Imperial College London

9 - Segmentation

Prof Peter YK Cheung

Dyson School of Design Engineering

URL: www.ee.ic.ac.uk/pcheung/teaching/DE4_DVS/
E-mail: p.cheung@imperial.ac.uk

The materials in this Lecture is based on the second half of Chapter 10 of the recommended textbook, "Digital Image Processing" by Gonzalez and Woods, and on "Introduction to Bioimage Analysis" (*https://bioimagebook.github.io*).

Segmentation subdivides an image into its constituent regions or objects. The level to which the subdivision is carried depends on the problem being solved. That is, segmentation should stop when the objects of interest have been isolated. For example, in the automated inspection of electronic assemblies, interest lies in analyzing images of the products with the objective of determining the presence or absence of specific anomalies, such as missing components or broken connection paths. There is no reason to carry segmentation past the level of detail required to identify those elements.

Segmentation of nontrivial images is one of the most difficult tasks in image processing. Segmentation accuracy determines the eventual success or failure of computerized analysis procedures. For this reason, considerable care should be taken to improve the probability of rugged segmentation. In some situations, such as industrial inspection applications, at least some measure of control over the environment is possible at times. In others, as in remote sensing, user control over image acquisition is limited principally to the choice of imaging sensors.

What is Segmentation?

- Segmentation is the **division** of an image into **spatial region**.
- This is achieved by **grouping pixels with similar visual characteristics** into regions.
- If R represent the entire spatial region, and R_i and R_j represents two separate regions i and j, image segmentation is def $\bigcup_{i=1}^{n} R_i = R$ the following five conditions:

a) Every pixel must be in the region R:

b) R_i is a region where pixels are connected (i.e. in connected set).

c) Region R_i must be disjoint: $R_i \cap R_j = \emptyset$ for all i and $j, i \neq j$.

d) All pixels must share some common properties Q: $Q(R_i) = TRUE$ for all pixels in R_i.

e) Two adjacent regions R_i and R_j must be different in some properties Q.

There are 5 conditions related to the process of segmentations. In the textbook, these five conditions are stated mathematically in terms of set theory. You are only expected to understand what they mean, not the mathematical formulation.

Condition (a) indicates that the segmentation must be complete, in the sense that every pixel must be in a region.

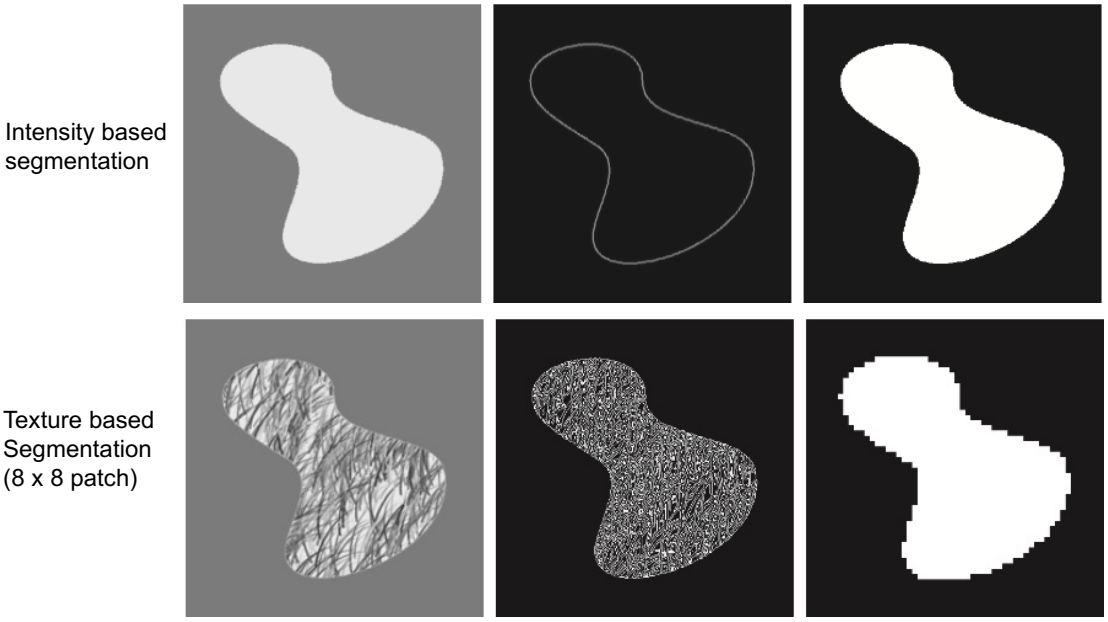
Condition (b) requires that points in a region be connected in some predefined sense (e.g., the points must be 8-connected).

Condition (c) says that the regions must be disjoint.

Condition (d) deals with the properties that must be satisfied by the pixels in a segmented region—for example, $Q(R_i) = TRUE$ if all pixels in R_i have the same intensity.

Finally, condition (e) indicates that two adjacent regions R_i and R_j must be different in the sense of predicate Q.

