Image Segmentation: Breaking it down

Matt Arnold

The talk about Image Segmentation: Breaking it down

1. Introduction
   - What is Image Segmentation, and why is it useful?
   - Applications
2. Binary thresholding
   - Otsu's Algorithm
   - Maximum Entropy
   - Balanced Histogram
3. K-Means applied to images
4. Graph-based methods
   - Region Growing
   - Minimum Spanning Tree-based
5. Other techniques
6. Conclusions

1/24

What is Image Segmentation?

What is Image Segmentation?

- Simplify an image into a set of segments
- Easier to analyse

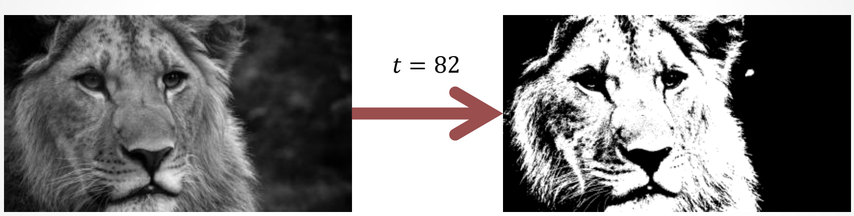
K-Means: k = 3

Why Image Segmentation?

- Content-based image retrieval
- Machine vision
- Medical imaging
- Automatic aesthetic image quality analysis
- Object detection and recognition
  - Pedestrian detection
  - Video surveillance
  - Face detection/recognition
  - Fingerprint recognition

Binary Thresholding

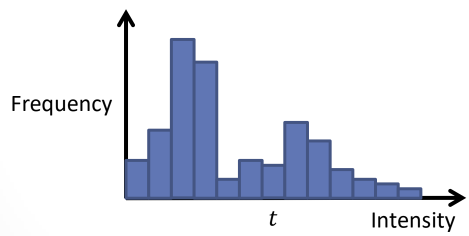
- Reduces a greyscale image to a binary image



$0 \leq p_{ij} \leq 255$ $p_{ij} \in \{0, 255\}$

Binary Thresholding

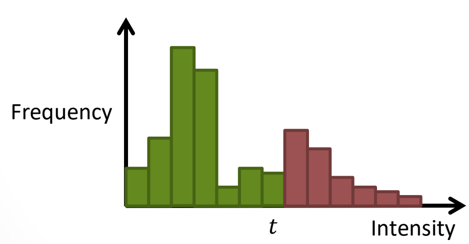
- Reduces a greyscale image to a binary image
- Not necessarily intensity, but we will assume
- Performs a split on the histogram



4/24

Binary Thresholding

- Reduces a greyscale image to a binary image
- Not necessarily intensity, but we will assume
- Performs a split on the histogram



4/24

Binary Thresholding

- Reduces a greyscale image to a binary image
- Not necessarily intensity, but we will assume
- Performs a split on the histogram
- Key is in selecting the threshold value
- Assumes bimodal intensity probability distribution (histogram)

Otsu's Method

- Exhaustive
- Return threshold that minimises the *intra-class variance*:

$$\sigma_w^2(t) = \omega_0(t)\sigma_0^2(t) + \omega_1(t)\sigma_1^2(t)$$

- Weighted sum of the variances of the two classes
- Equivalent to maximising *inter-class variance:*

$$\sigma_b^2(t) = \omega_0(t)w_1(t)(\mu_0(t) - \mu_1(t))^2$$

Maximum Entropy

- Exhaustive, like Otsu
- Maximises the sum of the entropies of both segments

$$T = \underset{t=0,\ldots,255}{\operatorname{argmax}}(H_B(t) + H_W(t))$$

probability of a pixel being a certain intensity

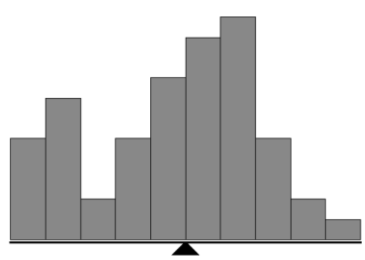
$$H_B(t) = -\sum_{i=0}^{t} \frac{h(i)}{\sum_{j=0}^{t} h(j)} \log \frac{h(i)}{\sum_{j=0}^{t} h(j)}$$

$$H_W(t) = -\sum_{i=t+1}^{255} \frac{h(i)}{\sum_{j=t+1}^{255} h(j)} \log \frac{h(i)}{\sum_{j=t+1}^{255} h(j)}$$

