Computer Vision
Prof. Dr. Songül Varlı

HISTOGRAM OF ORIENTED GRADIENTS

INTEREST POINT DETECTION

CORNER DETECTION

Histograms of Oriented Gradients for Human Detection N. Dalal and B. Triggs , CVPR 2005

- Detecting humans in images is a challenging task owing to their variable appearance and the wide range of poses that they can adopt. The first need is a robust feature set that allows the human form to be discriminated cleanly, even in cluttered backgrounds under difficult illumination
- The feature sets for **human detection**, showing that locally normalized Histogram of Oriented Gradient (HOG) descriptors provide excellent performance relative to other existing feature sets including wavelets

HOG feature extraction steps

1. Compute centered horizontal and vertical gradients with no smoothing
2. Compute gradient orientation and magnitudes
   - For color image, pick the color channel with the highest gradient magnitude for each pixel.
3. For a 64x128 image,
4. Divide the image into 16x16 blocks of 50% overlap.
   - 7x15=105 blocks in total
5. Each block should consist of 2x2 cells with size 8x8.
6. Quantize the gradient orientation into 9 bins
   - The vote is the gradient magnitude
   - Interpolate votes bi-linearly between neighboring bin center.
   - The vote can also be weighted with Gaussian to downweight the pixels near the edges of the block.
7. Concatenate histograms (Feature dimension: 105x4x9 = 3,780)

1- Computing Gradients
2- Compute Gradient Magnitude and Orientation

- Centered: $\grave{f}(x) = \lim_{k\rightarrow\infty}\left(\frac{f(x+h)-f(x-h)}{2h}\right)$

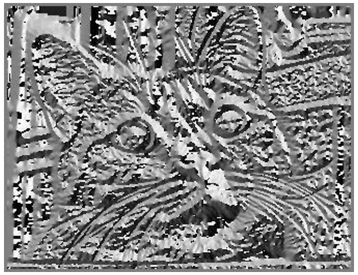
- Filter masks in x and y directions
  - Centered:

- 1	0	1

-1
0
1

- Gradient
  - Magnitude: $M = \sqrt{s_x^2 + s_y^2}$
  - Orientation: $\theta = \arctan(\frac{s_y}{s_x})$

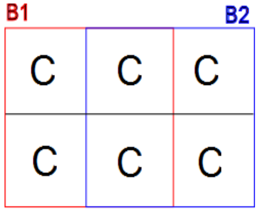




4- Divide Image into Blocks
5- Divide Blocks into Cells

For a 64x128 Image

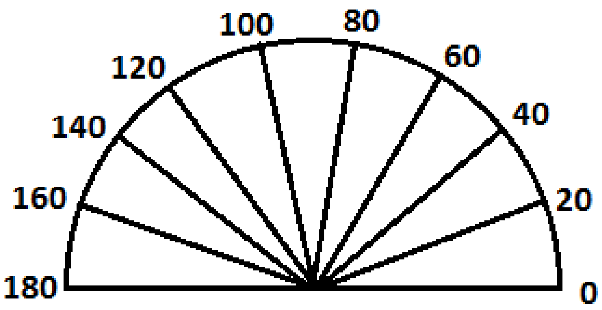
- Divide 16x16 blocks of 50% overlap.
  7x15=105 blocks in total
- Each block should consist of 2x2
  cells with size 8x8.



6- Quantize the Gradient Orientation into 9 bins

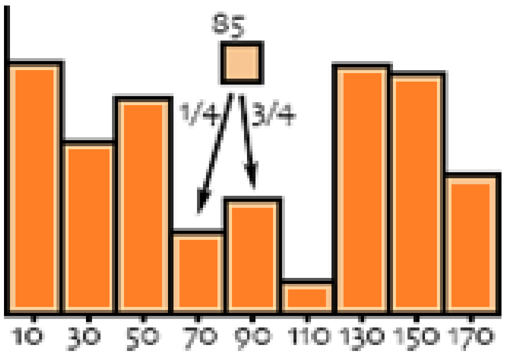
Each block consists of 2x2 cells with size 8x8

- Quantize the gradient orientation into 9 bins (0-180)
- The vote is the gradient magnitude interpolate votes linearly between neighboring bin centers.



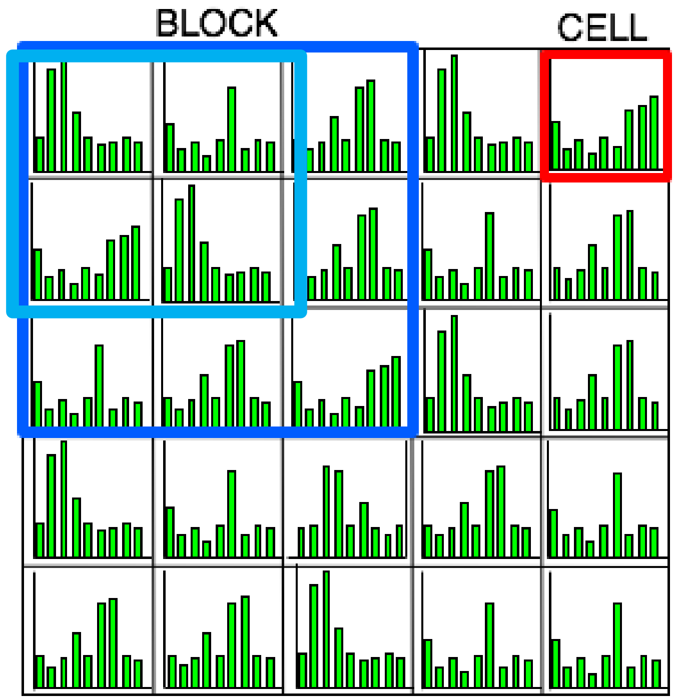
Example: if θ=85 degrees.

