

Computer Vision

CS308

Feng Zheng

SUSTech CS Vision Intelligence and Perception
Week 5



南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY



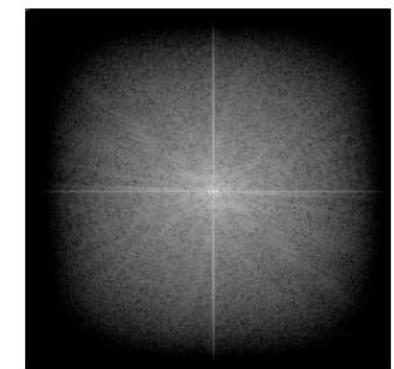
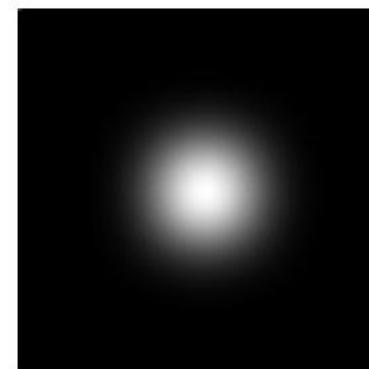
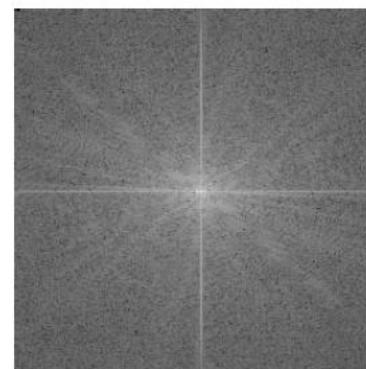
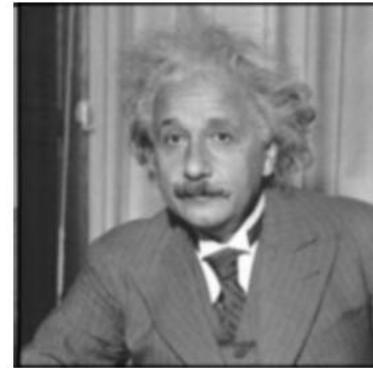
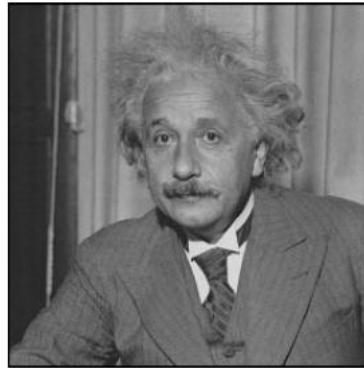
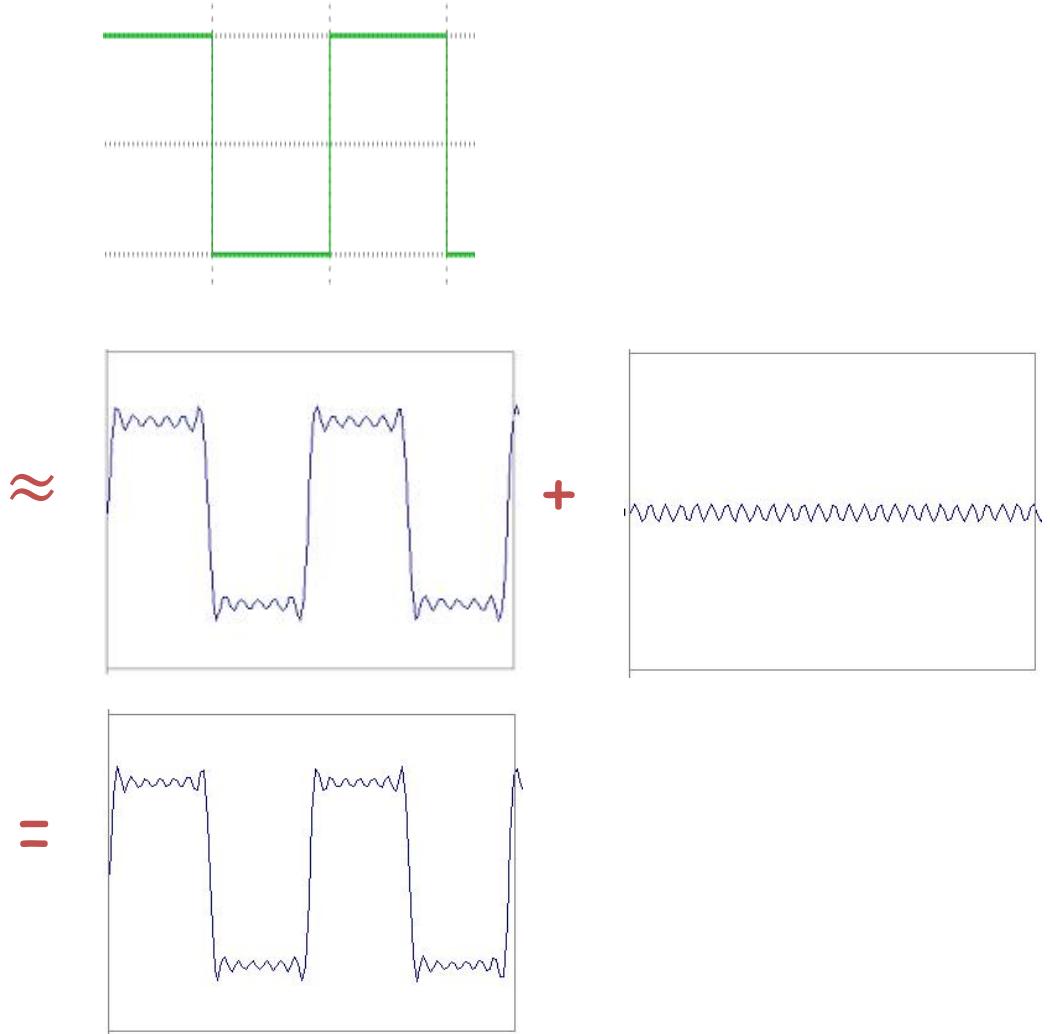
Content

- Brief Review
- Points Detection
- Points Descriptor
- Points Matching

Brief Review



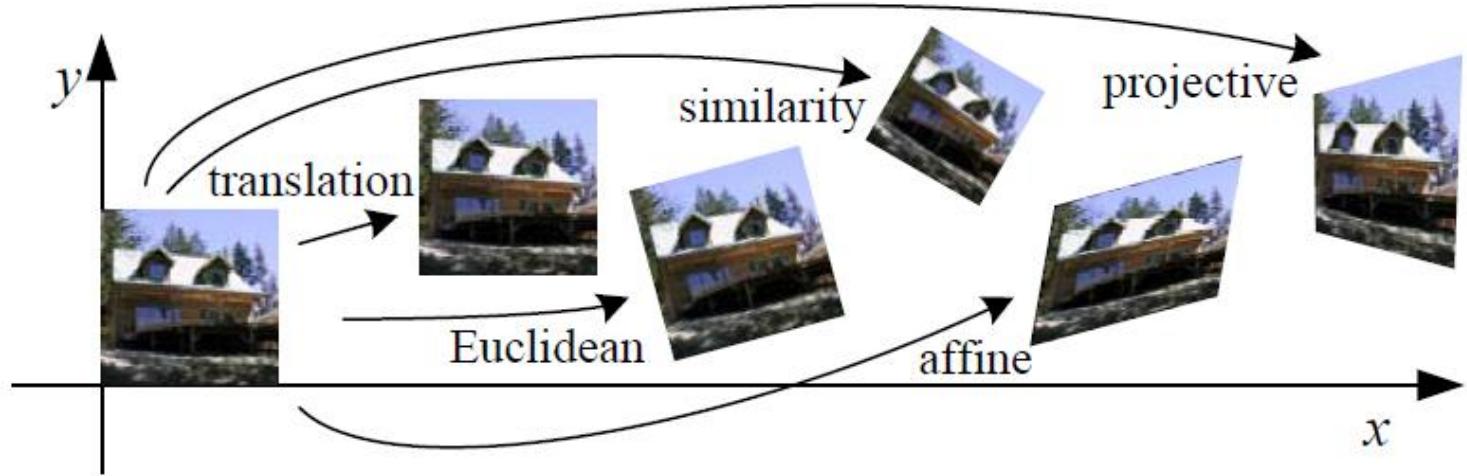
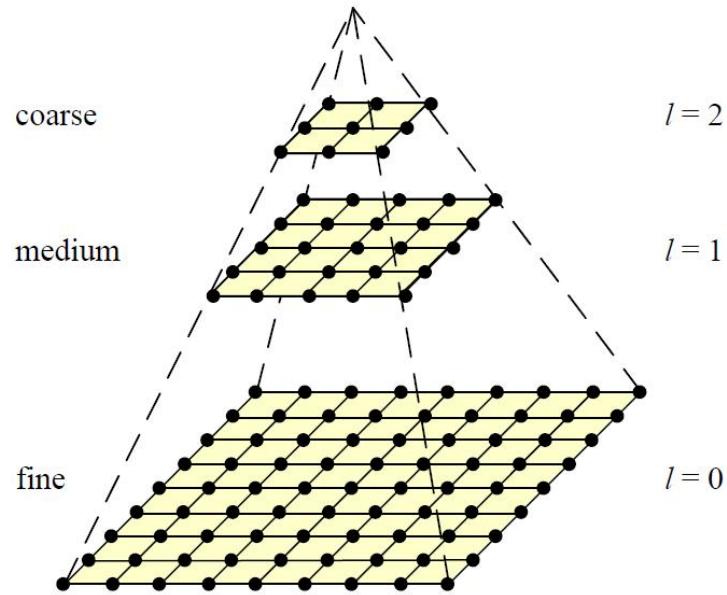
Review



Smoothing



Review

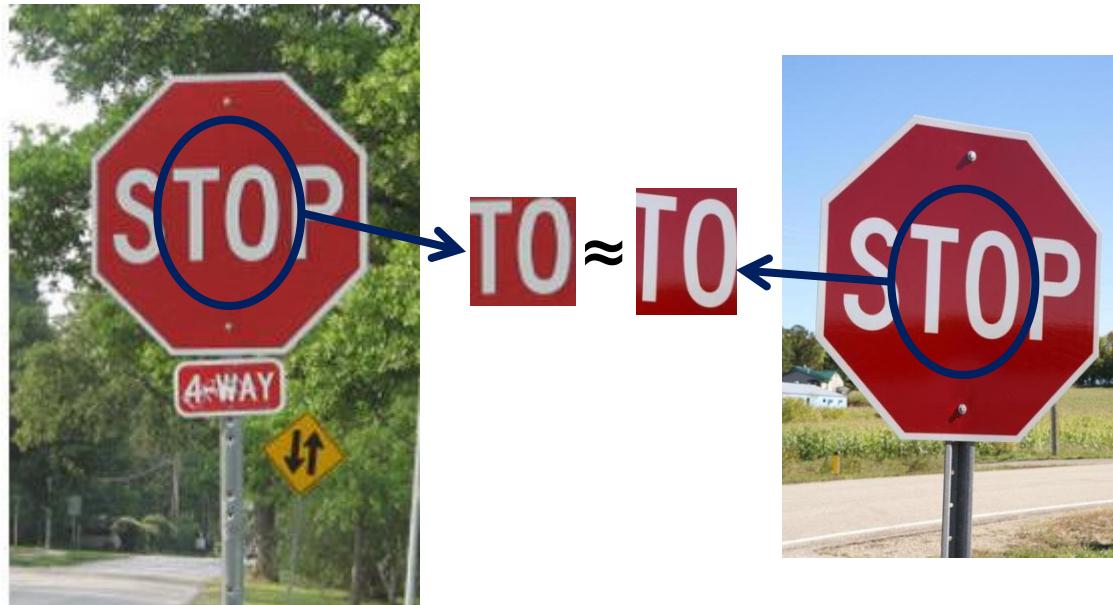


Points



Correspondence Across Views

- **Correspondence:** matching points, patches, edges, or regions across images



Example: structure from motion

How to confirm that the two images are similar with each other?



Applications

- Feature points are used for:
 - Image alignment
 - 3D reconstruction
 - Motion tracking
 - Indexing and database retrieval
 - Object recognition





An Example

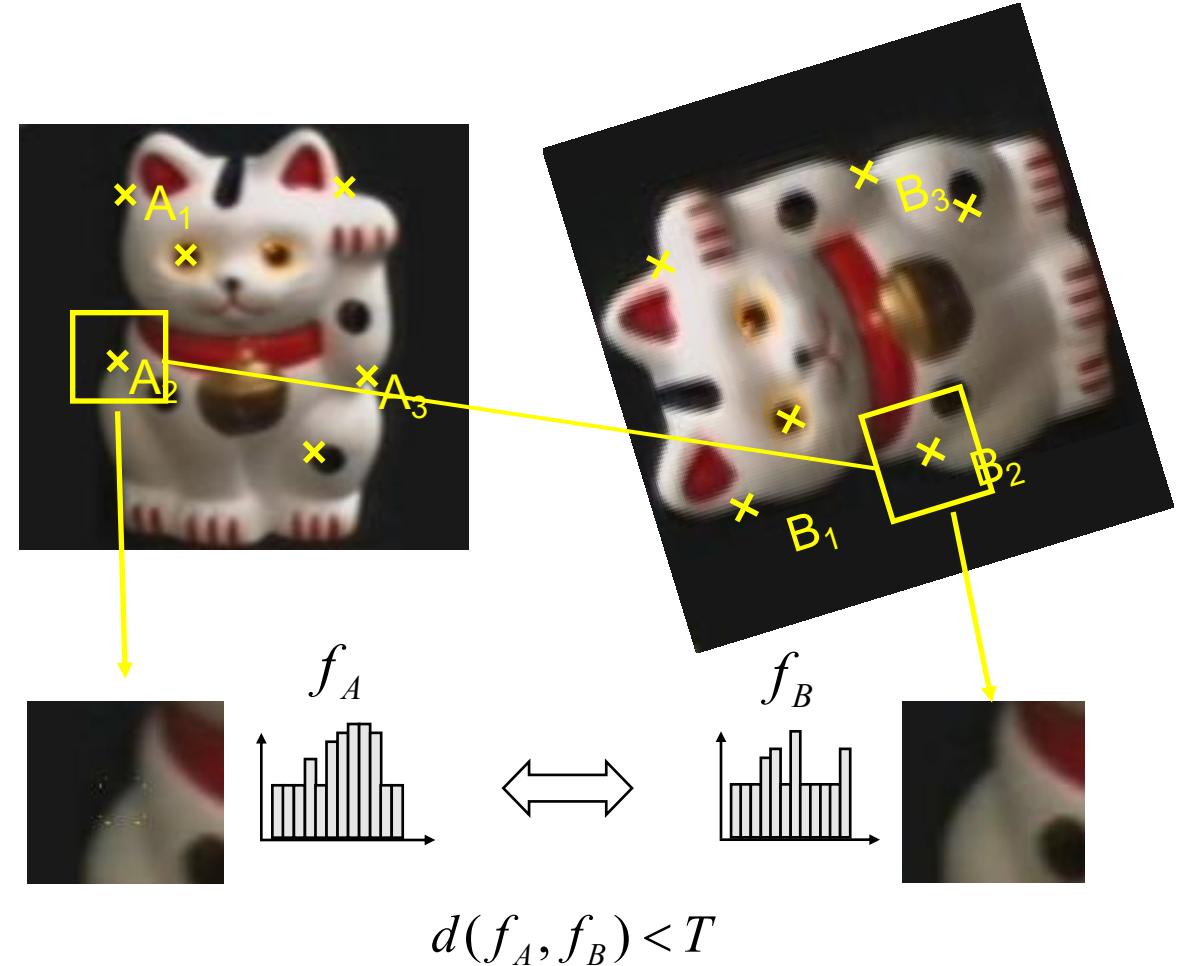
- Motivation: panorama stitching
 - We have two images - how do we combine them?





