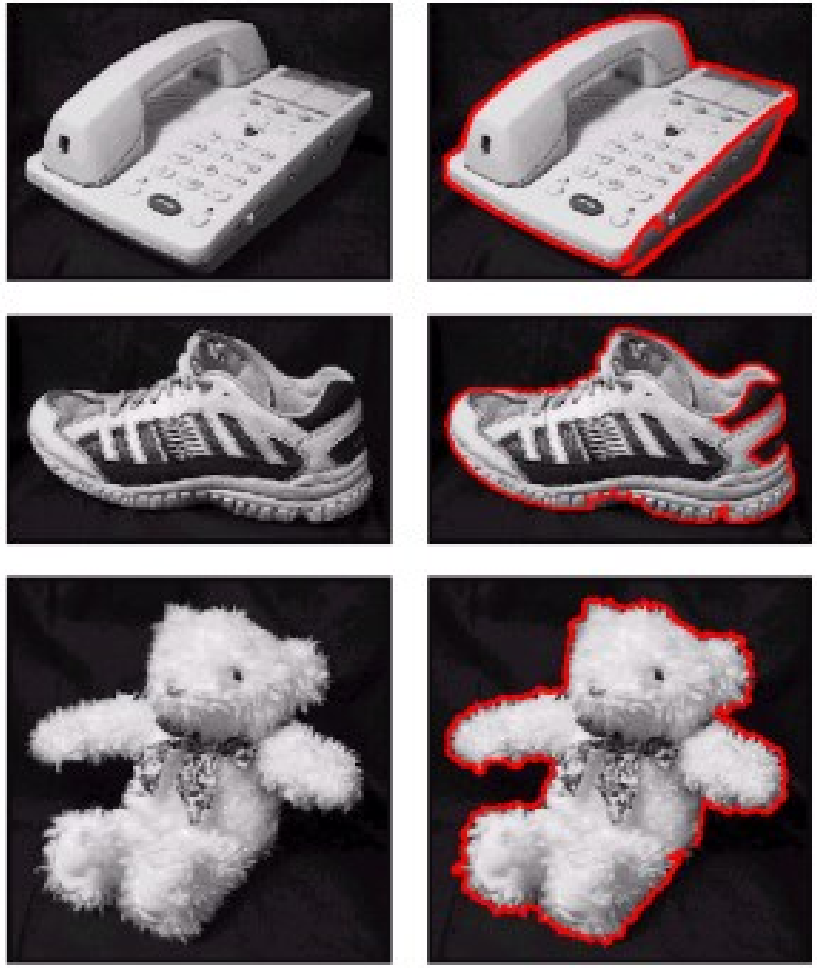
Feature Detection and Matching

Image by kirkh.deviantart.com

Object Recognition



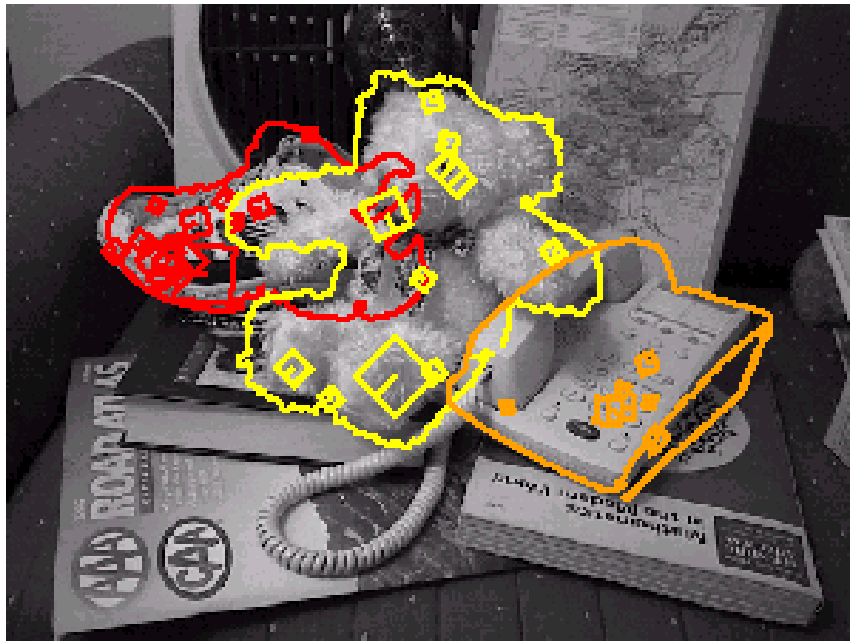


Image Matching

Introduction

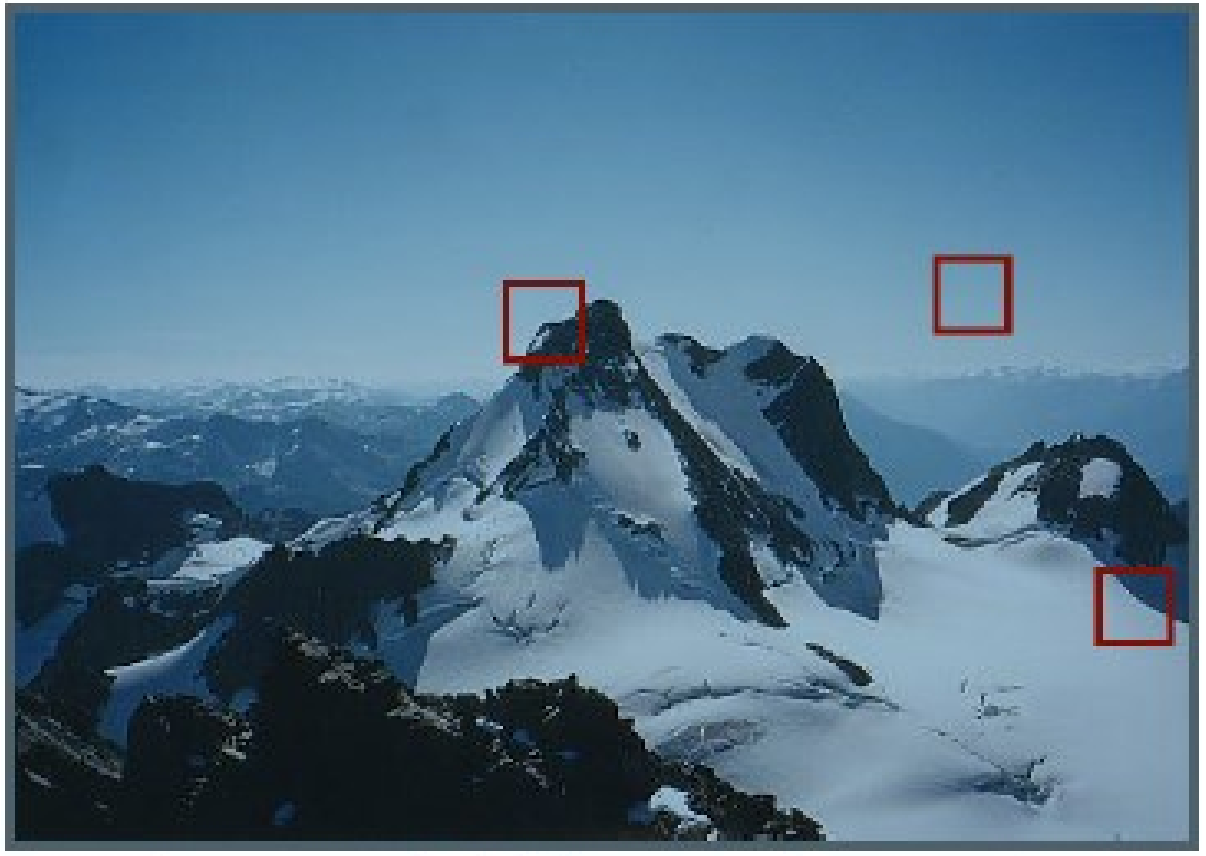
- Feature detection and matching are an essential component of many computer vision applications.
- **Specific locations** in the images, such as mountain peaks, building corners, doorways, or interestingly shaped patches of snow. These kinds of localized features are often called keypoint features or interest points (or even corners) and are often described by the appearance of pixel patches surrounding the point location
- **Edges**, e.g., the profile of mountains against the sky. These kinds of features can be matched based on their orientation and local appearance (edge profiles) and can also be good indicators of object boundaries and occlusion events in image sequences.
  - Edges can be grouped into **longer curves and contours**, which can then be tracked.
  - They can also be grouped into **straight line segments**, which can be directly matched or analyzed to find vanishing points and hence internal and external camera parameters.

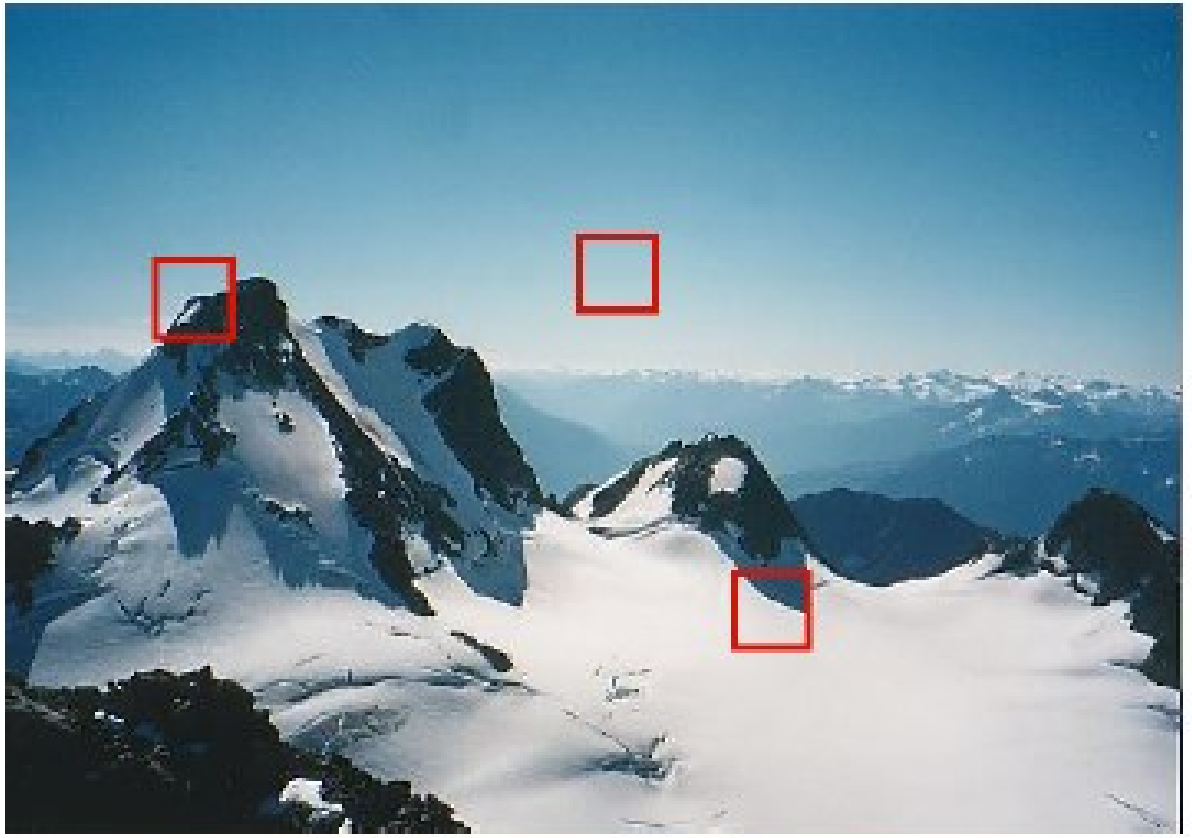
Points and patches

- Point features can be used to find a sparse set of corresponding locations in different images, often as a precursor to computing camera pose , which is a prerequisite for computing a denser set of correspondences using stereo matching.
  - Image alignment
  - Object instance recognition
- A key advantage of keypoints is that they permit matching even in the presence of clutter (occlusion) and large scale and orientation changes.

Points and patches (cont.)

