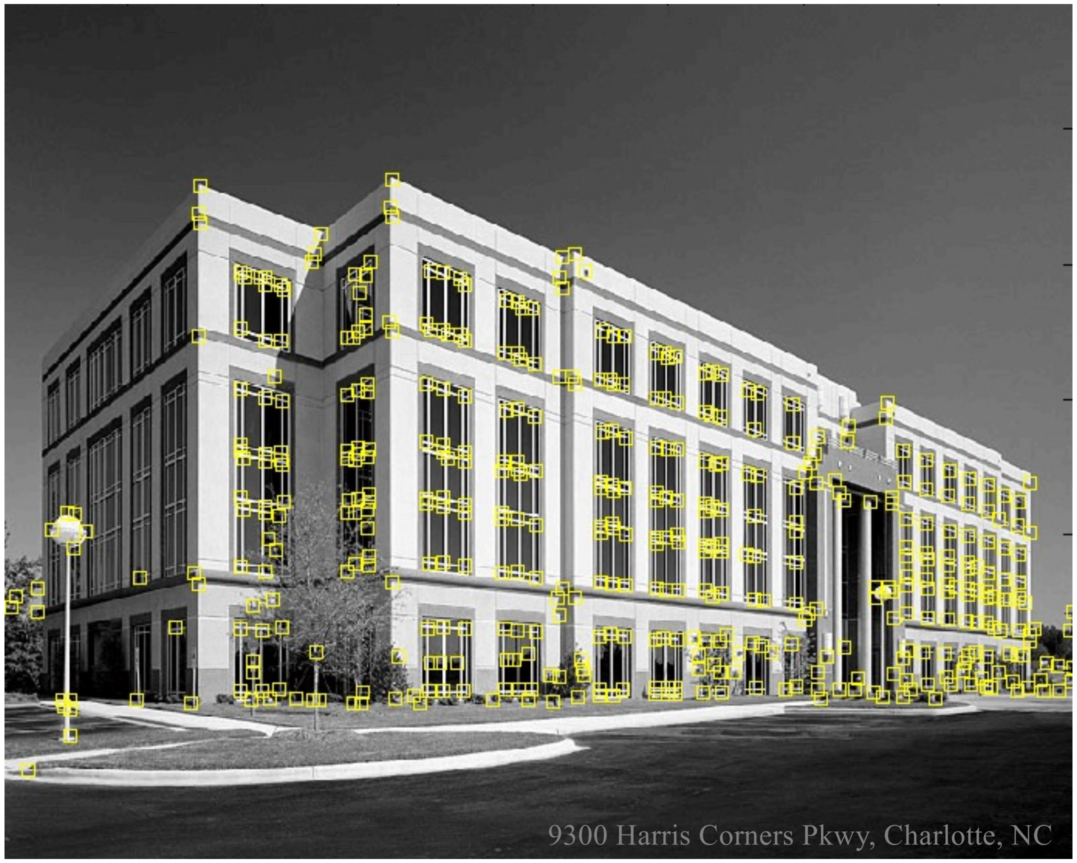
Corner detection

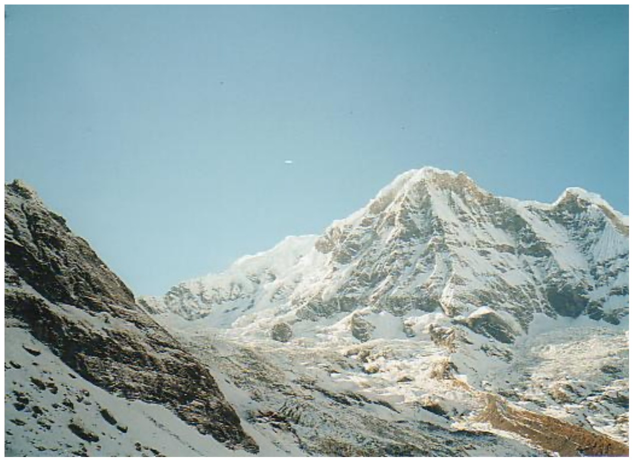
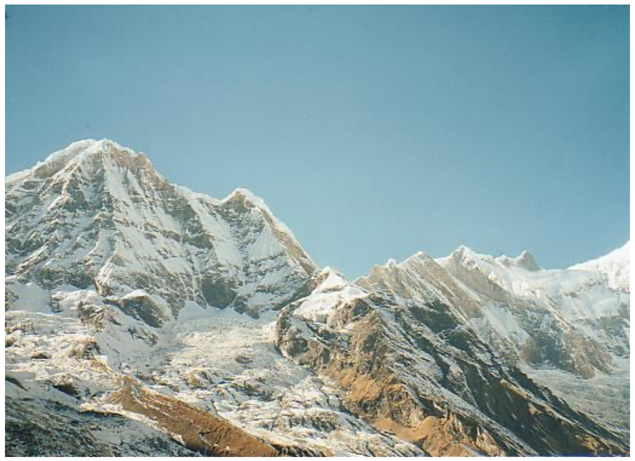


Outline

- Keypoint detection: Motivation
- Deriving a corner detection criterion
- The Harris corner detector
- Invariance properties of corners

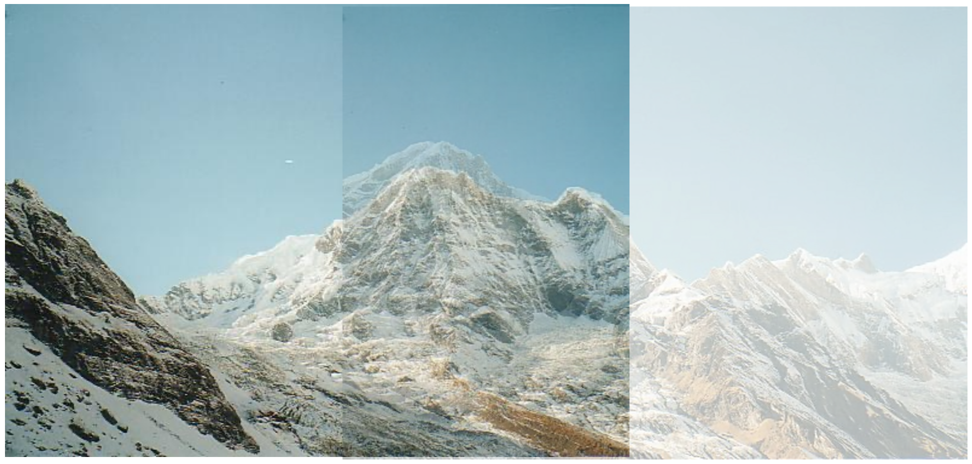
Why extract keypoints?

- Motivation: image alignment
  - We have two images – how do we combine them?

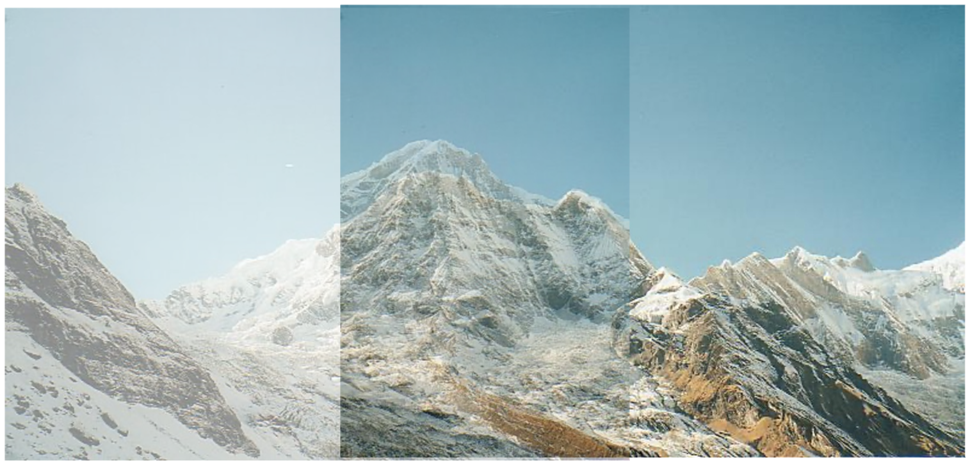
Why extract keypoints?

- Motivation: image alignment
  - We have two images – how do we combine them?



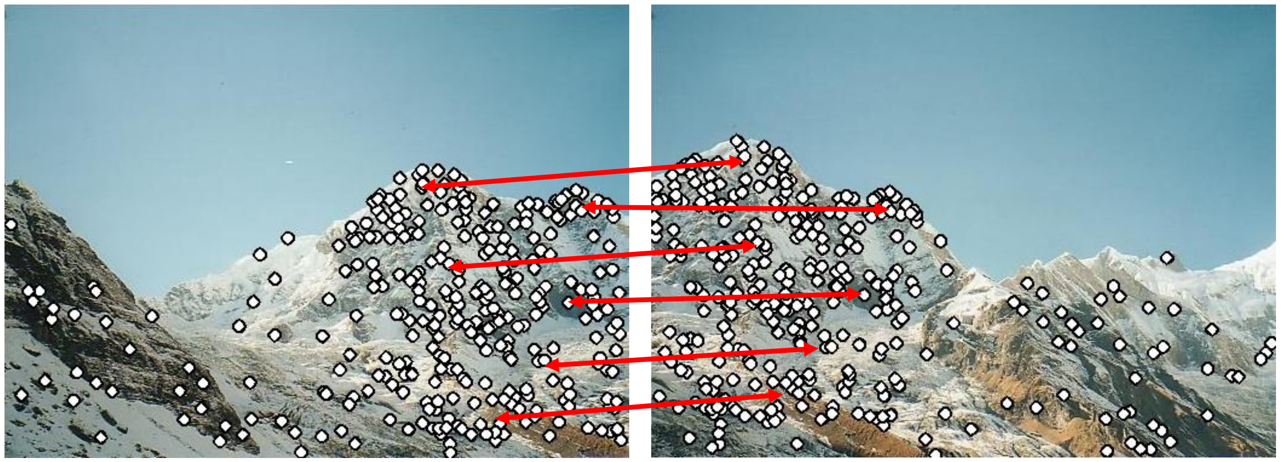
Why extract keypoints?

- Motivation: image alignment
  - We have two images – how do we combine them?



Why extract keypoints?

- Motivation: image alignment
  - We have two images – how do we combine them?

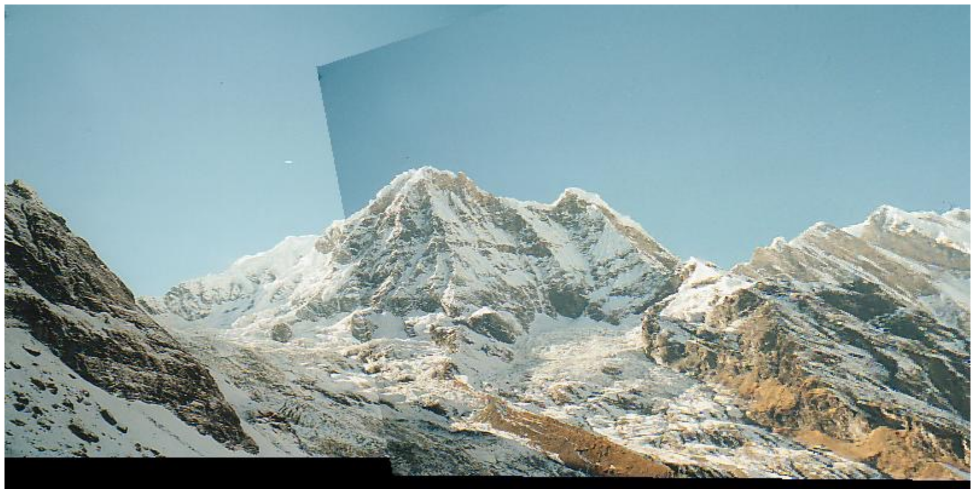


Step 1: extract keypoints

Step 2: match keypoint features

Why extract keypoints?

- Motivation: image alignment
  - We have two images – how do we combine them?



