CS-GY 6643
Computer Vision

Lecture 7: Segmentation

Prof. Erdem Varol

Today's menu

Announcements

Edges to segmentation

Mean-shift, Normalized Cuts and Graph-Based segmentation

Foreshadowing learning based methods

Coding lab

Announcements

Midterm exam next week (15% of grade)

- October 24, 11am. In-class, pen and paper.
  - No laptops.
  - Closed book.
  - May bring a single sheet of paper with written notes.
- Anything mentioned in lecture notes + Szeliski sub-chapters denoted in course website are fair game.
- Will test the ability to derive key mathematical concepts on the spot.
- The exam will be designed to be completed in 1 hour 15 mins although you are welcome to stay the full 2.5 hour time slot if needed.
- The style of questions will be like homework 2 with at least one proof based problem.
- There will be 4-5 questions with several subproblems.
- There will be 2 or 3 versions of exams distributed with altered inputs to prevent cheating.

Date	Topic	Material
September 5, 2024	Intro and survey of topics	Lecture 1 slides, Python tutorial, Google Colab Setup
September 12, 2024	Image formation and filtering	Lecture 2 slides, Colab for Lecture 2, Szeliski 2.2, 2.3, 3.1, 3.2
September 19, 2024	Non-linear filtering and edge detection	Lecture 3 slides, Colab for Lecture 3, Szeliski 3.3, 7.2
September 26, 2024	Image recognition, Feature detection and matching	Lecture 4 slides, Colab for Lecture 4, Szeliski 6,7
October 3, 2024	Contour tracking and Hough transforms	Lecture 5 slides, Colab for Lecture 5, Szeliski 7.3,7.4
October 10, 2024	Image alignment	Lecture 6 slides, Colab for Lecture 6, Szeliski 8.1,8.2
October 17, 2024	Segmentation	Szeliski 7.5

Midterm review session at next tuesday office hours

TA: Rishabh Raj

Email: rr4574@nyu.edu

Office Hours: Tuesdays 1-2pm on Zoom.

- Tuesday 1-2pm
- Zoom link
- We'll cover a set of sample questions and topics that will appear in the exam.

Project 2 grades and solutions out

If you have questions/disputes about your grades, please contact the TA's within 48 hours. After that we assume you are happy with the grade.

	10 runs	Image dims: 3000*2000
ID	Exec Time	Templ dims: 337*475
hz2095	0.926	
sw6071	0.952	
nar8991	1.007	
ap7949	1.222	
ak11115	1.22	
jd4587	1.356	
ck3740	1.434	

Top 7 students who have executed Q4 have received bonus points.

TA: Rishabh Raj

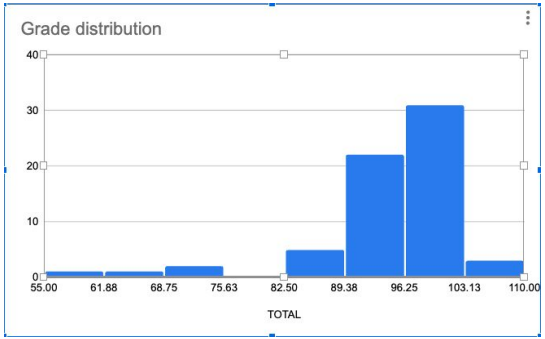
Email: rr4574@nyu.edu

Office Hours: Tuesdays 1-2pm on Zoom.

TA: Malhar Patel

Email: mkp6112@nyu.edu

Office Hours: Mondays 1-2pm on Zoom.



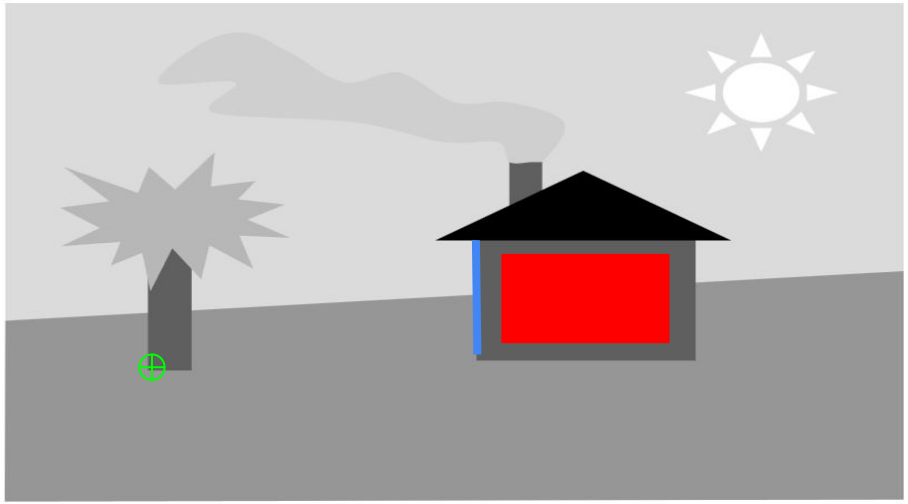
Segmentation

What is image segmentation?

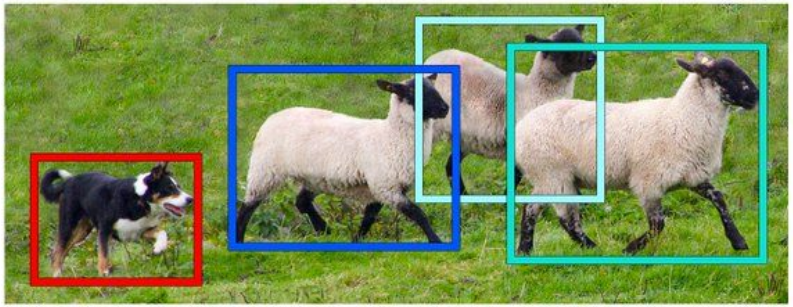
- Definition: Partitioning an image into meaningful regions

There is sparse amount of detail in each image quantified by flats, edges and corners.

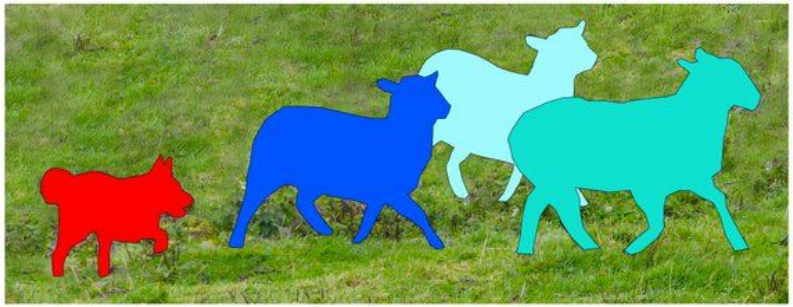
Segmentation = flats (mosty)



Difference between segmentation and detection



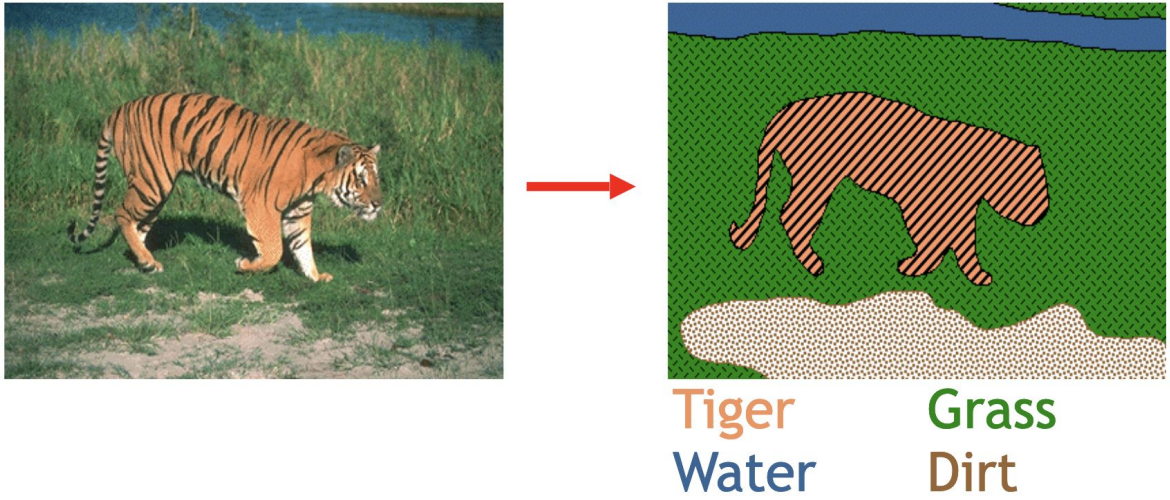
Object detection



Object segmentation

Types of segmentation

- Semantic Segmentation: Assign labels

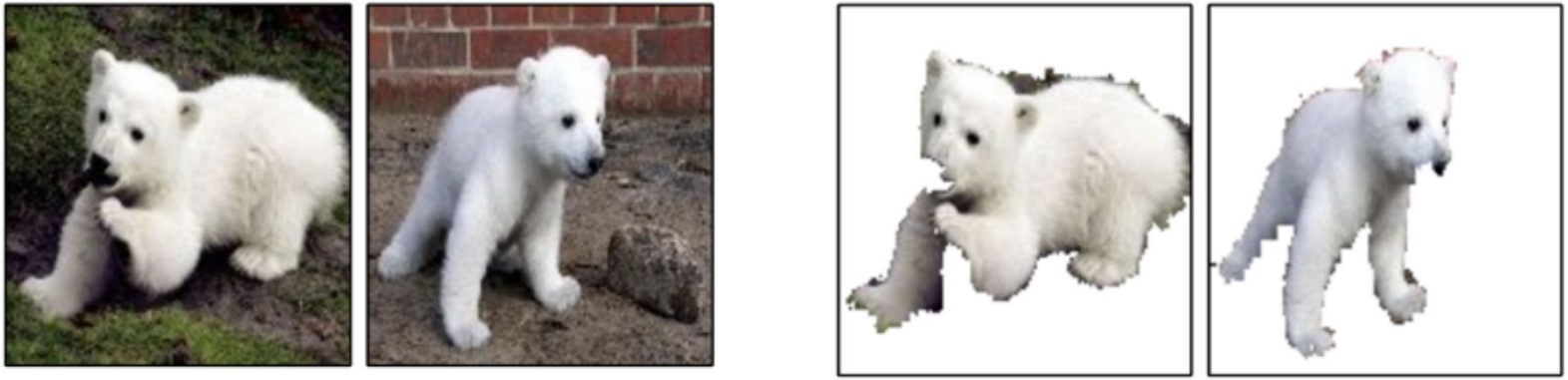


Types of segmentation

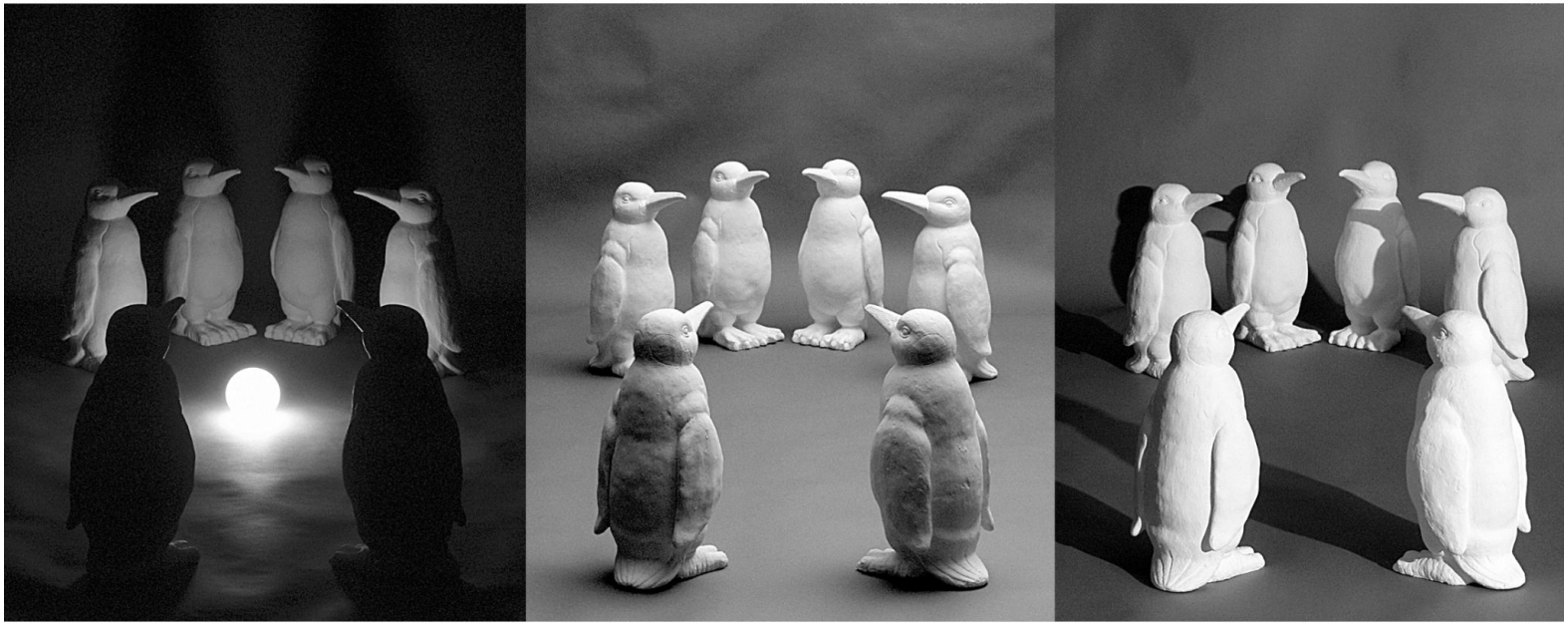
Foreground/background binary segmentation

Types of segmentation

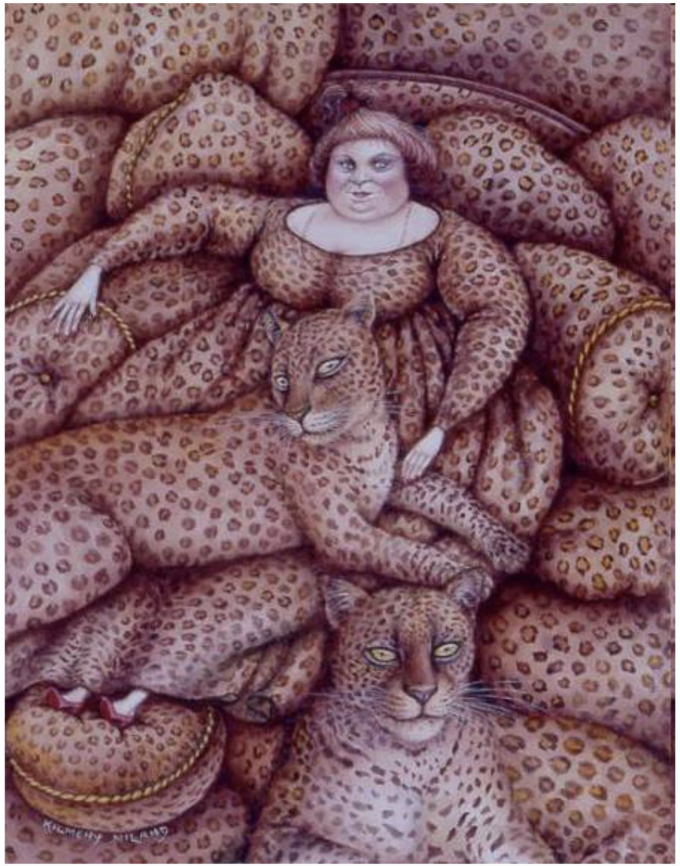
- Co-segmentation: Segment common object



