CAP5415
Computer Vision

Yogesh S Rawat

yogesh@ucf.edu

HEC-241

Questions?

Features

Lecture 9

Outline

- Features introduction - basics
- Key-points
- Histogram of Oriented gradients (HOG)
- Scale-Invariant Feature Transform (SIFT)

Features

Lecture 9

Basics

What is a Feature?

- Information extracted from an image/video.
  - Hand-crafted
  - Learned
- We can define a function
  - Takes an image/video as an input
  - Produces one or more numbers as output
- Hand-crafted features
  - Feature engineering
- Learned features
  - Automatically learned

Types of Features

- Global features
  - Extracted from the entire image
  - Examples: template (the image itself), HOG, etc.
- Region-based features
  - Extracted from a smaller window of the image.
  - Applying global method to a specific image region.
- Local features
  - Describe a pixel, and the vicinity around a specific pixel.
  - Local feature always refer to a specific pixel location.

Uses of Features

- Features can be used for many computer vision problems.
  - Detection.
  - Recognition.
  - Tracking.
  - Stereo estimation.
- Different types of features for different problems,
  - Different assumptions about the images.
  - That is why there are many different types of features.

Uses of Features: Matching

Uses of Features: Matching

Credit: Fei Fei Li

Uses of Features: structure from motion

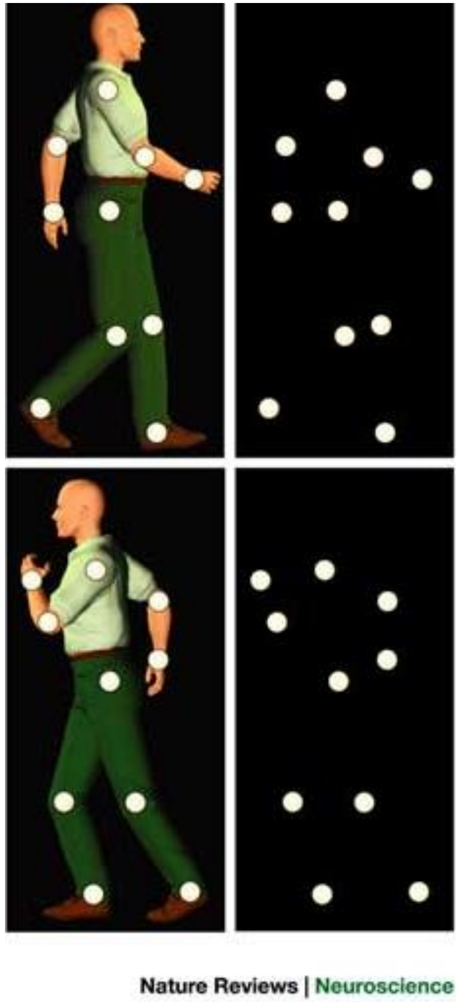
Uses of Features: structure from motion

Uses of Features: panorama stitching

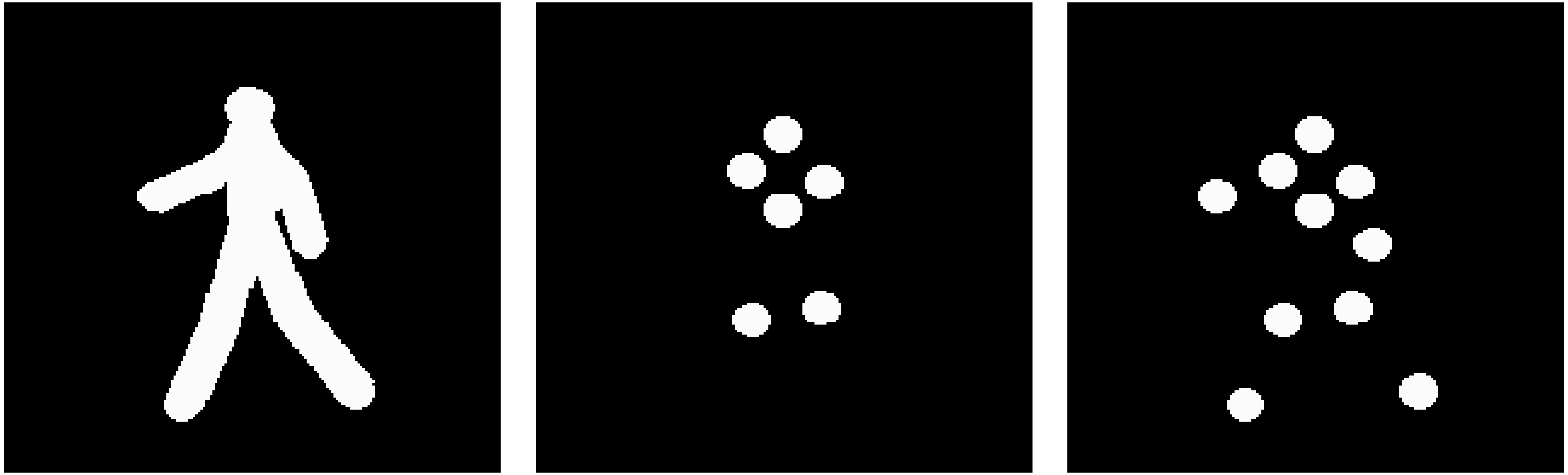
- Given two images
- How do we overlay them?


Finding Features in Videos

- Complex actions can be recognized on the basis of 'point-light displays',
  - Facial expressions,
  - Sign Language,
  - Arm movements,
  - Various full-body actions.



Finding Features in Videos



Characteristics of good features

- **Distinctiveness**

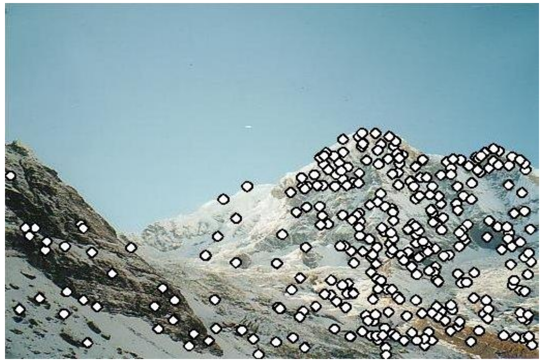
  Each feature can be uniquely identified

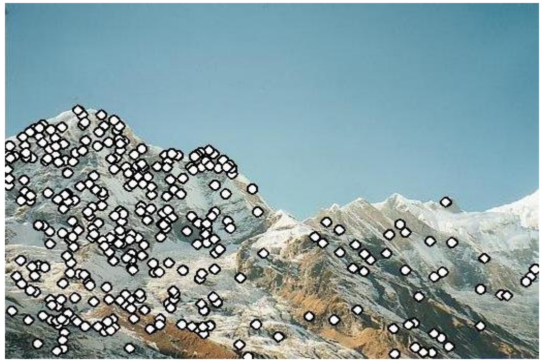
- **Repeatability**

  The same feature can be found in several images :
  - geometrically (translation, rotation, scale, perspective)
  - photometrically (reflectance, illumination)

- **Compactness and efficiency**

  - Many fewer features than image pixels
  - run independently per image





Kristen Grauman

Compactness and Efficiency

- We want the representation to be as small and as fast as possible
  - Much smaller than a whole image
- We'd like to be able to run the detection procedure *independently* per image
  - Match just the compact descriptors for speed.
  - *Difficult!* We don't get to see 'the other image' at match time, e.g., object detection.

Kristen Grauman

Questions?

Features

Lecture 9

Key-points

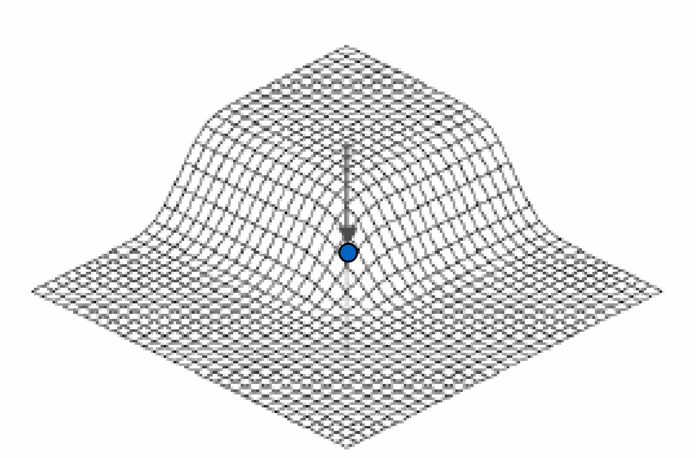
Choosing interest points

Where would you tell your friend to meet you?

Slide Credit: James Hays

What is an interest point?

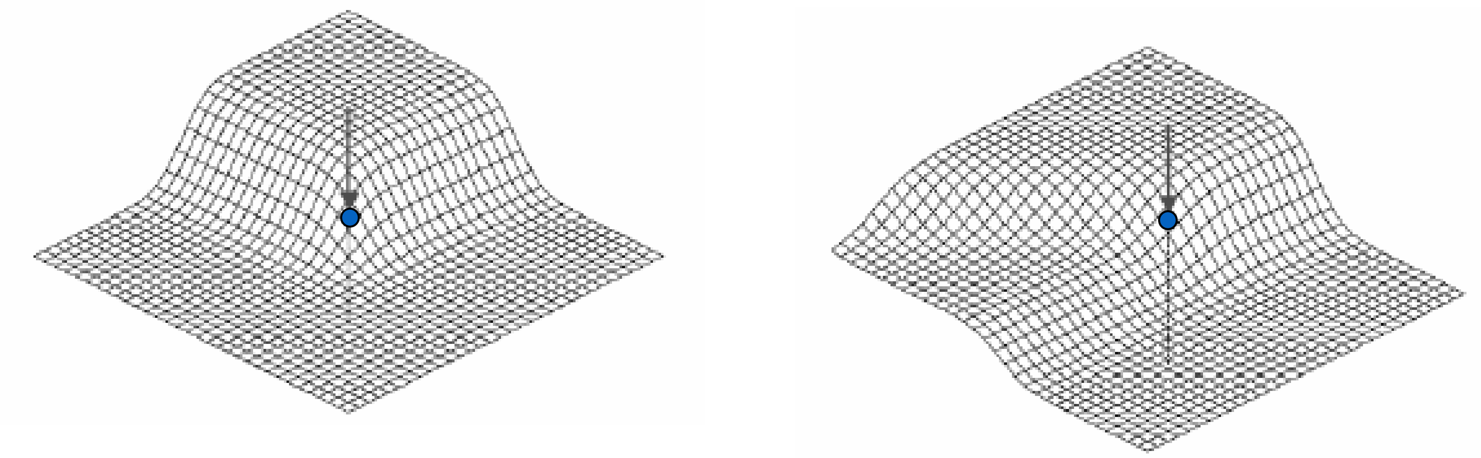
- Expressive texture
  - The point at which the direction of the boundary of object changes abruptly
  - Intersection point between two or more edge segments



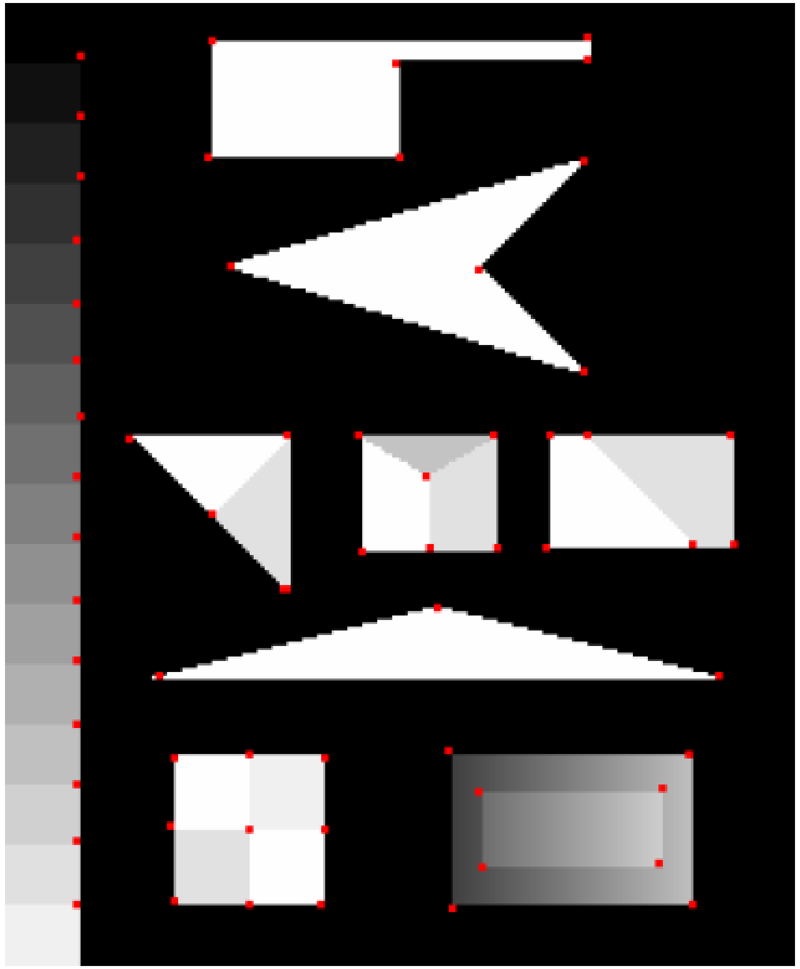
UCF Center for Research in Computer Vision

What is an interest point?

- Expressive texture
  - The point at which the direction of the boundary of object changes abruptly
  - Intersection point between two or more edge segments



What is an interest point?



Properties of Interest Points

- Detect all (or most) true interest points
- No false interest points
- Well localized
- Robust with respect to noise
- Efficient detection

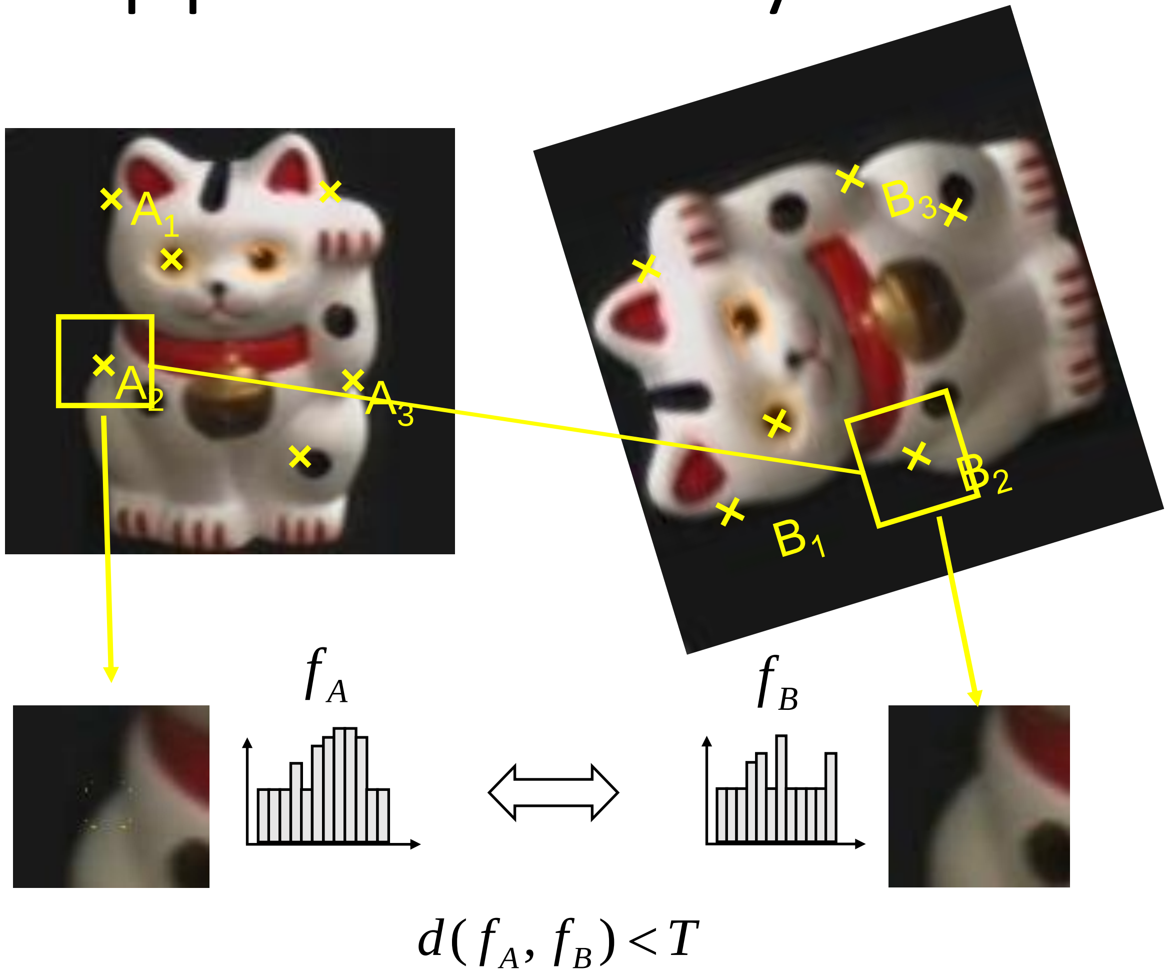
Possible approaches: corner detection

- Based on brightness of images
  - Usually image derivatives
- Based on boundary extraction
  - First step edge detection
  - Curvature analysis of edges

Goals for KeyPoints

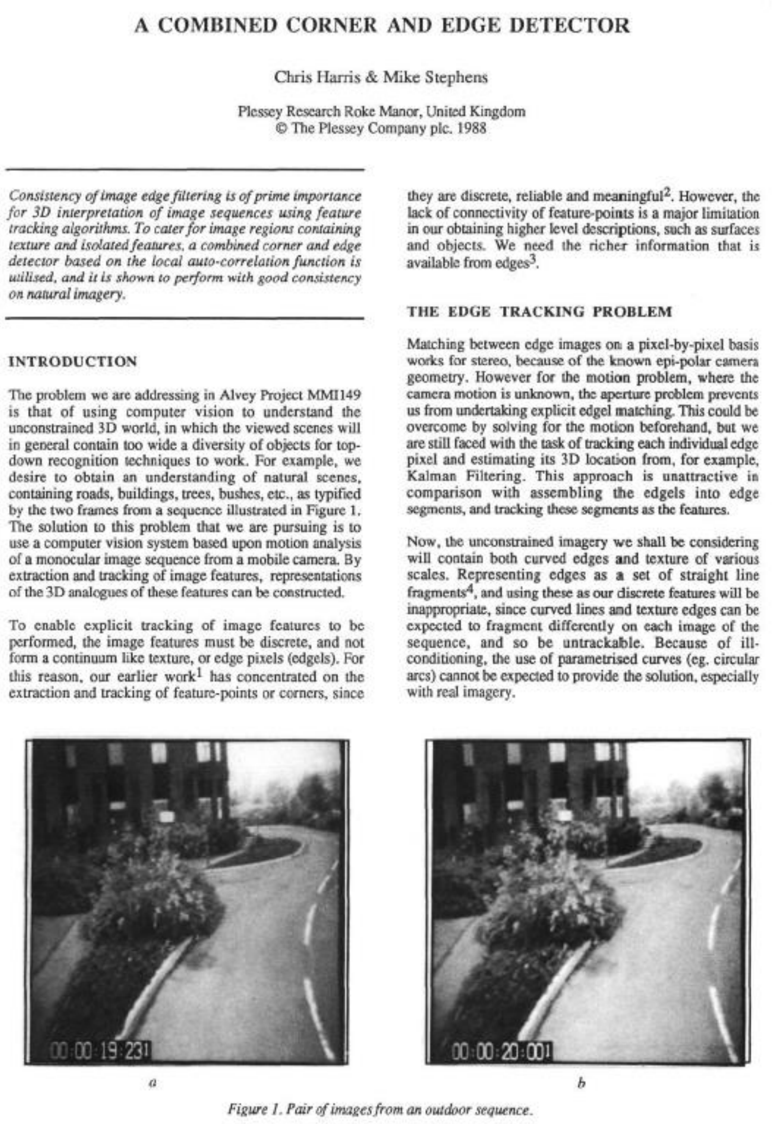
Detect points that are *repeatable* and *distinctive*

Application: KeyPoint Matching



1. Find a set of distinctive key-points
2. Define a region around each key-point
3. Extract and normalize the region content
4. Compute a local descriptor from the normalized region
5. Match local descriptors

K. Grauman, B. Leibe



A COMBINED CORNER AND EDGE DETECTOR

Chris Harris & Mike Stephens

Plessey Research Roke Manor, United Kingdom
© The Plessey Company plc. 1988

Consistency of image edge filtering is of prime importance for 3D interpretation of image sequences using feature tracking algorithms. To cater for image regions containing texture and isolated features, a combined corner and edge detector based on the local auto-correlation function is utilised, and it is shown to perform with good consistency on natural imagery.

