Object Recognition Using Scale Invariant Keypoints

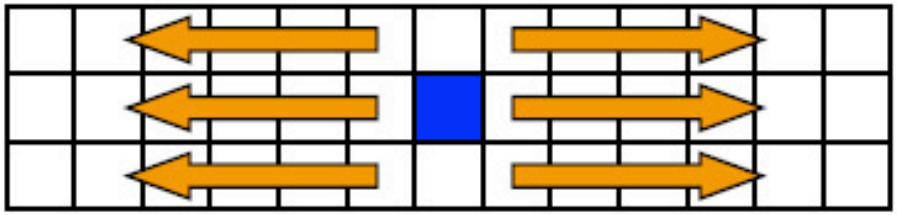
"Distinctive Image Features from Scale-Invariant Keypoints"
David Lowe. *Intl. Journal. of Computer Vision* 2006

(Available from the on-line journals section of the SFU library)

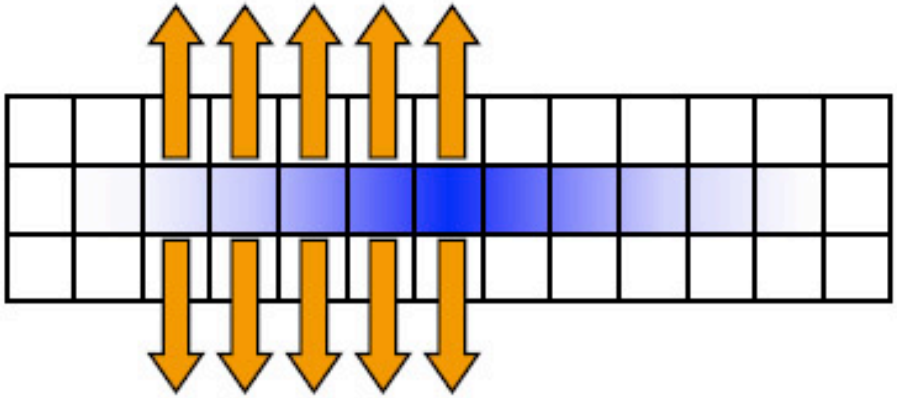
Figure 12: The training images for two objects are shown on the left. These can be recognized in a cluttered image with extensive occlusion, shown in the middle. The results of recognition are shown on the right. A parallelogram is drawn around each recognized object showing the boundaries of the original training image under the affine transformation solved for during recognition. Smaller squares indicate the keypoints that were used for recognition.

"Distinctive Image Features from Scale-Invariant Keypoints"
David Lowe. *Intl. Journal. of Computer Vision* 2006

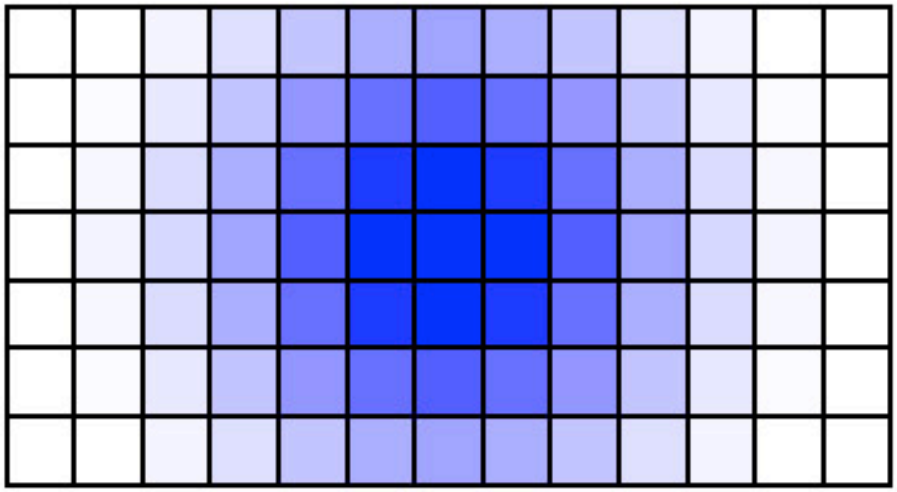
Separable Gaussian Convolution



Blur the sour
horizontally



Blur the blur
verticaly

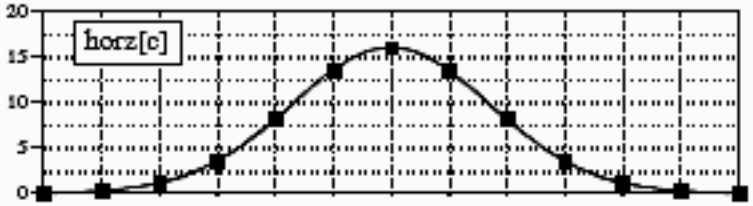


Result

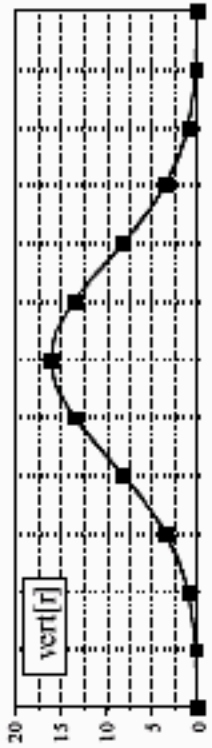
http://www.gamerendering.com/2008/10/11/gaussian-blur-filter-shader/

Separable Gaussian Convolution

FIGURE 24-7
Separation of the Gaussian. The Gaussian is the only PSF that is circularly symmetric *and* separable. This makes it a common filter kernel in image processing.



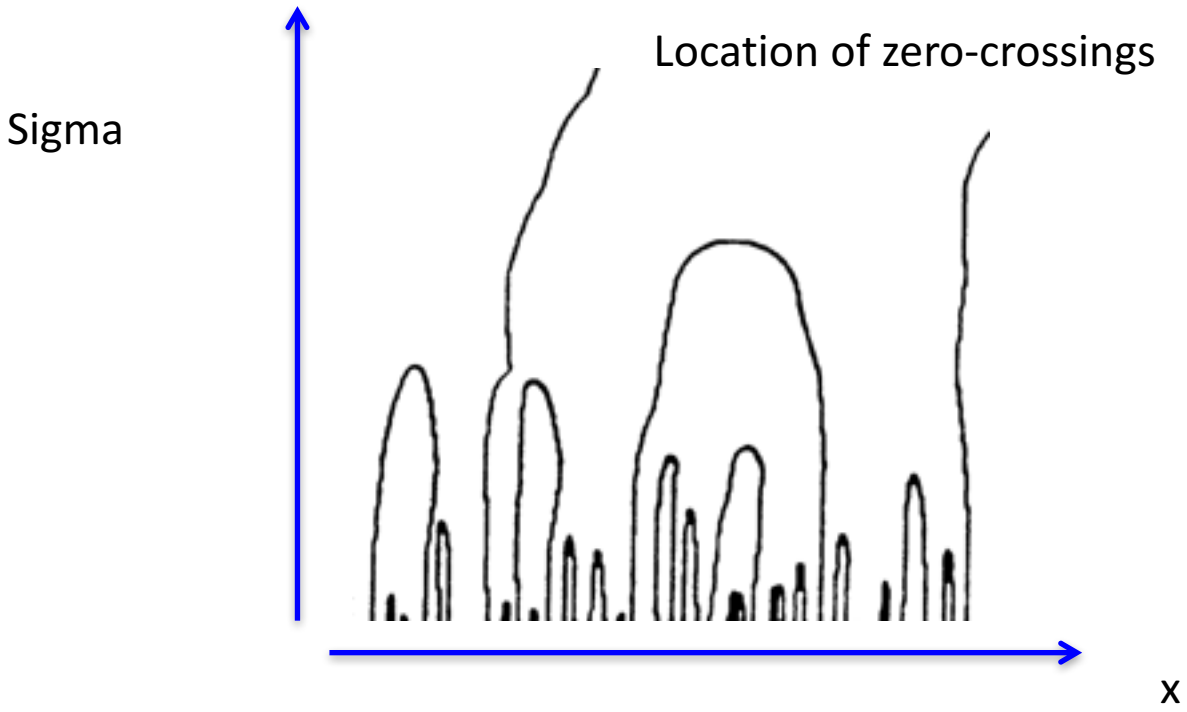
0.04	0.25	1.11	3.56	8.20	13.5	16.0	13.5	8.20	3.56	1.11	0.25	0.04