Challenges: illumination

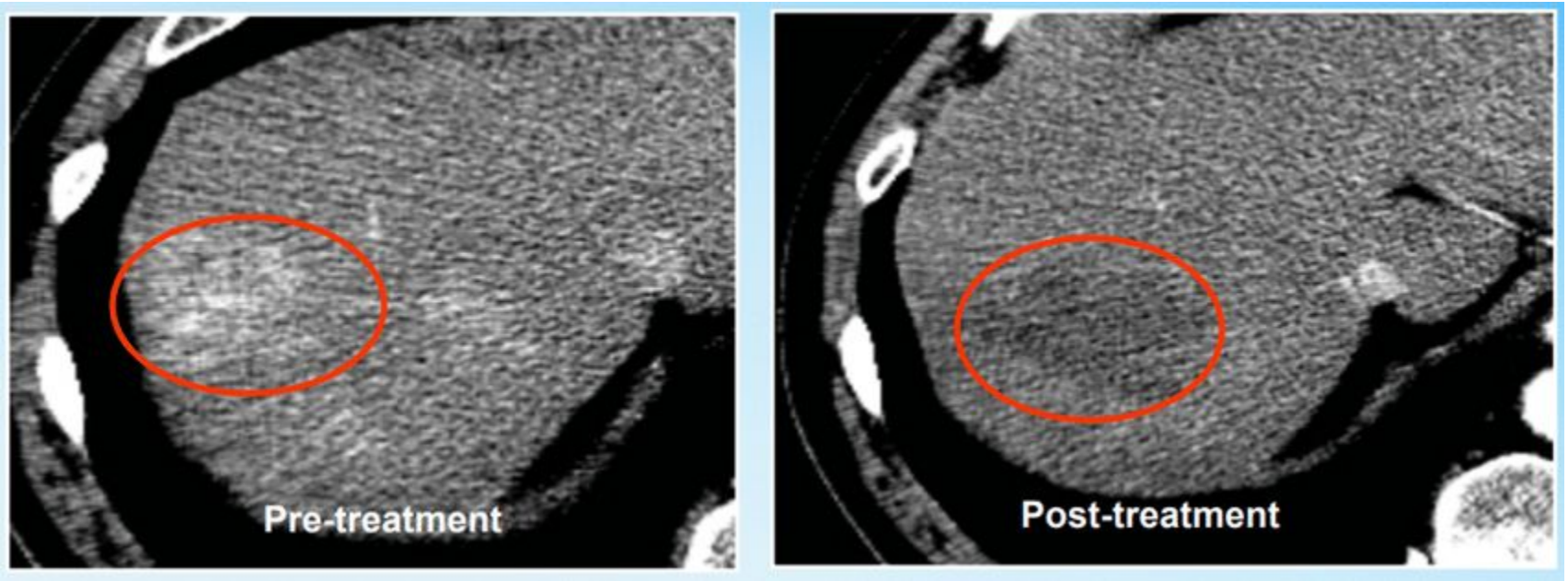


Challenges: background clutter



Kilmeny Niland. 1995

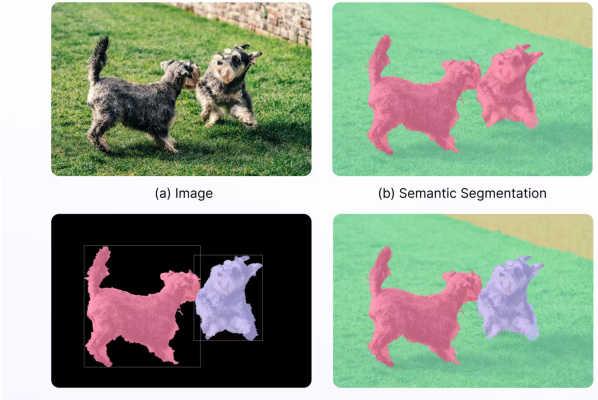
Challenges: unclear boundaries

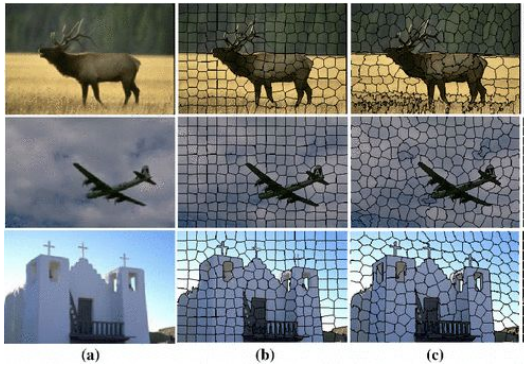


Brain tumor boundaries are vague

Why segment images?

- Object recognition - make it easier to classify objects
- Scene understanding - assign meaning to objects
- Image compression - sparsify images





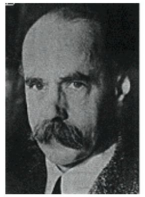
Gestalt theory and perception

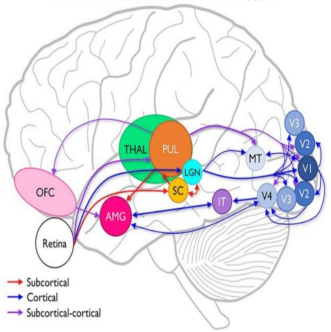
- Gestalt: whole or group
  - Whole is greater than sum of its parts
  - Relationships among parts can yield new properties/features

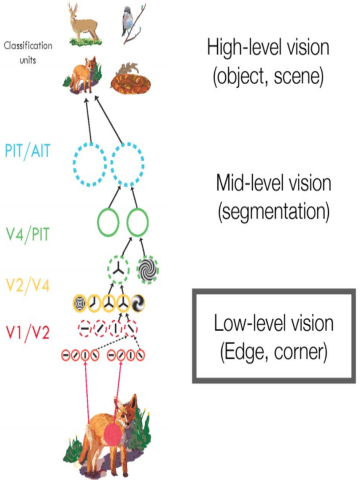
- Psychologists identified series of factors that predispose set of elements to be grouped (by human visual system)

"I stand at the window and see a house, trees, sky. Theoretically I might say there were 327 brightnesses and nuances of colour. Do I have "327"? No. I have sky, house, and trees."

Max Wertheimer
(1880-1943)

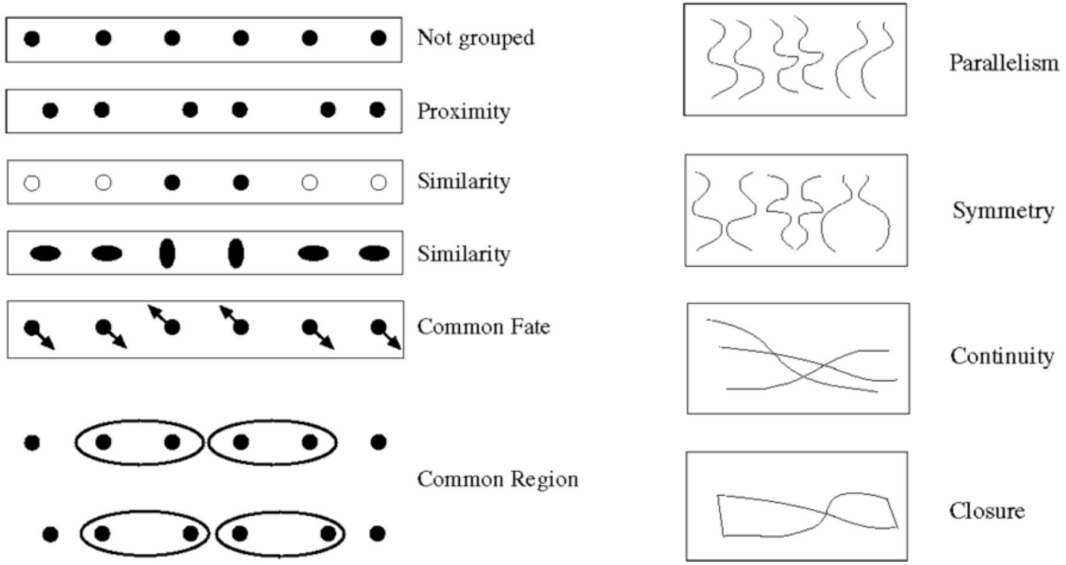




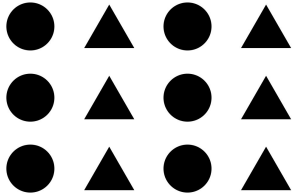


Higher order parts of our brains inherently "integrate" low level information to create meaning greater than the sum of its parts

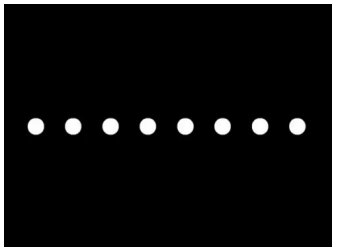
Gestalt examples



Closure example



Similarity example

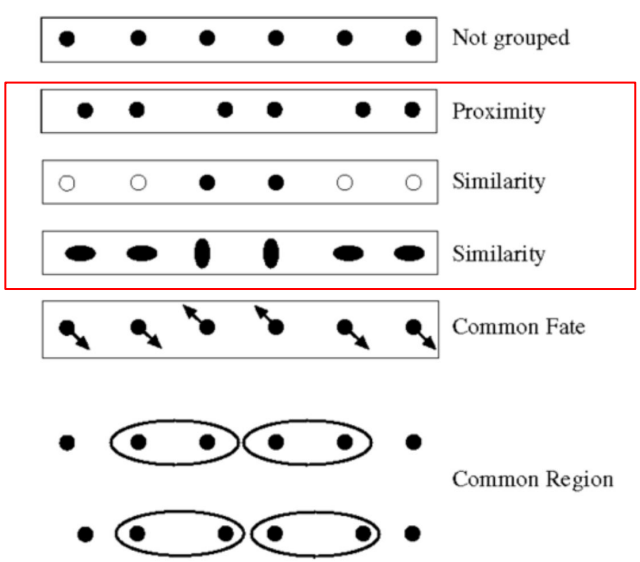


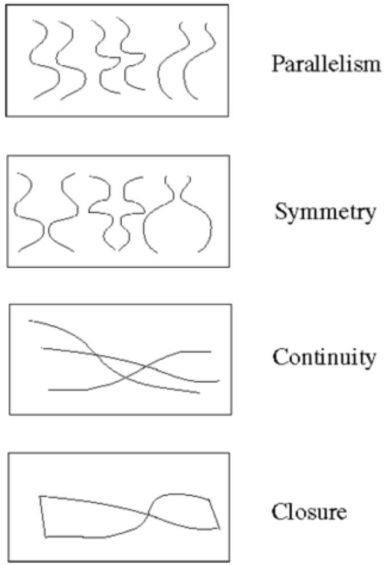
Common fate example

One of the goals of CV segmentation algorithms is to encode these concepts in mathematical models.

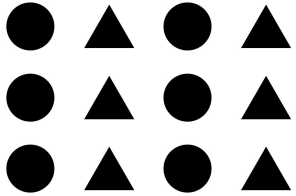
https://nanklesaria.medium.com/gestalt-principles-explained-through-gifs-d4c7893361d0

Gestalt examples

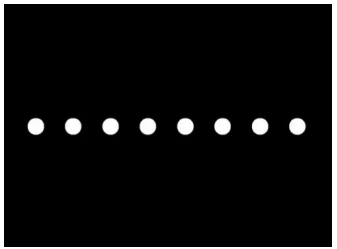




Closure example



Similarity example

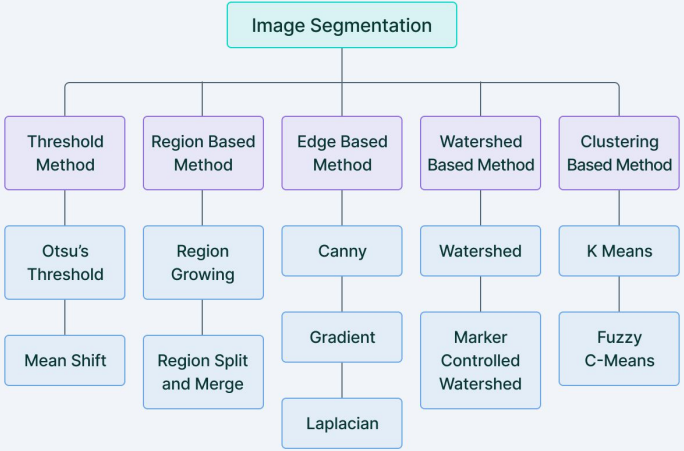


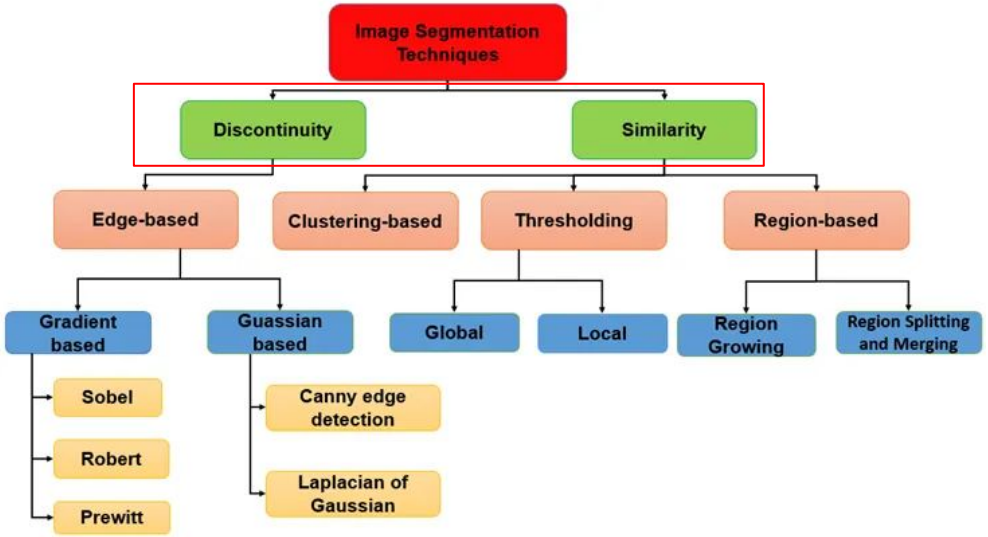
Common fate example

One of the goals of CV segmentation algorithms is to encode these concepts in mathematical models.

https://nanklesaria.medium.com/gestalt-principles-explained-through-gifs-d4c7893361d0

Types of segmentation algorithms - there are many taxonomies

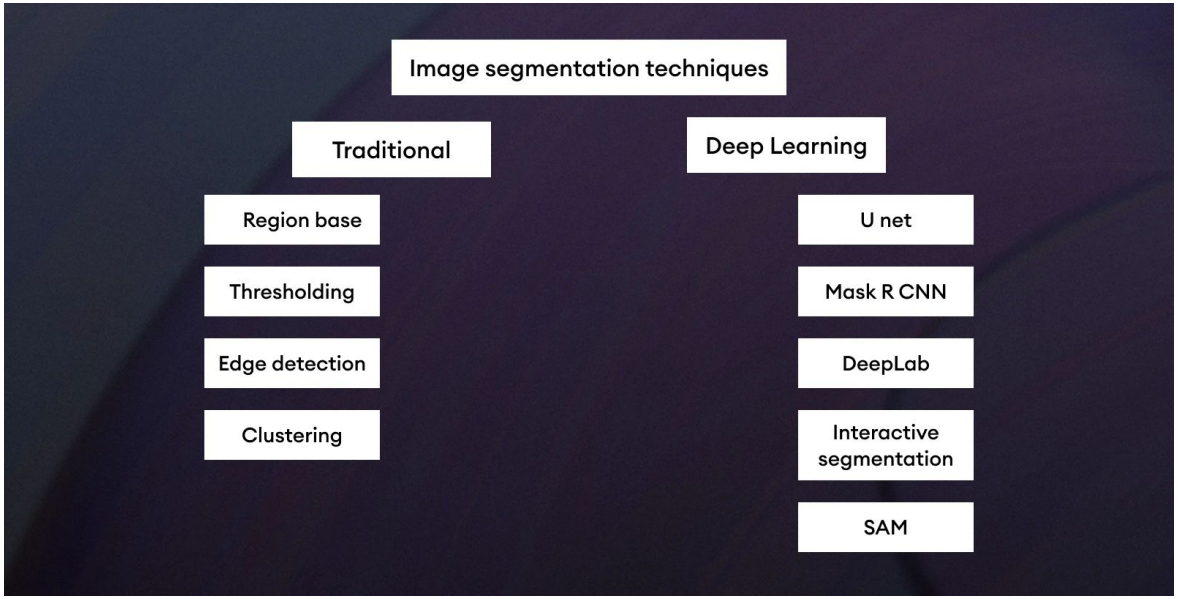




https://pooja-bagad18.medium.com/classification-of-image-segmentation-techniques-cc6a031b75fc

Image Segmentation Using Deep Learning: A Survey

In my opinion, the biggest divide is between model based vs. learning based approaches



