Introduction to Aerial Robotics
Lecture 5

Shaojie Shen
Associate Professor
Dept. of ECE, HKUST

4 March 2024

Course Outline

- Dynamics, Planning & Control
- Vision
- Estimation

Week	Lecture Date Tue 13:30-16:20	Topic	Assignment (due **23:59** on **Friday** of the corresponding week)	Lab Wed 18:00-20:50
1	2/4	Introduction		No Lab
2	2/11	Rigid Body Transformation Quaternion Quadrotor Modeling		No Lab
3	2/18	Control Basics Quadrotor Control Trajectory Generation	Project 1 Phase 1 Out	No Lab
4	2/25	Trajectory Generation Path Planning	Project 1 Phase 1 Due Project 1 Phase 2 Out	No Lab
5	3/4	Camera Modeling & Calibration Feature Detection & Matching	Project 1 Phase 2 Due Project 1 Phase 3 Out	No Lab
6	3/11	Midterm	Project 1 Phase 3 Due Project 1 Phase 4 Out	Lab Tutorial 1: Robot Assembly
7	3/18	Multi-View Geometry Pose Estimation	Project 1 Phase 3 Due Project 2 Phase 1 Out	Lab Tutorial 2: Prepare P1P4
8	3/25	Optical Flow Dense Stereo	Project 2 Phase 1 Due Project 2 Phase 2 Out	Free Lab Time
9	4/1	Midterm Break (No class)		No Lab
10	4/8	Probability Basics Bayesian Inferencing Kalman Filter	Project 2 Phase 2 Due Project 3 Phase 1 Out	Free Lab Time
11	4/15	Extended Kalman Filter Augmented State EKF Particle Filter	Project 1 Phase 4 Due Project 3 Phase 2 Out	Free Lab Time
12	4/22	SLAM	Project 3 Phase 3 Out Project 3 Phase 1 Due	Lab Tutorial 3: Prepare P3P3
13	4/29	x	Project 3 Phase 2 Due	Free Lab Time
14	5/6	x	Project 3 Phase 3 Due	Free Lab Time

Outline

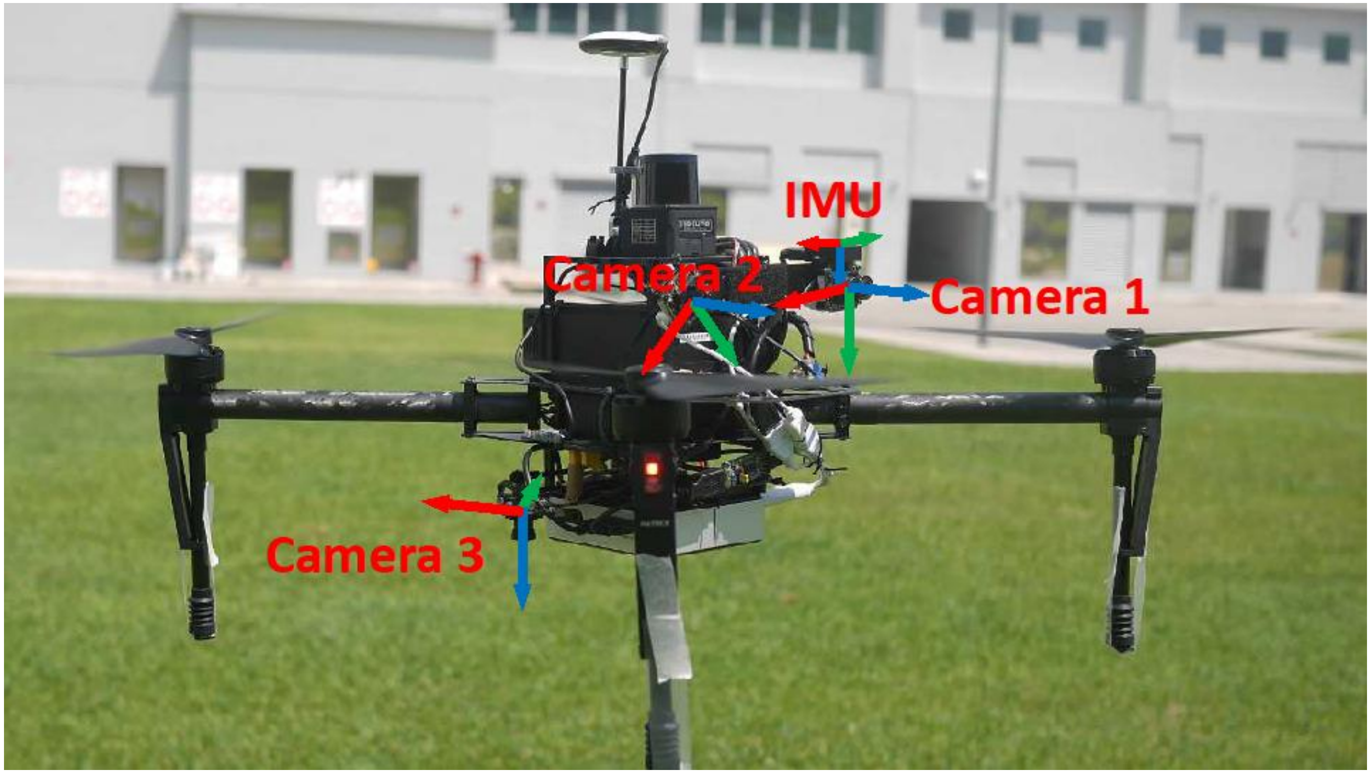
- Camera Modeling
- Robot Vision Pipeline
- Point Feature Detection & Matching

Camera Modeling

Goal: Fly Like Birds

Video from Smithsonian Channel

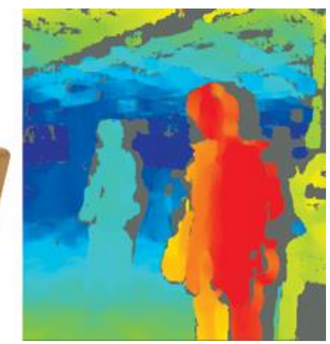
Robot Sees with Cameras



Cameras

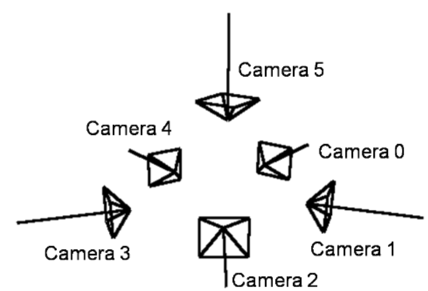
- Monocular
  - Simplest setup
  - Depth unknown

- Stereo
  - Able to compute depth
  - Depth accuracy affected by baseline, resolution, and calibration



- Multi-Camera
  - Overlapping / Non-overlapping field-of-view

Omnidirectional multi-camera system Ladybug



Relative position and posture of each camera

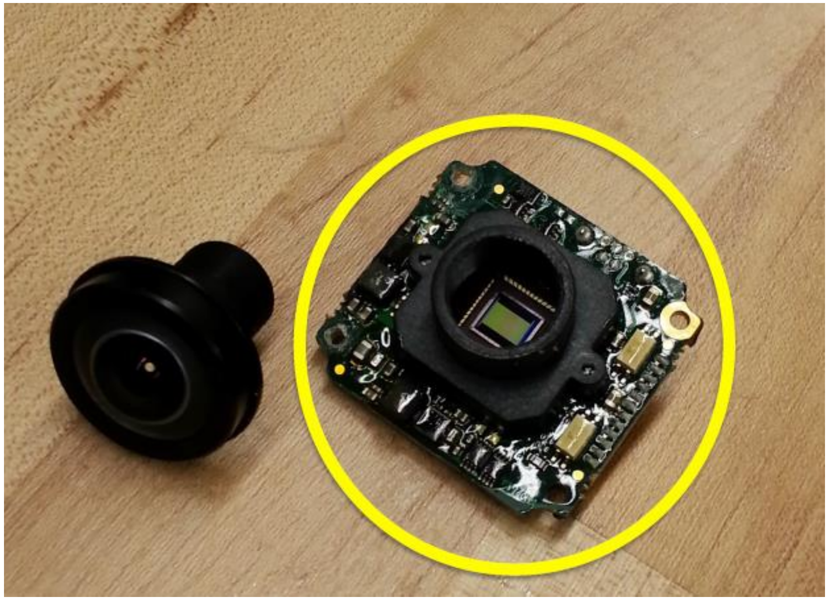
Cameras

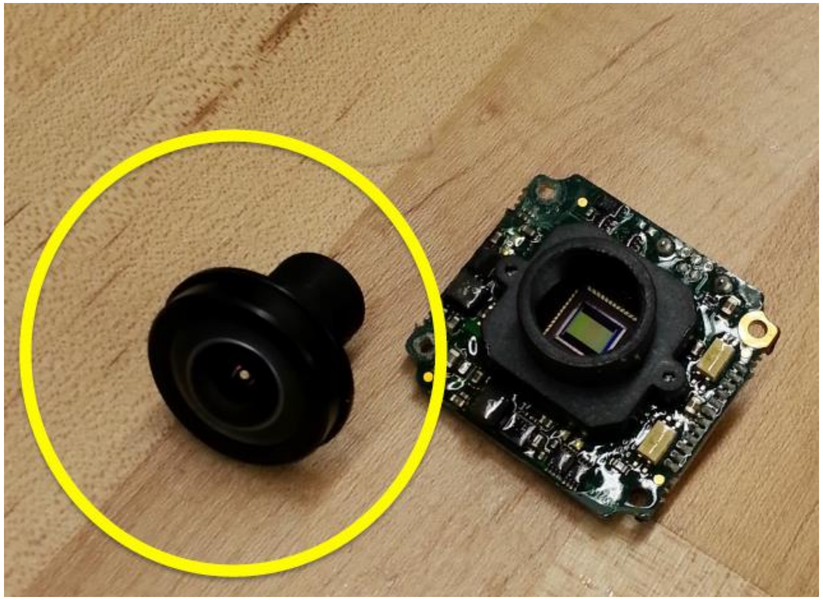
- RGB-D Sensor
  - Great depth
  - Does not work outdoors

- Omnidirectional camera
  - 360 capture
  - Strong distortion

Cameras

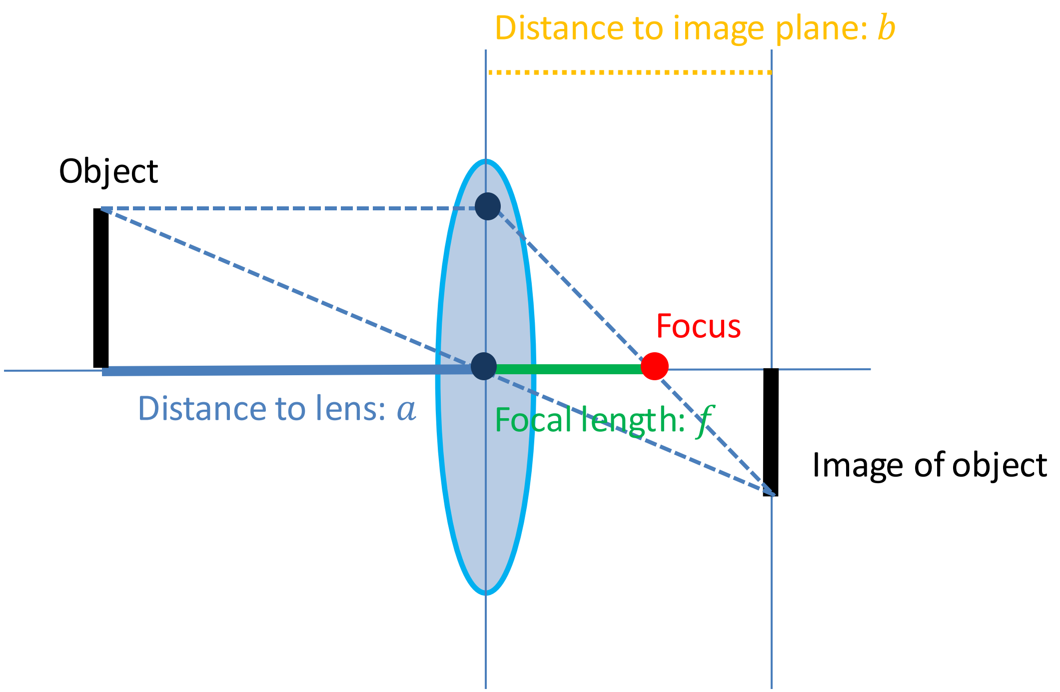
- Sensor
- Lens





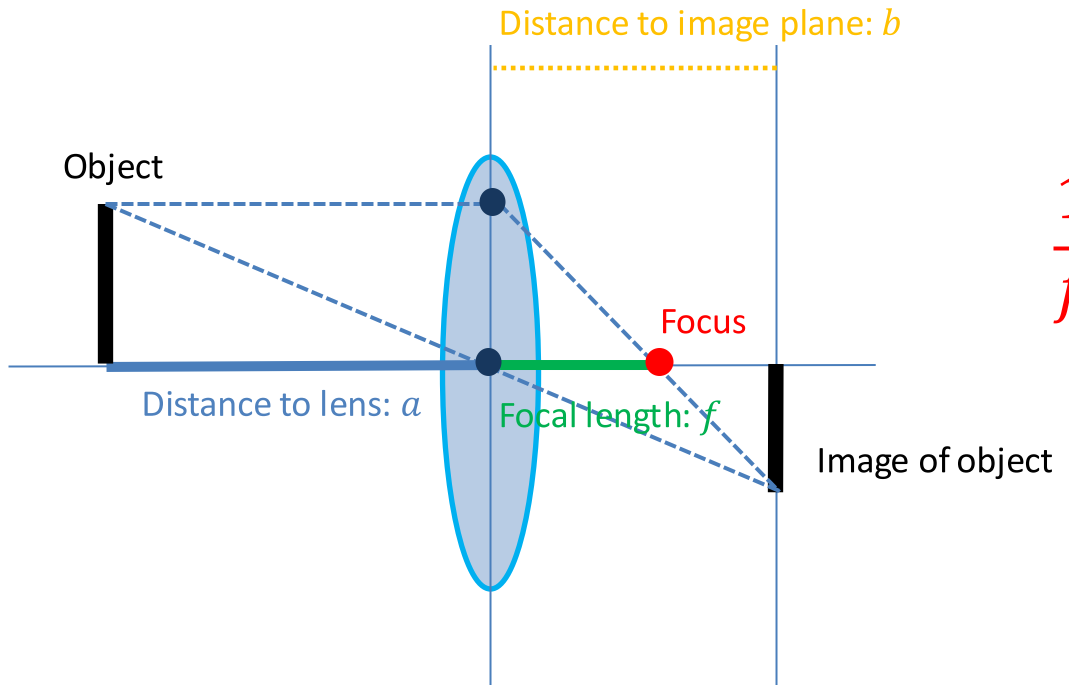
Thin Lens

- A lens with a thickness that is negligible compared to the radii of curvature of the lens surfaces



Thin Lens

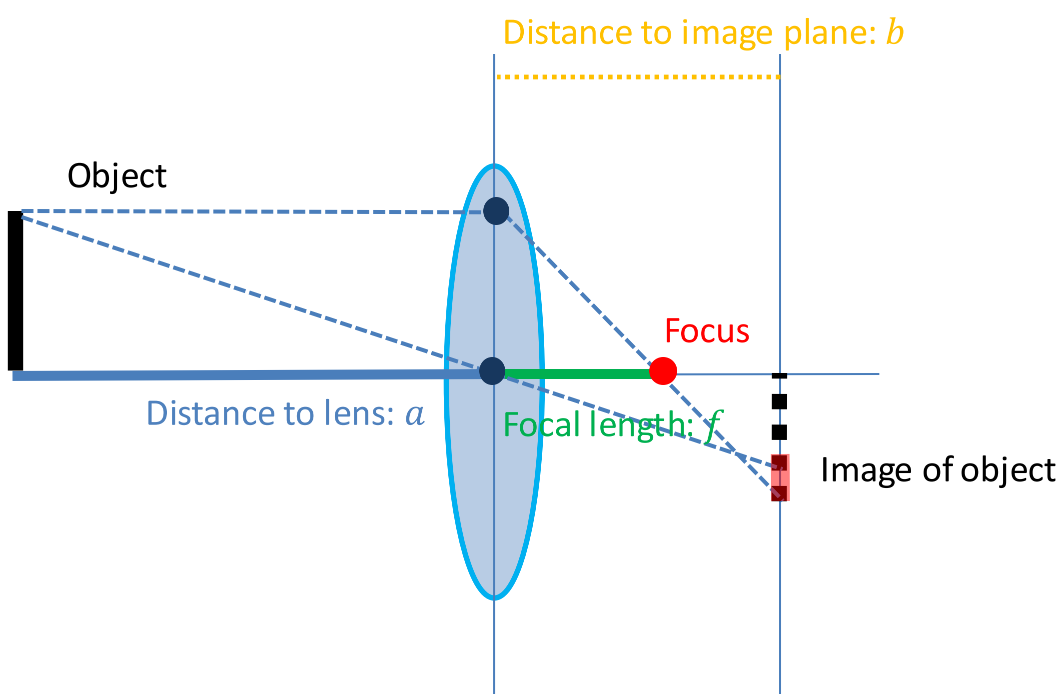
- Rays parallel to the optical axis meet focus after leaving the lens
- Rays through center of the lens do not change direction



$$\frac{1}{f} = \frac{1}{a} + \frac{1}{b}$$

Thin Lens

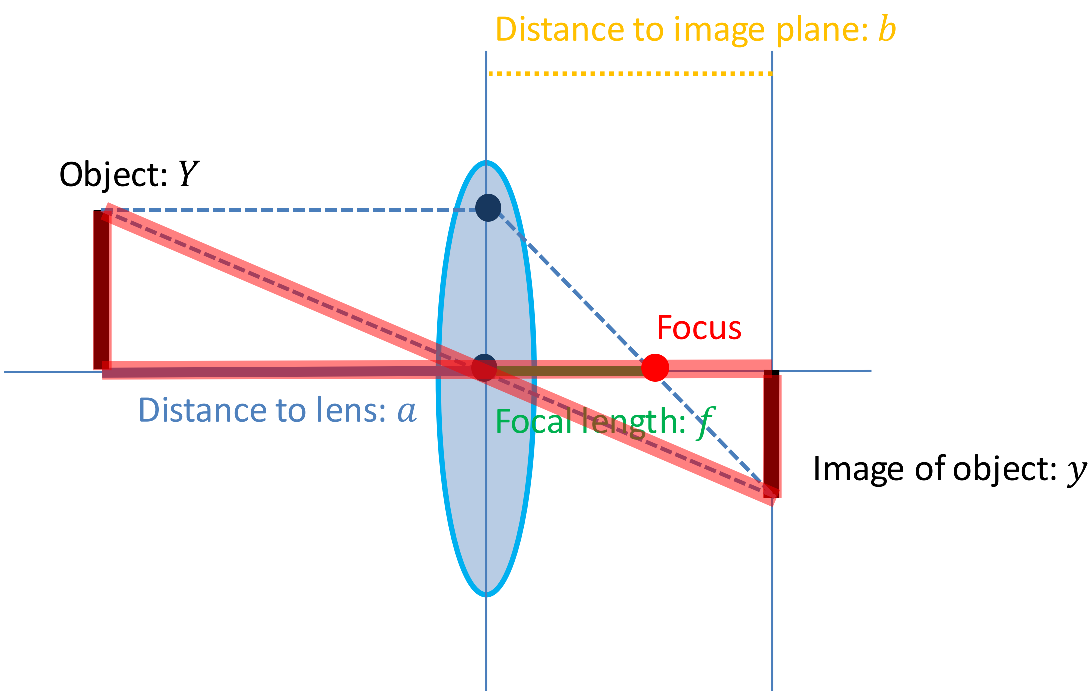
- De-focus if $$\frac{1}{f} \neq \frac{1}{a} + \frac{1}{b}$$



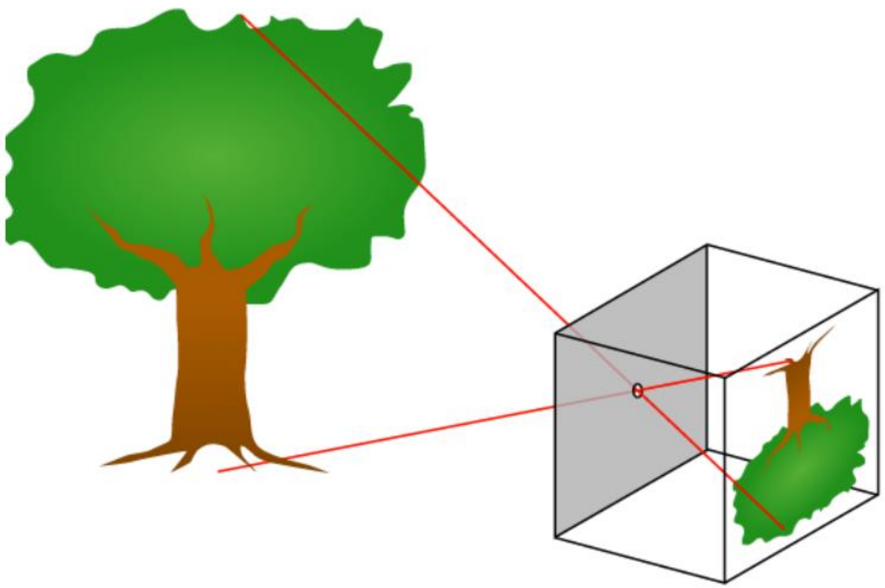
Perspective Projection

- The object comes closer, the image becomes bigger
- A point moving on the same ray does not change its image
- If we only look at rays going through center of the lens:

$$-\frac{Y}{a} = \frac{y}{b}$$



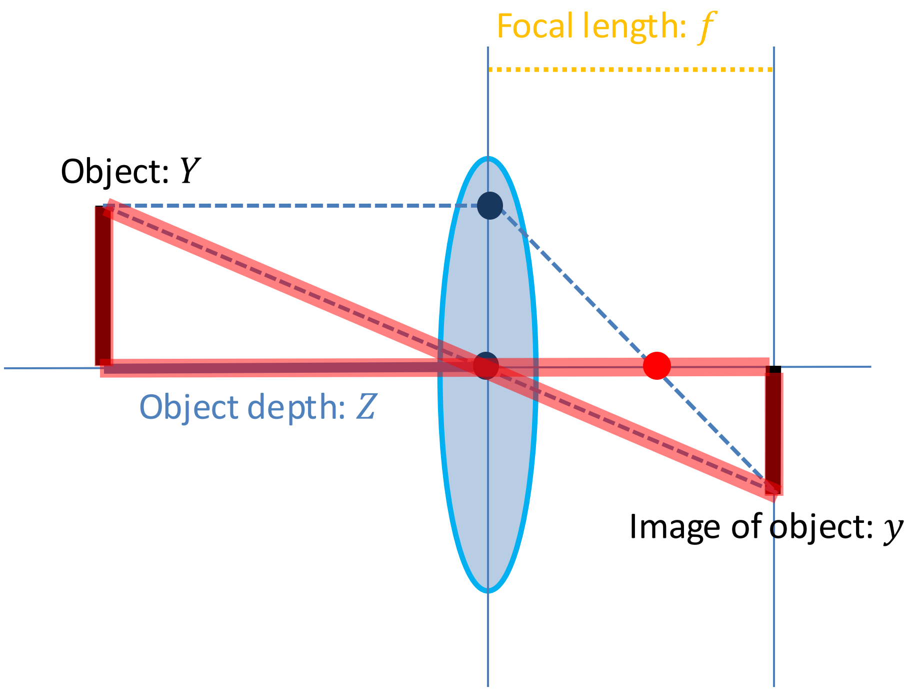
Pin-hole Camera Model



WIKIPEDIA
The Free Encyclopedia

Pin-hole Camera Model

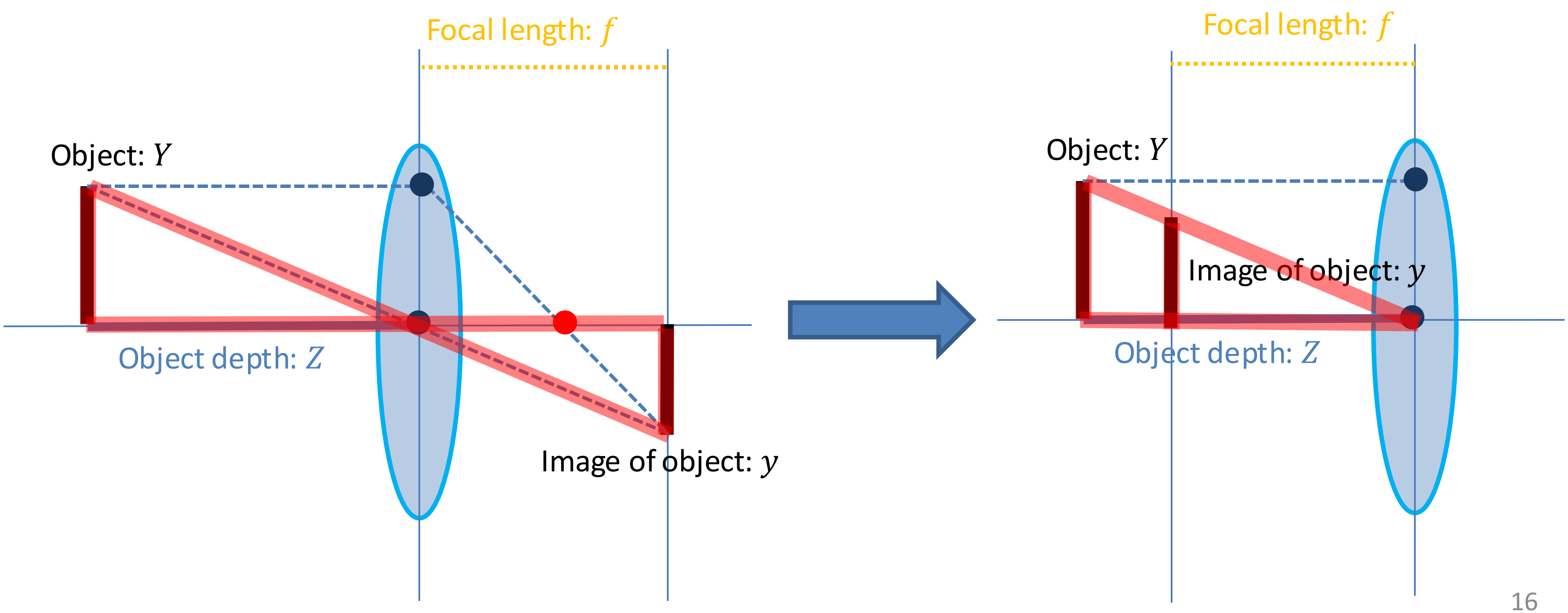
- If we replace b with f and include a minus because the object image is upside down ($Z = a, f = b$)
  - $y = -f\frac{Y}{Z}$