Segmentation based on discontinuity

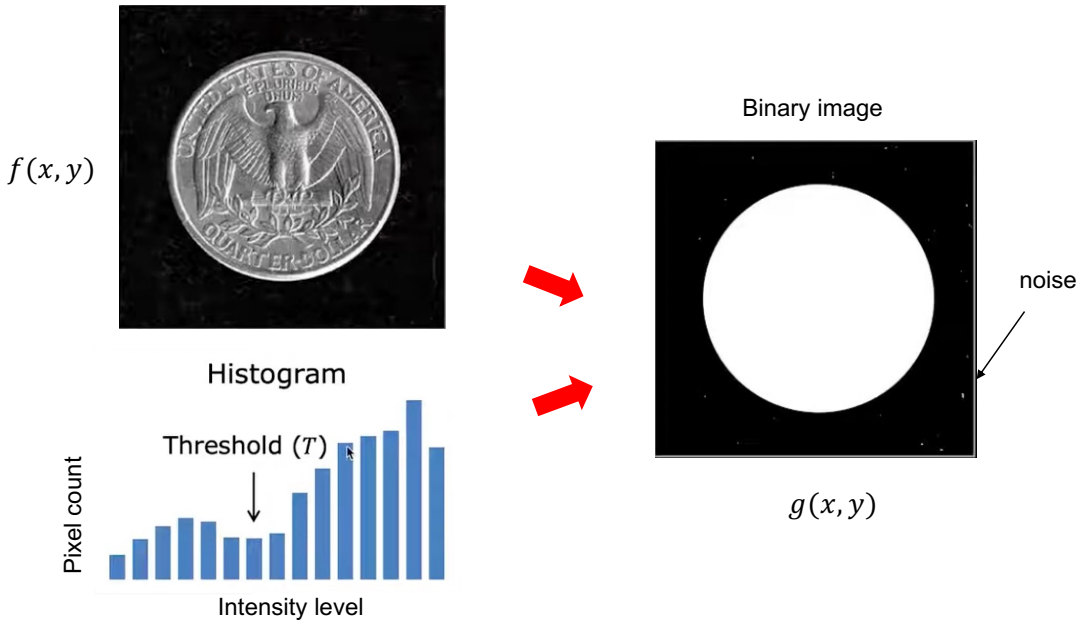


Segmentation algorithms in this lecture are for grayscale images. They are based on one of two basic properties of image intensity values: **discontinuity** and **sim**ilarity. In the first category, the approach is to partition an image based on **abrupt changes in intensity**, such as edges. The principal approaches in the second category are based on **partitioning an image into regions that are similar** according to a set of predefined criteria.

Top-left image on the slide has two regions: a white shape on a dark background. The top-middle image is the result of computing the boundary of the inner region based on intensity discontinuities. To segment the image, we assign white (i.e. 255) to the pixels on or inside the boundary, and black (i.e. 0) to all points exterior to the boundary. The top-right image shows the result of such a procedure. These three images demonstrates segmentation based on discontinuity.

The next three images illustrate region-based segmentation. Bottom-left image has a region based on textured pattern and not intensity. The bottom-middle image shows the result of segmentation using intensity discontinuity. It is clear that edge-based segmentation is not a suitable approach for this image. Instead, we can use the standard deviation of pixel values as a measure. The image on the bottom-right shows the result of dividing the original image into subregions of size 8×8. Each subregion was then labelled white if the standard deviation of its pixels was positive and zero otherwise. The result has a "blocky" appearance around the edge of the region because groups of 8×8 squares were labelled with the same intensity (smaller squares would have given a smoother region boundary).

Segmentation using Thresholding



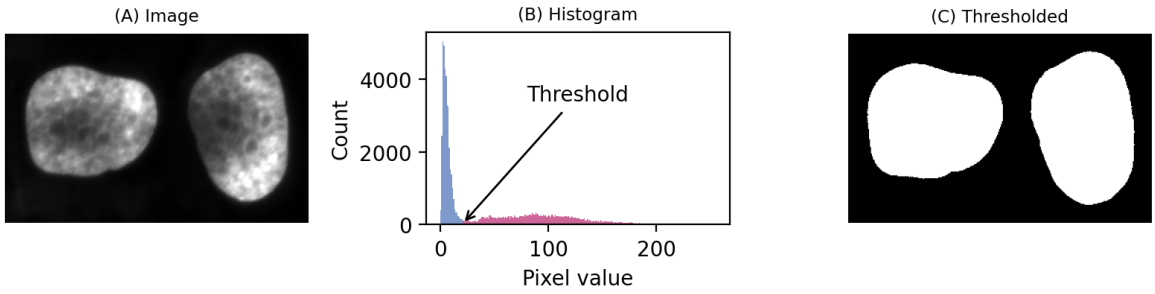
The simplest approach to segmentation is through thresholding. Given a grayscale image $f(x,y)$ with pixels at different intensity levels, a certain intensity level is chosen (how?) as a threshold value T. All pixels with intensity lower than the threshold is considered as black (background), and all those equal or higher than the threshold is considered as white (foreground). The end result is a black-and-white image $g(x,y)$.