Overview of Keypoint Matching

- Steps
 - Find a set of distinctive keypoints
 - Define a **region** around each keypoint
 - Compute a local **descriptor** from the region
 - Match local descriptors
- Goals
 - Detect points that are **repeatable** and **distinctive**





Interesting Points

- Goal: interest operator **repeatability**
 - Detect (at least some of) the **same points** in both images.
 - Run the detection procedure **independently** per image

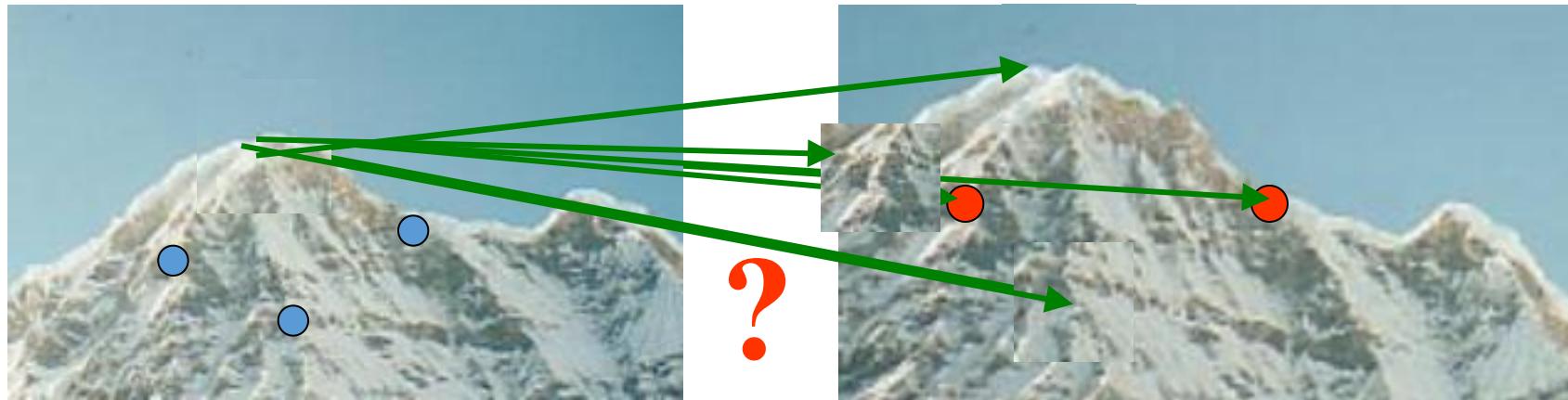


No chance to find true matches!



Interesting Points

- Goal: descriptor **distinctiveness**
 - Reliably determine which point goes with which
 - Must provide some invariance to geometric and photometric differences between the two views

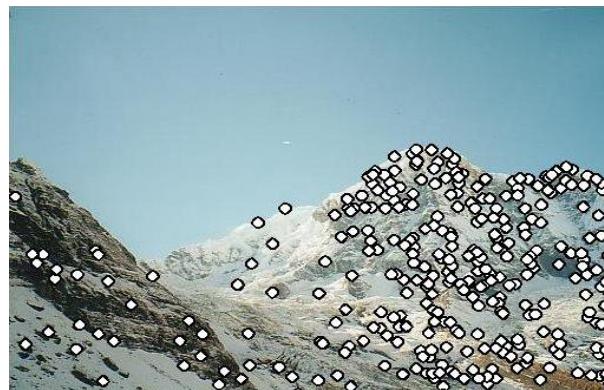




Interesting Points

- What are the characteristics of good features?

- Repeatability
 - ✓ The same feature can be found in several images despite geometric and photometric transformations
- Saliency
 - ✓ Each feature is distinctive
- Compactness and **efficiency**
 - ✓ Many fewer features than image pixels
- Locality
 - ✓ Relatively **small** area of the image
 - ✓ **Robust** to clutter and occlusion

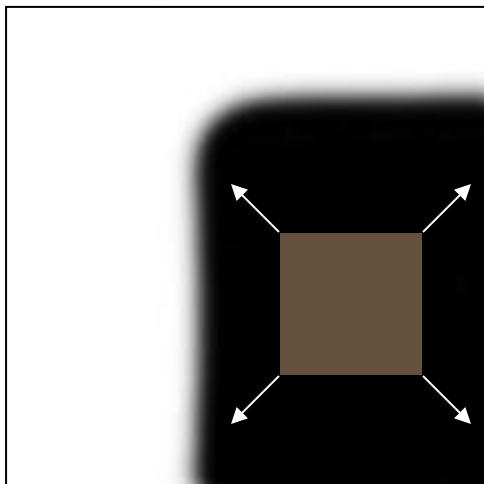


Points Detection

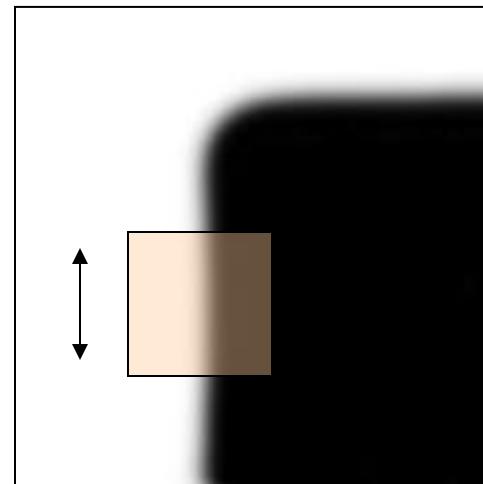


Corner Detection: Basic Idea

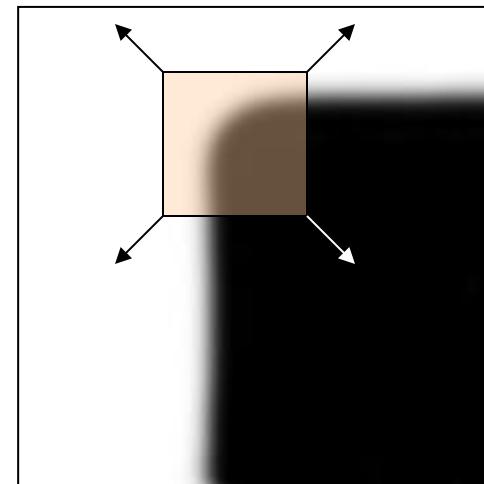
- We should **easily** recognize the point by looking through a small window
- Shifting a window in **any** direction should give a **large change** in intensity



“flat” region: no change
in all directions



“edge”: no change
along the edge direction



“corner”: significant
change in **all** directions

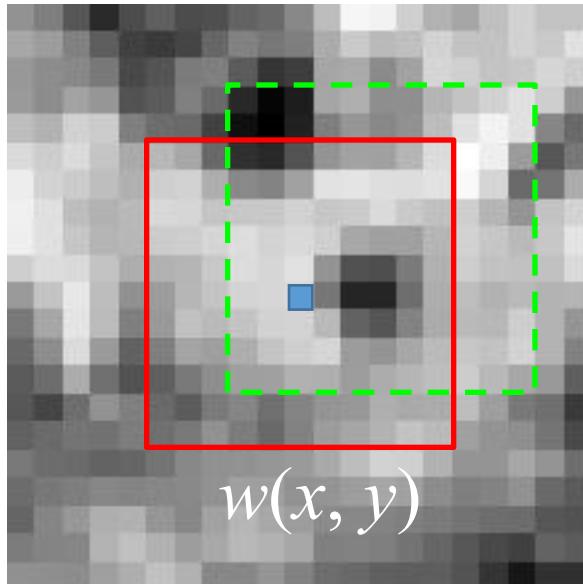


Corner Detection: Mathematics

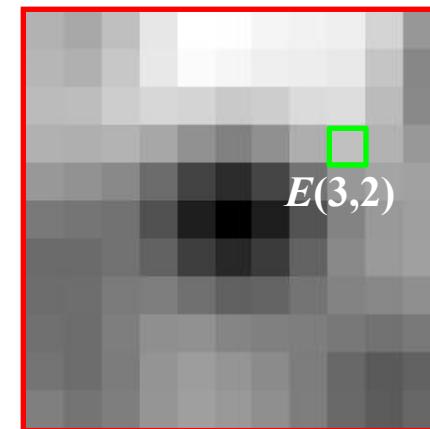
- Change in appearance of window $w(x, y)$ for the shift $[u, v]$:

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

$I(x, y)$



$E(u, v)$



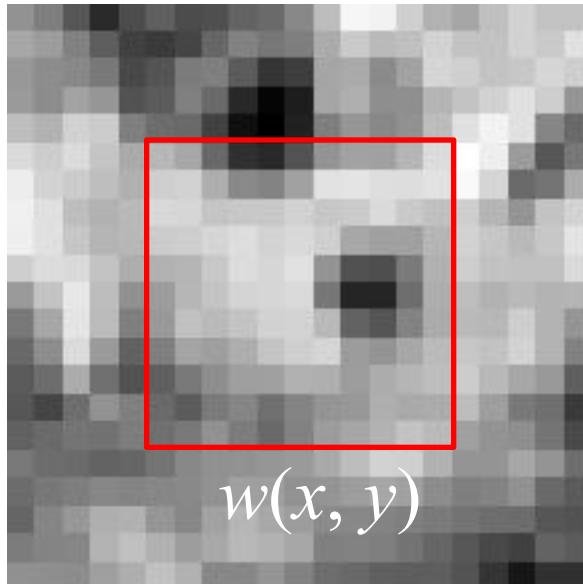


Corner Detection: Mathematics

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

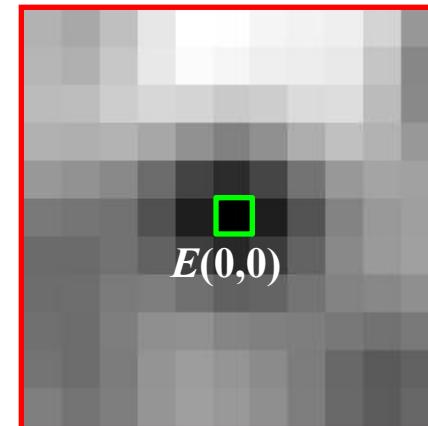
$$E(u, v) = \sum_{x, y} w(x, y) [I(x + u, y + v) - I(x, y)]^2$$

$I(x, y)$



$w(x, y)$

$E(u, v)$





Corner Detection: Mathematics

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

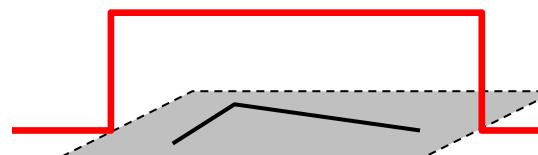
$$E(u, v) = \sum_{x, y} w(x, y) [I(x + u, y + v) - I(x, y)]^2$$

Window
function

Shifted
intensity

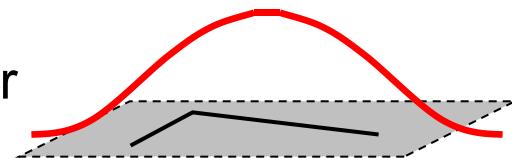
Intensity

Window function $w(x, y) =$



1 in window, 0 outside

or



Gaussian



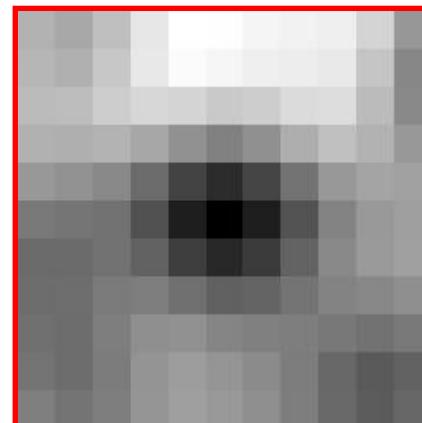
Corner Detection: Mathematics

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

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

- 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 $[u, v]$:

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



- We want to find out how this function behaves for small shifts
- But this is very slow to compute naively
 - $O(\text{window_width}^2 * \text{shift_range}^2 * \text{image_width}^2)$
 - $O(11^2 * 11^2 * 600^2) = 5.2 \text{ billion}$ of these
 - 14.6 thousand per pixel in your image

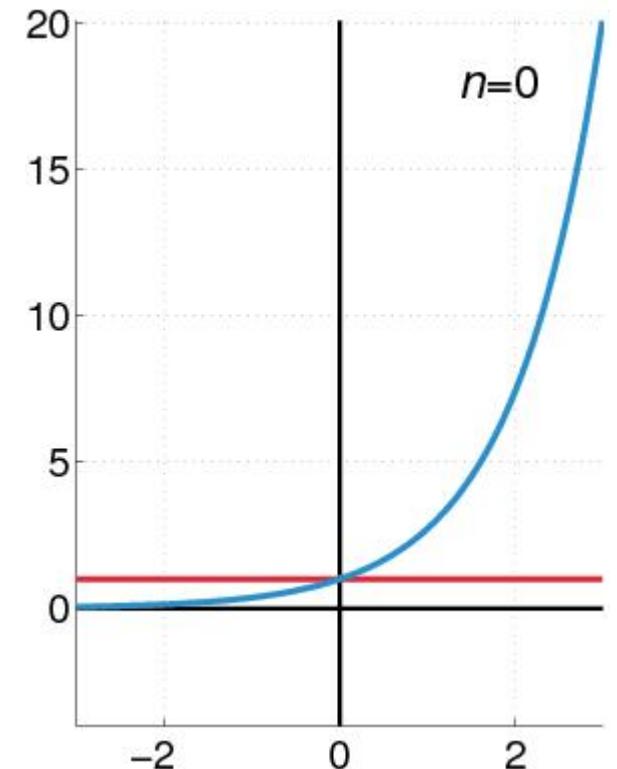


Corner Detection: Mathematics

- Recall Taylor series expansion. A function f can be approximated around point a as

$$f(a) + \frac{f'(a)}{1!}(x - a) + \frac{f''(a)}{2!}(x - a)^2 + \frac{f'''(a)}{3!}(x - a)^3 + \dots$$