Pin-hole Camera Model

- Assume that the image plane is in front of the lens:

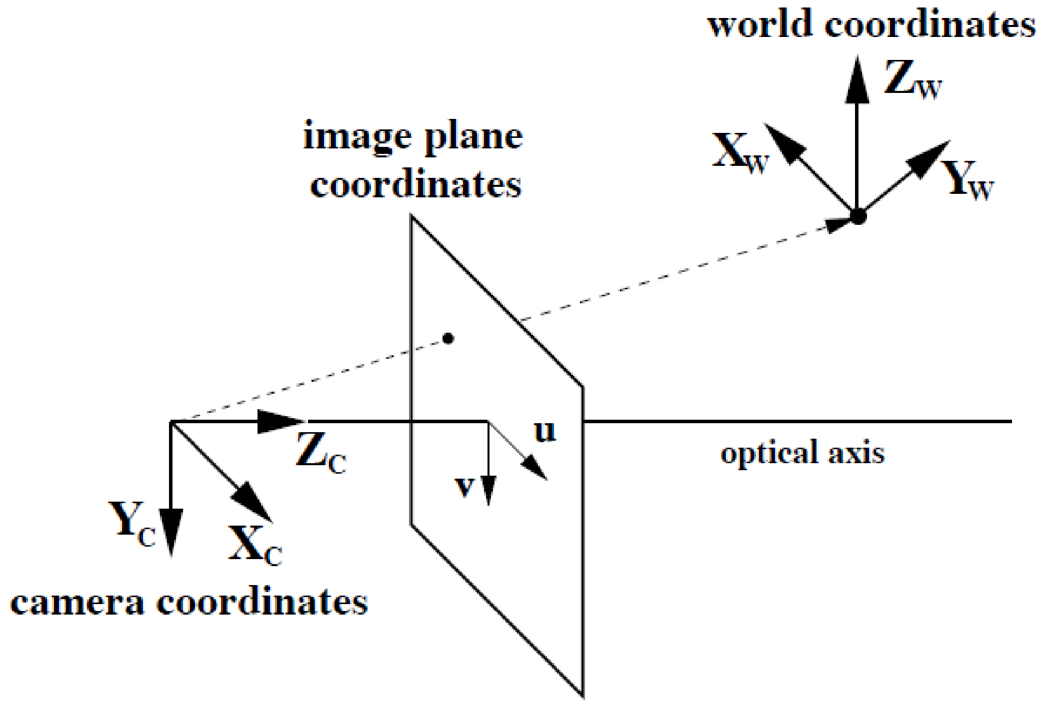
$$-y = f\frac{Y}{Z}$$



Effect of $f = b$?

- Theoretically ,we expect an offset in the x and y coordinates caused by the error $(f - b)$
- If the object is on focus: $\frac{b-f}{f} = \frac{b}{Z}$
  - Relative error depends on the ratio of focus length to depth.
  - This matters if we actually use the focus length from camera specs
  - In practice, we use a procedure called calibration to obtain f that best satisfies: $y = f\frac{Y}{Z}$

Camera Coordinate System

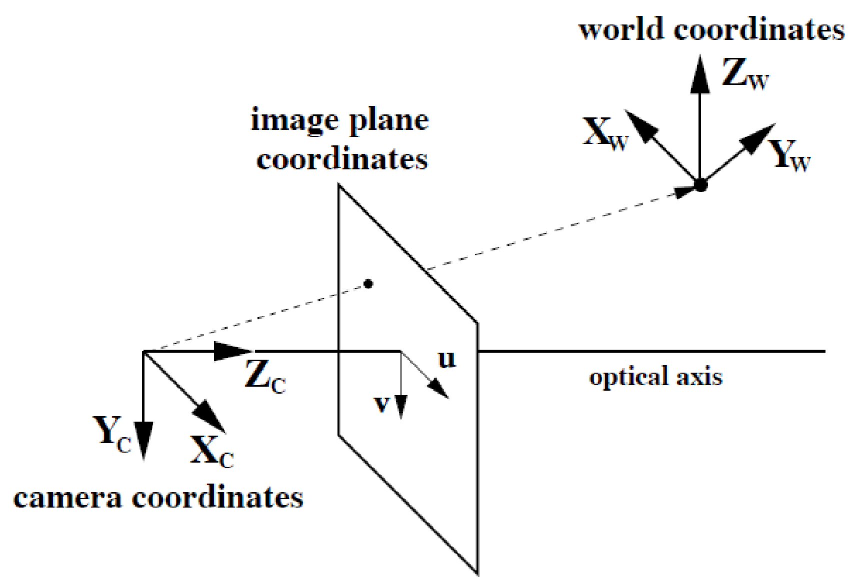


Courtesy Kostas Daniilidis

香港科技大學
THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY
電子及計算機工程學系
DEPARTMENT OF ELECTRONIC & COMPUTER ENGINEERING

Perspective Projection

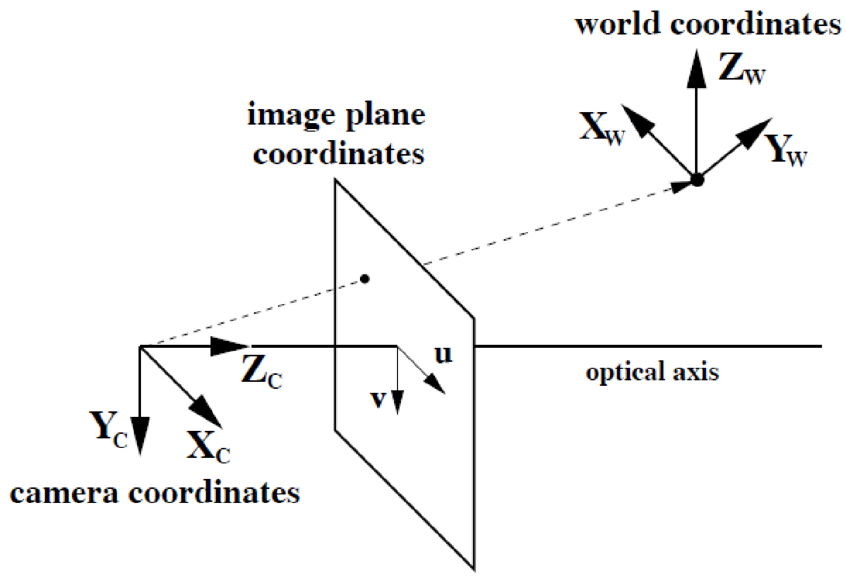
- Optical axis is the z-axis
- The image plane (u, v) is perpendicular to the optical axis
- Intersection of the image plane and the optical axis is the image center (u_0, v_0)
- f is the distance of the image plane from the origin (in pixels)
- Formulation:
  - $u = \frac{fX_c}{Z_c} + u_0$
  - $v = \frac{fY_c}{Z_c} + v_0$



Perspective Projection

- Matrix form:

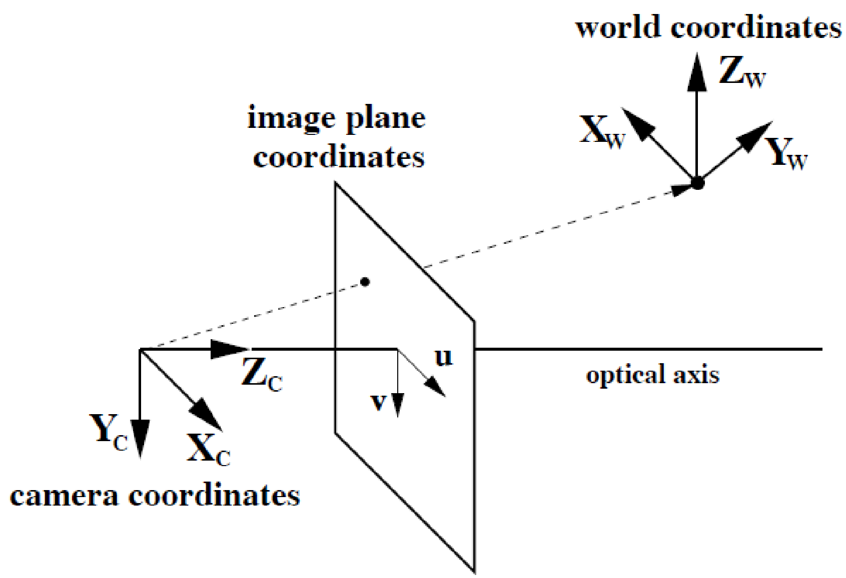
$$\lambda \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} f & 0 & u_o \\ 0 & f & v_o \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{pmatrix}$$



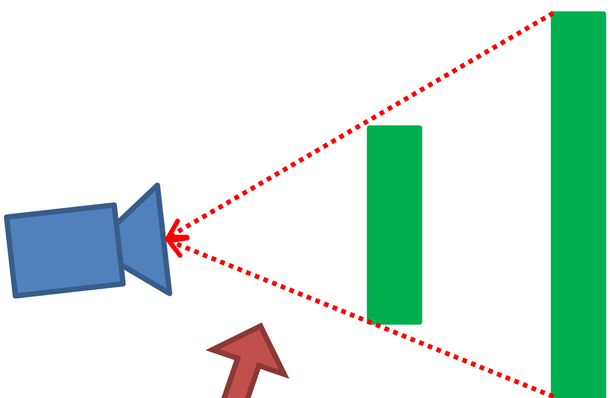
Perspective Projection

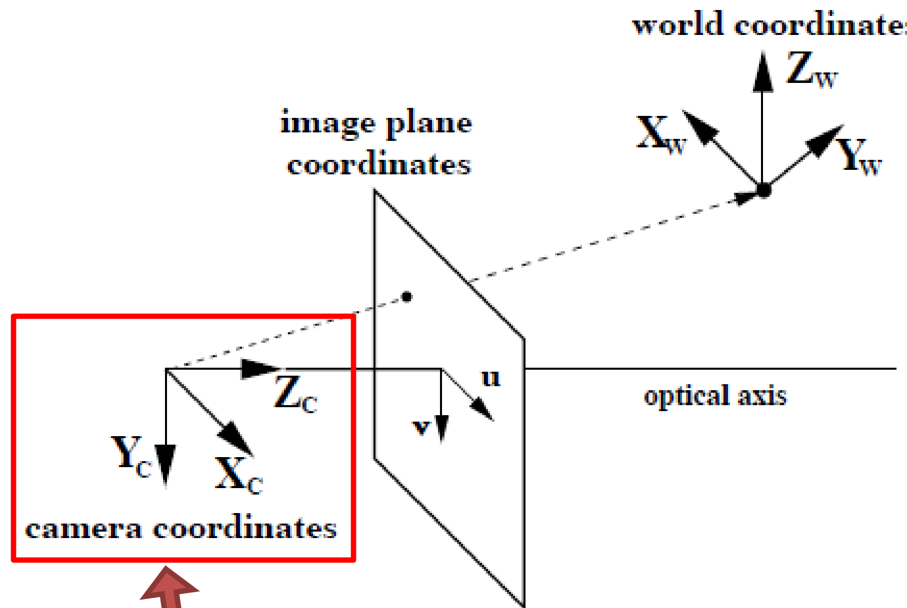
- From camera to world:

$$\begin{pmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{pmatrix} = \begin{pmatrix} R & t \\ 0 & 1 \end{pmatrix} \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix}$$



Pin-hole Camera Model





$$\boxed{\lambda}\boxed{\begin{pmatrix} u \\ v \\ 1 \end{pmatrix}} = \boxed{\begin{pmatrix} f & 0 & u_o \\ 0 & f & v_o \\ 0 & 0 & 1 \end{pmatrix}} \boxed{\begin{pmatrix} R & t \end{pmatrix}} \boxed{\begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix}} = P \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix}$$

Unknown Depth

Pixel Values

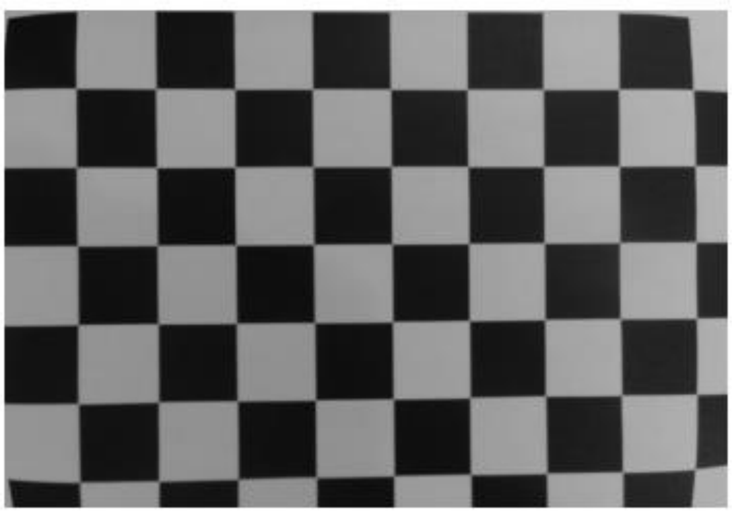
Intrinsic Calibration Matrix

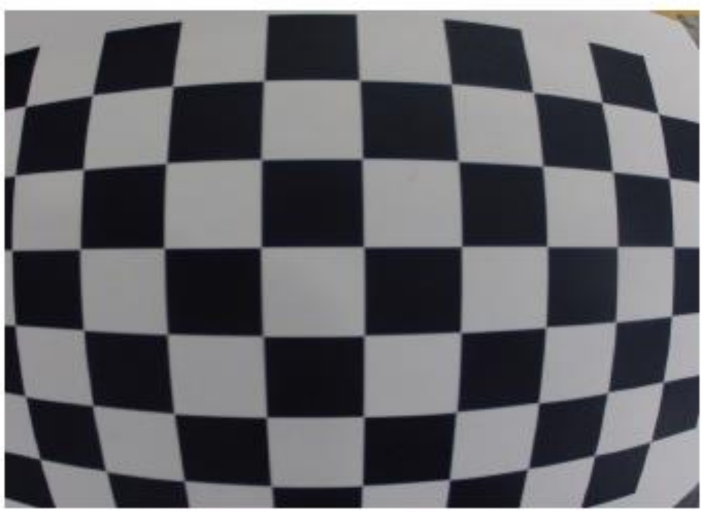
Camera Pose

World Point

Radial Distortions

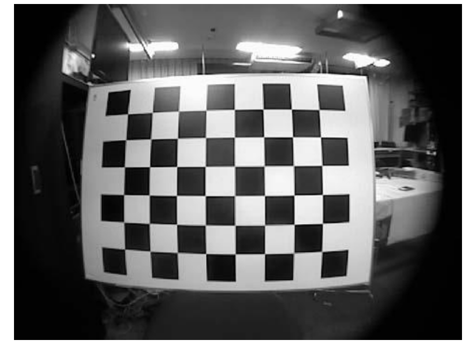
- Wide angle lenses have radial distortions
  - Straight lines become curves
- Formulation:
  - $r^2 = u^2 + v^2$
  - $u^{dist} = u(1 + k_1 r + k_2 r^2 + k_3 r^3 + \cdots)$
  - $v^{dist} = v(1 + k_1 r + k_2 r^2 + k_3 r^3 + \cdots)$





Camera Calibration

- Requires:
  - Calibration object
- Obtains:
  - Intrinsic parameters
  - Distortion parameters
  - Poses of cameras with respect to the calibration object



$$\lambda \underbrace{\boxed{\begin{pmatrix} u \\ v \\ 1 \end{pmatrix}}}_{\text{Pixel Values}} = \underbrace{\boxed{\begin{pmatrix} f & 0 & u_o \\ 0 & f & v_o \\ 0 & 0 & 1 \end{pmatrix}}}_{\text{Intrinsic Calibration Matrix}} \underbrace{\boxed{\begin{pmatrix} R & t \end{pmatrix}}}_{\text{Camera Pose}} \underbrace{\boxed{\begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix}}}_{\text{World Point}} = P \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix}$$