- Finding a compression that minimises total information lost

6/24

Balanced Histogram



- Histogram has physical weight
- Pivot starts in centre, and a bar falls off the heaviest end
- Repeats until only one bar remains

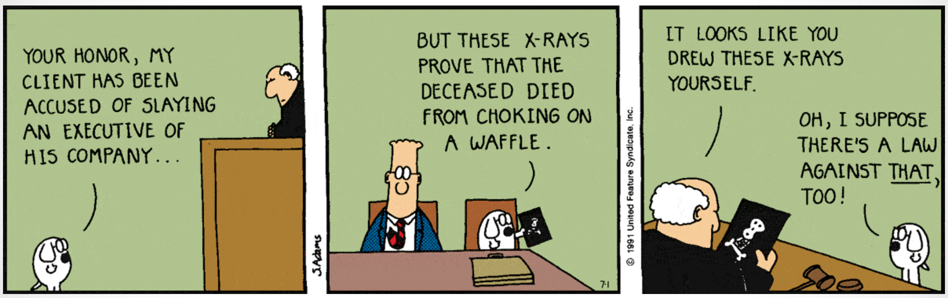
7/24

Demo

Binary Thresholding

- Main advantage: much faster than more involved techniques
  - Histogram holds much less data than the image
  - Computing histogram only requires one pass
- Main drawback: it only creates two segments
  - Difficult to apply to multi-channelled data
- Sensitive to noise and/or Intensity Inhomogeneity
  - IH unnoticeable to humans
- Circular thresholding

8/24



K-Means

- Iterative unsupervised clustering algorithm
- Input: set of data points, int k > 0, function d(p, q)
- Output: k disjoint, mutually spanning sets of data points

1. For each cluster, select an initial cluster mean.
   - Random from data space
   - Random from data points
   - K-Means++
2. Assign each data point to its 'closest' cluster.
3. Re-compute the cluster means using the recent assignments.
4. Repeat steps 2 and 3 until no changes.

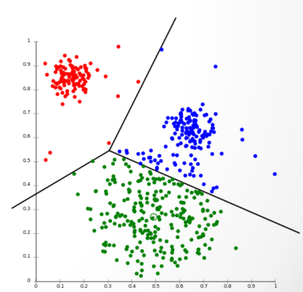


Image segmentation

K-Means

- Iterative unsupervised ~~clustering~~ algorithm
- Input: set of data points, int k > 0, function d(p, q)
- Output: k disjoint, mutually spanning sets of data points

1. For each cluster, select an initial cluster mean.
   - Random from data space
   - Random from data points
   - K-Means++
2. Assign each data point to its ‘closest’ cluster.
3. Re-compute the cluster means using the recent assignments.
4. Repeat steps 2 and 3 until no changes.

K-Means

- Iterative unsupervised image segmentation algorithm
- Input: set of ~~data points~~, int k > 0, function d(p, q) pixels
- Output: k disjoint, mutually spanning sets of ~~data points~~

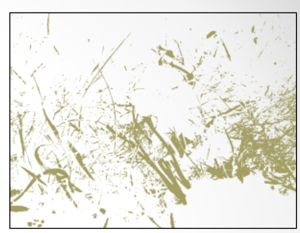
1. For each cluster, select an initial cluster mean.
   - Random from ~~data space~~
   - Random from ~~data points~~
   - K-Means++
2. Assign each ~~data point~~ to its ‘closest’ cluster.
3. Re-compute the cluster means using the recent assignments.
4. Repeat steps 2 and 3 until no changes.

