Segmentation

Segmentation

- Image segmentation is a long standing problem in vision
  - We want to divide an images into regions
  - The image within a region should be uniform in some way
  - Adjacent regions should be different in some way

- Segmentation is a case of clustering
  - Given a set of entities (eg pixels) we want to divide them into groups (eg regions)
  - Elements of each group should be

  similar
  - Elements of different groups should be different

Segmentation

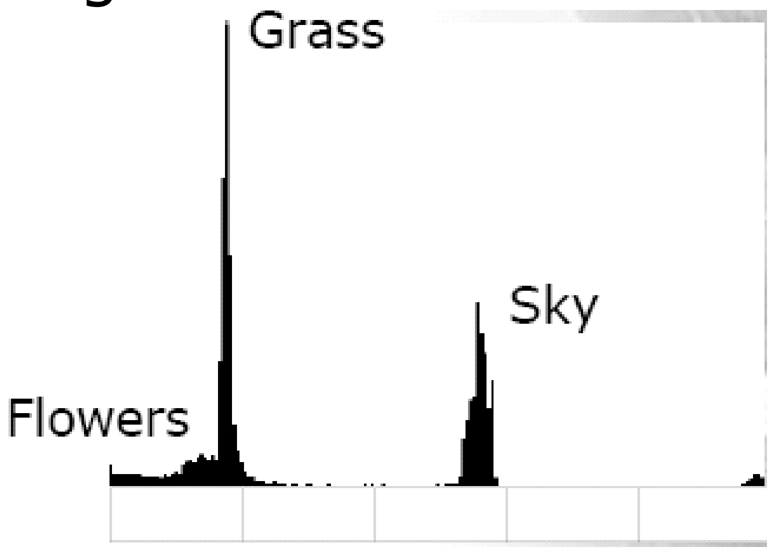
Image segmentation is a hard problem

- There is no 'correct solution – it depends very much on interpretation
- It is usually based on low-level cues (colour, texture, etc), but is trying to find high level things (objects)

Segmentation

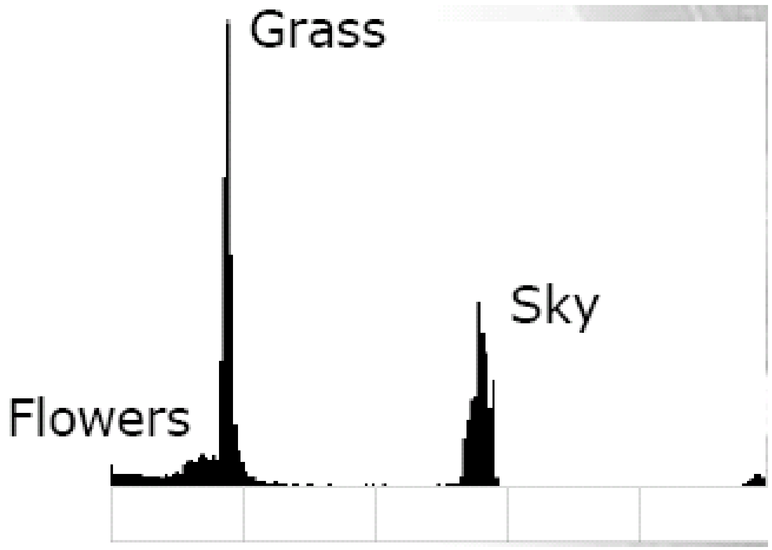
We find a histogram of some value we want to segment on

- Often intensity, or a colour metric
- We can use hue for the tree image since we have the green grass, flowers, and blue sky



Segmentation

- Assign a label based on pixel similarity to grass or sky



Clustering/Segmentation

- Clustering
  - To apply *k*-means to the image
  - We start with three hue estimates – say 0, 120, 240 which are red, green and blue
  - We cluster the pixels with *k*-means, being careful to remember that hue is a cyclic quantity

representation of pixels.