INTRODUCTION

The problem we are addressing in Alvey Project MMI149 is that of using computer vision to understand the unconstrained 3D world, in which the viewed scenes will in general contain too wide a diversity of objects for top-down recognition techniques to work. For example, we desire to obtain an understanding of natural scenes, containing roads, buildings, trees, bushes, etc., as typified by the two frames from a sequence illustrated in Figure 1. The solution to this problem that we are pursuing is to use a computer vision system based upon motion analysis of a monocular image sequence from a mobile camera. By extraction and tracking of image features, representations of the 3D analogues of these features can be constructed.

To enable explicit tracking of image features to be performed, the image features must be discrete, and not form a continuum like texture, or edge pixels (edgels). For this reason, our earlier work[1] has concentrated on the extraction and tracking of feature-points or corners, since they are discrete, reliable and meaningful[2]. However, the lack of connectivity of feature-points is a major limitation in our obtaining higher level descriptions, such as surfaces and objects. We need the richer information that is available from edges[3].

THE EDGE TRACKING PROBLEM

Matching between edge images on a pixel-by-pixel basis works for stereo, because of the known epi-polar camera geometry. However for the motion problem, where the camera motion is unknown, the aperture problem prevents us from undertaking explicit edgel matching. This could be overcome by solving for the motion beforehand, but we are still faced with the task of tracking each individual edge pixel and estimating its 3D location from, for example, Kalman Filtering. This approach is unattractive in comparison with assembling the edgels into edge segments, and tracking these segments as the features.

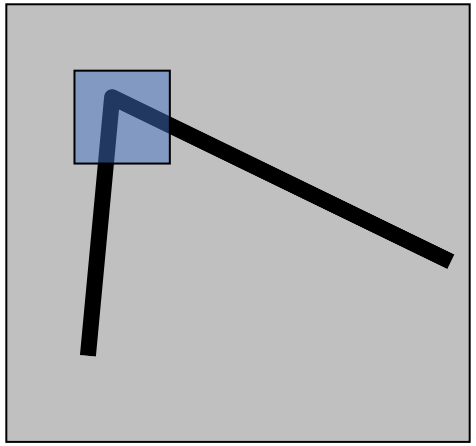
Now, the unconstrained imagery we shall be considering will contain both curved edges and texture of various scales. Representing edges as a set of straight line fragments[4], and using these as our discrete features will be inappropriate, since curved lines and texture edges can be expected to fragment differently on each image of the sequence, and so be untrackable. Because of ill-conditioning, the use of parametrised curves (eg. circular arcs) cannot be expected to provide the solution, especially with real imagery.

00:00:19:23 a

00:00:20:00 b

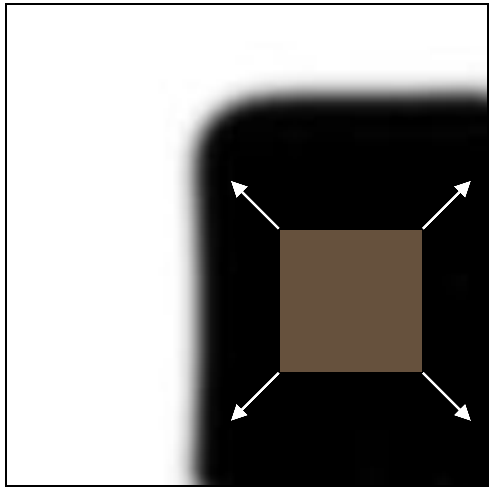
Figure 1. Pair of images from an outdoor sequence.

- Corner point can be recognized in a window
- Shifting a window in any direction should give a large change in intensity
- LOCALIZING and UNDERSTANDING shapes…

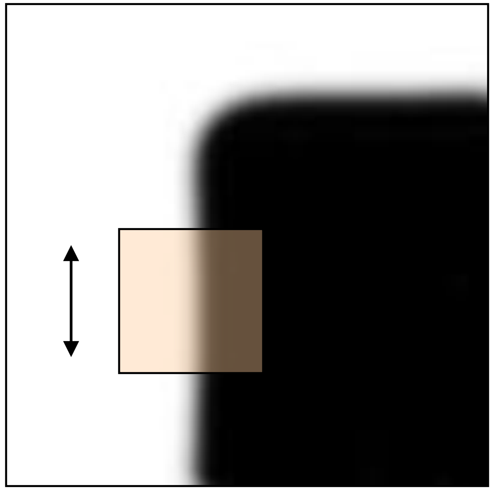


Basic Idea in Corner Detection

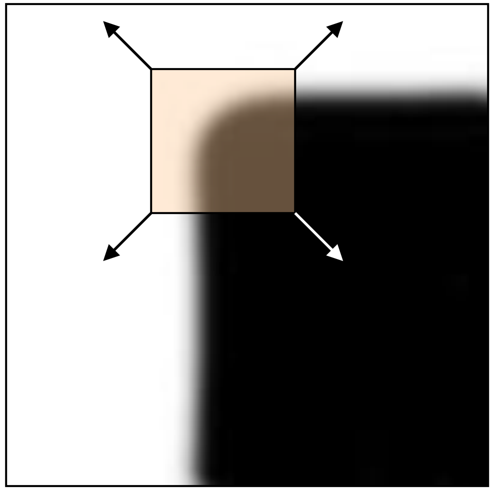
- Recognize corners by looking at small window.
- Shift in *any direction* to give *a large change* in intensity.



"Flat" region:
no change in all directions



"Edge":
no change along the edge direction



"Corner":
significant change in all directions

A. Efros

Template Matching

$$\begin{bmatrix} -4 & 5 & 5 \\ -4 & 5 & 5 \\ -4 & -4 & -4 \end{bmatrix} \qquad \begin{bmatrix} 5 & 5 & 5 \\ -4 & 5 & -4 \\ -4 & -4 & -4 \end{bmatrix}$$

Template Matching

$$\begin{bmatrix} -4 & 5 & 5 \\ -4 & 5 & 5 \\ -4 & -4 & -4 \end{bmatrix} \qquad \begin{bmatrix} 5 & 5 & 5 \\ -4 & 5 & -4 \\ -4 & -4 & -4 \end{bmatrix}$$

Complete set of eight templates can be generated by successive 90 degree of rotations.

Template Matching

$$\begin{bmatrix} -4 & 5 & 5 \\ -4 & 5 & 5 \\ -4 & -4 & -4 \end{bmatrix} \qquad \begin{bmatrix} 5 & 5 & 5 \\ -4 & 5 & -4 \\ -4 & -4 & -4 \end{bmatrix}$$

Complete set of eight templates can be generated by successive 90 degree of rotations.

Why the summation of filter is 0?

Template Matching

$$\begin{bmatrix} -4 & 5 & 5 \\ -4 & 5 & 5 \\ -4 & -4 & -4 \end{bmatrix} \qquad \begin{bmatrix} 5 & 5 & 5 \\ -4 & 5 & -4 \\ -4 & -4 & -4 \end{bmatrix}$$

Complete set of eight templates can be generated by successive 90 degree of rotations.

Why the summation of filter is 0? → Insensitive to absolute change In intensity!

Correlation - revisit

$$f \otimes h = \sum_k \sum_l f(k,l)h(k,l)$$

$$f = \text{Image}$$

$$h = \text{Kernel}$$

f

f_1	f_2	f_3
f_4	f_5	f_6
f_7	f_8	f_9

$\otimes$

h

h_1	h_2	h_3
h_4	h_5	h_6
h_7	h_8	h_9

$$\begin{aligned} f \otimes h = {} & f_1 h_1 + f_2 h_2 + f_3 h_3 \\ & + f_4 h_4 + f_5 h_5 + f_6 h_6 \\ & + f_7 h_7 + f_8 h_8 + f_9 h_9 \end{aligned}$$

Corner Detection by Auto-correlation

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

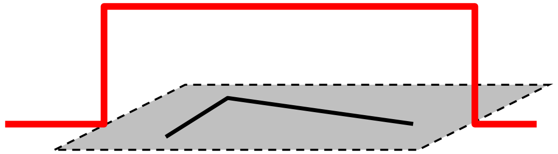
$$E(u,v) = \sum_{x,y} w(x,y)\left[I(x+u,y+v) - I(x,y)\right]^2$$

Window function

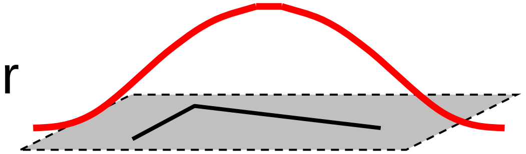
Shifted intensity

Intensity

Window function $w(x,y)$ =



or



1 in window, 0 outside

Gaussian

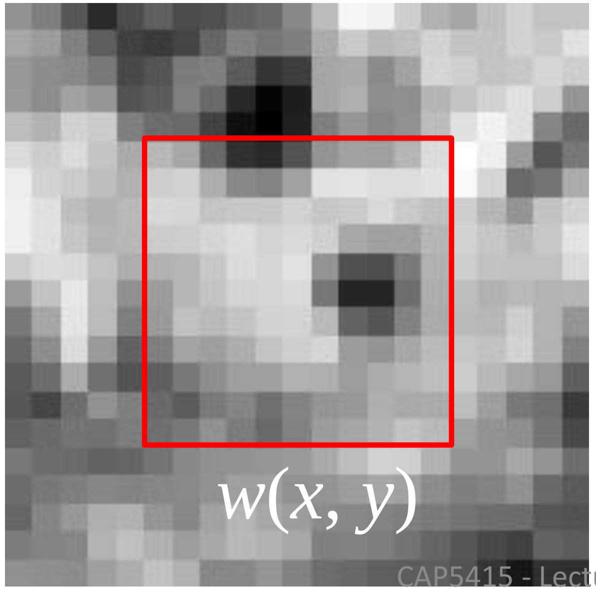
Source: R. Szeliski

Corner Detection by Auto-correlation

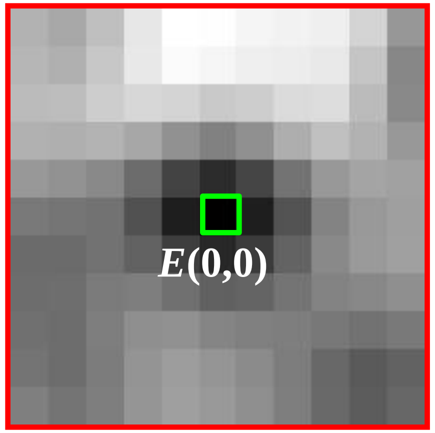
Change in appearance of window $w(x,y)$ for shift $[u,v]$:

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

$I(x, y)$



$E(u, v)$

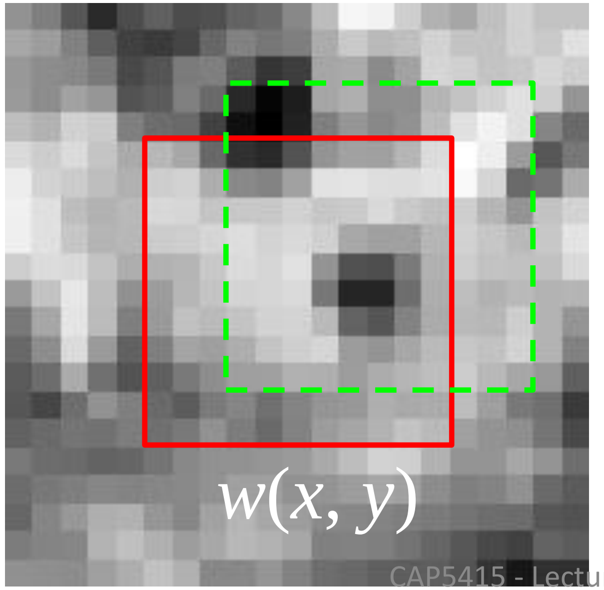


Corner Detection by Auto-correlation

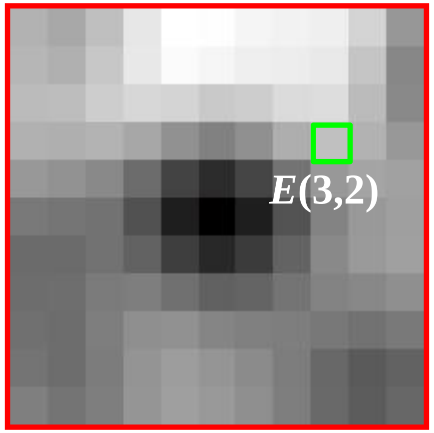
Change in appearance of window $w(x,y)$ for shift $[u,v]$:

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

$I(x, y)$



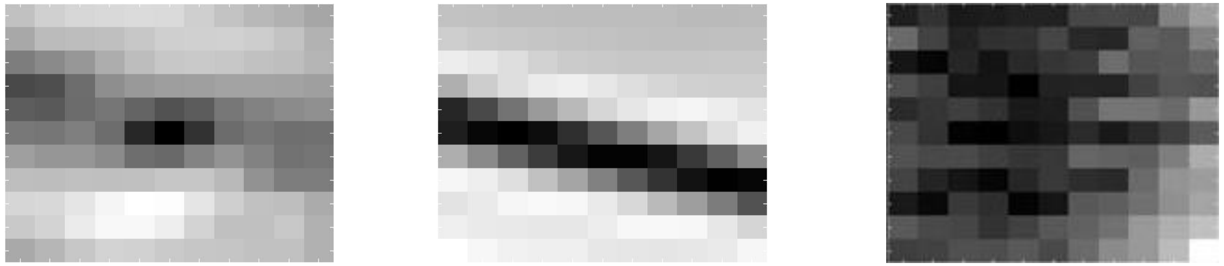
$E(u, v)$



Corner detection

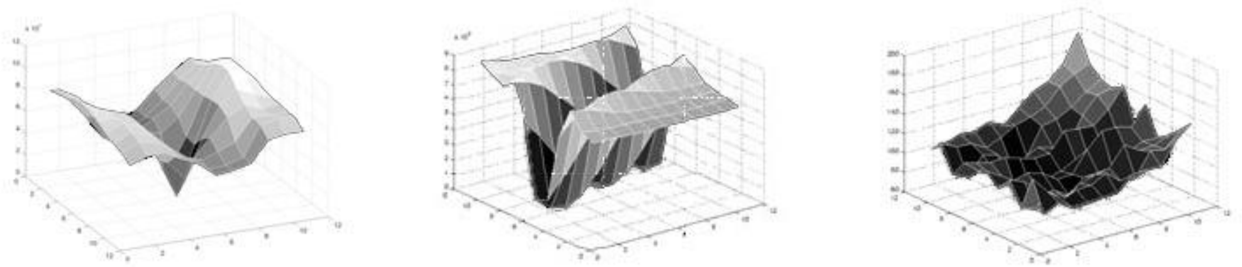
Three different cases

$E(u,v)$



$E(u,v)$

As a surface



Corner Detection by Auto-correlation

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

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

We want to discover how E behaves for small shifts

But this is very slow to compute naively.
O(window_width^2 * shift_range^2 * image_width^2)

O(11^2 * 11^2 * 600^2) = 5.2 billion of these
14.6k ops per image pixel

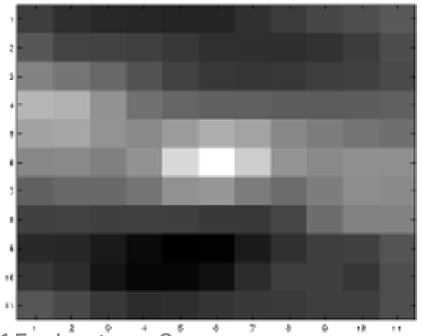
Corner Detection by Auto-correlation

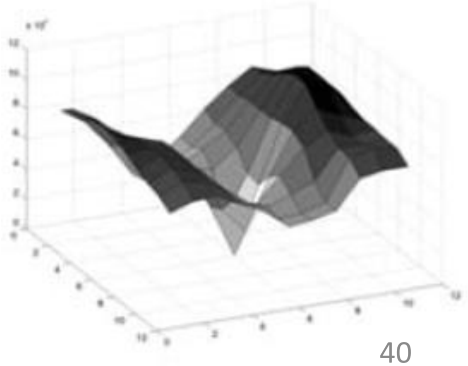
Change in appearance of window $w(x,y)$ for shift $[u,v]$:

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

We want to discover how E behaves for small shifts

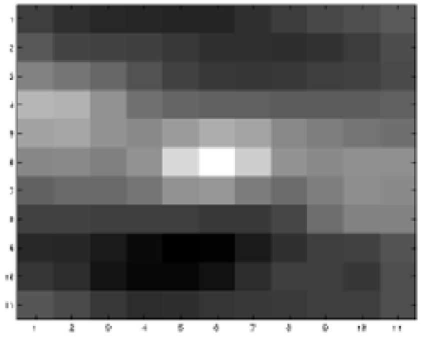
But we know the response in *E* that we are looking for – strong peak.

