Mid-Level CV: Feature Detection

ENEE 4584/5584 CV app in DL

Dr. Alsamman

Slide Credits:

Image Features

- Information in an image that is unique to the image
- Global features
  - One feature vector per image
  - Examples: histograms, shape, texture, eigenspace, etc.
  - Allow for compact representation of an image
  - Standard algorithms can be used
  - Sensitive to occlusion, clutter, deformations, etc.
  - Segmentation must be used.

Image Features

- Local features
  - Feature as a patch or region of interest.
  - No need for segmentation.
  - Robust to occlusion, clutter, deformations, etc.
  - Multiple feature vectors of varying sizes.
  - Non-standard algorithms

Local Features Desired Properties

- Repeatability
  - The same feature can be found in several images despite geometric and photometric transformations
- Saliency
  - Each feature has a distinctive description
- Compactness and efficiency
  - Many fewer features than image pixels
- Locality
  - A feature occupies a relatively small area of the image;
  - robust to clutter and occlusion

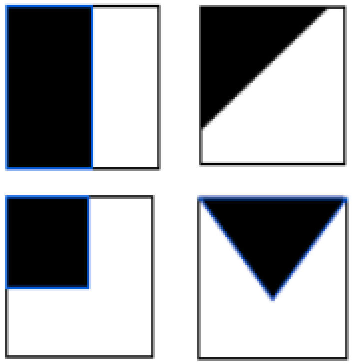
Popular Features

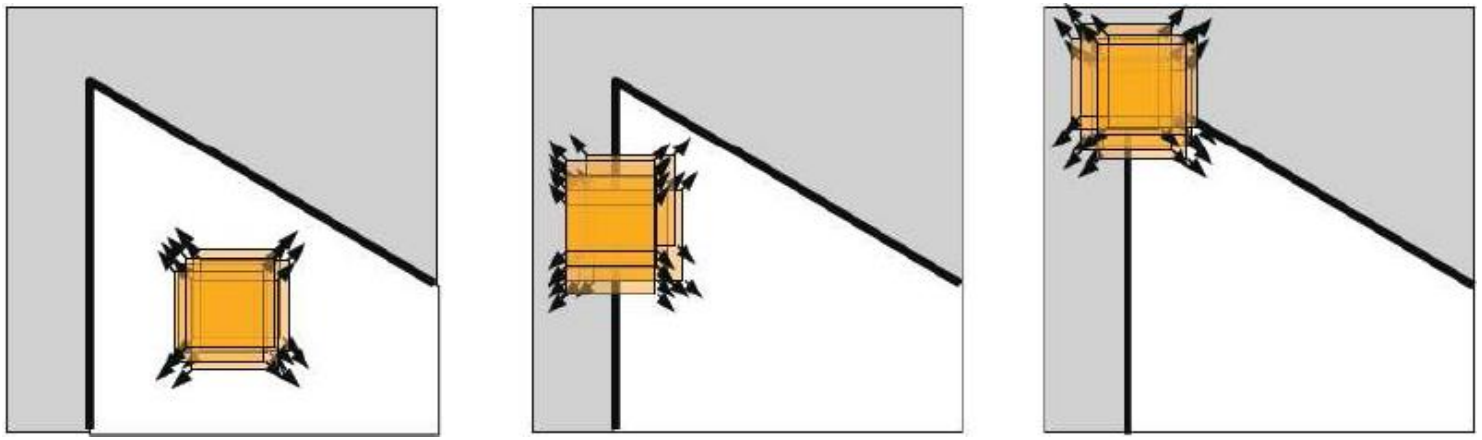
- Harris Corner Detection [1989]
  - https://docs.opencv.org/3.4/dc/d0d/tutorial_py_features_harris.html
  - Shi-Tomasi Corner Detector [1994]
    - https://docs.opencv.org/3.4/d4/d8c/tutorial_py_shi_tomasi.html
- Scale-Invariant Feature Transform (SIFT) [2004]
  - https://docs.opencv.org/3.4/da/df5/tutorial_py_sift_intro.html
  - Speeded-Up Robust Features (SURF) [2006]
    - https://docs.opencv.org/3.4/df/dd2/tutorial_py_surf_intro.html
- Features from Accelerated Segment Test (FAST) [2006,2010]
  - https://docs.opencv.org/3.4/df/d0c/tutorial_py_fast.html
- Binary Robust Independent Elementary Features (BRIEF) [2010]
  - https://docs.opencv.org/3.4/dc/d7d/tutorial_py_brief.html
- Oriented FAST and Rotated BRIEF (ORB) [2011]
  - https://docs.opencv.org/3.4/d1/d89/tutorial_py_orb.html

Harris Corner Detection

Corner Detection

- Local features that are insensitive to deformation (scale, rotation).
  - Use for detection, calibration, etc.
- Edge detection is not very good for corner detection.
  - Focuses on gradient in x-y direction.
  - Corner is a discontinuity: gradient is not defined.
- Idea: shifting a window in the image in multiple directions
  - Max difference when there is a corner





Windowed Square Sum Difference (SSD)

- Defined an a patch/window size
  - Size = 2mx2m
  - Can use a Gaussian window
- Windowed SSD
  - Shift the window by u,v

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

change

weighting window

shifted image

original image

$$E(x,y) = w(x,y) * [I(x+u, y+v) - I(x,y)]^2$$

2D convolution

Taylor Series Approximation

$$f(x+u, y+v) = f(x,y) + uf_x(x,y) + vf_y(x,y) +$$

First partial derivatives

$$\frac{1}{2!}\left[u^2 f_{xx}(x,y) + uvf_{xy}x,y + v^2 f_{yy}(x,y)\right] +$$

Second partial derivatives

$$\frac{1}{3!}\left[u^3 f_{xxx}(x,y) + u^2 v f_{xxy}(x,y) + uv^2 f_{xyy}(x,y) + v^3 f_{yyy}(x,y)\right]$$

Third partial derivatives

$+ \ \ldots$ (Higher order terms)

$$f(x+u, y+v) \approx f(x,y) + uf_x(x,y) + vf_y(x,y)$$

Taylor series expansion

For small shifts

$$\begin{aligned} u &\rightarrow 0 \\ v &\rightarrow 0 \end{aligned} \quad \Rightarrow \quad I(x+u, y+v) \approx I(x,y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v$$

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

image derivatives
along x and y axes

SSD Derivation

$$E(x,y) = w(x,y) * \left(I(x,y) + [I_x \quad I_y] \left[\begin{array}{c} u \\ v \end{array} \right] - I(x,y) \right)^2$$

$$= w(x,y) * \left([I_x \quad I_y] \left[\begin{array}{c} u \\ v \end{array} \right] \right)^2$$

$$= w(x,y) * \left([I_x \quad I_y] \left[\begin{array}{c} u \\ v \end{array} \right] \right)^{\mathrm{T}} \left([I_x \quad I_y] \left[\begin{array}{c} u \\ v \end{array} \right] \right)$$

$$E(x,y) = w(x,y) * [u \quad v] \begin{bmatrix} I_x \\ I_y \end{bmatrix} [I_x \quad I_y] \begin{bmatrix} u \\ v \end{bmatrix}$$

$$= [u \quad v] \underbrace{\left(w(x,y) * \begin{bmatrix} I_x^2(x,y) & I_x(x,y) I_y(x,y) \\ I_x(x,y) I_y(x,y) & I_y^2(x,y) \end{bmatrix} \right)}_{M(x,y)} \begin{bmatrix} u \\ v \end{bmatrix}$$

$$E(x,y) = [u \quad v] M(x,y) \begin{bmatrix} u \\ v \end{bmatrix}$$

For a Gaussian window

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

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

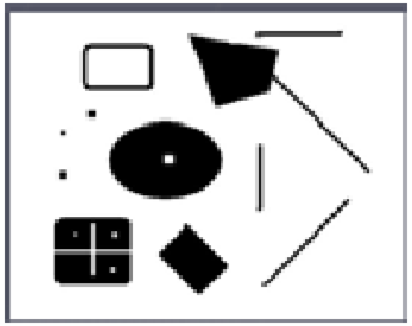
Image Gradient

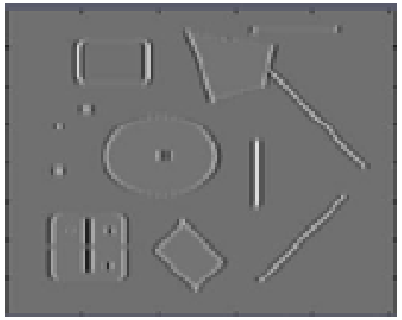
$$I_x(x, y) = I(x+1, y) - I(x, y)$$

$$= \underbrace{\begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 1 \\ 0 & 0 & 0 \end{bmatrix}}_{D_x(x,y)} * I(x, y)$$

