CSE 152A: Computer Vision

Manmohan Chandraker

Lecture 7: Matching

Overall goals for the course

- Introduce fundamental concepts in computer vision
- Enable one or all of several such outcomes
  - Pursue higher studies in computer vision
  - Join industry to do cutting-edge work in computer vision
  - Gain appreciation of modern computer vision technologies
- Engage in discussions and interaction
- This is a great time to study computer vision!

Course Details

Course details

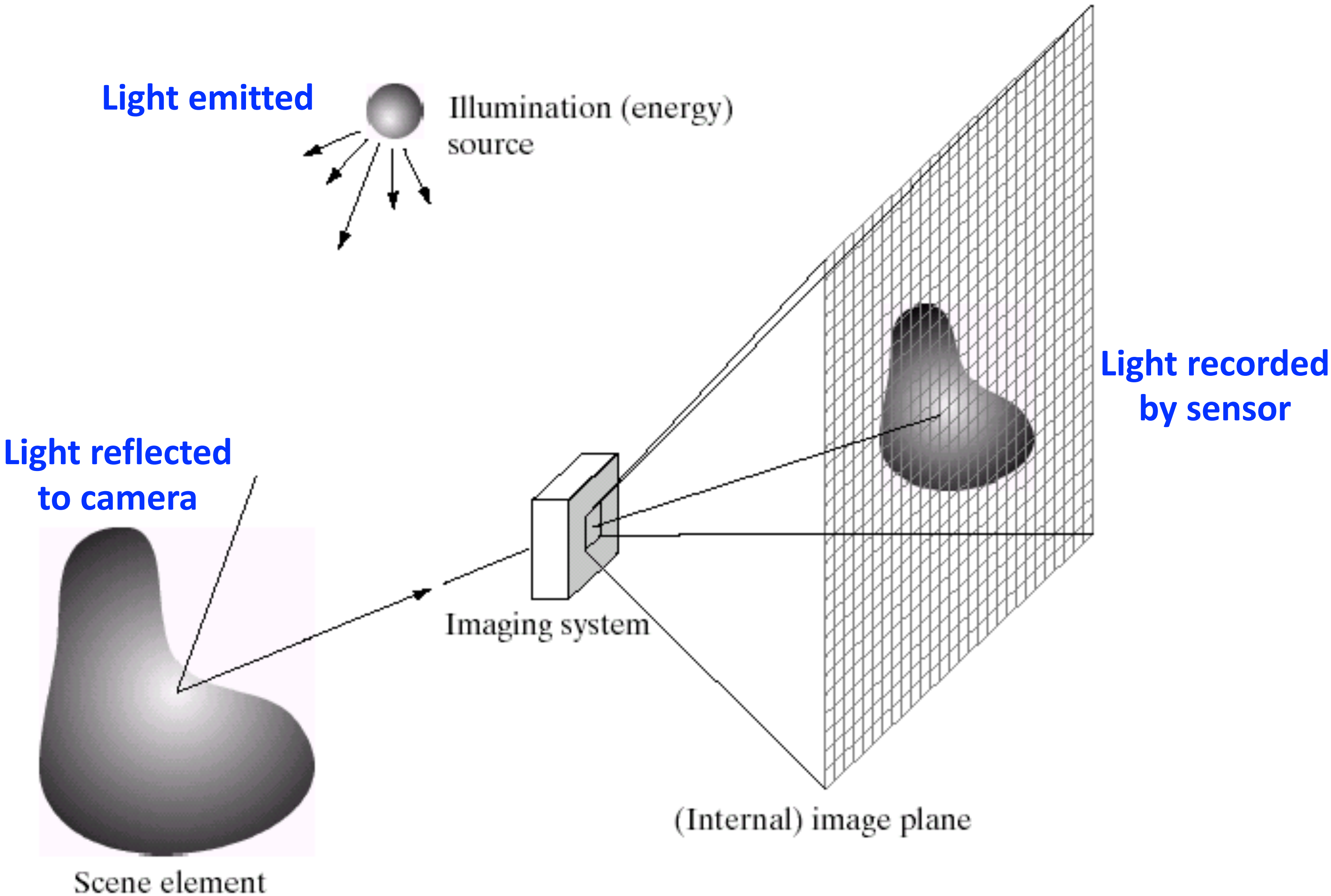
- Class webpage:
  - https://cseweb.ucsd.edu/~mkchandraker/classes/CSE152A/Winter2024/
- Instructor email:
  - mkchandraker@ucsd.edu
- Grading
  - 35% final exam
  - 40% homework assignments
  - 20% mid-term
  - 5% self-study exercise
  - Ungraded quizzes
- Aim is to learn together, discuss and have fun!

Course details

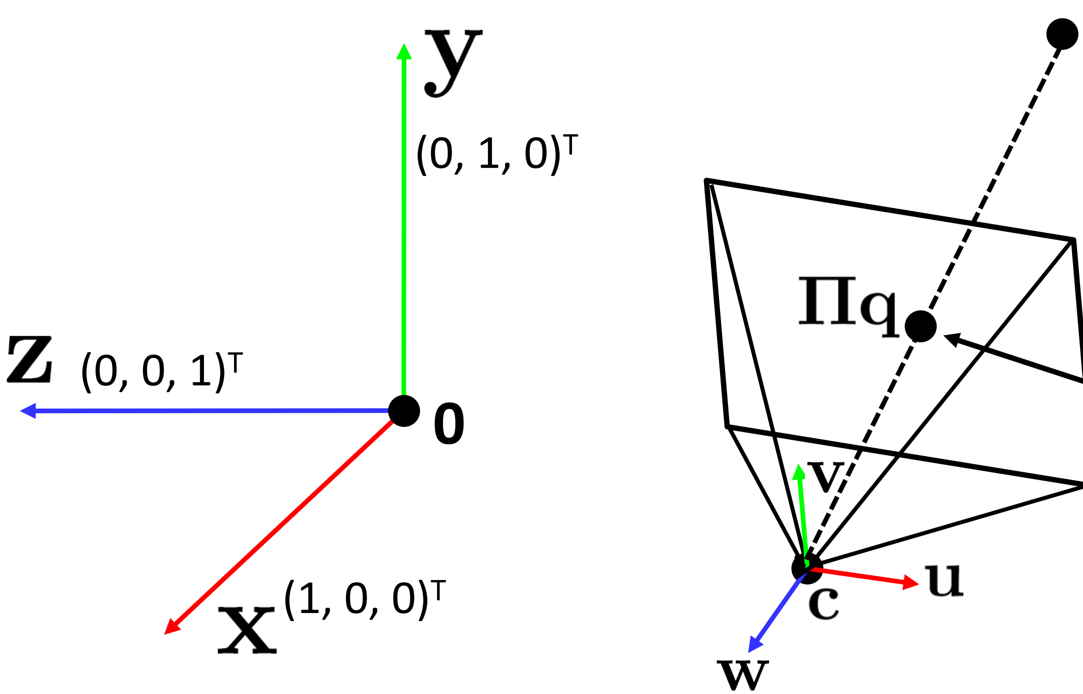
- TAs
  - Nicholas Chua: nchua@ucsd.edu
  - Tarun Kalluri: sskallur@ucsd.edu
  - Sreyas Ravichandran: srravichandran@ucsd.edu
- Tutors
  - Kun Wang, Kevin Chan, Zixian Wang: {kuw010, tsc003, ziw081}@ucsd.edu
- Discussion section: M 3-3:50pm
- TA office hours and tutor hours to be posted on webpage
- Piazza for questions and discussions:
  - https://piazza.com/ucsd/winter2024/cse152a

Recap

Photometric: Modeling appearance



Geometric: Modeling projection



$\mathbf{q} = (x, y, z, 1)$

(3D point in homogeneous coordinates, a vector of length 4)

$$\mathbf{\Pi} = \mathbf{K} \left[\ \mathbf{R} \ | \ -\mathbf{Rc} \ \right]$$

(2D image in homogeneous coordinates, a vector of length 3)

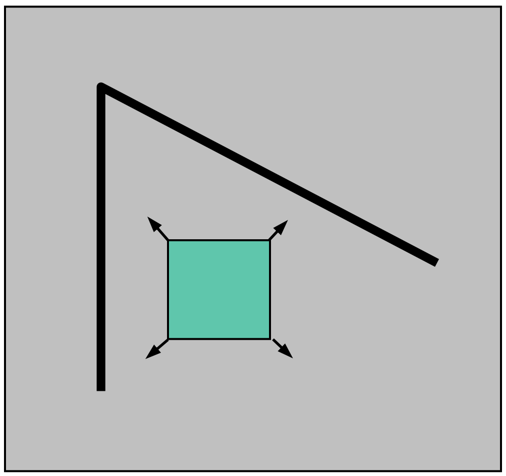
$$\left(\frac{(\mathbf{\Pi q})_1}{(\mathbf{\Pi q})_3} , \frac{(\mathbf{\Pi q})_2}{(\mathbf{\Pi q})_3} \right)$$

(Usual Cartesian coordinates, a vector of length 2)

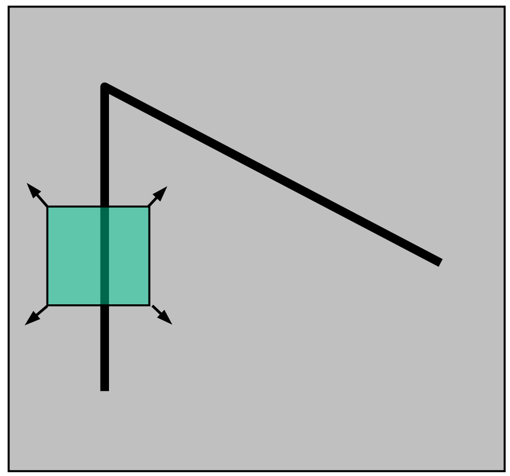
Feature detection

Local measure of feature uniqueness

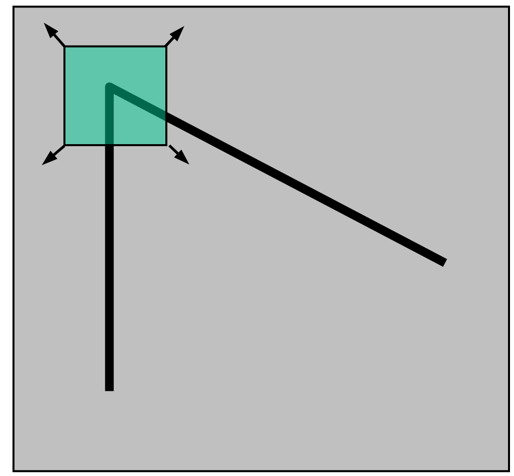
- How does the window change when you shift it?
- Shifting the window in *some direction* causes a *big change*



