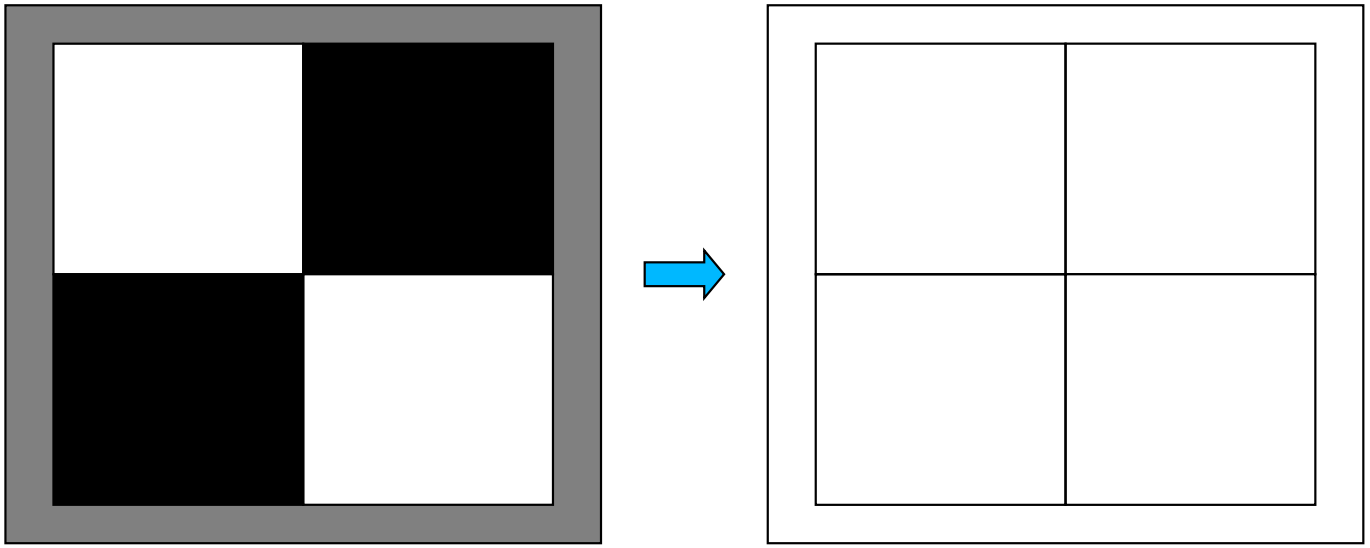
Regions

Pascal Fua
IC-CVLab

Reminder: Edges and Regions



Edges:

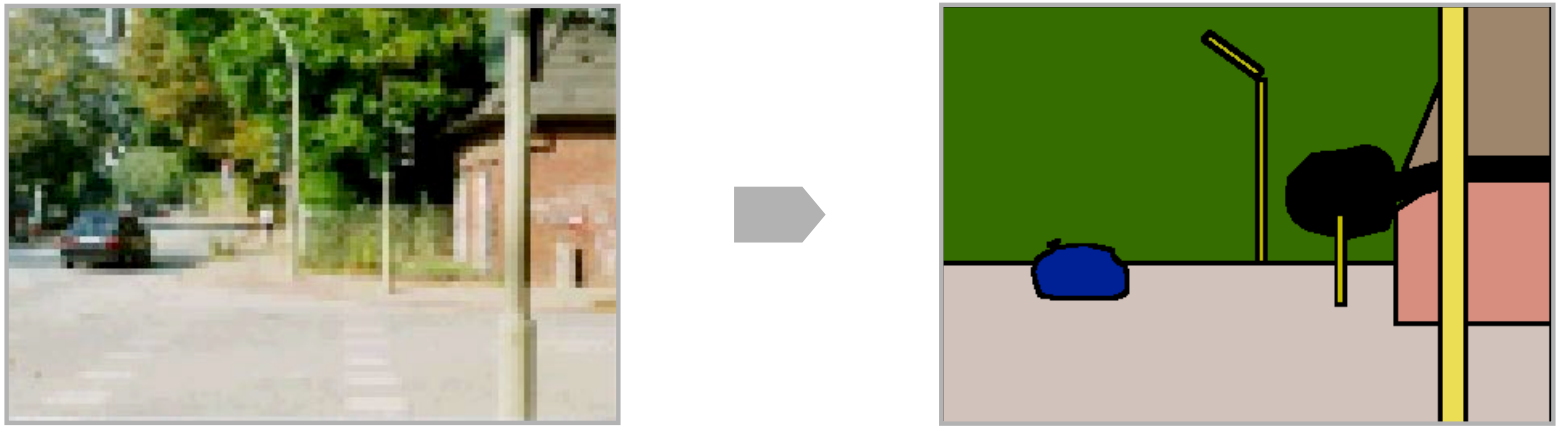
- Boundary between bland image regions.

Regions:

- Homogenous areas between edges.

→ Edge/Region Duality.

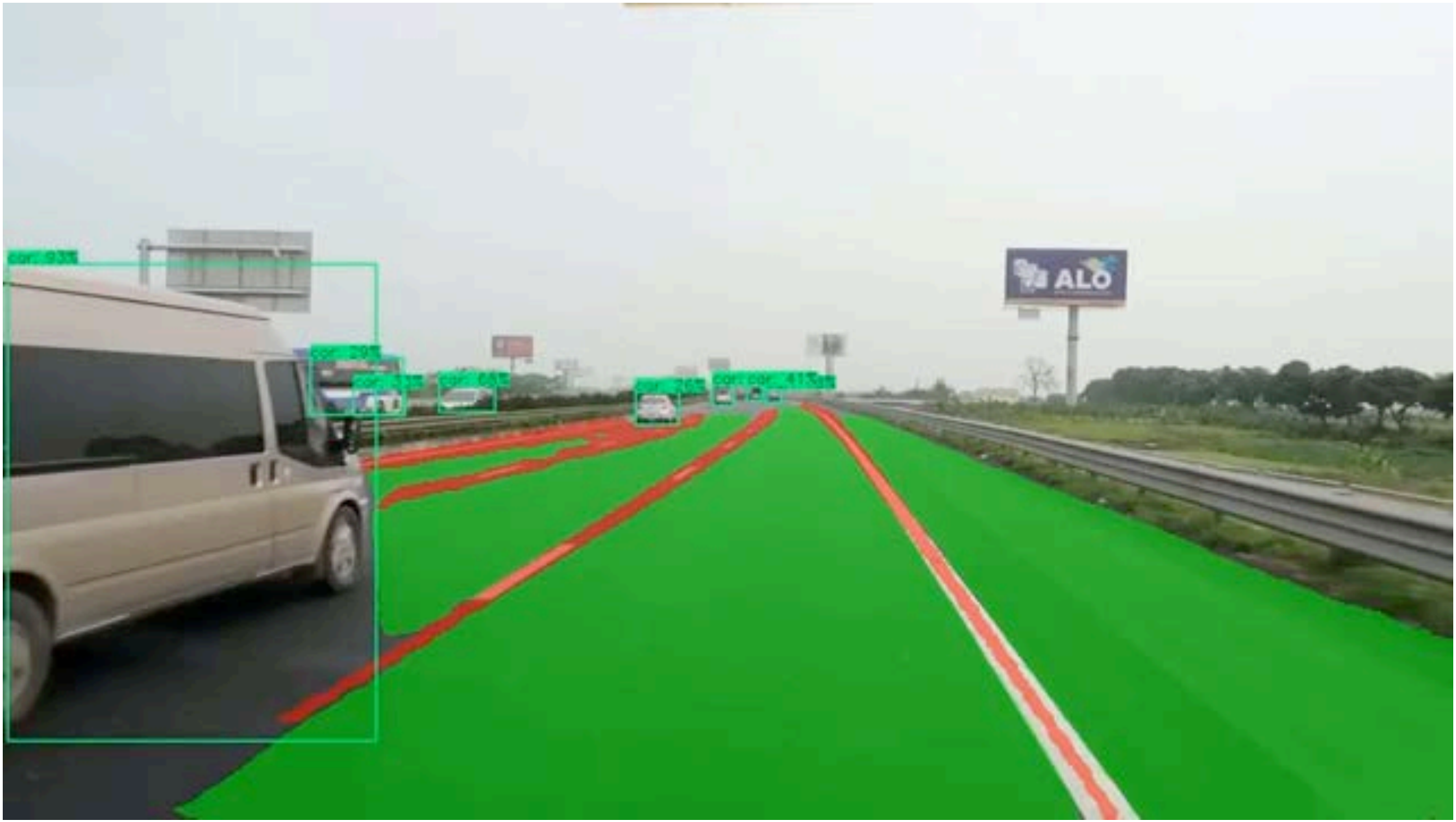
Region Segmentation



Ideal region: Set of pixels with the same statistical properties and corresponding to the same object.

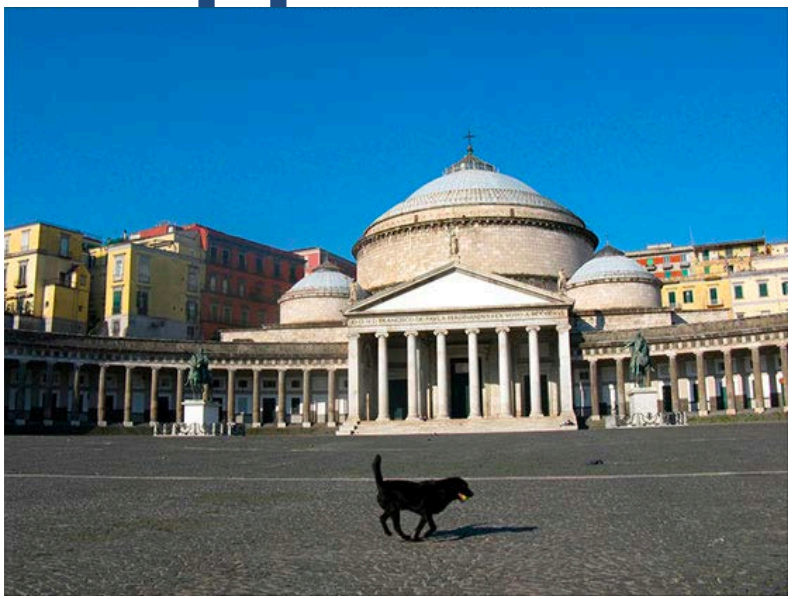
Purpose: Should help with recognition, tracking, image database retrieval, and image compression among other high-level vision tasks.

Application: Automated Driving

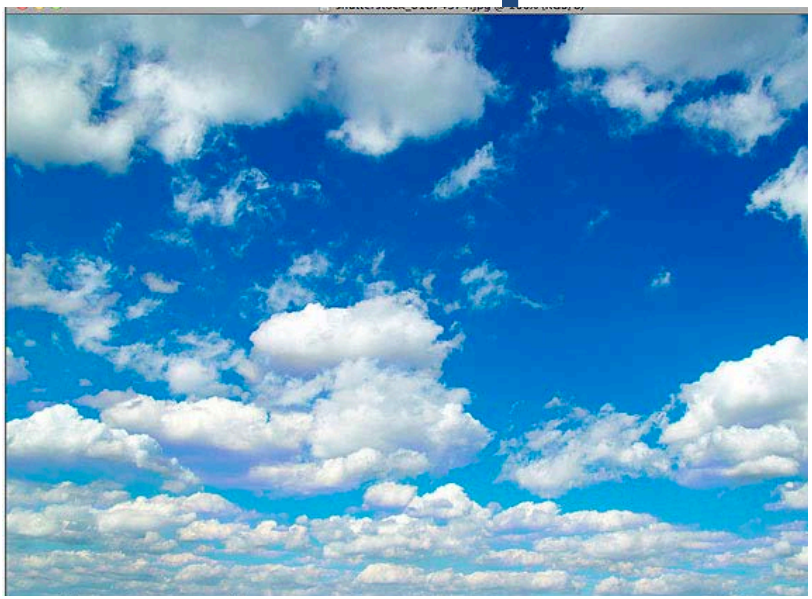


Vu et al. , ArXiv'22

Applications: Photoshop



I find this blue sky too bland!