- Distance to the bin centre Bin 70 and Bin 90 are 15 and 5 degrees, respectively.
- Hence, ratios are 5/20=1/4, 15/20=3/4.
- The vote can also be weighted with Gaussian to downweight the pixels near the edges of the block.



7- Concatenation of Histograms and Normalization

$$v(n) = \frac{v(n)}{\sqrt{\left(\sum_{k=1}^{2\times2\times9} v(k)^2\right)+1}}$$

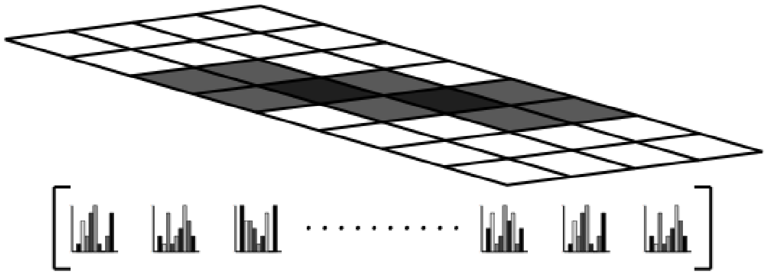


v is the magnitude of each direction

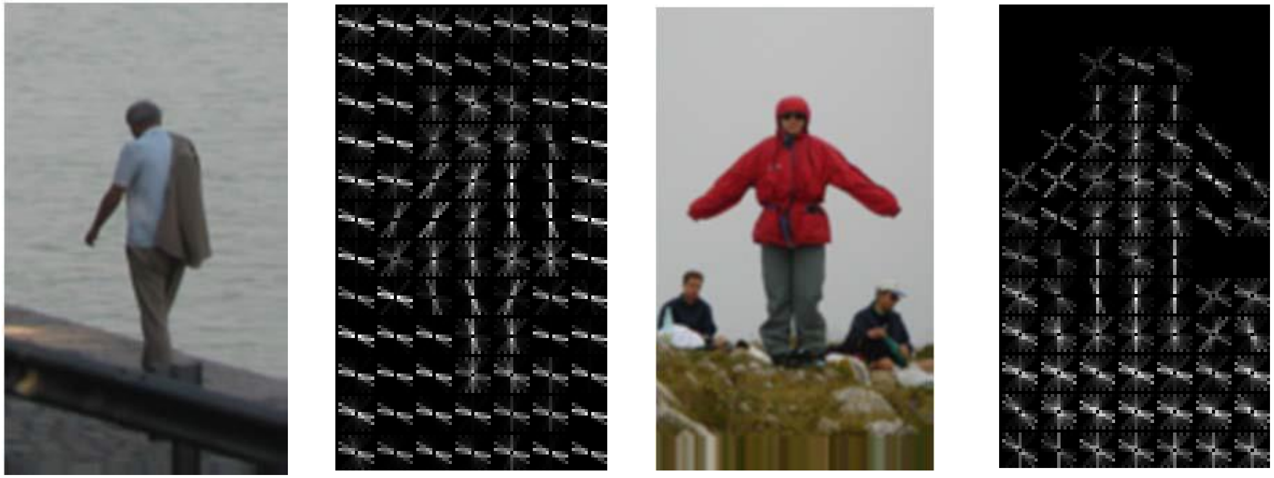
Block (2 × 2 cell) is performed by 50% overlap

Final Feature Vector

- Concatenate histograms
  - Make it a 1D matrix of length 3780.



- Visualization



Results

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

Example of Using HOG

HOG can represent a rough shape of the object, so that it has been used for general object recognition, such as people or cars.

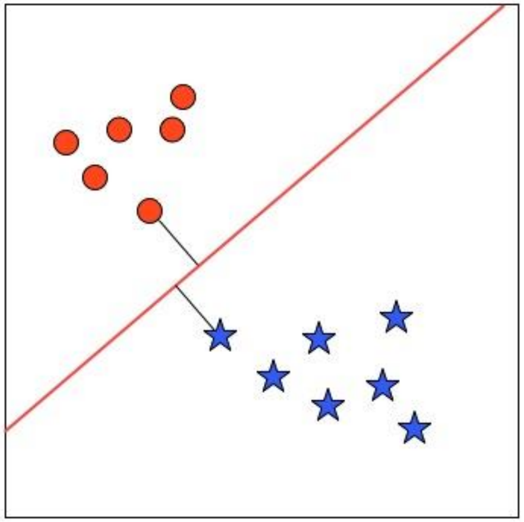
In order to achieve the general object recognition, the classifier (eg SVM) is be used.

1. To teach the classifier, the correct image and the incorrect image.
2. Scan the classifier to determine whether there are people in the detection window.

SVM Classifier

SVM divides space into two domains according to a teacher signal.

New examples are predicted to belong to a category based on which side of the gap domain.