"flat" region:
no change in all directions



"edge": large change perpendicular to the edge direction

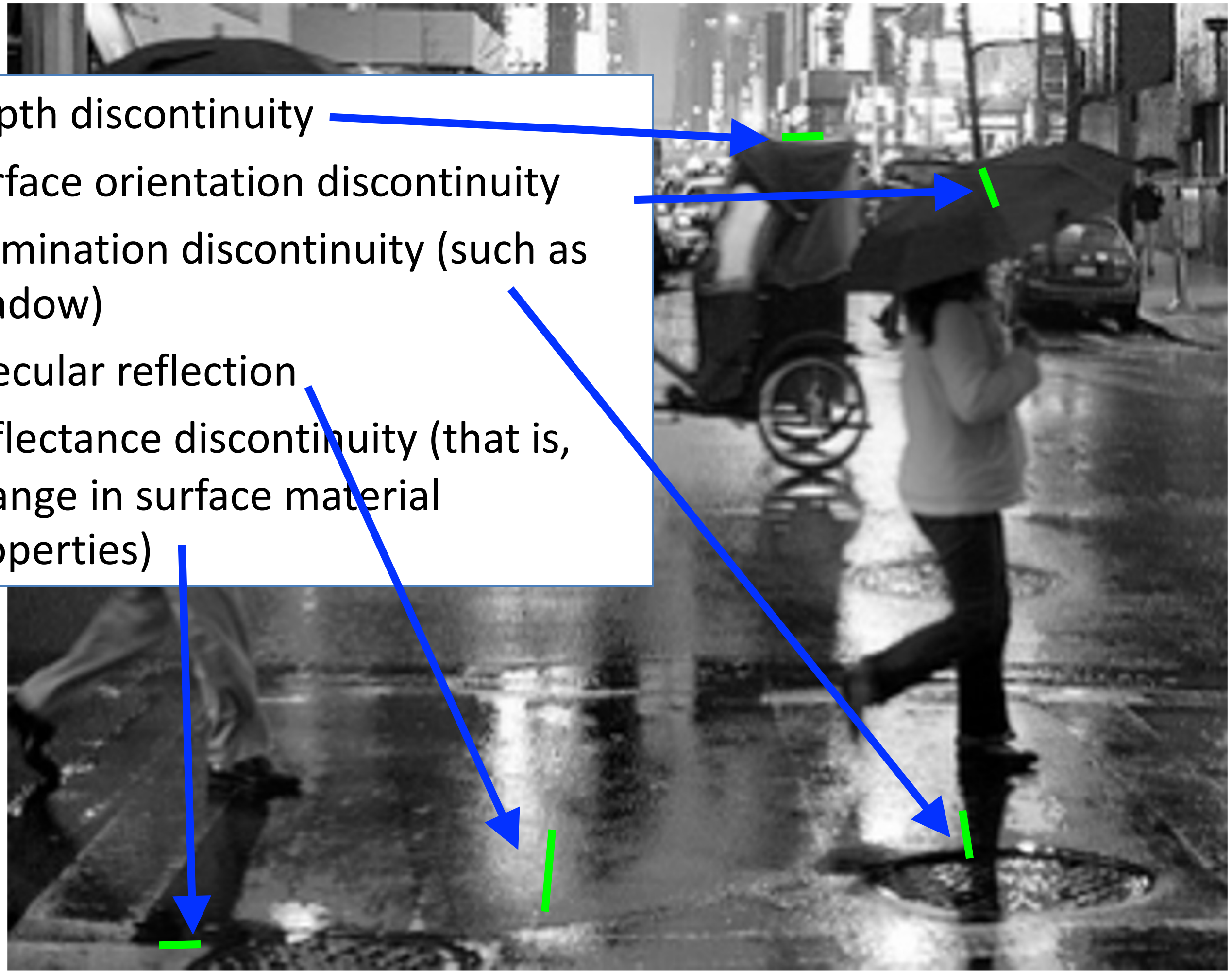


"corner": large change in all directions

[Darya Frolova, Denis Simakov, Weizmann Institute]

Edges in Natural Images

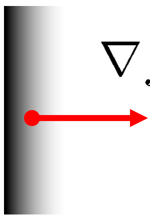
- Depth discontinuity
- Surface orientation discontinuity
- Illumination discontinuity (such as shadow)
- Specular reflection
- Reflectance discontinuity (that is, change in surface material properties)



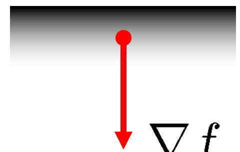
Source: Photografr.com

Edge Detection with Image Gradients

- Gradient represents direction of most rapid change in intensity

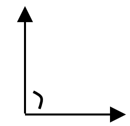


$$\nabla f = \left[\frac{\partial f}{\partial x}, 0\right]$$



$$\nabla f = \left[0, \frac{\partial f}{\partial y}\right]$$

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



- The gradient encodes *edge strength* and *edge direction* as

$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2} \qquad \theta = \tan^{-1}\left(\frac{\partial f}{\partial y} \Big/ \frac{\partial f}{\partial x}\right)$$

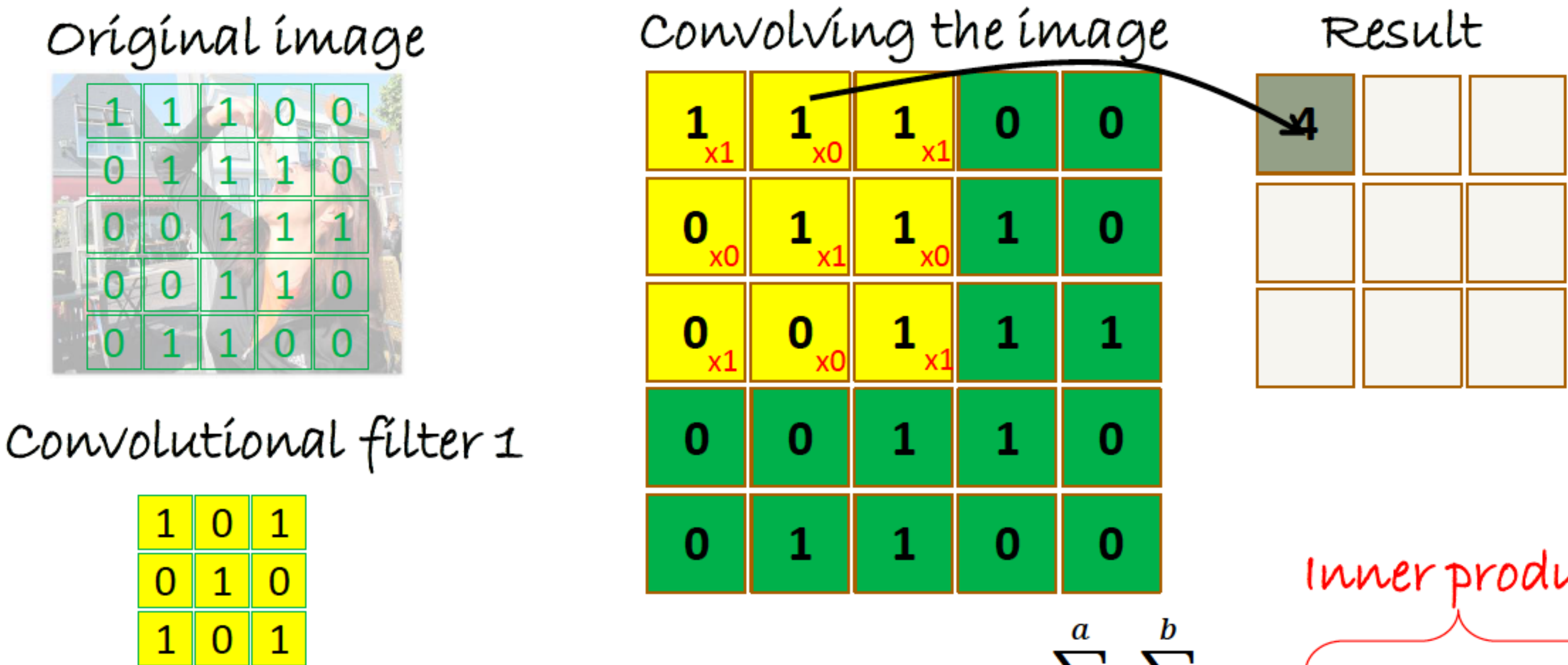
- Can efficiently compute gradient using convolutions

$$\boldsymbol{K}_x = \frac{1}{2}\begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & -1 \\ 0 & 0 & 0 \end{bmatrix} \qquad \boldsymbol{K}_y = \frac{1}{2}\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix}$$

- Sobel operator is often used in practice

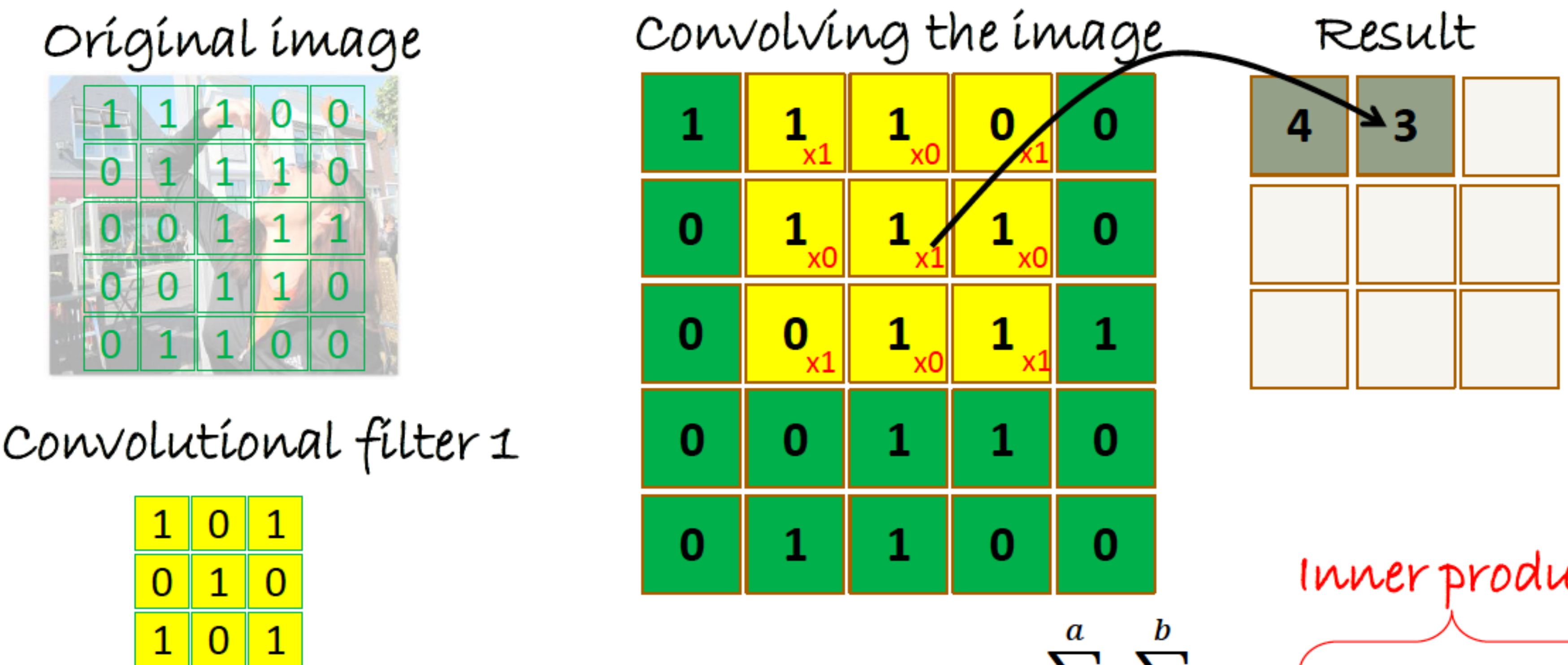
$$\boldsymbol{K}_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} \qquad \boldsymbol{K}_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Spatial filtering is convolution