Approximation of
 $f(x) = e^x$
centered at $f(0)$





Corner Detection: Mathematics

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

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

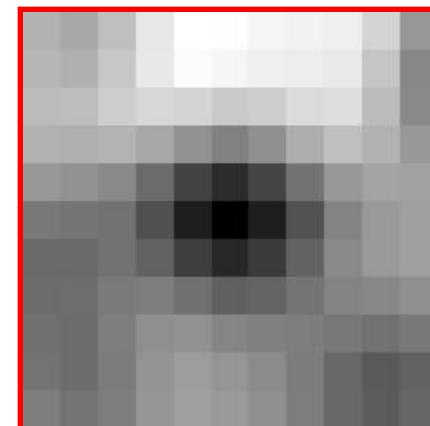


Always 0



First
derivative
is 0

$E(u, v)$





Corner Detection: Mathematics

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

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

$$E(u, v) \approx E(0,0)$$

$$+ [u \quad v] \begin{bmatrix} E_u(0,0) \\ E_v(0,0) \end{bmatrix}$$

$$+ \frac{1}{2} [u \quad v] \begin{bmatrix} E_{uu}(0,0) & E_{uv}(0,0) \\ E_{uv}(0,0) & E_{vv}(0,0) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

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

$$E_{uu}(u, v) = \sum_{x,y} 2w(x, y) I_x(x+u, y+v) I_x(x+u, y+v) \\ + \sum_{x,y} 2w(x, y) [I(x+u, y+v) - I(x, y)] I_{xx}(x+u, y+v)$$

$$E_{uv}(u, v) = \sum_{x,y} 2w(x, y) I_y(x+u, y+v) I_x(x+u, y+v) \\ + \sum_{x,y} 2w(x, y) [I(x+u, y+v) - I(x, y)] I_{xy}(x+u, y+v)$$



Corner Detection: Mathematics

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

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

$$E(u, v) \approx E(0,0)$$

$$+ [u \quad v] \begin{bmatrix} E_u(0,0) \\ E_v(0,0) \end{bmatrix}$$

$$+ \frac{1}{2} [u \quad v] \begin{bmatrix} E_{uu}(0,0) & E_{uv}(0,0) \\ E_{uv}(0,0) & E_{vv}(0,0) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

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

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

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

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

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

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



Corner Detection: Mathematics

- The quadratic approximation simplifies to

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

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

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

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

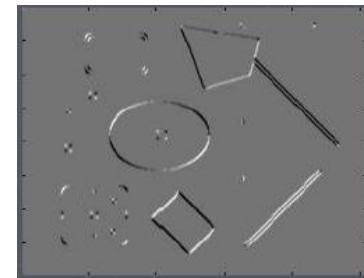
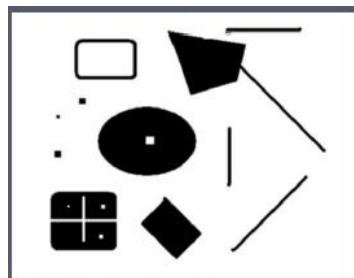


Corner Detection: Mathematics

- **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:

$$I_x \Leftrightarrow \frac{\partial I}{\partial x}$$

$$I_y \Leftrightarrow \frac{\partial I}{\partial y}$$

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

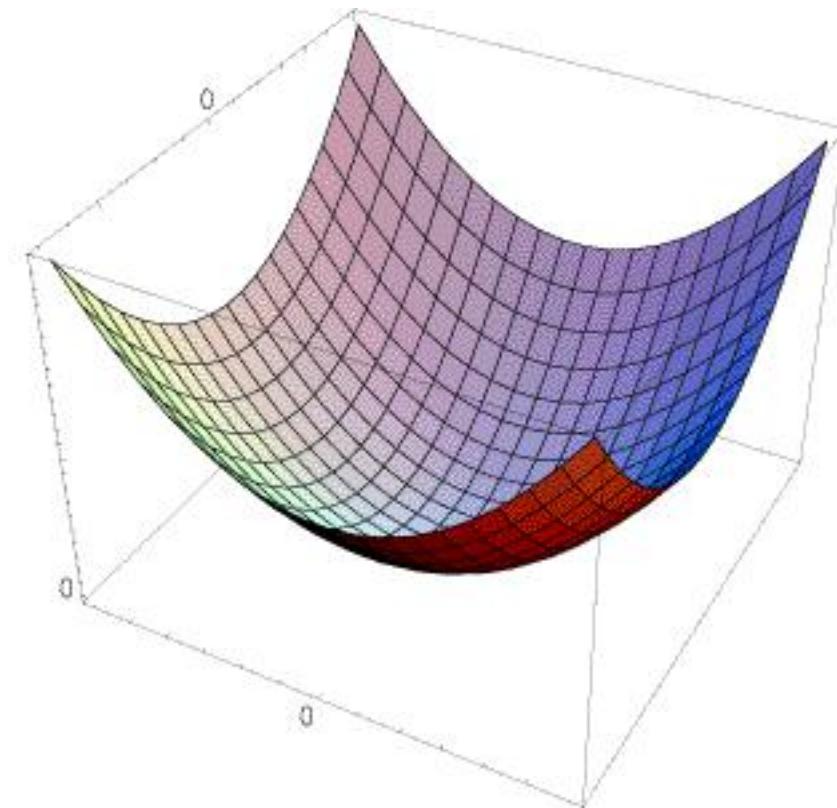


Corner Detection: Mathematics

- Interpreting the second moment matrix
 - The surface $E(u,v)$ is locally approximated by a quadratic form
 - Let's try to understand its shape

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

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



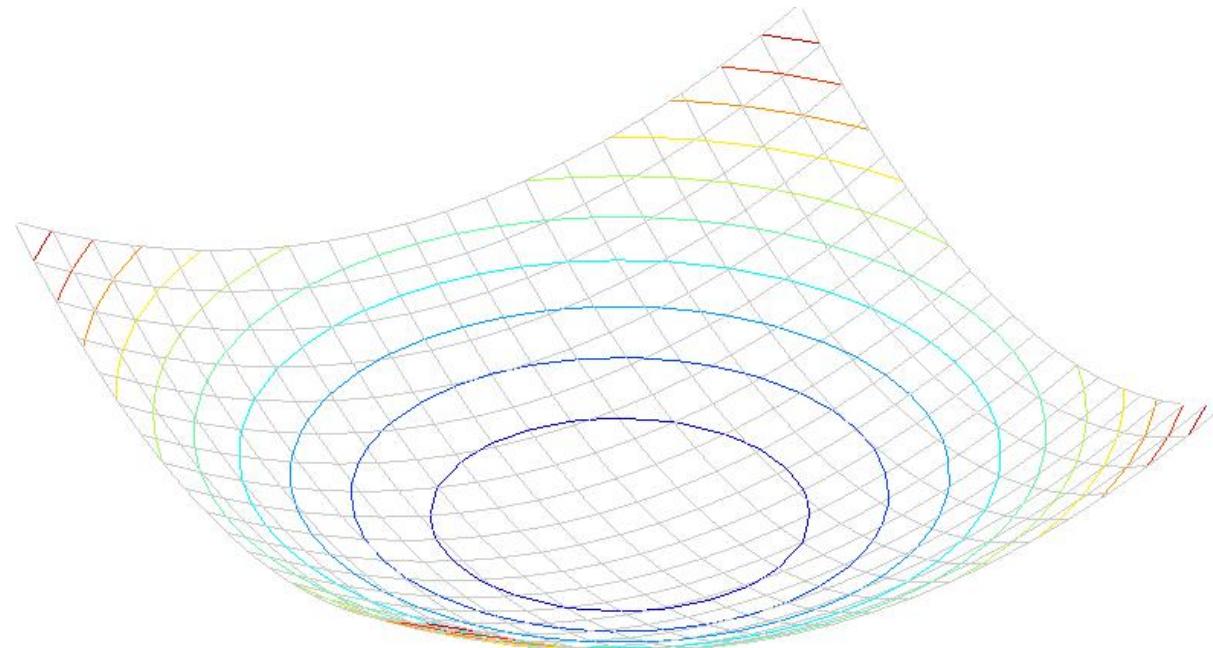


Corner Detection: Mathematics

- Interpreting the second moment matrix

- Consider a horizontal "slice" of $E(u, v)$:
- This is the equation of an ellipse

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





Corner Detection: Mathematics

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

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

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

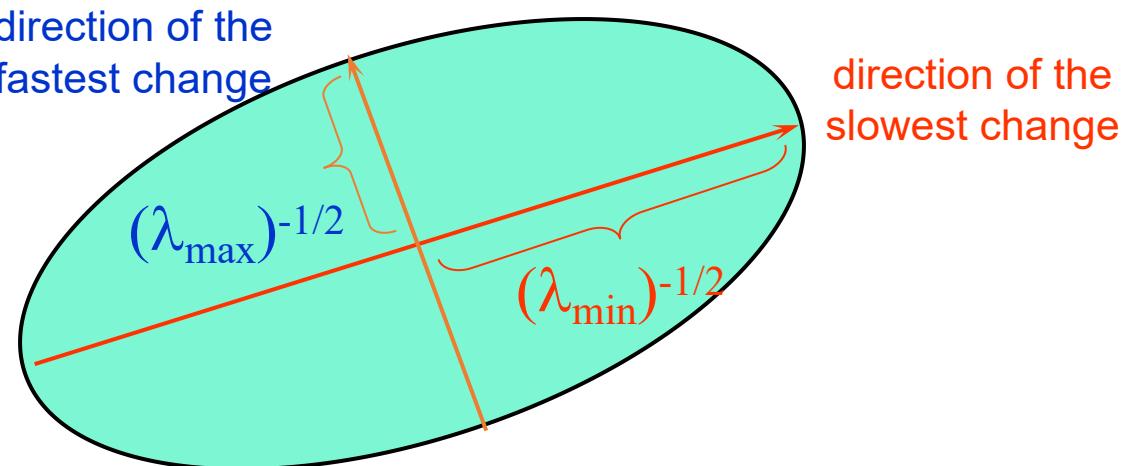


Corner Detection: Mathematics

- Interpreting the second moment matrix
 - Consider a horizontal "slice" of $E(u, v)$:
 - This is the equation of an ellipse
- Diagonalization of M :
 - The axis lengths of the ellipse are determined by the eigenvalues and the orientation is determined by R

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

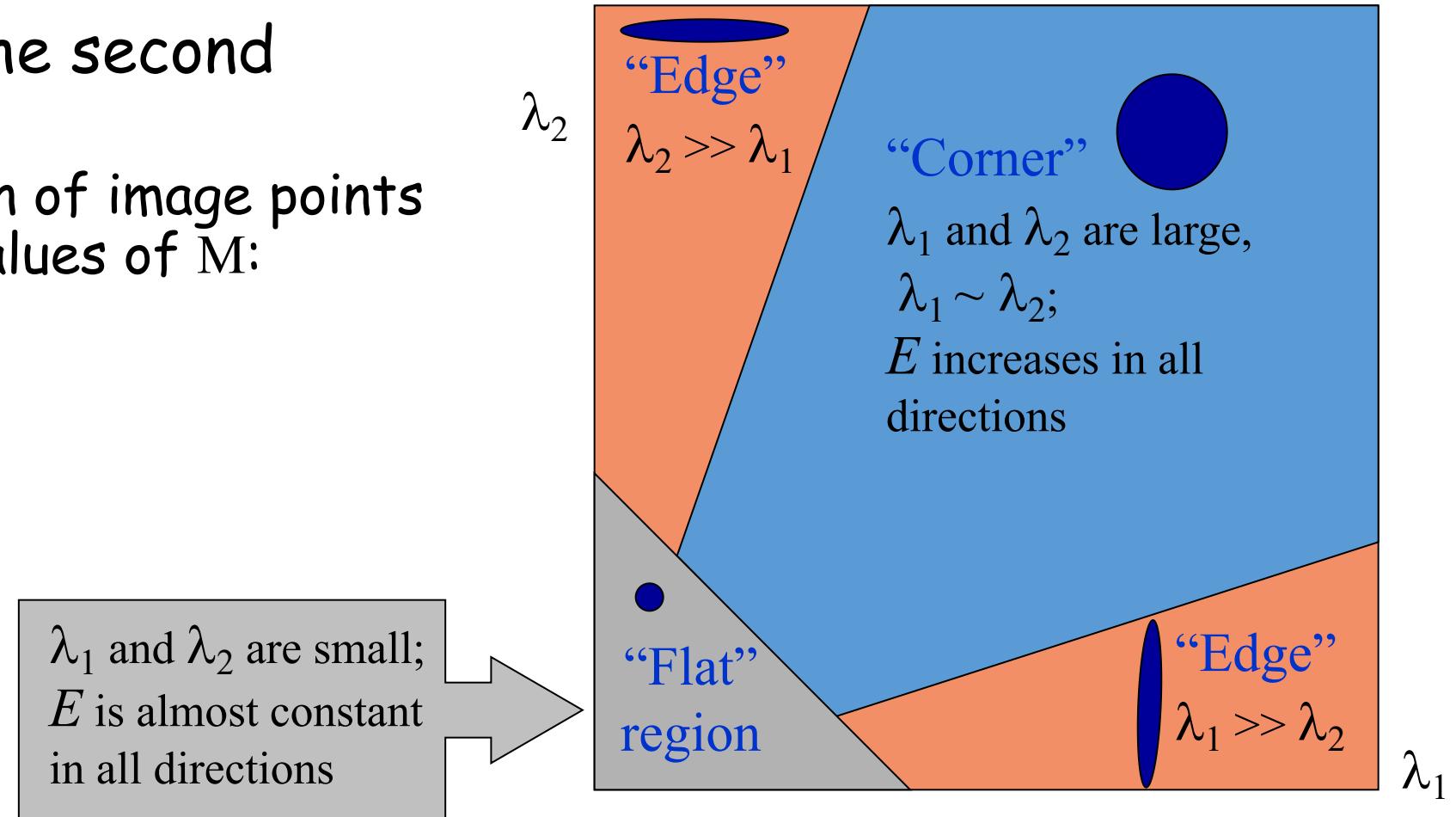
$$M = R^{-1} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} R$$





Corner Detection: Mathematics

- Interpreting the second moment matrix
 - Classification of image points using eigenvalues of M :



