ & & & & & \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ t_1 \\ t_2 \end{bmatrix} = \begin{bmatrix} \cdots \\ x_i' \\ y_i' \\ \cdots \end{bmatrix}$$

Slide from L. Lazebnik.

Fitting an affine transformation

$$\begin{bmatrix} & & \cdots & & & \\ x_i & y_i & 0 & 0 & 1 & 0 \\ 0 & 0 & x_i & y_i & 0 & 1 \\ & & \cdots & & & \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ t_1 \\ t_2 \end{bmatrix} = \begin{bmatrix} \cdots \\ x_i' \\ y_i' \\ \cdots \end{bmatrix}$$

- Linear system with six unknowns
- Each match gives us two linearly independent equations: need at least three to solve for the transformation parameters



Slide from L. Lazebnik.

Fitting a homography

- Recall: homogeneous coordinates

$$(x, y) \Rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Converting *to* homogeneous image coordinates

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} \Rightarrow (x/w, y/w)$$

Converting *from* homogeneous image coordinates

Slide from L. Lazebnik.

Fitting a homography

- Recall: homogeneous coordinates

$$(x, y) \Rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Converting *to* homogeneous image coordinates

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} \Rightarrow (x/w, y/w)$$

Converting *from* homogeneous image coordinates

- Equation for homography:

$$\lambda \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Slide from L. Lazebnik.

Fitting a homography

- Equation for homography:

$$\lambda \begin{bmatrix} x_i' \\ y_i' \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \qquad \lambda \mathbf{x}_i' = \mathbf{H}\mathbf{x}_i$$

$$\mathbf{x}_i' \times \mathbf{H}\mathbf{x}_i = 0$$

$$\begin{bmatrix} x_i' \\ y_i' \\ 1 \end{bmatrix} \times \begin{bmatrix} \mathbf{h_1}^T \mathbf{x}_i \\ \mathbf{h_2}^T \mathbf{x}_i \\ \mathbf{h_3}^T \mathbf{x}_i \end{bmatrix} = \begin{bmatrix} y_i' \mathbf{h}_3^T \mathbf{x}_i - \mathbf{h}_2^T \mathbf{x}_i \\ \mathbf{h}_1^T \mathbf{x}_i - x_i' \mathbf{h}_3^T \mathbf{x}_i \\ x_i' \mathbf{h}_2^T \mathbf{x}_i - y_i' \mathbf{h}_1^T \mathbf{x}_i \end{bmatrix}$$

$$\begin{bmatrix} 0^T & -\mathbf{x}_i^T & y_i' \mathbf{x}_i^T \\ \mathbf{x}_i^T & 0^T & -x_i' \mathbf{x}_i^T \\ -y_i' \mathbf{x}_i^T & x_i' \mathbf{x}_i^T & 0^T \end{bmatrix} \begin{pmatrix} \mathbf{h}_1 \\ \mathbf{h}_2 \\ \mathbf{h}_3 \end{pmatrix} = 0$$

3 equations, only 2 linearly independent

Slide from L. Lazebnik.

Fitting a homography

$$\begin{bmatrix} 0^T & \mathbf{x}_1^T & -y_1' \mathbf{x}_1^T \\ \mathbf{x}_1^T & 0^T & -x_1' \mathbf{x}_1^T \\ \cdots & \cdots & \cdots \\ 0^T & \mathbf{x}_n^T & -y_n' \mathbf{x}_n^T \\ \mathbf{x}_n^T & 0^T & -x_n' \mathbf{x}_n^T \end{bmatrix} \begin{pmatrix} \mathbf{h}_1 \\ \mathbf{h}_2 \\ \mathbf{h}_3 \end{pmatrix} = 0 \qquad \mathbf{A}\,\mathbf{h} = 0$$

- H has 8 degrees of freedom (9 parameters, but scale is arbitrary)
- One match gives us two linearly independent equations
- Homogeneous least squares: find $\mathbf{h}$ minimizing $||\mathbf{A}\mathbf{h}||^2$
  - Eigenvector of $\mathbf{A}^\mathbf{T}\mathbf{A}$ corresponding to smallest eigenvalue
  - Four matches needed for a minimal solution

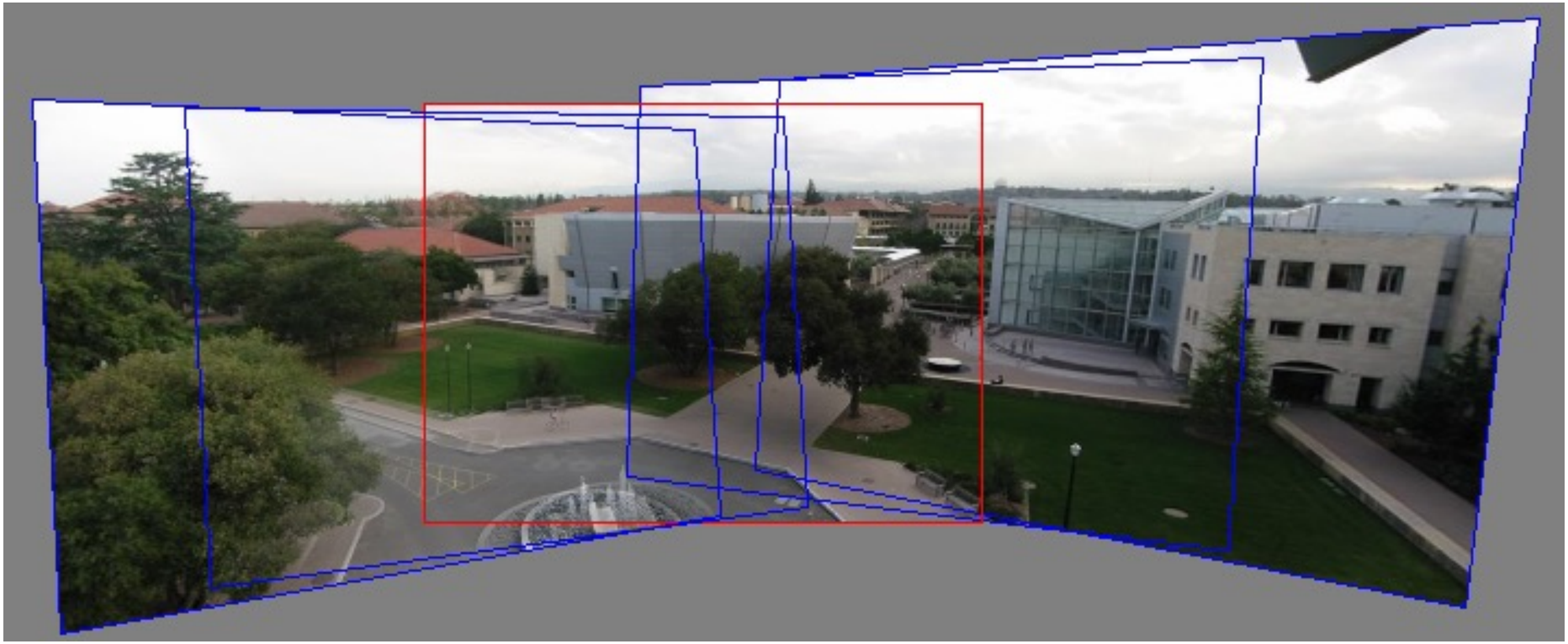
Slide from L. Lazebnik.

Feature-based alignment: Overview

- Alignment as fitting
  - Affine transformations
  - Homographies
- Robust alignment
  - Descriptor-based feature matching
  - RANSAC
- Applications

Slide from L. Lazebnik

Image alignment



Slides from Derek Hoiem, Svetlana Lazebnik

Image source

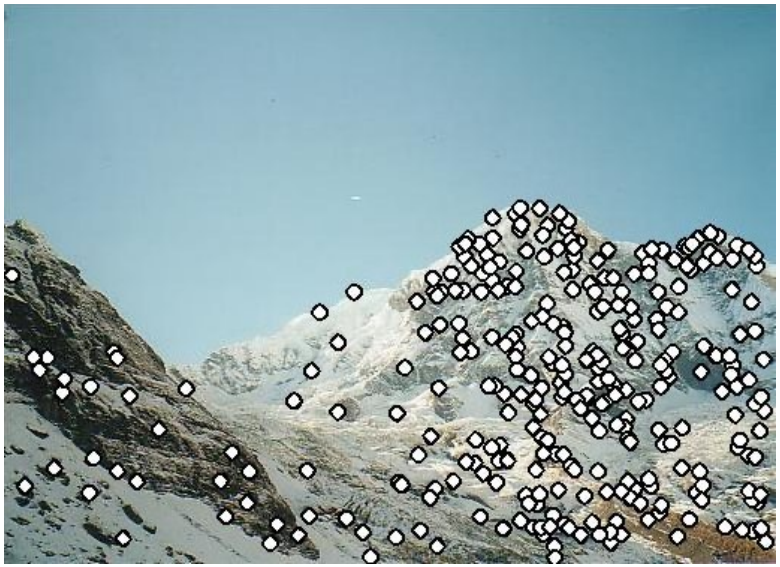
Image Alignment

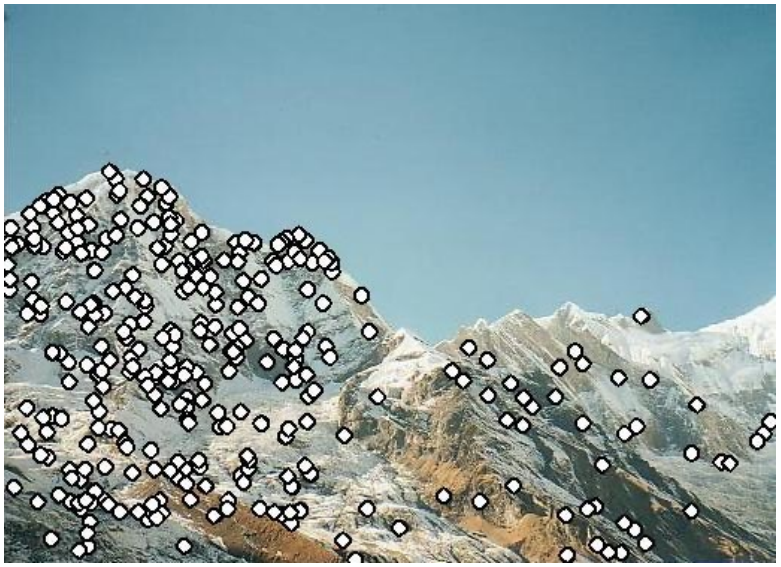
Slide from L. Lazebnik.