Clustering/Segmentation

Clustering, such as with *k*-means

- Finds groups of pixels with similar properties
- Doesn't guarantee that these groups form continuous areas in the image
- Even if it does the edges of these areas tend to be uneven

Region-Based Segmentation

We want smooth regions in the image

- We still want the pixels in each region to be similar, and those in adjacent regions to be different
- One way to do this is to work with regions rather than pixels

Region growing

- Start with a small 'seed' and expand by adding similar pixels
- Split and merge
  - Splitting divides regions that are inconsistent
  - Merging combines adjacent regions that are consistent

Region Growing

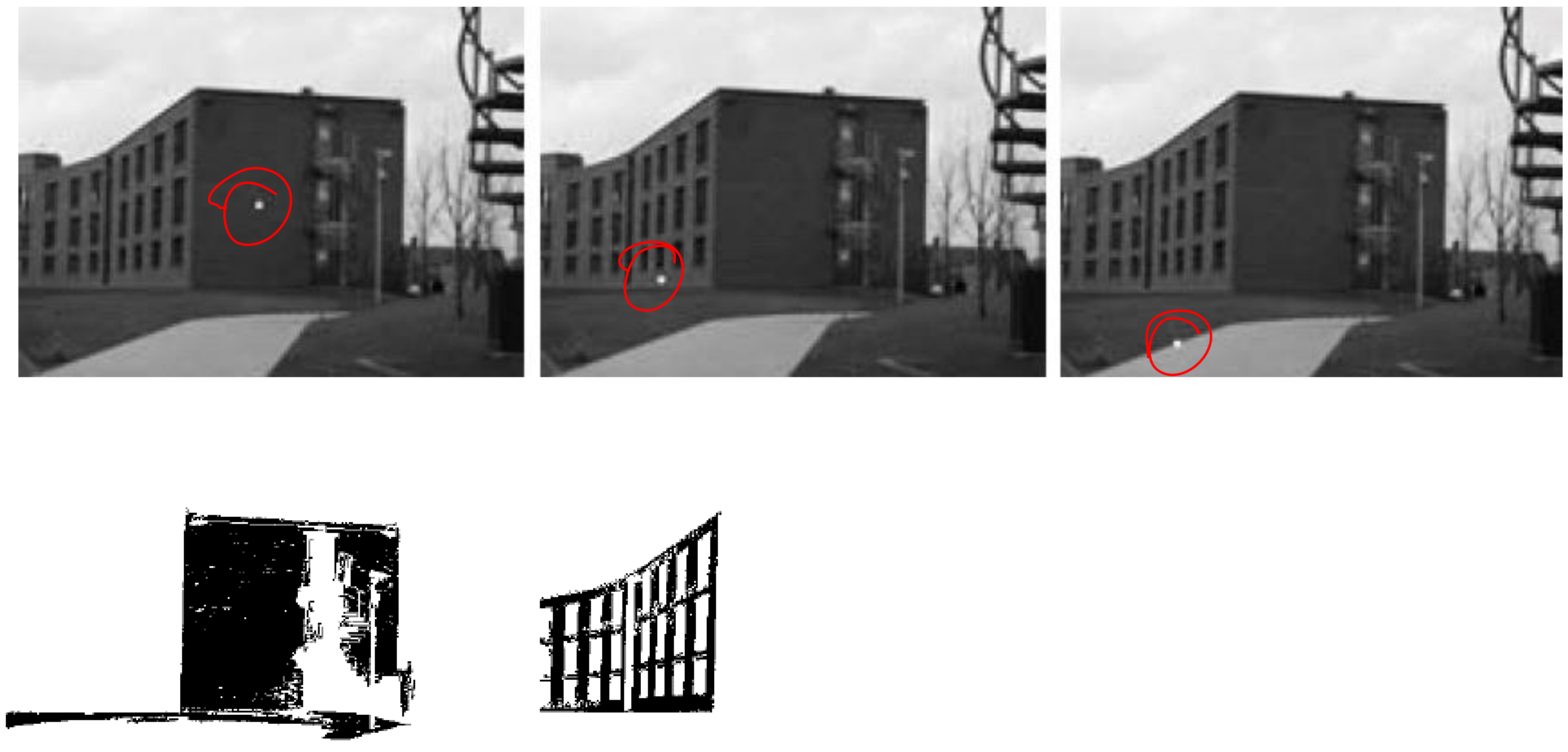
Region growing starts with a small patch of seed pixels