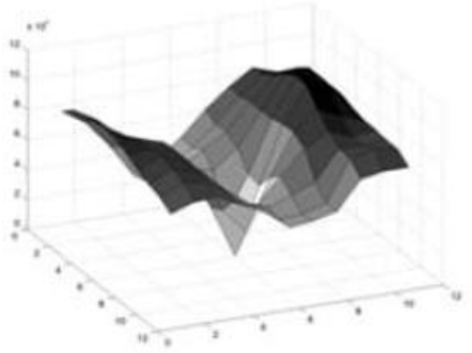




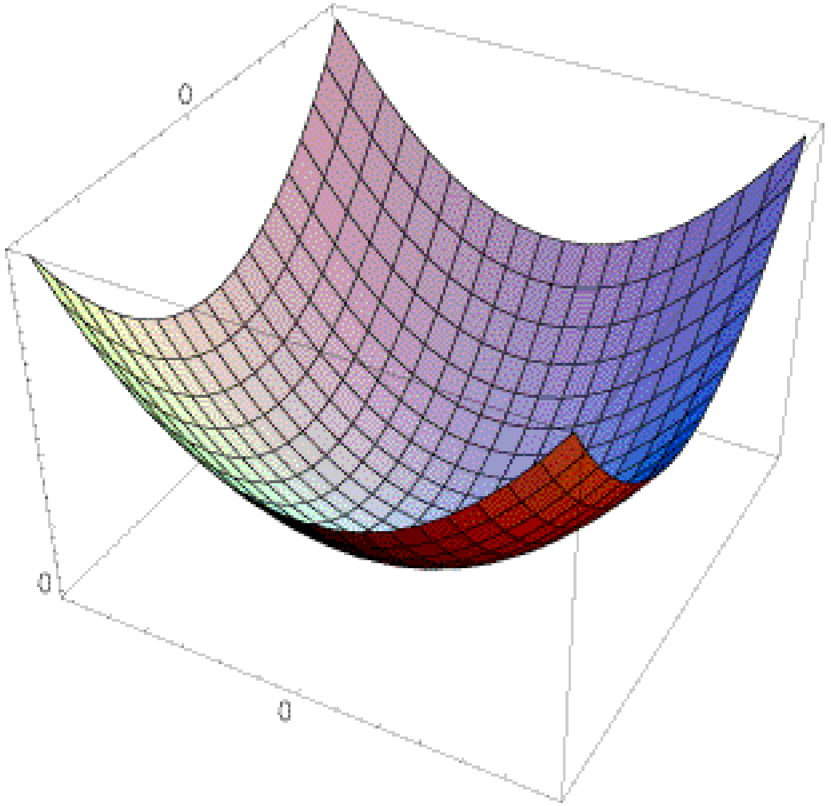
Corner Detection: strategy

Approximate *E*(*u*,*v*) locally by a quadratic surface





$\approx$



UCF Center for Research in Computer Vision

Recall: Taylor series expansion

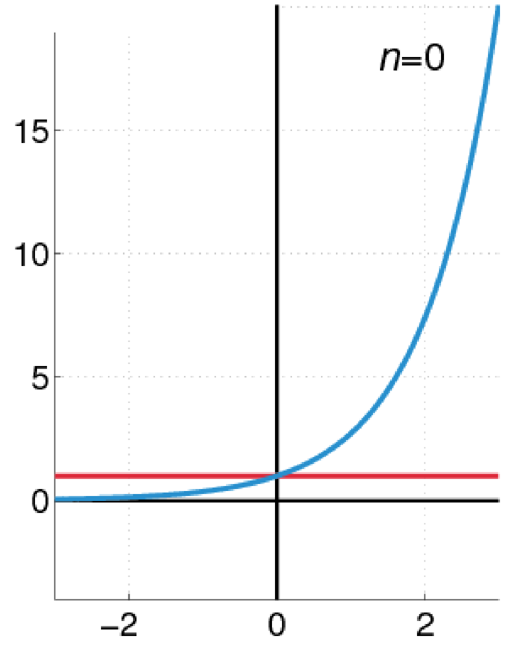
- A function f can be represented by
  - an infinite series of its derivatives at a single point *a*:

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

Wikipedia

As we care about window centered, we set *a = 0* (MacLaurin series)

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



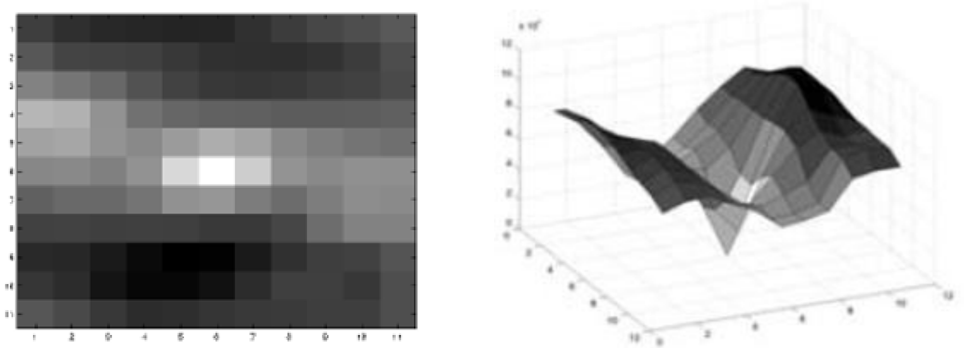
Corner Detection by Auto-correlation

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

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

We want to discover how E behaves for small shifts

But we know the response in *E* that we are looking for – strong peak.



Corner Detection: Mathematics

The quadratic approximation simplifies to

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

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

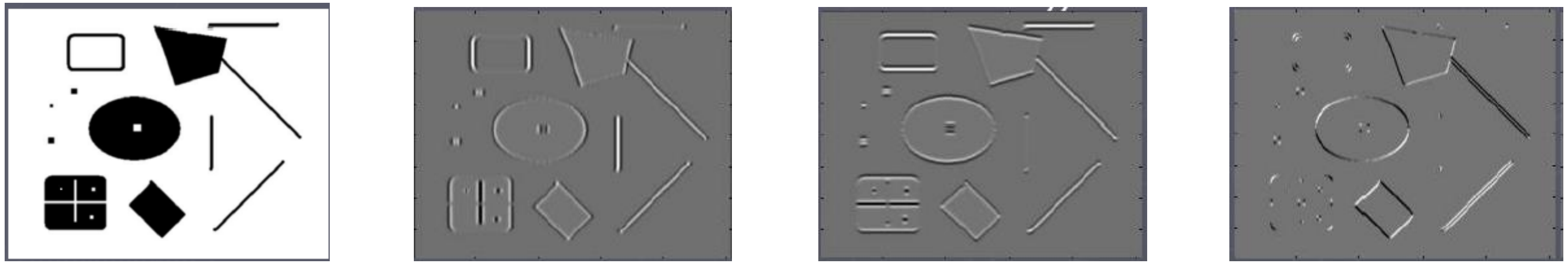
$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

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

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

James Hays

Harris corner detection

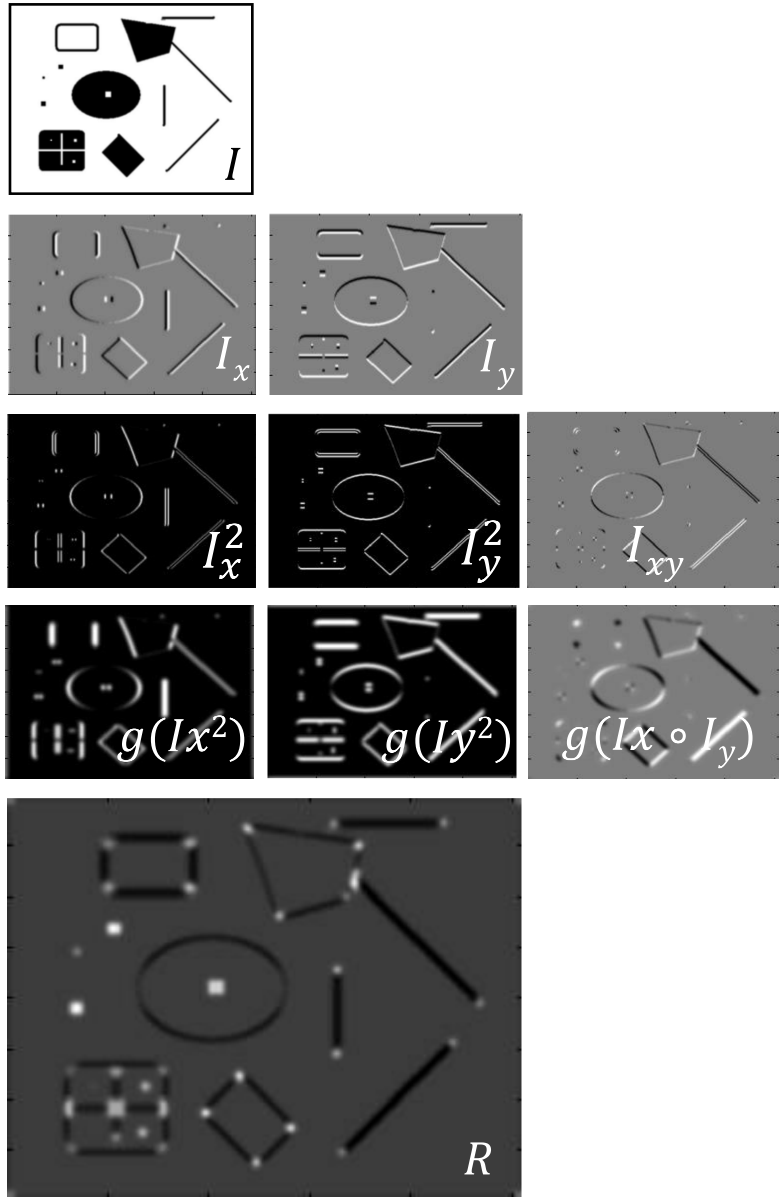
1) Compute M matrix for each window to recover a *cornerness* score C.

   Note: We can find M purely from the per-pixel image derivatives!

2) Threshold to find pixels which give large corner response C > threshold.

3) Find the local maxima pixels, i.e., non-maximal suppression.

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



0. Input image
   We want to compute M at each pixel.

1. Compute image derivatives (optionally, blur first).

2. Compute M components as squares of derivatives.

3. Gaussian filter *g()* with width *s*

   $= g(I_x^2), g(I_y^2), g(I_x \circ I_y)$

4. Compute cornerness

$$C = \det(M) - \alpha \operatorname{trace}(M)^2$$
$$= g(I_x^2) \circ g(I_y^2) - g(I_x \circ I_y)^2 \; - \alpha\left[g(I_x^2) + g(I_y^2)\right]^2$$

5. Threshold on C to pick high cornerness

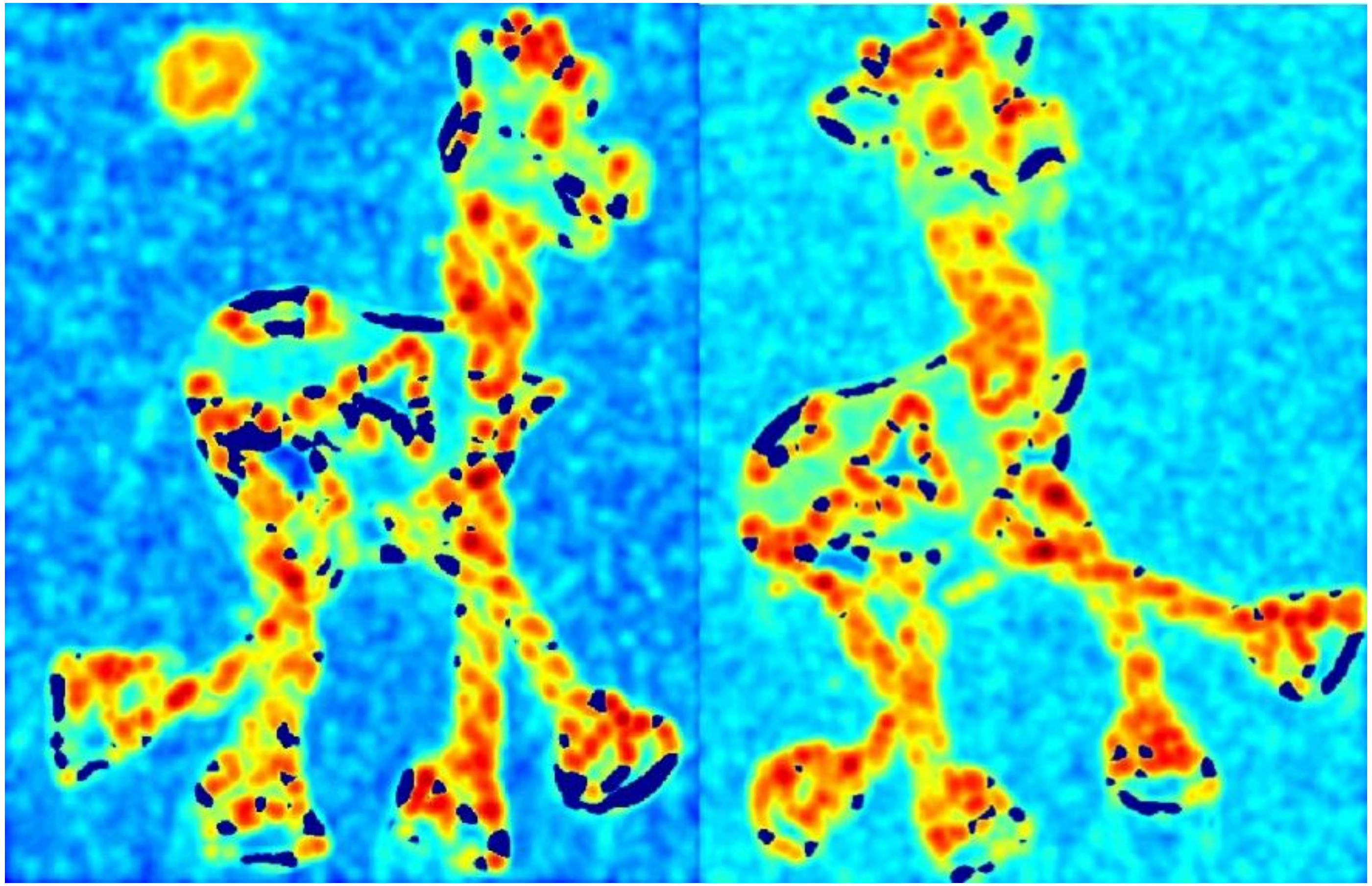
6. Non-maximal suppression to pick peaks.