Step 1: extract keypoints

Step 2: match keypoint features

Step 3: align images

Keypoint representations support very fast methods

In some applications, GPU just isn't available
   -too heavy; too much power; etc
And there isn't much CPU either

Idea: reduce image to keypoints, work with those

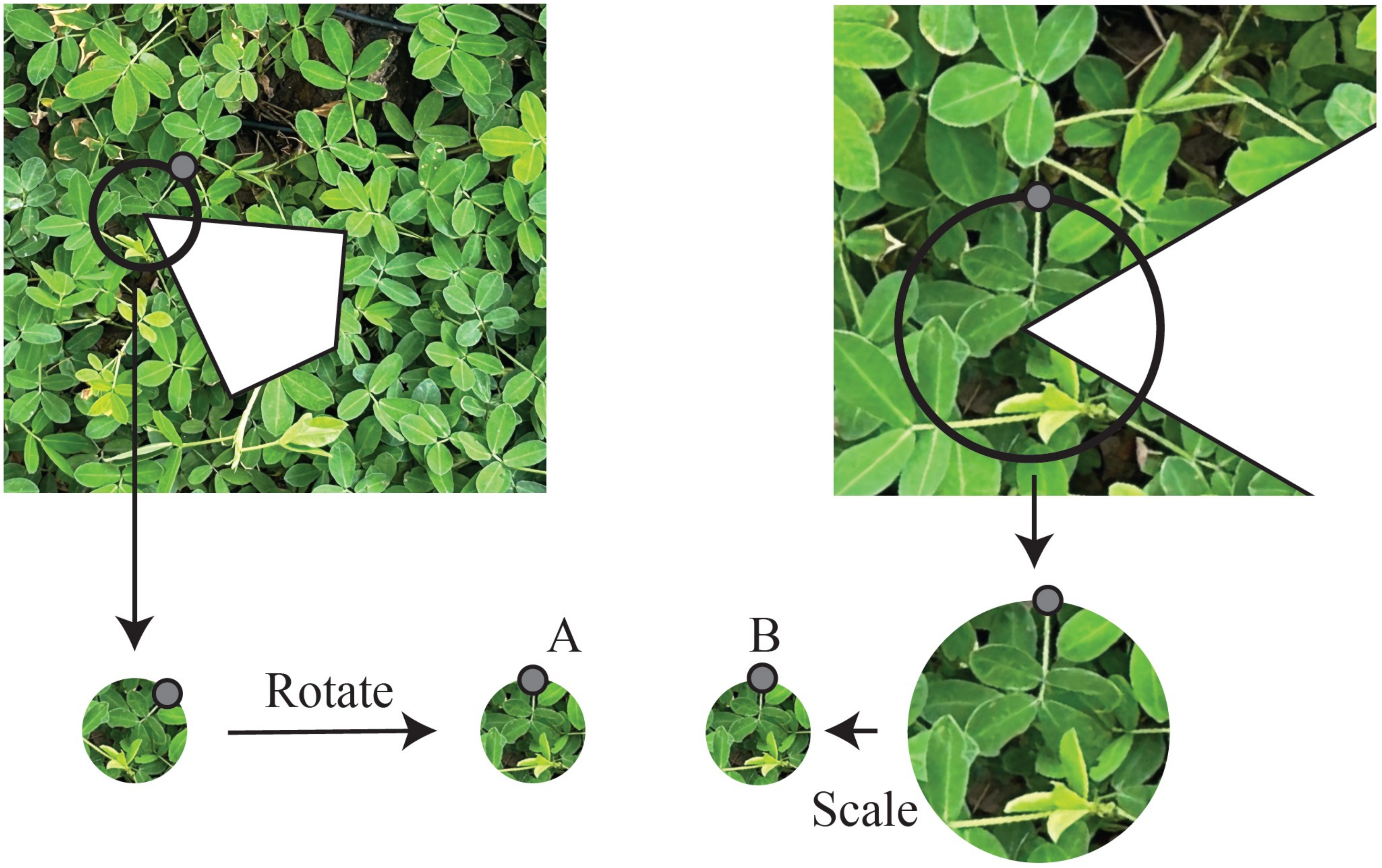
Need to find

Where is it:

What scale its at:

In a way that is (mostly) unaffected by image transformations

What orientation its at:


A
B
Rotate
Scale

Need to find

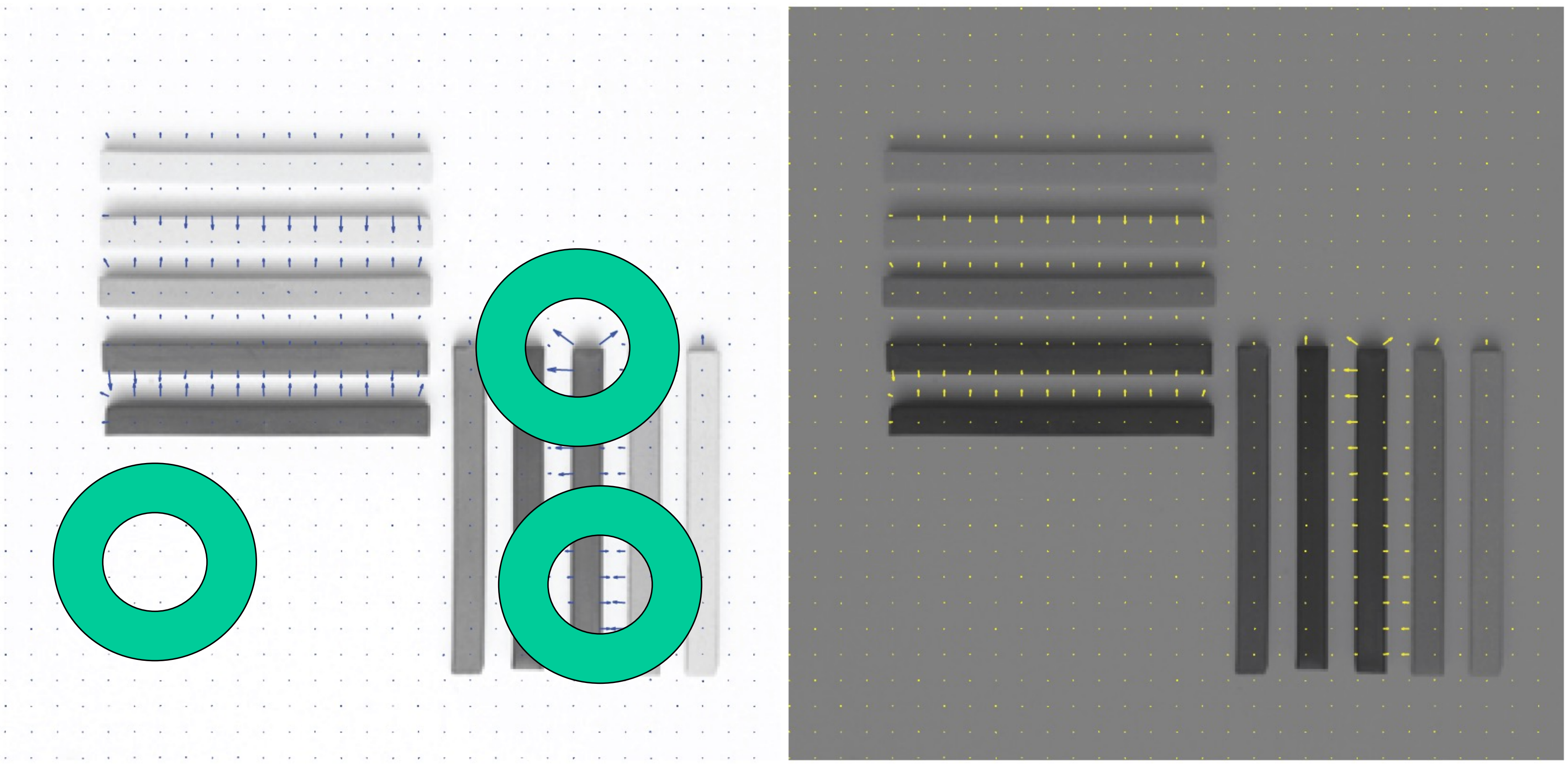
Where is it:

What scale its at:

In a way that is (mostly) unaffected by image transformations

What orientation its at:

Description of contents:



Corner detection matrix

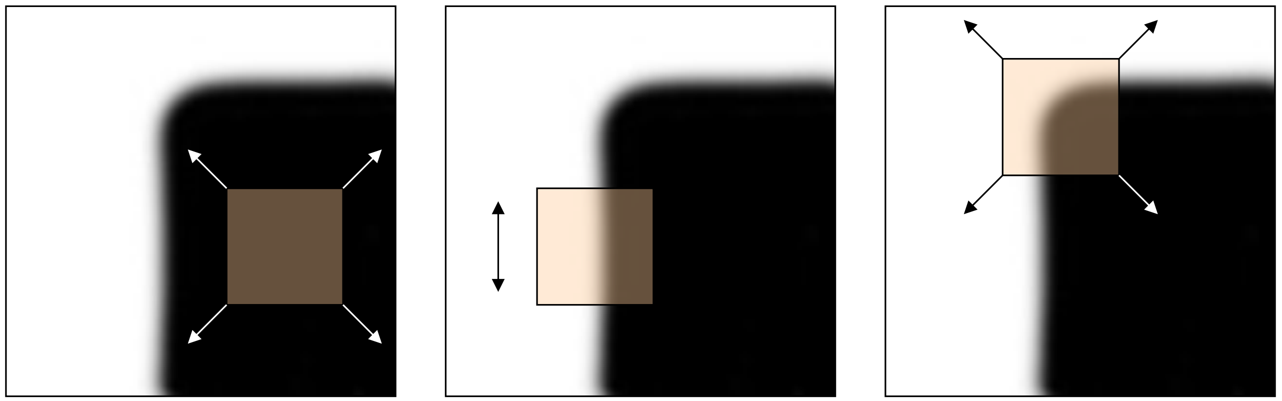
$$\mathcal{H} = \sum_{window} \{(\nabla I)(\nabla I)^T\}$$

$$\approx \sum_{window} \left\{ \begin{array}{cc} (\frac{\partial G_\sigma}{\partial x} **\mathcal{I})(\frac{\partial G_\sigma}{\partial x} **\mathcal{I}) & (\frac{\partial G_\sigma}{\partial x} **\mathcal{I})(\frac{\partial G_\sigma}{\partial y} **\mathcal{I}) \\ (\frac{\partial G_\sigma}{\partial x} **\mathcal{I})(\frac{\partial G_\sigma}{\partial y} **\mathcal{I}) & (\frac{\partial G_\sigma}{\partial y} **\mathcal{I})(\frac{\partial G_\sigma}{\partial y} **\mathcal{I}) \end{array} \right\}$$

This is very like a covariance matrix
Also, it’s a second order approximation to the image
Intensity (later slides)

Deriving a corner detection criterion

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



"flat" region:
no change in all directions

"edge":
no change along the edge direction

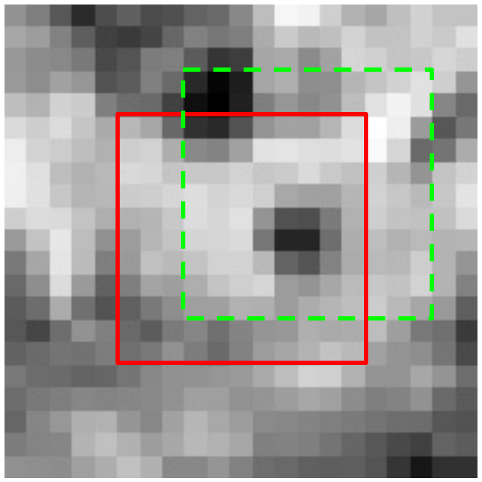
"corner":
significant change in all directions

Deriving a corner detection criterion

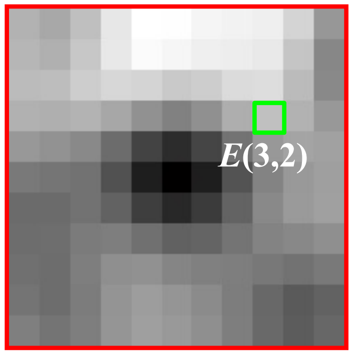
- Change in appearance of window W for the shift (u, v):

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

$I(x, y)$



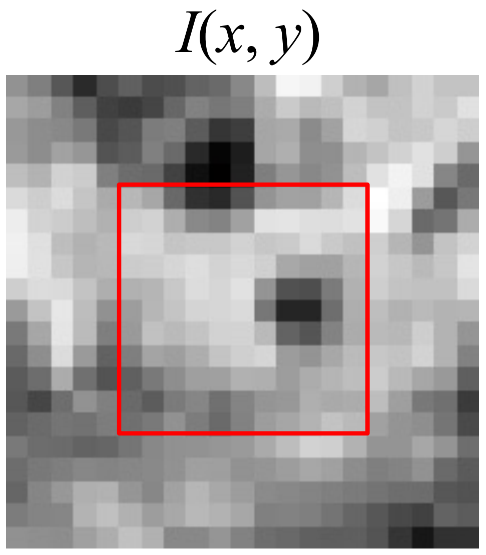
$E(u, v)$

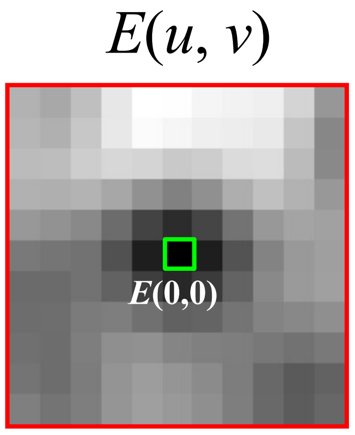


Deriving a corner detection criterion

- Change in appearance of window W for the shift (u, v):