- Compute statistics about the region
- Check neighbours to see if they can be added
- Recompute the statistics

This procedure repeats until the region stops growing

- Simple example: We compute the mean grey level of the pixels in the region
- Neighbours are added if their grey level is near the average

Region Growing



Split and Merge - Split

We start by taking the whole image to be one region

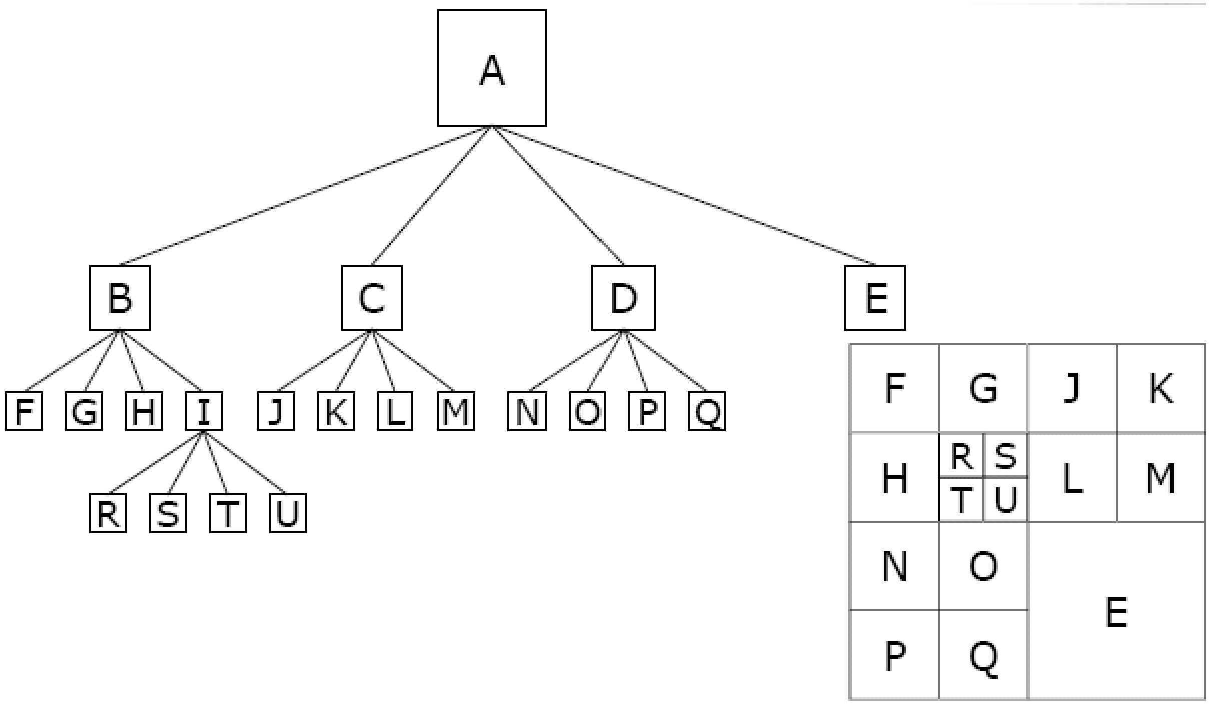
- We compute some measure of internal similarity
- If this indicates there is too much variety, we divide the region
- Repeat until no more splits, or we reach a minimum region size

Some details are needed

- How to we measure similarity – standard deviations are commonly used
- How do we determine whether to split or not
  - thresholding is easy
- How do we split regions – quadtrees are a common method

