

Workshops in Creative Computing: Computer Vision Module



Lecture 2: Image Features

Wednesday, Feb 27, 2013

Parag K Mital

Assignment 1: Still need to solve Computer Vision

Vision is based on **inference**

Popular Uses of Feature Detection:

Structure from Motion

Photo-montage

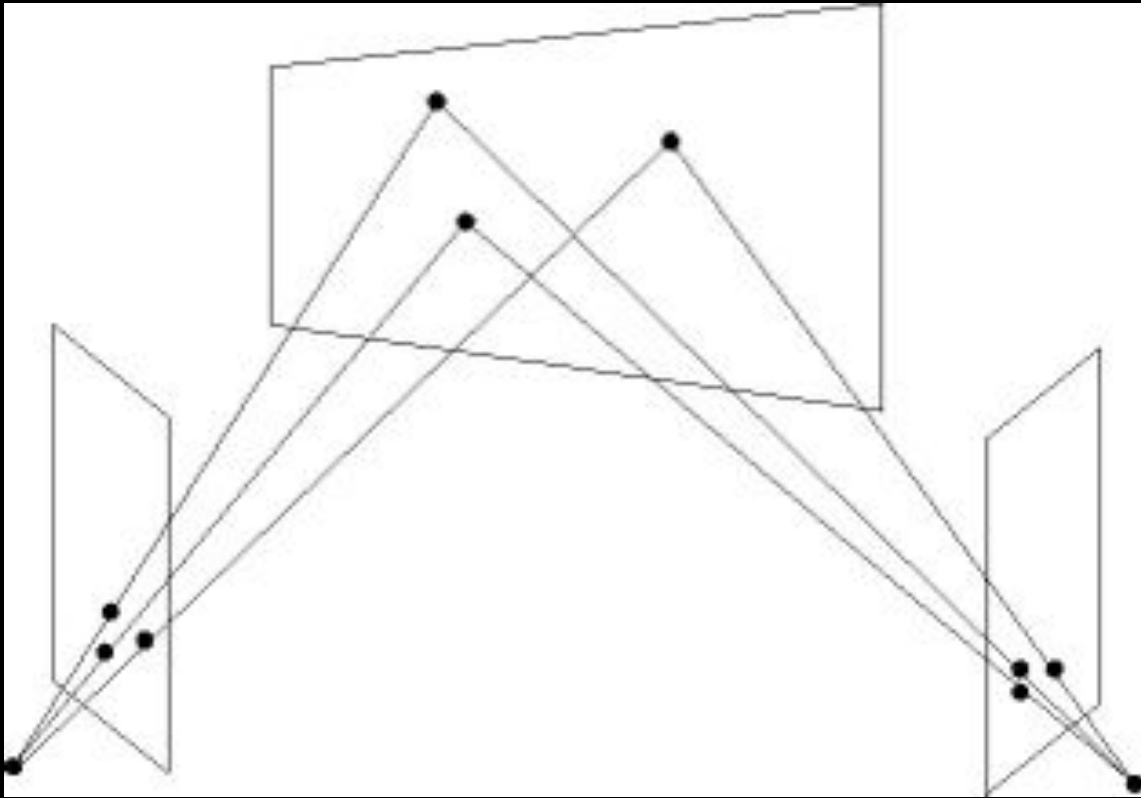
Panorama/Stitching/Mosaicing

Information Retrieval

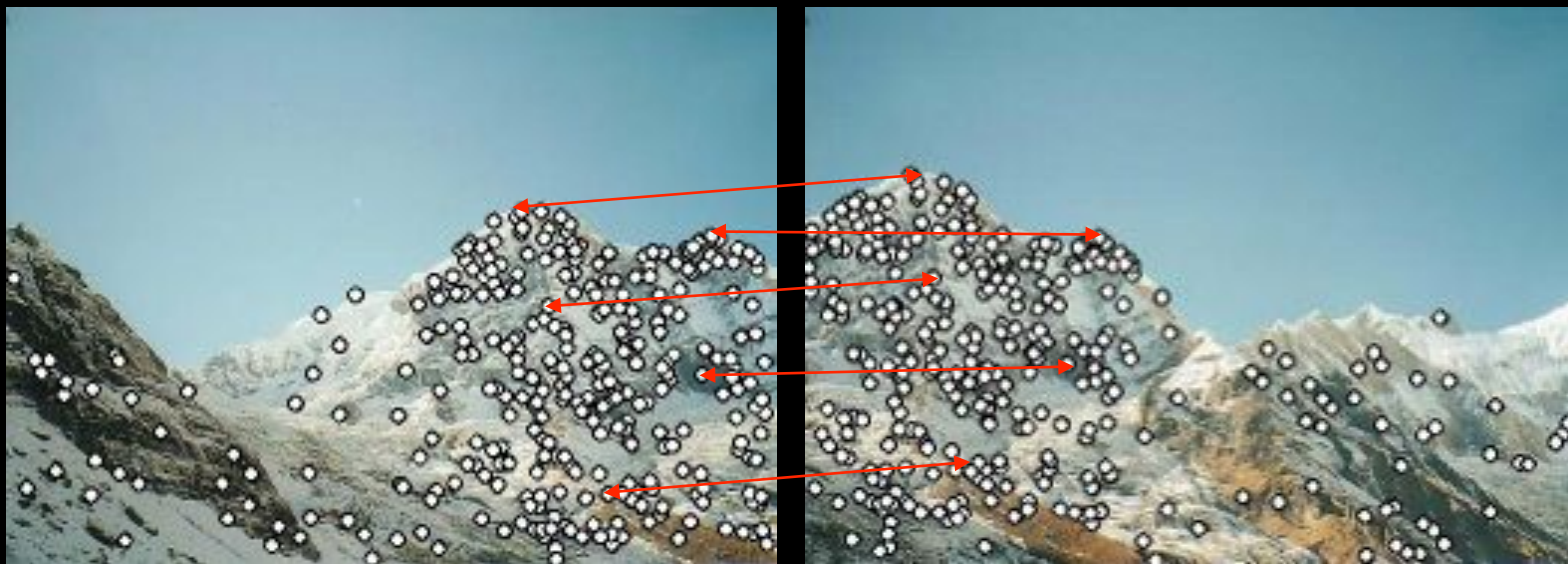
Object Detection

Scene Detection

Action Detection

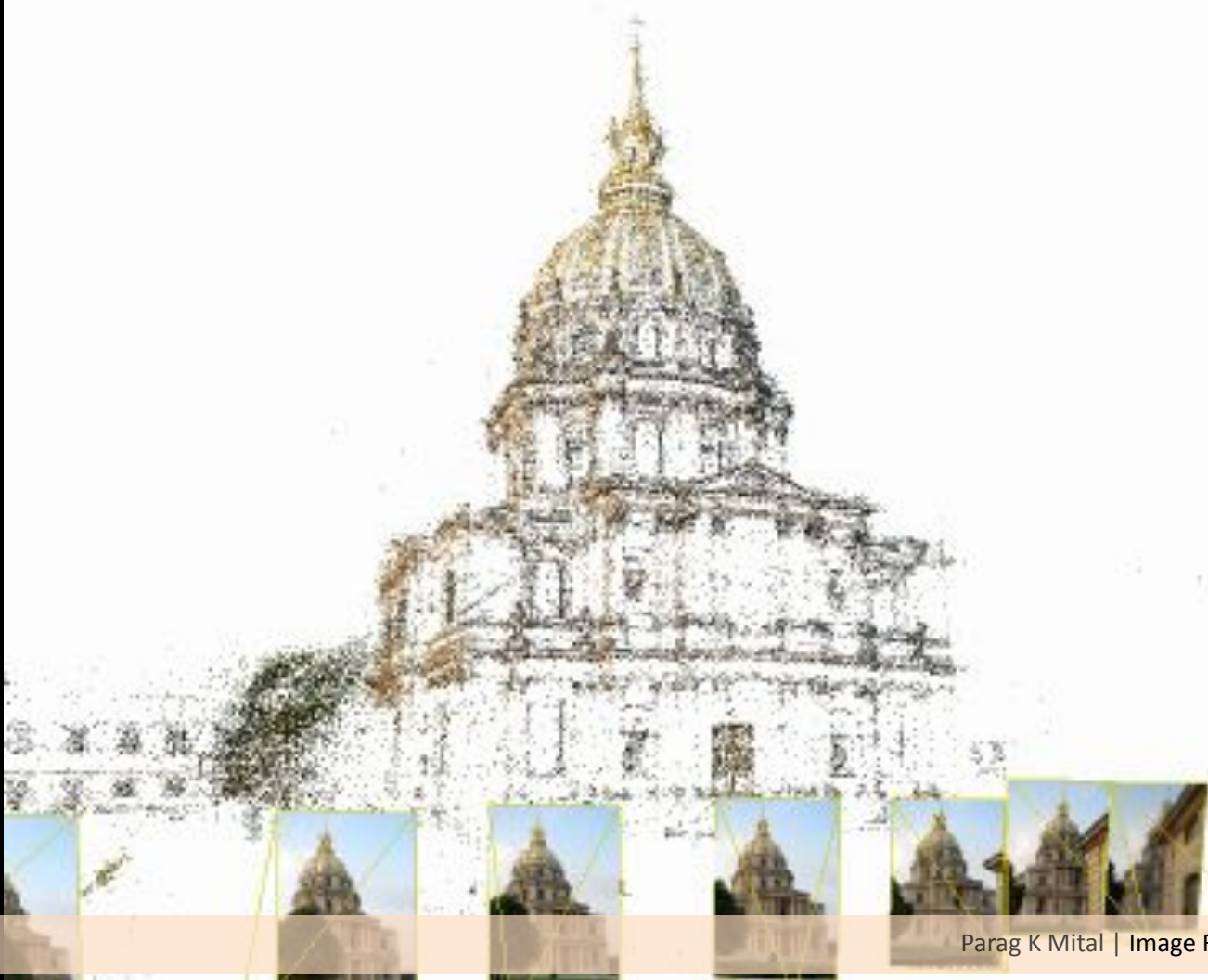












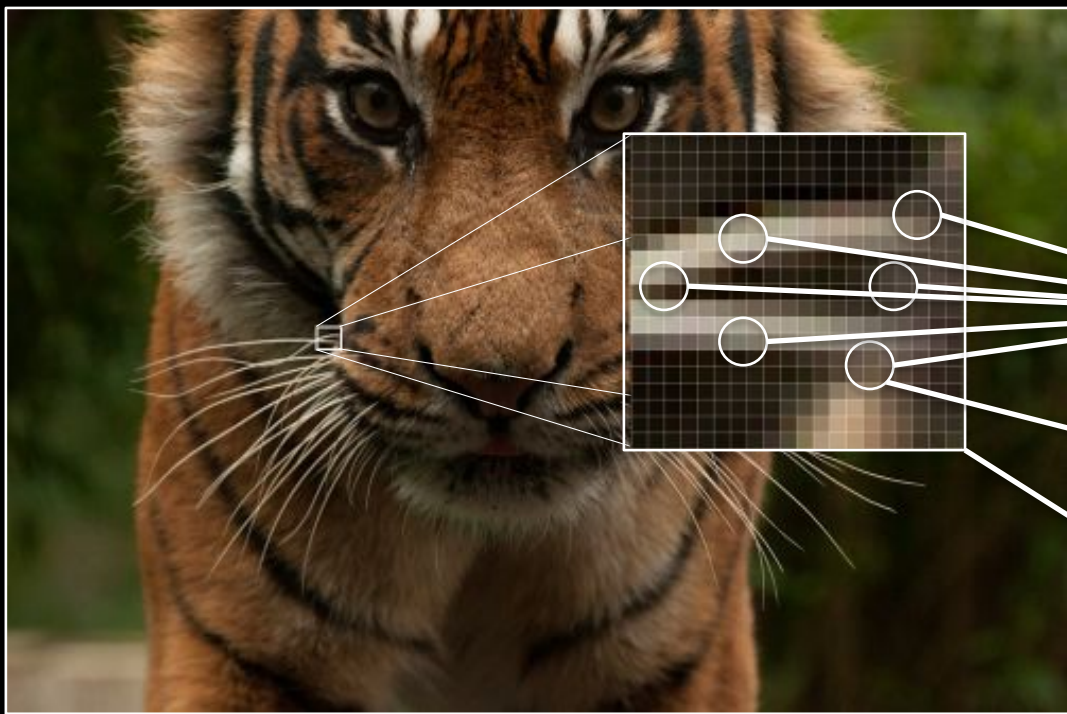


Google Image Search,
Google Goggles,
etc...

What makes us perceive objects in images?

Hypothesis: process images bottom-up

- Extract “features”
- Combine features with prior knowledge to classify objects in the image at a high-level



Semantic label =
High-level description



Grouping of Features =
Mid-level description

Single feature =
Low-level description

Pixels =
Low-level description

Generic Object Detection Workflow:

1. How do we **detect** features?
2. How do we **describe** features?
3. How do we **match** features?

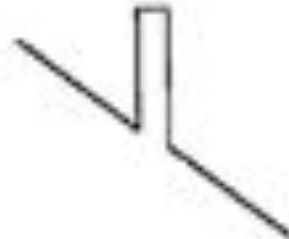
Pixels	Luminance; Color-spaces; Depth; Heat
Edges/Lists	Sobel; Canny; Hysteresis; Connected Components; Shape Models
Feature Points	SIFT; SURF; Harris Corners; HOG; FAST
Blobs/Regions	Mean-Shift; MSER; Watershed; Graph-Cuts; Background Subtraction; Appearance Models
Maps	Geodesics; Topography; Density