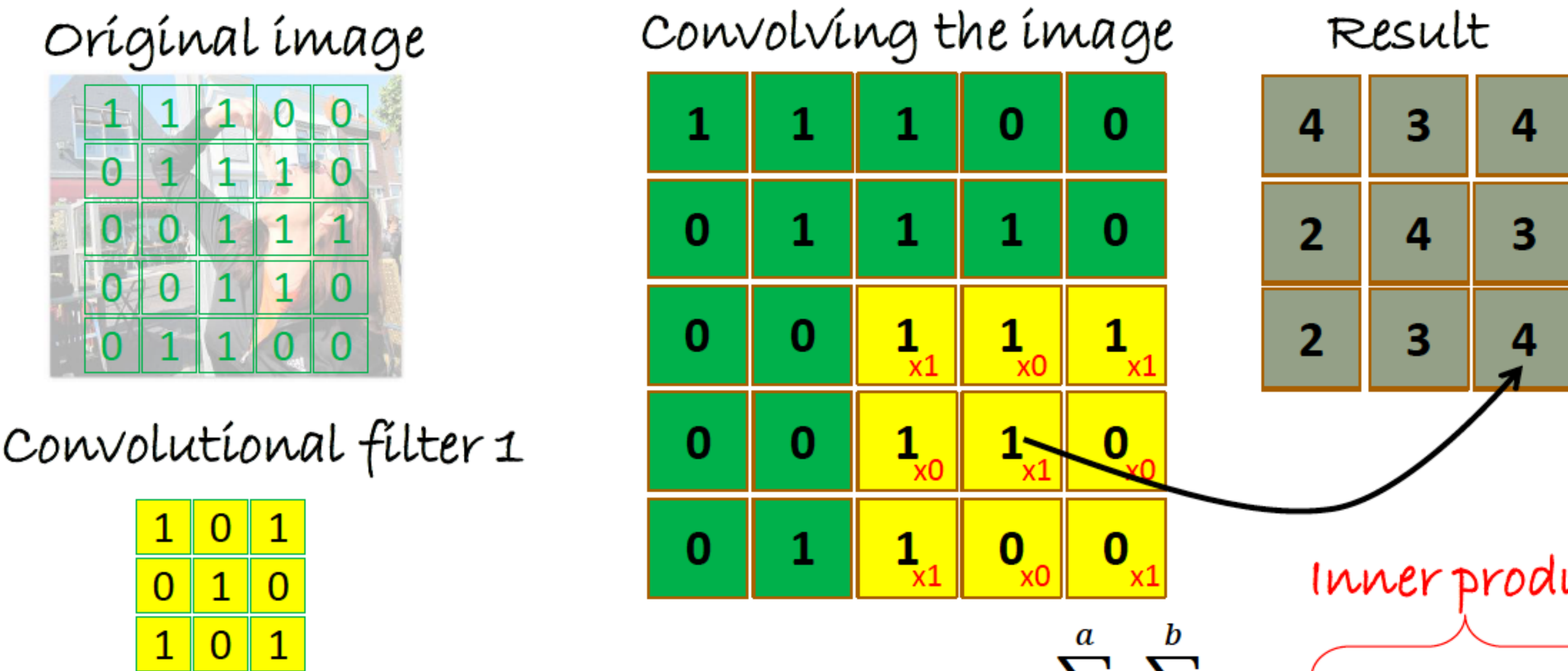
$$I(x,y) * h = \sum_{i=-a}^{a} \sum_{j=-b}^{b} I(\underbrace{x-i, y-j) \cdot h(i,j)}_{\text{Inner product}}$$

Spatial filtering is convolution



$$I(x,y) * h = \sum_{i=-a}^{a} \sum_{j=-b}^{b} \overbrace{I(x-i, y-j) \cdot h(i,j)}^{\text{Inner product}}$$

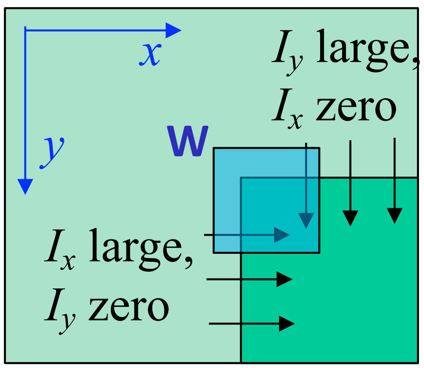
Spatial filtering is convolution



$$I(x,y) * h = \sum_{i=-a}^{a} \sum_{j=-b}^{b} \overbrace{I(x-i, y-j) \cdot h(i,j)}^{\text{Inner product}}$$

Harris Corner Detector

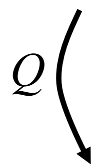
First, consider the second moment matrix for a simpler case:



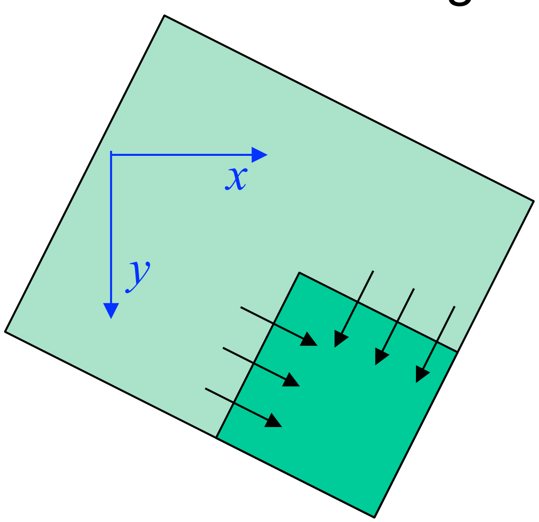
Sum over a small window W around hypothetical corner

$$C = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$

This means dominant gradient directions align with x or y axis.



In the general case, since C is symmetric, it can be shown:



$$C = Q^{-1} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} Q$$

Eigenvalues

Rotation

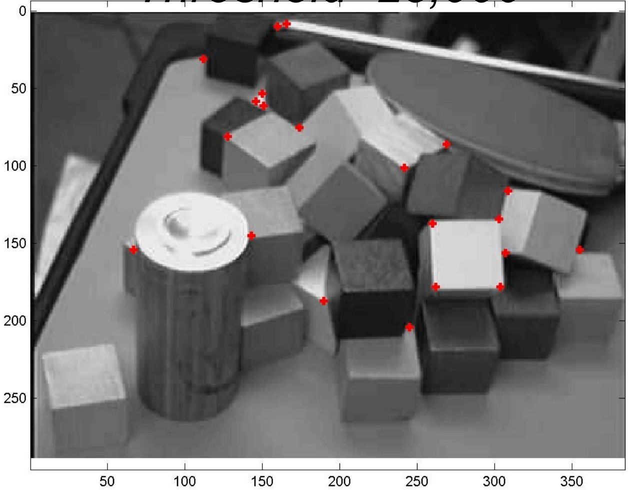
If either λ close to 0, then **not** a corner, so seek locations where both large.

Simple Corner Detector Implementation

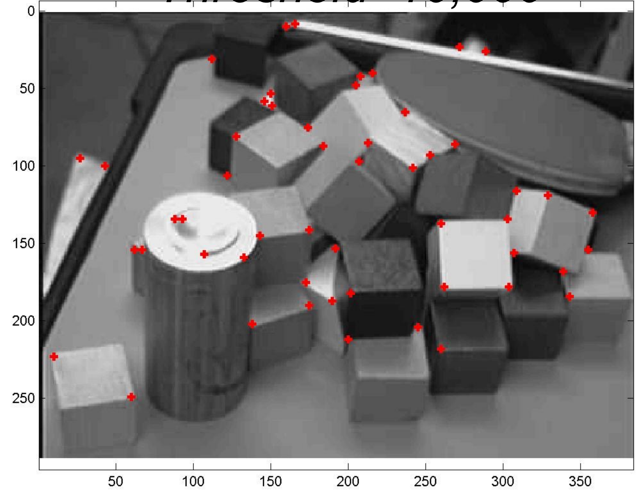
- Run a small window over an image and compute spatial gradient matrix ***C*** at every pixel
- Compute the minor eigenvalue of ***C*** at every pixel to obtain the corner response "image" ***R***
- Apply nonmaximal suppression to the "image" ***R***
  - Divide into grid, choose maximum within each grid cell
  - Resulting image ***R***' has only one corner candidate per grid cell
  - Prevents corners from being too close to each other
- Threshold resulting image ***R'*** using a global threshold *T*
  - Corners at pixels (*x*, *y*) corresponding to ***R***'(*x*, *y*) > *T*