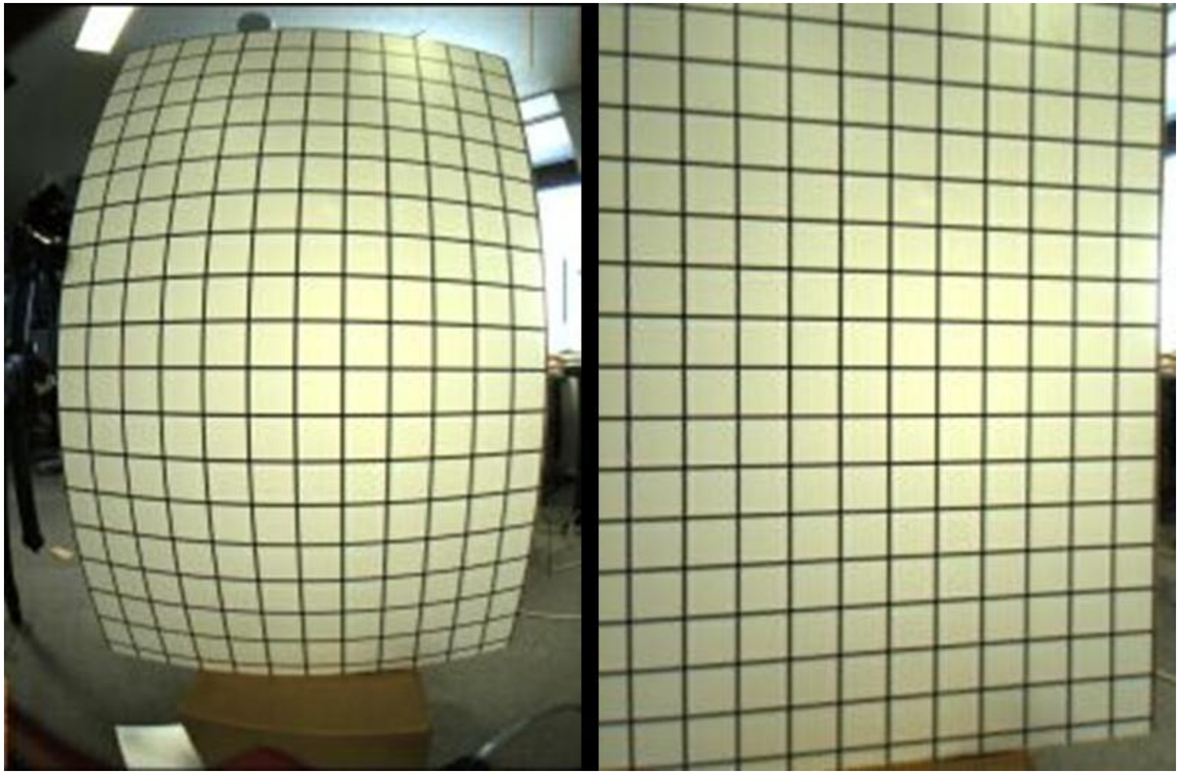
Known

Measurement

To be Estimated

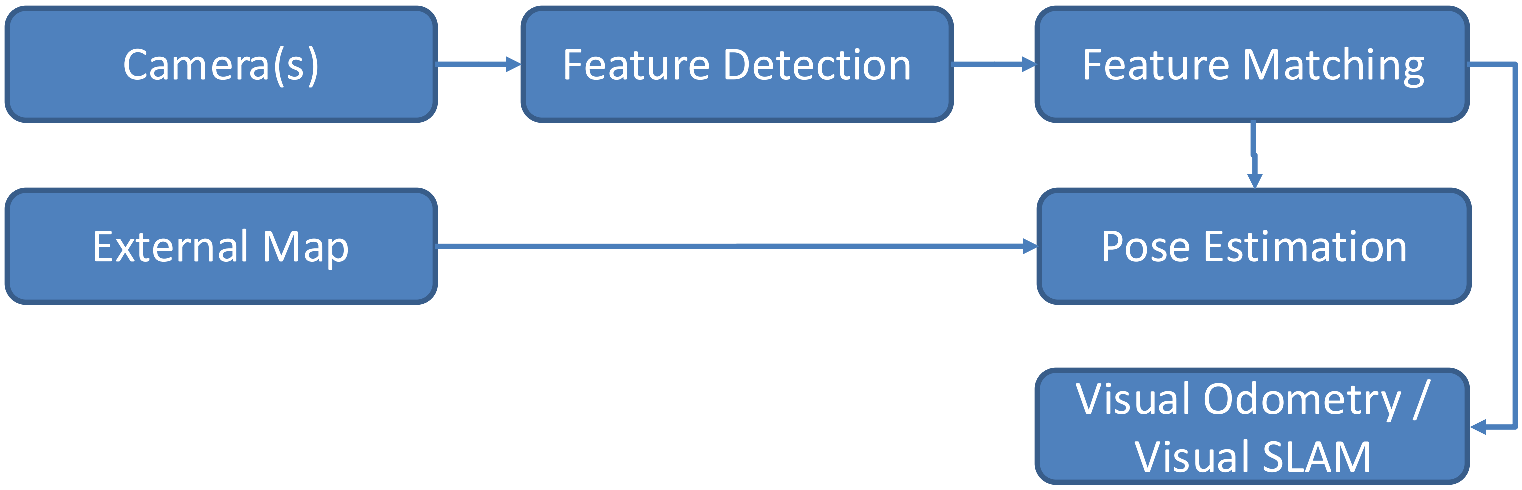
Camera Calibration

- Straight lines should be straight



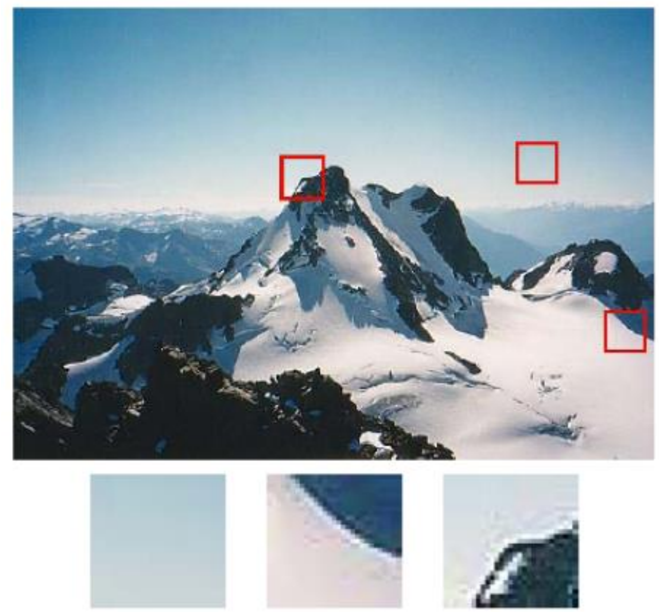
Robot Vision Pipeline

Vision-based State Estimation Pipeline



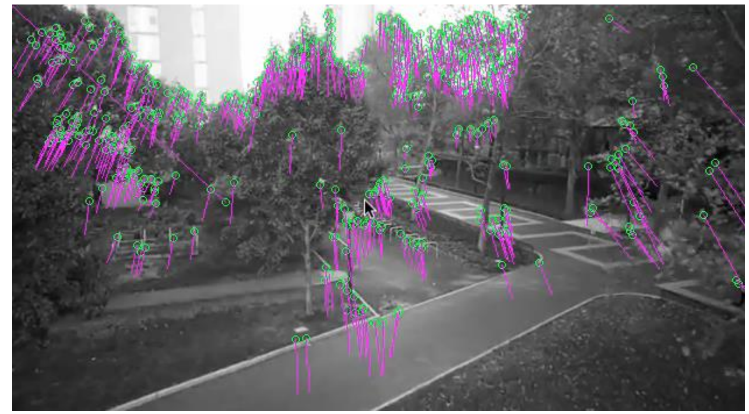
Feature Detection

- We cannot process the entire image directly
- Requirements:
  - Repeatability
  - Saliency
  - Locality
  - Compactness and efficiency
- Popular features
  - Corners (FAST, Harris, …)
  - Blob (SIFT, SURF, …)
  - Line (Canny, …)

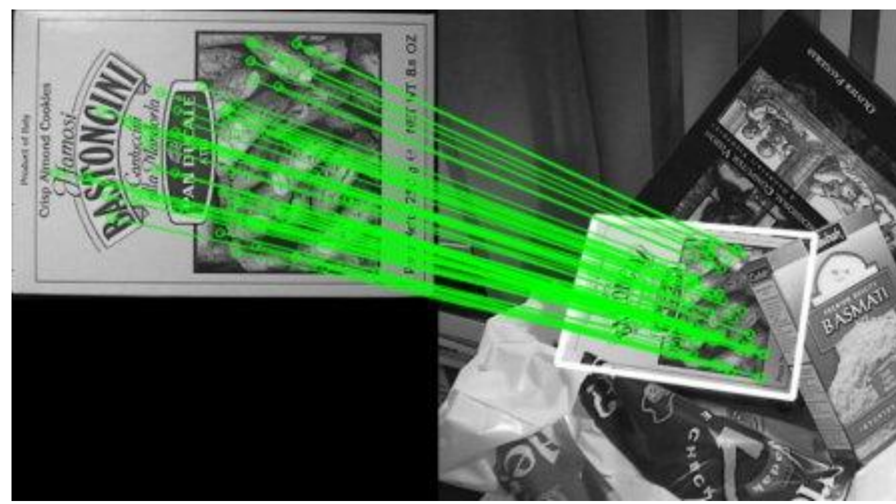


Feature Matching

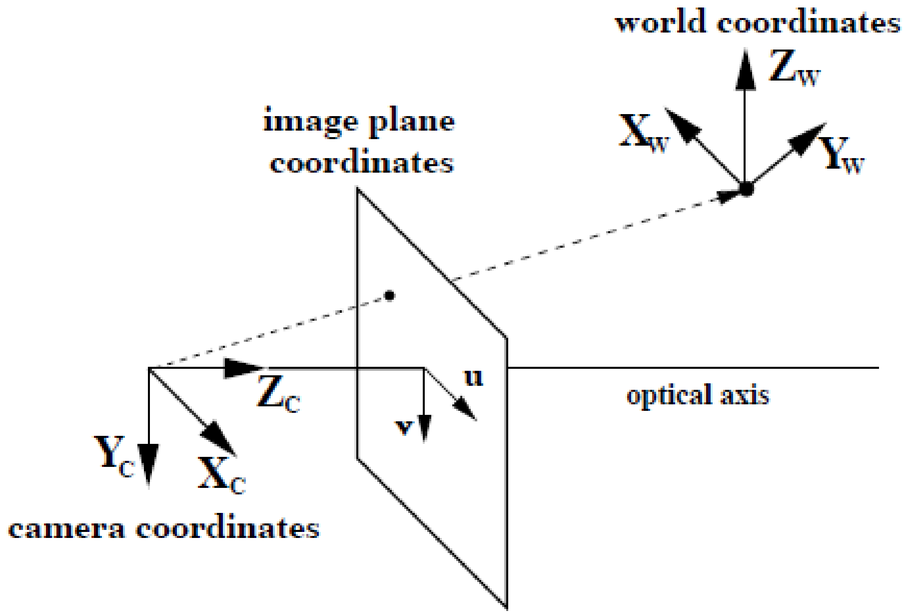
- Match features in different images
  - Across multiple cameras
  - Across time



- Common methods:
  - Descriptor matching
  - Optical flow



Pose Estimation



Known

Measurement

To be Estimated

$$\lambda \boxed{\begin{pmatrix}u\\ v\\ 1\end{pmatrix}} = \boxed{\begin{pmatrix}f & 0 & u_o\\ 0 & f & v_o\\ 0 & 0 & 1\end{pmatrix}} \boxed{\begin{pmatrix}R & t\end{pmatrix}} \boxed{\begin{pmatrix}X_w\\ Y_w\\ Z_w\\ 1\end{pmatrix}} = P\begin{pmatrix}X_w\\ Y_w\\ Z_w\\ 1\end{pmatrix}$$

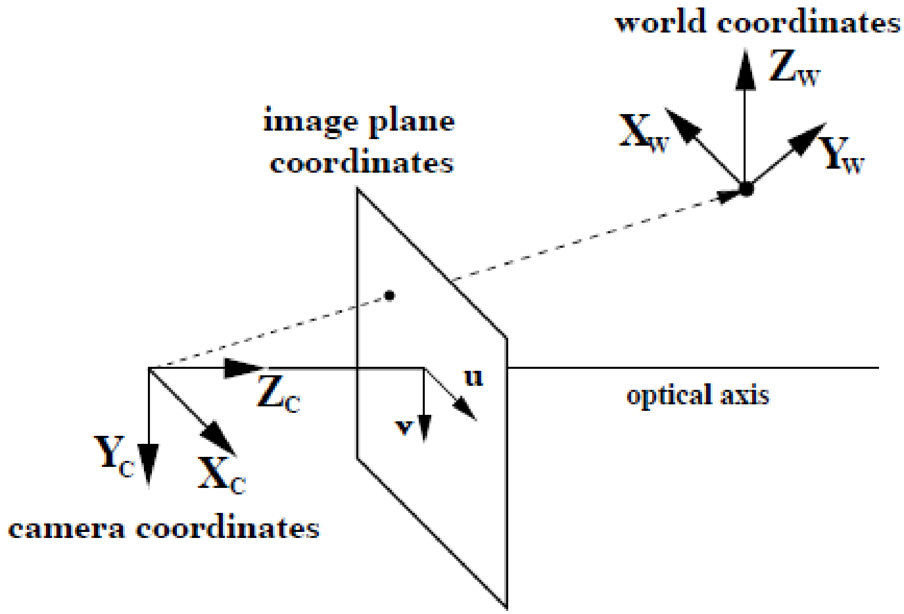
Pixel Values

Intrinsic Calibration Matrix

Camera Pose

World Point

Visual Odometry



Known

Measurement

To be Estimated

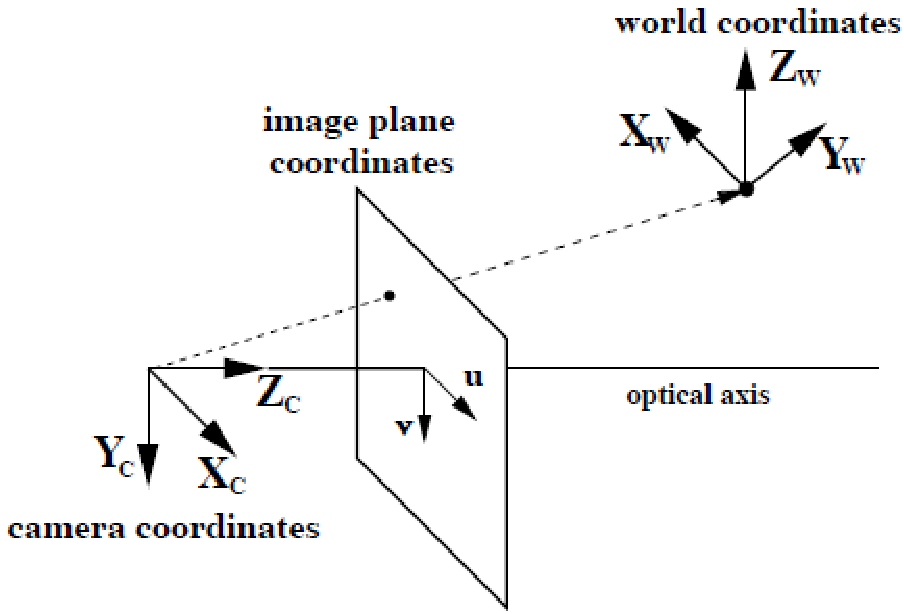
$$\lambda \boxed{\begin{pmatrix} u \\ v \\ 1 \end{pmatrix}} = \boxed{\begin{pmatrix} f & 0 & u_o \\ 0 & f & v_o \\ 0 & 0 & 1 \end{pmatrix}} \boxed{\begin{pmatrix} R & t \end{pmatrix}} \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix} = P \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix}$$

Pixel Values

Intrinsic Calibration Matrix

Camera (Incremental) Pose

SLAM



Known

Measurement

To be Estimated

$$\lambda \boxed{\begin{pmatrix} u \\ v \\ 1 \end{pmatrix}} = \boxed{\begin{pmatrix} f & 0 & u_o \\ 0 & f & v_o \\ 0 & 0 & 1 \end{pmatrix}} \boxed{\begin{pmatrix} R & t \end{pmatrix}} \boxed{\begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix}} = P \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix}$$

Pixel Values

Intrinsic Calibration Matrix

Camera Pose

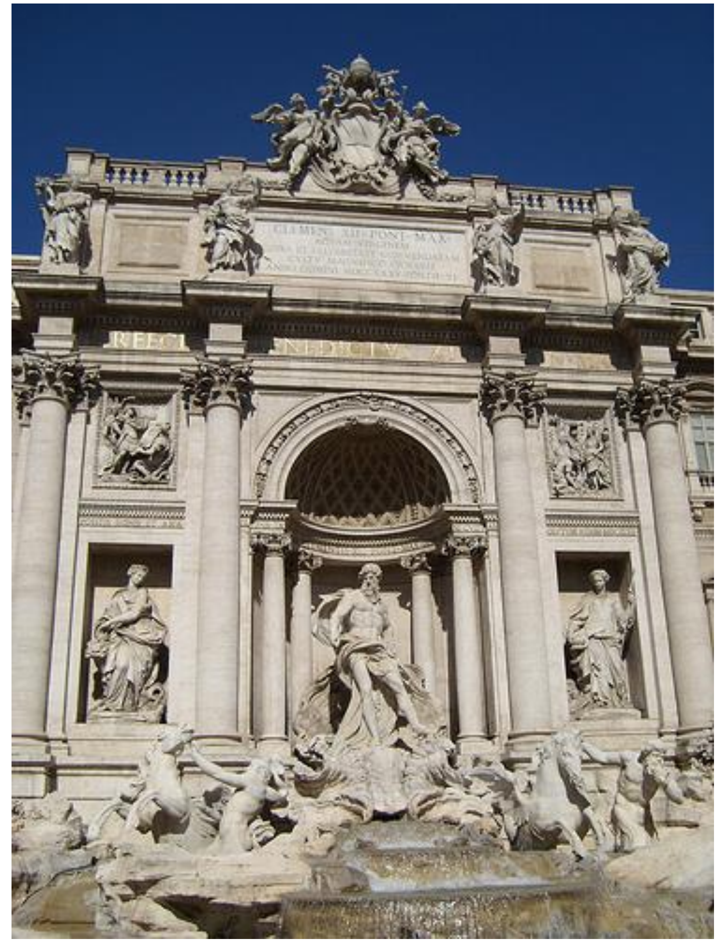
World Point

Point Feature Detection & Matching

Courtesy Steve Seitz and Richard Szeliski

Image matching

by Diva Sian



by swashford

Courtesy Steve Seitz and Richard Szeliski

Harder case

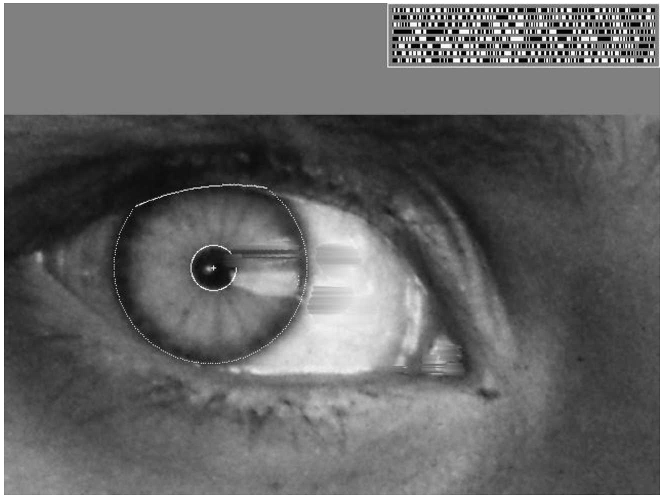
by Diva Sian

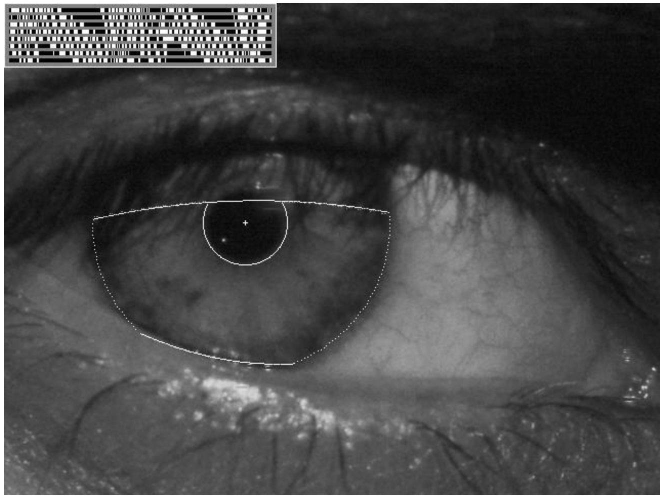
by scgbt

Courtesy Steve Seitz and Richard Szeliski

Even harder case

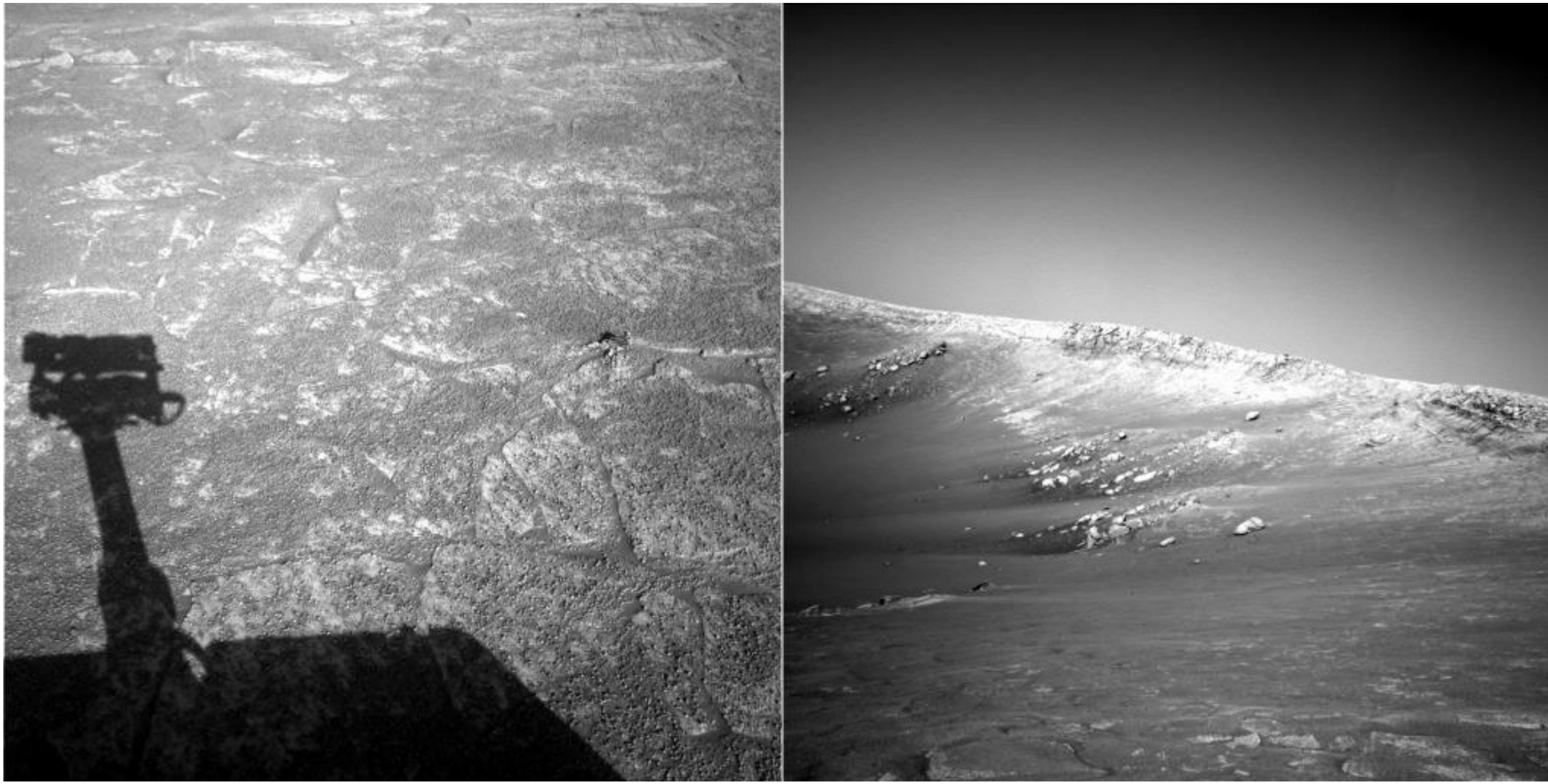
"How the Afghan Girl was Identified by Her Iris Patterns" **Read the story**





Courtesy Steve Seitz and Richard Szeliski

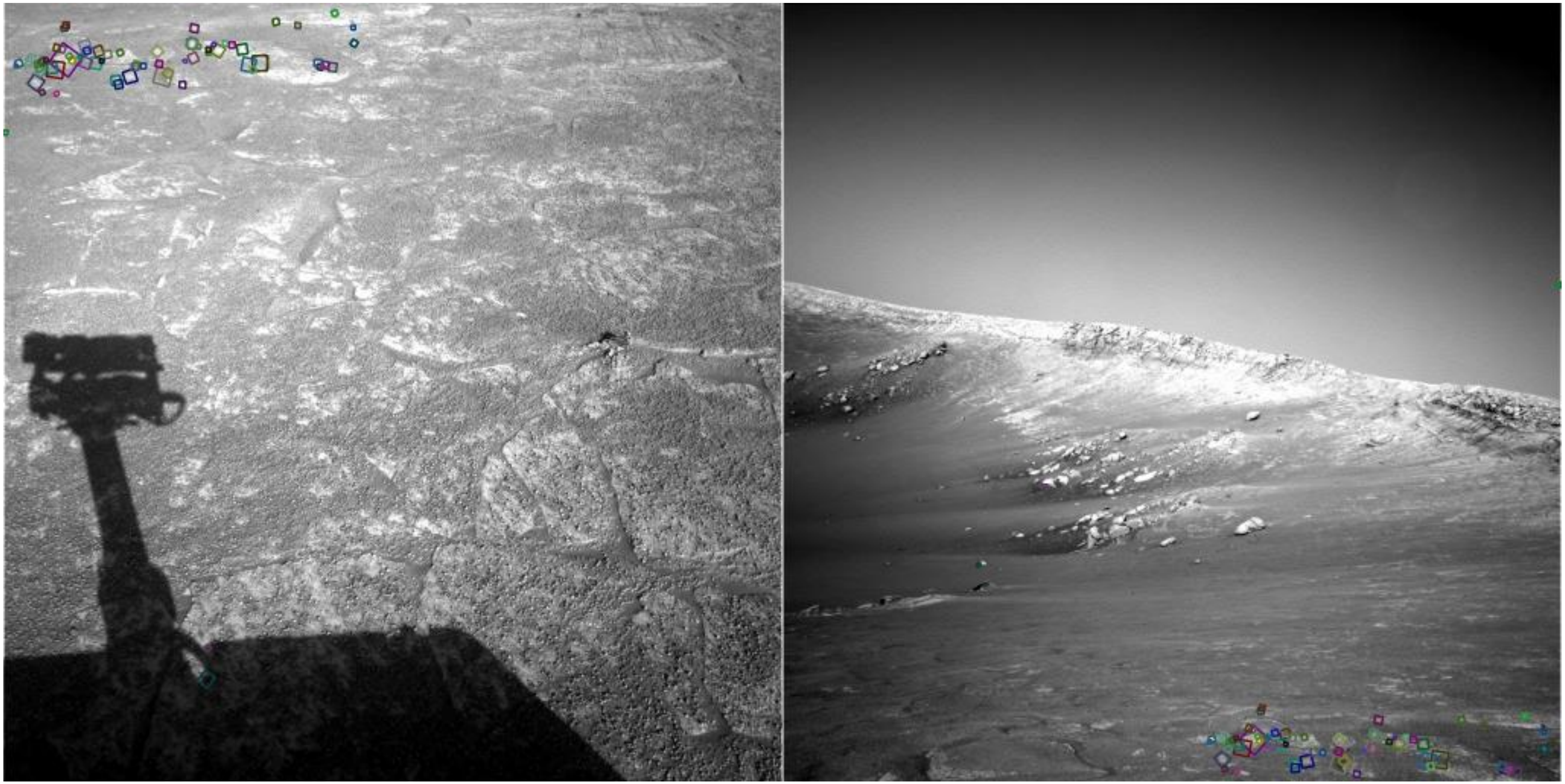
Harder still?