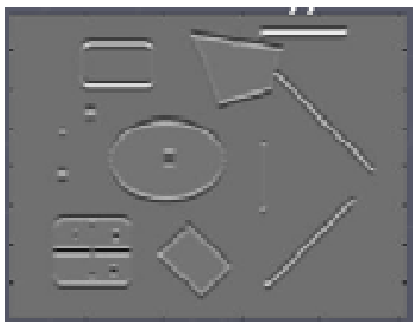
$$I_y(x,y) = I(x,y+1) - I(x,y)$$

$$= \underbrace{\begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 1 & 0 \end{bmatrix}}_{D_y(x,y)} * I(x,y)$$

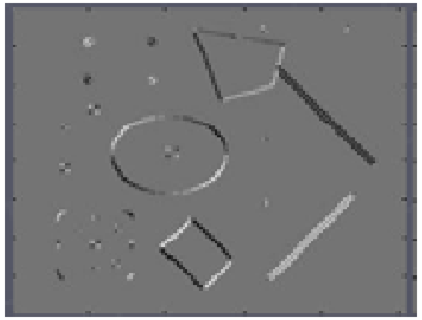




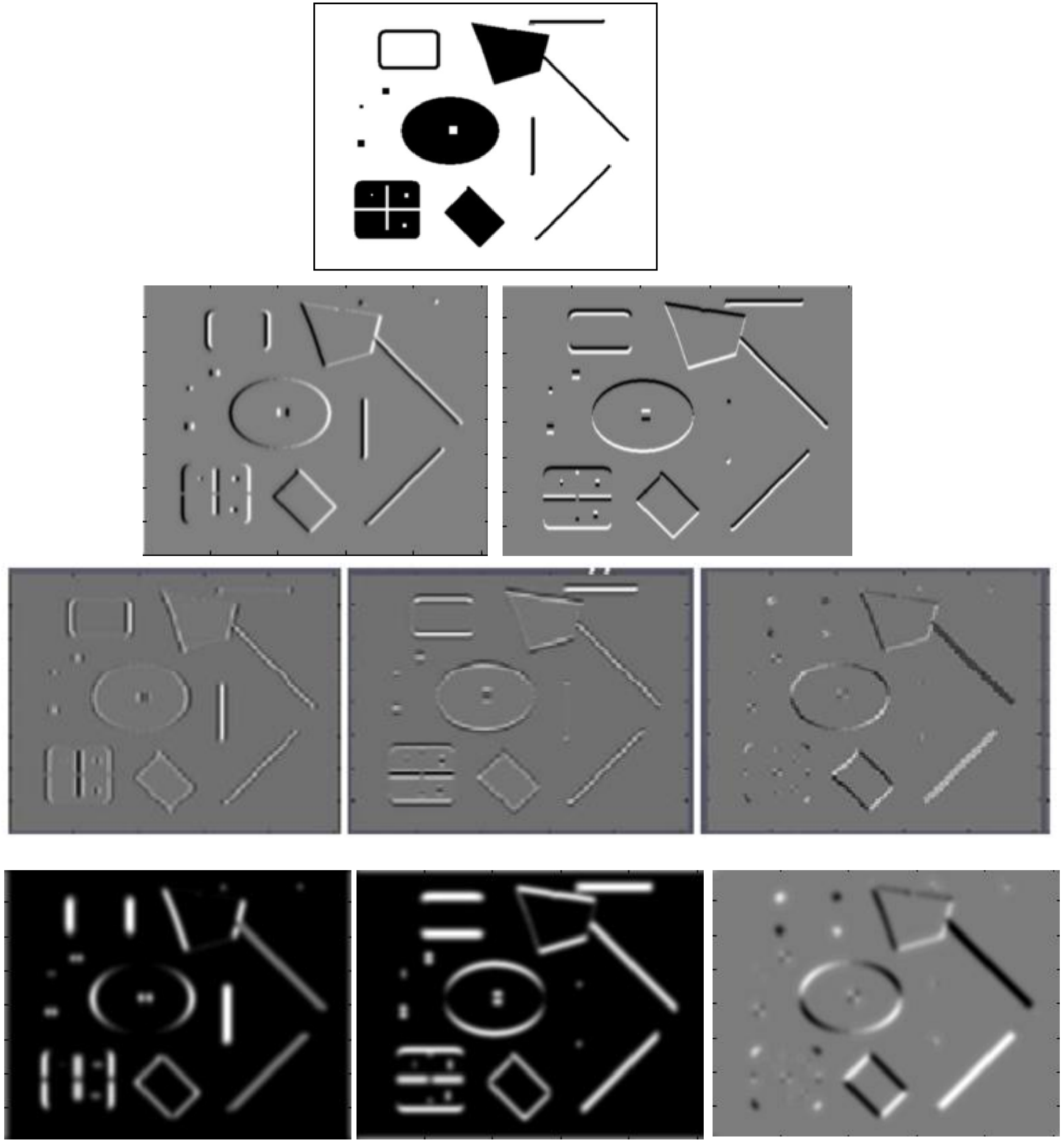
I_x



I_y

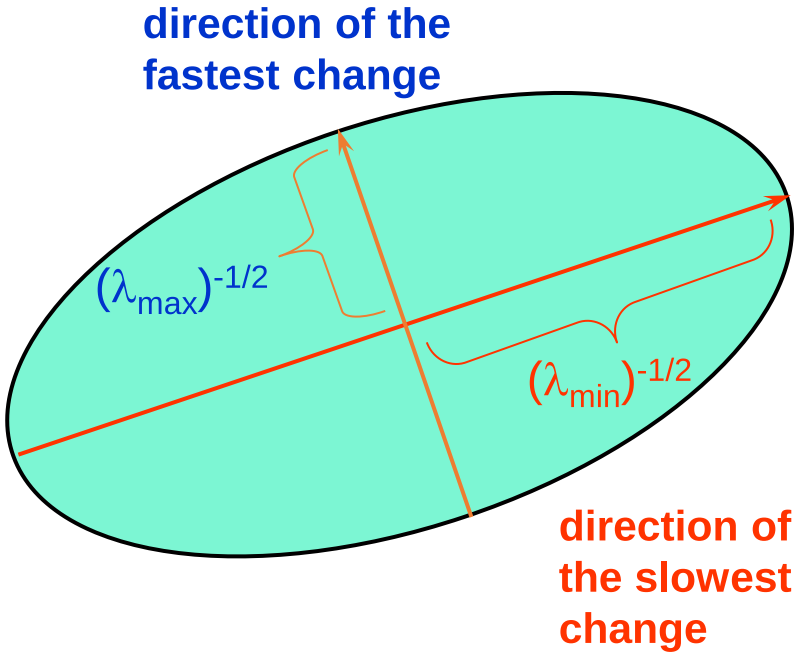


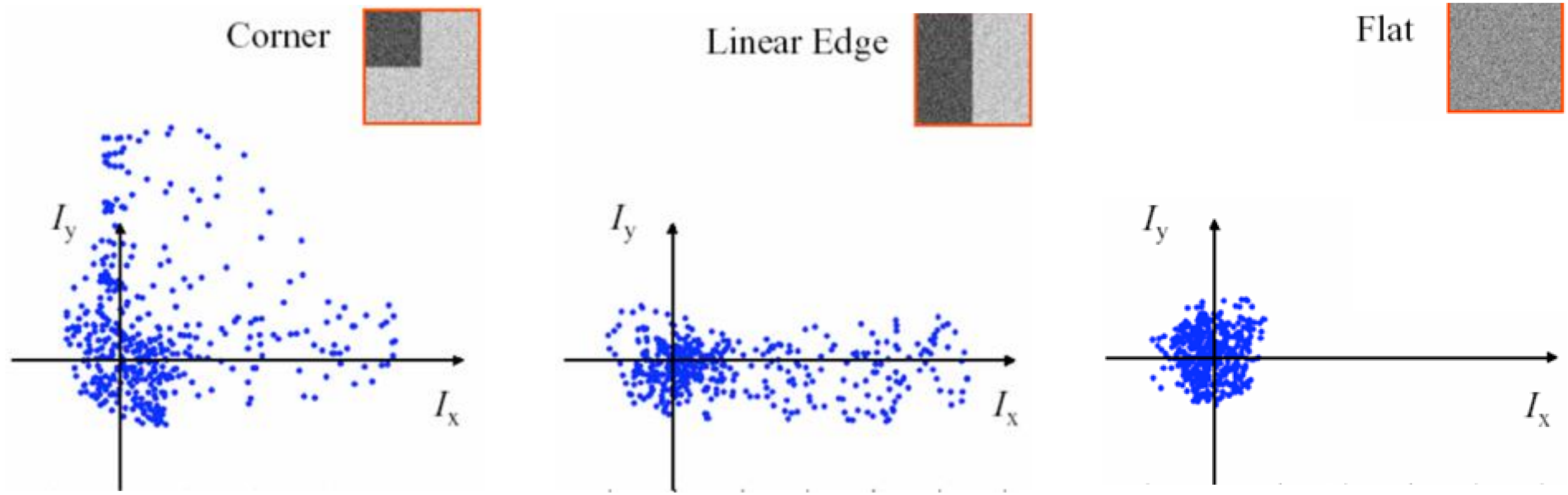
$I_x I_y$



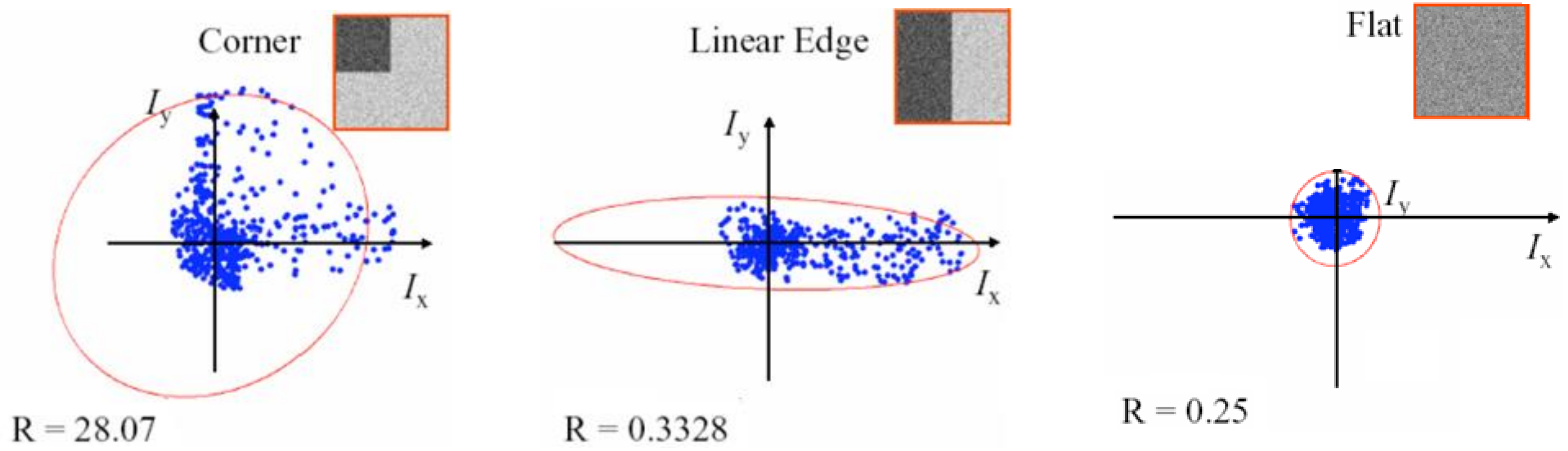
Matrix Form

- Automated method for edge detection
- 2nd order Taylor series approx of SSD:
  - Matrix form: $S \approx ½ [u\ v]\ M\ [u\ ;\ v]$
  - $M = \Sigma_x \Sigma_y\ w(x,y) * [I_x^2 \quad I_x I_y \quad ; \quad I_x I_y \quad I_y^2\ ]$
  - I_x , I_y = gradient in x, y
- S < Threshold is an ellipse
  - Ellipse can be described by the orthogonal basis of M
  - Orthogonal basis: direction of fastest and slowest change
  - Use eigenvectors of M to detect the 2 orthogonal basis: v_1, v_2
  - $\lambda_1 > \lambda_2$ => v_1 is direction of fastest change




Corner
Linear Edge
Flat
I_y
I_x
I_y
I_x
I_y
I_x

- Corners, edges, flat regions:
  - Edge point: $\lambda_1 > \lambda_2$ or $\lambda_2 > \lambda_1$
  - Flat regions: small λ_1 , λ_2
  - Corners: $\lambda_1 \approx \lambda_2$; λ_1 , λ_2 large
- λ’s can be used to generate a response parameter, R:
  - $R = \lambda_1 \lambda_2 – k(\lambda_1 - \lambda_2)^2$
  - k = constant (~0.04-0.06)
  - Edges: $R < 0$
  - Flat regions: $|R|$ is small
  - Corners: $R > 0$


Corner
I_y
I_x
R = 28.07
Linear Edge
I_y
I_x
R = 0.3328
Flat
I_y
I_x
R = 0.25

Harris Corner Detection Algorithm

- Smooth image to reduce noise
  - Harris is susceptible to noise.
- Compute magnitude of x, y gradient at each pixel
  - Remember images origin is upper-left corner.
  - Can be performed in a window. Harris uses Gaussian window.
- Find eigen values of A
  - Can be computed in closed form: A = [a b; c d]

    $\lambda_1 = (a+d - sqrt((a+d)^2 + 4bc))/2$

    $\lambda_2 = (a+d + sqrt((a+d)^2 + 4bc))/2$
- Check the λ_1, λ_2.
  - A response function can be used: $R = \lambda_1\lambda_2 - k(\lambda_1 - \lambda_2)^2$
  - k = constant (~0.04-0.06)
- Keep local maxima of R