K-Means

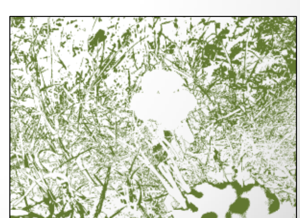
- Iterative unsupervised clustering algorithm
- Input: Image, int k > 0, function d(p, q)
- Output: k disjoint, mutually spanning sets of pixels

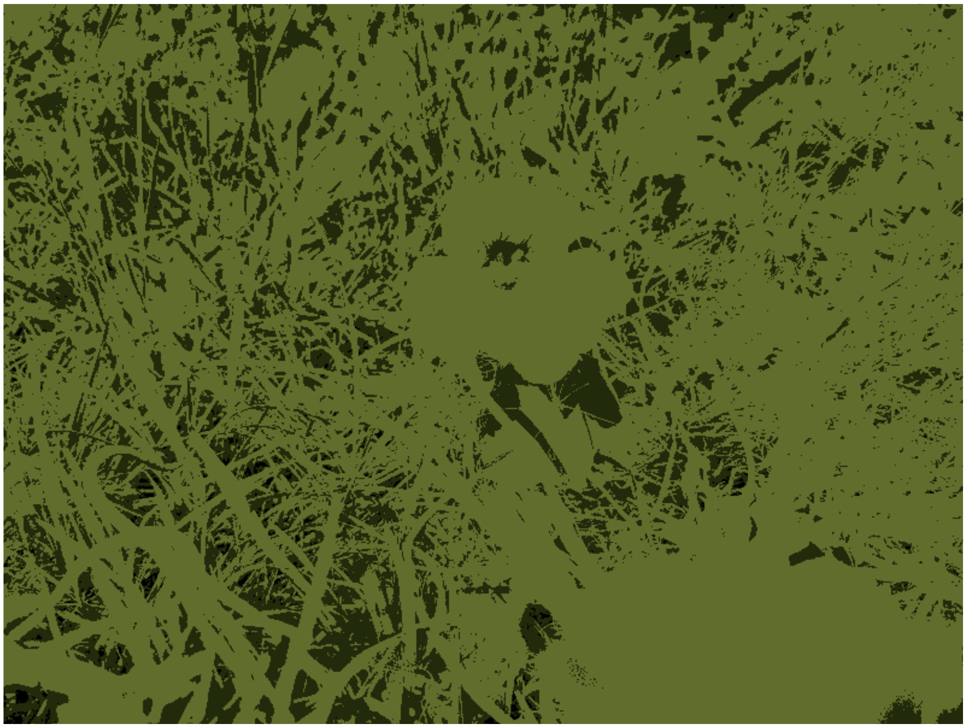
1. For each cluster, select an initial cluster mean.
    - Random from pixel colour space
    - Random from pixel colours
    - K-Means++
2. Assign each pixel to its 'closest' cluster.
3. Re-compute the cluster means using the recent assignments.
4. Repeat steps 2 and 3 until no changes.

K-Means



k = 4






k = 4
k = 8

i = number of iterations

K-Means

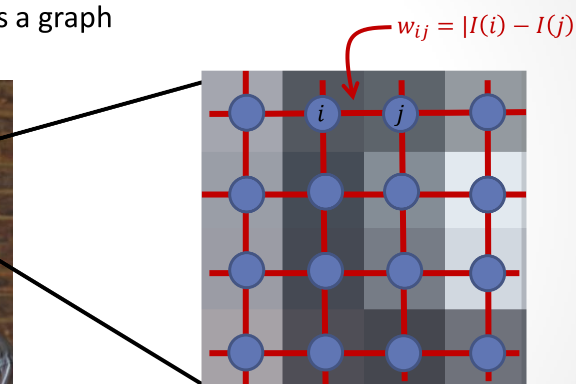
- Doesn’t incorporate spatial modelling
- We expect a segment of an image to be connected

K-Means

- Doesn't incorporate spatial modelling
- We expect a segment of an image to be connected
- Also susceptible to noise

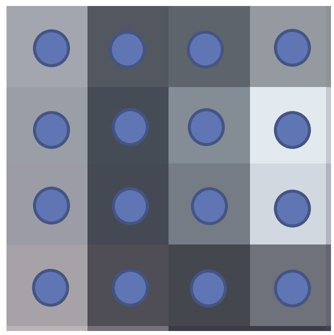
Graph-based methods

- Treat the image as a graph



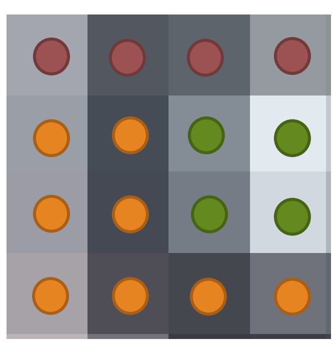
Graph-based methods

- Treat the image as a graph
- Segmentation is a partition of the vertices into *components*