- There are two main approaches to finding feature points and their correspondences.
- The first is to find features in one image that can be accurately tracked using a local search technique, such as correlation or least squares
- The second is to independently detect features in all the images under consideration and then match features based on their local appearance





Point vs. Line Features

Image Matching using Feature Points

NASA Mars Rover images

Feature Points are used for …

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

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

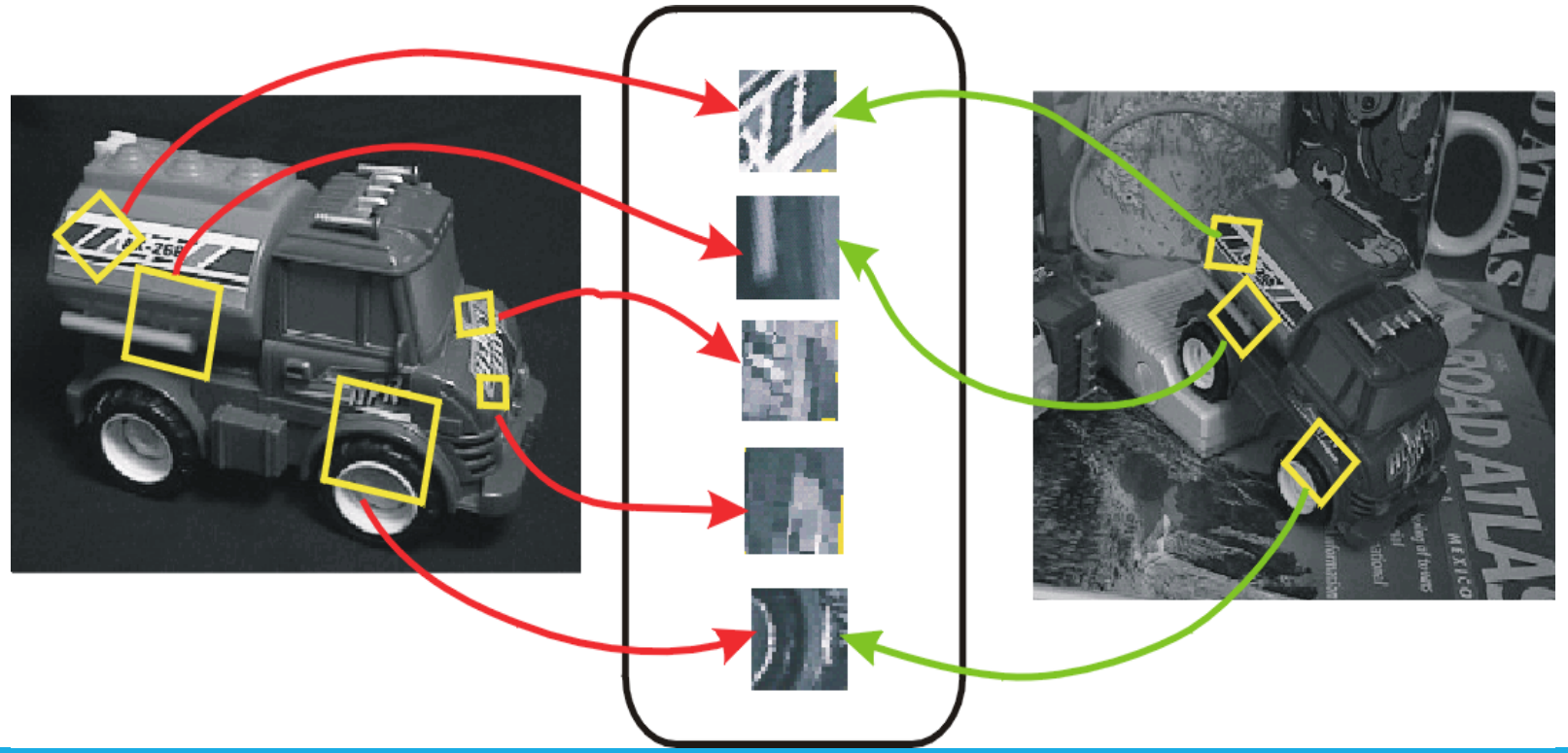
Processing Stages

- Feature **detection/extraction** (look for stable/accurate features)
- Feature **description** (aim for invariance to transformations)
- Feature **matching** (for photo collection) or **tracking** (for video)

Feature Detection

Invariant Local Features

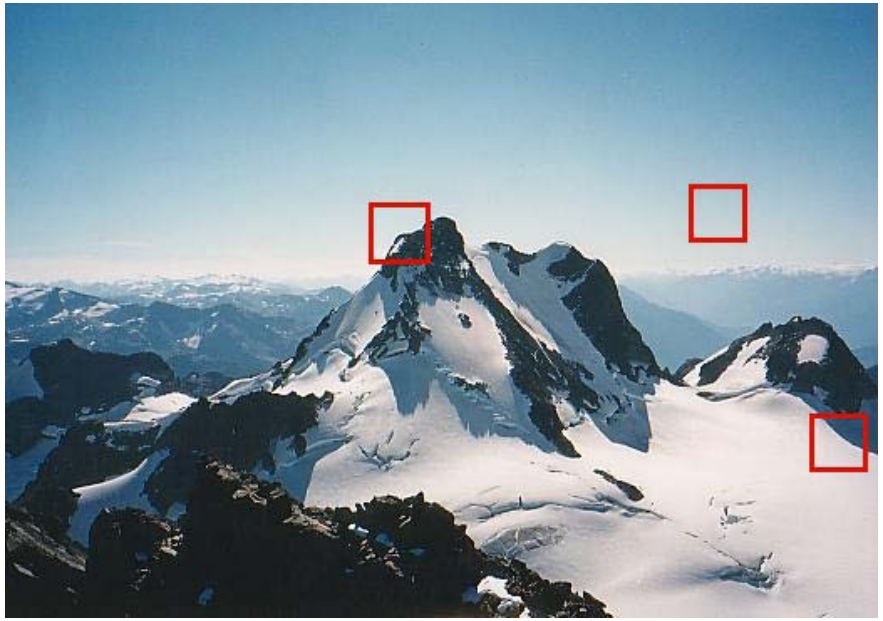
- Image content is transformed into local feature coordinates that are invariant to translation, rotation, scale, and other imaging parameters



Features Descriptors

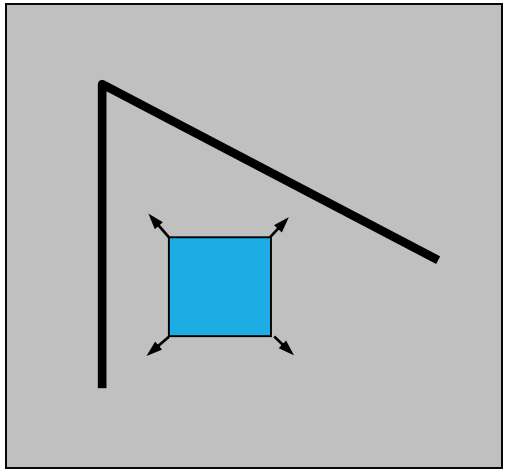
Choosing interest points

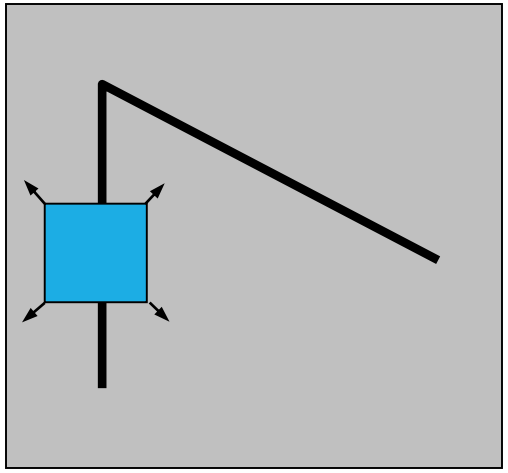
Stable (Good) Features: High Accuracy in Localization



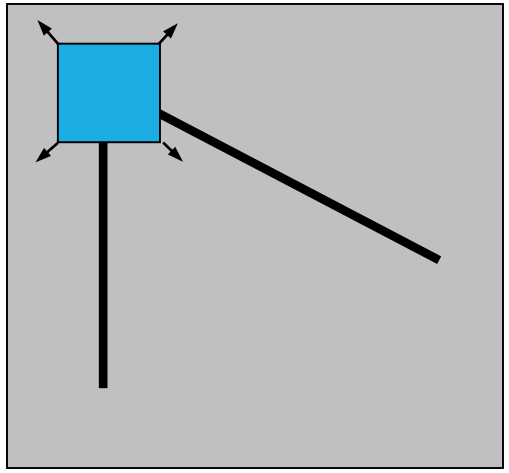
Stable (Good) Features

Aperture problem





Good feature

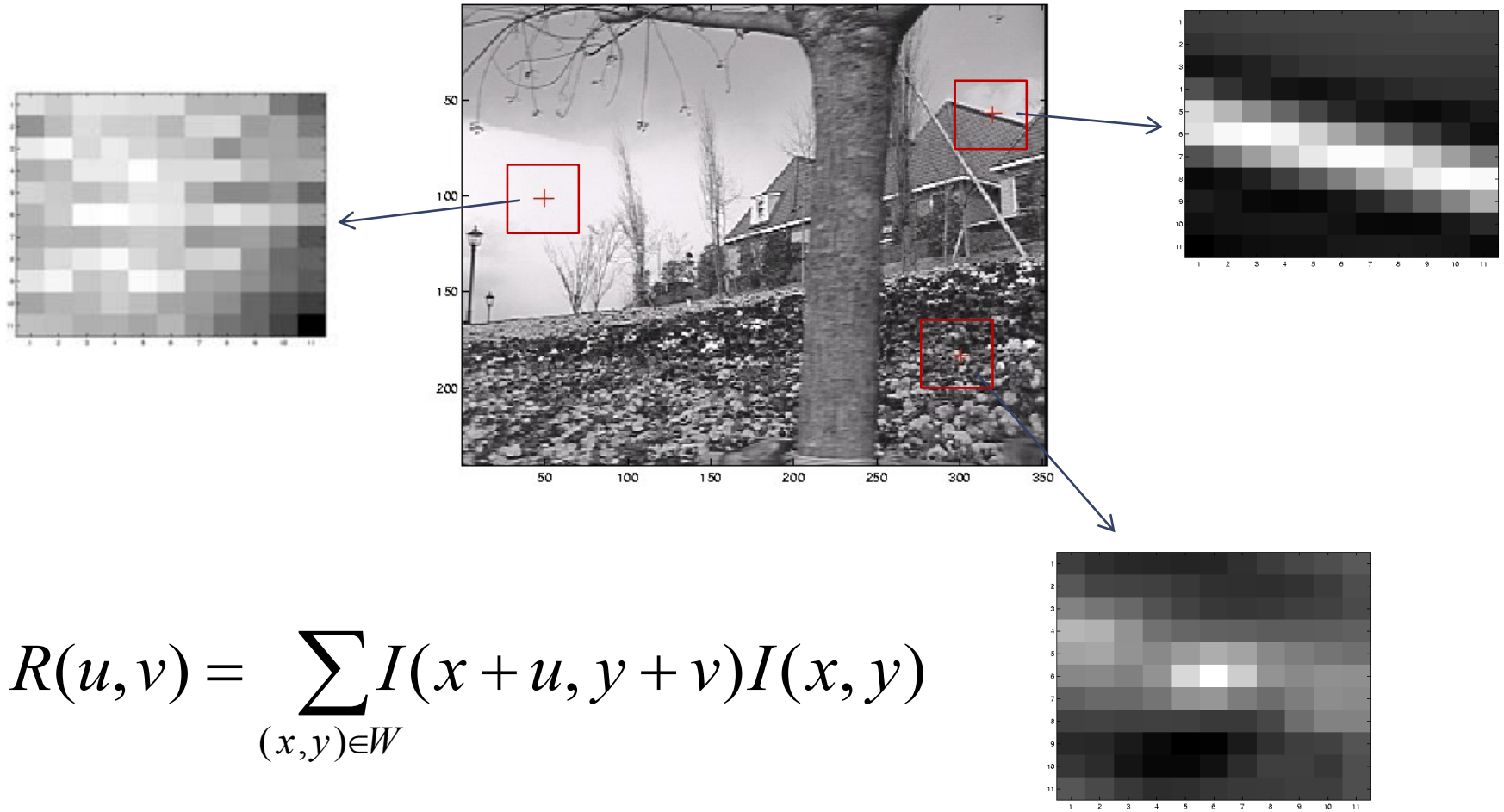


"flat" region:
no change in all directions

"edge":
no change along the edge direction

"corner":
significant change in all directions (*uniqueness*)

Autocorrelation: Indicator of “Good”ness



$$R(u,v) = \sum_{(x,y)\in W} I(x+u, y+v) I(x,y)$$

Autocorrelation Calculation

$$R(u,v) = \sum_{(x,y)\in W} I(x+u, y+v) I(x,y)$$

Autocorrelation can be approximated by sum-squared-difference (SSD):

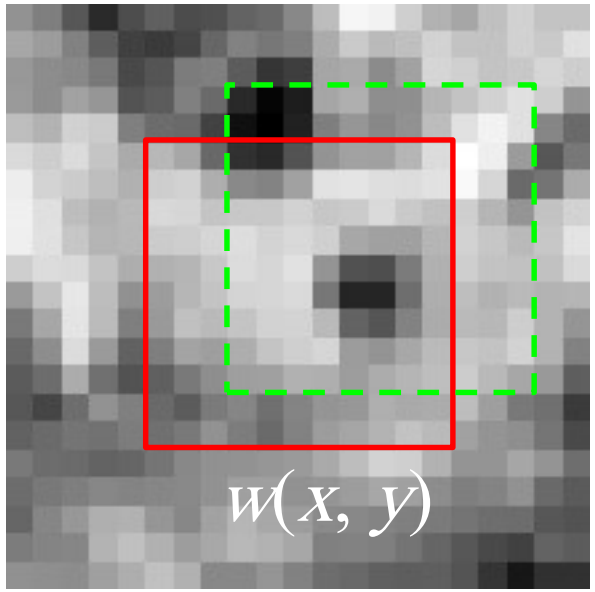
$$E(u,v) = \sum_{(x,y)\in W} [I(x+u, y+v) - I(x,y)]^2$$

Corner Detection: Mathematics

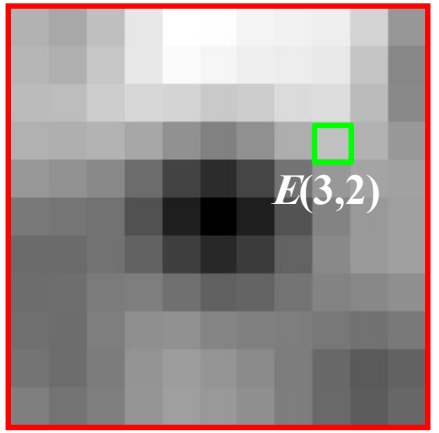
Change in appearance of window $w(x,y)$
for the shift $[u,v]$:

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

$I(x, y)$



$E(u, v)$

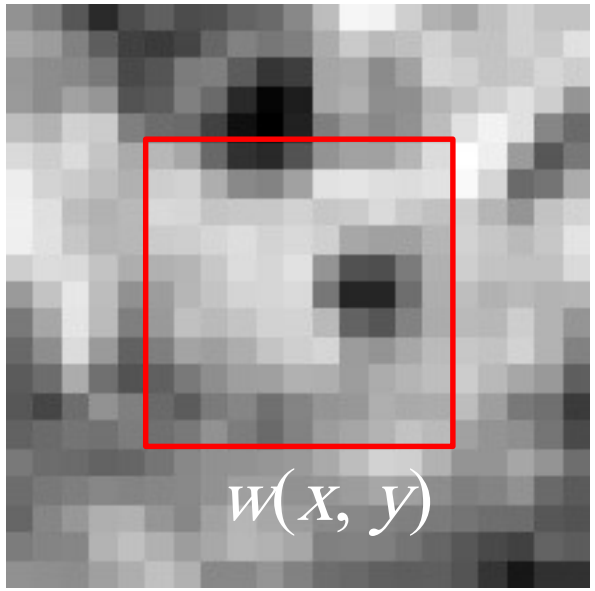


Corner Detection: Mathematics

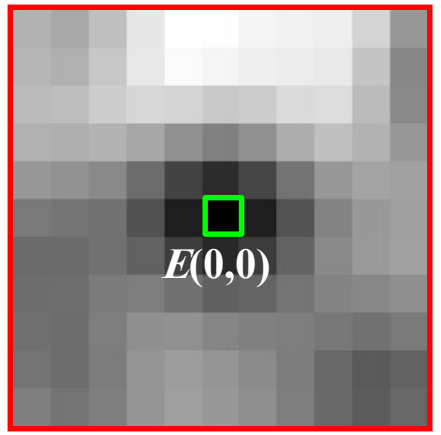
Change in appearance of window $w(x,y)$
for the shift $[u,v]$:

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

$I(x, y)$



$E(u, v)$



Corner Detection: Mathematics

Change in appearance of window $w(x,y)$ for the shift $[u,v]$:

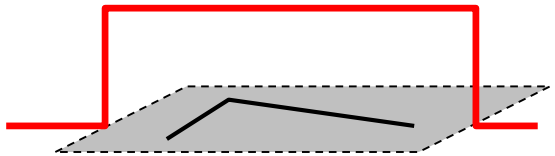
$$E(u,v) = \sum_{x,y} w(x,y)\left[I(x+u,y+v) - I(x,y)\right]^2$$

Window function

Shifted intensity

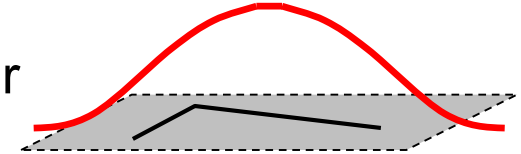
Intensity

Window function $w(x,y) =$



1 in window, 0 outside

or



Gaussian

Source: R. Szeliski

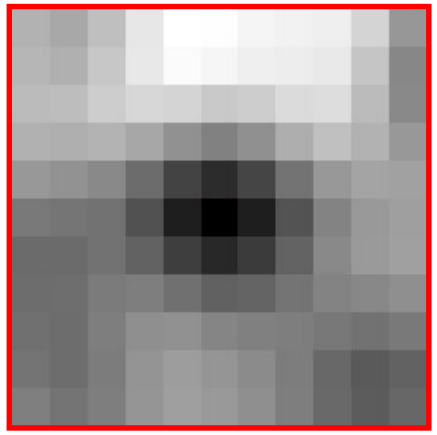
Corner Detection: Mathematics

Change in appearance of window $w(x,y)$
for the shift $[u,v]$:

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

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

$$E(u,\ v)$$



Corner Detection: Mathematics

Change in appearance of window $w(x,y)$ for the shift $[u,v]$:

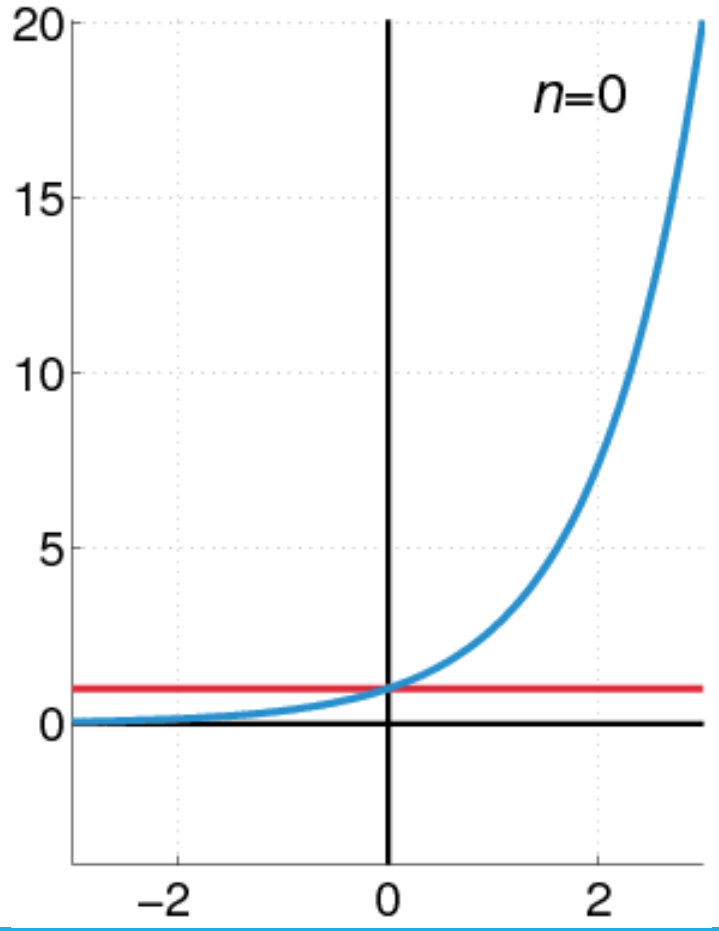
$$E(u,v) = \sum_{x,y} w(x,y)\left[I(x+u,y+v) - I(x,y)\right]^2$$

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

Local quadratic approximation of $E(u,v)$ in the neighborhood of (0,0) is given by the *second-order Taylor expansion*:

$$E(u,v) \approx E(0,0) + \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} E_u(0,0) \\ E_v(0,0) \end{bmatrix} + \frac{1}{2} \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} E_{uu}(0,0) & E_{uv}(0,0) \\ E_{uv}(0,0) & E_{vv}(0,0) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

Taylor Expansion



Corner Detection: Mathematics

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

Second-order Taylor expansion of $E(u,v)$ about (0,0):

$$E(u,v)\approx E(0,0)+\begin{bmatrix}u & v\end{bmatrix}\begin{bmatrix}E_u(0,0)\\ E_v(0,0)\end{bmatrix}+\frac{1}{2}\begin{bmatrix}u & v\end{bmatrix}\begin{bmatrix}E_{uu}(0,0) & E_{uv}(0,0)\\ E_{uv}(0,0) & E_{vv}(0,0)\end{bmatrix}\begin{bmatrix}u\\ v\end{bmatrix}$$

$$E_u(u,v)=\sum_{x,y} 2w(x,y)\left[I(x+u,y+v)-I(x,y)\right]I_x(x+u,y+v)$$

$$E_{uu}(u,v)=\sum_{x,y} 2w(x,y)I_x(x+u,y+v)I_x(x+u,y+v)$$

$$+\sum_{x,y} 2w(x,y)\left[I(x+u,y+v)-I(x,y)\right]I_{xx}(x+u,y+v)$$

$$E_{uv}(u,v)=\sum_{x,y} 2w(x,y)I_y(x+u,y+v)I_x(x+u,y+v)$$

$$+\sum_{x,y} 2w(x,y)\left[I(x+u,y+v)-I(x,y)\right]I_{xy}(x+u,y+v)$$

Corner Detection: Mathematics

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

Second-order Taylor expansion of $E(u,v)$ about (0,0):

$$E(u,v) \approx E(0,0) + \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} E_u(0,0) \\ E_v(0,0) \end{bmatrix} + \frac{1}{2} \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} E_{uu}(0,0) & E_{uv}(0,0) \\ E_{uv}(0,0) & E_{vv}(0,0) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

$$E(0,0) = 0$$

$$E_u(0,0) = 0$$

$$E_v(0,0) = 0$$

$$E_{uu}(0,0) = \sum_{x,y} 2w(x,y) I_x(x,y) I_x(x,y)$$

$$E_{vv}(0,0) = \sum_{x,y} 2w(x,y) I_y(x,y) I_y(x,y)$$

$$E_{uv}(0,0) = \sum_{x,y} 2w(x,y) I_x(x,y) I_y(x,y)$$

Corner Detection: Mathematics

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

Second-order Taylor expansion of $E(u,v)$ about (0,0):

$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} \sum_{x,y} w(x,y) I_x^2(x,y) & \sum_{x,y} w(x,y) I_x(x,y) I_y(x,y) \\ \sum_{x,y} w(x,y) I_x(x,y) I_y(x,y) & \sum_{x,y} w(x,y) I_y^2(x,y) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

$$E(0,0) = 0$$

$$E_u(0,0) = 0$$

$$E_v(0,0) = 0$$

$$E_{uu}(0,0) = \sum_{x,y} 2w(x,y) I_x(x,y) I_x(x,y)$$

$$E_{vv}(0,0) = \sum_{x,y} 2w(x,y) I_y(x,y) I_y(x,y)$$

$$E_{uv}(0,0) = \sum_{x,y} 2w(x,y) I_x(x,y) I_y(x,y)$$

Corner Detection: Mathematics

The quadratic approximation simplifies to

$$E(u,v) \approx [u \;\; v] \; M \begin{bmatrix} u \\ v \end{bmatrix}$$

where M is a *second moment matrix* computed from image derivatives:

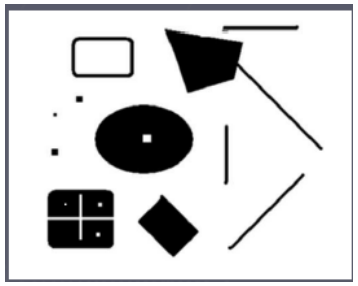
$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

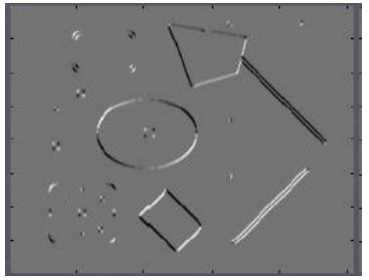
$$M = \begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} = \sum \begin{bmatrix} I_x \\ I_y \end{bmatrix} [I_x \;\; I_y] = \sum \nabla I (\nabla I)^T$$

Corners as distinctive interest points

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

2 x 2 matrix of image derivatives (averaged in neighborhood of a point).





Notation:

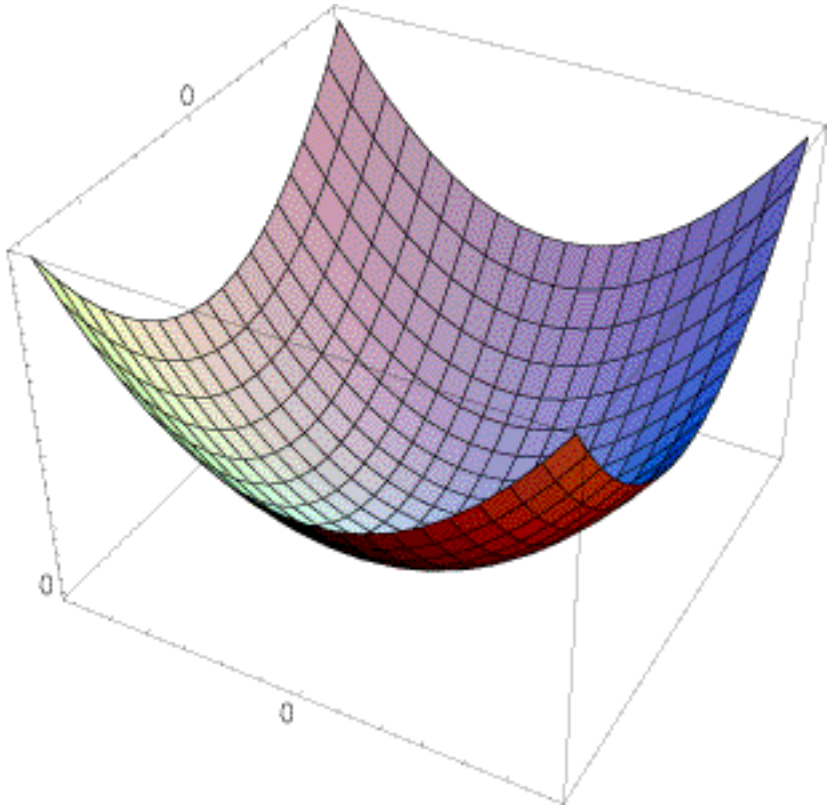
$$I_x \Leftrightarrow \frac{\partial I}{\partial x} \qquad I_y \Leftrightarrow \frac{\partial I}{\partial y} \qquad I_x I_y \Leftrightarrow \frac{\partial I}{\partial x}\frac{\partial I}{\partial y}$$

Interpreting the second moment matrix

The surface $E(u,v)$ is locally approximated by a quadratic form.

