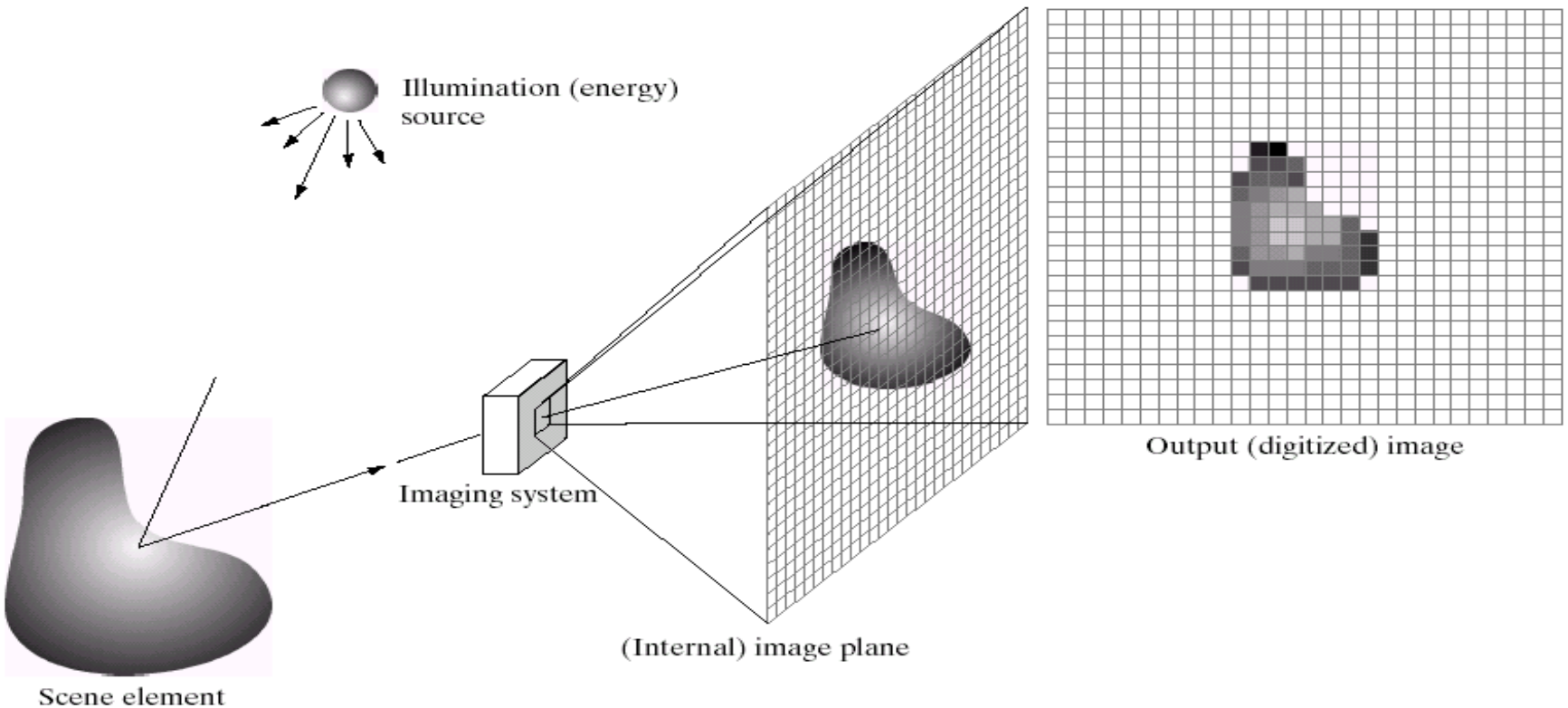
Computer Vision
SIFT
Scale Invariant Feature Transform

AGENDA

- **Introduction.**
- **The scale space.**
- **LoG Approximation .**
- **Finding keypoints.**
- **Get rid of bad key points.**
- **Assigning an orientation to the keypoints.**
- **Generate SIFT features.**

Image Formation



a
b c d e

FIGURE 2.15 An example of the digital image acquisition process. (a) Energy ("illumination") source. (b) An element of a scene. (c) Imaging system. (d) Projection of the scene onto the image plane. (e) Digitized image.

Gaussian filter- examples

- Gaussian distribution

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

original

sigma = 1

Gaussian filter-examples

original

sigma = 3

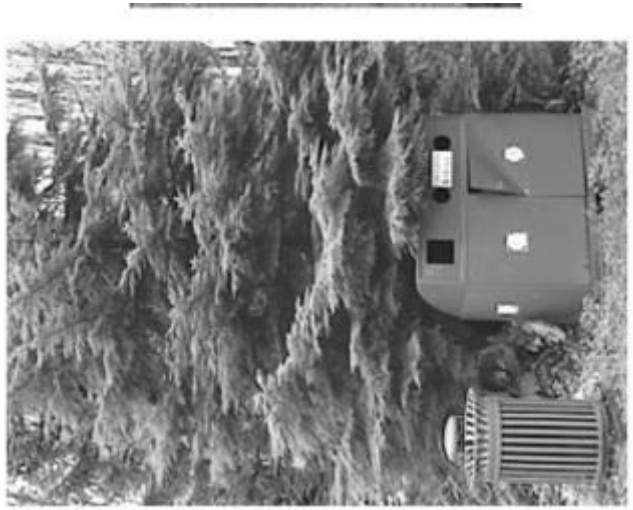
Gaussian filter-examples

original

sigma = 10

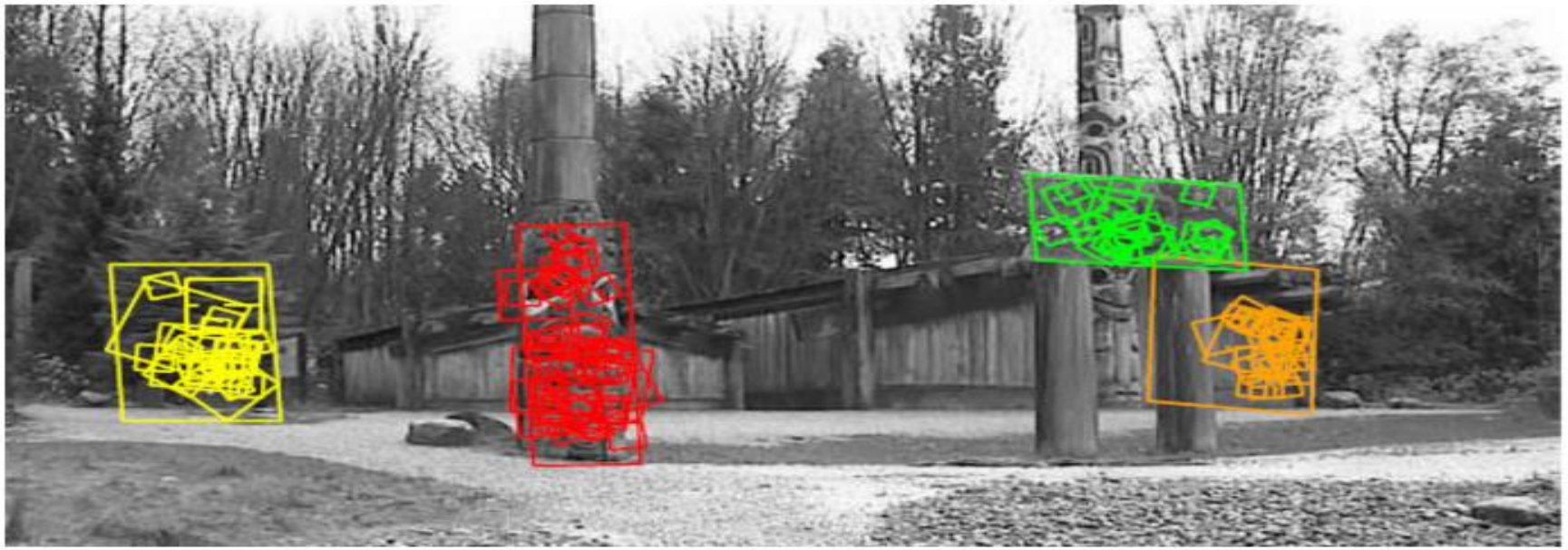
Why care about SIFT

- SIFT isn't just scale invariant. You can change the following, and still get good results:
- **Scale .**
- **Rotation.**
- **Illuminatio**
- **Viewpoint**.



And we want to find these objects in this scene:

Here's the result:



Types of invariance

- Illumination

Types of invariance

- Illumination
- Scale

Types of invariance

- Illumination
- Scale
- Rotation

Types of invariance

- Illumination
- Scale
- Rotation
- Affine

Types of invariance

- Illumination
- Scale
- Rotation
- Affine
- Full Perspective

AGENDA

- Introduction.
- The scale space.
- LoG Approximation .
- Finding keypoints.
- Get rid of bad key points.
- Assigning an orientation to the keypoints.
- Generate SIFT features.