Split and Merge - Split

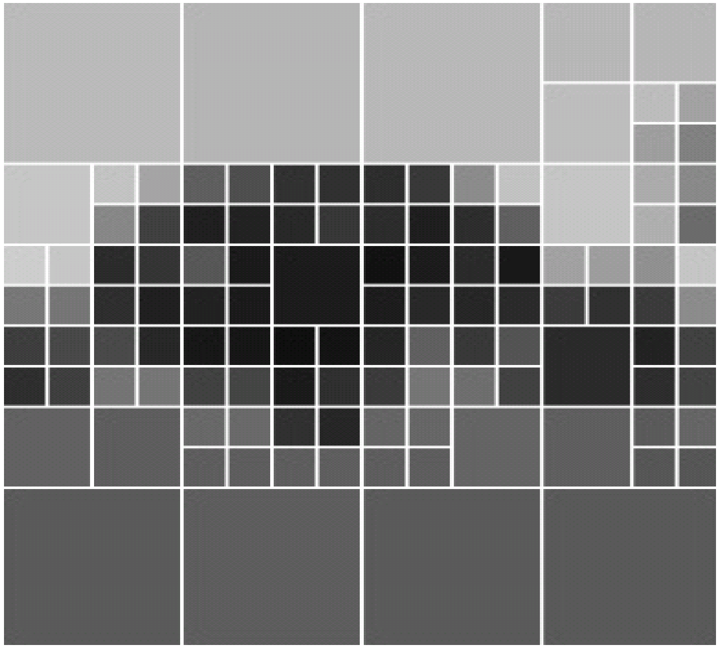
- Quadtrees



Split and Merge - Split

We'll use the tree image again

- Splitting based on intensity (could use something else)
- Splitting based on standard deviation, with a threshold of 25
- Split using a quadtree with a maximum of 5 levels



Split and Merge - Merge

Splitting gives us

- Regions that are small, consistent, or both
- Rather too many regions, as adjacent ones may be very similar
- We can now combine adjacent regions to make bigger ones

Merging

- We merge two regions if they are adjacent and similar
- Need a measure of similarity – can compare their mean grey level, or use statistical tests
- Repeat the merging until you can do no more

Split and Merge - Merge

- We consider merging adjacent regions
- Two regions are merged if their mean grey levels differ by less than 25
- This leads to less regularly shaped regions, but they are larger and still consistent

Split and Merge

Define a measure of internal region dissimilarity (e.g. entropy of colour histogram), a similarity threshold and minimum region size.

1. Start by taking the whole image to be one region.
2. Compute internal similarity.
3. Divide the region in 4 equal parts if the measure exceeds the threshold (indicates there is too much variety).
4. Repeat steps 2 and 3 for each part until there are no more splits, or we reach a minimum region size.
5. Compute internal similarity for each pair of adjacent regions at the bottom level of the tree.
6. Merge the regions if their similarity is higher than the threshold.
7. Repeat steps 5 and 6 until no merge is possible.

Normalised Cuts

Split and merge uses a regular division of the image

- This leads to a blocky final segmentation
- The regular division is what leads to an over segmentation by splitting, it forces e.g 4 sub-regions to be created, fewer or more may be better