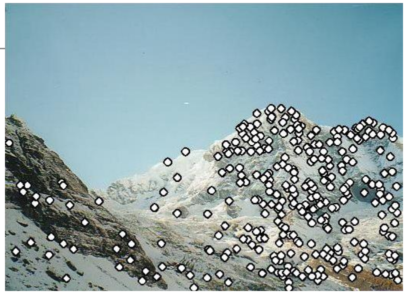


```
%matplotlib inline

import matplotlib.pyplot as plt

from skimage.feature import hog

from skimage import data, exposure

from skimage.color import rgb2gray

image1 = data.astronaut()

image=rgb2gray(image1)

print(image.shape)

fd, hog_image = hog(image, orientations=8, pixels_per_cell=(16, 16),cells_per_block=(1, 1), visualise=True )

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(16, 8), sharex=True, sharey=True)

ax1.axis('off')

ax1.imshow(image, cmap=plt.cm.gray)

ax1.set_title('Input image')

# Rescale histogram for better display

hog_image_rescaled = exposure.rescale_intensity(hog_image, in_range=(0, 10))

ax2.axis('off')

ax2.imshow(hog_image_rescaled, cmap=plt.cm.gray)

ax2.set_title('Histogram of Oriented Gradients')

plt.show()
```

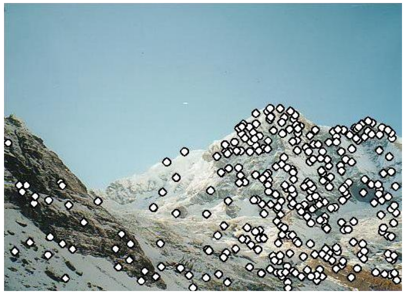
Interest Point Detection

Local features: main components

1) Detection: Identify the interest points



2) Description :Extract feature vector descriptor surrounding each interest point.



3) Matching: Determine correspondence between descriptors in two views

Interest Operator Repetability

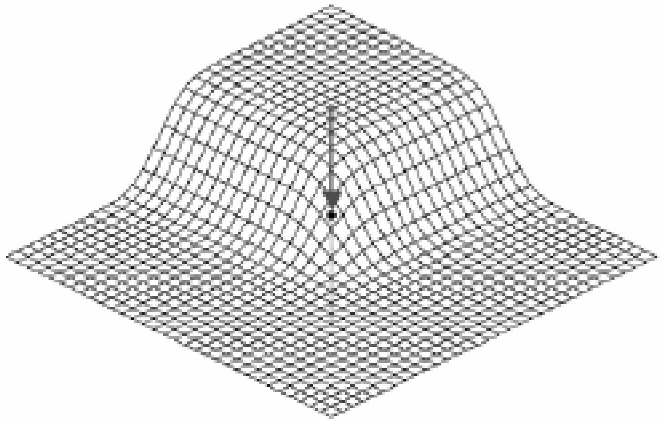
We want to detect (at least some of) the same points in both images.

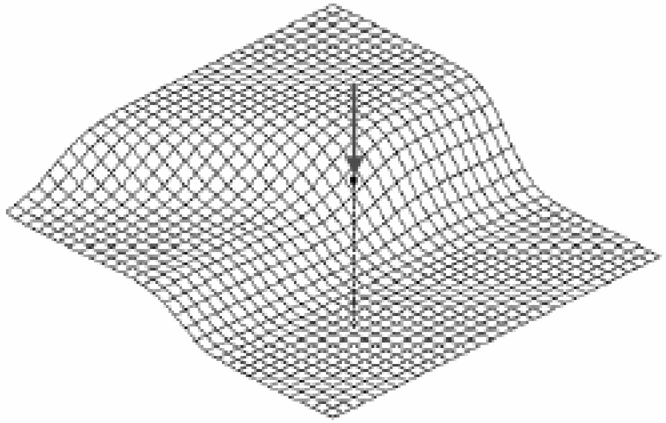
- Yet we have to be able to run the detection procedure *independently* per image.