I prefer this one.

Find the sky region.

Replace it.

https://www.photoshopessentials.com/basics/selections/magic-wand-tool/

In Theory

Look for an image partition such that:

$$I = \bigcup_{i=1}^{m} S_i$$

$$S_i \cap S_j = 0, \forall i \neq j$$

$$H(S_i) = True, \forall i$$

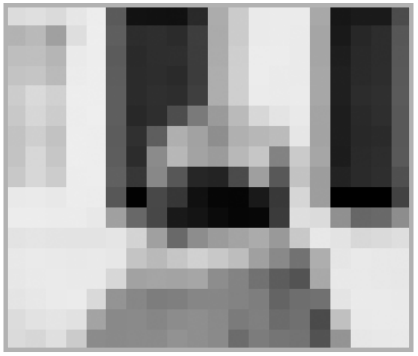
$$H(S_i \cup S_j) = False, \text{ if } S_i \text{ and } S_j \text{ are adjacent.}$$

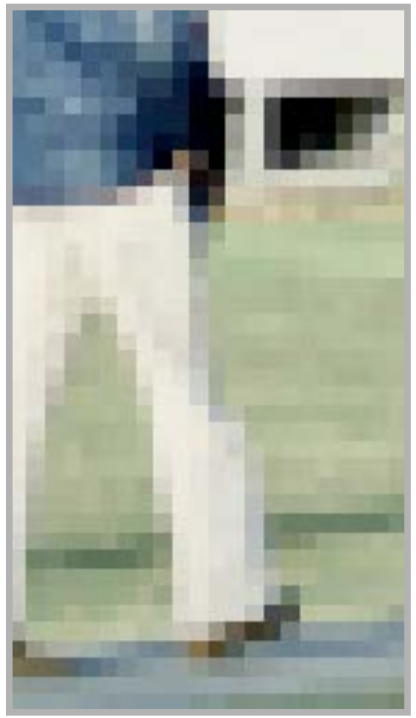
where H measures homogeneity.

Complex Gray Level Variations

- Simple thresholding and other basic image operations do not suffice.
- The H predicate is difficult to define.

Context is Essential





Without the whole image it is hard to make sense of small image windows.

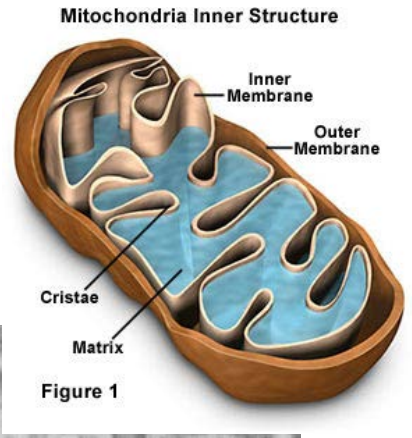
There is not always a Single Answer

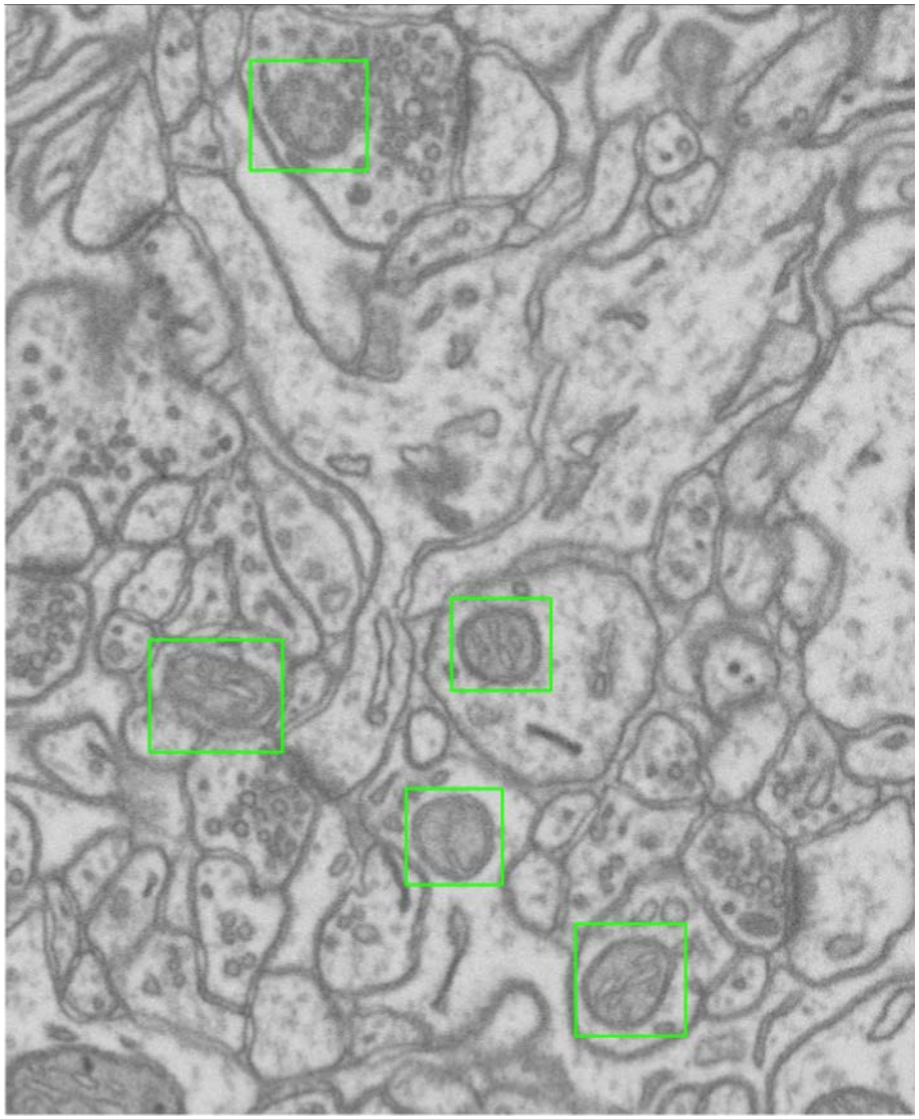
- Segmentations hand-drawn by 5 different people.
- We cannot say that one is right and the others wrong.

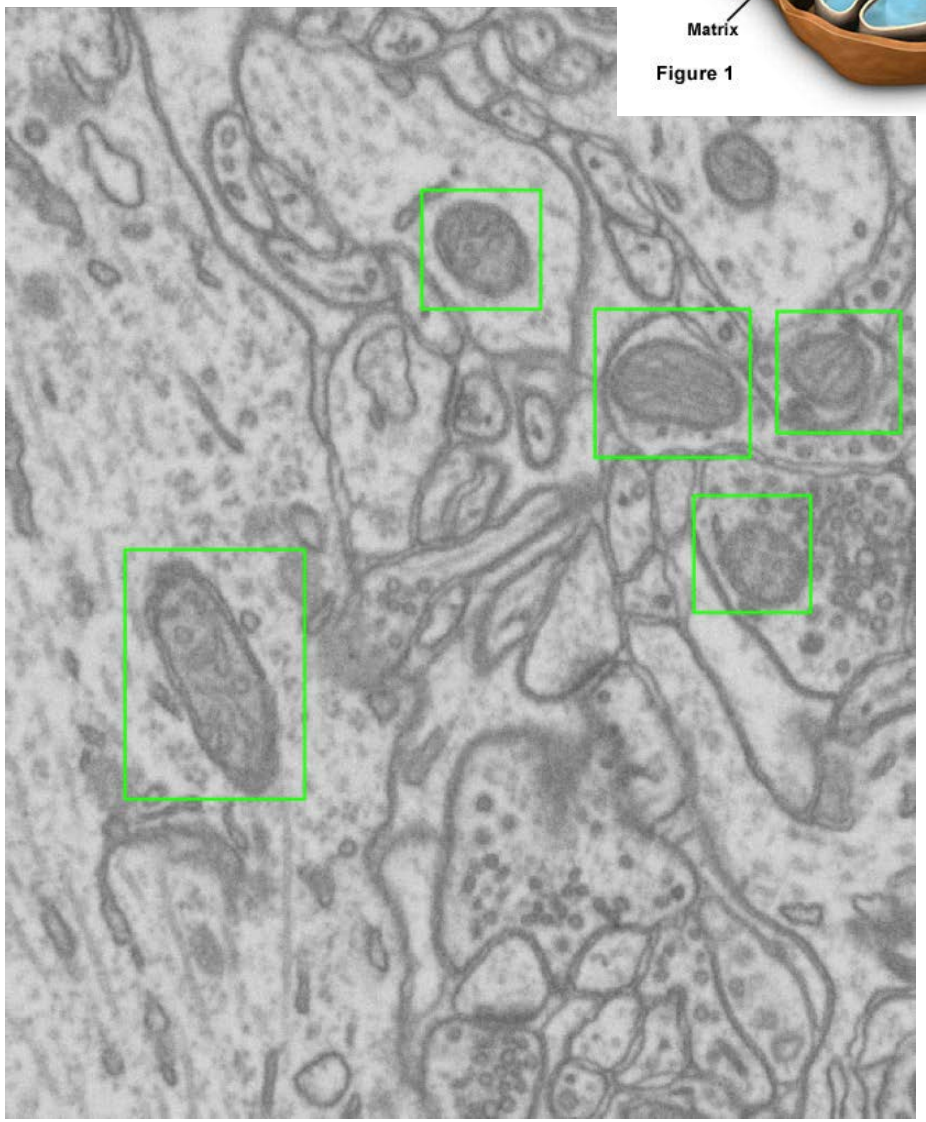
Homogeneous or Not?

What is homogeneous in some parts of these images are the statistical properties, not the actual pixel values.