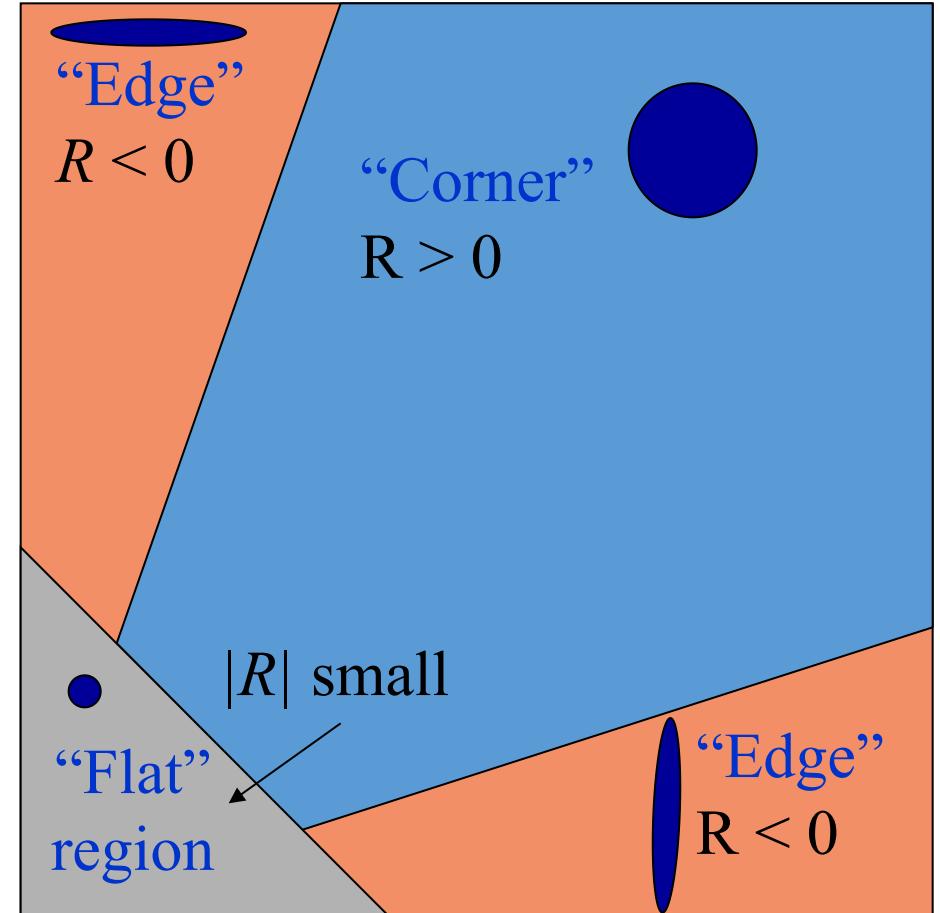


Corner Detection: Mathematics

- Corner response function

α : constant (0.04 to 0.06)

$$\begin{aligned} R &= \det(M) - \alpha \operatorname{trace}(M)^2 \\ &= \lambda_1 \lambda_2 - \alpha(\lambda_1 + \lambda_2)^2 \end{aligned}$$





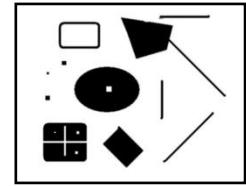
Harris Corner Detector

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

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



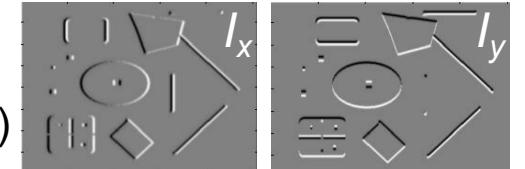
Harris Corner Detector: Steps



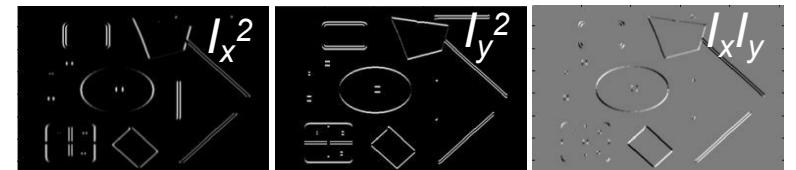
- Second moment matrix

$$\mu(\sigma_I, \sigma_D) = g(\sigma_I) * \begin{bmatrix} I_x^2(\sigma_D) & I_x I_y(\sigma_D) \\ I_x I_y(\sigma_D) & I_y^2(\sigma_D) \end{bmatrix}$$

1. Image derivatives
(optionally, blur first)



2. Square of derivatives



3. Gaussian filter g(\sigma_I)



4. Cornerness function – both eigenvalues are strong

$$har = \det[\mu(\sigma_I, \sigma_D)] - \alpha [\text{trace}(\mu(\sigma_I, \sigma_D))^2] =$$

$$g(I_x^2)g(I_y^2) - [g(I_x I_y)]^2 - \alpha[g(I_x^2) + g(I_y^2)]^2$$

5. Non-maxima suppression



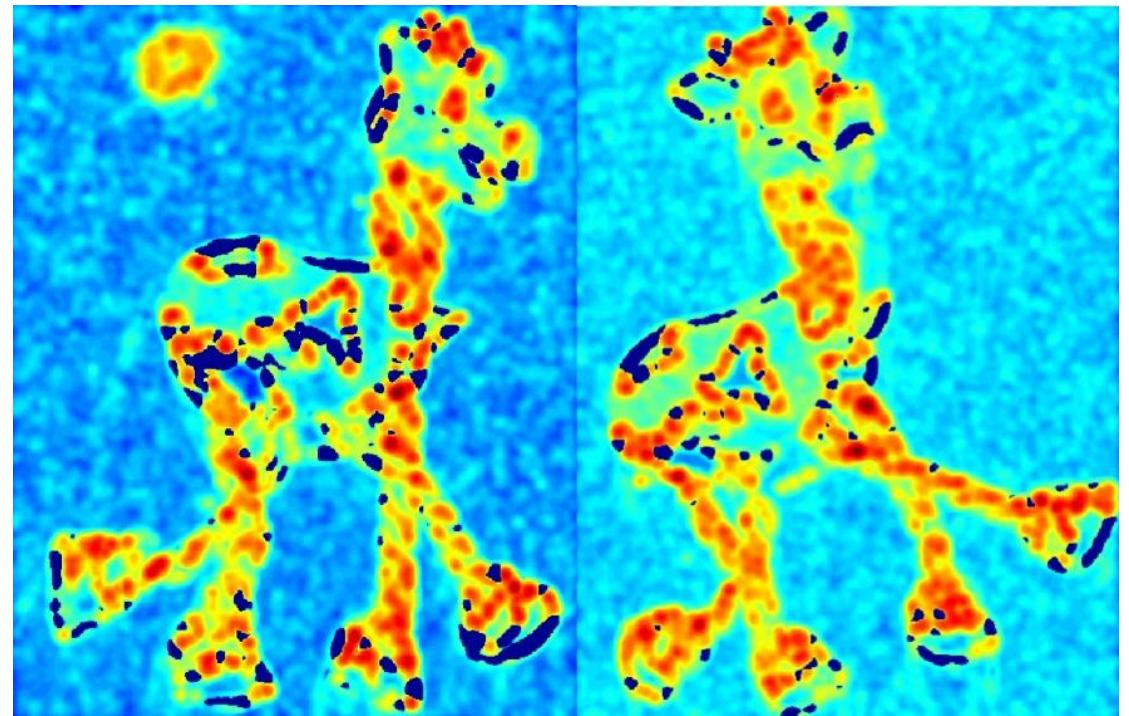


Harris Detector: Example

- Compute corner response R
 - $R_{\text{harris}}(x,y) = \det(M) - a * \text{trace}(M)$



Repeatability





Harris Detector: Example

- Find points with **large corner response**: $R > \text{threshold}$





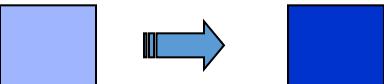
Invariance and Covariance

- We want corner locations to be invariant to **photometric** transformations and covariant to **geometric** transformations
 - Invariance: image is transformed and corner locations do not change
 - Covariance: if we have two transformed versions of the same image, features should be detected in **corresponding** locations

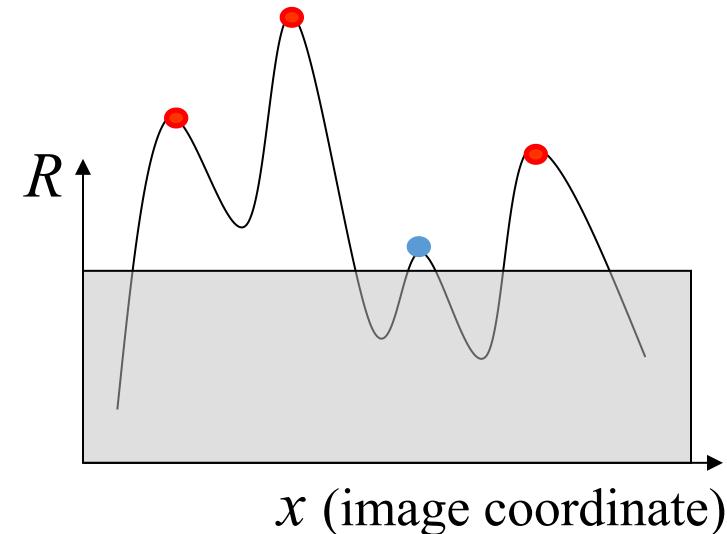
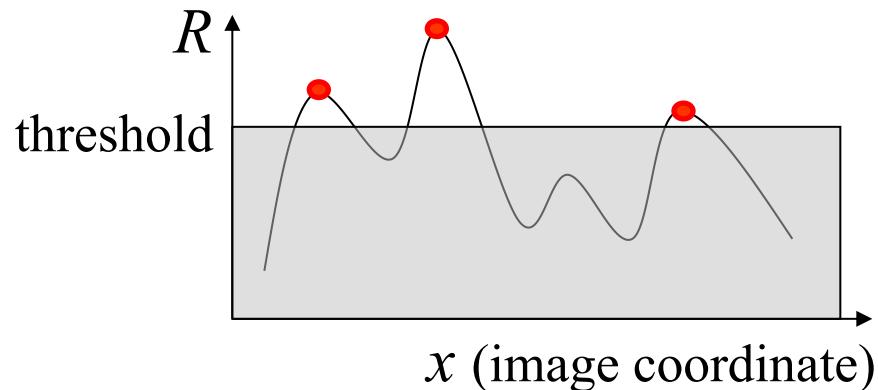




Affine Intensity Change

- Intensity change  $I \rightarrow a I + b$

- Only derivatives are used \Rightarrow invariance to intensity shift: $I \rightarrow I + b$
- Intensity scaling: $I \rightarrow a I$

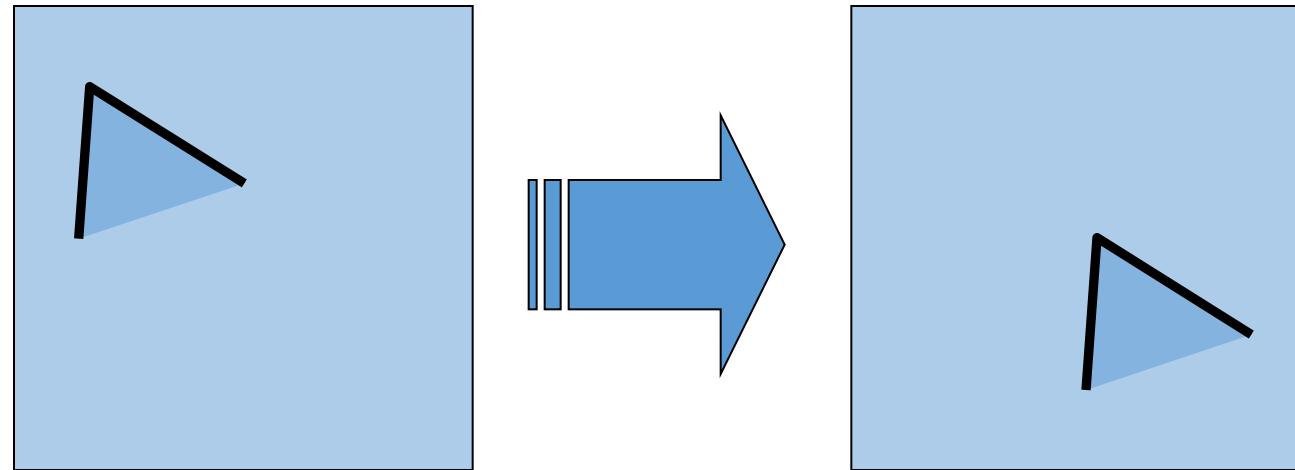


Partially invariant to affine intensity change



Image Translation

- Derivatives and window function are shift-invariant

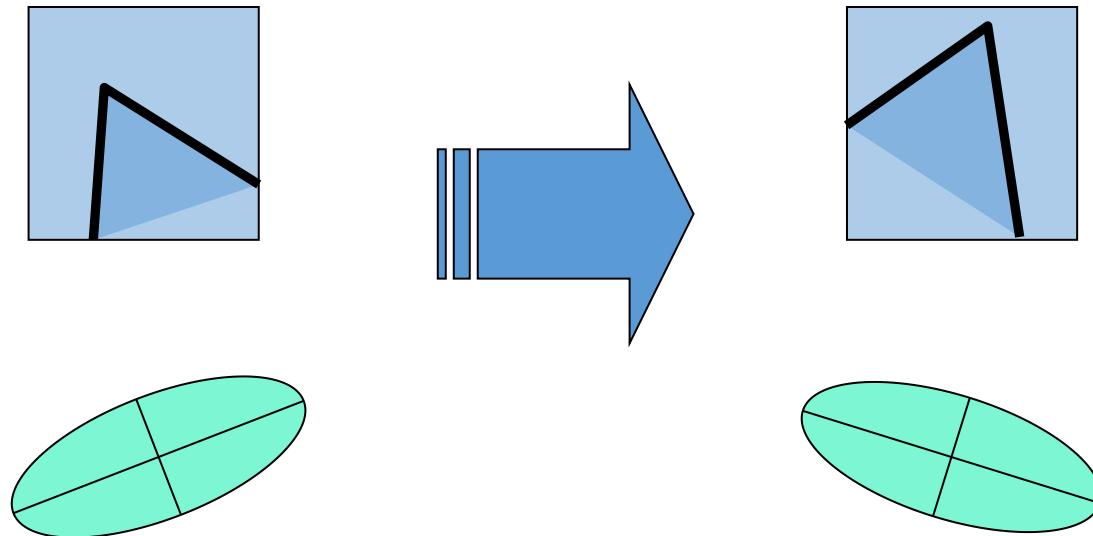


Corner location is covariant w.r.t. translation



Image Rotation

- Second moment ellipse rotates but its shape (i.e. eigenvalues) remains the same



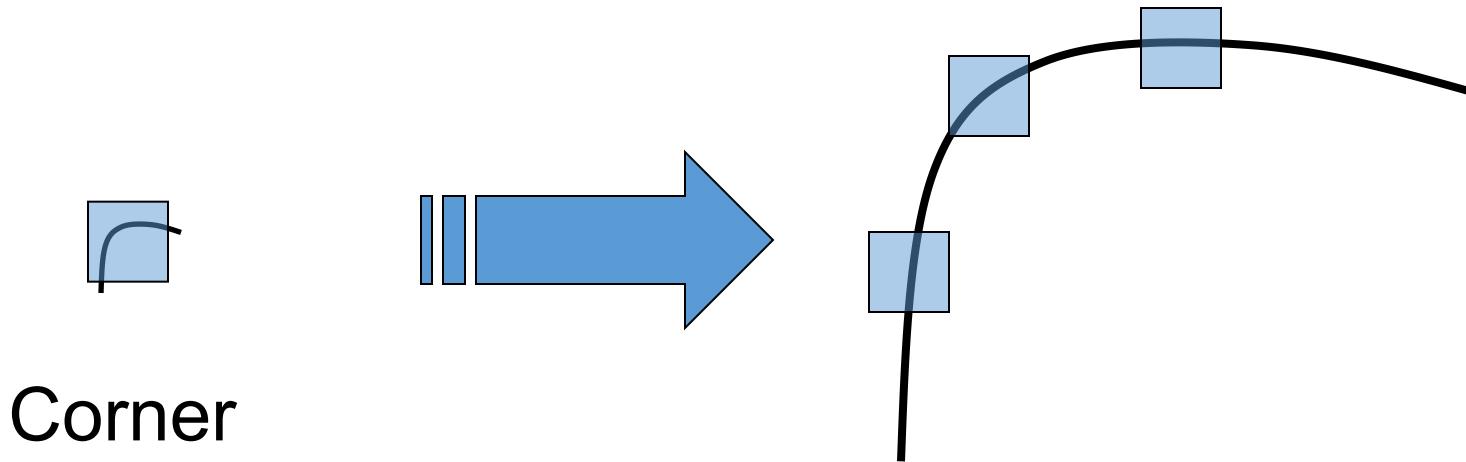
Corner location is covariant w.r.t. rotation



Scaling

- All points will be classified as edges

A Problem!