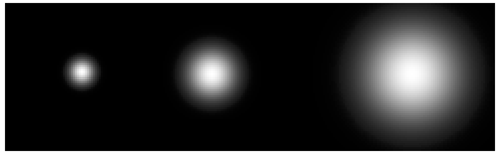
SIFT

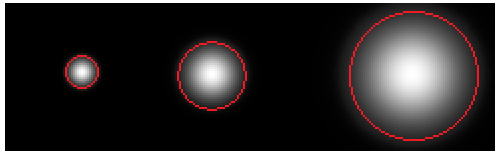
SIFT Features

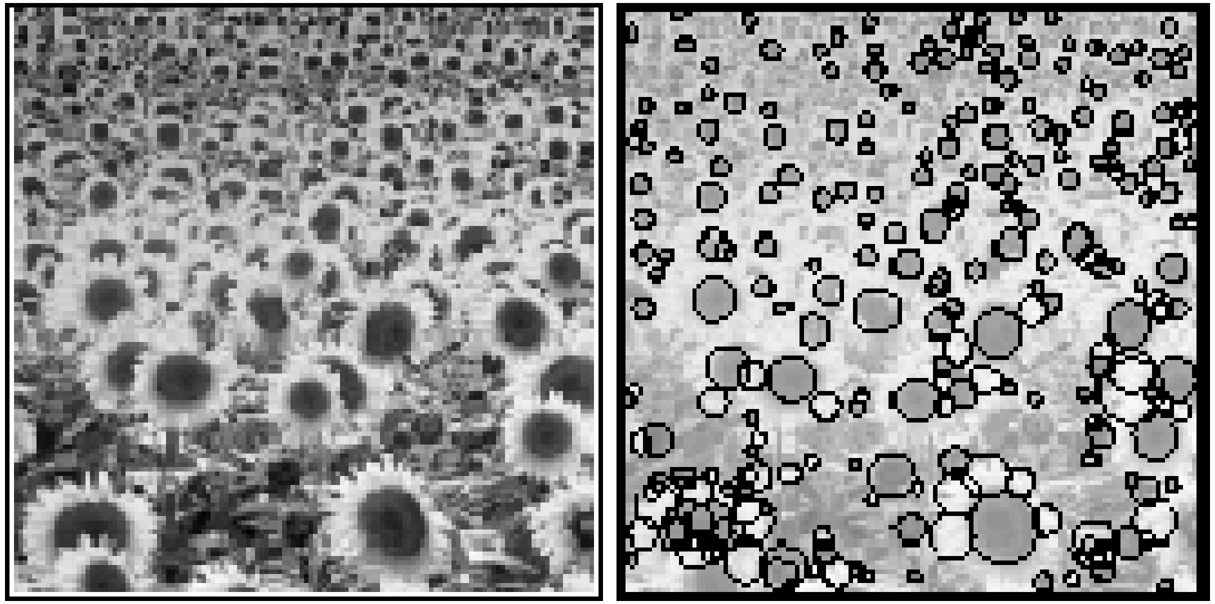
- Scale Invariant Feature Transform
  - Detects multiple blob sizes at various scales
  - Keeps blobs with the greatest response at each scale
  - Blob center located with sub-pixel accuracy
- Extracts local feature descriptors
  - Blobs are converted into a feature
  - Within each blob a gradients and their orientation are compiled into a histograms
  - Histograms provides some invariance to rotation
  - A feature vector descriptor makes matching a features easy
- *Reasonably* invariant to changes in illumination, image noise, rotation, scaling, and small changes in viewpoint.

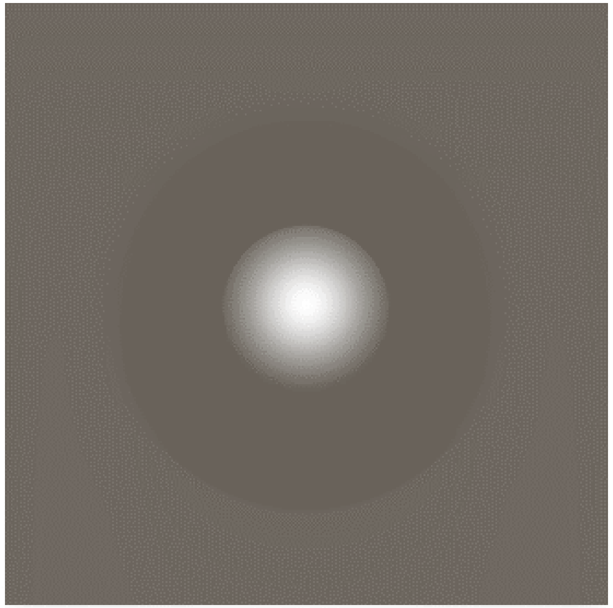
LoG Blob Detection

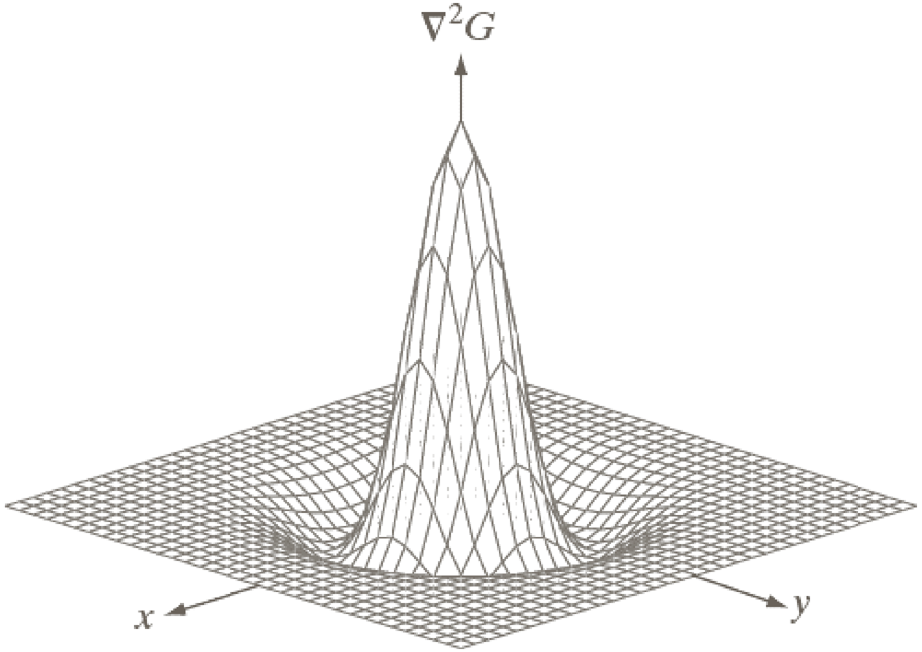
- Blobs have spatial size (scale)
  - Laplacian looks for points (independent of scale)
- Blobs tend to have a gaussian spread.
- By convolving the Laplacian with a Gaussian
  - AKA Laplacian of Gaussian (LoG)
  - Gaussian will blur smaller blobs/points
  - Laplacian will highlight edges
- Response is maximized when $\sigma = r/\sqrt{2}$
  - r is radius of the blob



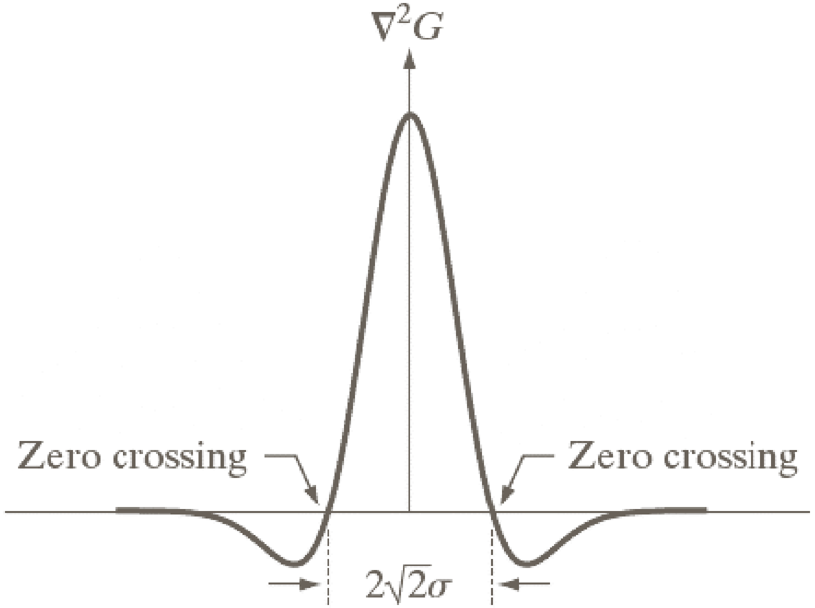








$\nabla^2 G$
x
y

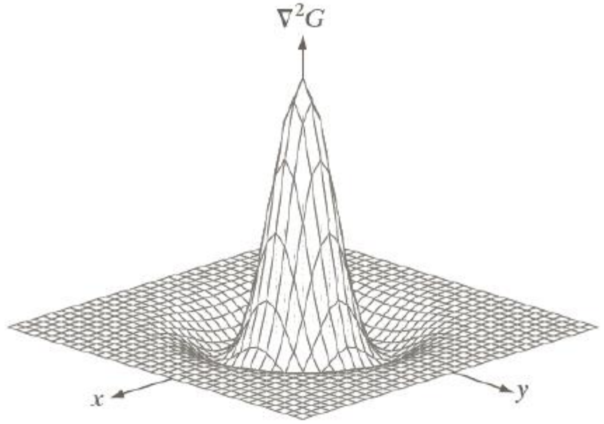


$\nabla^2 G$
Zero crossing
Zero crossing
$2\sqrt{2}\sigma$

Laplacian of Gaussian (LoG)