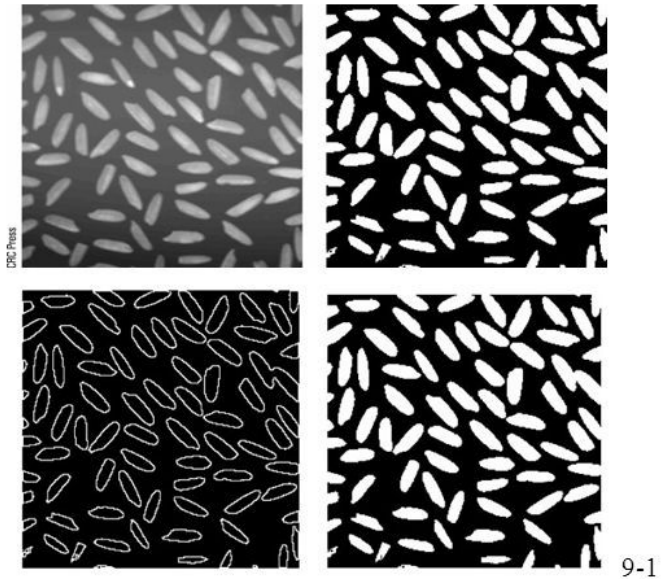
Today we will cover traditional methods as our last stop in model based computer vision - after midterm = learning based

Simplest segmentation methods we know already: thresholding and edge detection

Contents:

(1) Thresholding

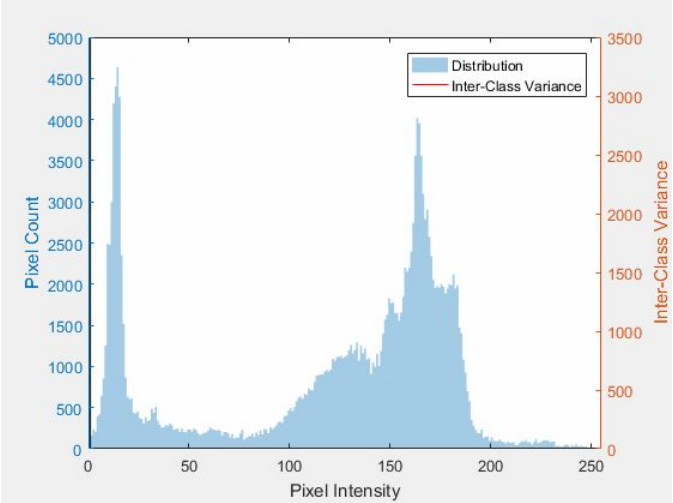
(2) Edge detection

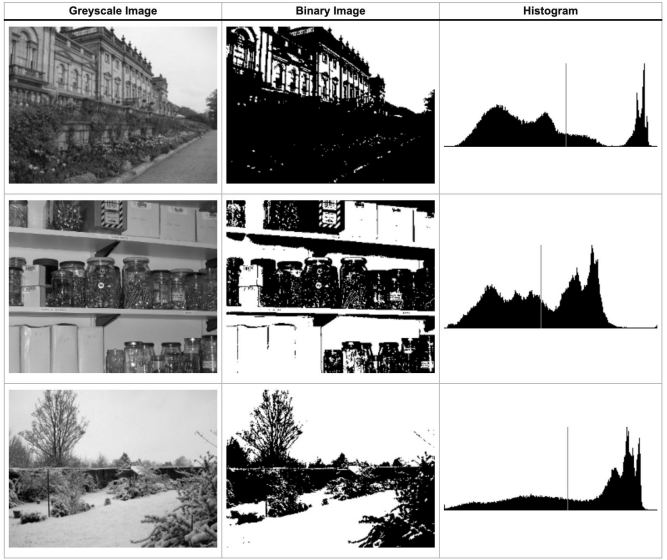


9-1

Intensity thresholding based approaches

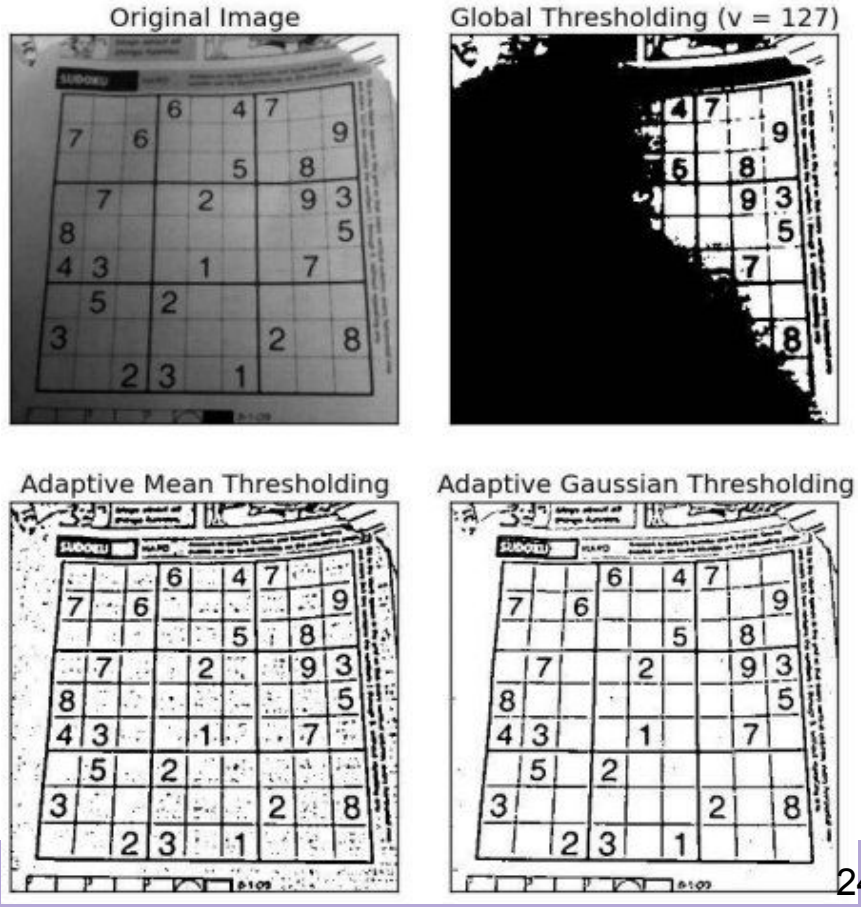
- Otsu's method:
  find a threshold that minimizes the intra-cluster variance
  bi-modal histogram



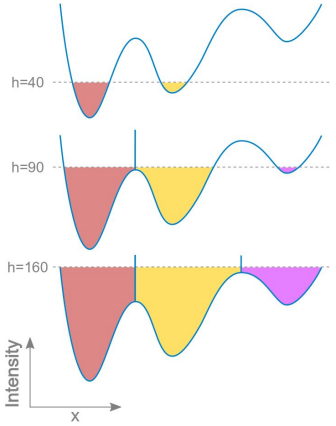


Intensity thresholding based approaches

- Adaptive thresholding (e.g Niblack's method)
  - If a pixel value is greater than N st.deviations from local mean
    - mask = 1
  - Otherwise
    - mask = 0



Intensity thresholding based approaches



Watershed

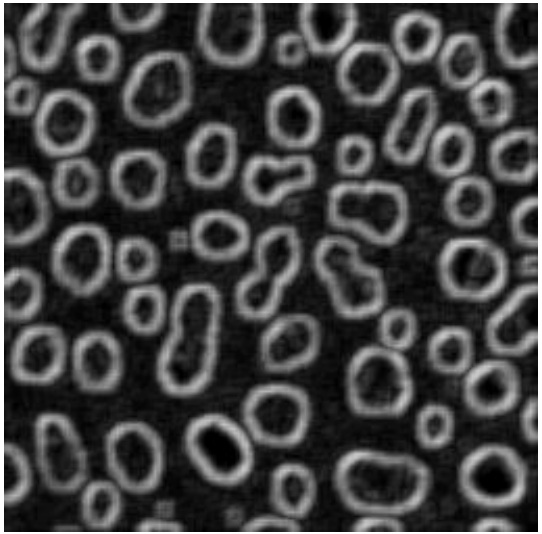
Initialize all local minima pixels as initial segments

Initialize T = “water level” at the value of lowest pixel.

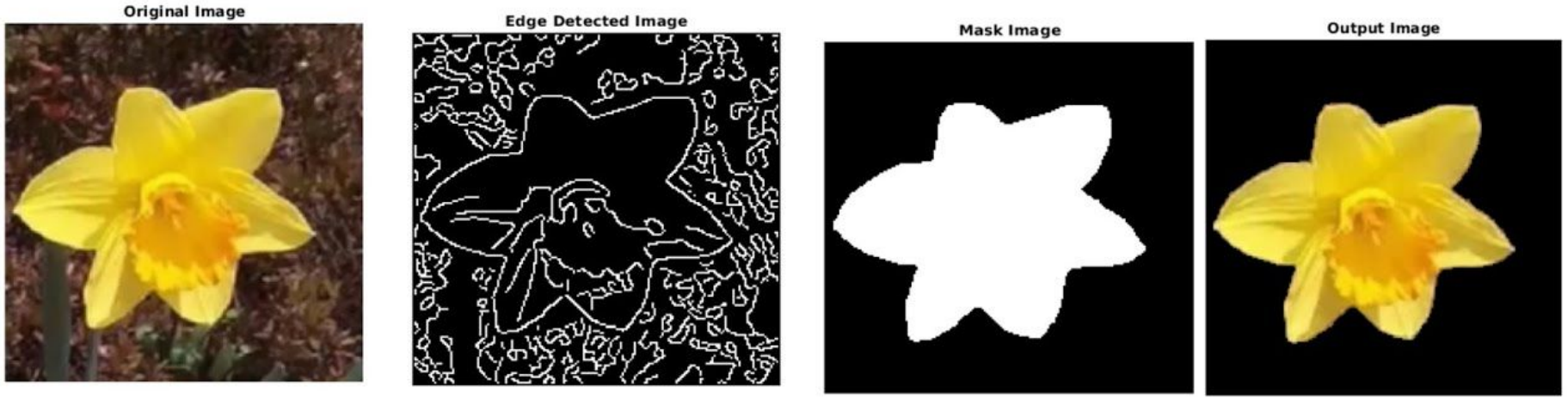
If an unassigned pixel in local neighborhood of a local minima (eg D8) has value T, connect this pixel to its closest local minima.

Do this for all pixels.

Increment T = T+1 until all pixels are assigned to a local minima.



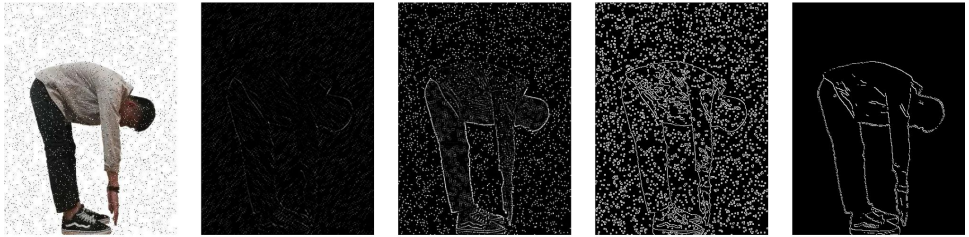
Edge based segmentation



Generate edge image (filter) → computer connected components

Limitations of edge and thresholding based segmentation methods

- Sensitivity to noise
- Incomplete/broken edges
- Can’t handle texture
- Assumption of binary context
- Prefer grayscale images, can’t handle complex scenes



(a) Image with noise **(b)** Sobel detection **(c)** Laplace detection **(d)** Canny detection **(e)** Proposed method

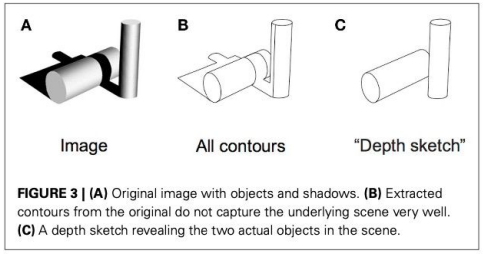


FIGURE 3 | (A) Original image with objects and shadows. **(B)** Extracted contours from the original do not capture the underlying scene very well. **(C)** A depth sketch revealing the two actual objects in the scene.

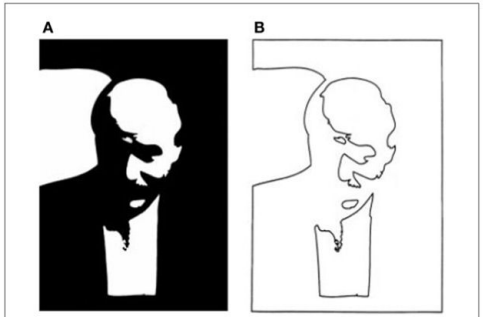


FIGURE 4 | (A) Two-tone image of Kennedy (1997) where the dark areas could be dark pigment or dark shadow. The face can be recovered only if the shadows are correctly identified. **(B)** A line version of the same contours is unrecognizable. The polarity information required for finding shadows is lost. The shadow boundaries can no longer be discounted and are taken as object contours that form meaningless shapes unrelated to the original.

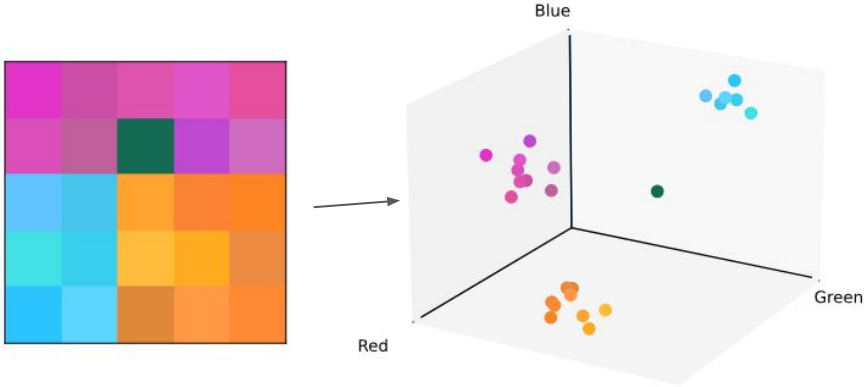
Can we generalize a bit better?

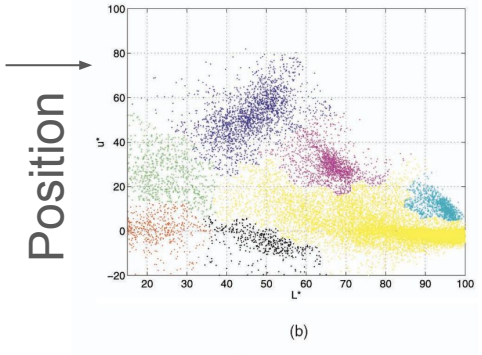
- Binarization/thresholding is a special case of clustering (with 2 clusters).
- **Image segmentation is can be thought of as a clustering problem:**

  group similar data points (pixels) together
- What are considered to be similar?

  represent each pixel with a feature vector (color intensity, locations, textures)
- Clustering methods:
  - Agglomerative clustering (bottom-up, region merging)
  - Divisive clustering (top down, region splitting)

Idea: represent all pixels as points in a feature space then cluster





Idea: represent all pixels as points in a feature space then cluster

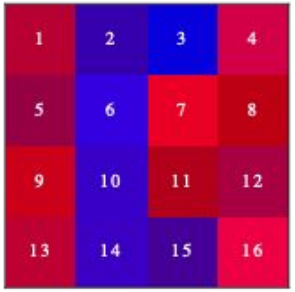
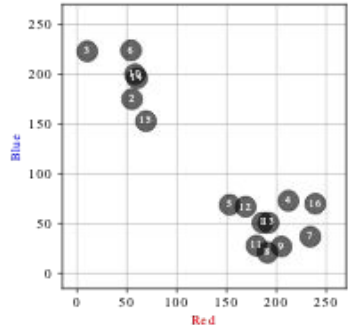


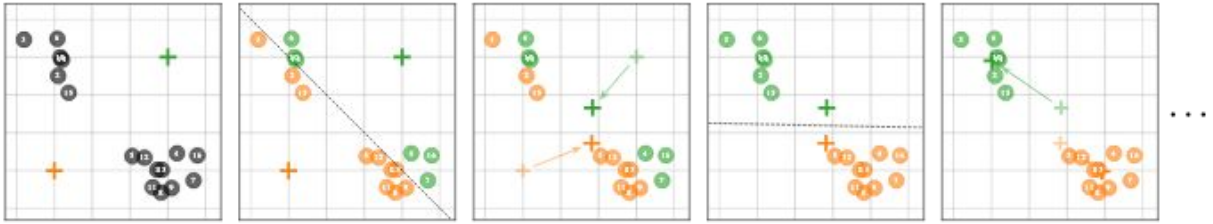
Image with bands red and blue.
Numbers are the pixel indices.