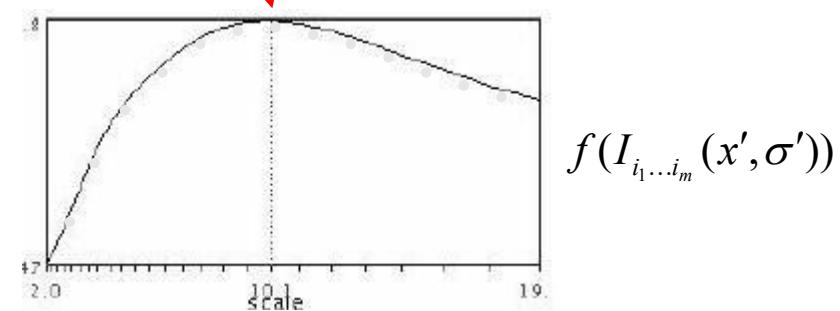
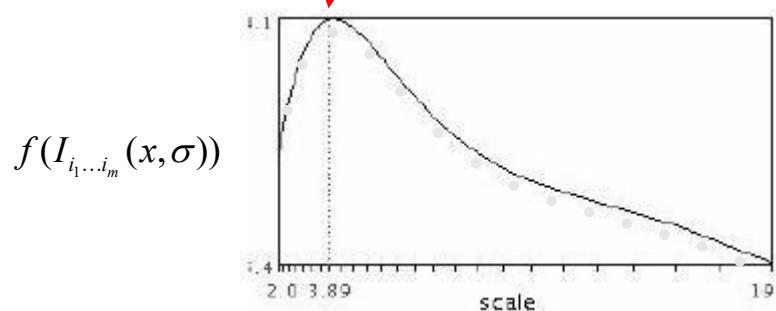
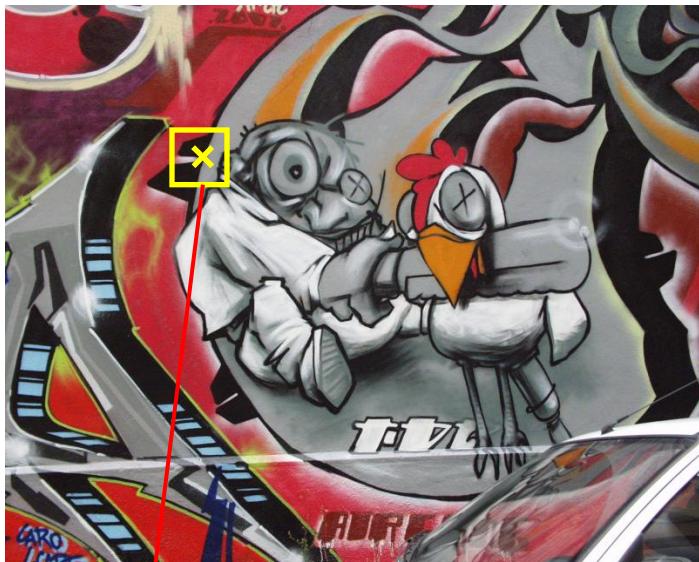
Corner

Corner location is **not** covariant to scaling!



Automatic Scale Selection

- Function responses for increasing scale (**scale** signature)





Difference-of-Gaussian (DoG)

- Laplace Operator

$$\Delta = \nabla \cdot \nabla = \nabla^2 = \begin{bmatrix} \frac{\partial}{\partial x} & \frac{\partial}{\partial y} \end{bmatrix} \begin{bmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{bmatrix} = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$$

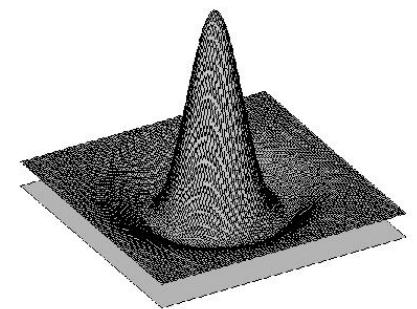
- Laplacian of Gaussian (LoG)

$$\Delta[G_\sigma(x, y) * f(x, y)] = [\Delta G_\sigma(x, y)] * f(x, y) = LoG * f(x, y) \quad LoG \triangleq \Delta G_\sigma(x, y) = \frac{\partial^2}{\partial x^2} G_\sigma(x, y) + \frac{\partial^2}{\partial y^2} G_\sigma(x, y) = \frac{x^2 + y^2 - 2\sigma^2}{\sigma^4} e^{-(x^2+y^2)/2\sigma^2}$$

- Difference of Gaussian (DoG)

$$g_1(x, y) - g_2(x, y) = G_{\sigma_1} * f(x, y) - G_{\sigma_2} * f(x, y) = (G_{\sigma_1} - G_{\sigma_2}) * f(x, y) = DoG * f(x, y)$$

$$DoG \triangleq G_{\sigma_1} - G_{\sigma_2} = \frac{1}{\sqrt{2\pi}} \left(\frac{1}{\sigma_1} e^{-(x^2+y^2)/2\sigma_1^2} - \frac{1}{\sigma_2} e^{-(x^2+y^2)/2\sigma_2^2} \right)$$



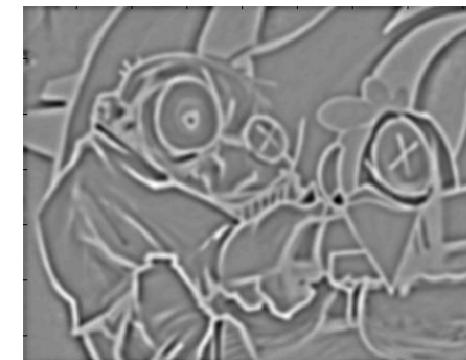
$$g_1(x, y) = G_{\sigma_1}(x, y) * f(x, y)$$



$$g_2(x, y) = G_{\sigma_2}(x, y) * f(x, y)$$

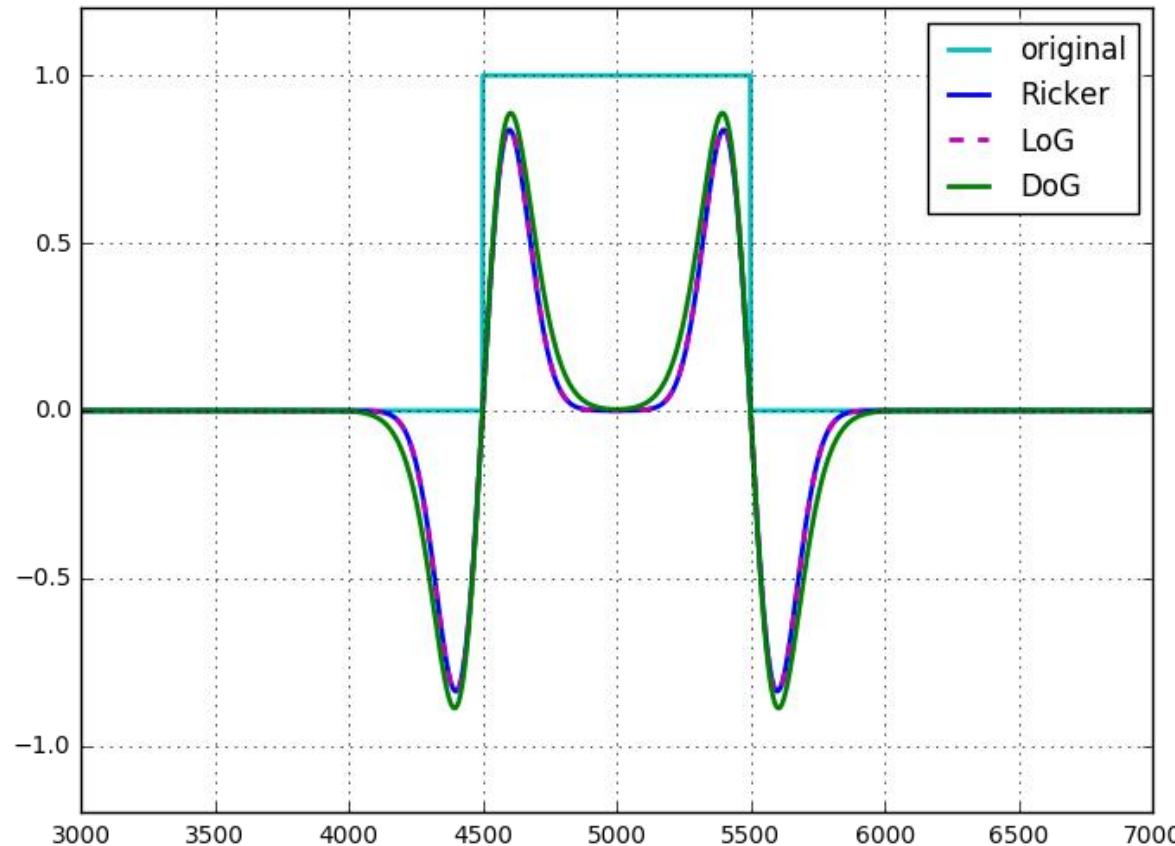


-





Difference-of-Gaussian (DoG)



<https://dsp.stackexchange.com/questions/37673/what-is-the-difference-between-difference-of-gaussian-laplace-of-gaussian-and>



DoG – Efficient Computation

- Computation in Gaussian scale **pyramid**

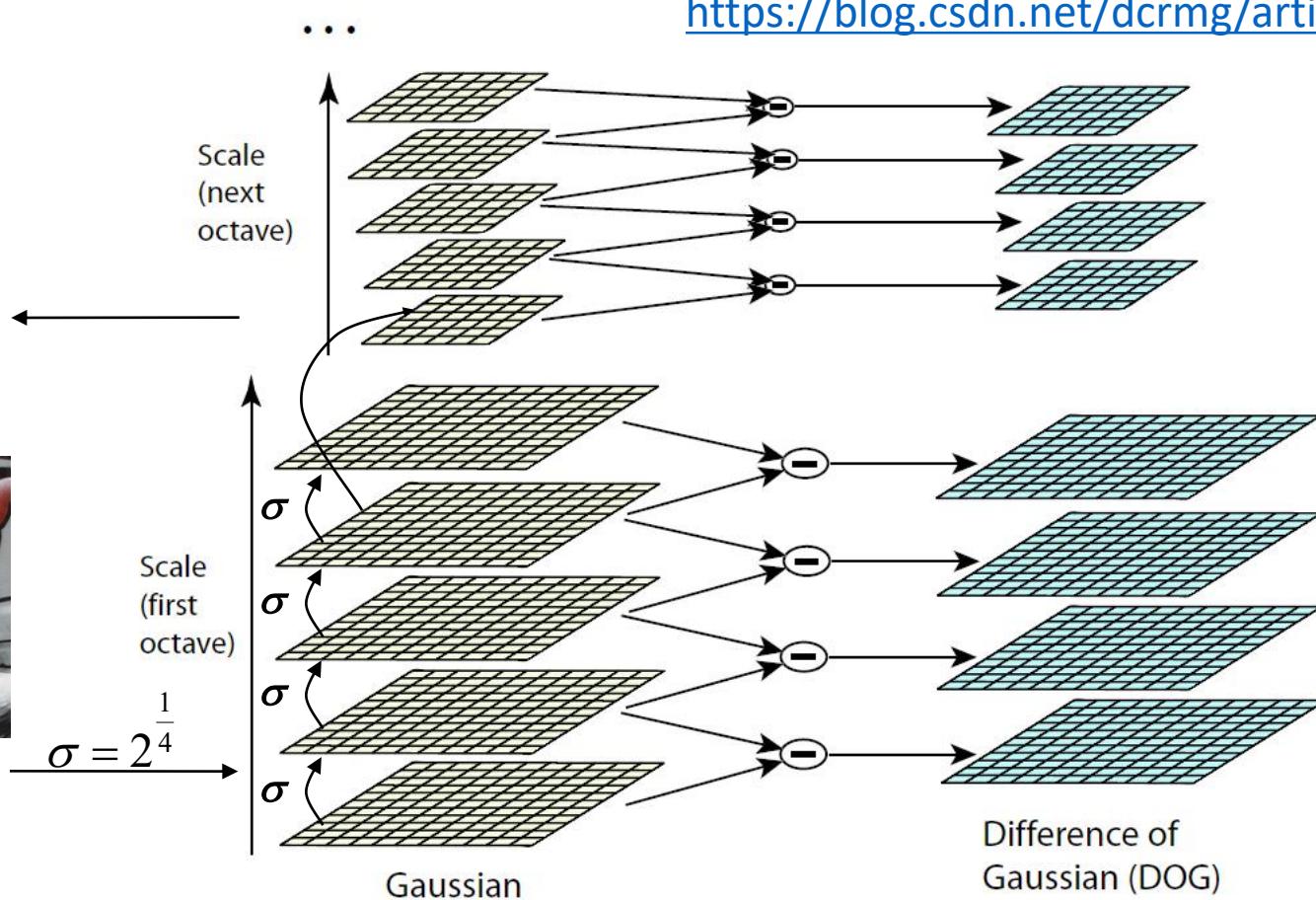
<https://blog.csdn.net/dcrmg/article/details/52561656>