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





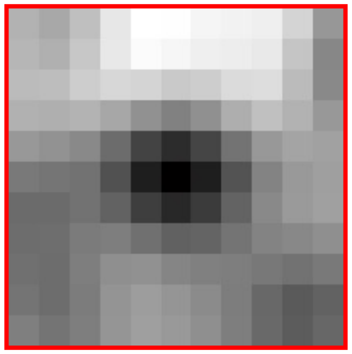
Deriving a corner detection criterion

- Change in appearance of window W for the shift (u, v):

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

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

$E(u, v)$

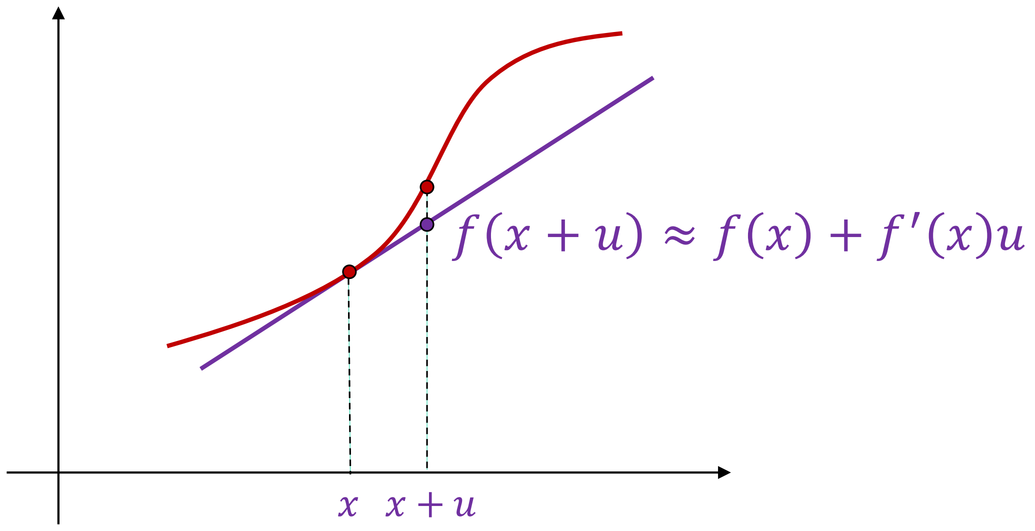


Deriving a corner detection criterion

- First-order Taylor approximation for small shifts (u, v):

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

- Recall: first-order Taylor approximation for a 1D function:



Deriving a corner detection criterion

- First-order Taylor approximation for small shifts (u, v):

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

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

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

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

$$= \sum_{(x,y)\in W} \left[I_x u + I_y v\right]^2 = \sum_{(x,y)\in W} I_x^2 u^2 + 2 I_x I_y uv + I_y^2 v^2$$

Deriving a corner detection criterion

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

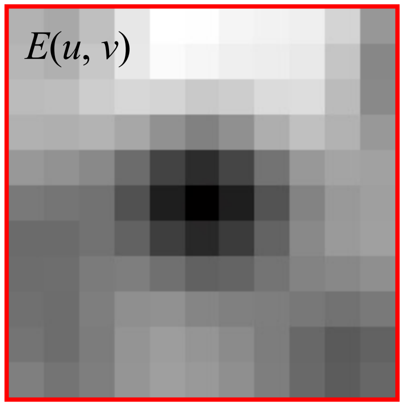
$$= \begin{pmatrix} u & v \end{pmatrix} \begin{bmatrix} \sum\limits_{x,y} I_x^2 & \sum\limits_{x,y} I_x I_y \\ \sum\limits_{x,y} I_x I_y & \sum\limits_{x,y} I_y^2 \end{bmatrix} \begin{pmatrix} u \\ v \end{pmatrix}$$

Second moment matrix M

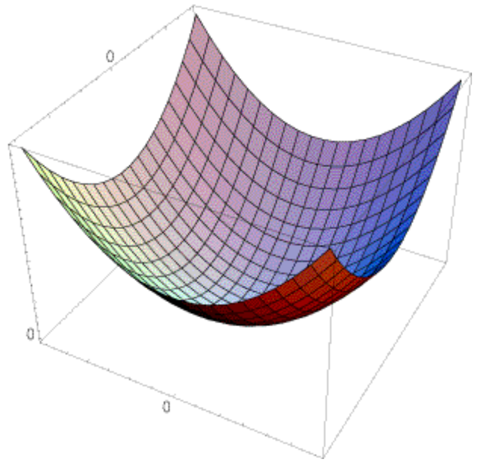
Deriving a corner detection criterion

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

- This is a quadratic function of (u,v):



$\approx$

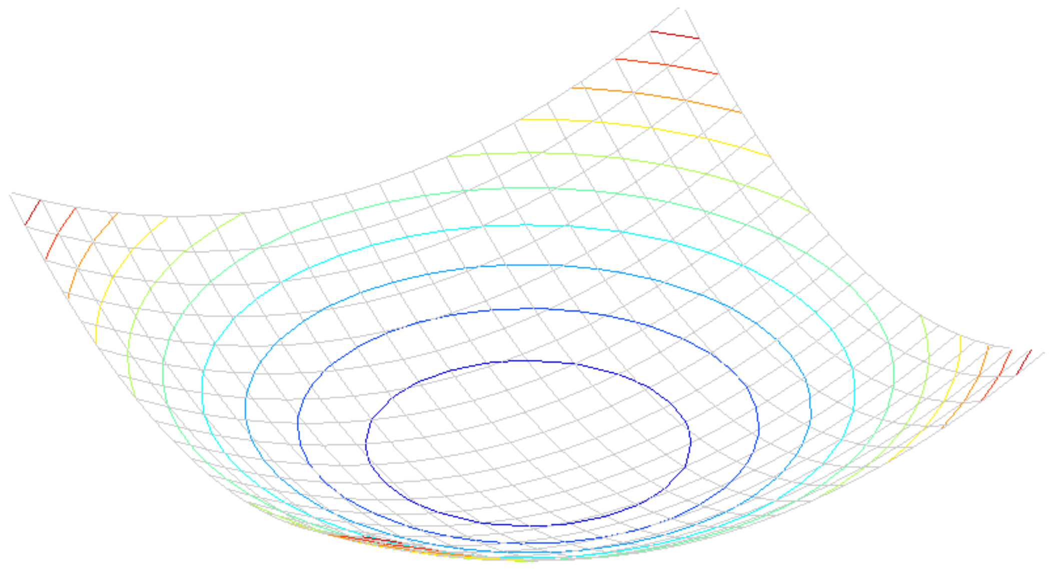


- How can we analyze the shape of this surface?

Interpreting the second moment matrix

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

$$(u \quad v) M \begin{pmatrix} u \\ v \end{pmatrix} = \text{const.}$$

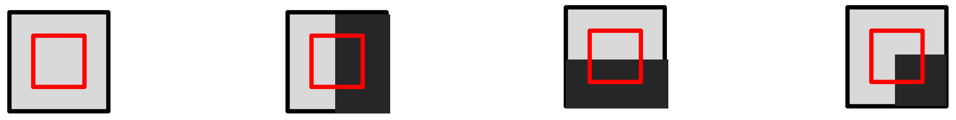


Interpreting the second moment matrix

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

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

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



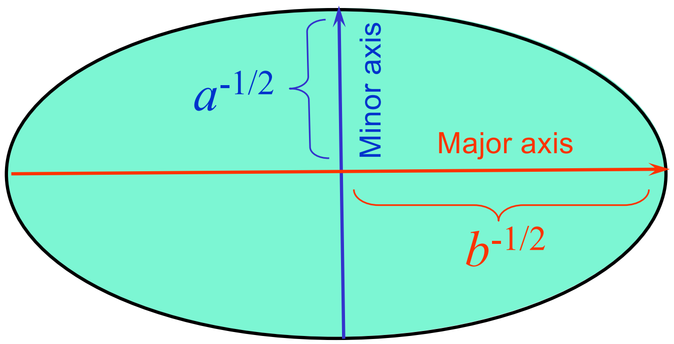
Interpreting the second moment matrix

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

$$\begin{pmatrix} u & v \end{pmatrix} \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = 1$$

$$au^2 + bv^2 = 1$$

$$\frac{u^2}{(a^{-1/2})^2} + \frac{v^2}{(b^{-1/2})^2} = 1$$

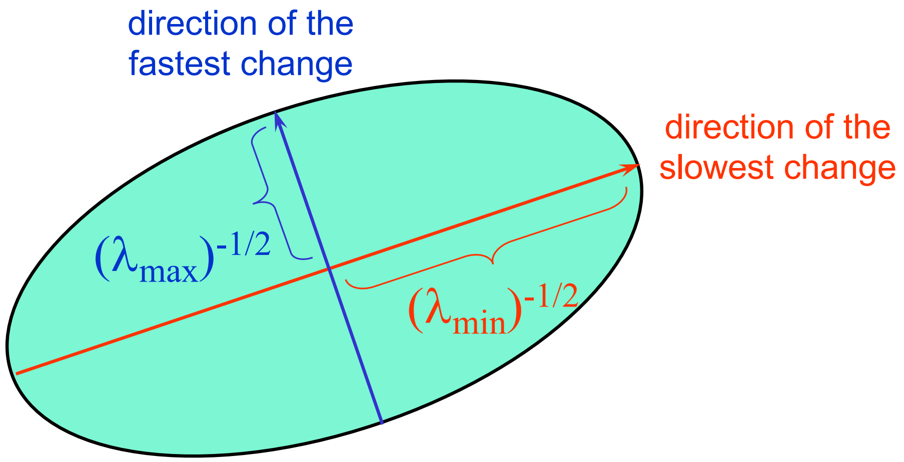


Interpreting the second moment matrix

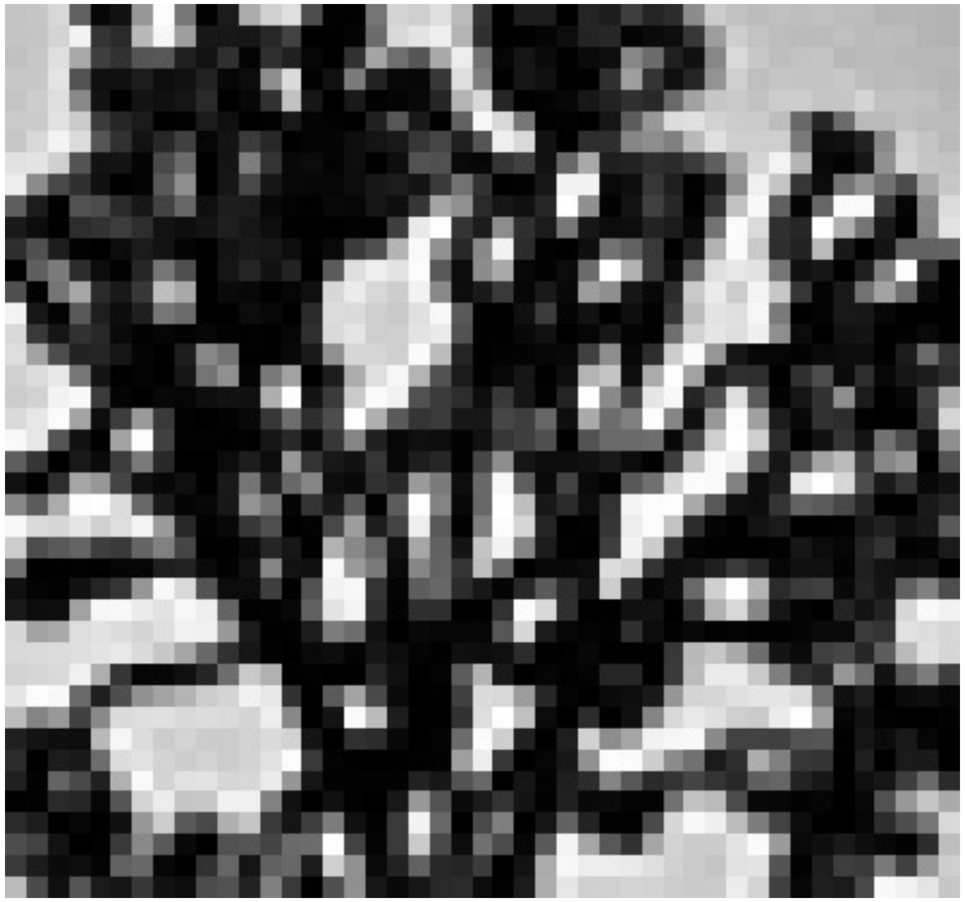
- In general, need to *diagonalize* M:

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

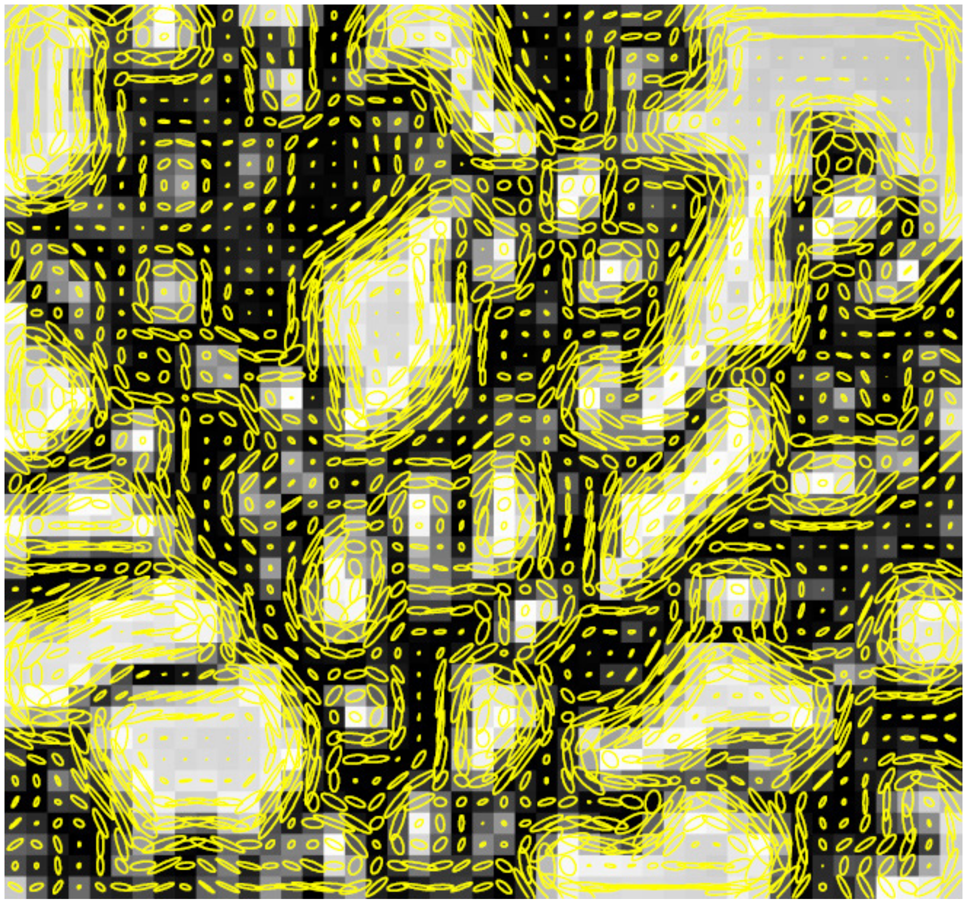
- The axis lengths of the ellipse are determined by the eigenvalues and the orientation is determined by R


direction of the fastest change
direction of the slowest change
$(\lambda_{max})^{-1/2}$
$(\lambda_{min})^{-1/2}$

Visualization of second moment matrices



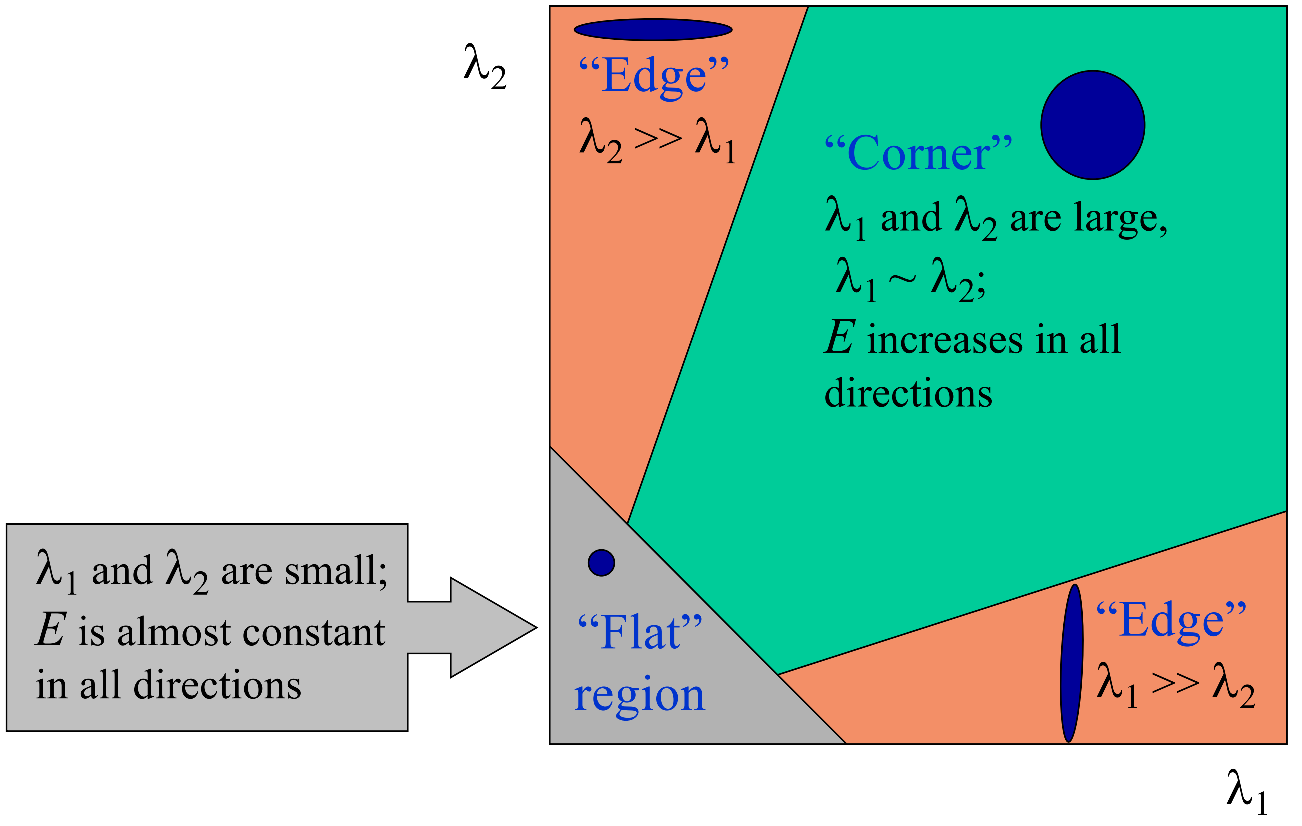
Visualization of second moment matrices



Note: axes are rescaled so ellipse areas are proportional to edge energy (i.e., bigger ellipses correspond to stronger edges)

Interpreting the eigenvalues

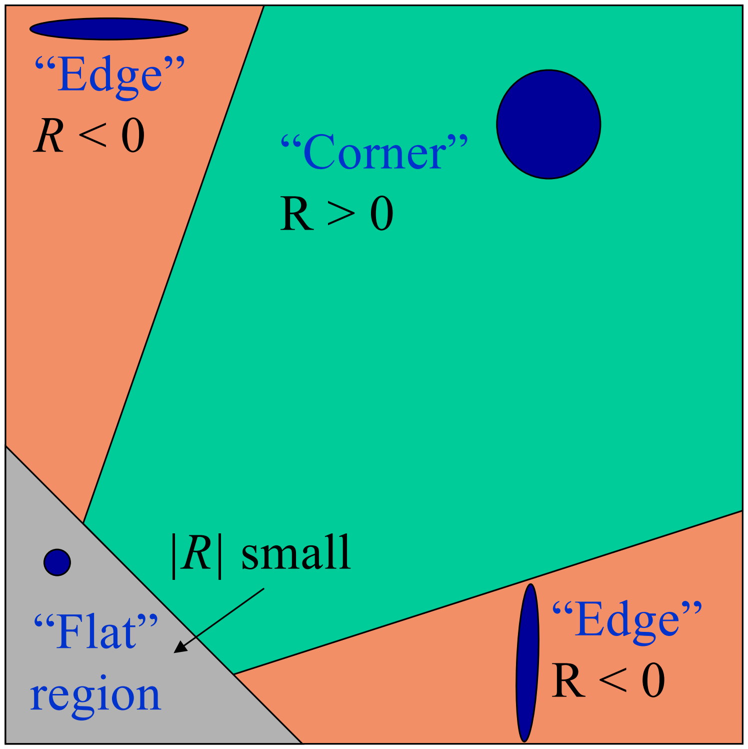
- Classification of image points using eigenvalues of M:

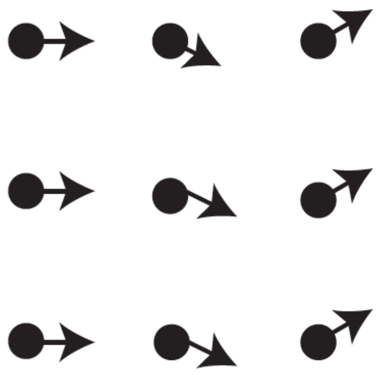

λ_2
"Edge"
$\lambda_2 >> \lambda_1$
"Corner"
λ_1 and λ_2 are large,
$\lambda_1 \sim \lambda_2$;
E increases in all directions
λ_1 and λ_2 are small;
E is almost constant in all directions
"Flat" region
"Edge"
$\lambda_1 >> \lambda_2$
λ_1

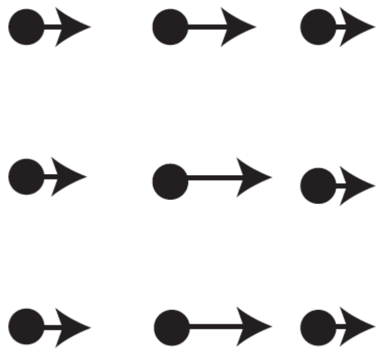
Corner response function

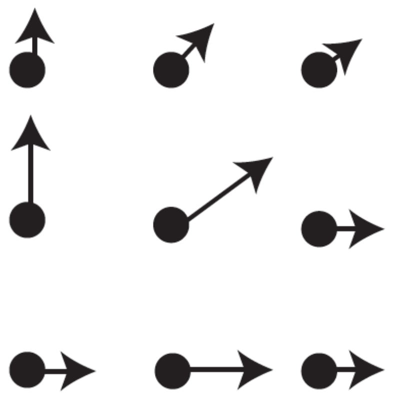
$$R = \det(M) - \alpha\,\mathrm{trace}(M)^2 = \lambda_1\lambda_2 - \alpha(\lambda_1 + \lambda_2)^2$$

α: constant (0.04 to 0.06)