186 51	55 175	10 223	212 73
153 69	54 224	234 37	191 21
205 27	58 200	180 28	169 67
192 51	60 196	69 153	239 70

Intensities (red and blue)
of the pixels.



Pixels plotted in
the (red,blue) space.



Initialization | Iteration 1 Step A | Iteration 1 Step B | Iteration 2 Step A | Iteration 2 Step B | ...

Progression of the K-means algorithm (only the initialization and the first two iterations are illustrated).
Centroids are represented by +, the dotted line indicates the boundary between the two classes.

The simplest agglomerative clustering: *superpixels (e.g. SLIC)*

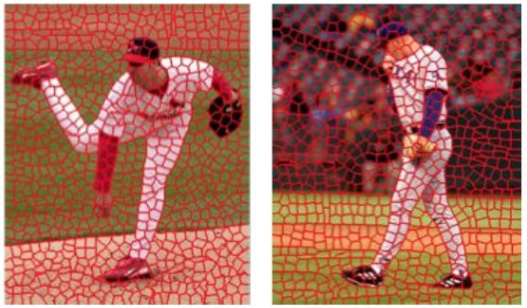
- much smaller region, oversegmented (shading field, recognition)
- dramatically reduces the complexity of images
- e.g., from 400K pixels to 200 superpixels
- Con: not semantically meaningful

$$L = \arg\min_{L} \min_{\{\mu_k\}} \sum_{k=1}^{K} \sum_{n:L[n]=k} \|I'[n] - \mu_k\|^2$$

n: pixels
μ: Cluster center (R^5)
I'[n]: augmented image (R^5)
L[n]: labeling

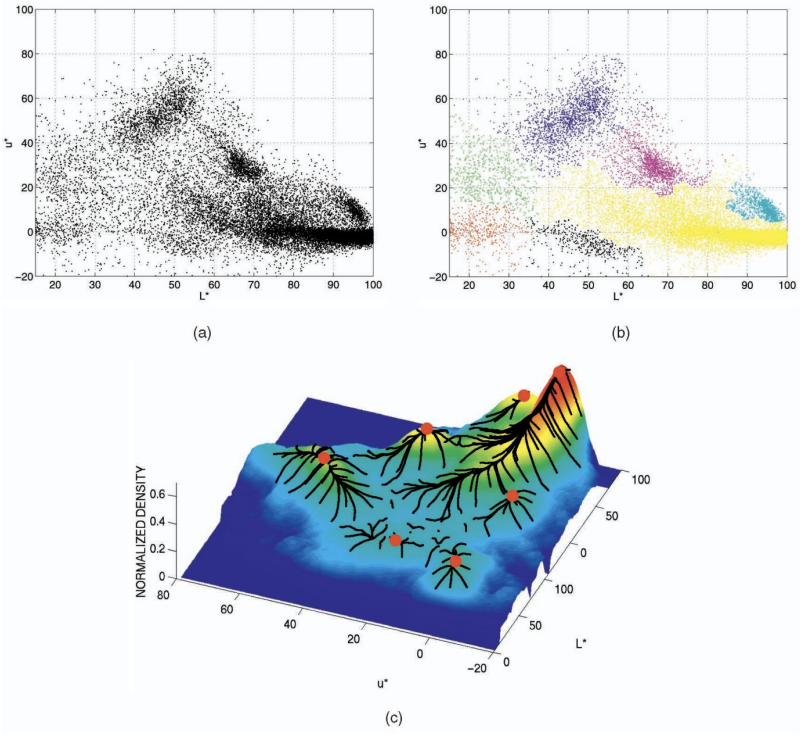
Image

Superpixels
(different levels)



Ren and Malik, 2003

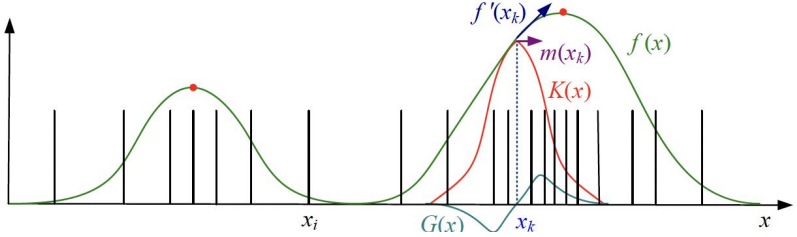
Mean shift



- Non-parametric clustering technique
- Key idea: look for modes in some feature space of pixels
- Use kernel density estimation and find the mode (red dots) by gradient ascent for each data point.

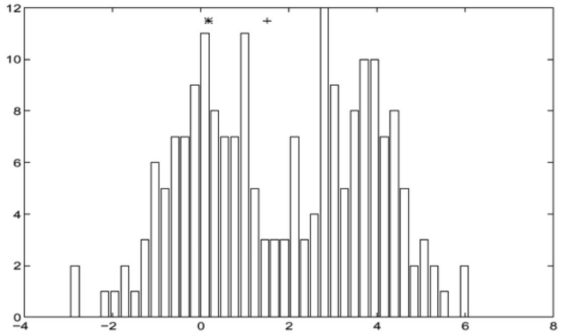
Comaniciu, 2002

Theoretical vs. practical implementation



$$f(\boldsymbol{x}) = \left(\frac{1}{n}\right) \sum_{i=1}^{n} K\left(\boldsymbol{x}_i - \boldsymbol{x}; h\right)$$

For each pixel, compute local density in feature space using kernel density estimation, take gradient steps towards local maxima (mode) until convergence.



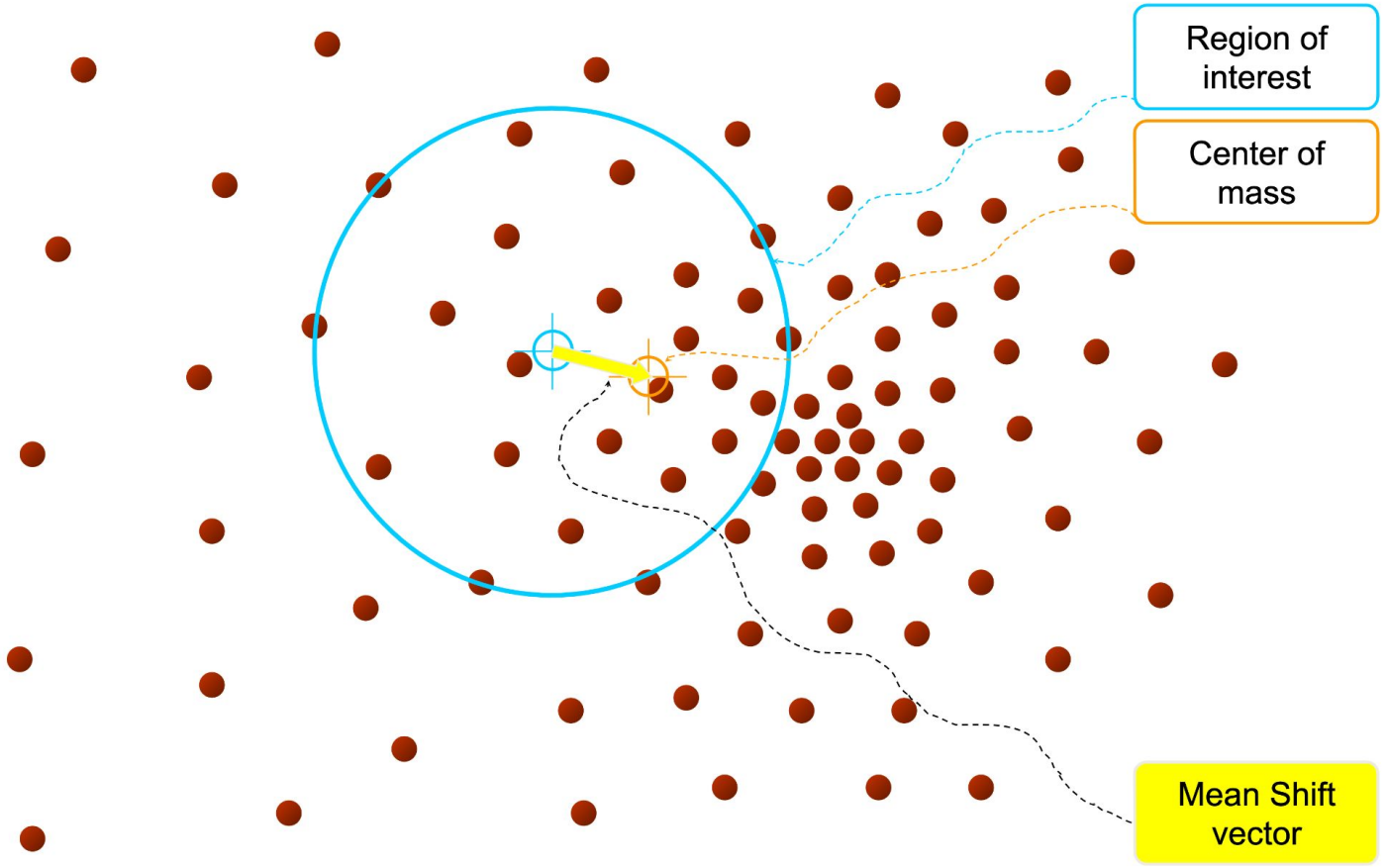
1. Initialize random seed, and window W
2. Calculate center of gravity (the "mean") of W: $\frac{1}{|W|}\sum_{x\in W} x$
   - Can generalize to arbitrary windows/kernels
3. Shift the search window to the mean
4. Repeat Step 2 until convergence