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

$$\nabla^2 g(x,y) = \left(\frac{x^2}{\sigma^4} + \frac{y^2}{\sigma^4} - \frac{2}{\sigma^2}\right)\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$$

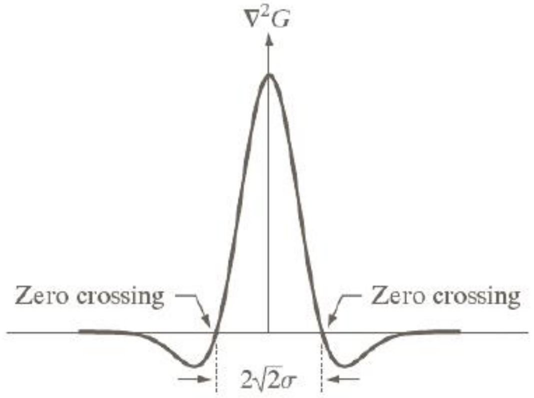


- The response is max when $\nabla^2 G$=0 or:

$$\left(\frac{x^2}{\sigma^4} + \frac{y^2}{\sigma^4} - \frac{2}{\sigma^2}\right) = 0 \quad \Rightarrow \quad \frac{x^2+y^2}{\sigma^4} = \frac{2}{\sigma^2}$$

$$x^2 + y^2 = 2\sigma^2 \quad \Rightarrow \quad r^2 = 2\sigma^2$$

$$\sigma = \frac{r}{\sqrt{2}}$$



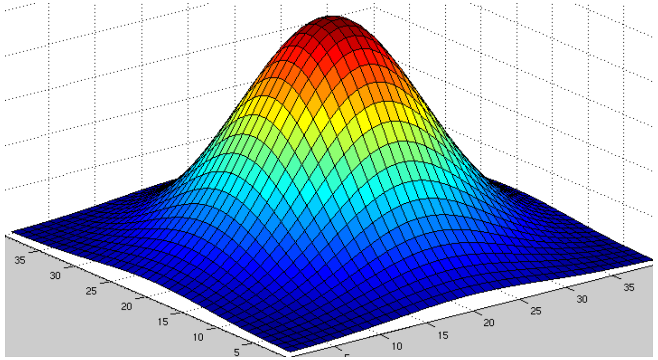
Difference of Gaussian

- LoG can be approximated by a DoG
- Subtract two Gaussians at slightly different scales
  - Correlates well with our understanding of how human vision works

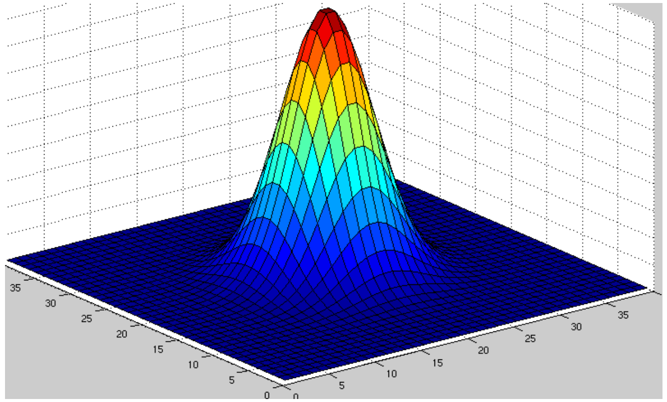
$$D(x, y) = \frac{1}{2\pi\sigma_1^2} \exp\left(-\frac{x^2 + y^2}{2\sigma_1^2}\right) - \frac{1}{2\pi\sigma_2^2} \exp\left(-\frac{x^2 + y^2}{2\sigma_2^2}\right)$$