Two small


One small, one large


Two large

The Harris corner detector

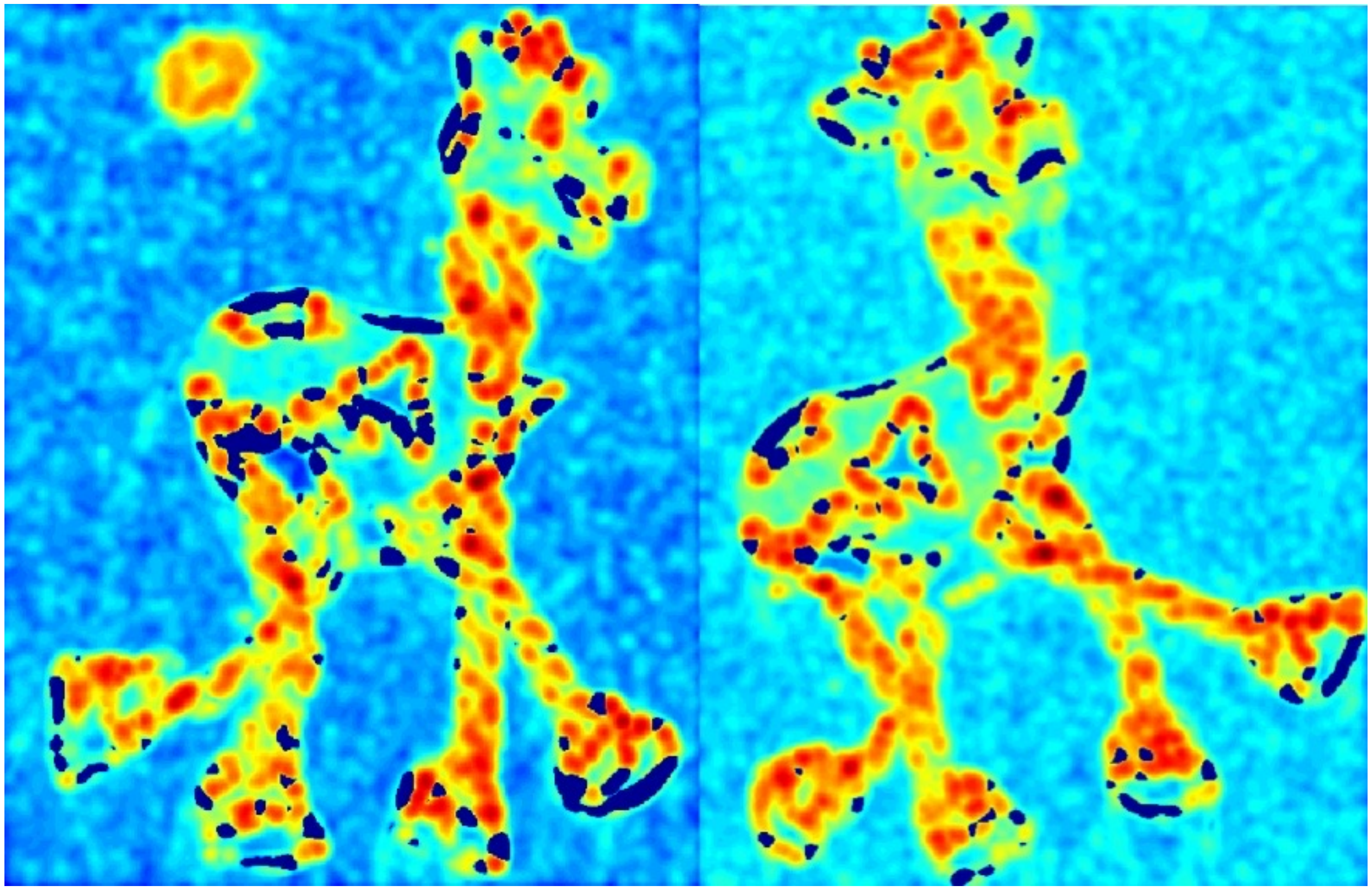
1. Compute partial derivatives I_x and I_y at each pixel
2. Compute second moment matrix in a *Gaussian window* around each pixel
3. Compute corner response function $R = \det(M) - \alpha \operatorname{trace}(M)^2$
4. Threshold R
5. Find local maxima of response function (NMS)

