8. Automatic Content Analysis

Automatic Content Analysis - What for?

The first generation of multimedia computers could only transmit digital video and audio streams and play them out on a device (monitor, speakers). Due to increased performance, current multimedia computers allow the **processing** of multimedia streams in real-time.

An expanding field of research is the **automatic content analysis** of audio and video streams. The computer tries to find out as much as possible about the content of a video. Application examples are:

- the automatic indexing of large video archives, e.g., at broadcasting stations,
- “query by image example” in a personal photo collection,
- the automatic filtering of raw video material,
- the automatic generation of video abstracts,
- the analysis of style characteristics in artistic film research.

The problem is called ”**bridging the semantical gap”**.

8.1 Statistics for Multimedia Content Analysis

MM content analysis intends to extract semantic information out of MM signals, in particular video or audio.

If we have a good understanding of a phenomenon we can hope to derive a **mathematical function** that relates the signal (or any kind of input) to specific semantics in a unique way. For example, the exact stimulation of the nerves in the human ear (the hair cells in the cochlea) by an audio signal is well known and can be expressed as a mathematical function.

Unfortunately, similar cases of functional descriptions are rare in MM content analysis. An ASCII transcription (the content) of a spoken text (the audio signal) is not perfect, there is no mathematical mapping, the transcription is a heuristic guess.

Statistics instead of Mathematical Functions

It is not well understood how humans see, hear and semantically interpret the physical phenomena. Until we have a deeper understanding of human perception, we can at least find **correlations** between MM signals and their meaning.

The attempt to extract information out of phenomena in spite of a vague understanding of the relationship is not unique to MM analysis. It is also popular in economics, psychology, biology and other sciences.

The following example is an instance of a statistical method that fits a line to a set of correlated edge pixels of an object in an image.

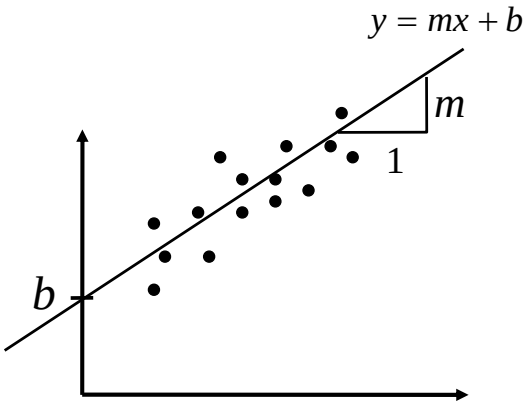
Finding a Line in an Image

If we assume that a set of points (x_i, y_i) forms a line we can use simple linear regression for fitting the optimal line:

Remember from your statistics lecture: The slope m and the offset b can be computed from all points (x_i, y_i):

$$m = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

$$b = \bar{y} - m\bar{x}$$



$\bar{x}$ = average value of all x_i

$\bar{y}$ = average value of all y_i

Straight Lines in Images

But how can we detect an unknown number of lines? Linear regression is only useful if we can assume that the entire set of points belongs to a *single* line.

Let us now assume that we have a set of edge pixels that form an **unknown** number of lines. For example, we want to find the lines of a tennis court in an image. To make matters worse, lines in real-world images are not perfect. They suffer from **outliers**, single **uncorrelated edge pixels,** and may be **interrupted**.

We will have to state more precisely what constraints have to be fulfilled for edge pixels to form a line.

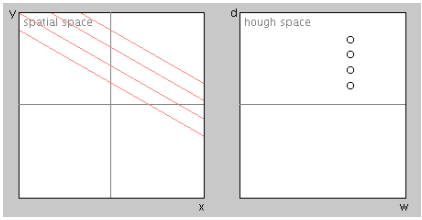
The Hough Transform

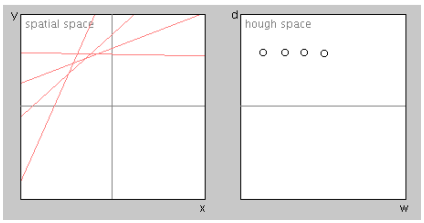
The Hough Transform can help us to solve this problem.

In the Hough space, an entire line is defined by a single point where

- the vertical axis defines the distance of the line from the origin,
- the horizontal axis defines the angle of the line with the x-axis.

