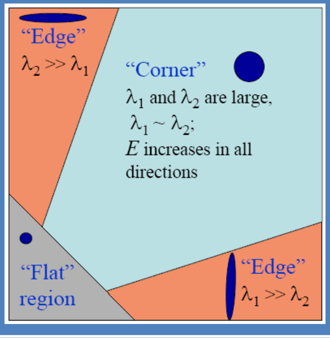
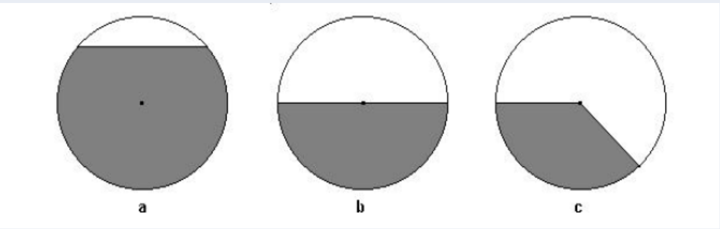
Corner 2nd

2009. 11. 16.

Jonghoon Seo
(jonghoon.seo@gmail.com)

Corner Detection

Geometry-based 	• **<u>Use the change of gradient magnitude</u>**, the surface curvature, and intensity variation occurs in every direction [1] • Image derivative required [2] • Noise Reduction required [2]	• Harris and Stephens (Plessey) • Shi and Tomasi • SIFT • Etc.
Template-based	• Determining the similarity, or correlation between a given template and all subwindows in a given image [1] • Examining a small patch of an image to see if it "look" like a corner [2] 	• SUSAN • Trajkovic and Hedley • FAST • Etc.

[1] Dongxiang Zhou; Yun-Hui Liu; Xuanping Cai, "An efficient and robust corner detection algorithm," *Intelligent Control and Automation, 2004. WCICA 2004. Fifth World Congress on* , vol.5, no., pp. 4020-4024 Vol.5, 15-19 June 2004
[2] Edward Rosten and Tom Drummond, "Machine learning for high-speed corner detection", *European Conference on Computer Vision, 2006. ECCV 2006.* pp. 430-443, May, 2006.

Harris and Stephens, 1988

Shi and Tomasi, 1994

Lowe, 2004

GEOMETRY-BASED CORNERS

Plessey - Harris and Stephens, 1988 (Brief Review)

- Computing an approximation to the second derivative of the SSD

$$\mathbf{C} = \begin{pmatrix} \sum \mathbf{I}_x^2 & \sum \mathbf{I}_x\mathbf{I}_y \\ \sum \mathbf{I}_x\mathbf{I}_y & \sum \mathbf{I}_y^2 \end{pmatrix}$$

- The two eigenvectors and eigenvalues λ_1, λ_2 of **C** encode the predominant directions and magnitudes of the gradient
- Corners are thus where min(λ_1, λ_2) is over a threshold.

- Measure of corner response:

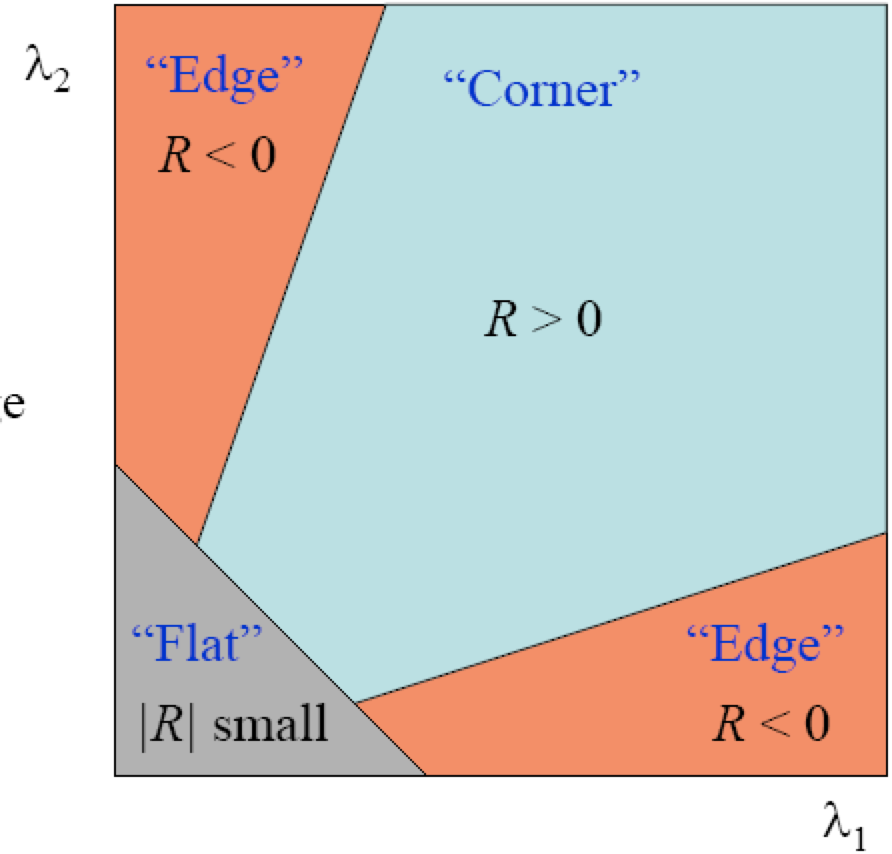
$$R = \det M - k\left(\operatorname{trace} M\right)^2$$

$$\det M = \lambda_1 \lambda_2$$
$$\operatorname{trace} M = \lambda_1 + \lambda_2$$

  – The trace of a matrix is the sum of its eigenvalues, making it an invariant with respect to a change of basis

Plessey - Harris and Stephens, 1988 (Brief Review)

- R depends only on eigenvalues of M
- R is large for a corner
- R is negative with large magnitude for an edge
- $|R|$ is small for a flat region



Shi and Tomasi, 1994 (Brief Review)

- To find location of **v** and **J** corresponding to **u** on **I**.
- How to detect **u** on **I**

$$\overline{\nu}_{\text{opt}} = G^{-1}\,\overline{b}.$$

$$G \doteq \sum_{x=p_x-\omega_x}^{p_x+\omega_x} \sum_{y=p_y-\omega_y}^{p_y+\omega_y} \begin{bmatrix} I_x^2 & I_x\, I_y \\ I_x\, I_y & I_y^2 \end{bmatrix} \quad \text{and} \quad \overline{b} \doteq \sum_{x=p_x-\omega_x}^{p_x+\omega_x} \sum_{y=p_y-\omega_y}^{p_y+\omega_y} \begin{bmatrix} \delta I\, I_x \\ \delta I\, I_y \end{bmatrix}.$$

- Find the point **u** whose **G** matrix is non-singular.
- <u>The minimum eigenvalue of **G** must larger than a threshold.</u>

Shi and Tomasi, 1994 (Brief Review)

1. Compute the G matrix and its minimum eigenvalue λ_m at every pixel in the image I.

2. Call λ_{max} the maximum value of λ_m over the whole image.

3. Retain the image pixels that have a λm value larger than a threshold.

4. Retain the local maximum pixels (a pixel is kept if its λ_m value is larger than that of any other pixel in its 3×3 neighborhood).

5. Keep the subset of those pixels so that the minimum distance between any pair of pixels is larger than a given threshold distance.

SIFT – Lowe, 2004

- There are four steps

Detector

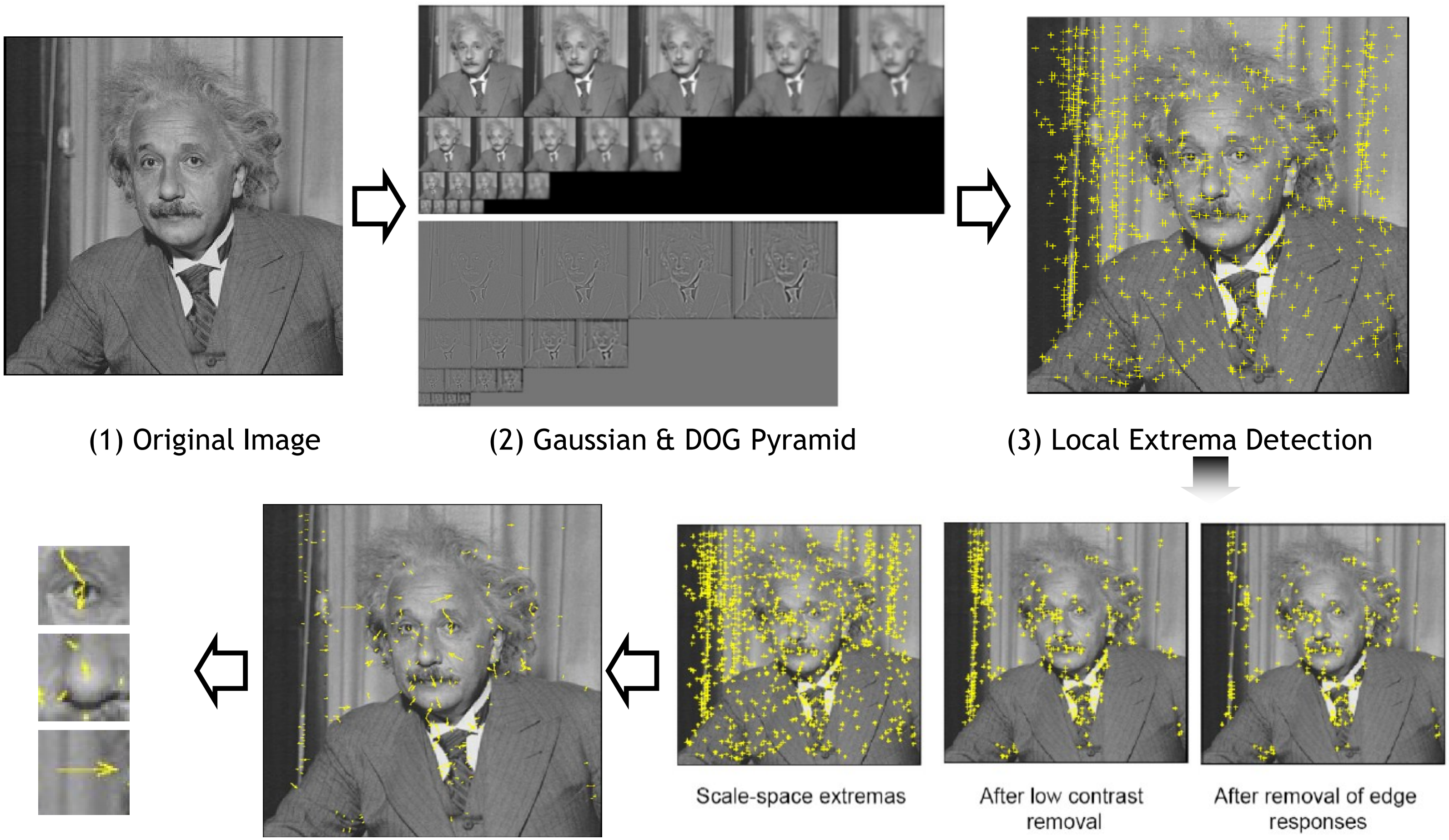
1. Find scale-space Extrema
   - Identify location and scale
2. Keypoint Localization & Filtering
   - Improve keypoints and throw out bad ones

Descriptor

3. Orientation Assignment
   - Remove effects of rotation and scale
4. Create descriptor
   - Using histograms of orientations

SIFT – Lowe, 2004

- SIFT Overview



(1) Original Image (2) Gaussian & DOG Pyramid (3) Local Extrema Detection

(6) Keypoint Descriptor (5) Keypoints with Orientation (4) Keypoint Localization

1. Scale-space extrema detection

- To indentify **locations and scales** that can be repeatably assigned under differing views of the same object.