What is an Interest Point

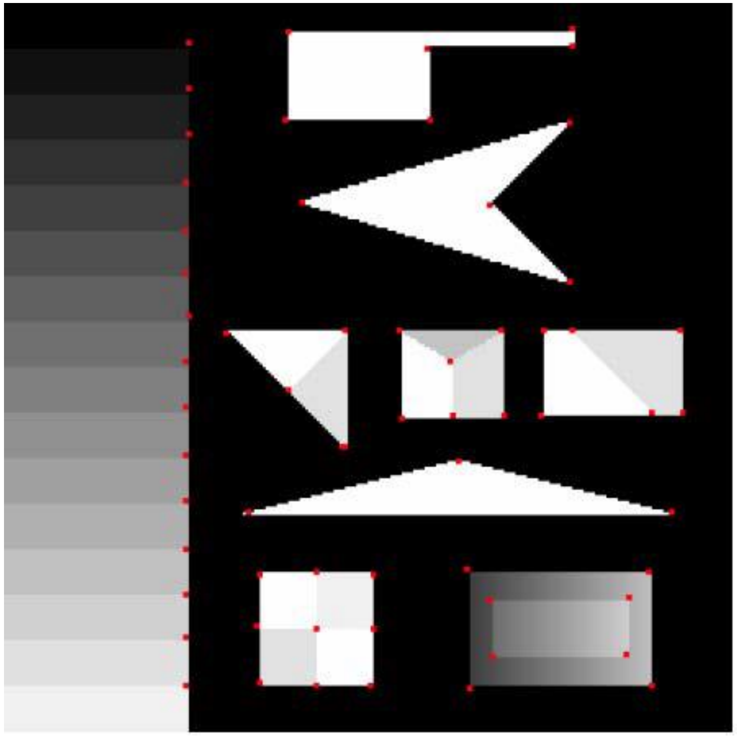
Expressive texture

The point at which the direction of the boundary of object changes abruptly





Synthetic and Real Interest Points



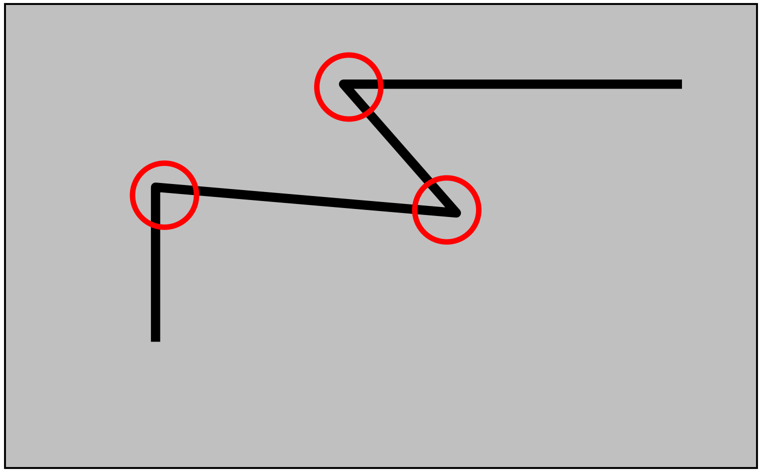
Corners are indicated in red

Properties of Interest Point Detectors

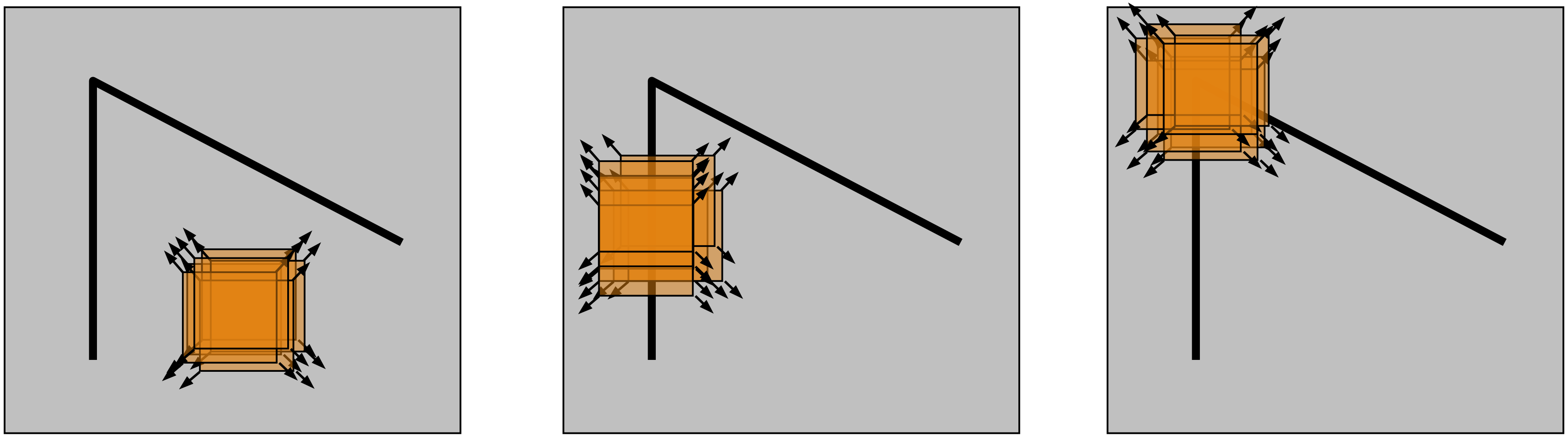
- Detect all (or most) true interest points
- No false interest points
- Well localized.
- Robust with respect to noise.
- Efficient detection

Harris Corner Detector

- Corner point can be recognized in a window
- Shifting a window in any direction should give a large change in intensity



Harris Detector: Basic Idea



"flat" region:
no change in
all directions

"edge":
no change along
the edge direction

"corner":
significant change
in all directions

Harris Detector : Mathematics

Change of intensity for the shift [u,v]:

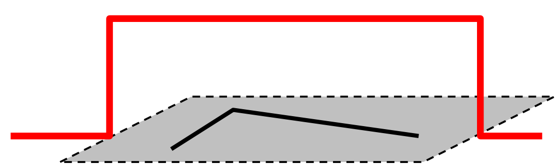
$$E(u,v) = \sum_{x,y} w(x,y)\left[I(x+u,y+v) - I(x,y)\right]^2$$

Window function

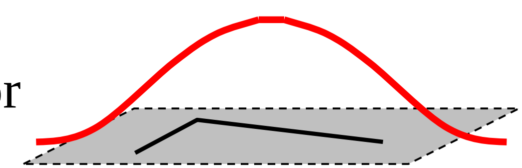
Shifted intensity

Intensity

Window function $w(x,y)$ =



or



1 in window, 0 outside

Gaussian

Harris Detector: Mathematics

For small shifts $[u,v]$ we have a *bilinear* approximation:

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

where M is a 2×2 matrix computed from image derivatives:

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

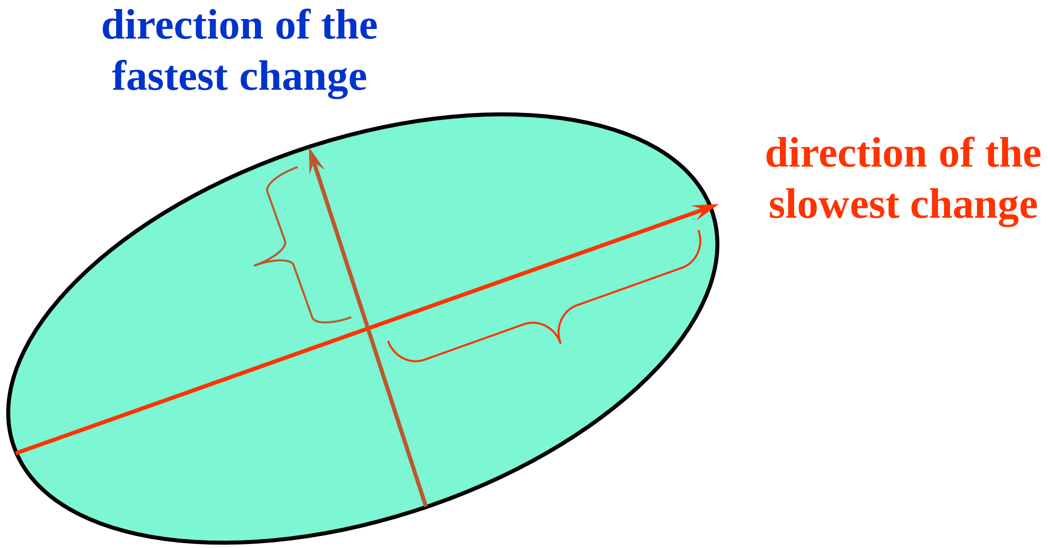
Harris Detector: Mathematics

Intensity change in shifting window: eigenvalue analysis

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

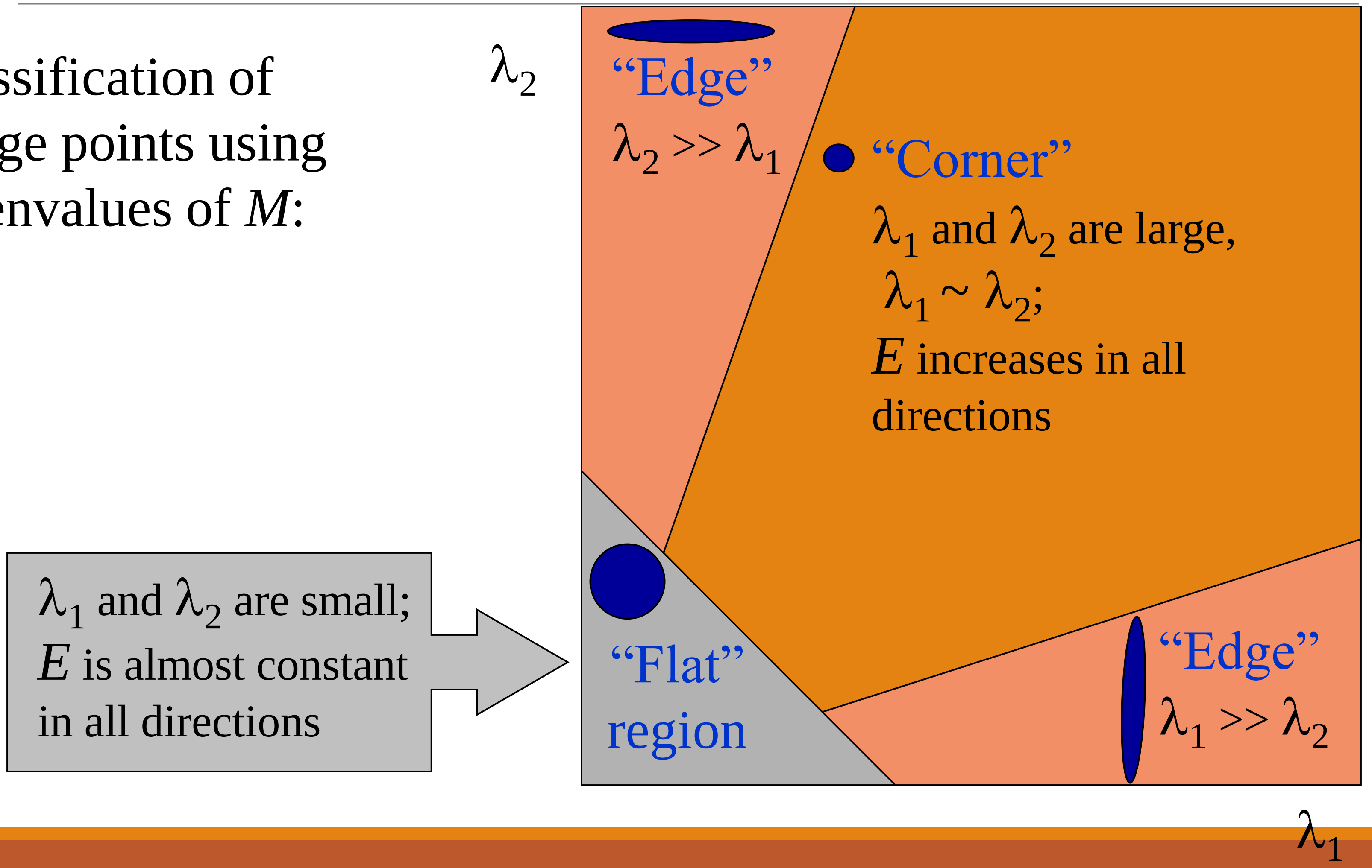
λ_1, λ_2 – eigenvalues of M

Ellipse $E(u,v)$ = const



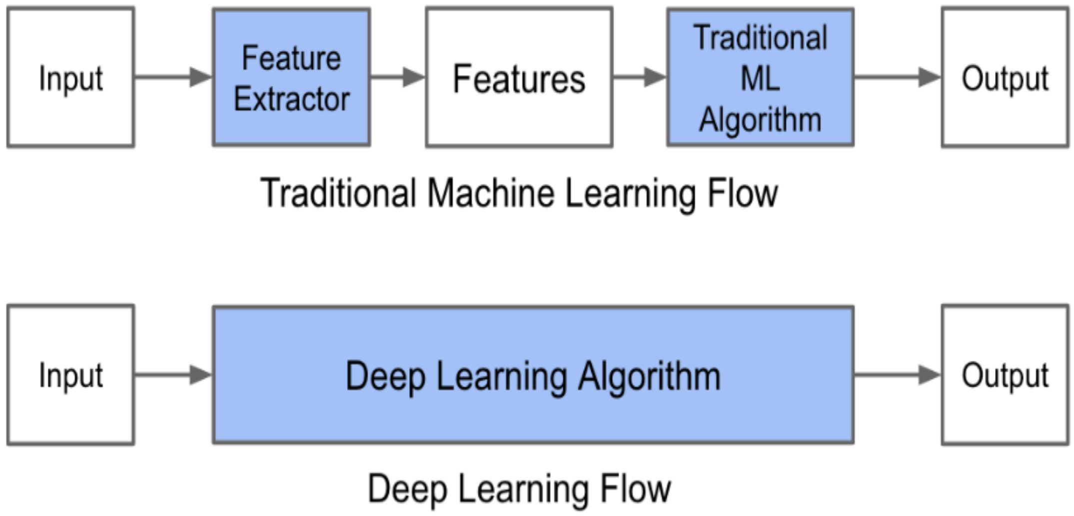
Harris Detector: Mathematics

Classification of image points using eigenvalues of M:

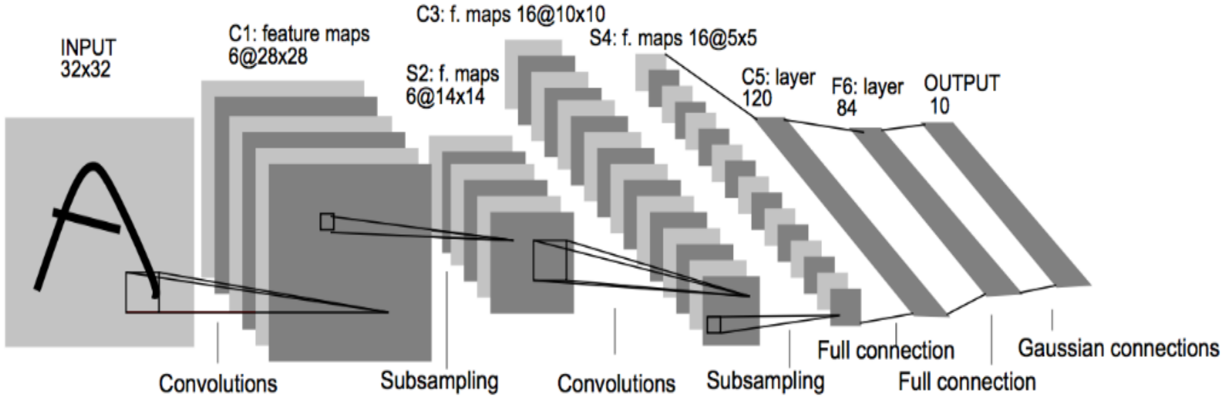


```
from matplotlib import pyplot as plt

from skimage import data

from skimage.feature import corner_harris, corner_subpix, corner_peaks

from skimage.transform import warp, AffineTransform

from skimage.draw import ellipse

tform = AffineTransform(scale=(1.3, 1.1), rotation=1, shear=0.7,   translation=(210, 50))

image = warp(data.checkerboard(), tform.inverse, output_shape=(350, 350))

rr, cc = ellipse(310, 175, 10, 100)

image[rr, cc] = 1

image[180:230, 10:60] = 1

image[230:280, 60:110] = 1

coords = corner_peaks(corner_harris(image), min_distance=5)

coords_subpix = corner_subpix(image, coords, window_size=13)

fig, ax = plt.subplots()

ax.imshow(image, interpolation='nearest', cmap=plt.cm.gray)

ax.plot(coords[:, 1], coords[:, 0], '.b', markersize=3)

ax.plot(coords_subpix[:, 1], coords_subpix[:, 0], '+r', markersize=15)

ax.axis((0, 350, 350, 0))

plt.show()
```