Mitochondria



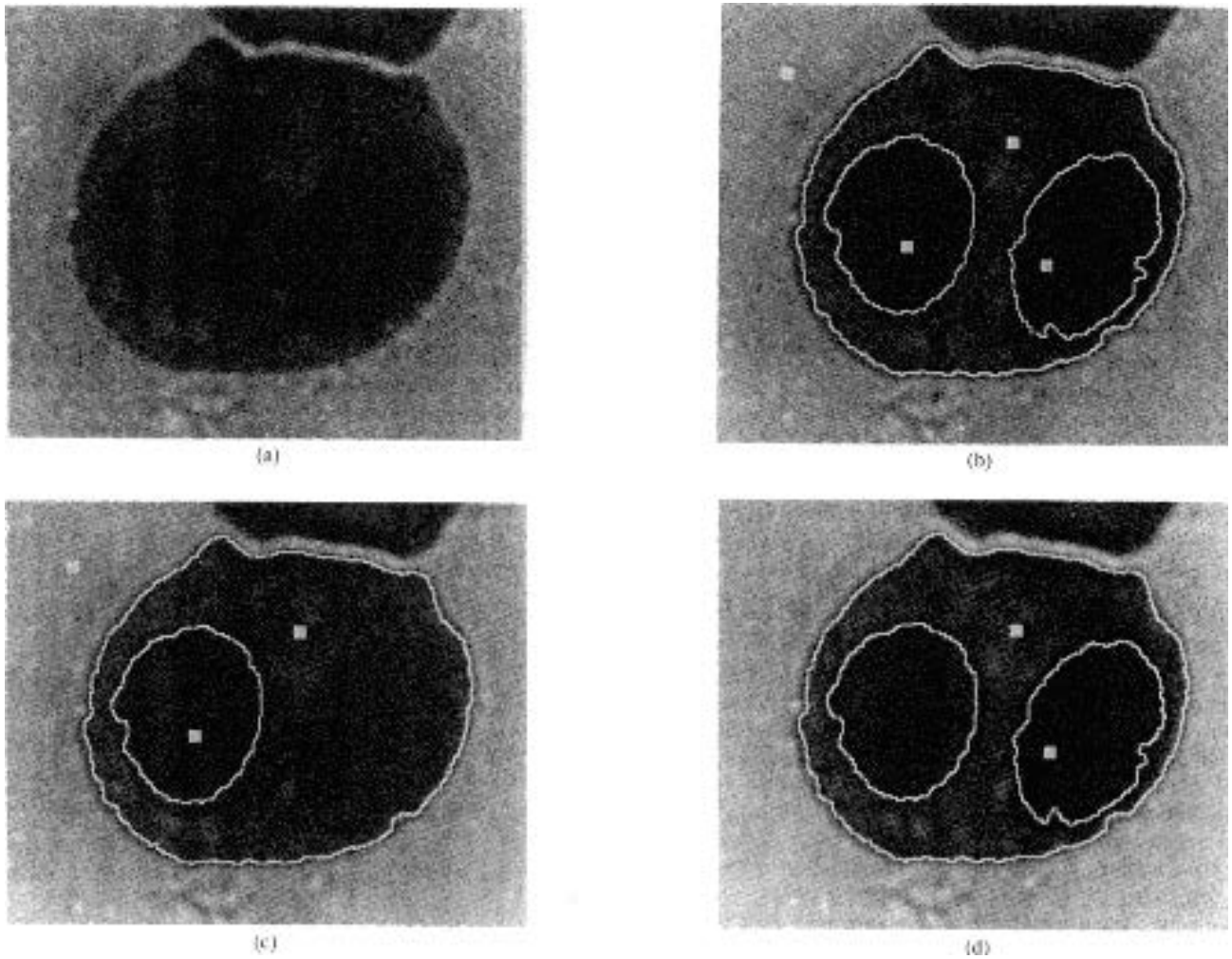




From Simple to Complex Algorithms

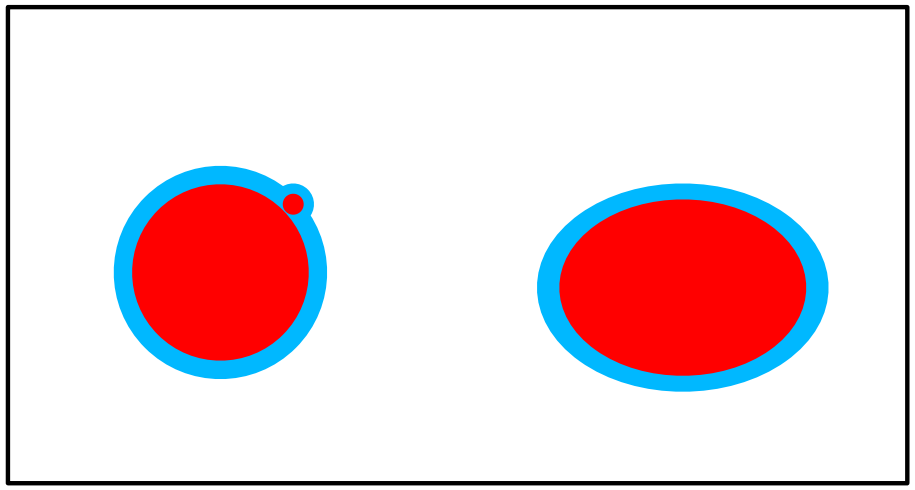
- Region Growing.
- Histogram Splitting.
- K-Means.
- Graph Theoretic Methods.
- Convolutional Neural Nets.

Region Growing



Interactive Segmentation of a Cell

Region Growing



- Labeled pixels.
- Unlabeled pixels.

Given a set of regions $A_1, \ldots, A_n$, let

$$T = \{x \notin (\cup_{i=1}^{n} A_i), N(x) \cap (\cup_{i=1}^{n} A_i) \neq \varnothing\},$$

the set of unlabeled pixels that are neighbors of already labeled ones and d be a metric, such as

$$\delta(x) = |g(x) - \text{mean}_{y \in A_{i(x)}}[g(y)]|.$$

Until all pixel are labeled:

1. Represent T as a sorted list, the SSL, according to this metric.
2. Label the first point in T.
3. Add its the neighbors to the SSL.

Region Growing

While SSL is not empty do

Remove first pixel y from SSL.

If all already labeled neighbors of y, other than boundary pixels, have the same label

then

Set y to this label.

Update running mean of corresponding region.

Add neighbors of y that are neither already set nor already in the SSL to the SSL according to their distance value.

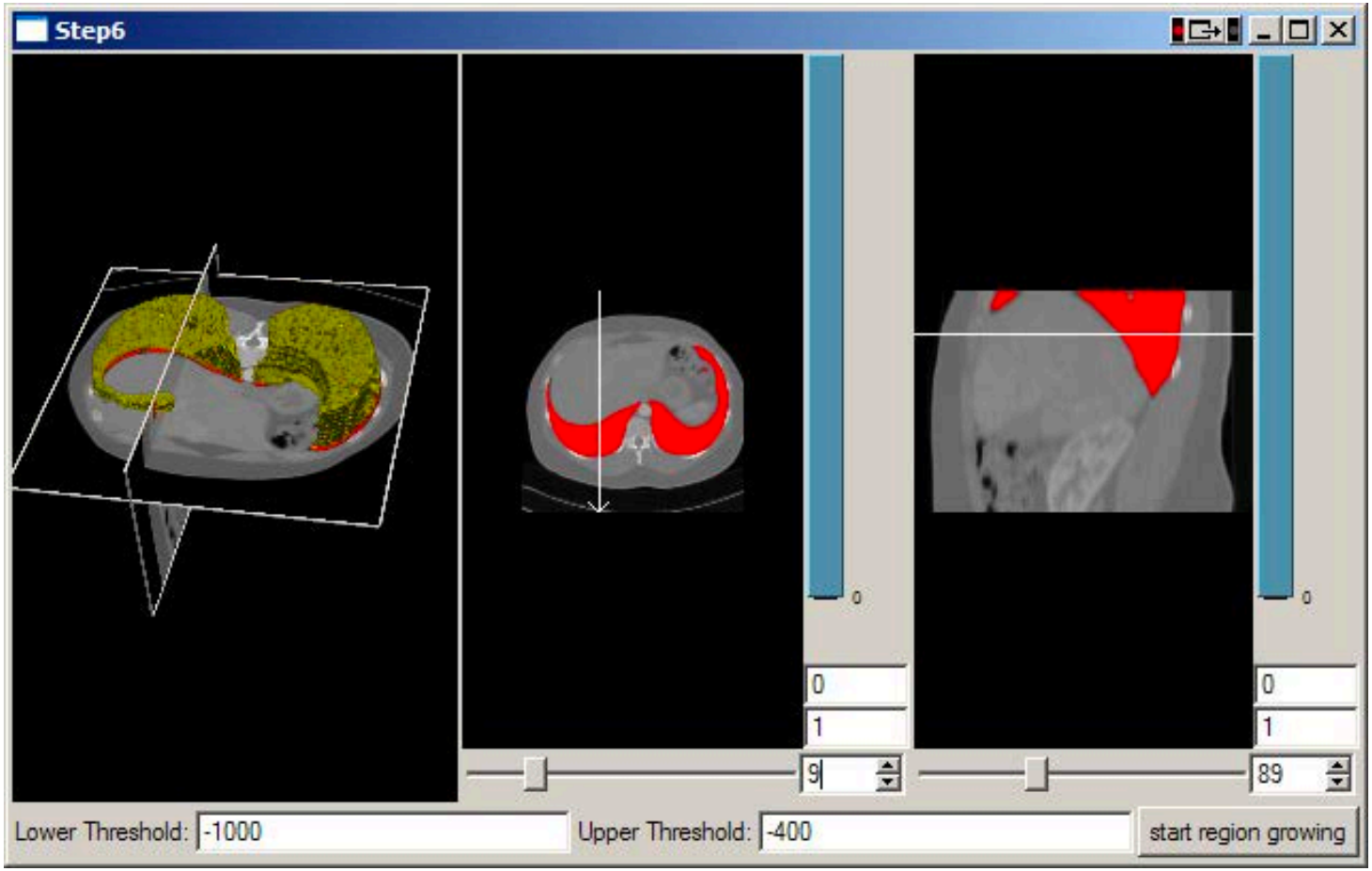
else

Flag y as a boundary pixel.

fi

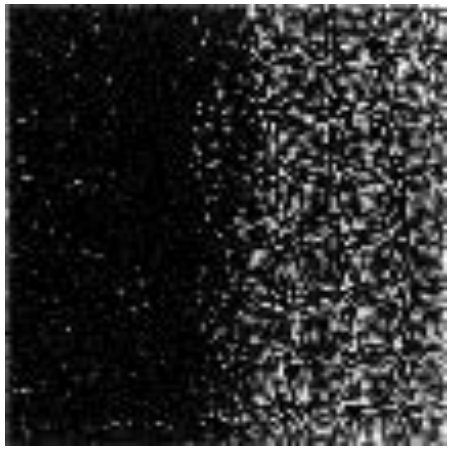
od

Interactive Region Grower

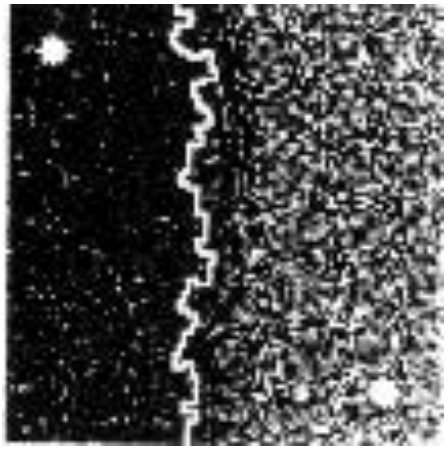


Medical Imaging Interaction Toolkit

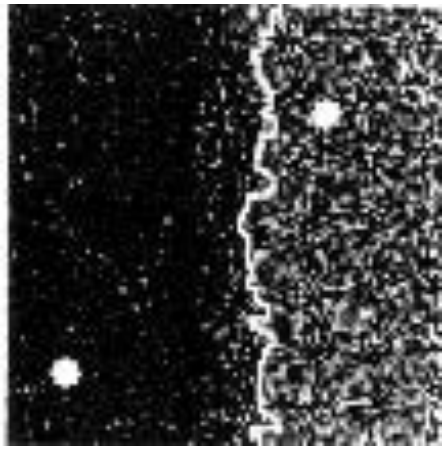
Limitations



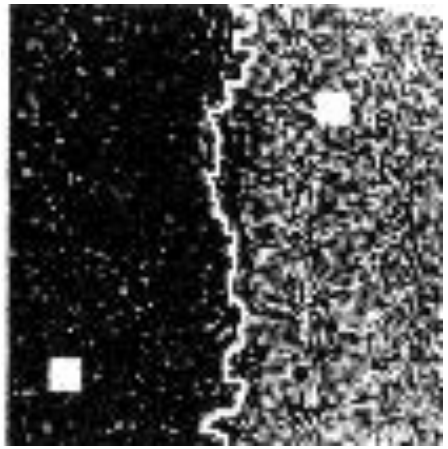
Original image



Result given
two seeds.