James Hays

Harris Detector: Steps

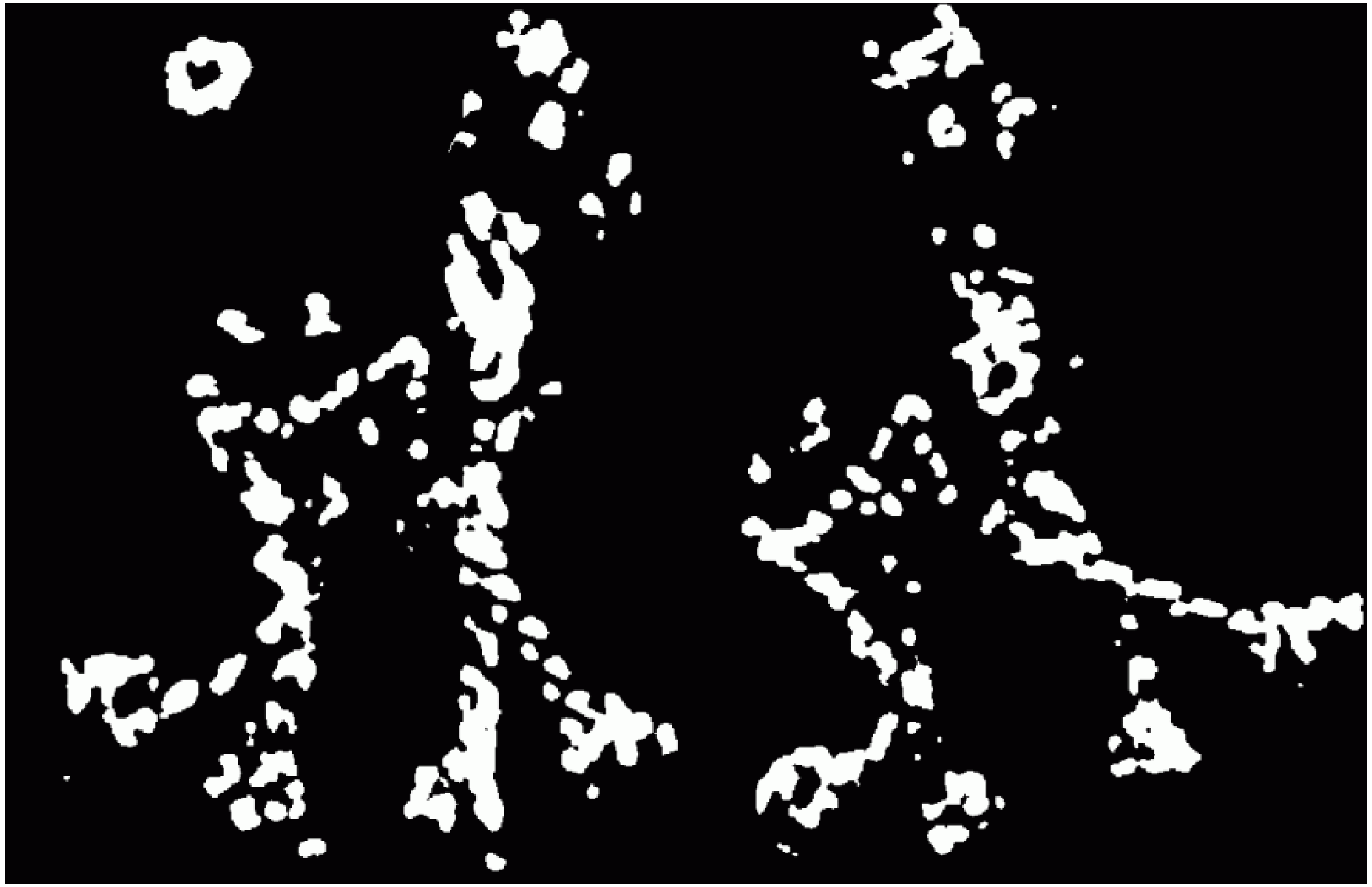
Harris Detector: Steps

Compute corner response C



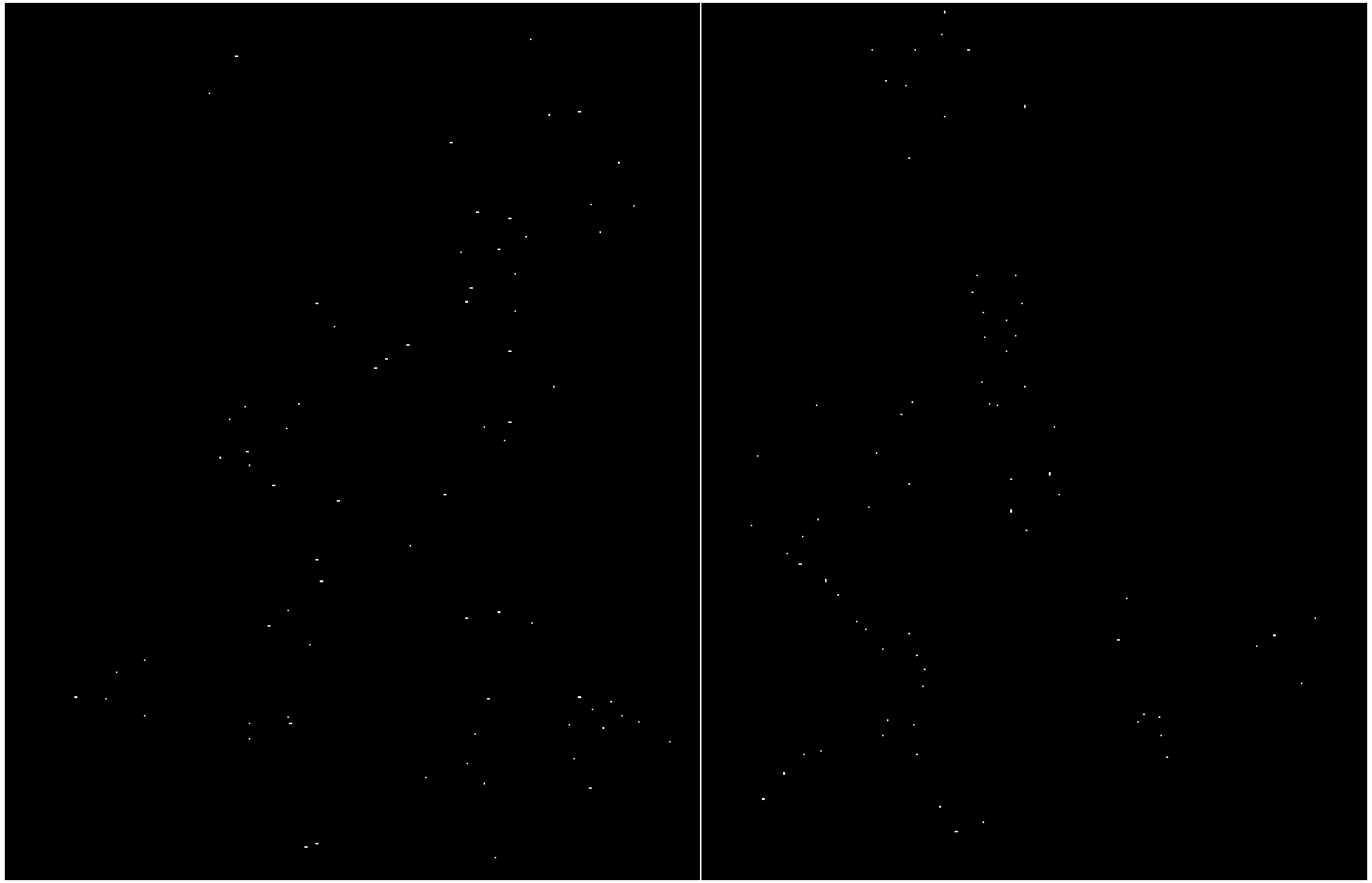
Harris Detector: Steps

Find points with large corner response: C > threshold



Harris Detector: Steps

Take only the points of local maxima of C



Harris Detector: Steps

Questions?

CENTER FOR RESEARCH IN COMPUTER VISION

CAP5415 Computer Vision

Yogesh S Rawat

yogesh@ucf.edu

HEC-241

UCF CENTER FOR RESEARCH IN COMPUTER VISION

Questions?

Features

Lecture 9

Histogram of Gradients (HoG)

Edges

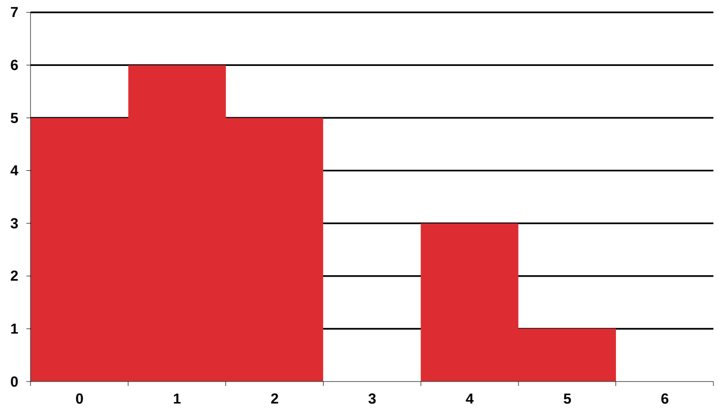
HOG: Human Detection

Navneet Dalal and Bill Triggs "Histograms of Oriented Gradients for Human Detection" CVPR05

Histogram - revisit

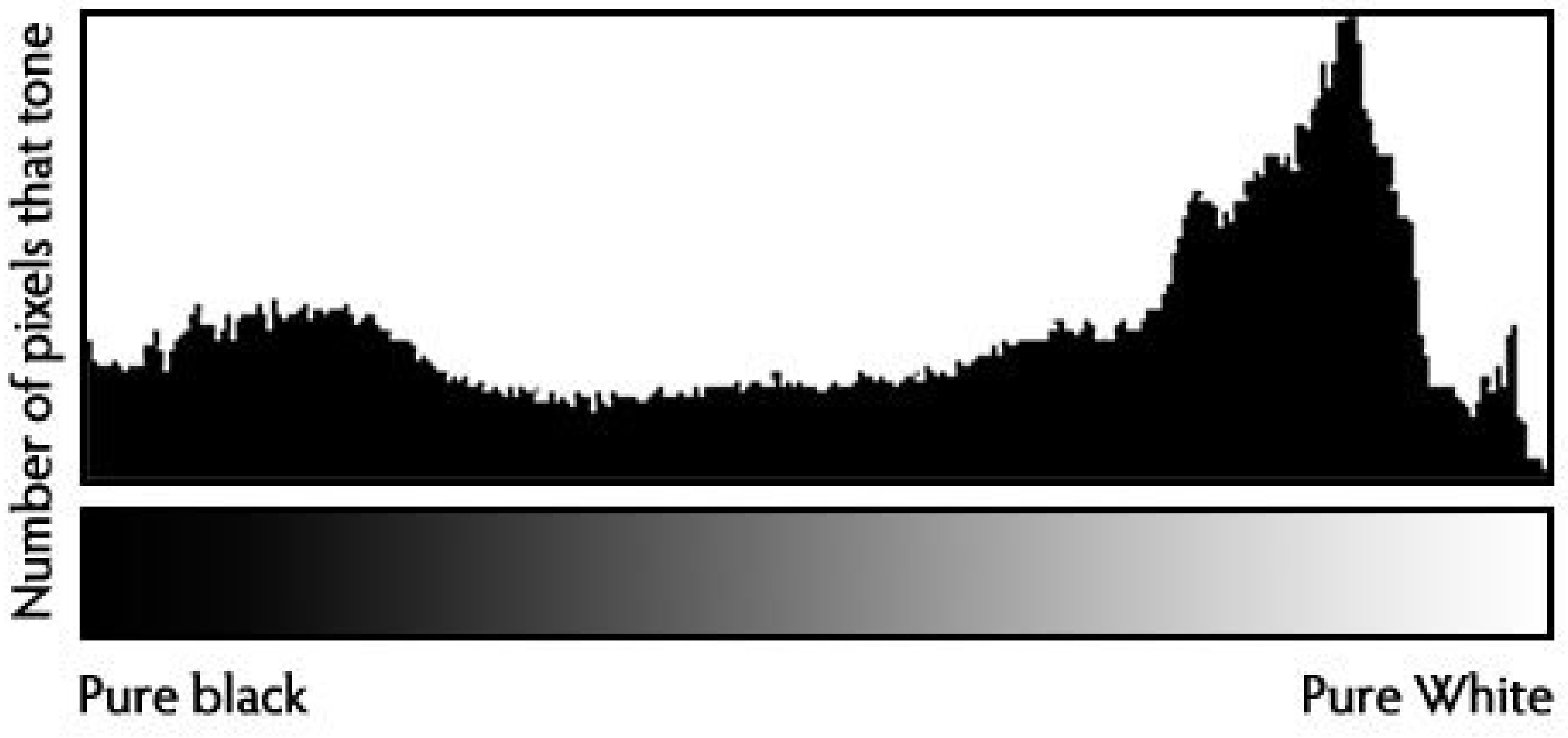
0	1	1	2	4
2	1	0	0	2
5	2	0	0	4
1	1	2	4	1

image



histogram

Image Histogram - revisit



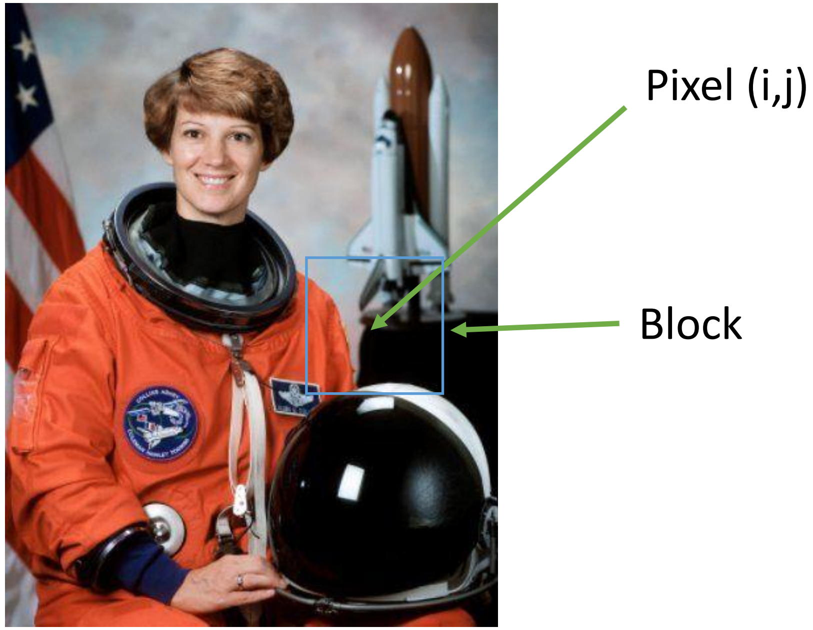
Histograms of Oriented Gradients

- Given an image I, and a pixel location (i,j).
- We want to compute the HOG feature for that pixel.
- The main operations can be described as a sequence of five steps.

Pixel (i,j)

Histograms of Oriented Gradients

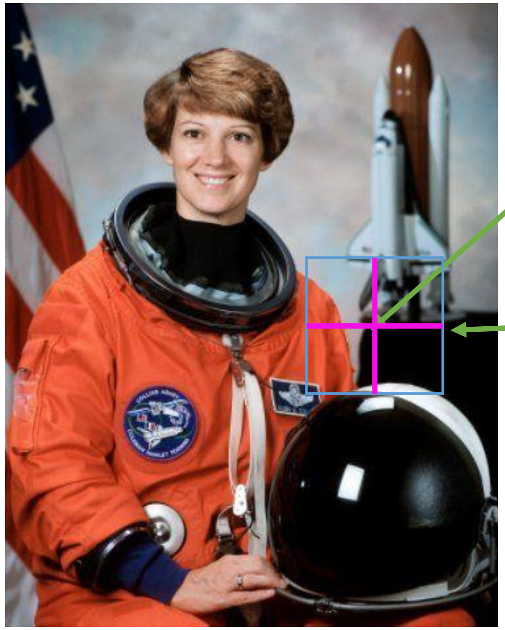
- Step 1: Extract a square window (called “block”) of some size.



Histograms of Oriented Gradients

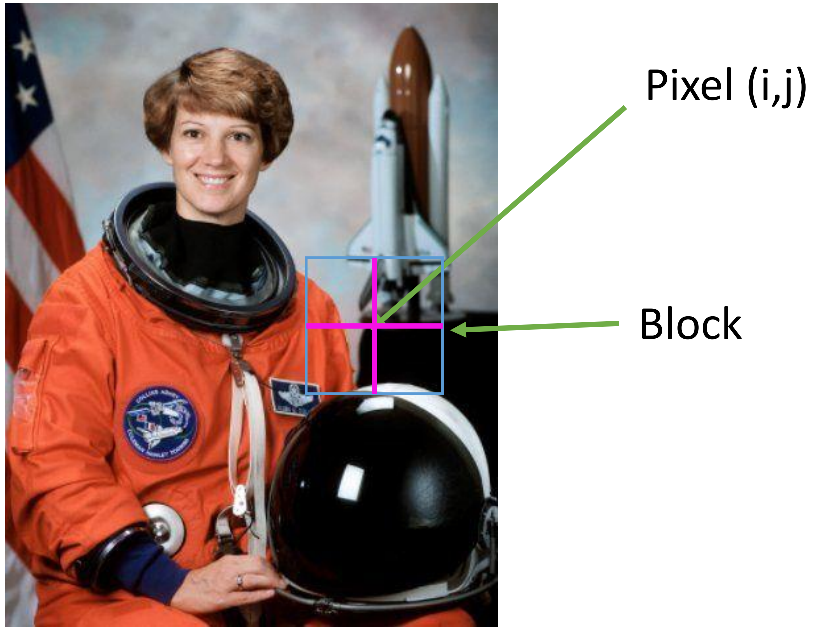
- Step 2: Divide block into a square grid of sub-blocks

(called “cells”) (2x2 grid in our example, resulting in four cells).



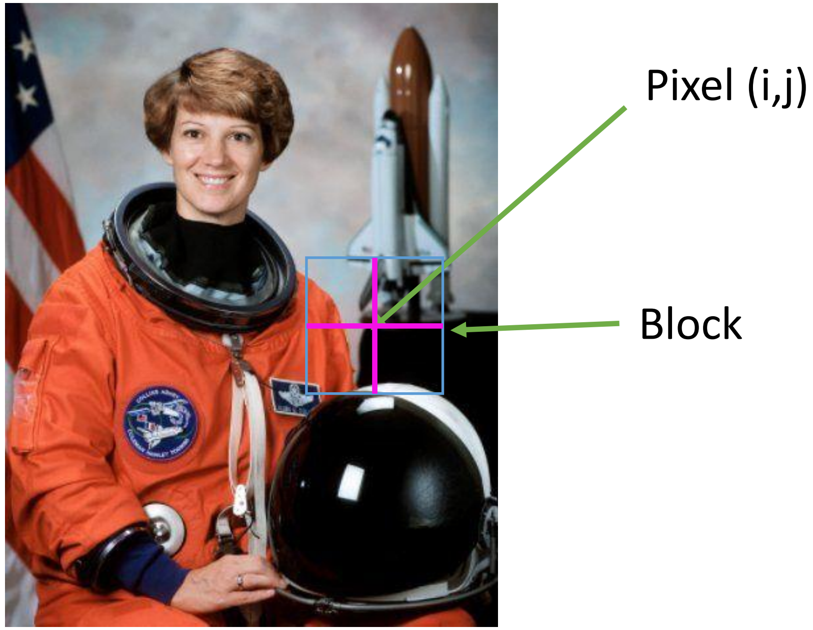
Histograms of Oriented Gradients

- Step 3: Compute orientation histogram of each cell.



Histograms of Oriented Gradients

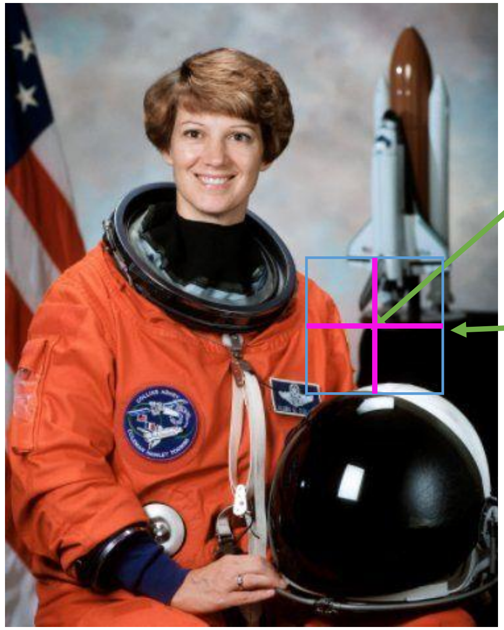
- Step 4: Concatenate the four histograms.



Histograms of Oriented Gradients

Let vector **v** be concatenation of the four histograms from step 4.

- Step 5: normalize **v**. Here we have three options for how to do it:
  - Option 1: Divide **v** by its Euclidean norm.



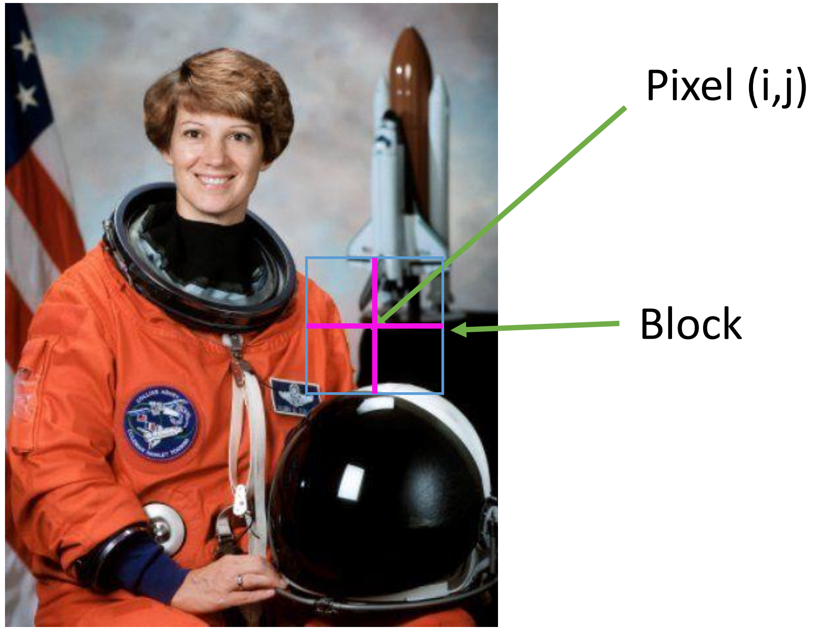
Pixel (i,j)

Block

Histograms of Oriented Gradients

Let vector **v** be concatenation of the four histograms from step 4.