Step Edges



Roof Edge



Line Edges

Edges are where change occurs

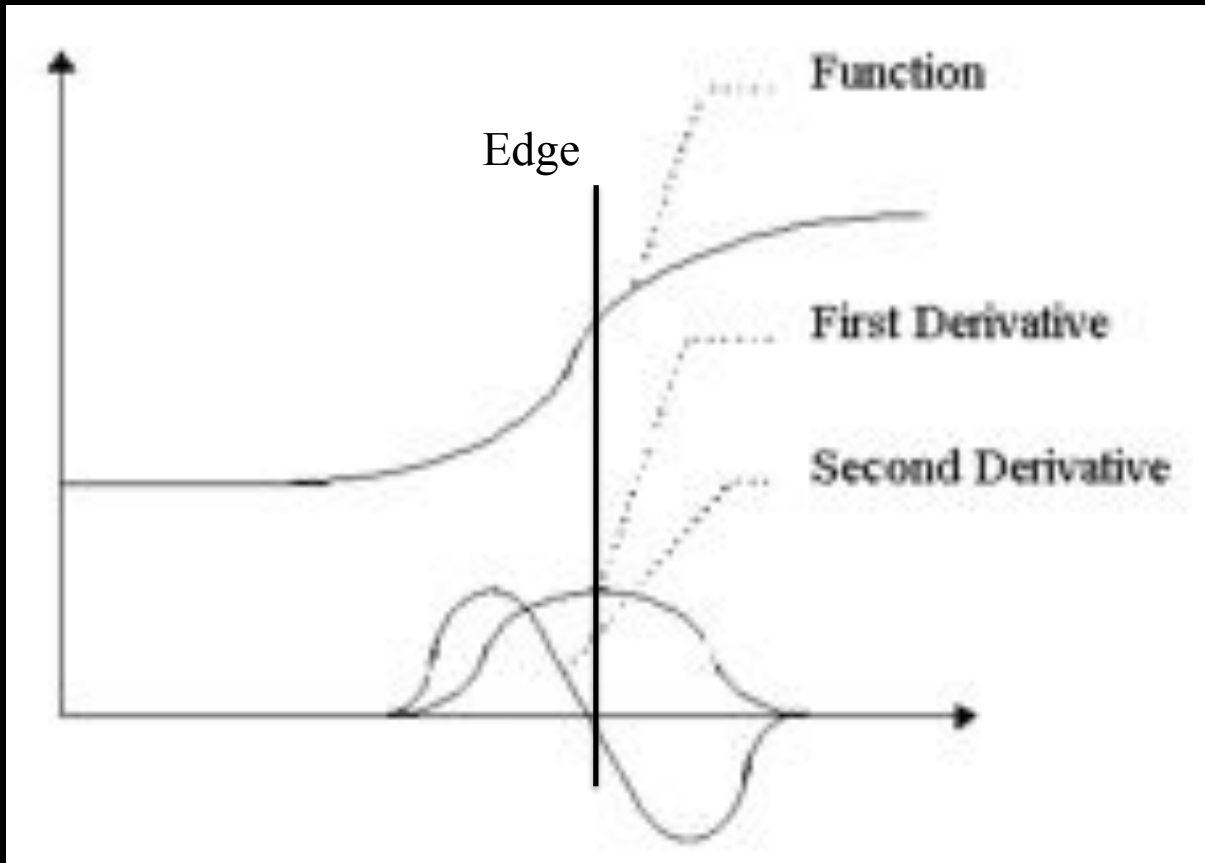
Images can be thought of as functions:

Pixel at location x :

$$P(x)$$

Then we can create a function f , which describes the intensity of pixel x :

$$f(x)$$



Derivative

$$\frac{df}{dx} = \lim_{h \rightarrow 0} \frac{f(x+h) - f(x)}{h}$$

Gradient

$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right]$$

Images are Discrete Functions

$$\frac{\partial f}{\partial x}[x, y] \approx f[x + 1, y] - f[x, y]$$

Sobel: Convolution Operators

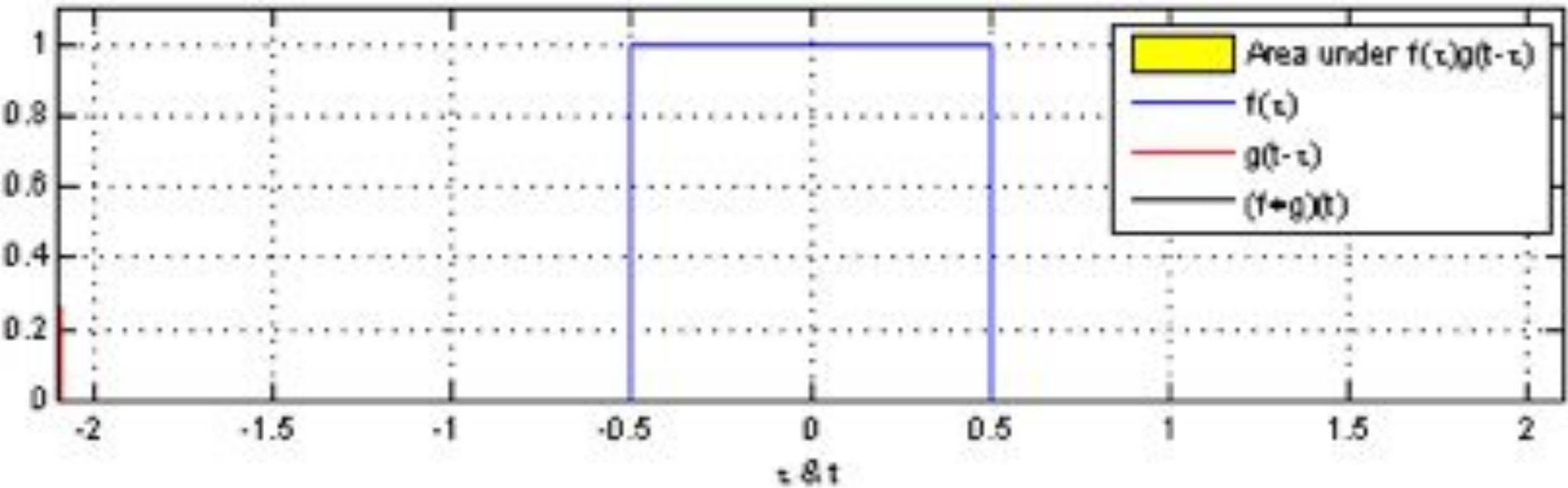
$$\frac{1}{8} \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

S_x

$$\frac{1}{8} \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

S_y

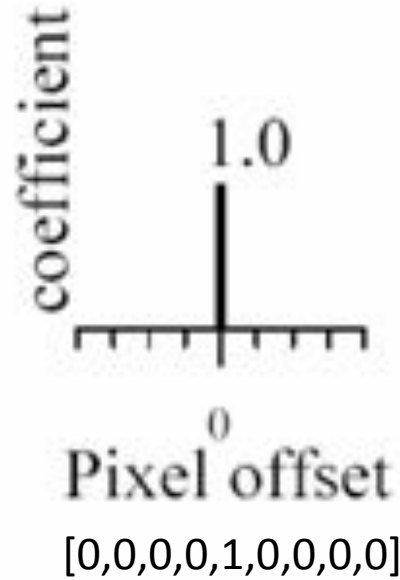
Convolution



Convolution



original

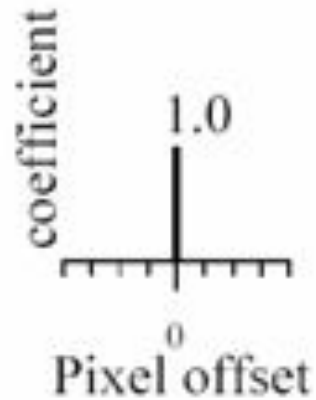


?