Robust feature-based alignment

Slide from L. Lazebnik.

Robust feature-based alignment

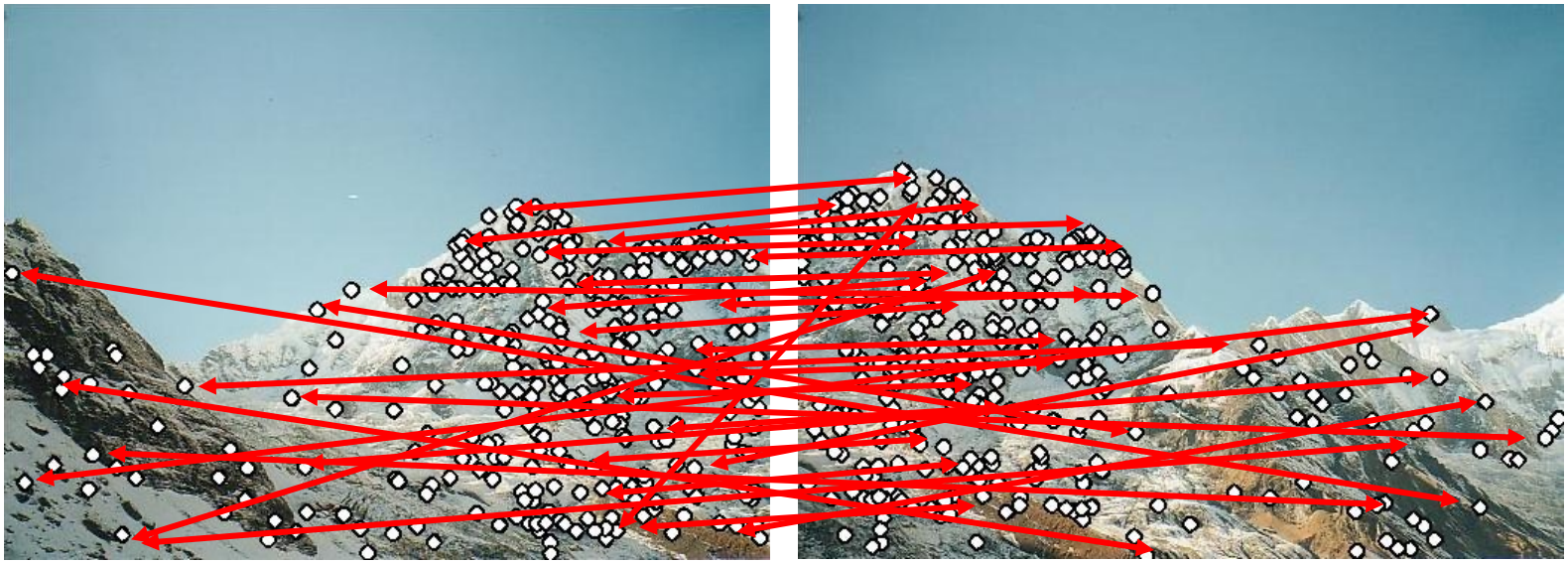




- Extract features

Slide from L. Lazebnik.

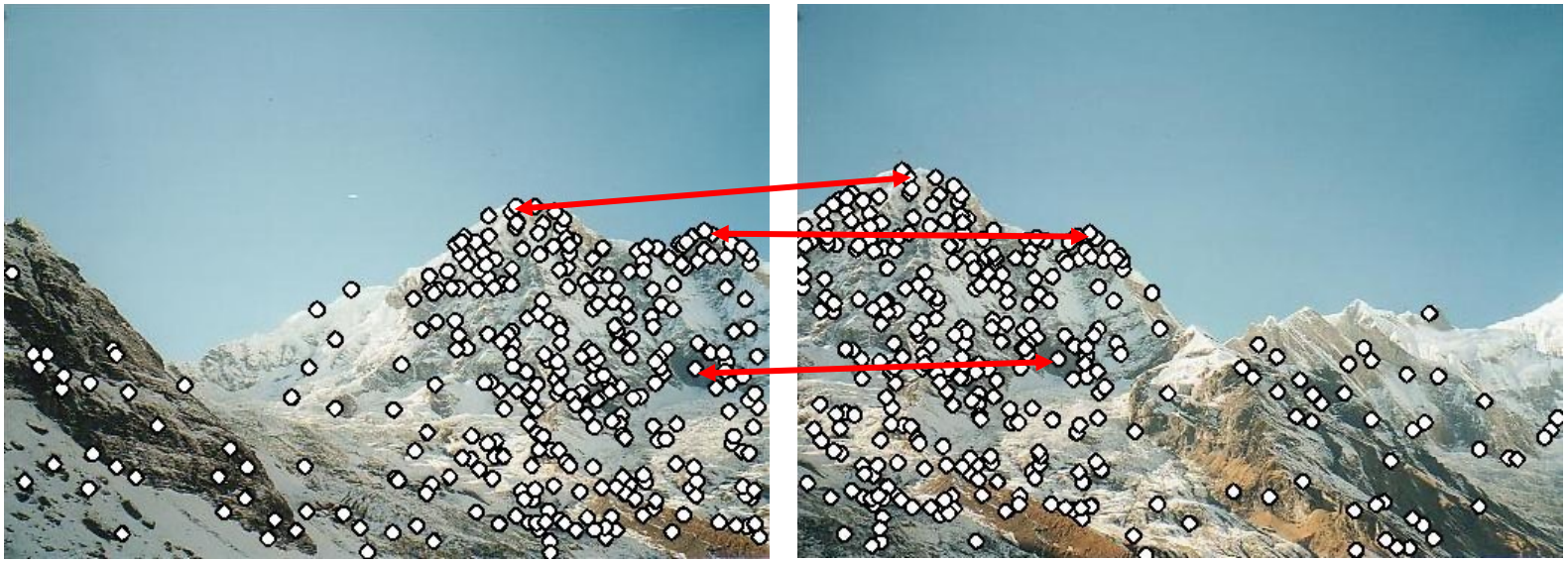
Robust feature-based alignment



- Extract features
- Compute *putative matches*

Slide from L. Lazebnik.

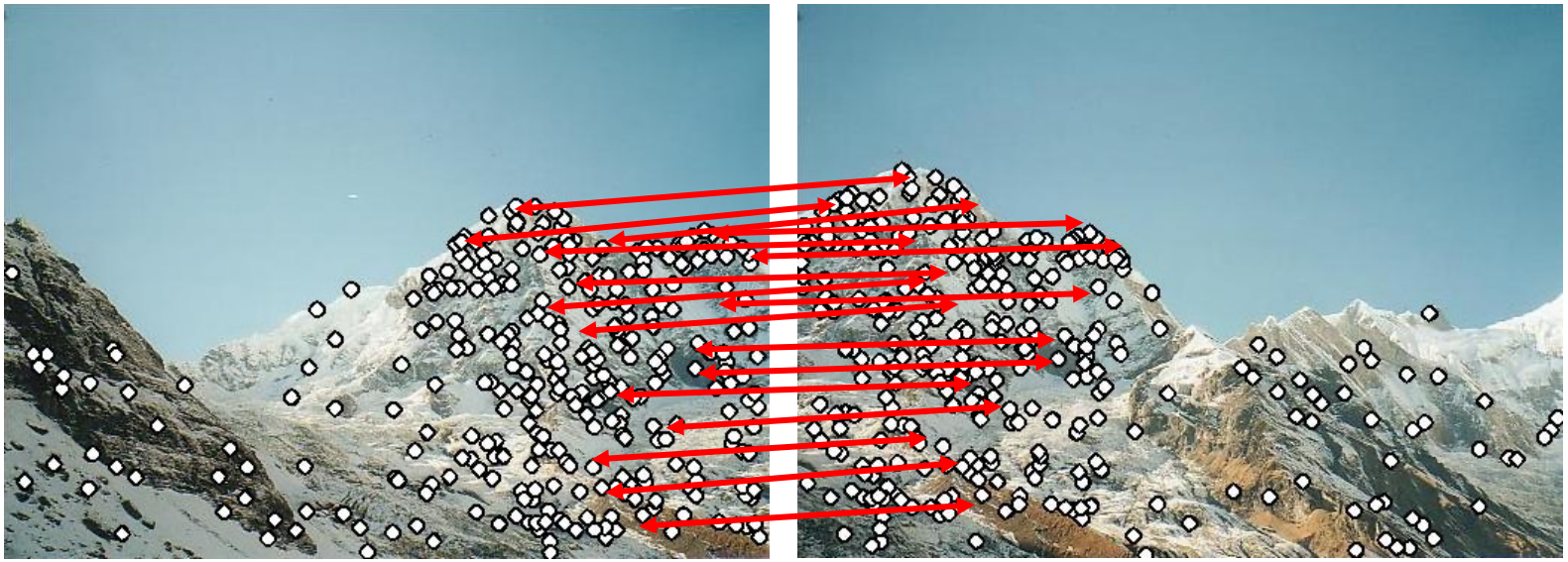
Robust feature-based alignment



- Extract features
- Compute *putative matches*
- Loop:
  - *Hypothesize* transformation *T*

Slide from L. Lazebnik.

