Computer Vision (7CCSMCVI / 6CCS3COV)

Recap

- **Image formation**
- **Low-level vision**
- **Mid-level vision**
  - grouping and segmentation of image elements
  - Biological
    - bottom-up influences (Gestalt cues)
    - top-down influences (knowledge, expectation, etc.)
  - Artificial ←Today
- **High-level vision**

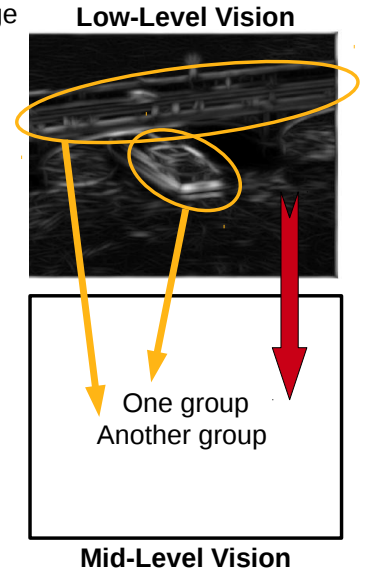
Today

group together those elements of an image that "belong together", and

segment these elements from all others.

Methods:

- Thresholding
  - morphological operators
- Region-based
  - region growing
  - region merging
  - split and merge
- Clustering
  - k-means clustering
  - hierarchical clustering
  - graph cutting
- Fitting
  - Hough transform
  - active contours



Features

Which elements "belong together"? (i.e. lie on the same object)

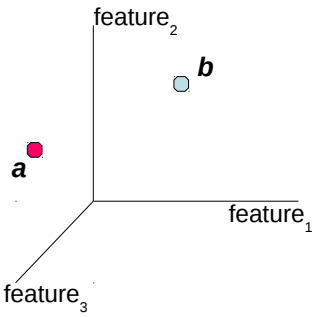
In computer vision many features are used (individually or in combination):

- location (= Gestalt law of proximity)
- colour / intensity (= Gestalt law of similarity)
- texture (= Gestalt law of similarity)
- size (= Gestalt law of similarity)
- depth
- motion (= Gestalt law of common fate)
- not separated by contour (= Gestalt law of common region)
- form a known shape when assembled (top-down)

Which feature(s) work best will depend on task

Feature Space

We can think of each image element being a point in feature space.
Similarity (and hence grouping) will be determined by the distance between points in this feature space.



a and ***b*** are vectors of feature values for two elements

$\boldsymbol{a}=(a_1,a_2,a_3)$

$\boldsymbol{b}=(b_1,b_2,b_3)$

Feature Space: similarity measures

We can think of each image element being a point in feature space.
Similarity (and hence grouping) will be determined by the distance between points in this feature space.

Similarity can be measured as:

Affinity= $\exp\left(\frac{-(distance^2)}{2\sigma^2}\right)$

Cross-correlation= $\sum_i a_i b_i$

Normalised Cross-correlation (NCC)= $\frac{\sum_i a_i b_i}{\sqrt{(\sum_i a_i^2)}\sqrt{(\sum_i b_i^2)}}$

Correlation coefficient= $\frac{\sum_i (a_i-\bar{a})(b_i-\bar{b})}{\sqrt{(\sum_i (a_i-\bar{a})^2)}\sqrt{(\sum_i (b_i-\bar{b})^2)}}$

Feature Space: similarity measures

We can think of each image element being a point in feature space.
Similarity (and hence grouping) will be determined by the distance between points in this feature space.

Alternatively, distance can be measured as:

Euclidean distance = $\sqrt{\sum_i^n (a_i-b_i)^2}$

Sum of Squared Differences (SSD)= $\sum_i^n (a_i-b_i)^2$

Sum of Absolute Differences (SAD)= $\sum_i^n |a_i-b_i|$

Other methods of calculating distance and similarity exist, and the choice of function will affect the segmentation that is produced.

Feature Space: feature scaling

Different features might be weighted differently in the calculation of distance/similarity:

e.g. for Euclidean distance:

$$\sqrt{w_1(a_1-b_1)^2+w_2(a_2-b_2)^2+w_3(a_3-b_3)^2+w_4(a_4-b_4)^2}$$

where w_1, ..., w_4 are weights that set relative importance of parameters

determining the weights that yield the best performance is a non-trivial task.

Feature Space: feature scaling

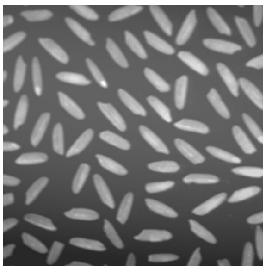
Features of different types may have different scales

e.g., pixel coordinates on a 2000x2000 pixel image vs. colour values in the range [0,1].

Problem: Features with smaller scales will appear more similar.

Solution: Scale the features to be in the same range.

Thresholding

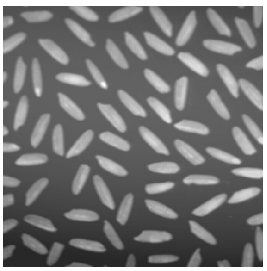


Regions defined by differences in brightness (could be colour, or output of some filtering process).

The feature space is 1-dimensional.

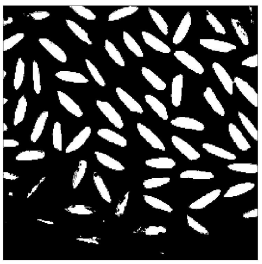
Can segment by thresholding, i.e.
I'(x,y) = 1, if I(x,y)>thres
I'(x,y) = 0, otherwise

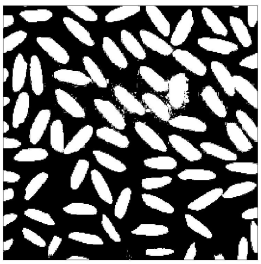
Thresholding



This results in two groups, foreground and background so is an example of figure-ground segmentation.

How to choose threshold?





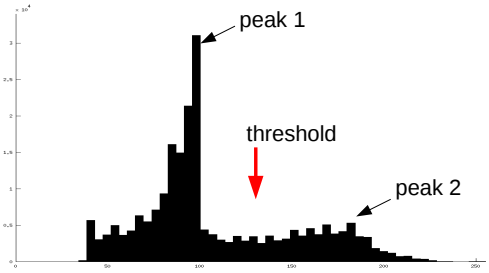
150 125 100

Thresholding: choosing the threshold

1. Use average intensity
2. Plot intensity histogram.

   Find peaks

   Assuming histogram is bimodal, place threshold in between the peaks



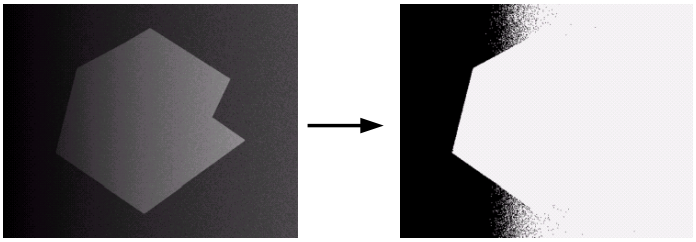
histogram for image on previous slide

3. Hysteresis thresholding. Define two thresholds (e.g. one each side of valley in histogram)
   - Pixels above the high threshold are classified as figure and below the low threshold as background
   - Pixels between the low and high thresholds are classified as figure only **if** they are adjacent to other figure pixels
4. Set threshold so that a certain fraction of image is figure (if fraction of image occupied by objects is known)

Thresholding: choosing the threshold

A single threshold is rarely appropriate for a whole image.

Especially, when we have uneven illumination due to shadows or due to the direction of illumination.