Convolution



original



$[0,0,0,0,1,0,0,0,0]$

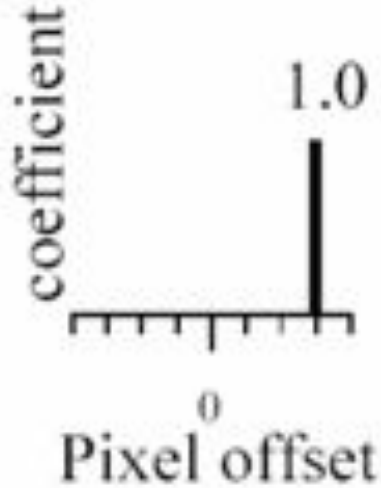


Filtered
(no change)

Convolution



original



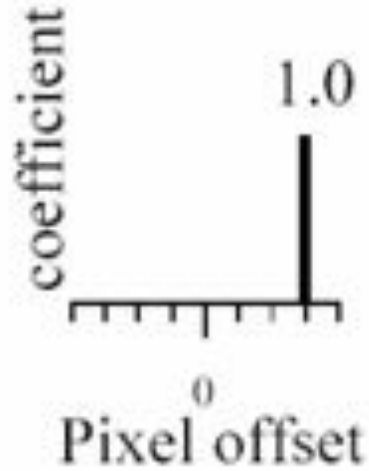
[0,0,0,0,0,0,0,1,0]

?

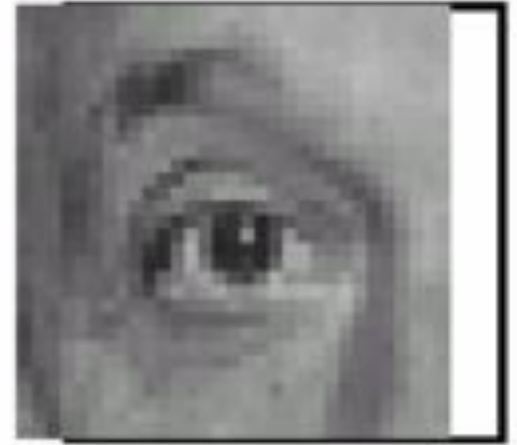
Convolution



original



[0,0,0,0,0,0,0,1,0]

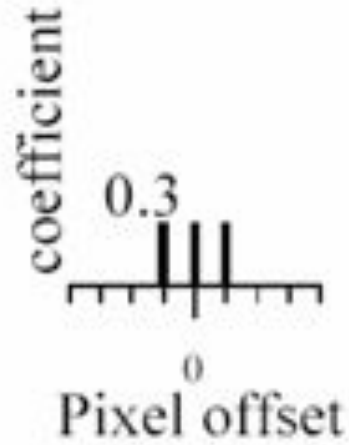


shifted

Convolution



original



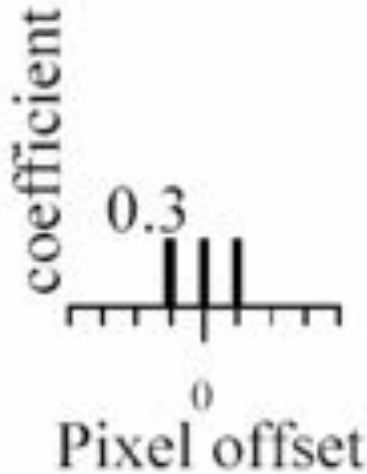
[0,0,0,0.333,0.333,0.333,0,0,0]

?

Convolution



original



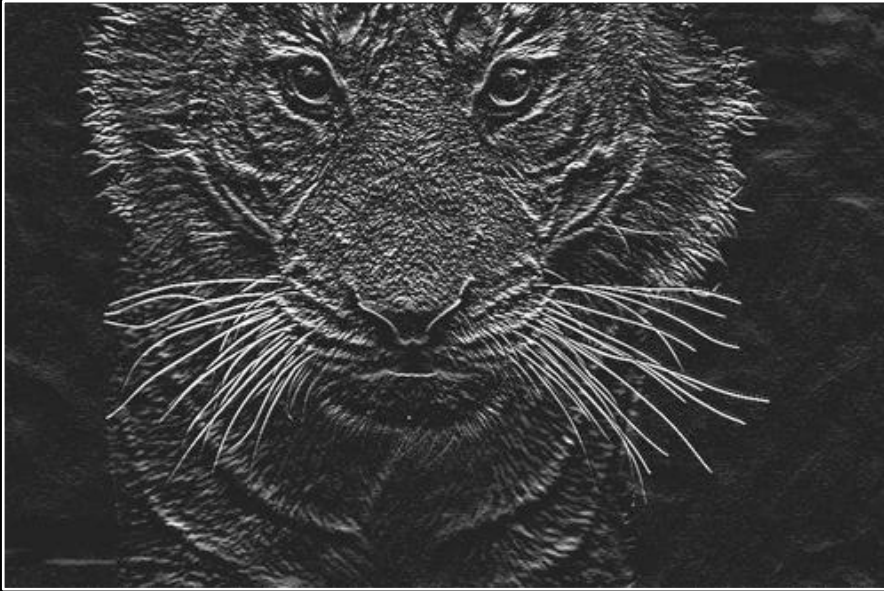
[0,0,0,0.333,0.333,0.333,0,0,0]



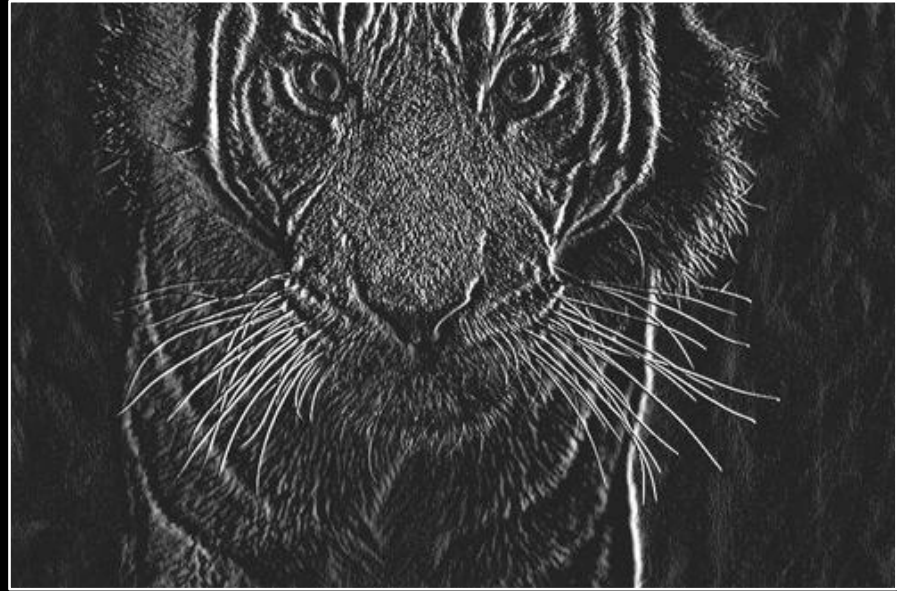
Blurred (filter applied in both dimensions).