0.04	0	0	0	0	0	1	1	1	0	0	0	0	0
0.25	0	0	0	1	2	3	4	3	2	1	0	0	0
1.11	0	0	1	4	9	15	18	15	9	4	1	0	0
3.56	0	1	4	13	29	48	57	48	29	13	4	1	0
8.20	0	2	9	29	67	111	131	111	67	29	9	2	0
13.5	1	3	15	48	111	183	216	183	111	48	15	3	1
16.0	1	4	18	57	131	216	255	216	131	57	18	4	1
13.5	1	3	15	48	111	183	216	183	111	48	15	3	1
8.20	0	2	9	29	67	111	131	111	67	29	9	2	0
3.56	0	1	4	13	29	48	57	48	29	13	4	1	0
1.11	0	0	1	4	9	15	18	15	9	4	1	0	0
0.25	0	0	0	1	2	3	4	3	2	1	0	0	0
0.04	0	0	0	0	0	1	1	1	0	0	0	0	0

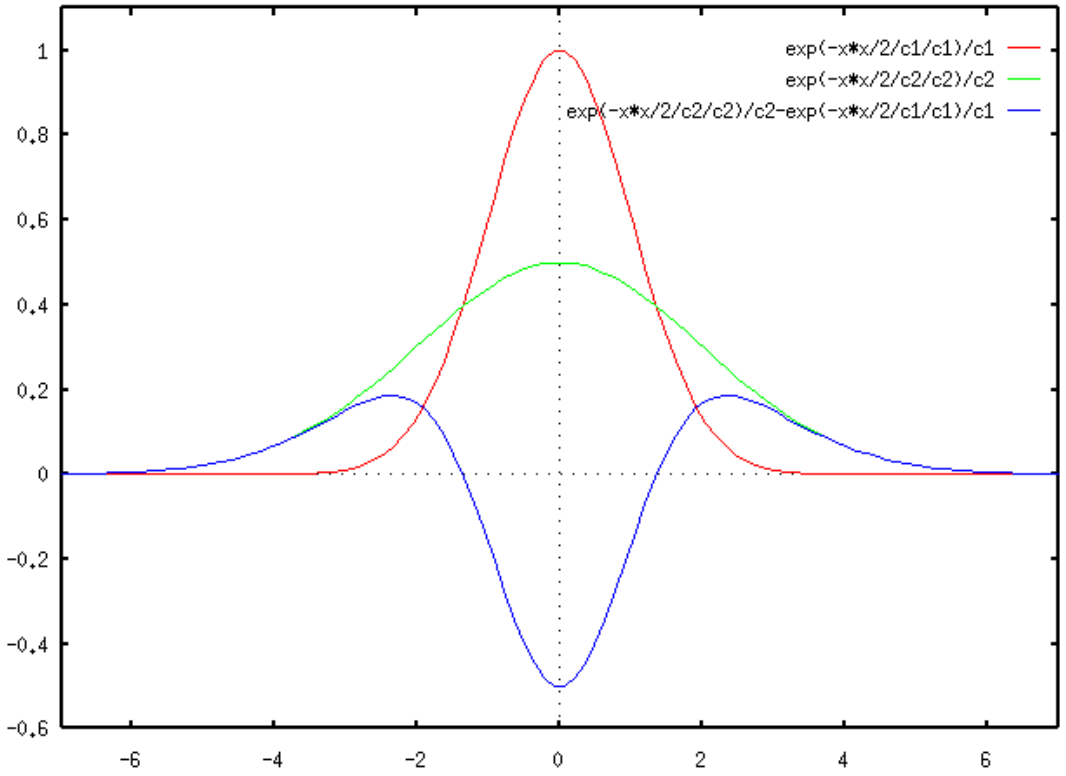
http://www.dspguide.com/graphics/F_24_7.gif

Scale Space



Andrew Witkin CMU

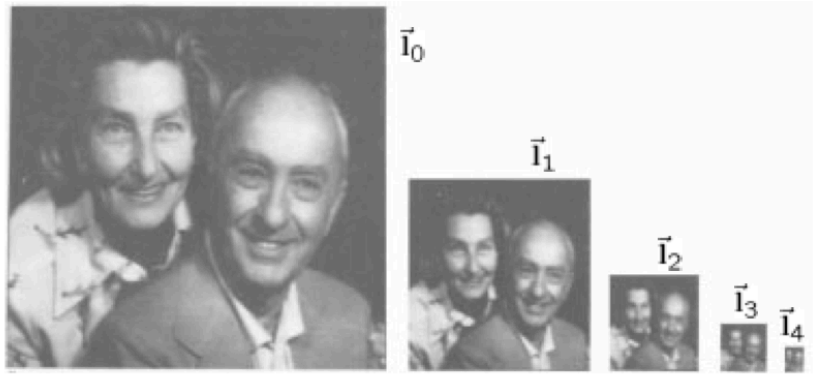
Laplacian can be approximated as the difference of two Gaussians



More on this a few slides from now

Gaussian Pyramid (cont)

Example of image and next four pyramid levels:



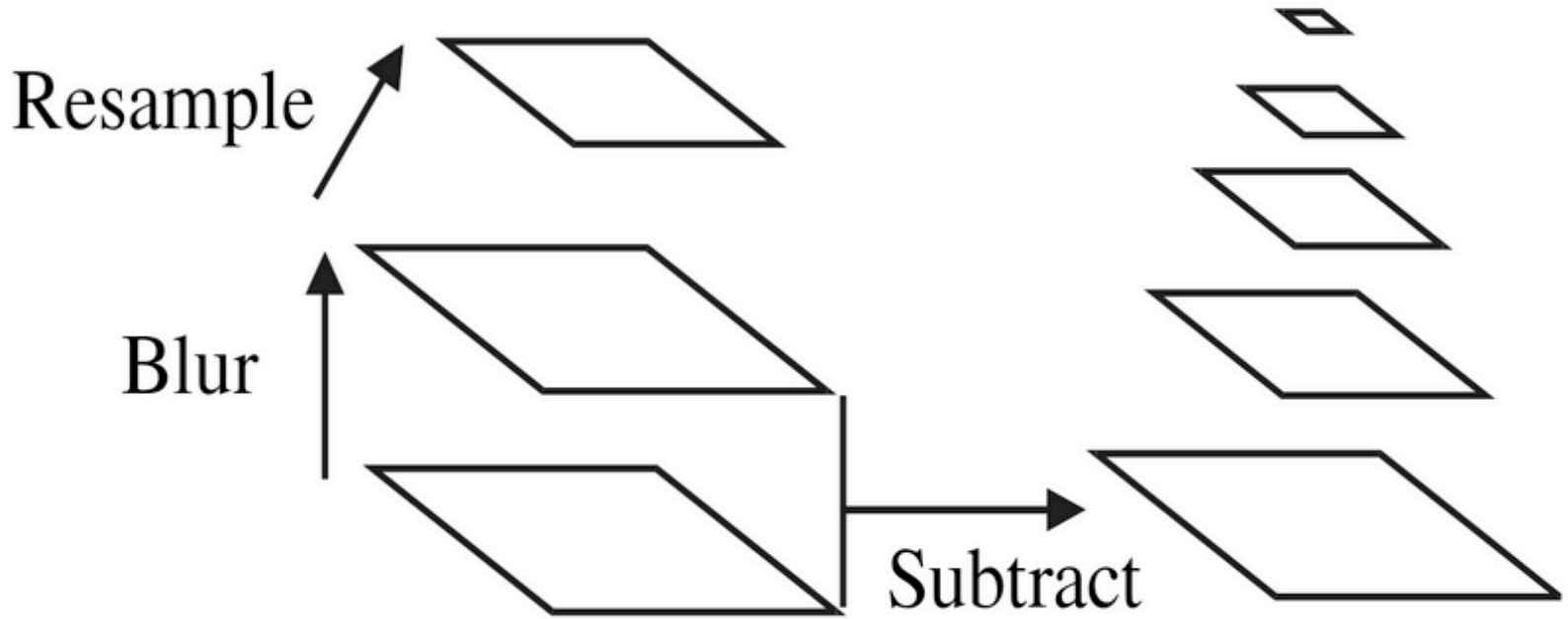
First three levels scaled to be the same size:

Properties of Gaussian pyramid:

- used for multi-scale edge estimation,
- efficient to compute coarse scale images. Only 5-tap 1D filter kernels are used,
- highly redundant, coarse scales provide much of the information in the finer scales.

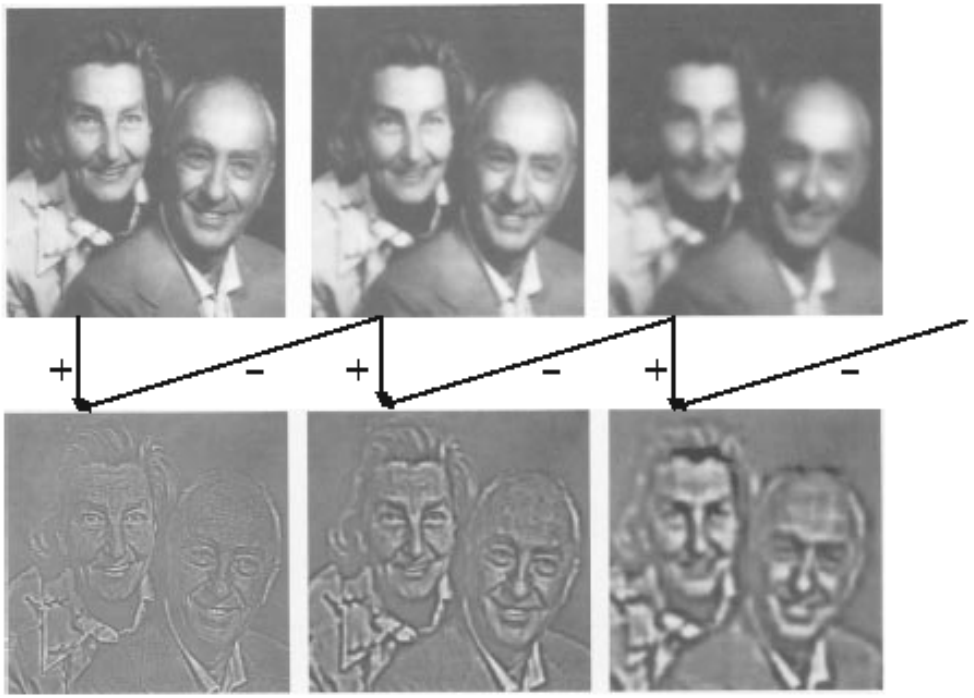
Fleet & Jepson U of T

- All scales must be examined to identify scale-invariant features
- DOG pyramid (Burt & Adelson, 1983)