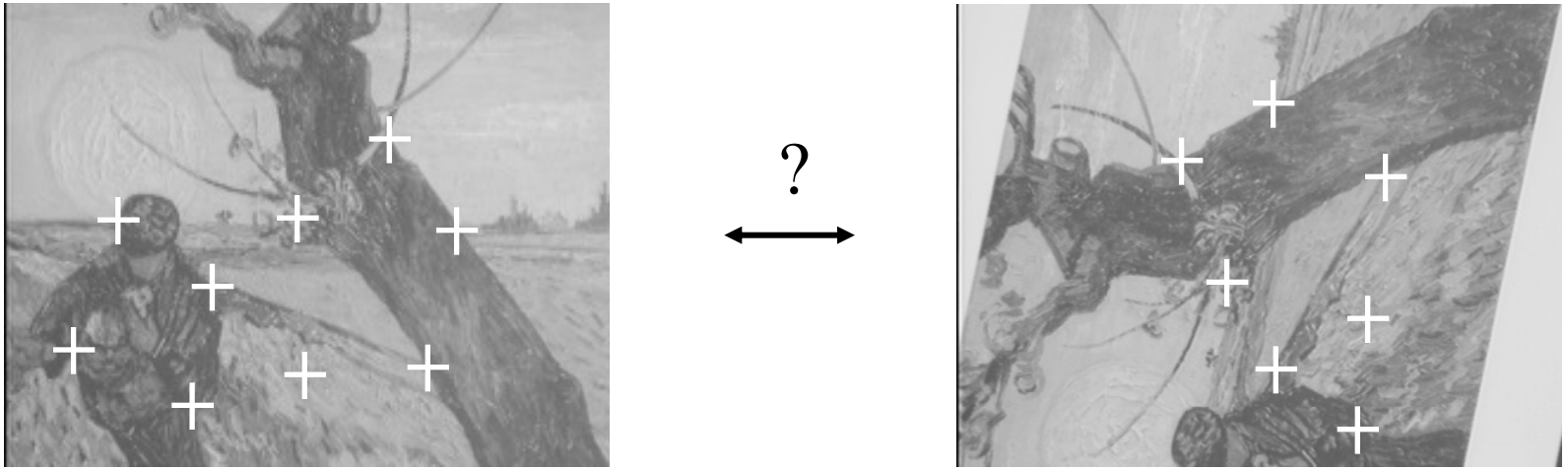
Robust feature-based alignment



- Extract features
- Compute *putative matches*
- Loop:
  - *Hypothesize* transformation *T*
  - *Verify* transformation (search for other matches consistent with *T*)

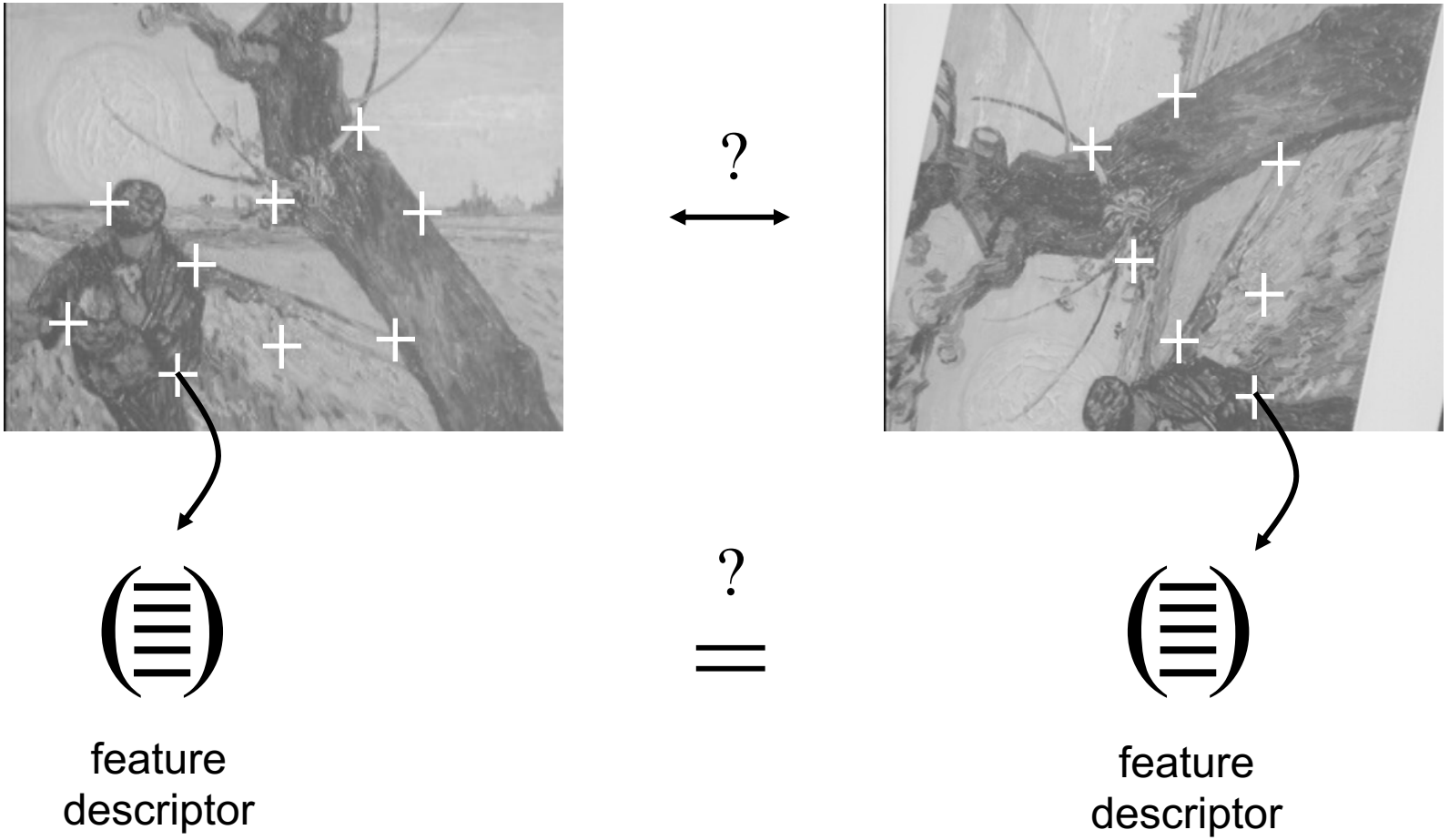
Slide from L. Lazebnik.

Generating putative correspondences



Slide from L. Lazebnik.

Generating putative correspondences

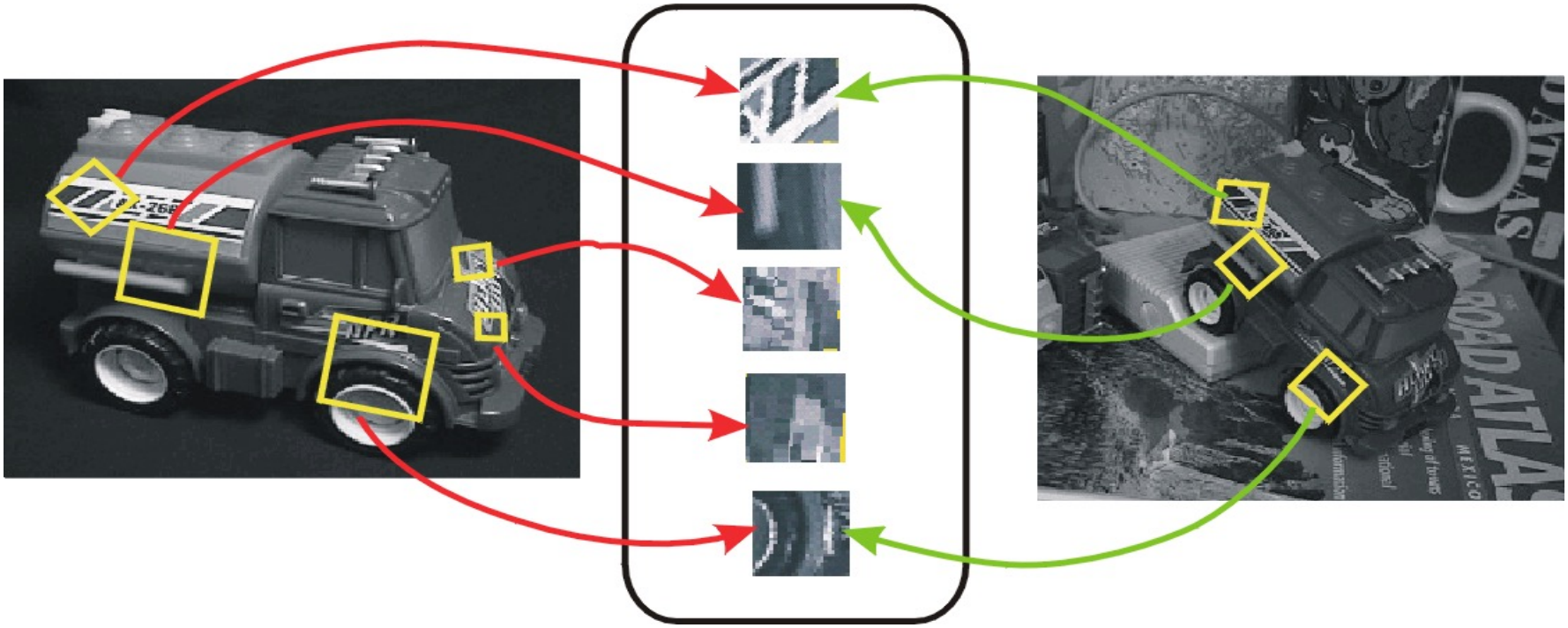


- Need to compare *feature descriptors* of local patches surrounding interest points

Slide from L. Lazebnik.

Feature descriptors

- Recall: feature detection and description



Slide from L. Lazebnik.