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

Harris Detector: Example

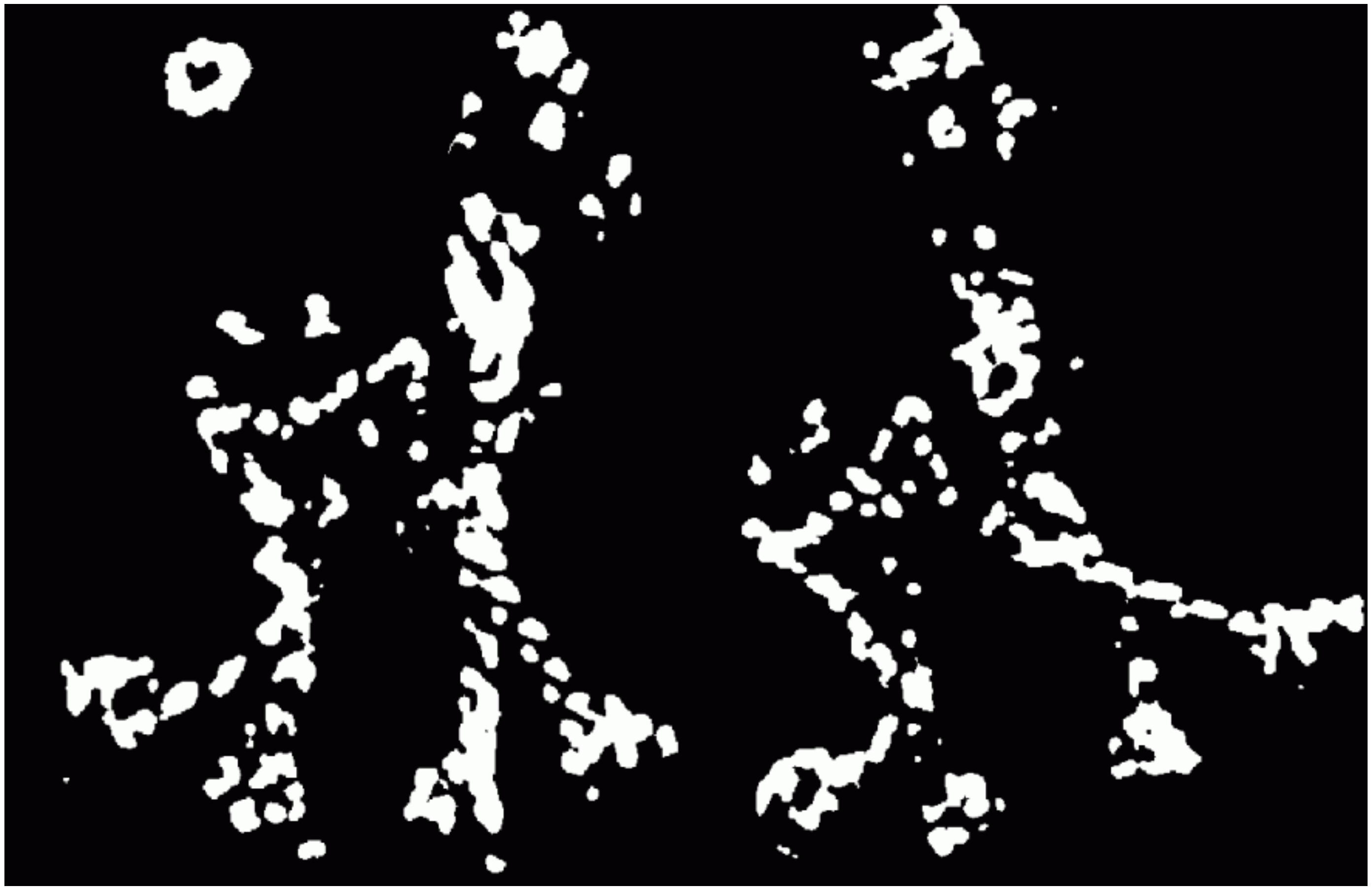
Harris Detector: Example

Compute corner response R



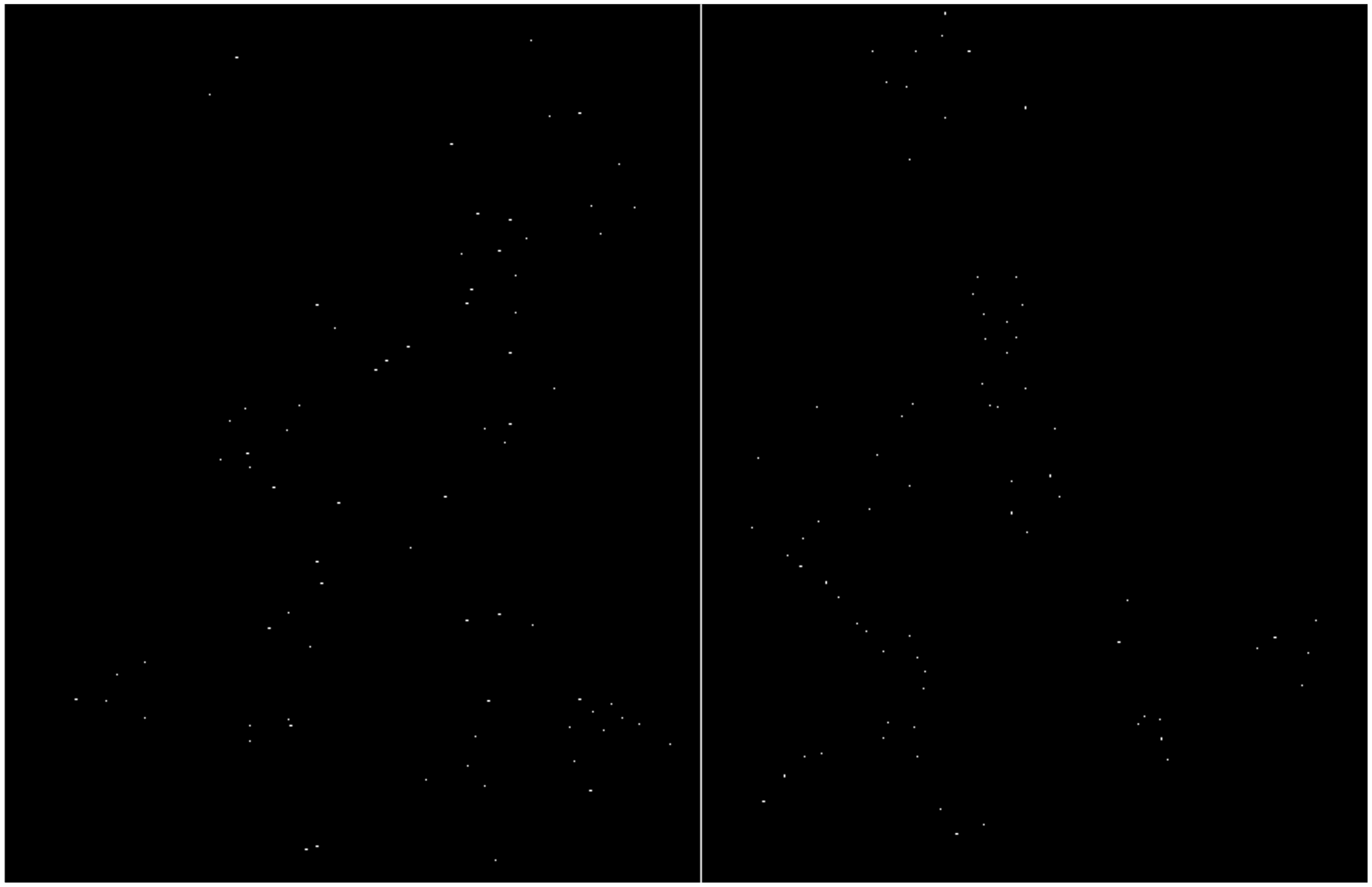
Harris Detector: Example

Find points with large corner response: R > threshold

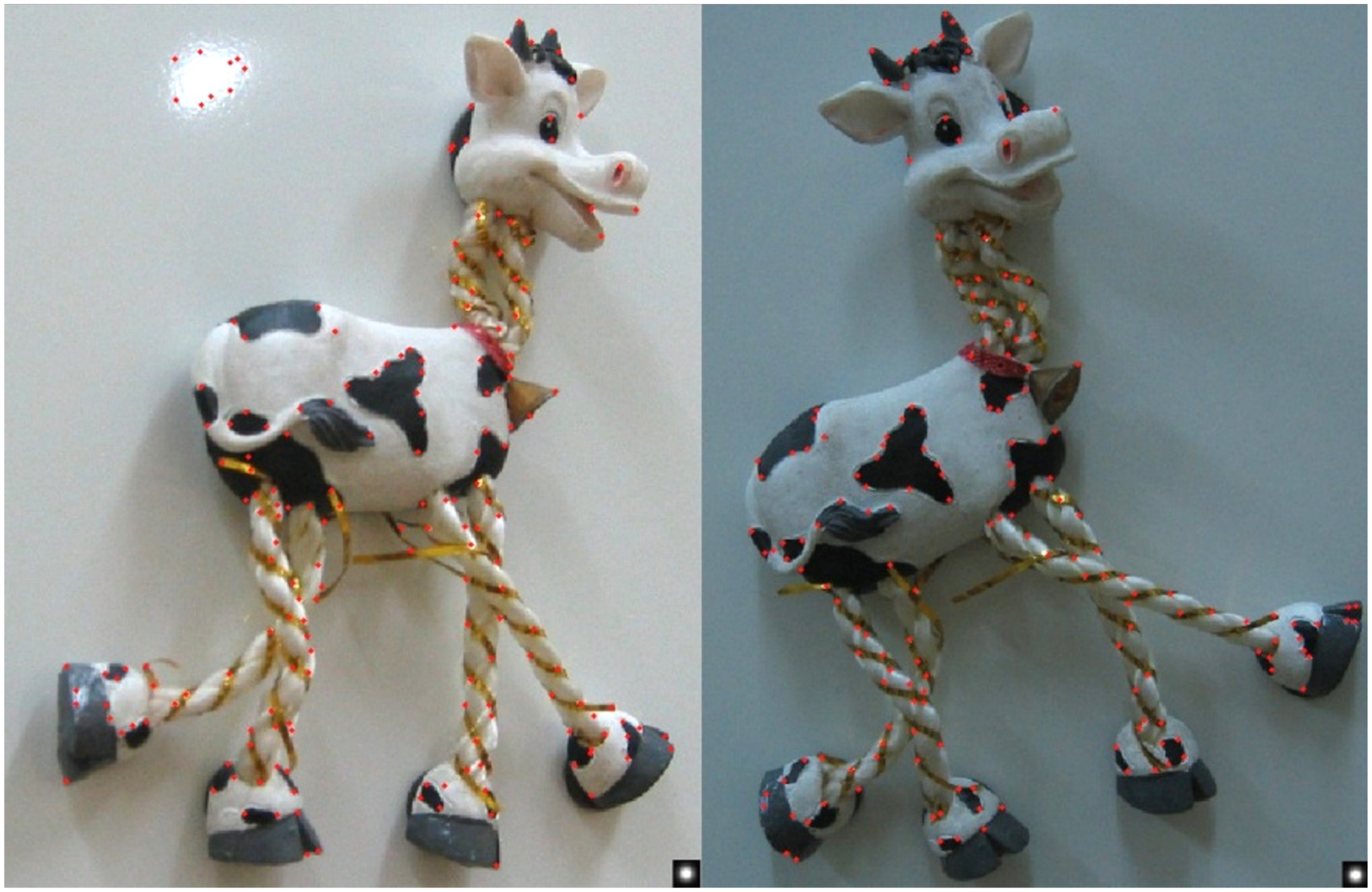


Harris Detector: Example

Take only the points of local maxima of R

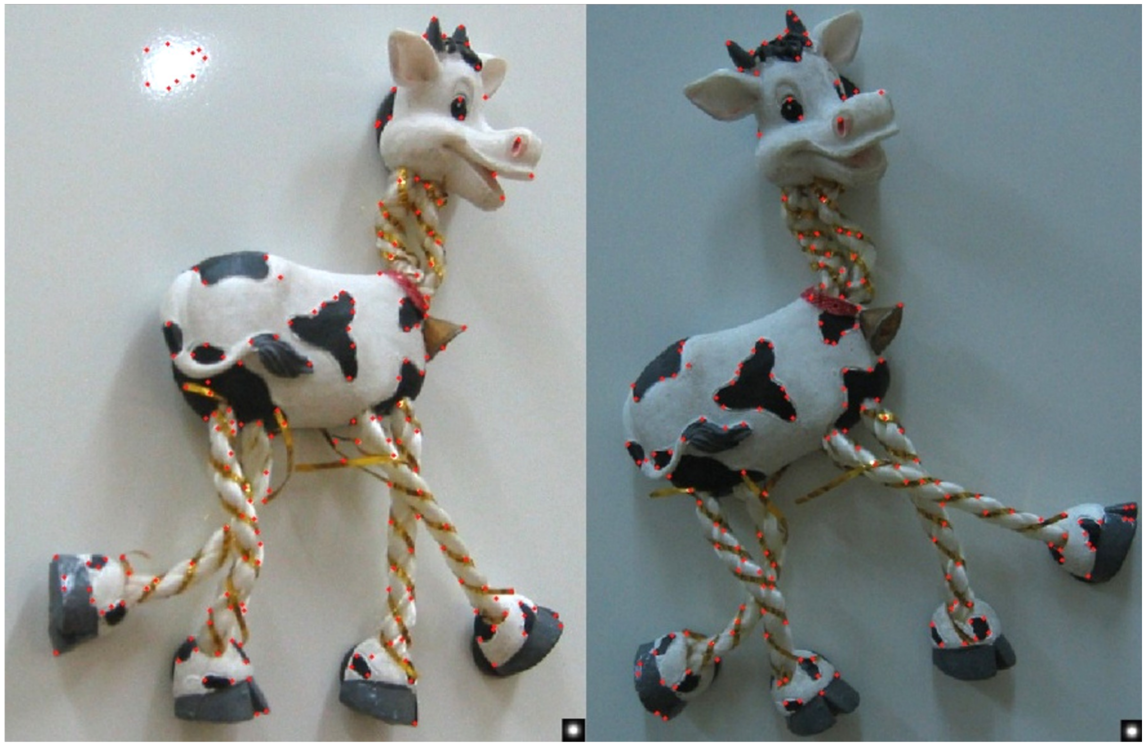


Harris Detector: Example

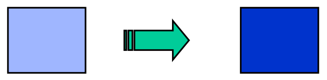


Behavior w.r.t. image transformations

- To be useful for image matching, “the same” corner features need to show up despite geometric and photometric transformations
- We need to analyze how *the corner response function* and *the corner locations* change in response to various transformations

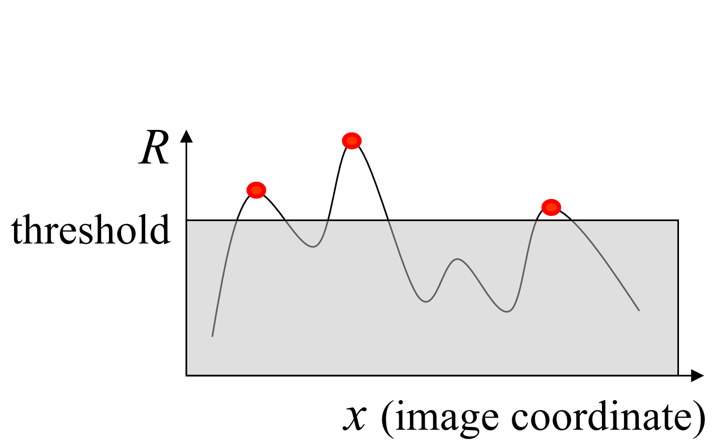


Affine intensity change



$I \rightarrow a\,I + b$

- What happens to the corner response function in case of intensity shifts ($I \rightarrow I + b$)?
  - It depends only on image derivatives, so it's *invariant* to intensity shifts
- What about intensity scaling ($I \rightarrow a\,I$)?
  - *Not fully invariant* if threshold stays constant



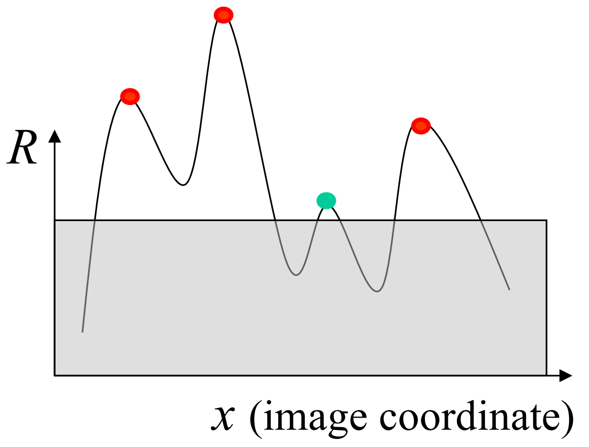
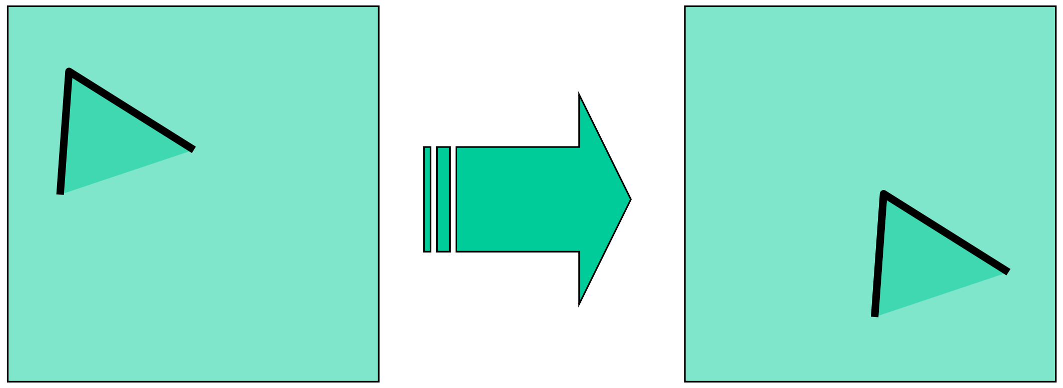


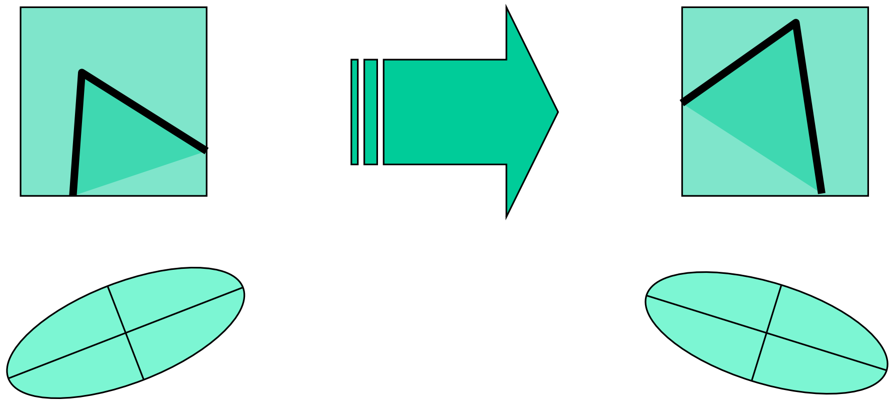
Image translation



- How do the detected corner locations change if the image pattern is translated?
  - All the ingredients of the second moment matrix are *shift-invariant,* and so is the corner response function
  - However, the locations of the corners are *equivariant* (or *covariant*) w.r.t. shifts

Translate the image – the corners translate

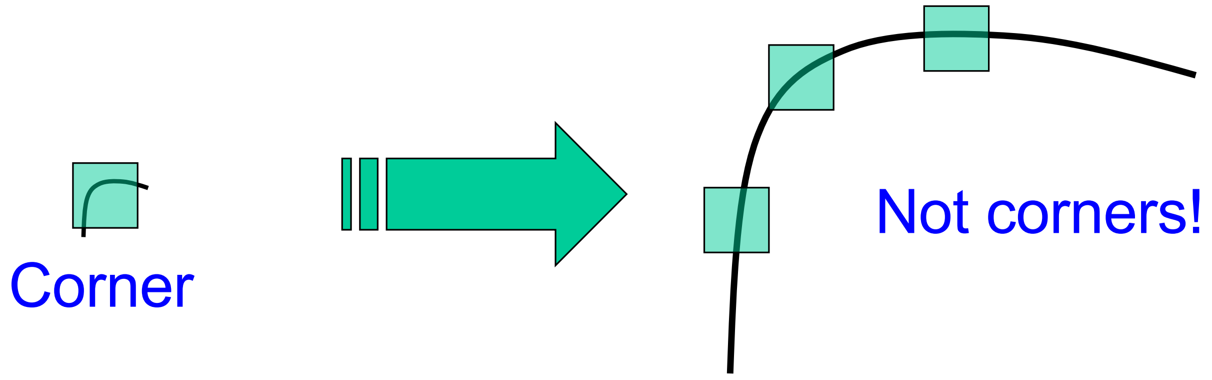
Image rotation



- How do the detected corner locations change if the image pattern is *rotated*?
  - Assuming the second moment matrix is calculated over a circular neighborhood (and ignoring resampling issues), the rotation changes but the eigenvalues stay the same, so the response function is *invariant*
  - The locations of the corners are *equivariant* (or *covariant*) w.r.t. rotations

Rotate the image – the corners rotate

Image scaling



- How do the detected corner locations change if the image pattern is *scaled*?
  - Assuming fixed-size neighborhoods for calculating the second moment matrix, the corner response function is *not* invariant and the corner locations are *not* equivariant w.r.t. scaling

Scale the image – lose/gain some corners (but not too bad)

Need to find

Where is it:

What scale its at:

In a way that is (mostly) unaffected by image transformations

What orientation its at:

Description of contents:

Laplacian of Gaussian

The *Laplacian* of a function in 2D is defined as

$$(\nabla^2 f)(x, y) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}.$$

It is natural to smooth the image before applying a Laplacian. Notice that the Laplacian is a linear operator and is shift invariant (exercises), meaning that we could represent taking the Laplacian as convolving the image with some kernel (which we write as K_{∇^2}). Because convolution is associative, we have that

$$(K_{\nabla^2} * (G_\sigma * \mathcal{I})) = (K_{\nabla^2} * G_\sigma) * \mathcal{I} = (\nabla^2 G_\sigma) * \mathcal{I}.$$

The reason this is important is that, just as for first derivatives, smoothing an image and then applying the Laplacian is the same as convolving the image with the Laplacian of the smoothing kernel. Figure 8.10 shows what happens for the usual case where smoothing is Gaussian smoothing. This kernel looks like a dark blob with a light ring around it on a gray background (closed form expression in the exercises). As the scale of the Gaussian gets larger, the blob gets bigger.