NASA Mars Rover images

Courtesy Steve Seitz and Richard Szeliski

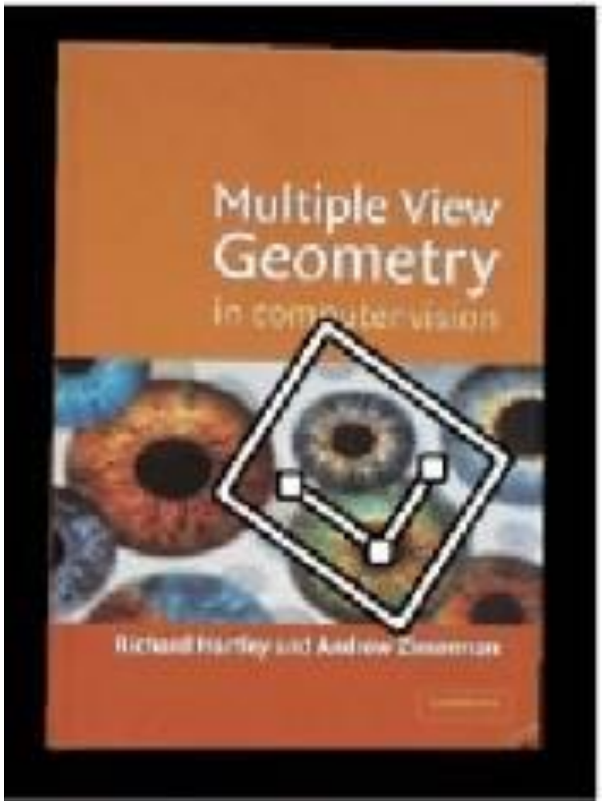
Answer below (look for tiny colored squares…)



NASA Mars Rover images
with SIFT feature matches
Figure by Noah Snavely

Courtesy Steve Seitz and Richard Szeliski

Image Matching



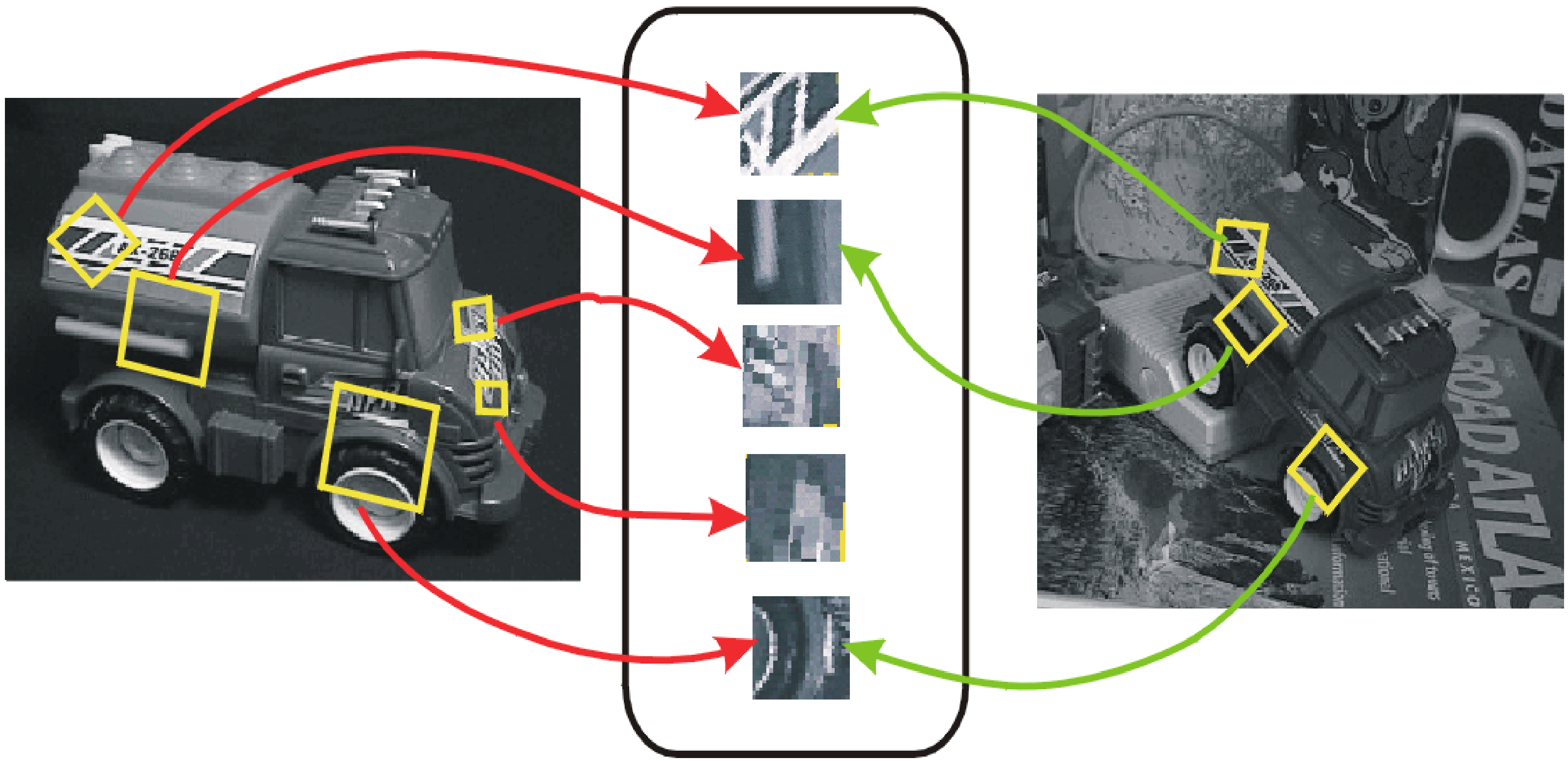
Courtesy Steve Seitz and Richard Szeliski

Image Matching

Courtesy Steve Seitz and Richard Szeliski

Invariant local features

- Find features that are invariant to transformations
  - geometric invariance: translation, rotation, scale
  - photometric invariance: brightness, exposure, …



Feature Descriptors

Courtesy Steve Seitz and Richard Szeliski

Advantages of local features

- Locality
  - Features are local, so robust to occlusion and clutter
- Distinctiveness
  - Can differentiate a large database of objects
- Quantity
  - Hundreds or thousands in a single image
- Efficiency
  - Real-time performance achievable
- Generality
  - Exploit different types of features in different situations

Courtesy Steve Seitz and Richard Szeliski

More motivation…

- Feature points are used for:
  - Image alignment (e.g., mosaics)
  - 3D reconstruction
  - Motion tracking
  - Object recognition
  - Indexing and database retrieval
  - Robot navigation
  - … other

Courtesy Steve Seitz and Richard Szeliski

Image Mosaics

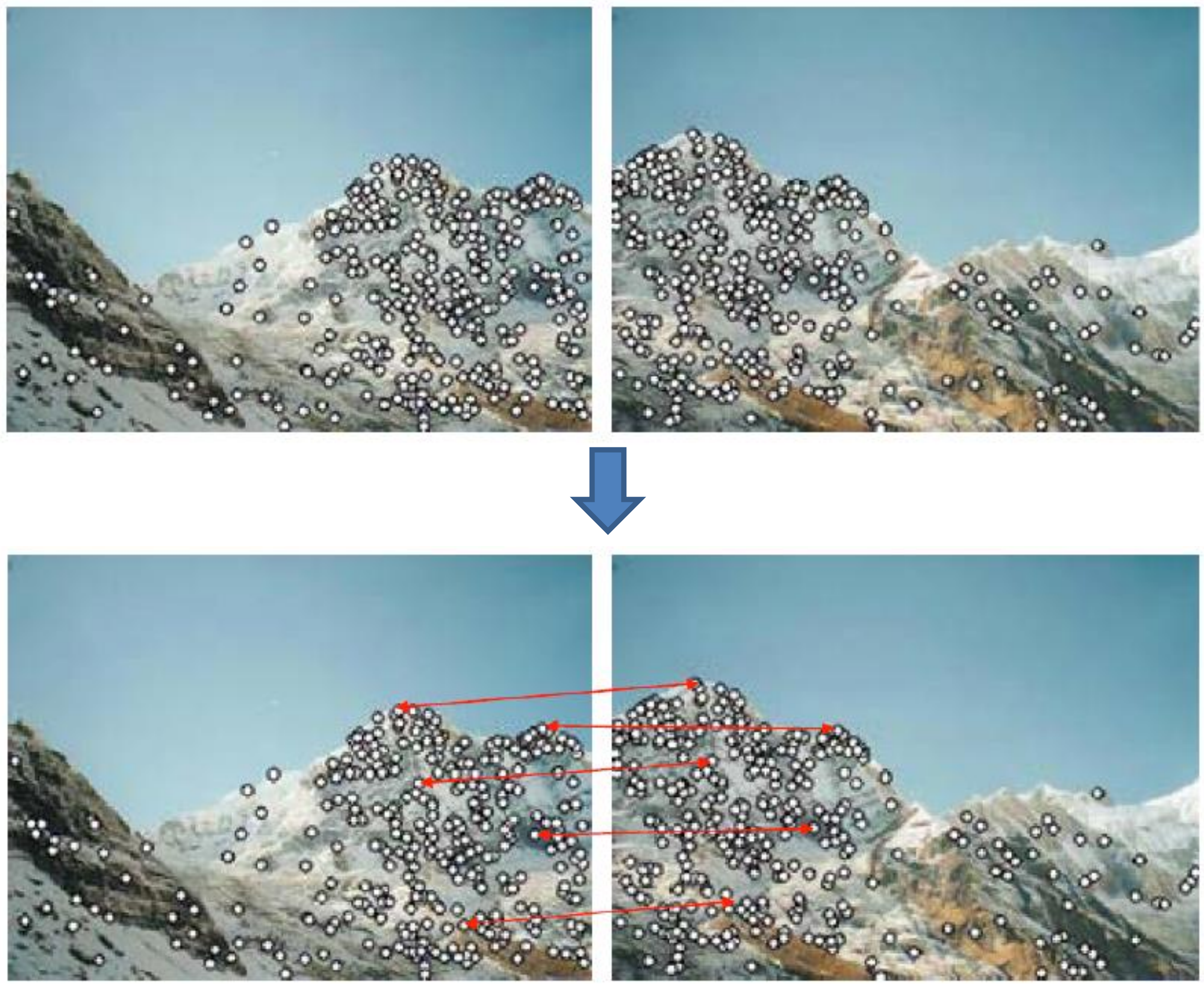
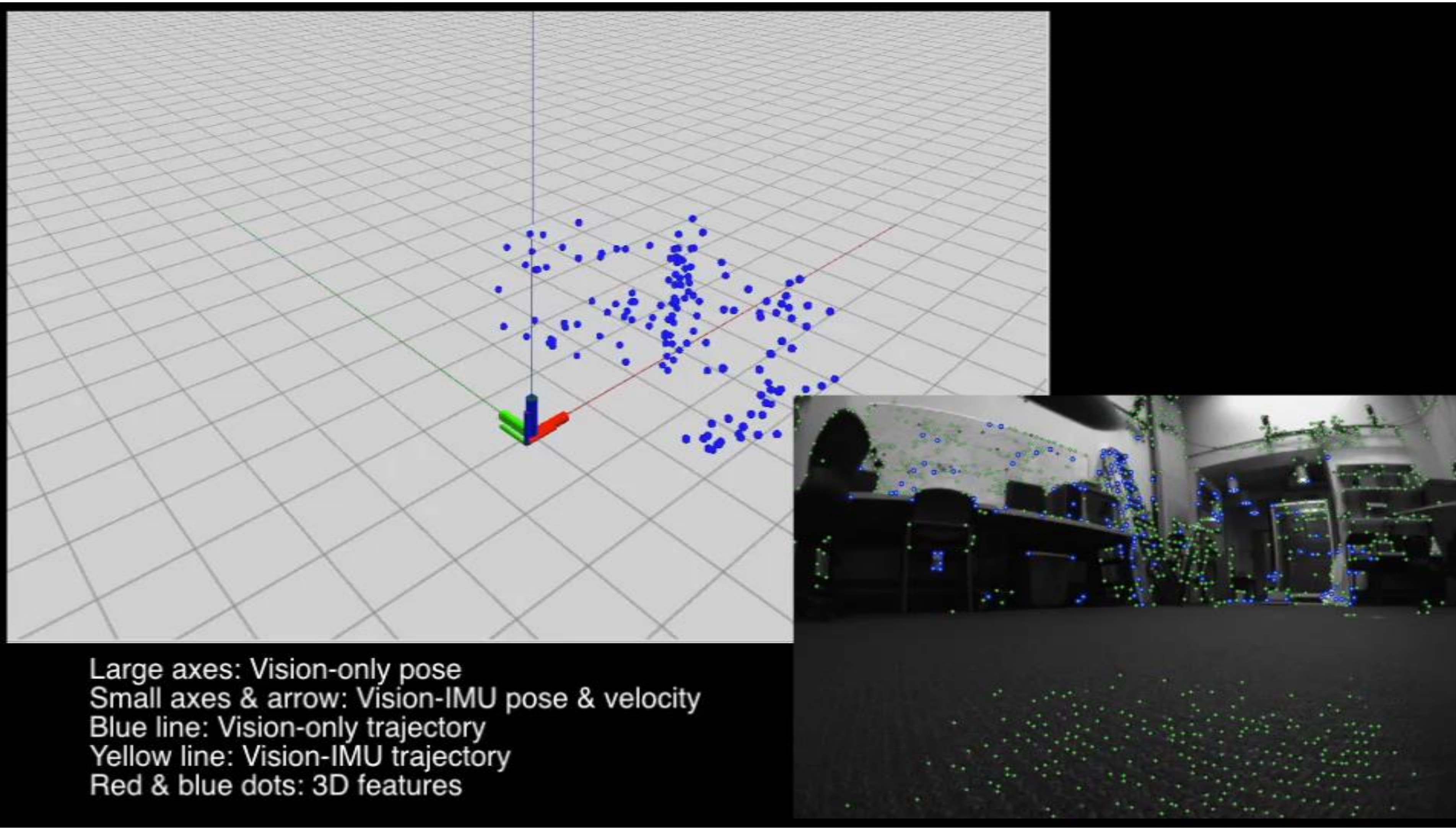


Image Mosaics

Motion Tracking



Want uniqueness

- Look for image regions that are unusual
  - Lead to unambiguous matches in other images

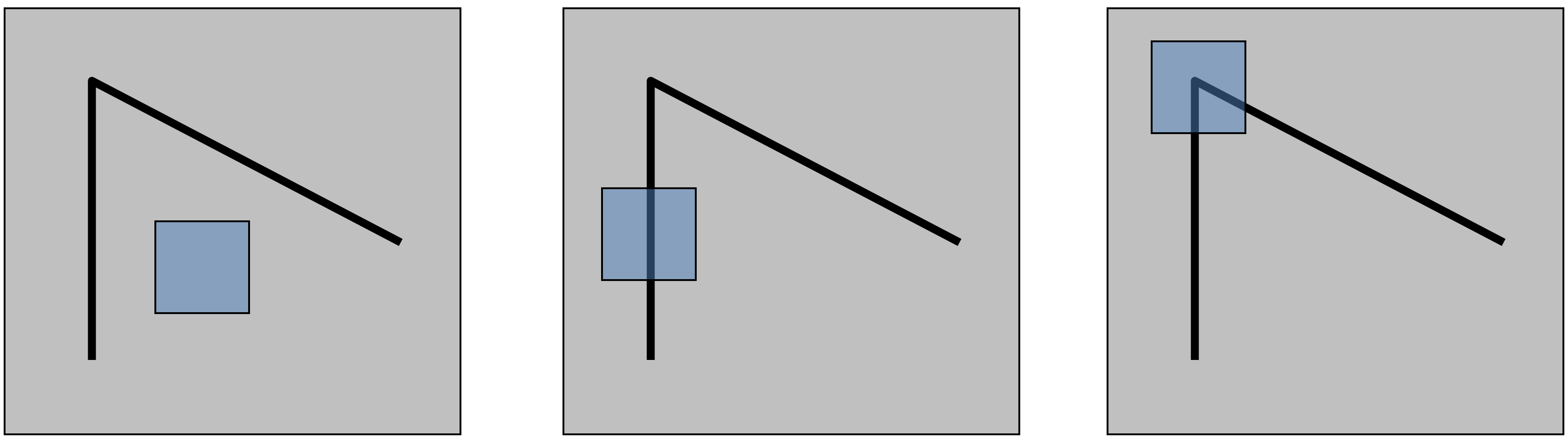
- How to define "unusual"?

Courtesy Steve Seitz and Richard Szeliski

Local measures of uniqueness

Suppose we only consider a small window of pixels

- What defines whether a feature is a good or bad candidate?

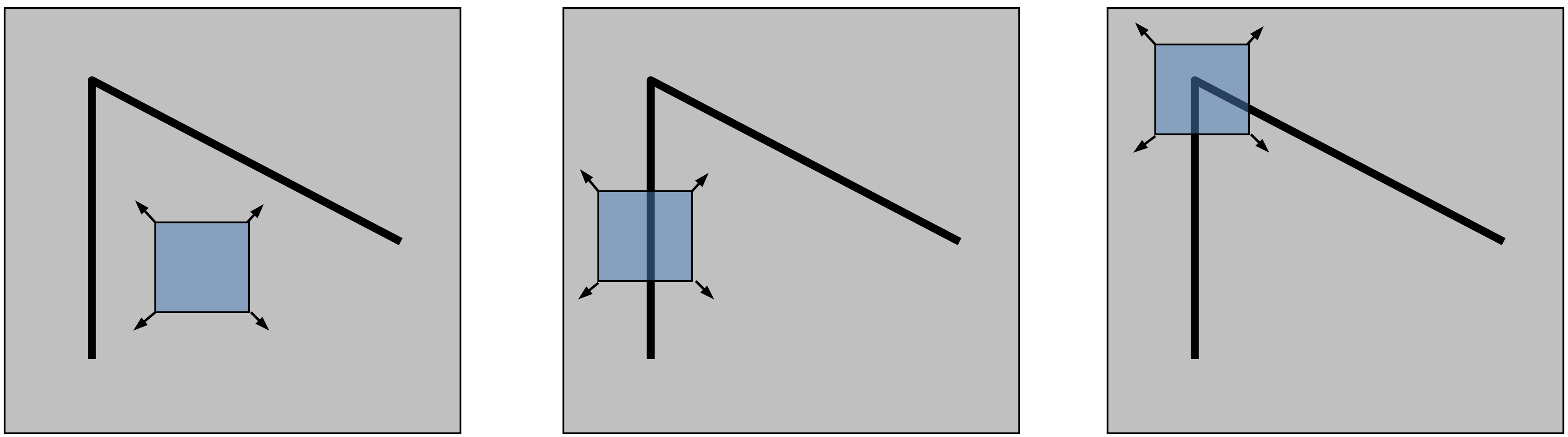


Slide adapted from Darya Frolova, Denis Simakov, Weizmann Institute.

Courtesy Steve Seitz and Richard Szeliski

Feature detection

- Local measure of feature uniqueness
  - How does the window change when you shift it?
  - Shifting the window in *any direction* causes a *big change*



"flat" region: no change in all directions

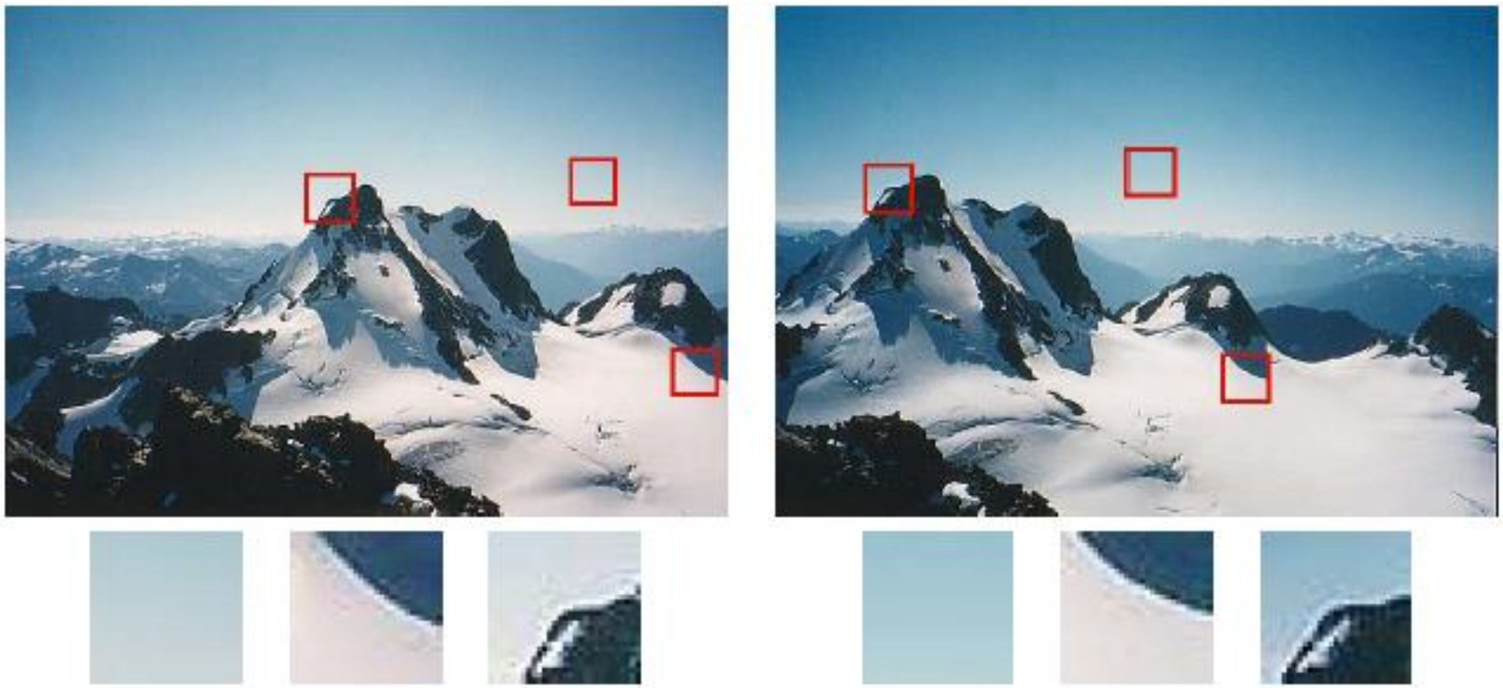
"edge": no change along the edge direction

"corner": significant change in all directions

Slide adapted from Darya Frolova, Denis Simakov, Weizmann Institute.