Feature descriptors

- Simplest descriptor: vector of raw intensity values
- How to compare two such vectors?
  - Sum of squared differences (SSD)

$$\mathrm{SSD}(\mathbf{u},\mathbf{v}) = \sum_i \left(u_i - v_i\right)^2$$

    - Not invariant to intensity change

  - Normalized correlation

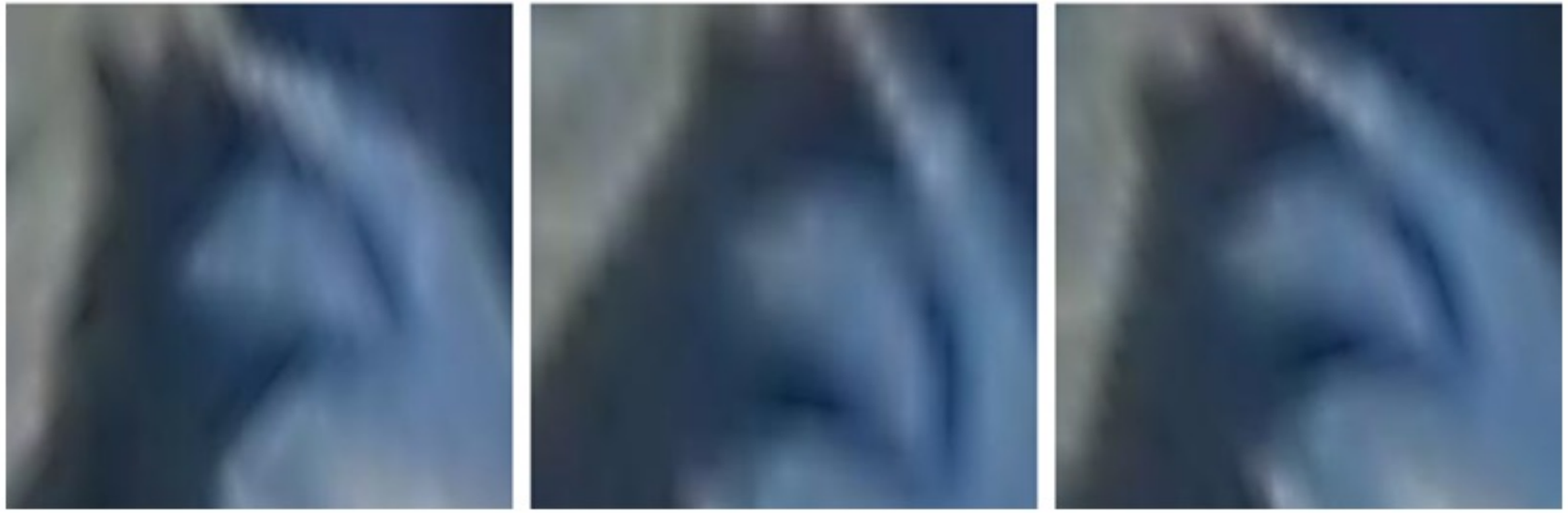
$$\rho(\mathbf{u},\mathbf{v}) = \frac{(\mathbf{u}-\bar{\mathbf{u}})}{\|\mathbf{u}-\bar{\mathbf{u}}\|} \cdot \frac{(\mathbf{v}-\bar{\mathbf{v}})}{\|\mathbf{v}-\bar{\mathbf{v}}\|} = \frac{\sum_i (u_i - \bar{\mathbf{u}})(v_i - \bar{\mathbf{v}})}{\sqrt{\left(\sum_j (u_j - \bar{\mathbf{u}})^2\right)\left(\sum_j (v_j - \bar{\mathbf{v}})^2\right)}}$$

    - Invariant to affine intensity change

Slide from L. Lazebnik.

Disadvantage of intensity vectors as descriptors

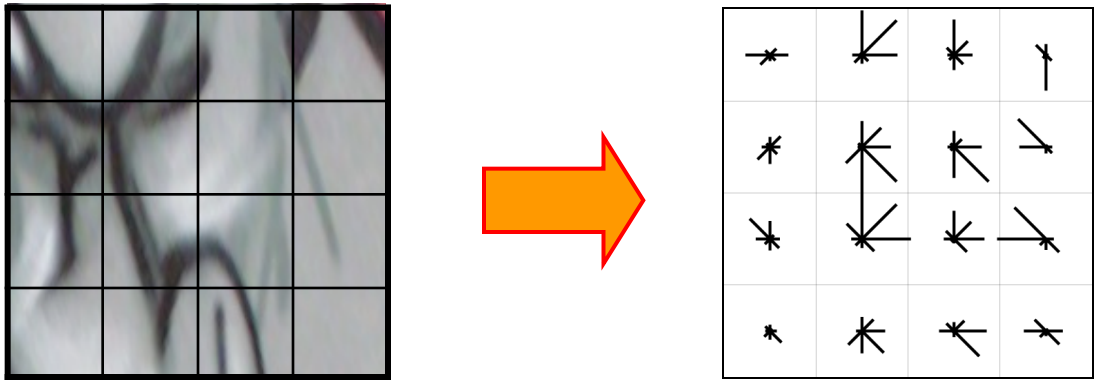
- Small deformations can affect the matching score a lot



Slide from L. Lazebnik.

Feature descriptors: SIFT

- Descriptor computation:
  - Divide patch into 4x4 sub-patches
  - Compute histogram of gradient orientations (8 reference angles) inside each sub-patch
  - Resulting descriptor: 4x4x8 = 128 dimensions



David G. Lowe. "Distinctive image features from scale-invariant keypoints." *IJCV* 60 (2), pp. 91-110, 2004.

Slide from L. Lazebnik.

Feature descriptors: SIFT

- Descriptor computation:
  - Divide patch into 4x4 sub-patches
  - Compute histogram of gradient orientations (8 reference angles) inside each sub-patch
  - Resulting descriptor: 4x4x8 = 128 dimensions

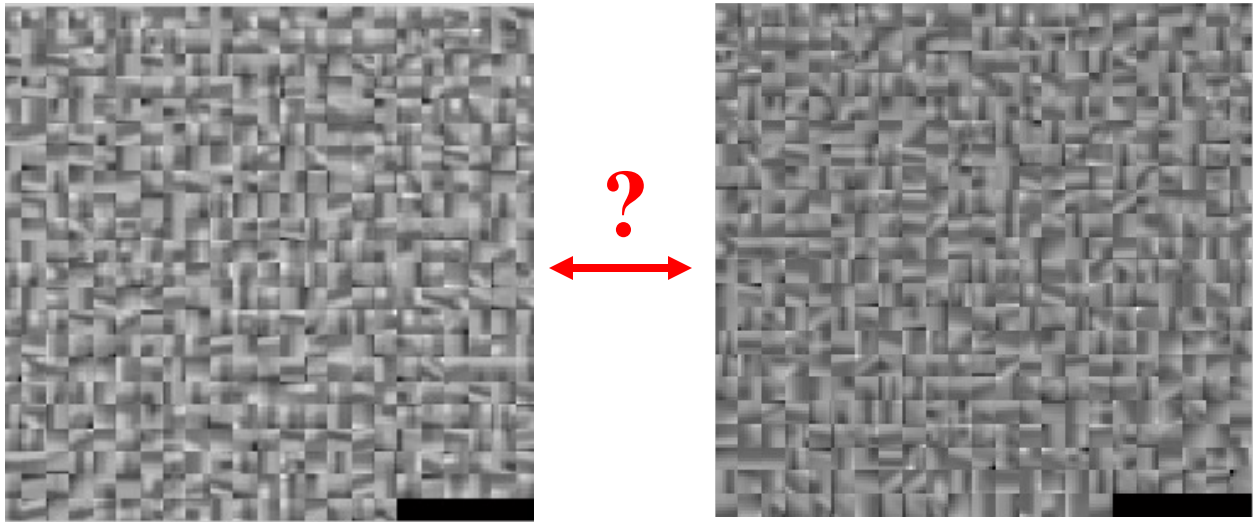
- Advantage over raw vectors of pixel values
  - Gradients less sensitive to illumination change
  - Pooling of gradients over the sub-patches achieves robustness to small shifts, but still preserves some spatial information

David G. Lowe. "Distinctive image features from scale-invariant keypoints." *IJCV* 60 (2), pp. 91-110, 2004.

Slide from L. Lazebnik.