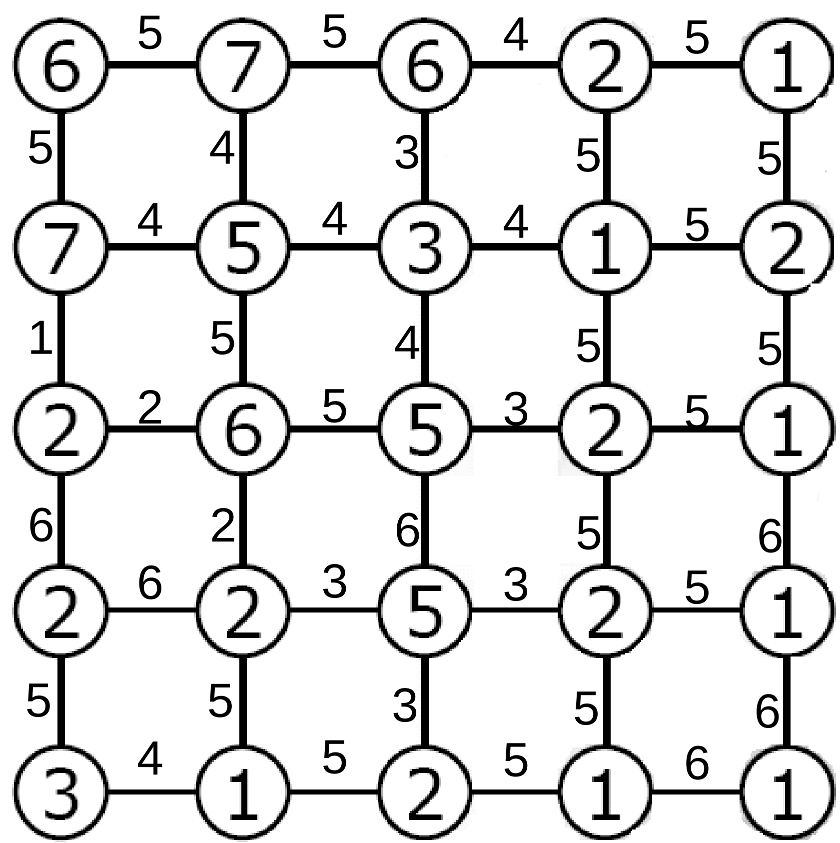
One alternative is normalised cuts

- It aims to do the splitting so that the division is optimal
- This gives better definition around the edges
- It (should) remove the need to merge regions after splitting

Normalised Cuts

Is based on graph theory

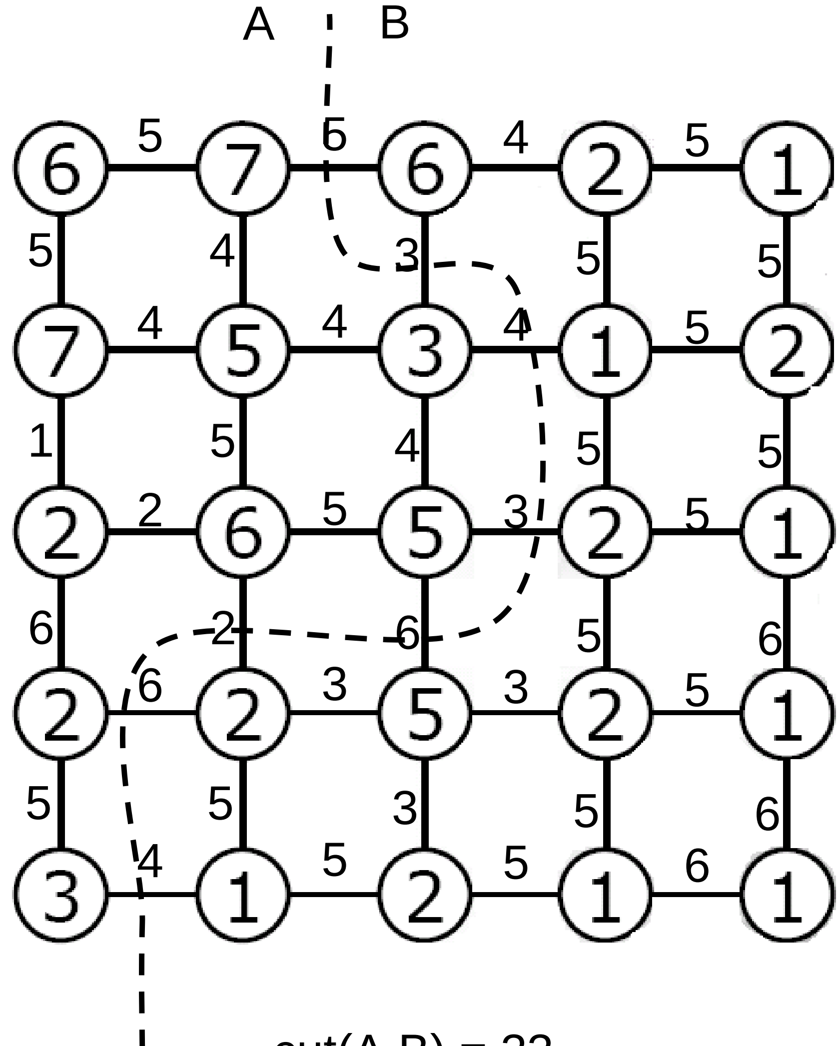
- Graphs are used to represent images
- Each pixel is a vertex in the graph, *vi*
- Edges, *e=(vi,vj)* link adjacent pixels, and are given weights so that if *vi* and *vj* are similar *w(vi, vj)* is large