- Detecting locations that are invariant to scale change of the image can be accomplished by searching for stable features across all possible scales, **using a continuous function of scale known as scale space** [Witkin, 1983]

•Witkin, A. P. 1983, "Scale-space filtering", In International Joint Conference on Artificial Intelligence, Karlsruhe, Germany, pp. 1019-1022

Scale Space

- Need to find 'characteristic scale' for feature
- Scale-Space: Continuous function of scale σ [Witkin, 1983]
  - Only reasonable kernel is Gaussian: [Koenderink 1984, Lindeberg 1994]

Scale space of image is defined as a function:

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y)$$

(Gaussian image)

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}$$

•Witkin, A. P. 1983, "Scale-space filtering", In International Joint Conference on Artificial Intelligence, Karlsruhe, Germany, pp. 1019-1022.
•Koenderink, J. J., 1984, "The structure of images", Biological Cybernetics, 50:363-396.
•Lindeberg, T. 1994., "Scale-space Theory: A basic tool for analysing structures at different scales", Journal of Applied Statistics, 21(2): 224-270.

Scale Selection

- Experimentally, Maxima of **<u>Laplacian-of-Gaussian</u>** gives **best notion of scale**: [Mikolajczyk, 2002]

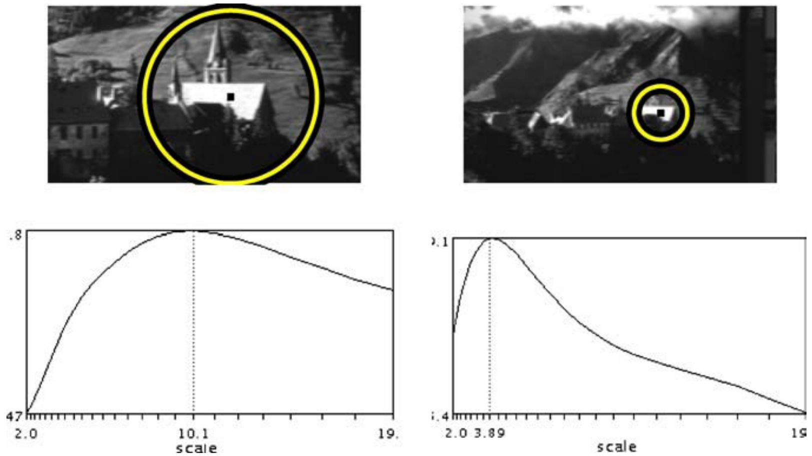


Figure 6. Example of characteristic scales. **Top row** shows **images taken with different zoom**. **Bottom row** shows **the responses of the Laplacian over scales**. The characteristic scales are 10.1 and 3.9 for the left and right image, respectively. The ratio of scales corresponds to the scale factor (2.5) between the two images. The radius of displayed regions in the top row is equal to 3 times the selected scales.

- Thus use Laplacian-of-Gaussian (LoG) operator:

$$\sigma^2 \nabla^2 G$$

Mikolajczyk, et al, “A Comparison of Affine Region Detectors”, IJCV, Vol. 65, n. 1-2, pp. 43-72 2002

Approximate LoG

- LoG is expensive, so let's approximate it
- Using the heat-diffusion equation:

$$\sigma \nabla^2 G = \frac{\partial G}{\partial \sigma} \approx \frac{G(k\sigma) - G(\sigma)}{k\sigma - \sigma}$$

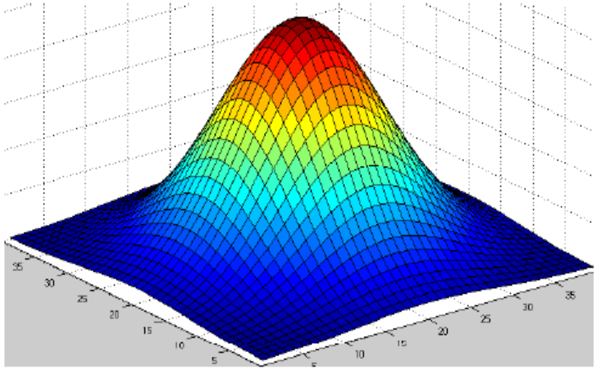
- Define Difference-of-Gaussians (DoG):

$$(k-1)\sigma^2 \nabla^2 G \approx G(k\sigma) - G(\sigma)$$

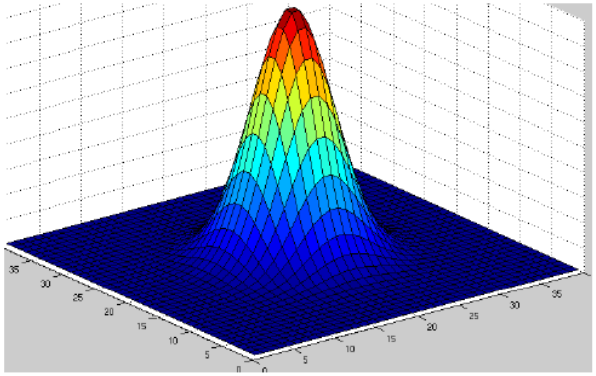
$$D(\sigma) \equiv \left(G(k\sigma) - G(\sigma)\right) * I$$

DoG Efficiency

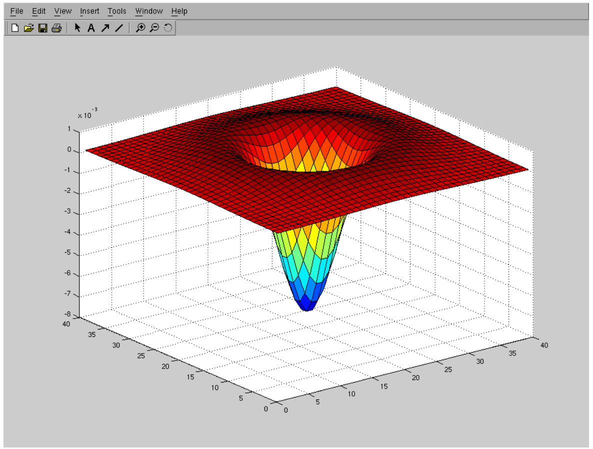
- The smoothed images need to be computed in any case for feature description.
- We need **<u>only to subtract two images</u>**.