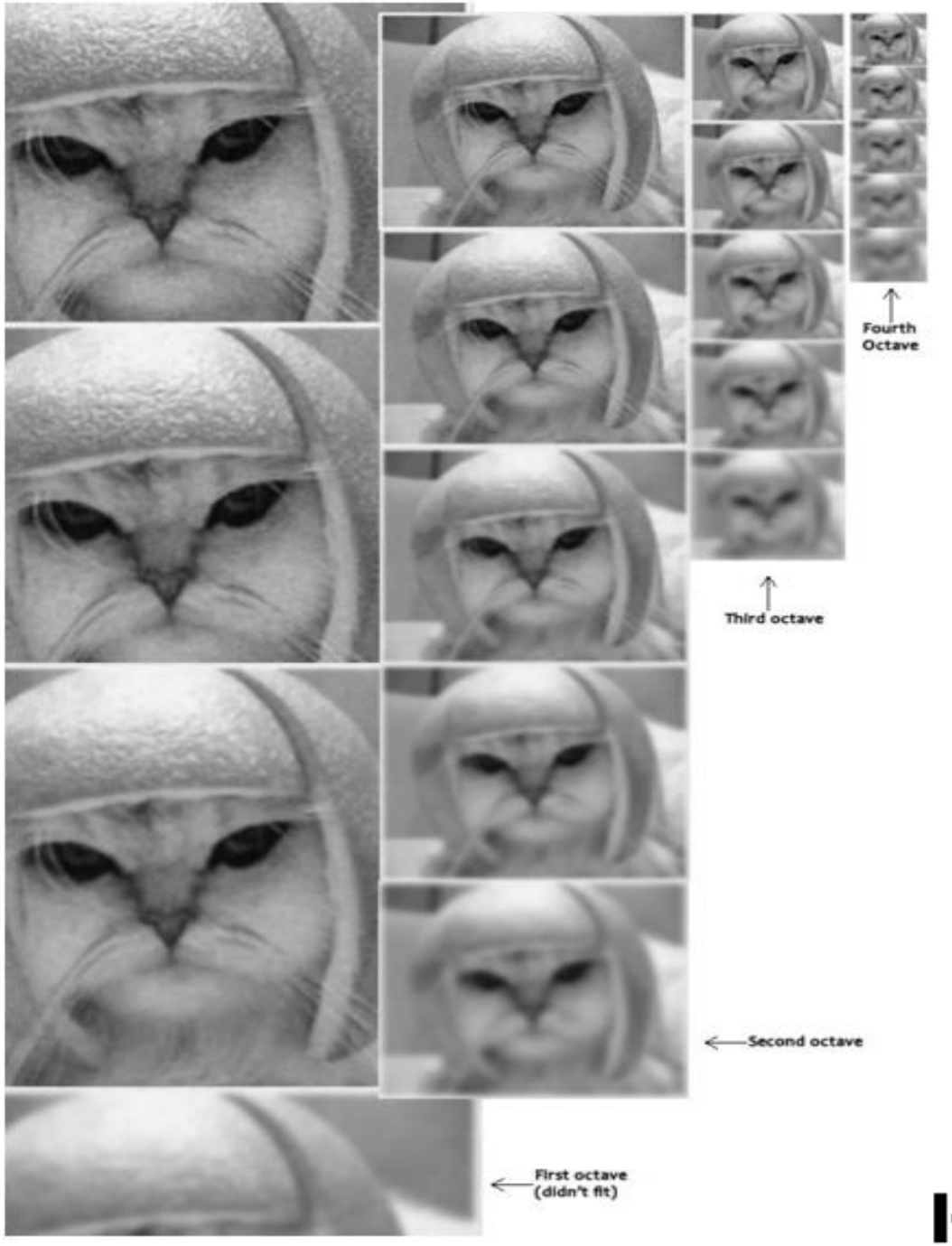
The scale space

- Do you want to look at a leaf or the entire tree? If it's a tree, get rid of some detail from the image (like the leaves, twigs, etc) intentionally.
- The only way to do that is with the Gaussian Blur.
- So to create a scale space, you take the original image and generate progressively **blurred** out images

Here's an example:

Scale spaces in SIFT

- SIFT takes scale spaces to the next level. You take the original image, and generate progressively blurred out images. Then, you resize the original image to half size. And you generate blurred out images again. And you keep repeating.
- Here's what it would look like in SIFT:



Technical Details

- **Octaves and Scales:**

The number of octaves and scale depends on the size of the original image. While programming SIFT, you'll have to decide for yourself how many octaves and scales you want. However, the creator of SIFT suggests that 4 octaves and 5 blur levels are ideal for the algorithm.

- **The first octave:**

If the original image is doubled in size and antialiased a bit (by blurring it) then the algorithm produces more four times more keypoints. The more the keypoints, the better!

Blurring

- Mathematically, "blurring" is referred to as the convolution of the gaussian operator and the image. Gaussian blur has a particular expression or "operator" that is applied to each pixel. What results is the blurred image.
- The symbols:
- L is a blurred image

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

- G is the Gaussian Blur operator
- I is an image
- x,y are the location coordinates
- σ is the "scale" parameter. Think of it as the amount of blur. Greater the value, greater the blur.
- The * is the convolution operation in x and y. It "applies" gaussian blur G onto the image I.
- This is the actual Gaussian Blur operator.

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

Amount of blurring

- The amount of blurring in each image is important. It goes like this. Assume the amount of blur in a particular image is σ. Then, the amount of blur in the next image will be k*σ. Here k is whatever

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

- See how each σ differs by a factor sqrt(2) from the previous one.

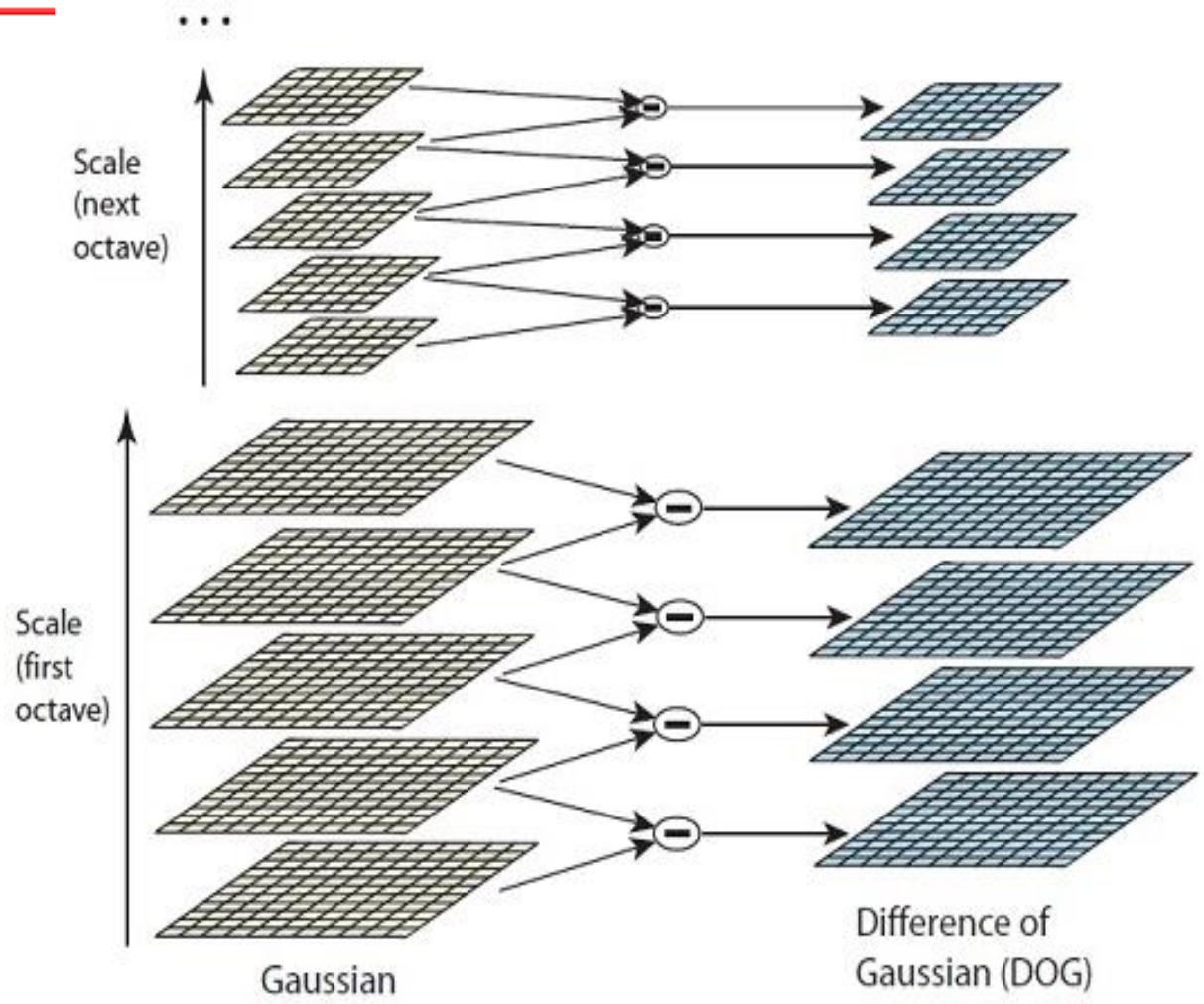
AGENDA

- Introduction.
- The scale space.
- LoG Approximation .
- Finding keypoints.
- Get rid of bad key points.
- Assigning an orientation to the keypoints.
- Generate SIFT features.