Graph Cuts

- A *cut* divides the vertices of a graph into 2 sets, *A* and *B*
- The weight of the cut is the sum of the edges between vertices in *A* and *B*

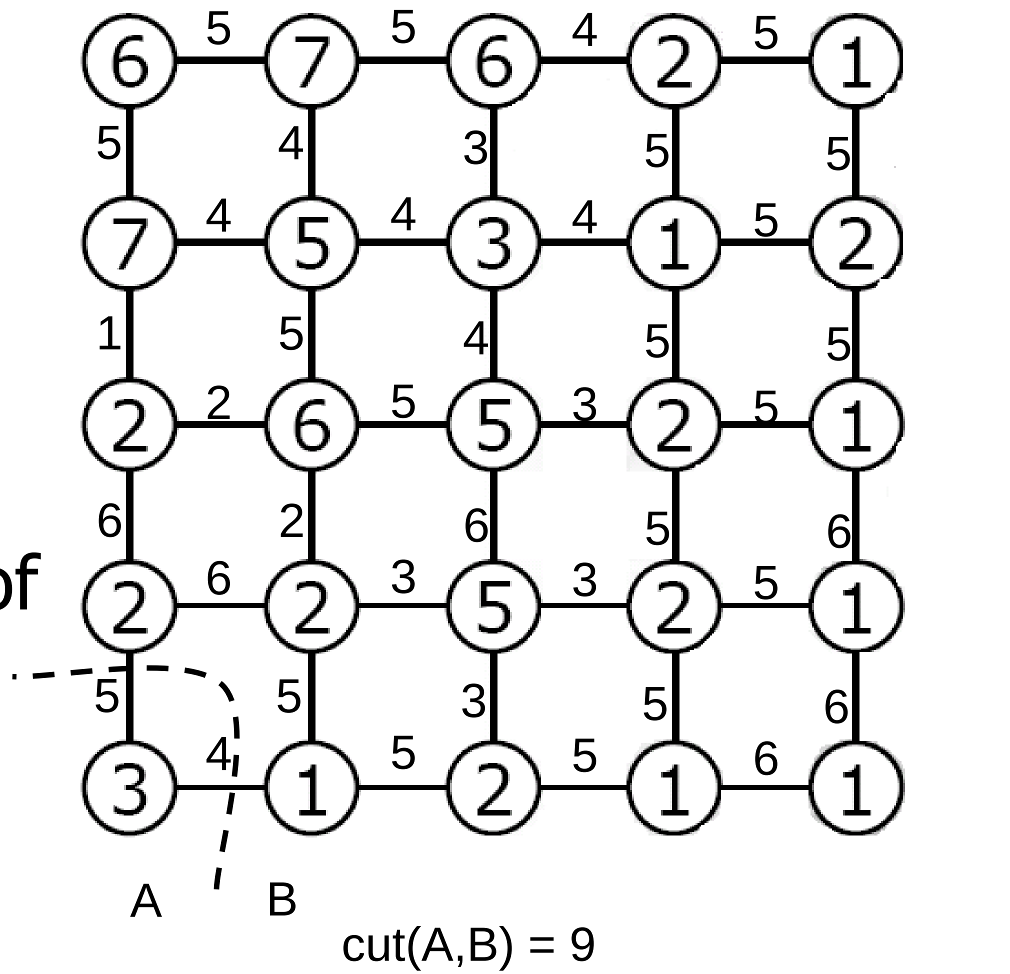
$$cut(A,B) = \sum_{u \in A,\, v \in B} w(u,v)$$



cut(A,B) = 33

Cuts and Segmentation

- We divide on cuts to segment an image
- We want the cuts to have a low weight
- This means that they don’t separate similar pixels
- However, the weight of a cut depends on its length, so short cuts have low weights



