Pattern Recognition

LECTURE SLIDE # 7

Pattern

Patterns include repeated trends in various forms of data. A pattern can either be observed physically, for example, in images and videos, or it can be observed mathematically by applying statistical algorithms. In computer science, a pattern is represented using vector feature values.

- **Example:** The colors on the clothes, speech patterns, etc.

Pattern recognition is the process of recognizing patterns. Pattern recognition can be defined as the classification of data based on knowledge already gained or on statistical information extracted from patterns and/or their representation.

- **Examples:** Speech recognition, speaker identification, multimedia document recognition (MDR), and automatic medical diagnosis.

Pattern Recognition

Pattern recognition involves the classification and cluster of patterns.

In classification, an appropriate class label is assigned to a pattern based on an abstraction that is generated using a set of training patterns or domain knowledge. Classification is used in supervised learning.

Clustering generated a partition of the data which helps decision making, the specific decision-making activity of interest to us. Clustering is used in unsupervised learning.

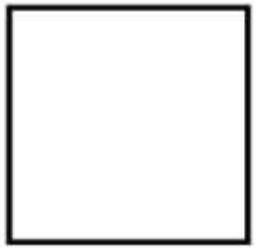
Pattern Recognition Approaches

- **Statistical.** This approach is based on statistical decision theory. Pattern recognizer extracts quantitative features from the data along with the multiple samples and compares those features. However, it does not touch upon how those features are related to each other.
- **Structural.** This approach is closer to how human perception works. It extracts morphological features from one data sample and checks how those are connected and related.
- **Neural.** In this approach, artificial neural networks are utilized. Compared to the ones mentioned above, it allows more flexibility in learning and is the closest to natural intelligence.

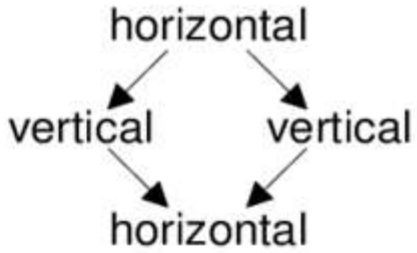
Pattern Recognition Approaches

Statistical

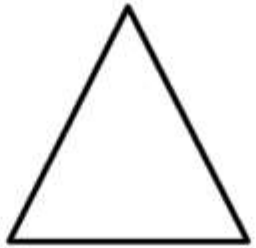
Number of segments: 4
Number of horizontal segments: 2
Number of vertical segments: 2
Number of diagonal segments: 0

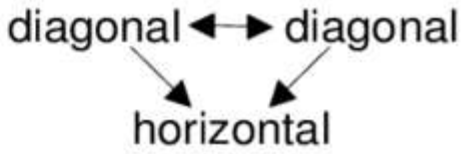


Structural

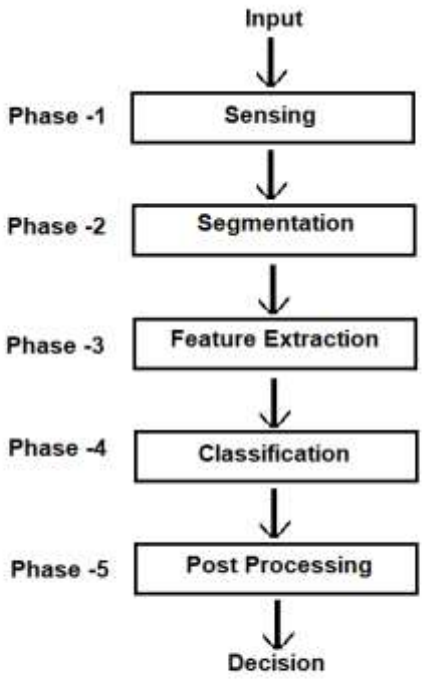


Number of segments: 3
Number of horizontal segments: 1
Number of vertical segments: 0
Number of diagonal segments: 2