This means the Laplacian of Gaussian can be used to find the radius of the neighborhood around a given corner located at i, j in $\mathcal{I}$. Write $\mathcal{L}_\sigma$ for a Laplacian of Gaussian kernel where the Gaussian has scale σ. Now, for each σ, place the kernel on the image, centered at the corner, and compute

$$V(\sigma) = \sum_{uv} \mathcal{I}_{i-u,j-v} \mathcal{L}_{\sigma;uv}.$$

The σ that maximizes (or minimizes) $V(\sigma)$ is the radius to use. This value has strong covariance properties. Imagine image $\mathcal{S}$ is the same as $\mathcal{I}$, but is upsampled factor of k. Then the value of σ chosen by this procedure for $\mathcal{S}$ will be k times the value chosen for $\mathcal{I}$. This is because the value is, essentially a dot-product (as in Section 41.2), and scores the match between the image and the kernel. If the image is upsampled by k, the best matching kernel should just k times the scale. This argument works for downsampling as well as upsampling, and is fine as long as there is no serious loss of information in the upsampling or downsampling (which is most of the time).

Need to find

Where is it:

What scale its at:

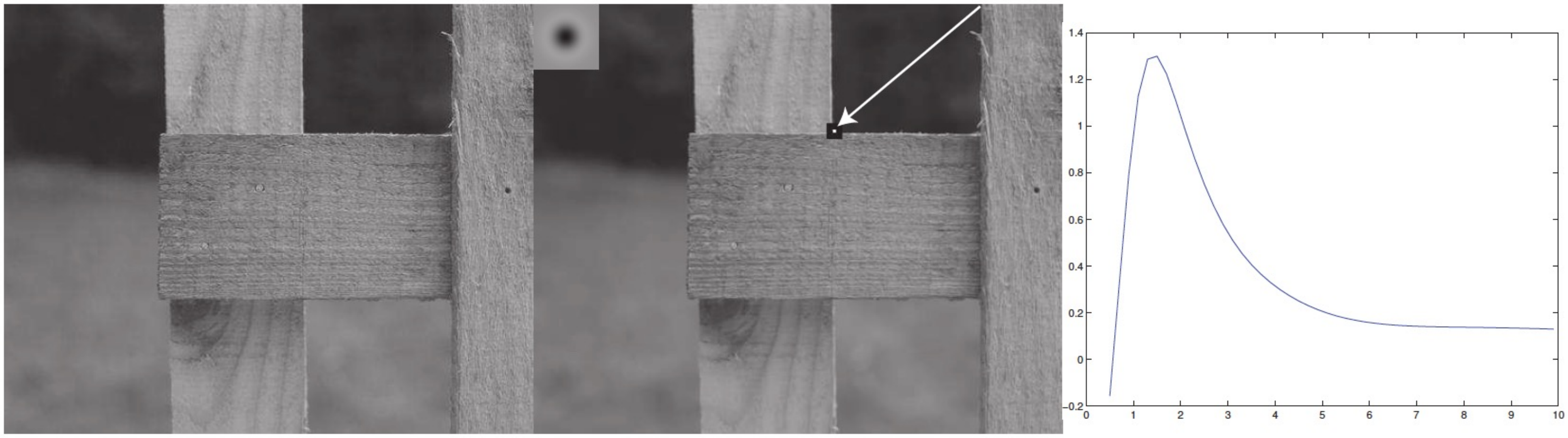
In a way that is (mostly) unaffected by image transformations

What orientation its at:

Description of contents:

Orientation histogram peak for orientation

Orientation histograms are a natural representation of image patches. However, you cannot represent orientations in image coordinates (for example, using the angle to the horizontal image axis), because the patch you are matching to might have been rotated. You need a reference orientation so all angles can be measured with respect to that reference (this is the filled gray circle in Figure 8.6). A natural reference orientation is the most common orientation in the patch. Compute a histogram of the gradient orientations in this patch, and find the largest peak. This peak is the reference orientation for the patch. If there are two or more peaks of the same magnitude, make multiple copies of the patch, one at each peak orientation.



Notice how useful a Gaussian pyramid could be here

Need to find

Where is it:

What scale its at:

In a way that is (mostly) unaffected by image transformations

What orientation its at:

Description of contents:

SIFT features

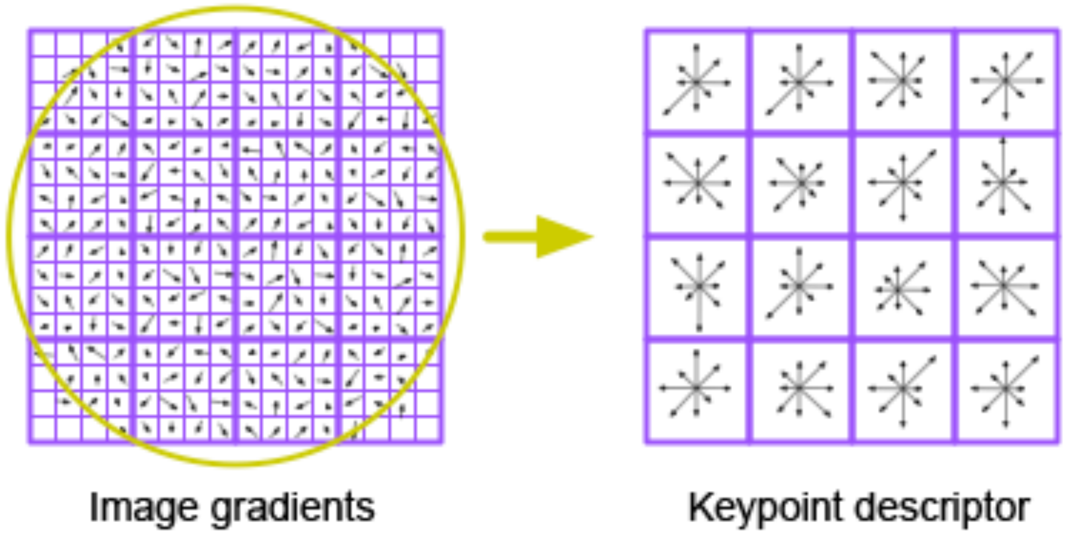
SIFT features

- SIFT=Scale Invariant Feature Transform
- Very strong record of effectiveness in matching applications
- SIFT features behave very well using nearest neighbors matching
  - i.e. the nearest neighbor to a query patch is usually a matching patch

Describing a window’s contents

We want description to be:

- Invariant to changes in image brightness: - use orientations
- Robust to noise: - ignore orientations with small magnitude
- Distinctive: - use lots of local orientations
- Invariant to small errors in location:
  - “bucket” the orientations in image
  - Use a histogram for each bucket



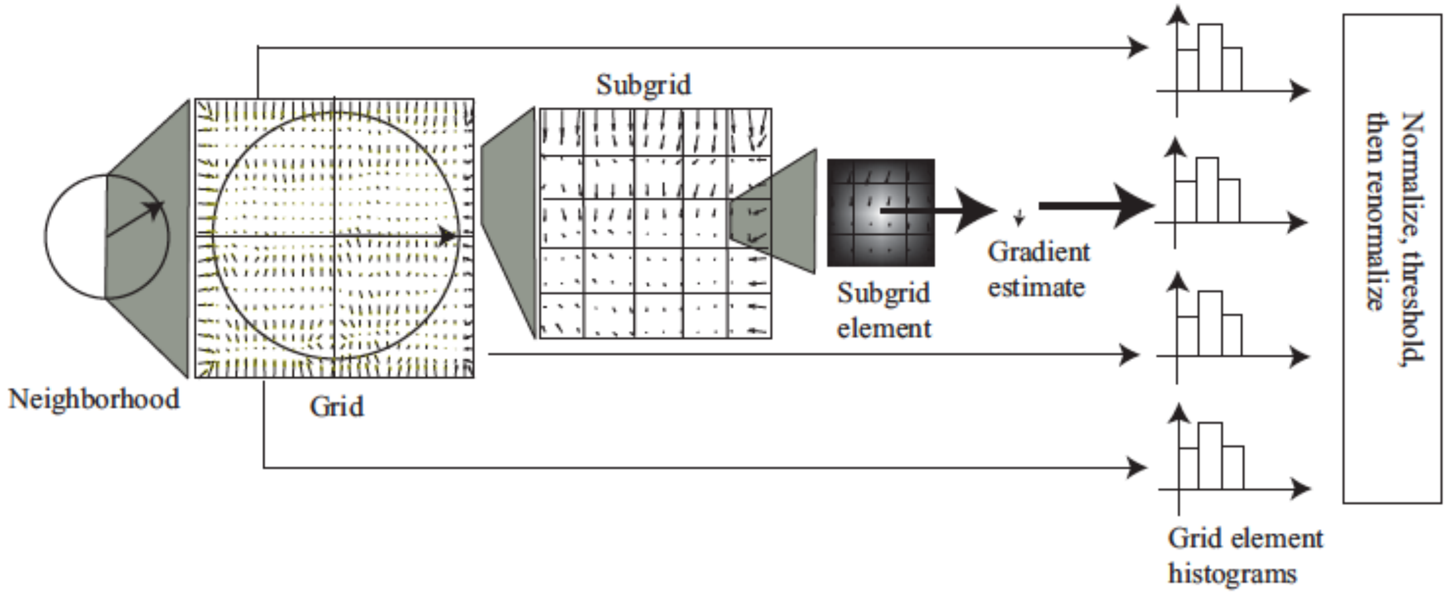
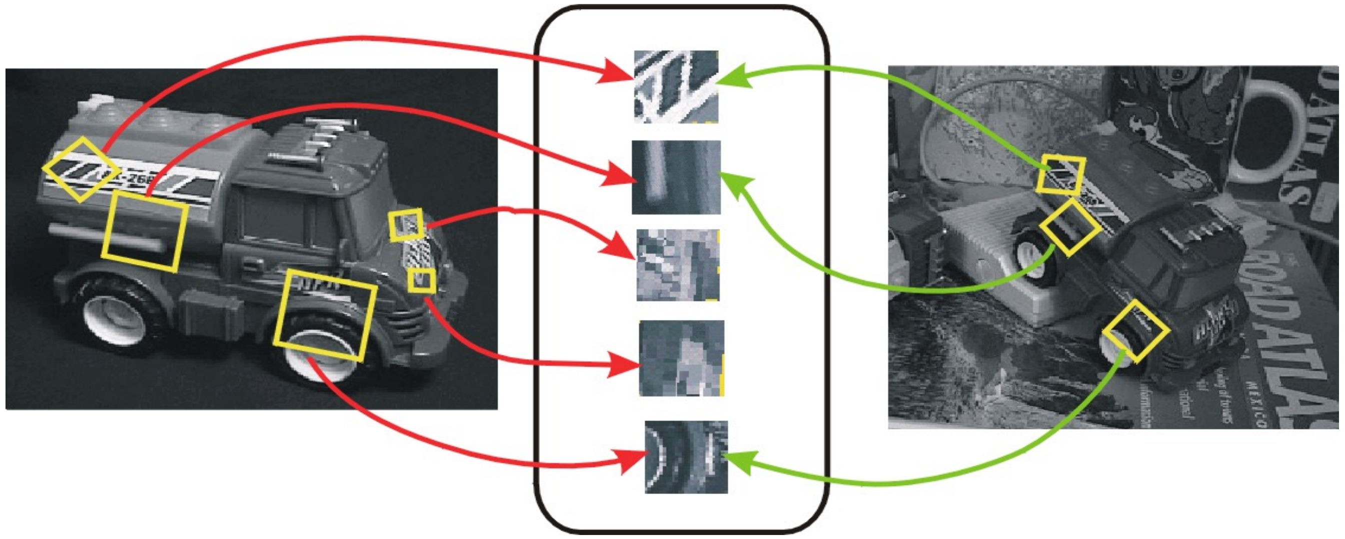