$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix}$$

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



Interpreting the second moment matrix

First, consider the axis-aligned case
(gradients are either horizontal or vertical)

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$

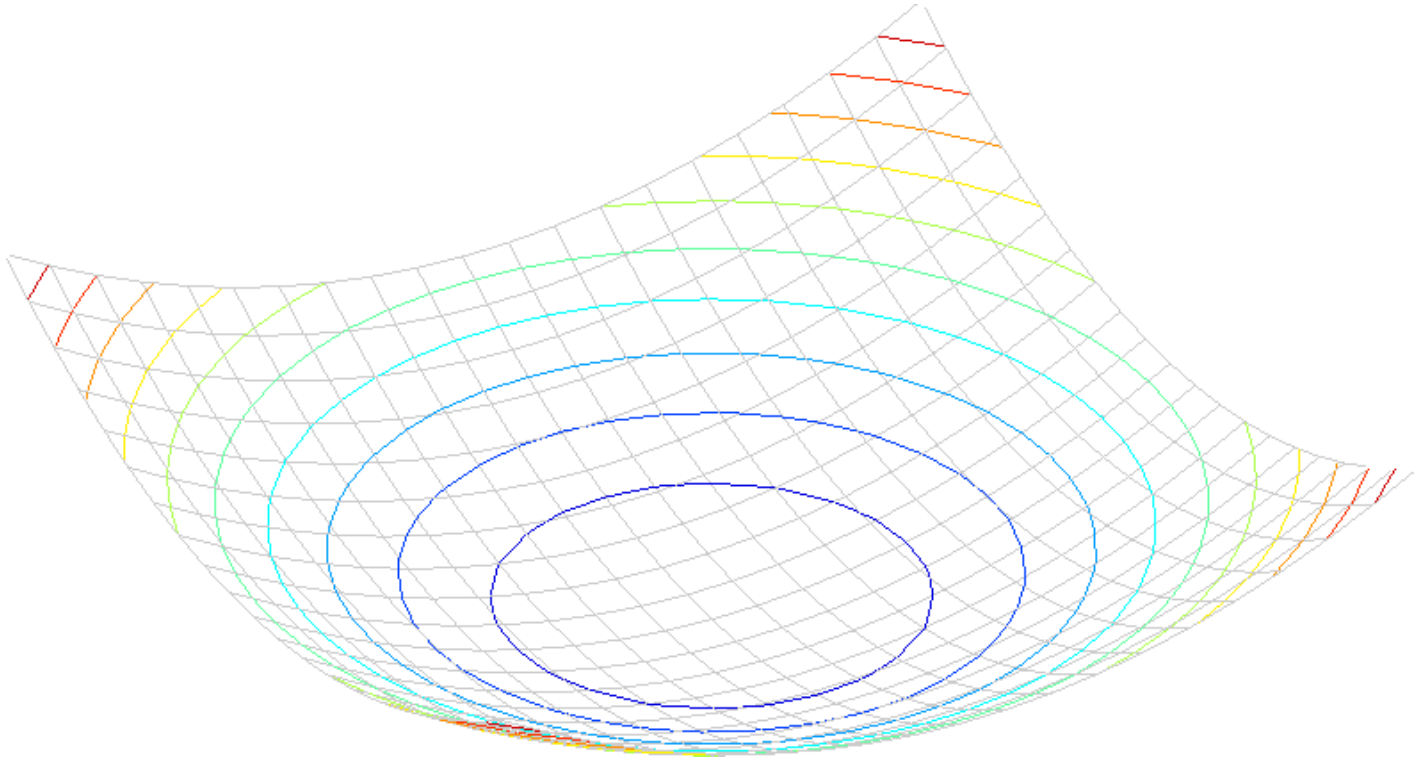
If either λ is close to 0, then this is **not** a corner, so look for locations where both are large.

Interpreting the second moment matrix

Consider a horizontal “slice” of $E(u, v)$:

This is the equation of an ellipse.

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



Interpreting the second moment matrix

Consider a horizontal "slice" of $E(u, v)$:

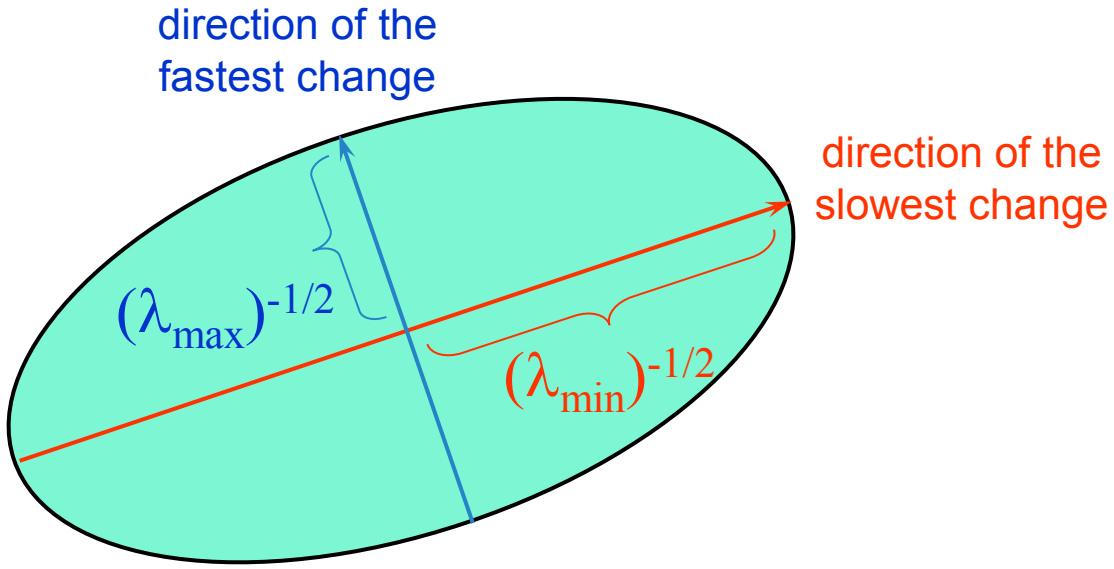
$$\begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix} = \text{const}$$

This is the equation of an ellipse.

Diagonalization of M:

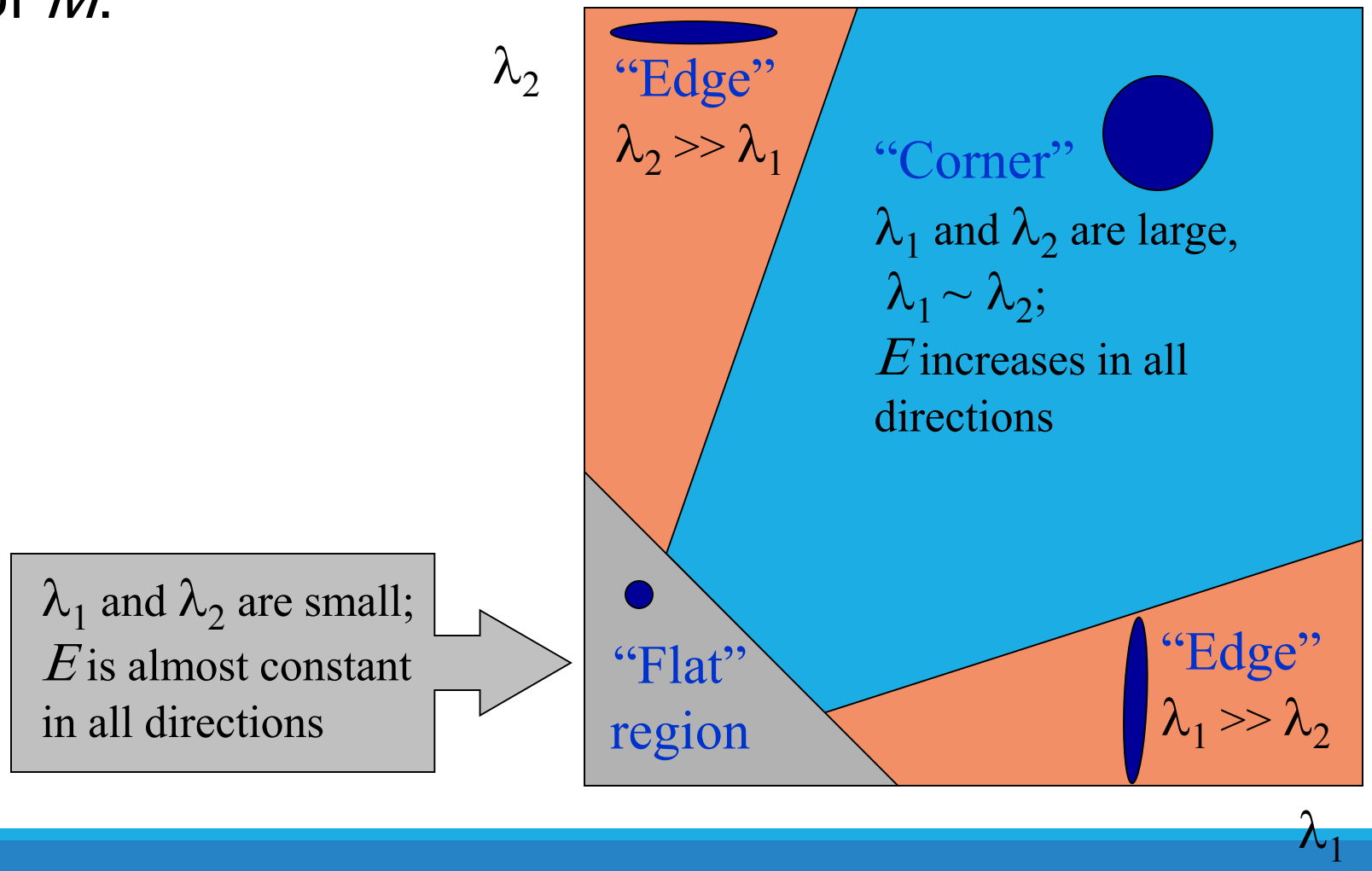
$$M = R^{-1} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} R$$

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



Interpreting the eigenvalues

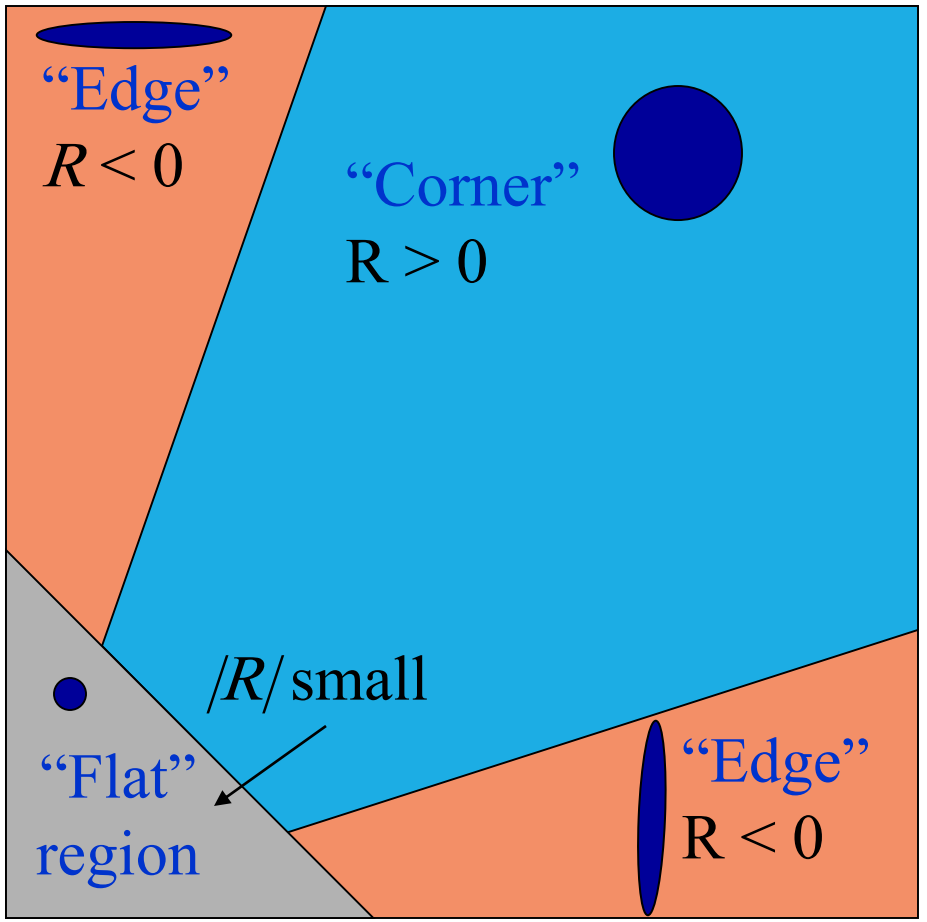
Classification of image points using eigenvalues of M:



Corner response function

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

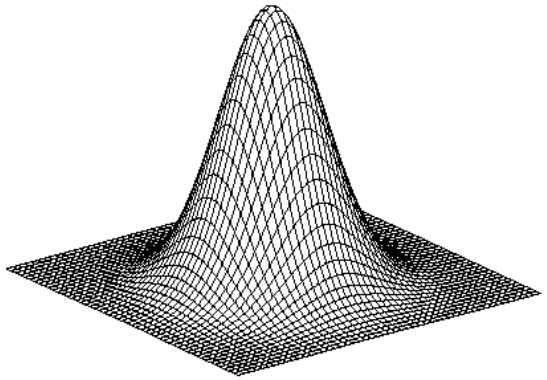
α: constant (0.04 to 0.06)



Harris corner detector

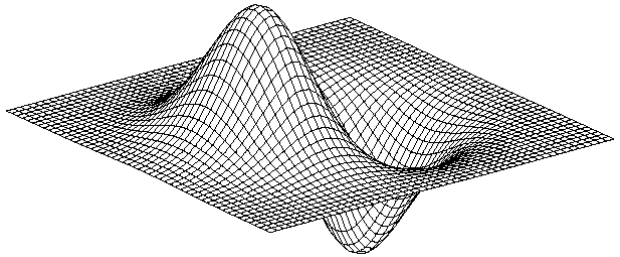
1) Compute *M* matrix for each image window to get their *cornerness* scores.
2) Find points whose surrounding window gave large corner response (*f*> threshold)
3) Take the points of local maxima, i.e., perform non-maximum suppression

Use Sobel Operator for Gradient Computation



Gaussian

$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$



derivative of Gaussian

$$\frac{\partial}{\partial x} h_\sigma(u,v)$$

-1	0	1
-2	0	2
-1	0	1

Horizontal derivative

1	2	1
0	0	0
-1	-2	-1

Vertical derivative

Eigenvalues and Eigenvectors of the Auto-correlation Matrix

lower limit

upper limit

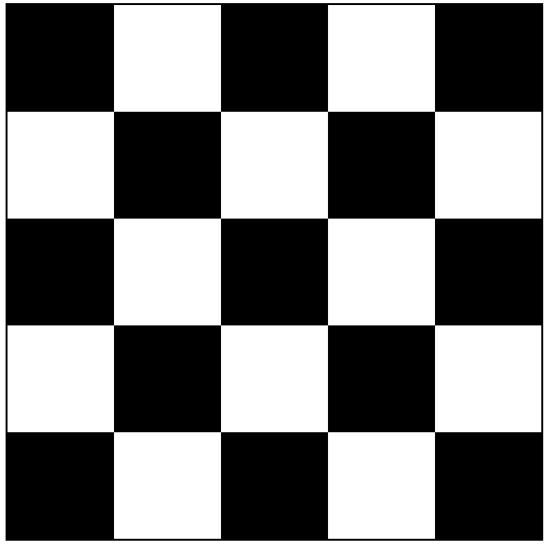
$$\lambda_{-} \le E(u,v) = u^T M u \le \lambda_{+}$$

where λ_{+} and λ_{-} are the two eigenvalues of M.

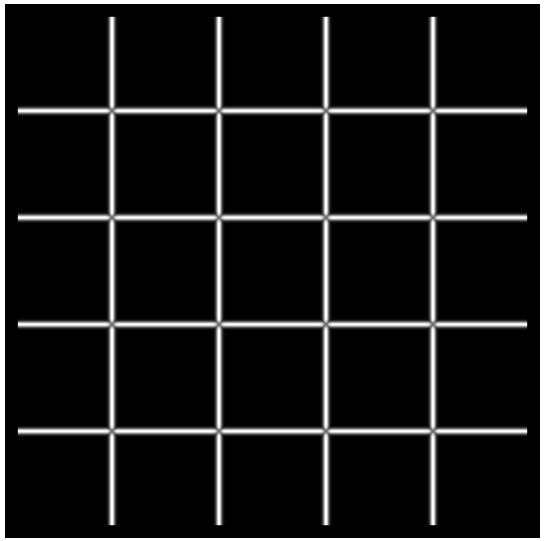
The eigenvector e_{+} corresponding to λ_{+} gives the direction of **largest** increase in ***E***,

while the eigenvector e_{-} corresponding to λ_{-} gives the direction of **smallest** increase in ***E***.

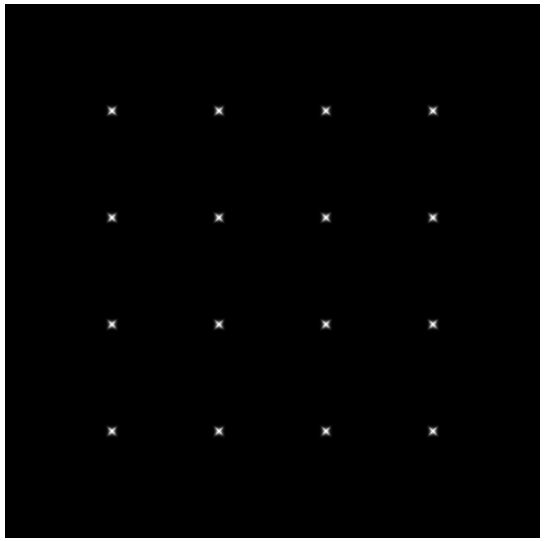
How are λ_+, e_+, λ_-, and e_+ relevant for feature detection?



I



λ_+



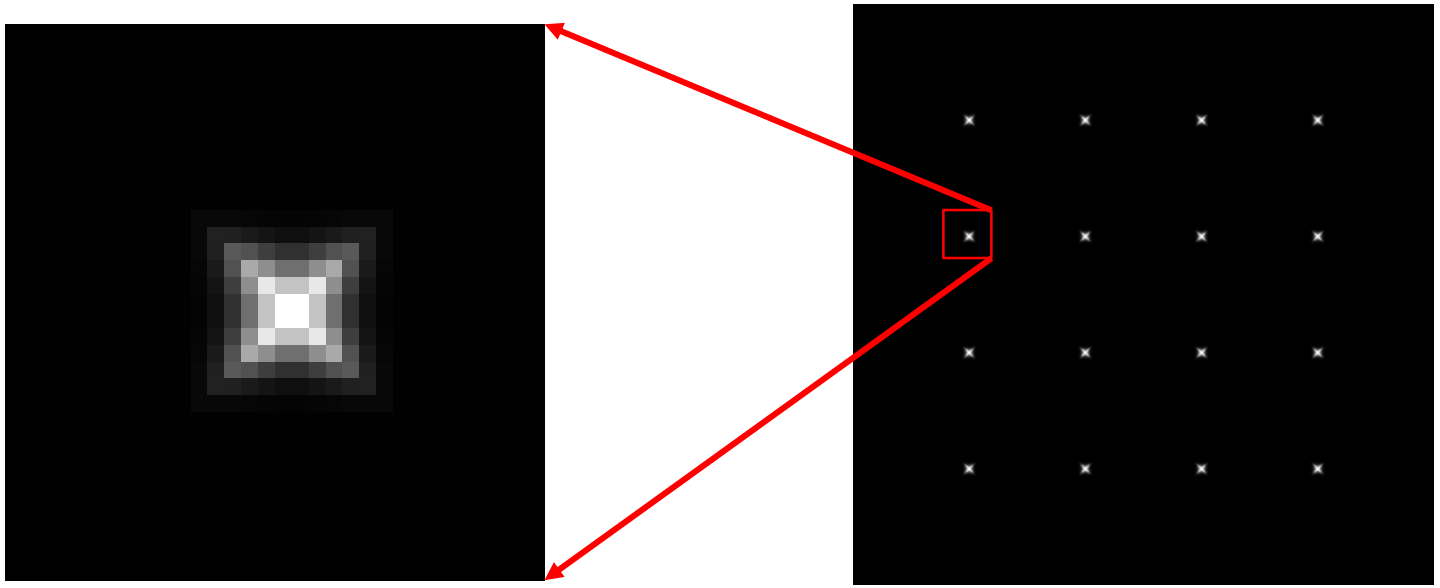
λ_-

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

Thus, the *minimum* of *E(u,v)* should be large, this minimum is given by the smaller eigenvalue λ_- of *M*

Feature Detection Algorithm

- Compute the gradient at each point in the image
- Create the *M* matrix for each point from the gradients in a window
- Compute the eigenvalues of each *M*
- Find points with large response (λ_- > threshold)
- Choose those points as features where λ_- is a local maximum



λ_-

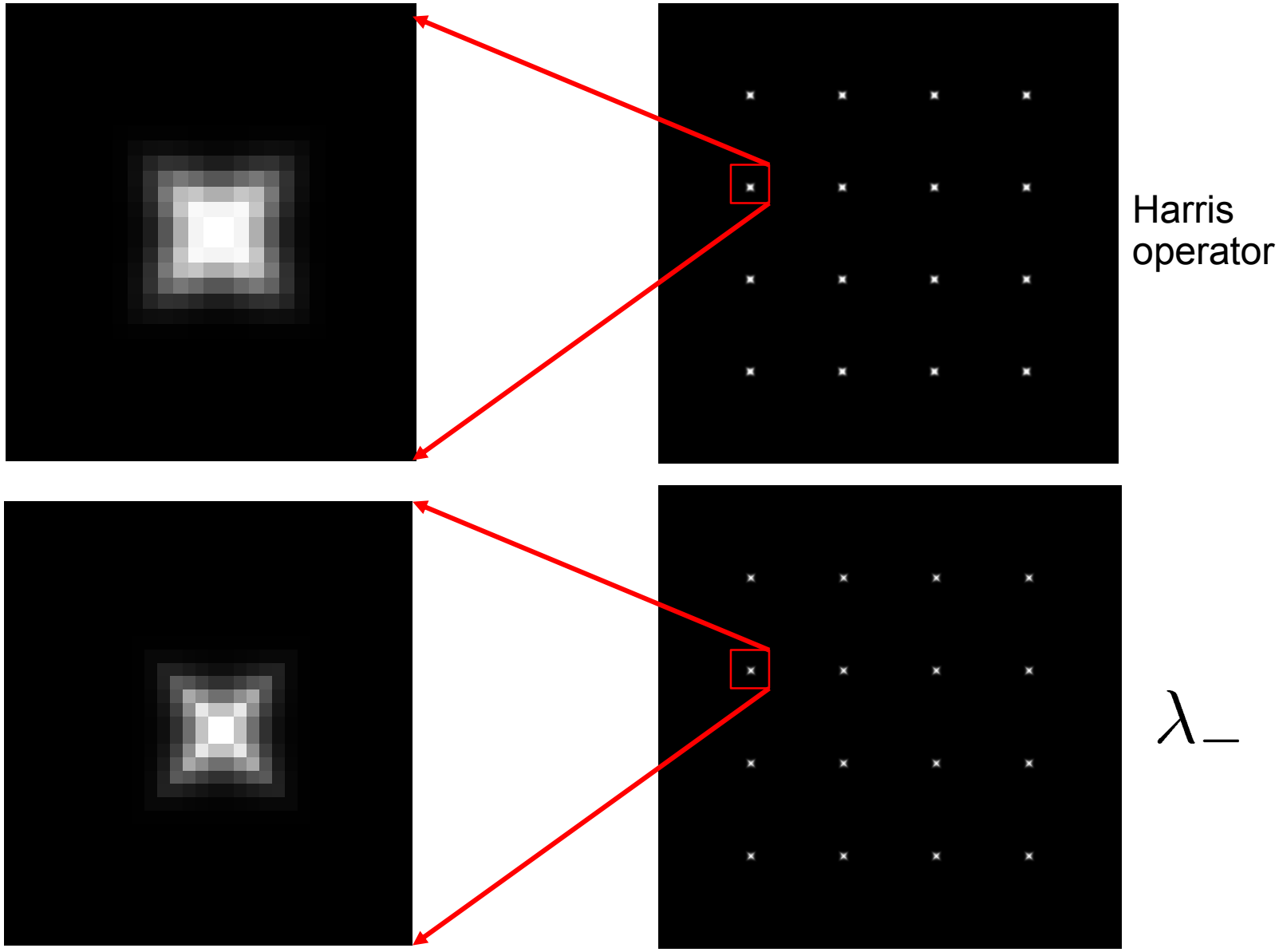
The Harris Operator

$$f = \frac{\lambda_- \lambda_+}{\lambda_- + \lambda_+} = \frac{\det(A)}{\mathrm{tr}(A)} = \frac{a_{11}a_{22} - a_{12}a_{21}}{a_{11} + a_{22}}$$