*Sampling with
step $\sigma^4 = 2$*



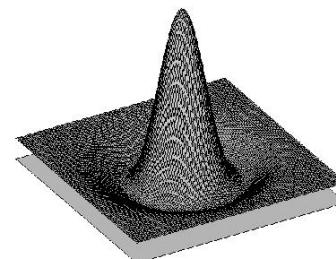
Original image



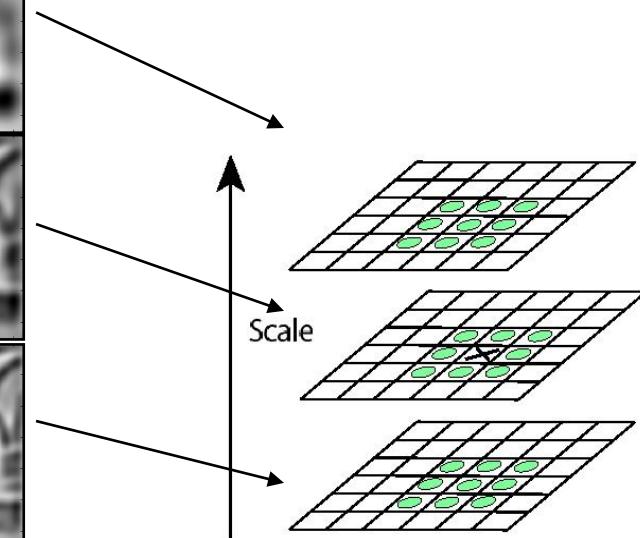
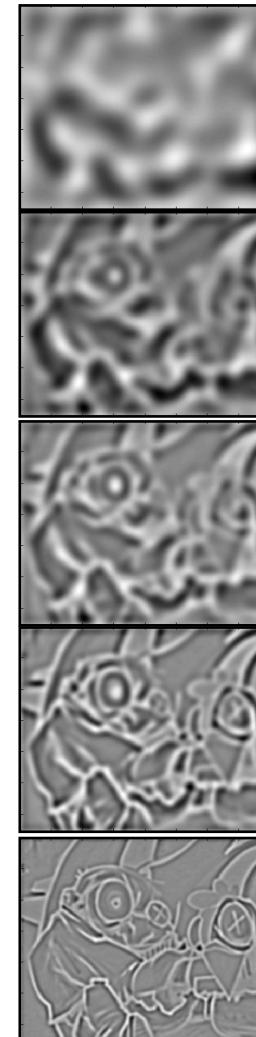


DoG – Efficient Computation

- Find **local maxima** in **position-scale** space of Difference-of-Gaussian in 26 values



$$L_{xx}(\sigma) + L_{yy}(\sigma) \rightarrow \sigma^5$$
$$\sigma^4$$
$$\sigma^3$$
$$\sigma^2$$
$$\sigma$$



⇒ List of
 (x, y, s)



DoG – Efficient Computation

- Keypoint Localization - **removing noises**

- The approach is similar to the one used in the Harris Corner Detector for removing edge features.

- Reject flats:

- $|D(\hat{x})| < 0.03$

- Reject edges:

$$\mathbf{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}$$

Let α be the eigenvalue with larger magnitude and β the smaller.

$$\text{Tr}(\mathbf{H}) = D_{xx} + D_{yy} = \alpha + \beta,$$

$$\text{Det}(\mathbf{H}) = D_{xx}D_{yy} - (D_{xy})^2 = \alpha\beta.$$

Let $r = \alpha/\beta$.
So $\alpha = r\beta$

$$\frac{\text{Tr}(\mathbf{H})^2}{\text{Det}(\mathbf{H})} = \frac{(\alpha + \beta)^2}{\alpha\beta} = \frac{(r\beta + \beta)^2}{r\beta^2} = \frac{(r+1)^2}{r},$$

- $r < 10$

$(r+1)^2/r$ is at a min when the 2 eigenvalues are equal.

Points Descriptor



Image Representations

- Templates
 - Intensity, gradients, etc.



- Histograms
 - Color, texture, SIFT descriptors, etc.

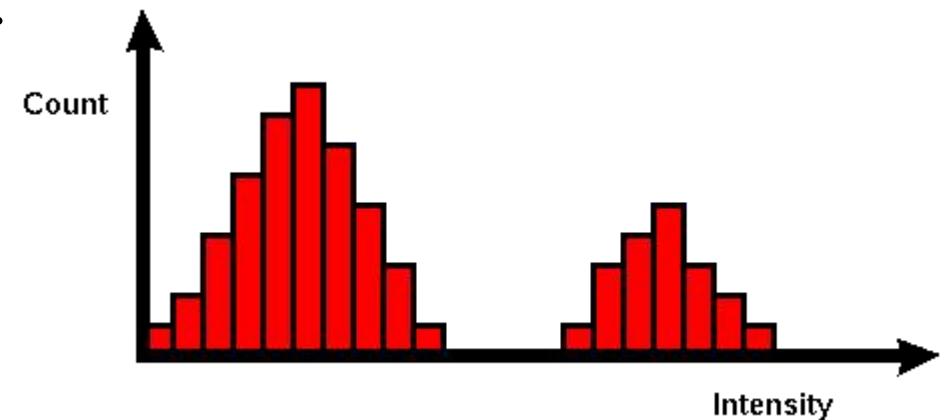




Image Representations: Histograms

- Global histogram
 - Represent distribution of features
 - Color, texture, depth, ...

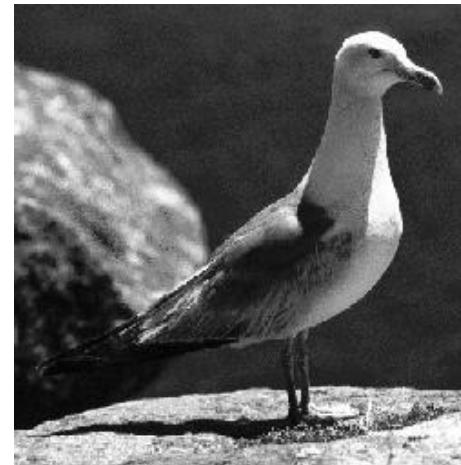
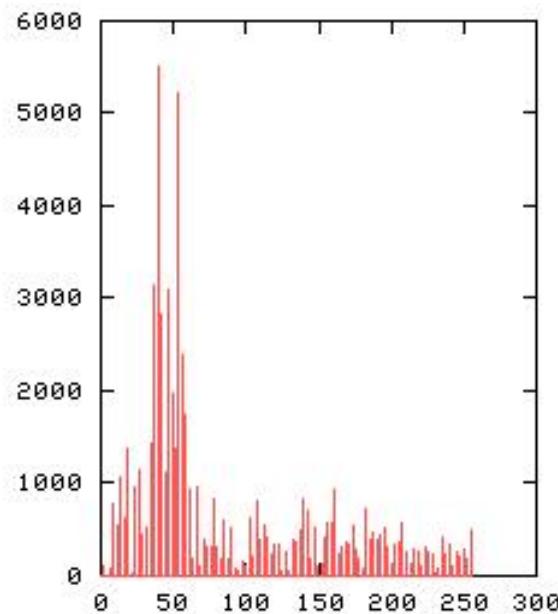




Image Representations: Histograms

- Histogram: probability or count of data in each bin
- Joint histogram
 - Requires lots of data
 - Loss of resolution to avoid empty bins
- Marginal histogram
 - Require independent features
 - More data/bin than joint histogram

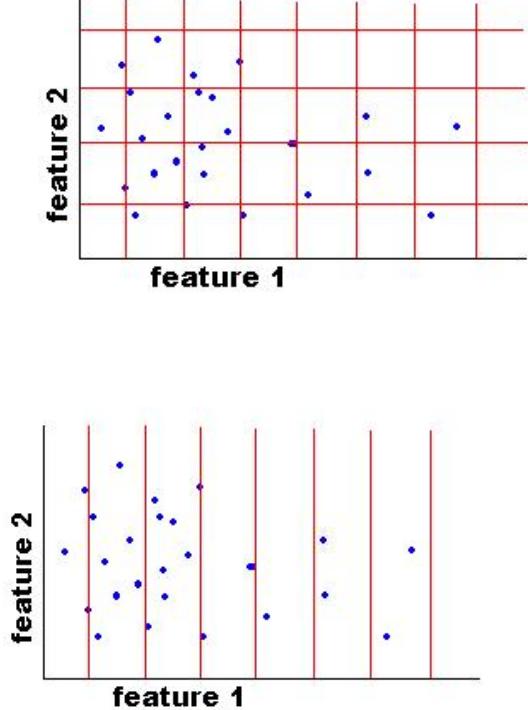
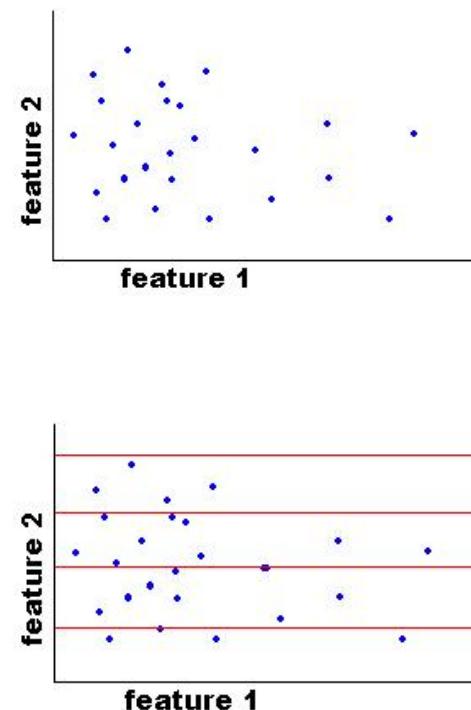
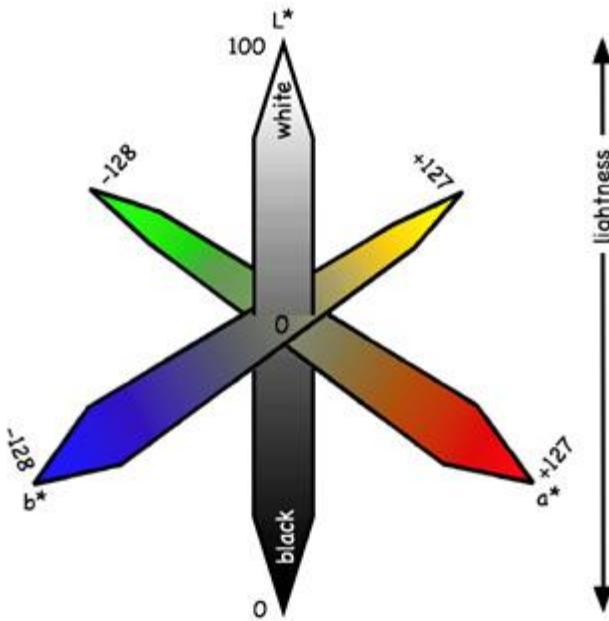




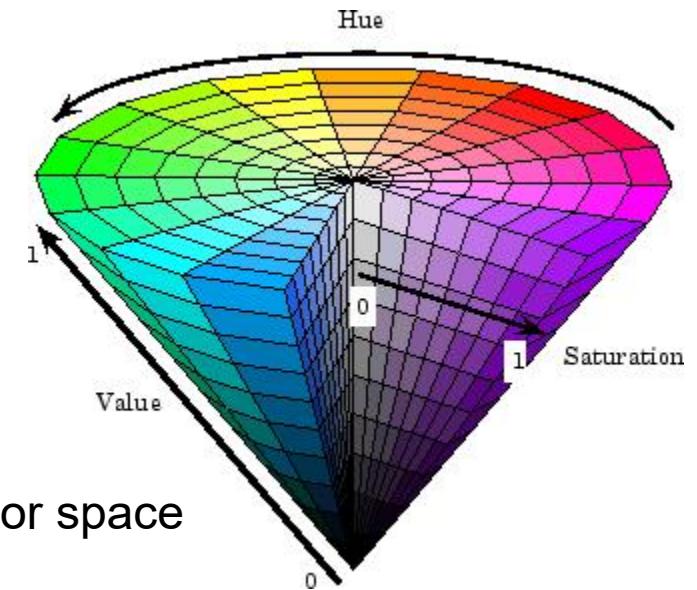
Image Representations

- What kind of things do we compute histograms of?
 - Color

L^{*}a^{*}b^{*} color space



HSV color space



- Texture (filter banks or HOG over regions)



Histogram of Oriented Gradients

- Gaussian-smoothed image at the **keypoint's scale** $L(x, y, \sigma)$

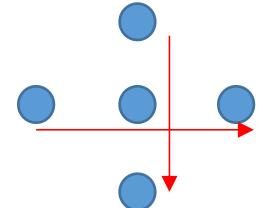
➤ Every pixel in a neighboring region around the keypoint

✓ Magnitude

✓ Direction

$$m(x, y) = \sqrt{(L(x+1, y) - L(x-1, y))^2 + (L(x, y+1) - L(x, y-1))^2}$$

$$\theta(x, y) = \text{atan2}(L(x, y+1) - L(x, y-1), L(x+1, y) - L(x-1, y))$$



- An **orientation histogram** is formed

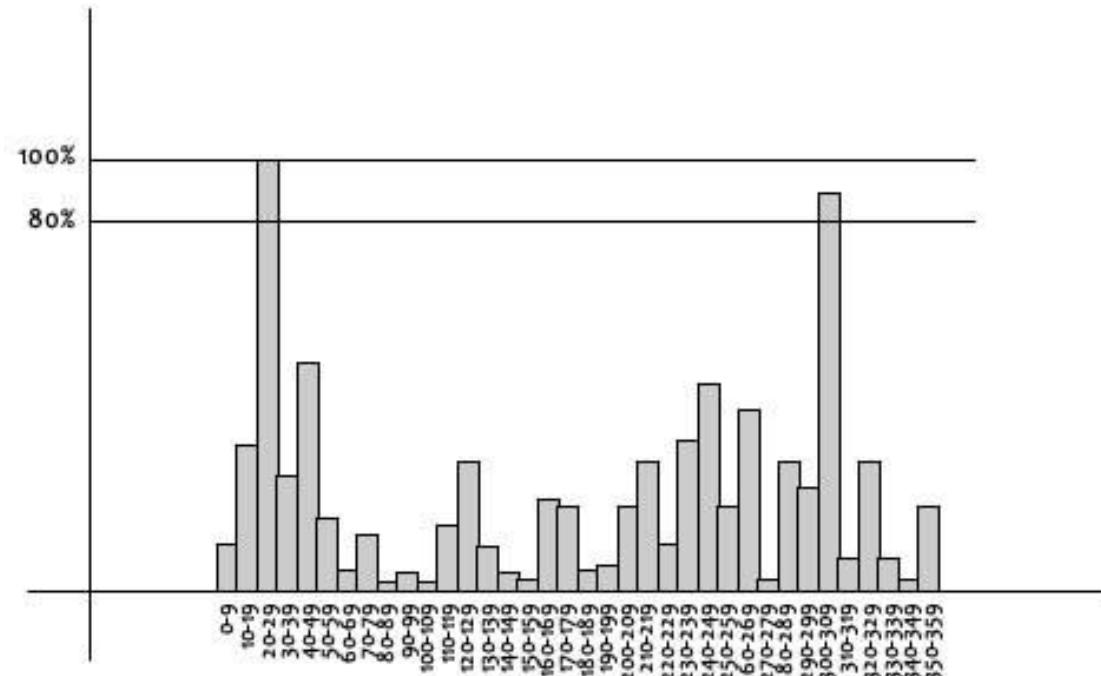
➤ Each sample in the **neighboring window** added to a histogram bin is weighted by its gradient magnitude and by a **Gaussian-weighted** circular window