$$\frac{1}{8} \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} s_y$$

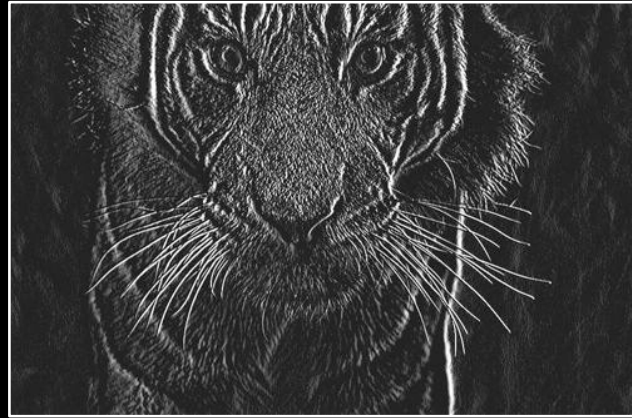


$$\frac{1}{8} \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} s_x$$





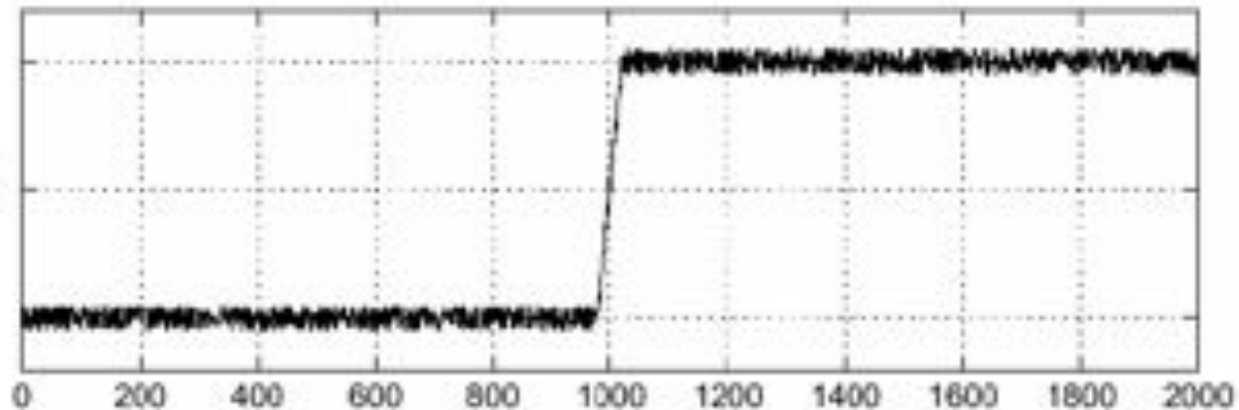
+



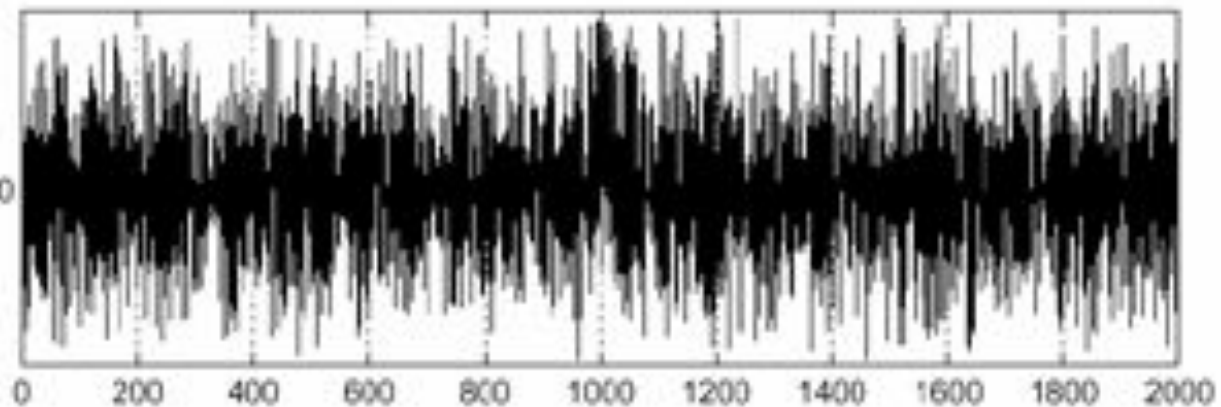
=



$f(x)$

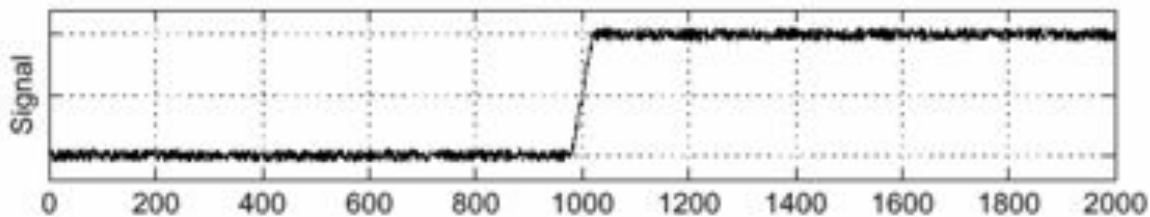


$\frac{d}{dx} f(x)$

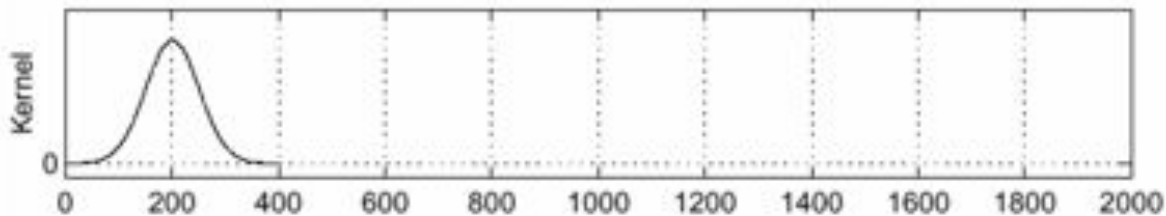


Sigma = 50

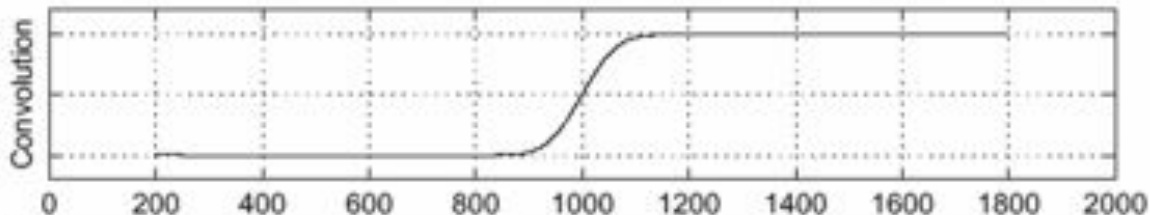
f



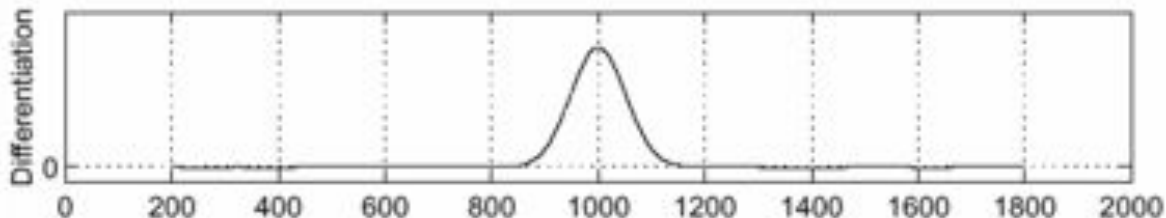