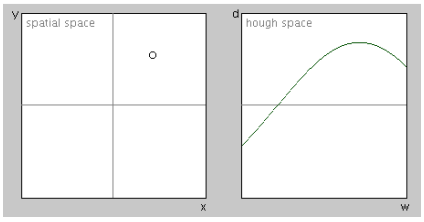
Examples: lines the spatial domain and the corresponding points in the Hough space



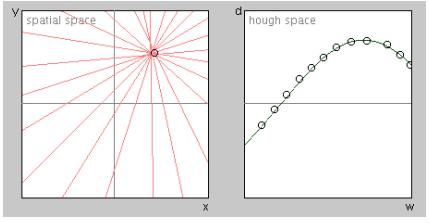


Points and Lines in under the Hough Transform

How is a point in the spatial domain transformed into the Hough space?



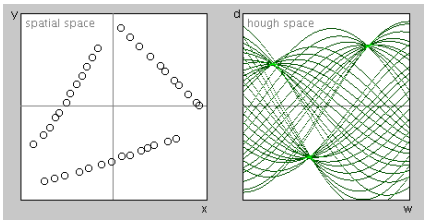
The point in the spatial domain (in the left part of the figure) does not define a line by itself. Thus we assume that any line passing through the point is a candidate line. All candidate lines correspond to the sine-shaped trajectory in the Hough space.



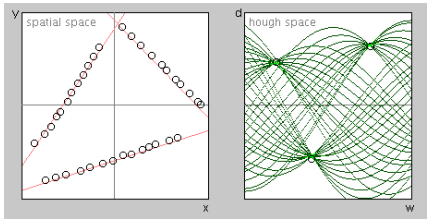
Taking more samples in the Hough space makes the issue clearer. The set of corresponding lines in the image space is shown on the left.

How To Find an Unknown Number of Lines

Let us now consider three roughly defined lines in the spatial domain. Each point defines a sine-shaped trajectory in the Hough space. All trajectories obviously meet in three distinct locations.



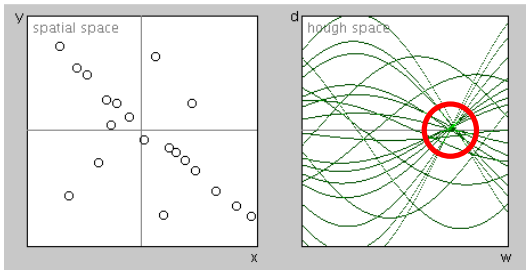
And in fact: If we mark each intersection in the Hough space with a point we get an approximation for the actual lines (shown in red).

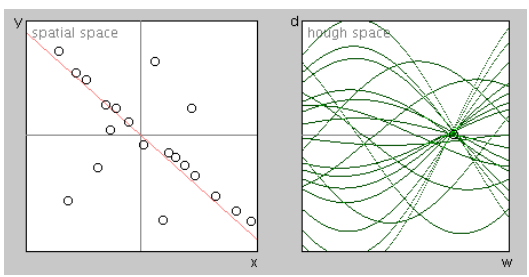


How To Define a Line

What do we consider a line?

Possible solution: If we encounter a predefined minimal line density within a local neighborhood in the Hough space (the red circle on the right) we define its center of gravity as the representing line.





Note that none of the points in the spatial domain necessarily touches the approximated line. Just as the center of gravity does not always lie in an object itself.

Clustering Algorithms

In more general terms: Rather than finding lines we might want to find the centers of distinct clusters. A well-known algorithm is the **k-means clustering algorithm**.

Assumption: We have a set of points, and we assume that the points form k clusters.

Problem: Find the centriod of each cluster.

Remark: This is very typical for features of images, such as color content, number of pixels on edges, etc. Clusters of points in the n-dimensional feature space correspond to similar images.

The k-means Clustering Algorithm

The K-means clustering algorithm

(1) Determine the number of clusters you expect

(2) Set the cluster centers anywhere within the feature space

Take care that the centers do not conglomerate in the first place. Their mutual distance can be arbitrarily large.

(3) Assign each point of the feature space to the nearest cluster.