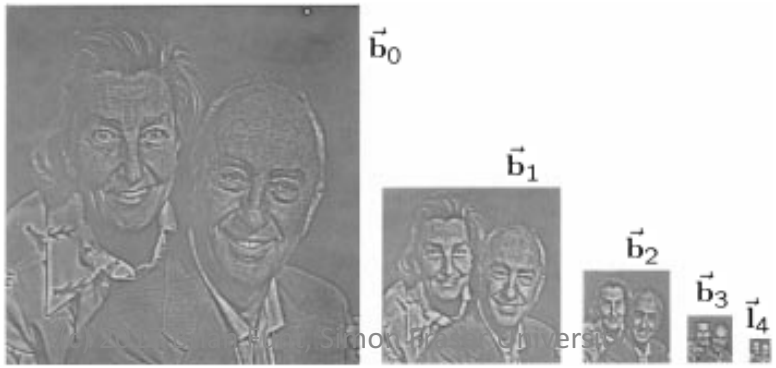


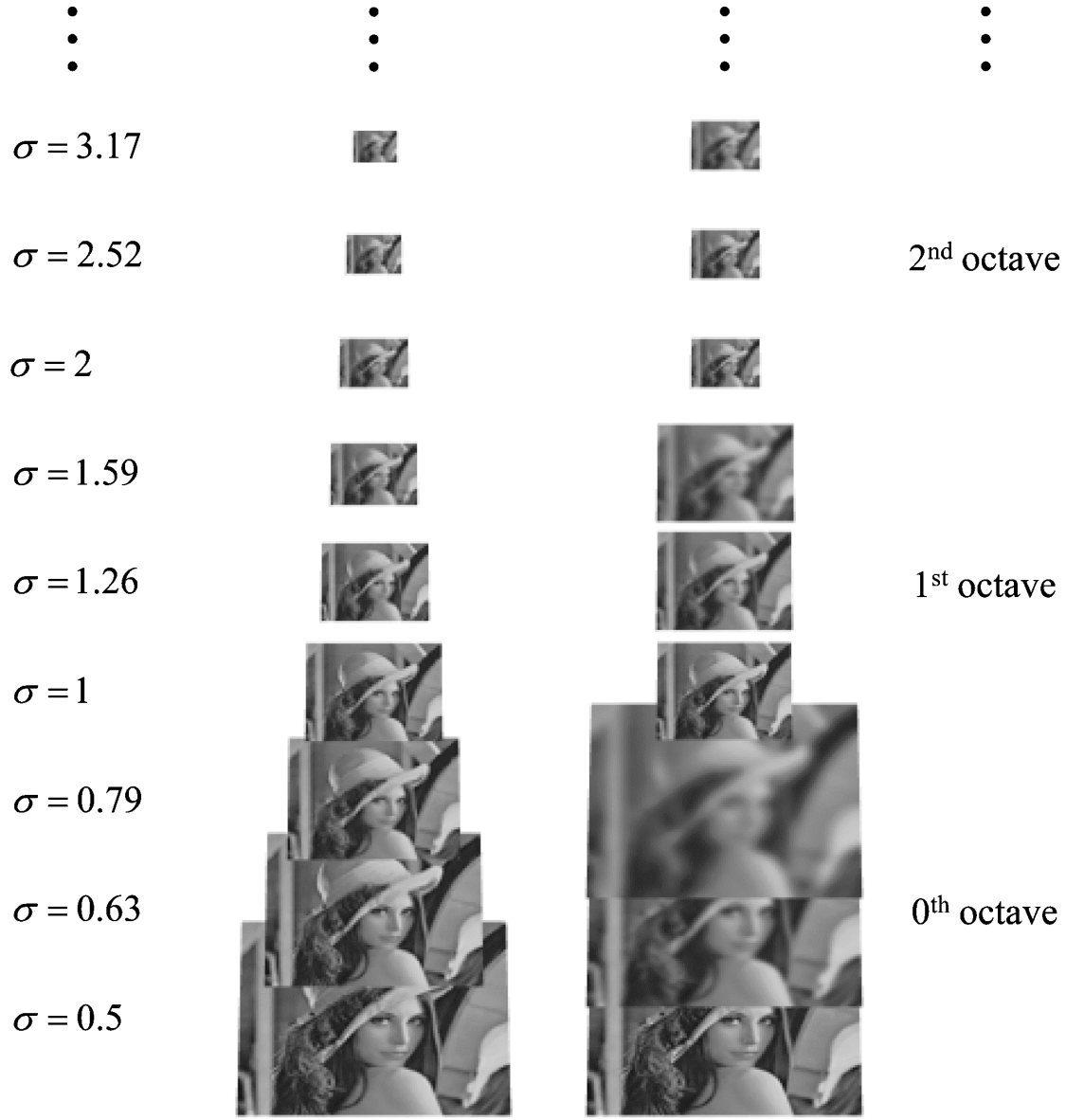
Laplacian Pyramid (cont)

Construction of the Laplacian bands:



A Laplacian pyramid with four levels:





Space Requirements

1+ 1/4 + 1/16 + 1/32 + 1/64 +…

For 4 levels:

1+ 1/4 + 1/16 + 1/32 = 1.34375

On the name "Laplacian"

The well-known Laplacian derivative operator (isotropic second derivative) is given by

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

For Gaussian kernels, $g(x;\sigma) = \frac{1}{\sqrt{2\pi}\sigma}\, e^{-x^2/2\sigma^2}$,

$$\begin{aligned}
\frac{dg(x;\sigma)}{dx} &= \frac{-x}{\sigma^2} g(x;\sigma) \\
\frac{d^2 g(x;\sigma)}{dx^2} &= \left(\frac{x^2}{\sigma^2} - 1\right) \frac{1}{\sigma^2} g(x;\sigma) \\
\frac{dg(x;\sigma)}{d\sigma} &= \left(\frac{x^2}{\sigma^2} - 1\right) \frac{1}{\sigma} g(x;\sigma)
\end{aligned}$$

Therefore

$$\frac{d^2 g(x;\sigma)}{dx^2} \;=\; c_0(\sigma) \frac{d\, g(x;\sigma)}{d\sigma} \;\approx\; c_1(\sigma)\,(g(x;\sigma) - g(x;\sigma + \Delta\sigma))$$

That is, if the low-pass filter h used to create the Laplacian pyramid is Gaussian, then the Laplacian pyramid levels approximate the second derivative of the image at scale σ.

Fleet & Jepson U of T

On the name "Laplacian"

The well-known Laplacian derivative operator (isotropic second derivative) is given by

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

For Gaussian kernels, $g(x;\sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-x^2/2\sigma^2}$,

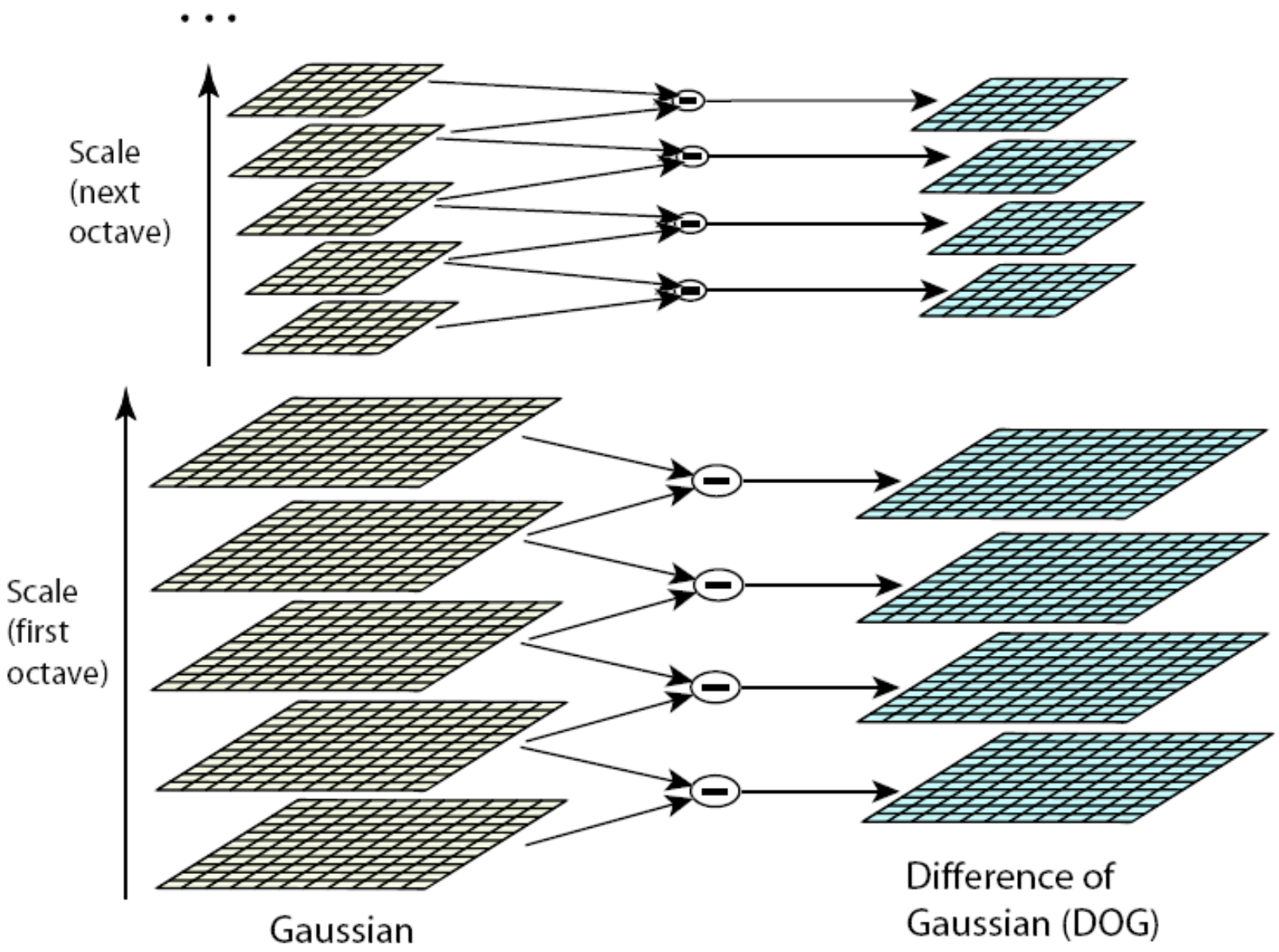
$$\begin{aligned}
\frac{dg(x;\sigma)}{dx} &= \frac{-x}{\sigma^2} g(x;\sigma) \\
\frac{d^2 g(x;\sigma)}{dx^2} &= \left(\frac{x^2}{\sigma^2} - 1\right) \frac{1}{\sigma^2} g(x;\sigma) \\
\frac{dg(x;\sigma)}{d\sigma} &= \left(\frac{x^2}{\sigma^2} - 1\right) \frac{1}{\sigma} g(x;\sigma)
\end{aligned}$$