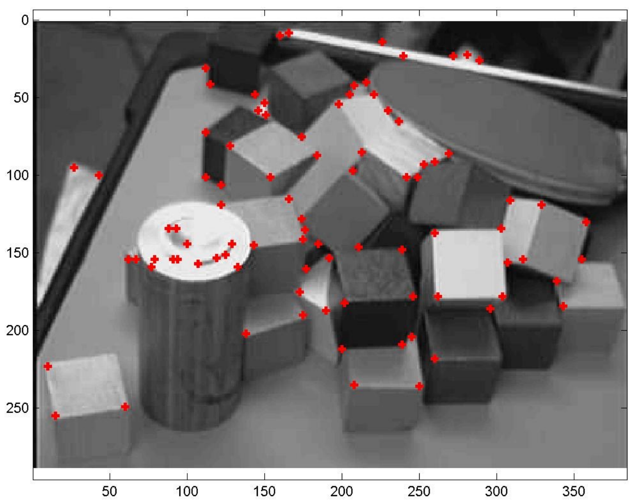
Simple Corner Detector: Outputs

Threshold=25,000



Threshold=10,000





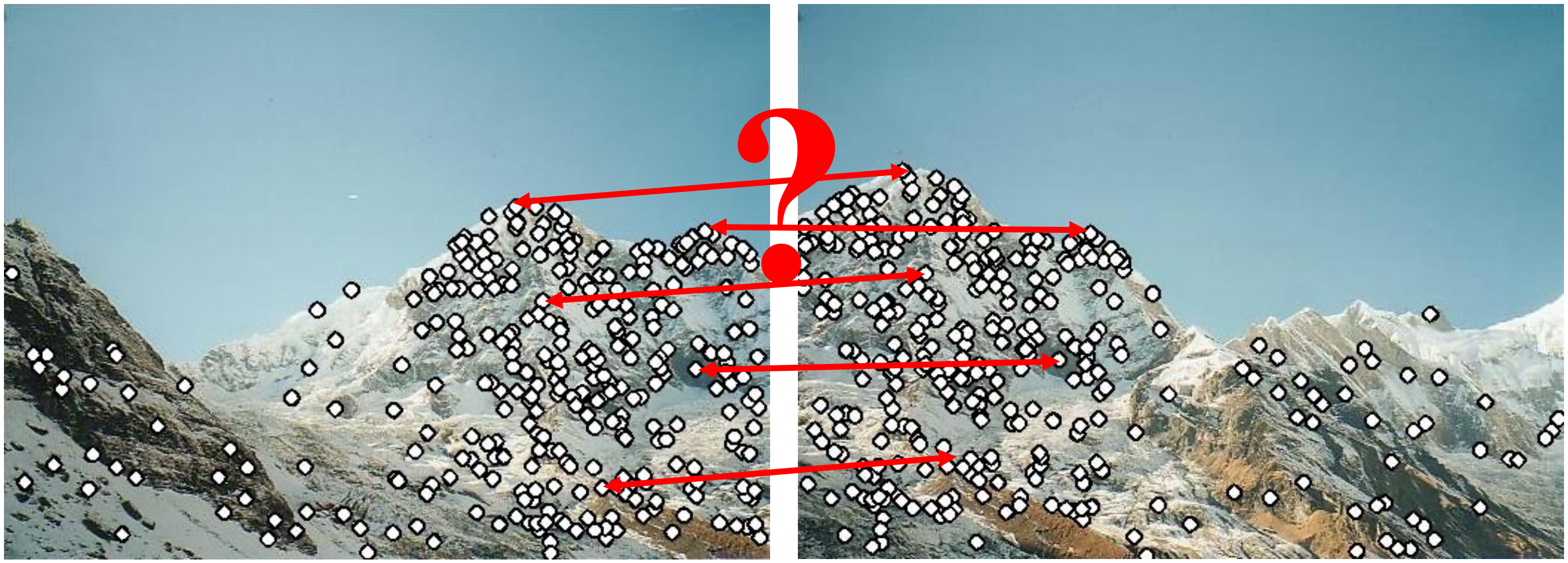
Threshold=5,000

Matching

Feature Descriptors

We know how to detect good points

Next question: **How to match them?**

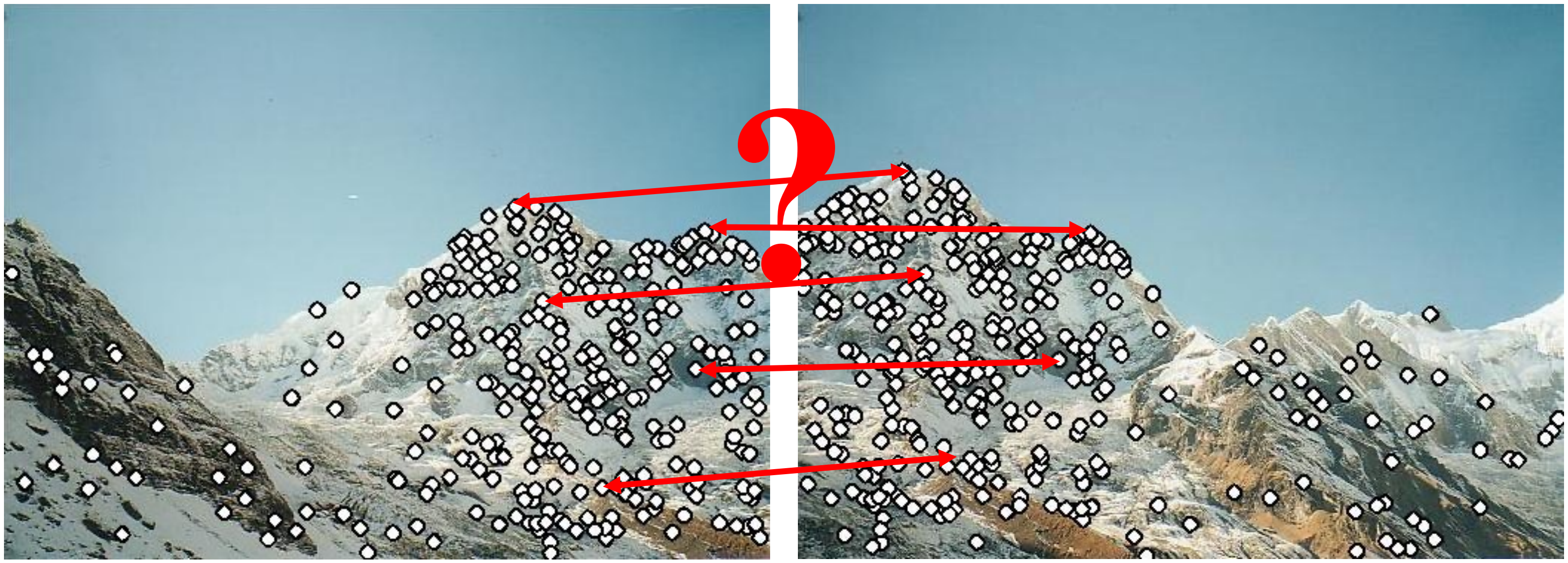


[Slide courtesy: Rick Szeliski]

Feature Descriptors

We know how to detect good points

Next question: **How to match them?**



Lots of possibilities (this is a popular research area)

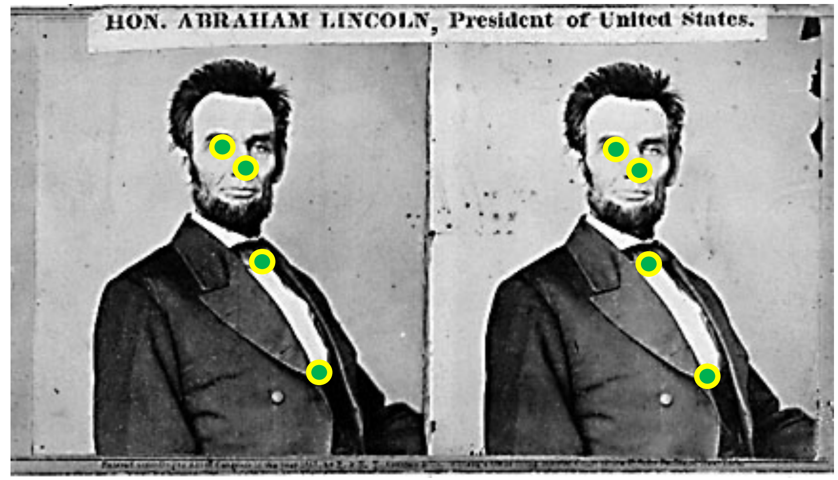
- Simple option: match square windows around the point
- State of the art: SIFT (http://www.cs.ubc.ca/~lowe/keypoints/)

[Slide courtesy: Rick Szeliski]

Simple matching methods

Interest point:

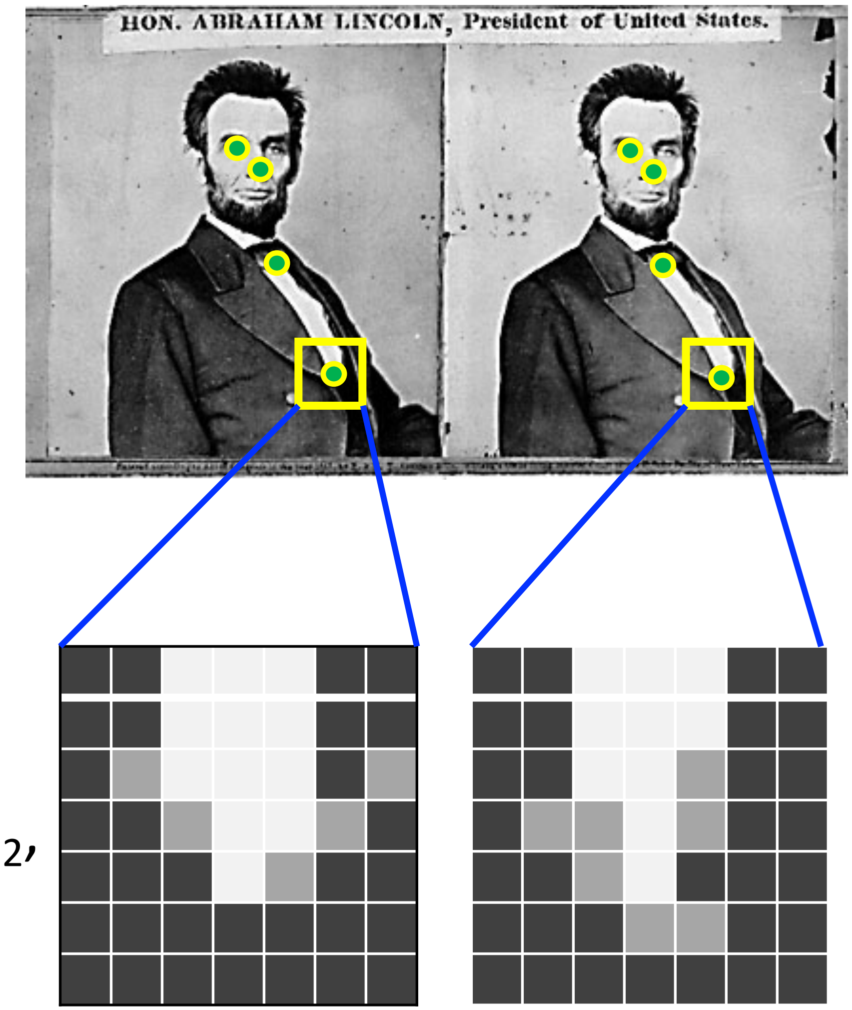
- Localized position
- Informative about image content
- Repeatable under variations



Simple matching methods

Interest point:

- Localized position
- Informative about image content
- Repeatable under variations



Descriptor:

- *Function* applied on each W_1 and W_2, to enable *comparing* them

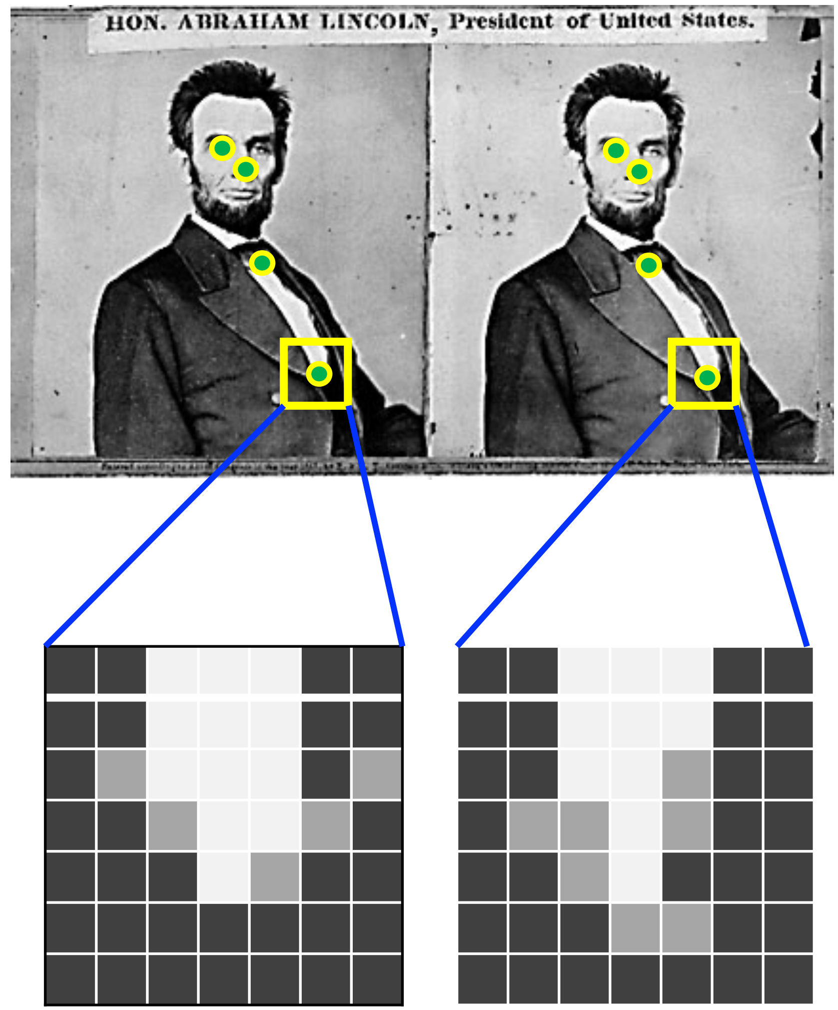
$W_1(x, y)$: k x k pixel patch in image 1