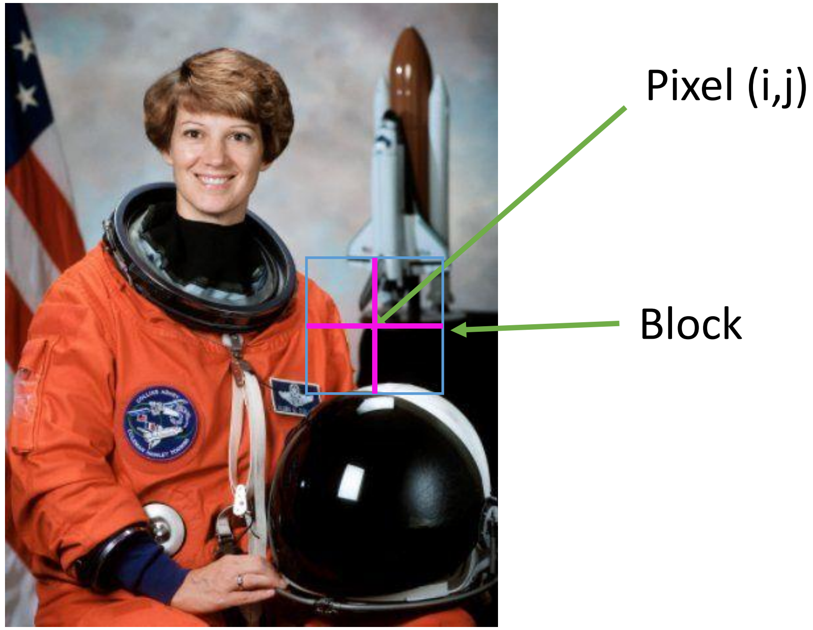
- Step 5: normalize **v**. Here we have three options for how to do it:
  - Option 2: Divide **v** by its L_1 norm (the L_1 norm is the sum of all absolute values of v).



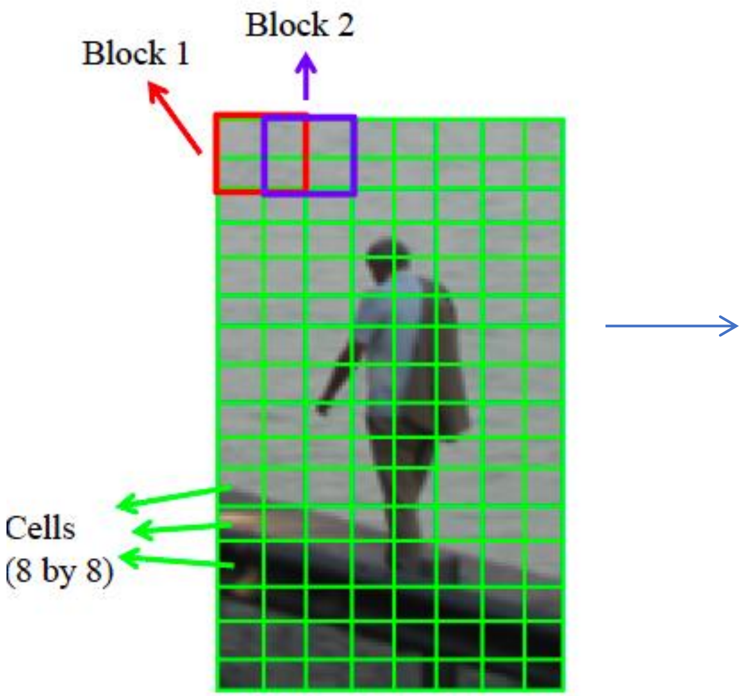
Histograms of Oriented Gradients

Let vector **v** be concatenation of the four histograms from step 4.

- Option 3:
  - Divide **v** by its Euclidean norm.
  - In the resulting vector, clip any value over 0.2
  - Then, renormalize the resulting vector by dividing again by its Euclidean norm.



Histogram of Oriented Gradients



- Each block consists of 2x2 cells with size 8x8

Image gradients

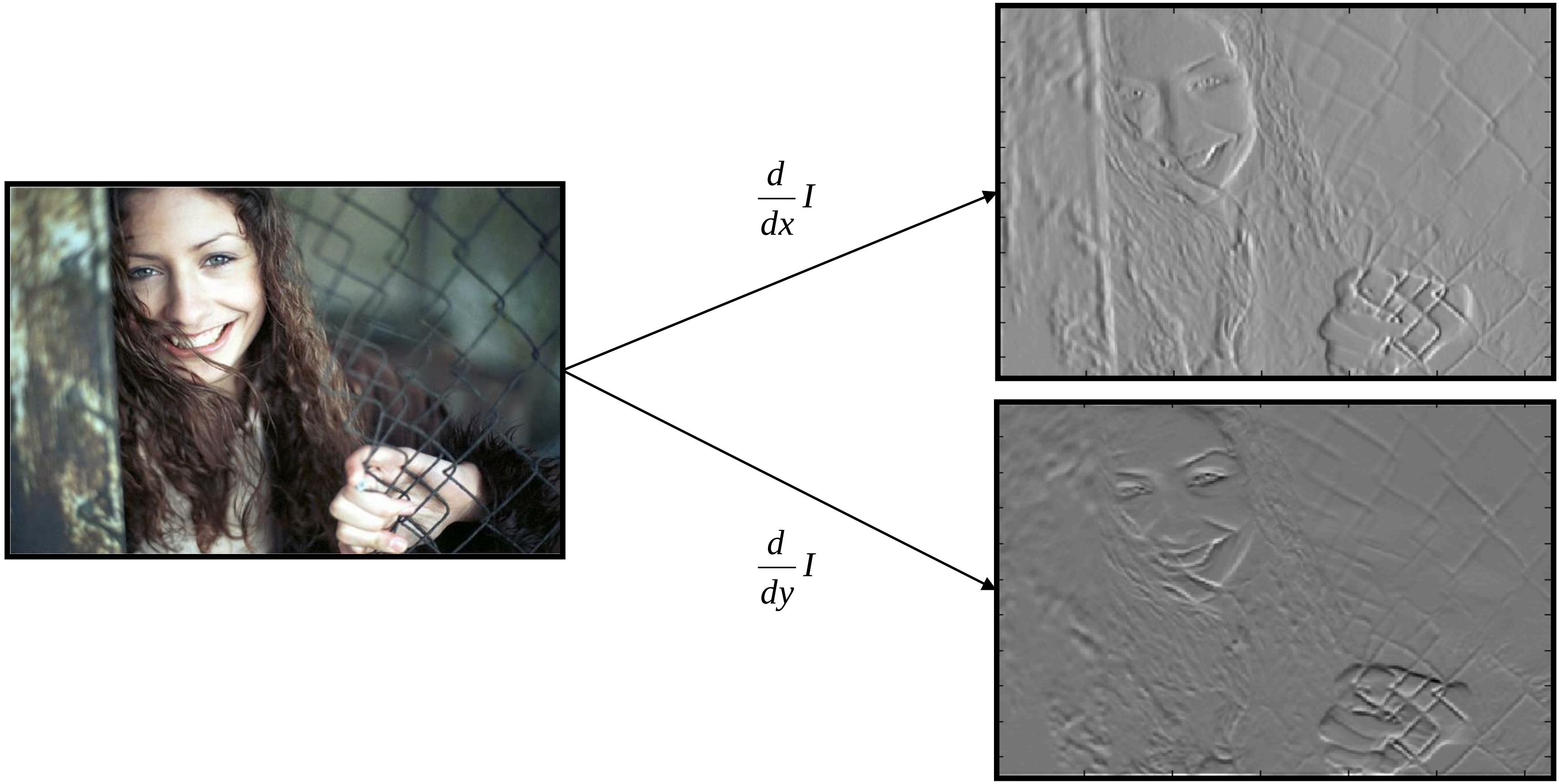
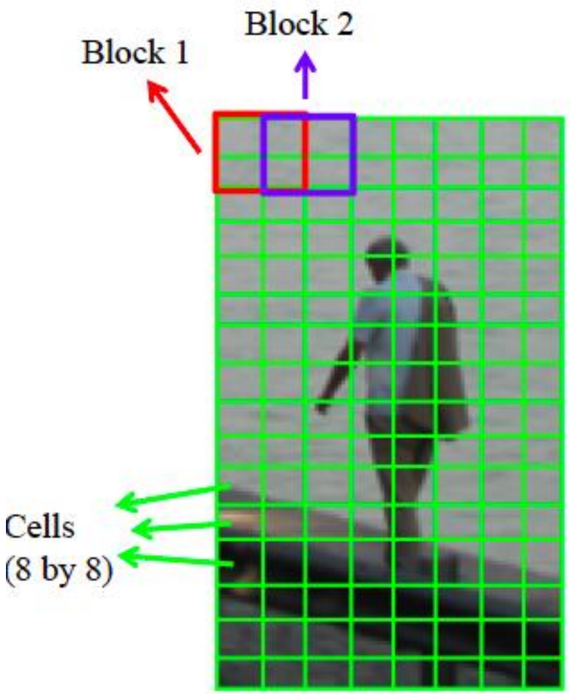


Image gradients

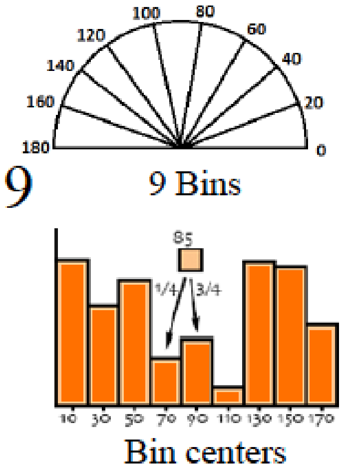
Gradient magnitude $\left|\nabla f(x,y)\right| = \sqrt{f_x^{\,2} + f_y^{\,2}}$

Gradient direction $\theta = \tan^{-1}\dfrac{f_x}{f_y}$

Histogram of Oriented Gradients



- Each block consists of 2x2 cells with size 8x8
- Quantize the gradient orientation into 9 bins (0-180)
  - The vote is the gradient magnitude



Summary of HOG Computation

- Step 1: Extract a square window (called “block”) of some size around the pixel location of interest.
- Step 2: Divide block into a square grid of sub-blocks (called “cells”) (2x2 grid in our example, resulting in four cells).
- Step 3: Compute orientation histogram of each cell.
- Step 4: Concatenate the four histograms.
- Step 5: normalize **v** using one of the three options described previously.

Histograms of Oriented Gradients

- Parameters and design options:
  - Angles range from 0 to 180 or from 0 to 360 degrees?
    - In the Dalal & Triggs paper, a range of 0 to 180 degrees is used,
    - and HOGs are used for detection of pedestrians.
  - Number of orientation bins.
    - Usually 9 bins, each bin covering 20 degrees.
  - Cell size.
    - Cells of size 8x8 pixels are often used.
  - Block size.
    - Blocks of size 2x2 cells (16x16 pixels) are often used.
- Usually a HOG feature has 36 dimensions.
  - 4 cells * 9 orientation bins.

HOG

Input image

Histogram of Oriented Gradients

Questions?

Features

Lecture 9

SIFT

UCF CENTER FOR RESEARCH IN COMPUTER VISION

Scale Invariant Feature Transform (SIFT)

- Lowe., D. 2004, IJCV

cited > 68K

Distinctive Image Features from Scale-Invariant Keypoints

DAVID G. LOWE
Computer Science Department, University of British Columbia, Vancouver, B.C., Canada
Lowe@cs.ubc.ca

Received January 10, 2003; Revised January 7, 2004; Accepted January 22, 2004

Abstract. This paper presents a method for extracting distinctive invariant features from images that can be used to perform reliable matching between different views of an object or scene. The features are invariant to image scale and rotation, and are shown to provide robust matching across a substantial range of affine distortion, change in 3D viewpoint, addition of noise, and change in illumination. The features are highly distinctive, in the sense that a single feature can be correctly matched with high probability against a large database of features from many images. This paper also describes an approach to using these features for object recognition. The recognition proceeds by matching individual features to a database of features from known objects using a fast nearest-neighbor algorithm, followed by a Hough transform to identify clusters belonging to a single object, and finally performing verification through least-squares solution for consistent pose parameters. This approach to recognition can robustly identify objects among clutter and occlusion while achieving near real-time performance.

Keywords: invariant features, object recognition, scale invariance, image matching

1. Introduction

Image matching is a fundamental aspect of many problems in computer vision, including object or scene recognition, solving for 3D structure from multiple images, stereo correspondence, and motion tracking. This paper describes image features that have many properties that make them suitable for matching differing images of an object or scene. The features are invariant to image scaling and rotation, and partially invariant to change in illumination and 3D camera viewpoint. They are well localized in both the spatial and frequency domains, reducing the probability of disruption by occlusion, clutter, or noise. Large numbers of features can be extracted from typical images with efficient algorithms. In addition, the features are highly distinctive, which allows a single feature to be correctly matched with high probability against a large database of features, providing a basis for object and scene recognition.

The cost of extracting these features is minimized by taking a cascade filtering approach, in which the more expensive operations are applied only at locations that pass an initial test. Following are the major stages of computation used to generate the set of image features:

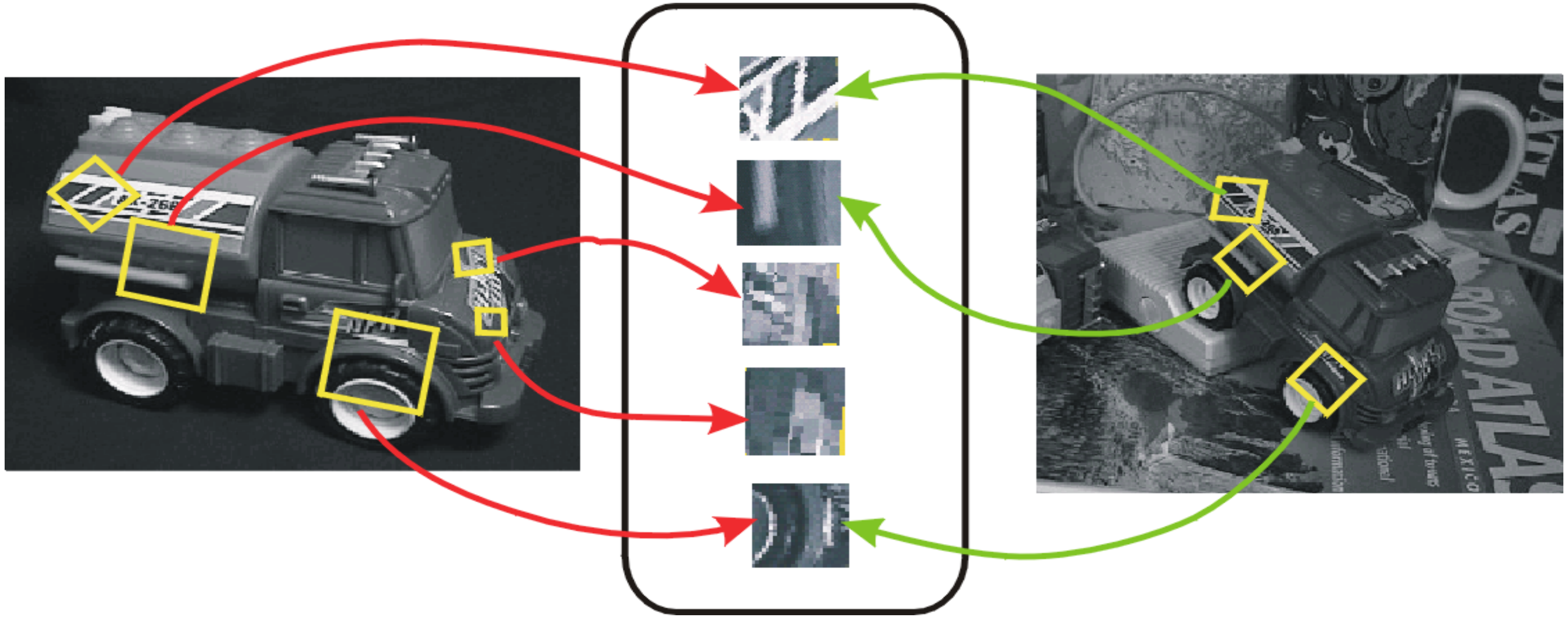
1. *Scale-space extrema detection*: The first stage of computation searches over all scales and image locations. It is implemented efficiently by using a difference-of-Gaussian function to identify potential interest points that are invariant to scale and orientation.
2. *Keypoint localization*: At each candidate location, a detailed model is fit to determine location and scale. Keypoints are selected based on measures of their stability.
3. *Orientation assignment*: One or more orientations are assigned to each keypoint location based on local image gradient directions. All future operations are performed on image data that has been transformed relative to the assigned orientation, scale, and location for each feature, thereby providing invariance to these transformations.

Scale Invariant Feature Transform (SIFT)

- Image content is transformed into local feature coordinates
- Invariant to
  - translation
  - rotation
  - scale, and
  - other imaging parameters

Scale Invariant Feature Transform (SIFT)

- Image content is transformed into local feature coordinates



Overall Procedure at a High Level

Scale-Space Extrema Detection	Search over multiple scales and image locations
KeyPoint Localization	Fit a model to determine location and scale. Select KeyPoints based on a measure of stability.
Orientation Assignment	Compute best orientation(s) for each keyPoint region.
KeyPoint Description	Use local image gradients at selected scale and rotation to describe each keyPoint region.

Automatic Scale Selection

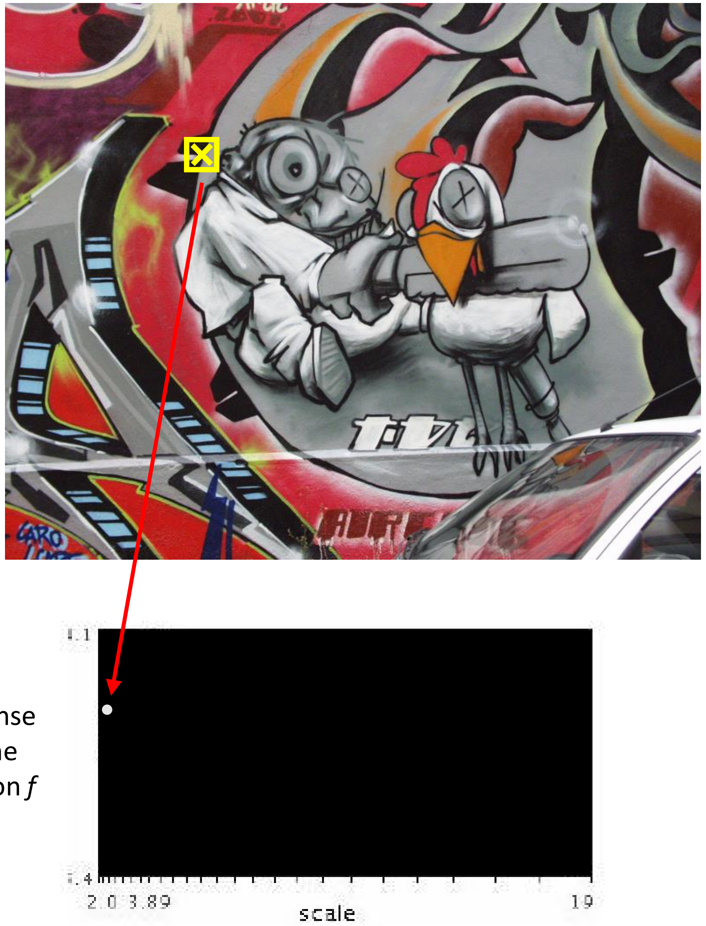
$$f(I_{i_1 \dots i_m}(x,\sigma)) = f(I_{i_1 \dots i_m}(x',\sigma'))$$

How to find patch sizes at which *f* response is equal?