Therefore

$$\frac{d^2 g(x;\sigma)}{dx^2} = c_0(\sigma) \frac{d\, g(x;\sigma)}{d\sigma} \approx c_1(\sigma) \left(g(x;\sigma) - g(x;\sigma + \Delta\sigma)\right)$$

That is, if the low-pass filter h used to create the Laplacian pyramid is Gaussian, then the Laplacian pyramid levels approximate the second derivative of the image at scale σ.

Fleet & Jepson U of T



David Lowe, IJCV paper

Keypoint Detection

Looking for points of high curvature

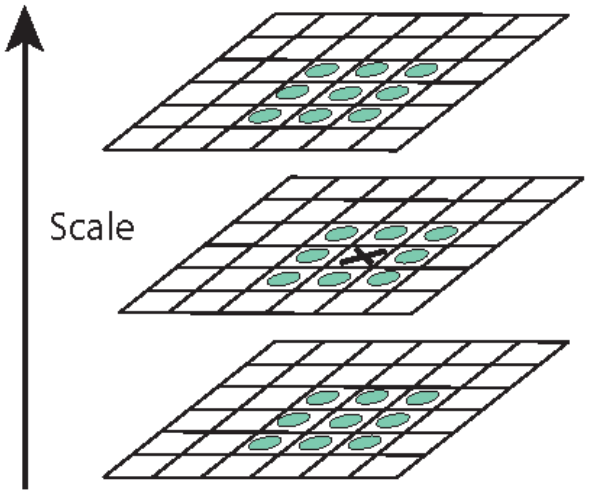


Figure 2: Maxima and minima of the difference-of-Gaussian images are detected by comparing a pixel (marked with X) to its 26 neighbors in 3x3 regions at the current and adjacent scales (marked with circles).

"Distinctive Image Features from Scale-Invariant Keypoints"
David Lowe. *Intl. Journal. of Computer Vision* 2006