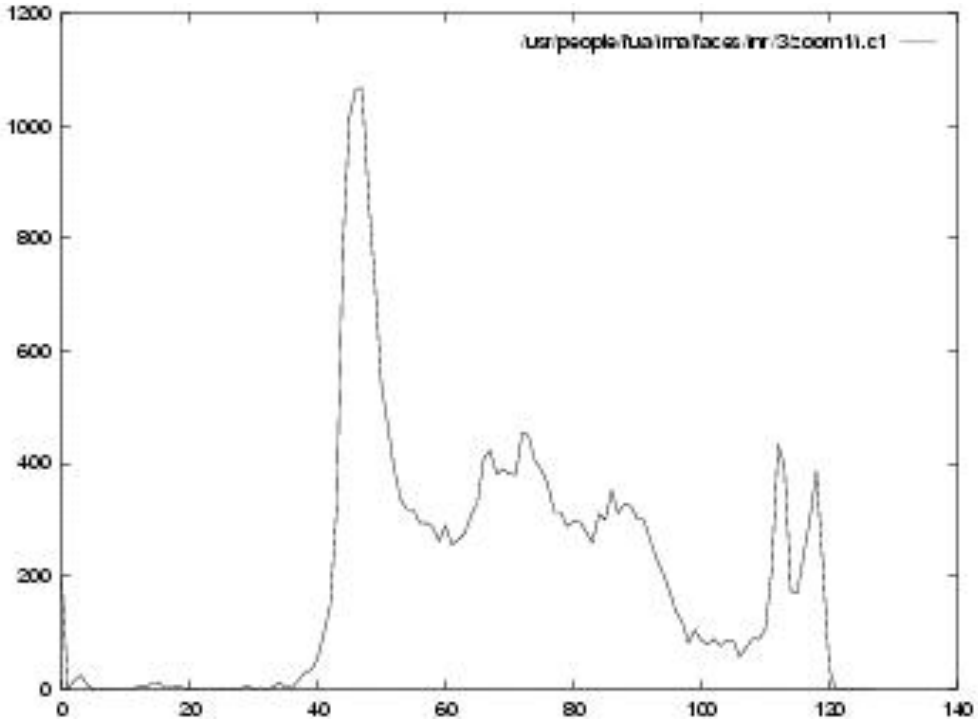
Result given
two different seeds.



Result given
larger seeds.

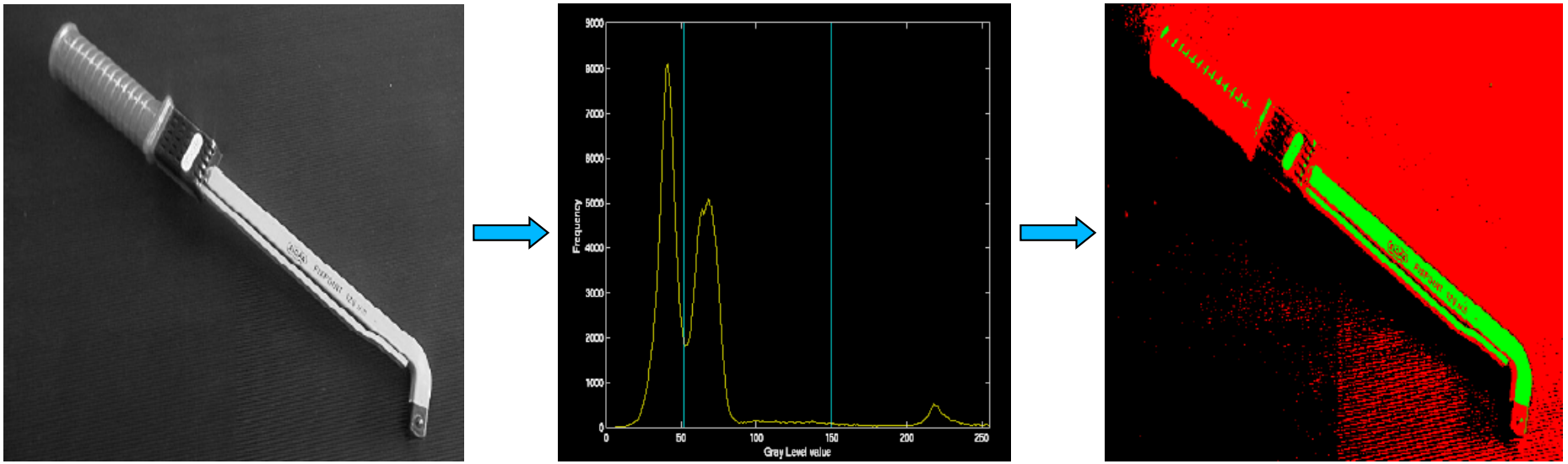
- The result depends on the order in which the pixels are taken into consideration.
- The homogeneity measure is noise sensitive.

Image Histogram



Number of pixels that have a given gray level.

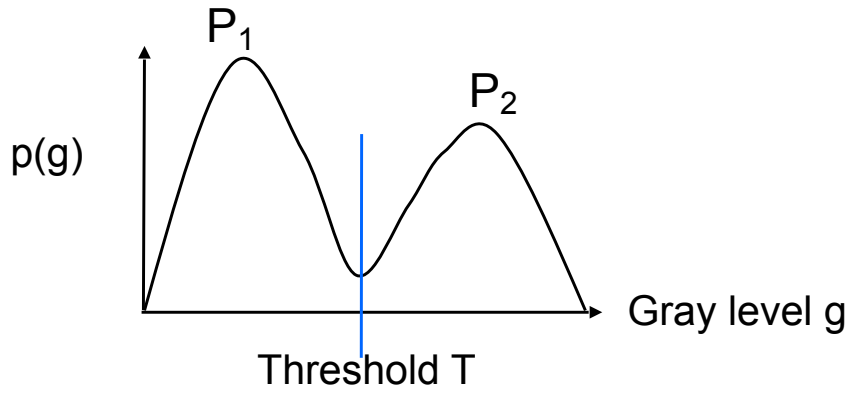
Histogram Splitting



- Groups of similar pixels appear as bumps in the brightness histogram
- Split the histogram at local-minima
- Label pixels according to which bump they belong to

—> Cannot stop there, must go on.

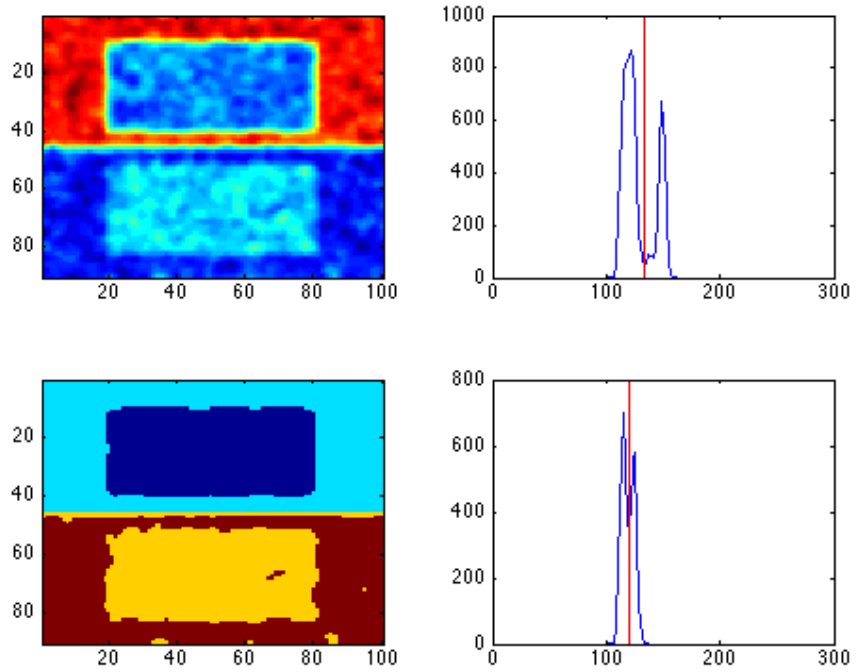
Recursive Splitting



- Compute image histogram.
- Smooth histogram.
- Look for peaks separated by deep valleys.
- Group pixels into connected regions.
- Smooth these regions.
- Iterate.

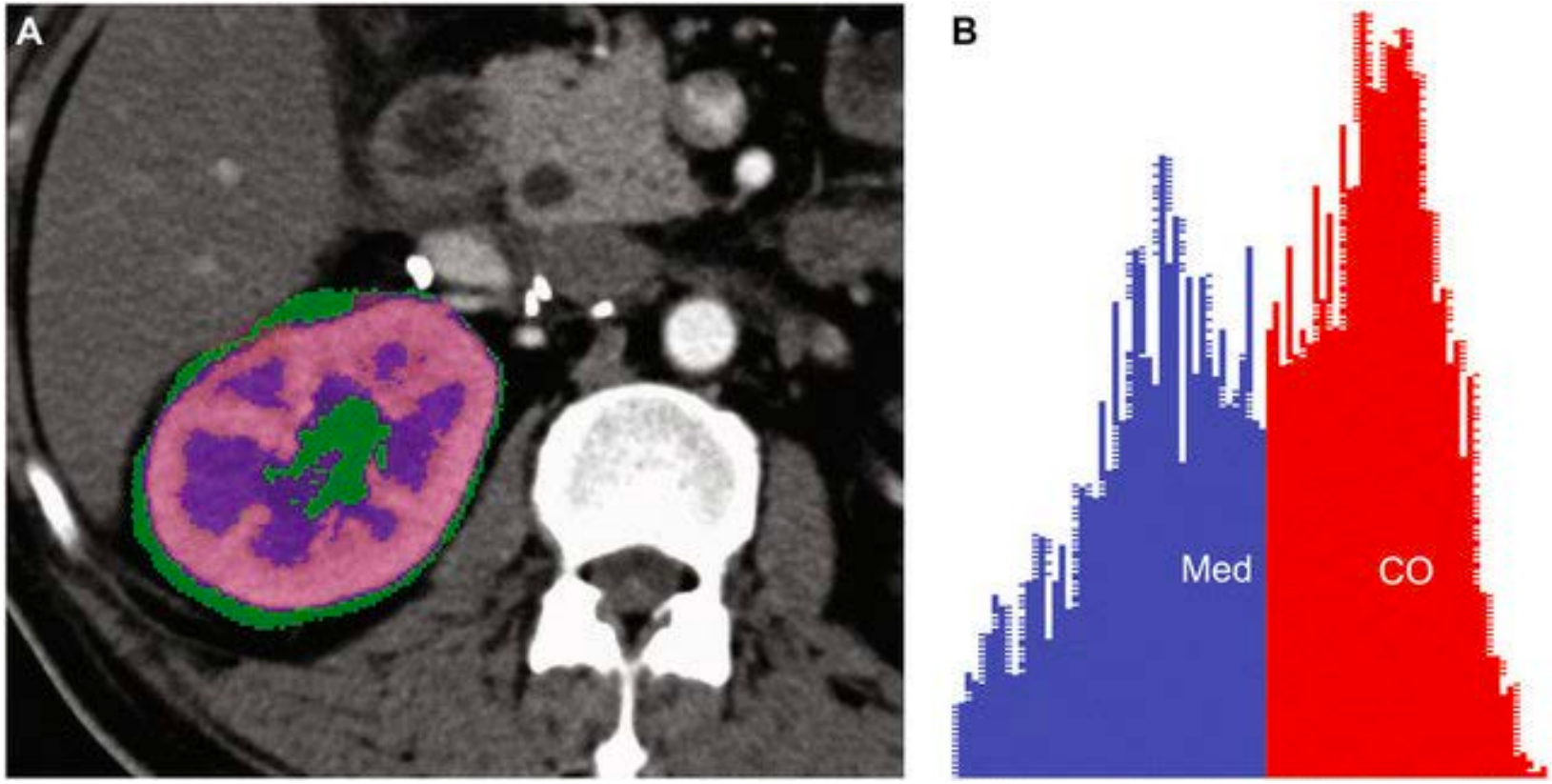
Recursive Splitting

- A first threshold is used to segment the dark pixels.
- This yields two regions, the bottom half of the picture and the dark rectangle at the top.
- The bottom half of the picture can now be more easily segmented into two regions.