  - $\sigma_1 > \sigma_2$.
  - $\sigma_1 = k\sigma_2$

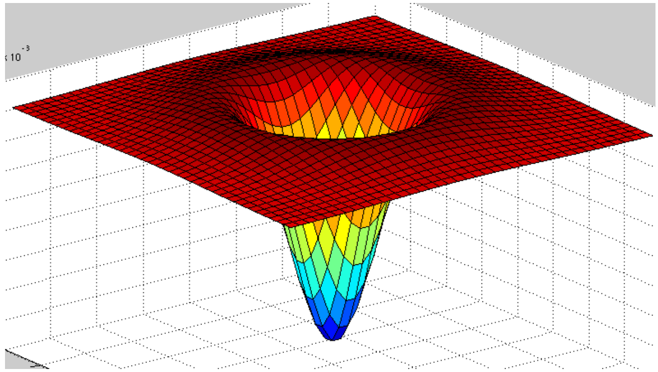
Difference of Gaussians

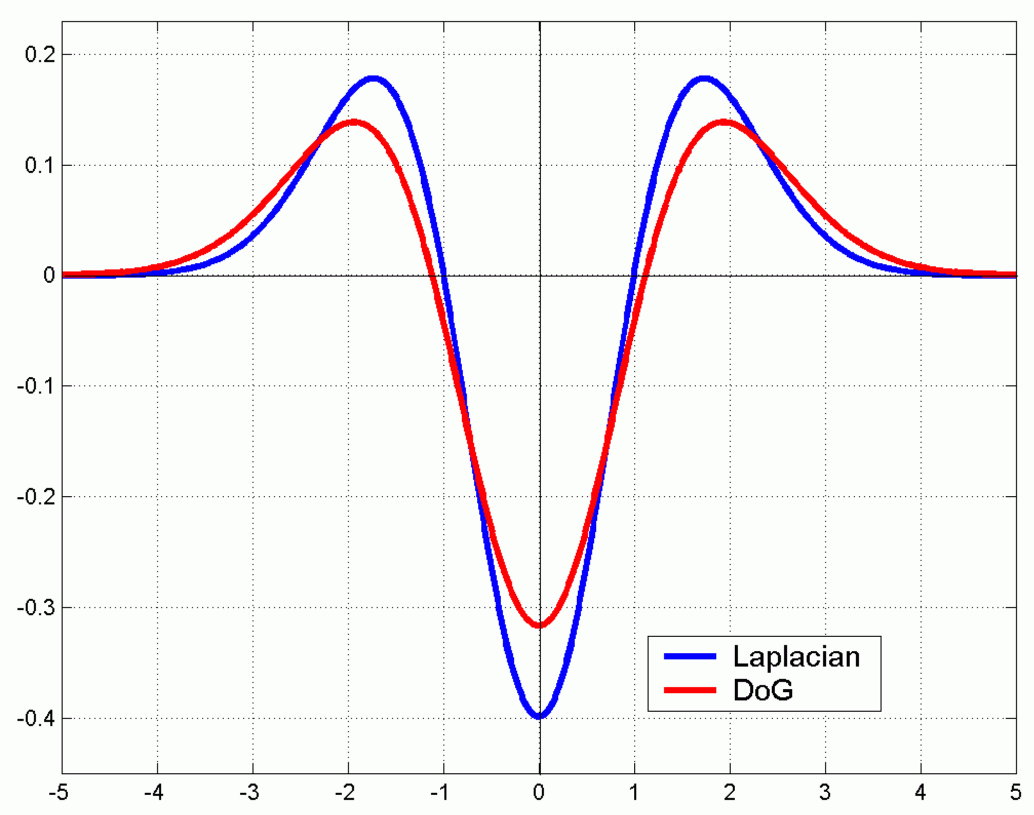


-

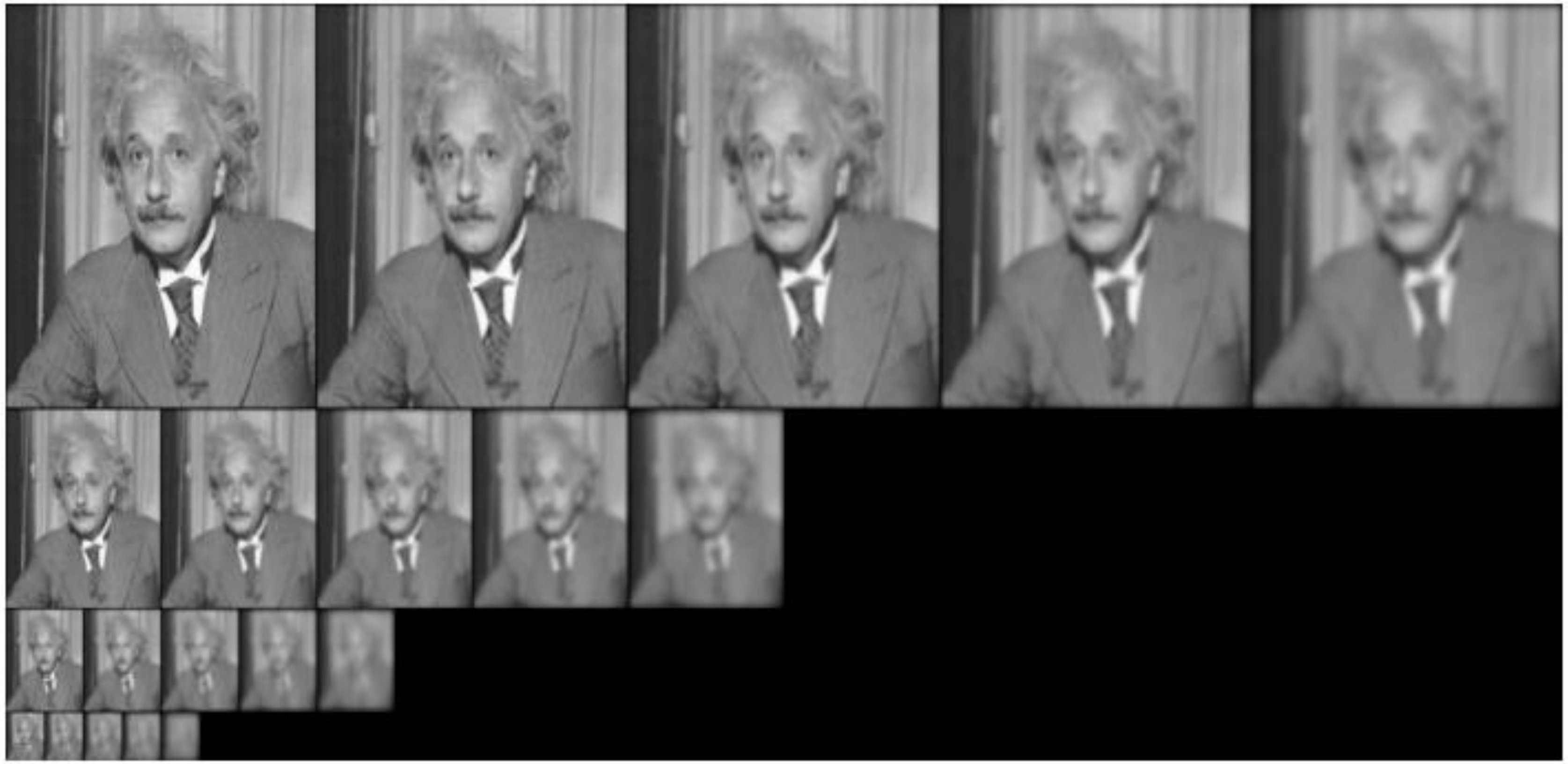


=

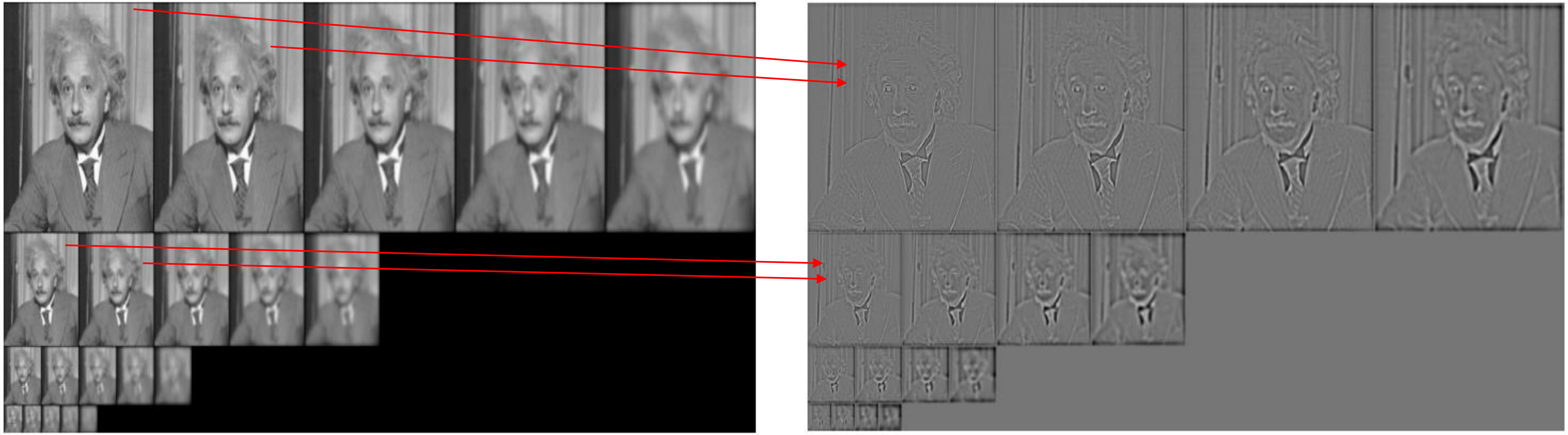




0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
-5
-4
-3
-2
-1
0
1
2
3
4
5
Laplacian
DoG