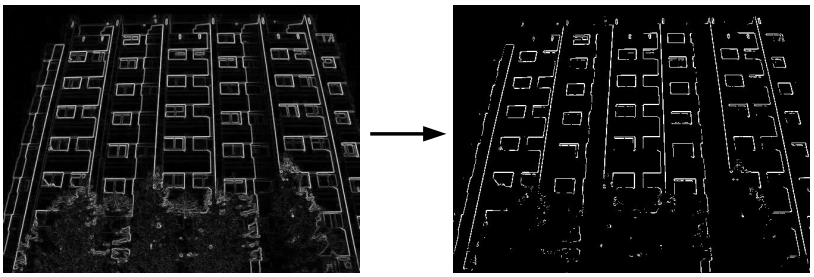
It is the local contrast between the object and the background that is important

Local thresholding / block thresholding: image split into rectangular blocks and threshold applied to each block

Thresholding

Thresholding might be applied after the image has been processed in some way (preprocessing defines feature space).

e.g. to segment all the edges from everything else:



Thresholding often results in

missing parts in the figure (e.g. edges with missing pixels)

unwanted parts in the figure (e.g. extra pixels that are not part of any edge)

Morphological Operations

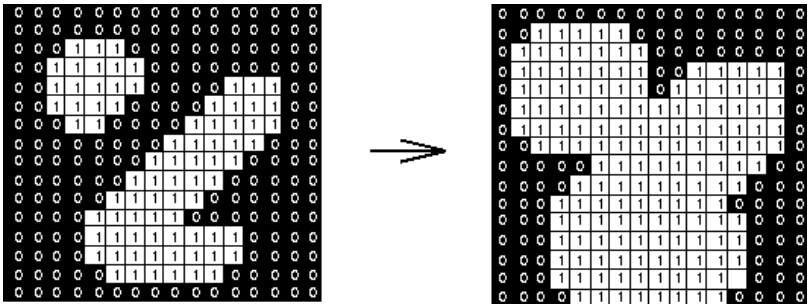
Used to clean up the results of thresholding

Dilation expands the area of foreground pixels (1s) in a binary image (e.g. to fill holes and gaps)

All background pixels that neighbour a foreground pixel are changed from 0 to 1

Neighbourhood defined by a “structuring element”

e.g. a 3x3 pixel square structuring element defines a neighbourhood where each pixel’s neighbours are the 8 adjacent pixels horizontally, vertically and diagonally



Morphological Operations

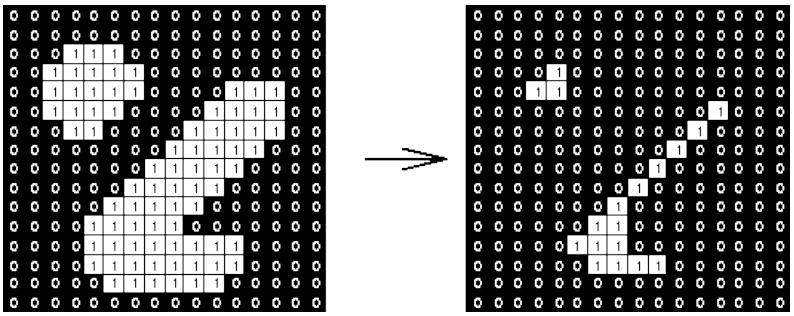
Used to clean up the results of thresholding

Erosion shrinks the area of foreground pixels (1s) in a binary image (e.g. to remove bridges, branches and small protrusions)

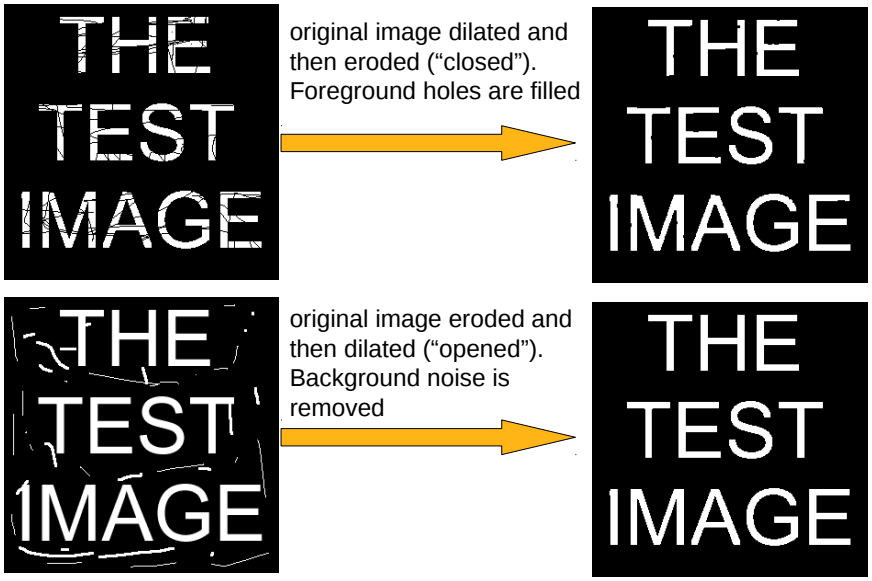
All foreground pixels that **neighbour** a background pixel are changed from 1 to 0

Neighbourhood defined by a “structuring element”

e.g. a 3x3 pixel square structuring element defines a neighbourhood where each pixel’s neighbours are the 8 adjacent pixels horizontally, vertically and diagonally



Morphological Operations



Region-based segmentation

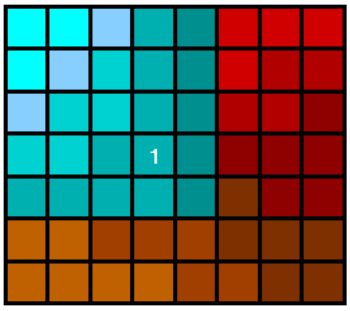
A fundamental drawback of thresholding is that it does not take into account spatial information.

Region based segmentation methods take into account the location of each pixel as well as its intensity, or colour, or texture (or other features or combination of features that are being used to define similarity).

The feature space can be multi-dimensional (following example has 3-d feature space: colour)

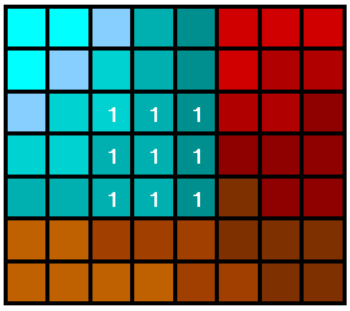
Region growing

- Start with one "seed" pixel, chosen arbitrarily
  - Give this pixel a label (defining the region it belongs to)
  - Examine all the unlabelled pixels neighbouring labelled pixels
  - If they are within similarity threshold, give them the same region label
- Repeat until region stops growing, then choose another seed pixel which does not yet belong to any region and start again.
- Repeat the whole process until all pixels have been assigned to a region.



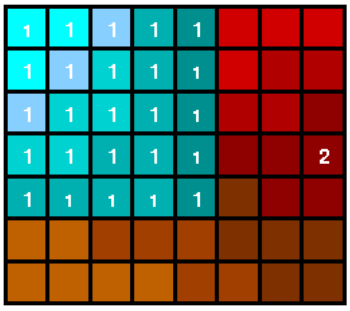
Region growing

- Start with one “seed” pixel, chosen arbitrarily
  - Give this pixel a label (defining the region it belongs to)
  - Examine all the unlabelled pixels neighbouring labelled pixels
  - If they are within similarity threshold, give them the same region label
- Repeat until region stops growing, then choose another seed pixel which does not yet belong to any region and start again.
- Repeat the whole process until all pixels have been assigned to a region.



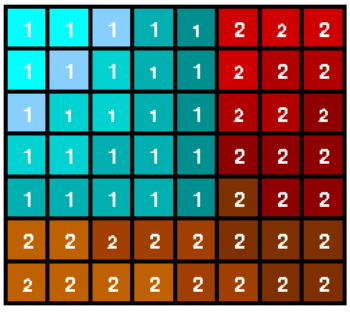
Region growing

- Start with one “seed” pixel, chosen arbitrarily
  - Give this pixel a label (defining the region it belongs to)
  - Examine all the unlabelled pixels neighbouring labelled pixels
  - If they are within similarity threshold, give them the same region label
- Repeat until region stops growing, then choose another seed pixel which does not yet belong to any region and start again.
- Repeat the whole process until all pixels have been assigned to a region.



Region growing

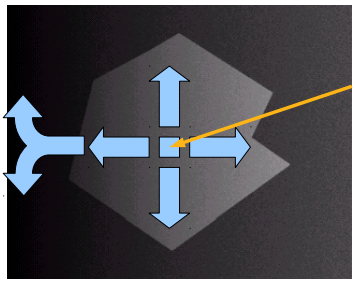
- Start with one “seed” pixel, chosen arbitrarily
  - Give this pixel a label (defining the region it belongs to)
  - Examine all the unlabelled pixels neighbouring labelled pixels
  - If they are within similarity threshold, give them the same region label
- Repeat until region stops growing, then choose another seed pixel which does not yet belong to any region and start again.
- Repeat the whole process until all pixels have been assigned to a region.