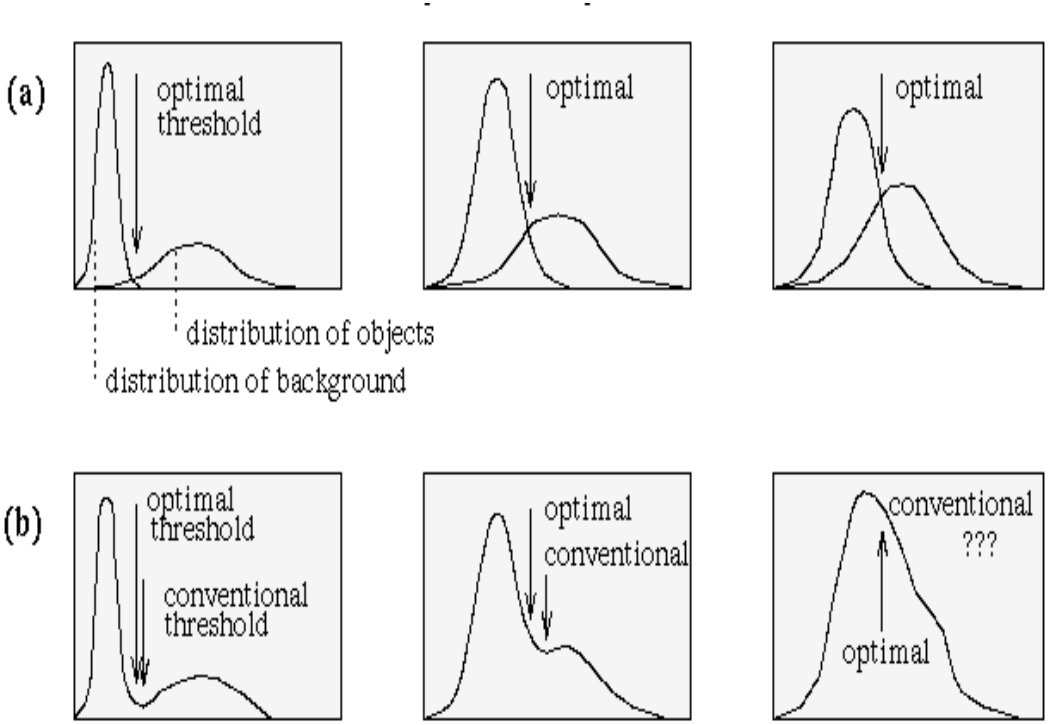
-> Decisions can be deferred until enough information becomes available.

Medical Application



It has its applications but .....

Finding Thresholds is Hard

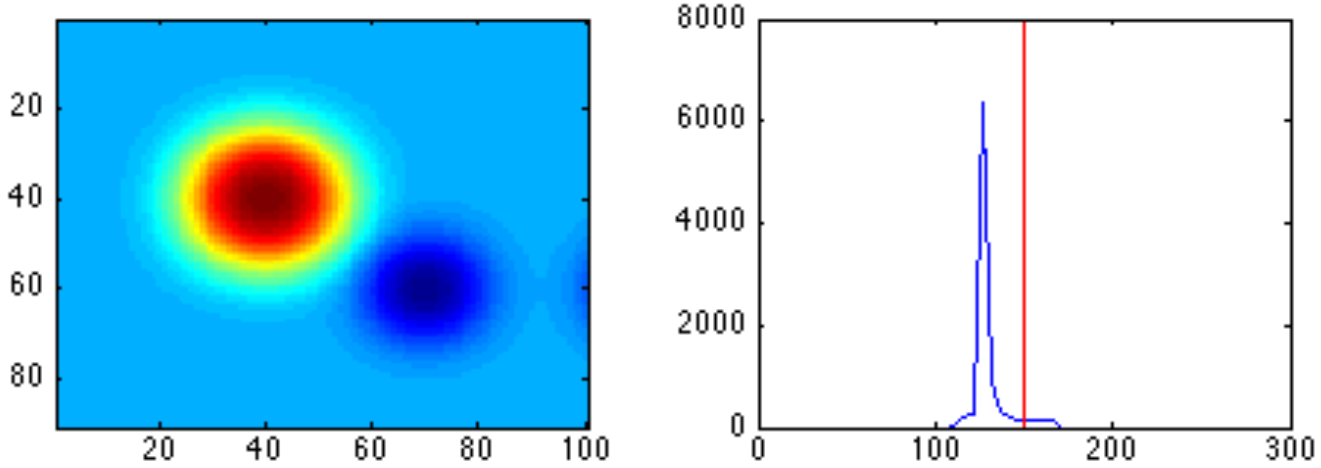


Probability distributions

Corresponding histograms

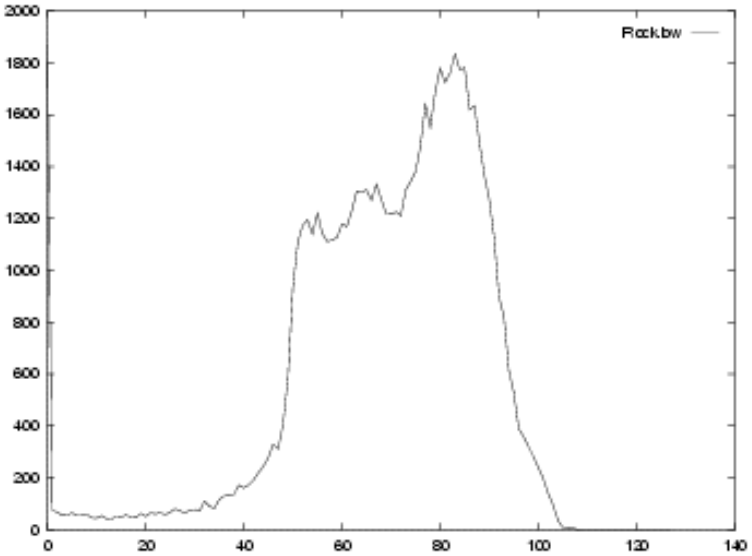
—> Choosing optimal thresholds is a difficult optimization problem.

No Obvious Threshold



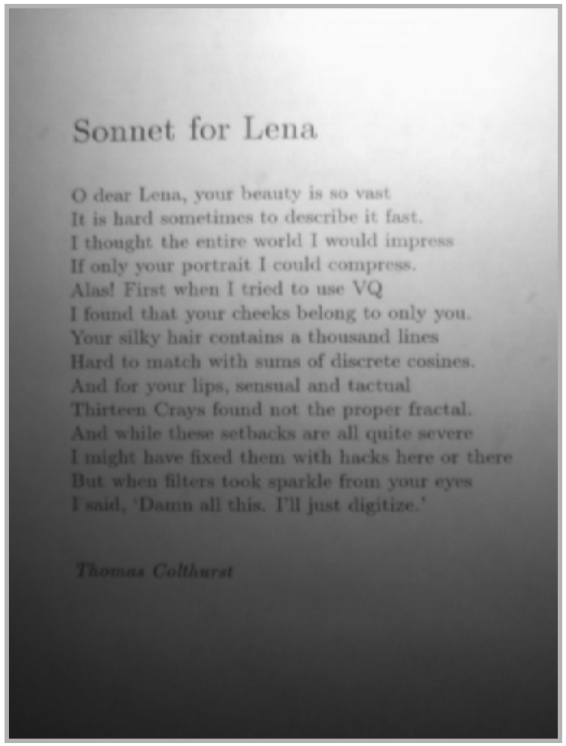
There are thresholds that would work but you can’t find them from the histogram only.

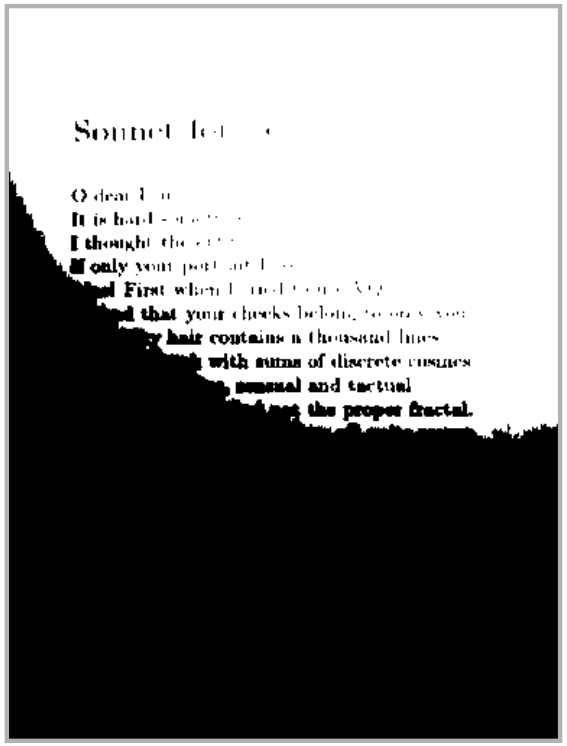
No Valid Threshold at All



In this image, no threshold can generates meaningful results.