Courtesy Steve Seitz and Richard Szeliski

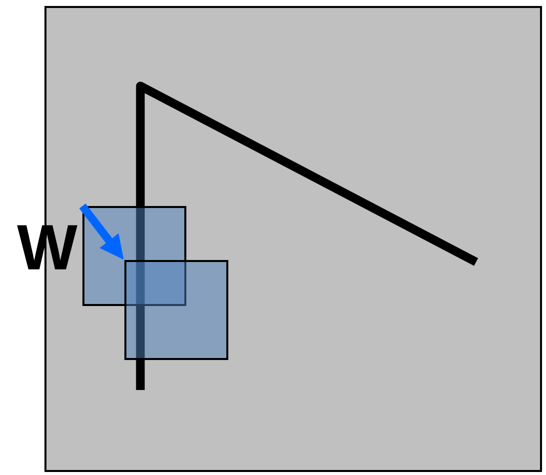
Feature detection



Courtesy Steve Seitz and Richard Szeliski

Feature detection: the math

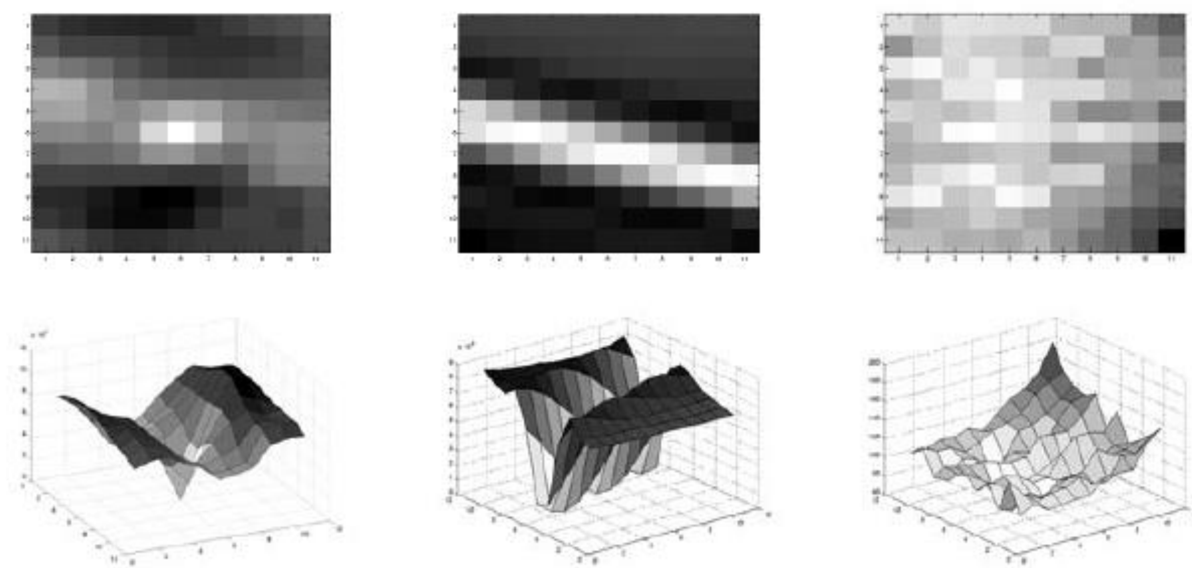
- Consider shifting the window **W** by (u,v)
  - How do the pixels in **W** change?
  - Compare each pixel before and after by summing up the squared differences (SSD)
  - This defines an SSD "error" of *E(u,v)*:



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

Courtesy Steve Seitz and Richard Szeliski

(a)



Courtesy Steve Seitz and Richard Szeliski

Small motion assumption

- Taylor Series expansion of I:

$$I(x+u, y+v) = I(x,y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \text{higher order terms}$$

- If the motion (u,v) is small, then first order approximation is good

$$I(x+u, y+v) \approx I(x,y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v$$

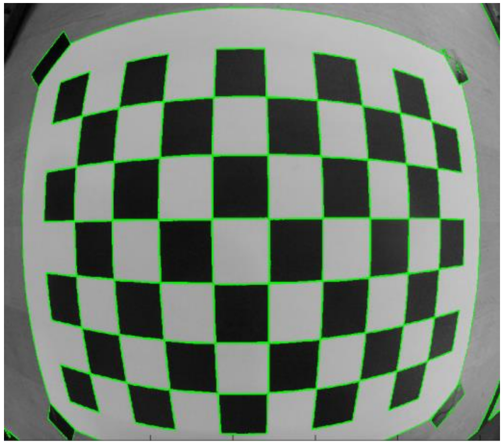
$$\approx I(x,y) + \begin{bmatrix} I_x & I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

$$\text{shorthand: } I_x = \frac{\partial I}{\partial x}$$

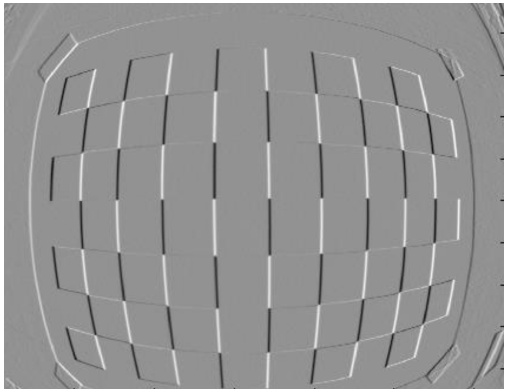
- Plugging this into the formula on the previous slide…

Courtesy Steve Seitz and Richard Szeliski

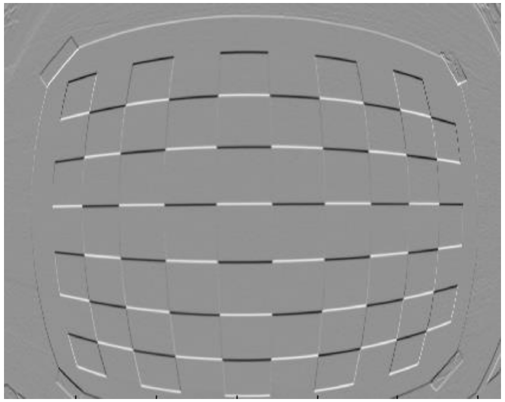
Image Gradients



I



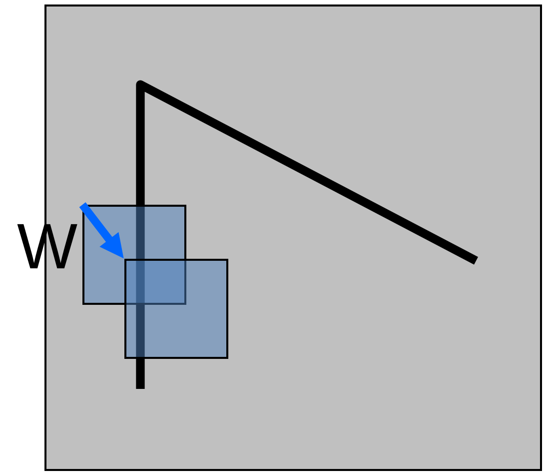
$$I_x = I(x+1, y) - I(x-1, y)$$



$$I_y = I(x, y+1) - I(x, y-1)$$

Feature detection: the math

- Consider shifting the window **W** by (u,v)
  - How do the pixels in **W** change?
  - Compare each pixel before and after by summing up the squared differences
  - This defines an "error" of *E(u,v)*:



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

$$\approx \sum_{(x,y)\in W} \left[I(x,y) + [I_x \quad I_y] \begin{bmatrix} u \\ v \end{bmatrix} - I(x,y)\right]^2$$

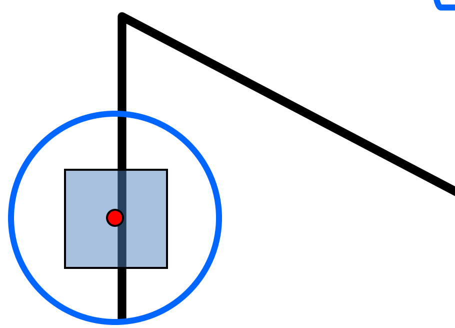
$$\approx \sum_{(x,y)\in W} \left[[I_x \quad I_y] \begin{bmatrix} u \\ v \end{bmatrix}\right]^2$$

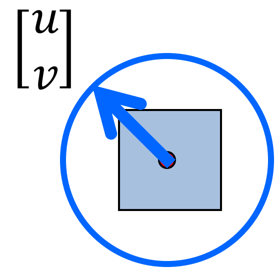
Courtesy Steve Seitz and Richard Szeliski

Feature detection: the math

- This can be rewritten:

$$E(u,v) = \sum_{(x,y)\in W} [u \quad v] \underbrace{\begin{bmatrix} I_x^2 & I_x I_y \\ I_y I_x & I_y^2 \end{bmatrix}}_{\boldsymbol{H}} \begin{bmatrix} u \\ v \end{bmatrix}$$





- For the example above
  - You can move the center of the green window to anywhere on the blue unit circle
  - Which directions will result in the largest and smallest E values?
  - We can find these directions by looking at the eigenvectors of *H*

Courtesy Steve Seitz and Richard Szeliski

Quick eigenvalue/eigenvector review

- The **eigenvectors** of a matrix **A** are the vectors **x** that satisfy:

$$Ax = \lambda x$$

- The scalar λ is the **eigenvalue** corresponding to **x**
  - The eigenvalues are found by solving:

$$det(A - \lambda I) = 0$$

  - In our case, ***A*** = ***H*** is a 2x2 matrix, so we have

$$det\begin{bmatrix} h_{11} - \lambda & h_{12} \\ h_{21} & h_{22} - \lambda \end{bmatrix} = 0$$

  - The solution:

$$\lambda_{\pm} = \frac{1}{2}\left[(h_{11} + h_{22}) \pm \sqrt{4h_{12}h_{21} + (h_{11} - h_{22})^2}\right]$$

- Once you know λ, you find **x** by solving

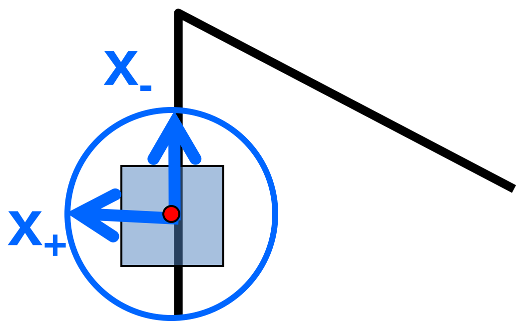
$$\begin{bmatrix} h_{11} - \lambda & h_{12} \\ h_{21} & h_{22} - \lambda \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} = 0$$

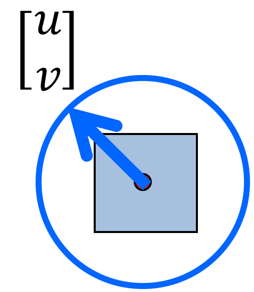
Courtesy Steve Seitz and Richard Szeliski

Feature detection: the math

- This can be rewritten:

$$E(u,v) = \sum_{(x,y)\in W} \begin{bmatrix} u & v \end{bmatrix} \underbrace{\begin{bmatrix} I_x^2 & I_x I_y \\ I_y I_x & I_y^2 \end{bmatrix}}_{\boldsymbol{H}} \begin{bmatrix} u \\ v \end{bmatrix}$$





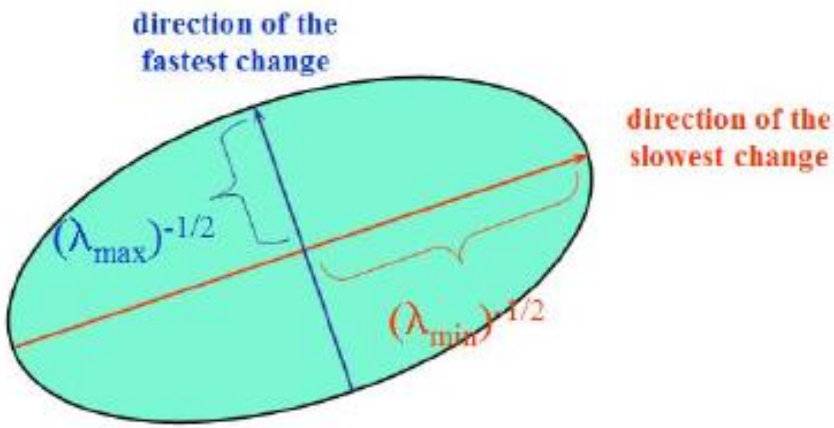
- Eigenvalues and eigenvectors of H
  - Define shifts with the smallest and largest change (E value)
  - x_+ = direction of **largest** increase in E.
  - λ_+ = amount of increase in direction x_+
  - x_- = direction of **smallest** increase in E.
  - λ- = amount of increase in direction x_+

$$Hx_+ = \lambda_+ x_+$$
$$Hx_- = \lambda_- x_-$$

Courtesy Steve Seitz and Richard Szeliski

Feature detection: the math

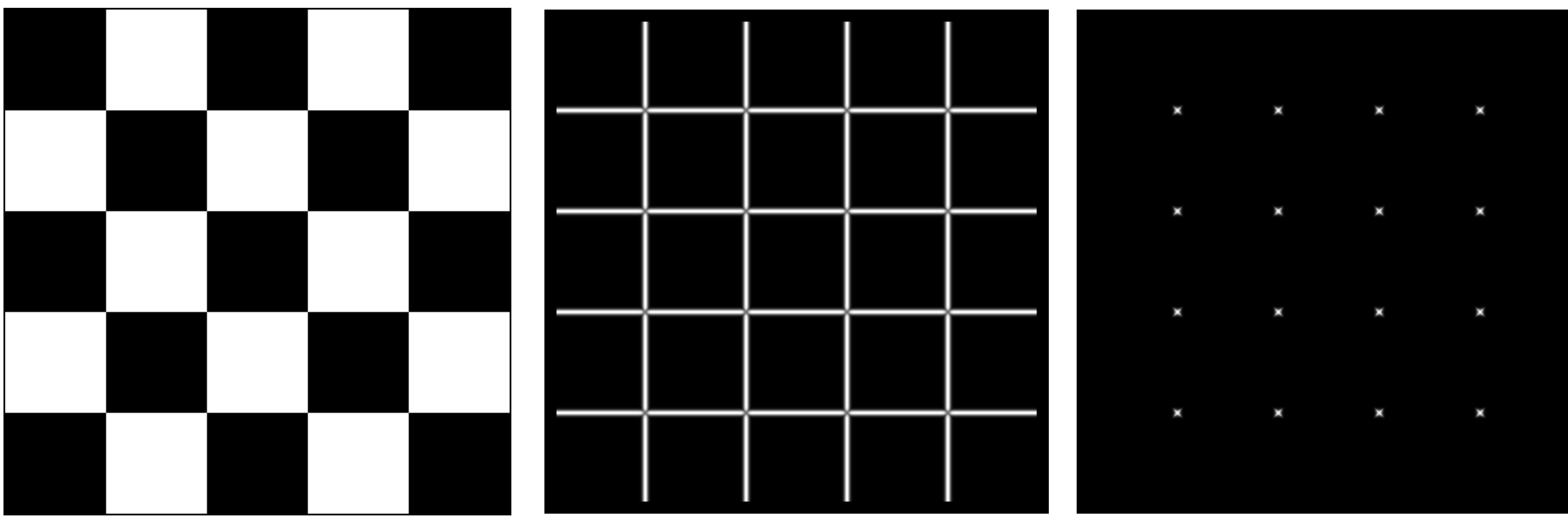
- How are λ_+, $\mathbf{x}_+$, λ_-, and $\mathbf{x}_-$ relevant for feature detection?
  - What's our feature scoring function?



Courtesy Steve Seitz and Richard Szeliski

Feature detection: the math

- How are λ_+, $\mathbf{x}_+$, λ_-, and $\mathbf{x}_-$ relevant for feature detection?
  - What's our feature scoring function?
- Want *E(u,v)* to be ***large*** for small shifts in ***all*** directions
  - the *minimum* of *E(u,v)* should be large, over all unit vectors [u v]
  - this minimum is given by the smaller eigenvalue (λ_-) of ***H***

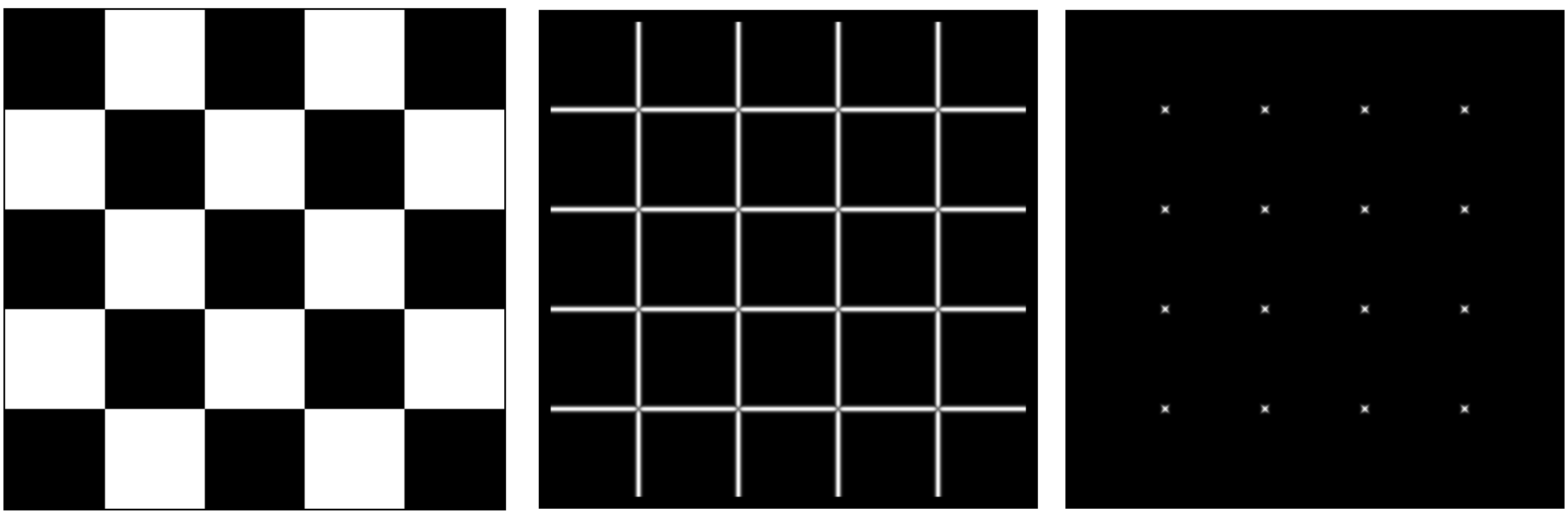


I λ_+ λ_-

Courtesy Steve Seitz and Richard Szeliski

Feature detection summary

- Here's what you do
  - Compute the gradient at each point in the image
  - Create the ***H*** matrix from the entries in the gradient
  - Compute the eigenvalues.
  - Find points with large response (λ_- > threshold)
  - Choose those points where λ_- is a local maximum as features

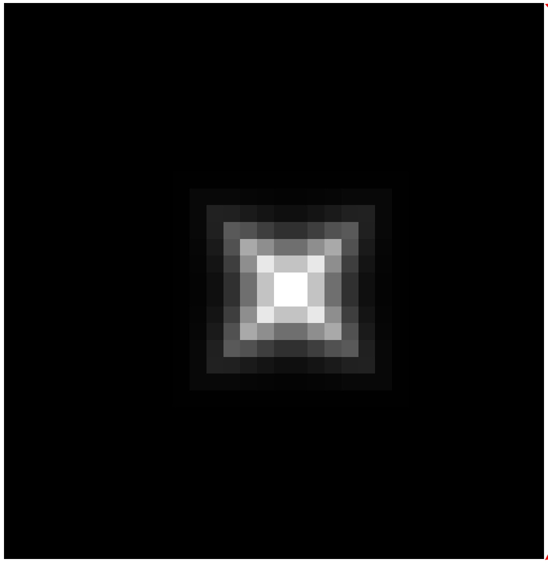


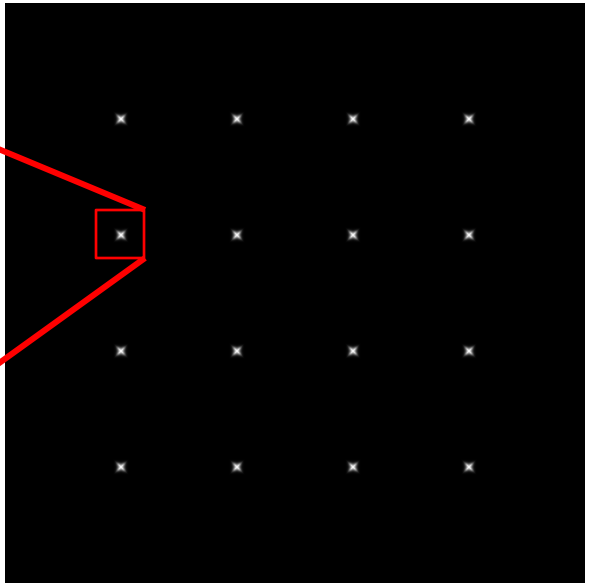
I λ_+ λ_-

Courtesy Steve Seitz and Richard Szeliski

Feature detection summary

- Here's what you do
  - Compute the gradient at each point in the image
  - Create the ***H*** matrix from the entries in the gradient
  - Compute the eigenvalues.
  - Find points with large response (λ_- > threshold)
  - Choose those points where λ_- is a local maximum as features