$W_2(x, y)$: k x k pixel patch in image 2

Simple matching methods

- SSD (Sum of Squared Differences)

$$\sum_{x,y} |W_1(x,y) - W_2(x,y)|^2$$



$W_1(x, y)$: k x k pixel patch in image 1

$W_2(x, y)$: k x k pixel patch in image 2

Simple matching methods

- SSD (Sum of Squared Differences)

$$\sum_{x,y} |W_1(x,y) - W_2(x,y)|^2$$

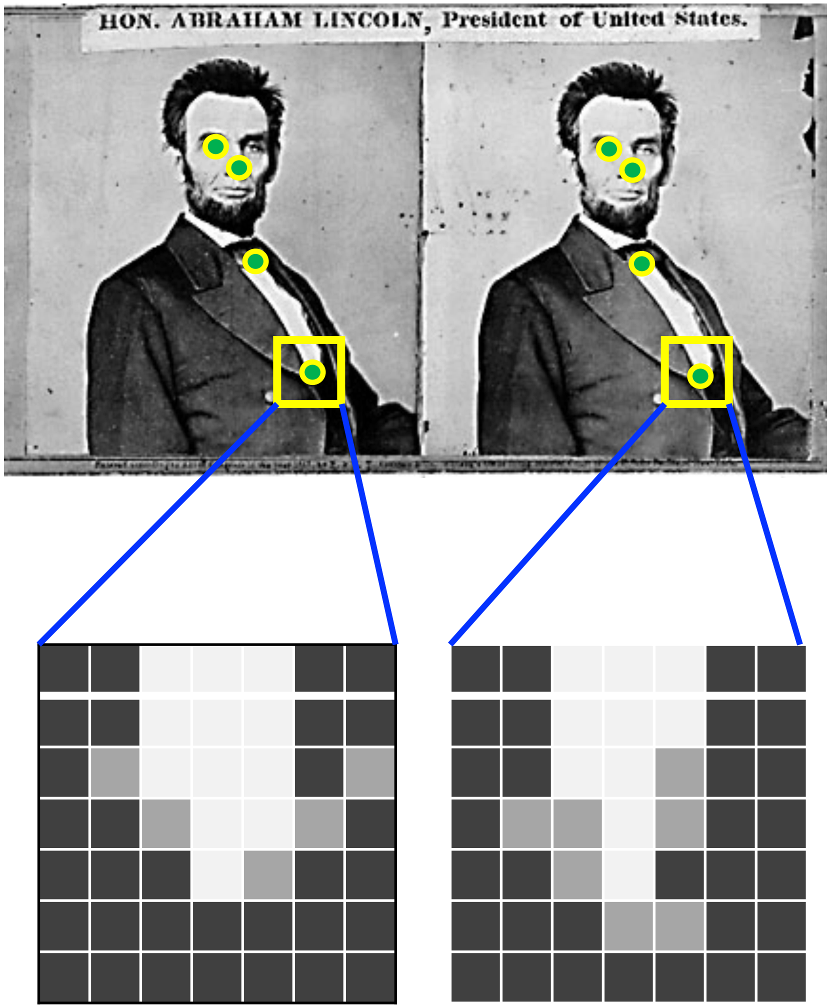
- NCC (Normalized Cross Correlation)

$$\sum_{x,y} \frac{(W_1(x,y) - \overline{W_1})(W_2(x,y) - \overline{W_2})}{\sigma_{W_1}\sigma_{W_2}}$$

$$\overline{W_i} = \frac{1}{n}\sum_{x,y} W_i \,, \quad \sigma_{W_i} = \sqrt{\frac{1}{n}\sum_{x,y}(W_i - \overline{W_i})^2}$$

(Mean) (Standard deviation)

- What advantages might NCC have over SSD?



$W_1(x, y)$: k x k pixel patch in image 1

$W_2(x, y)$: k x k pixel patch in image 2

Feature matching

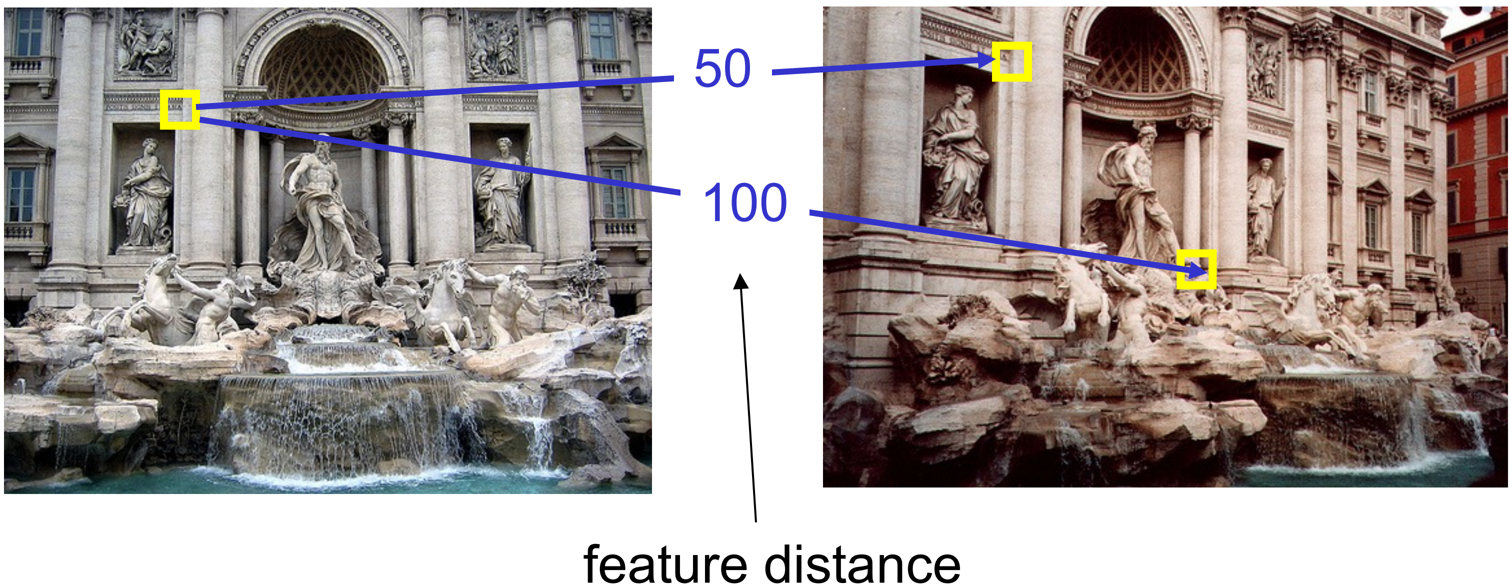
Given a feature in I_1, how to find the best match in I_2?

1. Define distance function that compares two descriptors
2. Test all the features in I_2, find the one with min distance

Feature distance

How to define the distance function between features f_1, f_2?

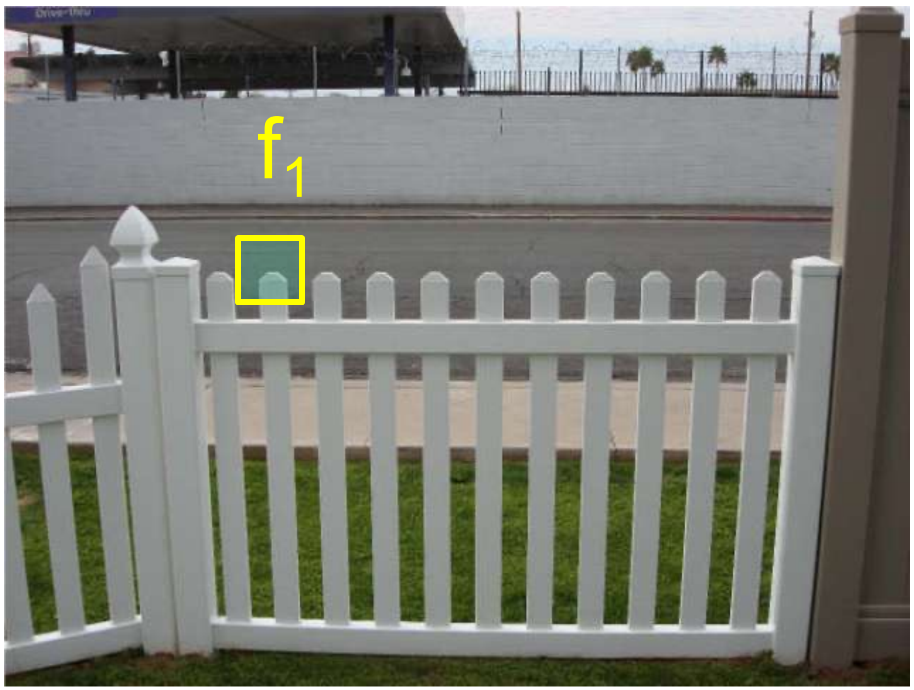
- Simple approach is SSD(f_1, f_2)
  - sum of square differences between entries of the two descriptors



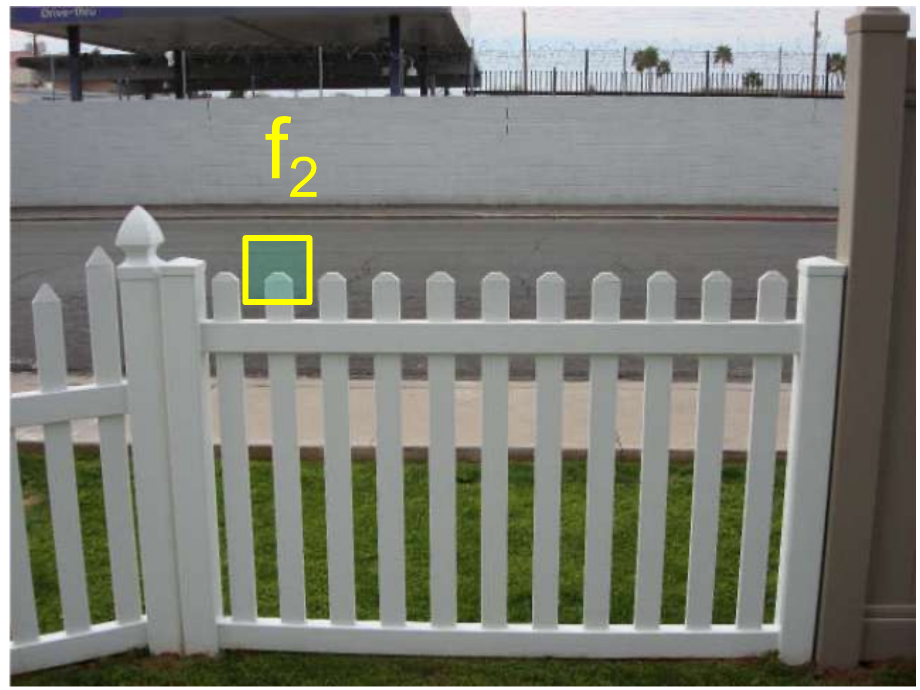
Feature distance

How to define the distance function between features f_1, f_2?