Deep Learning



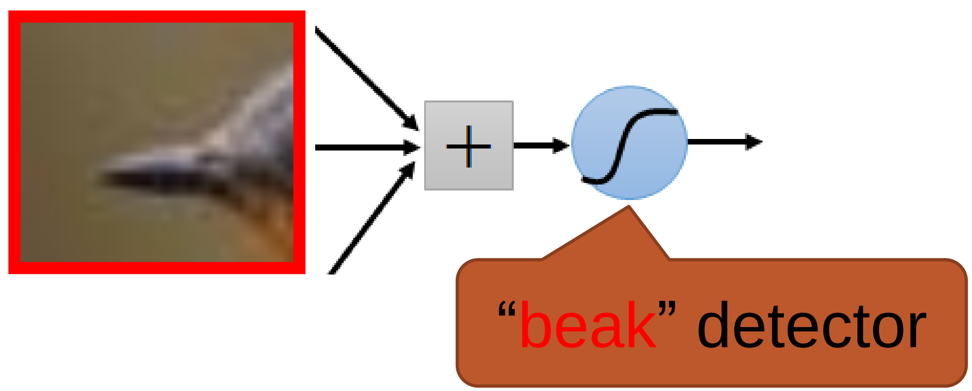
Feature Extraction by using Convolutional Neural Network-CNN



CNN called LeNet by Yann LeCun (1998)

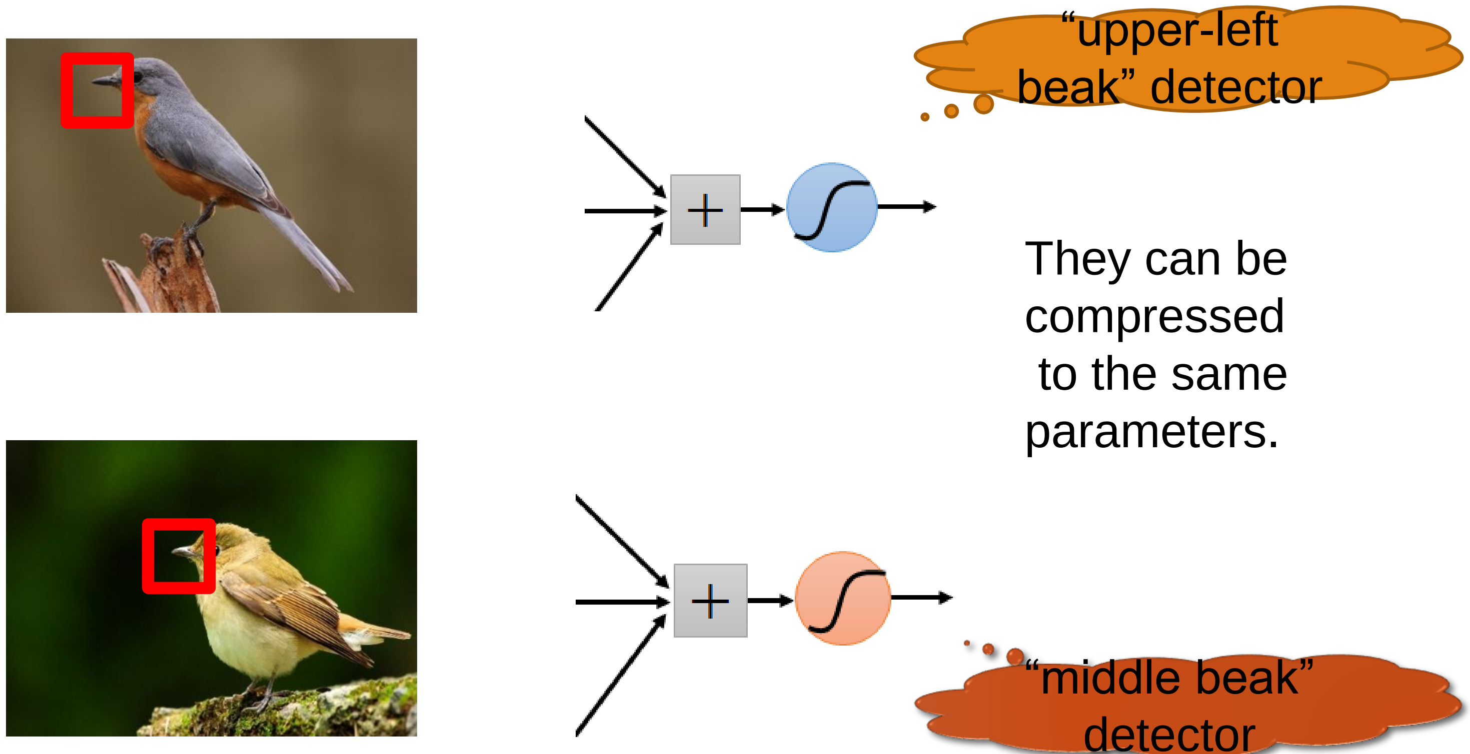
Convolutional Neural Networks-CNN

- ❑ Consider learning an image
- ❑ Some patterns are much smaller than the whole image