λ_-

Courtesy Steve Seitz and Richard Szeliski

The Harris operator

- λ_- is a variant of the "Harris operator" for feature detection

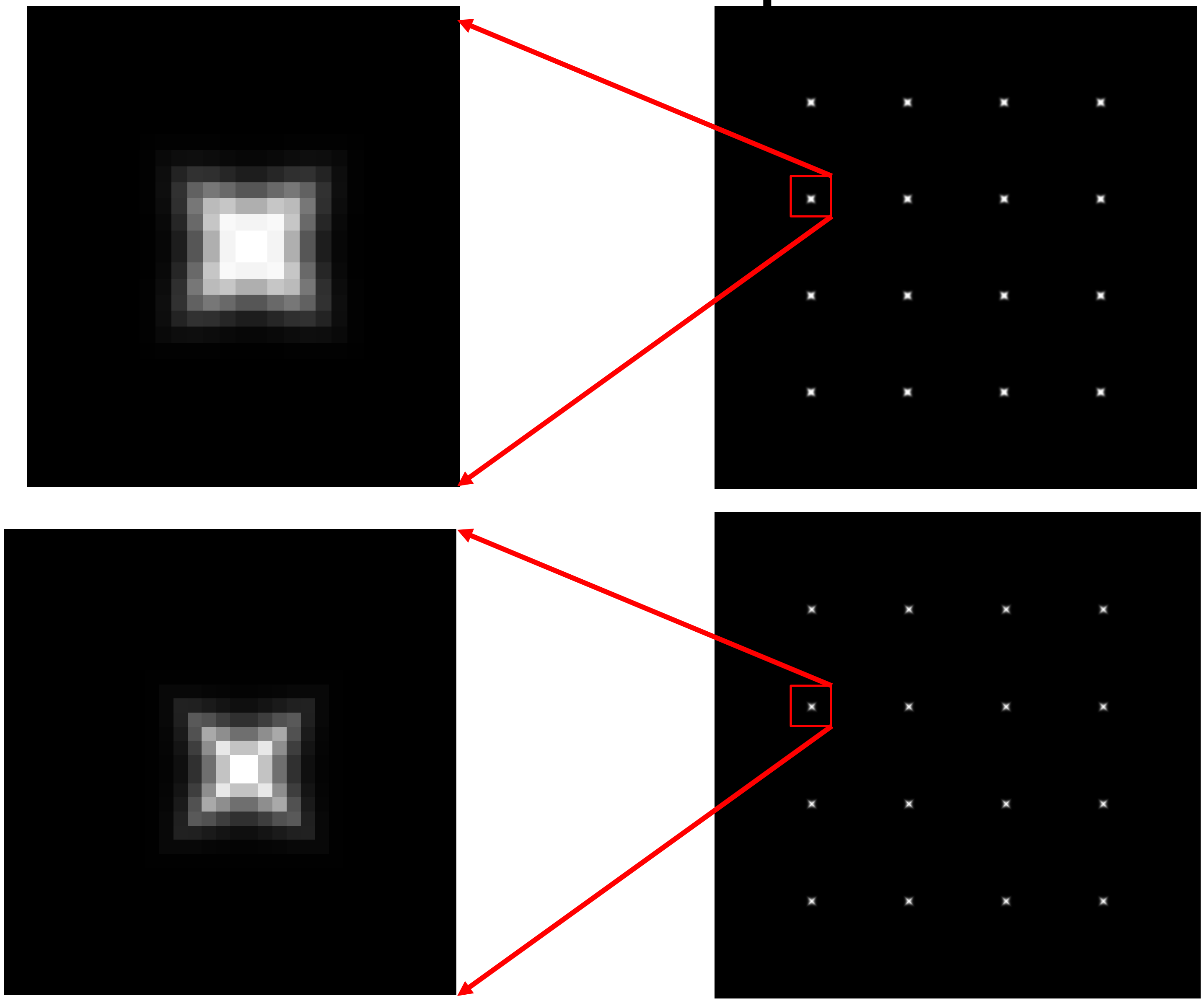
$$f = \frac{\lambda_1 \lambda_2}{\lambda_1 + \lambda_2}$$

$$= \frac{determinant(H)}{trace(H)}$$

  - The *trace* is the sum of the diagonals, i.e., $trace(H) = h_{11} + h_{22}$
  - Very similar to λ_- but less expensive (no square root)
  - Called the "Harris Corner Detector" or "Harris Operator"
  - Lots of other detectors, this is one of the most popular

Courtesy Steve Seitz and Richard Szeliski

The Harris operator



Harris operator

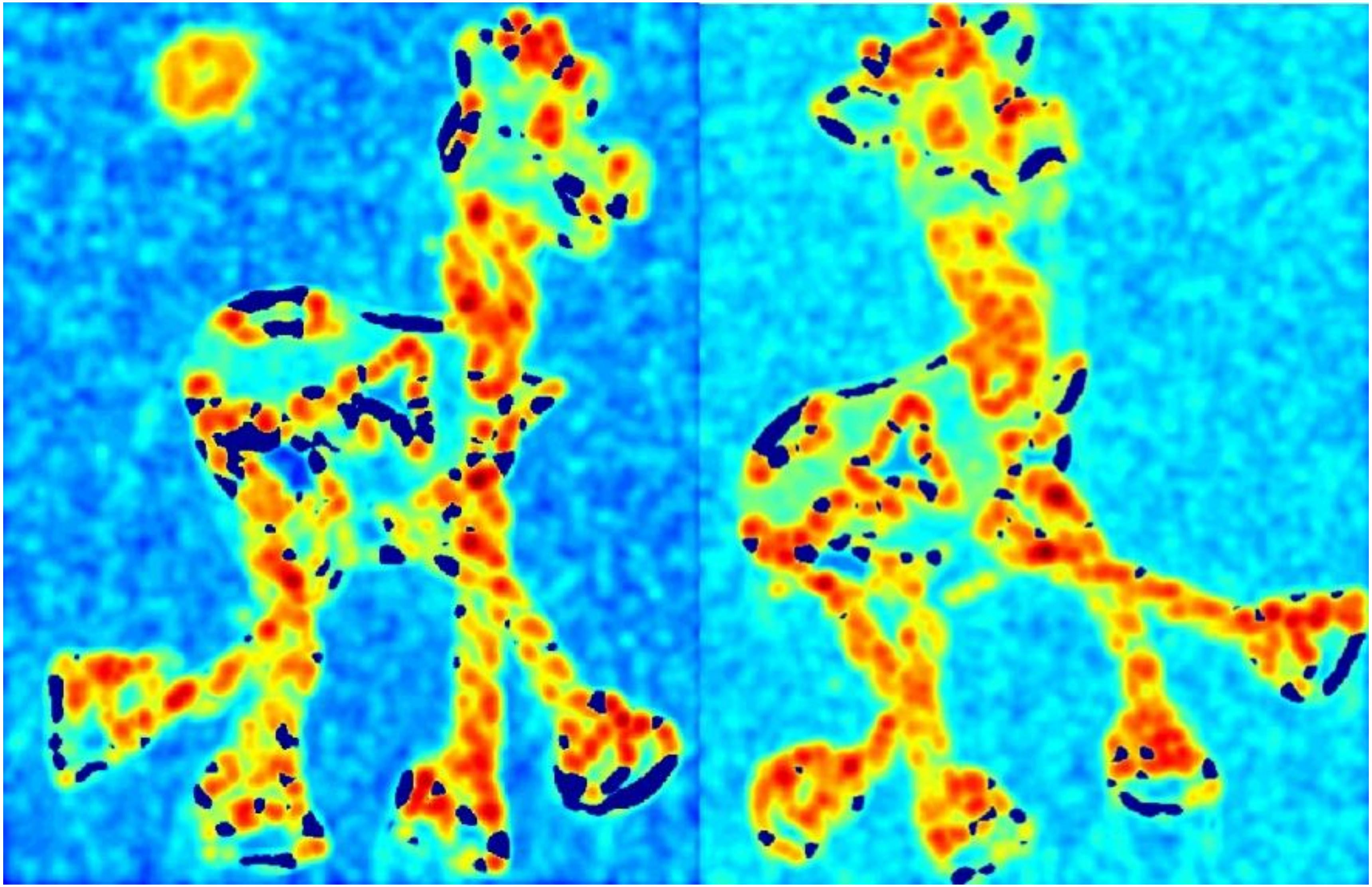
λ_-

Courtesy Steve Seitz and Richard Szeliski

Harris detector example

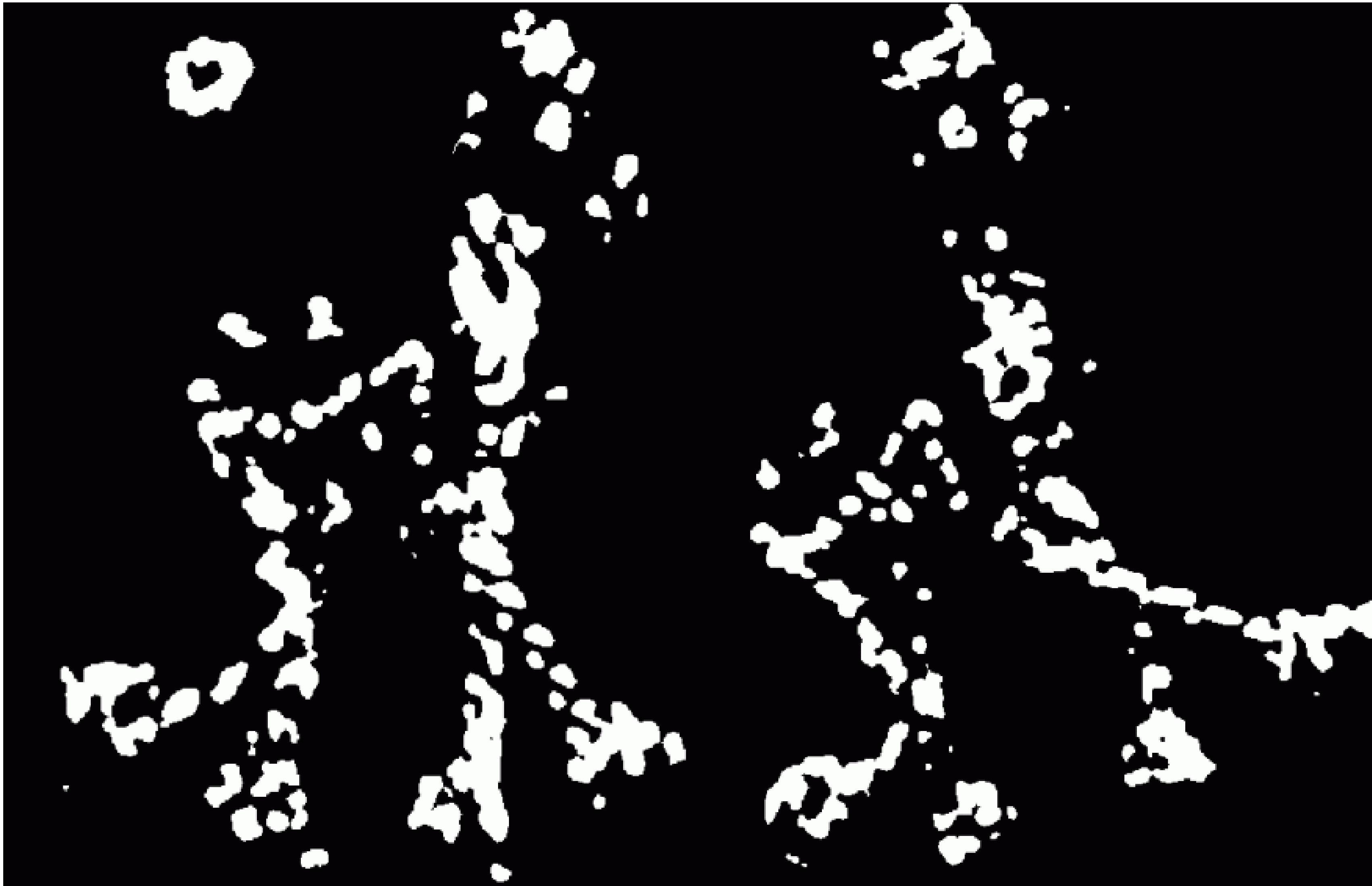
Courtesy Steve Seitz and Richard Szeliski

f value (red high, blue low)



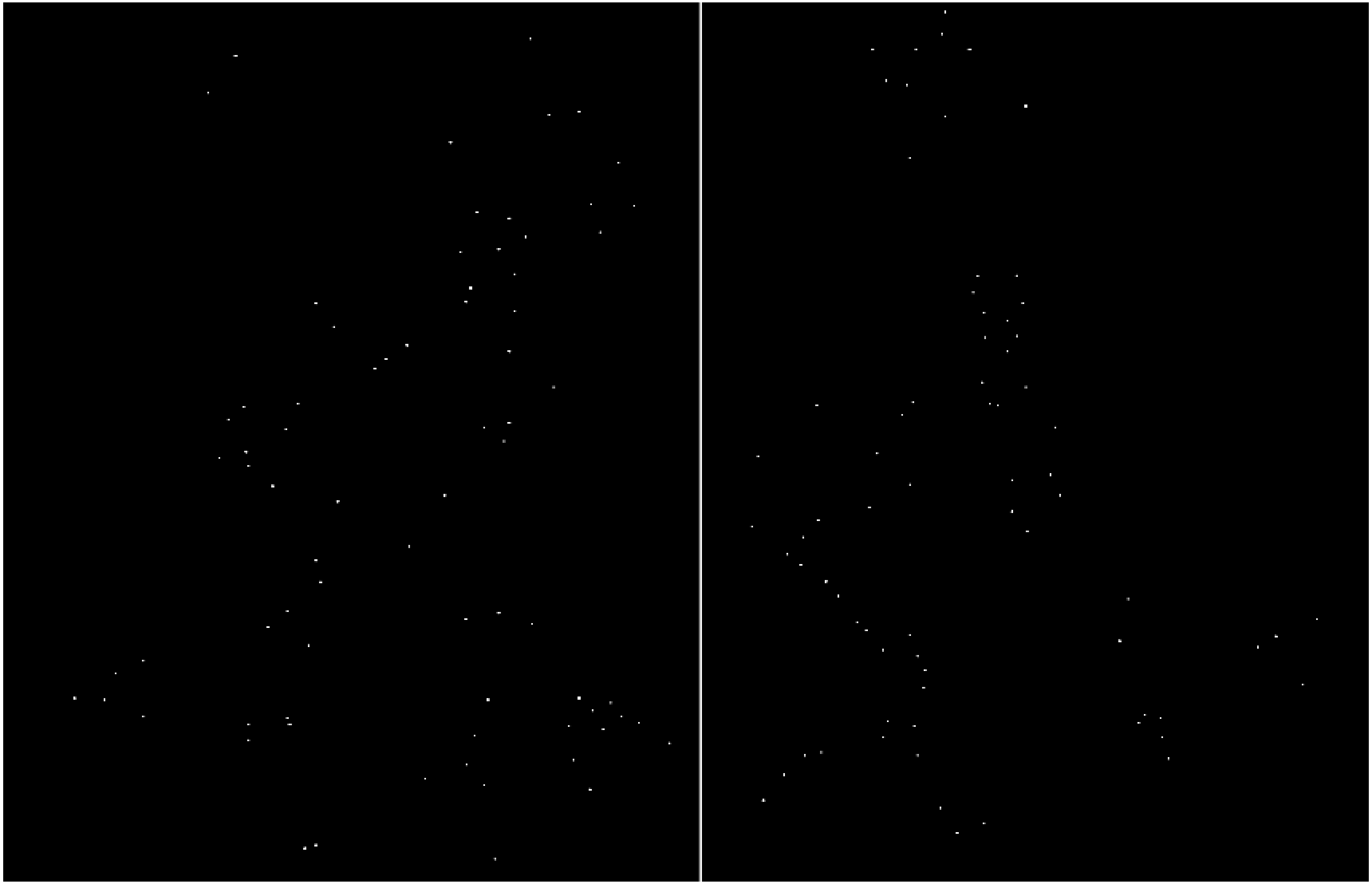
Courtesy Steve Seitz and Richard Szeliski

Threshold (f > value)



Courtesy Steve Seitz and Richard Szeliski

Find local maxima of f



Courtesy Steve Seitz and Richard Szeliski

Harris features (in red)

Courtesy Steve Seitz and Richard Szeliski

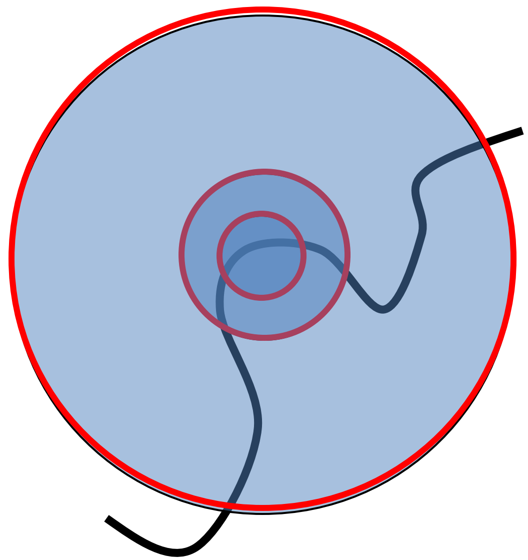
Invariance

- Suppose you **rotate** the image by some angle
  - Will you still pick up the same features?
- What if you change the brightness?
- Scale?

Courtesy Steve Seitz and Richard Szeliski

Scale invariant detection

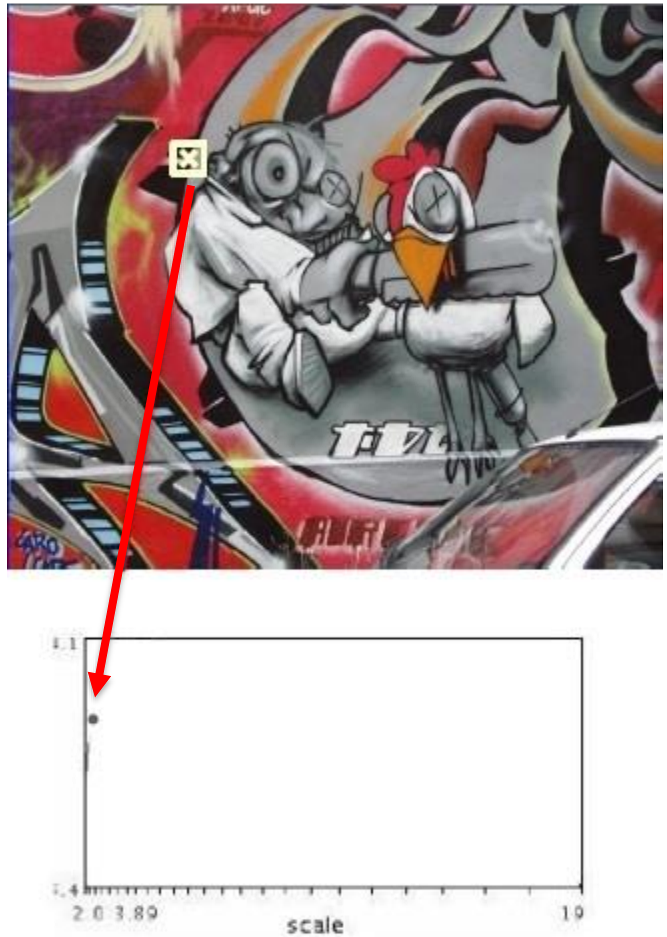
- Suppose you're looking for corners



- Key idea: find scale that gives local maximum of f
  - f is a local maximum in both position and scale