- Simple approach is SSD(f_1, f_2)
  - sum of square differences between entries of the two descriptors



I_1

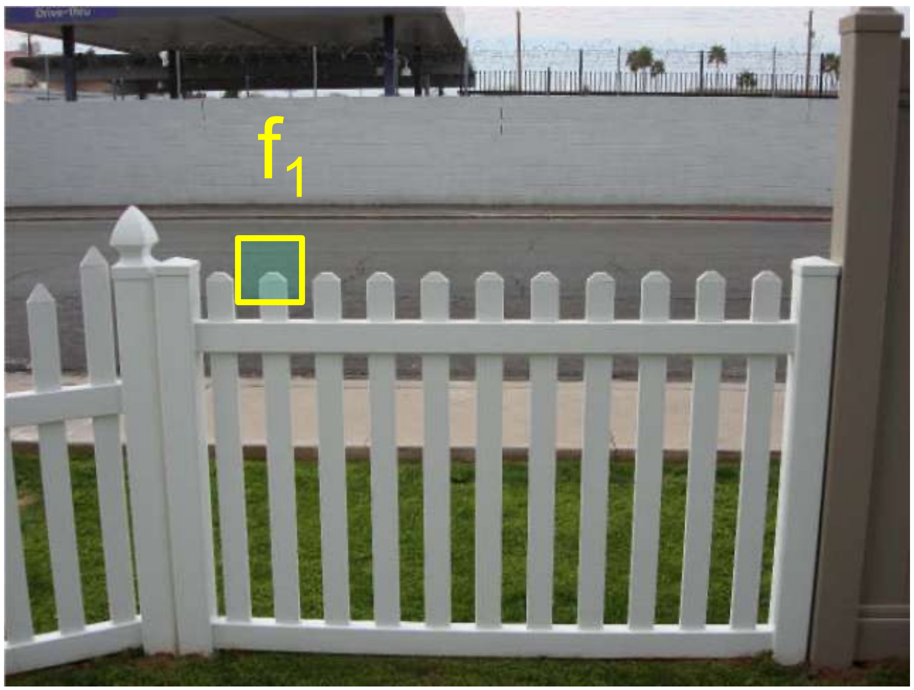


I_2

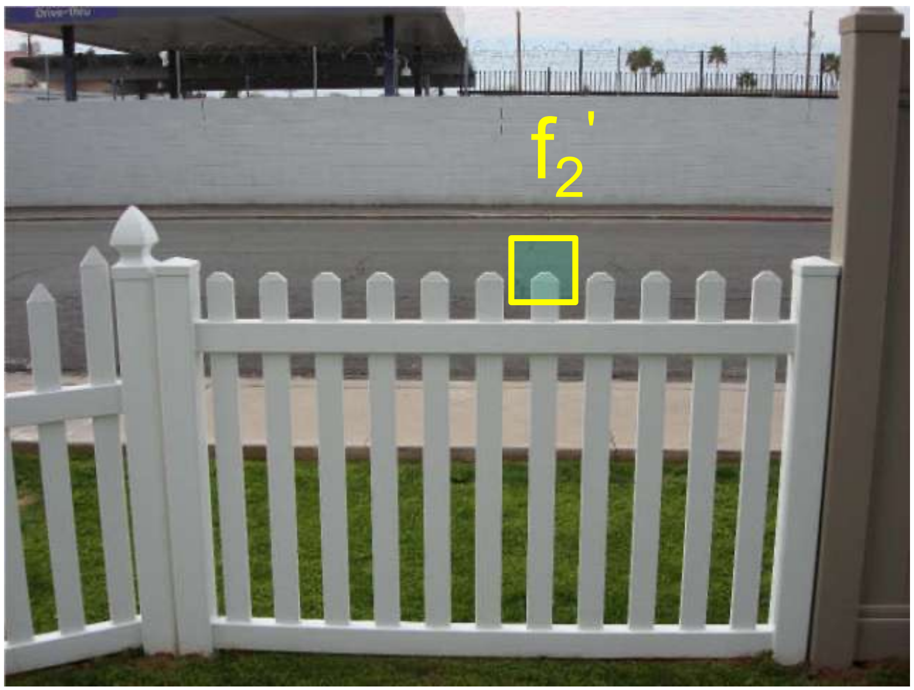
Feature distance

How to define the distance function between features f_1, f_2?

- Simple approach is SSD(f_1, f_2)
  - sum of square differences between entries of the two descriptors
  - can give good scores to ambiguous (bad) matches



I_1

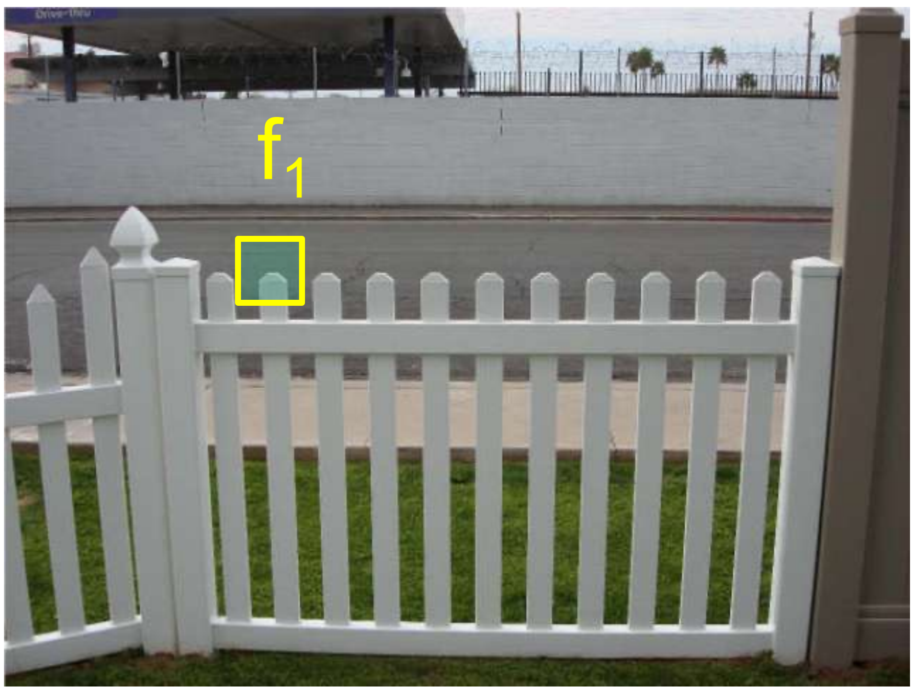


I_2

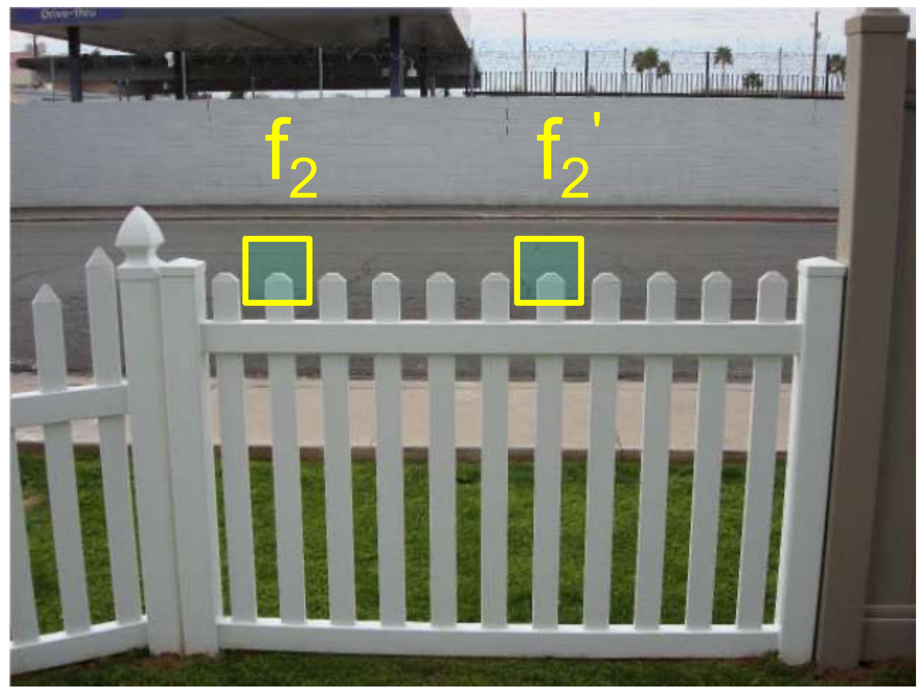
Feature distance

How to define the distance function between features f_1, f_2?

- Better approach: ratio distance = $SSD(f_1, f_2) / SSD(f_1, f_2')$
  - f_2 is best SSD match to f_1 in I_2
  - f_2' is 2^{nd} best SSD match to f_1 in I_2
  - gives small values for unambiguous matches