No Global Threshold

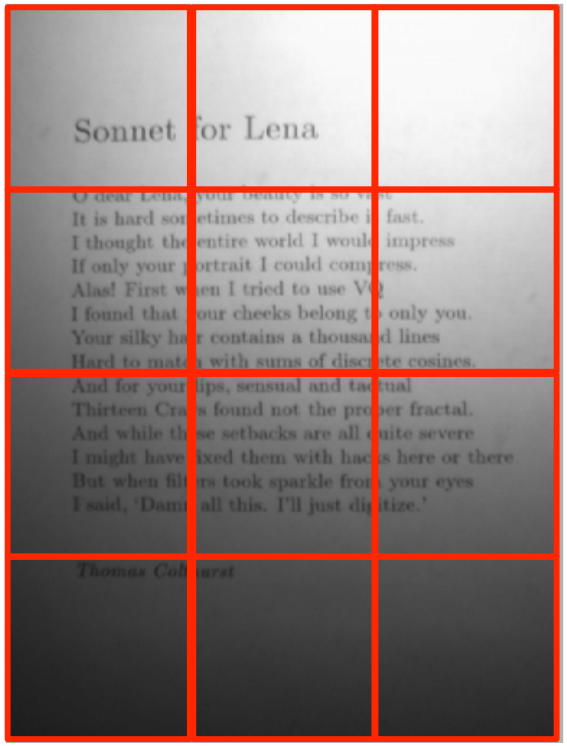


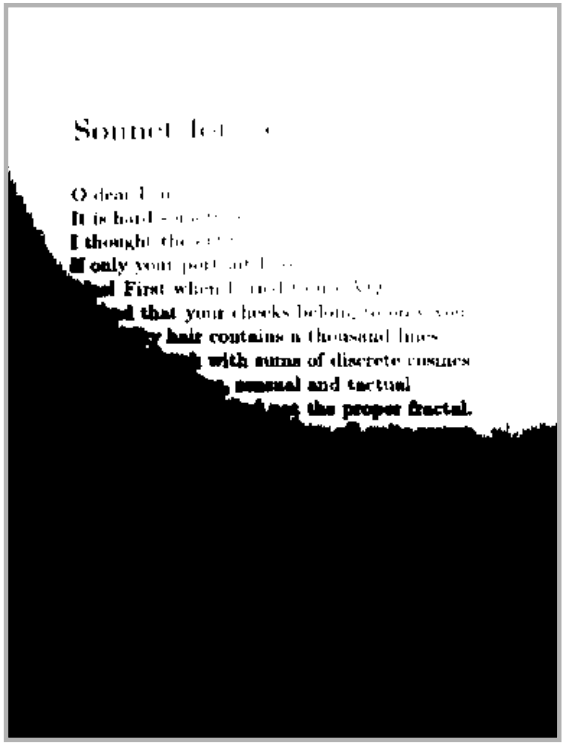


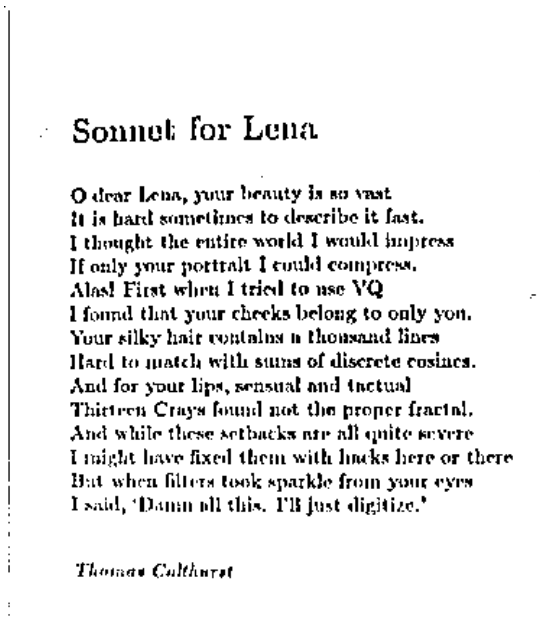
Because of the complex illumination:

- The top of the page is much lighter than the bottom.
- No global threshold can be found.
- Local thresholds can account for this.

Use Local Histograms



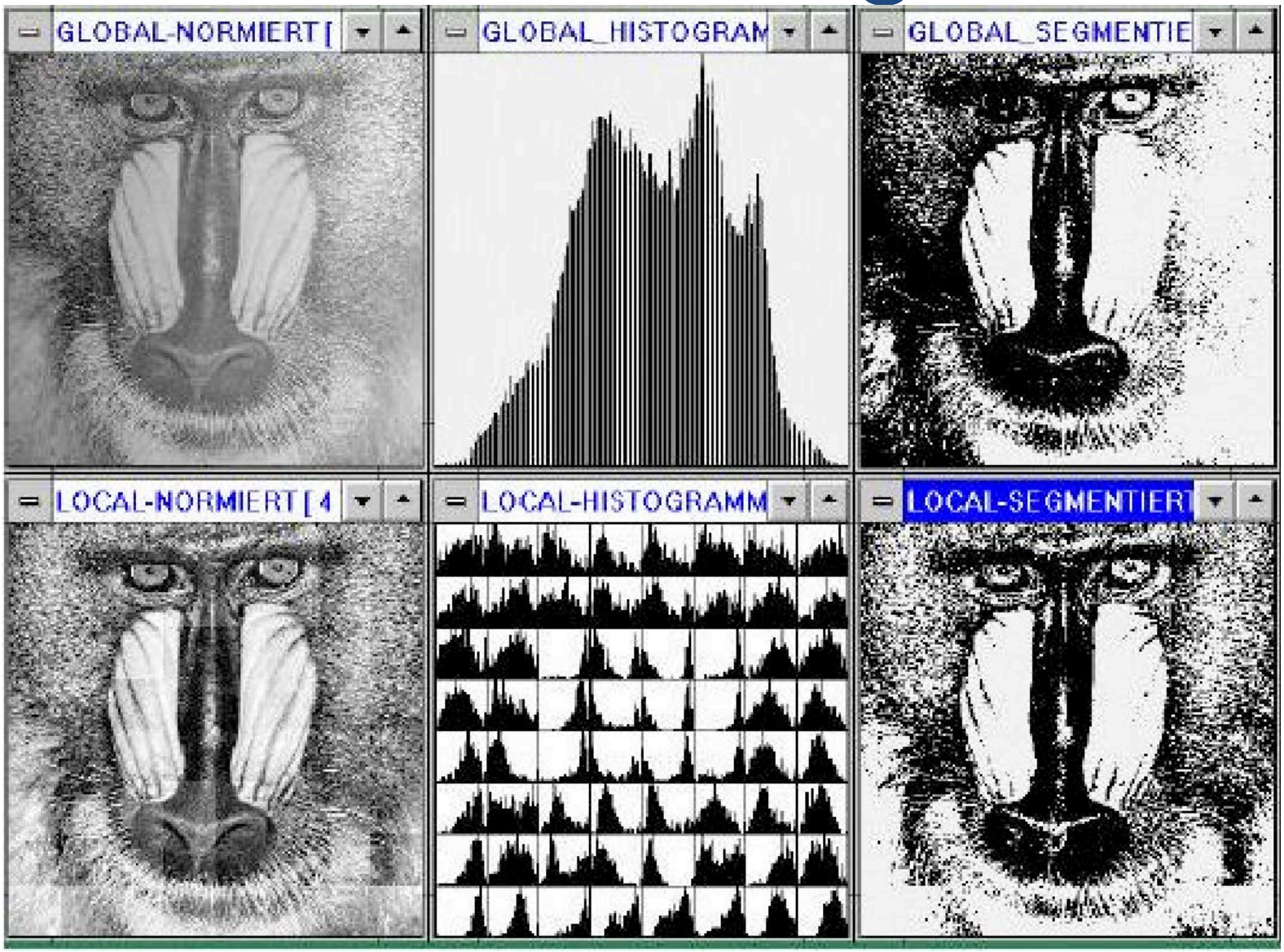




Global Local

- Compute a histogram in each red square.
- Find a local threshold.

Use Local Histograms



- Compute local histograms on a coarse grid.
- Use them instead of a global one.

Limitations of Global Histograms

- Histograms do not account for neighborhood relationships.
- Thresholds are hard to find.
- Some boundaries will not be found because the gray levels on both side belong to the same histogram peak.