Courtesy Steve Seitz and Richard Szeliski

Automatic scale selection

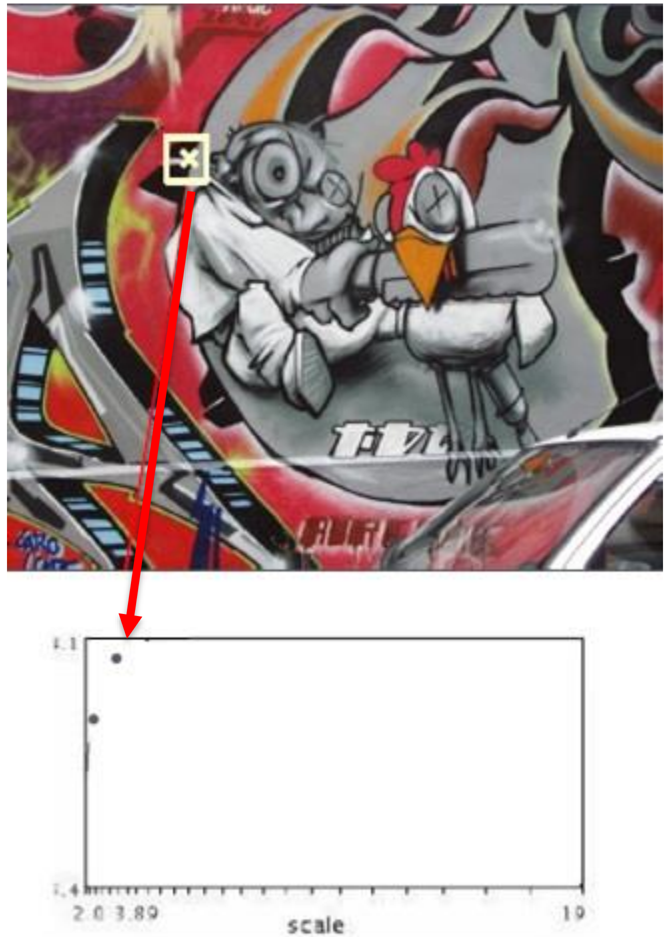


Lindeberg et al., 1996

$$f(I_{i_1 \dots i_m}(x,\sigma))$$

Courtesy Steve Seitz and Richard Szeliski

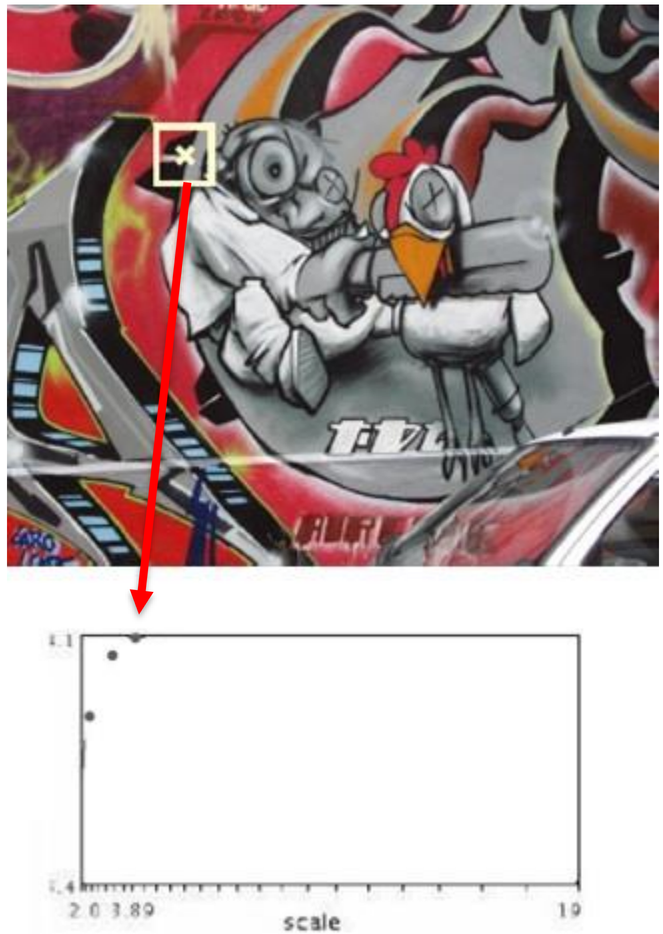
Automatic scale selection



$$f(I_{i_1 \dots i_m}(x, \sigma))$$

Courtesy Steve Seitz and Richard Szeliski

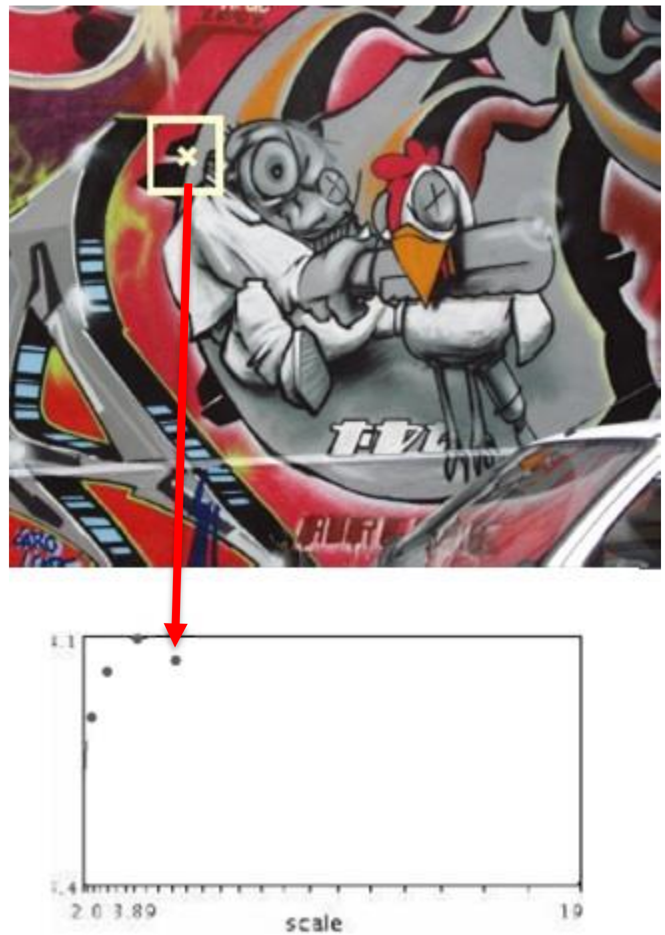
Automatic scale selection



$$f(I_{i_1 \dots i_m}(x, \sigma))$$

Courtesy Steve Seitz and Richard Szeliski

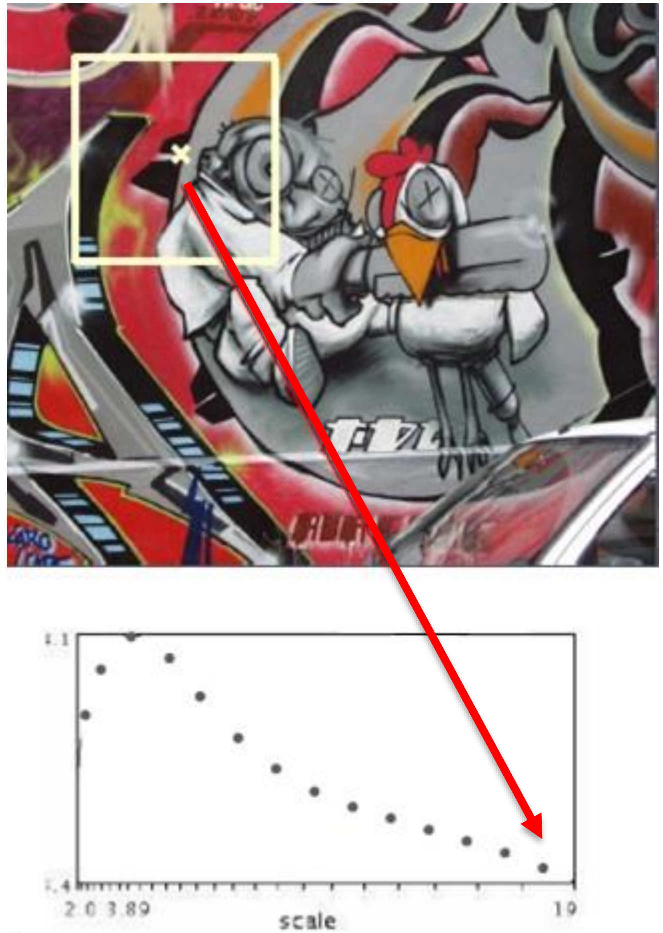
Automatic scale selection



$$f(I_{i_1 \ldots i_m}(x, \sigma))$$

Courtesy Steve Seitz and Richard Szeliski

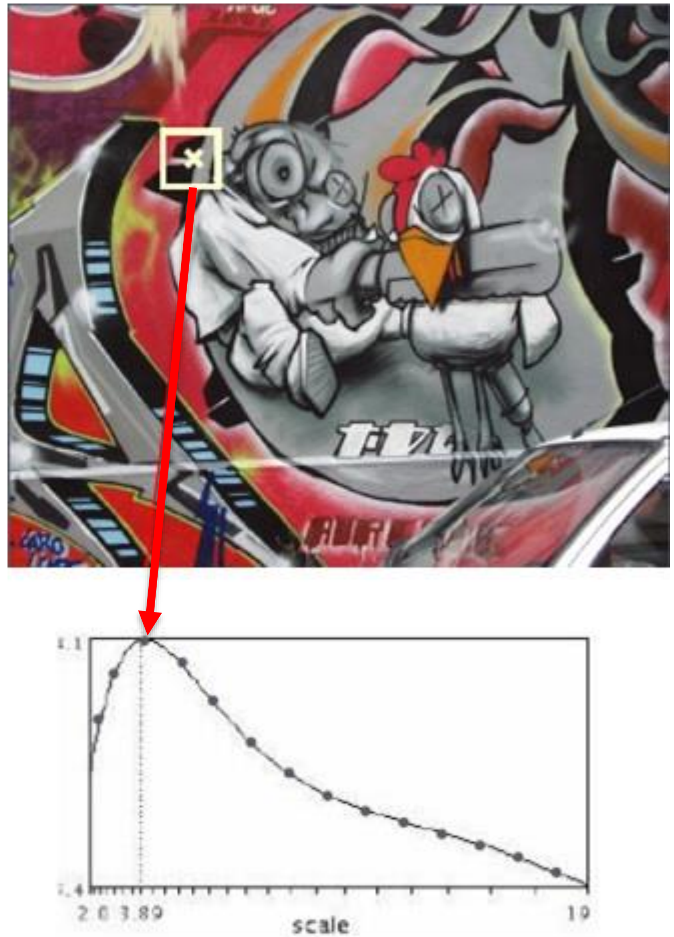
Automatic scale selection



$$f(I_{i_1 \ldots i_m}(x, \sigma))$$

Courtesy Steve Seitz and Richard Szeliski

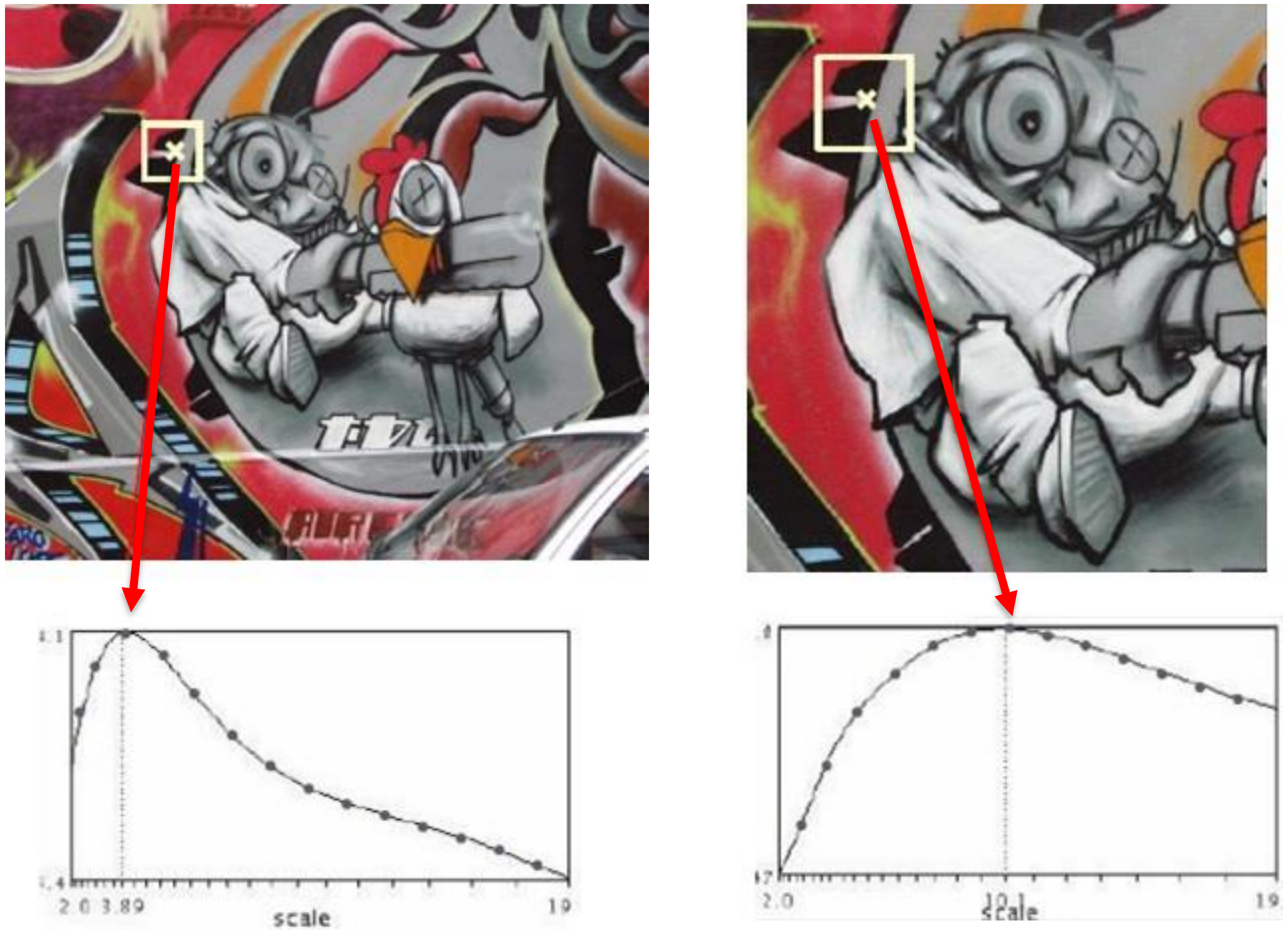
Automatic scale selection



$$f(I_{i_1 \ldots i_m}(x, \sigma))$$

Courtesy Steve Seitz and Richard Szeliski

Automatic scale selection



$$f(I_{i_1 \ldots i_m}(x, \sigma)) \qquad f(I_{i_1 \ldots i_m}(x', \sigma'))$$

Courtesy Steve Seitz and Richard Szeliski

Automatic scale selection

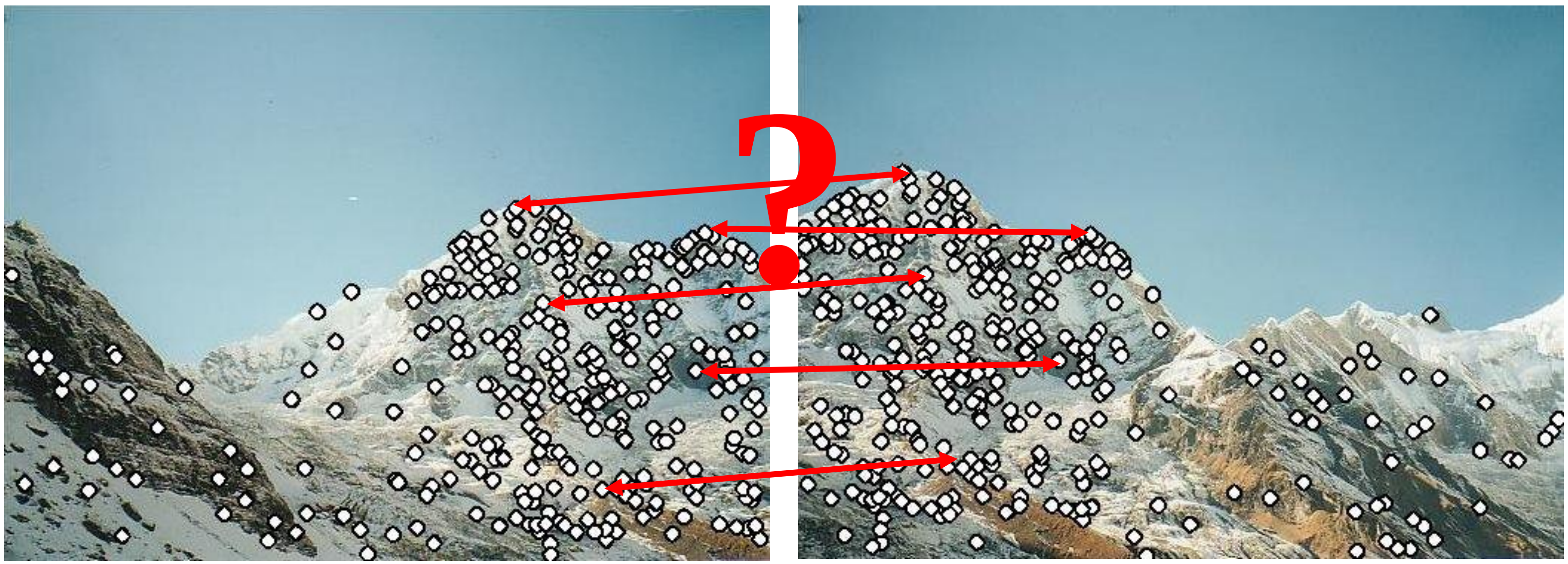
- Normalize: rescale to fixed size

Courtesy Steve Seitz and Richard Szeliski

Feature descriptors

We know how to detect good points

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

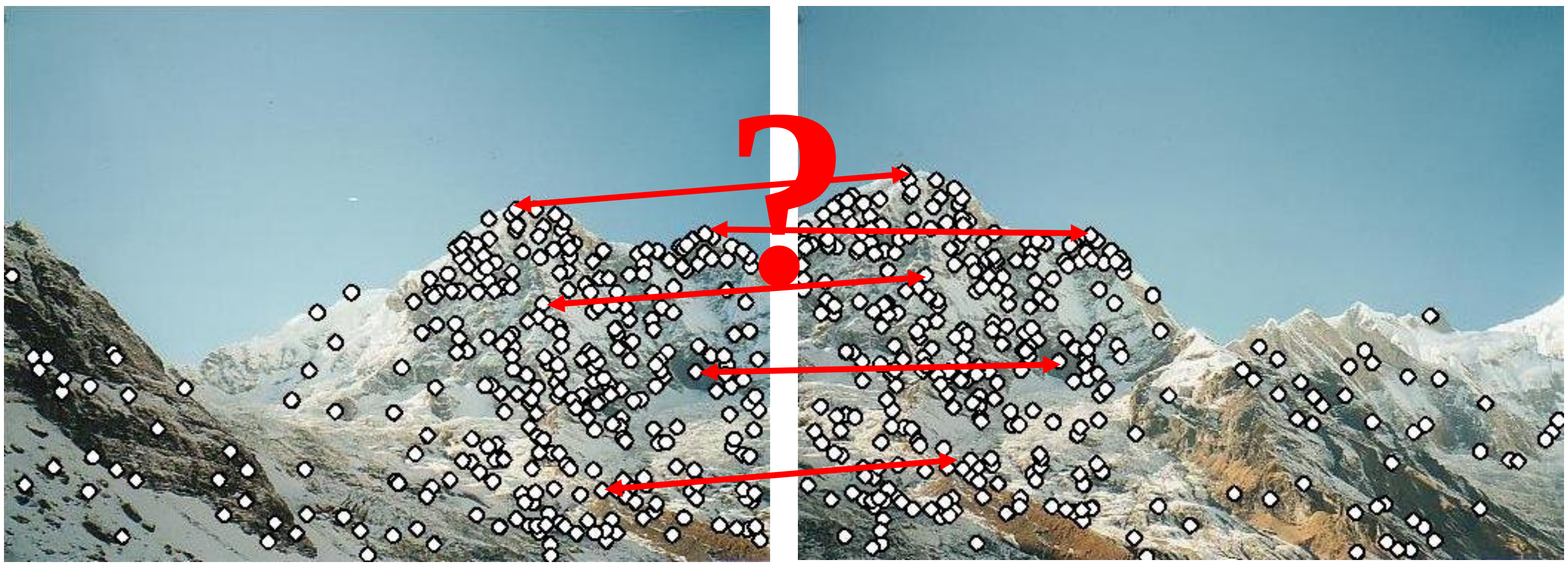


Courtesy Steve Seitz and Richard Szeliski

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
- Better approach: SIFT
  - David Lowe, UBC http://www.cs.ubc.ca/~lowe/keypoints/

Courtesy Steve Seitz and Richard Szeliski

Invariance

Suppose we are comparing two images I_1 and I_2

- I_2 may be a transformed version of I_1
- What kinds of transformations are we likely to encounter in practice?

Courtesy Steve Seitz and Richard Szeliski

Invariance

- Suppose we are comparing two images I_1 and I_2
  - I_2 may be a transformed version of I_1
  - What kinds of transformations are we likely to encounter in practice?

- We'd like to find the same features regardless of the transformation
  - This is called transformational ***invariance***
  - Most feature methods are designed to be invariant to
    - Translation, 2D rotation, scale
  - They can usually also handle
    - Limited 3D rotations (SIFT works up to about 60 degrees)
    - Limited affine transformations (some are fully affine invariant)
    - Limited illumination/contrast changes

Courtesy Steve Seitz and Richard Szeliski

How to achieve invariance

Need both of the following:

1. Make sure your detector is invariant
   - Harris is invariant to translation and rotation
   - Scale is trickier
     - common approach is to detect features at many scales using a Gaussian pyramid (e.g., MOPS)
     - More sophisticated methods find “the best scale” to represent each feature (e.g., SIFT)
2. Design an invariant feature *descriptor*
   - A descriptor captures the information in a region around the detected feature point
   - The simplest descriptor: a square window of pixels
     - What’s this invariant to?
   - Let’s look at some better approaches…

Courtesy Steve Seitz and Richard Szeliski

Rotation invariance for feature descriptors

Find dominant orientation of the image patch

- This is given by $\mathbf{x}_+$, the eigenvector of $\mathbf{H}$ corresponding to λ_+
  - λ_+ is the *larger* eigenvalue
- Rotate the patch according to this angle

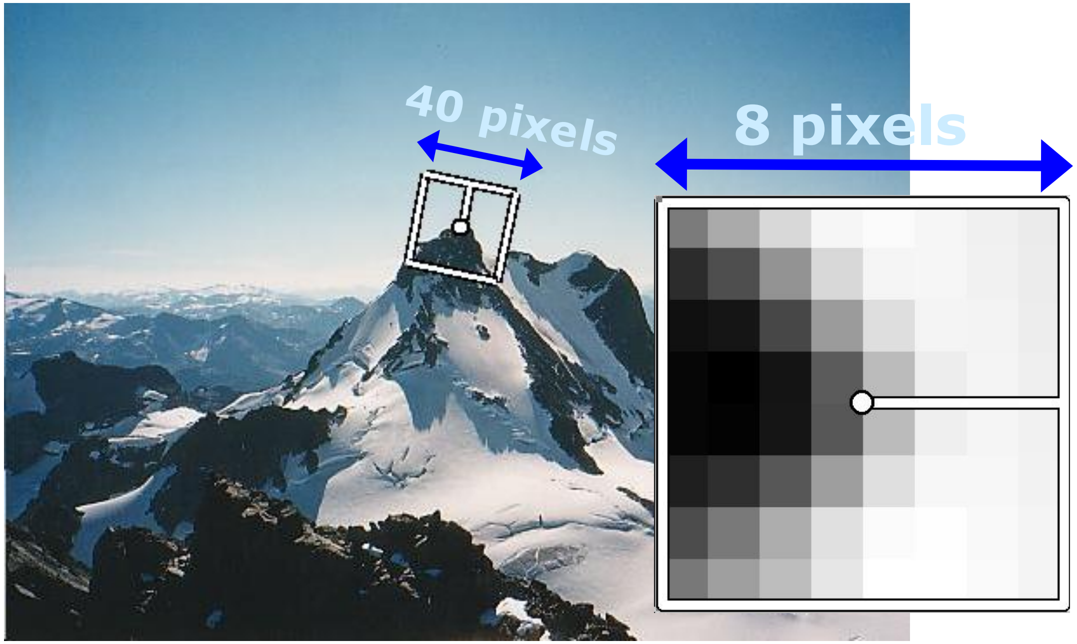
Figure by Matthew Brown

Courtesy Steve Seitz and Richard Szeliski

Multiscale Oriented PatcheS descriptor

Take 40x40 square window around detected feature

- Scale to 1/5 size (using prefiltering)
- Rotate to horizontal
- Sample 8x8 square window centered at feature
- Intensity normalize the window by subtracting the mean, dividing by the standard deviation in the window



Slide adapted from Matthew Brown

Courtesy Steve Seitz and Richard Szeliski

Detections at multiple scales

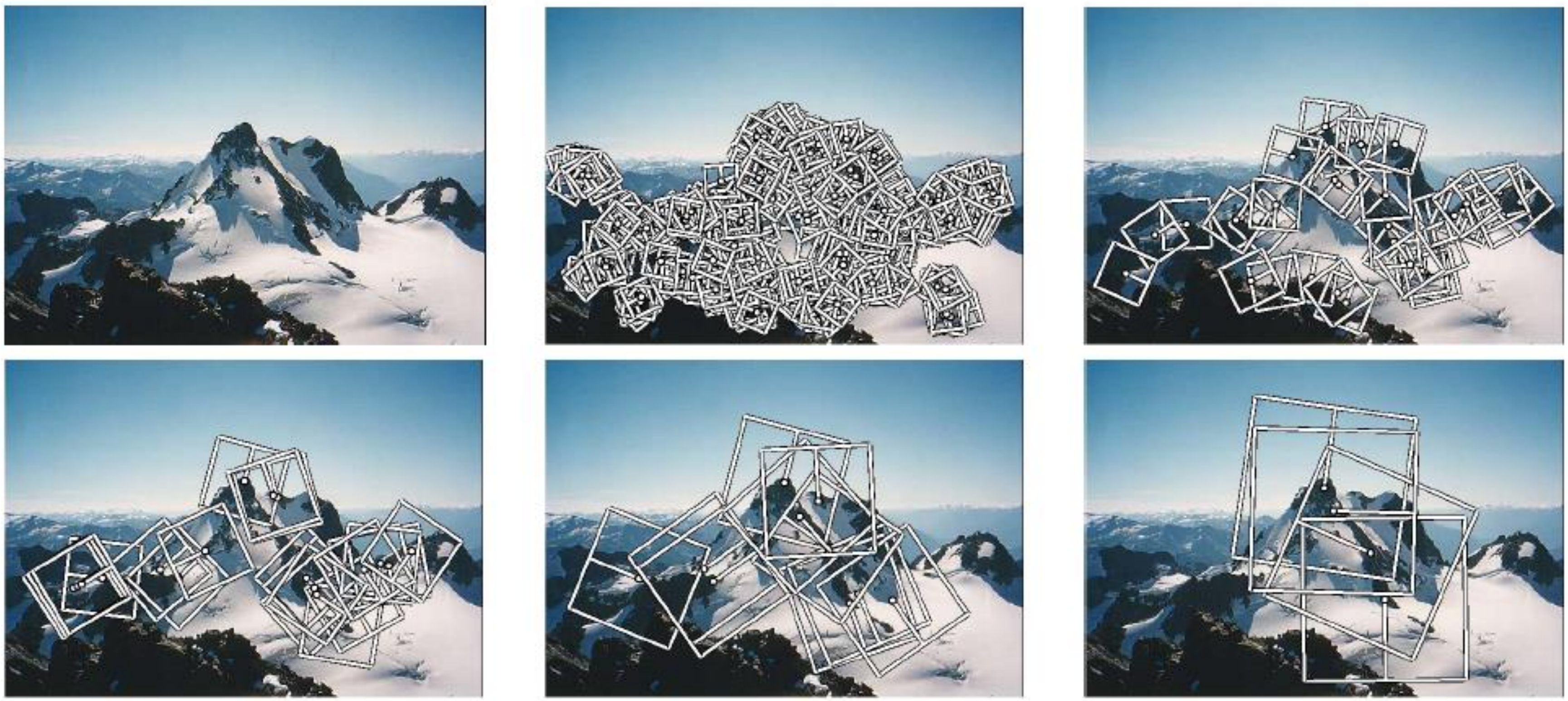


Figure 1. Multi-scale Oriented Patches (MOPS) extracted at five pyramid levels from one of the Matier images. The boxes show the feature orientation and the region from which the descriptor vector is sampled.