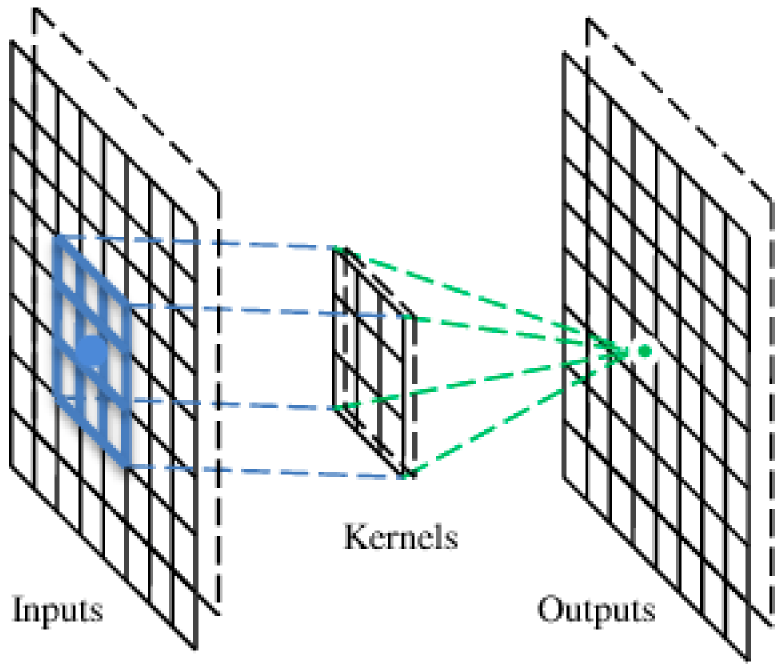
Convolutional Neural Networks-CNN

- Same pattern appears in different places: They can be compressed!
  What about training a lot of such "small" detectors and each detector must "move around".



Convolutional Neural Networks-CNN

A CNN is a neural network with some convolutional layers (and some other layers). A convolutional layer has a number of filters that does convolutional operation.



Convolution

These are the network parameters to be learned.

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

-1	1	-1
-1	1	-1
-1	1	-1

Filter 2

⋮ ⋮

Each filter detects a small pattern (3 x 3).

Convolution

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

stride=1

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

3 -1

Convolution

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

If stride=2

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

3 -3

Convolution

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

stride=1

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

3	-1	-3	-1
-3	1	0	-3
-3	-3	0	1
3	-2	-2	-1

Convolution

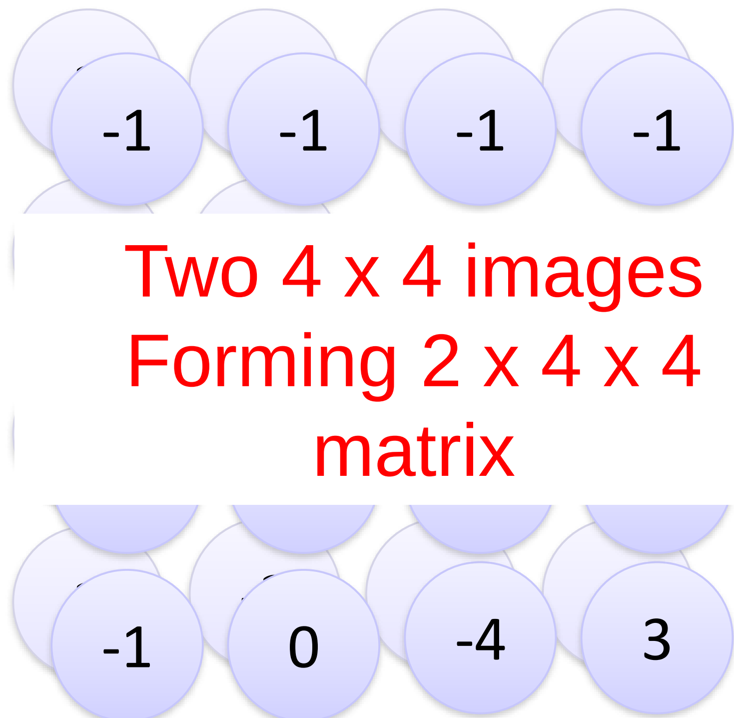
-1	1	-1
-1	1	-1
-1	1	-1

Filter 2

stride=1

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

Repeat this for each filter



Two 4 x 4 images
Forming 2 x 4 x 4 matrix

Convolution
Color image: RGB 3 channels

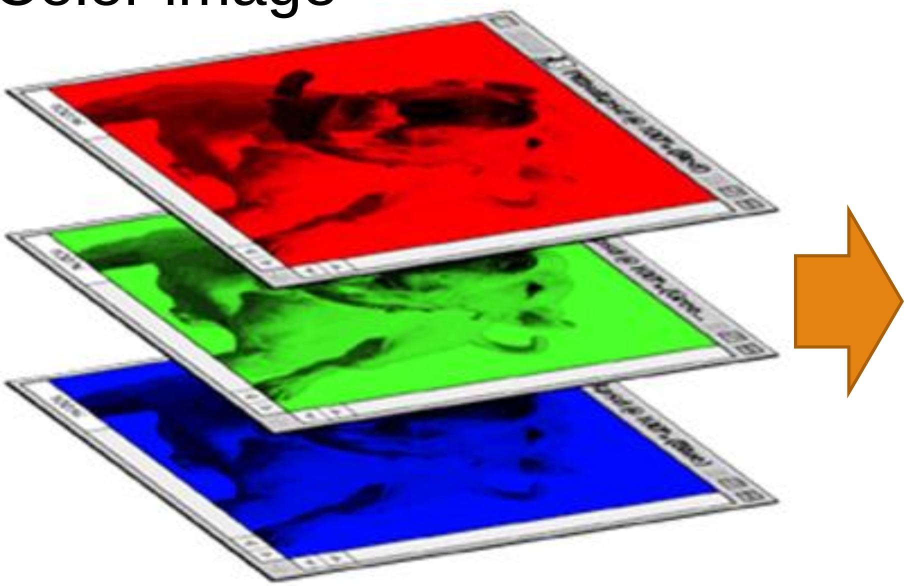
Filter 1

1	-1	-1
-1	1	-1
-1	-1	1

Filter 2

-1	1	-1
-1	1	-1
-1	1	-1

Color image



1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0