As shown above, such an approach could result in noise speckles in the background.

This approach is very simple. The only decision required is to decide what the threshold T should be.

The inevitable question to ask is: How can one decide what is the best threshold value to use?

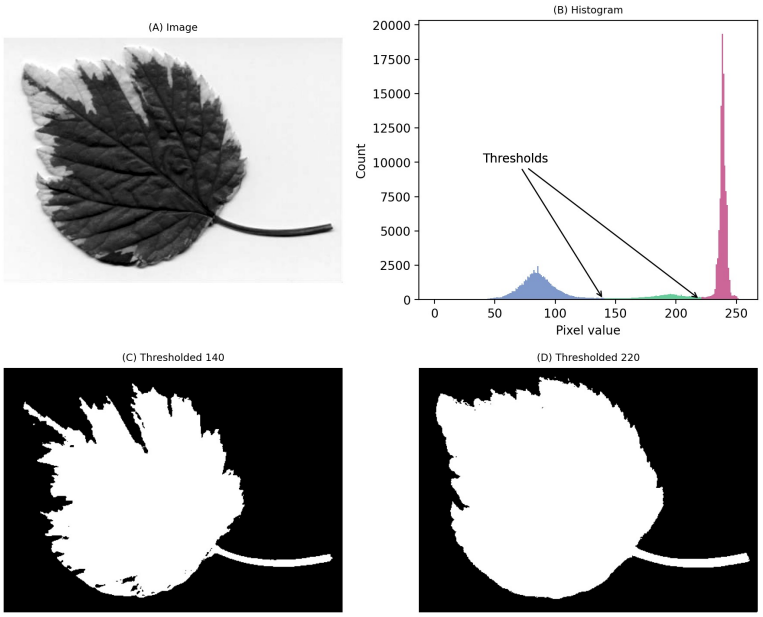
Histogram can be a guide



- The left image is to be segmented.
- Histogram is bimodal - the background and the object occupying two distinct regions.
- Putting the threshold between the two “hills” will easily segment the image.

For images with clear objects and background separations, determining the best threshold is easy. Here is an image with two objects and a clear background. The histogram is bimodal. The obvious best threshold is located in the middle of the two “humps”. The segmentation result is obvious and is of high quality. But then, this is a simple image to segment. What if it is not so obvious?

What if it is not bimodal?



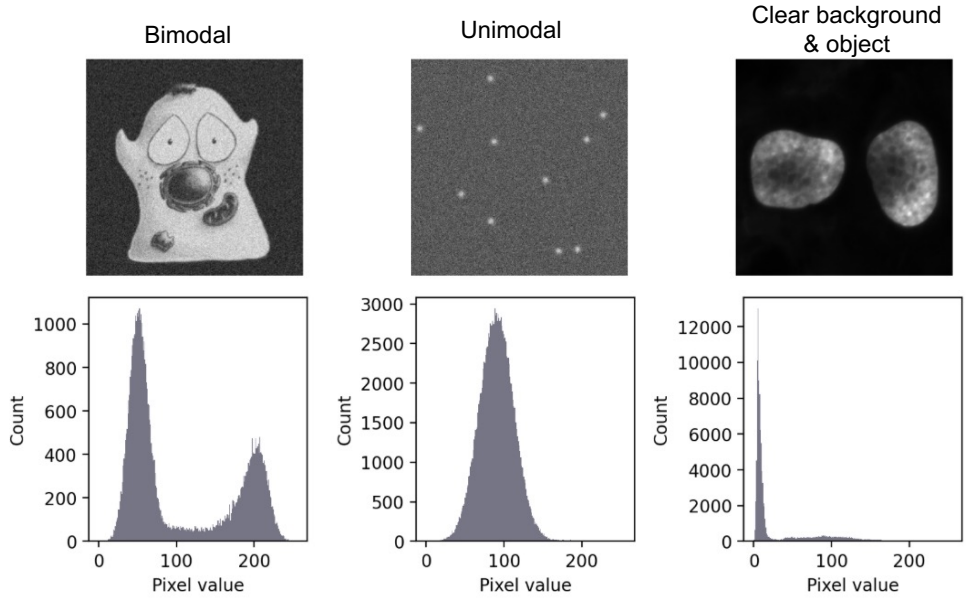
Here is an example where there are actually three ”humps”, the background and two other regions.

There could be two possible thresholds. Using either one will result in very different segmentation.

One solution is to segment this image into three instead of just two regions.

In general, it is actually not easy to take an arbitrary image and decide how many regions it should be segmented to.