Using Color

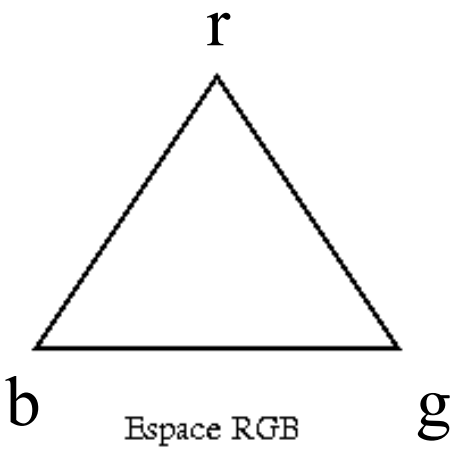
RGB

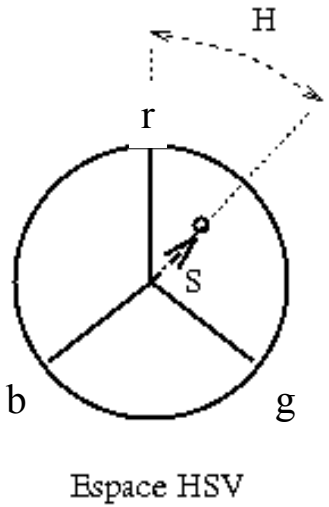
R

B

G

Color Space





$$V = R + G + B$$

$$r = \frac{R}{V}$$

$$g = \frac{G}{V}$$

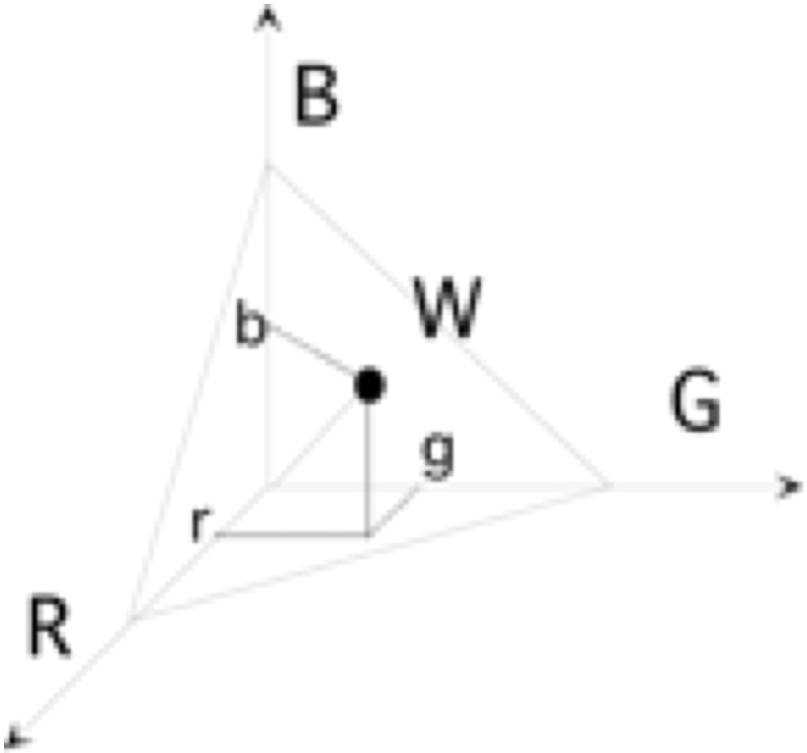
$$b = \frac{B}{V}$$

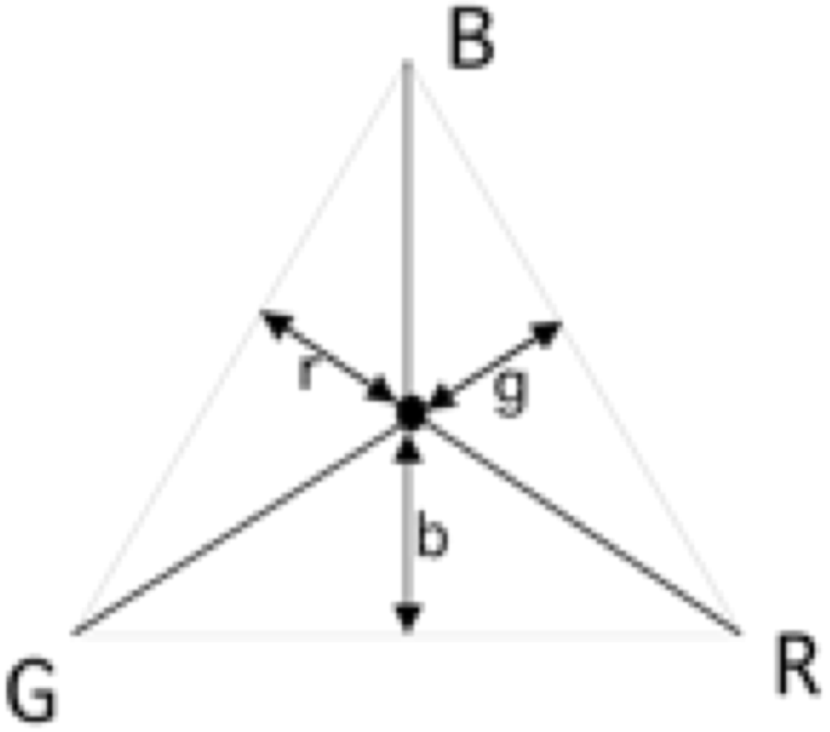
- Value

- Hue

- Saturation

RGB Chromaticity Diagram

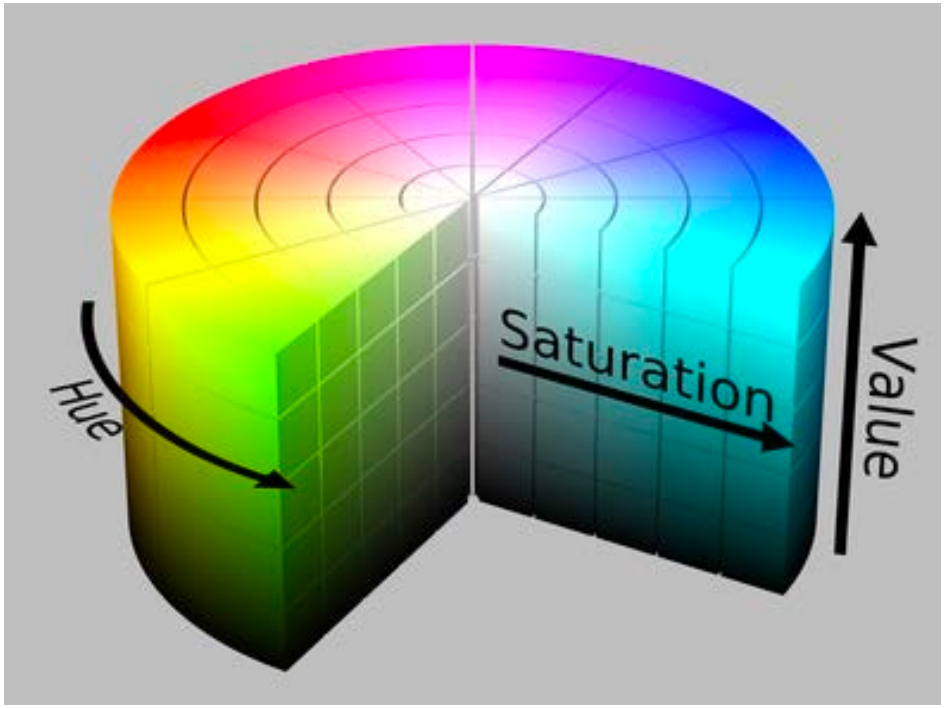




The Maxwell triangle involves projecting the colors in RGB space onto R+G+B=1 plane.

→ Chromaticity becomes independent from luminance.

HSV Space



Hue / Saturation / Value

$$H = \arccos\left(\frac{0.5(2r - g - b)}{\sqrt{(r-g)^2 + (r-g)(g-b)}}\right) \text{ if } b < g$$

