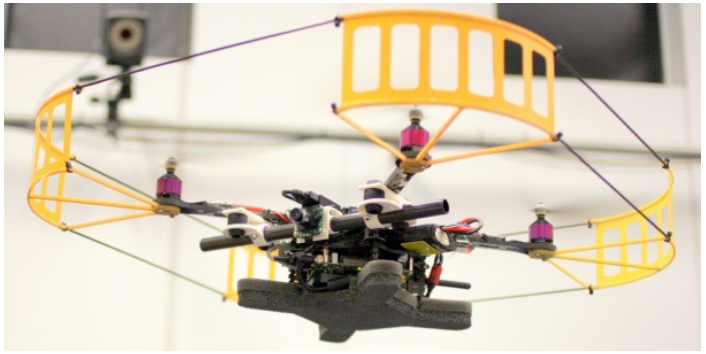
香港科技大學
THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY
電子及計算機工程學系
DEPARTMENT OF ELECTRONIC & COMPUTER ENGINEERING

# Introduction to Aerial Robotics
## Lecture 5

Shaojie Shen
Assistant Professor
Dept. of ECE, HKUST



6 October 2015

# Outline

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

# Camera Modeling

# Goal: Fly Like Birds


Video from Smithsonian Channel

# How can gannets' eyes and brain estimate distance to water?

*Nature Vol. 293 24 September 1981*

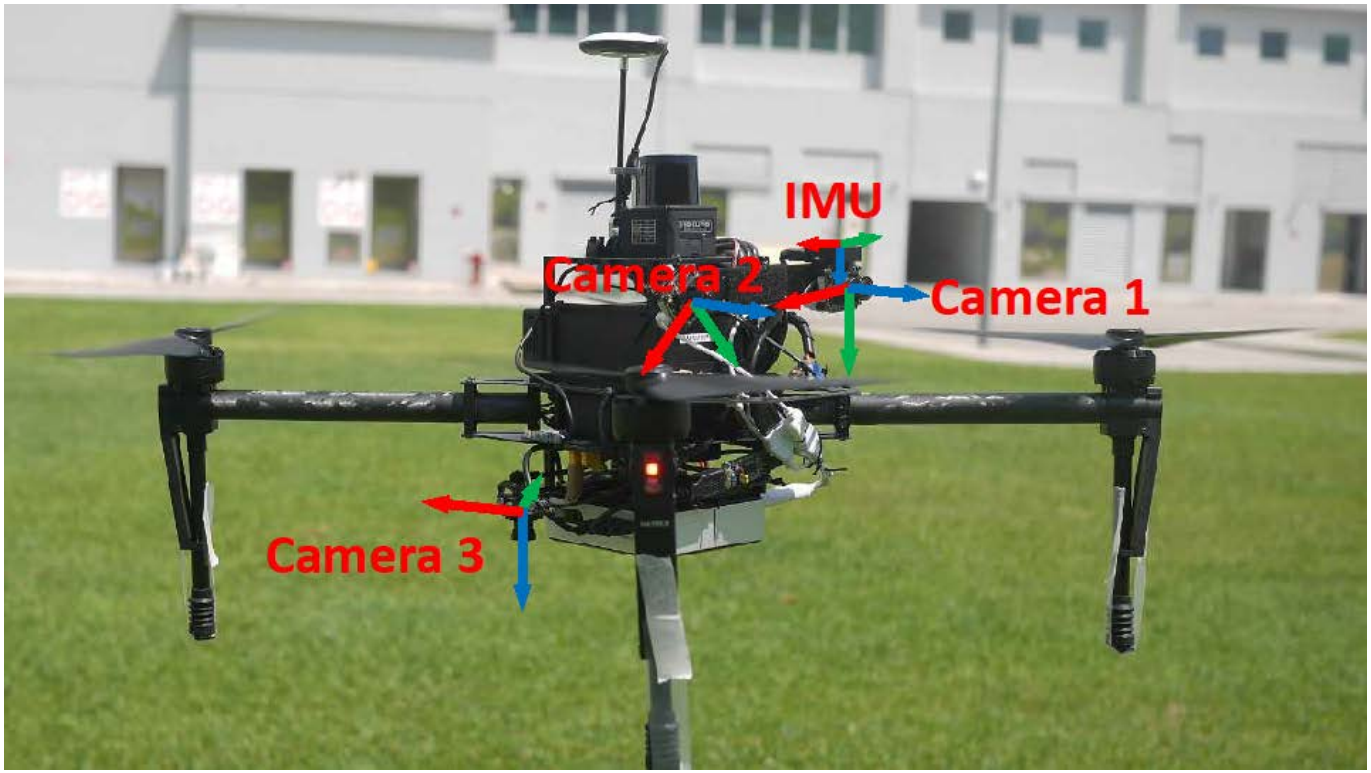
5. Spiess, F. N. *et al. Science* **207,** 1421–1433 (1980).
6. Corliss, J. B. *Science* **203,** 1073–1083 (1979).
7. Edmond, J. M. *et al. Earth planet. Sci. Lett.* **46,** 19–30 (1979).
8. Ruby, E. G., Wirsen, C. O. & Jannasch, H. W. *Appl. envir. Microbiol.* (in the press).
9. Cavanaugh, C. M., Gardiner, S. L., Jones, M. L., Jannasch, H. W. & Waterbury, J. B. *Science* **213,** 340–342 (1981).
10. Felbeck, H. *Science* **213,** 336–338 (1981).
11. Jones, M. L. *Science* **213,** 333–336 (1981).
12. Peck, H. D. Jr *Enzymes* **10,** 651–669 (1974).
13. Latzko, E. & Gibbs, M. *Pl. Physiol.* **44,** 295–300 (1969).
14. Reid, R. G. B. *Can. J. Zool.* **58,** 386–393 (1980).
15. Los Angeles County (California) Sanitation District files.
16. Rau, G. H. *Science* **213,** 338–340 (1981).
17. Rau, G. H. *Nature* **289,** 484–485 (1981).
18. Rau, G. H. & Hedges, J. I. *Science* **203,** 648–649 (1979).
19. Emery, K. O. & Hulsemann, J. *Deep-Sea Res.* **8,** 165–180 (1962).
20. Hartman, O. & Barnard, J. L. *Allan Hancock Pacific Exped.* **22,** (1958).
21. Nicholas, D. J. D., Ferrante, J. V. & Clarke, G. R. *Analyt. Biochem.* **95,** 24–31 (1979).
22. Lonsdale, P. *Nature* **281,** 531–534 (1979).

## Plummeting gannets: a paradigm of ecological optics

**David N. Lee & Paul E. Reddish**

Department of Psychology, University of Edinburgh, Edinburgh EH8 9JZ, UK

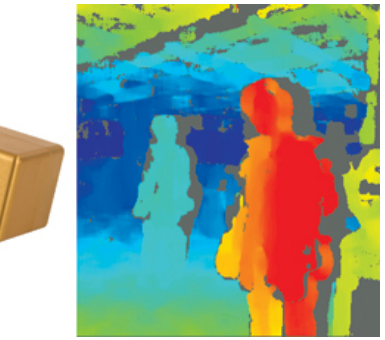
# Robot Sees with Cameras



# Cameras

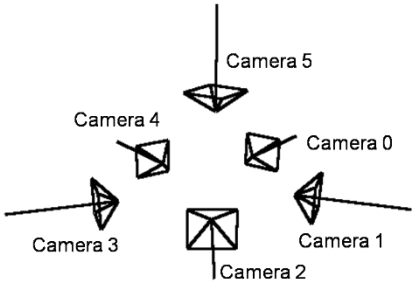
- Monocular
  - Simplest setup
  - Depth unknown

- Stereo
  - Able to compute depth
  - Depth accuracy affect 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

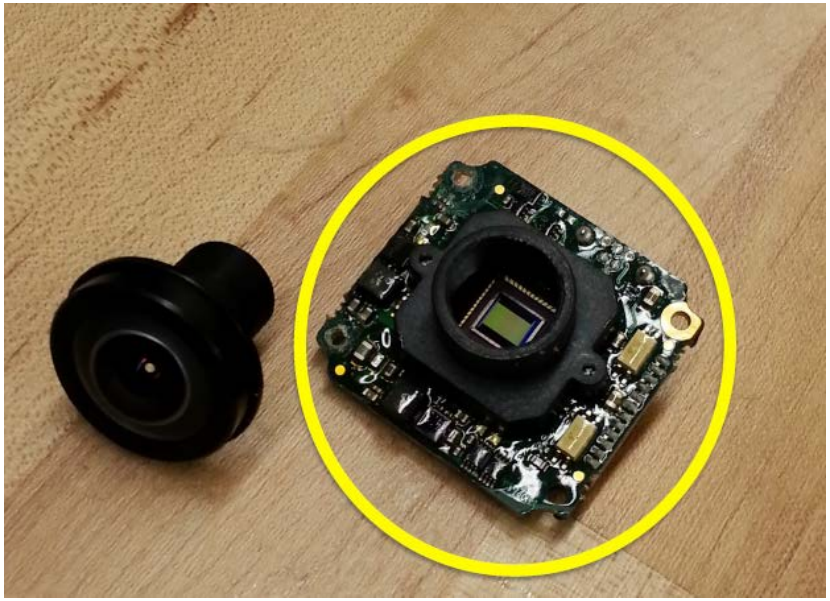
# Cameras

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

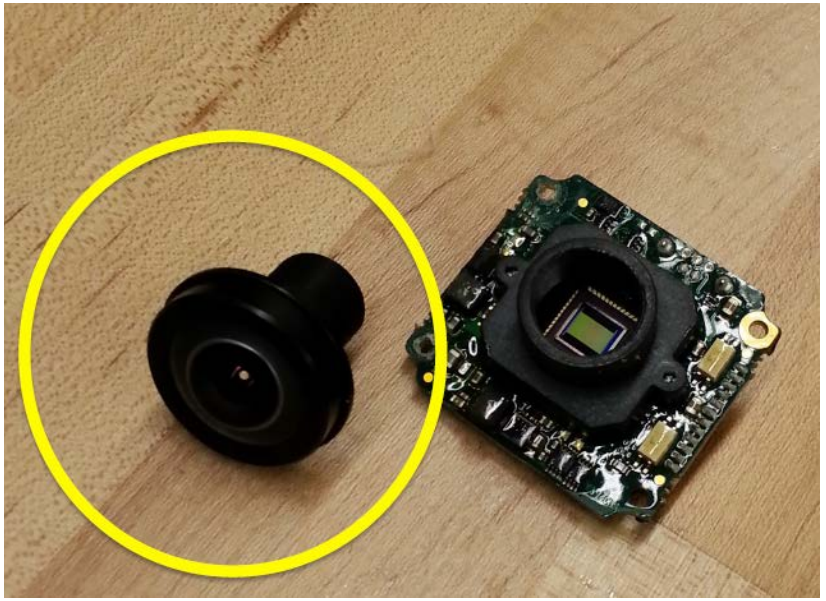

- Omnidirectional camera
  - 360 capture
  - Strong distortion


# Camera

- Sensor

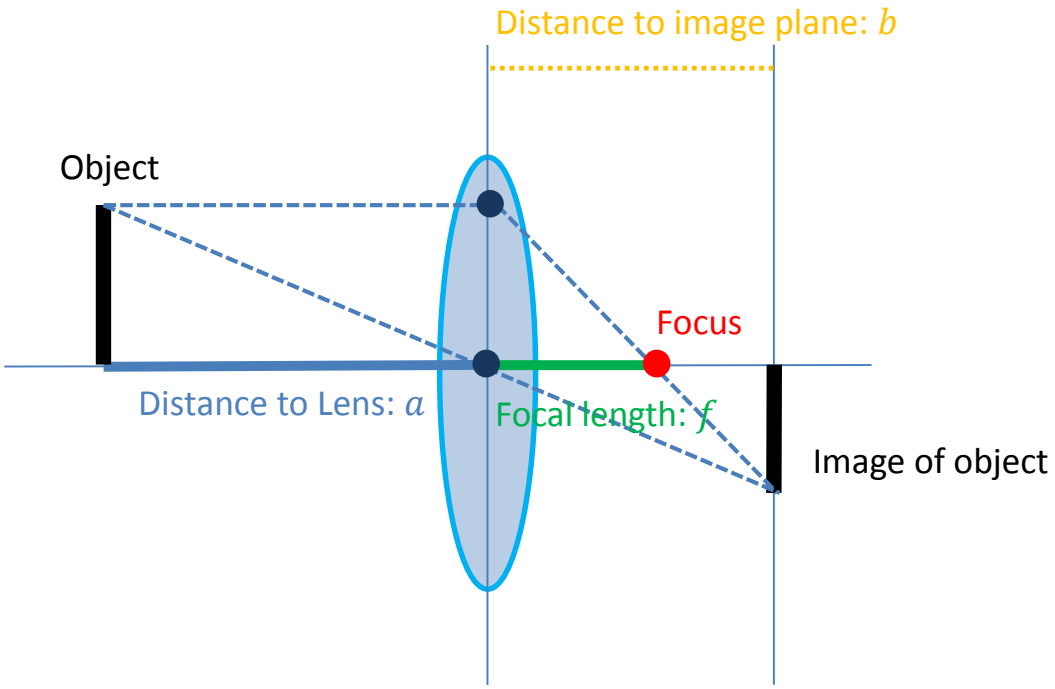


- Lens



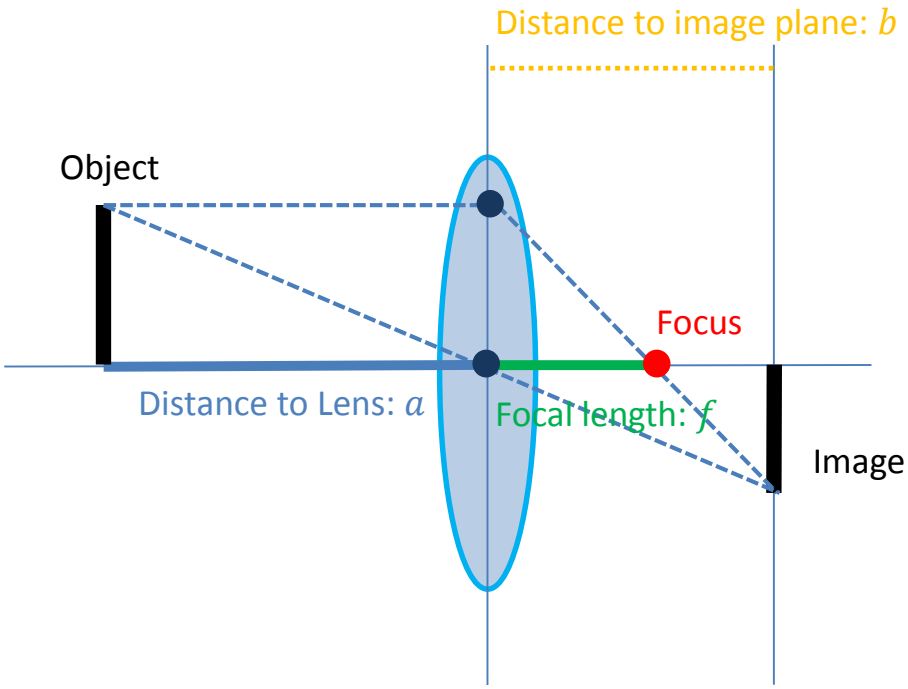
# Thin Lens

- 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

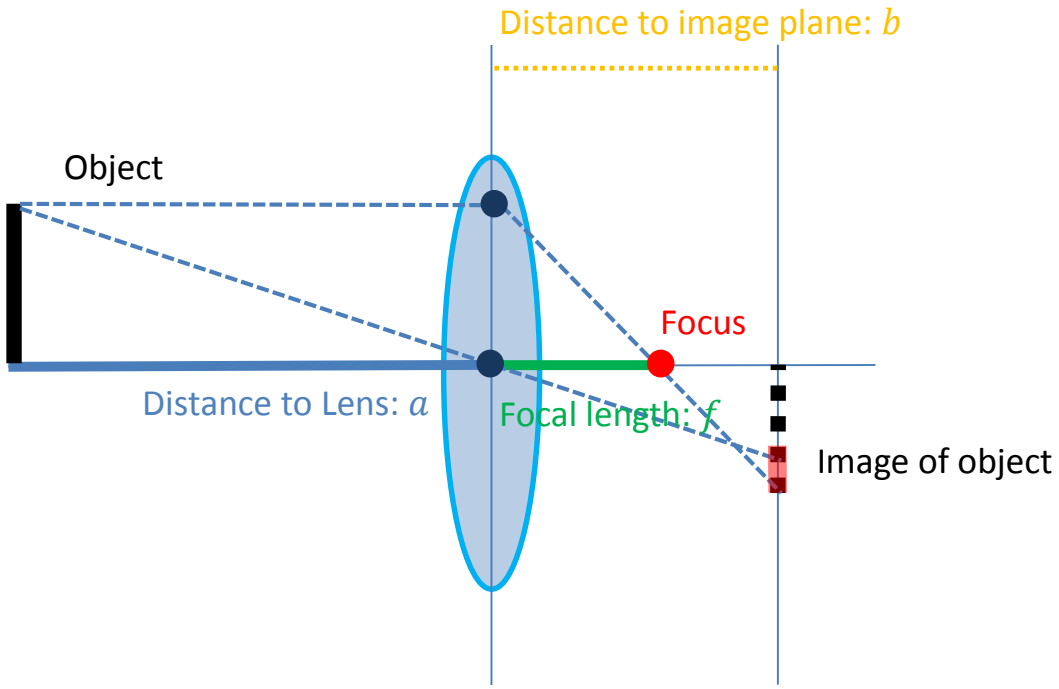



$$\frac{1}{f} = \frac{1}{a} + \frac{1}{b}$$

**WHY?**

# Thin Lens

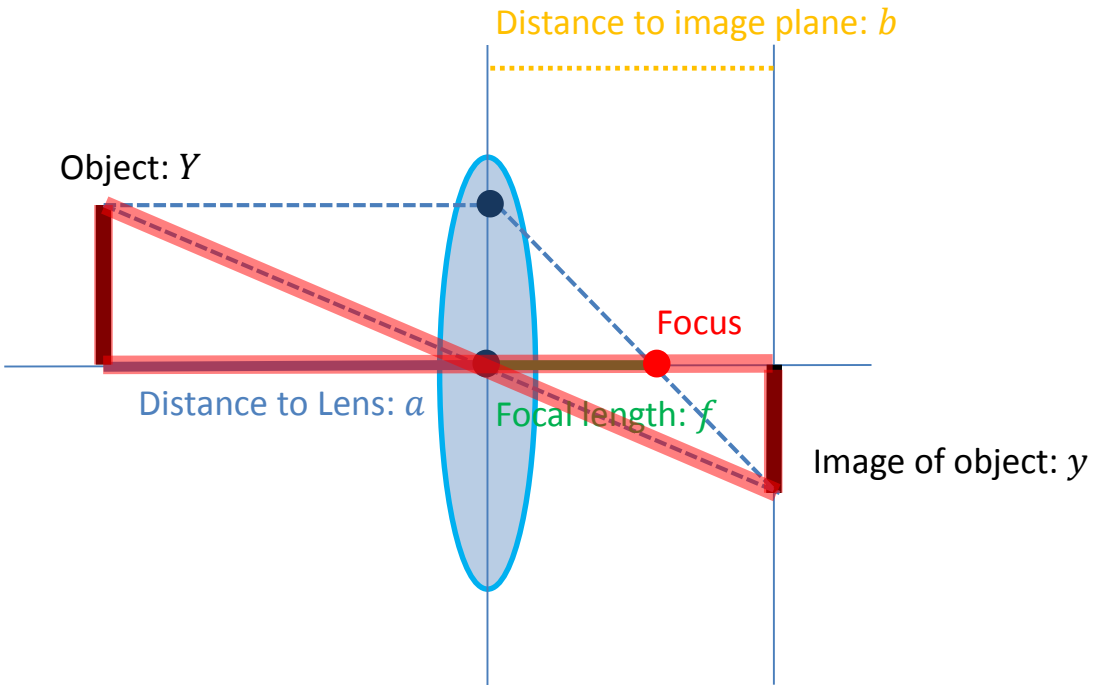
- De-focus if $\frac{1}{f} \neq \frac{1}{a} + \frac{1}{b}$



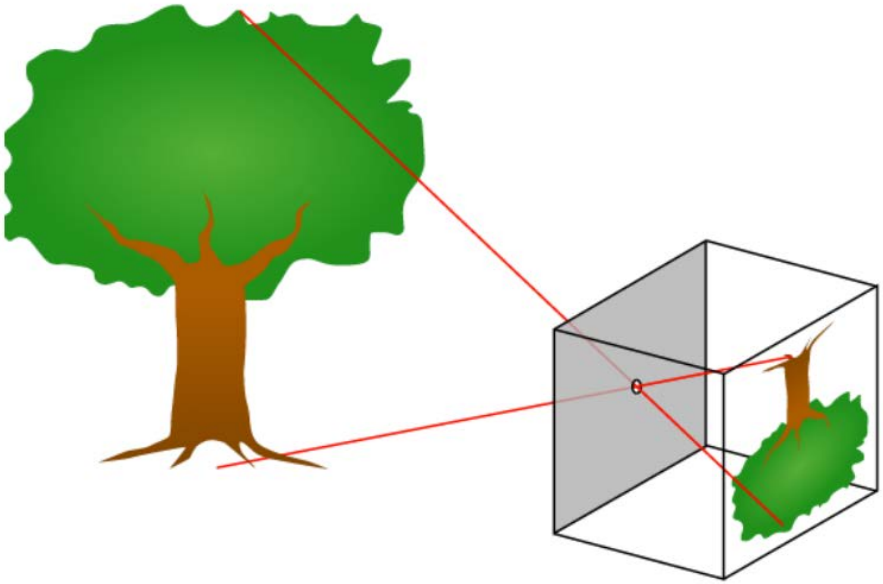
# Perspective Projection

- 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:

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



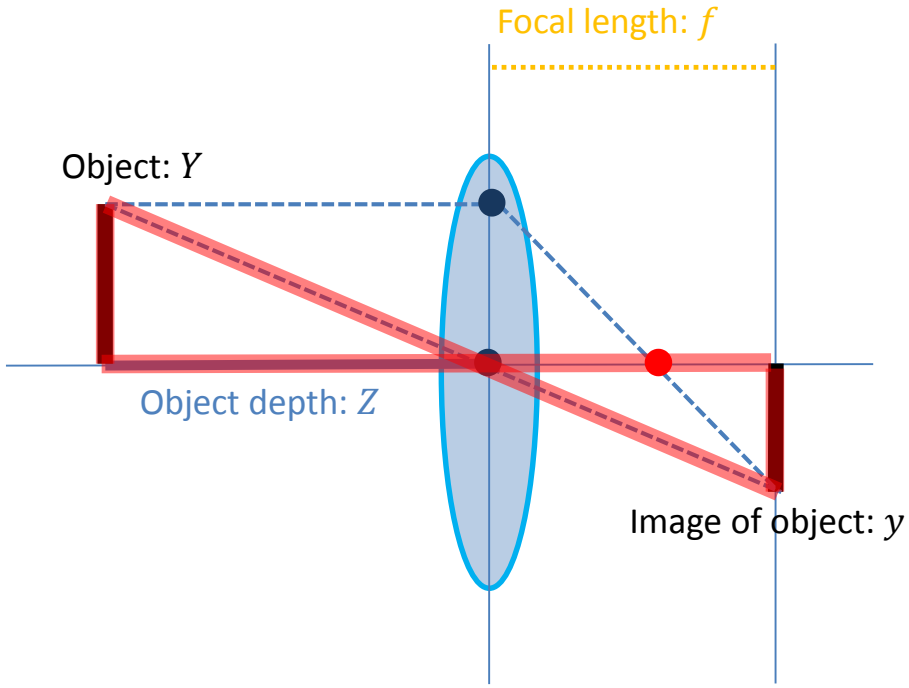
# Pin-hole Camera Model




WIKIPEDIA
The Free Encyclopedia

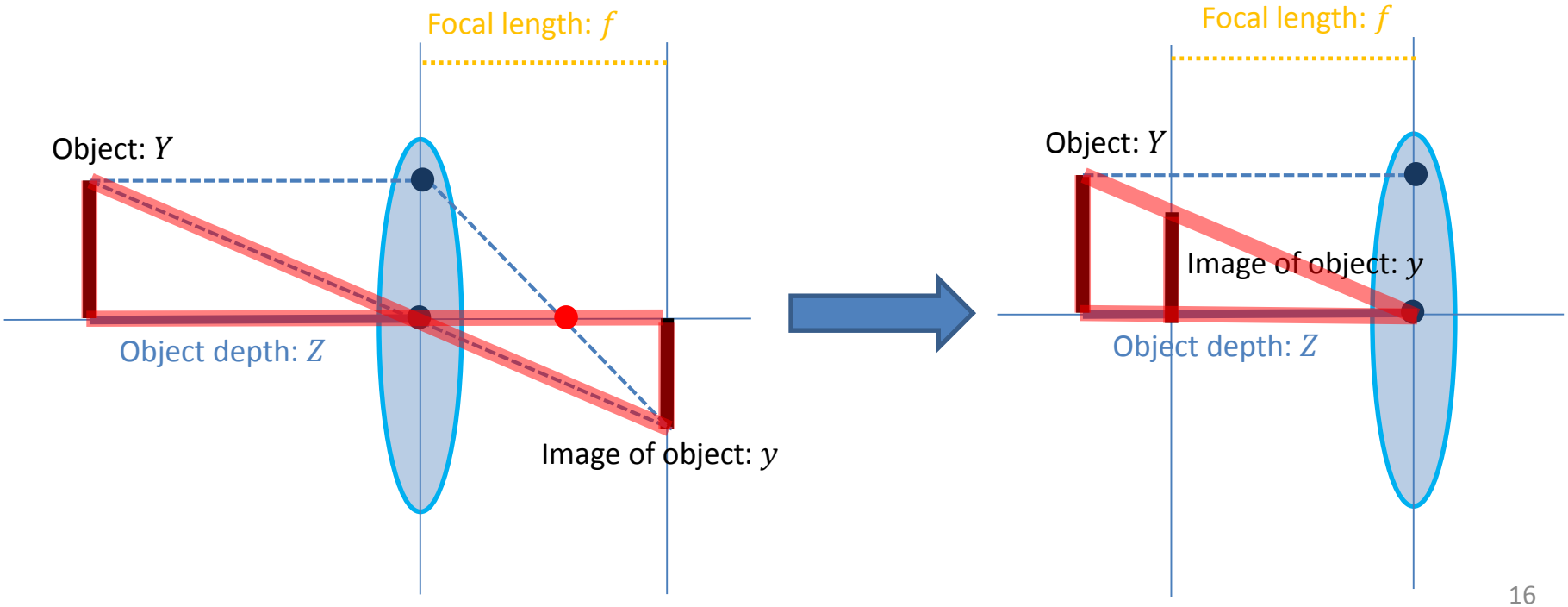
# Pin-hole Camera Model

- 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

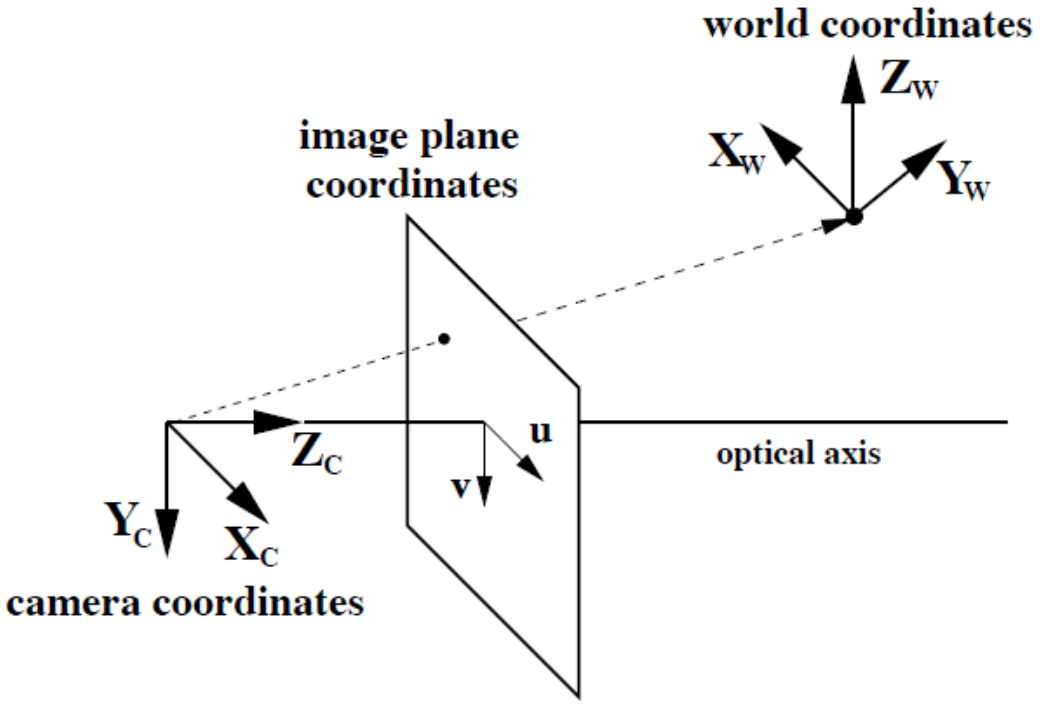
- Assume that the image plane is in front of the lens:
  - $y = f\frac{Y}{Z}$



# 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

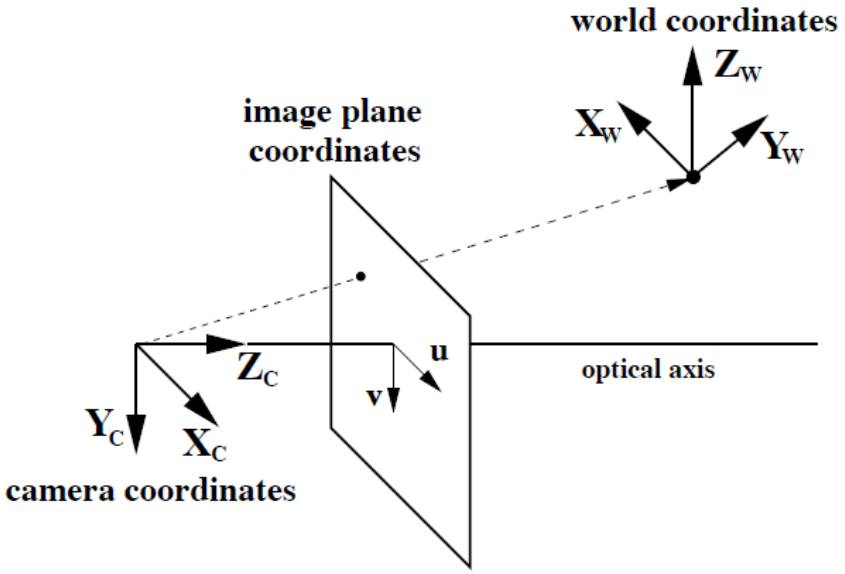





Courtesy Kostas Daniilidis

# Perspective Projection

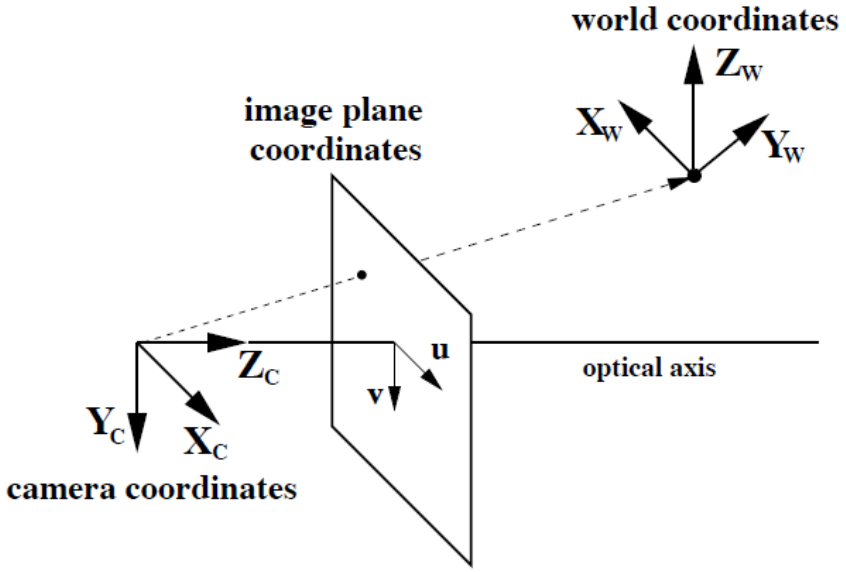
- Optical axis is the z-axis
- The image plane $(u, v)$ is perpendicular to the optical axis
- Intersection of the image plane and the optical axis is the image center $(u_0, v_0)$
- $f$ is the distance of the image plane from the origin (in pixels)
- Formulation:
  - $u = \frac{fX_c}{Z_c} + u_0$
  -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



# Perspective Projection

- Matrix form:

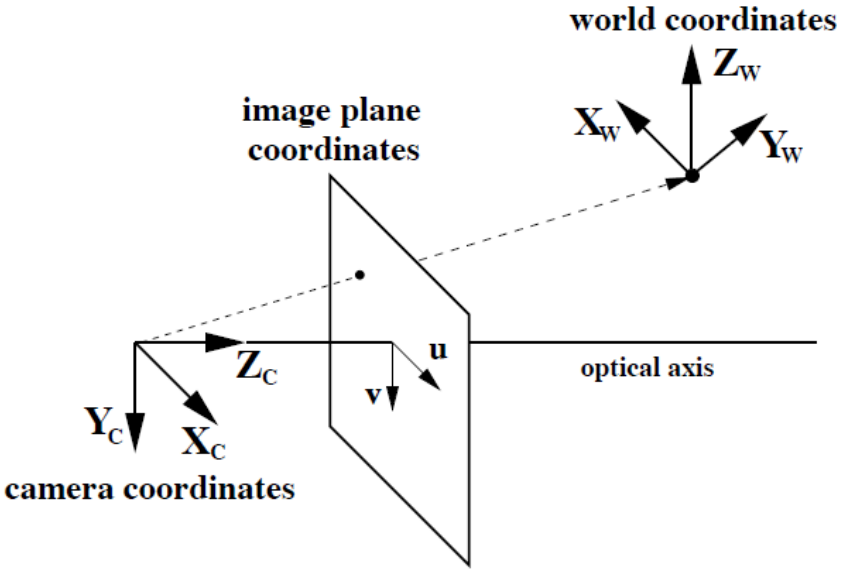
$$\lambda \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} f & 0 & u_o \\ 0 & f & v_o \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{pmatrix}$$



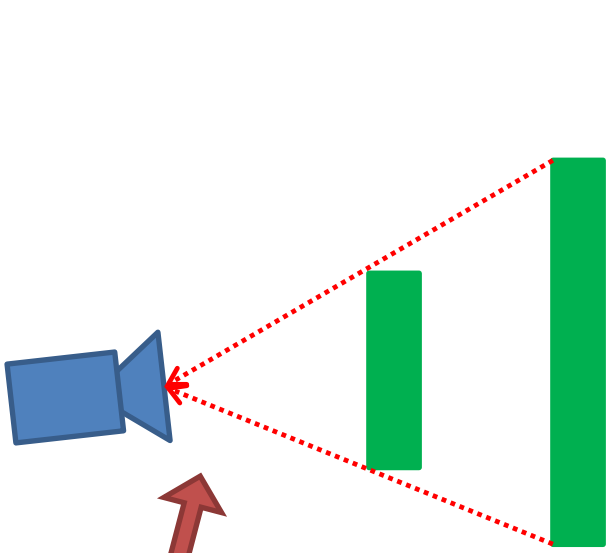
# Perspective Projection

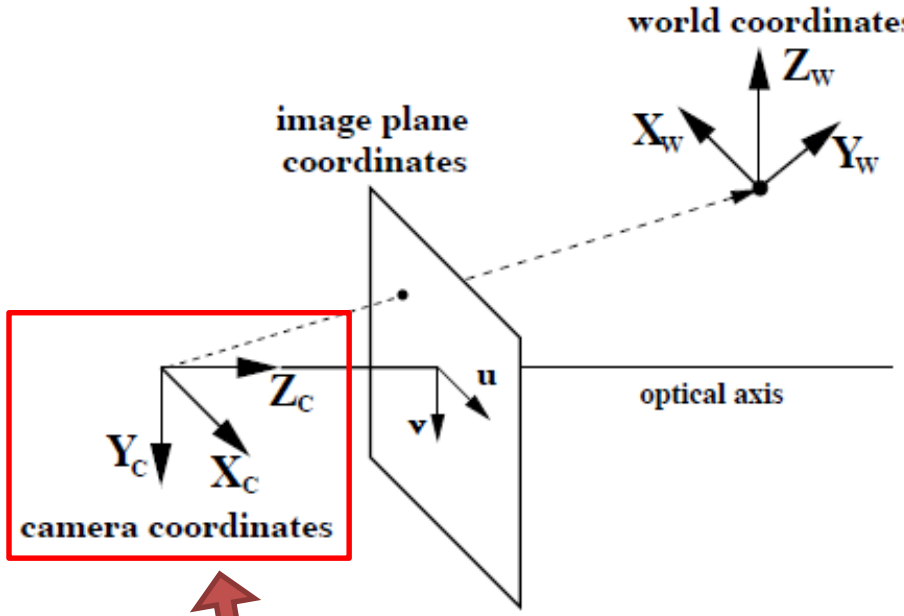
- From camera to world:

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



# Pin-hole Camera Model






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

Unknown Depth

Pixel Values

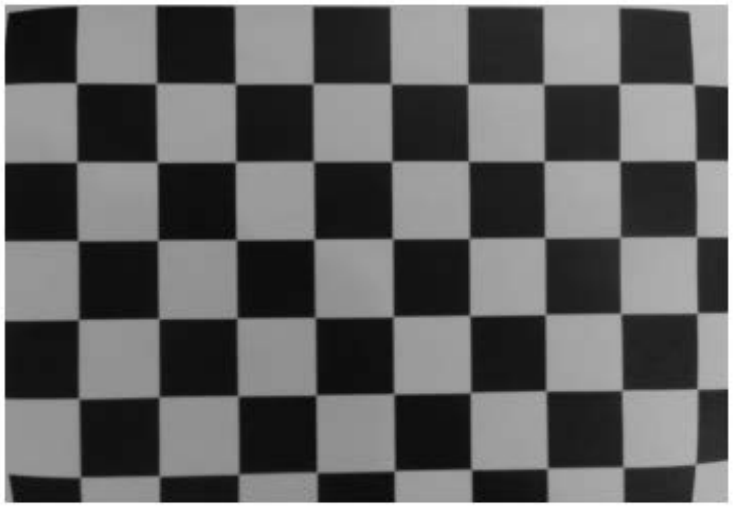
Intrinsic Calibration Matrix

Camera Pose

World Point

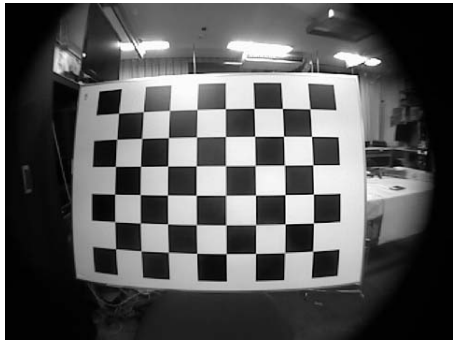
# Radial Distortions

- 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



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

Pixel Values

Intrinsic Calibration Matrix

Camera Pose

World Point

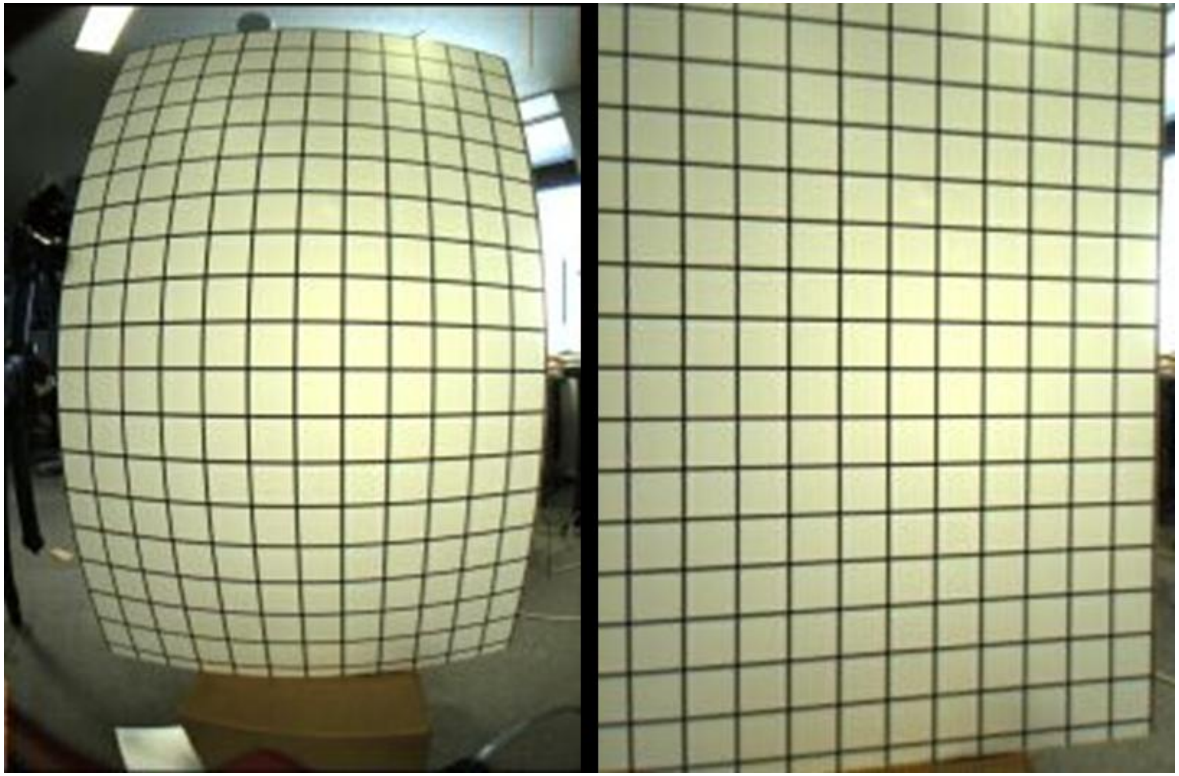
Known

Measurement

To be Estimated

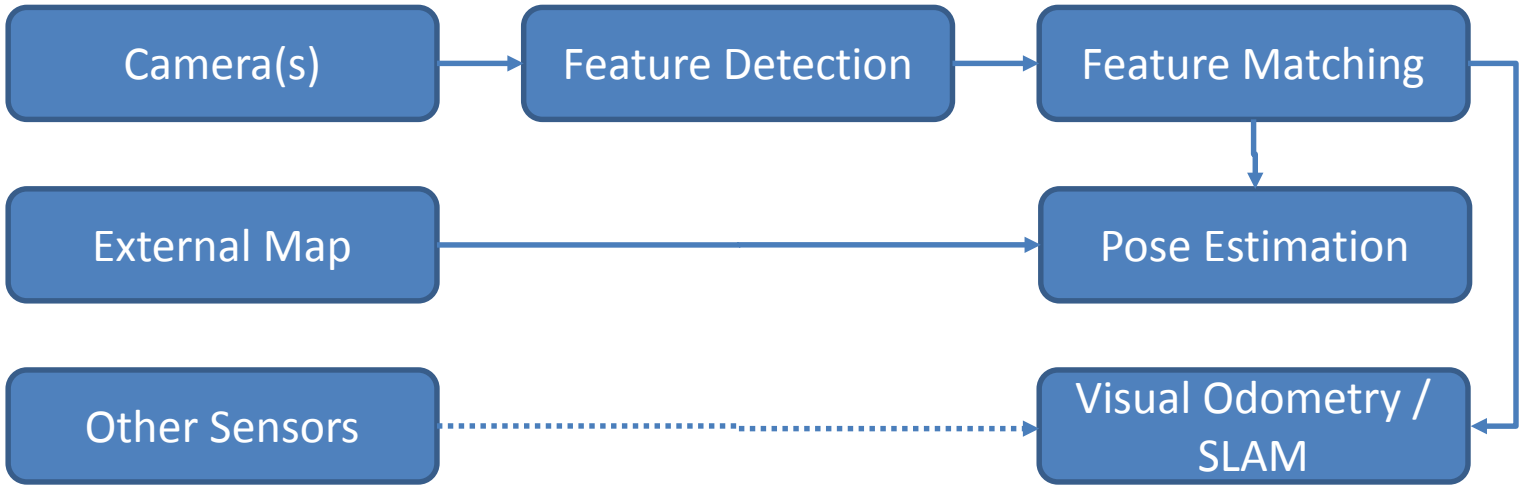
# Camera Calibration

- Straight lines should be straight



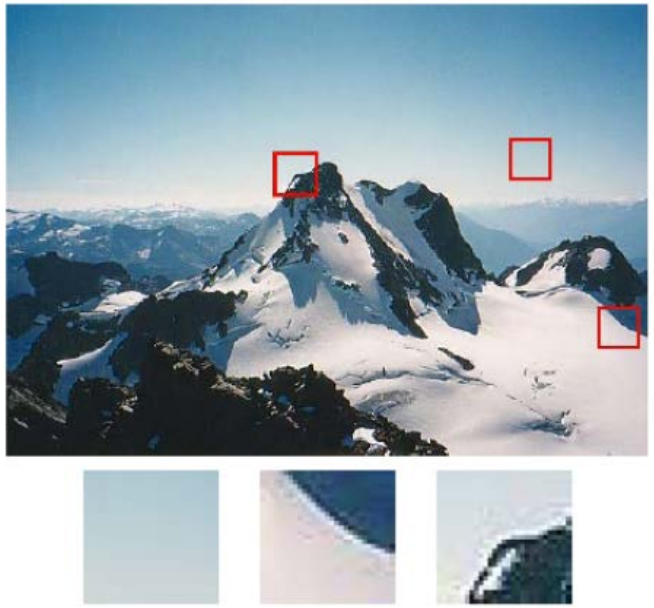
# Robot Vision Pipeline

# Vision-based State Estimation Pipeline



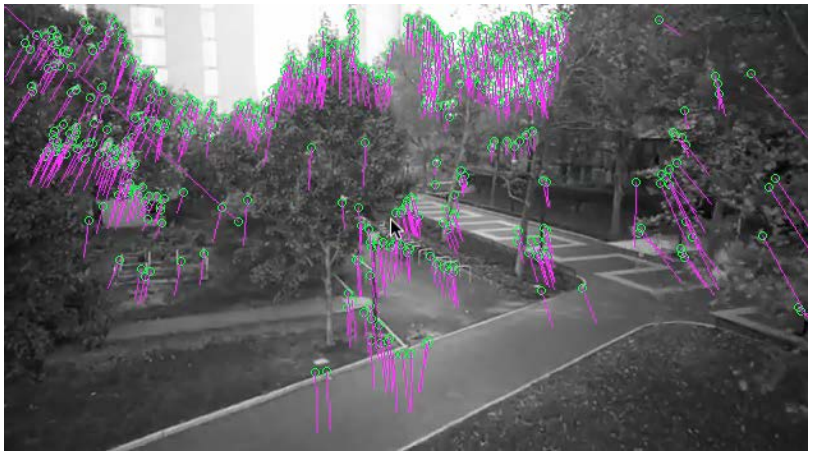
# Feature Detection

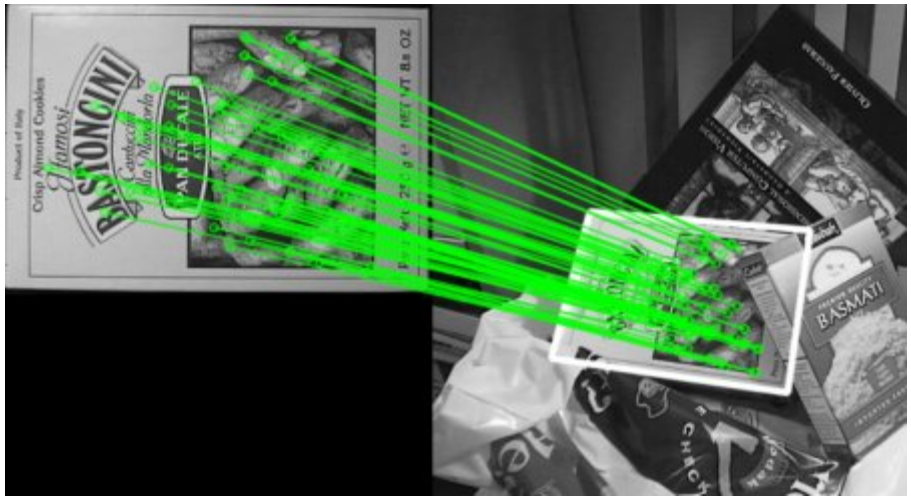
- We cannot process the entire image directly
- Requirements:
  - Repeatability
  - Saliency
  - Locality
  - Compactness and efficiency
- Popular features
  - Corners (FAST, Harris, …)
  - Blob (SIFT, SURF, …)
  - Line (Canny, …)



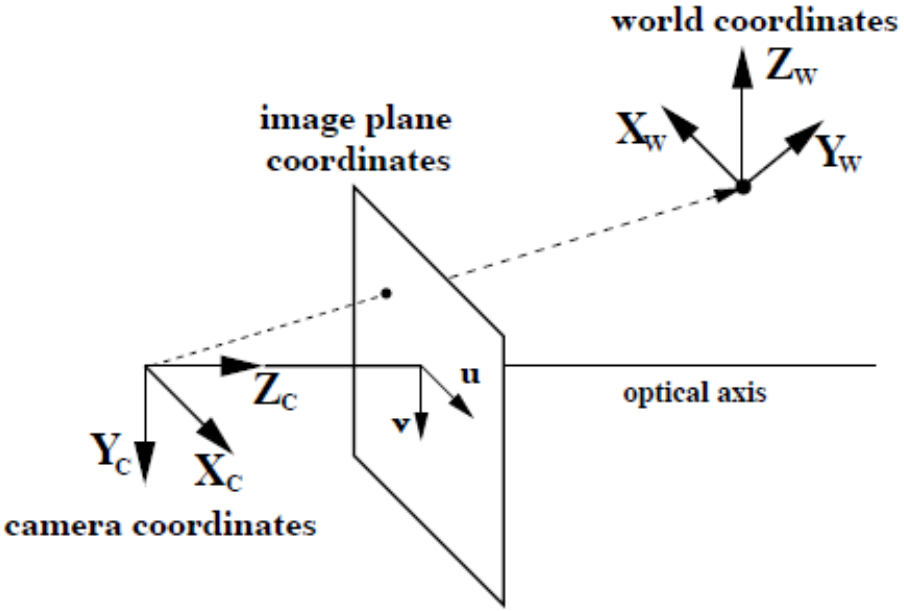
# Feature Matching

- Match features in different images
  - Across multiple cameras
  - Across time
- Common methods:
  - Descriptor matching
  - Optical flow





# Pose Estimation




Known

Measurement

To be Estimated

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

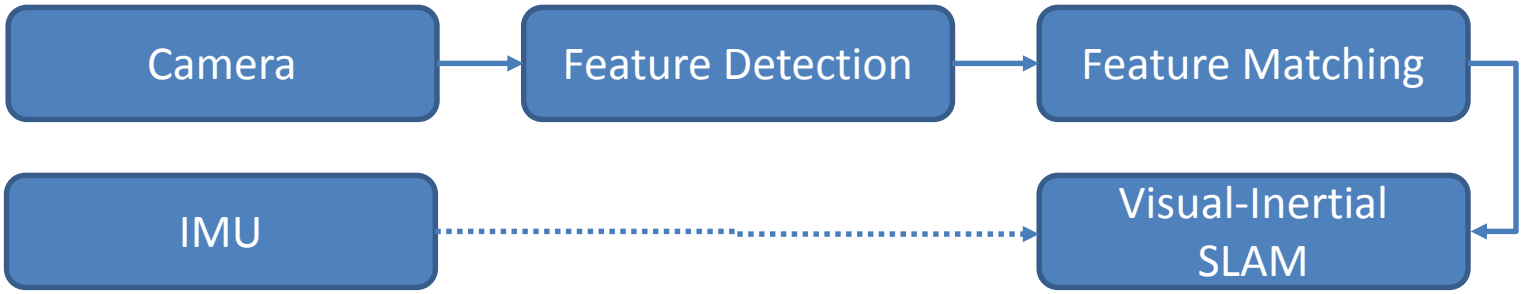
Pixel Values

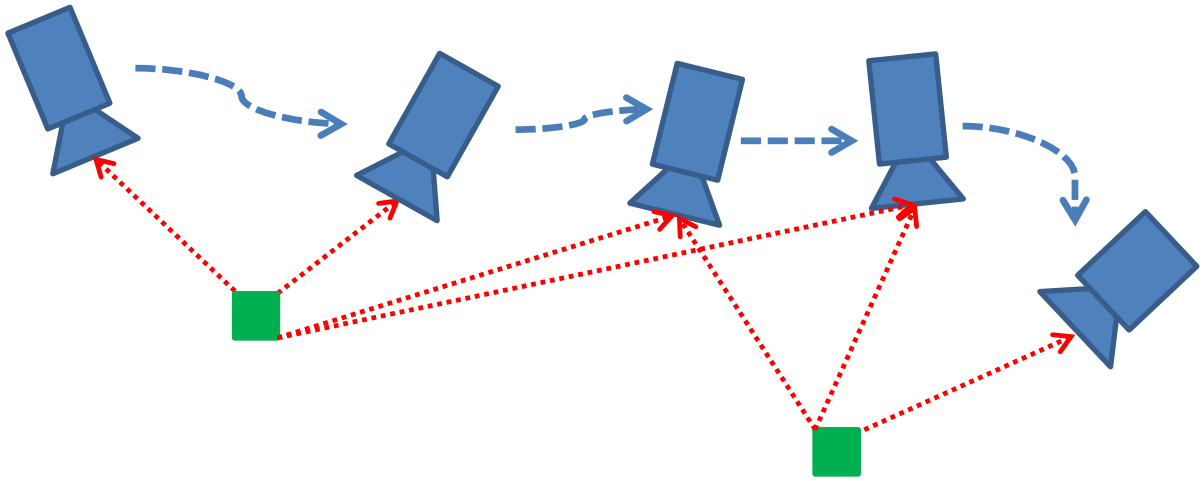
Intrinsic Calibration Matrix

Camera Pose

World Point

# Visual-Inertial SLAM



# Visual-Inertial SLAM

# 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

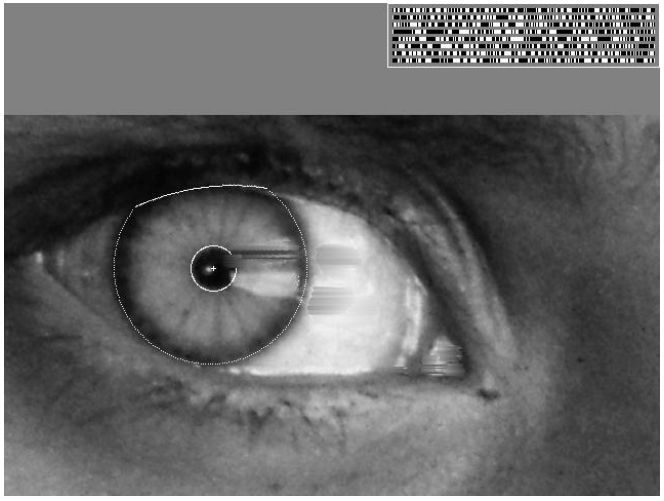
by Diva Sian

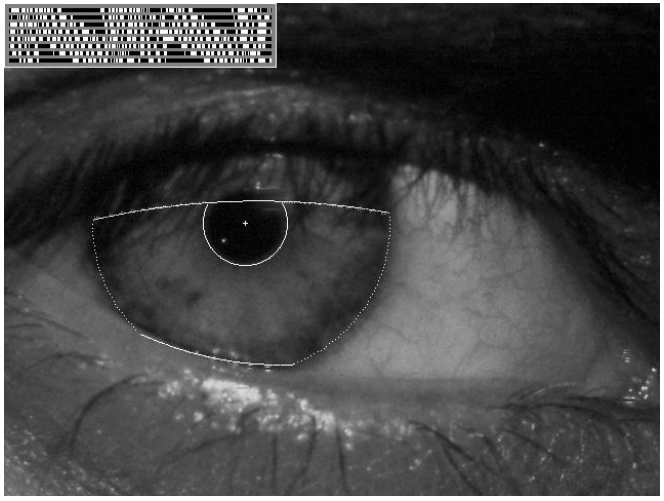
by scgbt

Courtesy Steve Seitz and Richard Szeliski

# Even harder case

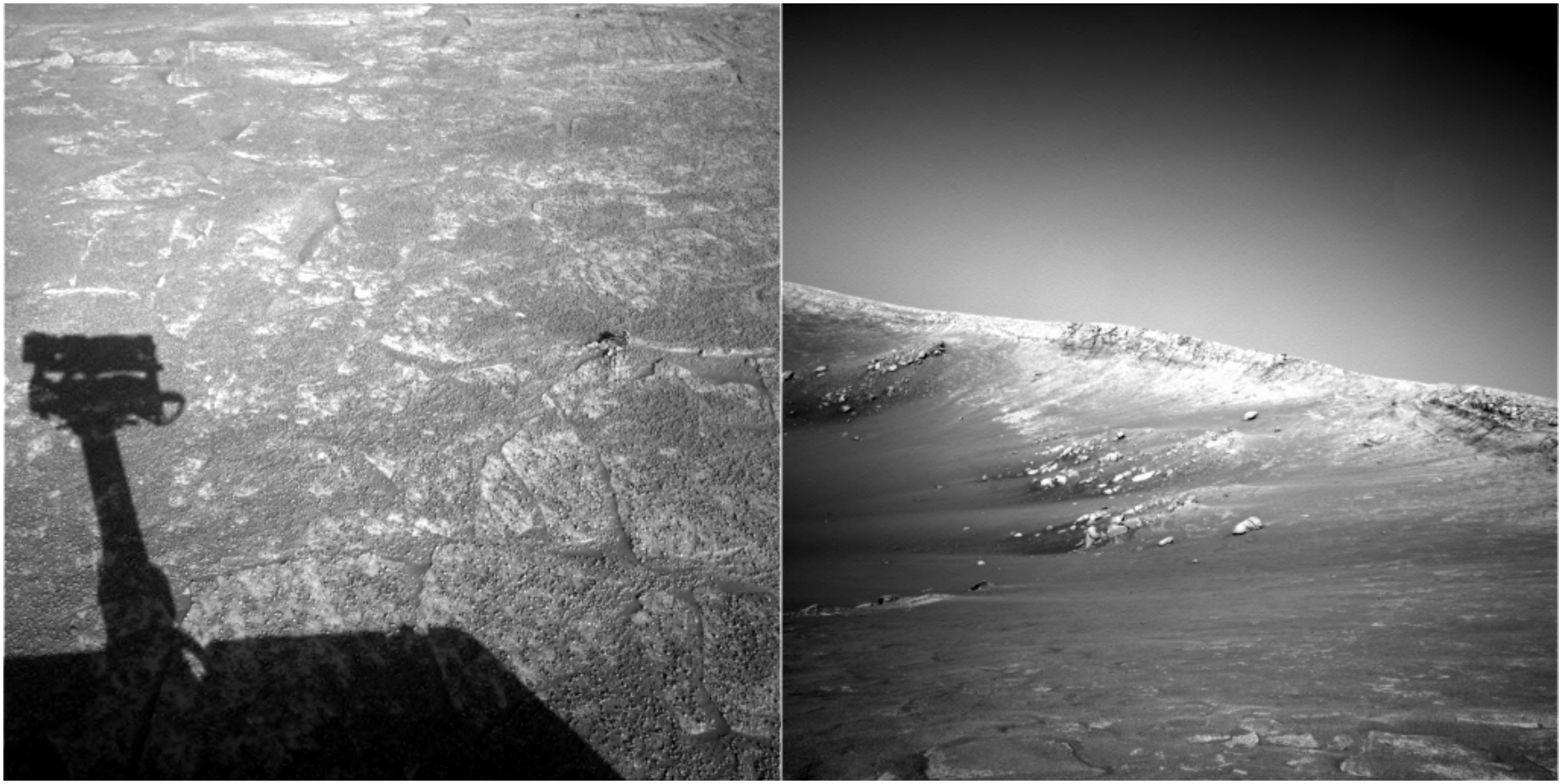
**"*How the Afghan Girl was Identified by Her Iris Patterns*" Read the [story]**





Courtesy Steve Seitz and Richard Szeliski

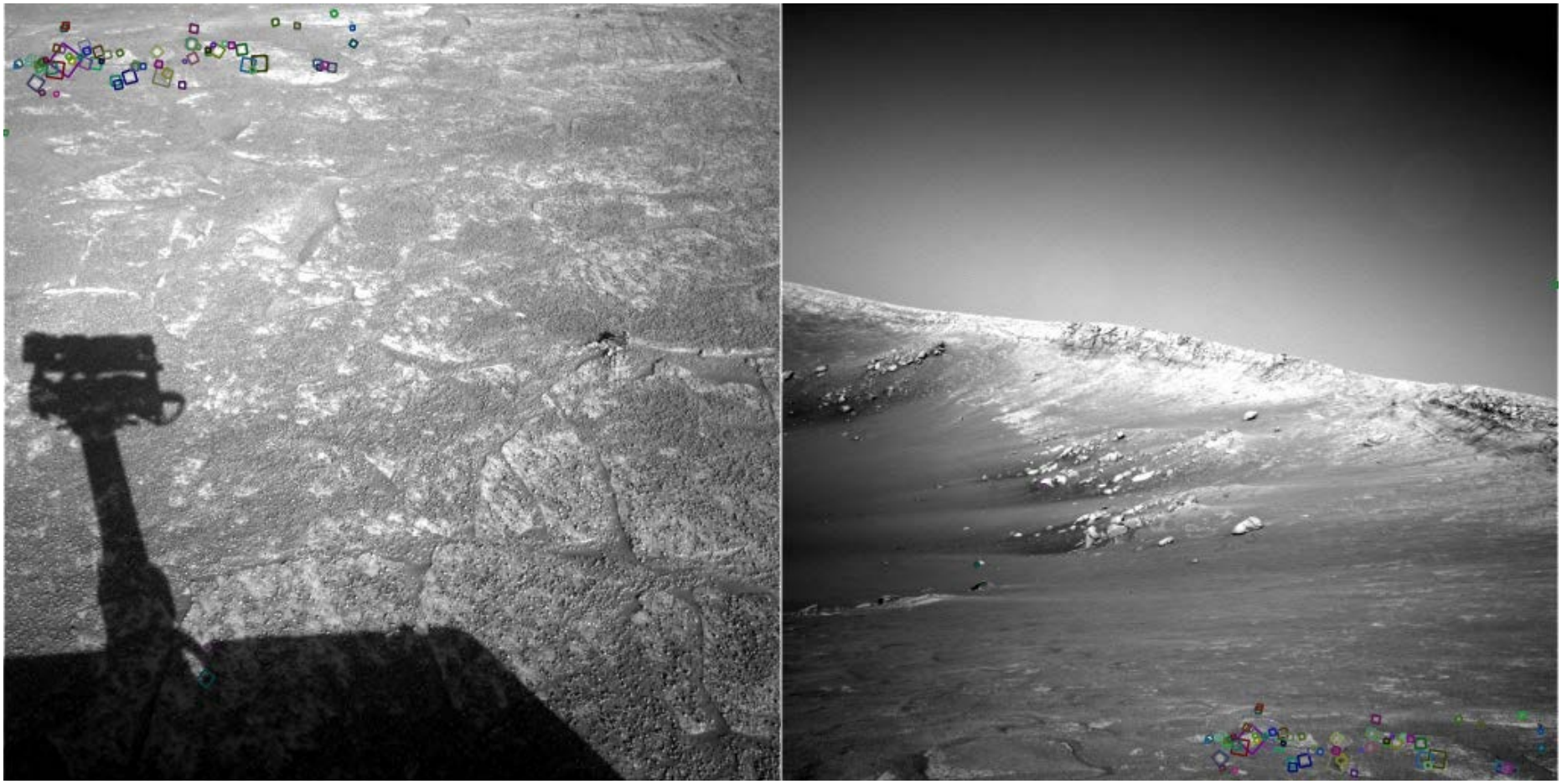
# Harder still?