Three Different Type of Images & Histograms

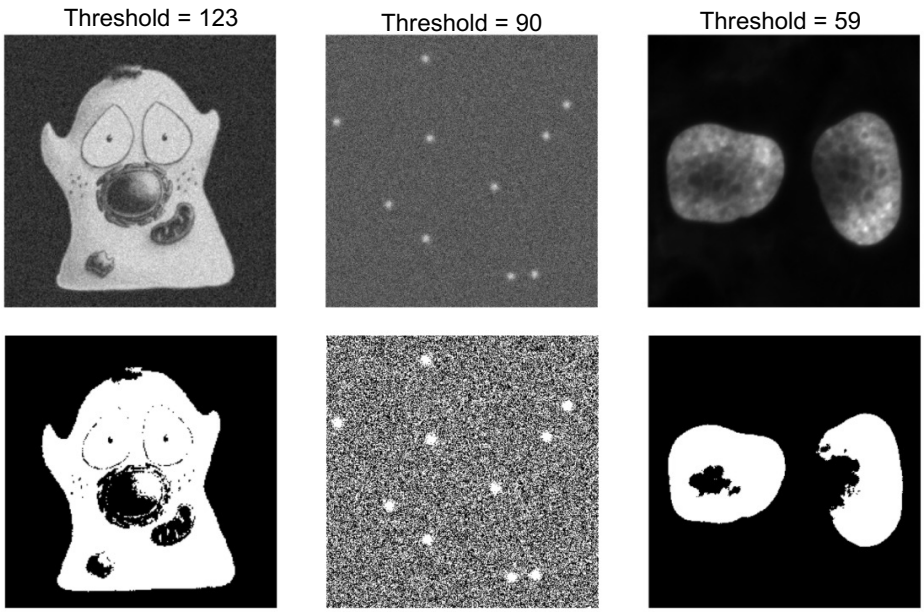


Different images have different histogram characteristics. Some are suitable for using global thresholding to perform segmentation as described in previous slides. Some are not.

Shown on the left is an image with bimodal histogram, but the optimum threshold level is not obvious, unlike that of the right-most image, which has clear foreground and background division.

The image in the middle is much more challenging using thresholding method because the histogram is unimodal (only one hump).

Otsu's method to find threshold



Otsu's method, introduced in 1979, is an effective and popular method in determining the threshold automatically, as long as the histogram is bimodal with a clear and relatively deep valley in between the two humps.

The exact algorithm is somewhat involved, and is described in detail in Gonzalez and Woods, p747-451.

In essence, Otsu's method involves adding the variance of the background pixels to the variance of the foreground pixels, for all possible thresholds. The threshold that is selected is the one for which the sum of the variances is smallest. This effectively is to keep the distributions of foreground and background pixels as close to each other as possible.

Otsu method does not work well with images with non-bimodal histogram.

As shown above, Otsu's method works well for the left and right images, but not at all for the middle image which does not have a bimodal histogram.

Variable Thresholding based on local statistics

- Threshold $T_{xy} = a\sigma_{xy} + bm_{xy}$, where σ_{xy} is local standard deviation, m_{xy} is local mean intensity, and a, b are positive constants.

Image of yeast cells

Using dual global thresholds

Using local thresholds

Another approach is to compute multiple thresholds using some local statistical property of the image such as mean and standard deviation. For example, for each pixel $f(x, y)$, take a $w \times w$ window in its neighbourhood, and calculate the mean m_{xy} and standard deviation σ_{xy} with (x, y) at its centre, and compute the threshold value:

$$T_{xy} = a\sigma_{xy} + bm_{xy}$$

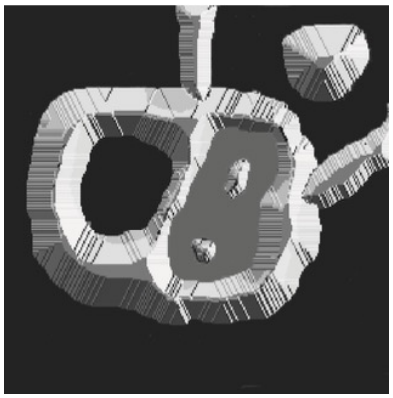
The pixel is set to white if it is above or equal to the threshold, otherwise, it is black.

In the example above, the image of some yeast cells shows having two distinct intensity levels. Using two threshold values, we can segment the image into three regions: yeast core, yeast body and background. However, the two thresholds are global to the image, and as a result, some cells are merge together as shown in the middle image.

If local threshold is used, we obtain the right most image which show clear separation between yeast cells.

Segmentation using Watershed Transform (1)

- A watershed is the ridge that divides areas drained by different river systems.
- A catchment basin is the geographical area draining into a river or reservoir.
- The segmented regions are the basins that catch the rainwater as water rises.
- The watershed ridge line partition the image into regions thus achieving segmentation.