What is a good *f* ?

Automatic Scale Selection

- Function responses for increasing scale (scale signature)



Response of some function *f*

$f(I_{i_1 \ldots i_m}(x, \sigma))$

$f(I_{i_1 \ldots i_m}(x', \sigma'))$

Automatic Scale Selection

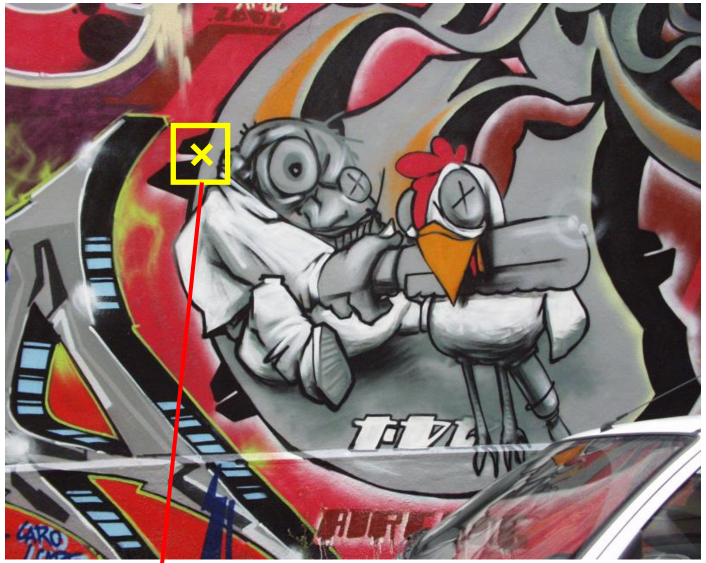
- Function responses for increasing scale (scale signature)

Response of some function *f*

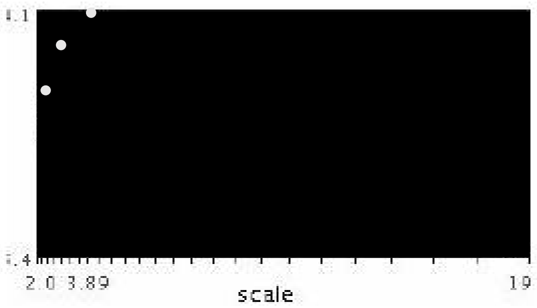
$f(I_{i_1 \ldots i_m}(x,\sigma))$ $f(I_{i_1 \ldots i_m}(x',\sigma'))$

Automatic Scale Selection

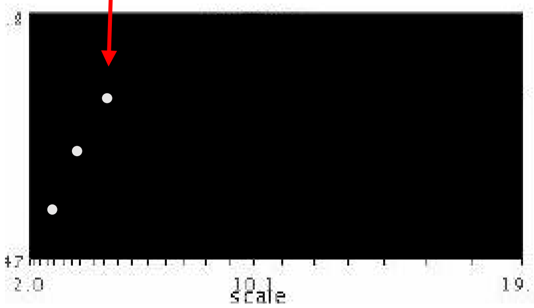
- Function responses for increasing scale (scale signature)



Response of some function *f*



$f(I_{i_1 \ldots i_m}(x, \sigma))$



$f(I_{i_1 \ldots i_m}(x', \sigma'))$

Automatic Scale Selection

- Function responses for increasing scale (scale signature)

Response
of some
function *f*

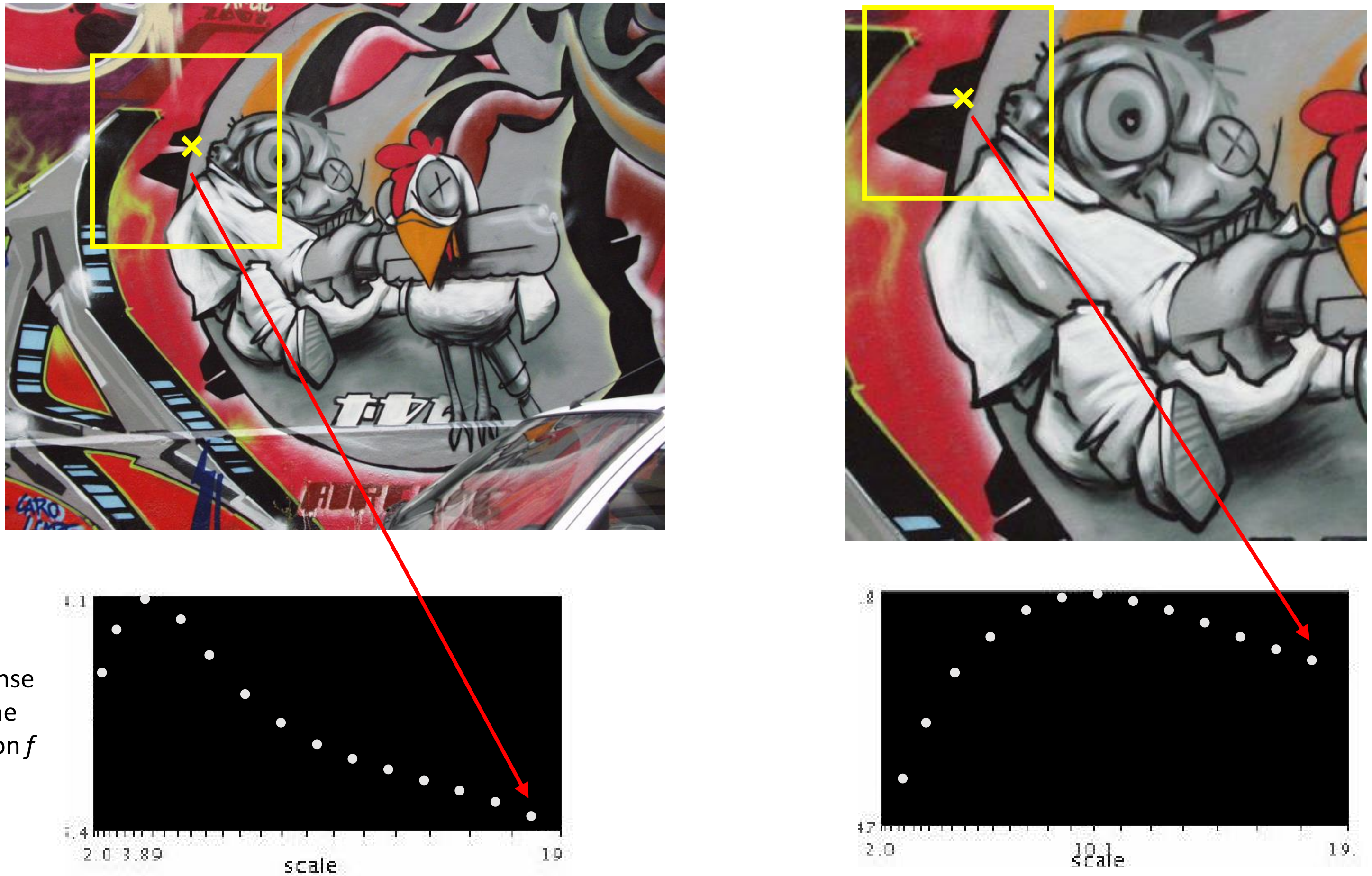
$f(I_{i_1 \ldots i_m}(x, \sigma))$

$f(I_{i_1 \ldots i_m}(x', \sigma'))$

Automatic Scale Selection

- Function responses for increasing scale (scale signature)

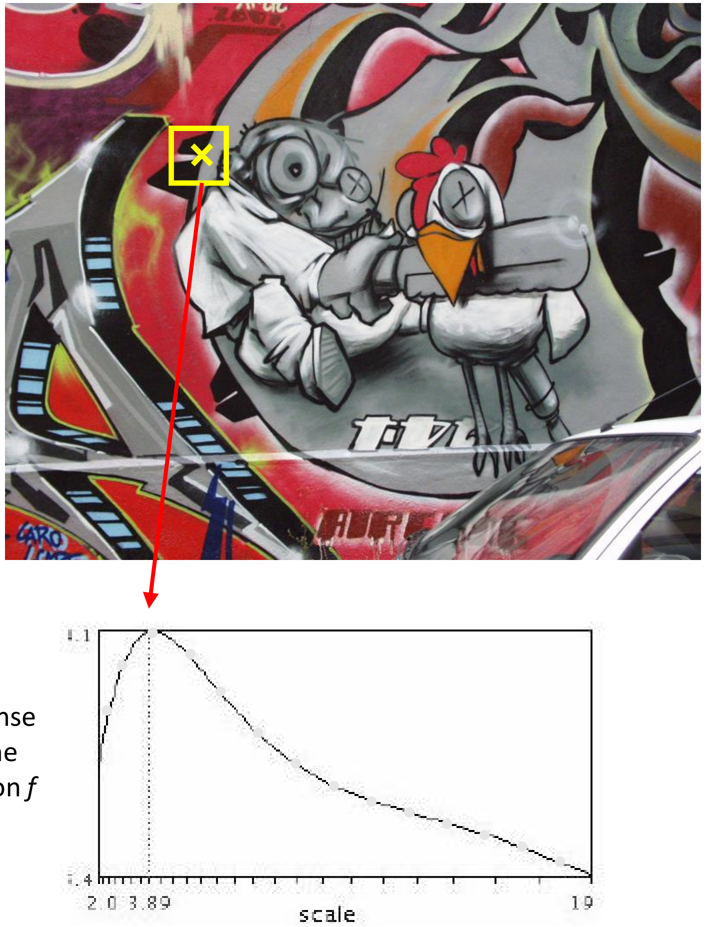
Response of some function f



$f(I_{i_1\ldots i_m}(x,\sigma))$ $f(I_{i_1\ldots i_m}(x',\sigma'))$

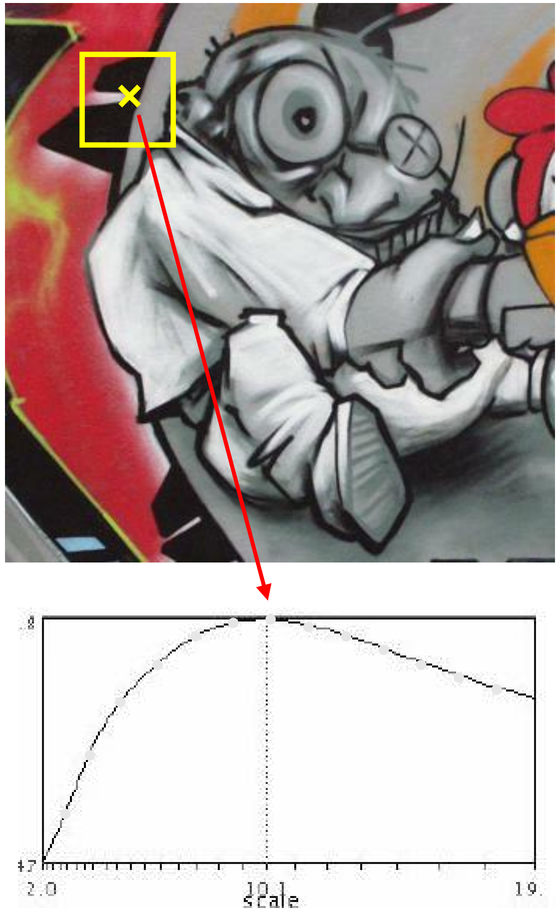
Automatic Scale Selection

- Function responses for increasing scale (scale signature)



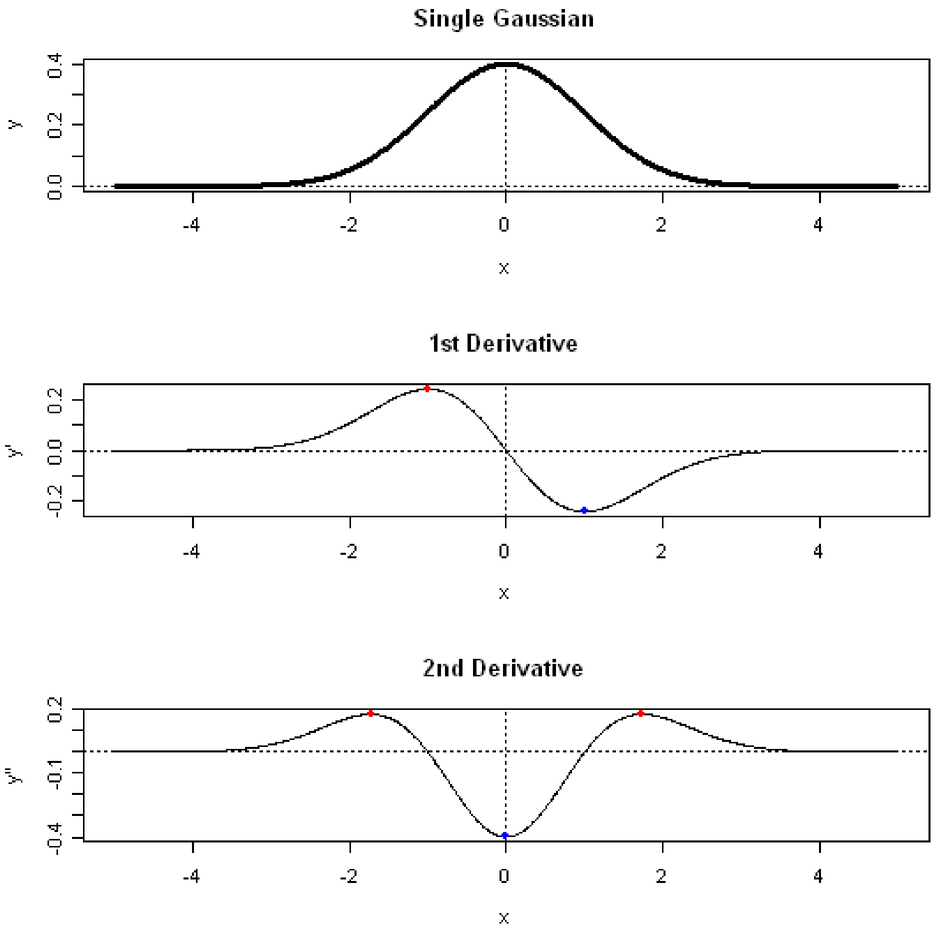
Response of some function *f*

$f(I_{i_1 \dots i_m}(x,\sigma))$

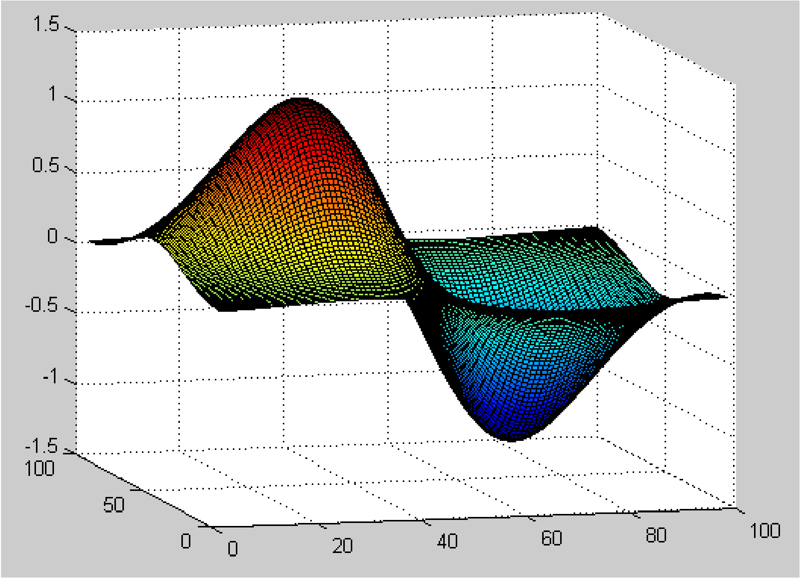


$f(I_{i_1 \dots i_m}(x',\sigma'))$

What Is A Useful Signature Function f ?



1st Derivative of Gaussian



(Laplacian of Gaussian)

Earl F. Glynn

CAP5415 Computer Vision

Yogesh S Rawat

yogesh@ucf.edu

HEC-241

Questions?

Features

Lecture 9

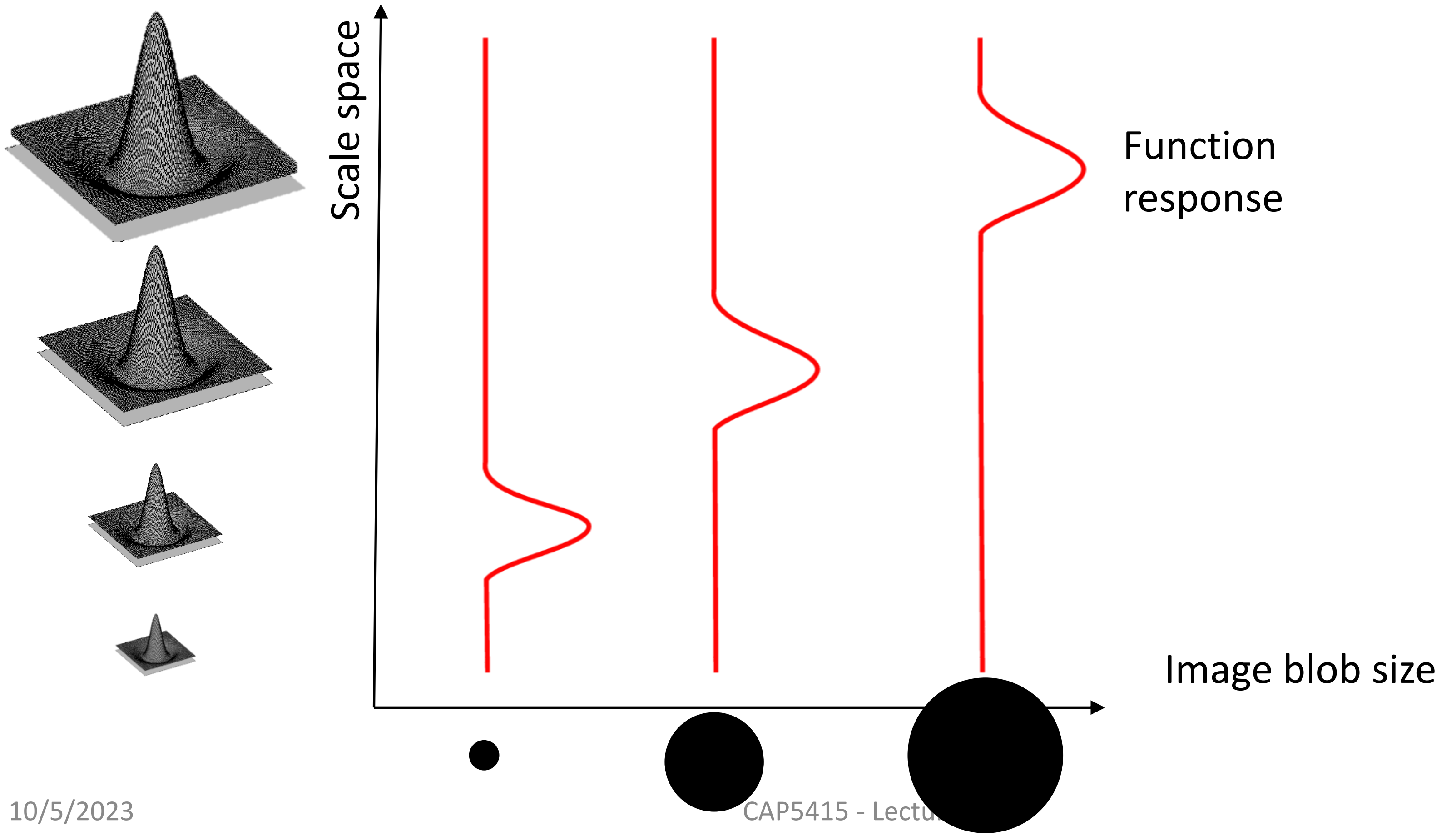
SIFT

Continued...