NASA Mars Rover images

Courtesy Steve Seitz and Richard Szeliski

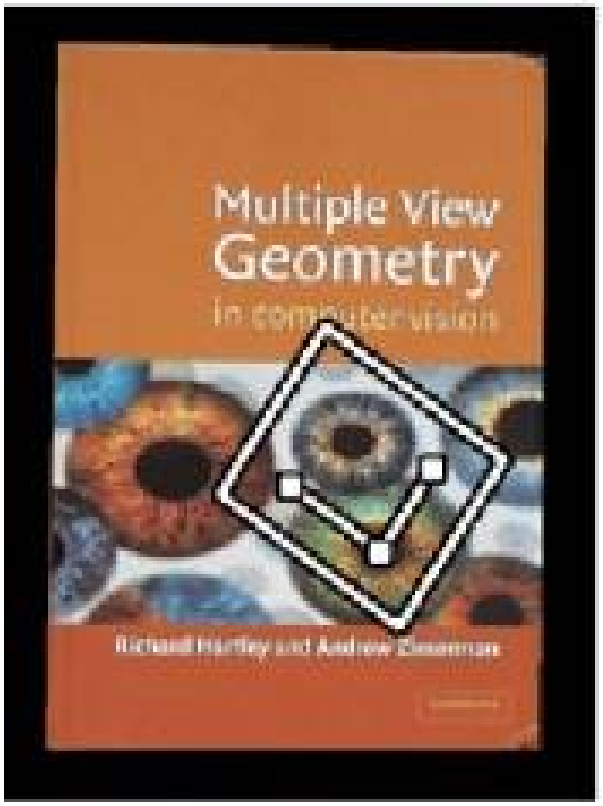
# Answer below (look for tiny colored squares…)



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

Courtesy Steve Seitz and Richard Szeliski

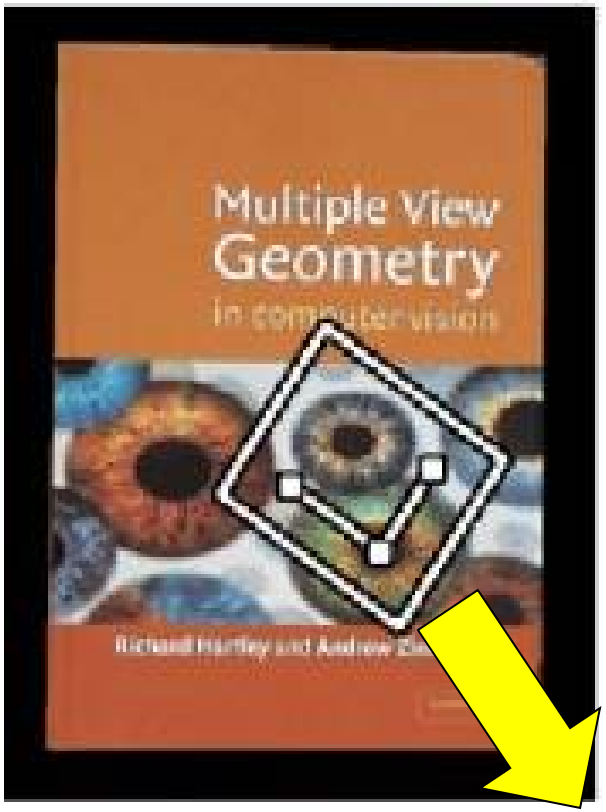
# Image Matching



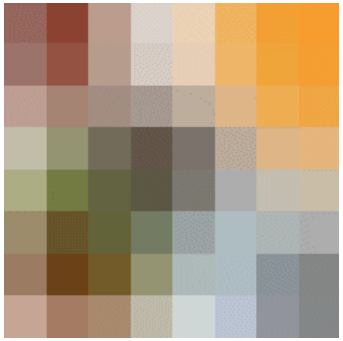


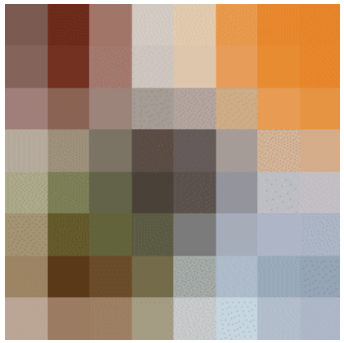
Courtesy Steve Seitz and Richard Szeliski

# Image Matching







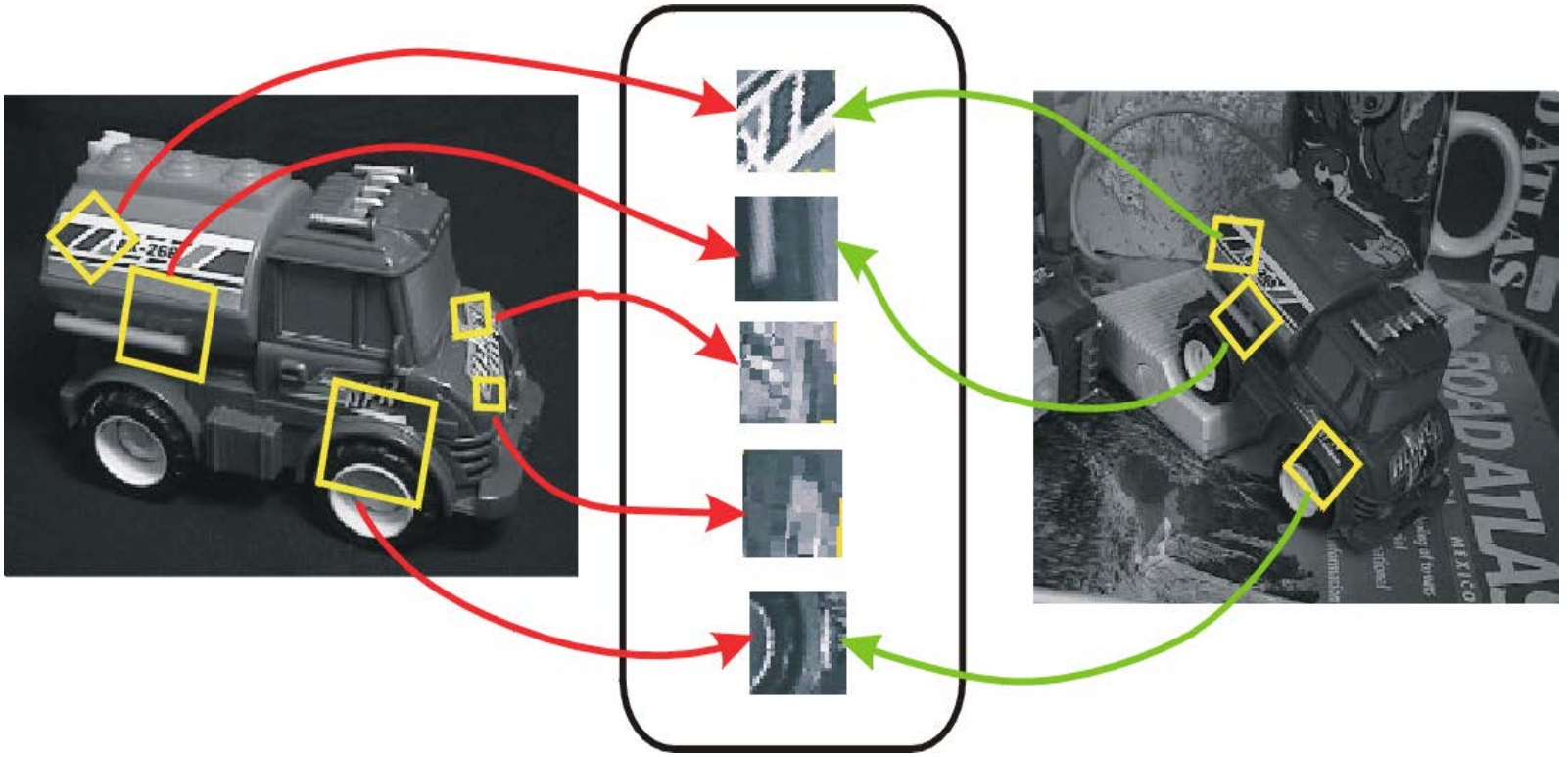


Courtesy Steve Seitz and Richard Szeliski

# 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

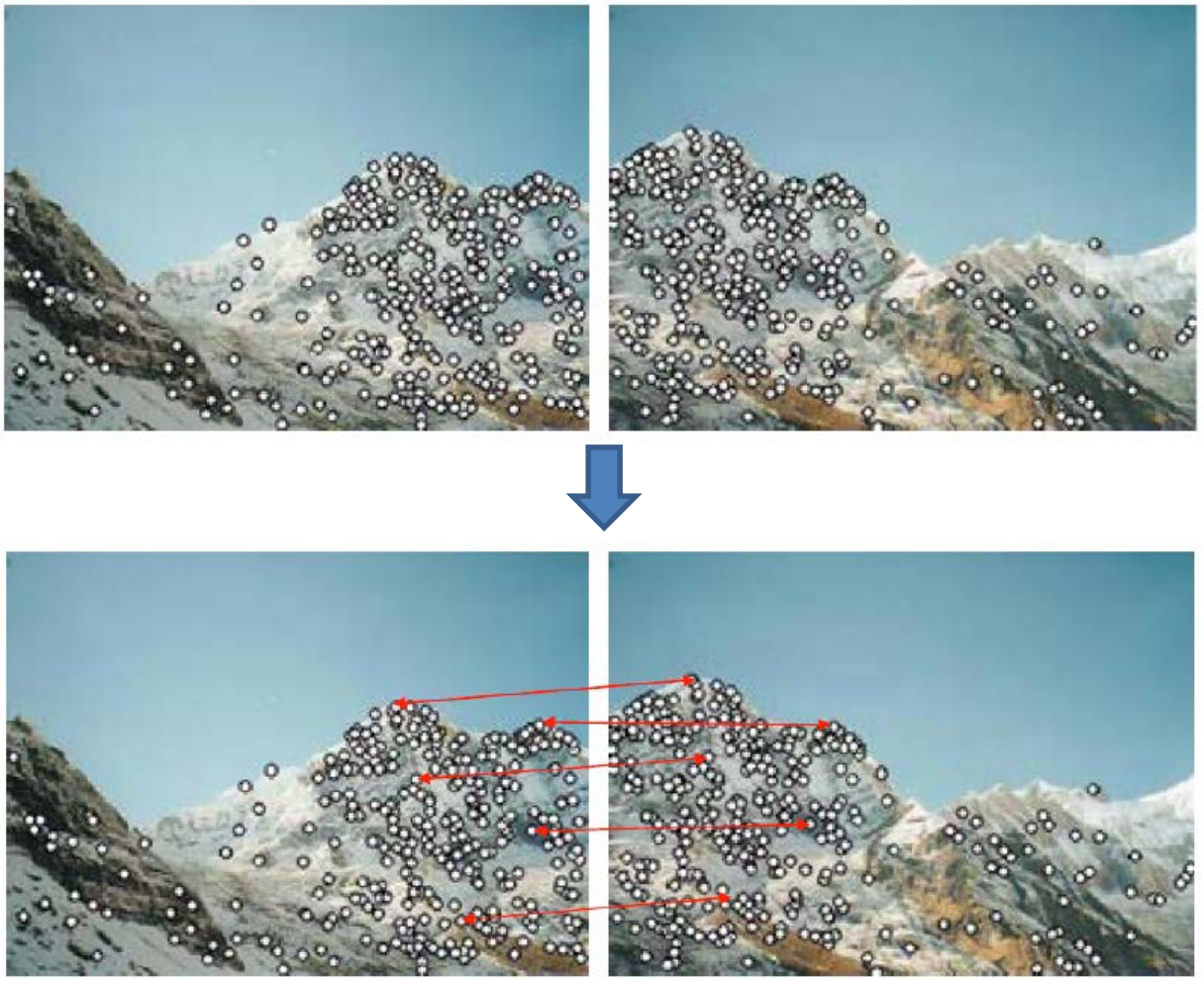
# More motivation…

Feature points are used for:

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

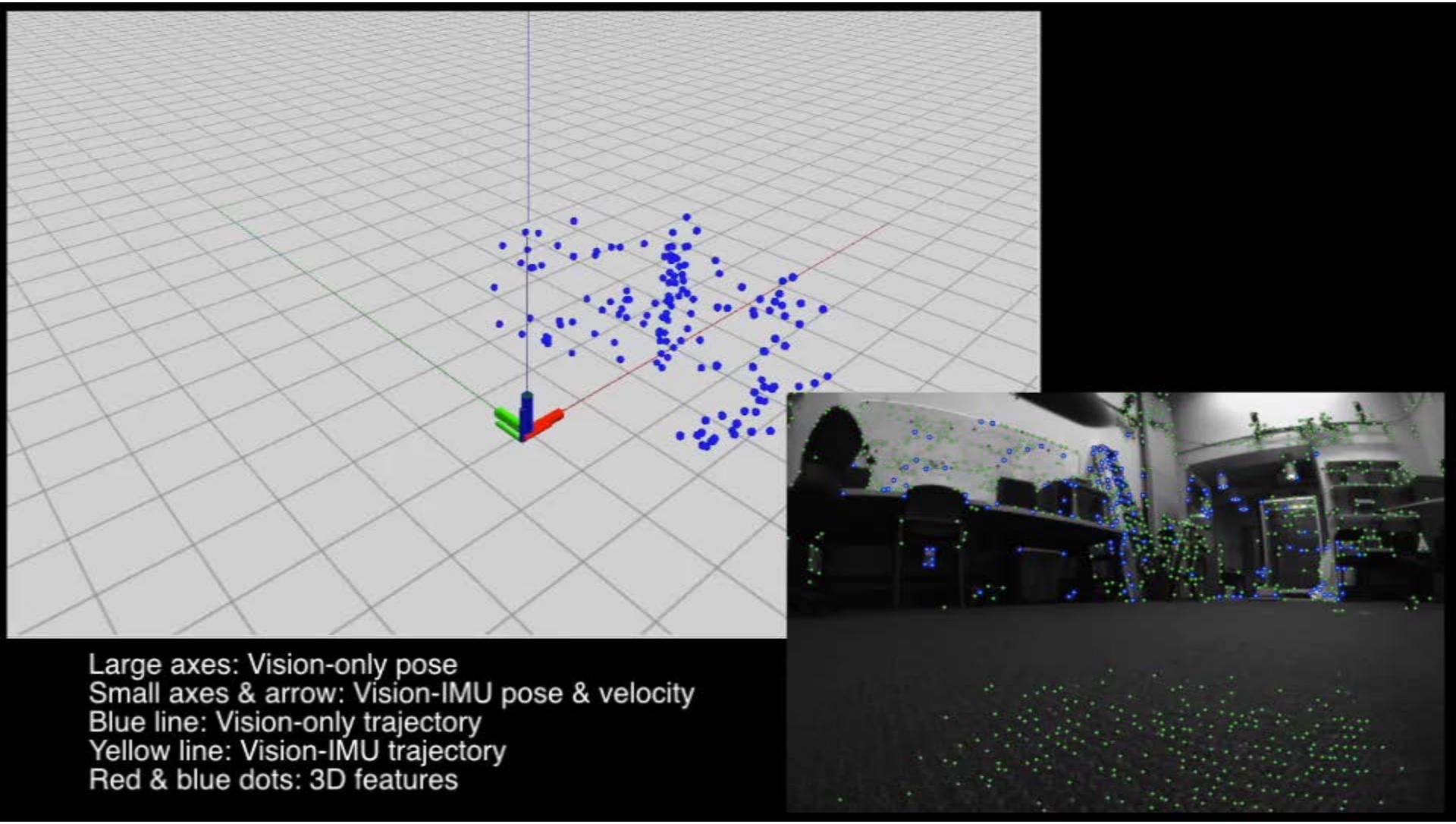
Courtesy Steve Seitz and Richard Szeliski

# Image Mosaics



# Image Mosaics

# Motion Tracking



# Want uniqueness

Look for image regions that are unusual

– Lead to unambiguous matches in other images

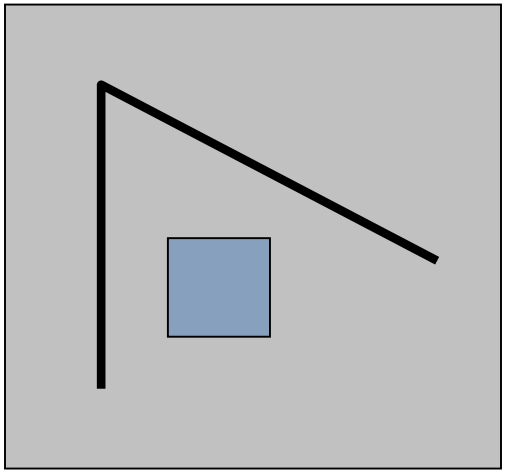
How to define “unusual”?

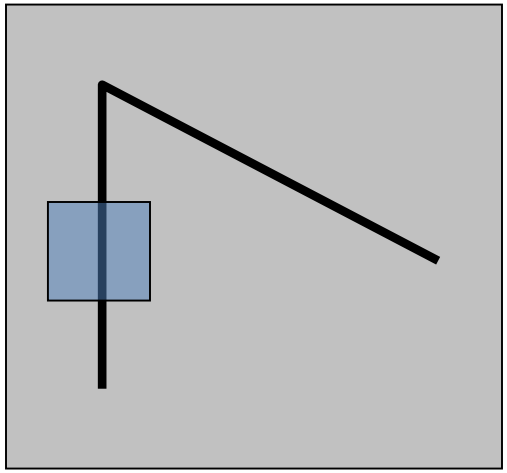
Courtesy Steve Seitz and Richard Szeliski

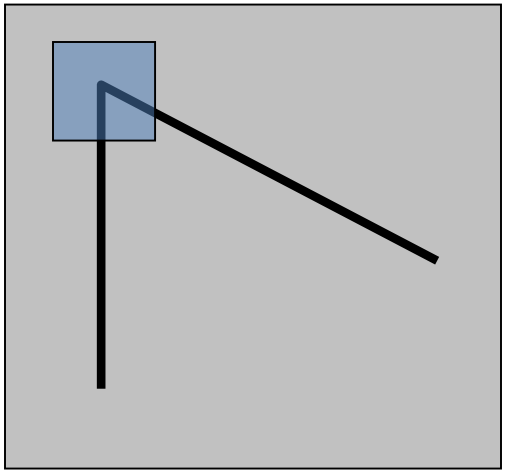
# Local measures of uniqueness

Suppose we only consider a small window of pixels

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







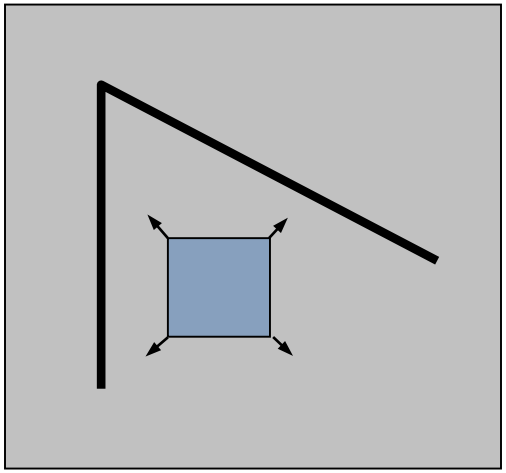
Slide adapted from Darya Frolova, Denis Simakov, Weizmann Institute.

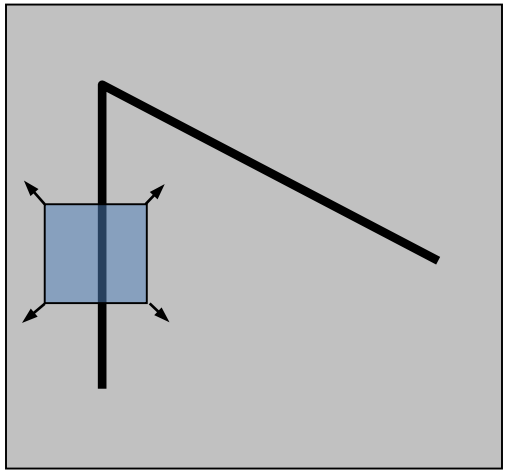
Courtesy Steve Seitz and Richard Szeliski

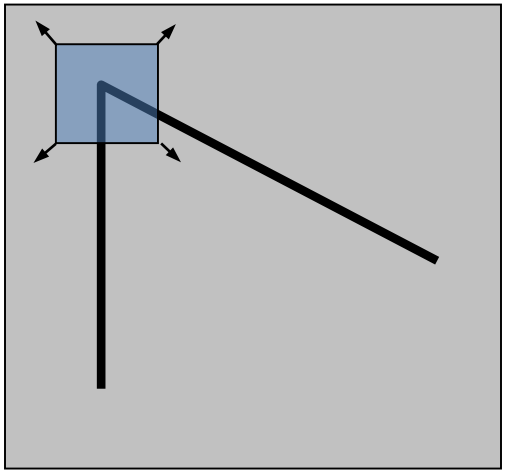
# Feature detection

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

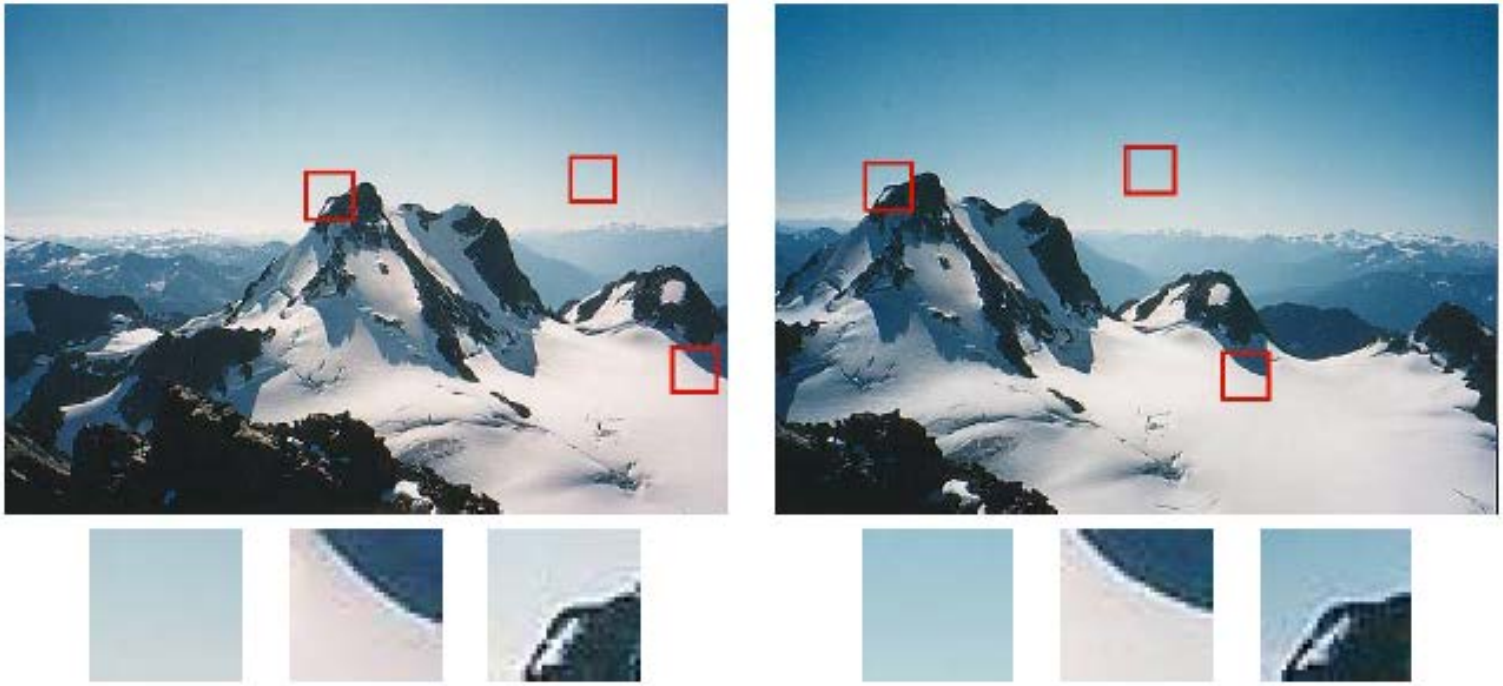
"edge":
no change along the edge direction

"corner":
significant change in all directions

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

Courtesy Steve Seitz and Richard Szeliski

# Feature detection

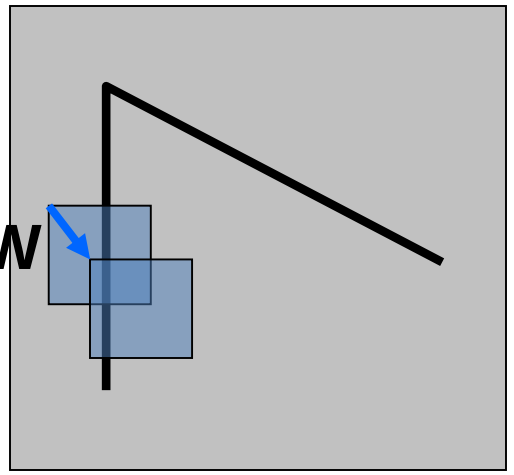


Courtesy Steve Seitz and Richard Szeliski

# Feature detection: the math

Consider shifting the window **W** by (u,v)