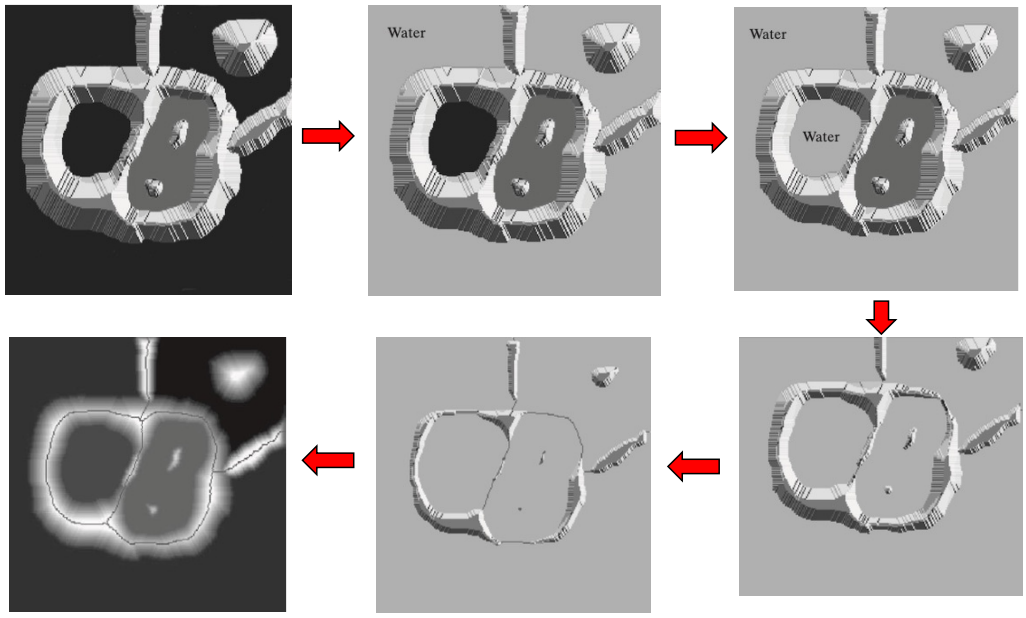
In geography, a watershed is the ridge that divides areas drained by different river systems. A catchment basin is the geographical area draining into a river or reservoir. The watershed transform applies these ideas to grayscale image processing in a way that can be used to solve a variety of image segmentation problems.

In watershed transform, a grayscale image is considered as a topological surface, where the values of $f(x, y)$ are interpreted as heights. For example, the left image above can be view as a three-dimensional surface shown on the left.

If we imagine rain falling on this surface, it is clear that water would collect in the two areas labelled as catchment basins. Rain falling exactly on the watershed ridge line would be equally likely to collect in either of the two catchment basins.

The watershed transform finds the catchment basins and ridge lines in a grayscale image.

Segmentation using Watershed Transform (2)



To perform watershed segmentation algorithm, we take the topological surface and start flooding the landscape. As water is falling, the catchment basic starts filling with water as shown in the right most figure.

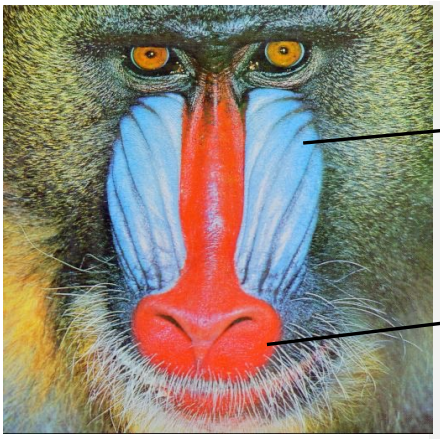
As flooding progresses, the two regions start to merge together as shown in bottom-right image. To prevent the two regions from merging into one, a dam is built between them as shown in the lower-middle image.

Eventually, the different regions become separated by dams. The dams become the boundaries of the segmentation of the image!

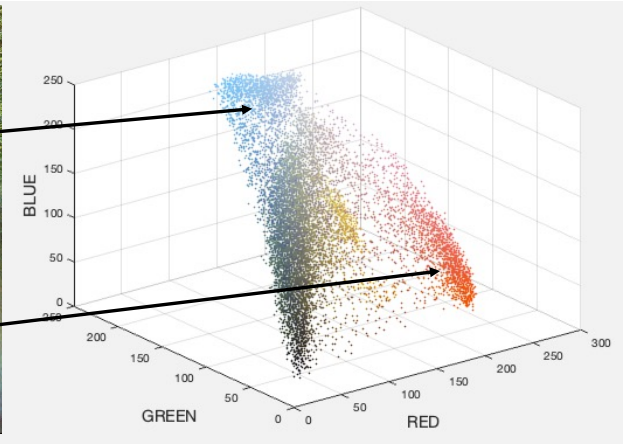
Segmentation based on clustering

- Visual characteristics: Intensity, colour, position, texture, motion, depth
- Each pixel as feature vector: [R, G, B, x, y …..]

Colour image



Pixel colour in feature vector [R G B]



Using pixel intensity and thresholds to perform segmentation has many limitations. A more general and far more powerful approach is to the clustering approach. The idea is that we group together those pixels that share some common visual characteristics. For example, they could be roughly of the same colour or close in location.

Every pixel is then map to such feature space, and those pixels that are “close” in the feature space are group together as a cluster.