Graph-based methods

- Treat the image as a graph
- Segmentation is a partition of the vertices into *components*

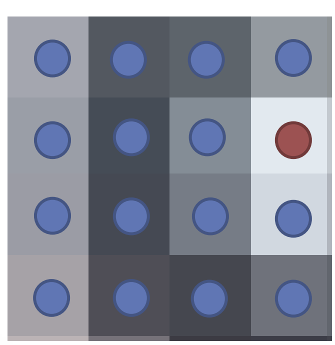


Graph-based methods

- Treat the image as a graph
- Segmentation is a partition of the vertices into *components*
- Incorporate local structure, by considering only neighbours

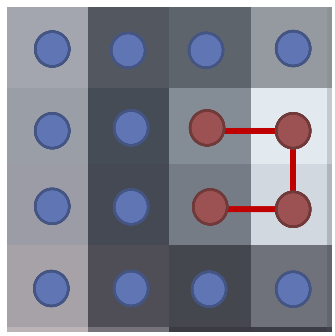
Region Growing

- Seed is chosen by user. From here, "connected" pixels are added to the segment



Region Growing

- Seed is chosen by user. From here, “connected” pixels are added to the segment



Region Growing

- Seed is chosen by user. From here, “connected” pixels are added to the segment
- Repeated until no new pixels are added
- Multiple seeds for multiple segments
- Have to manually mark each image, and know number of regions from start
- Unseeded region growing: but noise, and no global view of problem

MST-based

- Compute the Minimum Spanning Tree of the graph
- Then remove edges that are too heavy by some criteria
- Result is a spanning forest of MSTs

Felzenszwalb & Huttenlocher

- Based on Kruskal's MST algorithm
- Only allows edges to be added if they pass a certain criterion based on the components they merge
- Only merge two components if they are below a threshold of 'difference' from each other

Felzenszwalb & Huttenlocher

$$Int(C) = \max_{e \in MST(C,E)} w(e)$$

$$Dif(C_1, C_2) = \min_{v_i \in C_1, v_j \in C_2} w((v_i, v_j))$$

$$MInt(C_1, C_2) = \min(Int(C_1) + \tau(C_1), Int(C_2) + \tau(C_2))$$

$$\tau(C) = \frac{k}{|C|}$$

Felzenszwalb & Huttenlocher

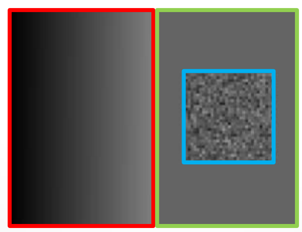
1. Sort edges into non-decreasing order
2. Start with each vertex as its own component
3. Considering the next edge (v_i, v_j), it will connect two components C_1 and C_2. If $C_1 \neq C_2$, and

   $$w\left((v_i, v_j)\right) \leq MInt(C_1, C_2)$$

   then merge the two components by adding the edge. Otherwise, do nothing
4. Repeat for all edges
5. Return the final segmentation

Felzenszwalb & Huttenlocher

- Based on Kruskal's MST algorithm
- Only allows edges to be added if they pass a certain criterion based on the components they merge
- Only merge two components if they are below a threshold of 'difference' from each other
- Satisfies global properties, despite being a greedy algorithm using pixel-wise comparisons



Other methods

- Watershed transformation: looking at colour/intensity gradient map
- Edge detection: extend edges to form region boundaries
- Histogram-based: includes methods already mentioned, but also analysing structure of a smoothed histogram
- Normalised cuts: top-down approach, which reduces graph partition to an eigenvector problem
- and many more…

Conclusions

- Many different forms of and approaches to image segmentation (e.g. Thresholding, Clustering, Graph-Based)
- Speed vs Accuracy
- Global vs Local
- Interactive vs Automatic

Thank you

Any questions?