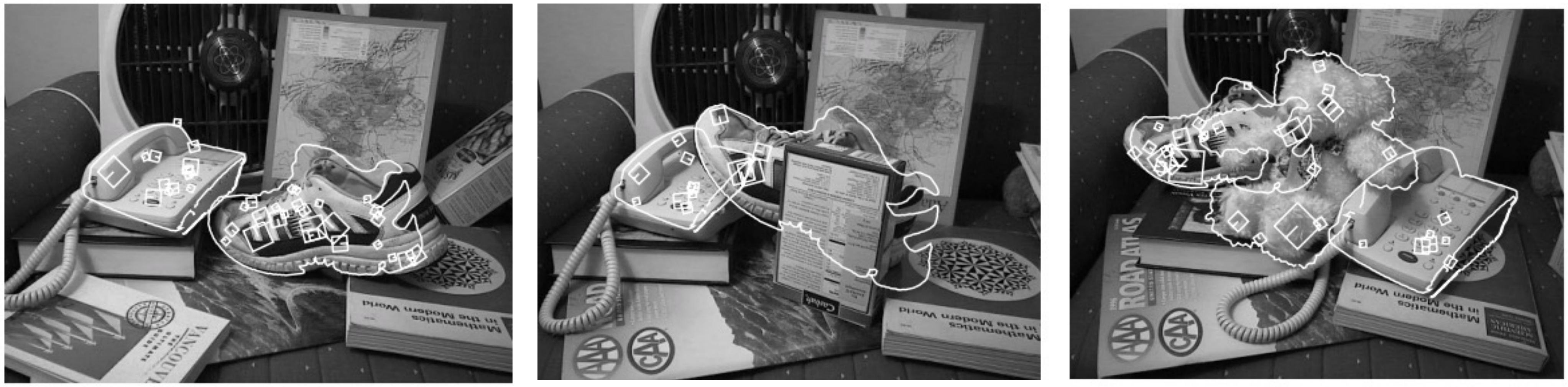
FIGURE 5.14: To construct a SIFT descriptor for a neighborhood, we place a grid over the rectified neighborhood. Each grid is divided into a subgrid, and a gradient estimate is computed at the center of each subgrid element. This gradient estimate is a weighted average of nearby gradients, with weights chosen so that gradients outside the subgrid cell contribute. The gradient estimates in each subgrid element are accumulated into an orientation histogram. Each gradient votes for its orientation, with a vote weighted by its magnitude and by its distance to the center of the neighborhood. The resulting orientation histograms are stacked to give a single feature vector. This is normalized to have unit norm; then terms in the normalized feature vector are thresholded, and the vector is normalized again.

SIFT for matching

- The main goal of SIFT is to enable image matching in the presence of significant transformations
  - To recognize the same keypoint in multiple images, we need to match appearance descriptors or "signatures" in their neighborhoods
  - Descriptors that are *locally* invariant w.r.t. scale and rotation can handle a wide range of *global* transformations



SIFT: Scale-invariant feature transform

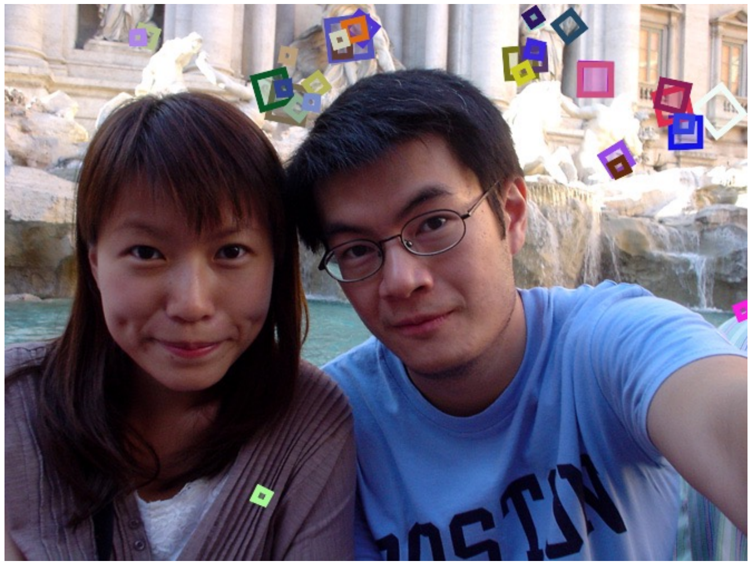


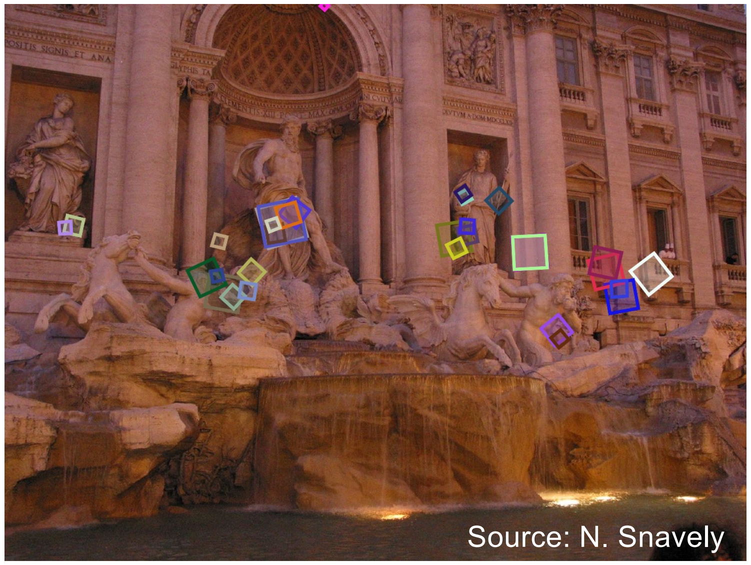
D. Lowe. Object recognition from local scale-invariant features. ICCV 1999

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

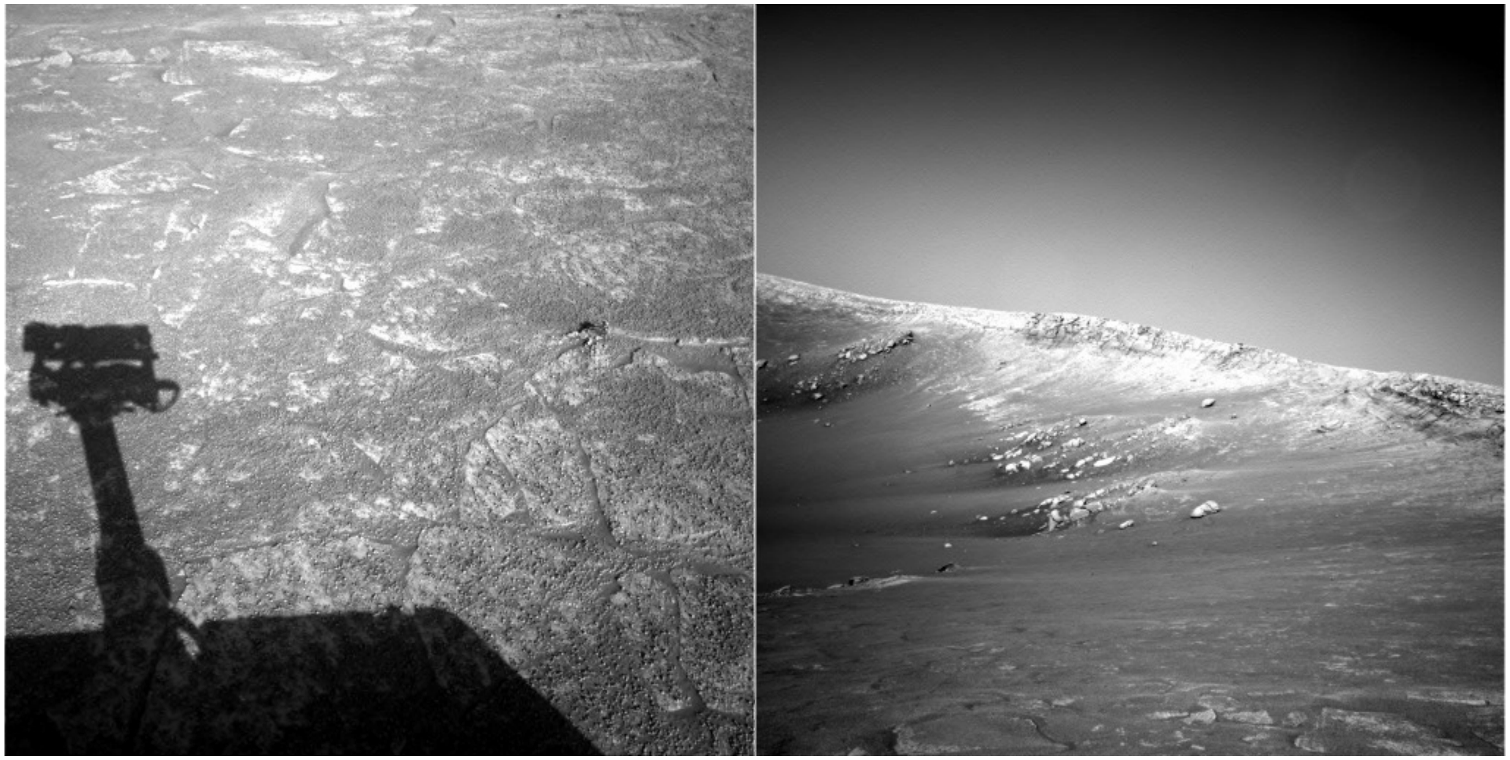
SIFT for matching

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

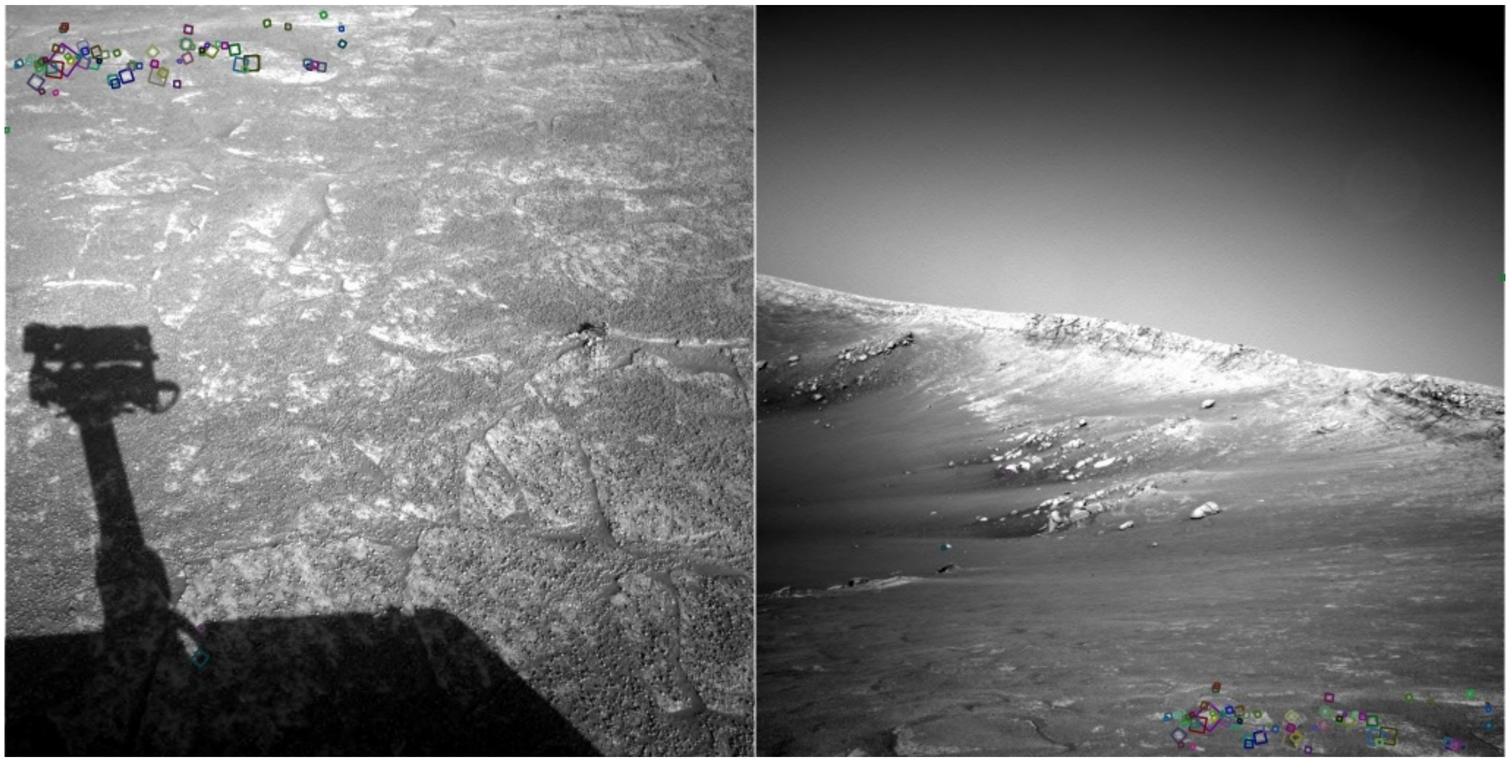




A hard matching problem


NASA Mars Rover images

Answer below (look for tiny colored squares…)



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