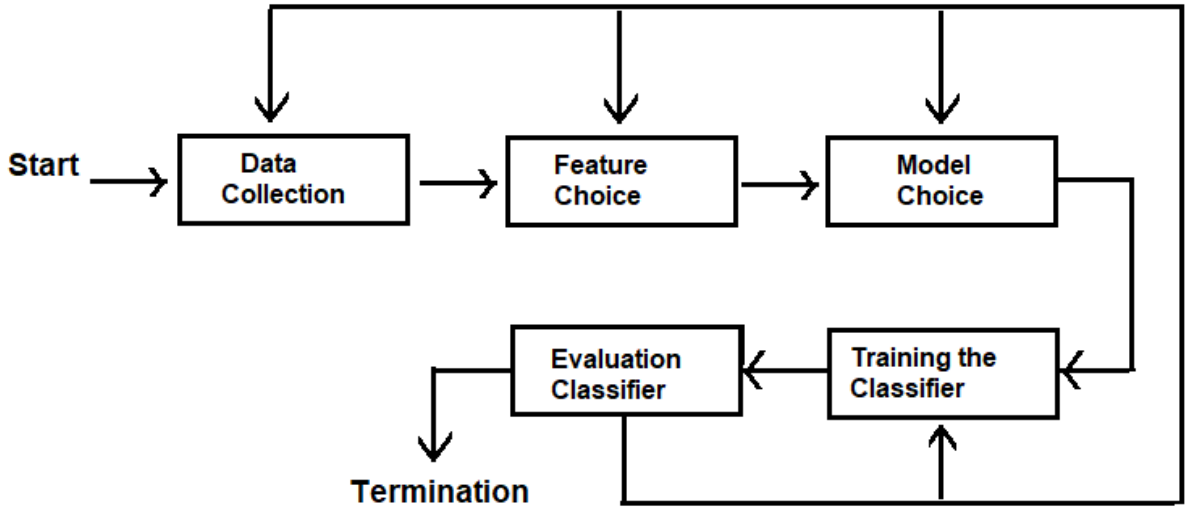


Phases in Pattern Recognition System



Phases in Pattern Recognition

Activities for designing the Pattern Recognition Systems



Activity Cycle

Features

Features may be represented as continuous, discrete, or discrete binary variables. A feature is a function of one or more measurements, computed so that it quantifies some significant characteristics of the object.

- **Example:** consider our face then eyes, ears, nose, etc are features of the face.
  A set of features that are taken together, forms the **features vector**.

Example: In the above example of a face, if all the features (eyes, ears, nose, etc) are taken together then the sequence is a feature vector([eyes, ears, nose]). The feature vector is the sequence of a feature represented as a ***d***-dimensional column vector.

A feature is a significant piece of information extracted from an image that provides a more detailed understanding of the image.

Features

A good feature set contains discriminating information, which can distinguish one object from other objects. It must be as robust as possible in order to prevent generating different feature codes for the objects in the same class.

Features can be classified into two categories:

- **Local features** are usually geometric (e.g. concave/convex parts, number of endpoints, branches, Joints, and corners)
- **Global features** are those features that are extracted from the overall character images. Global features are usually topological or statistical.

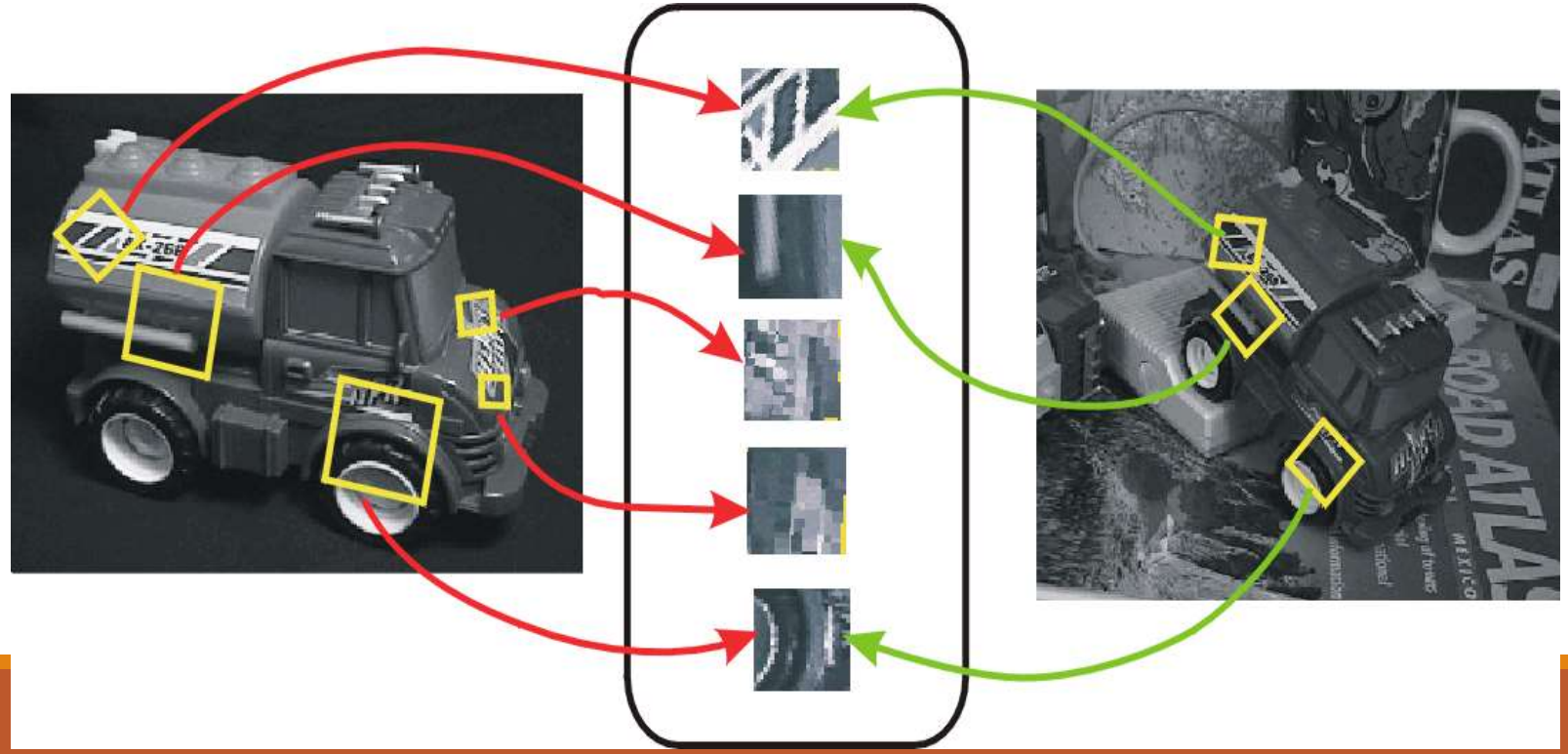
Global features describe the entire image, whereas local features describe the image patches (small group of pixels)