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$$

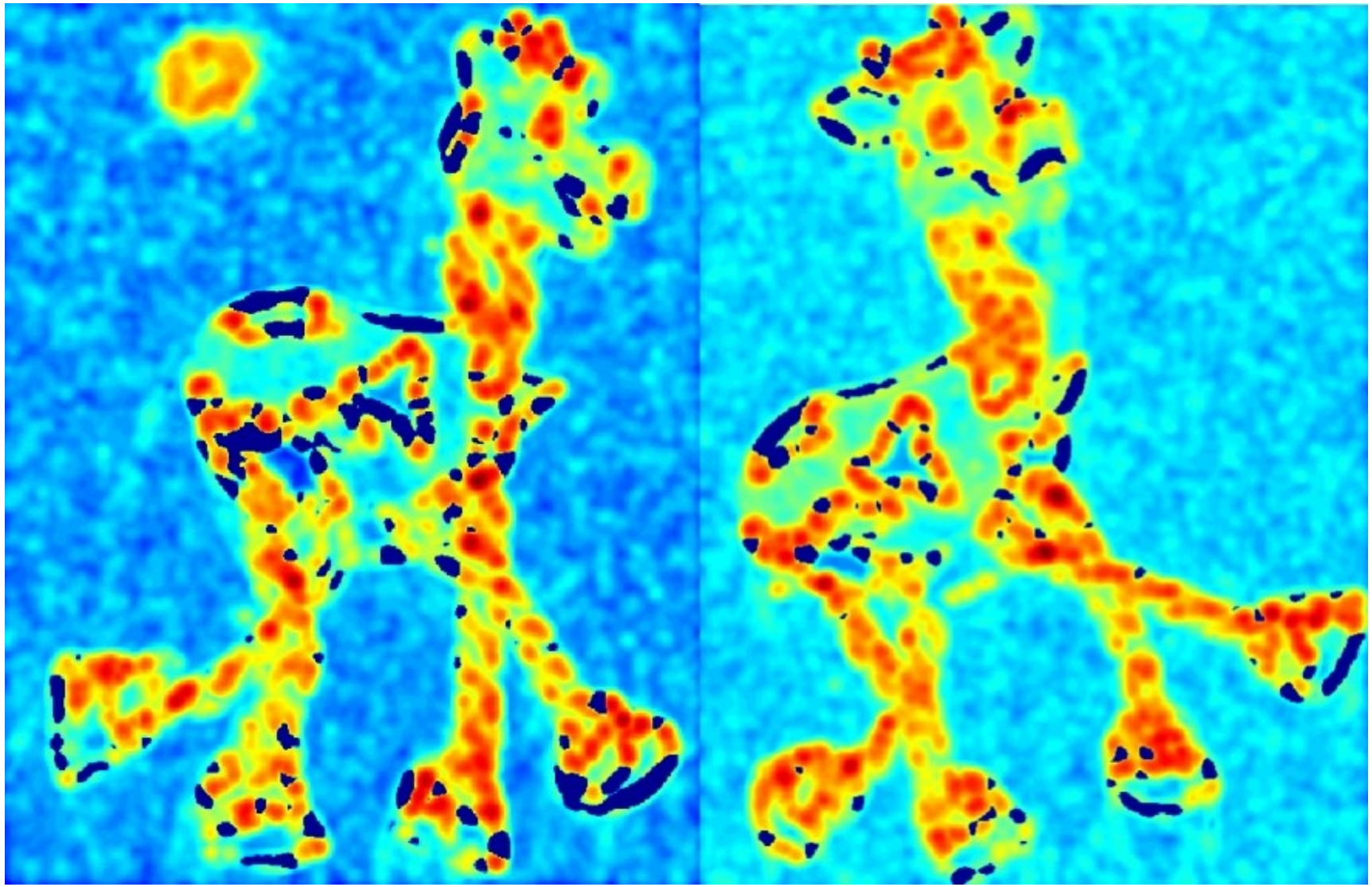
- Called the “Harris Corner Detector” or “Harris Operator”
- Very similar to λ_- but less expensive (no square root)
- Lots of other detectors, this is one of the most popular

The Harris Operator

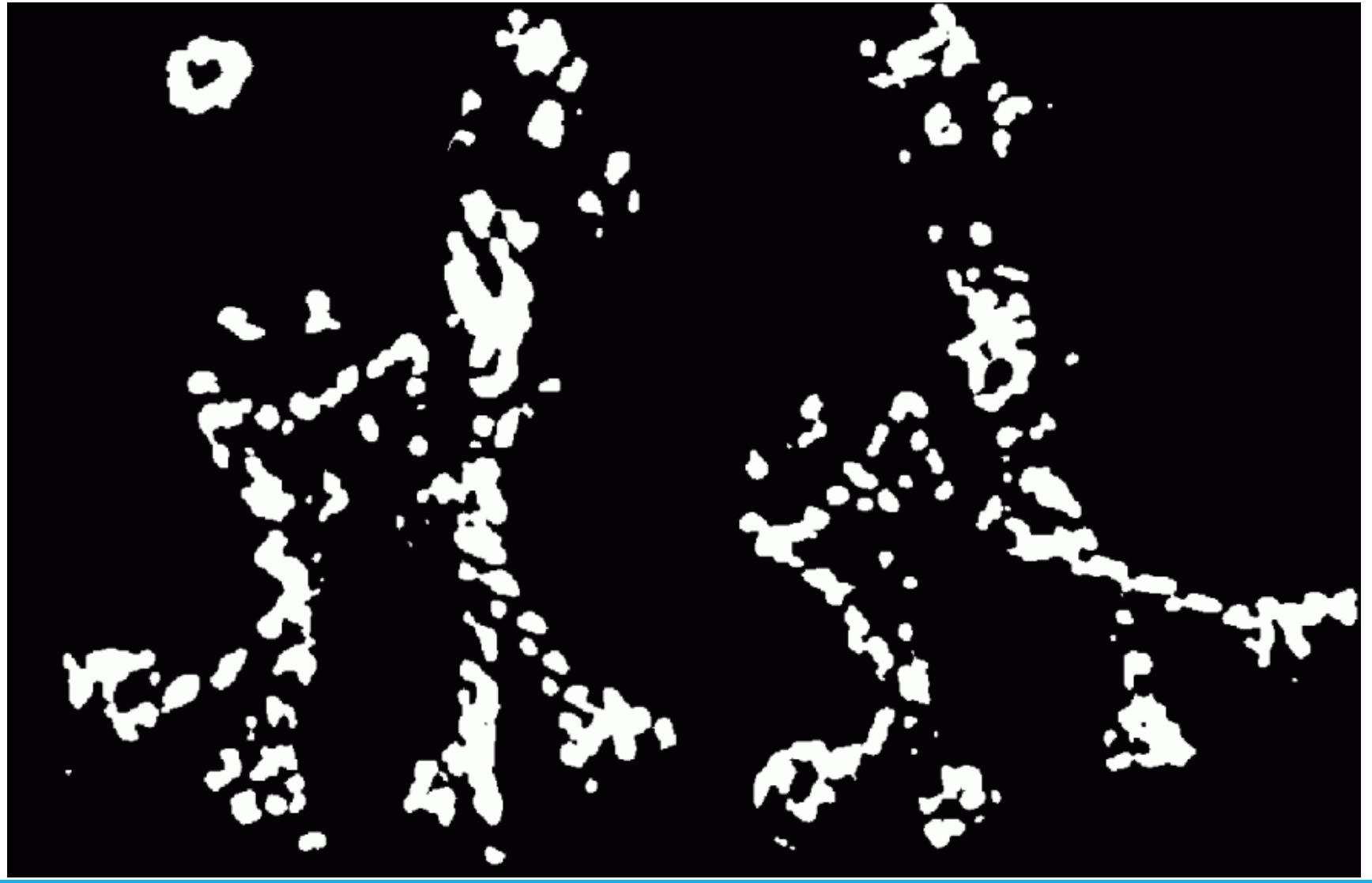


Harris Detector Example

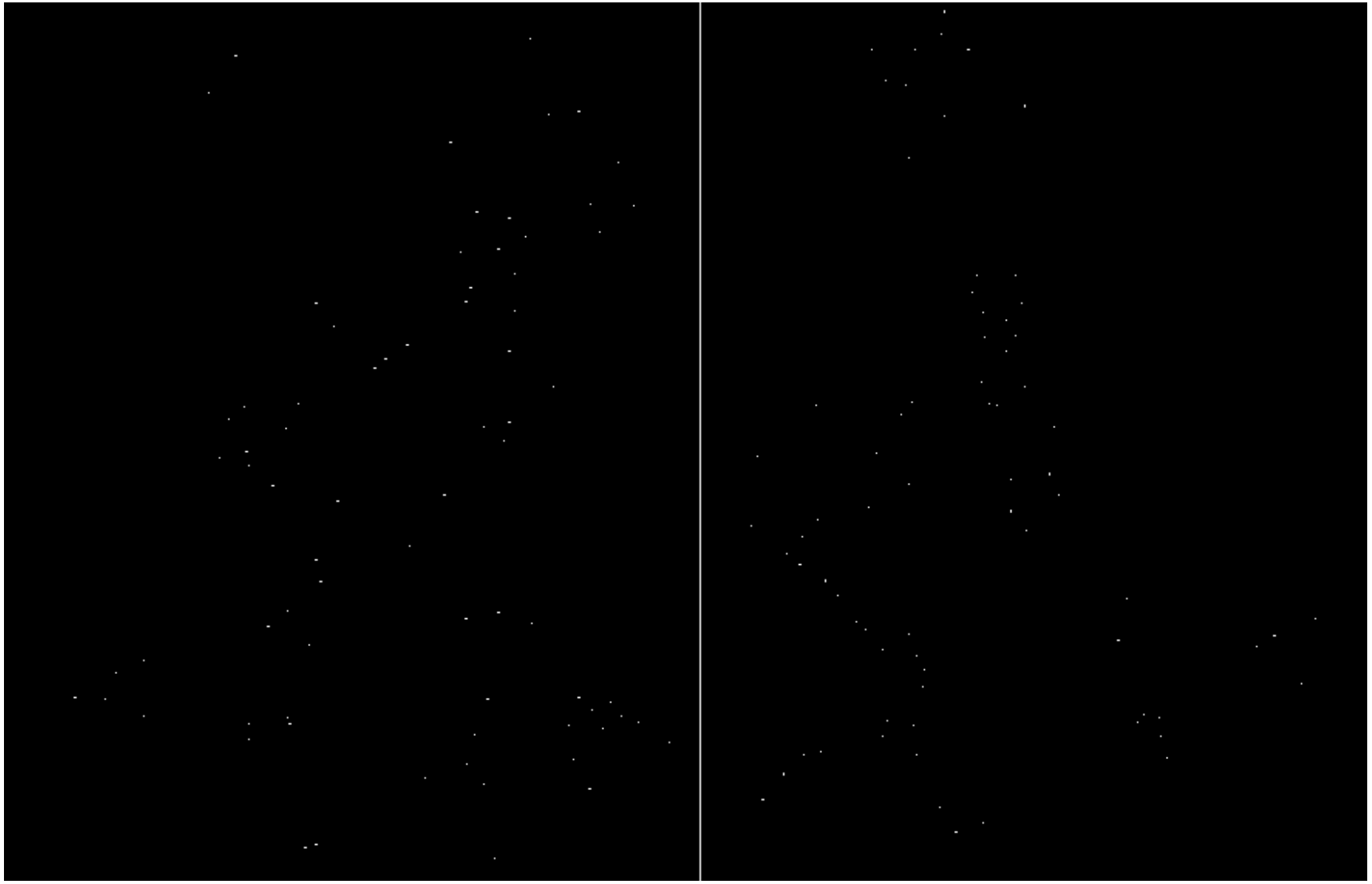
f value (red = high, blue = low)



Threshold (f > value)



Find Local Maxima of f

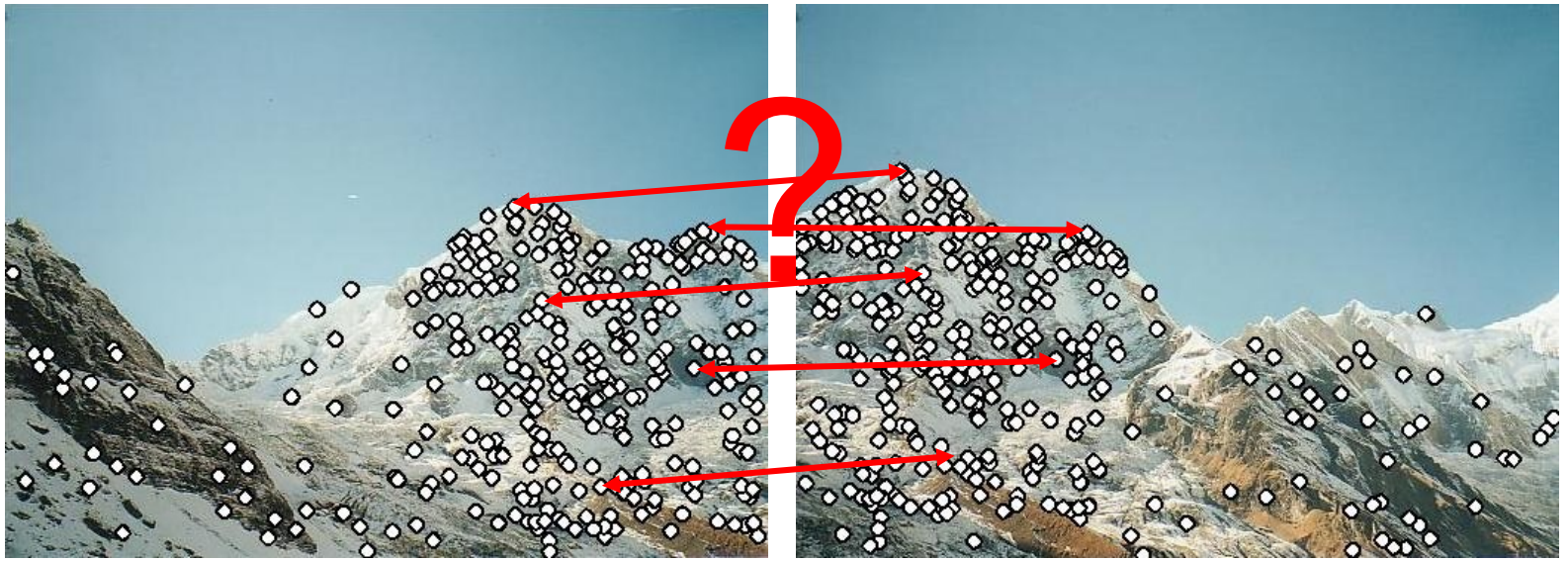


Harris Features (in red)