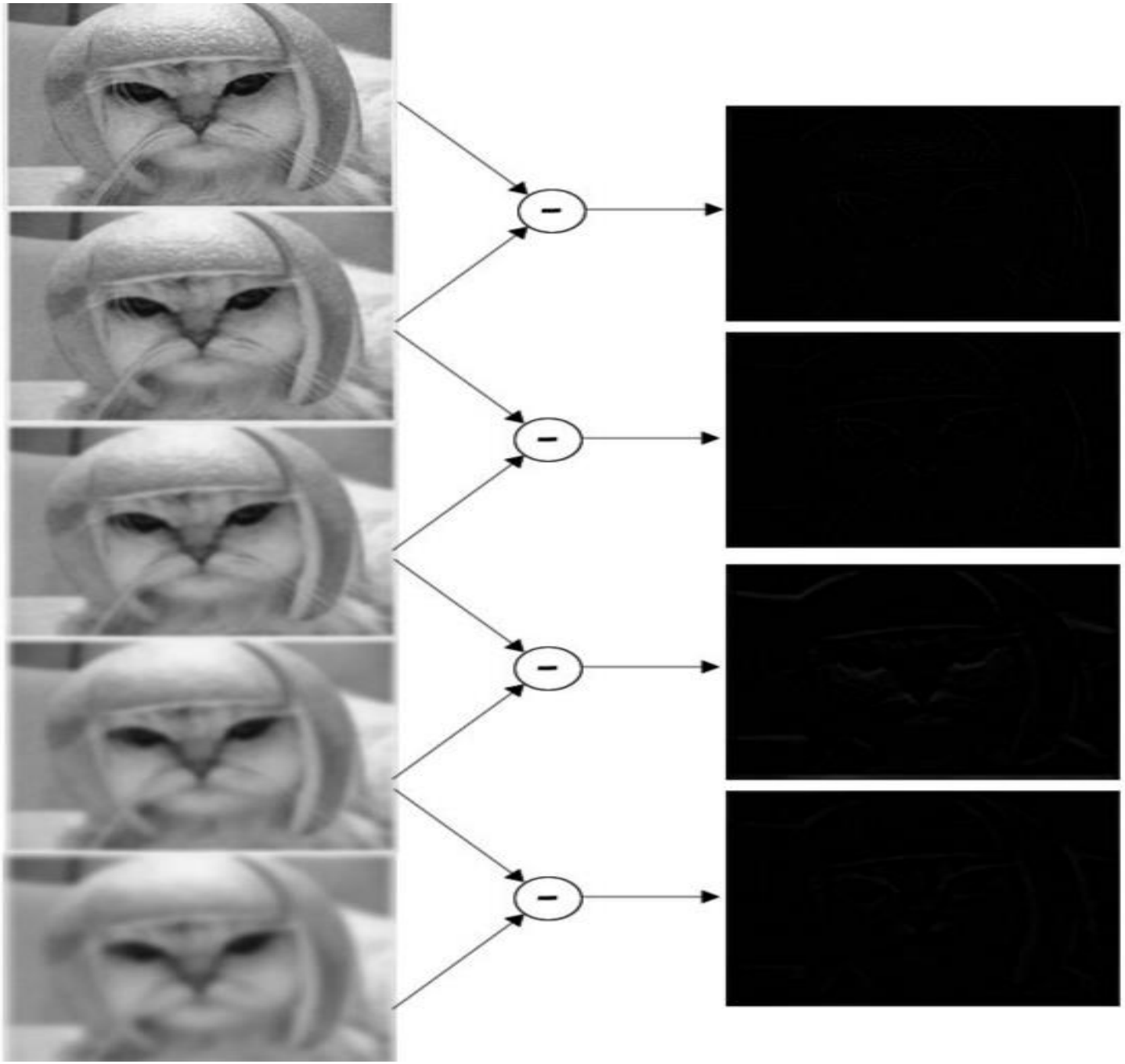
LoG approximations

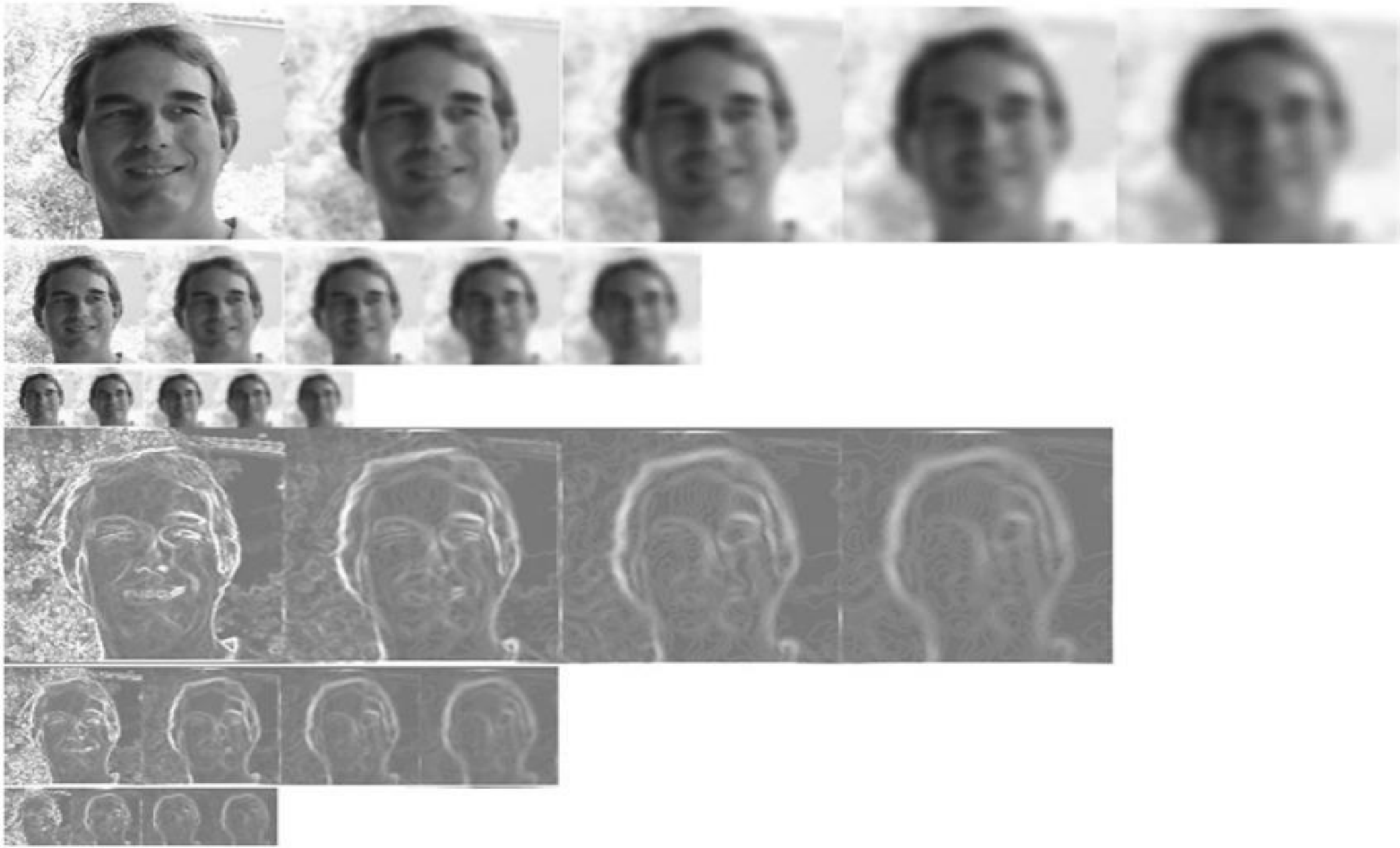
- Laplacian of Gaussian:

These Difference of Gaussian images are approximately equivalent to the Laplacian of Gaussian. And we've replaced a computationally intensive process with a simple subtraction (fast and efficient).



Example





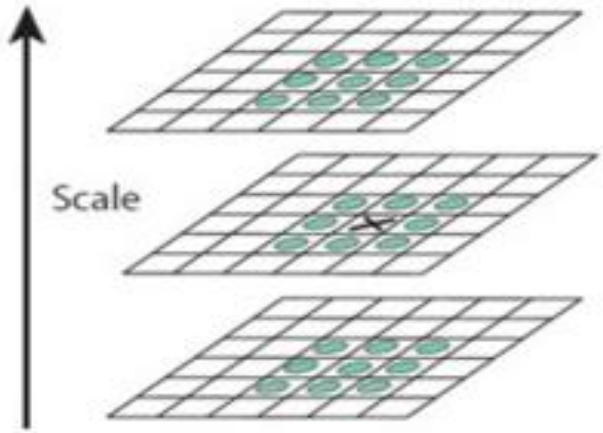
AGENDA

- Introduction.
- The scale space.
- LoG Approximation .
- Finding keypoints.
- Get rid of bad key points.
- Assigning an orientation to the keypoints.
- Generate SIFT features.