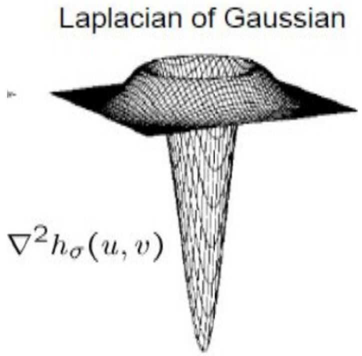
\-



=

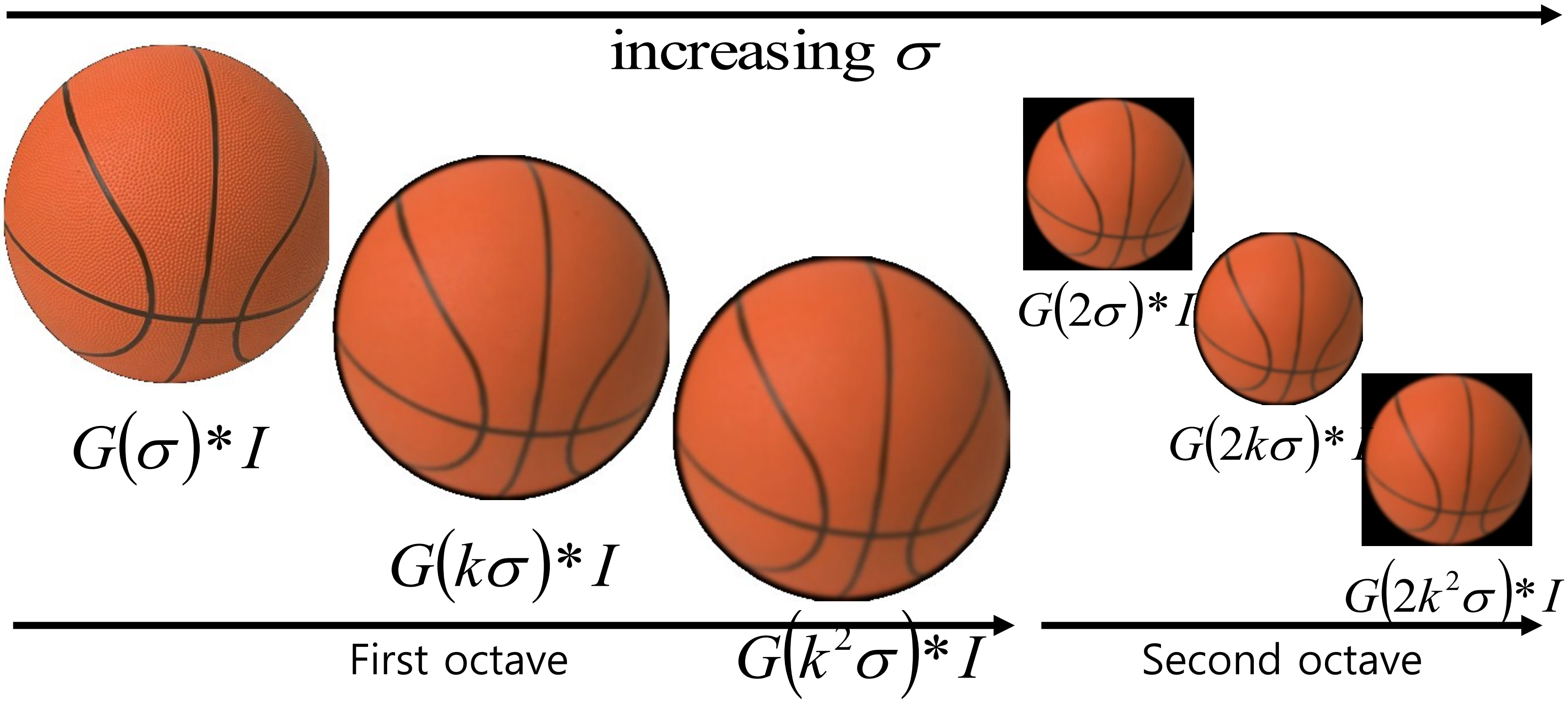


≡

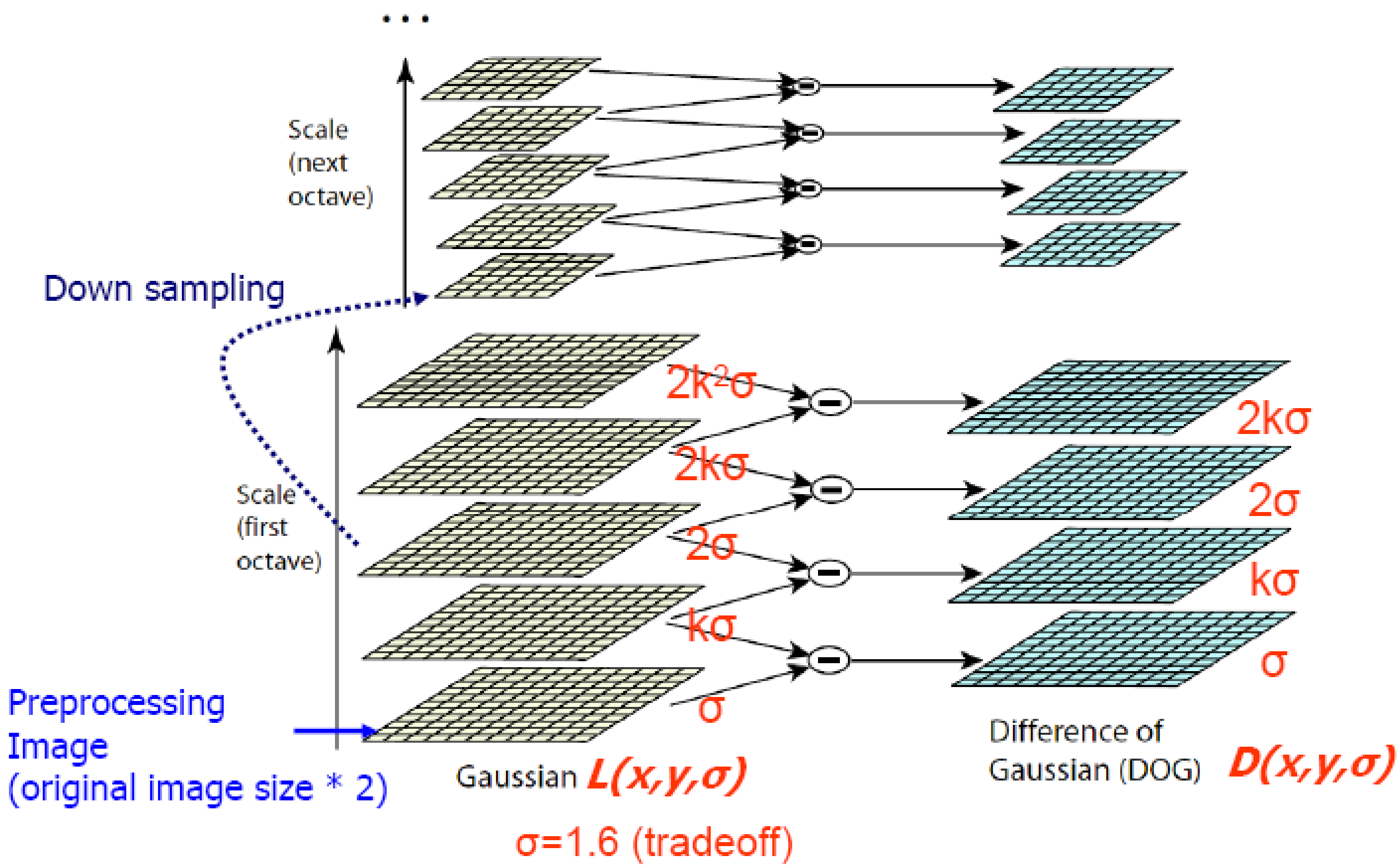


Scale-Space Construction

- First construct scale-space:

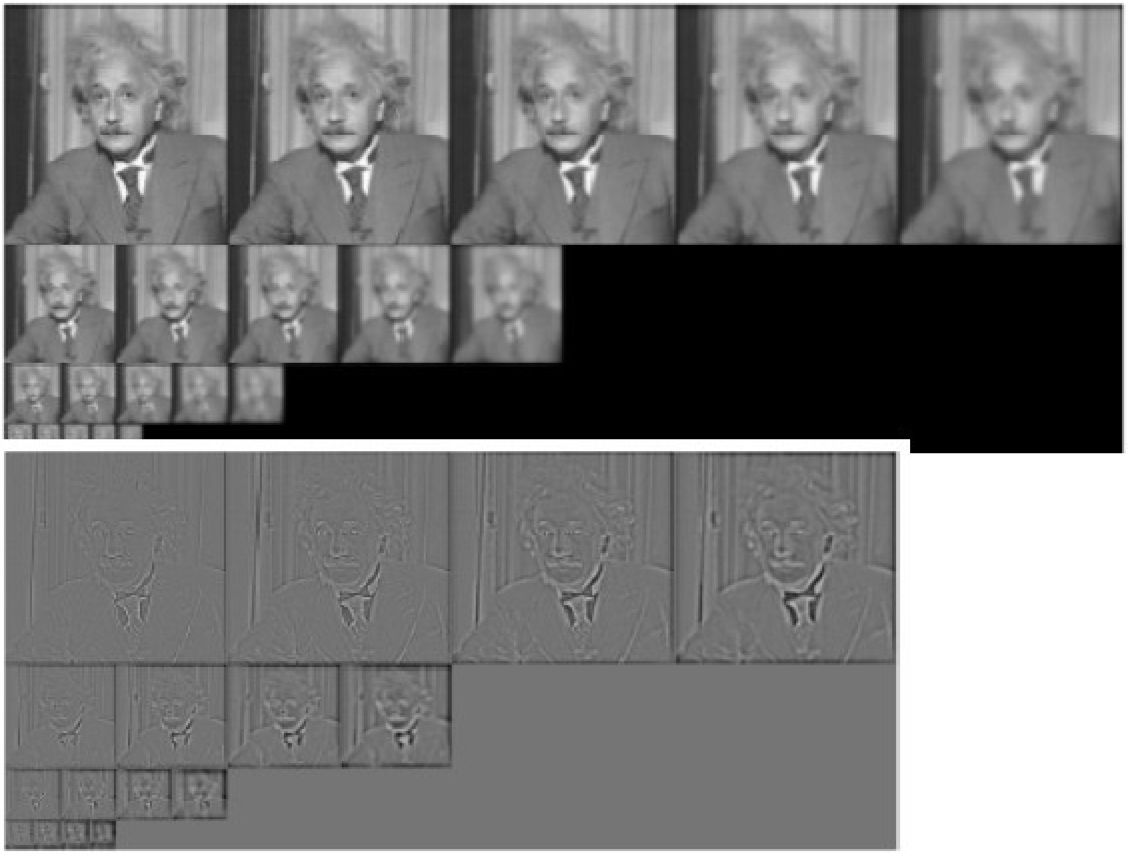


Scale-Space Construction



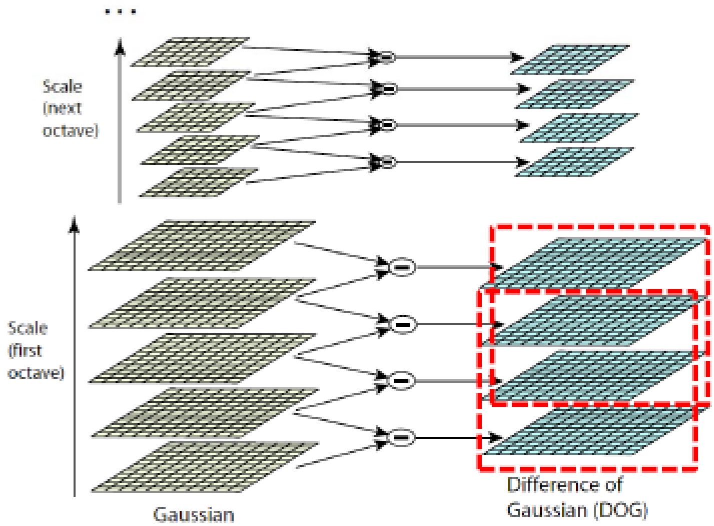
This diagram shows the gaussian and DOG pyramids

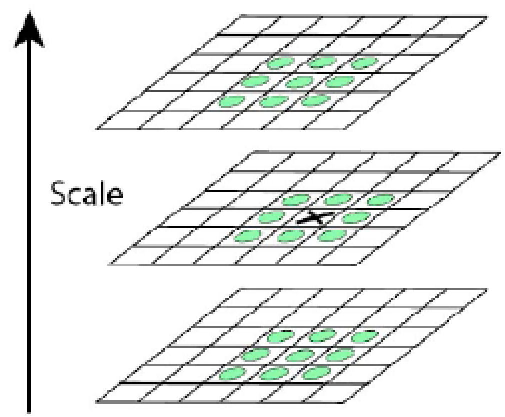
Scale-Space Construction



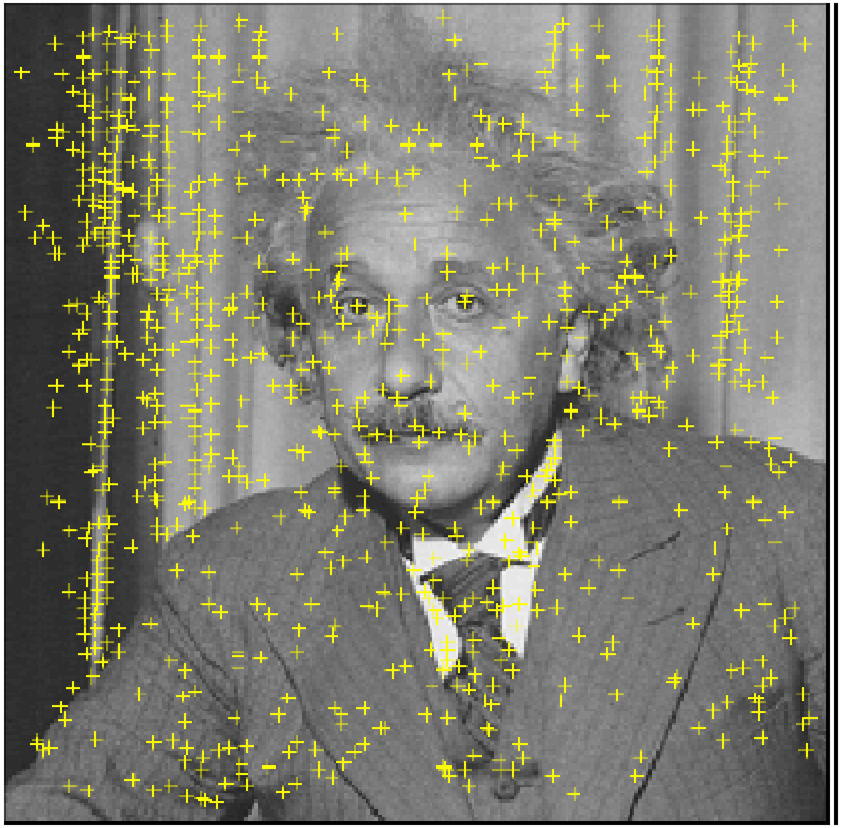
Scale-space extrema detection

- **Scale-space extrema detection**
  - Maxima and minima of the DoG are detected by comparing a pixel
  - Pixel marked with "x" is compared to 26 neighbors in a 3x3x3 window that spans adjacent pixels and scales
  - Sample point is selected only if it is a minimum or a maximum of these points