Feature Description

Feature Descriptor

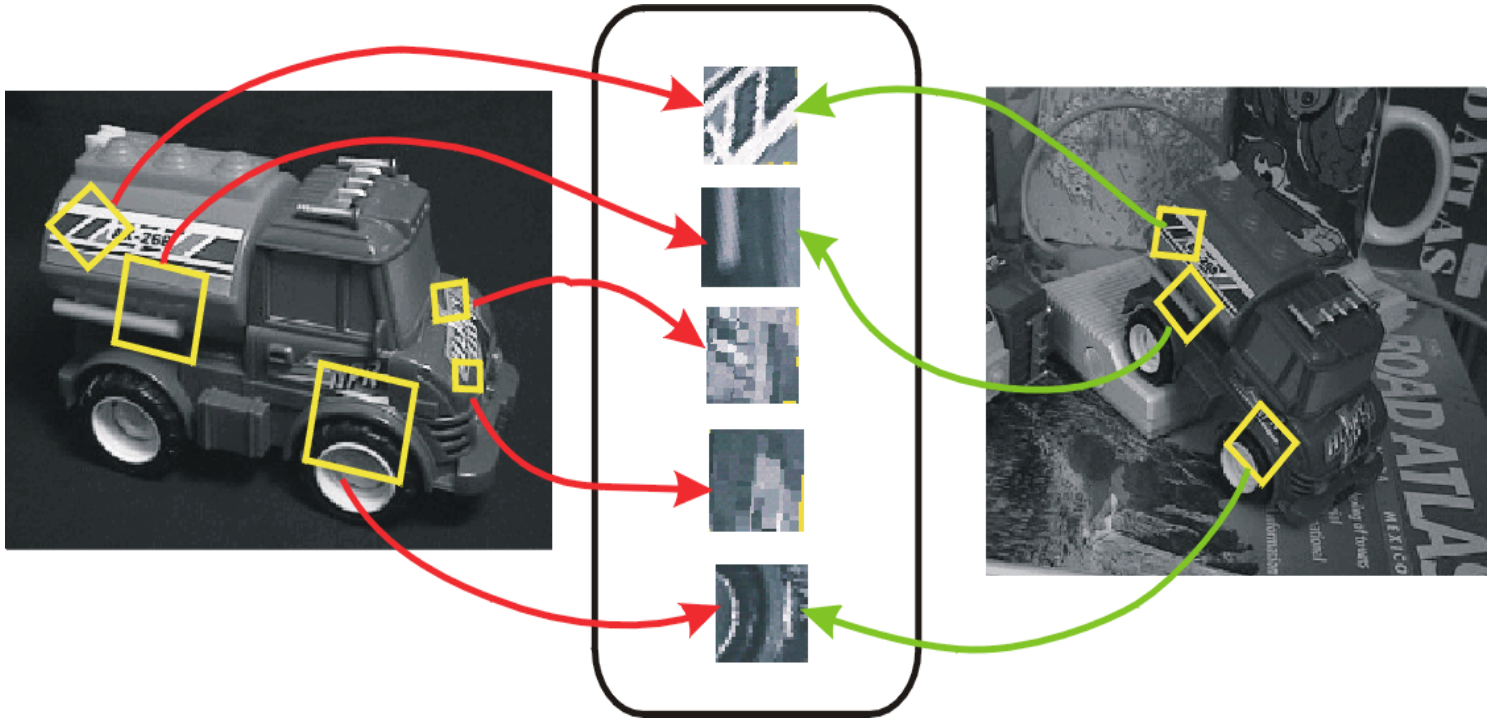
- How can we extract feature **descriptors** that are invariant to variations (transformations) and yet still discriminative enough to establish correct correspondences?



Transformation Invariance

Find features that are invariant to transformations

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



Transformation Invariance

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

Types of invariance

- Illumination

Types of invariance (cont.)

- Scale

Types of invariance (cont.)

- Rotation

Types of invariance (cont.)

- Affine

Types of invariance (cont.)

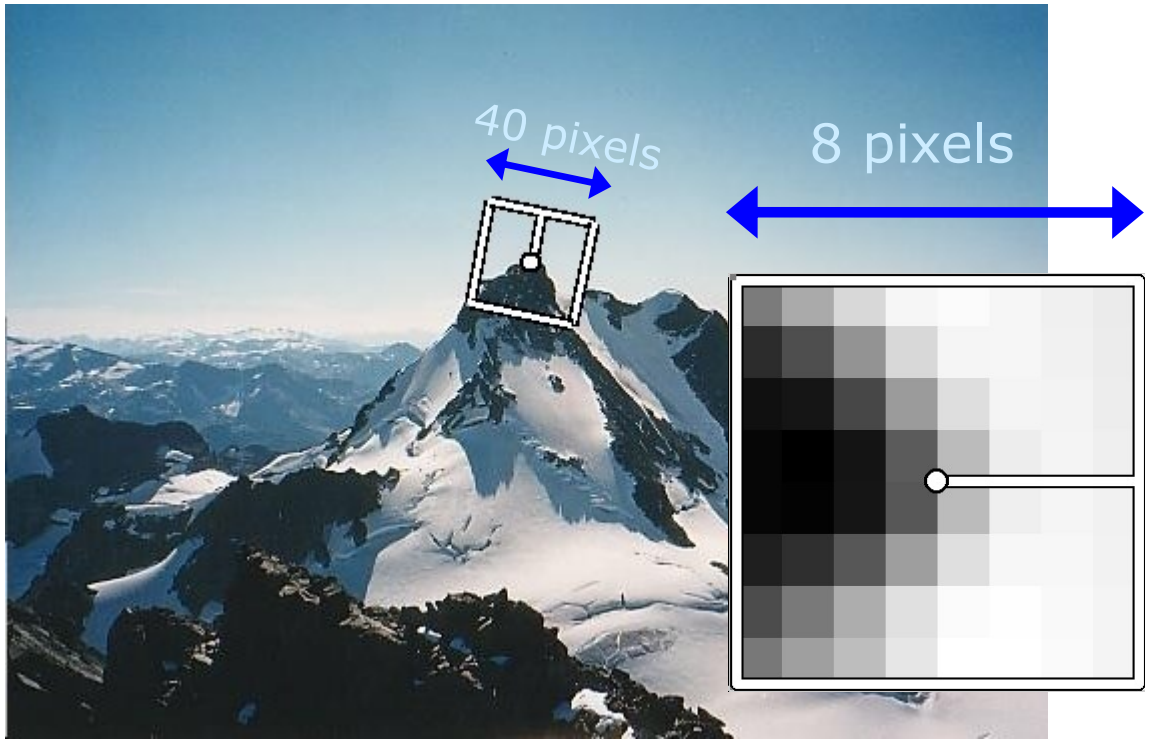
- Full Perspective

How to Achieve Invariance

1. Make sure the **detector** is invariant
   - Harris is invariant to translation and rotation
   - Scale is trickier
     - A 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?

Rotational Invariance for Feature Descriptors

- Find dominant orientation of the image patch. This is given by $\mathbf{e}_+$, the eigenvector of $\mathbf{M}$ corresponding to λ_+
- Rotate the patch to horizontal direction
- Intensity normalize the window by subtracting the mean, dividing by the standard deviation in the window



Scale Invariance

- Detecting features at the finest stable scale possible may not always be appropriate
  - e.g., when matching images with little high frequency detail (e.g., clouds), fine-scale features may not exist.
- One solution to the problem is to extract features at a variety of scales
  - e.g., by performing the same operations at multiple resolutions in a pyramid and then matching features at the same level.
- This kind of approach is suitable when the images being matched do not undergo large scale changes
  - e.g., when matching successive aerial images taken from an airplane or
  - stitching panoramas taken with a fixed-focal-length camera.

Scale Invariance for Feature Descriptors

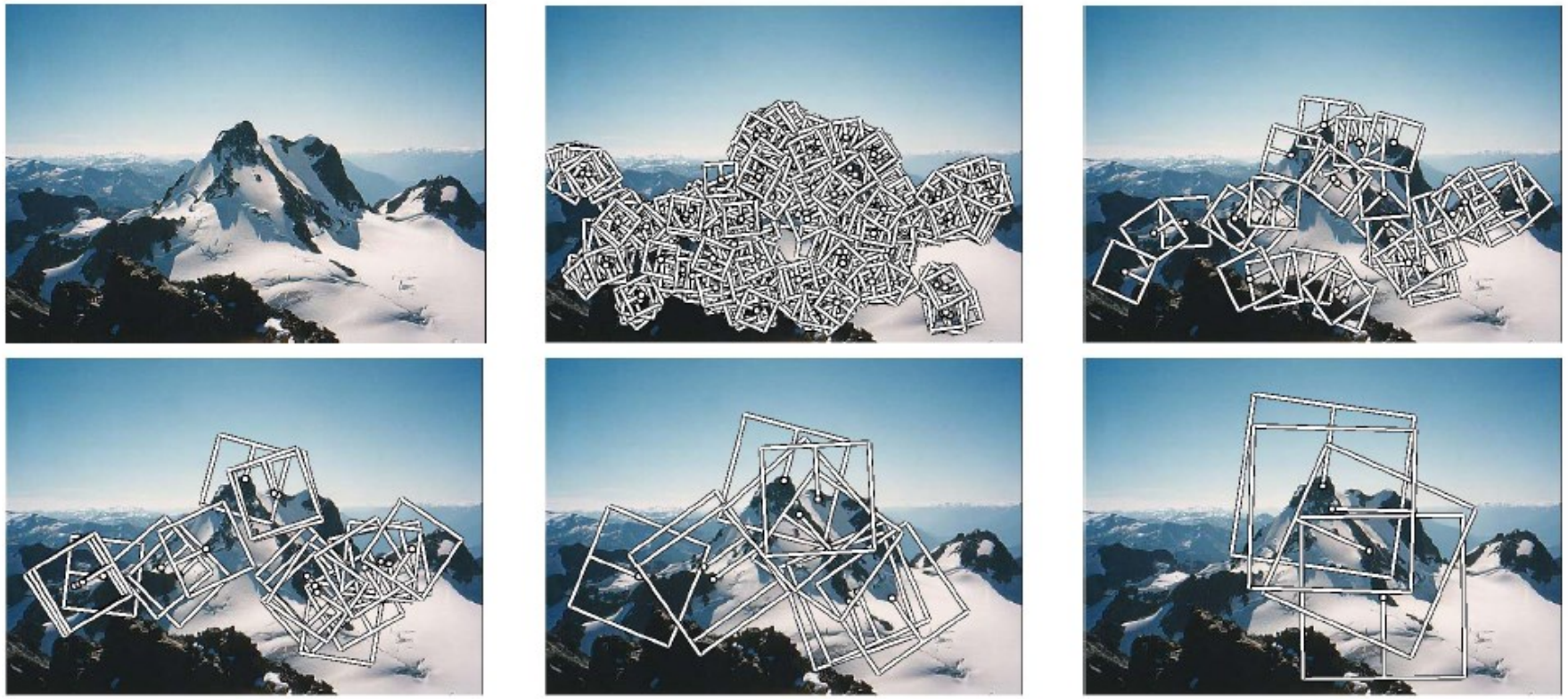
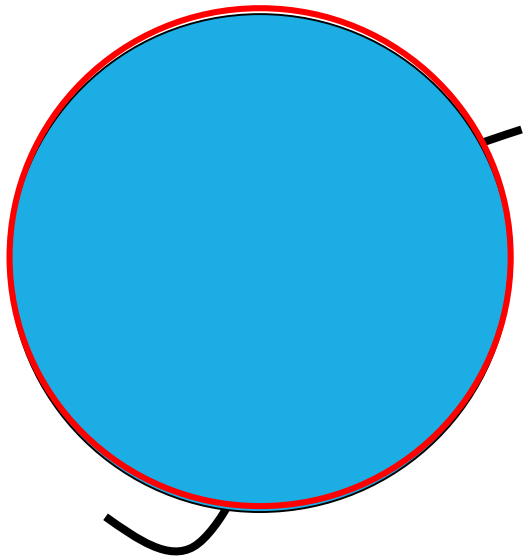


Figure 1. Multi-scale Oriented Patches (MOPS) extracted at five pyramid levels from one of the Matier images. The boxes show the feature orientation and the region from which the descriptor vector is sampled.