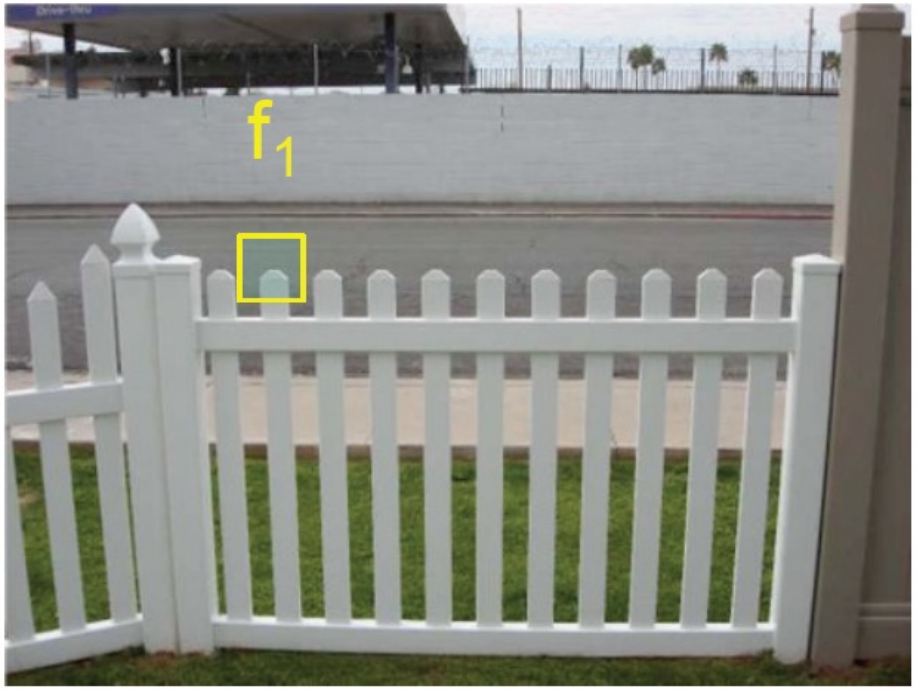
Feature matching

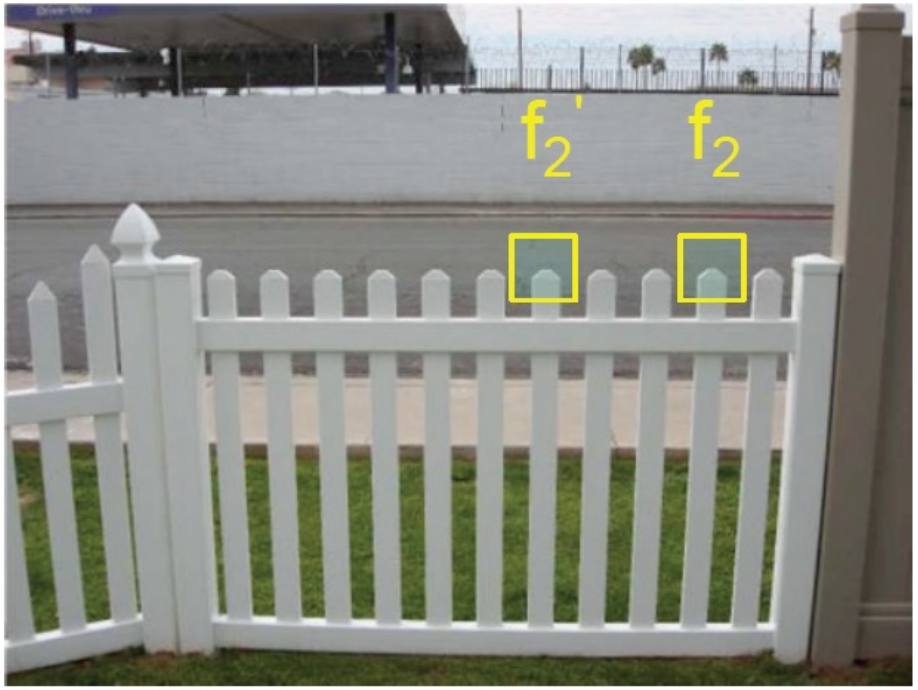
- Generating *putative matches*: for each patch in one image, find a short list of patches in the other image that could match it based solely on appearance



Slide from L. Lazebnik.

Problem: Ambiguous putative matches

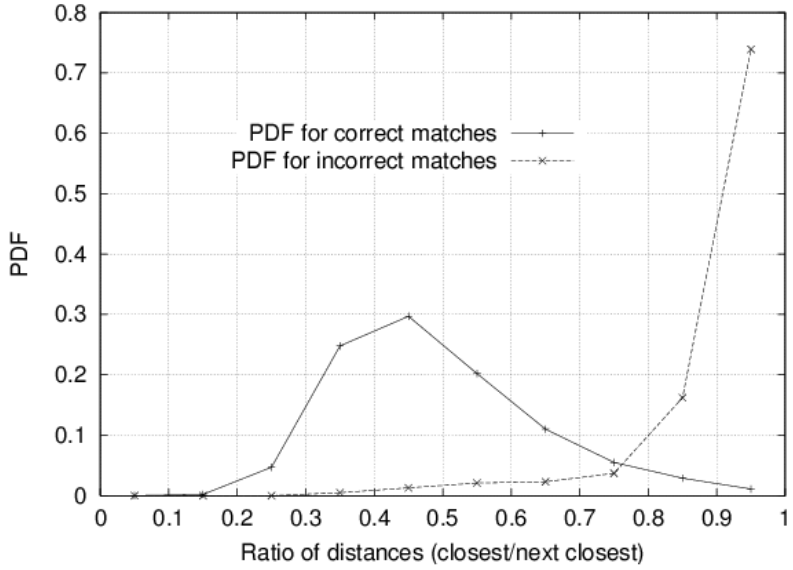




Source: Y. Furukawa

Rejection of unreliable matches

- How can we tell which putative matches are more reliable?
- Heuristic: compare distance of **nearest** neighbor to that of **second** nearest neighbor
  - Ratio of closest distance to second-closest distance will be *high* for features that are *not* distinctive



Threshold of 0.8 provides good separation

David G. Lowe. "Distinctive image features from scale-invariant keypoints." *IJCV* 60 (2), pp. 91-110, 2004.

Slide from L. Lazebnik.

RANSAC

- The set of putative matches contains a very high percentage of outliers

RANSAC loop:

1. Randomly select a *seed group* of matches
2. Compute transformation from seed group
3. Find *inliers* to this transformation
4. If the number of inliers is sufficiently large, re-compute least-squares estimate of transformation on all of the inliers

Keep the transformation with the largest number of inliers

Slide from L. Lazebnik.

Robust feature-based alignment

- Extract features
- Compute *putative matches*
- Loop:
  - *Hypothesize* transformation *T*
  - *Verify* transformation (search for other matches consistent with *T*)

Slide from L. Lazebnik.

Common transformations

original

Transformed

translation

rotation

aspect

affine

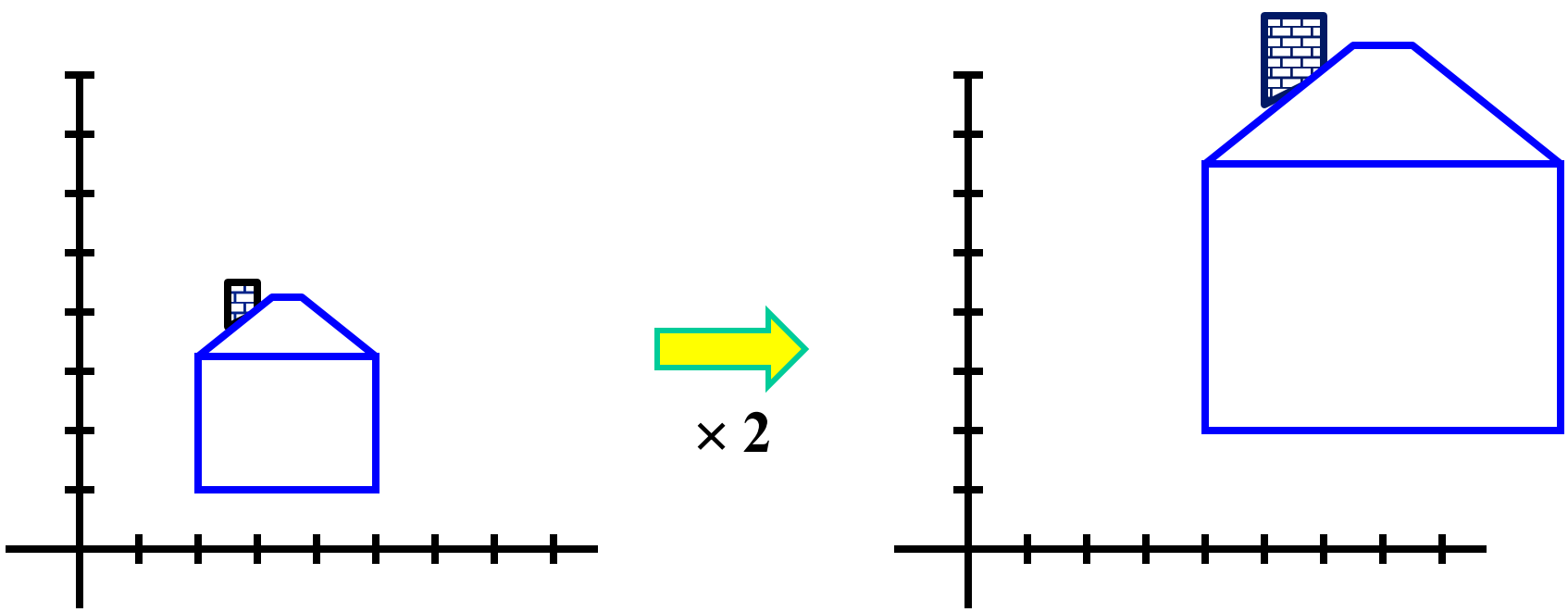
perspective

Slide from A. Efros, S. Seitz, D. Hoiem

Scaling

Scaling a coordinate means multiplying each of its components by a scalar

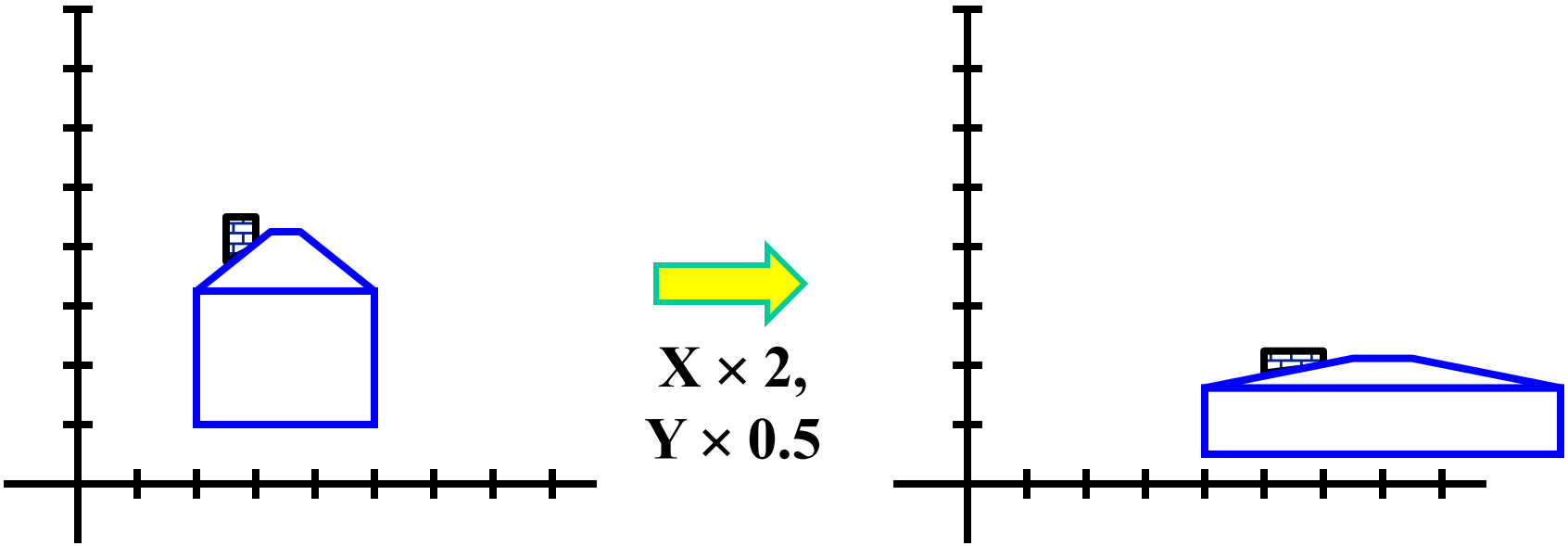
Uniform scaling means this scalar is the same for all components:



Slide from A. Efros, S. Seitz, D. Hoiem

Scaling

Non-uniform scaling: different scalars per component:



Slide from A. Efros, S. Seitz, D. Hoiem

Scaling

Scaling operation:

$$x' = ax$$

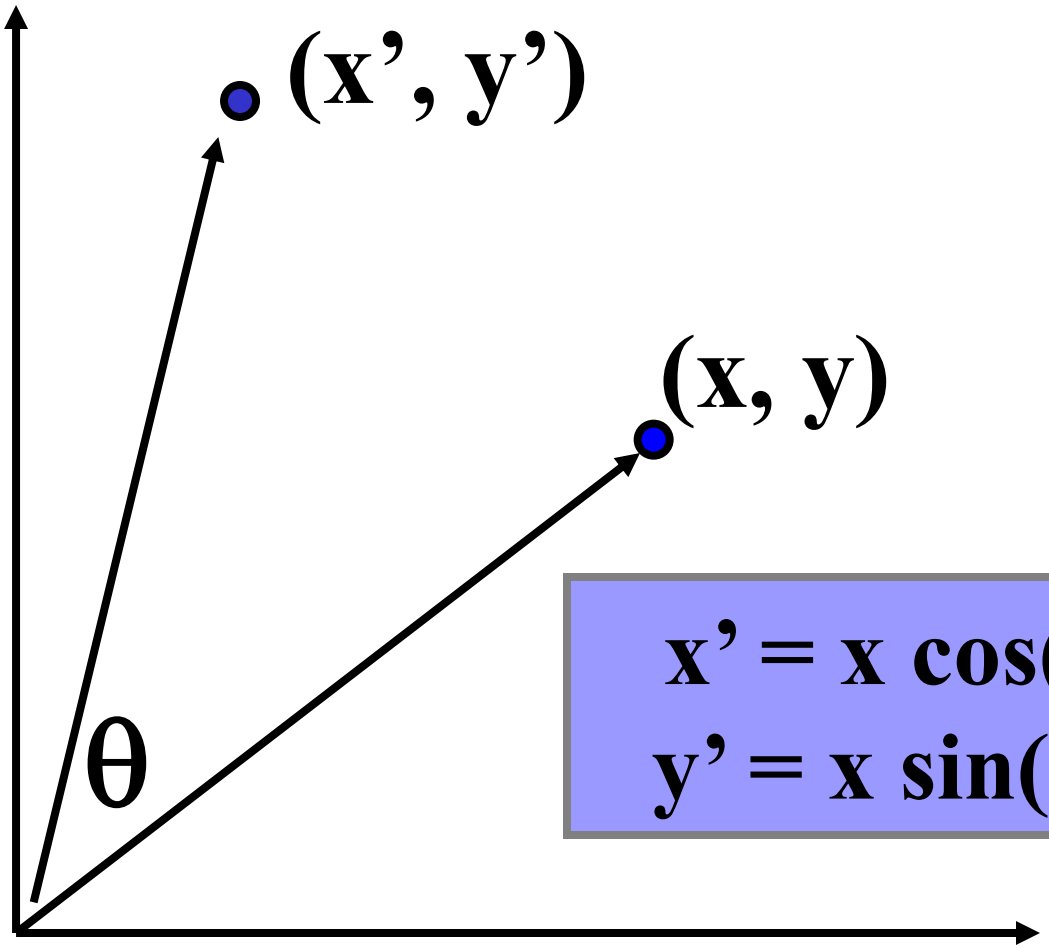
$$y' = by$$

Or, in matrix form:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \underbrace{\begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}}_{\textbf{scaling matrix S}} \begin{bmatrix} x \\ y \end{bmatrix}$$

Slide from A. Efros, S. Seitz, D. Hoiem

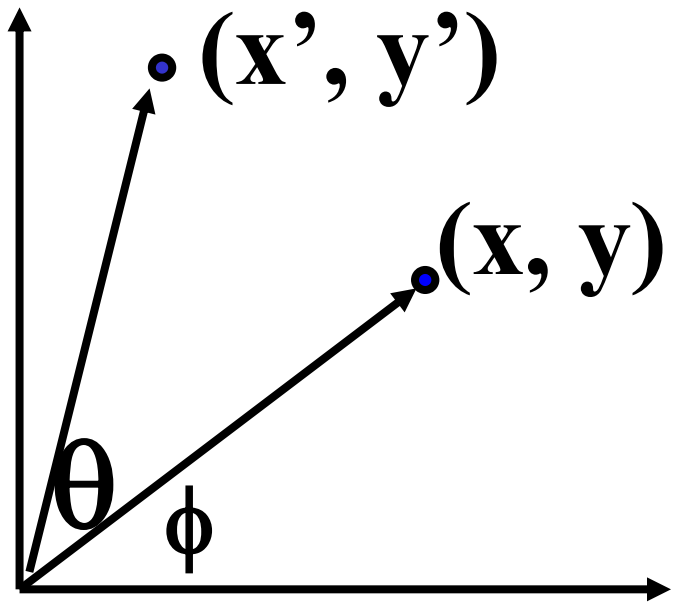
2-D Rotation



$$x' = x \cos(\theta) - y \sin(\theta)$$
$$y' = x \sin(\theta) + y \cos(\theta)$$

Slide from A. Efros, S. Seitz, D. Hoiem

2-D Rotation



Polar coordinates…

$x = r \cos(\phi)$

$y = r \sin(\phi)$

$x' = r \cos(\phi + \theta)$

$y' = r \sin(\phi + \theta)$

Trig Identity…

$x' = r \cos(\phi) \cos(\theta) - r \sin(\phi) \sin(\theta)$

$y' = r \sin(\phi) \cos(\theta) + r \cos(\phi) \sin(\theta)$

Substitute…

$x' = x \cos(\theta) - y \sin(\theta)$

$y' = x \sin(\theta) + y \cos(\theta)$

Slide from A. Efros, S. Seitz, D. Hoiem

2-D Rotation

This is easy to capture in matrix form:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \underbrace{\begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}}_{\mathbf{R}} \begin{bmatrix} x \\ y \end{bmatrix}$$

Even though sin(θ) and cos(θ) are nonlinear functions of θ,

- ***x' is a linear combination of x and y***
- ***y' is a linear combination of x and y***

What is the inverse transformation?

- Rotation by –θ
- For rotation matrices $\mathbf{R}^{-1} = \mathbf{R}^{T}$

Slide from A. Efros, S. Seitz, D. Hoiem

Basic 2D transformations

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Scale

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & \alpha_x \\ \alpha_y & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Shear

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\Theta & -\sin\Theta \\ \sin\Theta & \cos\Theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Rotate

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Translate

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Affine

Affine is any combination of translation, scale, rotation, shear

Slide from A. Efros, S. Seitz, D. Hoiem

Affine Transformations

Affine transformations are combinations of

- Linear transformations, and
- Translations

Properties of affine transformations:

- Lines map to lines
- Parallel lines remain parallel
- Ratios are preserved
- Closed under composition

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

or

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Slide from L. Lazebnik.

Projective Transformations

Projective transformations are combos of

- Affine transformations, and
- Projective warps

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

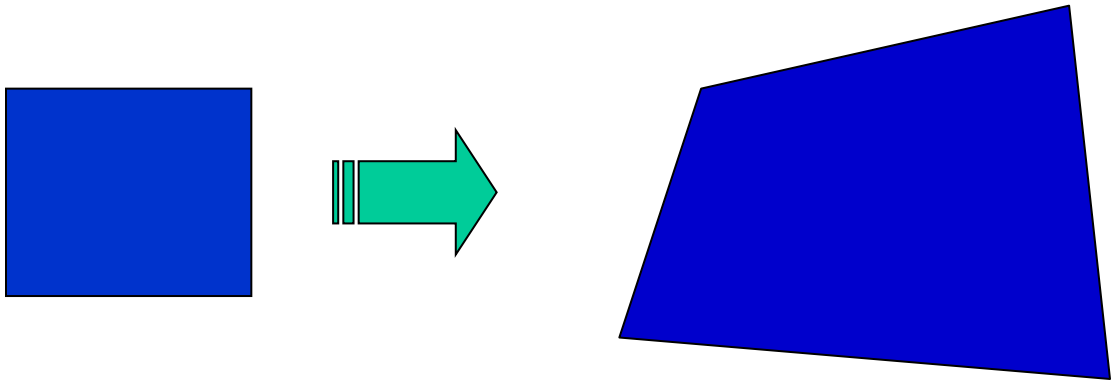
Properties of projective transformations:

- Lines map to lines
- Parallel lines do not necessarily remain parallel
- Ratios are not preserved
- Closed under composition
- **Projective matrix is defined up to a scale (8 DOF)**

Slide from L. Lazebnik.

Fitting a plane projective transformation

- **Homography:** plane projective transformation (transformation taking a quad to another arbitrary quad)

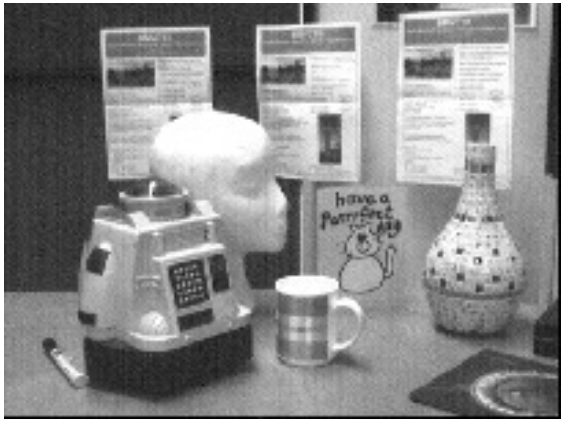


Slide from L. Lazebnik.