Region growing

If similarity is based on intensity, then regions should stop growing at discontinuities in intensity, i.e. edges.

However, region growth may "leak" through a single weak spot in the boundary



Region merging

- Start with each pixel (or small square regions of pixels, e.g. 2x2, or 4x4 pixels) being labelled as separate regions.
  - A region's properties are compared with those of an adjacent region
  - If they match, they are merged into a larger region and the properties of the new region are computed.
- Continue merging of adjacent regions until a region cannot be merged with any of its neighbours, it is marked `final'.
- The whole process repeats until all image regions are marked final.

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56

Region merging

- Start with each pixel (or small square regions of pixels, e.g. 2x2, or 4x4 pixels) being labelled as separate regions.
  - A region's properties are compared with those of an adjacent region
  - If they match, they are merged into a larger region and the properties of the new region are computed.
- Continue merging of adjacent regions until a region cannot be merged with any of its neighbours, it is marked `final'.
- The whole process repeats until all image regions are marked final.

1	1	3	4	5	6	7	8
1	1	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56

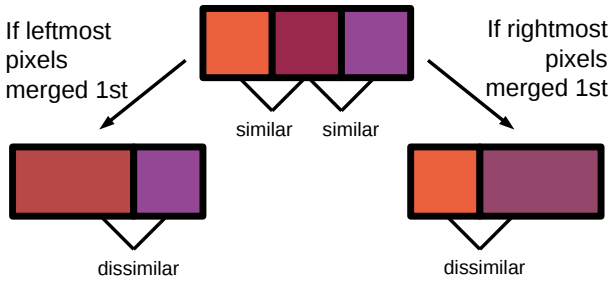
Region merging

- Start with each pixel (or small square regions of pixels, e.g. 2x2, or 4x4 pixels) being labelled as separate regions.
  - A region's properties are compared with those of an adjacent region
  - If they match, they are merged into a larger region and the properties of the new region are computed.
- Continue merging of adjacent regions until a region cannot be merged with any of its neighbours, it is marked `final'.
- The whole process repeats until all image regions are marked final.

Region merging

The result of region merging usually depends on the order in which regions are merged.

Due to the properties of the merged region being the average of the properties of the constituent parts.

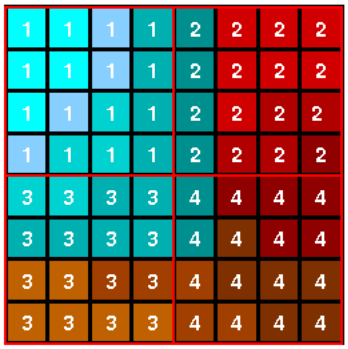


Region splitting and merging

- Start with the whole image labelled as a single region
  - For each region:
    - If all pixels are not similar, split the four quadrants into different regions. Continue until each region is homogeneous.
  - For each region:
    - Compare to neighbours and merge neighbouring regions which are similar. Continue until no more regions can merge.

Region splitting and merging

- Start with the whole image labelled as a single region
  - For each region:
    - If all pixels are not similar, split the four quadrants into different regions. Continue until each region is homogeneous.
  - For each region:
    - Compare to neighbours and merge neighbouring regions which are similar. Continue until no more regions can merge.



Region splitting and merging

- Start with the whole image labelled as a single region
  - For each region:
    - If all pixels are not similar, split the four quadrants into different regions. Continue until each region is homogeneous.
  - For each region:
    - Compare to neighbours and merge neighbouring regions which are similar. Continue until no more regions can merge.

Region splitting and merging

- Start with the whole image labelled as a single region
  - For each region:
    - If all pixels are not similar, split the four quadrants into different regions. Continue until each region is homogeneous.
  - For each region:
    - Compare to neighbours and merge neighbouring regions which are similar. Continue until no more regions can merge.

Region-based segmentation

General problems with region-based methods

"Meaningful" regions may not be uniform: the brightness and colour of a surface will vary due to lighting effects or curvature.

It is very unusual in practice for an image to be composed of uniform regions of similar intensity, or colour, or texture etc.

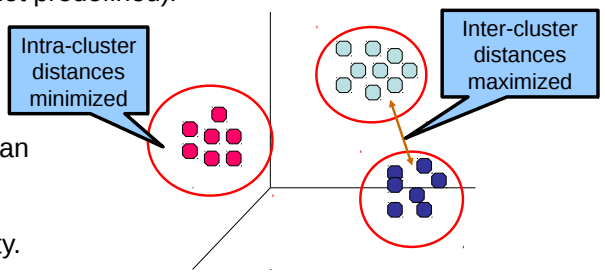
Example of image segmentation by region growing using colour and texture similarity.

Different colours represent different region labels.

Clustering

A general class of methods for unsupervised classification of data (i.e. classes are not predefined).

These methods can be used to group image elements based on similarity.



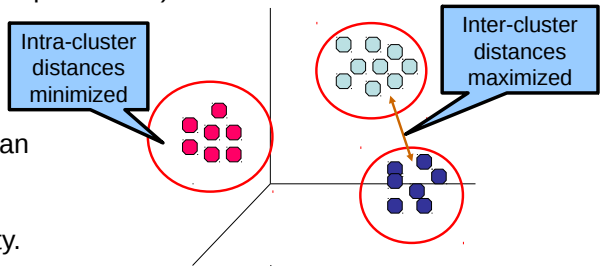
Two main sub-classes of algorithm:

- Partitional Clustering
  - data divided into non-overlapping subsets (clusters) such that each data element is in exactly one subset
- Hierarchical clustering
  - A set of nested clusters organized as a hierarchical tree

Clustering

A general class of methods for unsupervised classification of data (i.e. classes are not predefined).

These methods can be used to group image elements based on similarity.



The feature space is multi-dimensional (following examples have 2-d feature space)

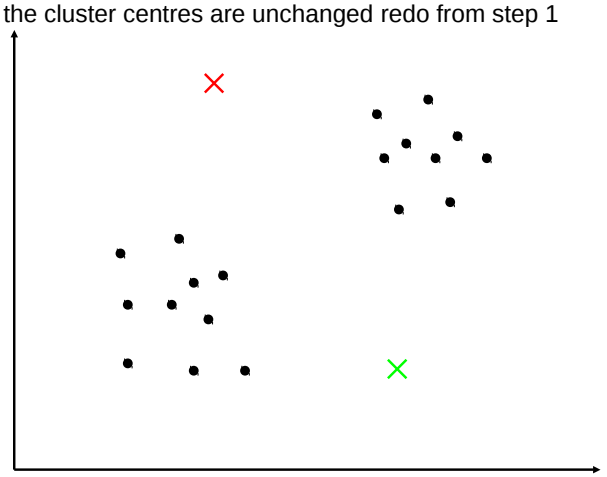
K-means clustering

A simple Partitional Clustering algorithm which assumes that there are k clusters (k is a parameter input to the algorithm)

0. Randomly choose k points to act as cluster centres
1. Allocate each element to the cluster with the closest centre
2. Compute new cluster centres as the mean position of the elements in each cluster
3. Until the cluster centres are unchanged redo from step 1

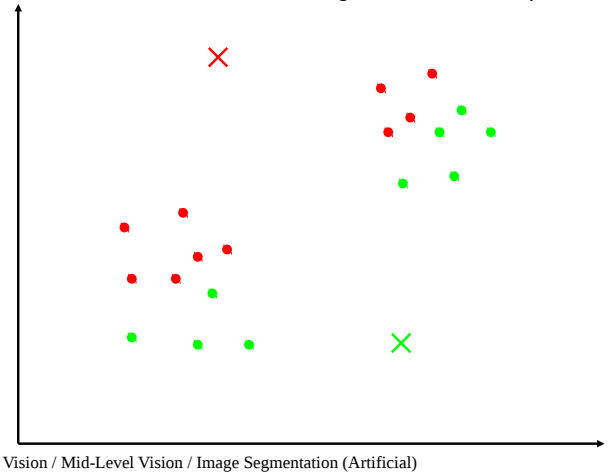
K-means clustering: example

0. Randomly choose k points to act as cluster centres
1. Allocate each element to the cluster with the closest centre
2. Compute new cluster centres as the mean position of the elements in each cluster
3. Until the cluster centres are unchanged redo from step 1