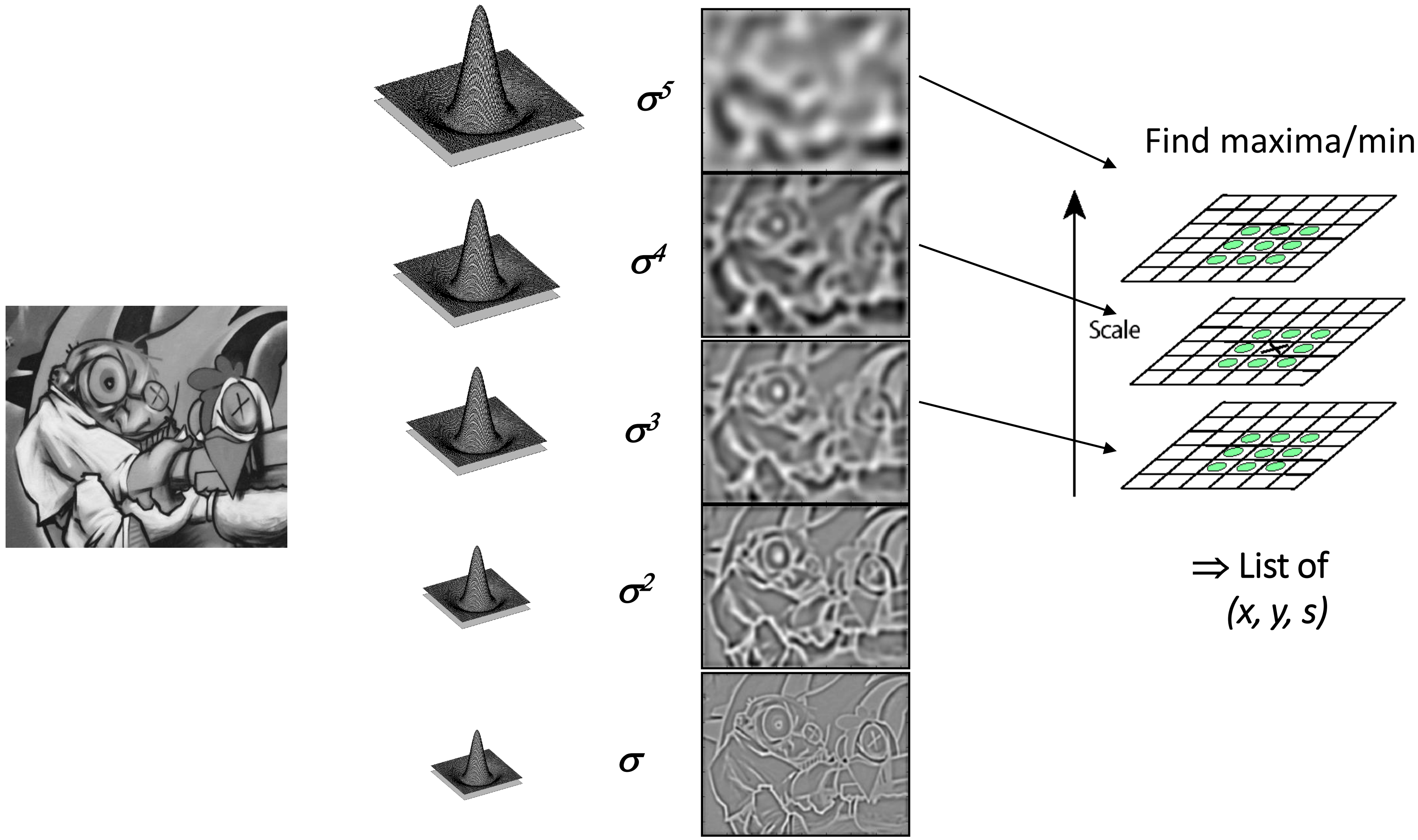
What Is A Useful Signature Function f ?

"Blob" detector is common for corners

- Laplacian (2nd derivative) of Gaussian (LoG)



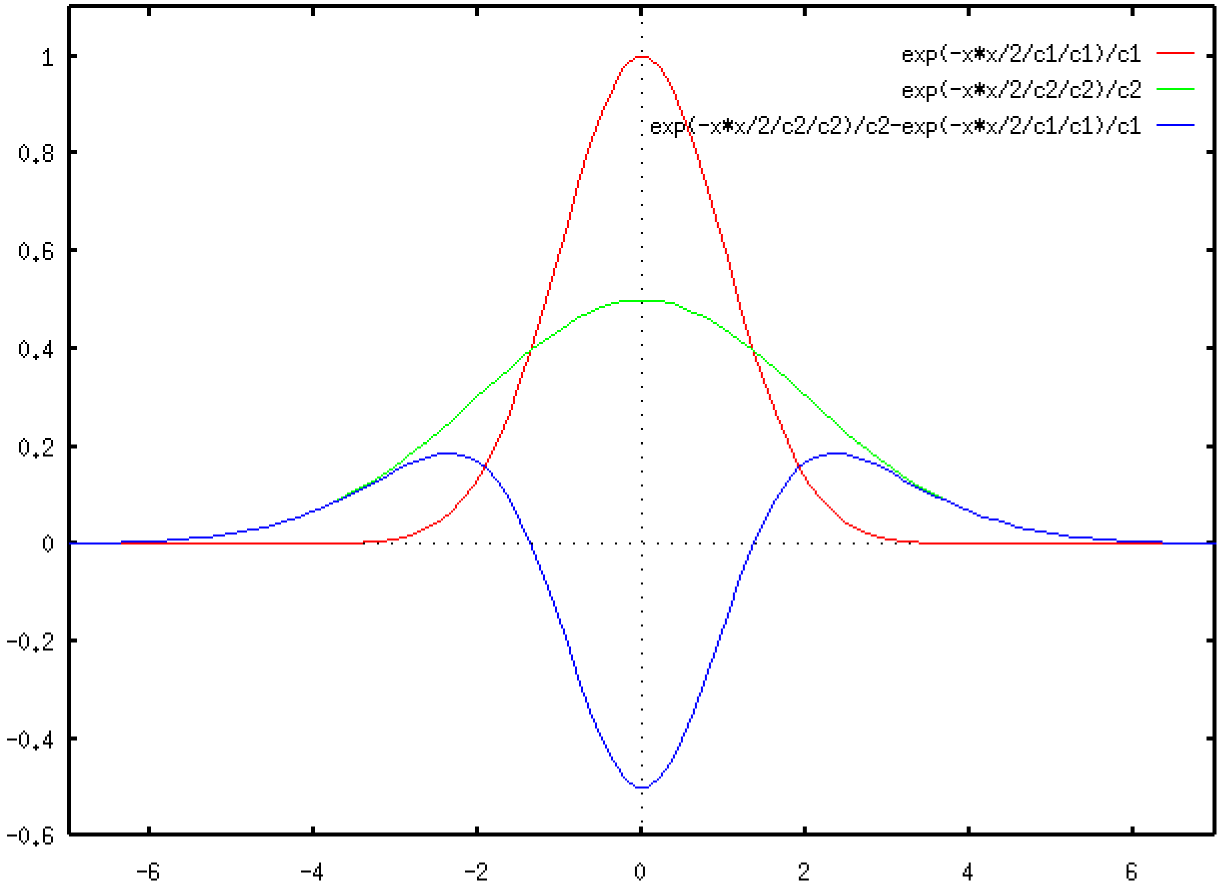
Find local maxima in position-scale space



$\Rightarrow$ List of
(x, y, s)

Alternative kernel

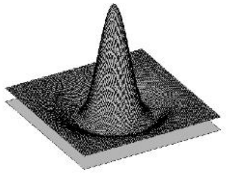
Approximate LoG with Difference-of-Gaussian (DoG).



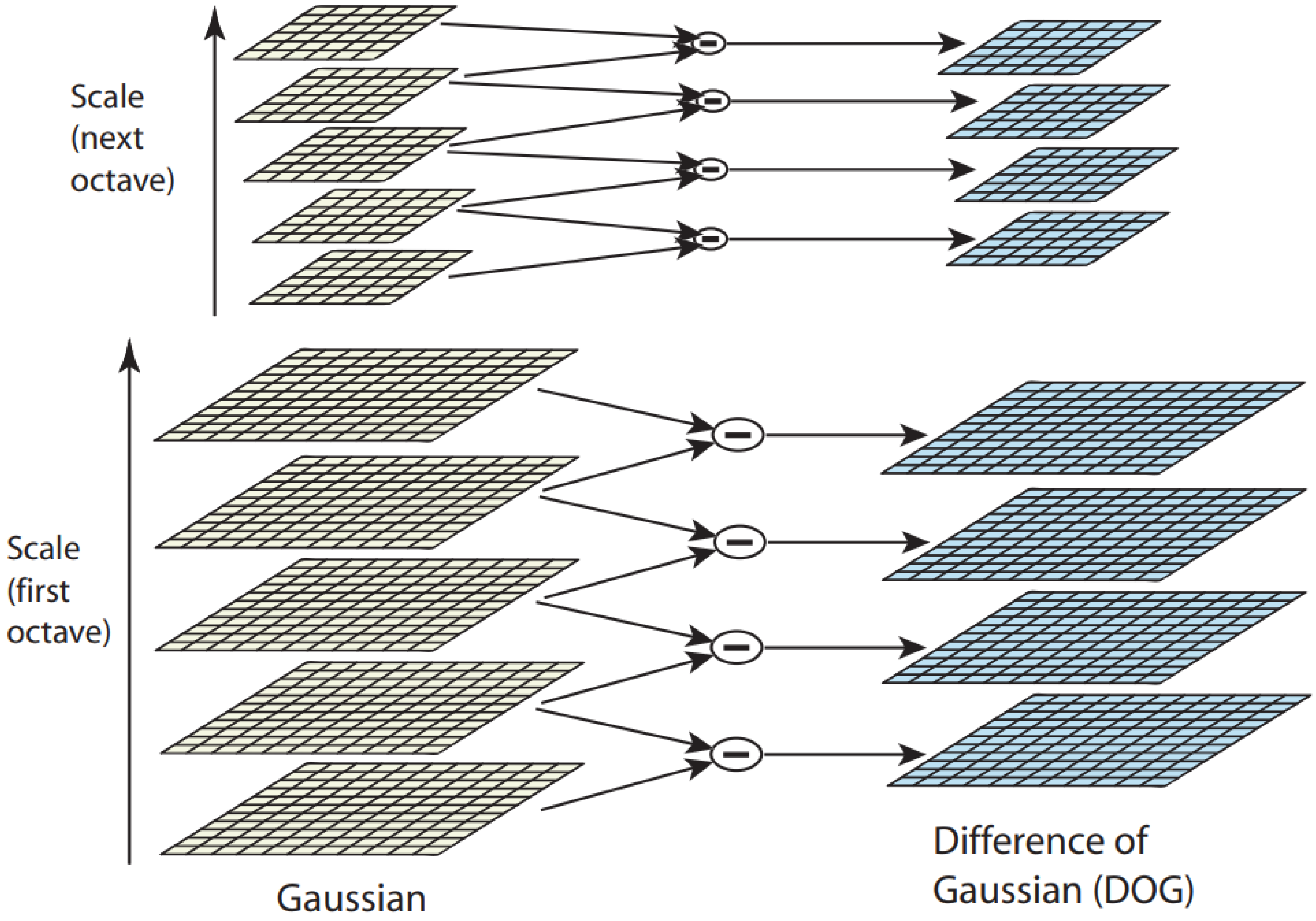
Alternative kernel

Approximate LoG with Difference-of-Gaussian (DoG).

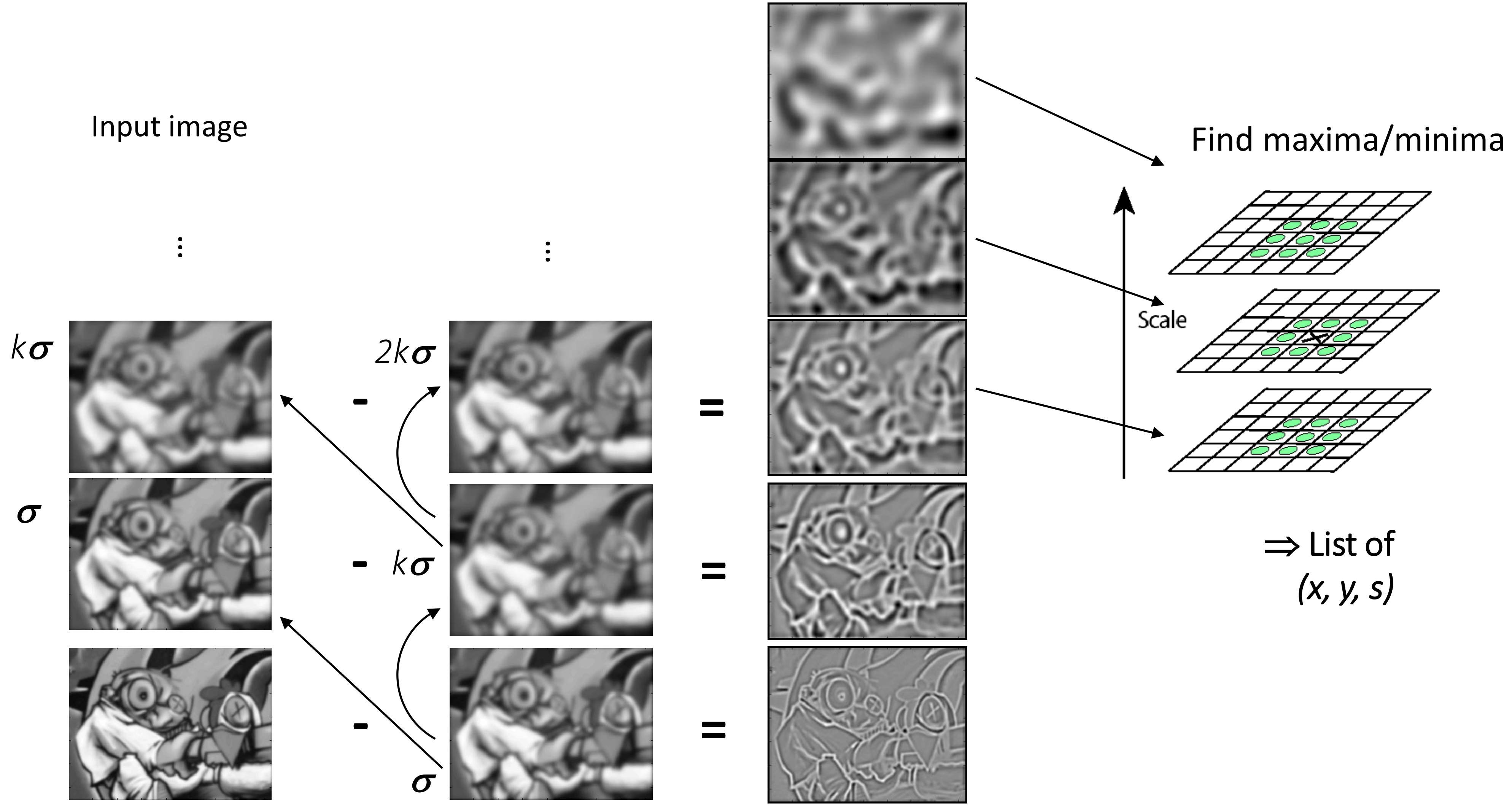
1. Blur image with σ Gaussian kernel
2. Blur image with $k\sigma$ Gaussian kernel
3. Subtract 2. from 1.