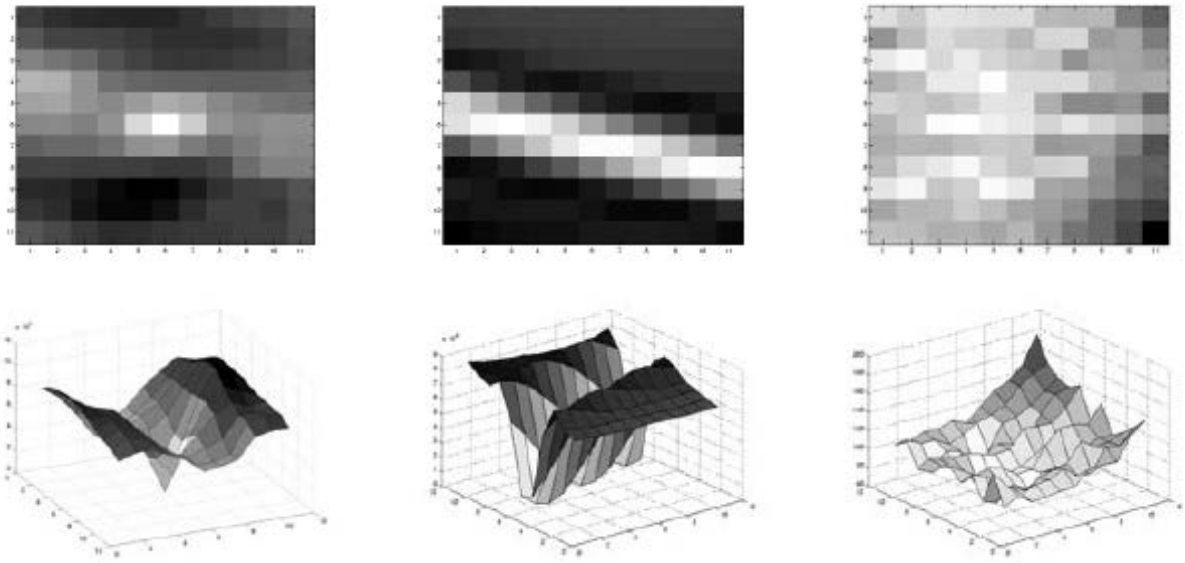
- how do the pixels in **W** change?
- compare each pixel before and after by summing up the squared differences (SSD)
- this defines an SSD "error" of *E(u,v)*:




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

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 approx is good

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

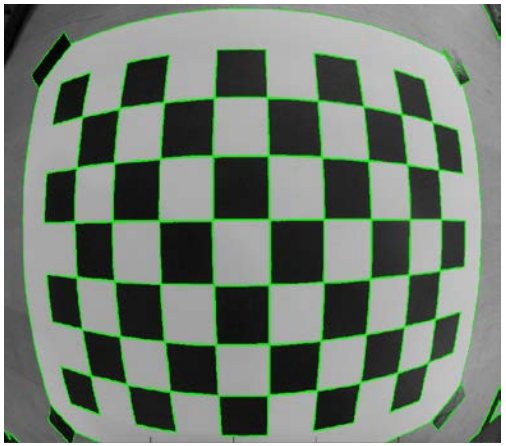
$$\approx I(x,y) + [I_x \ I_y] \begin{bmatrix} u \\ v \end{bmatrix}$$

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

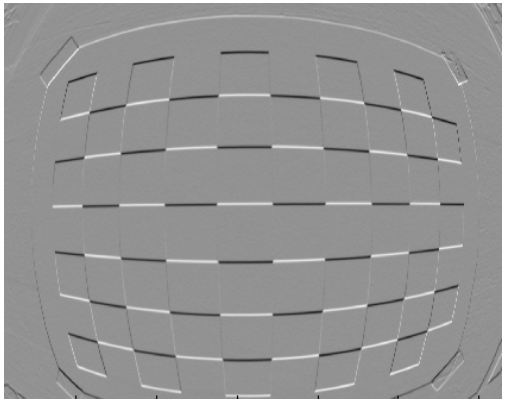
Plugging this into the formula on the previous slide…

Courtesy Steve Seitz and Richard Szeliski

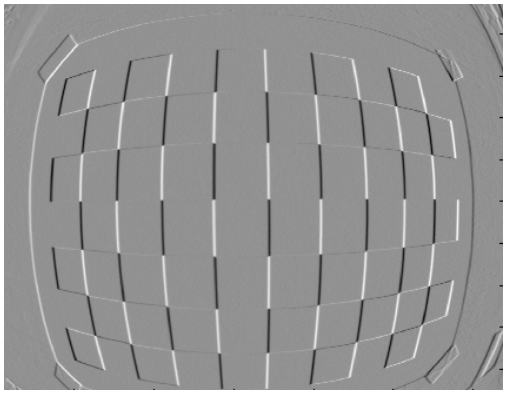
# Image Gradients



$I$



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

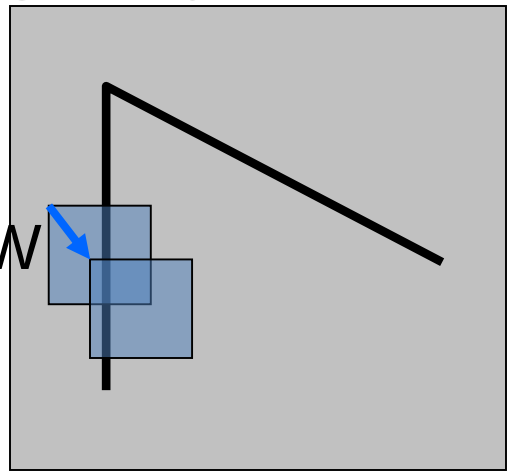


$I_y = I(x, y+1) = I(x, y-1)$

# 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):




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

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

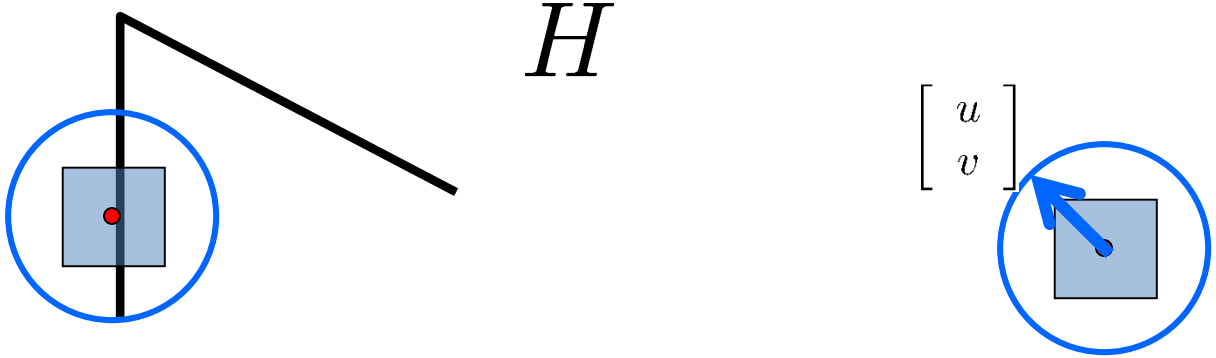
$$\approx \sum_{(x,y)\in W} \left[ [I_x \; I_y] \begin{bmatrix} u \\ v \end{bmatrix} \right]^2$$

Courtesy Steve Seitz and Richard Szeliski

# Feature detection: the math

This can be rewritten:

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




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

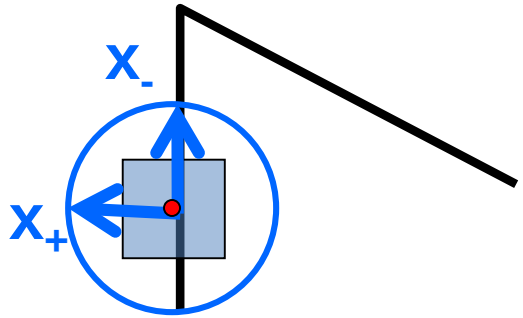
$$\begin{bmatrix} h_{11} - \lambda & h_{12} \\ h_{21} & h_{22} - \lambda \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = 0$$

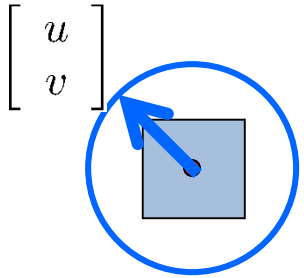
Courtesy Steve Seitz and Richard Szeliski

# Feature detection: the math

This can be rewritten:

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




Eigenvalues and eigenvectors of H

- Define shifts with the smallest and largest change (E value)
- $x_+$ = direction of **largest** increase in E.
- $\lambda_+$ = amount of increase in direction $x_+$
- $x_-$ = direction of **smallest** increase in E.
- $\lambda_-$ = amount of increase in direction $x_+$

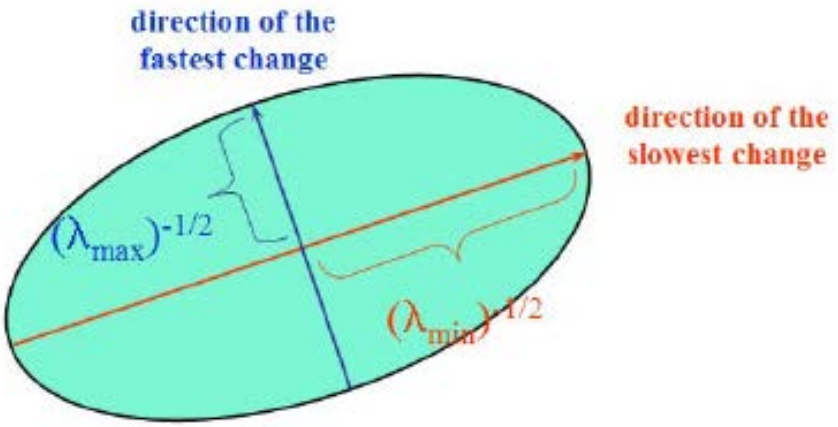
$$Hx_+ = \lambda_+ x_+$$
$$Hx_- = \lambda_- x_-$$

Courtesy Steve Seitz and Richard Szeliski

# Feature detection: the math

How are $\lambda_+$, $\mathbf{x}_+$, $\lambda_-$, and $\mathbf{x}_+$ relevant for feature detection?

- What's our feature scoring function?




Courtesy Steve Seitz and Richard Szeliski

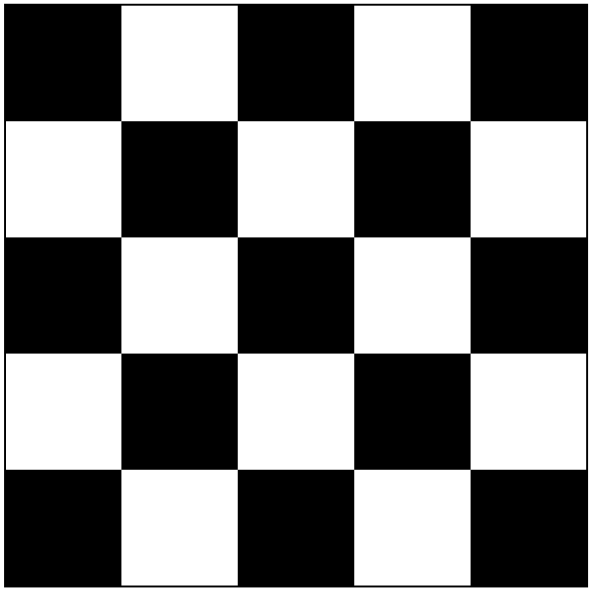
# Feature detection: the math

How are $\lambda_+$, $\mathbf{x}_+$, $\lambda_-$, and $\mathbf{x}_+$ relevant for feature detection?

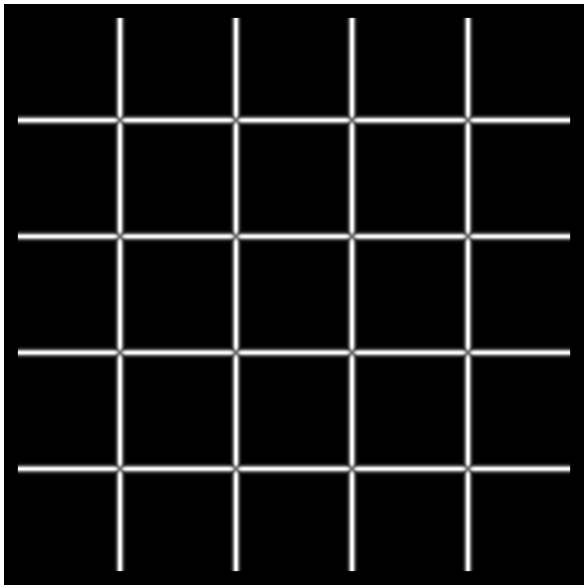
- What's our feature scoring function?

Want *E(u,v)* to be ***large*** for small shifts in ***all*** directions

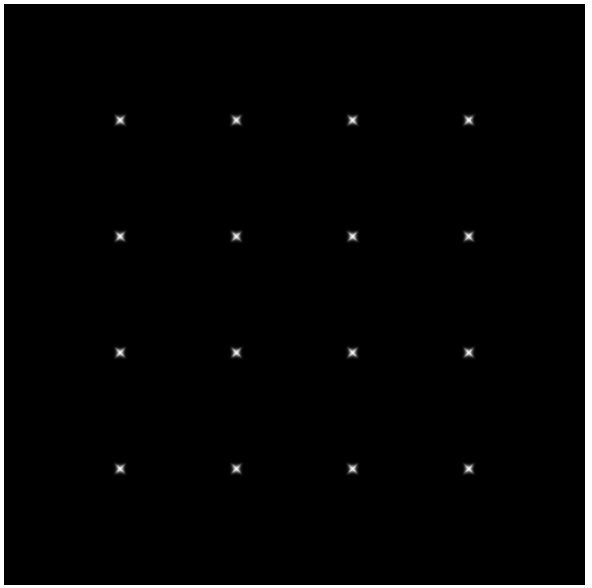
- the *minimum* of *E(u,v)* should be large, over all unit vectors [u v]
- this minimum is given by the smaller eigenvalue ($\lambda_-$) of ***H***



$I$



$\lambda_+$



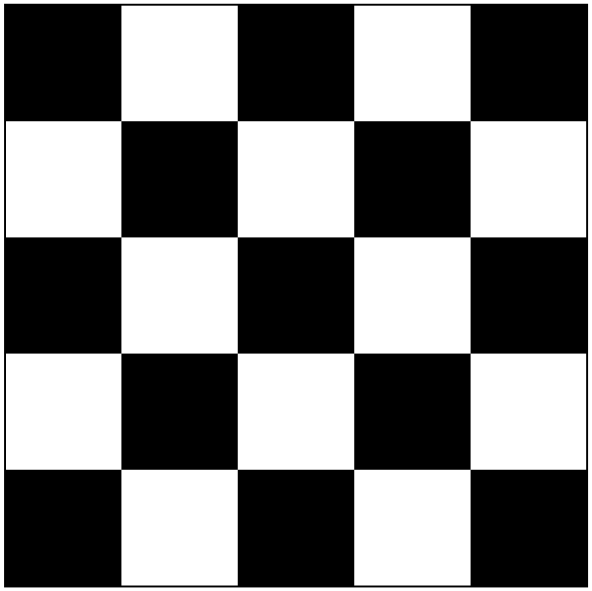
$\lambda_-$

Courtesy Steve Seitz and Richard Szeliski

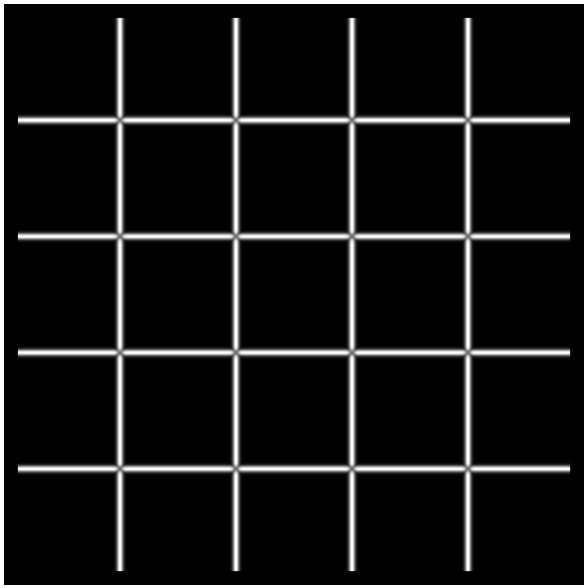
# Feature detection summary

Here's what you do

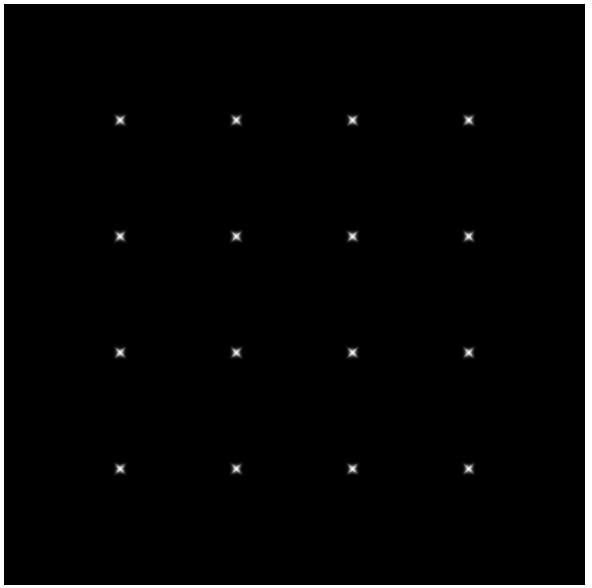
- Compute the gradient at each point in the image
- Create the ***H*** matrix from the entries in the gradient
- Compute the eigenvalues.
- Find points with large response ($\lambda_-$ > threshold)
- Choose those points where $\lambda_-$ is a local maximum as features



$I$



$\lambda_+$



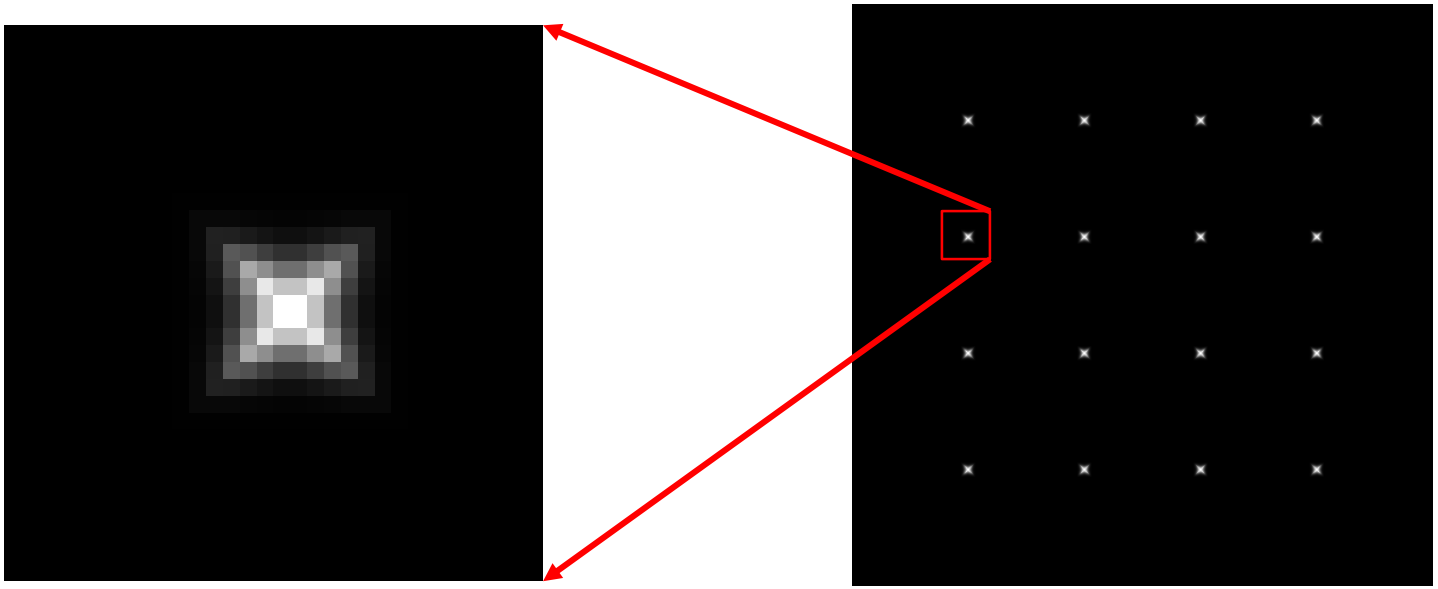
$\lambda_-$

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 ($\lambda_-$ > threshold)
- Choose those points where $\lambda_-$ is a local maximum as features



$\lambda_-$

Courtesy Steve Seitz and Richard Szeliski

# The Harris operator

$\lambda_-$ is a variant of the "Harris operator" for feature detection

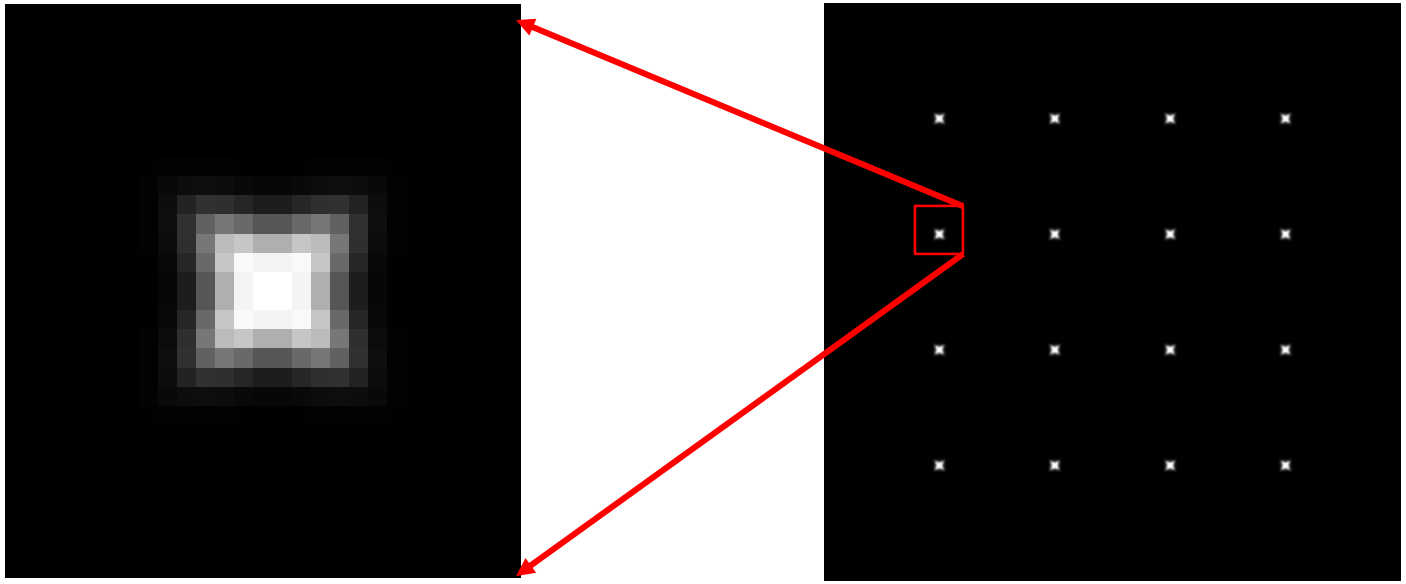
$$f = \frac{\lambda_1 \lambda_2}{\lambda_1 + \lambda_2}$$