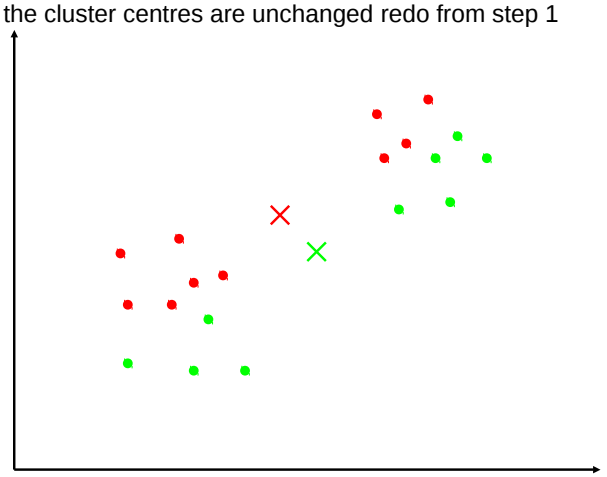
K-means clustering: example

0. Randomly choose k points to act as cluster centres
1. Allocate each element to the cluster with the closest centre
2. Compute new cluster centres as the mean position of the elements in each cluster
3. Until the cluster centres are unchanged redo from step 1



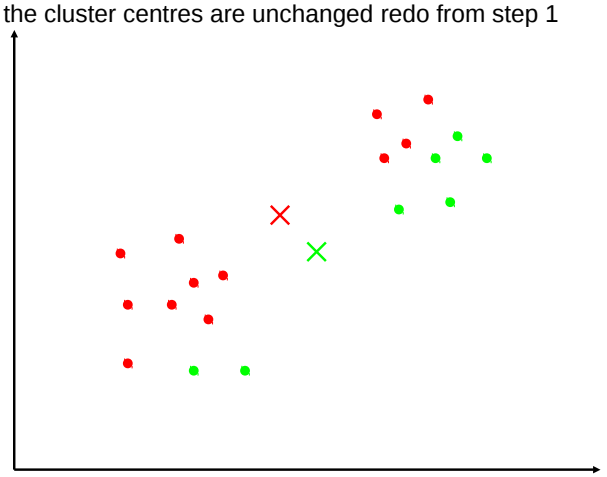
K-means clustering: example

0. Randomly choose k points to act as cluster centres
1. Allocate each element to the cluster with the closest centre
2. Compute new cluster centres as the mean position of the elements in each cluster
3. Until the cluster centres are unchanged redo from step 1



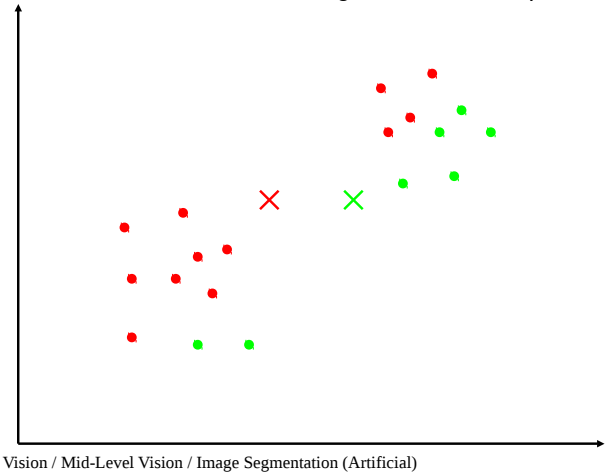
K-means clustering: example

0. Randomly choose k points to act as cluster centres
1. Allocate each element to the cluster with the closest centre
2. Compute new cluster centres as the mean position of the elements in each cluster
3. Until the cluster centres are unchanged redo from step 1



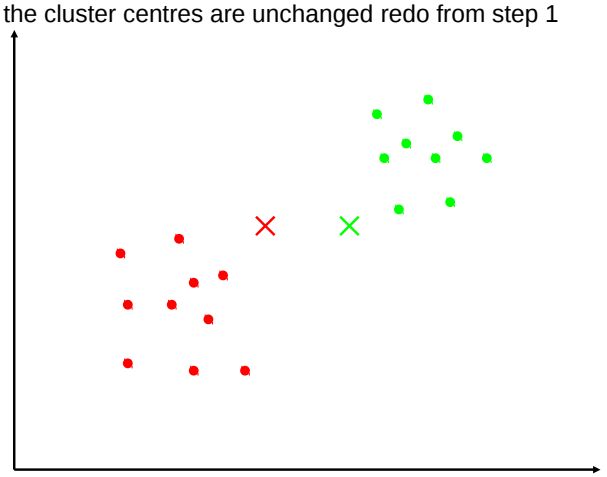
K-means clustering: example

0. Randomly choose k points to act as cluster centres
1. Allocate each element to the cluster with the closest centre
2. Compute new cluster centres as the mean position of the elements in each cluster
3. Until the cluster centres are unchanged redo from step 1



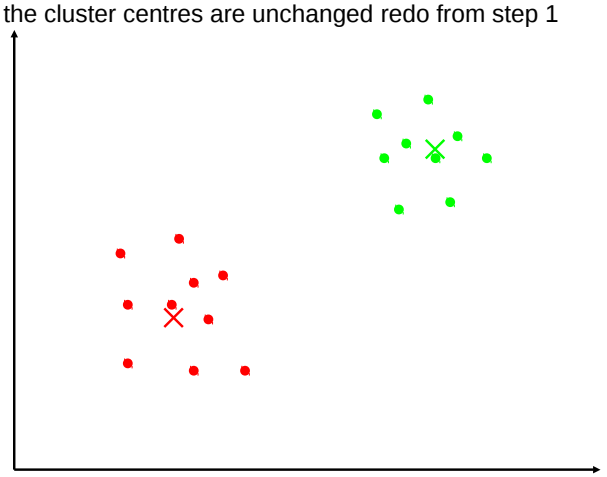
K-means clustering: example

0. Randomly choose k points to act as cluster centres
1. Allocate each element to the cluster with the closest centre
2. Compute new cluster centres as the mean position of the elements in each cluster
3. Until the cluster centres are unchanged redo from step 1



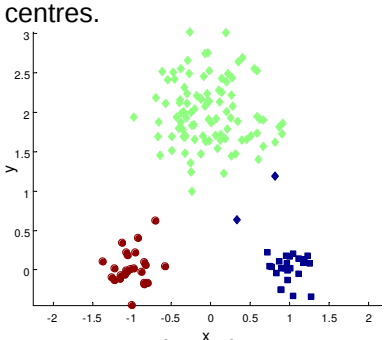
K-means clustering: example

0. Randomly choose k points to act as cluster centres
1. Allocate each element to the cluster with the closest centre
2. Compute new cluster centres as the mean position of the elements in each cluster
3. Until the cluster centres are unchanged redo from step 1

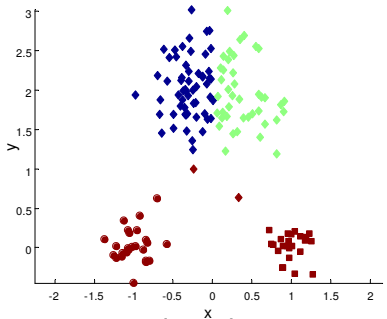


K-means clustering: problems

Results may differ depending on the location of the original cluster centres.



3-means clustering, run 1

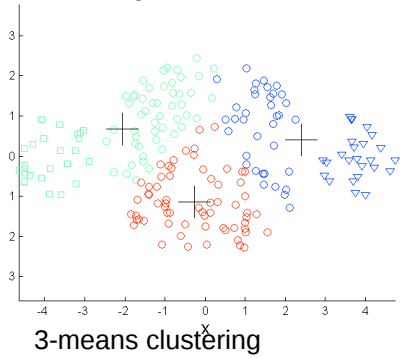


3-means clustering, run 2

K-means clustering: problems

Clustering is poor if true clusters are of differing sizes.