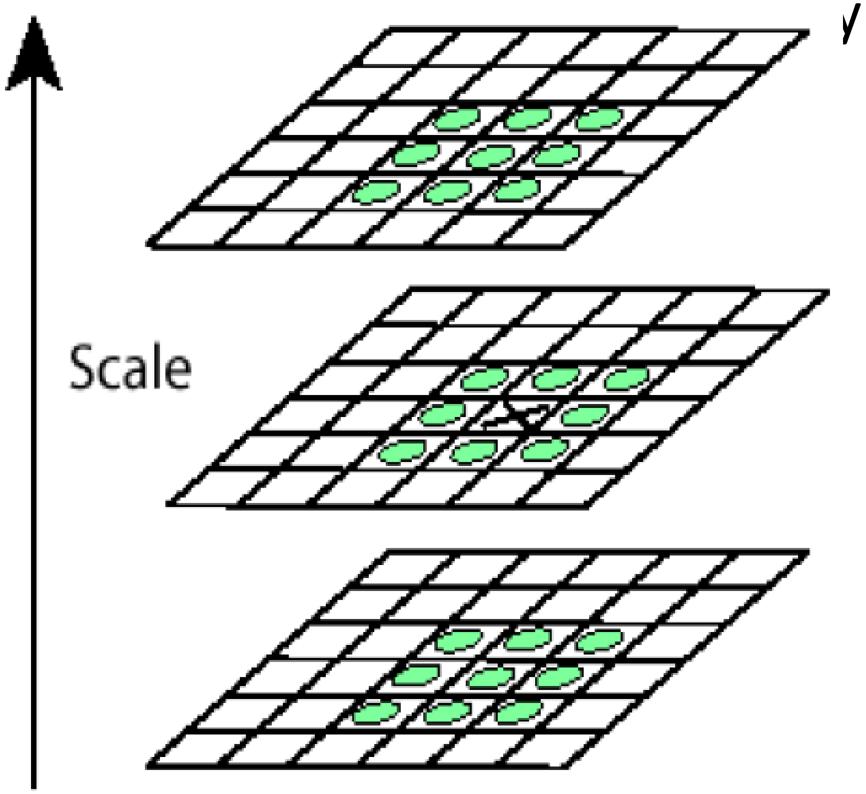


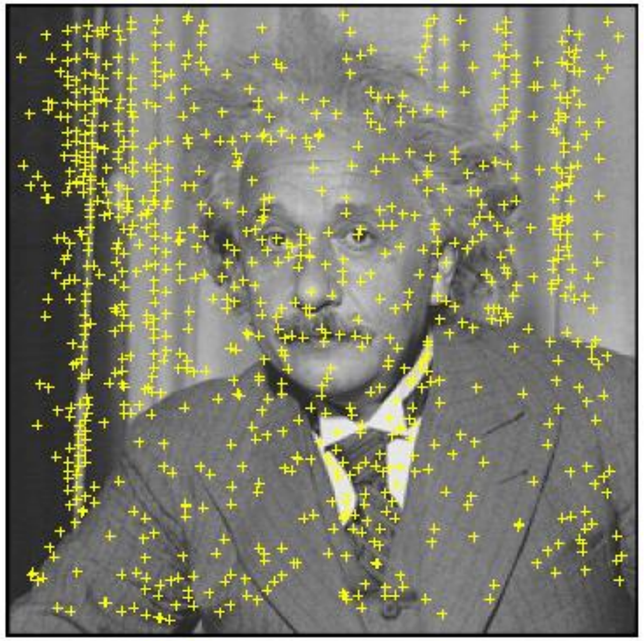
octave \ scale →				
0.707107	1.000000	1.414214	2.000000	2.828427
1.414214	2.000000	2.828427	4.000000	5.656854
2.828427	4.000000	5.656854	8.000000	11.313708
5.656854	8.000000	11.313708	16.000000	22.627417



Scale-space Extrema

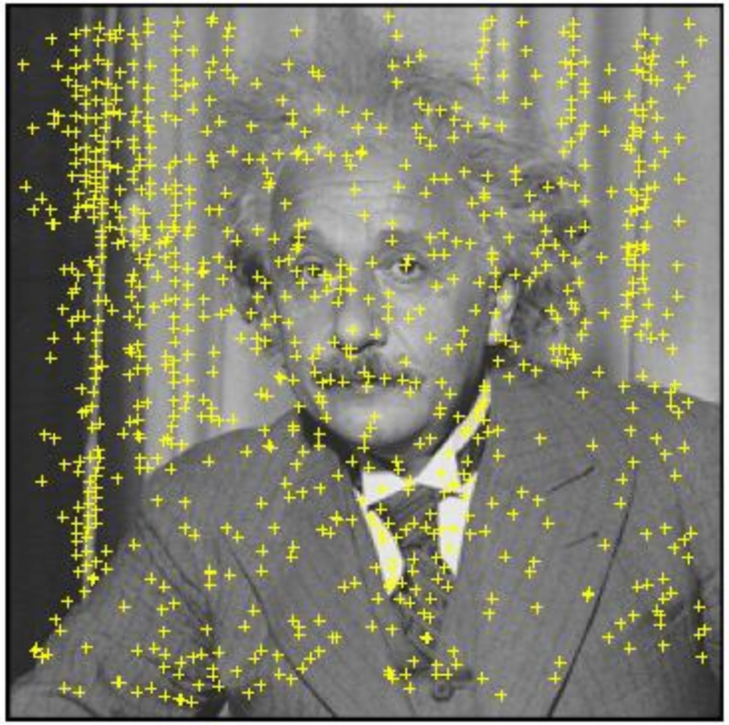
- Local maxima/mimima are detected in 3x3x3 neighborhoods
- Spans adjacent DoG images across scales.
- Interpolation of nearby data determine its position.
- Keypoints with low contrast
- Responses along edges are
- The keypoint is assigned an
  - gradient orientation histogram

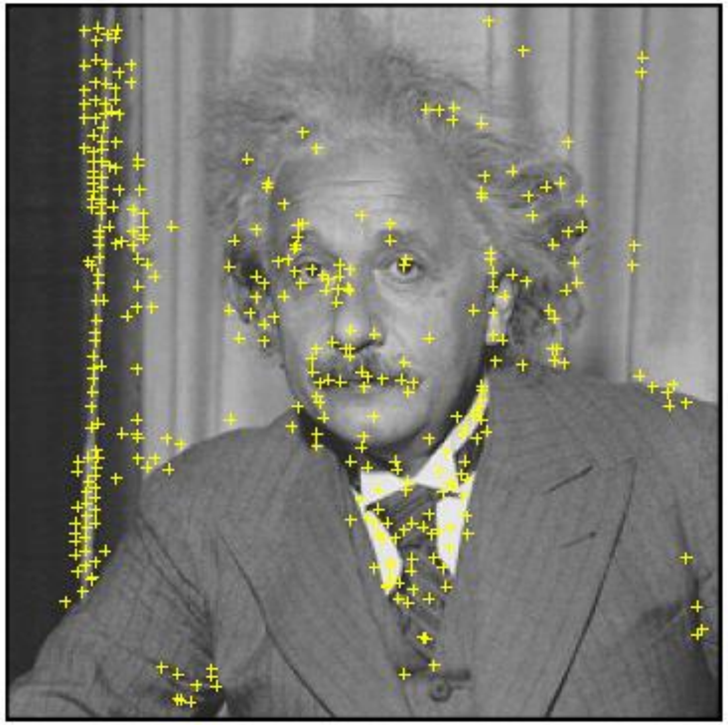




Reject Low Contrast Points

- Reject points with bad contrast:
  - D < 0.03 (image values [0,1])





Reject Points along Edges

- Problem: Strong responses along edges
- Solution eliminate edge points
- Use Hessian to determine edges

$$H = \begin{bmatrix} I_x^2 & I_{xy} \\ I_{xy} & I_y^2 \end{bmatrix}$$

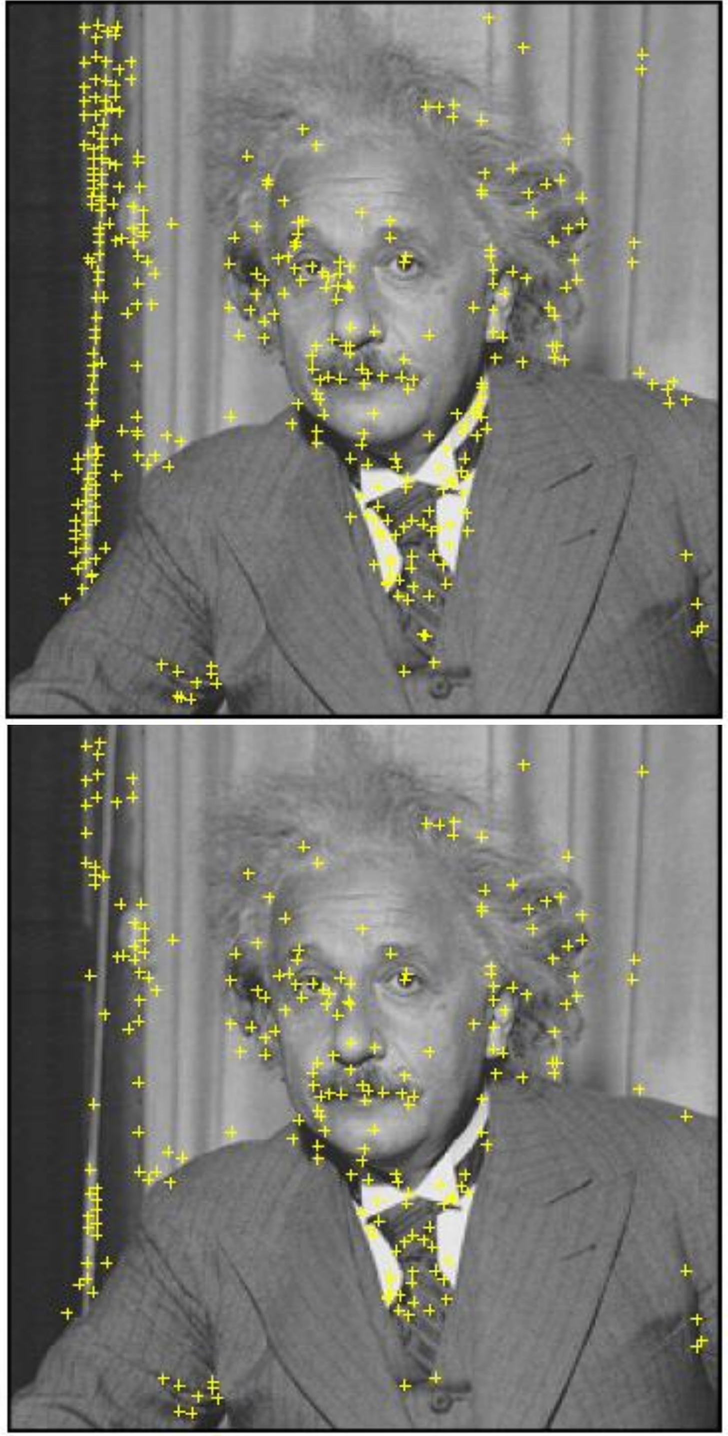
- $I_x = D \otimes \begin{bmatrix} -1 \\ 1 \end{bmatrix}$ = derivative along x
- $I_y = D \otimes [-1\ 1]$ = derivative along y
- $I_x^2 = I_x\, I_x,$
- $I_y^2 = I_y\, I_y\ ,$
- $I_{xy} = I_x\, I_y$