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

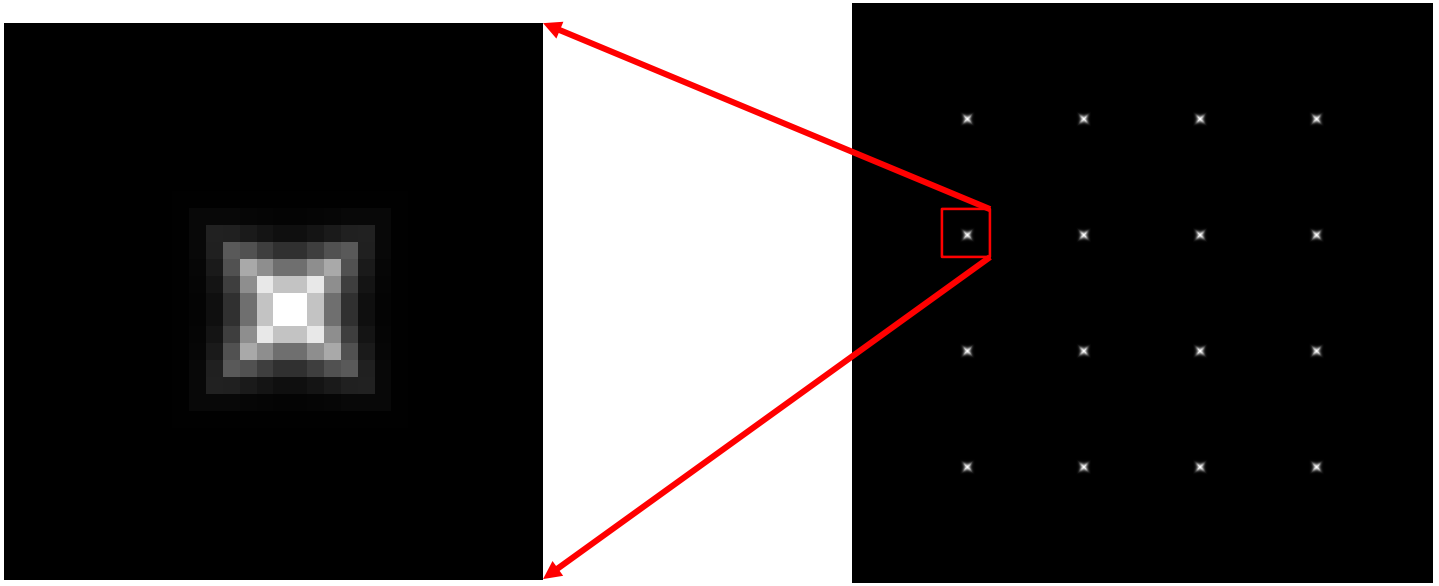
- The *trace* is the sum of the diagonals, i.e., *trace(H)* = $h_{11} + h_{22}$
- Very similar to $\lambda_-$ but less expensive (no square root)
- Called the "Harris Corner Detector" or "Harris Operator"
- Lots of other detectors, this is one of the most popular

Courtesy Steve Seitz and Richard Szeliski

# The Harris operator



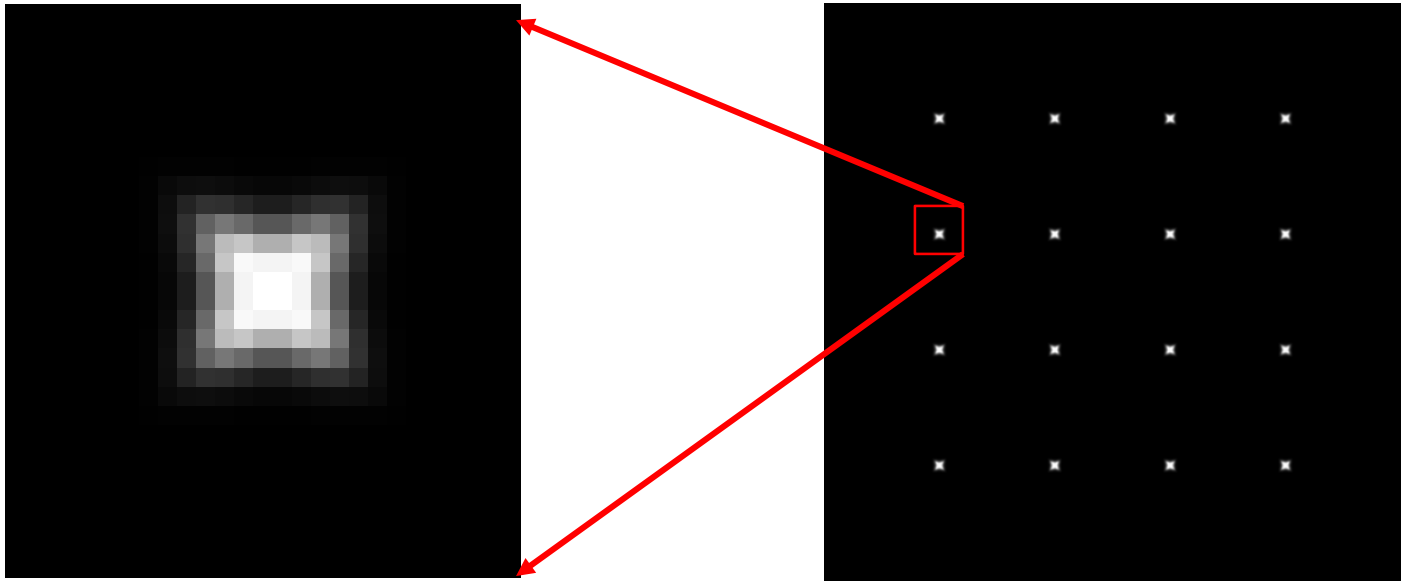
Harris operator



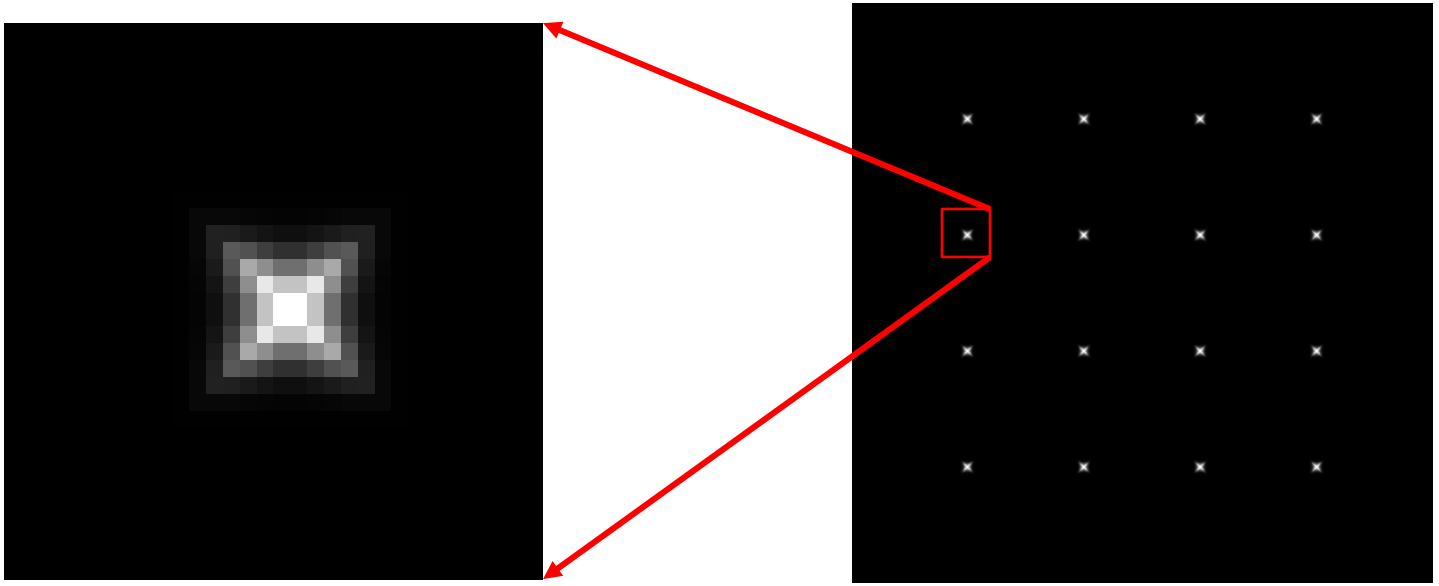
$\lambda_-$

Courtesy Steve Seitz and Richard Szeliski

# The Harris operator



Harris operator



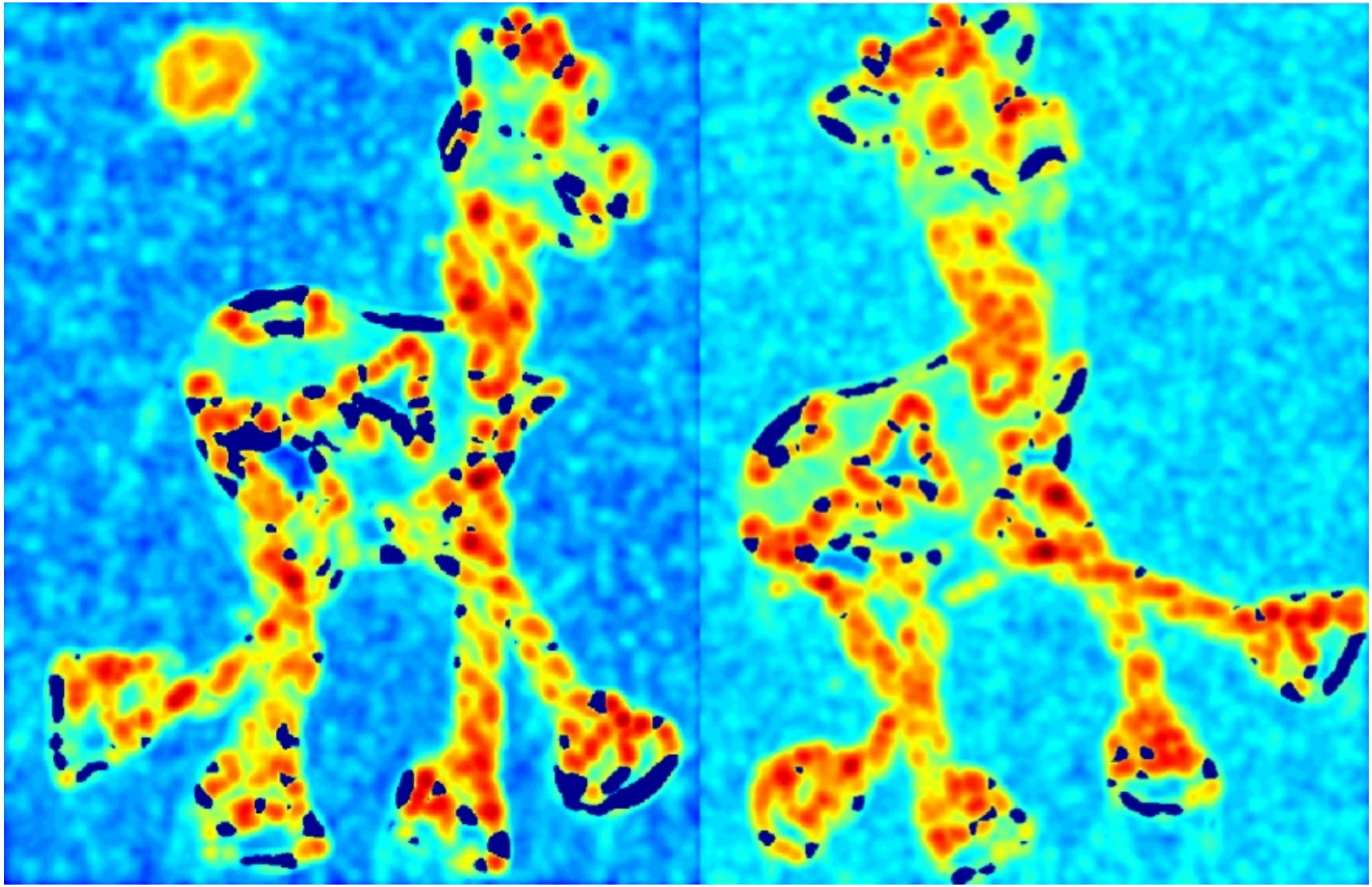
$\lambda_-$

Courtesy Steve Seitz and Richard Szeliski

# Harris detector example

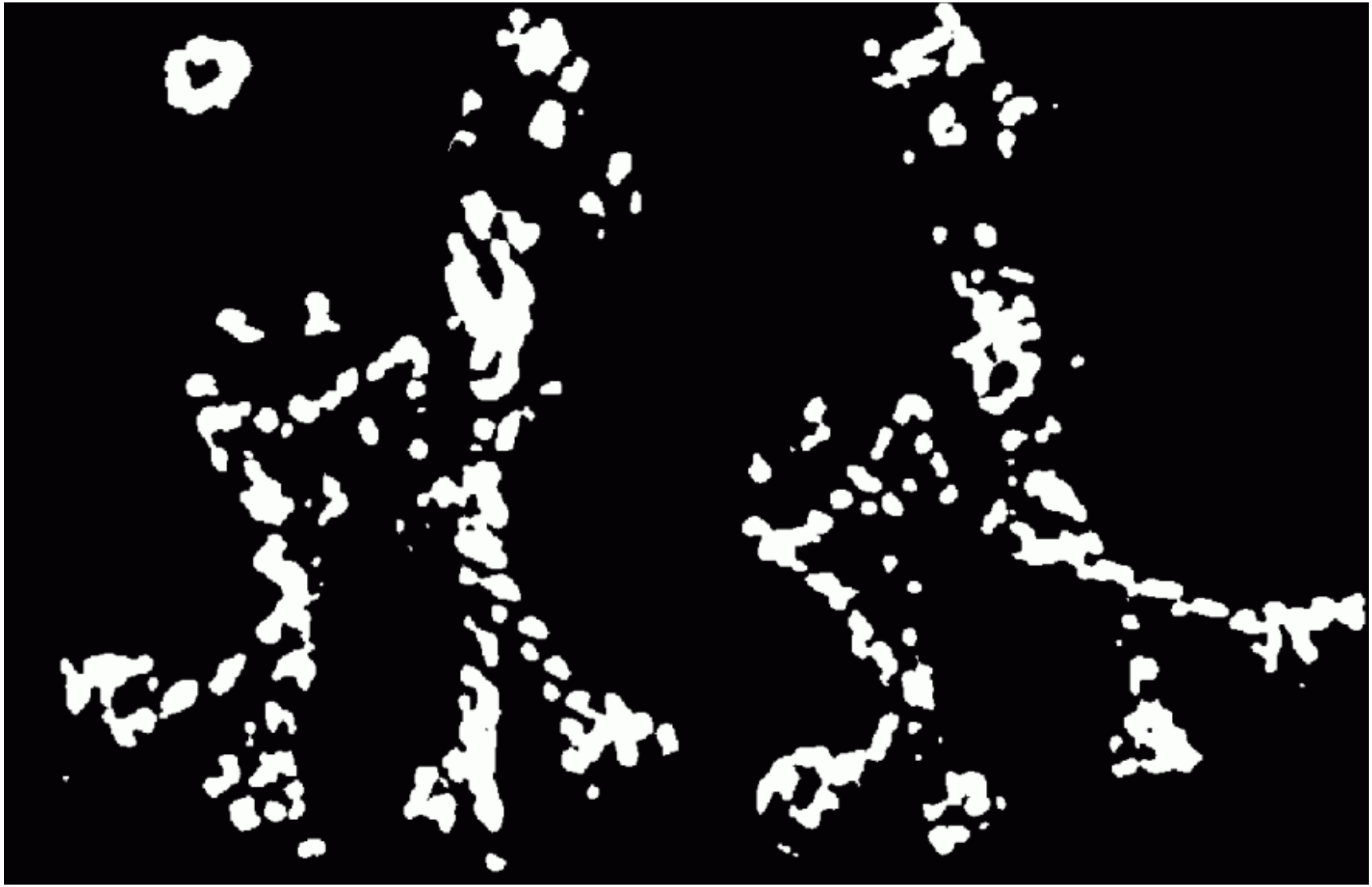
Courtesy Steve Seitz and Richard Szeliski

# f value (red high, blue low)



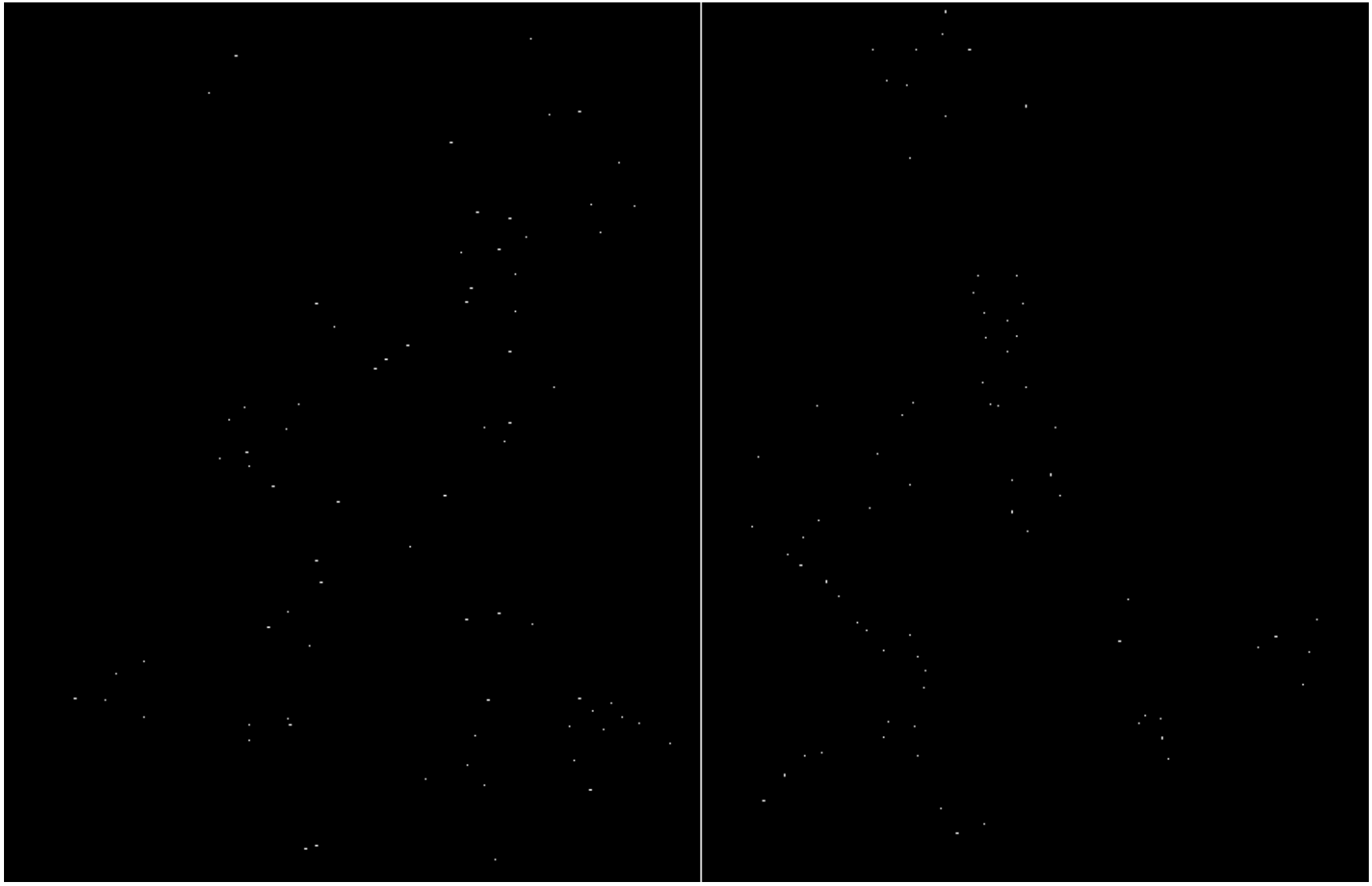
Courtesy Steve Seitz and Richard Szeliski

# 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?

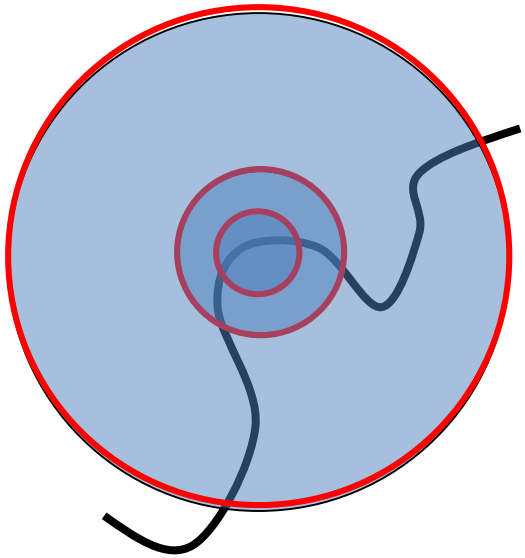
What if you change the brightness?

Scale?