Homography

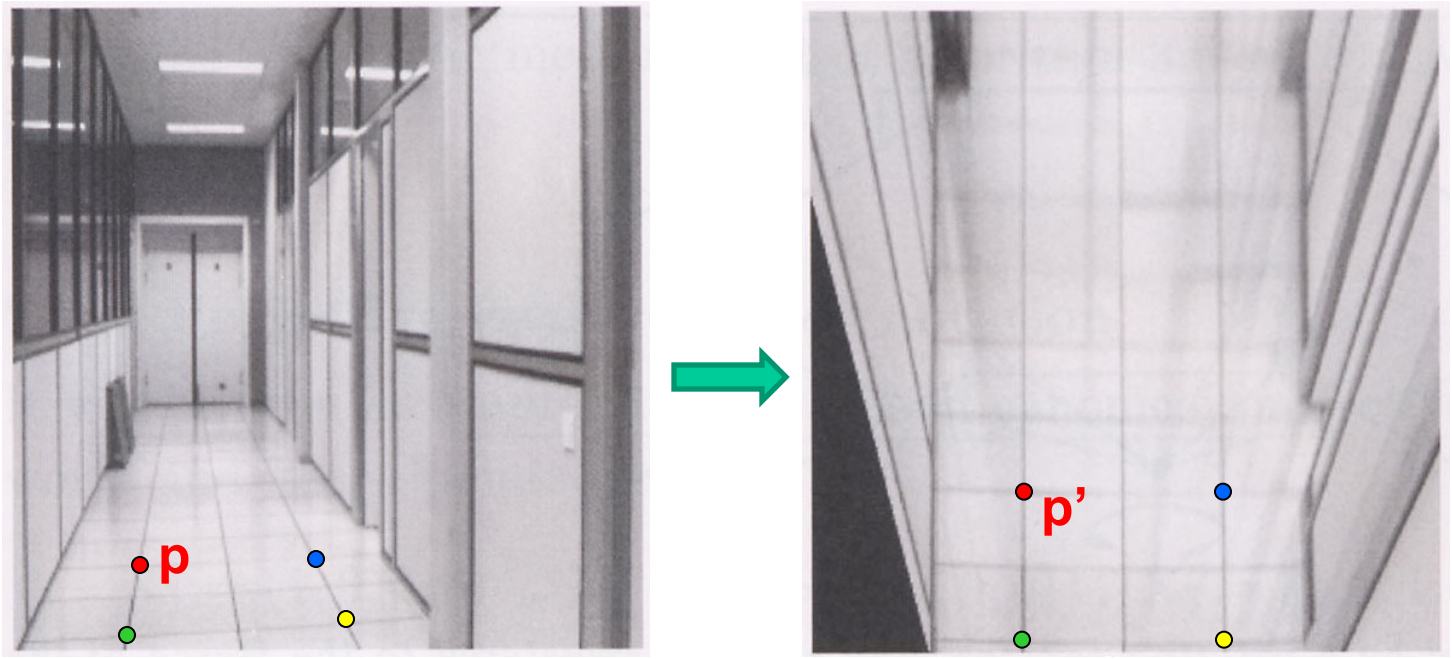
- The transformation between two views of a planar surface


- The transformation between images from two cameras that share the same center

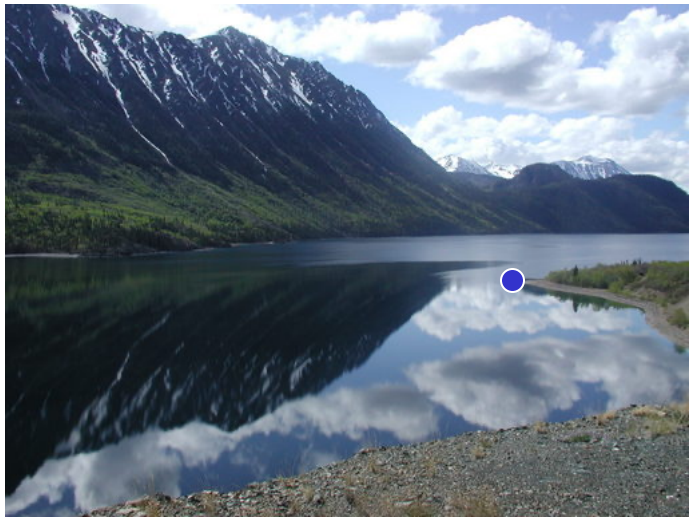
Slide from L. Lazebnik.