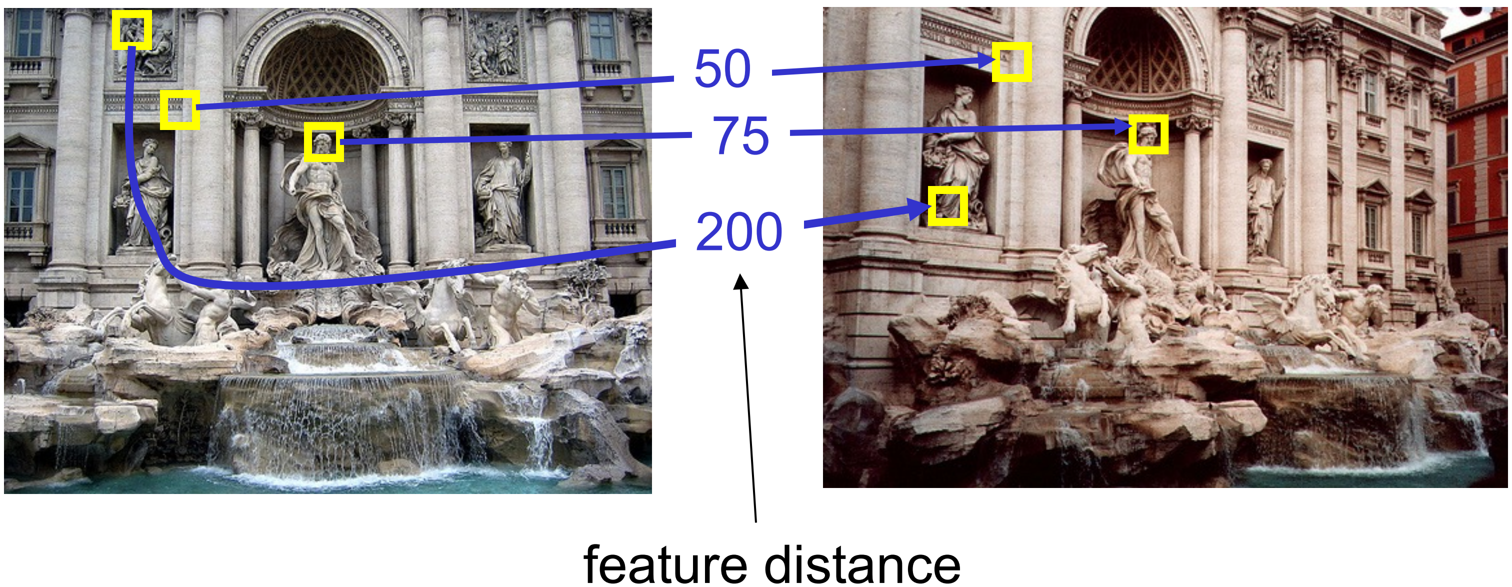
I_1



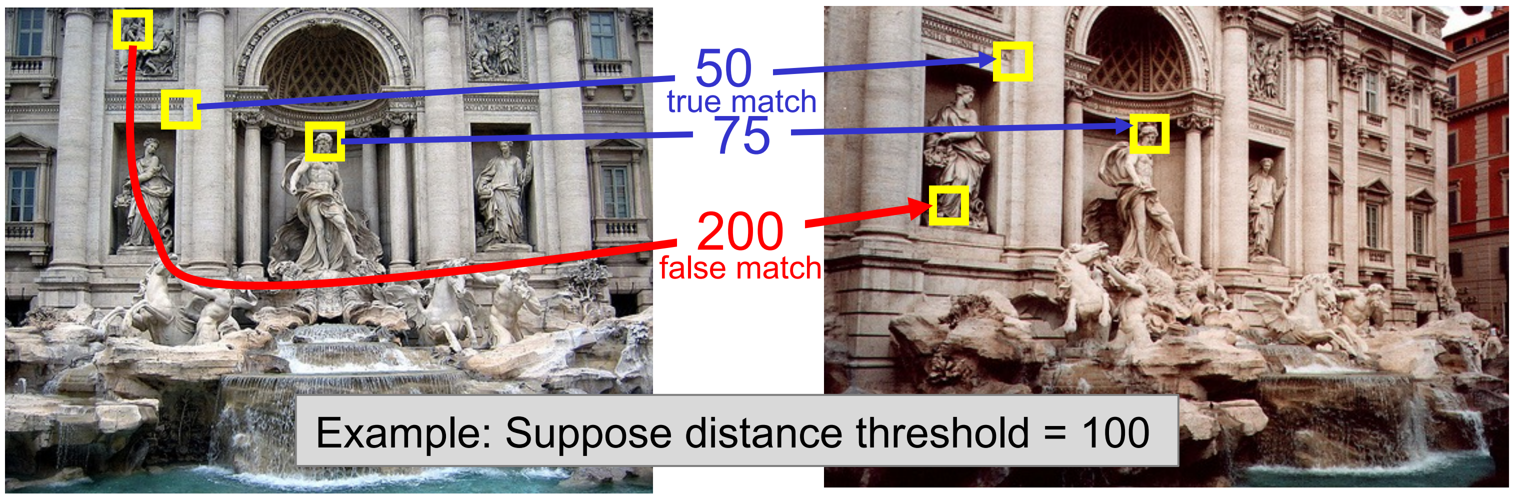
I_2

Evaluating the results

How can we measure the performance of a feature matcher?



True or false positives

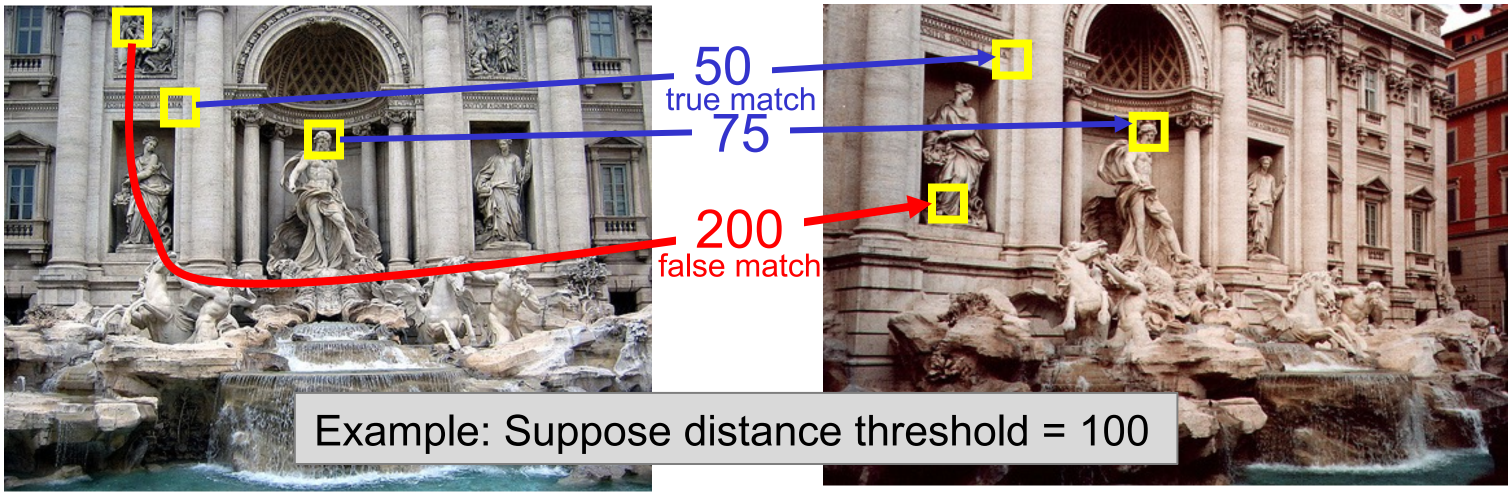


Positive match if SSD < distance threshold

The distance threshold affects performance

- True positives = number of detected matches that are correct
  - Suppose we want to maximize these—how to choose threshold?
- False positives = number of detected matches that are incorrect
  - Suppose we want to minimize these—how to choose threshold?

True or false positives



Positive match if SSD < distance threshold

The distance threshold affects performance

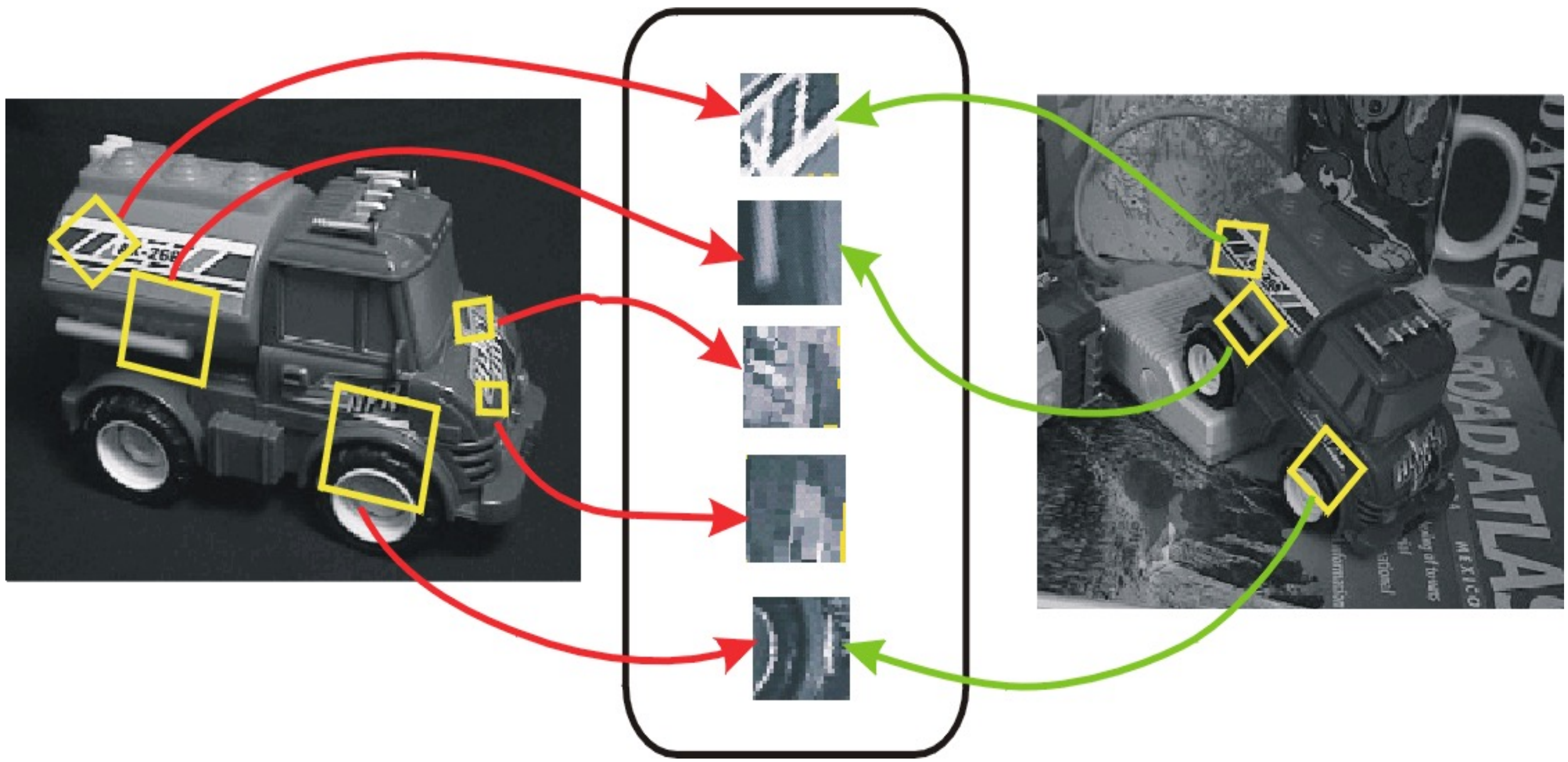
- True positives = number of detected matches that are correct
  - Suppose we want to maximize these—how to choose threshold?
  - Increase threshold (uncertain matches are also allowed)

- False positives = number of detected matches that are incorrect
  - Suppose we want to minimize these—how to choose threshold?
  - Decrease threshold (matches discarded unless they are very certain)

Desirable property: invariance

Find features that are invariant to transformations across two images

- geometric invariance: translation, rotation, scale
- photometric invariance: brightness, exposure, …



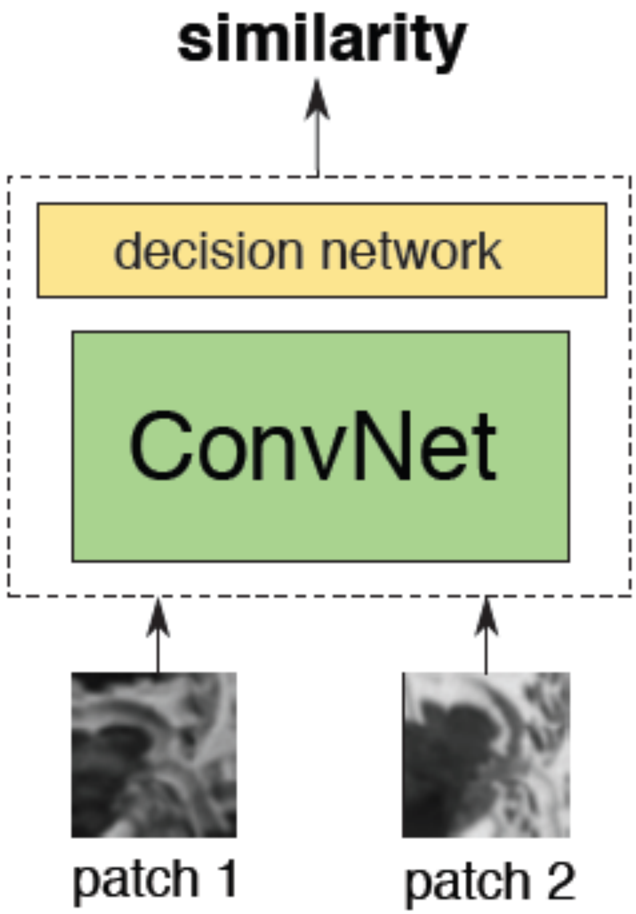
Feature Descriptors

SIFT: Popular for Feature Matching

Extraordinarily robust matching technique

- Can handle changes in viewpoint
  - Up to about 30 degrees out of plane rotation
- Can handle significant changes in illumination
  - Sometimes even day vs. night (below)
- Fast and efficient—can run in real time
- Lots of code available
  - http://people.csail.mit.edu/albert/ladypack/wiki/index.php/Known_implementations_of_SIFT