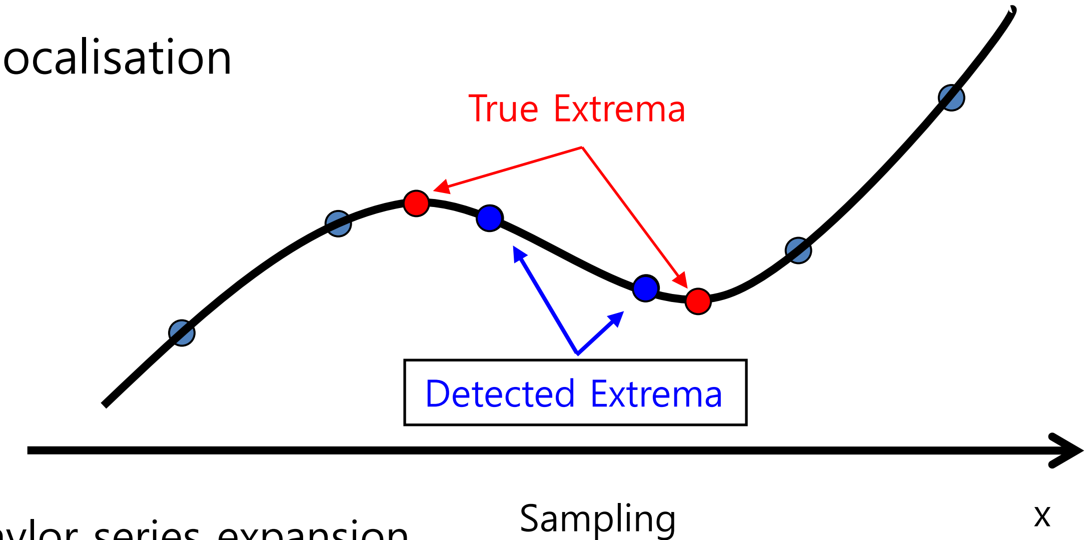


Scale-space extrema detection



2. Accurate keypoint localization

- Keypoint localisation



  - Take Taylor series expansion

$$D(\mathbf{x}) = D + \frac{\partial D}{\partial \mathbf{x}}^T \mathbf{x} + \frac{1}{2}\mathbf{x}^T \frac{\partial^2 D}{\partial \mathbf{x}^2}\mathbf{x} \qquad (2)$$

D and derivatives are evaluated at the sample point and $\mathbf{x} = (x, y, \sigma)^T$ is the offset from this point

  - Minimize to get true location of extrema:

$$\hat{x} = -\frac{\partial^2 D}{\partial \vec{x}^2}^{-1} \frac{\partial D}{\partial \vec{x}}$$

2. Accurate keypoint localization

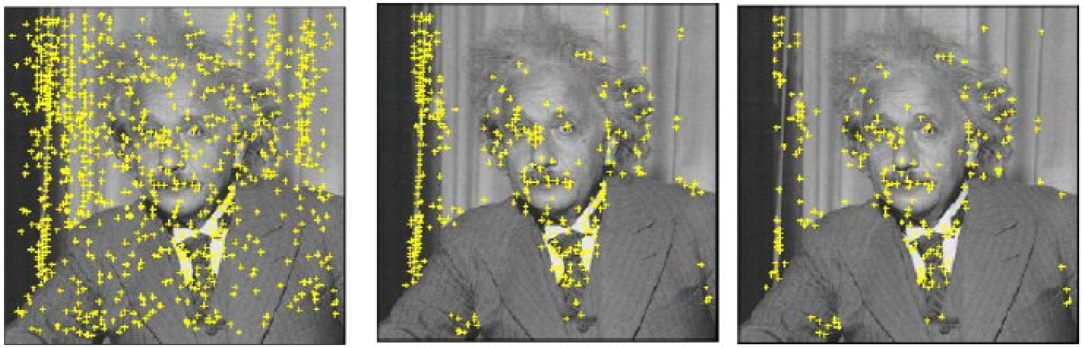
- Filtering
  - Remove the **low contrast**
    - The function value at the extremum is useful for rejecting unstable extrema with low contrast.

$$D(\hat{\mathbf{x}}) = D + \frac{1}{2}\frac{\partial D}{\partial \mathbf{x}}^T \hat{\mathbf{x}}.$$

  - Throw out low contrast:

$D(\hat{x})$ < 0.03 (image values in [0, 1])

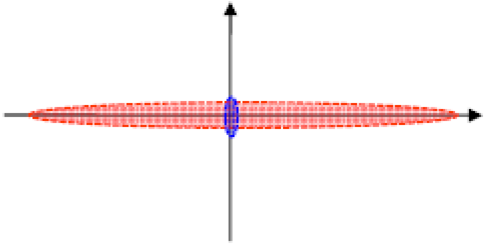
  - Remove the **edge responses**



Scale-space extremas | After low contrast removal | After removal of edge responses

2. Accurate keypoint localization

- A poorly defined peak in the DOG function will have a large principal curvature across the edge but a small one in the perpendicular direction.



- Principal curvature can be computed from a 2x2 Hessian matrix

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

$$D_{xx} = D(x+1, y) + D(x-1, y) - 2D(x, y)$$
$$D_{yy} = D(x, y+1) + D(x, y-1) - 2D(x, y)$$
$$D_{xy} = \left(D(x+1, y+1) - D(x-1, y+1) - D(x+1, y-1) + D(x-1, y-1)\right)/4$$

  – The eigenvalues of H are proportional to the principal curvatures of D.

2. Accurate keypoint localization

- α be the **eigenvalue** with the **largest** magnitude
- β be the **smaller** one
- γ be the ration between the largest magnitude eigenvalue and the smaller one.

  $\alpha = r\beta$

- Using **Harris and Stephens(1998)** Approach

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

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

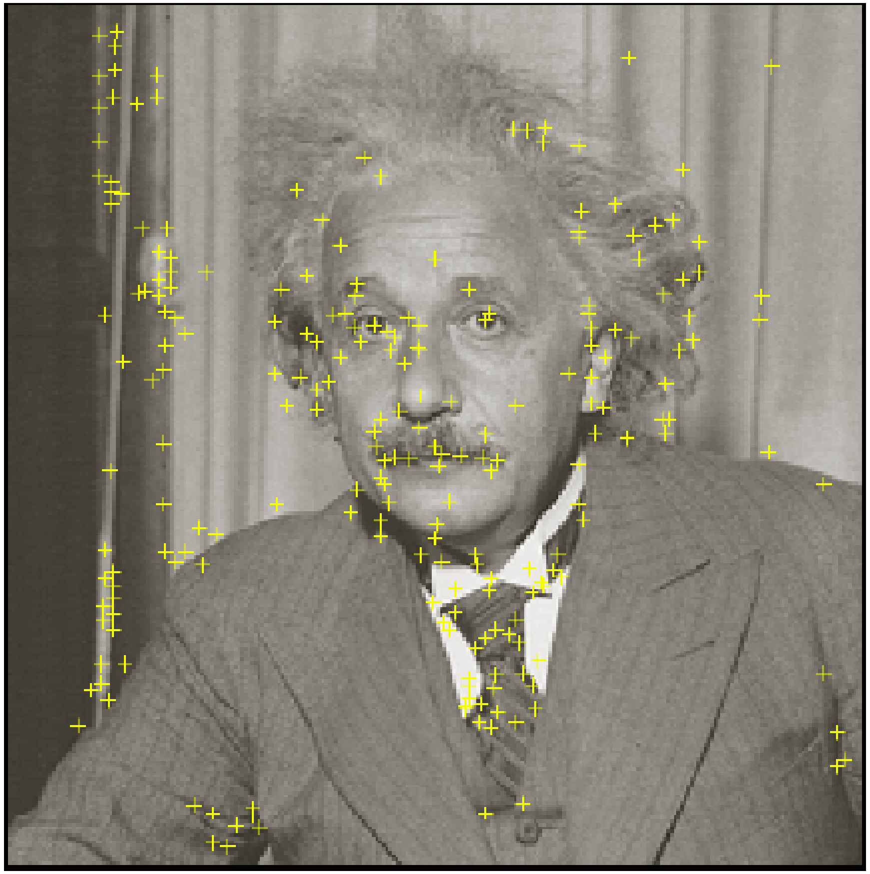
  $$\frac{Tr(H)^2}{Det(H)} = \frac{(\alpha+\beta)^2}{\alpha\beta} = \frac{(r\beta+\beta)^2}{r\beta^2} = \frac{(r+1)^2}{r}$$

  → will be **minimum** when **two eigenvalues are equal**, it **increases** with **r**

- Check that the ratio of principal curvatures is below some threshold r (r = 10)

  $$\frac{Tr(H)^2}{Det(H)} < \frac{(r+1)^2}{r}$$

2. Accurate keypoint localization



SIFT – Lowe, 2004

- There are four steps

Detector

1. Find scale-space Extrema
    - Identify location and scale
2. Keypoint Localization & Filtering
    - Improve keypoints and throw out bad ones

Descriptor

3. Orientation Assignment
    - Remove effects of rotation and scale
4. Create descriptor
    - Using histograms of orientations

3. Orientation Assignment

- Now we have set of good points
- Choose a region around each point
  - Remove **effects of scale and rotation**