In the example above, each pixel in image of a baboon is mapped to the R, G and B space as a dot (in that colour). For this, the feature set is a vector [R G B].

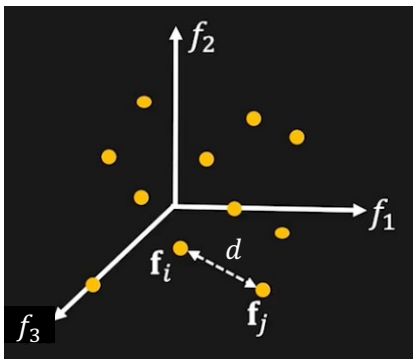
We may also define the feature vector to include the x, y coordinate as a 5 dimensional vector [R G B x y].

Pixel Similarity Measure

- Let f_i and f_j be the feature vector for pixels i and j respectively.
- The Euclidian distance between f_i and f_j is given by:

$$d(f_i, f_j) = \sqrt{\sum_k (f_{ik} - f_{jk})^2}$$

- Smaller the value of d, the greater the similarity.



We can then work out how similar each pixel is relative to other pixel by comuting the Euclidian distance in the feature space.

In the sample above, each pixel is mapped to the feature space $[f_1\ f_2\ f_3]$. To find the distance between two such pixels i and j in the feature space, we simple calculate the sum of the squares of the differences between the two pixels, then take the square root.

$$d(f_i, f_j) = \sqrt{\sum_k (f_{ik} - f_{jk})^2}$$

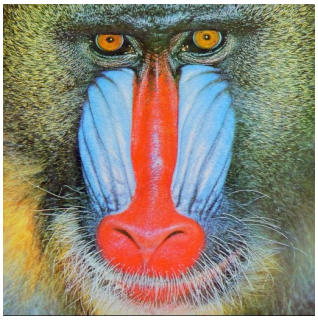
This is known as the Euclidean distance.

The shorter the distance, the more similar they are.

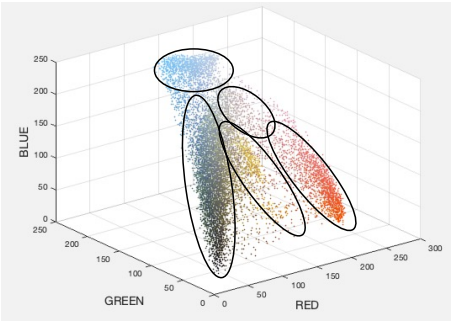
Segmentation by Clustering [R G B] vectors

- Group together those pixels that have high similarity in colour (i.e. short $d(r, g, b)$ distances) to form clusters.
- Assign a "mean" colour to each cluster.
- Now we have a segmented image by colour.

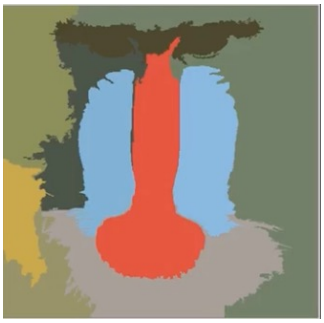
Original image



Features divided into 5 clusters



Segmented by colour

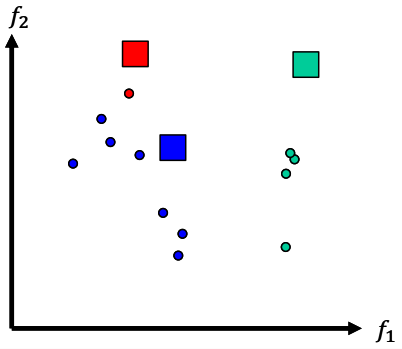


Going back to our baboon image and their [R G B] feature space. If these are somehow clustered into five groups, and for each group we compute the "average" feature values, then we can assign the same average (or mean) values to all the pixels belonging to that cluster.

This results in a segmented image as shown on the right.

Segmentation by k-Means Clustering (1)

- Segment image into k clusters using pixel characteristics.
  Assume k = 3
- Step 1: Generate 3 random initial means (centroids) in feature space.
- Step 2: Cluster each pixel to the mean with shortest distance to form 3 clusters as shown.



There are various algorithms which can be employed to perform clustering. One simple algorithm is known as *k*-means clustering.

Here is a simple example of pixels having only two features f_1 and f_2. The pixels are shown in green dots in the feature space.

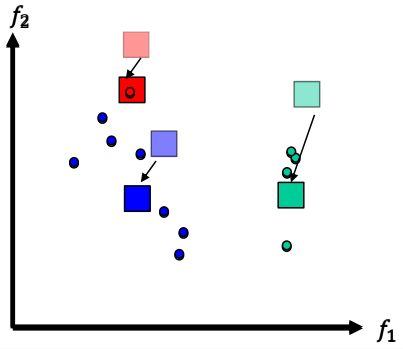
We wish to segment the image into 3 regions (i.e. $k = 3$).