➤ The **peaks** in this histogram correspond to **dominant** orientations



Histogram of Oriented Gradients

- An orientation histogram with 36 bins covering 360 degrees is created
 - A histogram bin is **weighted by its gradient magnitude**





Orientation Normalization

- Compute orientation histogram
- Select **dominant** orientation
- Normalize: rotate to fixed orientation

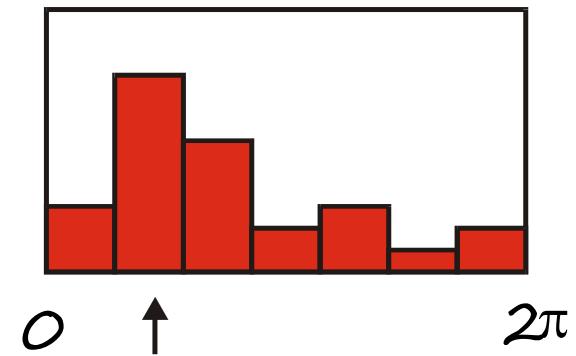
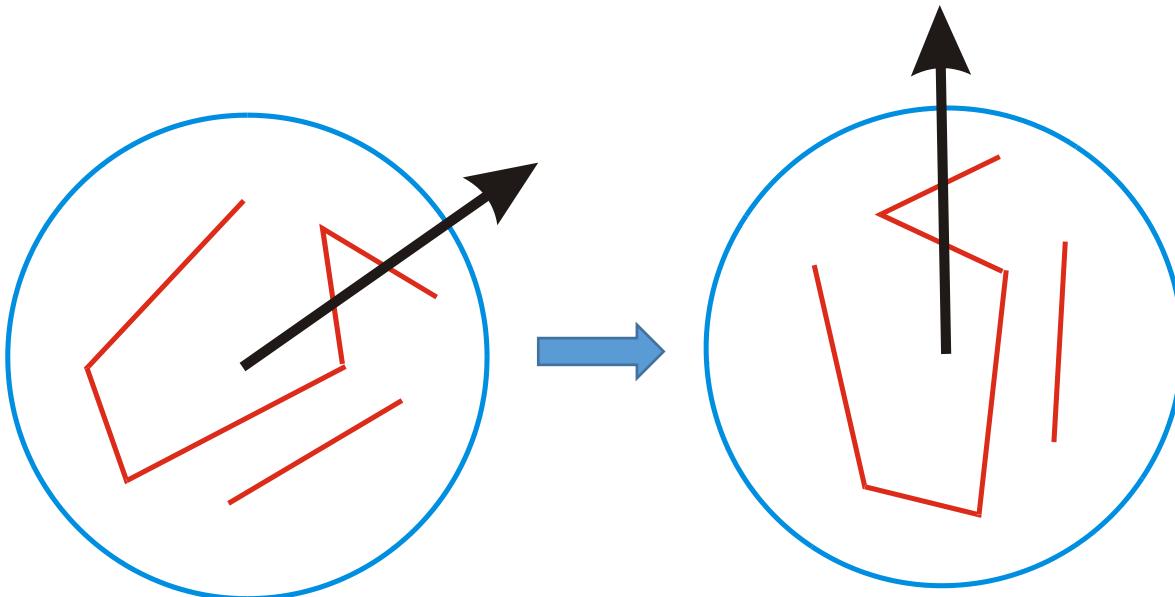
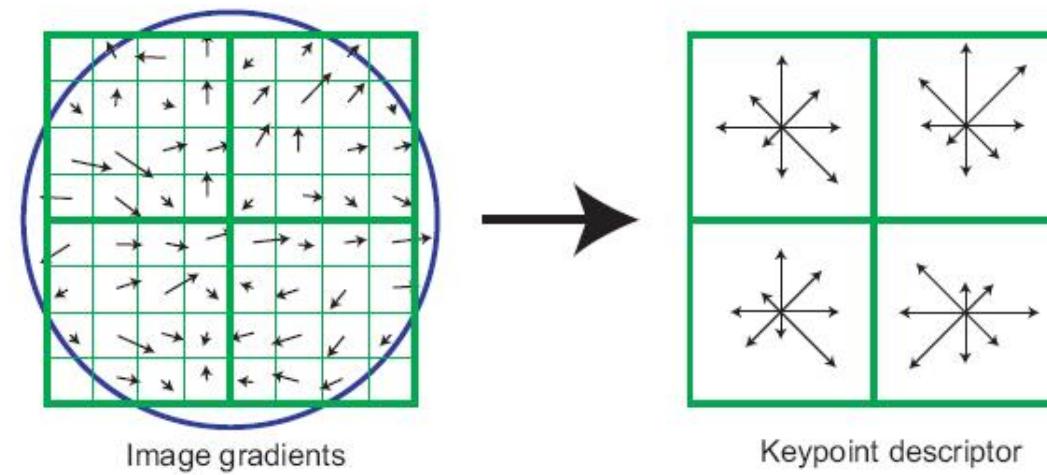
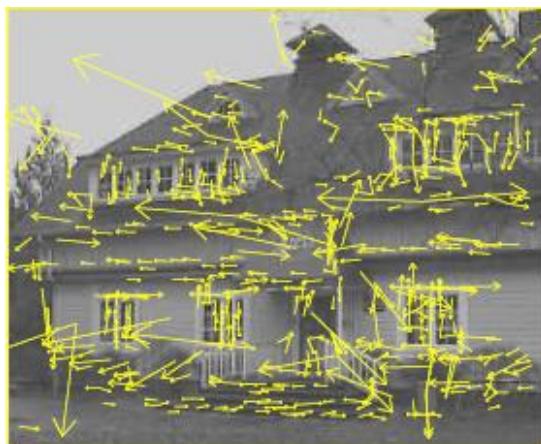




Image Representations

- At this point, each keypoint has a **location, scale, orientation**
- What kind of things do we compute histograms of?

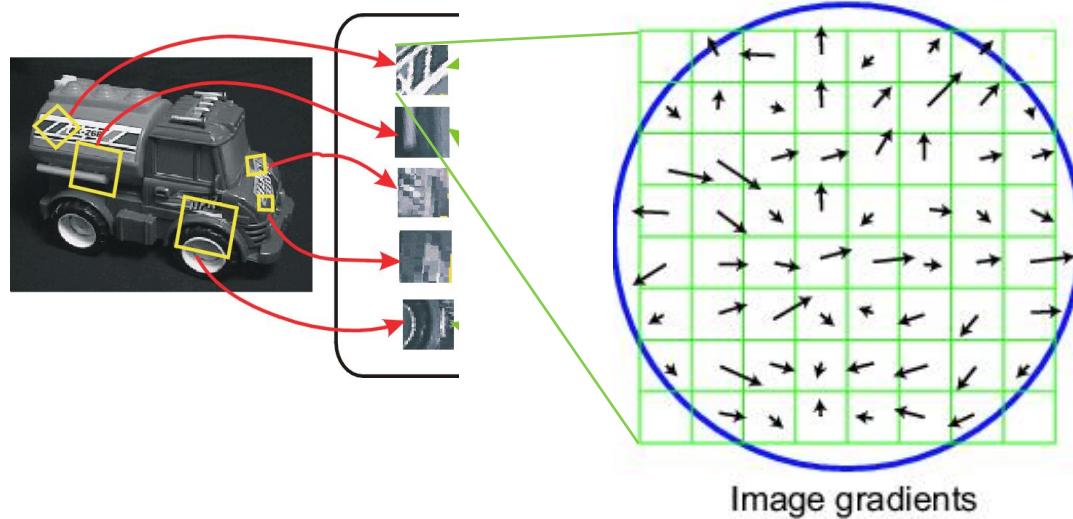


SIFT – Lowe IJCV 2004



SIFT Vector Formation

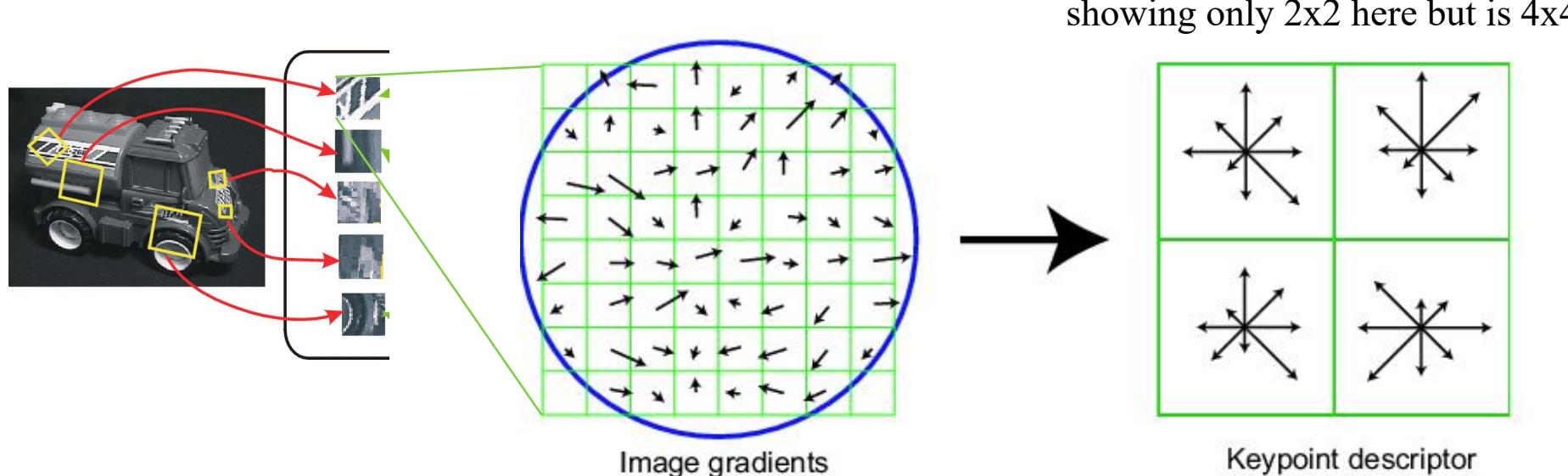
- Computed on **rotated** and **scaled** version of window according to computed orientation & scale
 - Resample the window
- Based on gradients weighted by a Gaussian of variance half the window (for smooth falloff)





SIFT Vector Formation

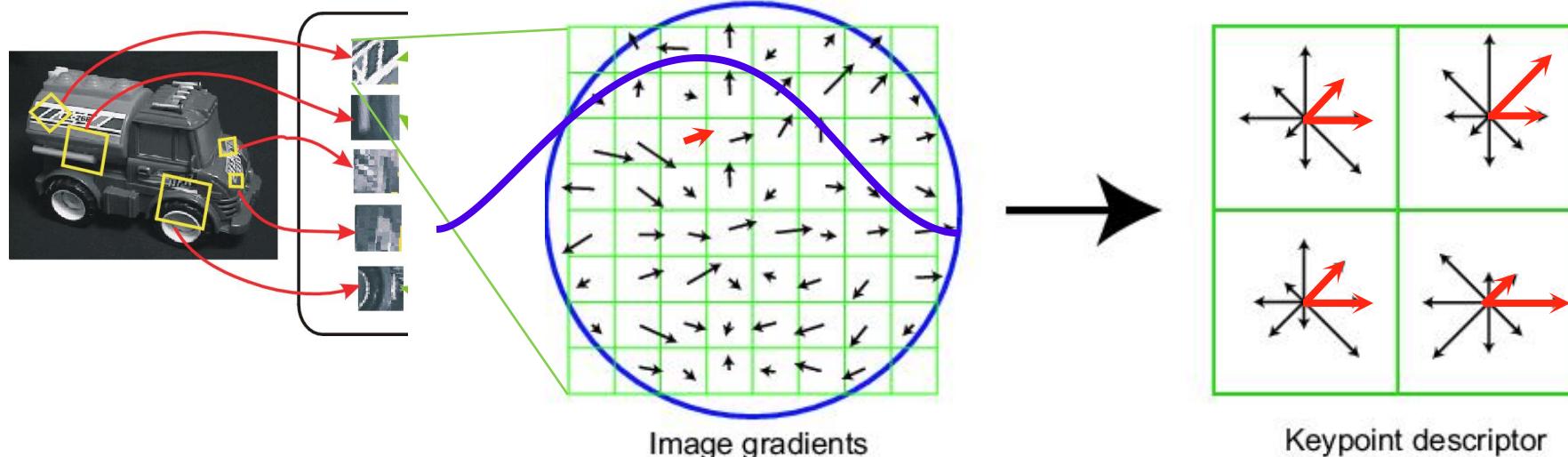
- 4x4 array of gradient orientation histogram weighted by magnitude
- 8 orientations \times 4x4 array = 128 dimensions
- Motivation: some sensitivity to spatial layout, but not too much.