(4) For each cluster, compute the center of the associated points

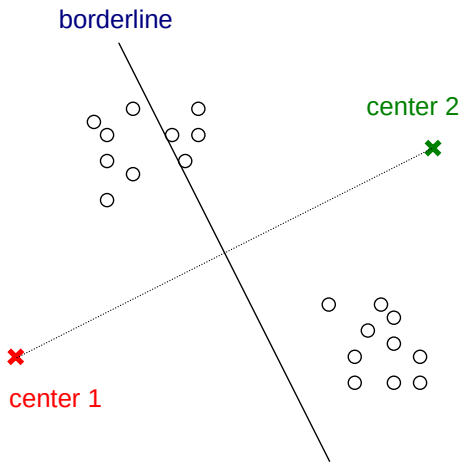
(5) Goto (3) until all centers have settled

Example of k-means Clustering

Example: We want to determine the two centers which are defined by 2D feature vectors (points on the right). Obviously the initial predictions for the two centers, marked by **x**, are not very good.

The borderline in the middle partitions the feature space into the two subspaces belonging to each initial center.

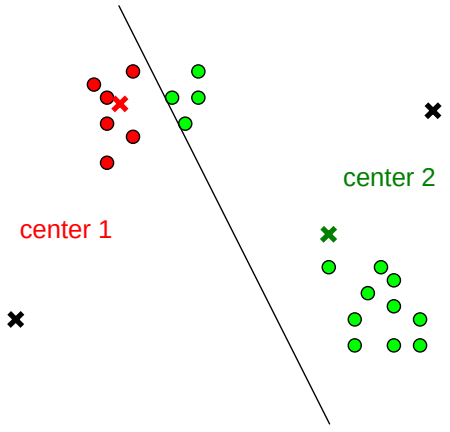
For k = 2: The borderline is perpendicular to the line between center 1 and center 2.



Example of k-means Clustering: Step 2

Each feature point is associated with the initial center 1 or center 2 (shown in black). The new centers of the red and the green cluster are then computed (shown in red and green).

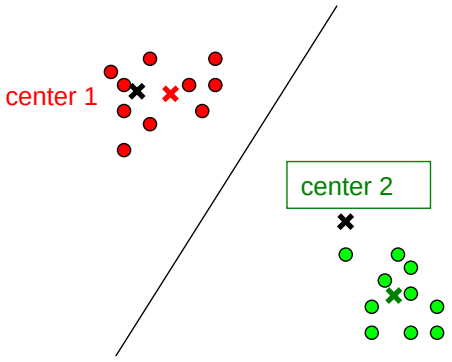
Though not yet perfect, the new centers have moved into the right direction.



Example of k-means Clustering: Step 3

Again, each feature point is associated with the nearest current center (shown in black).

The newly determined centers (shown in red and green) do not influence the borderline in such a way that a point would change its center. Thus the final state of the algorithm is reached with the red and green centers, as marked.

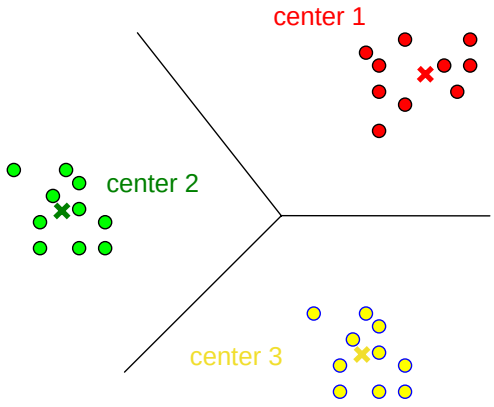


Clustering for Three Centers

Clustering for k > 2 works in an analog fashion. The well-known **Voronoi diagram** (lines of equal distances to the closest centers) partitions the feature space.

Remarks:

- Convergence of the algorithm is usually quick, even for higher-dimensional feature spaces.
- k has to be known in advance.
- The algorithm can run into a local minimum.
- The outcome is not always predictable (think of three actual centers and k = 2).



Conclusion

- Mathematical transformations (such as the Hough transform, the Fourier transform or the Wavelet transform) are often useful to *discover structure* in spatial and temporal phenomena.
- Statistical methods are useful to derive correlations between physical-level parameters and higher-level semantics.
- The k-means clustering algorithm is useful to detect similarity between images or image objects in multi-dimensional feature spaces.