h

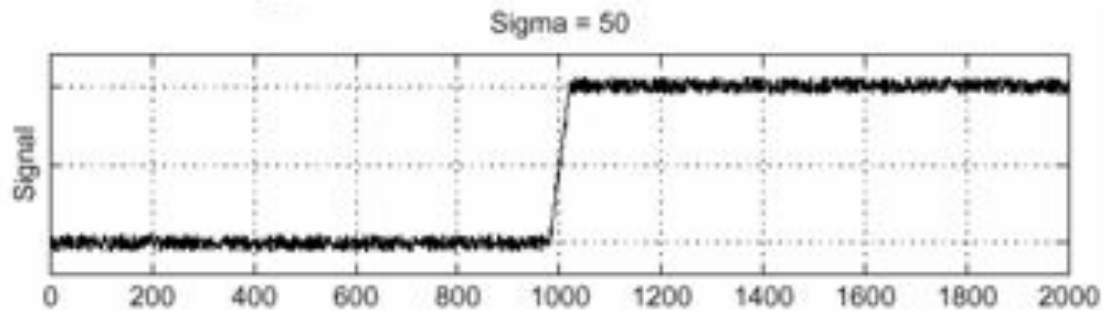
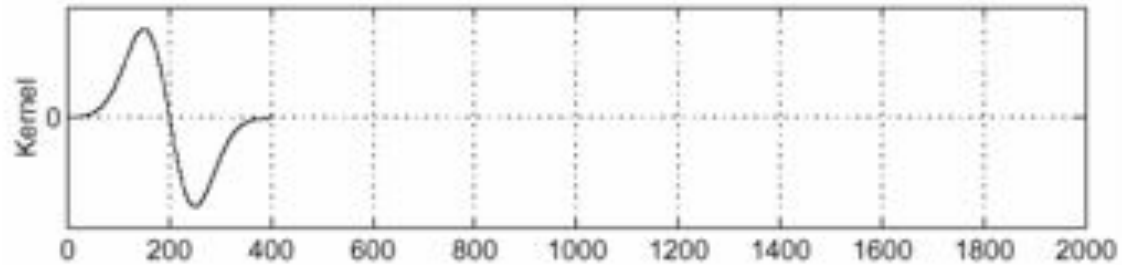
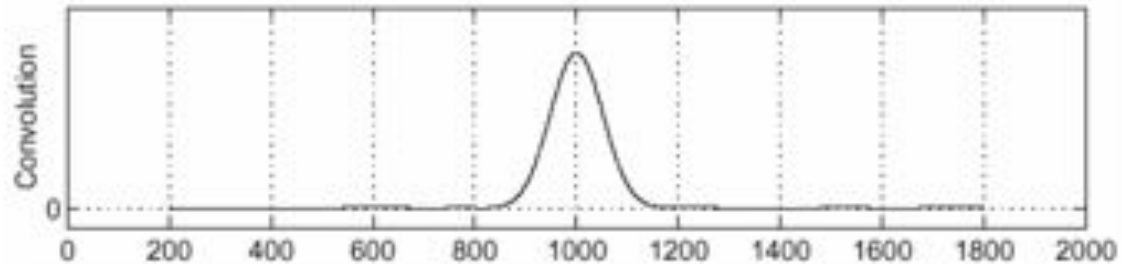


$h \star f$

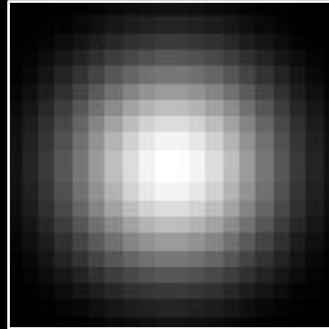


$\frac{\partial}{\partial x}(h \star f)$



f  $\frac{\partial}{\partial x} h$  $(\frac{\partial}{\partial x} h) \star f$ 

Gaussian Kernel

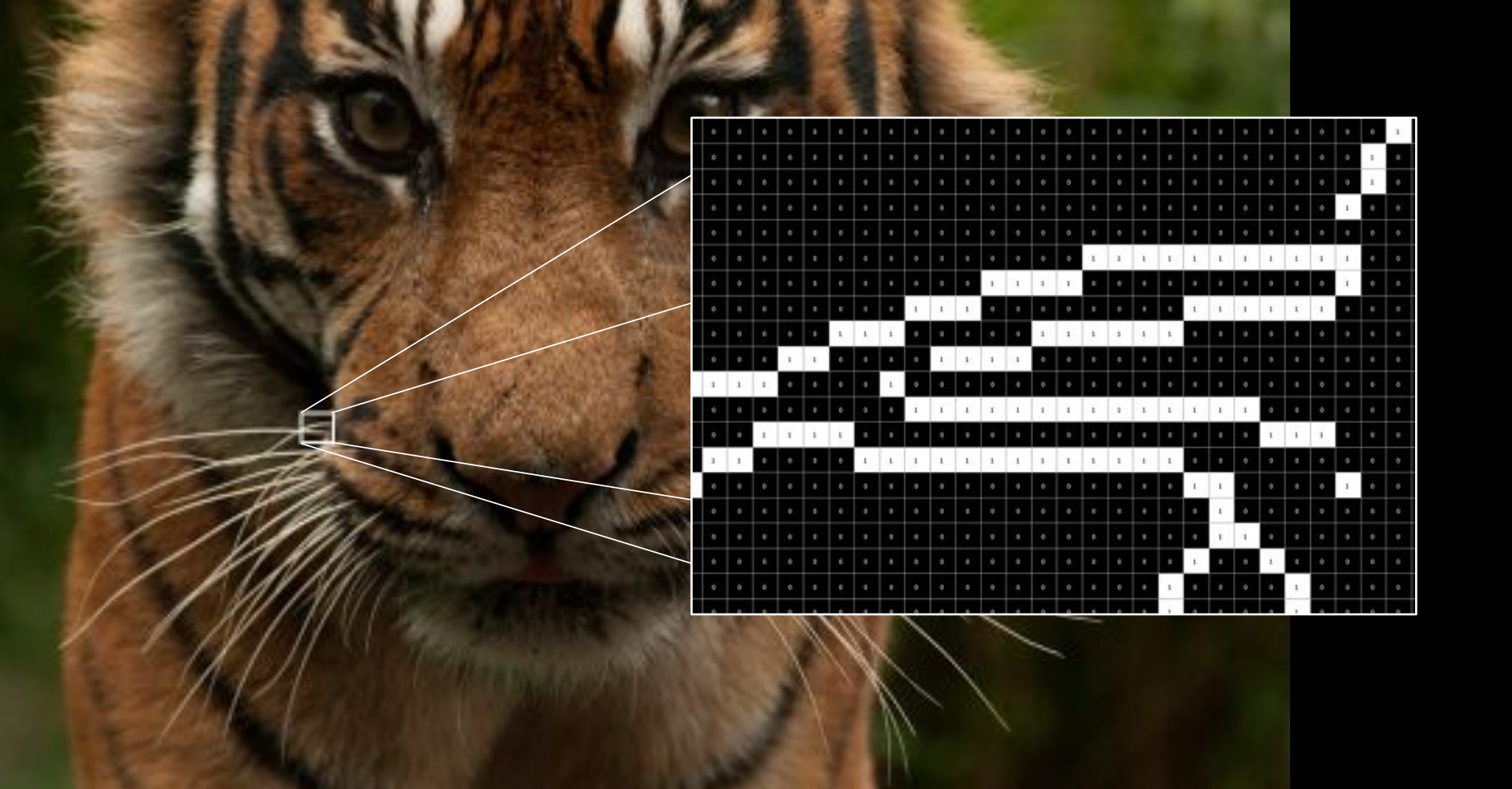


At what **scale** are our edges defined?









What kind of **invariance** does our algorithm have?