Scale-space

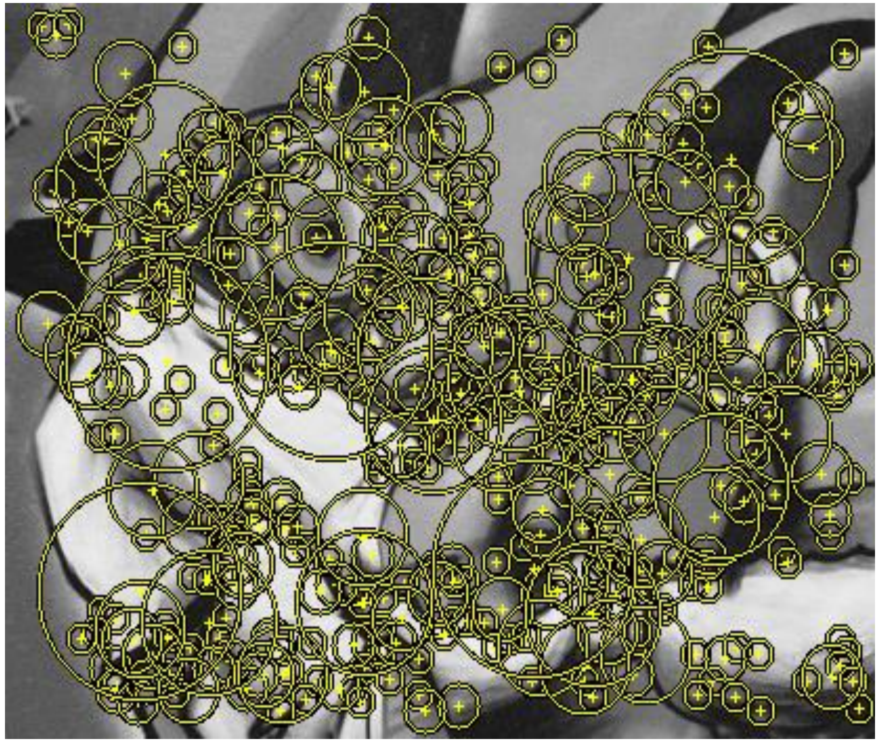

Scale
(next
octave)
Scale
(first
octave)
Gaussian
Difference of
Gaussian (DOG)

Find local maxima in position-scale space of DoG



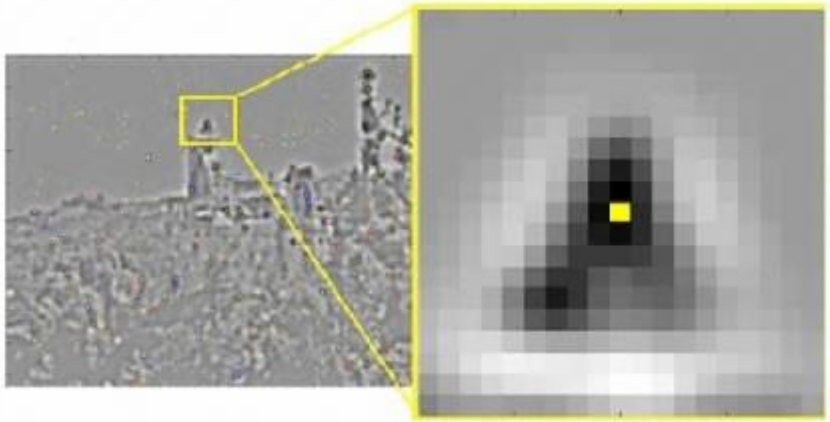
Results: Difference-of-Gaussian

- Larger circles = larger scale
- Descriptors with maximal scale response

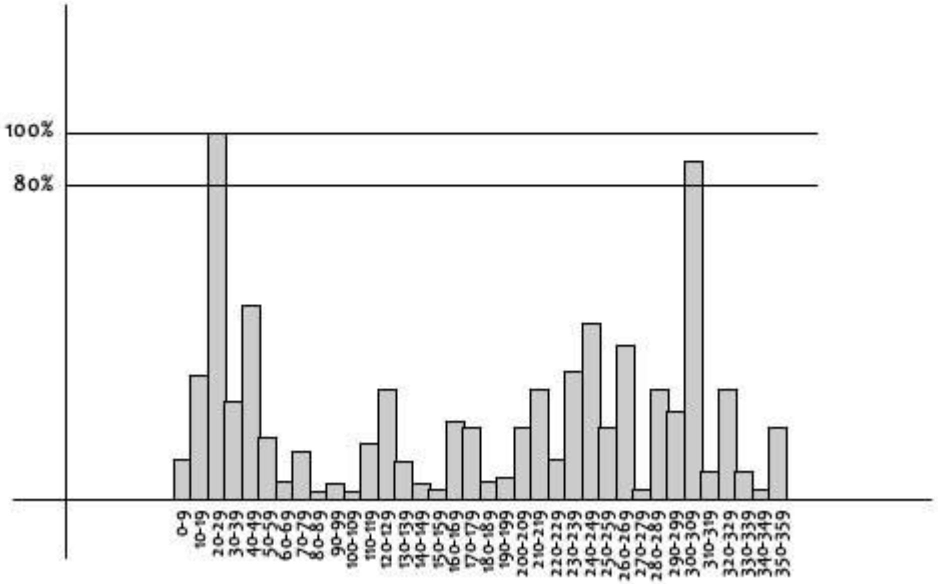


SIFT Orientation estimation

- Compute gradient orientation histogram
- Select dominant orientation θ

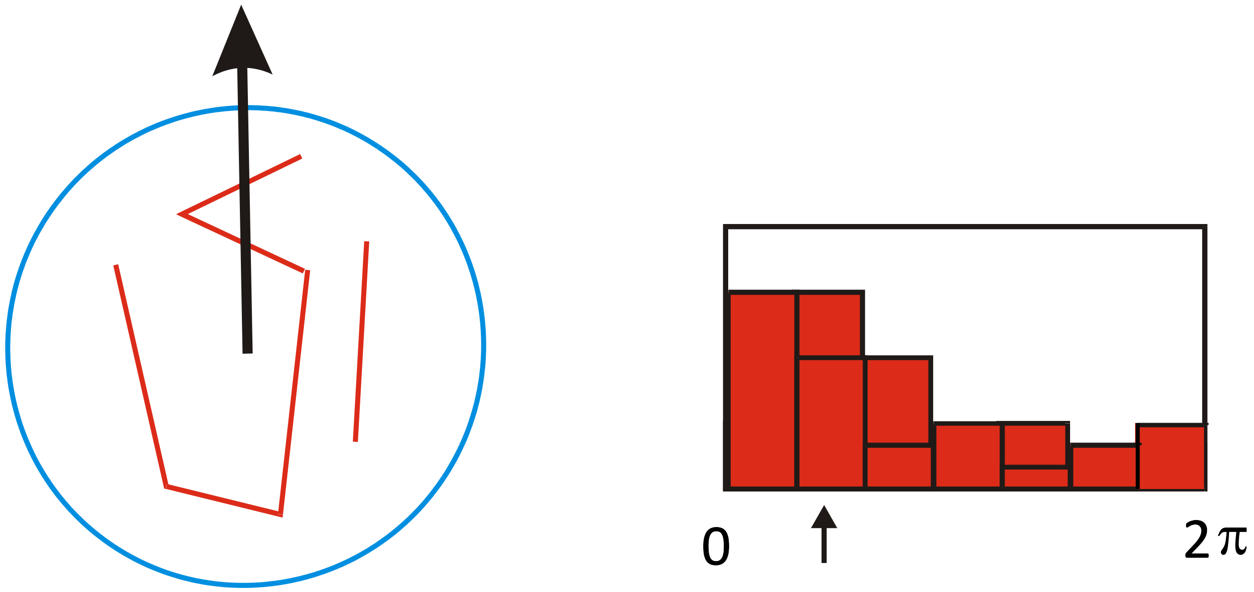


A keypoint



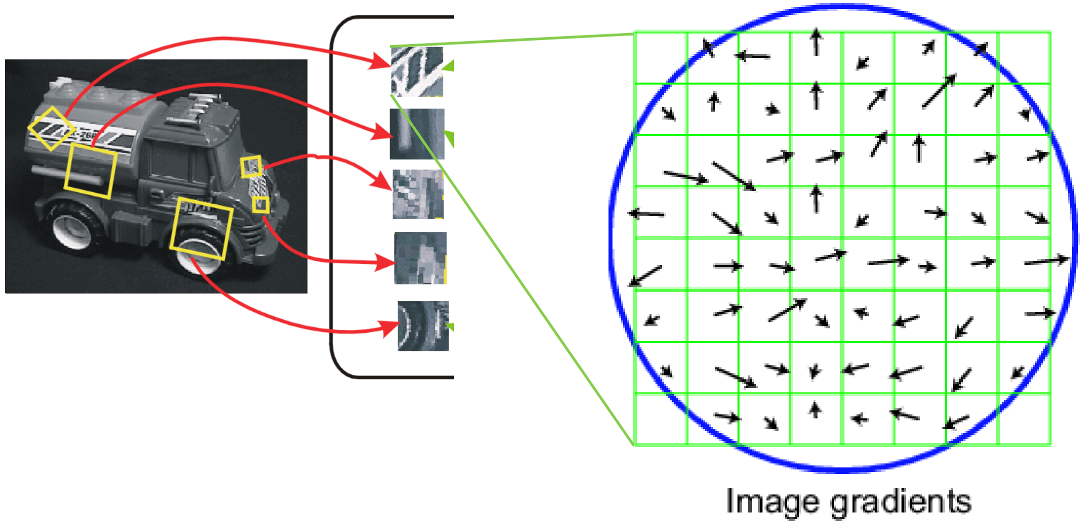
SIFT Orientation Normalization

- Compute gradient orientation histogram
- Select dominant orientation θ
- Normalize: rotate to fixed orientation



SIFT descriptor formation

- Compute on local 16 x 16 window around detection.
- Rotate and scale window according to discovered orientation *Θ* and scale *σ* (gain invariance).
- Compute gradients weighted by a Gaussian of variance half the window (for smooth falloff).



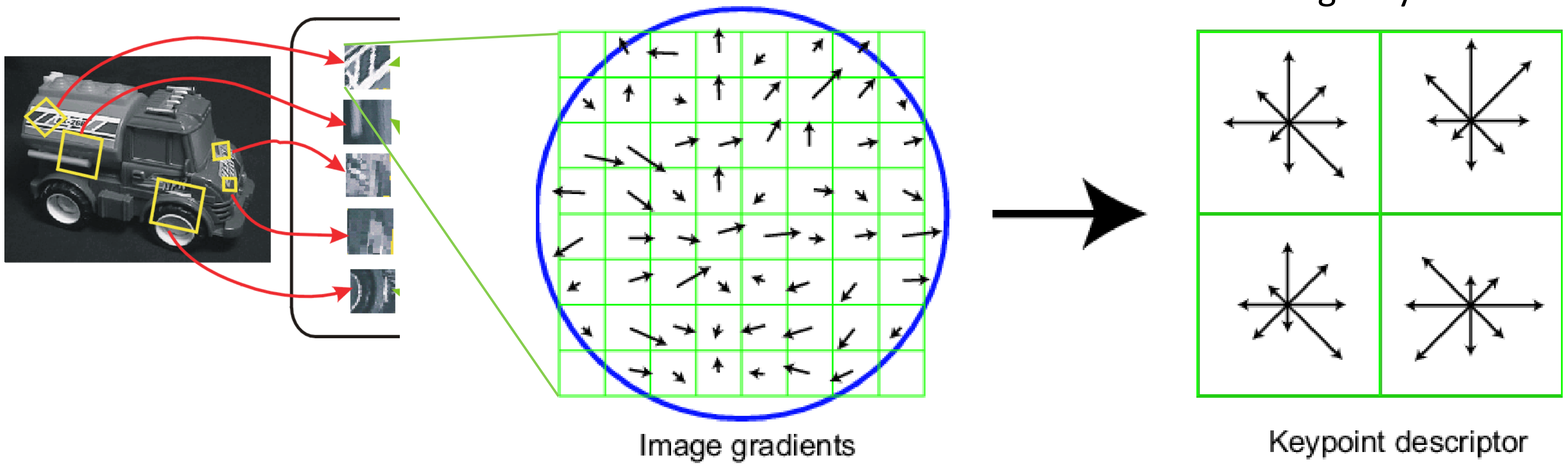
Actually 16x16, only showing 8x8

James Hays

SIFT descriptor formation

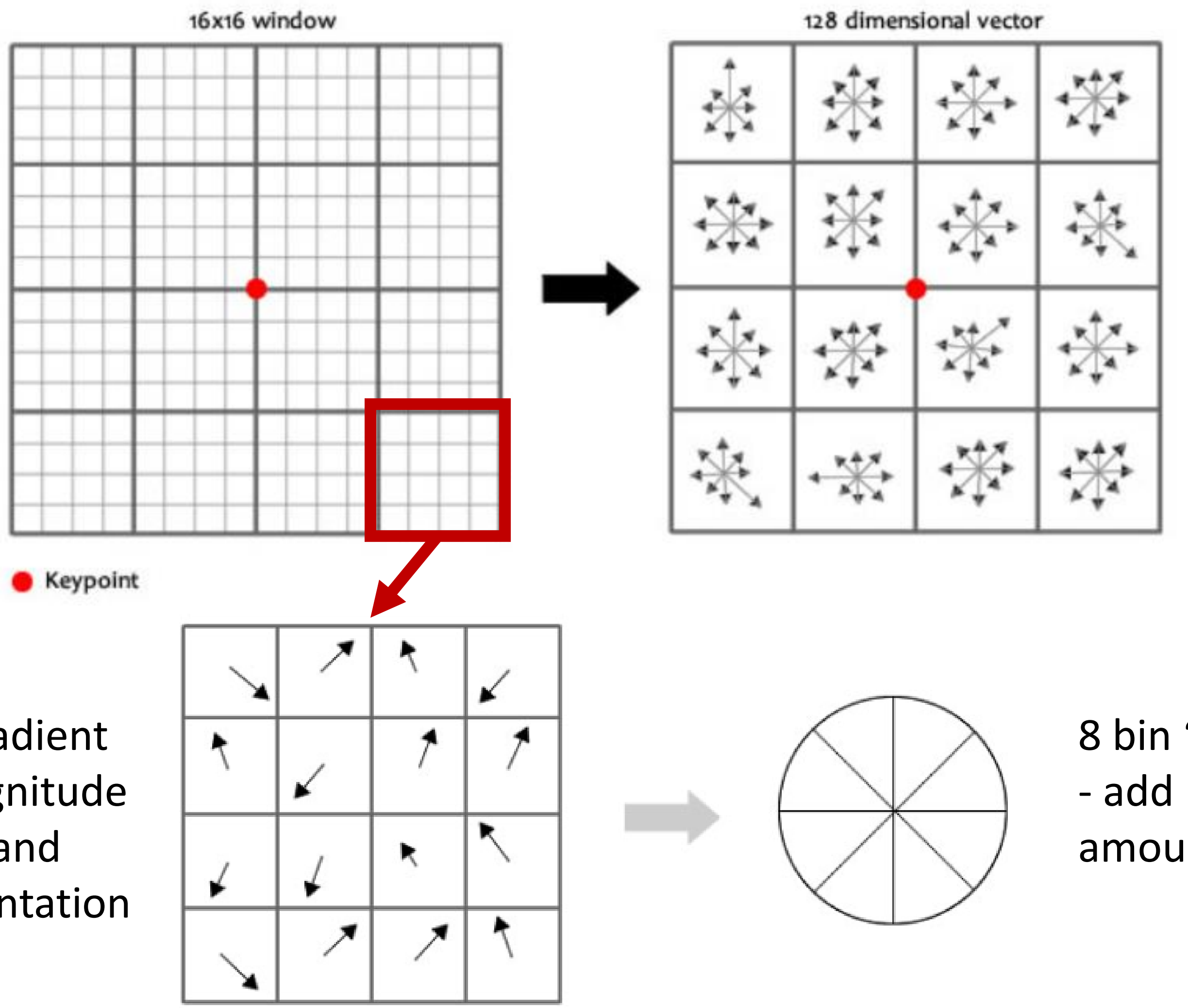
- 4x4 array of gradient orientation histograms weighted by gradient magnitude.
- Bin into 8 orientations x 4x4 array = 128 dimensions.

Showing only 2x2 here but is 4x4



James Hays

SIFT Descriptor Extraction



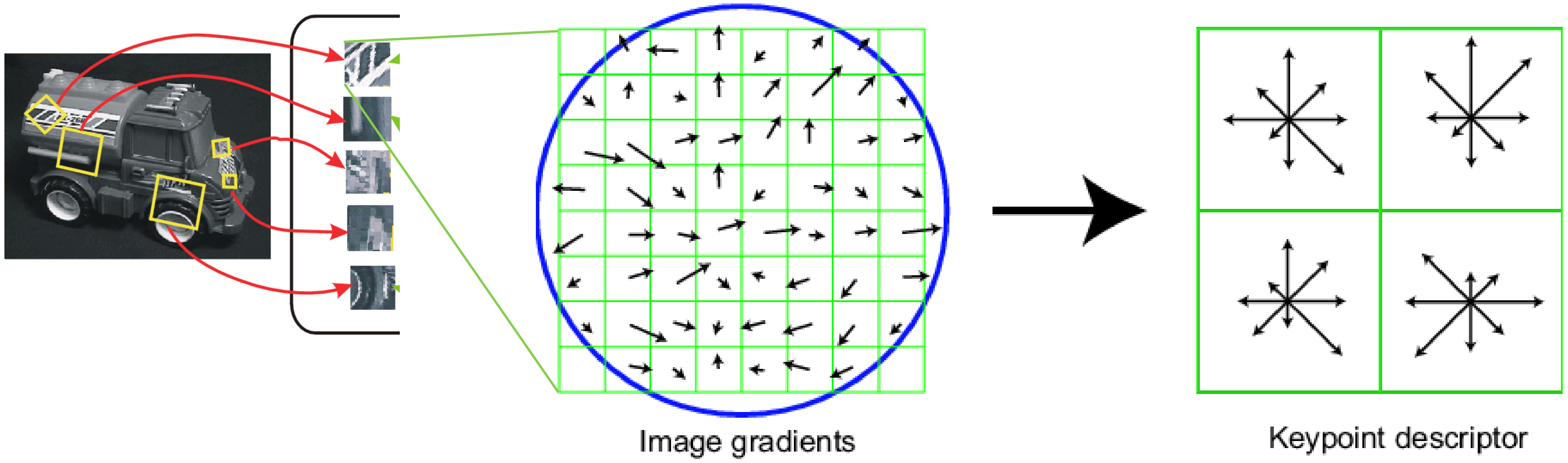
Gradient magnitude and orientation

8 bin 'histogram' - add magnitude amounts!

Utkarsh Sinha

Reduce effect of illumination

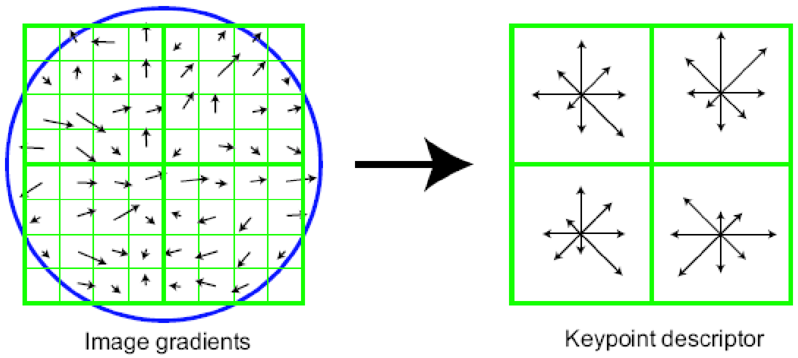
- 128-dim vector normalized to 1
- Threshold gradient magnitudes to avoid excessive influence of high gradients
  - After normalization, clamp gradients > 0.2
  - Renormalize



James Hays

Review: Local Descriptors

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



Questions?

Sources for this lecture include materials from works by Mubarak Shah, S. Seitz, James Tompkin and Ulas Bagci

Questions?

Sources for this lecture include materials from works by Mubarak Shah, S. Seitz, James Tompkin and Ulas Bagci