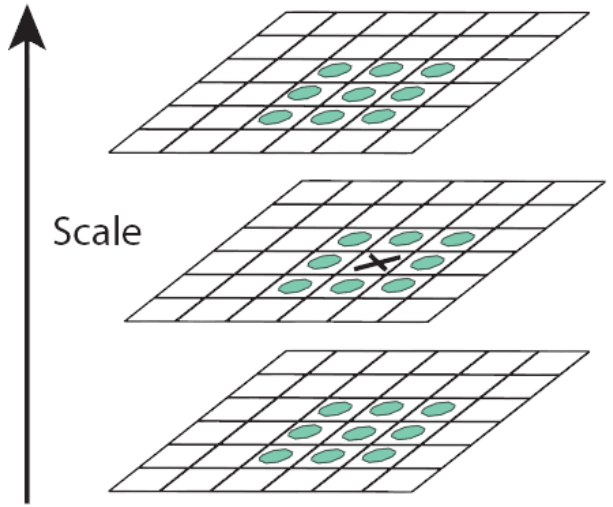
Scale Invariant Detection

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



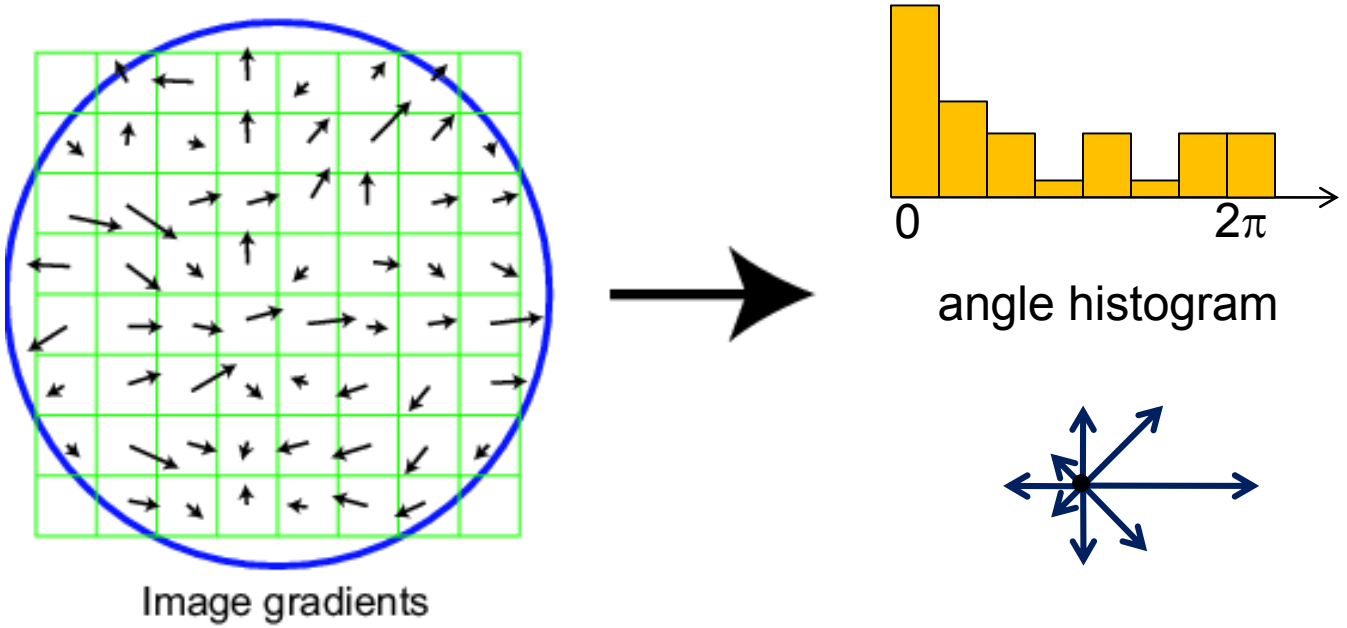
Scale Invariant Feature Transform (SIFT) Features

- In order to detect the local maxima and minima of f, each sample point is compared to its eight neighbors in the current image and nine neighbors in the scale above and below
  - It is selected only if it is larger than all of these neighbors or smaller than all of them.
  - Instead of extracting features at many different scales and then matching all of them, it is more efficient to extract features that are stable in both **location** and **scale**.



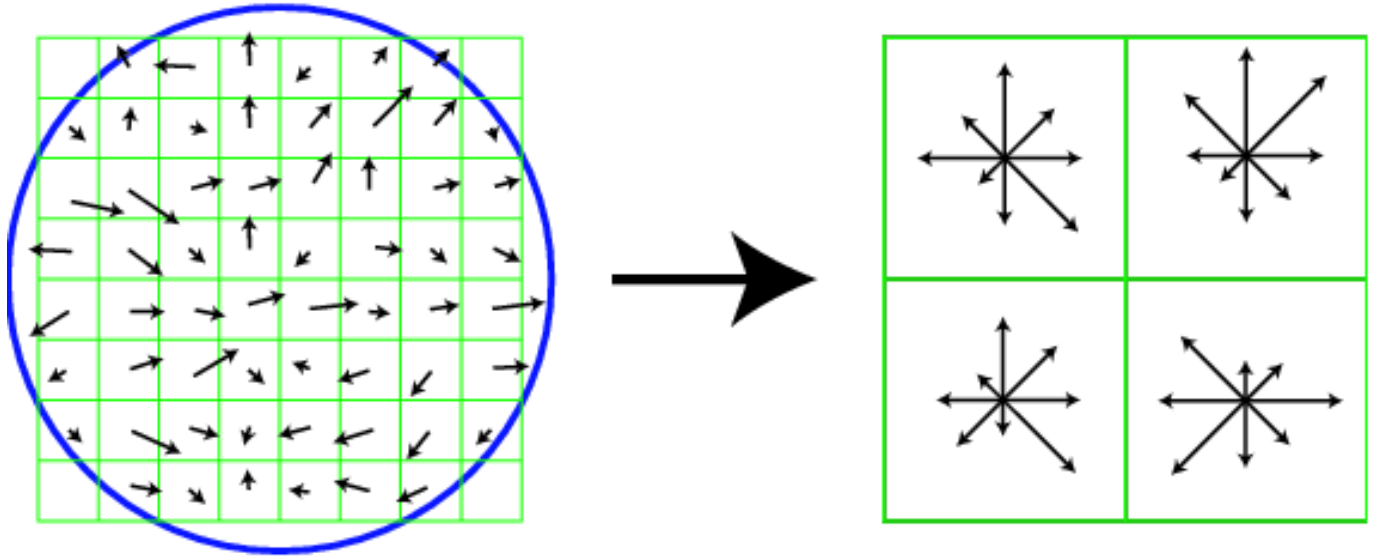
Scale Invariant Feature Transform

- 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



Scale Invariant Feature Transform

- 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



Properties of SIFT

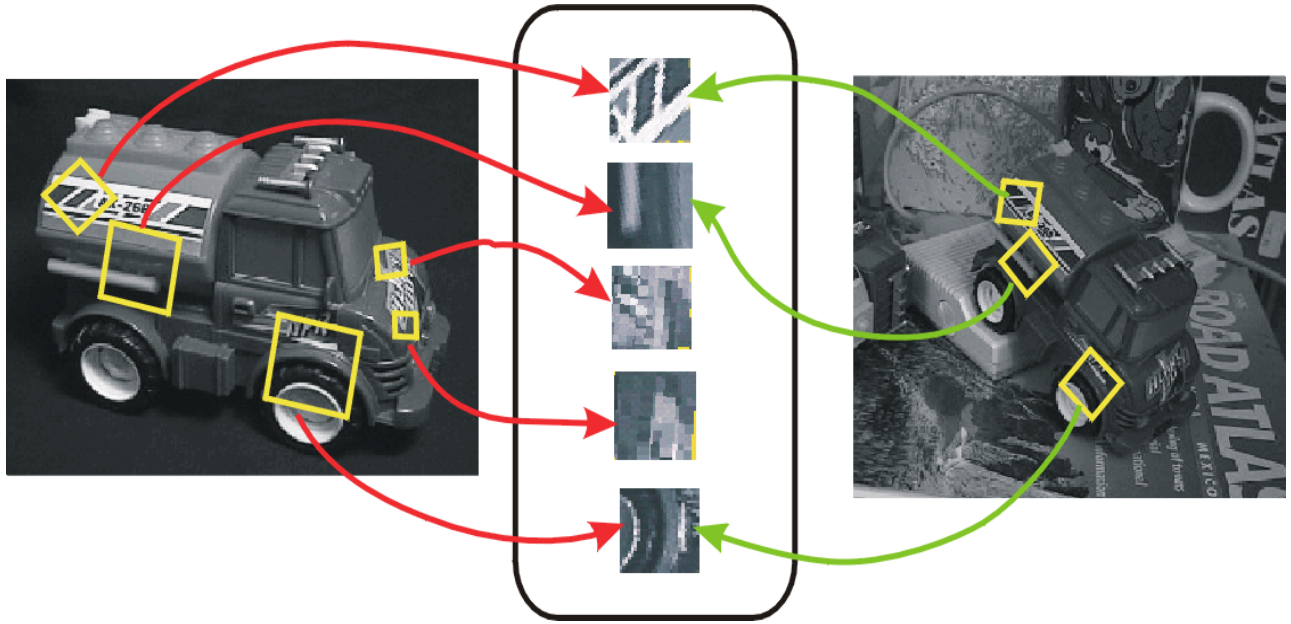
Extraordinarily robust matching technique

- 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

Feature Matching

Feature Matching

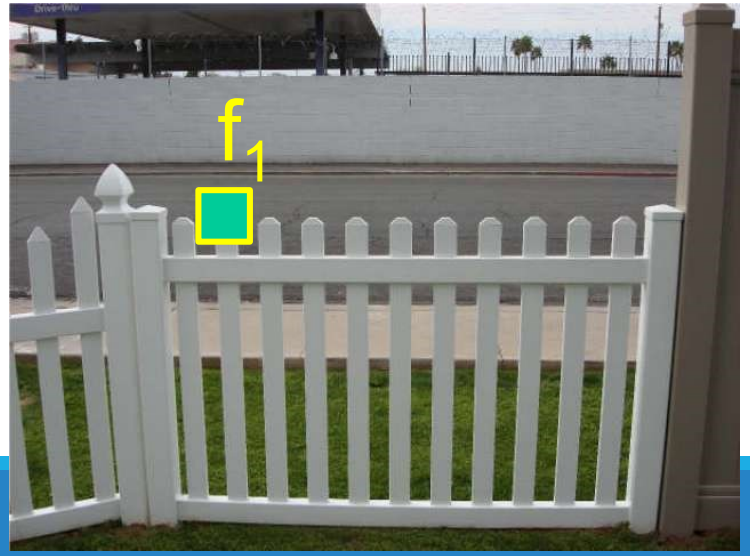
- Given a feature in image I_1, how to find the best match in image I_2?
  1. Define a distance function that compares two descriptors
  2. Test all the features in I_2, find the one with minimum distance

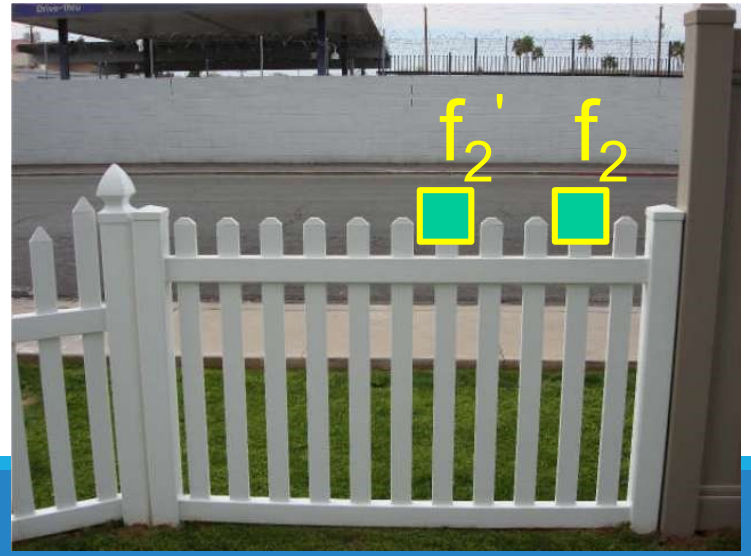


Feature Distance

- SSD is commonly used to measure the difference between two features f_1 and f_2
- Use ratio-of-distance = SSD(f1, f2) / SSD(f1, f2') as a measure of reliability of the feature match:
  - 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
  - Ratio-of-distance is small for ambiguous matches

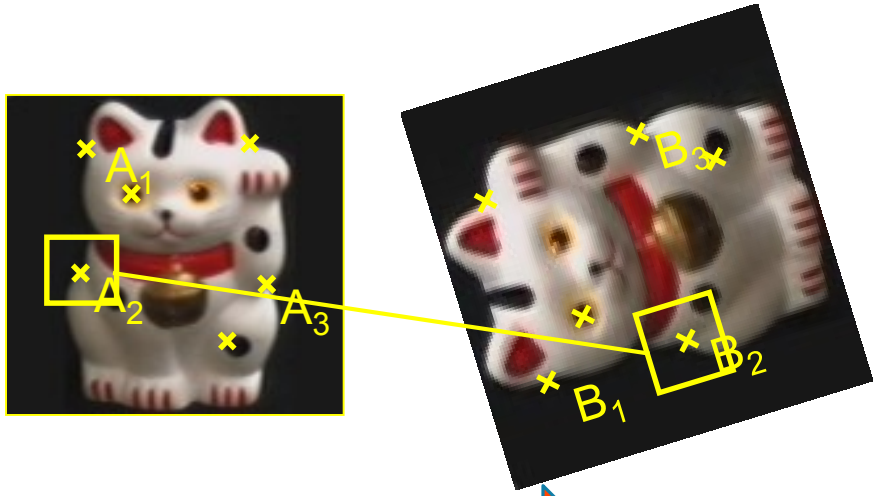
I_1





I_2

Key trade-offs



Detection of interest points

More Repeatable

Robust detection
Precise localization

More Points

Robust to occlusion
Works with less texture

Description of patches

More Distinctive

Minimize wrong matches

More Flexible

Robust to expected variations
Maximize correct matches

Thank You