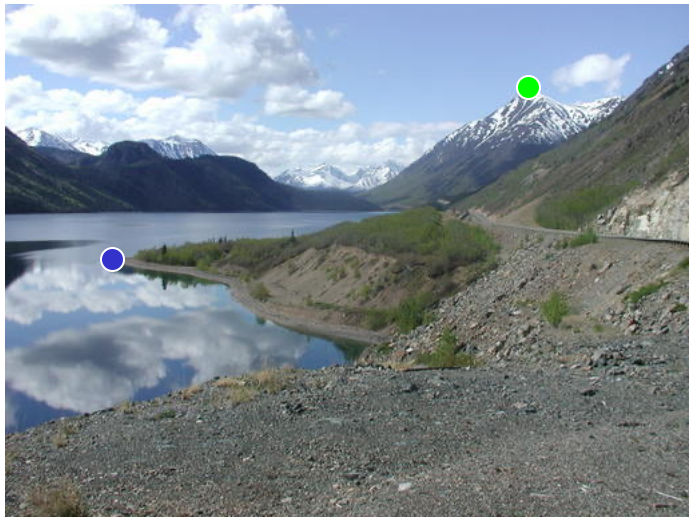
Homography example: Image rectification

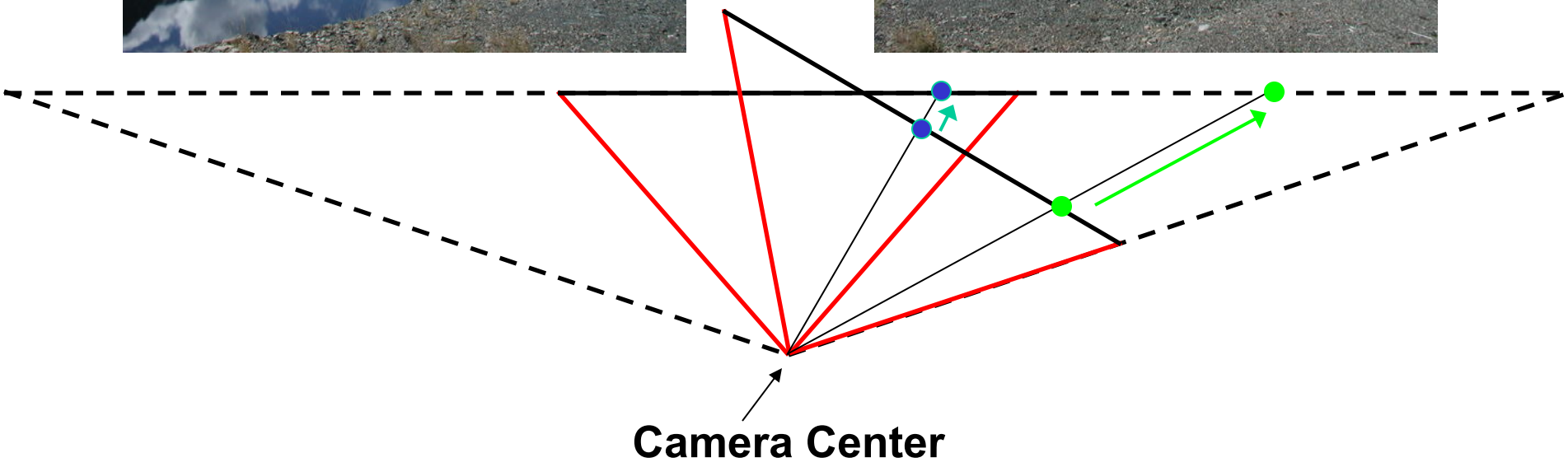


Slide from A. Efros, S. Seitz, D. Hoiem

Homography Example

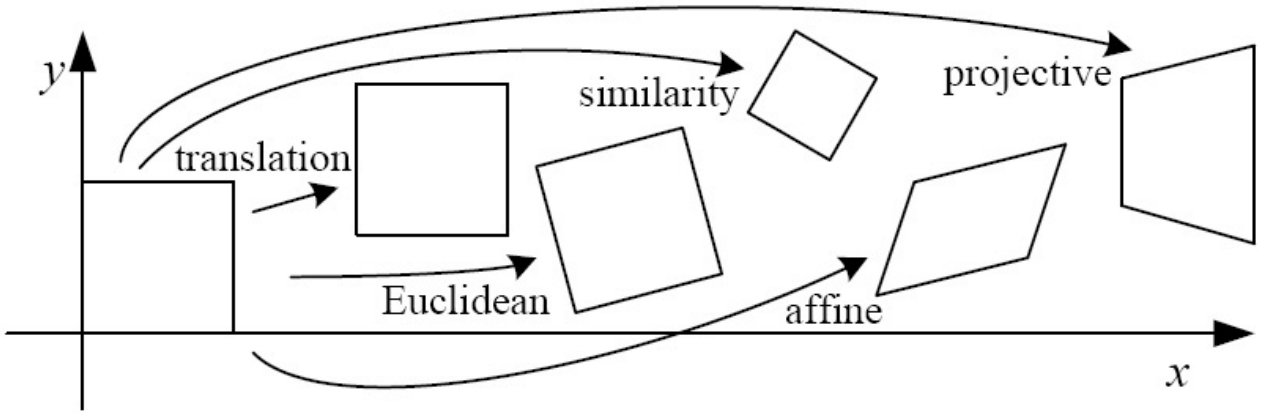






Slide from A. Efros, S. Seitz, D. Hoiem

2D image transformations

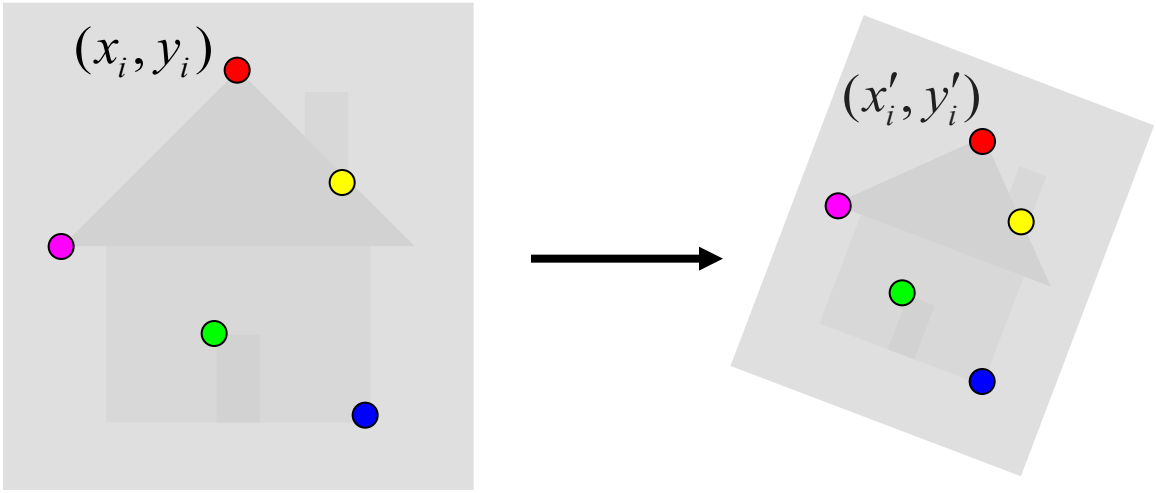


Name	Matrix	# D.O.F.	Preserves:	Icon
translation	$\left[\begin{array}{c\|c} \boldsymbol{I} & \boldsymbol{t} \end{array}\right]_{2\times 3}$	2	orientation $+\cdots$	
rigid (Euclidean)	$\left[\begin{array}{c\|c} \boldsymbol{R} & \boldsymbol{t} \end{array}\right]_{2\times 3}$	3	lengths $+\cdots$	
similarity	$\left[\begin{array}{c\|c} s\boldsymbol{R} & \boldsymbol{t} \end{array}\right]_{2\times 3}$	4	angles $+\cdots$	
affine	$\left[\begin{array}{c} \boldsymbol{A} \end{array}\right]_{2\times 3}$	6	parallelism $+\cdots$	
projective	$\left[\begin{array}{c} \tilde{\boldsymbol{H}} \end{array}\right]_{3\times 3}$	8	straight lines	

Slide from R. Szeliski

Fitting an affine transformation

- Assume we know the correspondences, how do we get the transformation?



$$\begin{bmatrix} x_i' \\ y_i' \end{bmatrix} = \begin{bmatrix} m_1 & m_2 \\ m_3 & m_4 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} + \begin{bmatrix} t_1 \\ t_2 \end{bmatrix}$$

$$\mathbf{x}_i' = \mathbf{M}\mathbf{x}_i + \mathbf{t}$$

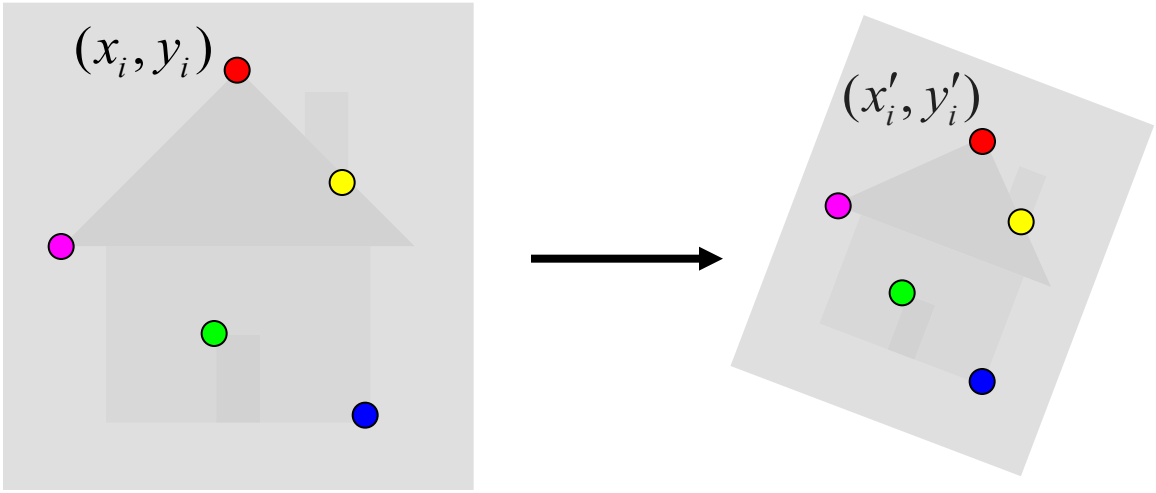
Want to find **M**, **t** to minimize

$$\sum_{i=1}^{n} || \mathbf{x}_i' - \mathbf{M}\mathbf{x}_i - \mathbf{t} ||^2$$

Slide from L. Lazebnik.

Fitting an affine transformation

- Assume we know the correspondences, how do we get the transformation?



$$\begin{bmatrix} x_i' \\ y_i' \end{bmatrix} = \begin{bmatrix} m_1 & m_2 \\ m_3 & m_4 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} + \begin{bmatrix} t_1 \\ t_2 \end{bmatrix}$$

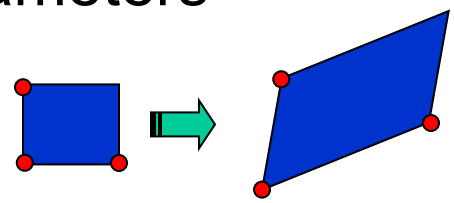
$$\begin{bmatrix} & & & & & \\ & & & & & \\ & & & & & \\ & & & & & \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ t_1 \\ t_2 \end{bmatrix} = \begin{bmatrix} \cdots \\ x_i' \\ y_i' \\ \cdots \end{bmatrix}$$

Slide from L. Lazebnik.

Fitting an affine transformation

$$\begin{bmatrix} & & \cdots & & & \\ x_i & y_i & 0 & 0 & 1 & 0 \\ 0 & 0 & x_i & y_i & 0 & 1 \\ & & \cdots & & & \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ t_1 \\ t_2 \end{bmatrix} = \begin{bmatrix} \cdots \\ x_i' \\ y_i' \\ \cdots \end{bmatrix}$$

- Linear system with six unknowns
- Each match gives us two linearly independent equations: need at least three to solve for the transformation parameters



Slide from L. Lazebnik.

Fitting a homography

- Recall: homogeneous coordinates

$$(x, y) \Rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Converting *to* homogeneous image coordinates

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} \Rightarrow (x/w, y/w)$$

Converting *from* homogeneous image coordinates

Slide from L. Lazebnik.

Fitting a homography

- Recall: homogeneous coordinates

$$(x, y) \Rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Converting *to* homogeneous image coordinates

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} \Rightarrow (x/w, y/w)$$

Converting *from* homogeneous image coordinates

- Equation for homography:

$$\lambda \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Slide from L. Lazebnik.

Fitting a homography

- Equation for homography:

$$\lambda \begin{bmatrix} x_i' \\ y_i' \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \qquad \lambda \mathbf{x}_i' = \mathbf{H}\mathbf{x}_i$$

$$\mathbf{x}_i' \times \mathbf{H}\mathbf{x}_i = 0$$

$$\begin{bmatrix} x_i' \\ y_i' \\ 1 \end{bmatrix} \times \begin{bmatrix} \mathbf{h_1}^T \mathbf{x}_i \\ \mathbf{h_2}^T \mathbf{x}_i \\ \mathbf{h_3}^T \mathbf{x}_i \end{bmatrix} = \begin{bmatrix} y_i' \mathbf{h}_3^T \mathbf{x}_i - \mathbf{h}_2^T \mathbf{x}_i \\ \mathbf{h}_1^T \mathbf{x}_i - x_i' \mathbf{h}_3^T \mathbf{x}_i \\ x_i' \mathbf{h}_2^T \mathbf{x}_i - y_i' \mathbf{h}_1^T \mathbf{x}_i \end{bmatrix}$$

$$\begin{bmatrix} 0^T & -\mathbf{x}_i^T & y_i' \mathbf{x}_i^T \\ \mathbf{x}_i^T & 0^T & -x_i' \mathbf{x}_i^T \\ -y_i' \mathbf{x}_i^T & x_i' \mathbf{x}_i^T & 0^T \end{bmatrix} \begin{pmatrix} \mathbf{h}_1 \\ \mathbf{h}_2 \\ \mathbf{h}_3 \end{pmatrix} = 0$$

3 equations, only 2 linearly independent

Slide from L. Lazebnik.

Fitting a homography

$$\begin{bmatrix} 0^T & \mathbf{x}_1^T & -y_1' \mathbf{x}_1^T \\ \mathbf{x}_1^T & 0^T & -x_1' \mathbf{x}_1^T \\ \cdots & \cdots & \cdots \\ 0^T & \mathbf{x}_n^T & -y_n' \mathbf{x}_n^T \\ \mathbf{x}_n^T & 0^T & -x_n' \mathbf{x}_n^T \end{bmatrix} \begin{pmatrix} \mathbf{h}_1 \\ \mathbf{h}_2 \\ \mathbf{h}_3 \end{pmatrix} = 0 \qquad \mathbf{A}\,\mathbf{h} = 0$$

- H has 8 degrees of freedom (9 parameters, but scale is arbitrary)
- One match gives us two linearly independent equations
- Homogeneous least squares: find $\mathbf{h}$ minimizing $||\mathbf{A}\mathbf{h}||^2$
  - Eigenvector of $\mathbf{A}^\mathbf{T}\mathbf{A}$ corresponding to smallest eigenvalue
  - Four matches needed for a minimal solution

Slide from L. Lazebnik.

Feature-based alignment: Overview

- Alignment as fitting
  - Affine transformations
  - Homographies
- Robust alignment
  - Descriptor-based feature matching
  - RANSAC
- Applications

Slide from L. Lazebnik