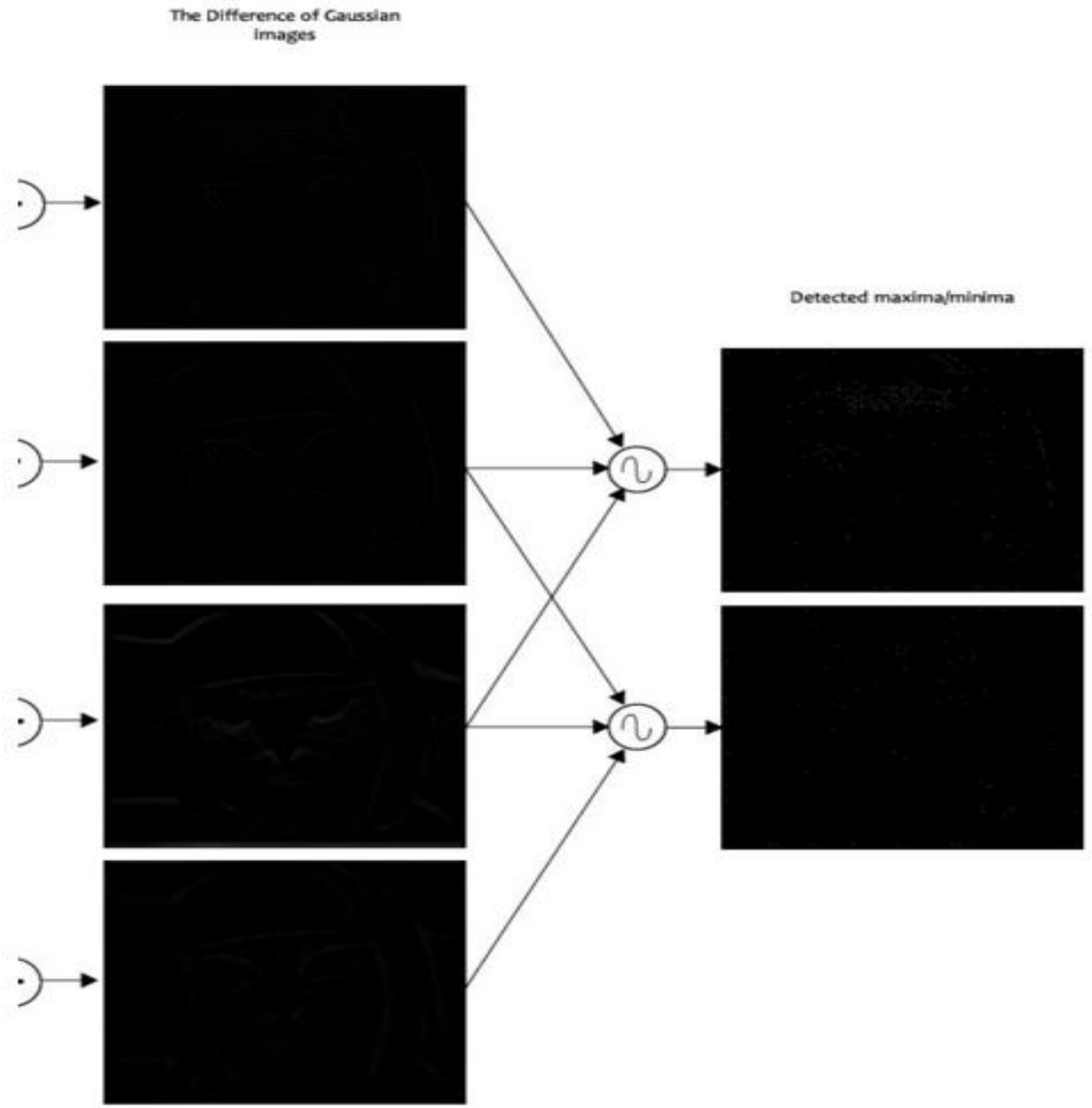
Finding keypoints

- Locate maxima/minima in DoG images

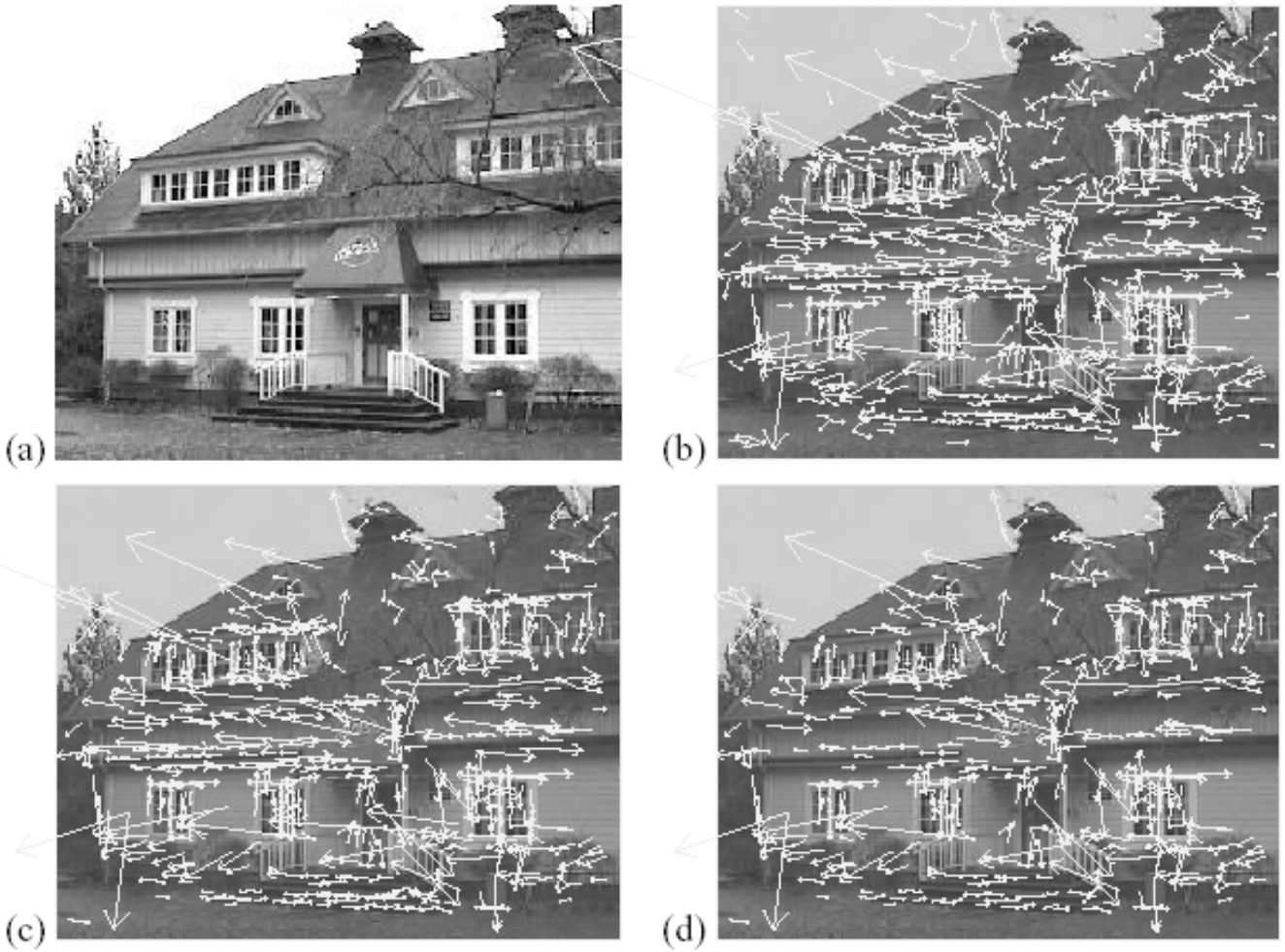
The first step is to coarsely locate the maxima and minima. This is simple. You iterate through each pixel and check all it's neighbours. The check is done within the current image, and also the one above and below it. Something like this:



Example



Keypoint Localization (cont'd)



(a) 233x189 image

(b) 832 DoG extrema

(c) 729 left after low contrast threshold

ratio based on Hessian

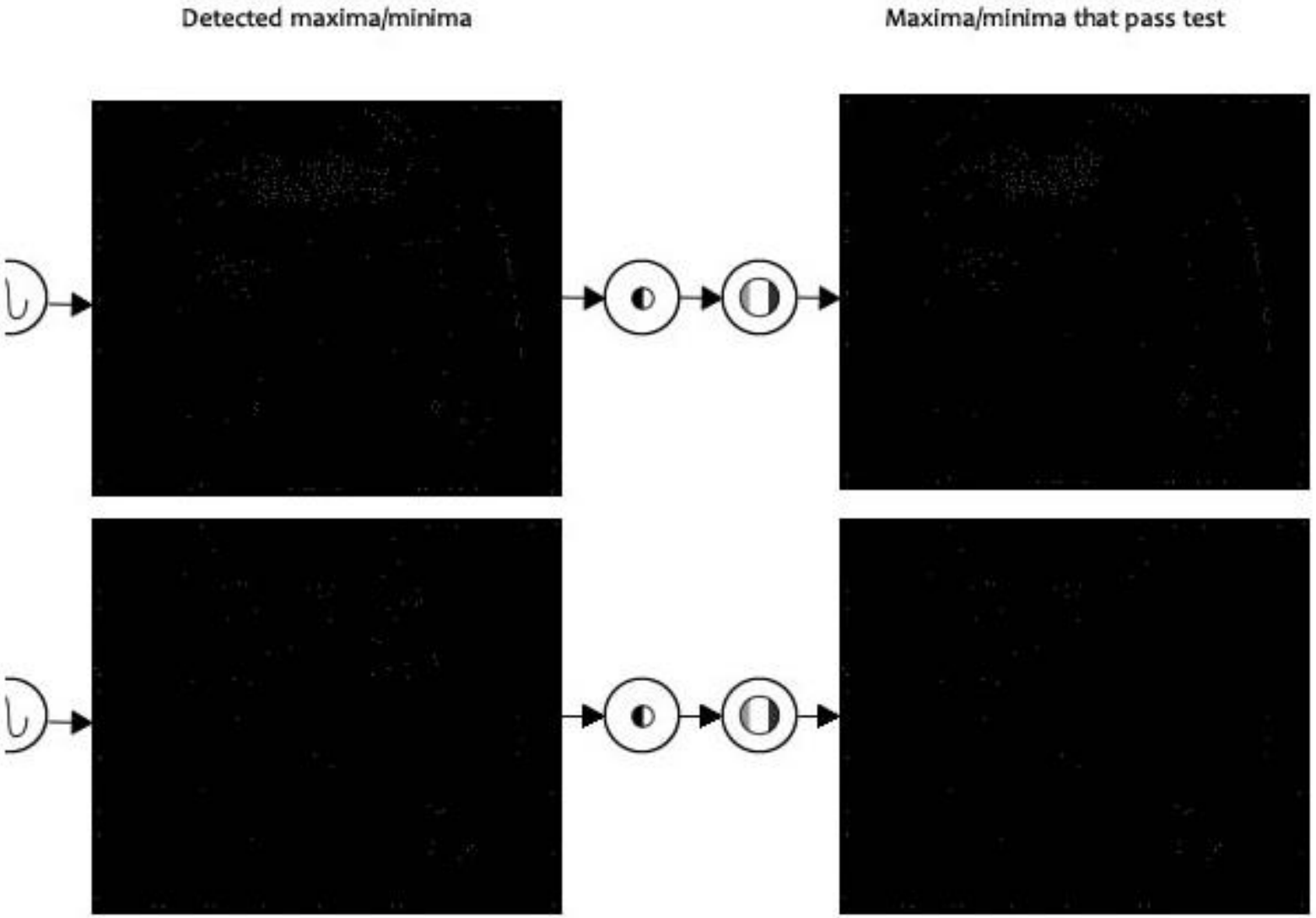
AGENDA

- Introduction.
- The scale space.
- LoG Approximation .
- Finding keypoints.
- Get rid of bad key points.
- Assigning an orientation to the keypoints.
- Generate SIFT features.

Getting rid of low contrast keypoints

- Removing low contrast features
- This is simple. If the magnitude of the intensity (i.e., without sign) at the current pixel in the DoG image (that is being checked for minima/maxima) is less than a certain value, it is rejected.

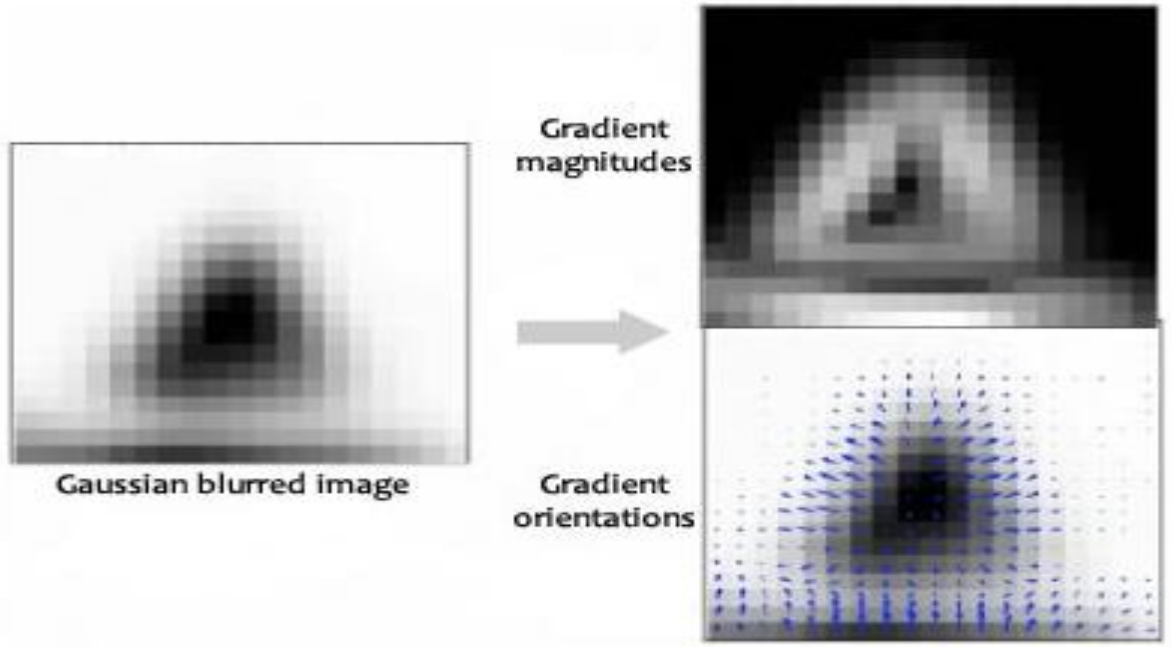
Example



AGENDA

- Introduction.
- The scale space.
- LoG Approximation .
- Finding keypoints.
- Get rid of bad key points.
- Assigning an orientation to the keypoints.
- Generate SIFT features.