- Orientation Assignment
  - Use **scale** of point **to choose correct image**:

$$L(x,y) = G(x,y,\sigma) * I(x,y)$$

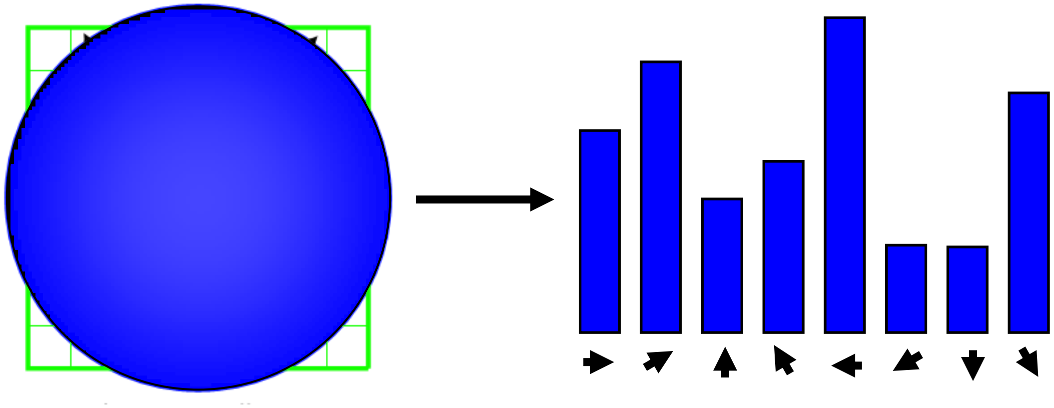
  - **Compute gradient magnitude** and **orientation** using finite differences:

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

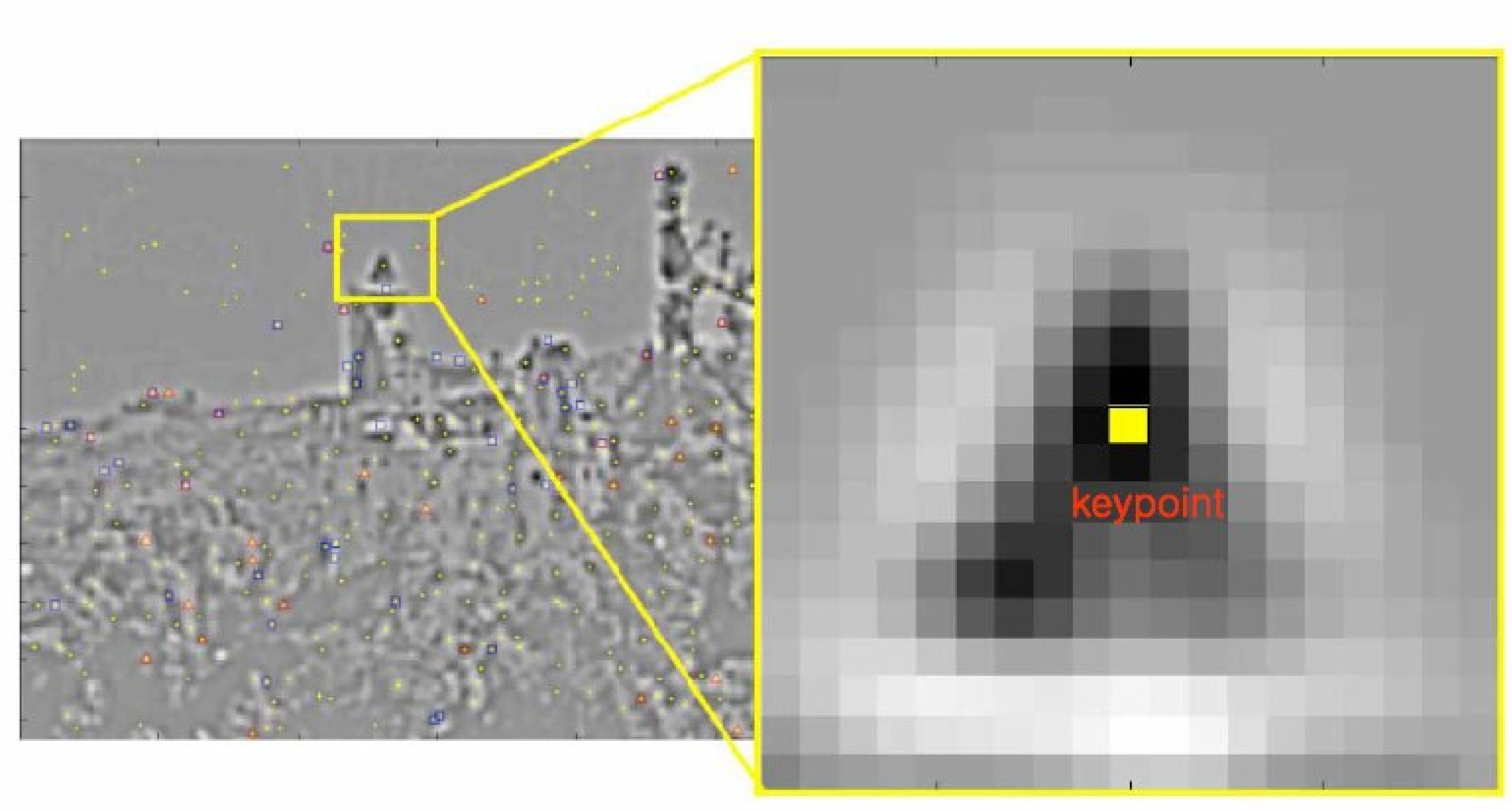
$$\theta(x,y) = \tan^{-1}\left(\frac{(L(x,y+1) - L(x,y-1))}{(L(x+1,y) - L(x-1,y))}\right)$$

3. Orientation Assignment

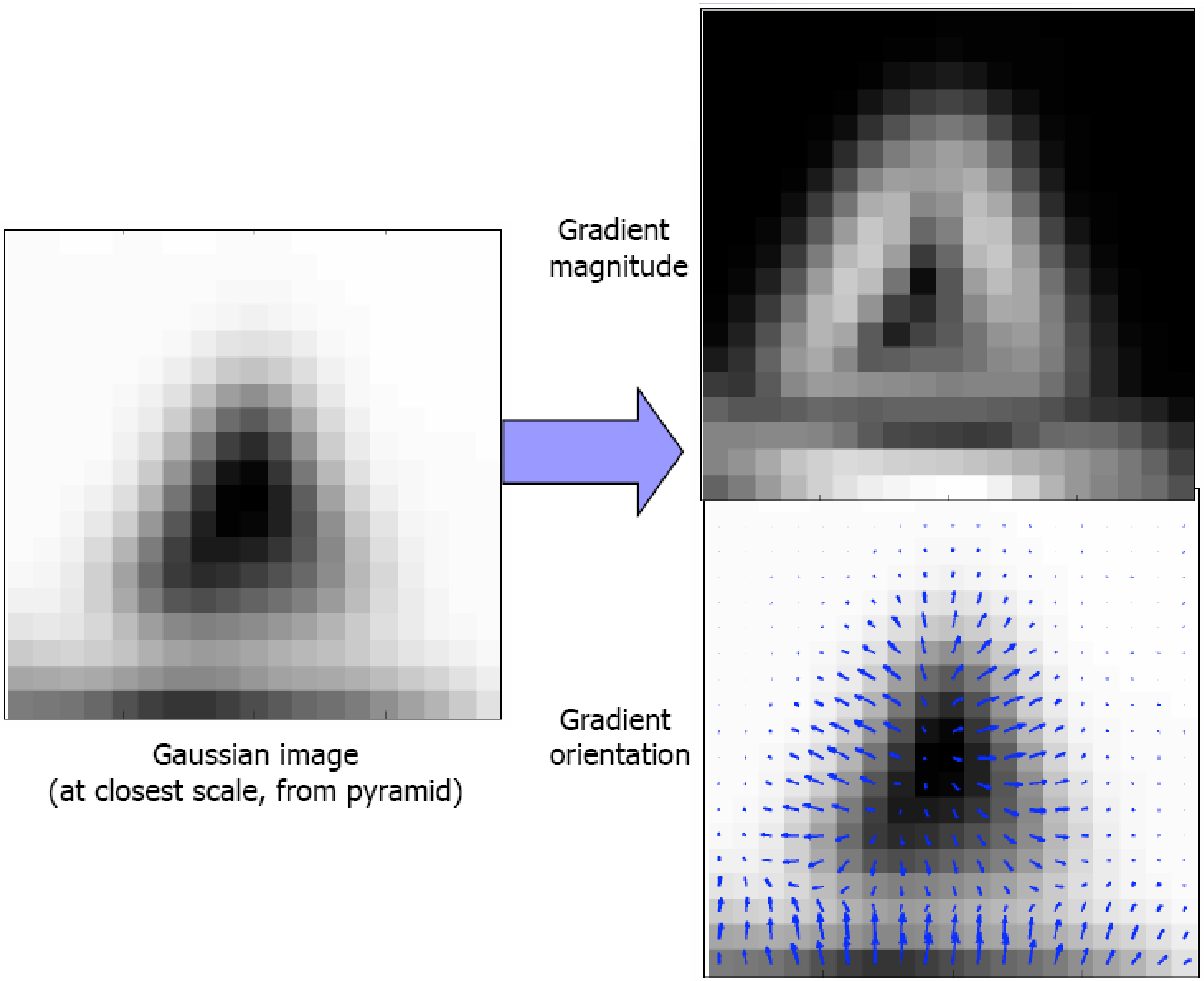
- Create **gradient histogram** (36 bins)
  - 16 by 16 matrix: magnitude, orientation
  - Orientation → 36 bins(0~10, 10~20, …)
  - Weighted by magnitude:
  - Gaussian-weighted circular window
    (σ is 1.5 times that of the scale of a keypoint)