Courtesy Steve Seitz and Richard Szeliski

Scale Invariant Feature Transform

- Basic idea:
  - Take 16x16 square window around detected feature
  - Compute edge orientation (angle of the gradient - 90°) for each pixel
  - Throw out weak edges (threshold gradient magnitude)
  - Create histogram of surviving edge orientations

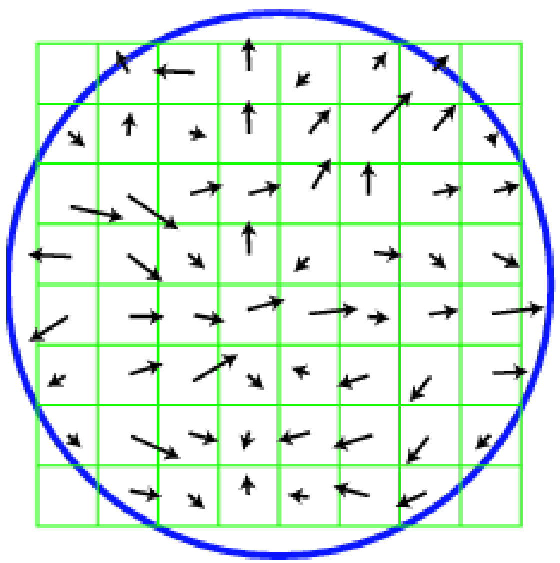
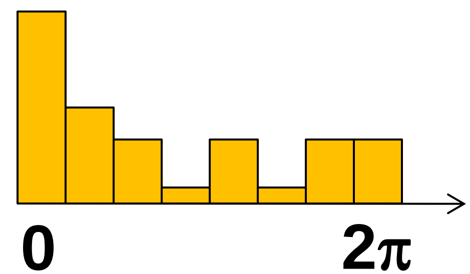
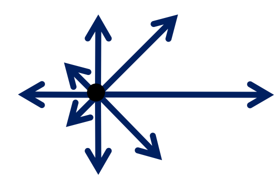


Image gradients



angle histogram

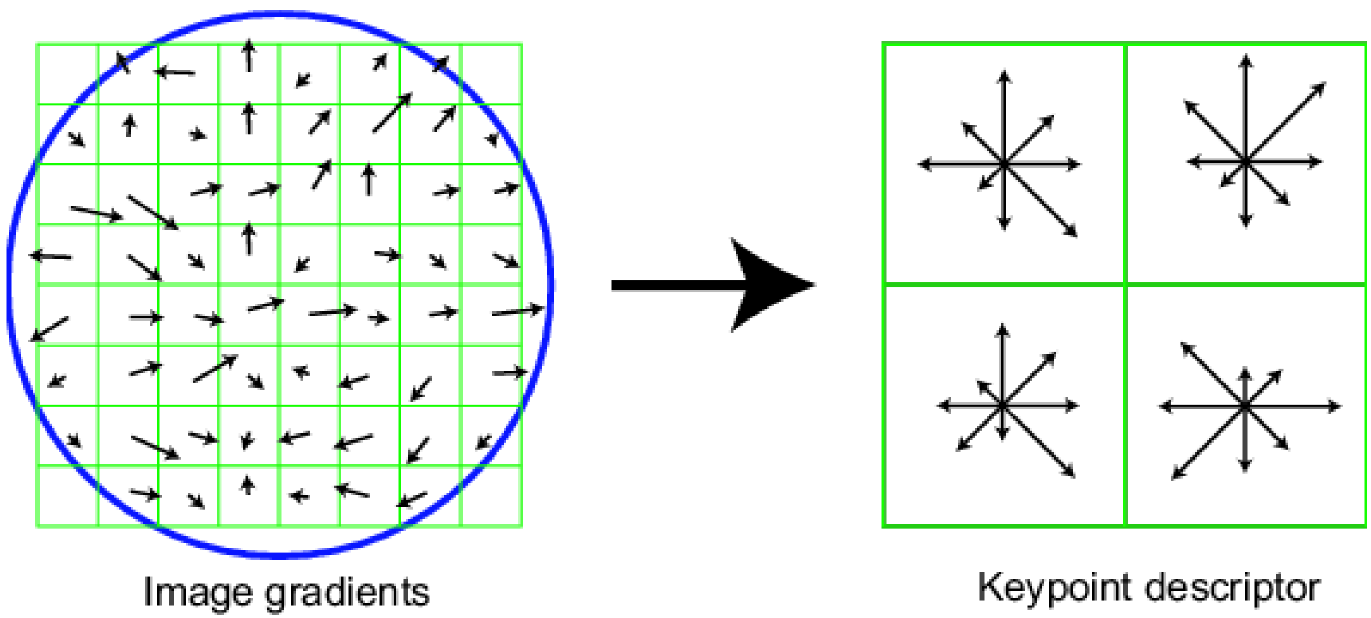


Slide adapted from David Lowe

Courtesy Steve Seitz and Richard Szeliski

SIFT descriptor

- Full version
  - Divide the 16x16 window into a 4x4 grid of cells (2x2 case shown below)
  - Compute an orientation histogram for each cell
  - 16 cells * 8 orientations = 128 dimensional descriptor



Slide adapted from David Lowe

Courtesy Steve Seitz and Richard Szeliski

Properties of SIFT

- Extraordinarily robust matching 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 (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

Courtesy Steve Seitz and Richard Szeliski

Feature matching

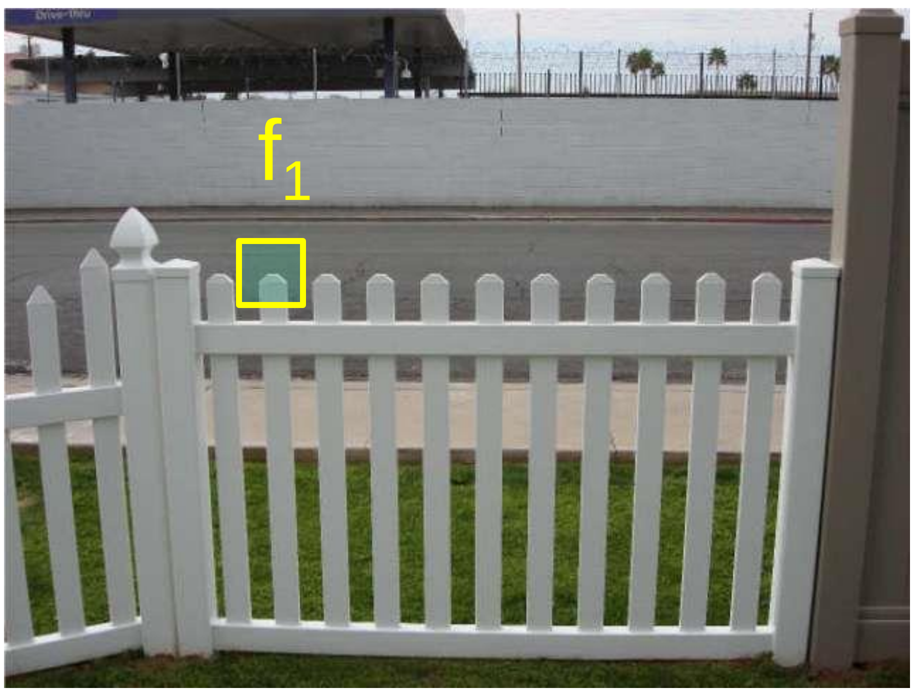
Given a feature in I_1, how to find the best match in I_2?

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

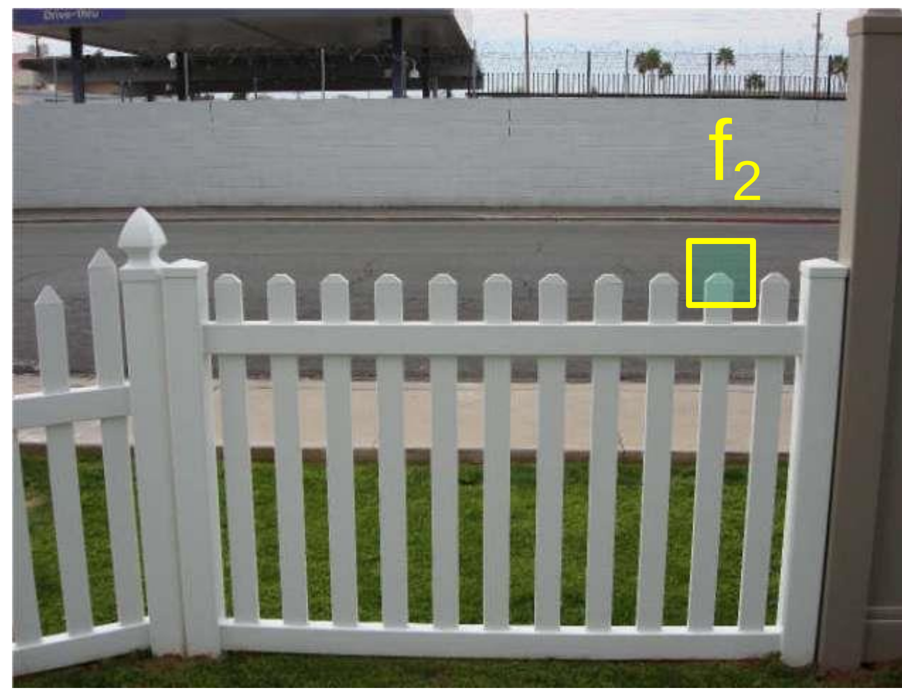
Courtesy Steve Seitz and Richard Szeliski

Feature distance

- How to define the difference between two 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 very ambiguous (bad) matches



I_1

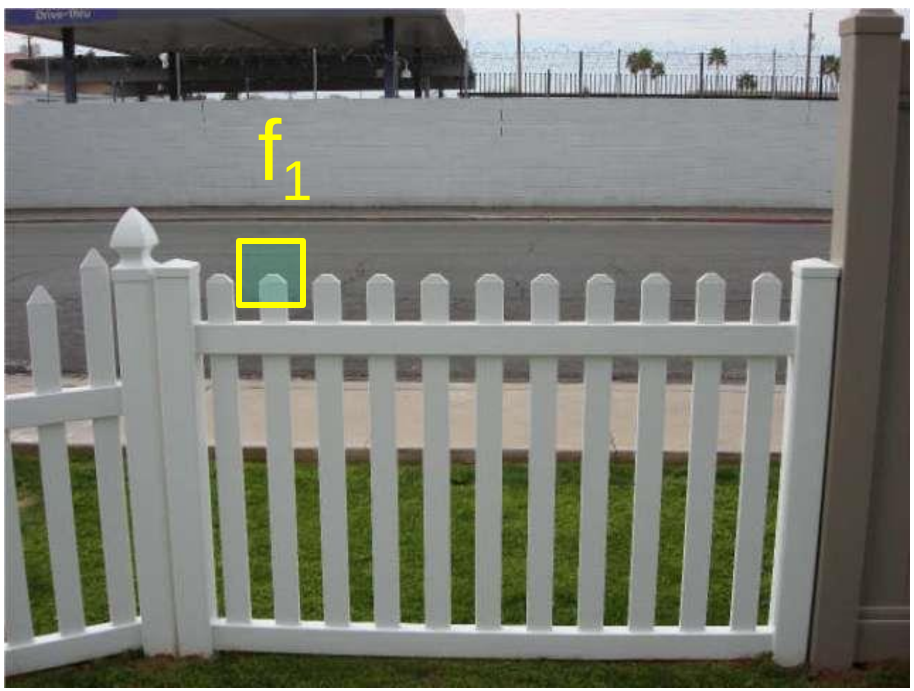


I_2

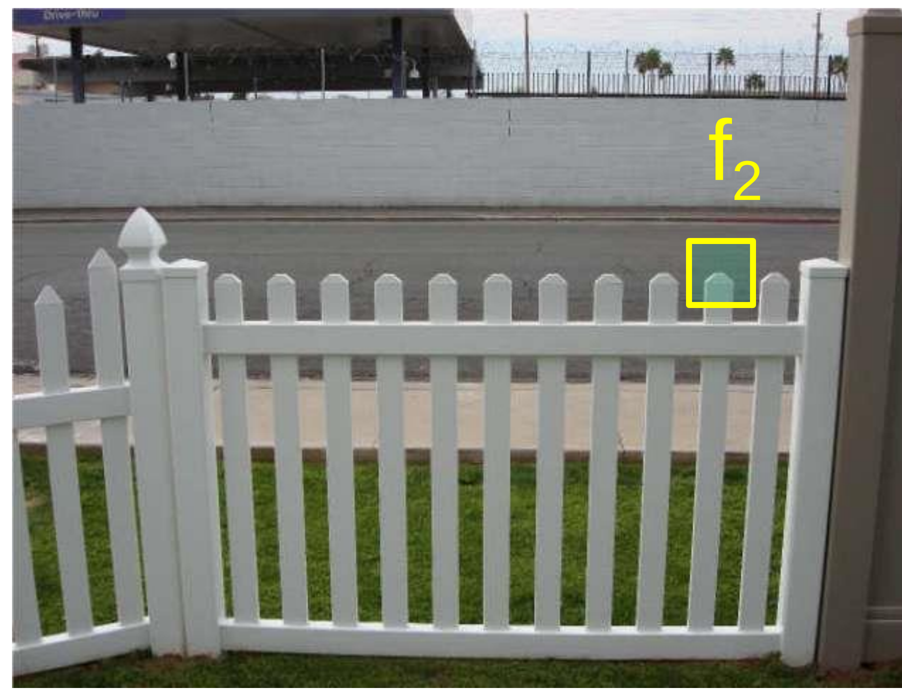
Courtesy Steve Seitz and Richard Szeliski

Feature distance

- How to define the difference between two 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 ambiguous matches



I_1

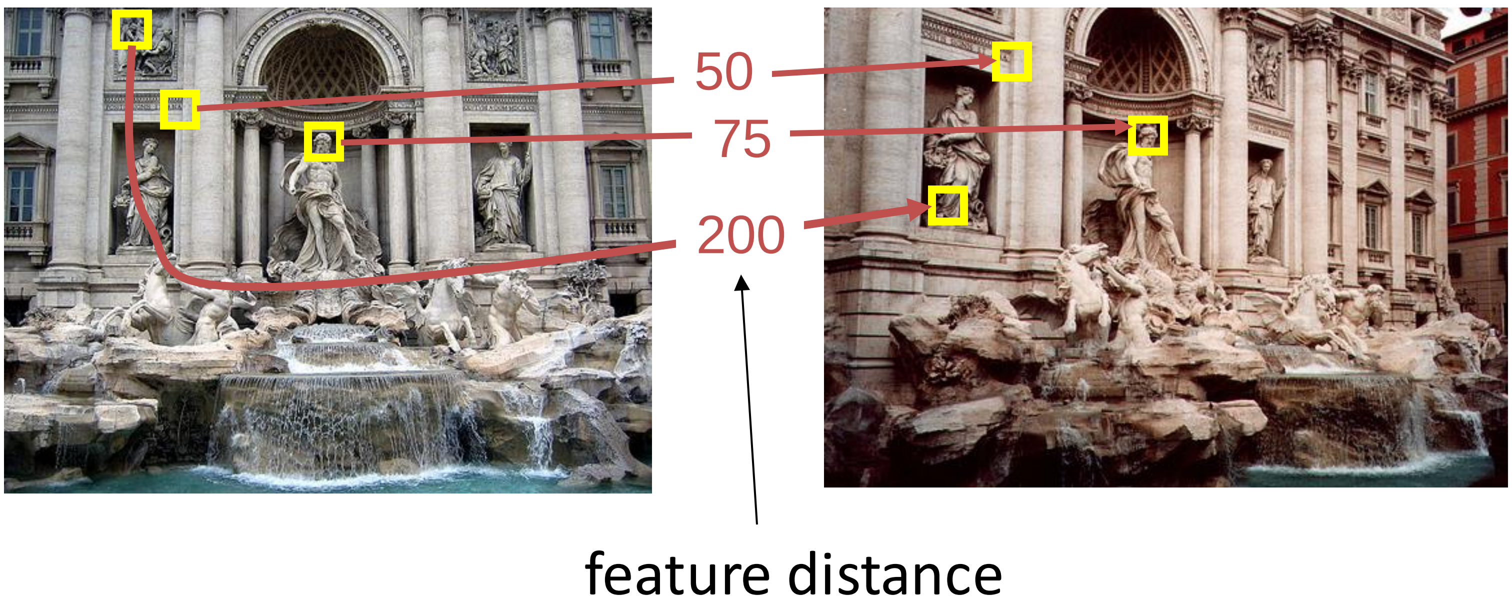


I_2

Courtesy Steve Seitz and Richard Szeliski

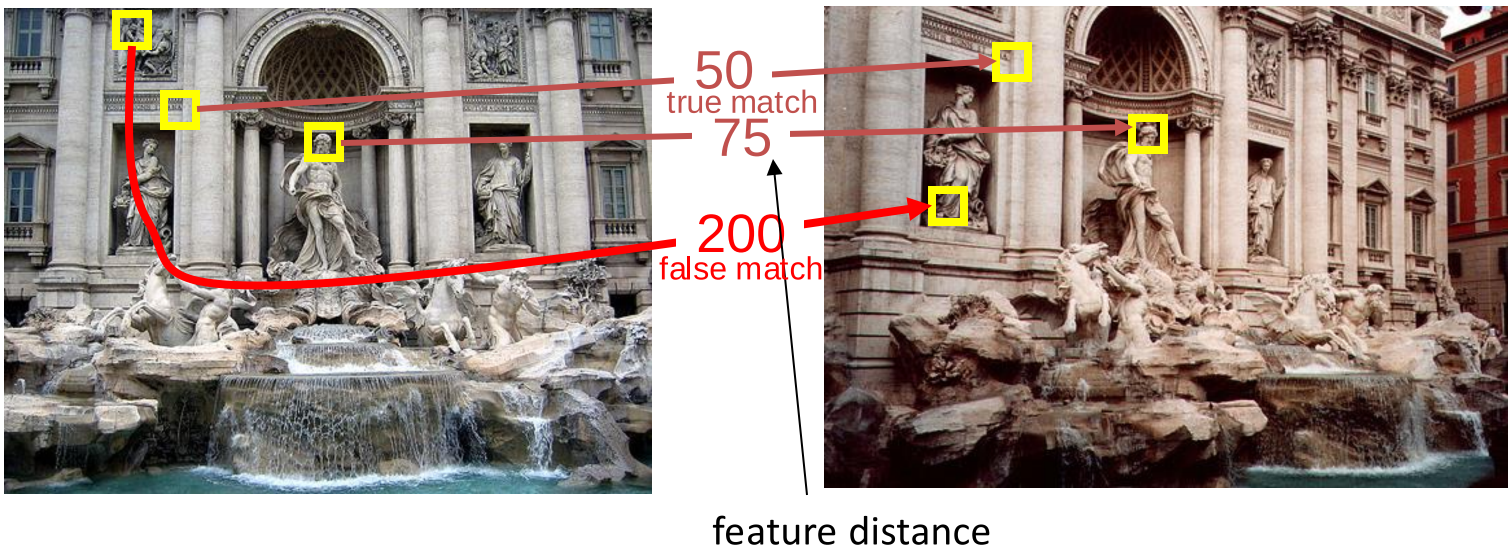
Evaluating the results

How can we measure the performance of a feature matcher?



Courtesy Steve Seitz and Richard Szeliski

True/false positives



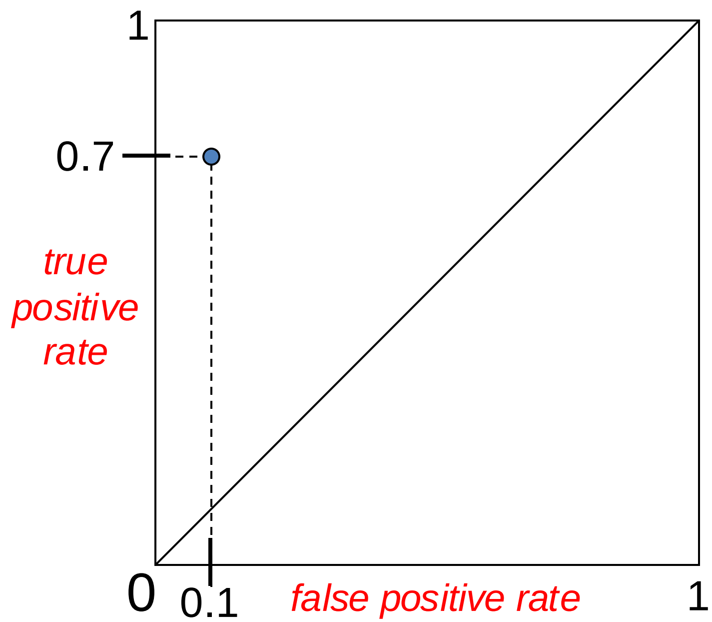
- The distance threshold affects performance
  – True positives = # of detected matches that are correct
    - Suppose we want to maximize these—how to choose threshold?
  – False positives = # of detected matches that are incorrect
    - Suppose we want to minimize these—how to choose threshold?

Courtesy Steve Seitz and Richard Szeliski

Evaluating the results

How can we measure the performance of a feature matcher?

$$\frac{\text{\# true positives}}{\text{\# matching features (positives)}}$$



$$\frac{\text{\# false positives}}{\text{\# unmatched features (negatives)}}$$

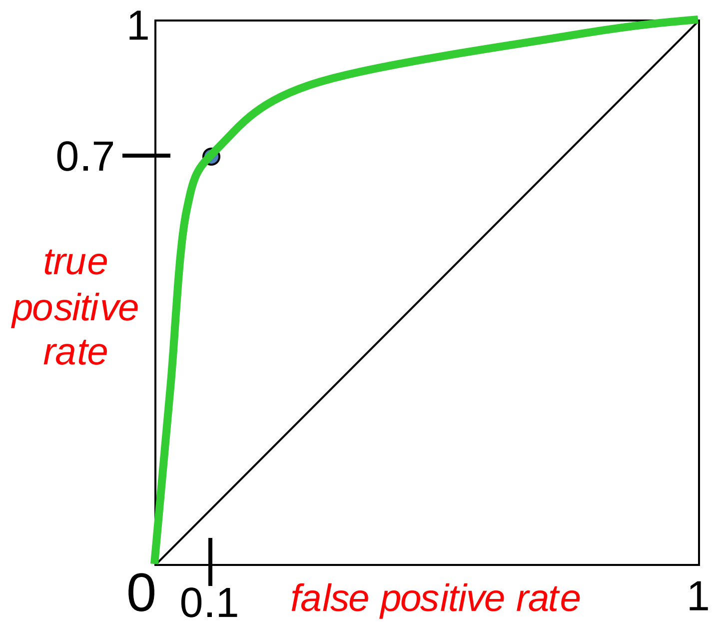
Courtesy Steve Seitz and Richard Szeliski

Evaluating the results

- How can we measure the performance of a feature matcher?

ROC curve (“Receiver Operator Characteristic”)

$$\text{true positive rate} = \frac{\text{\# true positives}}{\text{\# matching features (positives)}}$$



$$\text{false positive rate} = \frac{\text{\# false positives}}{\text{\# unmatched features (negatives)}}$$

- ROC Curves
  - Generated by counting # current/incorrect matches, for different thresholds
  - Want to maximize area under the curve (AUC)
  - Useful for comparing different feature matching methods

Courtesy Steve Seitz and Richard Szeliski

Lots of applications

- Features are used for:
  - Image alignment (e.g., mosaics)
  - 3D reconstruction
  - Motion tracking
  - Object recognition
  - Indexing and database retrieval
  - Robot navigation
  - … other

Courtesy Steve Seitz and Richard Szeliski

Logistics

- Project 1, phase 2 due this Friday (03/07)
- Project 1, phase 3 due next Friday (03/14)
- Lab tutorial next Wednesday (3/12)
  - 6:00pm-8:50pm @ Robotics Institute G/F, Flight Area
  - 2~3 students per group.
  - Try your best to find your groupmates, or we will randomly assign for you.
- Midterm next Tuesday (03/11)
  - 1:30pm-3:30pm @ Rm5506
  - Open book, open notes, close Internet
  - No communication with your classmates, honor code is strictly enforced
  - Covers lectures 1-4
  - 2 hours exam