Only parameter: window size

Local density ⟷ Window

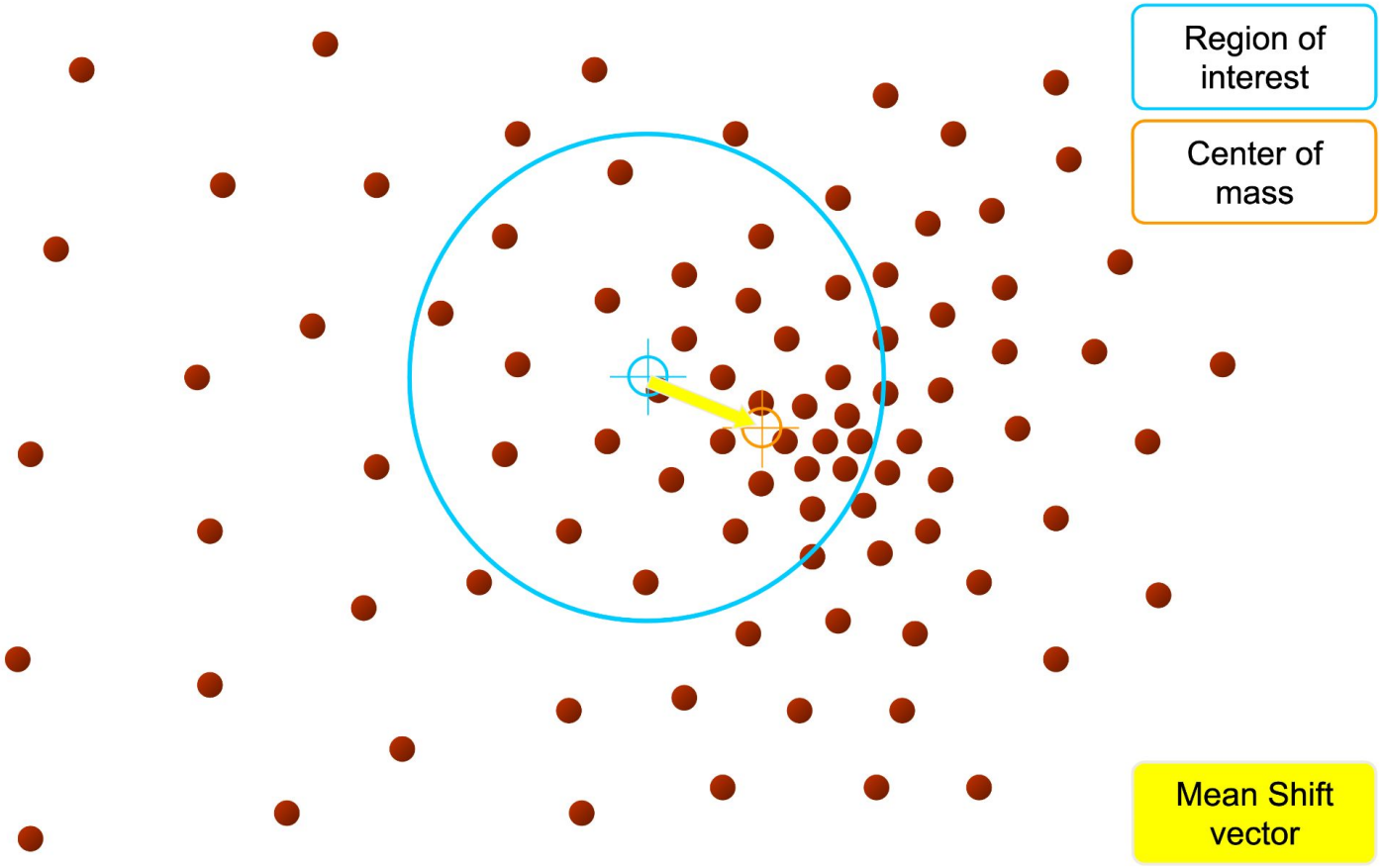
Mean-Shift

For each pixel, find the mode of the local density it belongs to



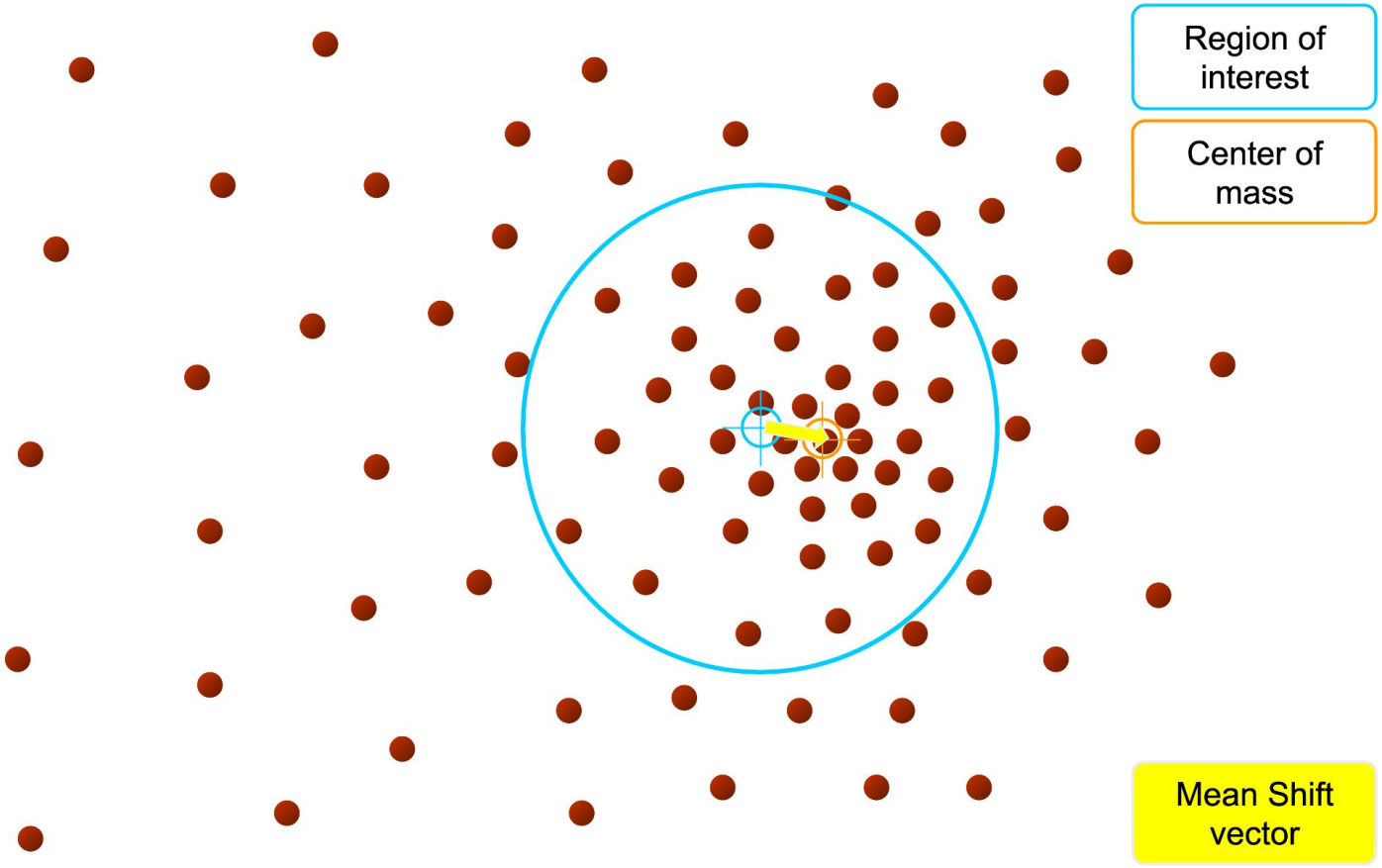
Mean-Shift

For each pixel, find the mode of the local density it belongs to



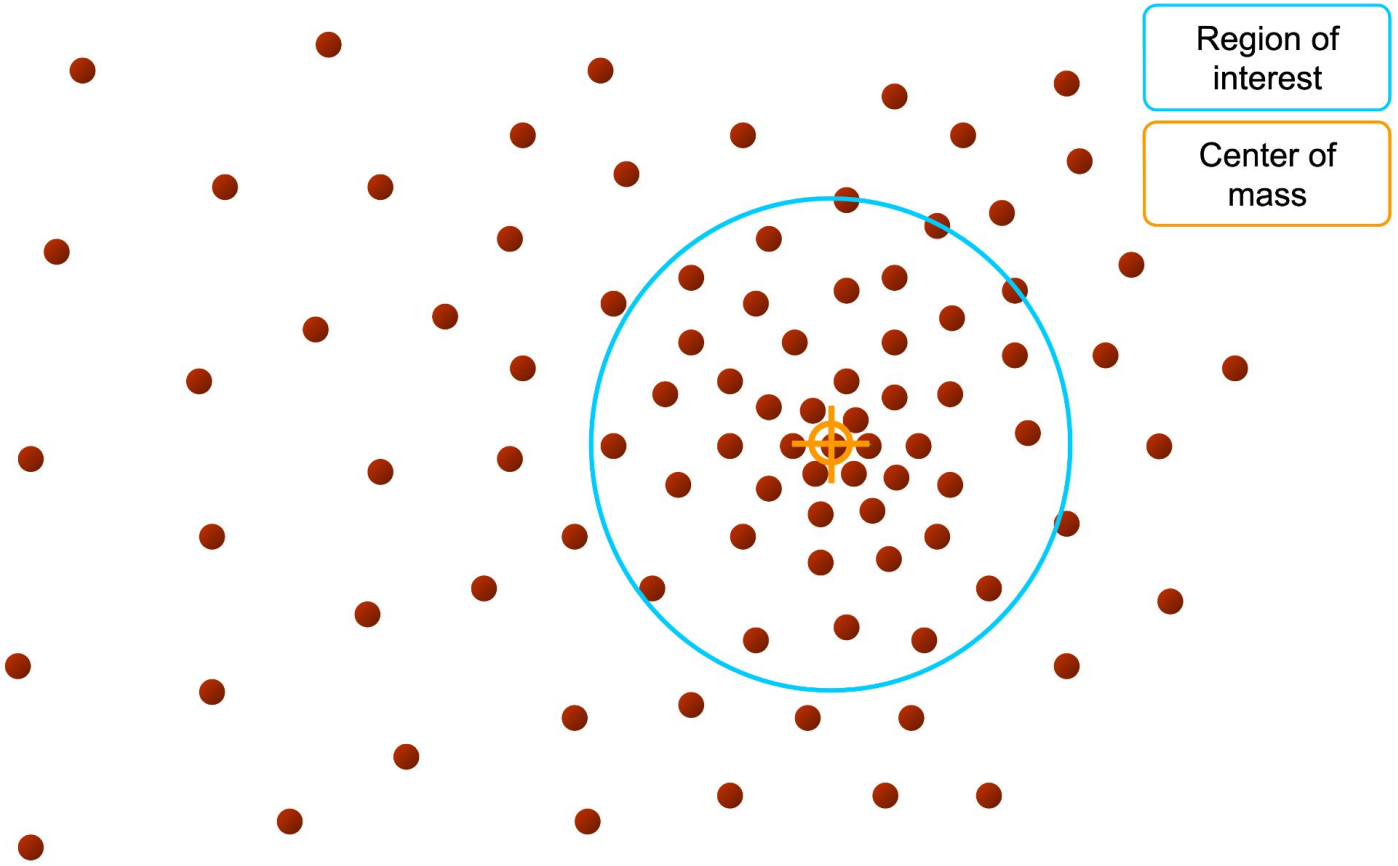
Mean-Shift

For each pixel, find the mode of the local density it belongs to



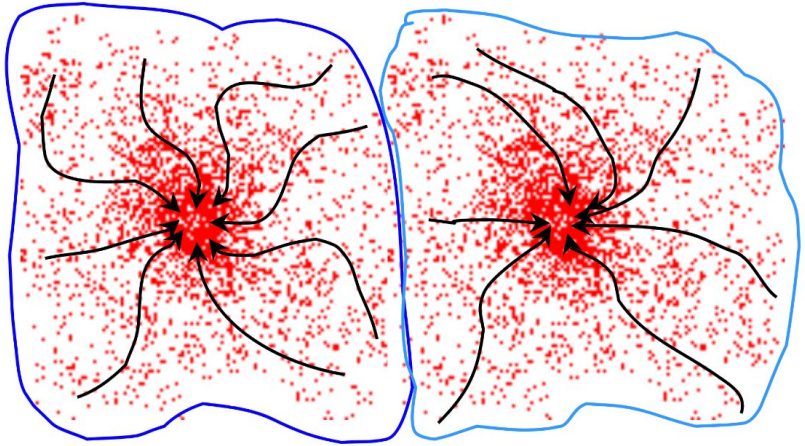
Mean-Shift

For each pixel, find the mode of the local density it belongs to



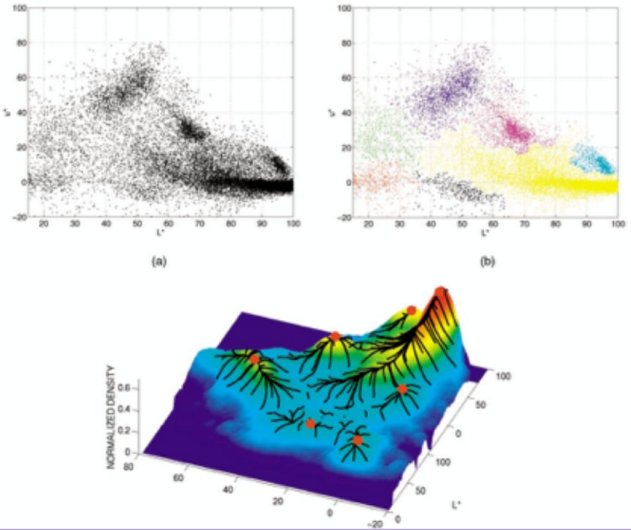
Clustering: Mean-shift

- Cluster: all data points in the attraction basin of a mode
- Attraction basin: the region for which all trajectories lead to the same mode



Mean-shift for segmentation

- Find features (color, gradients, texture, etc)
- Initialize windows at individual pixel locations
- Perform mean shift for each window until convergence
- Merge windows that end up near the same “peak” or mode

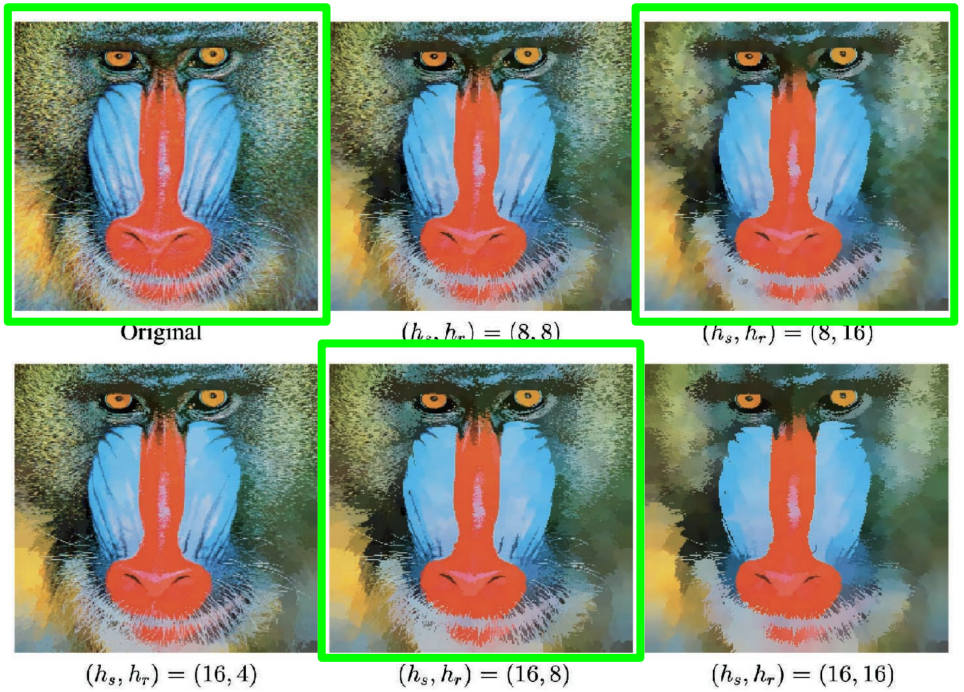


Mean-shift for segmentation

Mean shift kernel effects (in the kernel implementation)

When color (x_r) and locations (x_s) are at different scales

$$K(\boldsymbol{x}_j) = k\left(\frac{\|\boldsymbol{x}_r\|^2}{h_r^2}\right) k\left(\frac{\|\boldsymbol{x}_s\|^2}{h_s^2}\right)$$

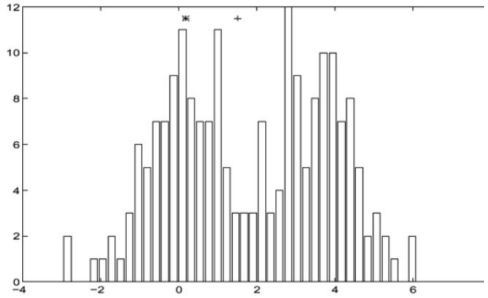


Comaniciu, 2002

Mean Shift

Pro:

- No number of clusters assumption
- Handle unusual distributions
- Simple



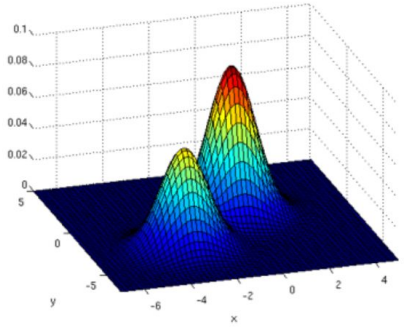
Con:

- Choice of window size
- Can be somewhat expensive

Clustering based segmentation in general

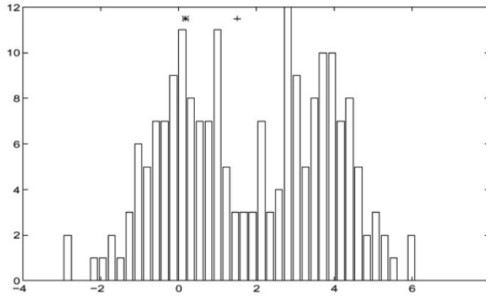
Pro:

- Generally simple
- Can handle most data distributions with sufficient effort.

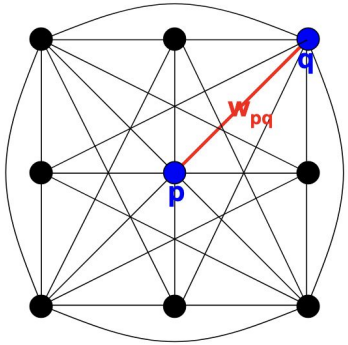


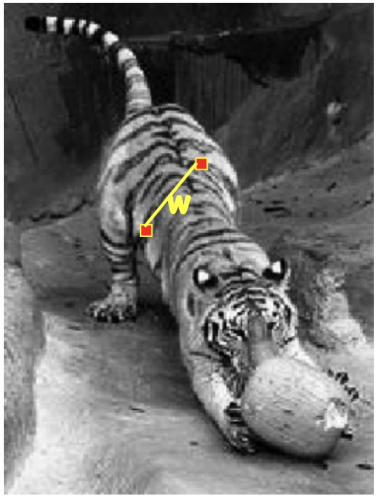
Con:

- Hard to capture global structure
- Performance is limited by simplicity



Idea: represent all pixels as nodes in a graph and then find subgraphs





- Node (vertex) for every pixel
- Edge between pairs of pixels, (p,q)
- Affinity weight w_{pq} for each edge
  - w_{pq} measures similarity
  - Similarity is inversely proportional to difference (in color and position…)

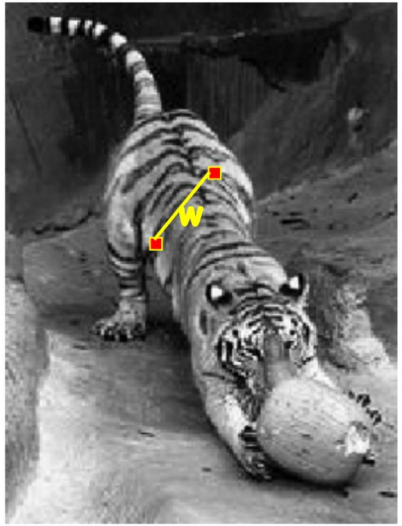
How to construct a pixel graph?

Which edges to include?

Fully connected:
- Captures all pairwise similarities
- Infeasible for most images

Neighboring pixels:
- Very fast to compute
- Only captures very local interactions

Local neighborhood:
- Reasonably fast, graph still very sparse
- Good tradeoff



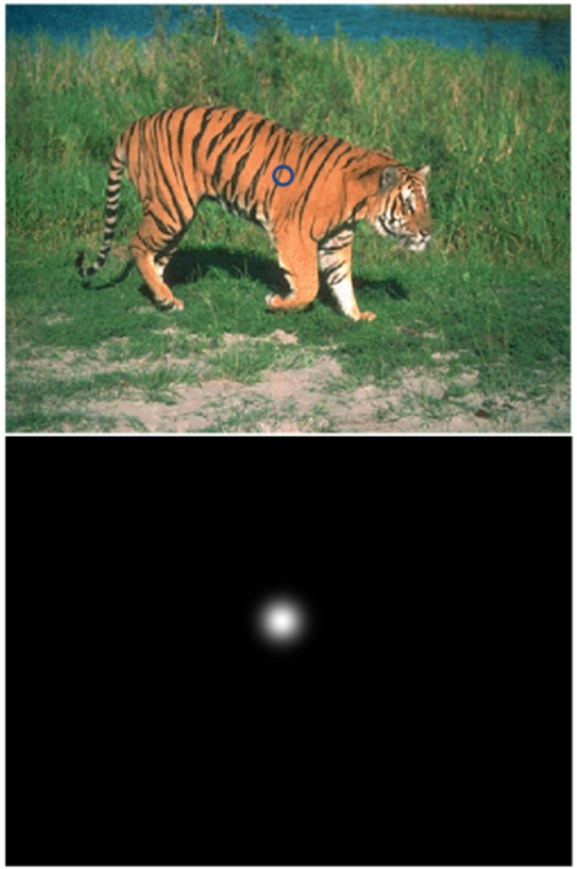
How to compute affinities?