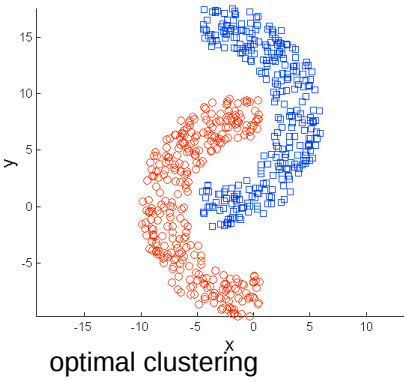
optimal clustering



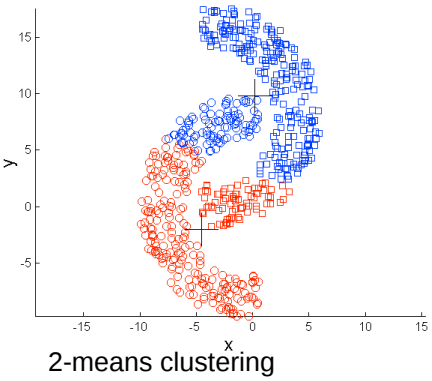
3-means clustering

K-means clustering: problems

Clustering is poor if true clusters are not “globular”.



optimal clustering



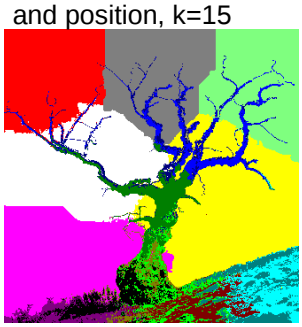
2-means clustering

K-means clustering: results

Original image

Clustering using colour alone, k=15

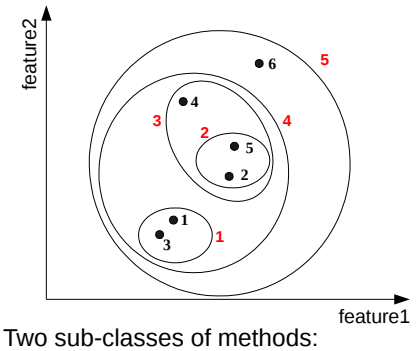
Clustering using colour and position, k=15



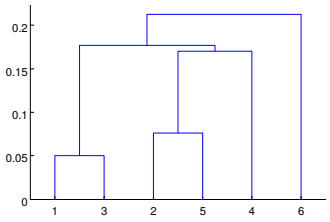
Different colours represent different region labels.

Hierarchical Clustering

Produces a set of nested clusters organized as a hierarchical tree



Can be visualized as a dendrogram:



Two sub-classes of methods:

Divisive clustering – the data set is regarded as a single cluster and then clusters are recursively split along best boundary

Agglomerative clustering – each data item is regarded as a cluster, and the clusters are recursively merged by choosing two most similar clusters

Agglomerative Clustering Algorithm

Basic algorithm:

- Let each data point be a cluster
- Compute the proximity matrix
- **Repeat**
- Merge the two closest clusters
- Update the proximity matrix
- **Until** only a single cluster remains

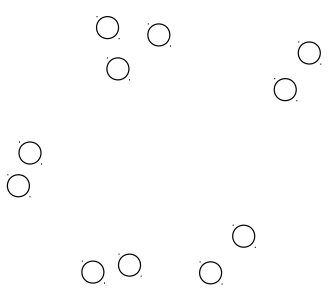
Key operation is the computation of the proximity of two clusters

Different approaches to defining the distance between clusters distinguish the different algorithms

Agglomerative Clustering Algorithm: example

Start with clusters of individual points and calculate proximity matrix

Data points in (2d) feature space:



Proximity Matrix:

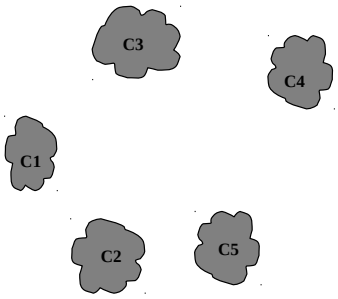
	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Dendrogram:

Agglomerative Clustering Algorithm: example

After some merging steps, we have some clusters and a revised proximity matrix

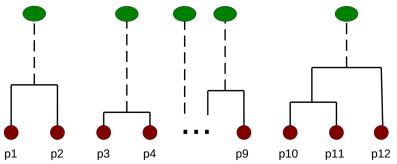
Data points in feature space:



Proximity Matrix:

	C1	C2	C3	C4	C5
C1					
C2					
C3					
C4					
C5					

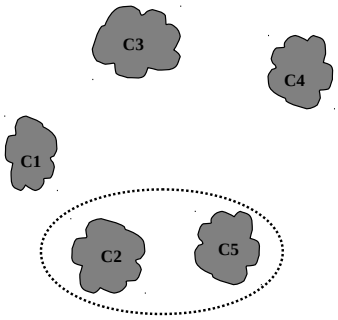
Dendrogram:



Agglomerative Clustering Algorithm: example

The two closest clusters are now C2 and C5. These need to be merged and the proximity matrix updated

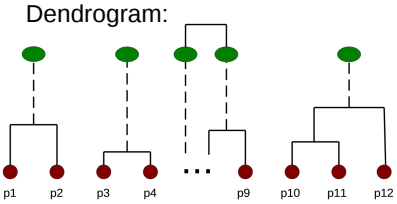
Data points in feature space:



Proximity Matrix:

	C1	C2	C3	C4	C5
C1					
C2					
C3					
C4					
C5					

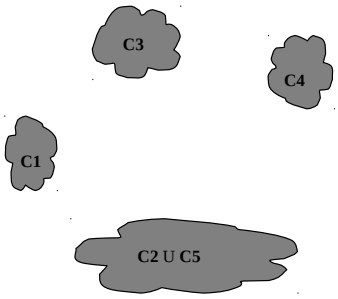
Dendrogram:



Agglomerative Clustering Algorithm: example

How we update the proximity matrix depends on how we define inter-cluster similarity

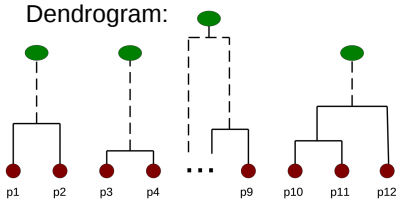
Data points in feature space:



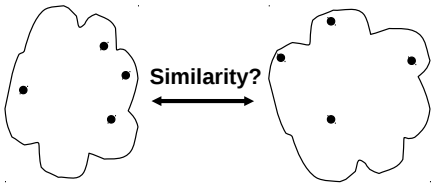
Proximity Matrix:

	C1	C2 U C5	C3	C4
C1		?		
C2 U C5	?	?	?	?
C3		?		
C4		?		

Dendrogram:



Agglomerative Clustering Algorithm

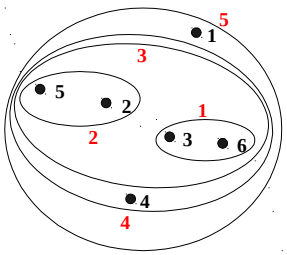


How to Define Inter-Cluster Similarity?

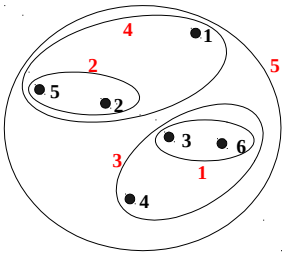
- single-link clustering - distance between clusters is shortest distance between elements (MIN distance)
- complete-link clustering - distance between clusters is longest distance between elements (MAX distance)
- group-average clustering - distance between clusters is average of all distances between elements (AVERAGE distance)
- centroid clustering - distance between clusters is the distance between the average of the feature vectors in each cluster

Agglomerative Clustering Algorithm

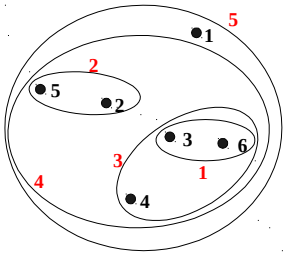
Results will vary depending on the method used to calculate cluster similarity.



MIN distance



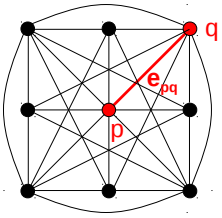
MAX distance

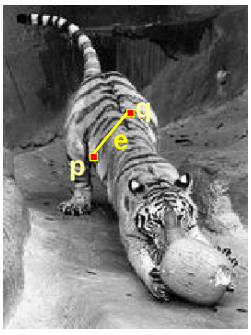


AVERAGE distance

Segmentation by graph cutting

Feature space can be considered to form a graph G = (V, E)





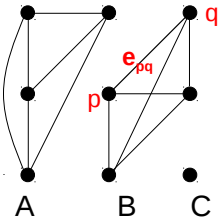
vertices (V) represent image elements (multi-dimensional feature vectors)

edges (E) connect pair of vertices

each edge encodes the similarity between the two connected elements (e.g. similarity in colour, position, intensity, etc.)