Normalised Cuts

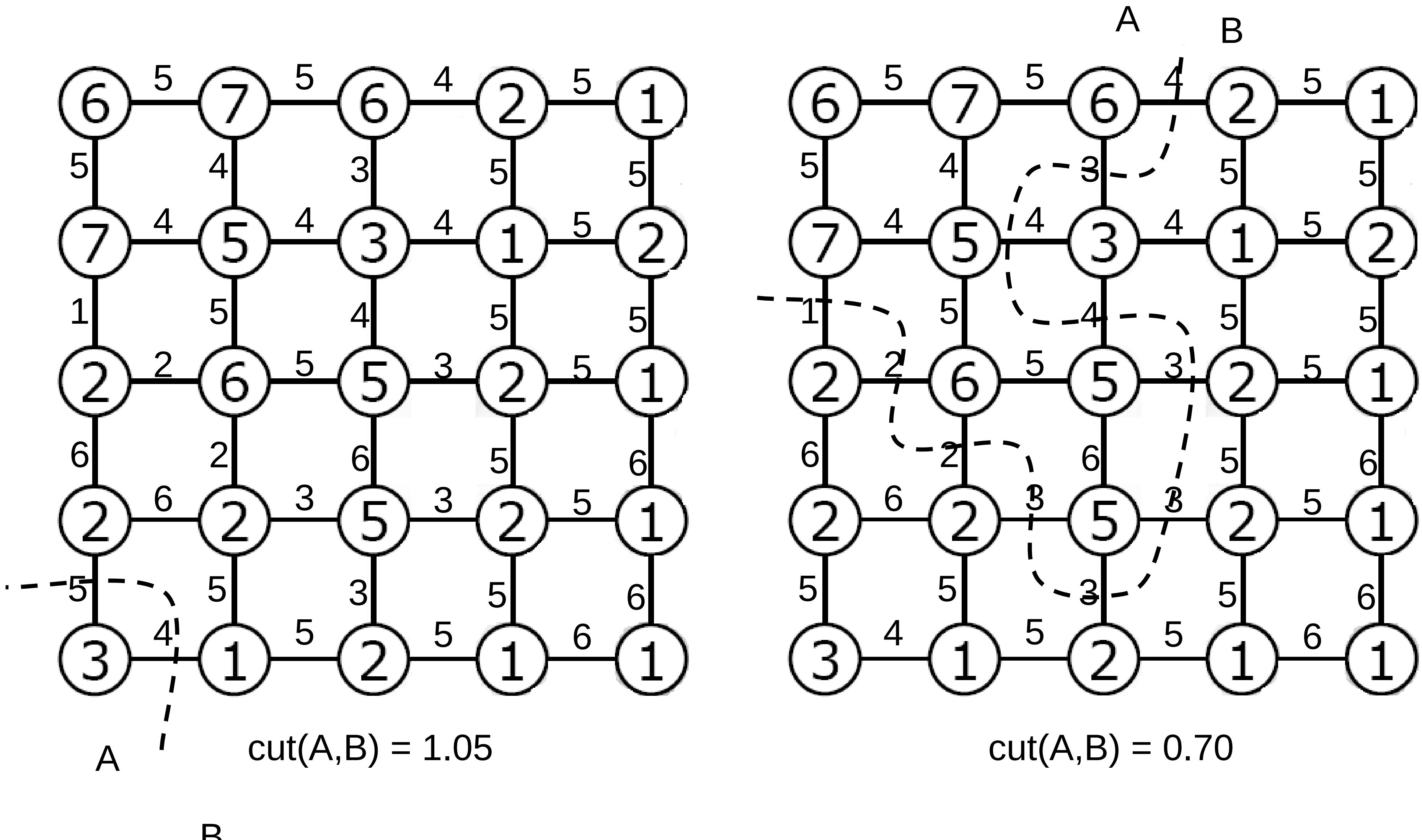
- A normalised cut removes this bias
- It also takes into account the total weights of edges in the two sets, called the association where V is the set of all vertices in the graph