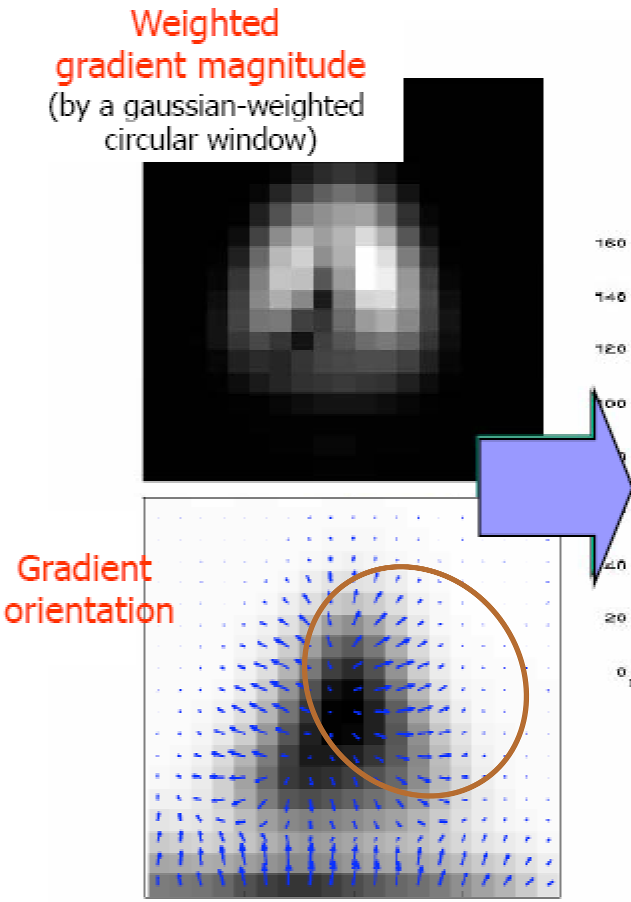
3. Orientation Assignment



3. Orientation Assignment



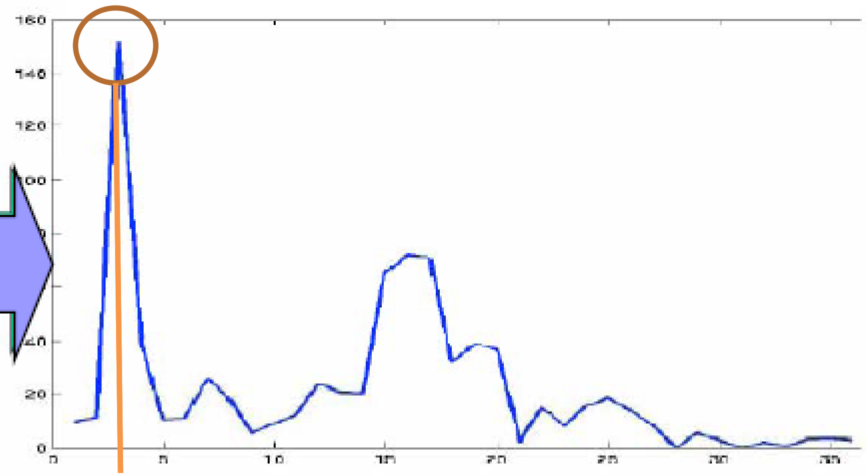
3. Orientation Assignment



Weighted orientation histogram

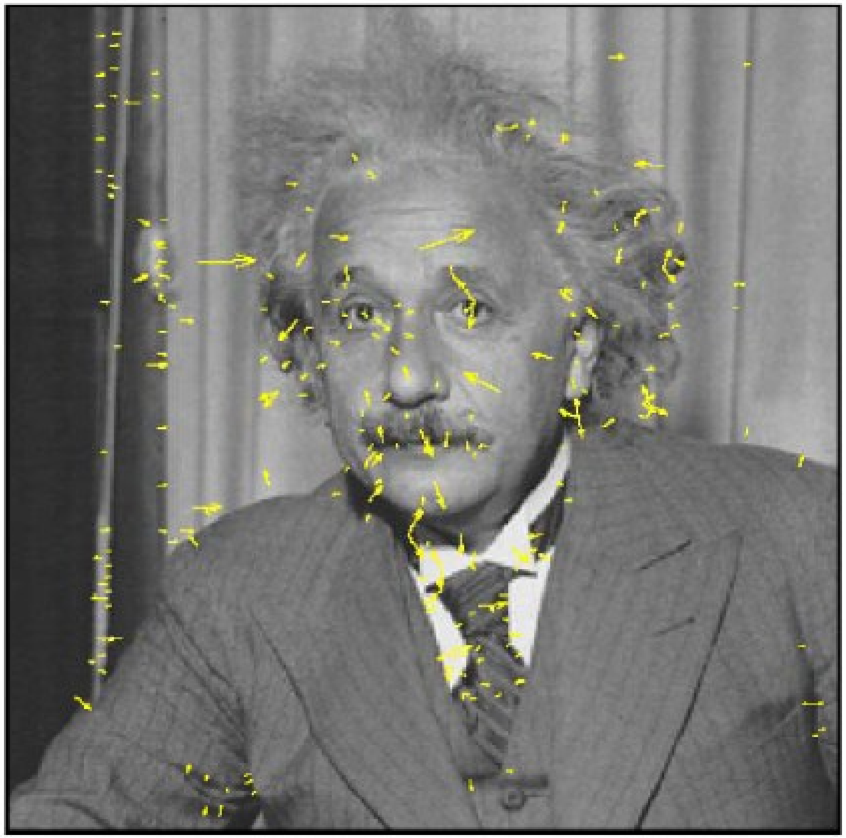
-If the magnitude over the 80% of peak value,
we select the all orientation

-A keypoint may be assigned more than one orientation.



Orientation of keypoints is approximately 25 degrees

3. Orientation Assignment



4. Creating Descriptor

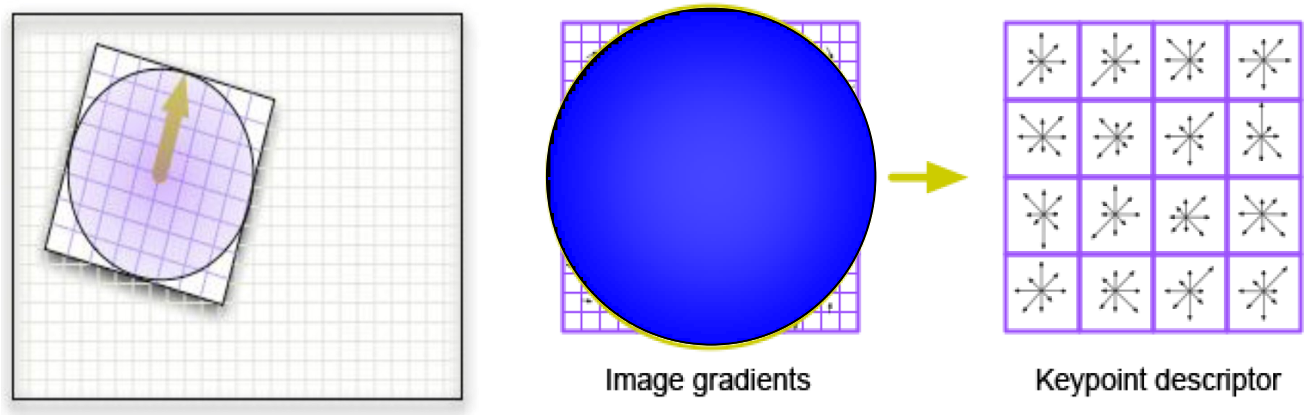
- Each point so far has x, y, σ, m, θ

- Now we need a descriptor for the region
  - Could **sample intensities** around point, but…
    - Sensitive to lighting changes
    - Sensitive to slight errors in x, y, θ
    - Sensitive 3D viewpoint change(affine,perspective…)
  - Look to **biological vision**
    - [Edelman, Intrator & Poggio, 1997]
    - Complex neurons in primary visual cortex
    - Complex neurons respond to gradients at certain frequency and orientation
      - But location of gradient can shift slightly!

- The descriptors are a grid of gradient orientation histograms, where the sampling grid for the histograms is rotated to the main orientation of each keypoint.

4. Creating Descriptor

- 4x4 Gradient window
- Histogram of 4x4 samples per window in 8 directions
- Gaussian weighting around center(σ is 0.5 times that of the scale of a keypoint)
- 4x4x8 = 128 dimensional feature vector