Ensure Smoothness

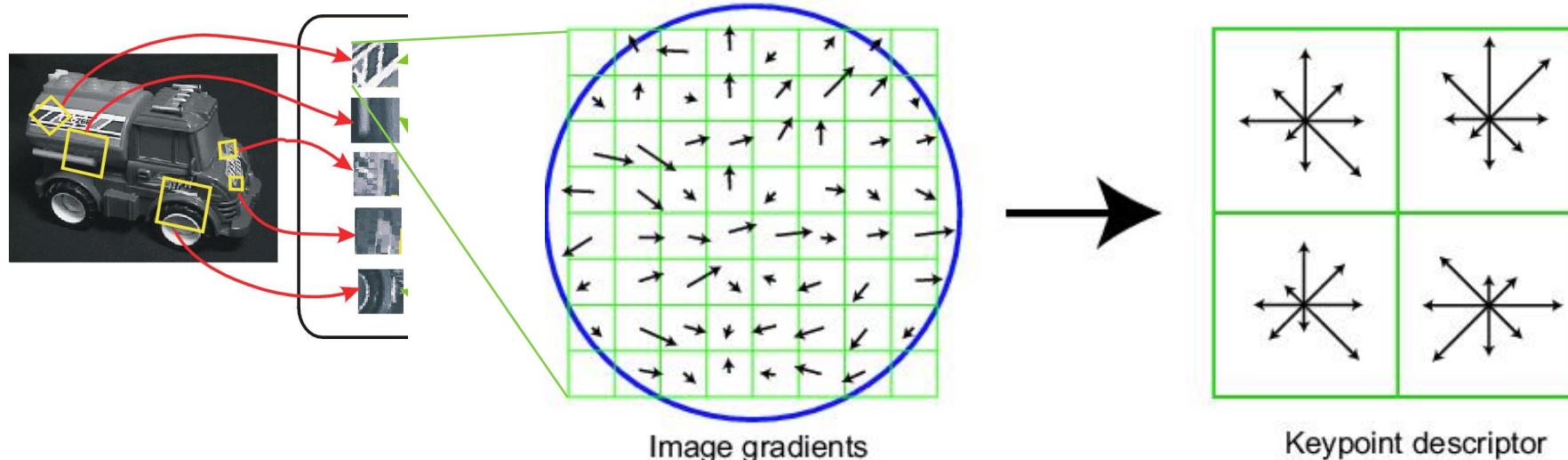
- Gaussian weight
- Interpolation
 - A given gradient contributes to 8 bins: 4 in space times 2 in orientation





Reduce Effect of Illumination

- 128-dim vector normalized to 1
- Threshold gradient magnitudes to avoid excessive influence of high gradients
 - After normalization, clamp gradients > 0.2
 - Renormalize





Local Descriptors

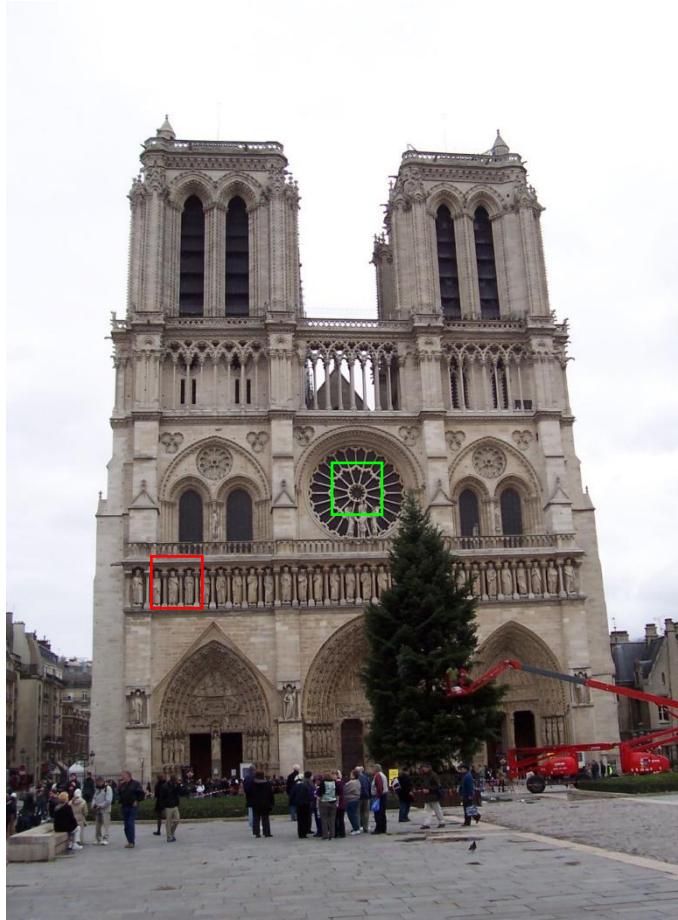
- Most features can be thought of as **templates, histograms** (counts), or combinations
- The ideal descriptor should be
 - Robust
 - Distinctive
 - Compact
 - Efficient
- Most available descriptors focus on edge/gradient information
 - Capture texture information
 - Color rarely used

Points Matching

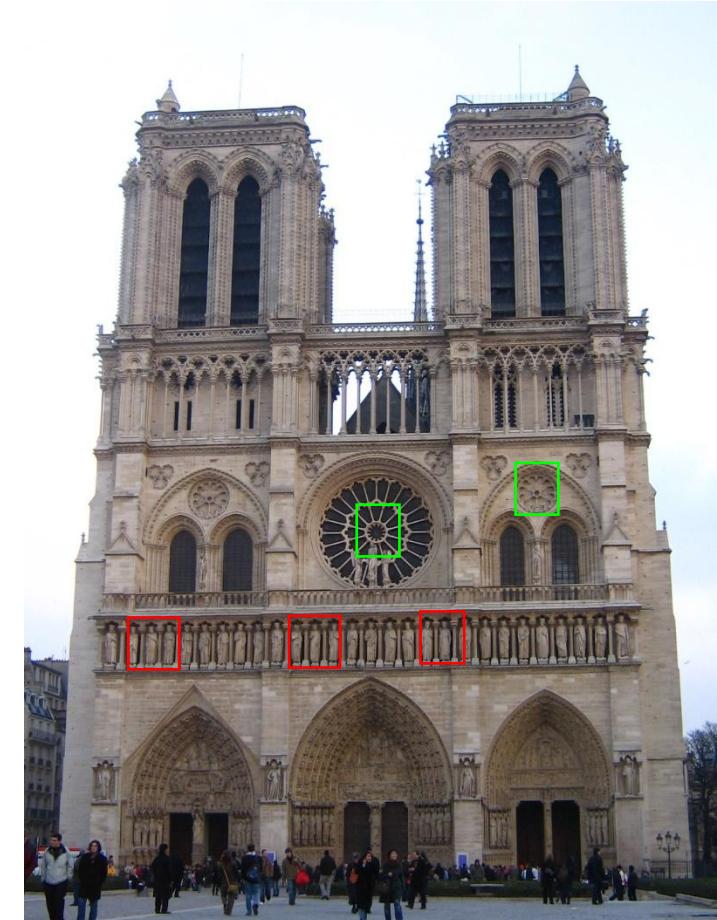


Matching

- Simplest approach: Pick the nearest neighbor
- Threshold on absolute distance
- Problem: Lots of self similarity in many photos



Distance: 0.34, 0.30, 0.40

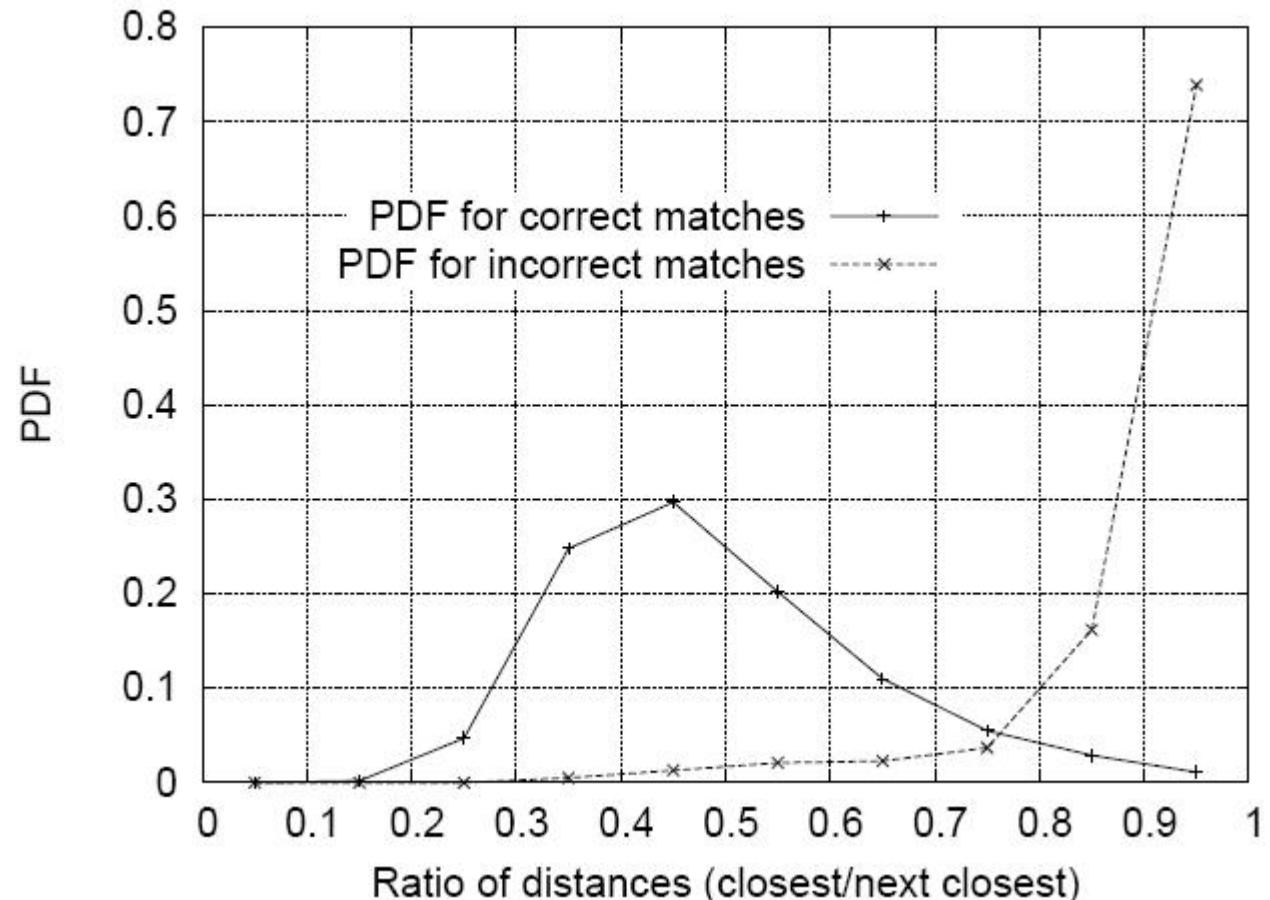


Distance: 0.61, 1.22



Nearest Neighbor Distance Ratio

- $\frac{NN1}{NN2}$ where NN1 is the distance to the **first** nearest neighbor and NN2 is the distance to the **second** nearest neighbor
- Sorting by this ratio puts matches in order of confidence



Conclusions



Choosing a Detector

- What do you want it for?
 - Precise **localization** in x-y: Harris
 - Good localization in **scale**: Difference of Gaussian
- Best choice often application dependent
 - Harris-/Hessian-Laplace/DoG work well for many natural categories
- There have been extensive evaluations/comparisons
 - [Mikolajczyk et al., IJCV'05, PAMI'05]
 - All detectors/descriptors shown here work well



Comparison of Keypoint Detectors

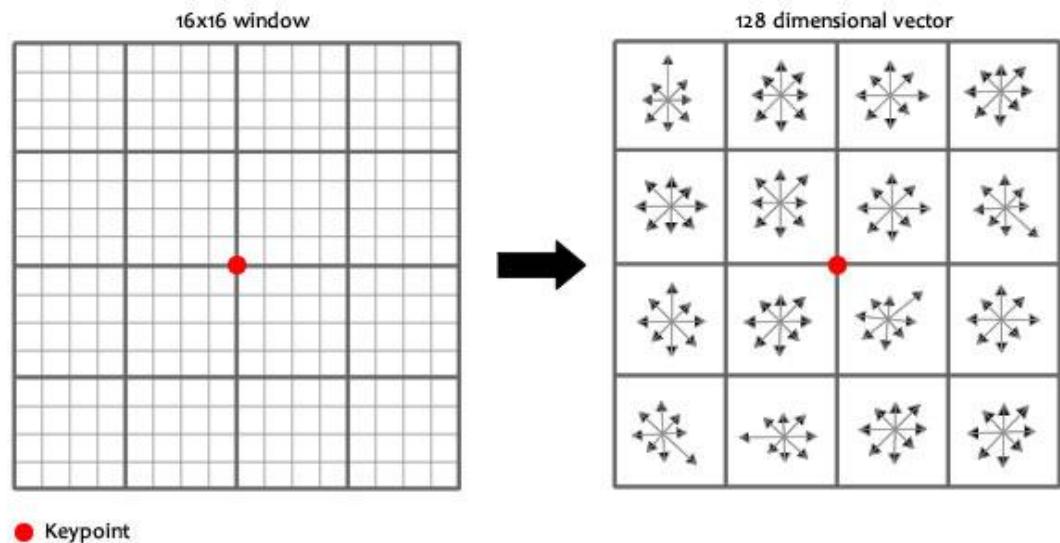
Table 7.1 Overview of feature detectors.

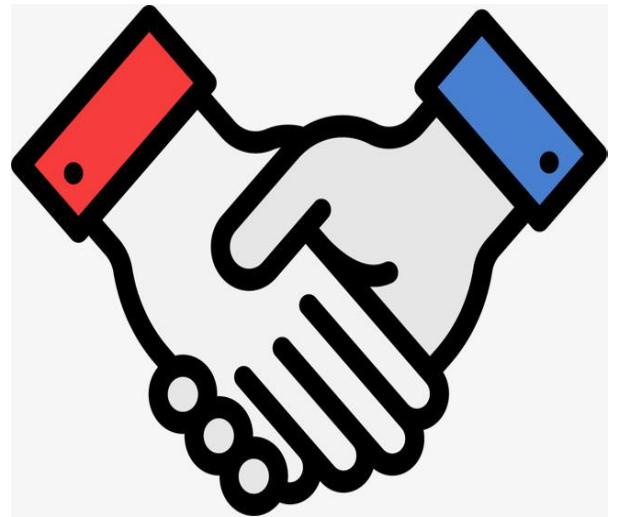
Feature Detector	Corner	Blob	Region	Rotation invariant	Scale invariant	Affine invariant	Repeatability	Localization accuracy	Robustness	Efficiency
Harris	✓			✓			+++	+++	+++	++
Hessian		✓		✓			++	++	++	+
SUSAN	✓			✓			++	++	++	+++
Harris-Laplace	✓	(✓)		✓	✓		+++	+++	++	+
Hessian-Laplace	(✓)	✓		✓	✓		+++	+++	+++	+
DoG	(✓)	✓		✓	✓		++	++	++	++
SURF	(✓)	✓		✓	✓		++	++	++	+++
Harris-Affine	✓	(✓)		✓	✓	✓	+++	+++	++	++
Hessian-Affine	(✓)	✓		✓	✓	✓	+++	+++	+++	++
Salient Regions	(✓)	✓		✓	✓	(✓)	+	+	++	+
Edge-based	✓			✓	✓	✓	+++	+++	+	+
MSER			✓	✓	✓	✓	+++	+++	++	+++
Intensity-based			✓	✓	✓	✓	++	++	++	++
Superpixels			✓	✓	(✓)	(✓)	+	+	+	+



Things to Remember

- Keypoint detection:
repeatable and distinctive
 - Corners, blobs, stable regions
 - Harris, DoG
- Descriptors: robust and selective
 - Spatial histograms of orientation
 - SIFT





Thanks



zhengf@sustc.edu.cn