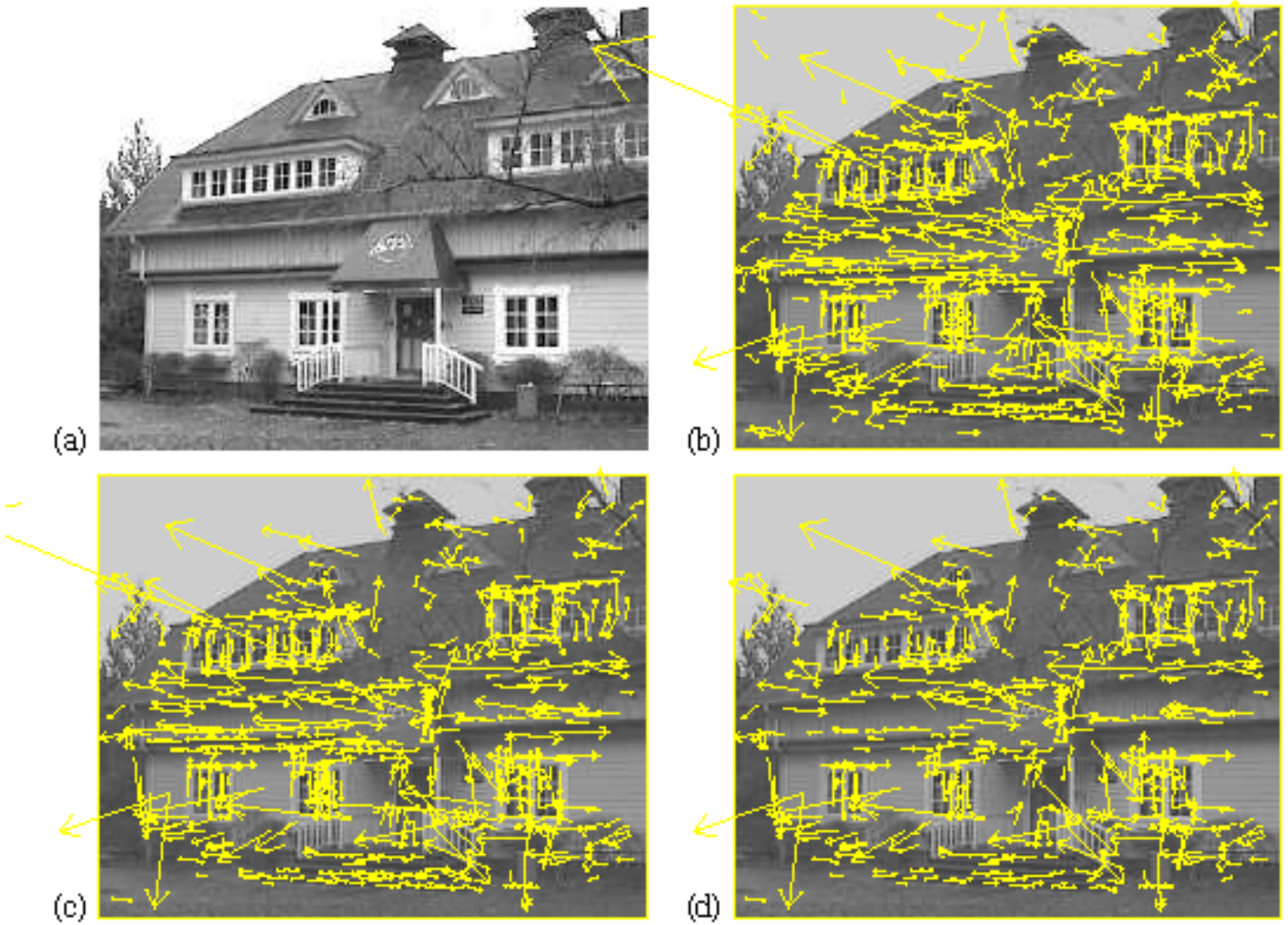


Figure 5: This figure shows the stages of keypoint selection. (a) The 233x189 pixel original image. (b) The initial 832 keypoints locations at maxima and minima of the difference-of-Gaussian function. Keypoints are displayed as vectors indicating scale, orientation, and location. (c) After applying a threshold on minimum contrast, 729 keypoints remain. (d) The final 536 keypoints that remain following an additional threshold on ratio of principal curvatures.

David Lowe, IJCV

Keypoint Descriptor

Description is different from location

Keypoint is described relative to its local coordinate system

Vector of 128 features is used as the descriptor

What are these features?

Object is recognized by comparing the feature vectors

Keypoint Coordinate System

Sample points in region around the keypoint

Histogram their Magnitude and Orientation of Gradient

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

Every 10 degrees => 36 bins

Weight each contribution by the gradient magnitude

Use dominant bin as the direction of the gradient

Result is a local coordinate system for the keypoint

Building the Keypoint Descriptor

David Lowe, IJCV

These gradients are relative to the keypoint coordinate system

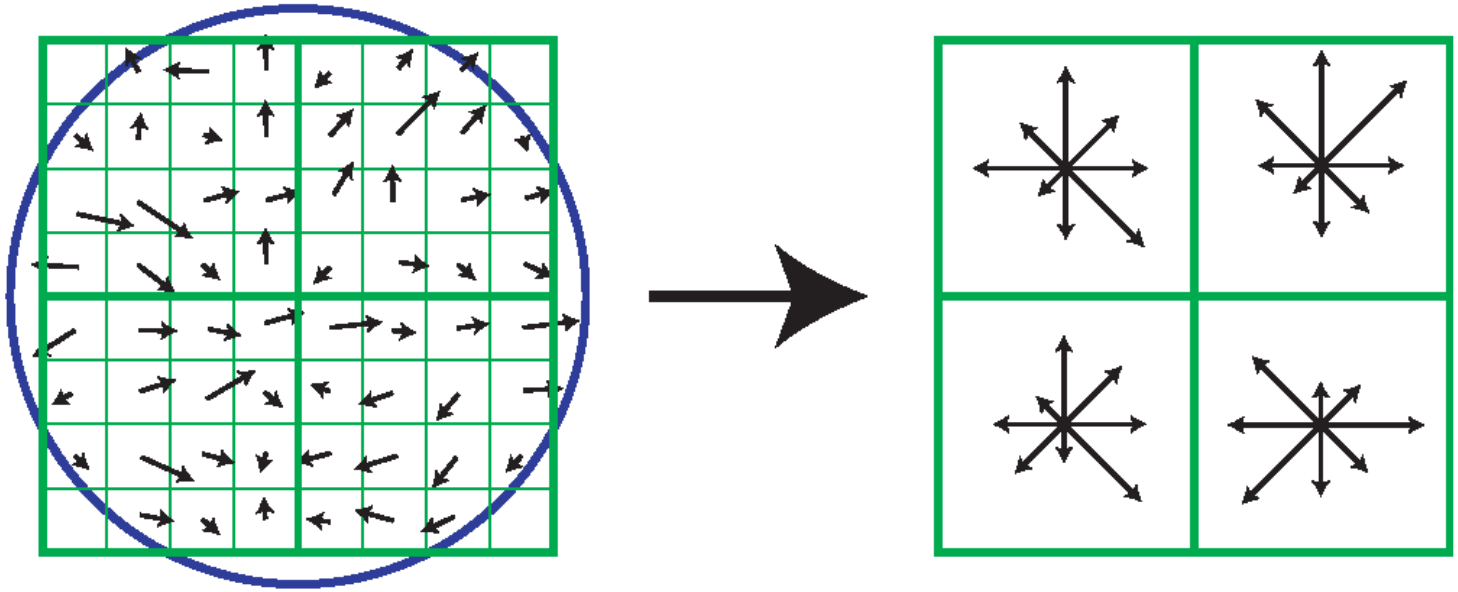
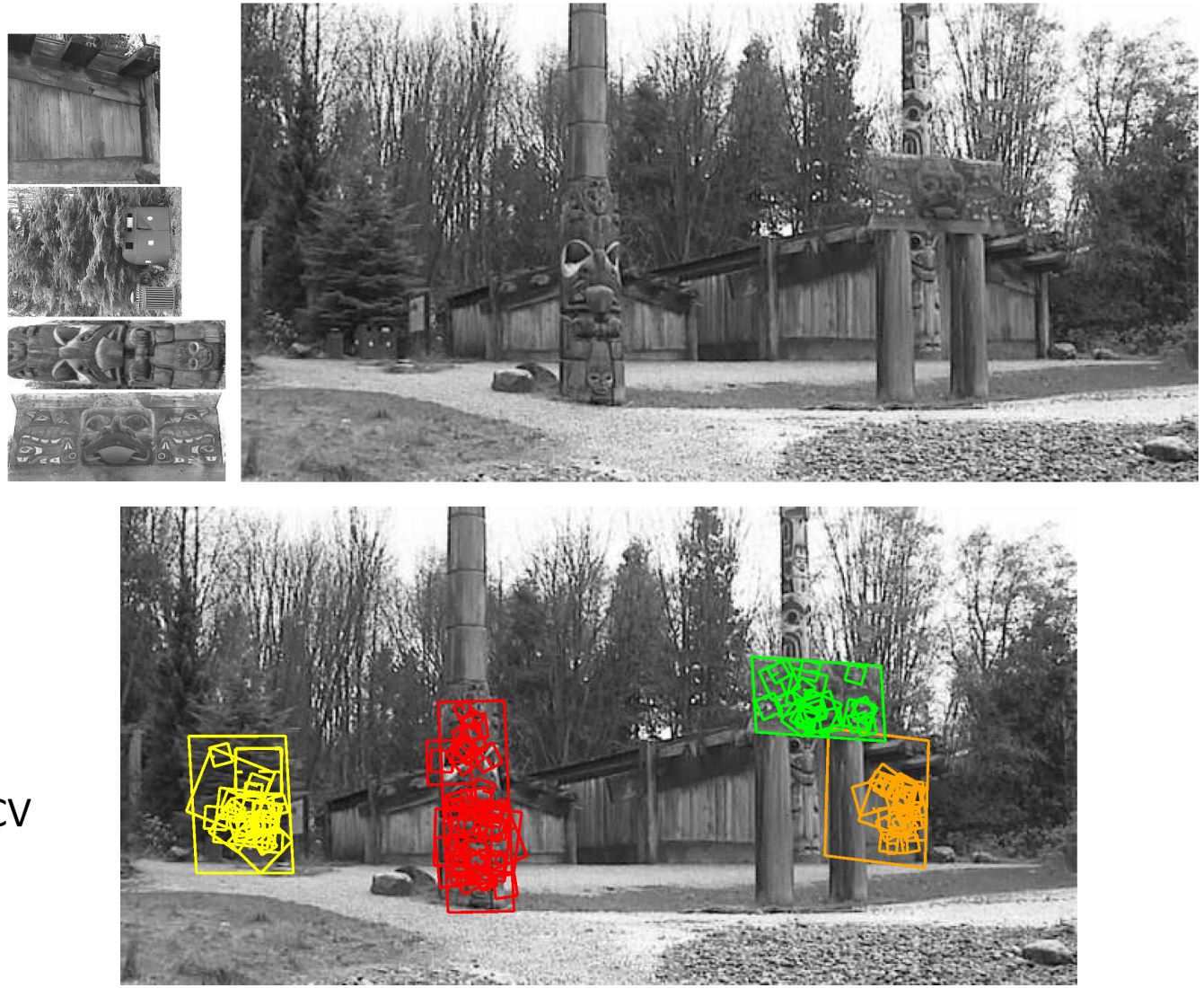