Smith and Brady, 1997

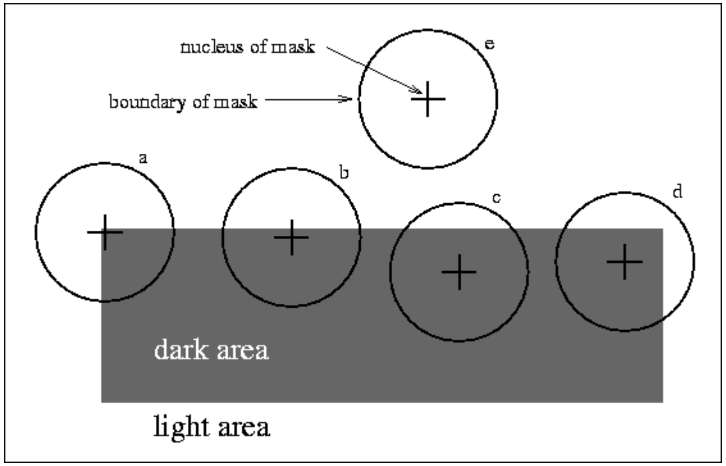
Trajkovic and Hedley, 1998

Rosten and Drummond, 2006

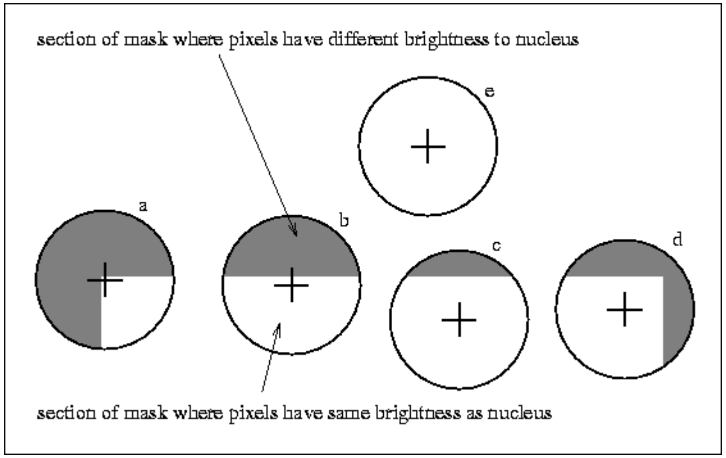
TEMPLATE-BASED CORNERS

SUSAN – Smith and Brady, 1997

- Masking based approach

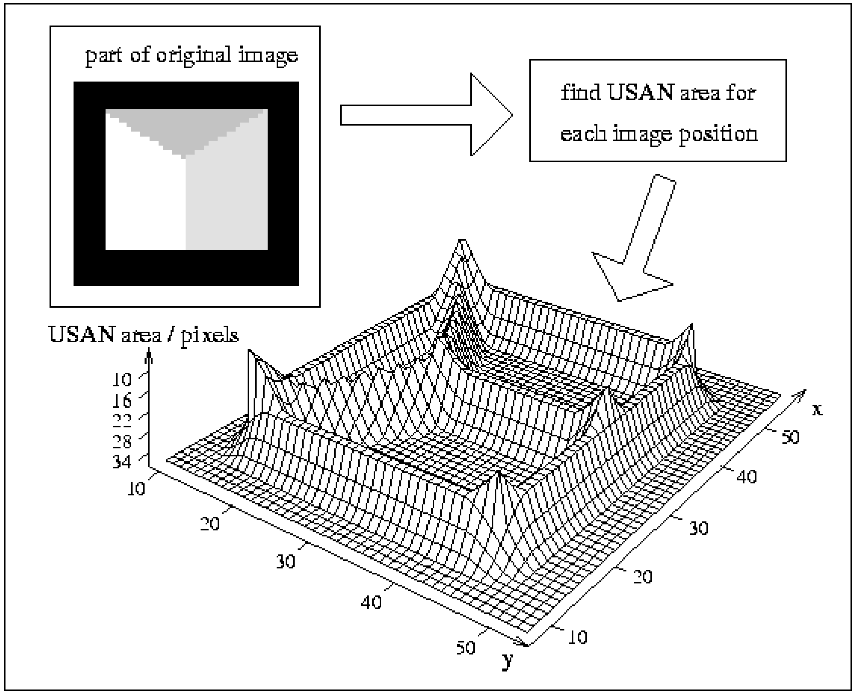


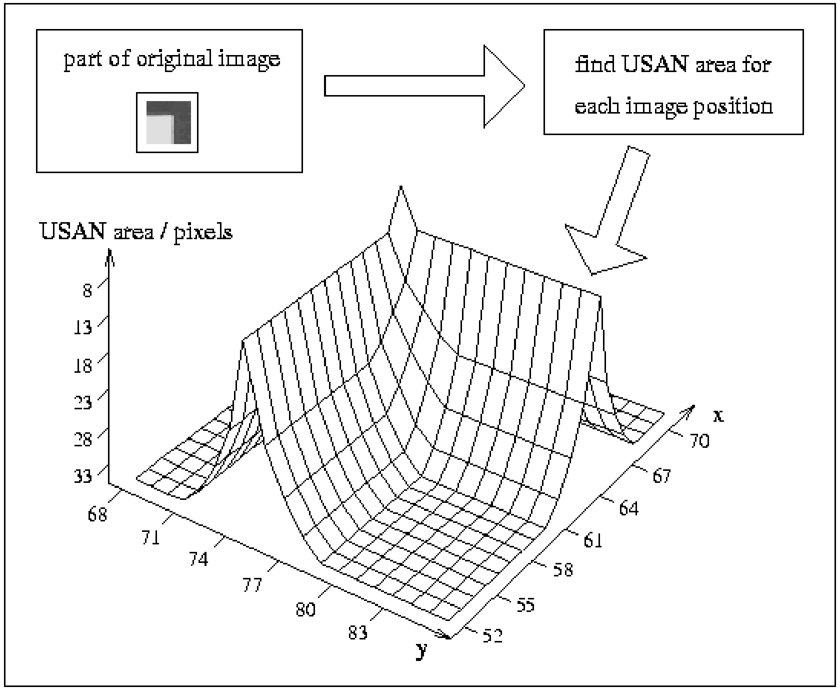
- USAN (Univalue Segment Assimilating Nucleus)



SUSAN – Smith and Brady, 1997

- USAN Example





- An image processed to give as output **<u>inverted USAN area</u>** has **<u>edges and two dimensional features</u>** strongly enhanced, with the two dimensional features more strongly enhanced than edges.

SUSAN – Smith and Brady, 1997

- **SUSAN Corner Detector**

1. Place a circular mask around the pixel in question (the nucleus).

2. Calculate **the number of pixels within the circular mask** which have similar brightness to the nucleus. (These pixels define the USAN).

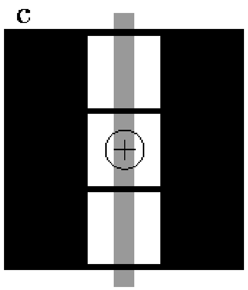
3. **Subtract the USAN size from the geometric threshold** (set lower than when finding edges) to produce a corner strength image.

4. **Test for false positives** by finding the USAN's centroid and its contiguity.

5. Use **non-maximum suppression** to find corners.

SUSAN – Smith and Brady, 1997

- False positive problems
  - This can occur with real data where **blurring of boundaries** between regions occurs.
  - Below case where there is **<u>a thin line</u>** with a brightness approximately half way between the two surrounding regions. A thin line such as this will **<u>usually be broken up</u>**, and, being usually only one pixel thick, it may cause corners to be wrongly reported.



  - Solution
    - Find the USAN's **<u>centroid</u>** and its **<u>contiguity</u>**.

Trajkovic and Hedley, 1998

- An arbitrary line *l* containing the nucleus and intersecting the boundary of the circular window at two opposite points *P* and *P′*

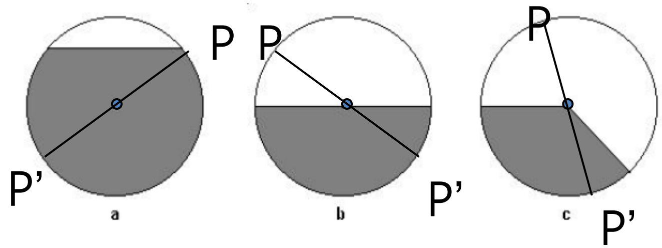


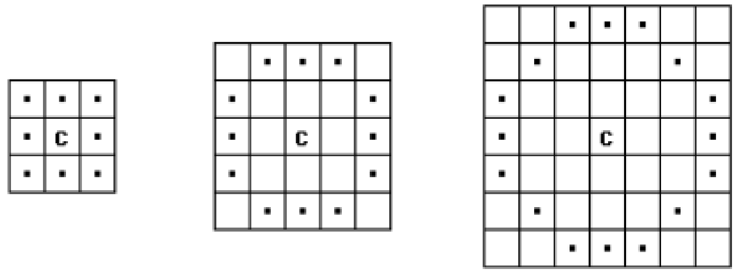
Figure 3.1: Representative shapes of USAN: a) nucleus is within the USAN; b) nucleus is an edge point; c) nucleus is a corner point;

- CRF $$R_N = \min\left((f_P - f_N)^2 + (f_{P'} - f_N)^2\right)$$
  (where *N* is the central point and f_P refers to the image intensity at the point *P*)

	Case A	Case B	Case C
Case Description	The nucleus is within the uniform area.	The nucleus is the edge point.	The nucleus is a corner point.
CRF	low	low	High
CRF Description	There are more than one line *l*, s.t. both *P* and *P′* belong to the USAN	There is exactly one line (tangential to the edge), s.t. both *P* and *P′* belong to the USAN.	There are no line *l*, s.t. at least one of points *P* and *P′* does not belong to the USAN.

Trajkovic and Hedley, 1998

- Discrete approximation

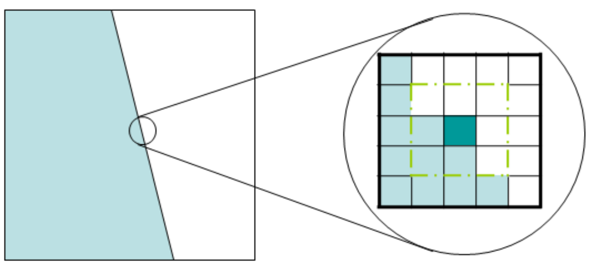


  - CRF becomes

$$R_N = \min_{P,P' \in S_n} \left((f_P - f_N)^2 + (f_{P'} - f_N)^2 \right)$$

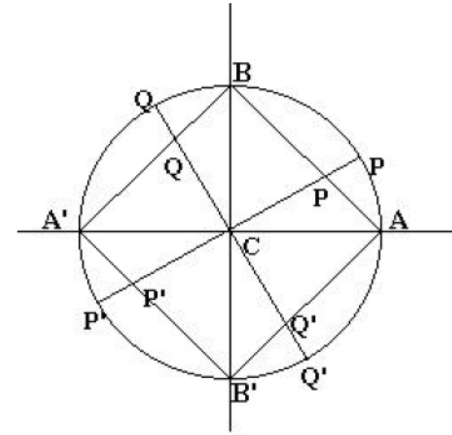
(where N is the nucleus and P and P' are opposite with respect to N)

- **<u>Wrong CRF by Discrete Approximation</u>**



  - the point is on the edge, but CRF may be high.

Trajkovic and Hedley, 1998



- If using <u>bigger window</u> ➔ worse localisation!

- <u>**Interpixel Approximation**</u>
  - Compute horizontal (r_A) and vertical (r_B) intensity variation

    $$r_A = (f_A - f_C)^2 + (f_{A'} - f_C)^2$$
    $$r_B = (f_B - f_C)^2 + (f_{B'} - f_C)^2$$
  - The CRF is computed $R = \min(r_A, r_B)$
  - If R is **less than a given threshold**, the nucleus **is not a corner** point and no further computation is necessary
  - Otherwise, <u>**the interpixel approximation**</u> is applied to check for <u>**diagonal edges**</u>.

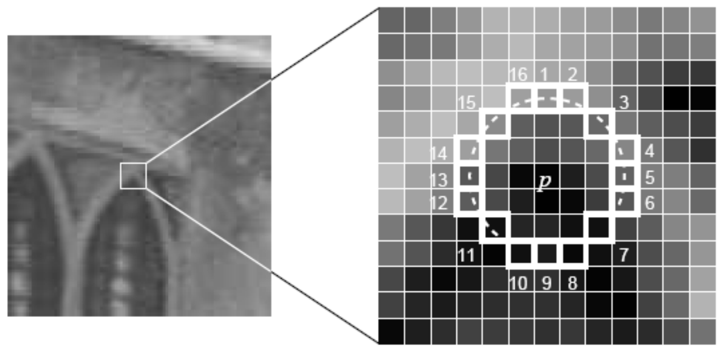
  - The CRF is computed $R = \min_{x \in (0,1)} (r_1(x), r_2(x))$
    where x is a parameter which determines position of the point on the square
  - The response function and intensity at interpixel location is calculated as below:

$$r_1(x) = (f_P - f_C)^2 + (f_{P'} - f_C)^2 \quad f_P = (1-x) \cdot f_A + x \cdot f_B, \; f_{P'} = (1-x) \cdot f_{A'} + x \cdot f_{B'}$$
$$r_2(x) = (f_Q - f_C)^2 + (f_{Q'} - f_C)^2 \quad f_Q = (1-x) \cdot f_{A'} + x \cdot f_B, \; f_{Q'} = (1-x) \cdot f_A + x \cdot f_{B'}$$

FAST – Rosten and Drummond, 2006

- Used "**Machine Learning**" approach

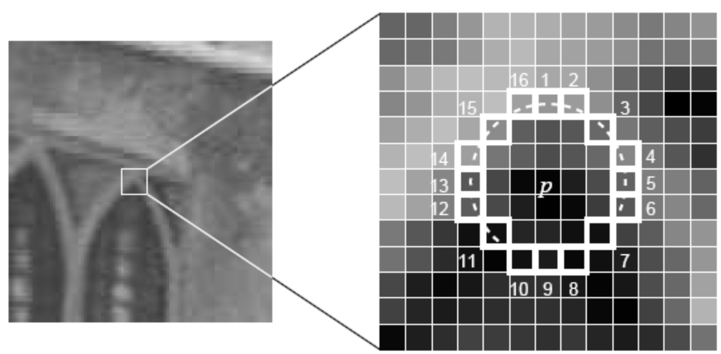
- **Segment-Test Algorithm**



  – If there exists a set of ***n* contiguous pixels** in the circle (a Bresenham circle) which are **all brighter** than the intensity of the candidate pixel I_p plus a threshold t, or **all darker** than $I_p - t$

FAST – Rosten and Drummond, 2006

- High speed Segment-Test Algorithm
  - Consider n = 12 and r = 3
  - The test **examines only the four pixels** at 1, 5, 9, and 13 (**the four compass directions**),
    - If p is a corner then at **least three of these** must all **be brighter or darker** than $I_p \pm t$
    - Otherwise, the point **cannot be corner**.
  - Problem
    - The high-speed test **does not generalise** well for **n<12**.
    - The choice and ordering of the fast test pixels contains **implicit assumptions about the distribution of feature appearance.**
    - Knowledge from the first 4 tests is discarded.
    - Multiple features are detected adjacent to one another.