- In general: $aff(x, y) = \exp\left(-\frac{1}{2\sigma_d^2}\|f(x) - f(y)\|^2\right)$
- Examples:
  - Distance: $f(x) = location(x)$
  - Intensity: $f(x) = intensity(x)$
  - Color: $f(x) = color(x)$
  - Texture: $f(x) = filterbank(x)$
- Note: Can also modify distance metric

Measuring Affinity

Distance:

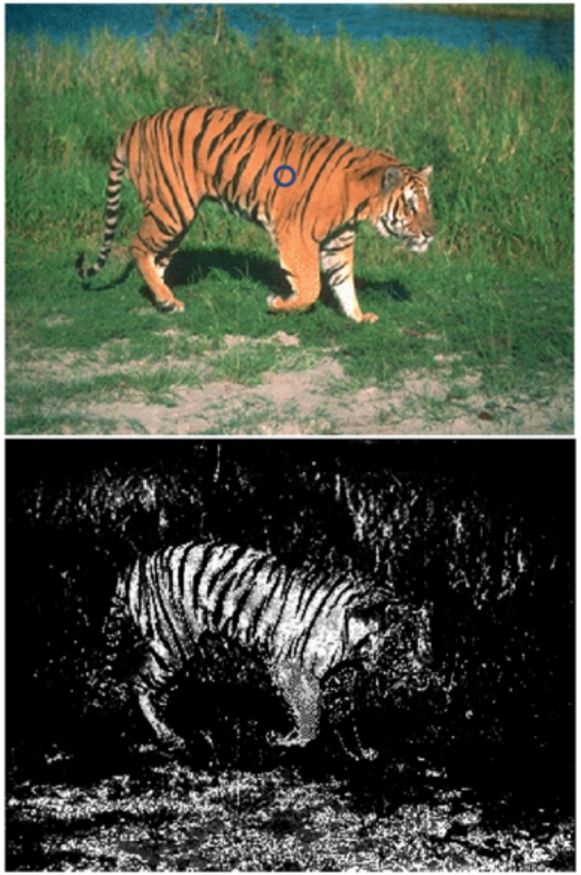
$$f(x) = location(x)$$



Measuring Affinity

Intensity:

$$f(x) = intensity(x)$$



Measuring Affinity

Color:

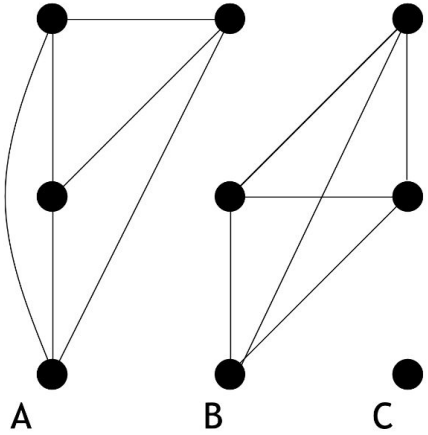
$$f(x) = color(x)$$

Measuring Affinity

Texture:

$$f(x) = filterbank(x)$$

Segmentation as Graph Cuts



- Break Graph into Segments
  - Delete links that cross between segments
  - Easiest to break links that have low similarity (low weight)
    - Similar pixels should be in the same segments
    - Dissimilar pixels should be in different segments

Graph Cut with Eigenvalues

- Given: Affinity matrix *W*
- Goal: Extract a single good cluster *v*
  - *v(i)*: score for point *i* for cluster *v*

$$\max_{v} \; v^T W v$$
$$\text{s.t.} \; v^T v = 1$$

W: Pixels x pixels matrix
v: pixels x 1 vector

Optimizing

$$\max_{v} \; v^T W v \quad \text{s.t.} \; v^T v = 1 \qquad \longleftrightarrow \qquad \min_{v} \; -\tfrac{1}{2} v^T W v \quad \text{s.t} \; v^T v = 1$$

Lagrangian: $$-\frac{1}{2} v^T W v + \lambda(v^T v - 1)$$

$$-Wv + \lambda v = 0$$

$$Wv = \lambda v$$

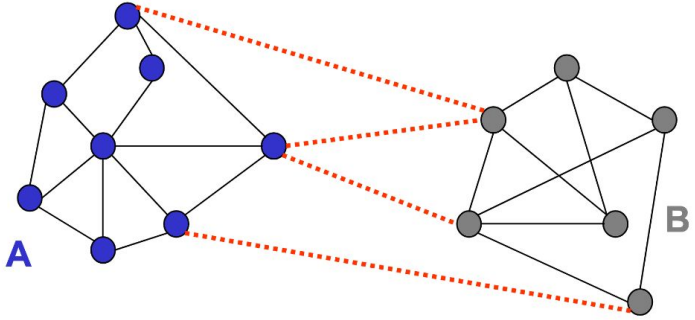
v is an eigenvector of *W*

Clustering via Eigenvalues

1. Construct affinity matrix *W*
2. Compute eigenvalues and vectors of *W*
3. Until done
    1. Take eigenvector of largest unprocessed eigenvalue
    2. Zero all components of elements that have already been clustered
    3. Threshold remaining components to determine cluster membership

Note: This is an example of a *spectral clustering* algorithm

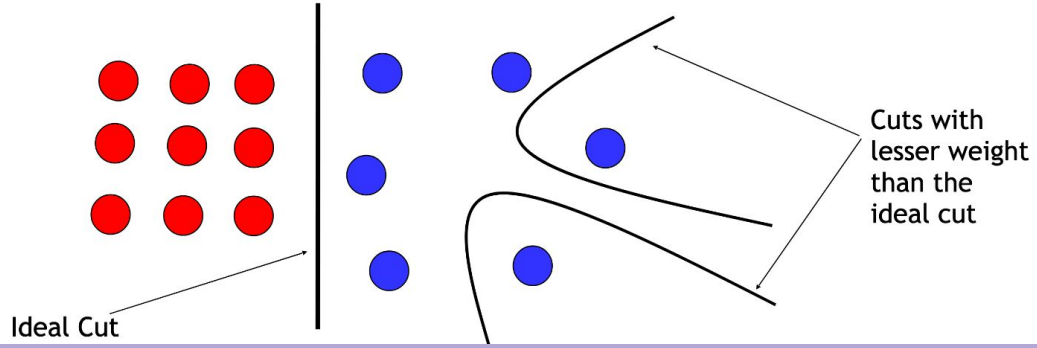
Graph Cuts - Another Look



- Set of edges whose removal makes a graph disconnected
- Cost of a cut
  - Sum of weights of cut edges: $cut(A, B) = \sum_{p \in A, q \in B} w_{pq}$
- A graph cut gives us a segmentation
  - What is a “good” graph cut and how do we find one?

Formulation: Min Cut

- We can do segmentation by finding the *minimum cut*
  - either smallest number of elements (unweighted) or smallest sum of weights (weighted)
  - efficient algorithms exist
- Drawback
  - Weight of cut proportional to number of edges
  - Biased towards cutting small, isolated components



Formulation: Normalized Cuts

- Key idea: normalize segment size
  - Fixes min cut's bias
- Formulation:

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

$$= cut(A,B)\left[\frac{1}{\sum_{p\in A} w_{p,q}} + \frac{1}{\sum_{q\in B} w_{p,q}}\right]$$

$assoc(A,V)$ = sum of weights of edges in *V* that touch *A*

- NP-hard, but can approximate

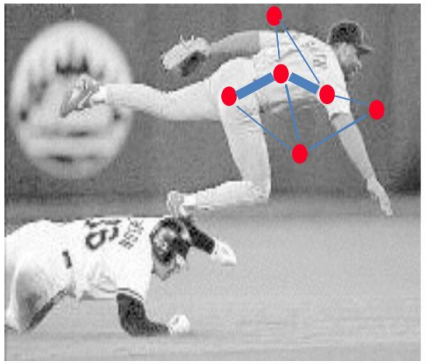
NCuts as Generalized Eigenvector Problem

Definitions:

W : **affinity** matrix

D : **diagonal** matrix $D(i,i) = \sum_j w_{i,j}$

z : **vector** in $\{-1, 1\}^N, z_i = 1 \Leftrightarrow i \in A$



In matrix form:

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

$$= \frac{(1+z)^T(D-W)(1+z)}{k1^TD1} + \frac{(1-z)(D-W)(1-z)}{(1-k)1^TD1}; \quad k = \frac{\sum_{z_i>0} D(i,i)}{\sum_i D(i,i)}$$

$$= \ldots$$

After a lot of math…

- After simplification, we get

$$NCut(A,B) = \frac{y^T(D-W)y}{y^TDy},\ y_i \in \{1,-b\},\ y^TD1 = 0$$

This is hard, y is discrete!

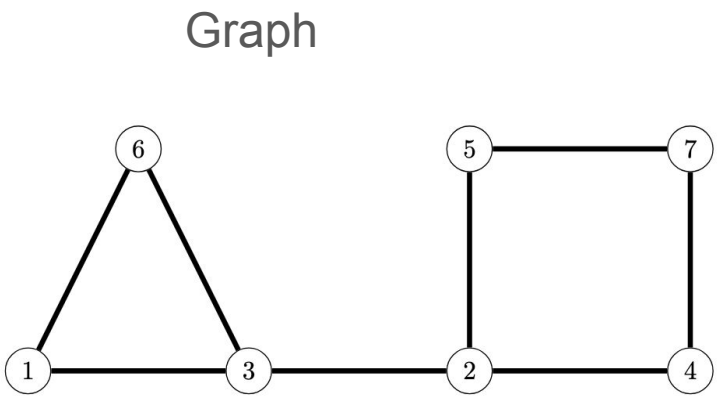
- This is a Rayleigh Quotient
  - Solution given by the "generalized" eigenvalue problem

$$(D-W)y = \lambda Dy$$

Relaxation: continuous y

- Subtleties
  - Optimal solution is second smallest eigenvector
  - Gives continuous result—must convert into discrete values of y

Brief detour in graph theory



$$D = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 \end{bmatrix}$$

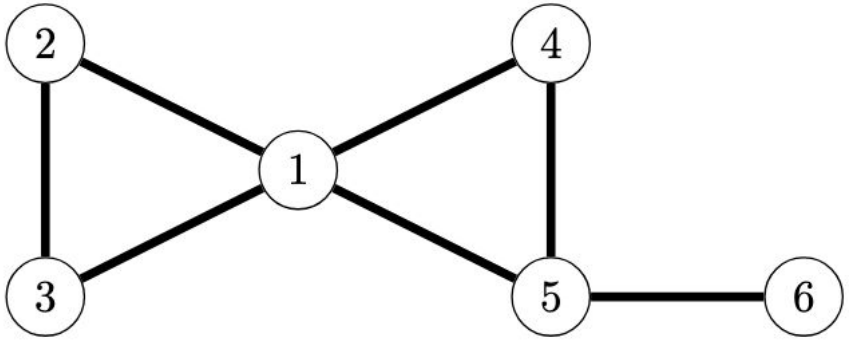
$$A = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}$$

$$L = D - A = \begin{bmatrix} 2 & 0 & -1 & 0 & 0 & -1 & 0 \\ 0 & 3 & -1 & -1 & -1 & 0 & 0 \\ -1 & -1 & 3 & 0 & 0 & -1 & 0 \\ 0 & -1 & 0 & 2 & 0 & 0 & -1 \\ 0 & -1 & 0 & 0 & 2 & 0 & -1 \\ -1 & 0 & -1 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & -1 & -1 & 0 & 2 \end{bmatrix}$$

Both the adjacency matrix and the Laplacian matrix contain all information about the graph and both can be used to analyze the graph.

Brief detour in graph theory

Example 12.3. Consider this example:



A minimum cut can be achieved by removing the $(5, 6)$ edge. However the result (the bow-tie on the left and the single vertex on the right) isn't very satisfactory

The usual solution to this is to minimize cut(P) with the added condition that we try to keep the number of vertices in each of the two remaining subgraphs as equal as possible.

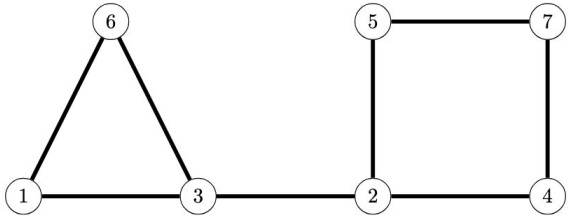
Fiedler value / method

If G is a simple connected graph with n vertices and if L is the Laplacian matrix for G then L has n real eigenvalues satisfying

$$0 = \lambda_1 < \lambda_2 \leq \lambda_3 \leq ... \leq \lambda_n$$

Definition 12.4.3.1. The *Fiedler Value* or the *algebraic connectivity* of a graph is the second smallest eigenvalue of its Laplacian matrix L.

Example 12.1 Revisited. In our example:



we saw that:

$$L = \begin{bmatrix} 2 & 0 & -1 & 0 & 0 & -1 & 0 \\ 0 & 3 & -1 & -1 & -1 & 0 & 0 \\ -1 & -1 & 3 & 0 & 0 & -1 & 0 \\ 0 & -1 & 0 & 2 & 0 & 0 & -1 \\ 0 & -1 & 0 & 0 & 2 & 0 & -1 \\ -1 & 0 & -1 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & -1 & -1 & 0 & 2 \end{bmatrix}$$

the eigenvalues in order are:

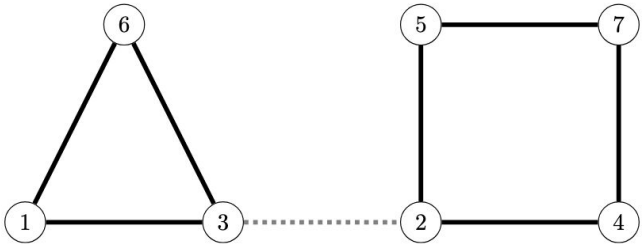
$$0,\ 0.3588,\ 2.0000,\ 2.2763,\ 3.0000,\ 3.5892,\ 4.7757$$

Note that the Fiedler Value is 0.3588. A Fiedler Vector is an eigenvector corresponding to this. Any nonzero multiple of the following unit vector will suffice:

$$\bar{v} = \begin{bmatrix} 0.48 \\ -0.15 \\ 0.31 \\ -0.35 \\ -0.35 \\ 0.48 \\ -0.42 \end{bmatrix}$$

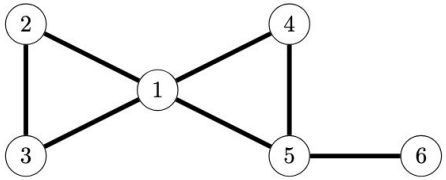
At its most basic, the Fiedler Method basically states that we can achieve a "reasonable" partition into two subgraphs by separating the vertices according to the sign of the values in a Fiedler Vector $\bar{v}$ where each entry corresponds to a vertex. This means we group together the vertices i with $v_i = +$ and we group

Example 12.1 Revisited. In our example above $v_i = +$ for $i = 1, 3, 6$ and $v_i = -$ for $i = 2, 4, 5, 7$, so that $P = (\{1, 3, 6\}, \{2, 4, 5, 7\})$. This means we separate the vertices accordingly:



This is just what we predicted!