- Edge point: $\lambda_1 \gg \lambda_2$
  - Use a measure :

$$r = \frac{trace^2}{determinant} = \frac{(\lambda_1 + \lambda_2)^2}{\lambda_1 \lambda_2}$$

  - reject point if : $\lambda_1 > 10\lambda_2$:

$$r > \frac{(10+1)^2}{10} = 12.1$$

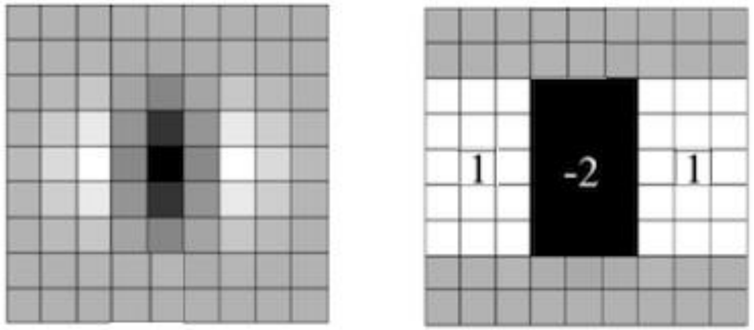


SURF

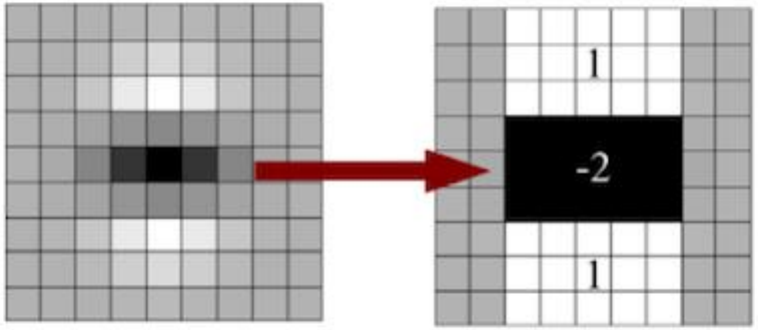
Speed Up Robust Feature (SURF)

- Improvements to SIFT
- Approximates LoG with Box Filter
  - SIFT used DoG
  - 9x9 Haar-like wavelet response
  - Convolution with box filter can be easily calculated from integral images.
- Scale up the filter instead of resizing image
  - Double filter
  - Search for extrema in 3x3x3 neighborhood.
- Determinant of Hessian matrix for both scale and location
  - $det\{H\} = I_{xx}I_{yy} - \left(wI_{xy}\right)^2$
  - w = 0.9

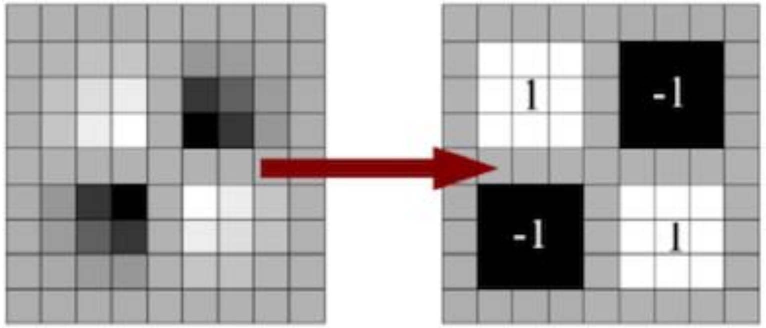
$I_x \, I_x$



$I_y \, I_y$



$I_x \, I_y$



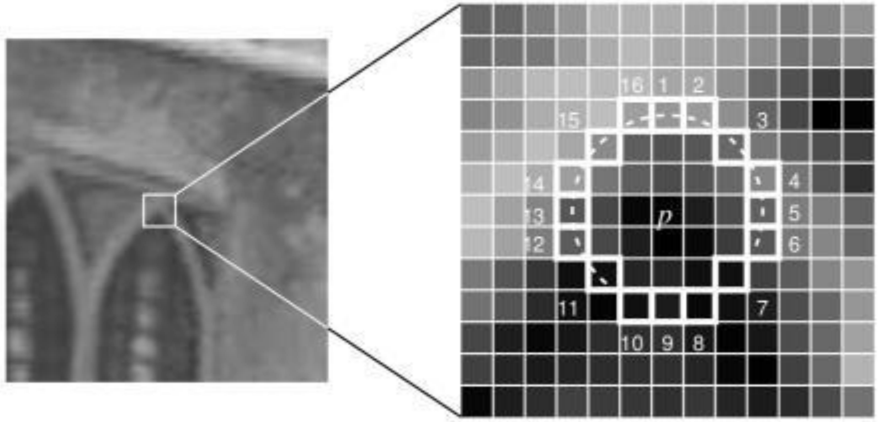
FAST

Features from accelerated Segment Test (FAST)

- Focuses on real-time application
- Corner detection technique
- Criteria for corner:
  - Each pixel, p_i, has a circle of 16 pixels around it, p_{i_1} to $p_{i_{16}}$
    - Radius = 3
  - p_i is a corner if at least 12, p_{i_j} are brighter or darker than a threshold

$$\left|p_i - p_{i_j}\right| > Threshold$$

- High speed test:
  - Examine only the four pixels at j = 1, 9, 5 and 13
  - At least 3 must be brighter/darker



Machine Learning

- Machine learning for better performance
- Weaknesses of previous algorithm: High acceptance rate: not inclusive of all j candidates
- Solution: Train for optimal performance
  1. Select a training data set
     - Identify corners
  2. Run FAST for all 16 neighbors on a circle
     - Store all 16 neighbor values
  3. Process neighbors into 3 states: darker (d=-1), similar (0), brighter (1)
  4. Use decision tree classifier
     - find the neighbors that are most important to determine if p_i is a corner
  5. Use decision tree for test images

BRIEF

Descriptor Vectors

- Feature vectors can be extremely large
  - SIFT: 128 values to describe each feature
  - Values are floating point numbers
  - Single precision = 4B/value, Double precision = 8 B/value
- Use dimension reduction techniques
  - Principal component analysis (PCA)
  - Linear discriminant analysis (LDA)
- Use hashing methods to convert floating point values into binary strings
  - Locality Sensitive Hashing (LSH)
  - Allows for fast comparison of features: XOR then count bits

Binary Robust Independent Elementary Features (BRIEF)

- Detecting a feature
- Create an SxS patch around the feature
- Randomly choose n random pairs in SxS
  - Uniform random selection works well
  - Gaussian works well
  - Polar
- Each pair (i,j) is given a binary value
  - P(i) >= P(j) => 0
  - P(i) < P(j) => 1
- Scale each by 2^k

ORB

Oriented FAST and Rotated BRIEF

- Developed by OpenCV Labs
  - No patented as SIFT & SURF are patented
- Use FAST to find corners
  - Apply to many scales as in SIFT
- Use Harris score to pick top corners
- New: orientation of corner
  - From corner center to weighted patch center
  - Discretized to 2pi/30 (12)
- Use BRIEF for feature vectors