Step 1 is to somehow (say randomly) generate three initial "mean" or centroid in the feature space as shown as RED, GREEN and BLUE squares above.

Step 2 is to label all pixels in the feature space closes to the mean as belong to the same cluster. That is, all the red pixels (only one here) closes to the red square is now belonging to the red group etc.

Segmentation by 3-Means Clustering (2)

- Step 3: Recompute the mean of each cluster
- Step 4: Repeat steps 2 and 3 until mean (or centroid) values change below a small margin. Then segmentation is completed and converges to a final solution.
- Needs to determine k. Also need to select initial values.
- Best initial values: Perform k-means clustering on a subset of pixels, and use that solution as initial values.



In Step 3, recompute a new mean (or centroid) for each cluster using the newly assigned member pixels as shown in the slide.

With the new centroids, do step 2 and step 3 repeated until the mean values no long change within a given margin. The algorithm is then deemed to have converged to a solution.

So in general, k-means clustering steps are:

1. Pick k random initial locations of means in the feature space $\{m_1, m_2, \ldots., m_k\}$.
2. For each pixel g_j find the nearest mean m_i in the feature space and assign the pixel to cluster i.
3. Recompute the mean for each cluster using the newly assigned pixels.
4. Calculate the change in k means. If it is less than a predefined error ε, finish; else go back to Step 2.

You need to determine the value of k and ε, and the initial locations.

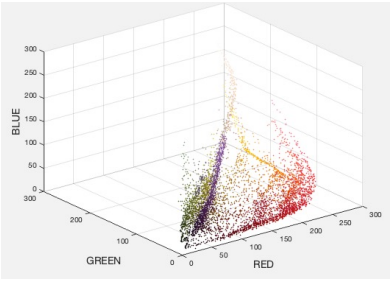
The initial locations can be determined by:

Method 1: Randomly select k initial feature points and re-do if two points are close.

Method 2: Select k uniformly distributed means.

Method 3 (Best): Perform k-means clustering on a subset of randomly selected pixels and use that result as the initial mean values.

Example of k-Means Clustering (k = 16)



k = 16, {R, G, B} space only

k = 16, {R, G, B, x, y} space

Here is an example of using k-means clustering for segmentation. The peppers image has many objects of different colours. The {R, G, B} feature vector space mapping of all pixels are as shown. If k is set 16, then the resultant segmentation is shown on the bottom-left. As can be seen, many of the 16 segments of different shades of background colours and there are many disjointed regions.

However, if we extend the feature vector to 5 dimensions including the location of the pixel, i.e. {R, G, B, x, y}, then the segmentation result is considerably better with many of the small regions merged.

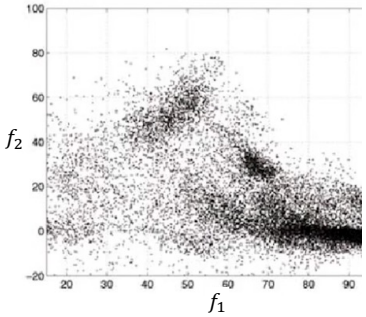
k-means clustering algorithm are

- Simple and fast.
- Need to predetermine the value of k.
- Sensitive to initial location of means – different initial location will give different results.
- Sensitive to outliers – outlying pixels will create false cluster regions.

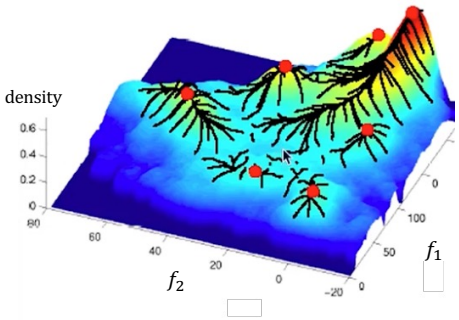
There is a better method: the mean-shift clustering algorithm.

Idea behind Mean Shift Clustering

Pixel feature distribution



Density (normalized to [0 1])



- Mean Shift determines how many clusters automatically depending on image.
- Each hill of the distribution represents a cluster.
- Each peak of the hill represents “mean” or ”centroid” of a cluster.
- Each pixel climbs the steepest hill within its neighbourhood.
- Pixel takes on the the peak of the hill in its cluster.

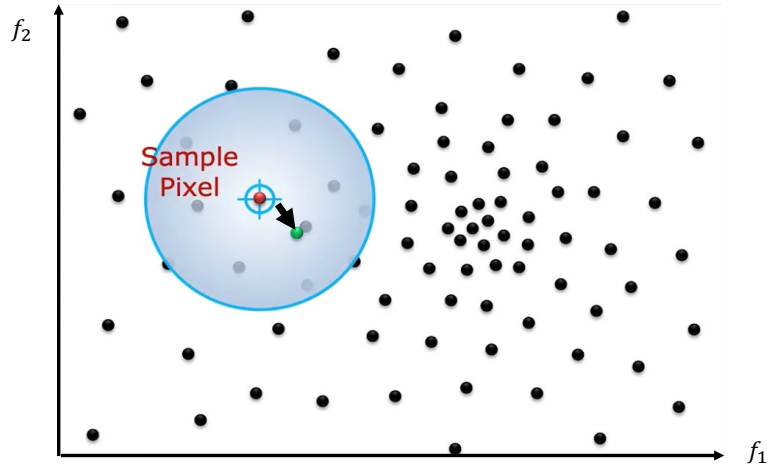
The idea behind the mean shift clustering algorithm is to work with the density distribution of pixels in the feature space.