Keypoint orientations



Gradient magnitudes and orientations are calculated using these formulae:

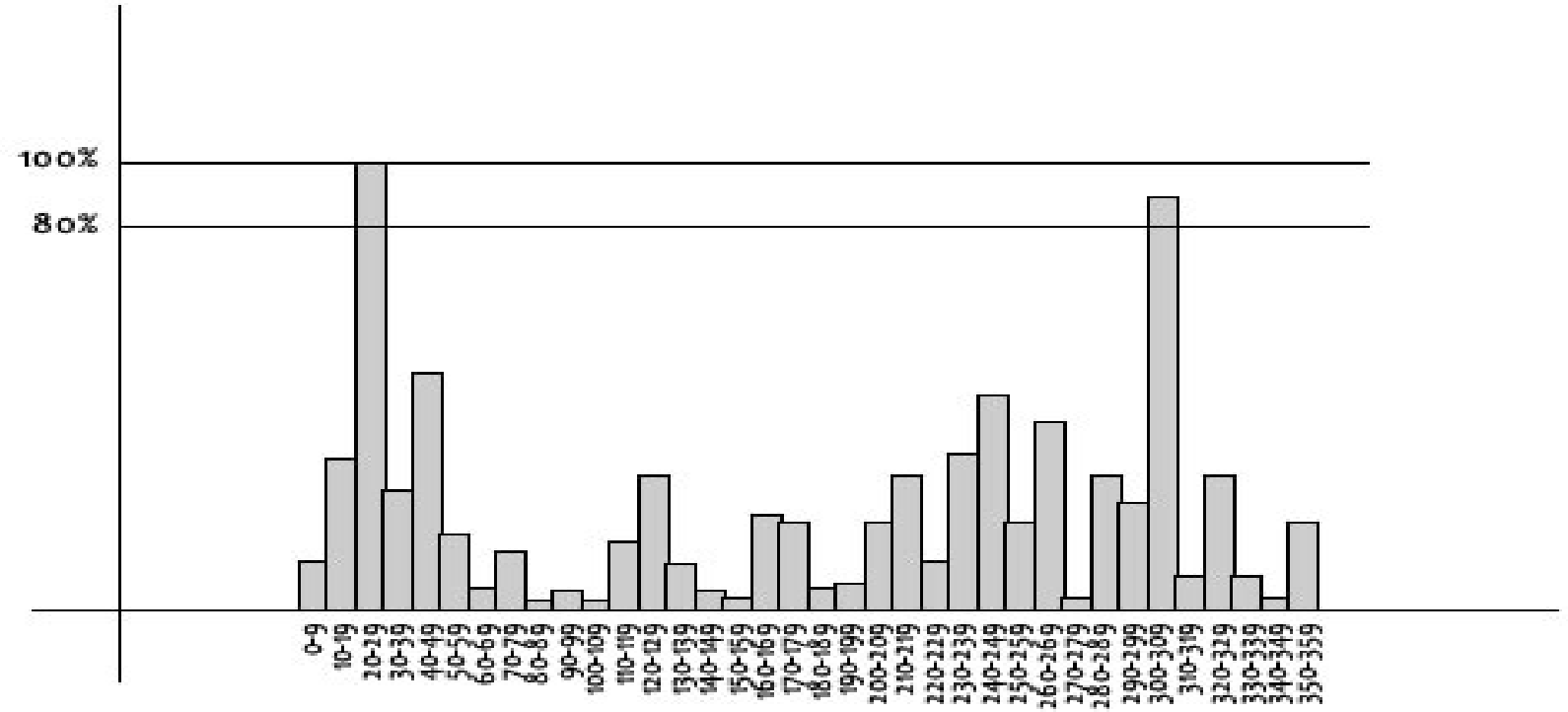
$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2}$$

$$\theta(x,y) = a\tan 2((L(x,y+1) - L(x,y-1)) / (L(x+1,y) - L(x-1,y)))$$

Keypoint orientations

The magnitude and orientation is calculated for all pixels around the keypoint. Then, A histogram is created for this.
In this histogram, the 360 degrees of orientation are broken into 36 bins (each 10 degrees). Lets say the gradient direction at a certain point (in the "orientation collection region") is 18.759 degrees, then it will go into the 10-19 degree bin. And the "amount" that is added to the bin is proportional to the magnitude of gradient at that point.
Once you've done this for all pixels around the keypoint, the histogram will have a peak at some point.
you see the histogram peaks at 20-29 degrees. So, the keypoint is assigned orientation 3 (the third bin)
Also, any peaks above 80% of the highest peak are converted into a new keypoint. This new keypoint has the same location and scale as the original. But it's orientation is equal to the other peak.
So, orientation can split up one keypoint into multiple keypoints.

Keypoint orientations



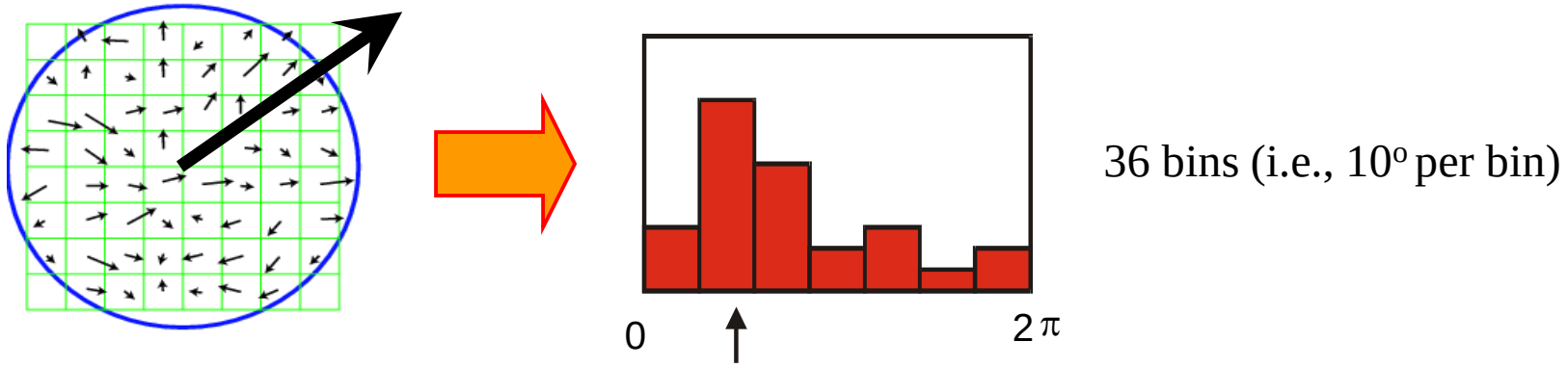
Orientation Assignment

- Create histogram of gradient directions, within a region around the keypoint, at selected scale:

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

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

$$\theta(x, y) = a\tan 2((L(x, y+1) - L(x, y-1)) / (L(x+1, y) - L(x-1, y)))$$

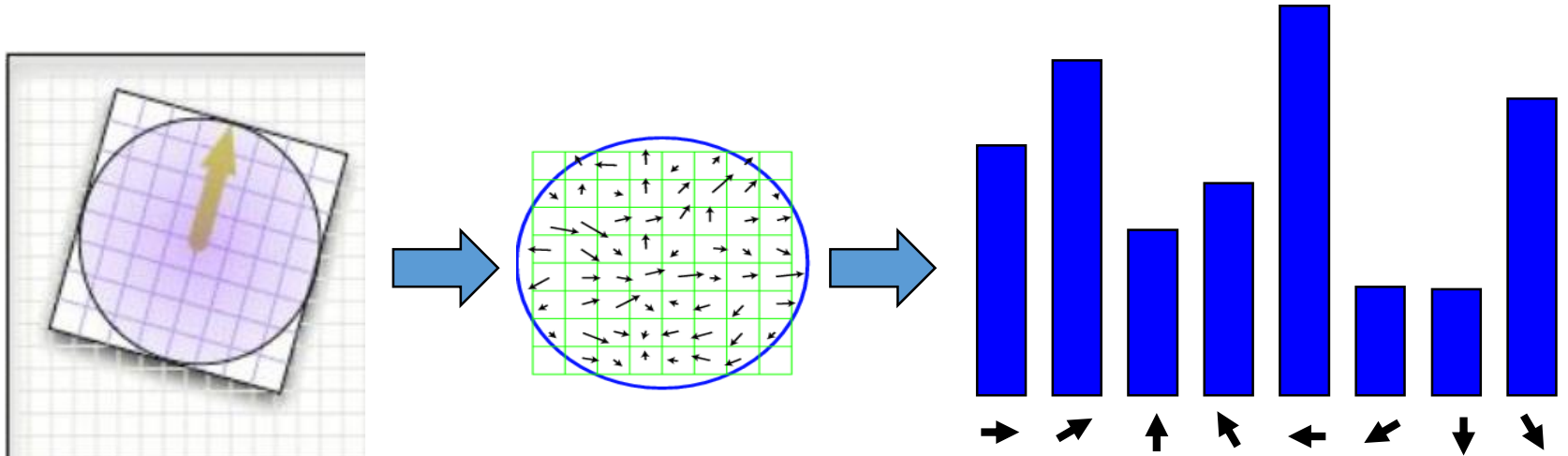


36 bins (i.e., 10^o per bin)

AGENDA

- Introduction.
- The scale space.
- LoG Approximation .
- Finding keypoints.
- Get rid of bad key points.
- Assigning an orientation to the keypoints.
- Generate SIFT features.