Example 12.4. Let's go back to the bow-tie:



Here we have Laplacian Matrix:

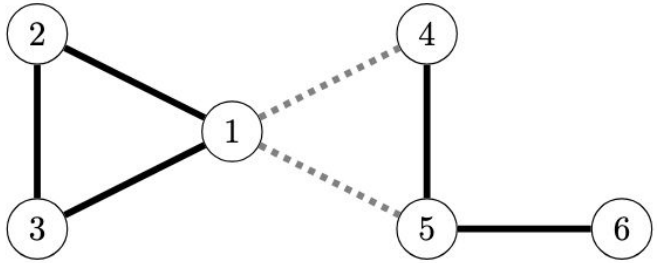
$$L = \begin{bmatrix} 4 & -1 & -1 & -1 & -1 & 0 \\ -1 & 2 & -1 & 0 & 0 & 0 \\ -1 & -1 & 2 & 0 & 0 & 0 \\ -1 & 0 & 0 & 2 & -1 & 0 \\ -1 & 0 & -1 & 3 & -1 & \\ 0 & 0 & 0 & 0 & -1 & -1 \end{bmatrix}$$

The eigenvalues in order are:

$$0, 0.6314, 1.4738, 3, 3.7877, 5.1071$$

$$\bar{v} = \begin{bmatrix} -0.16 \\ -0.44 \\ -0.44 \\ 0.07 \\ 0.26 \\ 0.71 \end{bmatrix}$$

e separate the vertices accordingly:



This is a more-balanced partition than the minimum cut.

https://www.math.umd.edu/~immortal/MATH401/book/ch_graph_theory.pdf

NCuts example

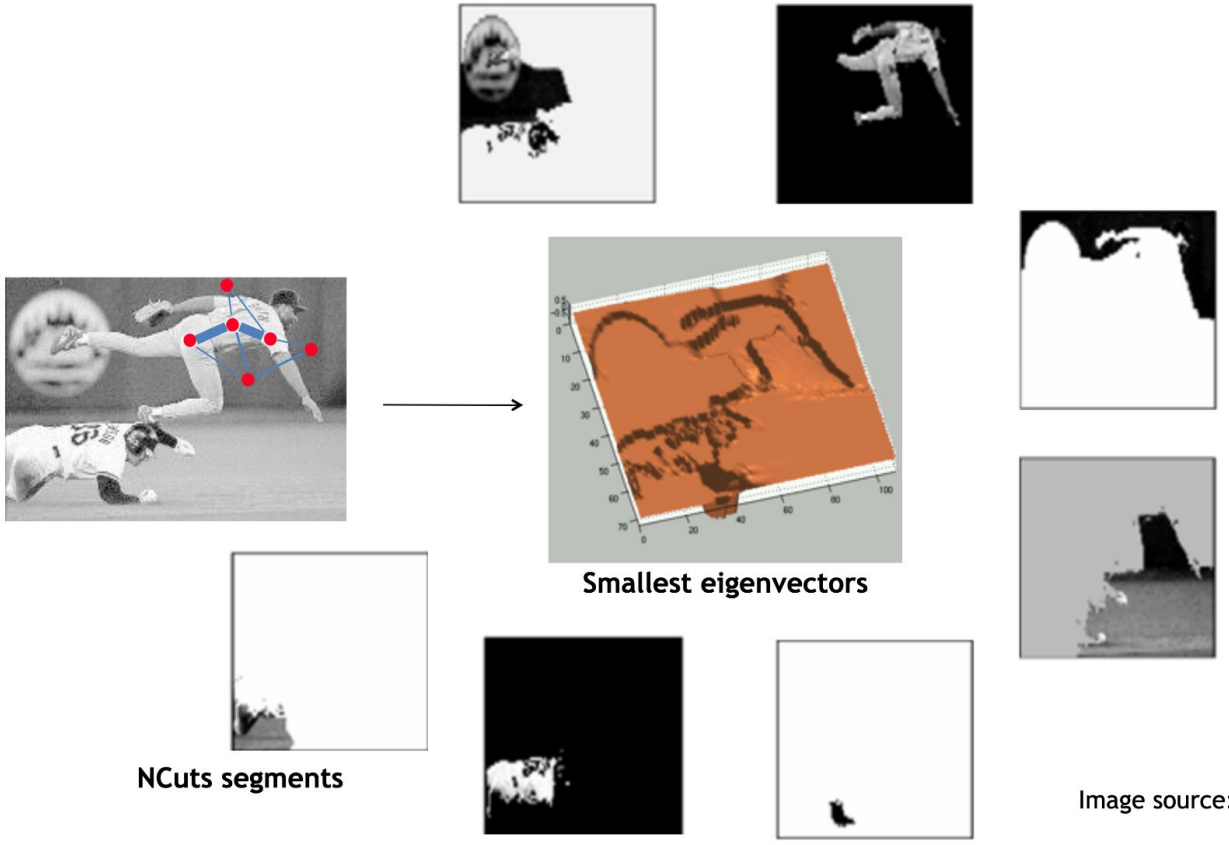


Image source: Shi & Malik

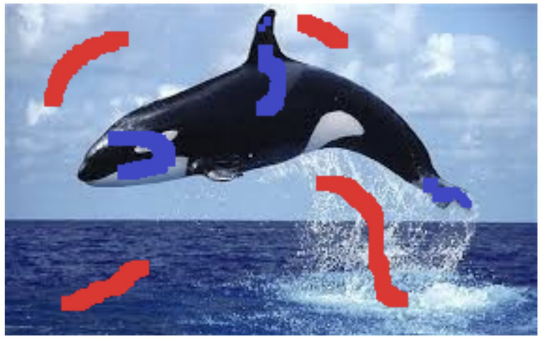
NCuts: Algorithm Summary

1. Construct weighted graph $G = (V, E)$
2. Construct affinity matrix W
3. Solve $(D - W)y = \lambda Dy$ for smallest few eigenvectors.
   - This is a continuous solution
4. Threshold eigenvectors to get a discrete cut
   - This is the approximation
   - As before, several heuristics for doing this
5. Recursively subdivide as desired.

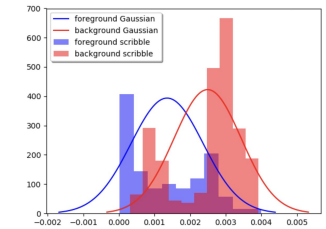
Graph cut in practice

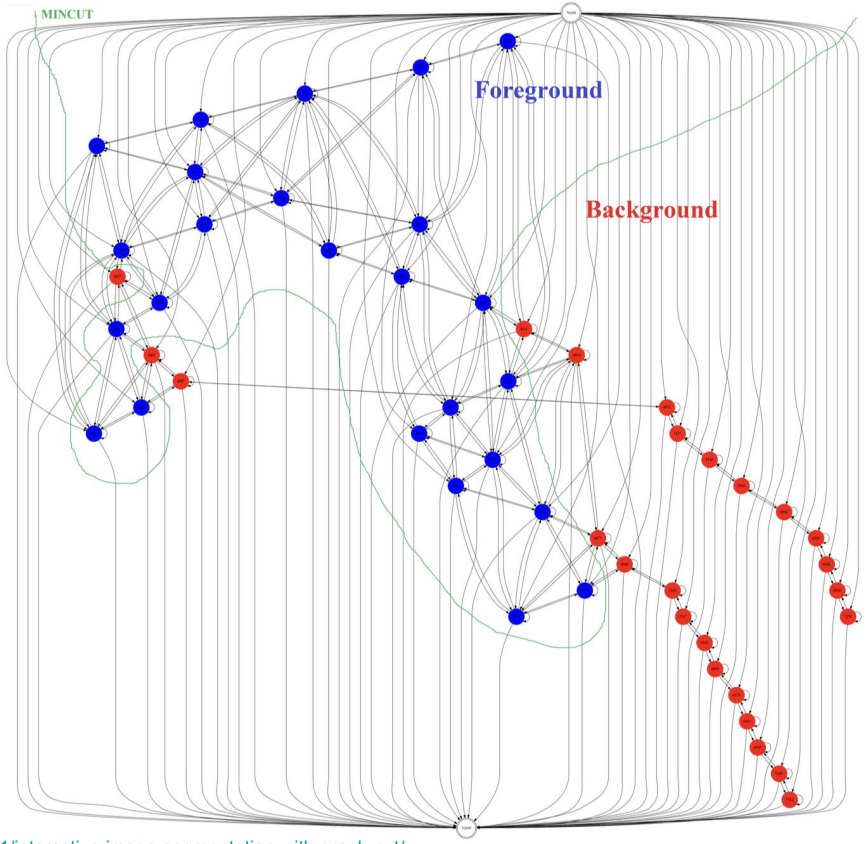
Input Image

Input Image with Scribbles



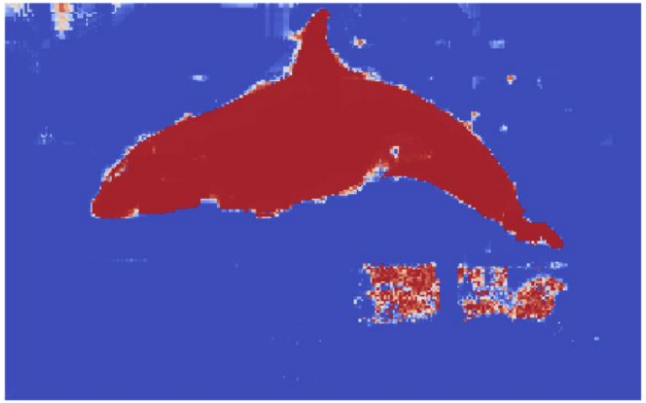
Fitted Densities from Color Scribbles



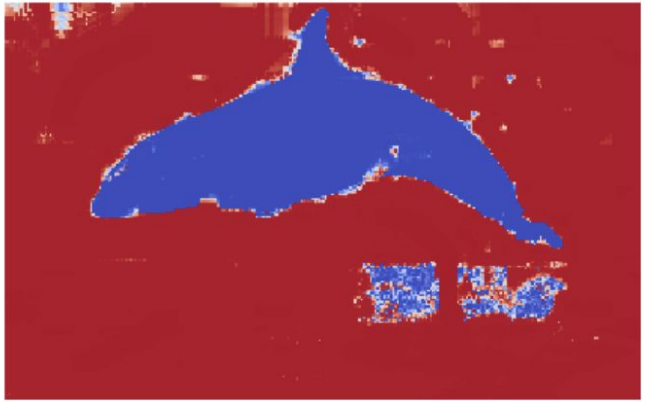


https://sandipanweb.wordpress.com/2018/02/11/interactive-image-segmentation-with-graph-cut/
https://sandipanweb.wordpress.com/2018/02/25/graph-based-image-segmentation-in-python/

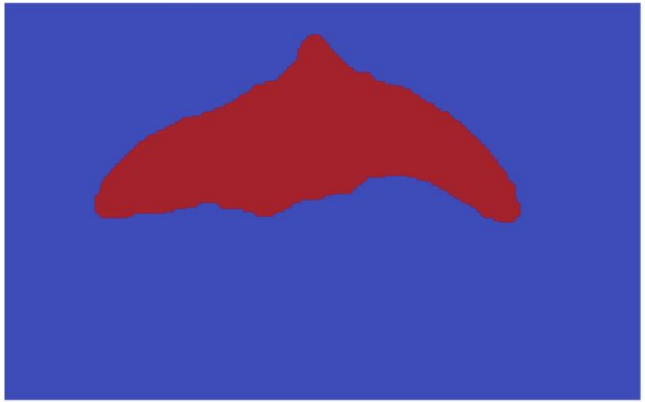
Foreground Probabilities



Background Probabilities



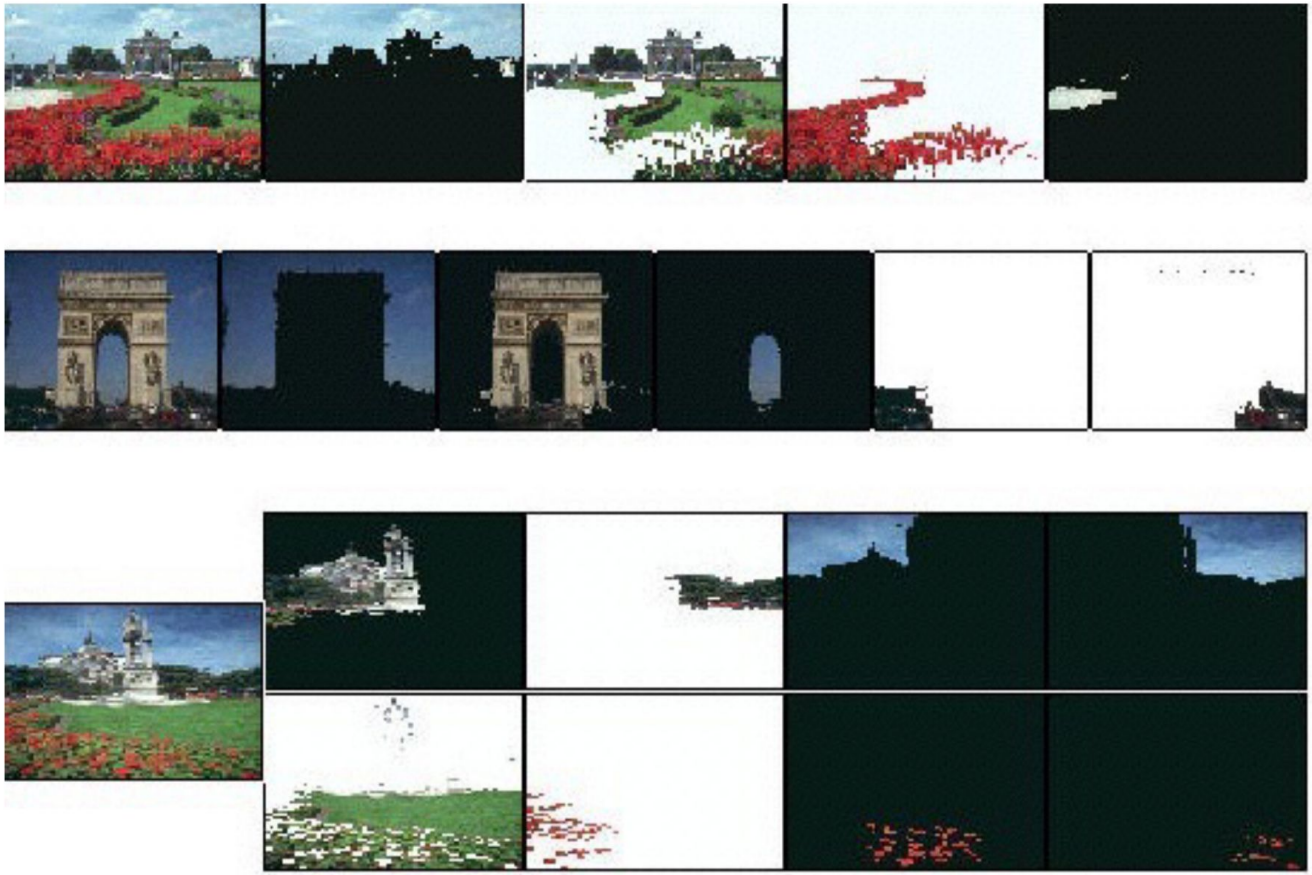
Foreground with Maxflow/Mincut



Segmented Foreground with GraphCut

https://sandipanweb.wordpress.com/2018/02/11/interactive-image-segmentation-with-graph-cut/
https://sandipanweb.wordpress.com/2018/02/25/graph-based-image-segmentation-in-python/

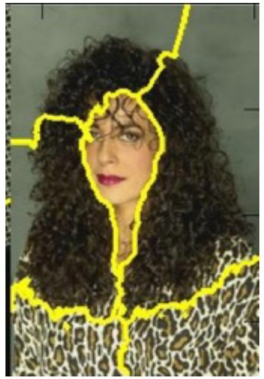
NCuts examples



NCuts examples

NCuts Pro and Con

- Pro
  - Flexible to choice of affinity matrix
  - Generally works better than other methods we've seen so far

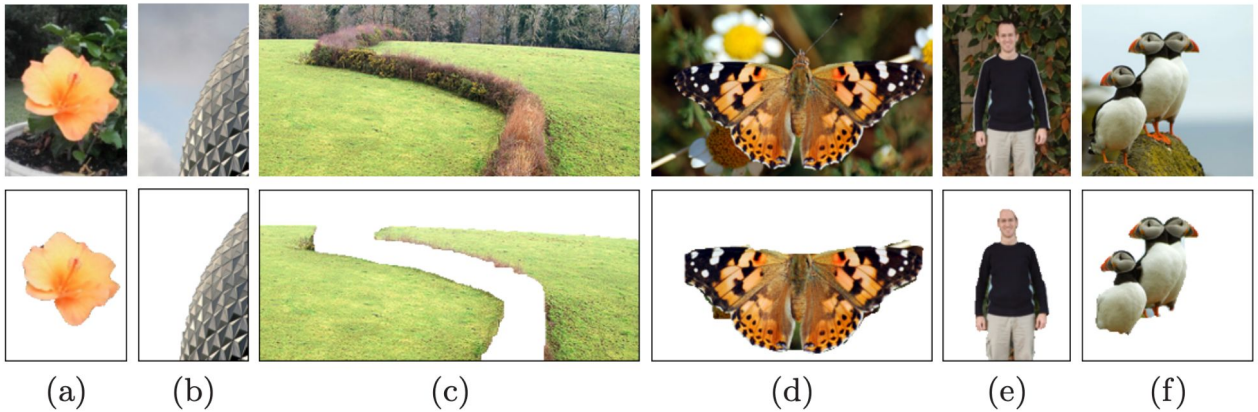


- Con
  - Can be expensive, especially with many cuts.
  - Bias toward balanced partitions
  - Constrained by affinity matrix model

How can we measure the segmentation model worked? Evaluation metrics!