Courtesy Steve Seitz and Richard Szeliski

# Scale invariant detection

Suppose you're looking for corners



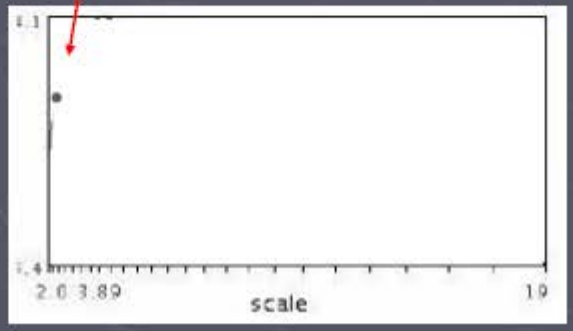
Key idea: find scale that gives local maximum of f

- f is a local maximum in both position and scale
- Common definition of f: Laplacian
  (or difference between two Gaussian filtered images with different sigmas)

Courtesy Steve Seitz and Richard Szeliski

# Automatic scale selection

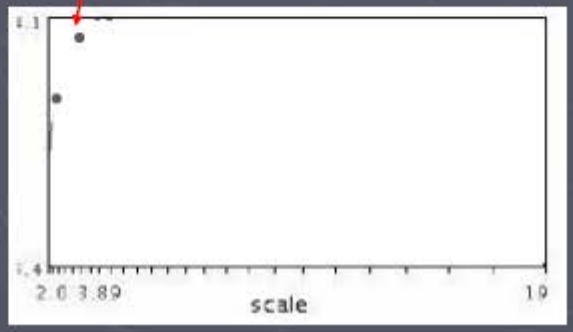
Lindeberg et al., 1996




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

Slide from Tinne Tuytelaars Courtesy Steve Seitz and Richard Szeliski

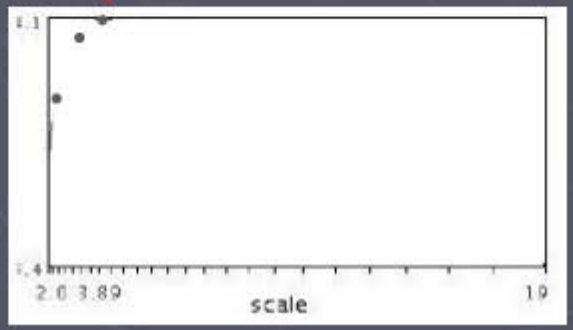
# Automatic scale selection




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

Courtesy Steve Seitz and Richard Szeliski

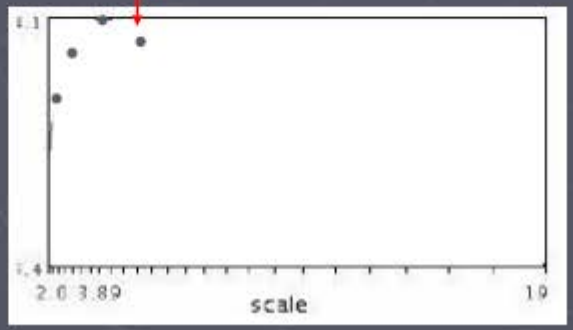
# Automatic scale selection




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

Courtesy Steve Seitz and Richard Szeliski

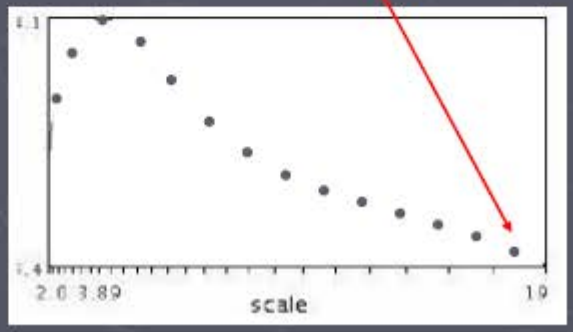
# Automatic scale selection




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

Courtesy Steve Seitz and Richard Szeliski

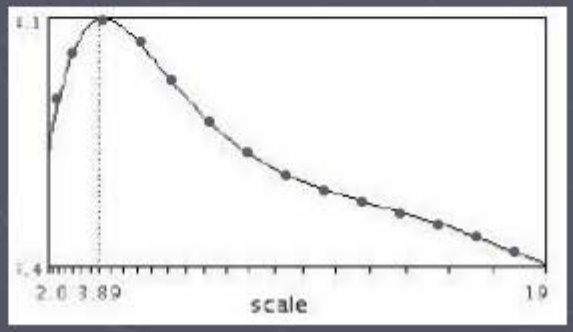
# Automatic scale selection




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

Courtesy Steve Seitz and Richard Szeliski

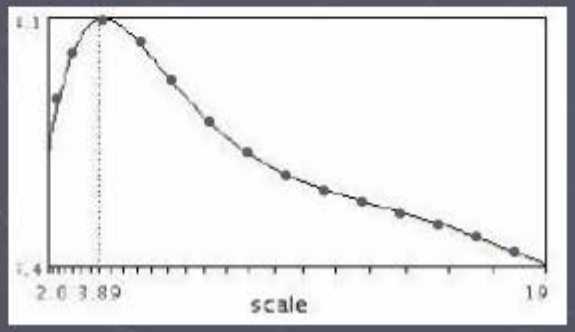
# Automatic scale selection



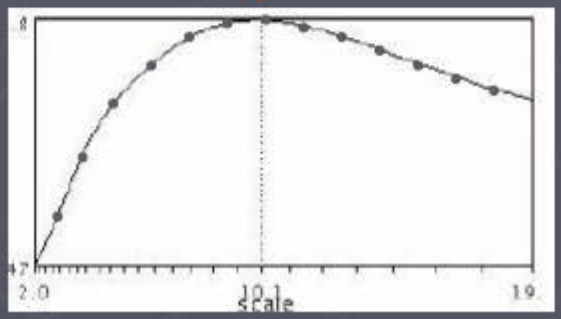

$f(I_{i_1 \ldots i_m}(x,\sigma))$

Courtesy Steve Seitz and Richard Szeliski

# Automatic scale selection




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

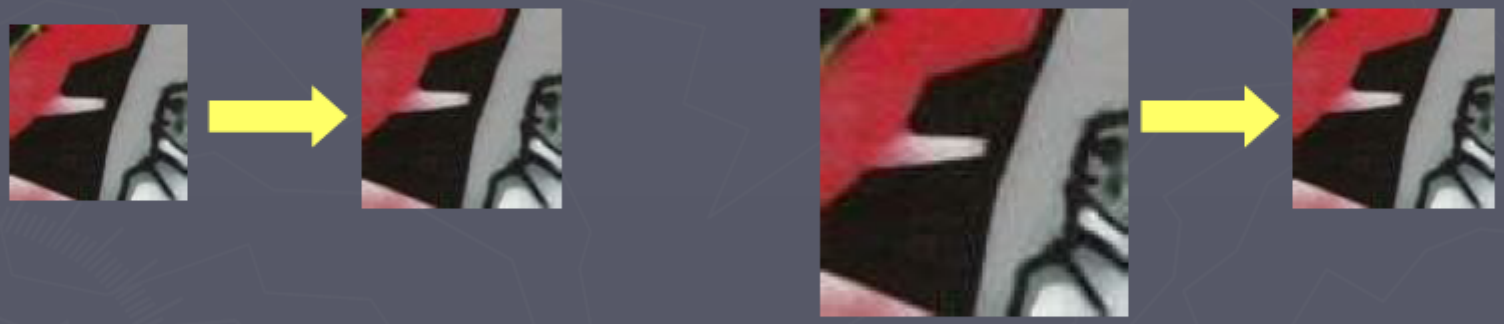



$f(I_{i_1 \ldots i_m}(x',\sigma'))$

Courtesy Steve Seitz and Richard Szeliski

# Automatic scale selection

Normalize: rescale to fixed size

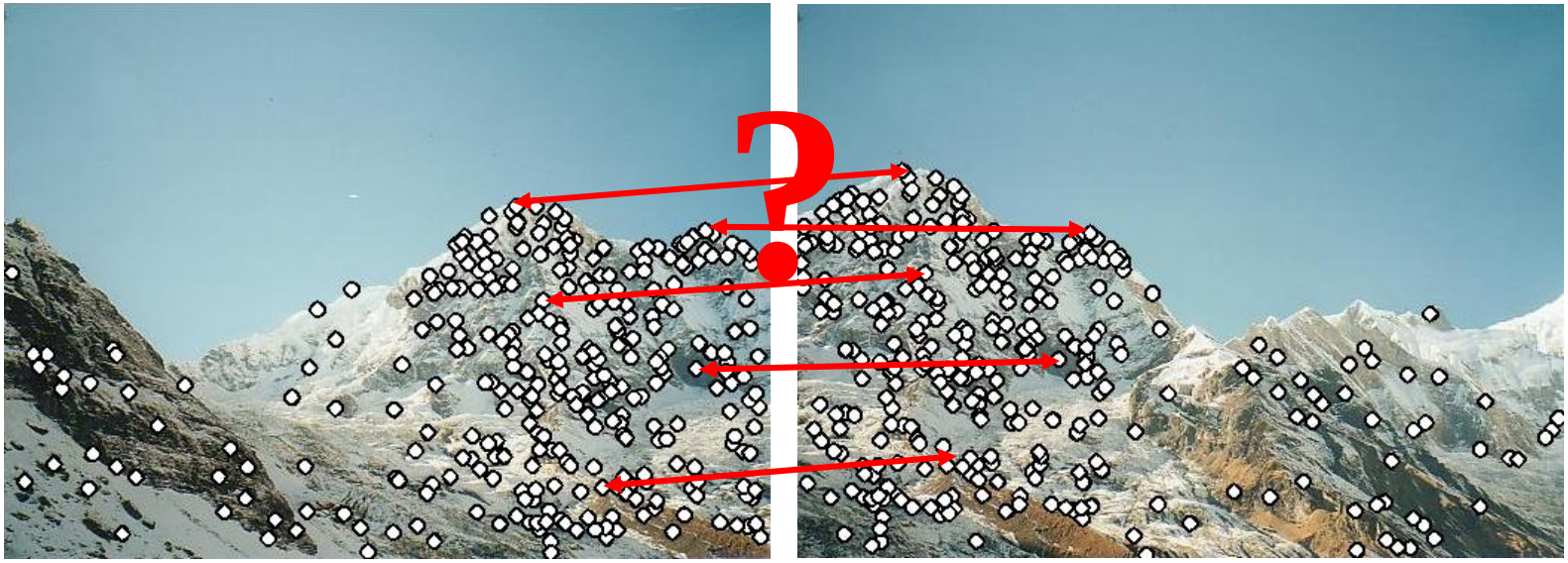


Courtesy Steve Seitz and Richard Szeliski

# Feature descriptors

We know how to detect good points

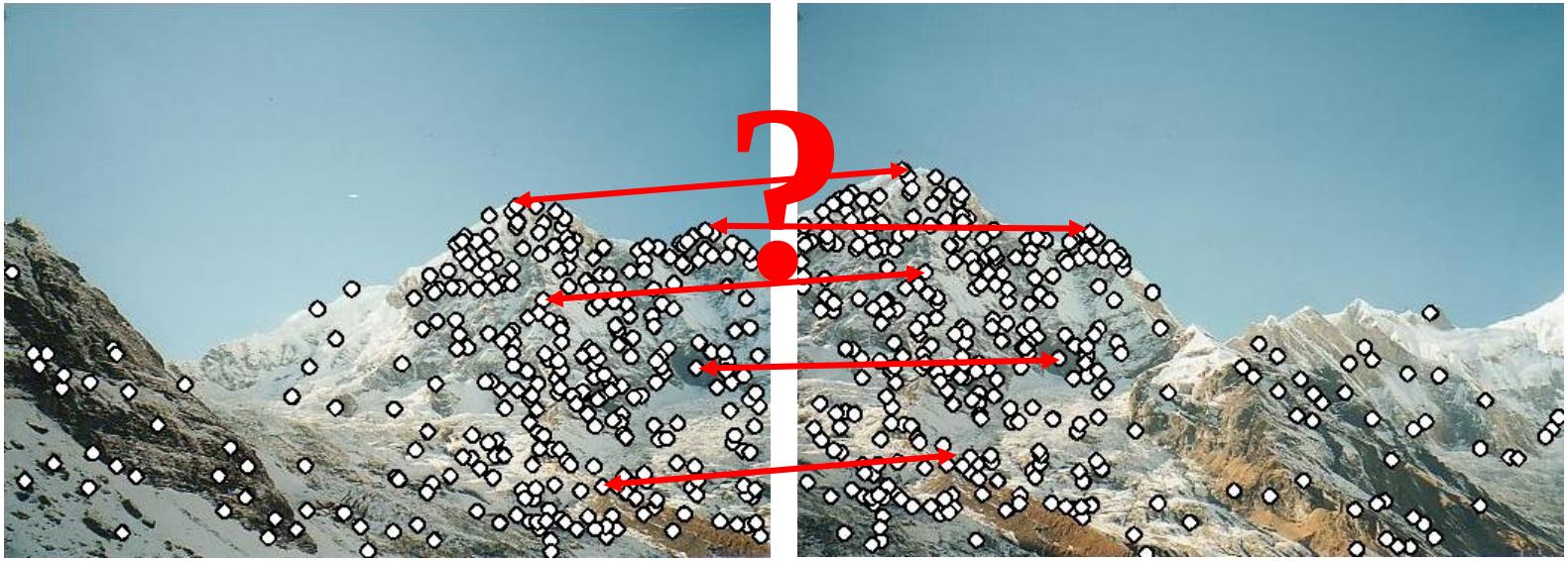
Next question: **How to match them?**




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
- State of the art 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 $\lambda_+$
  - $\lambda_+$ is the *larger* eigenvalue
- Rotate the patch according to this angle

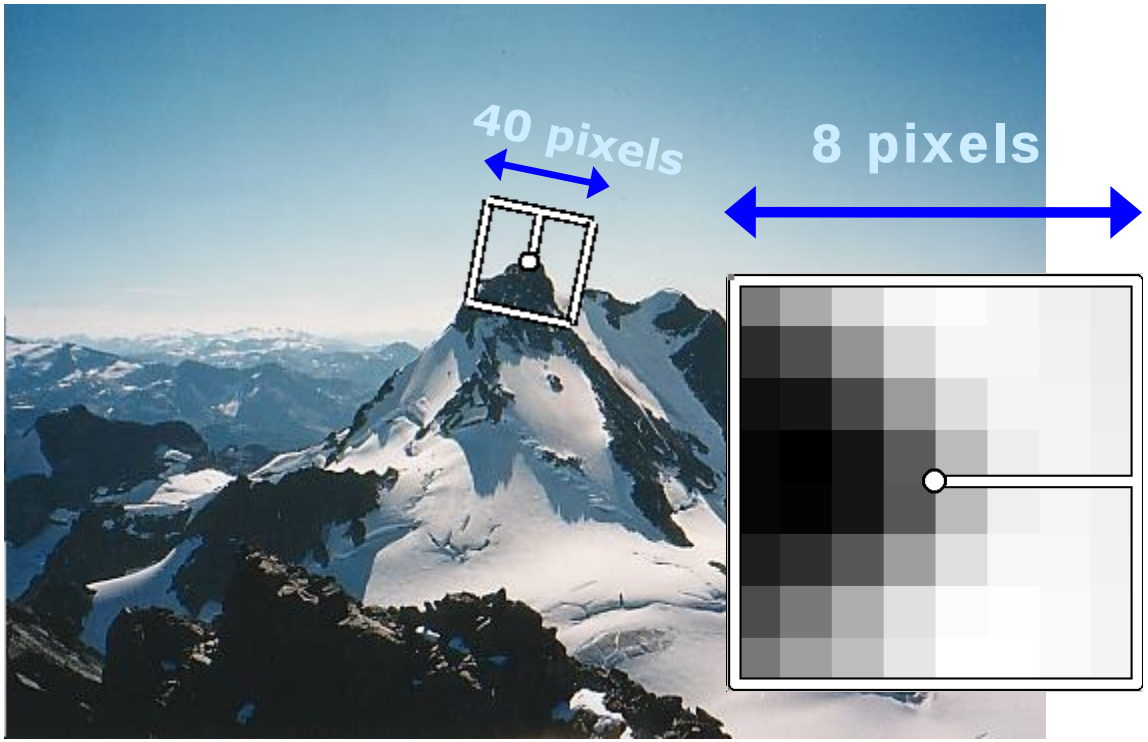
Figure by Matthew Brown

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




Adapted from slide by Matthew Brown

Courtesy Steve Seitz and Richard Szeliski

# Detections at multiple scales

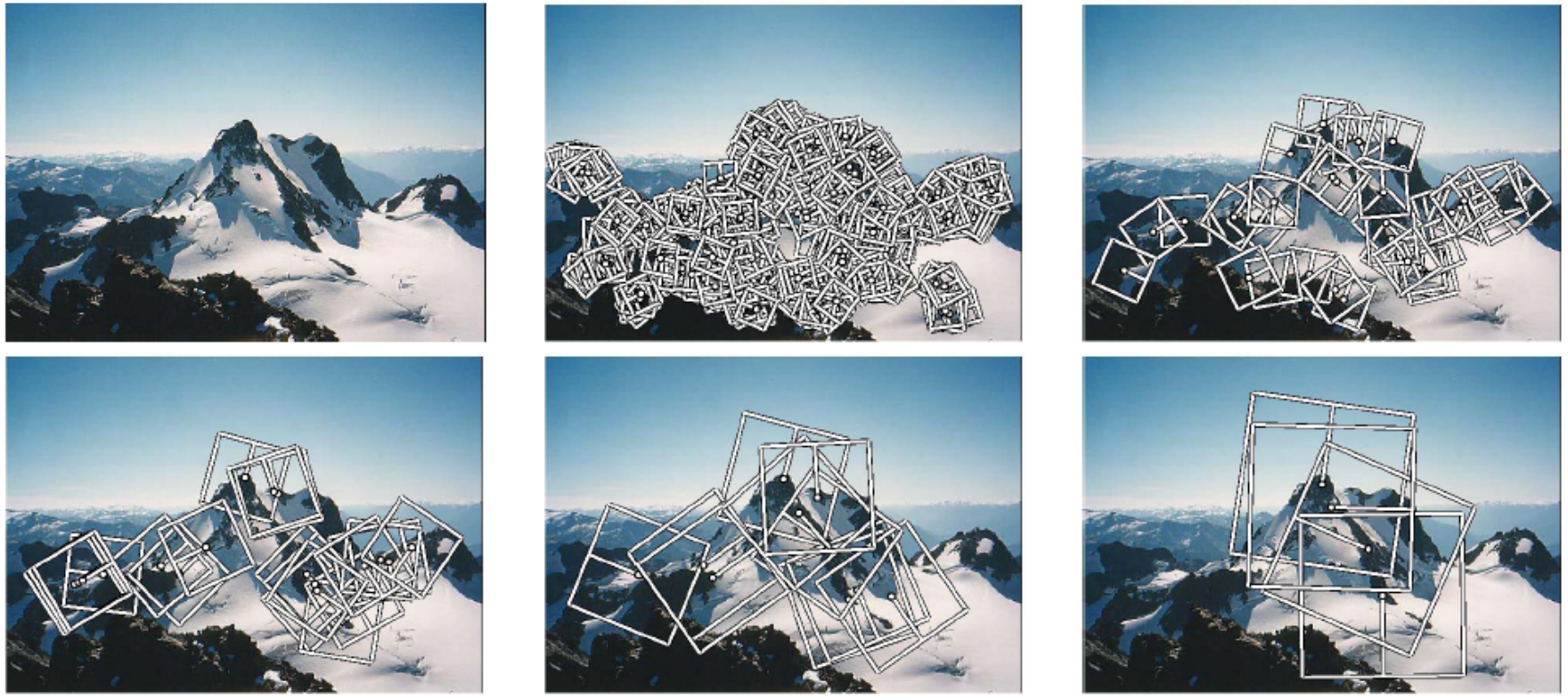


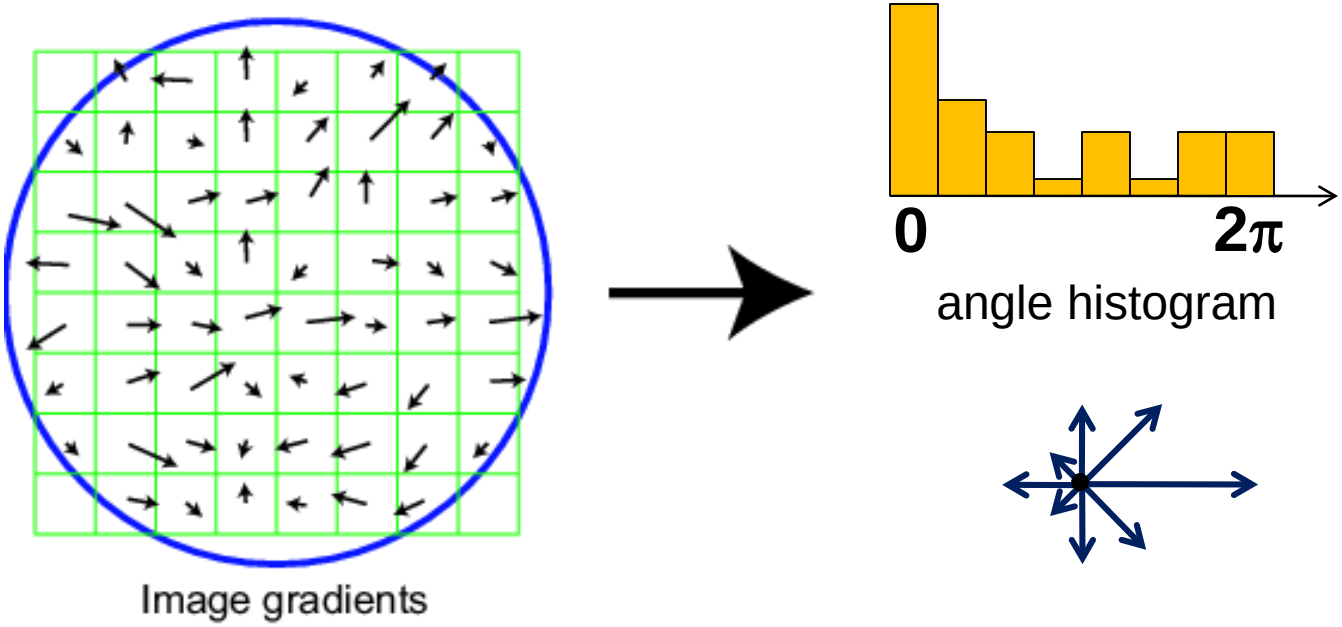
*Figure 1. Multi-scale Oriented Patches (MOPS) extracted at five pyramid levels from one of the Matier images. The boxes show the feature orientation and the region from which the descriptor vector is sampled.*

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



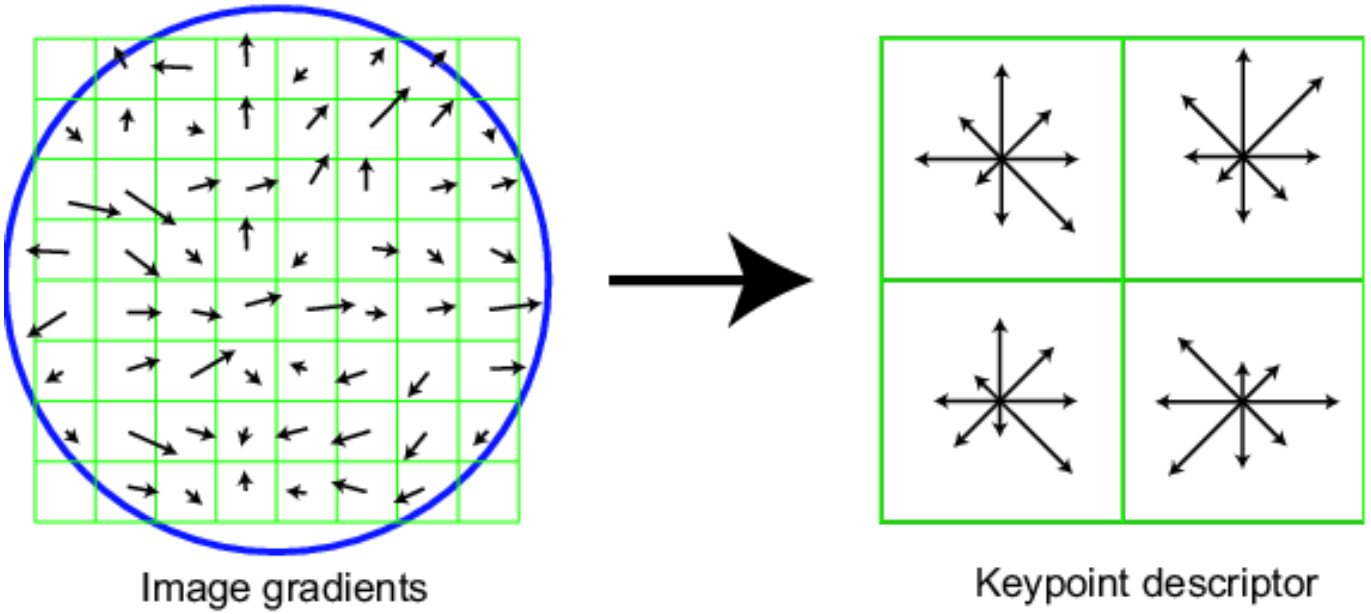

Adapted from slide by David Lowe

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




Adapted from slide by 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 $I_2$?