Keypoint Descriptor



8 bins

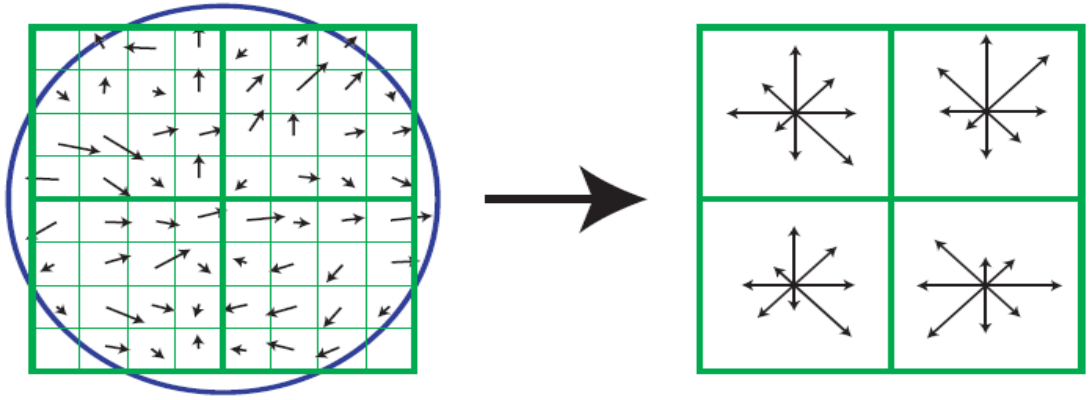
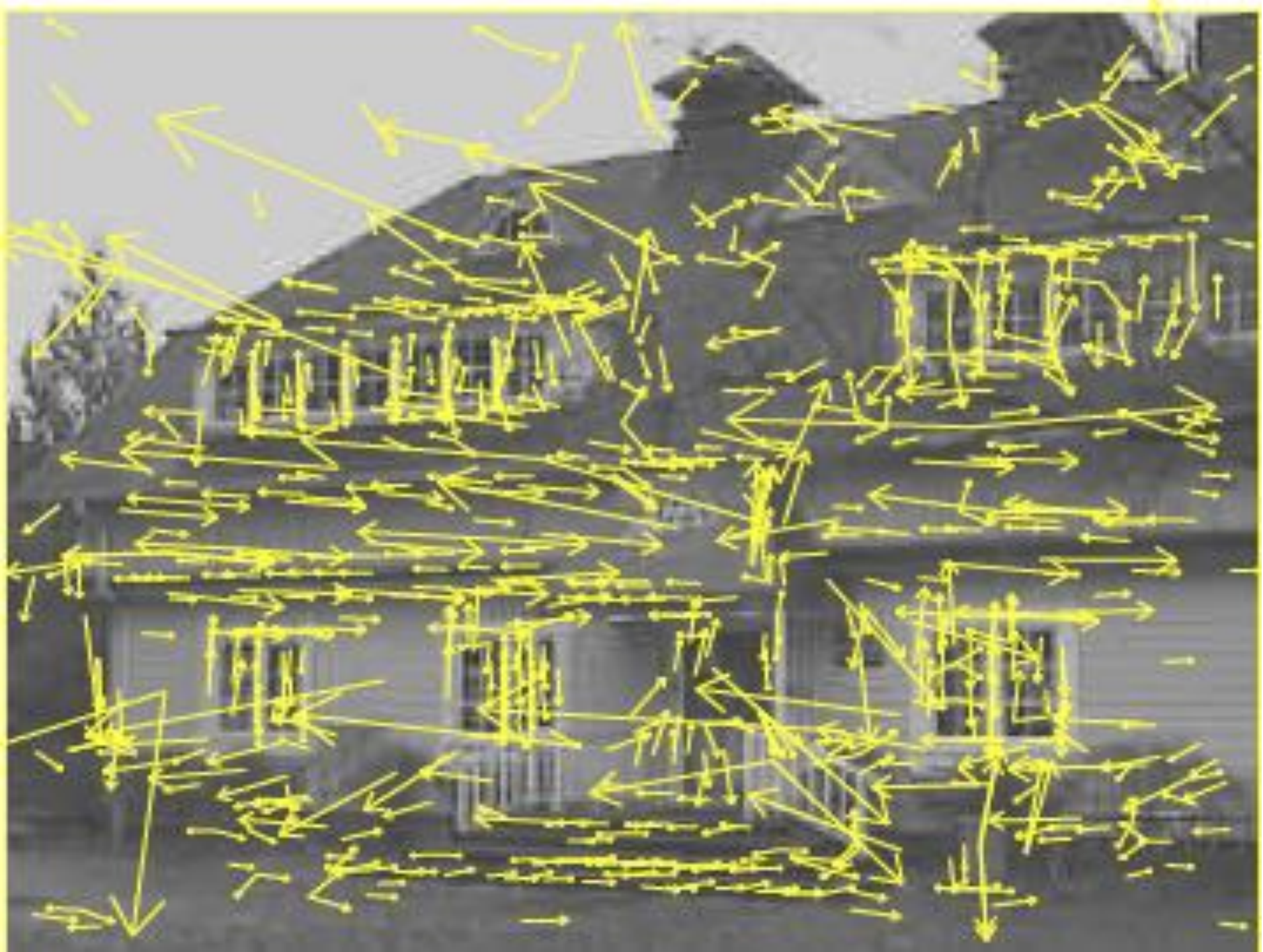
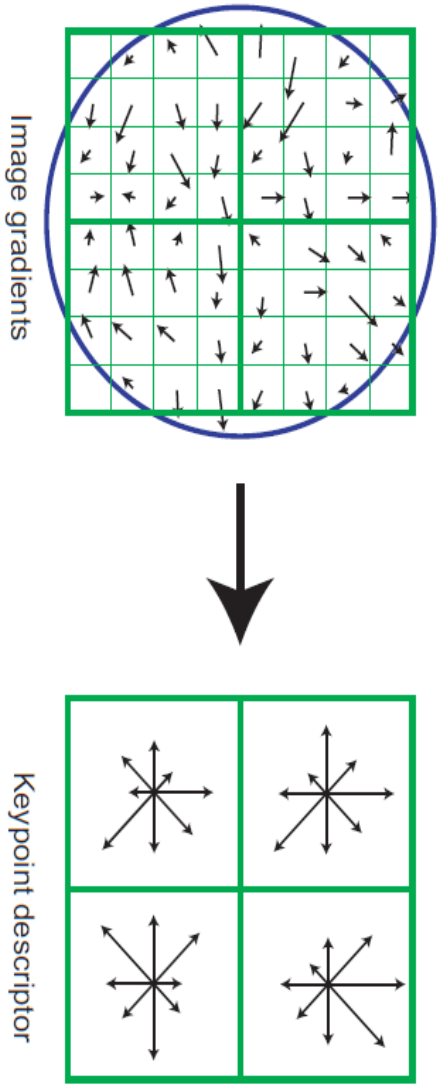


Image gradients

Keypoint descriptor



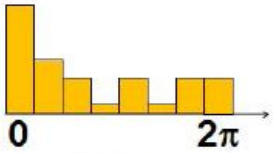


Keypoint Descriptor (cont'd)

1. Take a 16 x16 window around detected interest point.

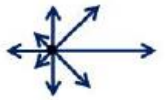
2. Divide into a 4x4 grid of cells.

3. Compute histogram in each cell.



angle histogram

(8 bins)



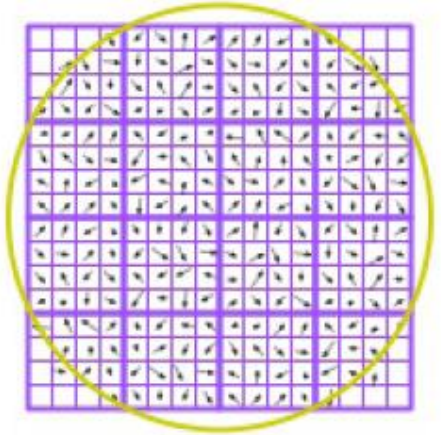
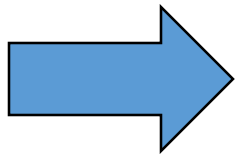
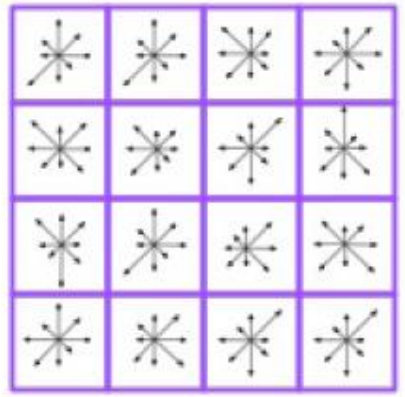


Image gradients




Keypoint descriptor

16 histograms x 8 orientations
= 128 features

Applications of SIFT

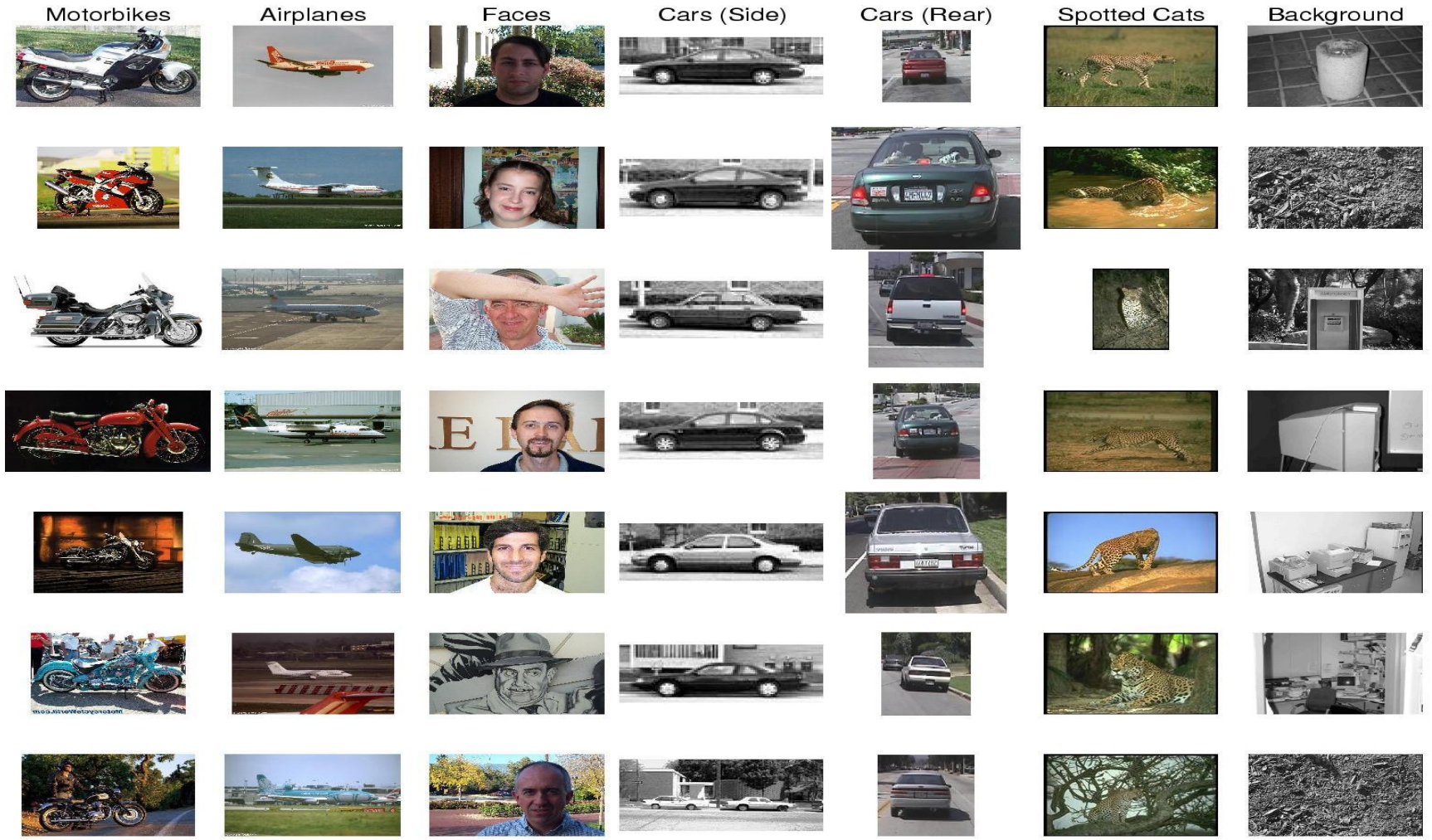
- Object recognition
- Object categorization
- Location recognition
- Robot localization
- Image retrieval
- Image panoramas