Figure 7: A keypoint descriptor is created by first computing the gradient magnitude and orientation at each image sample point in a region around the keypoint location, as shown on the left. These are weighted by a Gaussian window, indicated by the overlaid circle. These samples are then accumulated into orientation histograms summarizing the contents over 4x4 subregions, as shown on the right, with the length of each arrow corresponding to the sum of the gradient magnitudes near that direction within the region. This figure shows a 2x2 descriptor array computed from an 8x8 set of samples, whereas the experiments in this paper use 4x4 descriptors computed from a 16x16 sample array.

4x4 samples, 8 directions => 128-element descriptor

Figure 12: The training images for two objects are shown on the left. These can be recognized in a cluttered image with extensive occlusion, shown in the middle. The results of recognition are shown on the right. A parallelogram is drawn around each recognized object showing the boundaries of the original training image under the affine transformation solved for during recognition. Smaller squares indicate the keypoints that were used for recognition.

"Distinctive Image Features from Scale-Invariant Keypoints"
David Lowe. *Intl. Journal. of Computer Vision* 2006



David Lowe, IJCV

Figure 13: This example shows location recognition within a complex scene. The training images for locations are shown at the upper left and the 640x315 pixel test image taken from a different viewpoint is on the upper right. The recognized regions are shown on the lower image, with keypoints shown as squares and an outer parallelogram showing the boundaries of the training images under the affine transform used for recognition.

Database of Objects

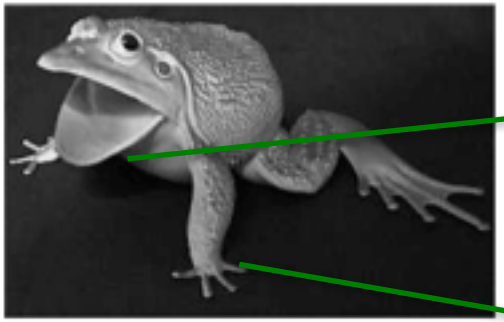
Descriptor for Each Keypoint

(g1, g2, …., g128)

(g1, g2, …., g128)

…

(g1, g2, …., g128)



(g1, g2, …., g128)

…

…

Locating objects in this input image

Image's Keypoint Descriptors

(g1, g2, …., g128)

(g1, g2, …., g128)

...

(g1, g2, …., g128)

Search database of known objects for matching keypoint descriptors

(a) Exhaustive search
(b) ANN (approximate nearest neighbour) methods

Lots of matches tell us the object is there in the image and its rough location

Object Pose

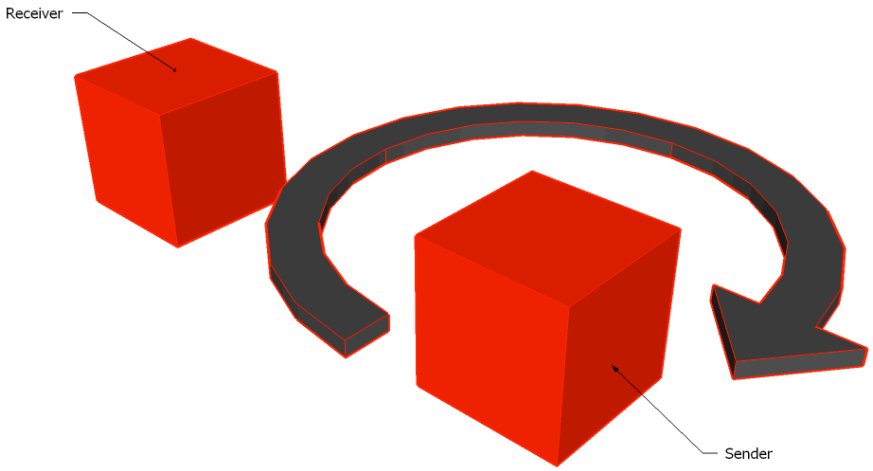
Pose: “The combination of position and orientation is referred to as the pose of an object, even though this concept is sometimes used only to describe the orientation.”

That quote is from Wikipedia entry on pose
http://en.wikipedia.org/wiki/Pose_%28computer_vision%29

It’s a good page to look at on pose and pose estimation

Confirmation of Pose for Keypoints

Small Rotations in 3D approximated by affine transformation in 2D



Affine includes 2D scaling plus translation

Image from http://www.percussa.com/manuals/getting-started/

2D Affine Transformation

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} m_1 & m_2 \\ m_3 & m_4 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

Output Point

Scaling
Rotation
Shear

Input Point

Translation Component

Solving for the transformation's parameters m_i

Image's Keypoint locations

$$\begin{bmatrix} x & y & 0 & 0 & 1 & 0 \\ 0 & 0 & x & y & 0 & 1 \\ & & \cdots & & & \\ & & \cdots & & & \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ t_x \\ t_y \end{bmatrix} = \begin{bmatrix} u \\ v \\ \vdots \end{bmatrix}$$

Model's keypoint locations

Solve for the best-fitting values

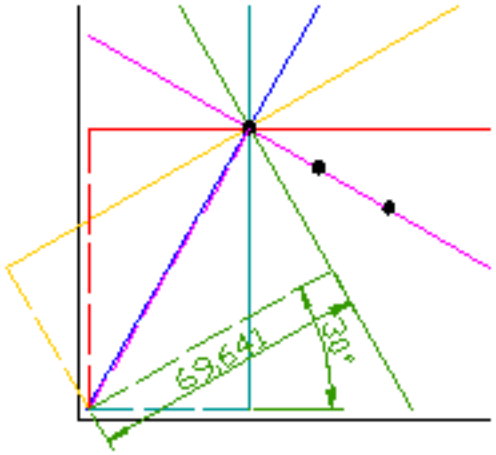
$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} m_1 & m_2 \\ m_3 & m_4 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