Feature Detection

Invariant local features

Find features that are invariant to transformations

- geometric invariance: translation, rotation, scale
- photometric invariance: brightness, exposure, …



Hough Transform

- The Hough Transform (HT) is a popular feature extraction technique that converts an image from Cartesian to polar coordinates
- You want some mechanism that gives more weightage to pixels that are already in a line. This is exactly what the Hough Transform does.
- In this transform image space (x, y) is transformed into a (ρ, θ) parameter space
- Successful detection of linear features using the HT requires pre-processing to threshold the input image into binary layers. The benefits of the HT are that it detects lines with some fragmentation and it is reasonably unaffected by random noise

Image and Parameter Spaces

Equation of Line: $y = mx + c$

Find: (m, c)

Consider point: (x_i, y_i)

$$y_i = mx_i + c \quad or \quad c = -x_i m + y_i$$

Parameter space also called Hough Space

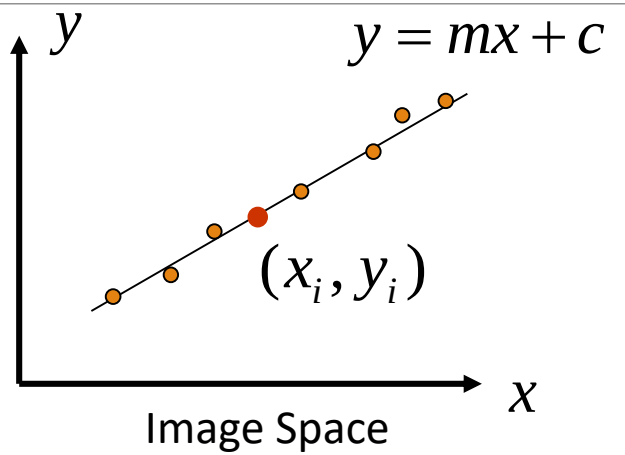
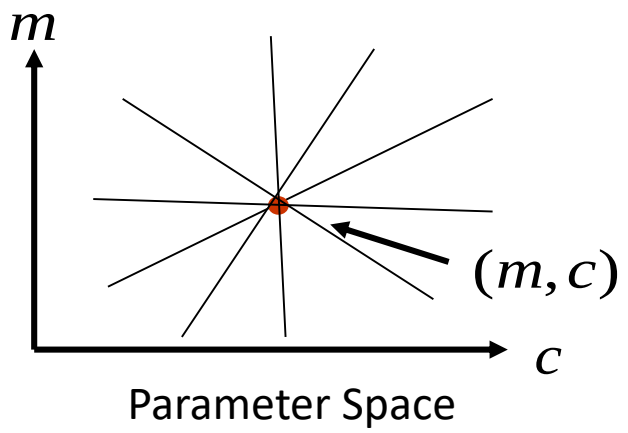