Object Recognition

Object Models

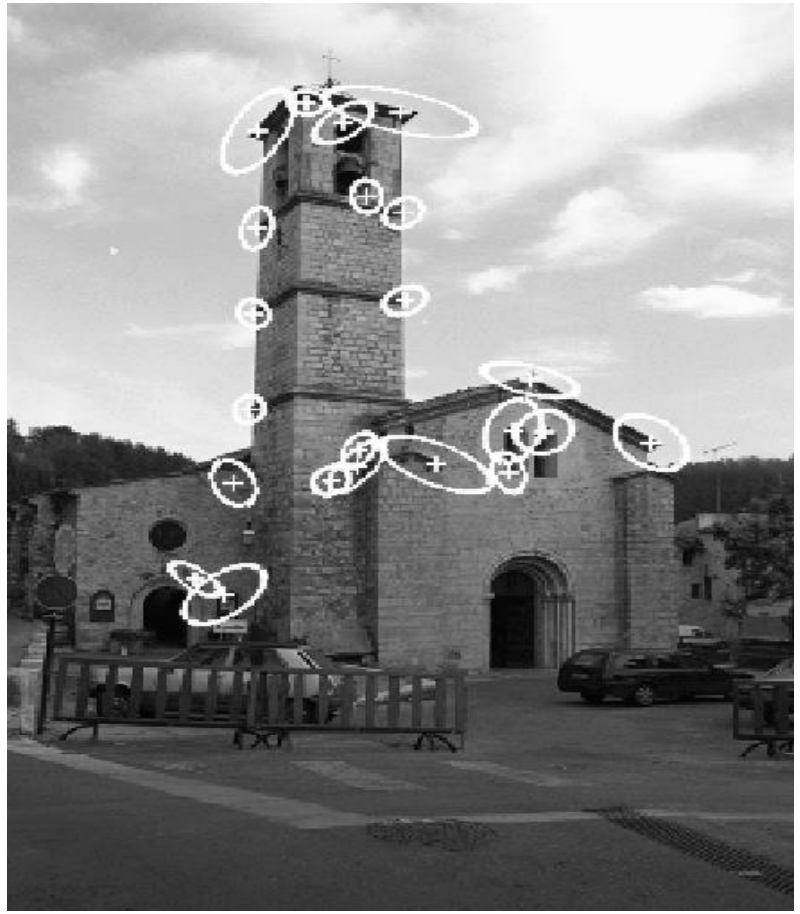

ROAD ATLAS

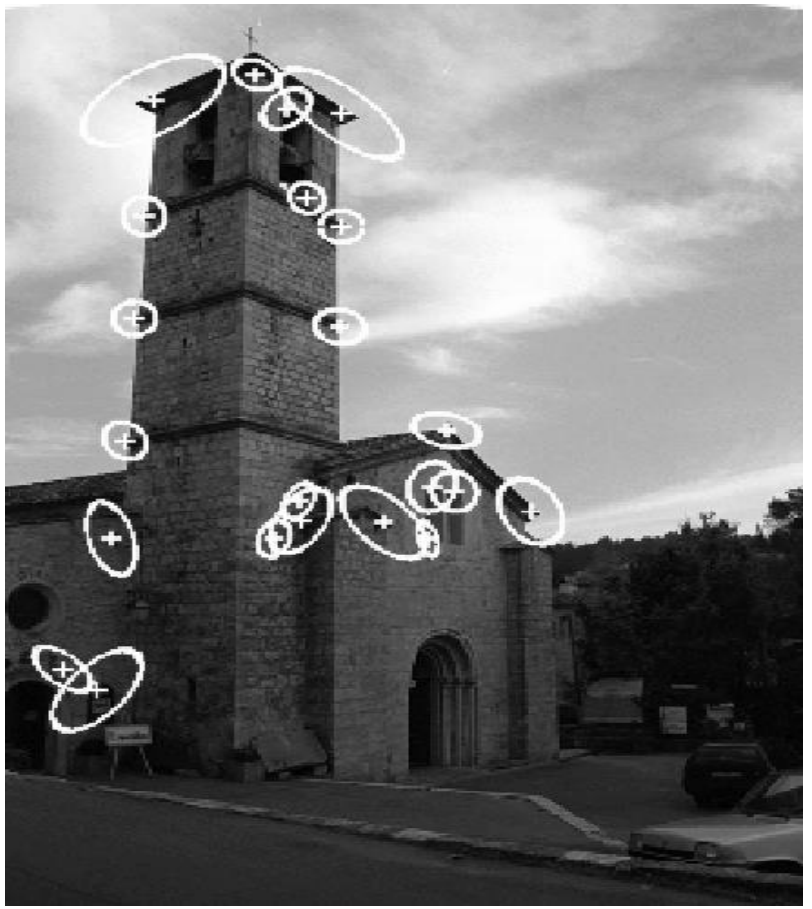
Object Categorization



Location recognition

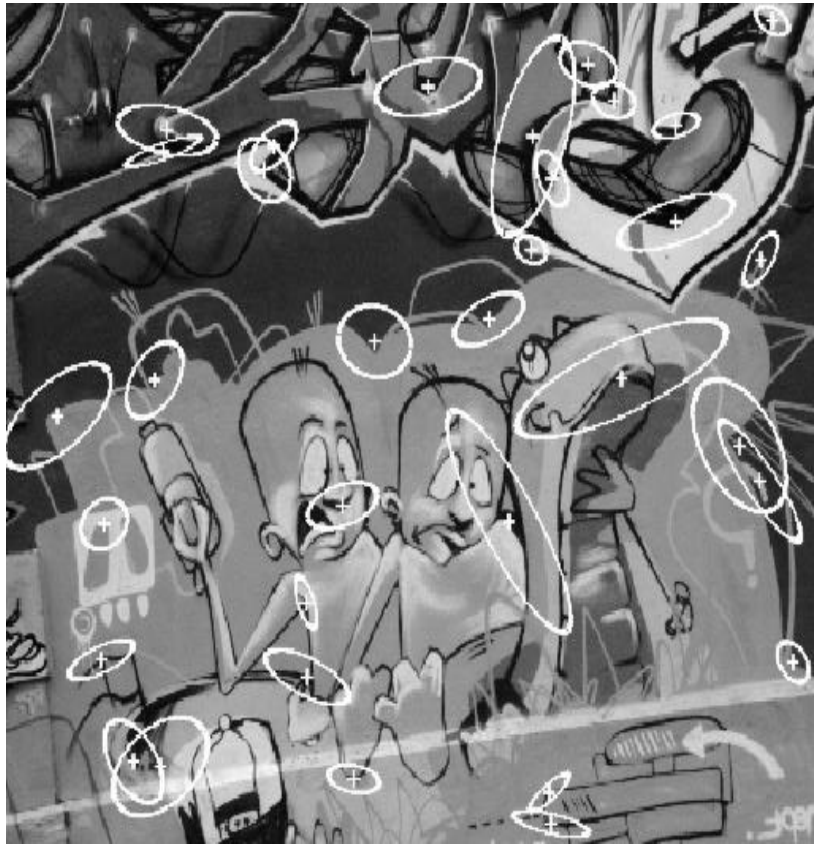
Matches





22 correct matches

Matches



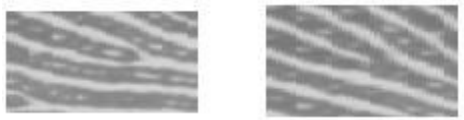
33 correct matches

Image Registration

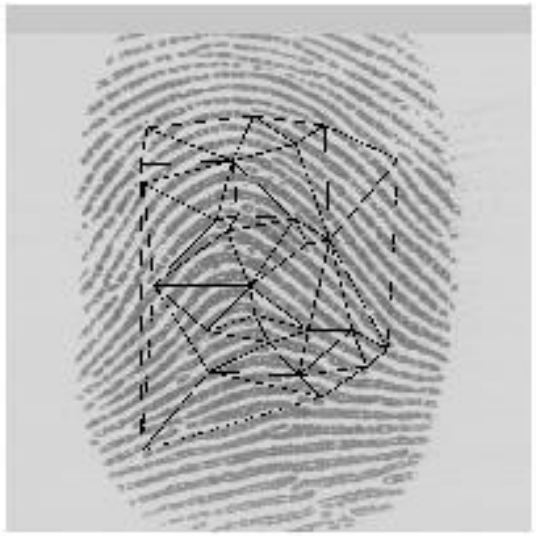
Applications

Fingerprint Identification Research at UNR

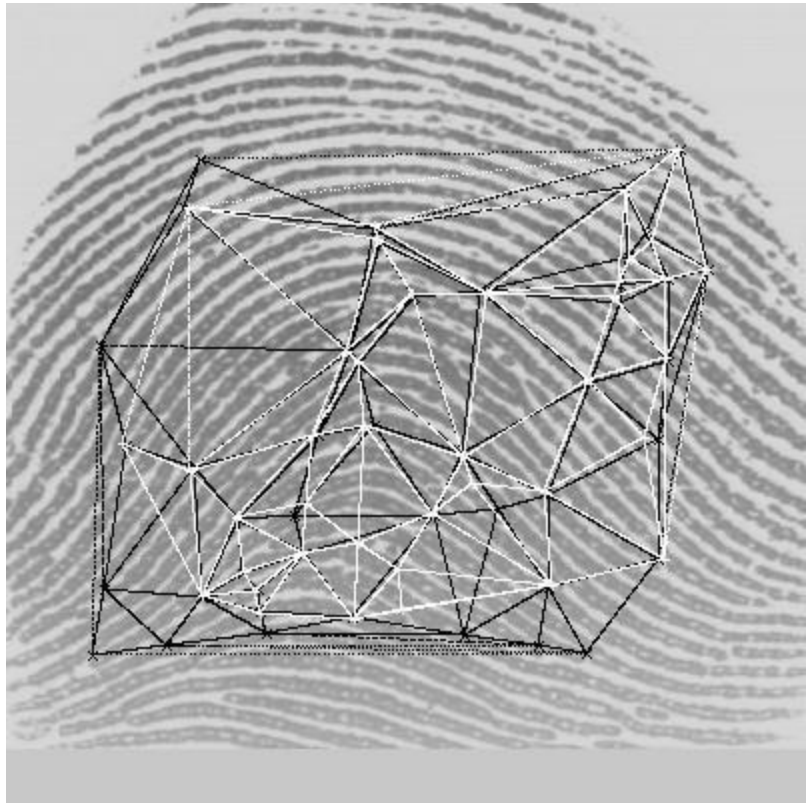
Minutiae



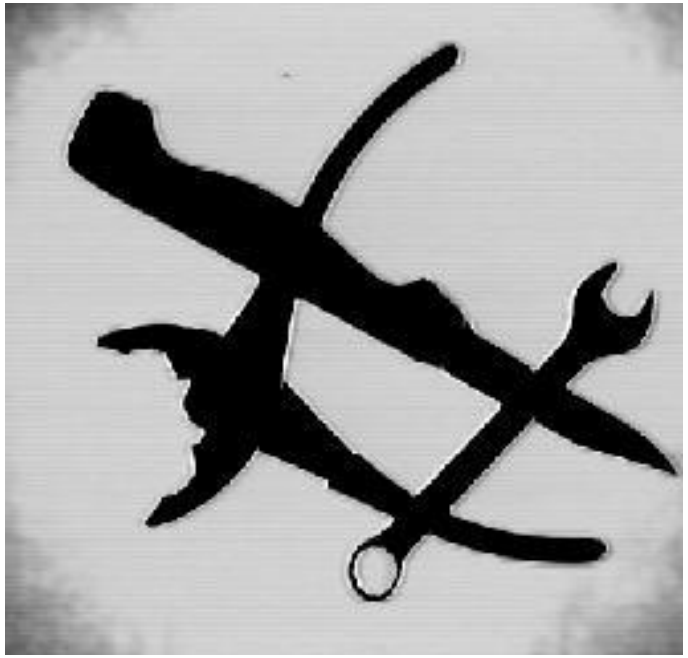