FAST – Rosten and Drummond, 2006

- Machine learning a corner detector
  - 2 stage algorithm

- Stage 1) **detecting corners** and **making attribute map**
  - Corners are detected targeted at every pixels from a set of images
  - **Using the segment-test** criterion for *n* and a convenient threshold *t*
  - This uses **a slow algorithm** which for **each pixel simply tests all 16 locations** on the circle around it.
  - Preferably from the target application domain

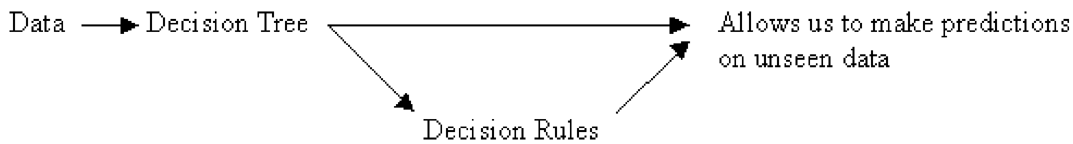
  - For each location on the circle $x \in \{1..16\}$, the pixel at that position relative to *p* (denoted by *p*→*x*) can have one of three states:

$$S_{p\to x} = \begin{cases} d, & I_{p\to x} \le I_p - t \quad \text{(darker)} \\ s, & I_p - t < I_{p\to x} < I_p + t \quad \text{(similar)} \\ b, & I_p + t \le I_{p\to x} \quad \text{(brighter)} \end{cases} \qquad (5)$$

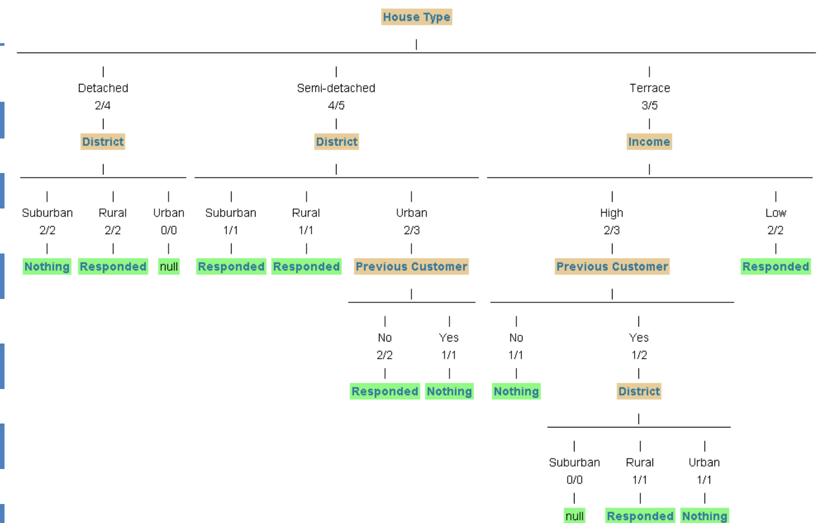
  - Choosing an x and computing $S_{p\to x}$ for all $p \in P$ (the set of all pixels in all training images) partitions *P* into three subsets, P_d, P_s, P_b, where each *p* is assigned to $P_{S_{p\to x}}$.

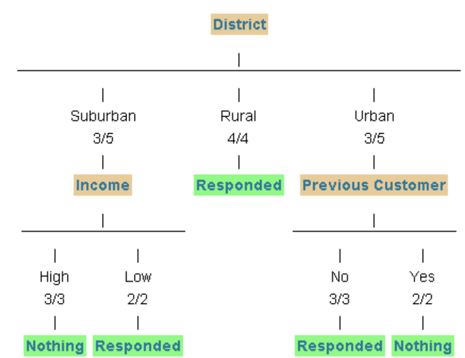
FAST – Rosten and Drummond, 2006

- ID3 (Iterative Dichotomiser 3) Decision Tree
  - Invented by Ross Quinlan
  - A decision tree is constructed by looking for regularities in data.



District	House Type	Income	Previous Customer	Outcome
Suburban	Detached	High	No	Nothing
Suburban	Detached	High	Yes	Nothing
Rural	Detached	High	No	Responded
Urban	Semi-detached	High	No	Responded
Urban	Semi-detached	Low	No	Responded
Urban	Semi-detached	Low	Yes	Nothing
Rural	Semi-detached	Low	Yes	Responded
Suburban	Terrace	High	No	Nothing
Suburban	Semi-detached	Low	No	Responded
Urban	Terrace	Low	No	Responded
Suburban	Terrace	Low	Yes	Responded
Rural	Terrace	High	Yes	Responded
Rural	Detached	Low	No	Responded
Urban	Terrace	High	Yes	Nothing





FAST – Rosten and Drummond, 2006

- **ID3**
  - Examine the attributes to add at the next level of the tree **using an entropy calculation**.
  - Choose the attribute that **minimizes the entropy**.
- **Entropy**
  - The entropy of a dataset can be considered to be **how disordered it is**.
  - Entropy is related to information, in the sense that **the higher the entropy**, or uncertainty, of some data, then the **more information is required in order to completely describe that data**.

$$Entropy(s) = -\sum_{i=1}^{C} p_i \log_2 p_i$$

  - P_i is the proportion of instances in the dataset that take the $_ith$ value of the target attribute
- **Information Gain**
  - Calculates **the reduction in entropy** (*Gain in information*) that would result on splitting the data on an attribute

$$Gain\ (S, A) = Entropy\ \ (S) - \sum_{v \in A} \frac{|S_v|}{|S|} Entropy\ \ (S_v)$$

- **Input:** A data set, S
  Output: A decision tree
- If all the instances have the same value for the target attribute then return a decision tree that is simply this value (not really a tree - more of a stump).
- Else
  1. **Compute Gain** values **for all attributes** and **select** an attribute with the **highest value** and create a node for that attribute.
  2. **Make a branch from this node** for every value of the attribute
  3. **Assign** all possible **values of the attribute** to branches.
  4. Follow each branch by partitioning the dataset to be only instances whereby the value of the branch is present and then go back to 1.

FAST – Rosten and Drummond, 2006

- State 2) **making ID3 decision tree**
  - Let K_p be a boolean variable which is true if p is a corner and false otherwise.
  - The entropy of K for the set P is:

$$H(P) = (c + \bar{c})\log_2(c + \bar{c}) - c\log_2 c - \bar{c}\log_2 \bar{c}$$

$$\text{where} \quad c = \left|\{p | K_p \text{ is true}\}\right| \quad \text{(number of corners)}$$

$$\text{and} \quad \bar{c} = \left|\{p | K_p \text{ is false}\}\right| \quad \text{(number of non corners)}$$

  - The information gain:

$$H(P) - H(P_d) - H(P_s) - H(P_b)$$

  - Practical implementation ➔ series of if statements

```
if(*cache_1 > cb)
  if(*cache_2 > cb)
    if(*(cache_0+3) > cb)
      if(*(cache_0 + pixel[11]) > cb)
        if(*(cache_0 + pixel[14]) > cb)
          if(*(cache_0 + pixel[6]) > cb)
            if(*(cache_0 + pixel[13]) > cb)
              if(*(cache_2+2) > cb)
                if(*(cache_0 + pixel[10]) > cb)
                  if(*(cache_1+1) > cb)
                    if(*(cache_1+1) > cb)
                      if(*(cache_2+1) > cb)
                        goto success;
                      else if(*(cache_2+1) < c_b)
                        continue;
                      else
                        if(*(cache_0+3) > cb)
                          goto success;
                        else
                          continue;
                    else if(*(cache_1+1) < c_b)
                      continue;
                    else
                      if(*(cache_0+3) > cb)
                        if(*(cache_0 + pixel[2]) > cb)
                          if(*(cache_2+1) > cb)
                            if(*(cache_0 + pixel[3]) > cb)
                              goto success;
                            else
                              continue;
                          else
                            continue;
                        else
```

FAST – Rosten and Drummond, 2006

- Non-maximal suppression
  - Since **the segment test does not compute a corner response function**
  - **Non maximal suppression** can not be applied directly
  - **A score function**, ***V*** must be computed for each detected corner
    - The sum of the absolute difference between the pixels in the contiguous arc and the center pixel.

$$V = \max\left(\sum_{x \in S_{\text{bright}}} |I_{p \to x} - I_p| - t \ , \ \sum_{x \in S_{\text{dark}}} |I_p - I_{p \to x}| - t \right)$$

References

1. Harris, C. and Stephens, M., "A combined corner and edge detector", In Alvey Vision Conference, pp. 147-151, 1988
   - http://www.csse.uwa.edu.au/~pk/research/matlabfns/Spatial/Docs/Harris/
   - Jung, H.G., "4-3. Corner", in Computer Vision Lecture Notes, EEE6503-01, Yonsei Univ.
2. Shi, J. and Toamasi, C., "Good features to track", In 9th IEEE Conference on Computer Vision and Pattern Recognition, Springer, 1994
   - Suhr, J.K., "8-1. KLT Feature Tracking", in Computer Vision Lecture Notes, EEE6503-01, Yonsei Univ.
3. Lowe, D.G., "Distinctive image features from scale-invariant keypoints", International Journal of Computer Vision,
4. Smith, S.M. and Brady, J.M., "SUSAN – a new approach to low level image processing", International Journal of Computer Vision, vol. 23, pp. 45-78, 1997
   - http://users.fmrib.ox.ac.uk/~steve/susan/
5. Trajkovic, M. and Hedley, M., "Fast corner detection", Image and Vision Computing, vol. 16, pp. 45-78, 1998
6. Rosten, E. and Drummond, T., "Machine learning for high-speed corner detection", European Conference on Computer Vision, 2006. ECCV 2006. pp. 430-443, May, 2006.
   - http://svr-www.eng.cam.ac.uk/~er258/work/fast
7. Quinlan, J.R., "Induction of induction of decision trees", Mach, Learn. 1. 1(Mar. 1986), pp. 81-106.
   - "Decision Trees tutorial > ID3 & Entropy", Decision Trees & Data Mining, http://www.decisiontrees.net/node/27