Clustering Matching Features using Hough Transform

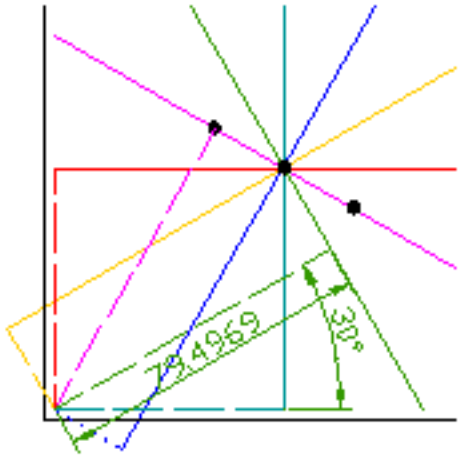
- Want to identify clusters of image features
  - Identifying single model
  - And its pose

- Hough-like clustering
  - Consider original Hough transform for hypothesizing lines…

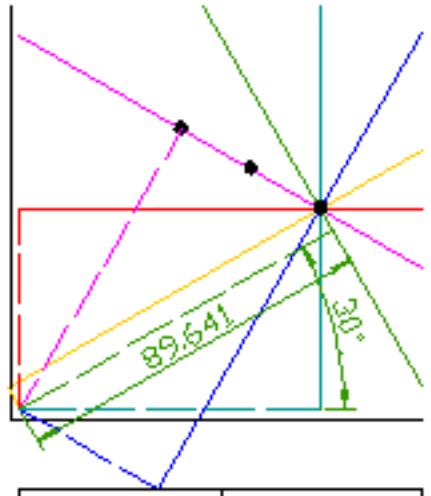
Hough Transform



Angle	Dist.
0	40
30	69.6
60	81.2
90	70
120	40.6
150	0.4



Angle	Dist.
0	57.1
30	79.5
60	80.5
90	60
120	23.4
150	−19.5

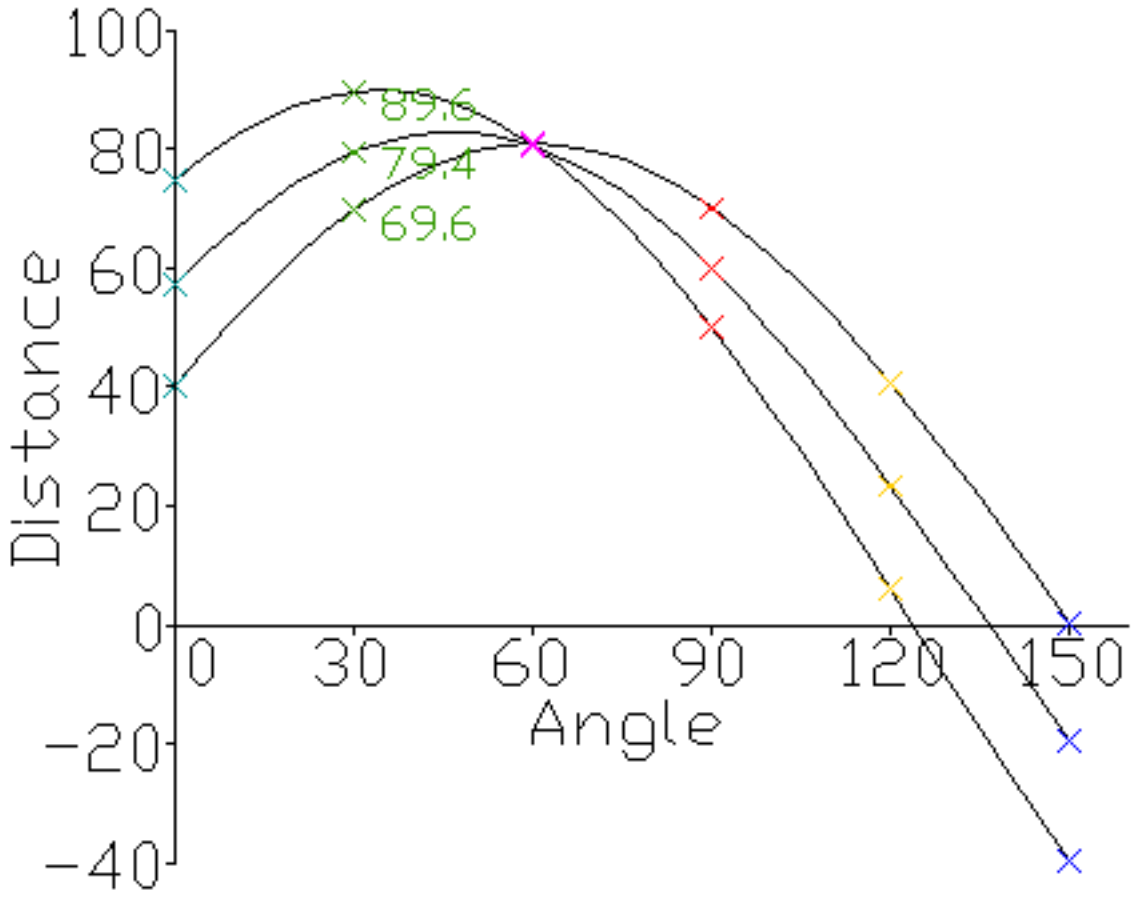


Angle	Dist.
0	74.6
30	89.6
60	80.6
90	50
120	6.0
150	−39.6

Retrieved from: http://en.wikipedia.org/wiki/Hough_transform

Example is for finding straight lines, but it generalizes to other things

Hough Transform Voting



Retrieved from: http://en.wikipedia.org/wiki/Hough_transform

Clustering Matching Features using Hough Transform

- Hough-like clustering
  - Each feature vector votes for all pose-model pairs
  - Limited to models with matching keypoint descriptors
  - High count indicates a model at the given pose.

- Once model and pose are established
  - Confirm using affine transform

SIFT Object Recognition Summary

- Keypoint identification
  - Based on high curvature
  - Scale space
    - Laplacian of Gaussian
    - Implementation by Difference of Gaussian
    - Laplacian pyramid using different scales
- Keypoint descriptor
  - Use dominant local gradient as coordinates
  - 16 histogram of gradients in neighbouring regions
  - Total of 128-elements in descriptor

Summary Continued

- Match image's keypoint descriptors to database
  - Approximate nearest neighbour method
- Cluster descriptors by pose-model pair
  - Use Hough transform voting
- Confirm model
  - Find best 2D affine transformation
    - i.e., 2D scaling, rotation, translation
    - Mapping model's keypoints to image's
    - Good fit implies right model in right place

Summary Continued

- Other applications
- Navigation
  - Find where you are by recognizing landmarks
- Image stitching
  - Building Panoramas
  - Creating 3D models from multiple views
    - http://apps.123dapp.com/catch/
    - Also free app on iTunes called 123D catch