Image Space



Parameter Space

Alternative Parameter Space (Polar Coordinates)

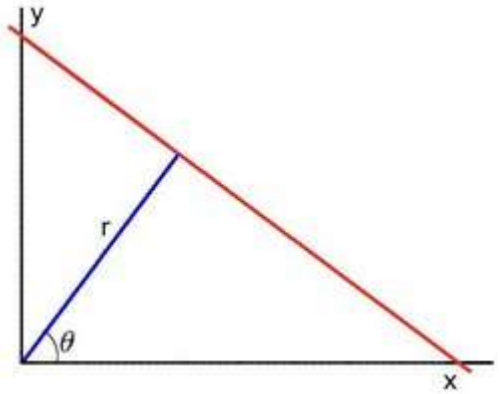
A line in the image space can be expressed with two variables. For example:

a.In the **Cartesian coordinate system:** Parameters: (m, c).

b.In the **Polar coordinate system:** Parameters: (r,θ)

$$y = \left(-\frac{\cos\theta}{\sin\theta}\right)x + \left(\frac{r}{\sin\theta}\right)$$

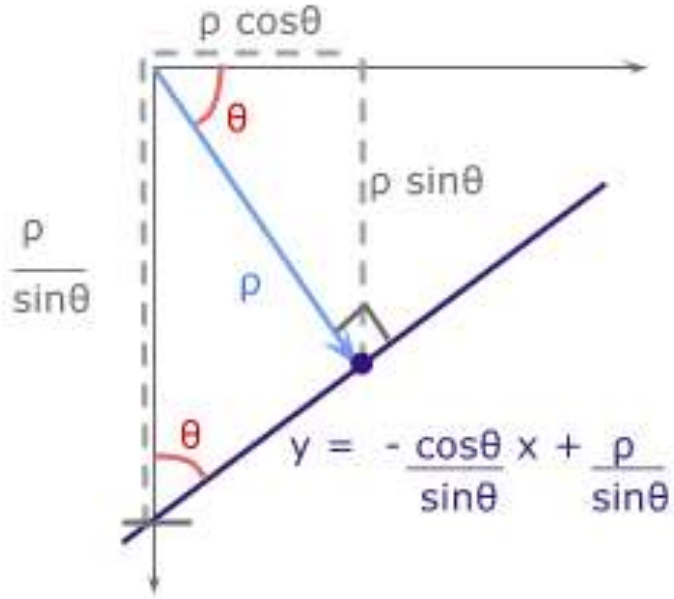
Arranging the terms: $r = x\cos\theta + y\sin\theta$



Alternative Parameter Space (Polar Coordinates)

$$y = \frac{-cos(\theta)}{sin(\theta)}x + \frac{\rho}{sin(\theta)}$$

$$\rho = xcos(\theta) + ysin(\theta)$$



Hough Transform Algorithm

Extract edge using any edge detection algorithm

1. Initialize H[r, θ]=0
2. for each edge point *I*[x, y] in the image

   for θ = 0 to 180

   - Calculate $r = xcos\Theta + ysin\Theta$

     H[r, θ] += 1
3. Find the value(s) of (r, θ) where H[r, θ] is maximum
4. The detected line in the image is given by $r = xcos\Theta + ysin\Theta$

What's the running time (measured in # votes)?

Extensions

Extension 1: Use the image gradient

1. same
2. for each edge point I[x,y] in the image

   compute unique (d, θ) based on image gradient at (x,y)

   H[d, θ] += 1
3. same
4. same

What's the running time measured in votes?

Extension 2

- give more votes for stronger edges

Extension 3

- change the sampling of (d, θ) to give more/less resolution

Extension 4

- The same procedure can be used with circles, squares, or any other shape

The Algorithm

- Decide on the range of ρ and θ. Often, the range of θ is [0, 180] degrees and *ρ* is [-*d*, *d*] where *d* is the length of the edge image's diagonal. It is important to quantize the range of ρ and θ meaning there should be a finite number of possible values.
- Create a 2D array called the accumulator representing the Hough Space with dimension (*num_rhos*, *num_thetas*) and initialize all its values to zero.
- Perform edge detection on the original image. This can be done with any edge detection algorithm of your choice.
- For every pixel on the edge image, check whether the pixel is an edge pixel. If it is an edge pixel, loop through all possible values of θ, calculate the corresponding ρ, find the θ and ρ index in the accumulator, and increment the accumulator base on those index pairs.
- Loop through all the values in the accumulator. If the value is larger than a certain threshold, get the ρ and θ index, get the value of ρ and θ from the index pair which can then be converted back to the form of $y = ax + b$.