Luminance?

Color?

Translation?

Rotation?

Scale?

Skew? (Perspective?)

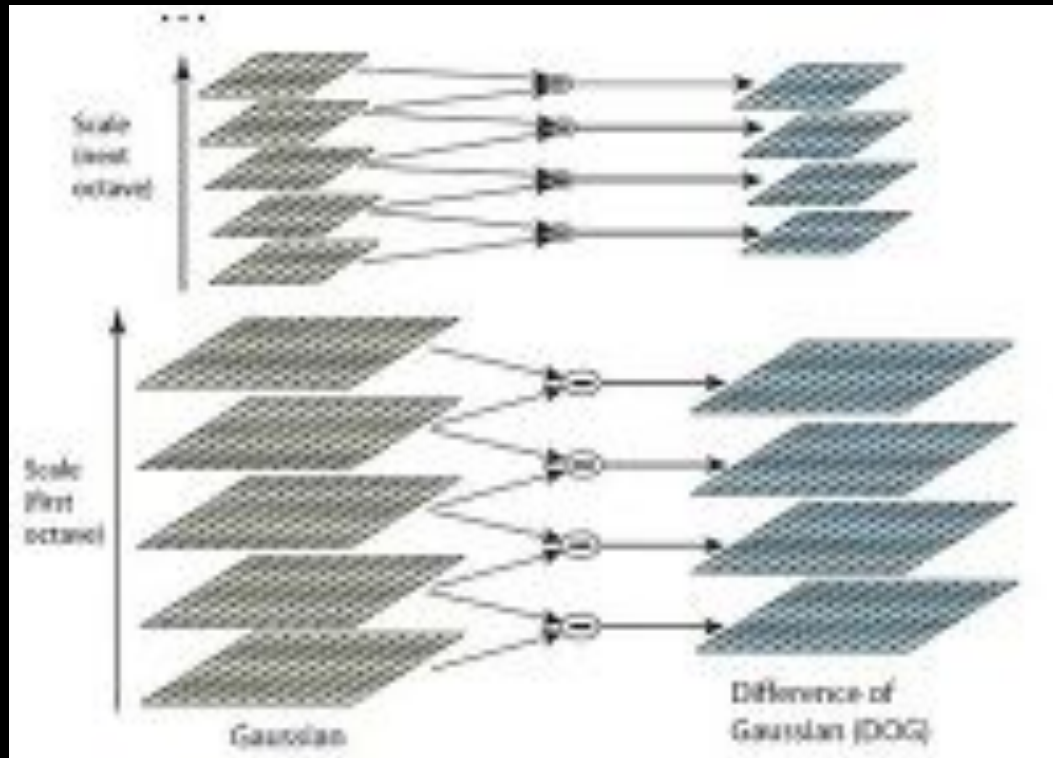
Rather than describe every pixel of an image,
we need to find the **keypoints**

Invariance to: **luminance, color, rotation,
translation, scale, skew...**

Should be fast to detect, and cheap to store!

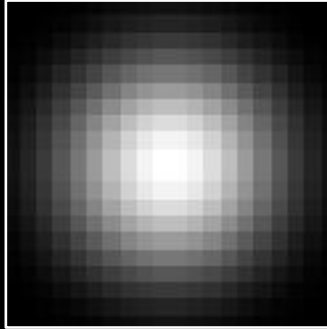
Scale Invariant Feature Transform (SIFT)

- Generate a Difference of Gaussian(DoG) or a laplacian pyramid
- Extrema detection from the DoG pyramid which is the local maxima and minima, the point found is an extrema
- Eliminate low contrast or poorly localized points, what remains are the keypoints
- Assign an orientation to the points based on the image properties
- Compute and generate keypoint descriptors





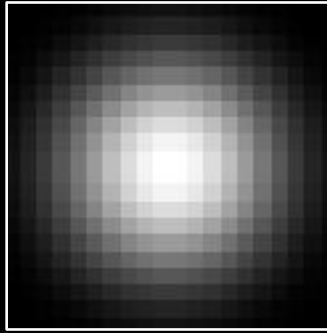
*



=



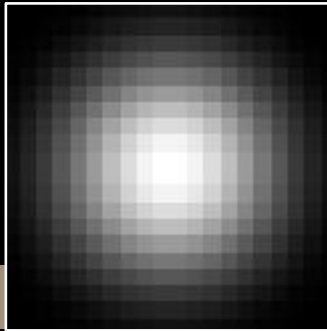
*



=

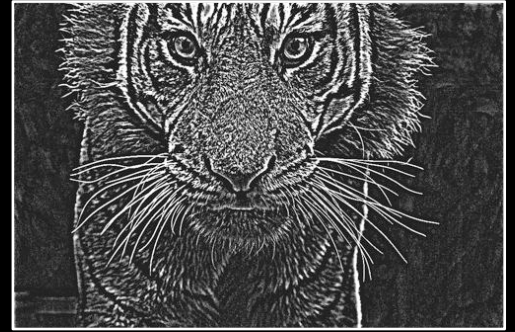
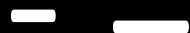
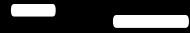
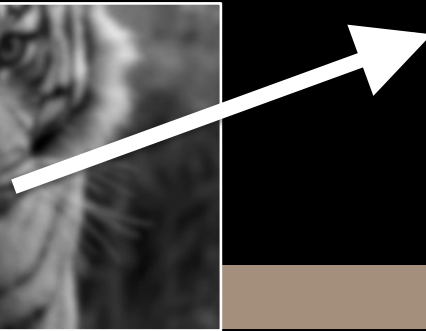
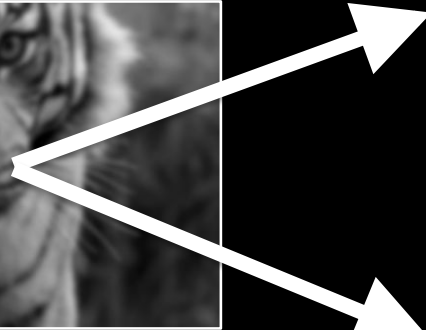
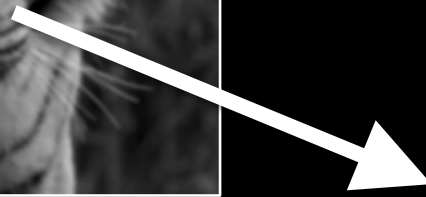


*



=





Popular Feature Detectors:

SIFT: Scale Invariant Feature Transform

SURF: Speeded-Up Robust Features

Harris: Corner detector

FAST: It's a really fast Corner detector

STAR: Center Surround Extractor (CenSurE)