$$assoc(A,V) = \sum_{u \in A,\, v \in V} w(u,v)$$

A split is now made along the normalised cut with the smallest weight, given by

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

Normalised Cuts



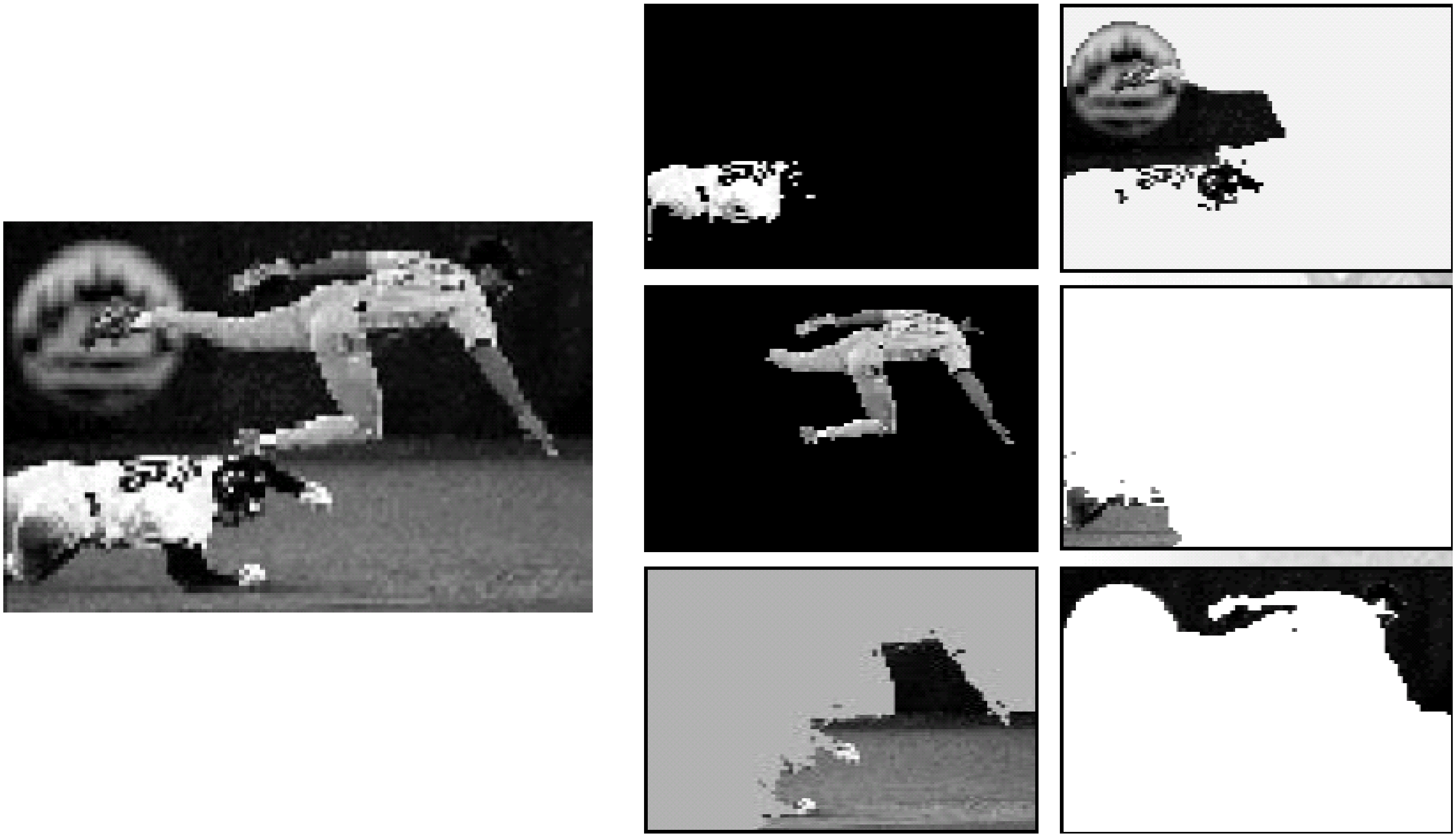
$$assoc(A,V) = \sum_{u \in A,\, v \in V} w(u,v)$$

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

Computing Normalised Cuts

- We want to find a minimal cut
- It turns out that this is NP-complete so we need $O(2^n)$ time to solve it in general n here is the number of edges – about 3 billion for a 320x240 image, so 2^n is very large indeed
- An approximate solution is found
- This is based on a branch of maths called spectral graph theory

Normalised Cuts Example



Other Segmentation methods

- Watershed
- Expectation Maximisation EM + probabilistic models

Revision

- Segmentation methods
  - Pixel based classification
  - Region growing
  - Split and merge
  - Normalized cuts