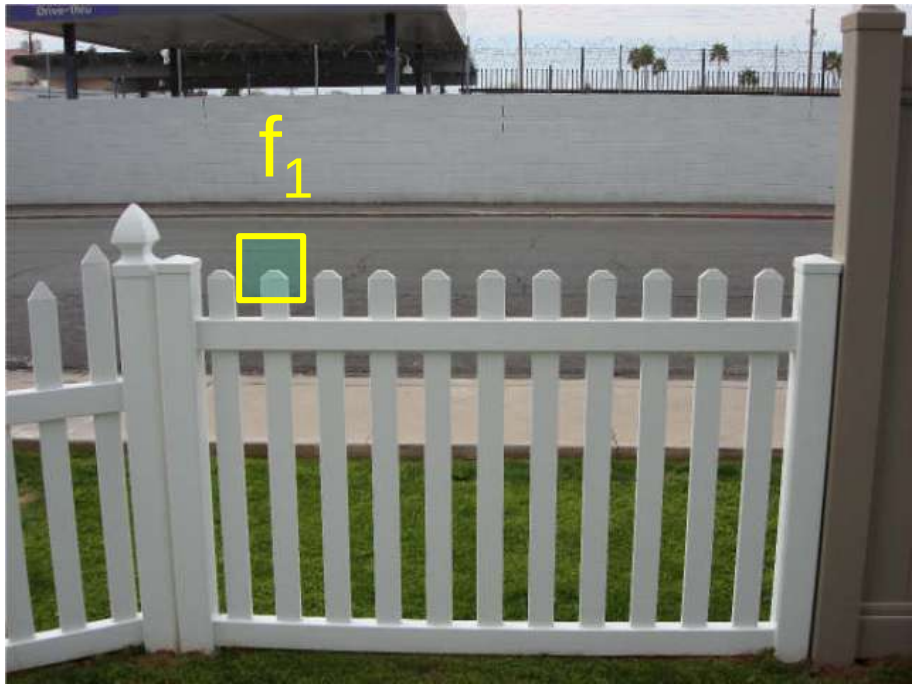
1. Define distance function that compares two descriptors
2. Test all the features in $I_2$, find the one with min distance

Courtesy Steve Seitz and Richard Szeliski

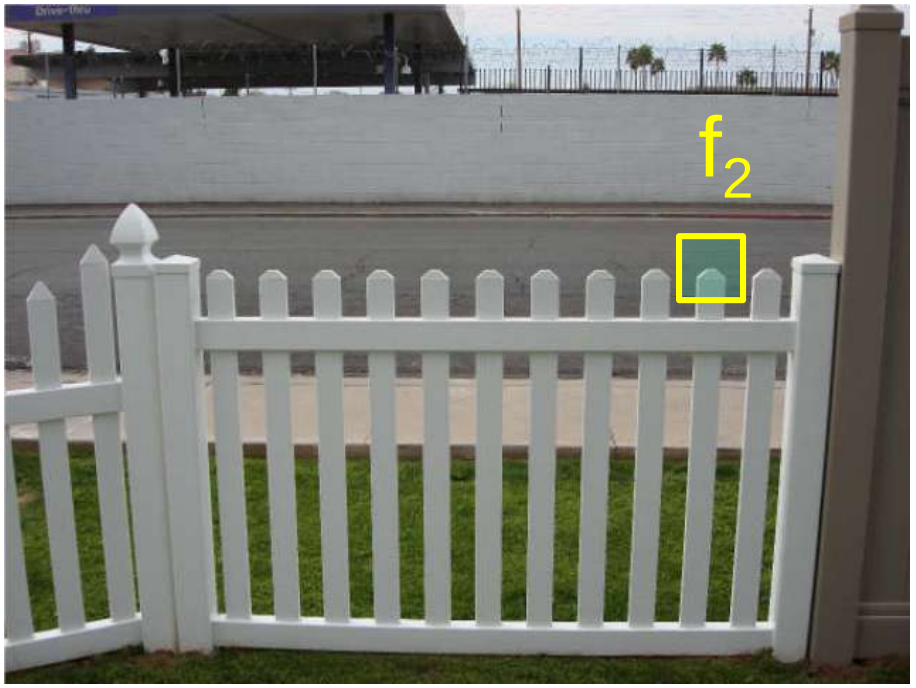
# Feature distance

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




$I_1$



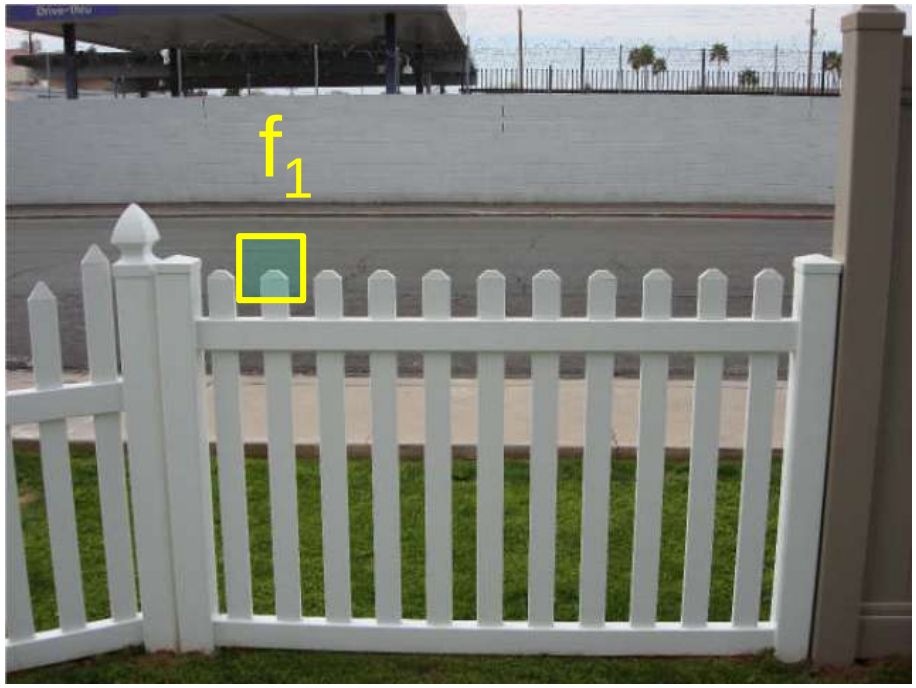

$I_2$

Courtesy Steve Seitz and Richard Szeliski

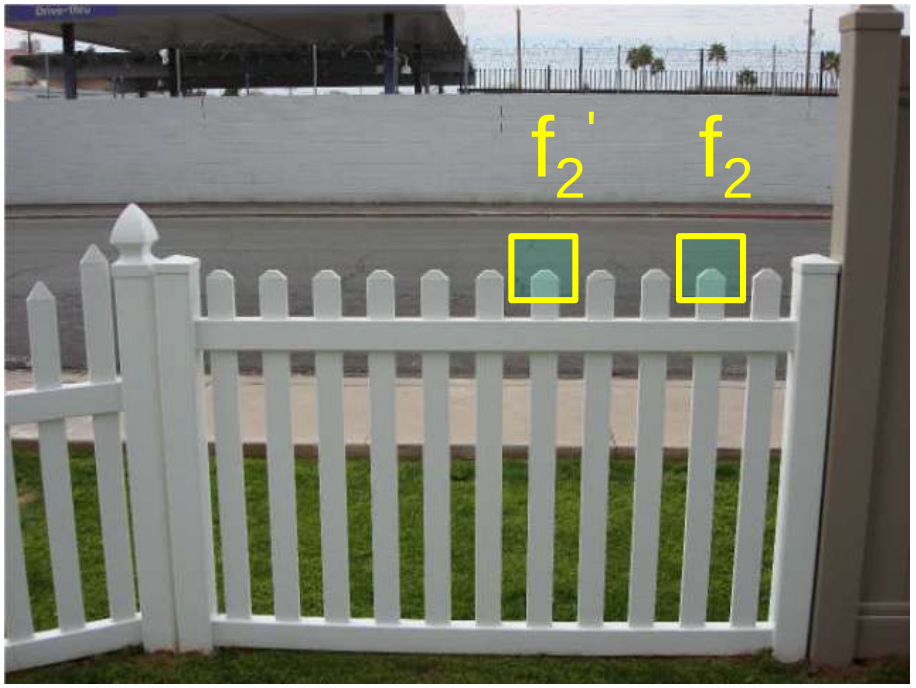
# Feature distance

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 2nd best SSD match to $f_1$ in $I_2$
  - gives small values for ambiguous matches




$I_1$

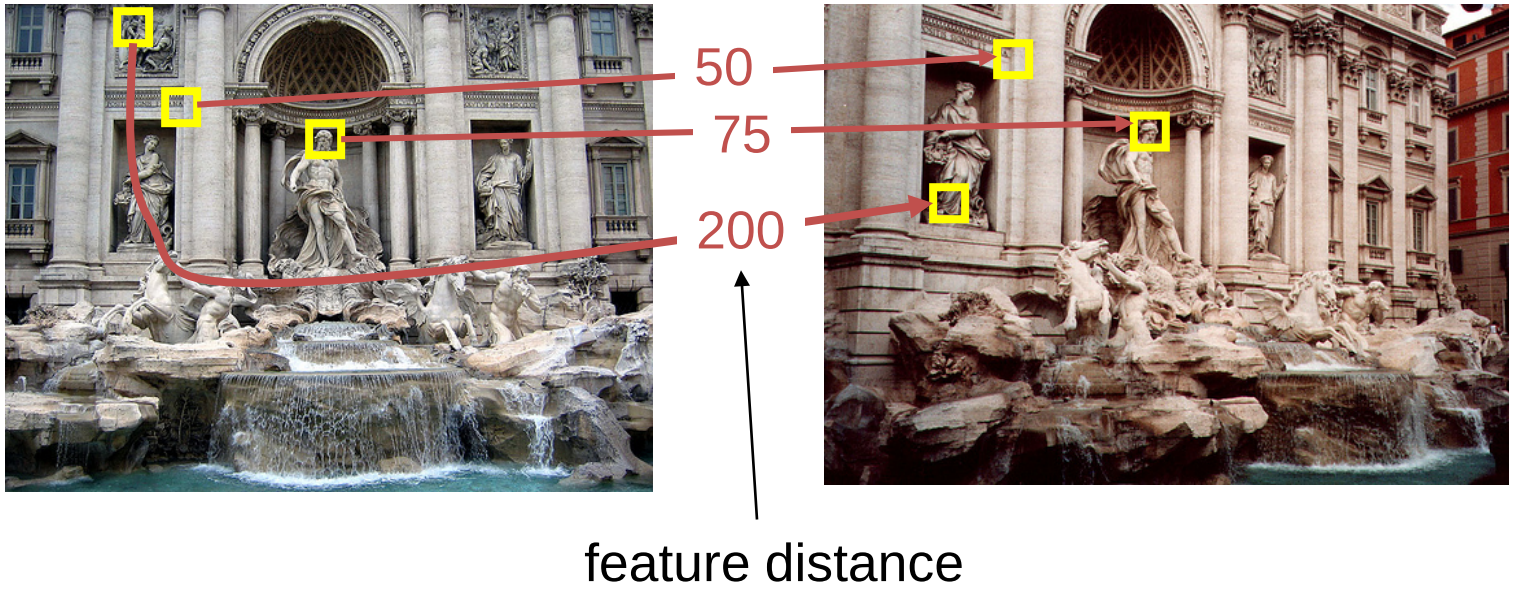



$I_2$

Courtesy Steve Seitz and Richard Szeliski

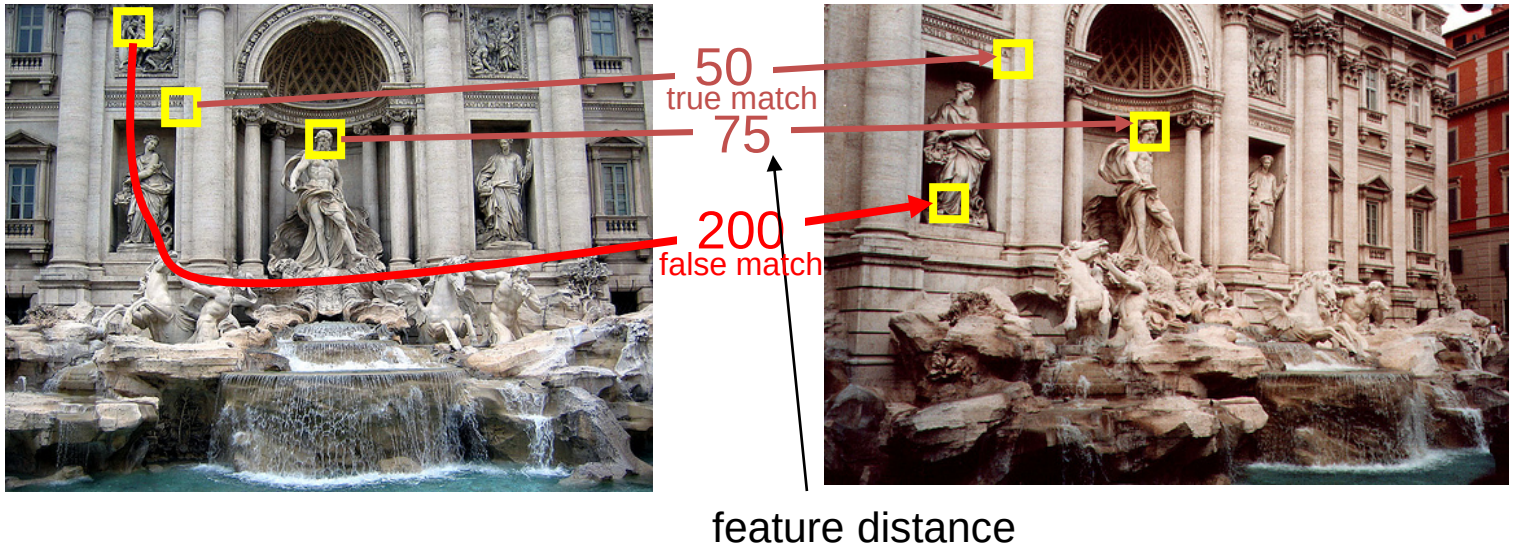
# Evaluating the results

How can we measure the performance of a feature matcher?




Courtesy Steve Seitz and Richard Szeliski

# True/false positives



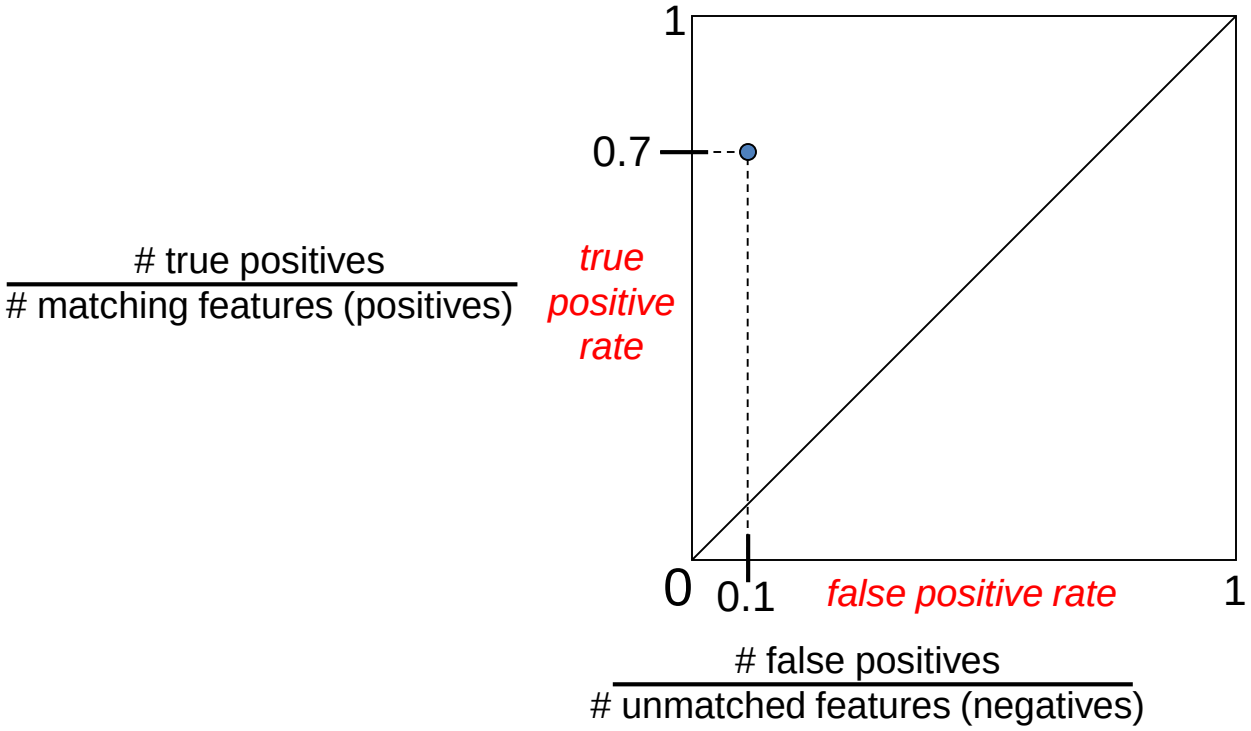

The distance threshold affects performance

- 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?

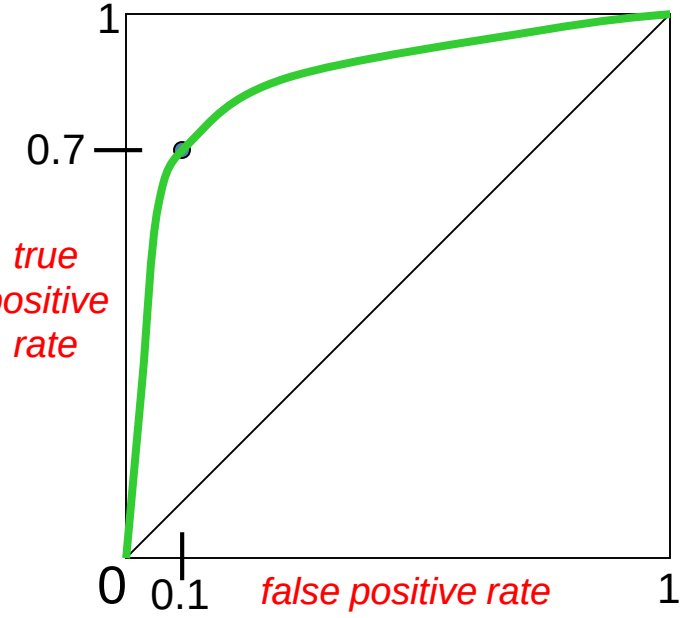



Courtesy Steve Seitz and Richard Szeliski

# Evaluating the results

How can we measure the performance of a feature matcher?

**ROC curve** **(“Receiver Operator Characteristic”)**




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

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

ROC Curves

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

Courtesy Steve Seitz and Richard Szeliski

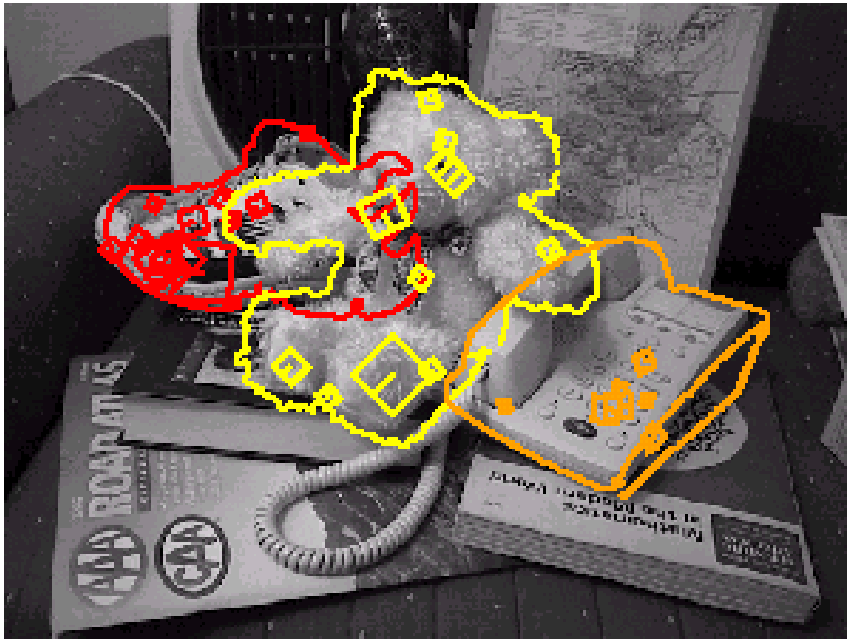
# Lots of applications

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

# Object recognition (David Lowe)



Courtesy Steve Seitz and Richard Szeliski

# Reading

- "Computer Vision: Algorithms and Applications", Richard Szeliski, Chapter 4.1
  - Download at: http://szeliski.org/Book/

# Logistics

- Lab session this week (CYT-2014)
  - ROS tutorial
  - Experimental setup
- Form project groups
  - 3 students/group
  - Let us know by sending email to elec6910p@gmail.com by this Friday
- Project 1, phase 3 will be released tomorrow (7/10)
  - Lab sessions for the weeks of 12/10 & 19/10
  - Additional testing time slots will be announced this Friday, you will need to signup for the additional time slots.