Segmentation by graph cutting

Image segmentation can be viewed as finding an optimal partitioning of the graph (i.e. cutting the graph into disjoint subgraphs).



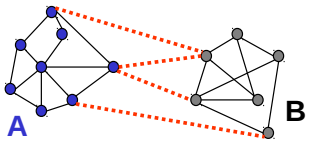
Graph cutting attempts to produce subgraphs such that the vertices within each subgraph represent elements within the same image region.

Achieved by partitioning the graph so that:

- similarity between subgraphs is minimized (i.e. cut edges are weak)
- similarity within subgraphs is maximized (i.e. subgraphs have strong interior links)

Segmentation by graph cutting

Cutting the graph into disjoint subgraphs will generally require several edges to be cut.

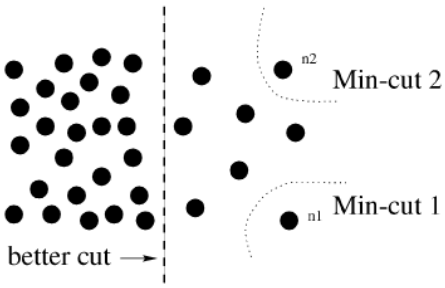


Cost of cutting edges whose removal makes two sets of vertices (A and B) disconnected:

$$cut(A,B) = \sum_{p \in A, q \in B} e_{pq}$$

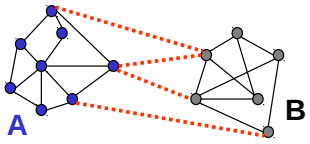
Finding sets of vertices A and B which give minimum cut cost, will favour cutting small sets of nodes over large sets

(cost is proportional to the number of edges in the cut)



Normalized Cuts (Ncuts)

Bias towards small subgraphs can be overcome by normalising the cut cost by the total strength of all the connections in the two segments.



Normalised cost of cutting edges whose removal makes two sets of vertices (A and B) disconnected:

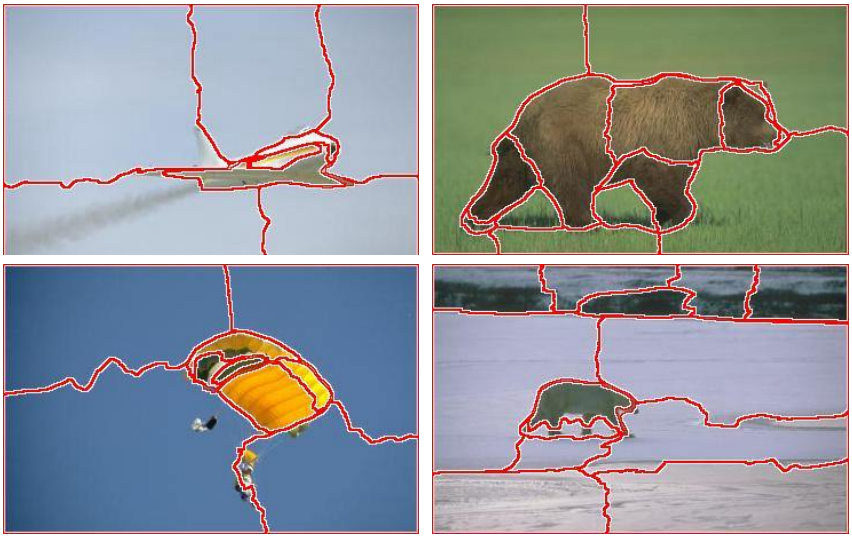
$$Ncut(A,B) = \frac{cut(A,B)}{assoc(A,V)} + \frac{cut(A,B)}{assoc(B,V)}$$

Where:

$$assoc(A,V) = \sum_{p \in A, q \in V} e_{pq}$$

assoc(A,V) is the total strength of all connections from vertices in set A to all the vertices in the graph

Normalized Cuts (Ncuts): results



Red lines indicate estimated region boundaries.

Normalized Cuts (Ncuts): problems

- Finding sets of nodes (A and B) which produce the minimum cut is NP-hard:
  - only approximate solutions are possible for real images.
- Bias towards partitioning into equal segments
- Has problems with textured backgrounds (as do most image segmentation algorithms)

Fitting

A class of methods that try to use a mathematical model to represent a set of elements.

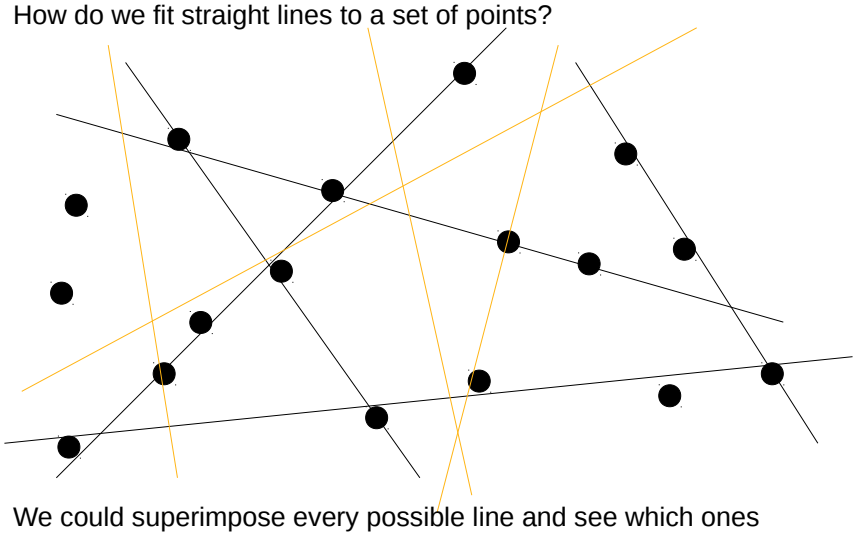
Can be used to find the boundaries of objects, and hence, segment the image.

Example:

- A model of the outline of an object that can be rotated and translated to compare it with the image.
- If this model is found to closely fit a set of points or line segments this is unlikely to be due to chance.
- So we represent those elements using the model, i.e. the elements that fit the model are grouped together.

Fitting: example fitting straight lines

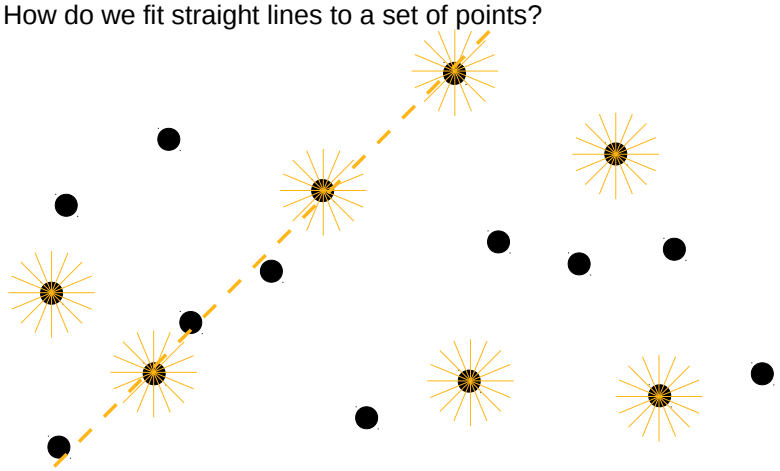
How do we fit straight lines to a set of points?



We could superimpose every possible line and see which ones account for the most data.

Fitting: Hough Transform for fitting lines

How do we fit straight lines to a set of points?



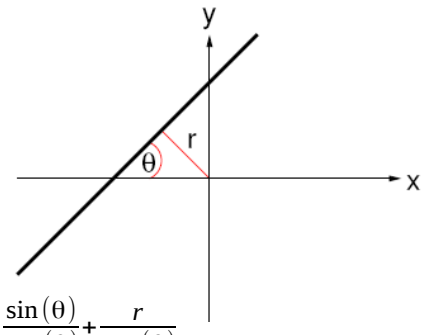
Alternatively, we could see which lines can potentially pass through each point, and count which of these possible lines recur most often.
i.e. each data point votes for all the lines that could pass through it, and lines that have a lot of votes are ones that pass through most points.