Learning correspondence

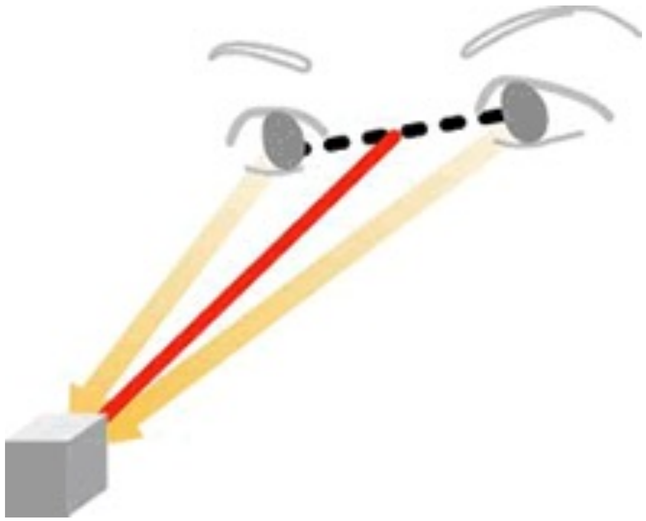


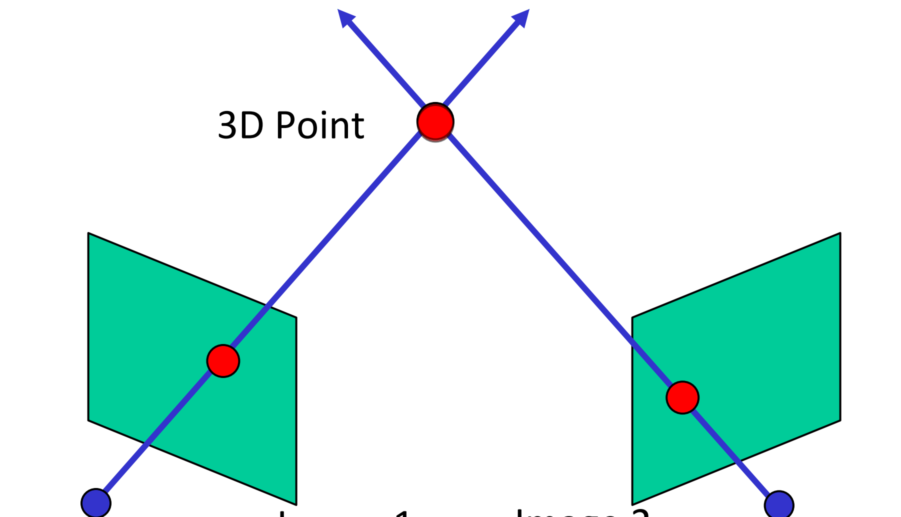
Idea:

- Siamese network to decide patch similarity
- Use intermediate activations as features.

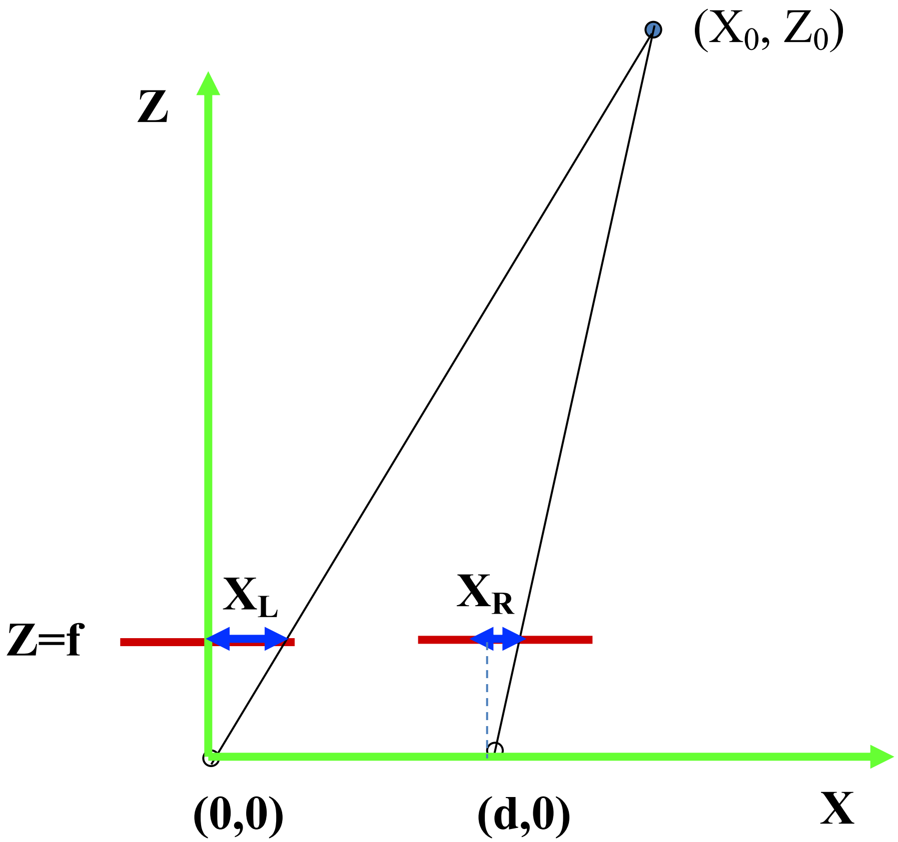
[Zagoruyko and Komodakis, CVPR 2015]

Correspondence is a vital 3D cue





Depth from correspondence

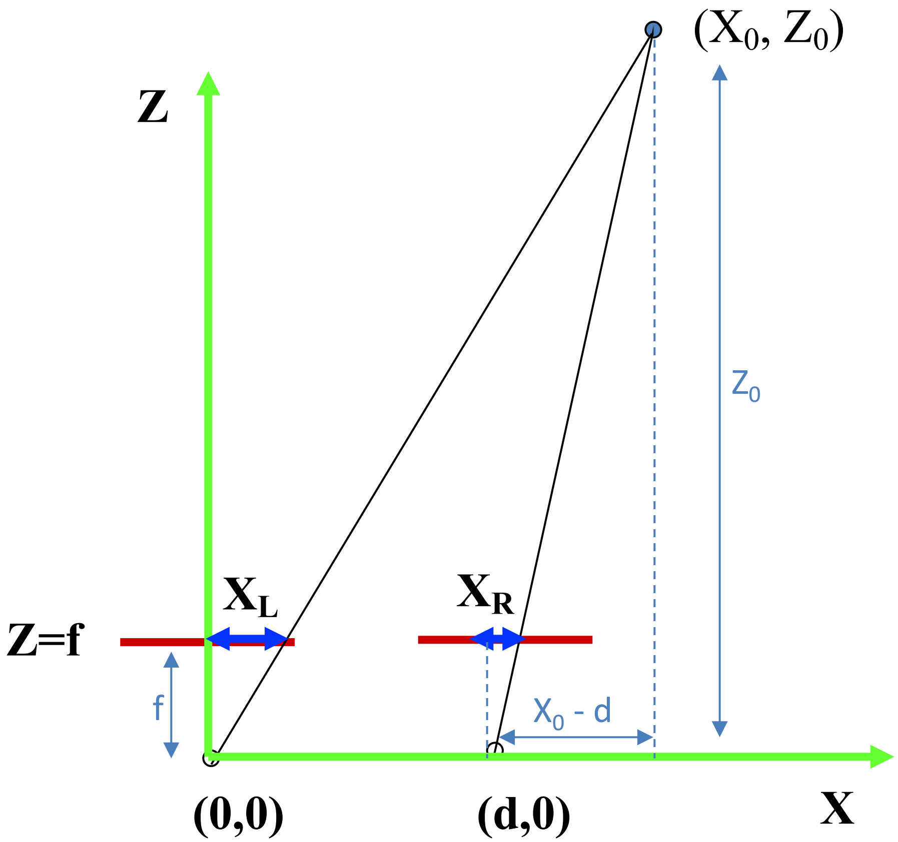


Two measurements: X_L, X_R
Two unknowns: X_0, Z_0

Constants:
- **Baseline: d**
- **Focal length: f**

(Adapted from Hager)

Depth from correspondence



Two measurements: X_L, X_R
Two unknowns: X_0, Z_0

Constants:
Baseline: d
Focal length: f

$$X_0 = \frac{d\, X_L}{(X_L - X_R)}$$

$$Z_0 = \frac{d\, f}{(X_L - X_R)}$$

Disparity: $(X_L - X_R)$

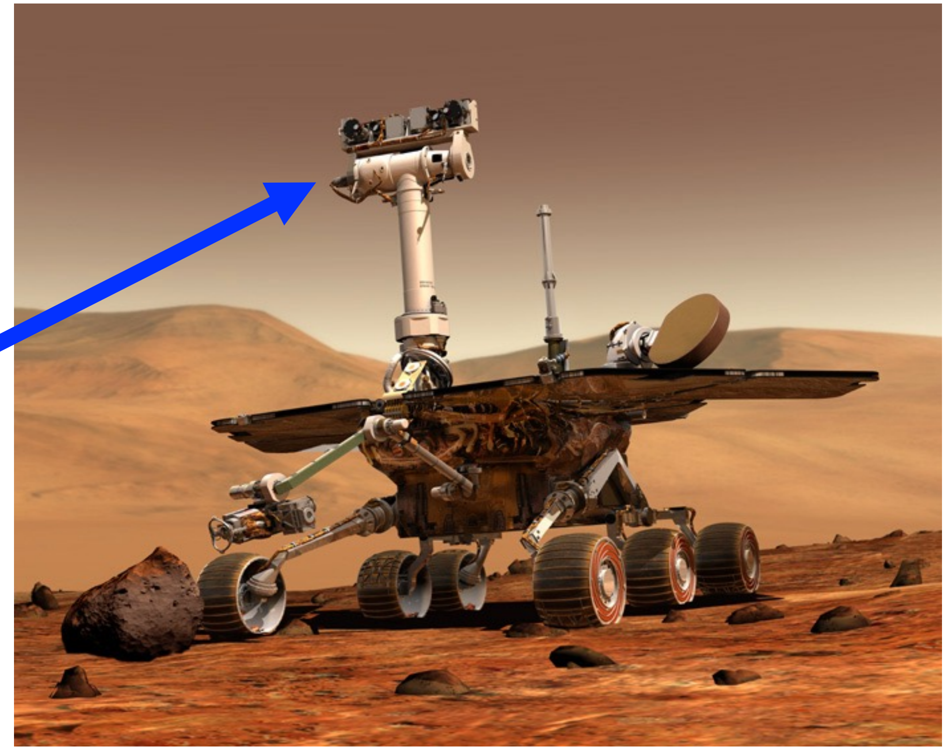
Using similar triangles:

$$\frac{X_L}{f} = \frac{X_0}{Z_0} \qquad \frac{X_R}{f} = \frac{X_0 - d}{Z_0}$$

Depth is inversely proportional to disparity

(Adapted from Hager)

Mars Exploratory Rovers:
Spirit and Opportunity,
2004



Stereo camera