Delaunay Triangulation

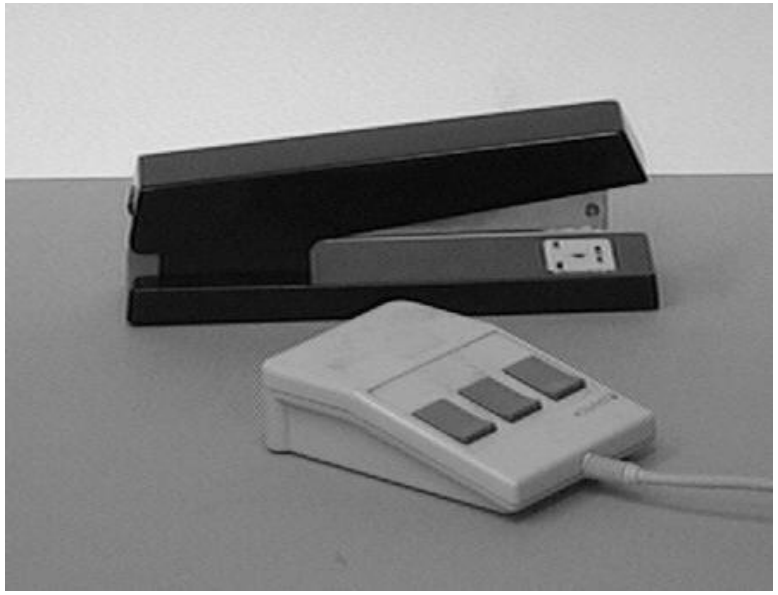


Matching



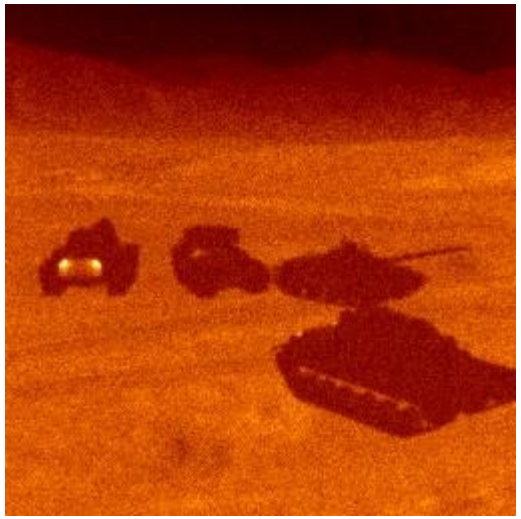
Object Recognition



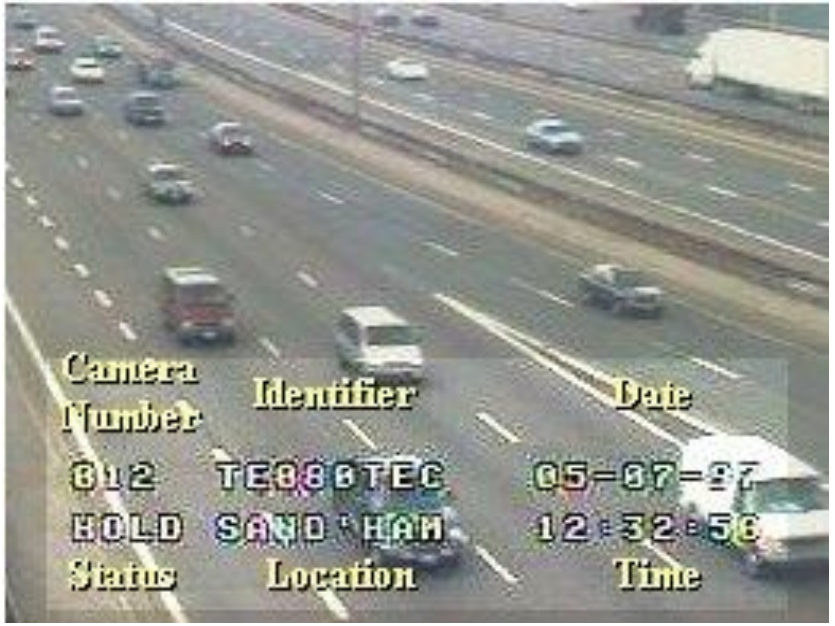


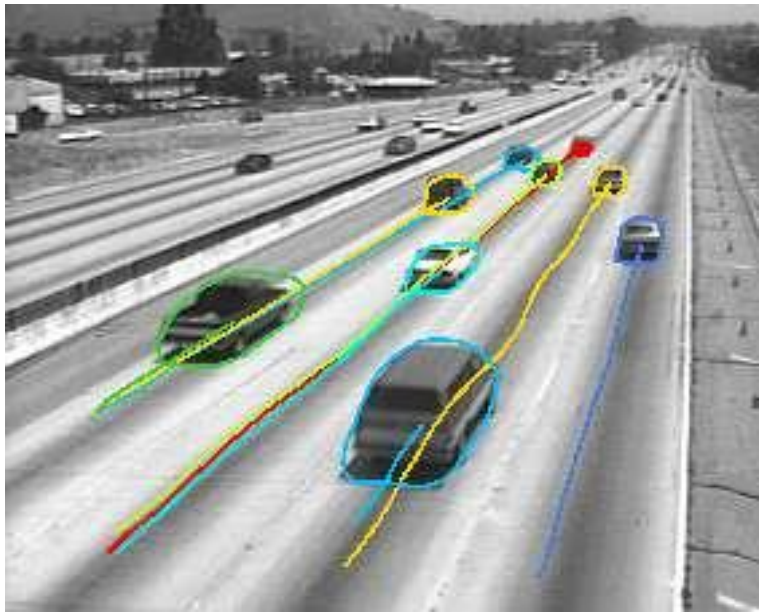
Target Recognition

- Department of Defense (Army, Airforce, Navy)

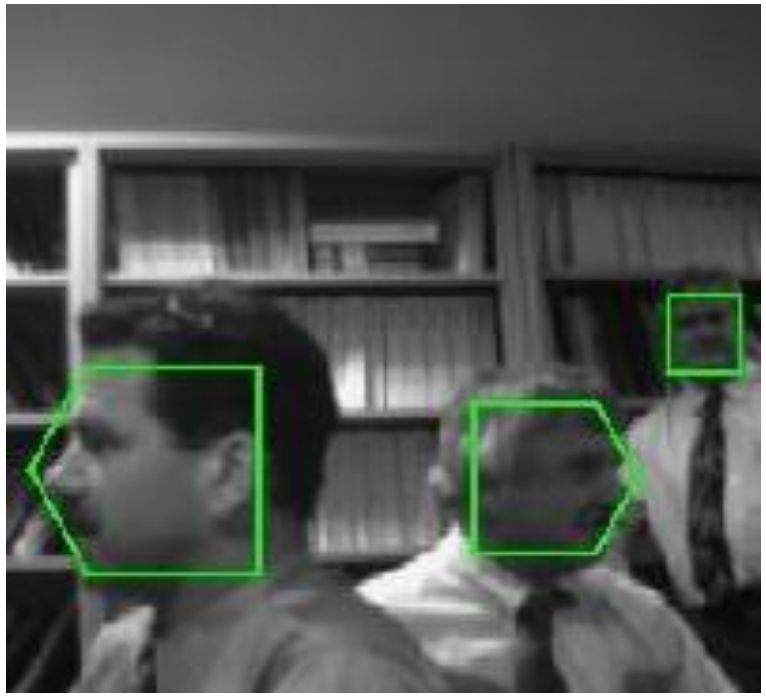


Traffic Monitoring



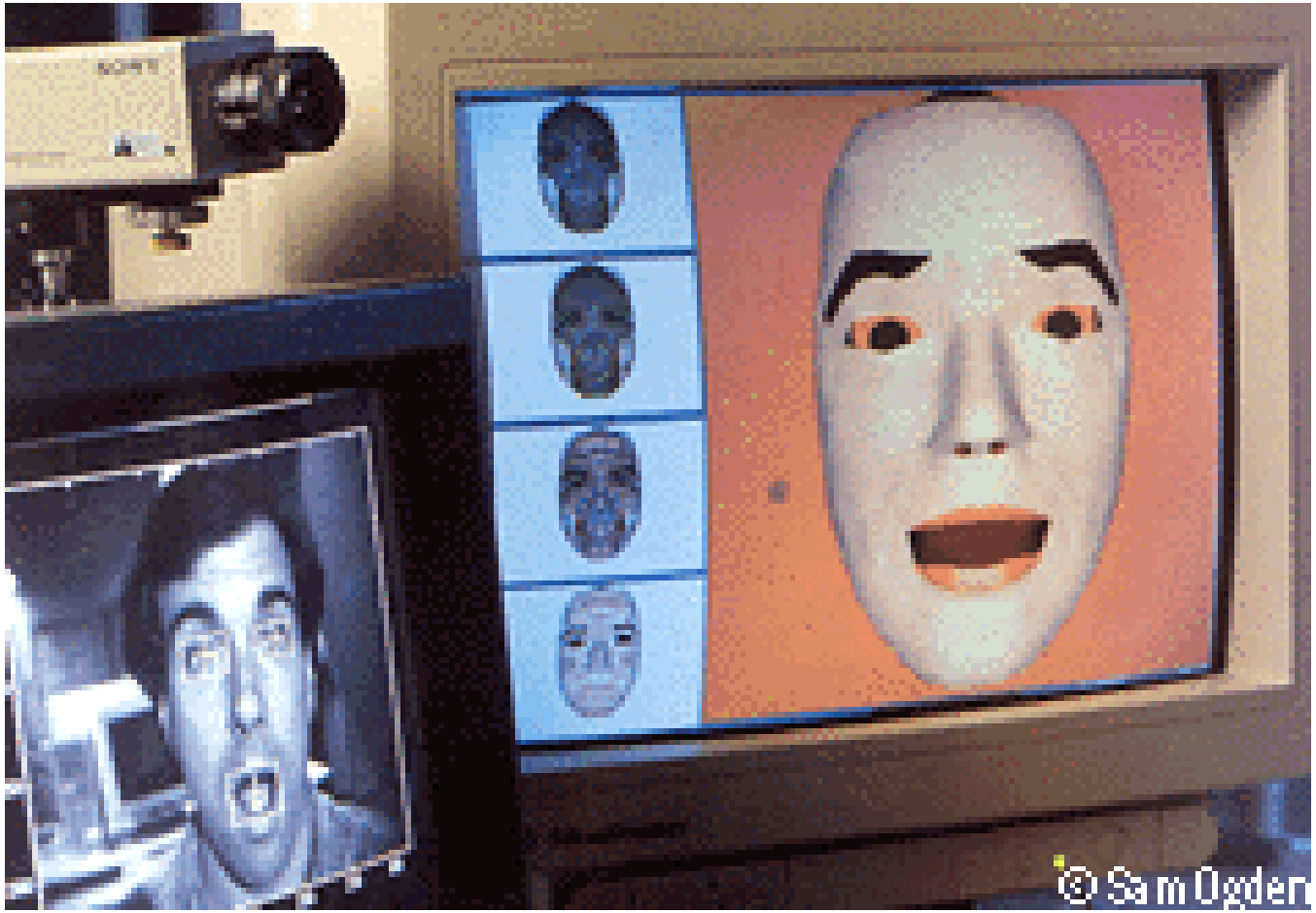


Face Detection



Face Recognition

Facial Expression Recognition



Human Activity Recognition