Evaluation metrics for segmentations

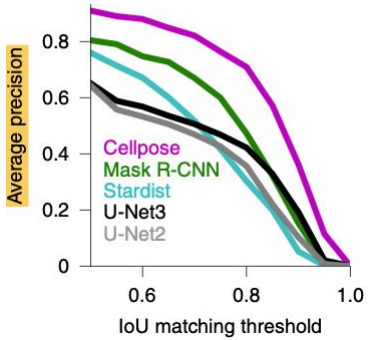
- There's no theory on what should be clustered together..
- We should form the cluster that is helpful for the particular applications
- '*There is a huge diversity of definitions of **"visually meaningful" image segments**, ranging from simple uniformly colored segments, textured segments, through symmetric patterns, and up to complex semantically meaningful objects.' – What Is a Good Image Segment? A Unified Approach to Segment Extraction*, Bagon et al., 2008



Evaluation metrics for segmentations

Some methods:

- Count-based - just count the pixel agreement/disagreement
- Set matching methods - find the matching clusters first!



Evaluations 1: count-based

The pairs of pixels/points from the image/data → if the pixel is in the same category by clustering result C and C':

- N_{00}: # of pixels in the different cluster in C and C'
- N_{01}: # of pixels in the same cluster in C' but NOT in C
- N_{10}: # of pixels in the same cluster in C but NOT in C'
- N_{11}: # of pixels in the same cluster in both C and C'

So,

- $N_{00}+N_{01}+N_{10}+N_{11}$ = n(n-1)/2 (total number of pairs)
- $N_{00}+N_{11}$: # of agreements between C and C'
- $N_{01}+N_{10}$: # of disagreements between C and C'

Rand index → Accuracy

$R(C,C') = [N_{00}+N_{11}] / n(n-1)/2$

$R(C,C') = [TP+TN] / [TP+FP+FN+TN]$

Jaccard index → Intersection-over-Union (IoU)

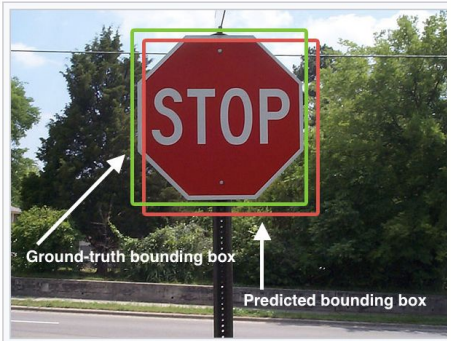
$J(C,C') = N_{11} / (N_{01}+N_{10}+N_{11})$

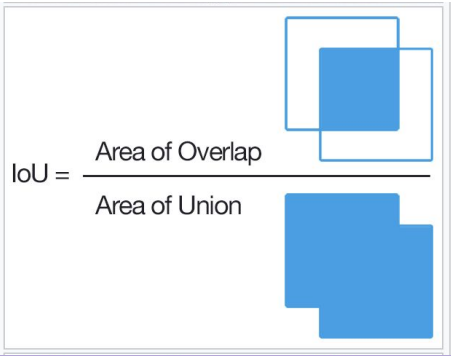
Dice coefficient → F1 score

$DC(C,C') = 2*TP / (2*TP + FP + FN)$

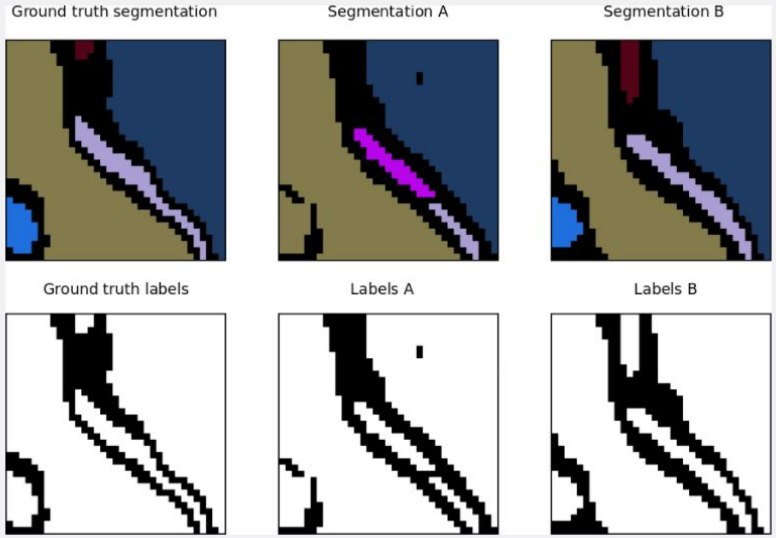
Pixel error:

Hamming distance in binary case



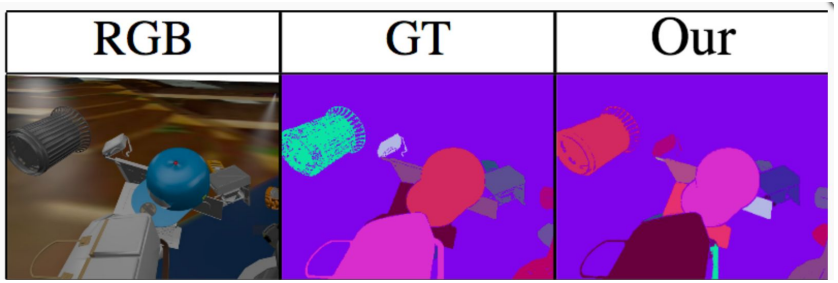


Evaluations 2: warping error, Jain et al., 2010



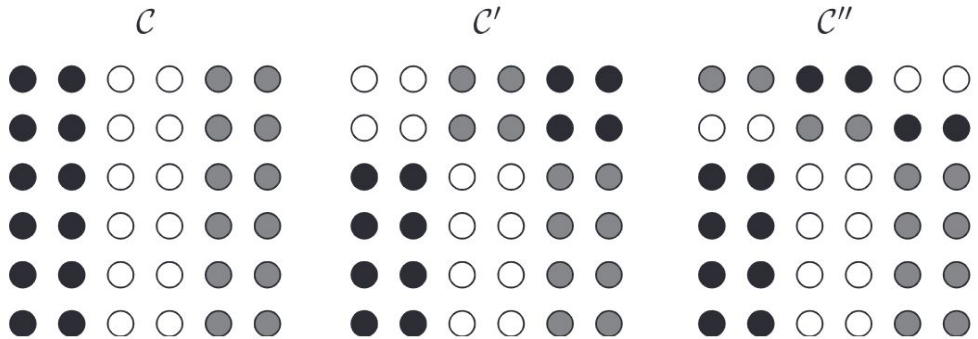
Application of the topology-preserving warping error. Example A and B have almost the same amount of pixel error with respect to the ground truth, however, example B has no topological error.

Evaluations 3:
Set matching methods



$$\mathcal{H}(\mathcal{C}, \mathcal{C}') = 1 - \frac{1}{n} \max_{\pi} \sum_{k=1}^{K} n_{k,\pi(k)}.$$

- First, each cluster in C is given a ‘best matching cluster’ in C’
- H is the total ‘unmatched’ probability mass in the confusion matrix
- π is injection matrix of {1,..., K} to {1,..., K’} – correspondence bw labels in C and C’



Common threads of classical segmentation

- Define a feature representation of each pixel (DATA)
  - It can be just its intensity (like otsu) or intensity + position (like in mean-shift or graph cuts) or some higher order descriptor.
- Define a objective function to allocate pixels into different groups (MODEL)
  - Agglomerative (like watershed or superpixels) or divisive (like graph cuts)
- Define an inference routine (INFERENCE)
  - Greedy (like watershed)
  - Optimal (like graph cut)
- Measure error at the end(LOSS)

1. Define Model **F**

Input image

F

1. Define Model **F**

2. Infer an output that satisfies model

Input image

F

1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	3	3	1	1	1	1	4
1	3	3	3	3	3	4	4
1	3	3	3	3	3	3	4

Segmentation map

1. Define Model **F**

2. Infer an output that satisfies model

Input image

F

1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	3	3	1	1	1	1	4
1	3	3	3	3	3	4	4
1	3	3	3	3	3	3	4

Segmentation map

2	2	2	2	2	1	1	1
2	2	2	2	2	1	1	1
2	1	1	1	1	1	1	1
2	1	1	1	1	1	1	4
2	1	1	1	1	1	4	4
3	3	3	1	4	4	4	4
3	3	3	3	3	4	4	4
3	3	3	3	3	4	4	4

Ground truth

3. Measure error

Comparative analysis

- All model based segmentation methods have implicit assumptions / biases e.g.
  - Threshold approaches = Pixels above or below threshold come from the same object
  - Watershed = Pixels with smoothly changing intensities and positions come from the same object
  - Mean-shift = the pixels around each mode in a feature space represent the same object
  - Graph cut = different objects have have cohesive representations that are different

All models are wrong, but some are useful.

— George E.P. Box —

Model based segmentation: common assumption

Bottom-up modeling:

Pixels belong together because they look similar to each other (and/or dissimilar to other pixels).

Pros:

- Easy to generate models.
- Interpretable assumptions.
- No data needed to train!

Cons:

- Strong inductive biases.
- Evaluation is not part of the model.

What if we thought of the problem from top-down?

Pixels belong together because they come from the same object.

Learning based segmentation: common assumption

Top-down modeling:

We don’t know ahead of time the representation under which pixels should be similar, but we know which pixels should belong together since they belong to the same object. Learn a representation that satisfies this.

Pros:

- Less inductive biases
- We can learn anything based on training data since the loss function is now part of the model.

Cons:

- Need lots of training data
- Model is often interpretable.

1. Define a flexible model **F**

Input image

F

1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	3	3	1	1	1	1	4
1	3	3	3	3	3	4	4
1	3	3	3	3	3	3	4

Segmentation map

2	2	2	2	2	1	1	1
2	2	2	2	2	1	1	1
2	1	1	1	1	1	1	1
2	1	1	1	1	1	1	4
2	1	1	1	1	1	4	4
3	3	3	1	4	4	4	4
3	3	3	3	3	4	4	4
3	3	3	3	3	4	4	4

Ground truth

1. Define error

1. Define a flexible model **F**

Input image

F

1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	3	3	1	1	1	1	4
1	3	3	3	3	3	4	4
1	3	3	3	3	3	3	4

Segmentation map

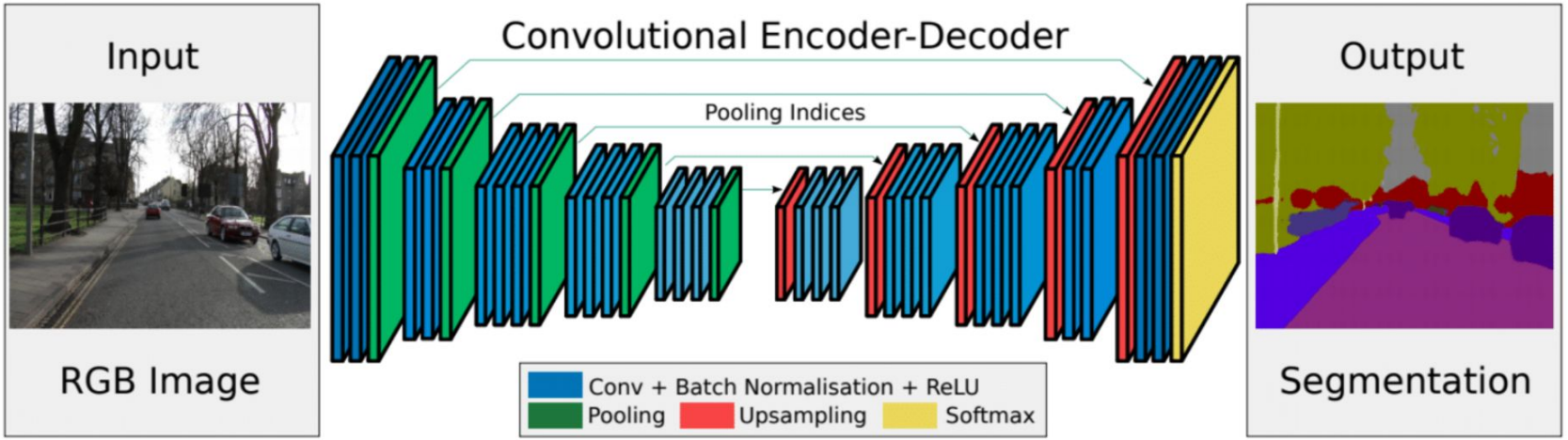
2	2	2	2	2	1	1	1
2	2	2	2	2	1	1	1
2	1	1	1	1	1	1	1
2	1	1	1	1	1	1	4
2	1	1	1	1	1	4	4
3	3	3	1	4	4	4	4
3	3	3	3	3	4	4	4
3	3	3	3	3	4	4	4

Ground truth

1. Define error

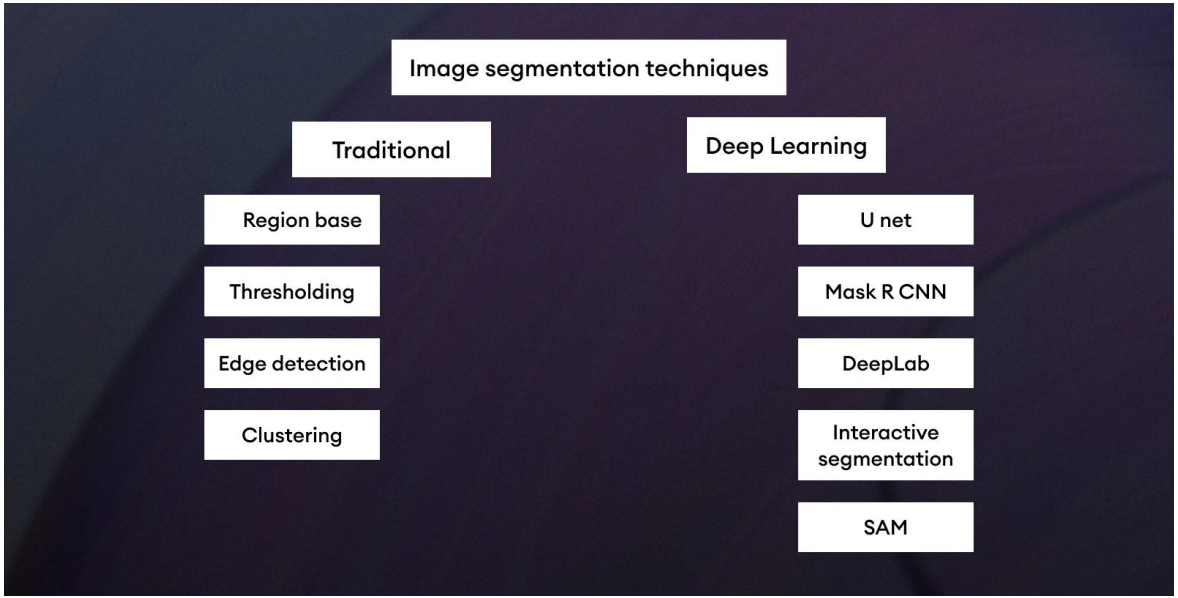
2. Learn model parameters that output a segmentation map in agreement with GT

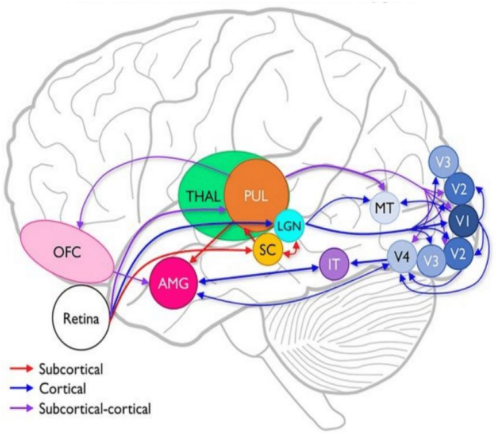
An example of a learning based segmentation algo

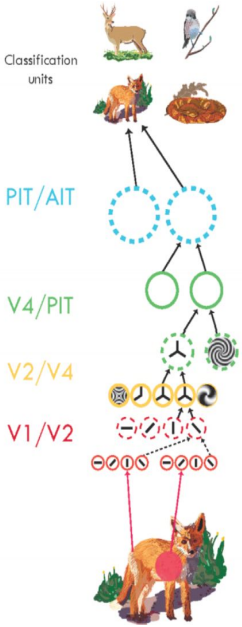


Second half of the semester - Learning based methods!

Paradigm shift in approach







High-level vision (object, scene)

After midterm - Deeper computer vision

Mid-level vision (segmentation)

Up to midterm - shallow computer vision

Low-level vision (Edge, corner)

features

243	239	240
242	239	218
243	242	123

patches

243

pixels

start

Coding lab

Next up: Midterm exam in class 10/24 11am!