Show above is an example in a 2D feature space. Each pixel in the original image is map to this space. There is clearly a variation in the number of pixel located at different part of the features space.

We can produce a density map for all the pixels in the feature space as shown on the right. There are area where more pixels share similar feature parameters than other areas. This distribution map (normalised to the range of 0 to 1) looks like a landscape with hills and valleys.

We now use the peaks (also called “modes”) of the hills as the centre of each cluster. All pixels that are surrounding this would “climb” the hill, until they reach their nearest peak. Then they take on the feature values of the peak.

Mean Shift Hill Climbing



- Consider one pixel (red) in feature space. Form a window centred at pixel with with w.
- Calculate the weighted means of feature parameters to find a new location in green.
- Shift the original pixel in feature space to the new mean (hence name "mean shift").
- Keep doing this until there is no shift – we are done for this pixel!

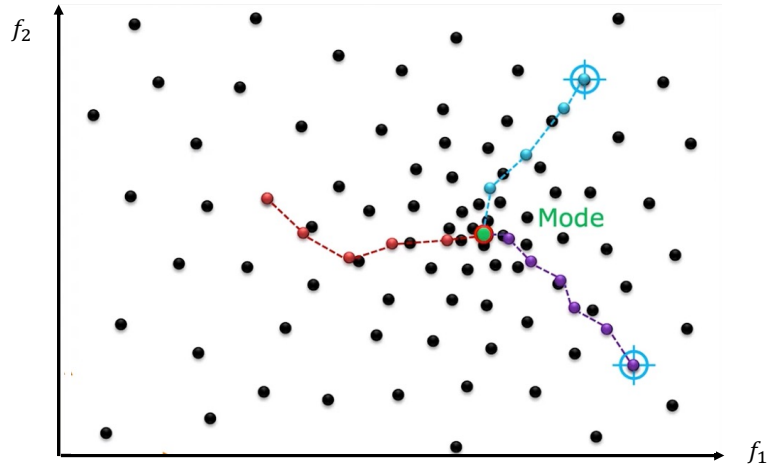
Here is an illustration of how one pixel would "climb" the hill in this algorithm.

Consider one pixel shown in as a red dot above. We define the neighbourhood of this pixel as a circle of radius w.

We now calculate the mean feature of all the pixel within the circuit (this could be a simple mean or a weighted mean with weighting decreasing with distance from the centre), to find a new mean value.

We then move the original pixel to this new mean values, and repeat the process again until the new mean is no longer different from the old mean (within as certain error). This results in the original pixel eventually finding its local peak and take on the feature values of that peak (and belong to that cluster).

Mean Shift Hill Climbing



- Therefore, the original pixel (in red), successfully climb the hill to the MODE (or peak).
- Other pixels nearby also eventually reach the same peak.
- These are ALL belonging the same cluster. They are then assigned the same feature vector as that of the peak.

As we repeat this process for the feature point of each pixel, many pixels will climb to the same peak as shown above. All of them are deemed to belong to the same cluster. They are assigned the feature values of that of the peak (or mode).

The mean shift clustering algorithm can therefore be define as:

1. Set $m_i = f_i$ as the initial mean value for each pixel.
2. For each mean m_i, do:
   a) Put a window of size w around m_i.
   b) Compute new centroid $\hat{m}_i$, and set $m_i = \hat{m}_i$.
   c) If shift in m_i to $\hat{m}_i$ is smaller than a threshold $\in$, peak is found.
3. Relabel all pixels reaching to the same peak with the same feature vector as the peak, and they belong to the same cluster.

Example of Mean Shift Clustering

k-Means (k = 16)

Mean Shift ($w = 21$)

Here is the same example of the pepper image. We are using the 5 dimensional feature vector {R, G, B, x, y}. As can be seen, the mean shift clustering segmentation produces much better results with all the background "climbing" the same hill to the peak, and therefore merged.

Here are the advantages of the mean shift clustering segmentation method:

- Simple idea, but more expensive than k-means clustering.
- No need to specify number of clusters (segments) – find its own "best" value.
- No initial feature means required.
- Robust against outliers.

However, one still needs to determine the window size w. Different value of w can results in different segmentation. Furthermore, mean shift method is computationally very expensive and therefore slow.

Finally, there is yet another popular segmentation method, known as "min-cut" method. Here the image is represented as a graph with nodes and edges. The method would cut the graph into a number of subgraphs (regions) such that some properties are minimized. This method is deliberately omitted in this lecture due to constraint of time.