MSER: Maximally Stable Extremal Regions

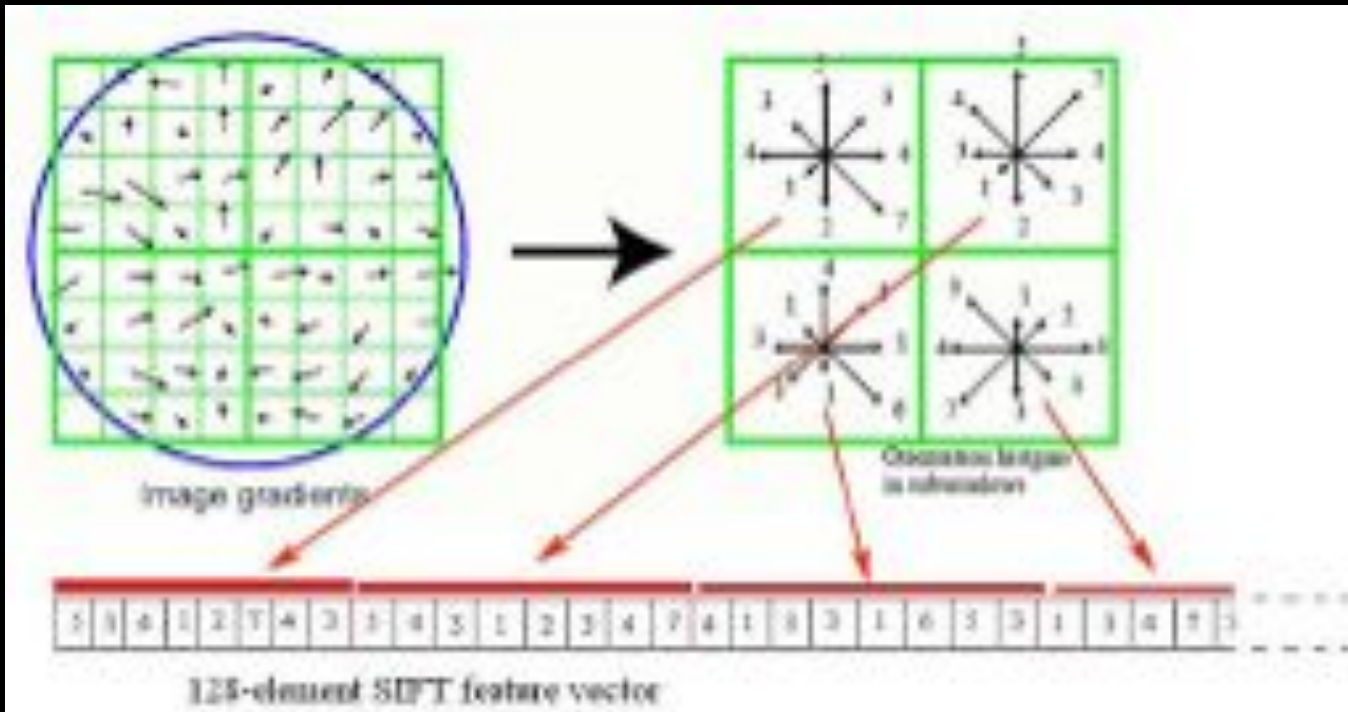
GFTT: Good Features To Track

GIST: Global scene feature

HOG: Histogram of Oriented Gradients

1. ~~How do we~~ detect features?
2. How do we describe features?
3. How do we match features?

Now we've **detected** features, but
how do we **describe** them, and
match similar groups of them?



128 element vector * 320 pixels wide * 240 pixels high

= 38 MB per image!

128 element vector * 300 features =

0.15 MB per image

Store all **keypoints** describing our object in a matrix

5	1	4	1	2	7	4	3	1	4	5	1	2	3	4	7	4	1	8	3	1	6	3	3	1	3	4	7	3	
.
5	1	4	1	2	7	4	3	1	4	5	1	2	3	4	7	4	1	8	3	1	6	3	3	1	3	4	7	3	

Popular Feature Descriptors:

SIFT: Scale Invariant Feature Transform

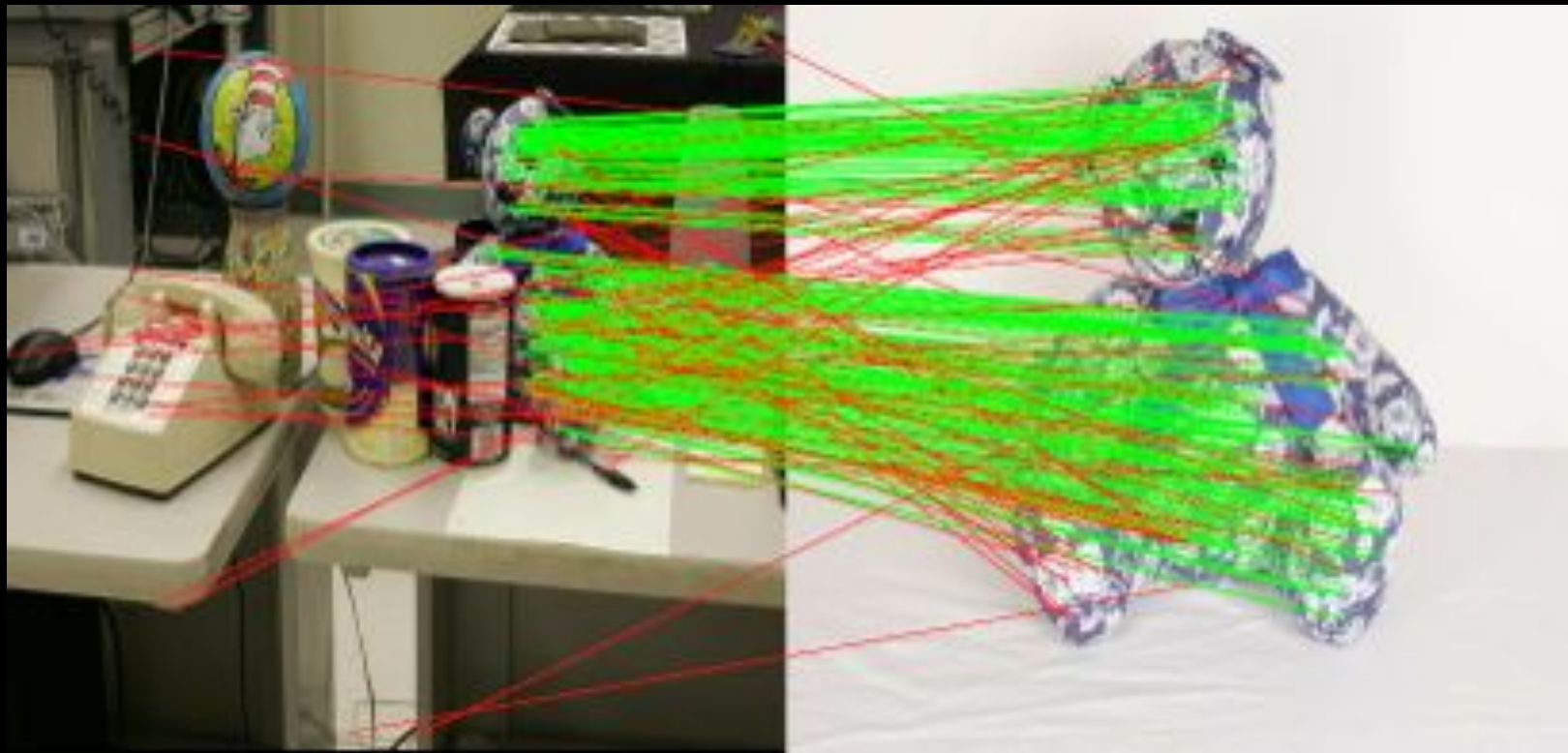
SURF: Speeded-Up Robust Features

BRIEF: Binary string descriptor

Geometric Blur: Samples image from small
deviations

Self-Similarity

1. ~~How do we~~ detect features?
2. ~~How do we~~ describe features?
3. How do we match features?



Nearest neighbors

Hash Table

Approximate Nearest Neighbors

PCA

ICP