Hough Transform: line parametrisation

Assume that we want to find elements that form straight lines (i.e. our model is a line)

Representing a line in the usual form, $y = mx + c$, has the problem that m goes to infinity for vertical lines.

A better choice of parameters for the line is angle, θ, and perpendicular distance from the origin, r.



A line is then the set of points (x, y) such that:

$$y = x\tan(\theta) + \frac{r}{\cos(\theta)} = x\frac{\sin(\theta)}{\cos(\theta)} + \frac{r}{\cos(\theta)}$$

$$y\cos(\theta) - x\sin(\theta) = r$$

We can represent any line by the two parameters θ and r.

Hough Transform: example

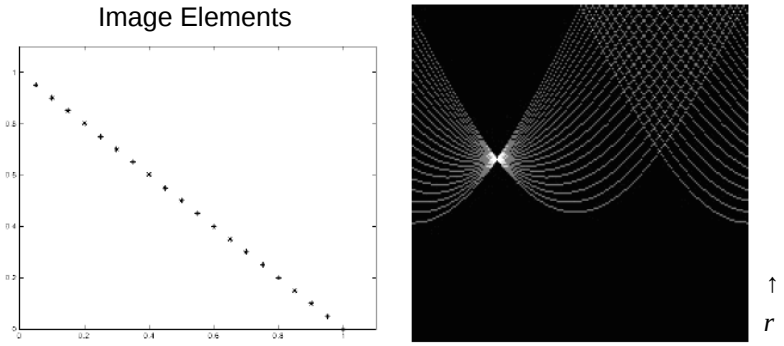
Any point (x, y) in an image could potentially form part of of a family of lines that pass through this point.

All these lines obey the following relationship: $r = y\cos(\theta) - x\sin(\theta)$

An accumulator array is used to count the votes for each set of parameter values.

Votes

Image Elements



↑ r

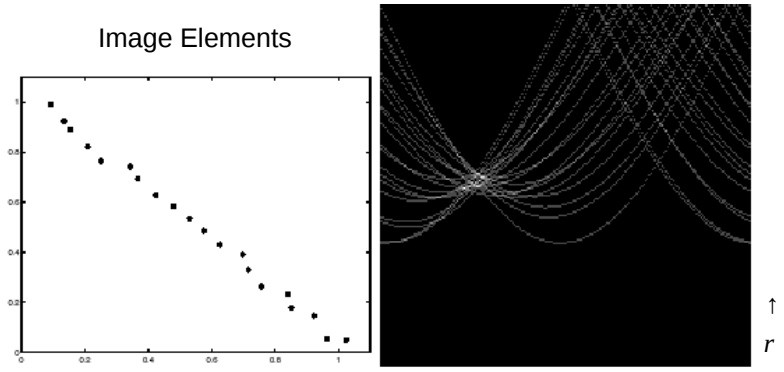
θ →

Hough Transform: example

Peak in Hough space becomes blurred as elements are less well aligned

Votes

Image Elements



↑ r

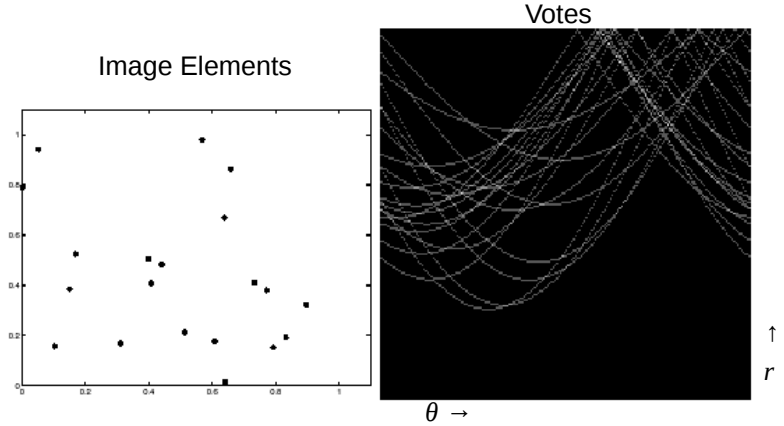
θ →

Hough Transform: example

Multiple, weak, peaks in Hough space result from random, unaligned, points.

Votes

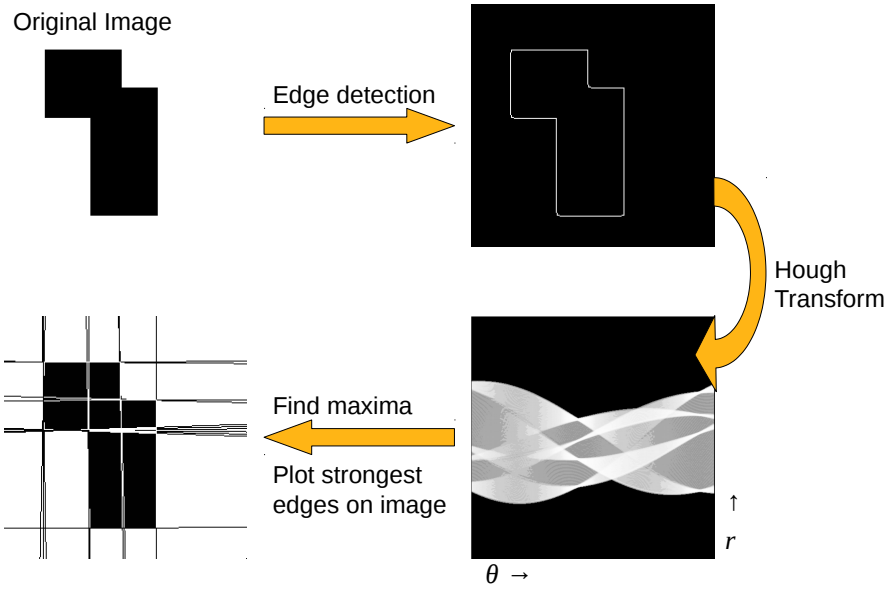
Image Elements



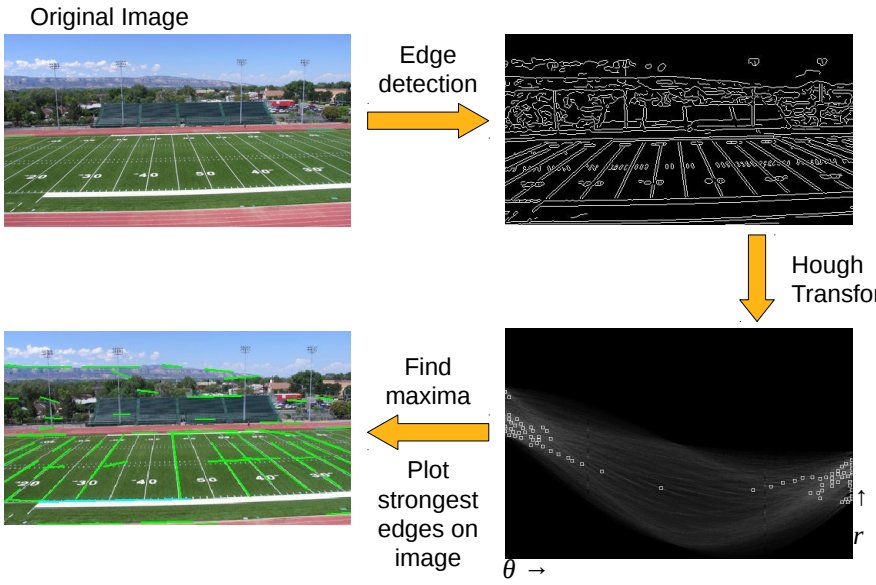
↑ r

θ →

Hough Transform: example



Hough Transform: example



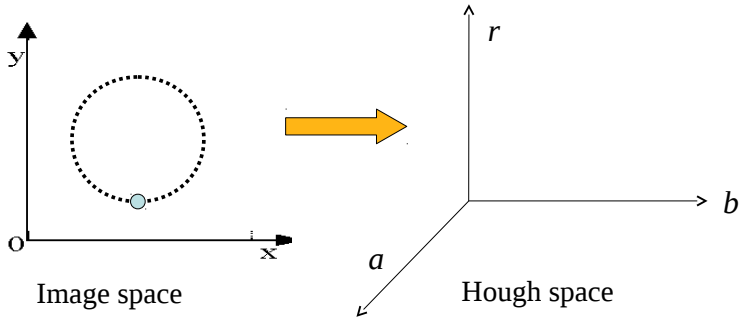
Hough Transform: generalised

The Hough transform can be extended to other shapes that can be expressed parametrically.

e.g. a circle can be described using 3 parameters (a, b, r), that define the centre (a,b) and radius (r) of the circle.

Each point in the image can vote for the (a, b, r) values that encode circles that could potentially pass through that point.

Hence, the accumulator array is 3D in this case.



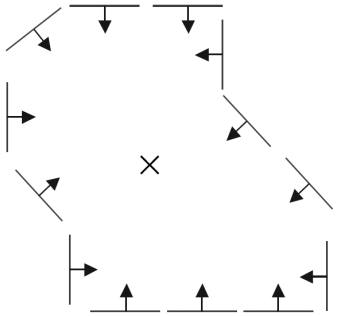
Hough Transform: generalised

The Hough transform can also be extended to arbitrary shapes that can *not* be expressed parametrically.

To do so, make a table that describes the location of each edge pixel in the target shape relative to some reference point.

For each point in the image, assume that it is each edge location in turn, and vote for the location of the reference point.

This is called the Generalised Hough Transform.



Hough Transform

Advantages

- tolerant of gaps in the edges
- Tolerant to occlusion in the image
- relatively unaffected by noise
- can detect multiple occurrences of a shape in the same pass

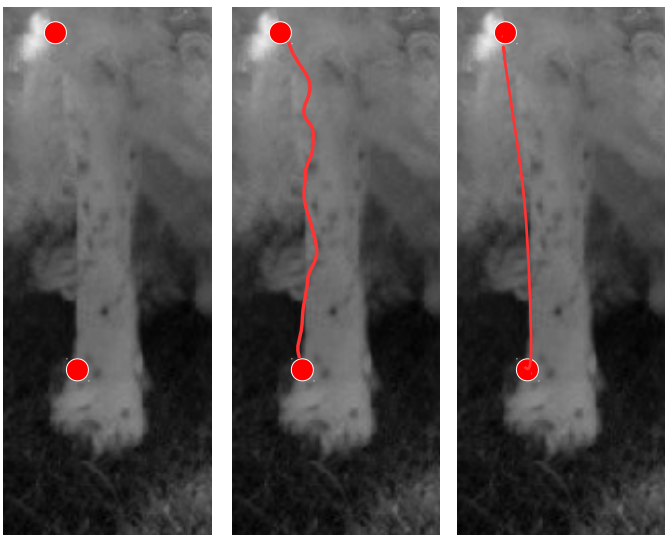
Disadvantages

- expensive in terms of memory and computation time
- localisation information is lost during transformation, e.g. end points of lines are unknown
- peaks in the accumulator can be difficult to locate in presence of noisy or parallel edges
- difficult to determine how coarse or fine the quantization of the accumulator should be: too coarse, and we merge quite different lines; too fine, and noise causes lines to be missed

Active contours (“snakes”)

Assume we want to find the boundary between these two points.

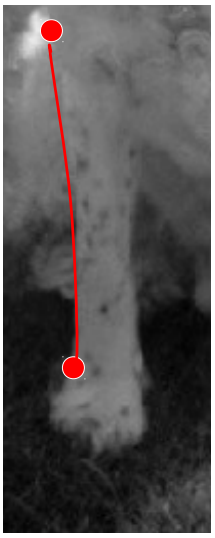
Which is the best?



Active contours (“snakes”)

Contour should be:

- near the edge
- smooth



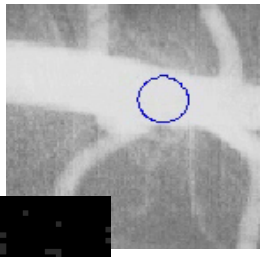
A snake is a spline curve that moves so as to minimise “energy”.

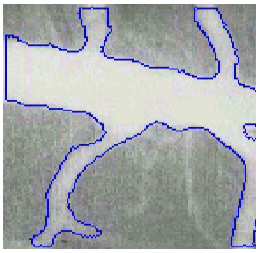
Energy is the sum of two terms:

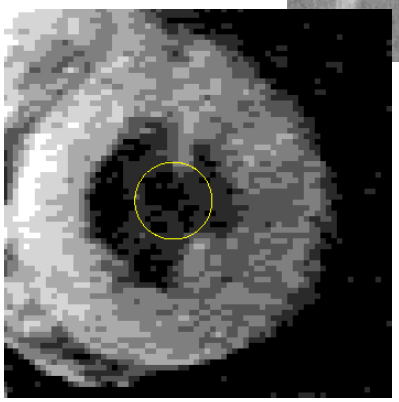
- internal energy determined by shape of the snake
  - bending and stretching curve increases energy
- external energy determined by proximity of the snake to image features
  - large intensity gradients decrease energy

Minimizing the energy of the snake results in a curve that is short, smooth, and close to the edge.

Active contours (“snakes”): examples





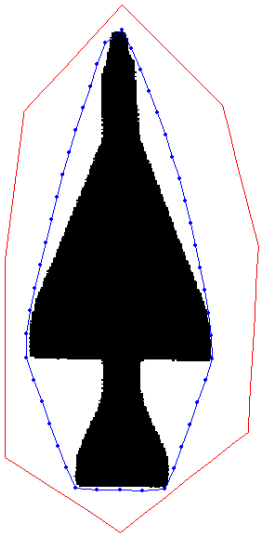


Snake deforms and moves towards an object boundary.

In the end it completely “shrink-wraps” around the object.

Active contours (“snakes”): problems

- Only works for shapes that are smooth and form a closed contour.
- Extremely sensitive to parameters.
  - parameters control the relative strength of different energy terms (i.e. how big the influence of smoothness, shortness, feature proximity).
- Convergence is dependent on initial position.
  - snake usually placed around object by hand!
- no external force acts on points where there is no intensity gradient, e.g. points which are far away from the boundary.

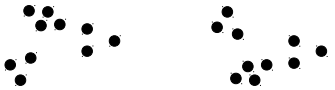


Summary

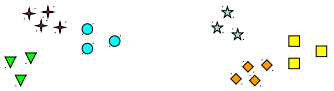
There are lots of methods of image segmentation.

None work very reliably!

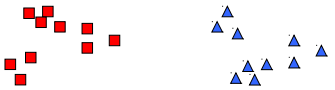
Part of the problem may be that grouping is inherently ambiguous:



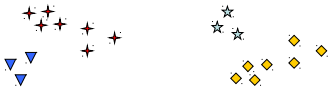
How many clusters?



Six Clusters



Two Clusters



Four Clusters

Rather than attempting to produce one result, it might be better if segmentation methods produced multiple possible segmentations.

Summary

We can classify segmentation methods in two different ways:

edge-based vs region-based, or

model-based vs model-free

Edge-based methods

e.g. Hough transform, active contours, thresholding (applied to the output of an edge detector)

The approach is to partition an image based on abrupt changes in feature values (i.e. by looking for discontinuities / boundaries)

Region-based methods

e.g. thresholding (applied to intensity), region growing, region merging, split and merge, k-means clustering, hierarchical clustering, graph cutting

The approach is to group together image elements that have similar feature vectors (i.e. that look similar)

Summary

Edge-based methods

attempt to locate the *discontinuities* in image features that define a boundary *surrounding* a region.

Region-based methods

region-based methods attempt to locate the *similarities* in image features that define the set of pixels *within* a region.

One is the dual of the other.

Segmentations resulting from edge-based methods and region-based methods are not usually exactly the same, and a combination of results may often be a good idea.

Summary

Model-based methods

Hough transform, active contours

The approach is to fit the data to a predefined model.

Model-free methods

thresholding, region growing, region merging, split and merge, k-means clustering, hierarchical clustering, graph cutting

The data isn't (explicitly) assumed to fit any particular model, segmentation is based purely on similarity of image features.

Summary

Recall:

Top-down

- elements belong together because they lie on the same object

Bottom-up

- elements belong together because they are similar or locally coherent

Many computer vision systems attempt to perform segmentation before recognition (i.e. they use a bottom-up approach).

However, most of these systems implicitly include some top-down knowledge as the system's designer has made choices about what features to group based on the task they are trying to perform.

Other computer vision systems explicitly use a top-down approach and attempt to fit a model to the data.