$$S = \max(r, g, b) - \min(r, g, b)$$

$$V = R + G + B$$

HSV Images

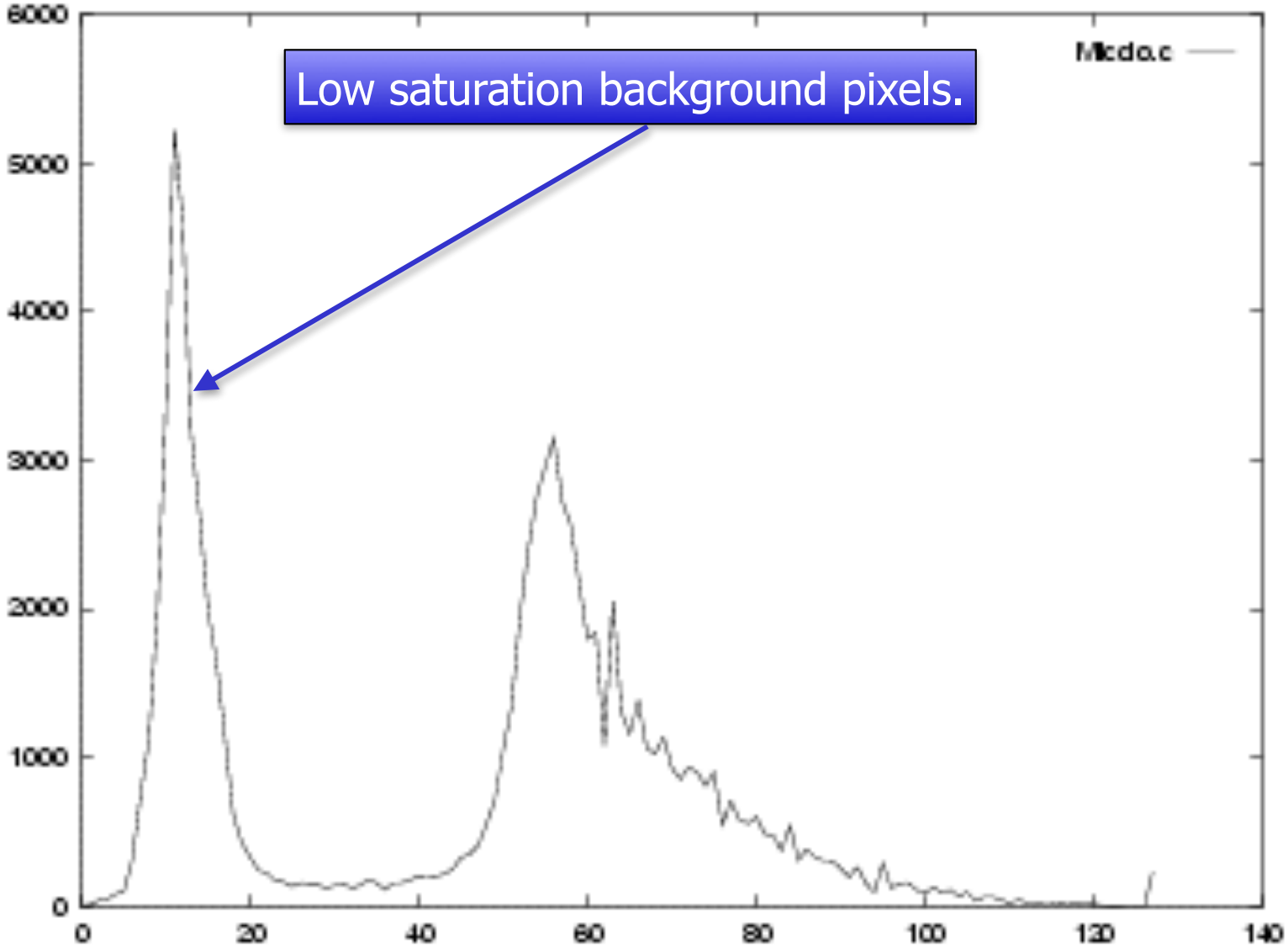
RGB

Value

Saturation

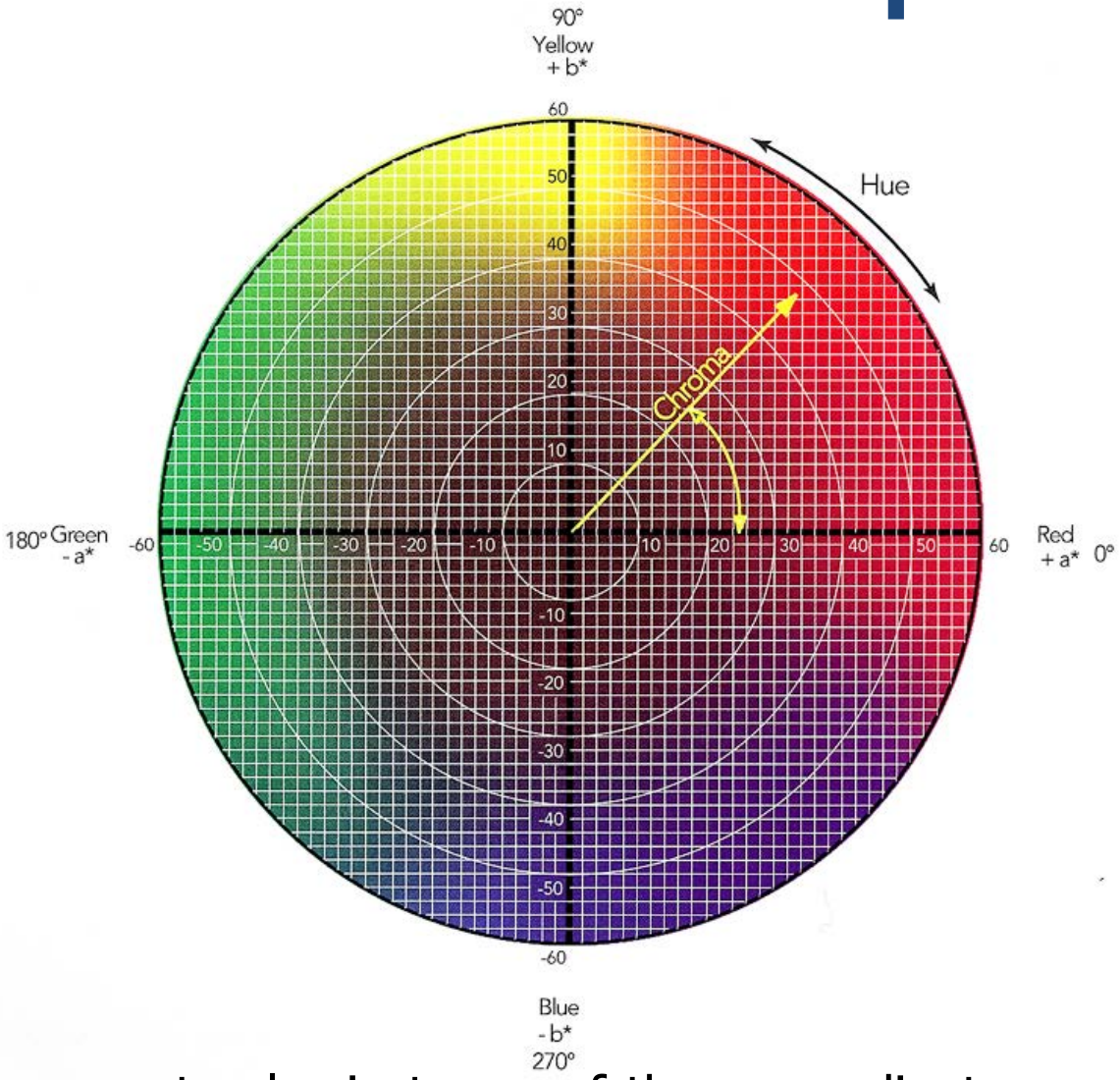
Hue

Saturation Histogram



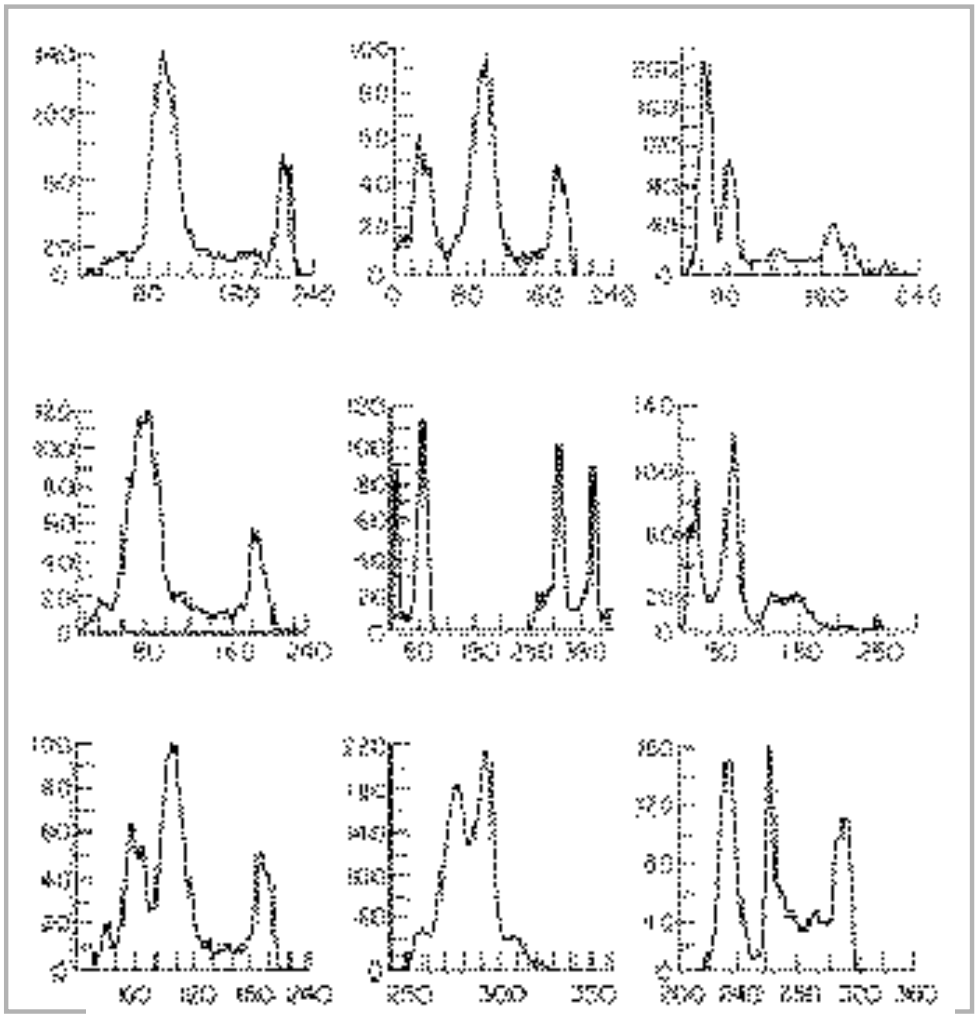
- This histogram is a lot easier to split!
- It makes it easy to segment the head from the background.

CIE LAB Color Space



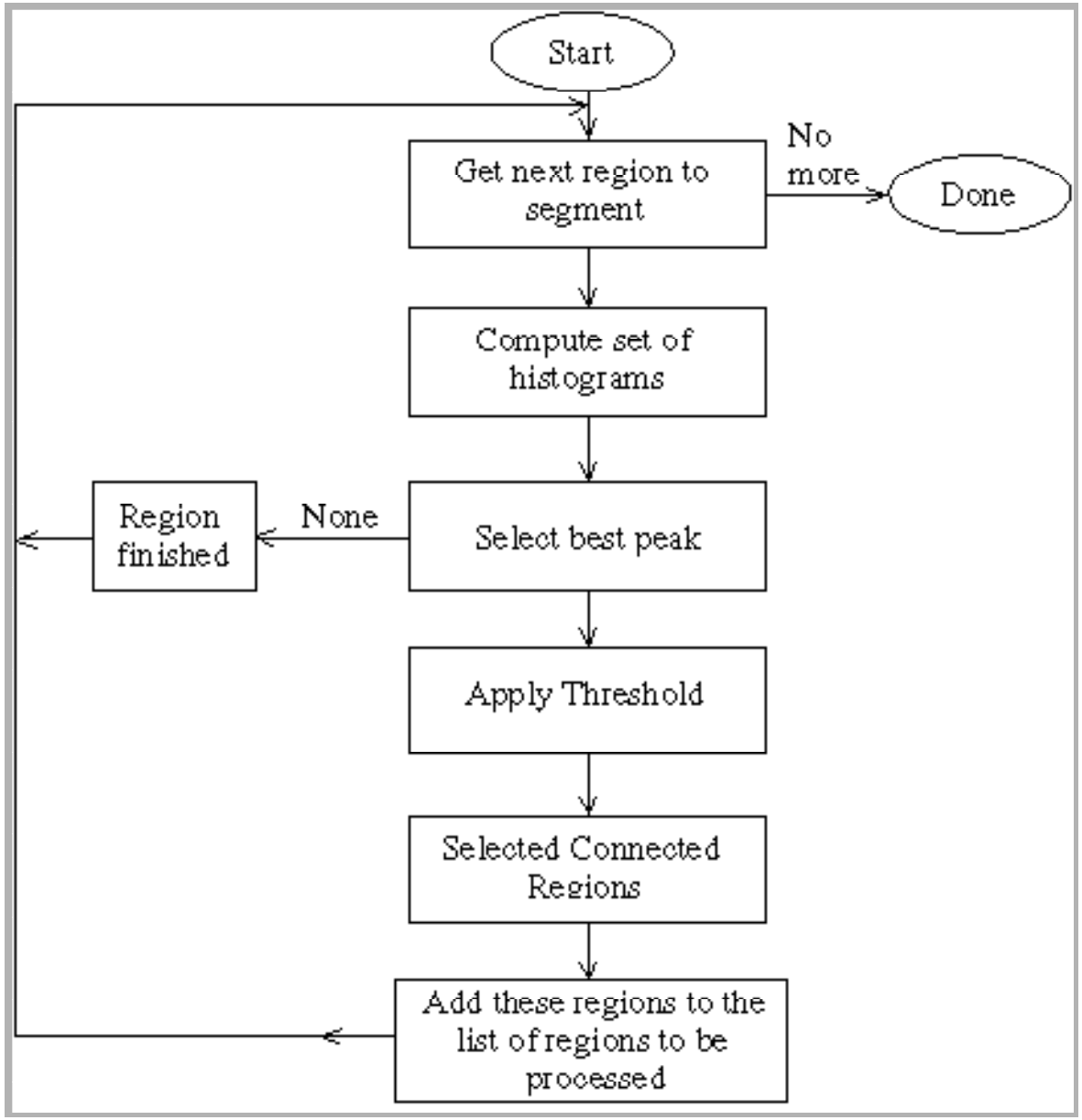
- Another way to represent color in terms of three coordinates, one for "brightness" and the other two for chromaticity.
- Designed so that the same amount of numerical change in these values corresponds to roughly the same amount of visually perceived change.

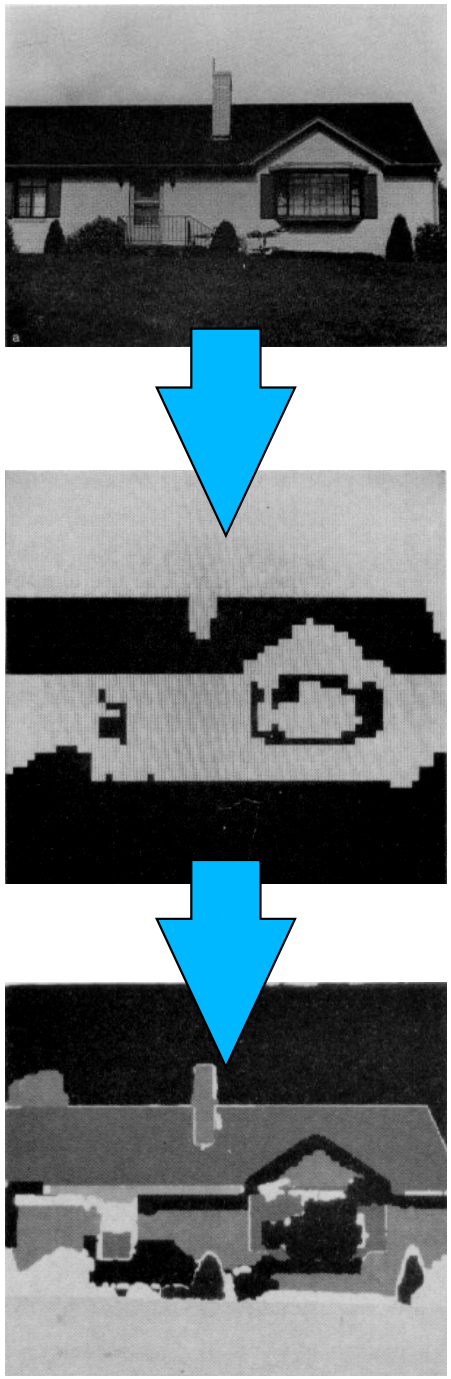
Using Multiple Histograms



- Compute multiple histograms for each segment.
- Use one of them to split each segment.
- Repeat on the resulting smaller segments.

Recursive Algorithm





Early Color Segmentation

Ohlander & Price, CGIP'78

More Recent Color Segmentation

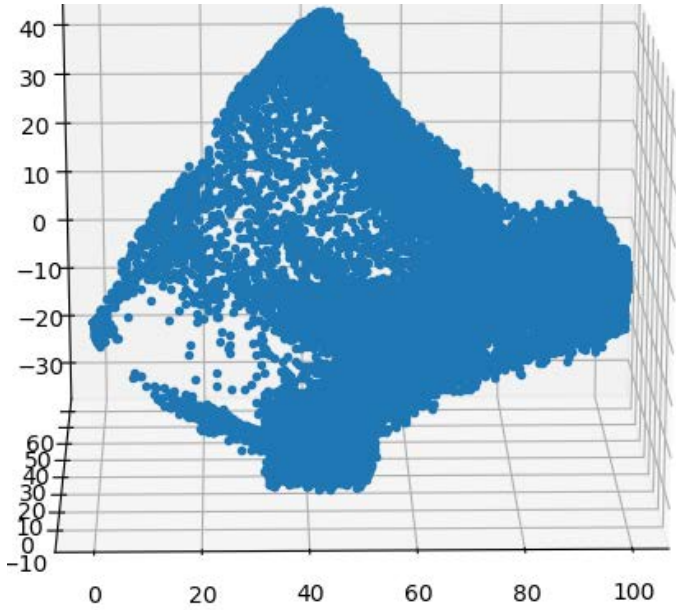
I find this blue sky too bland!

Replace it.

Manual intervention was still required to find the whole sky region the sky region!

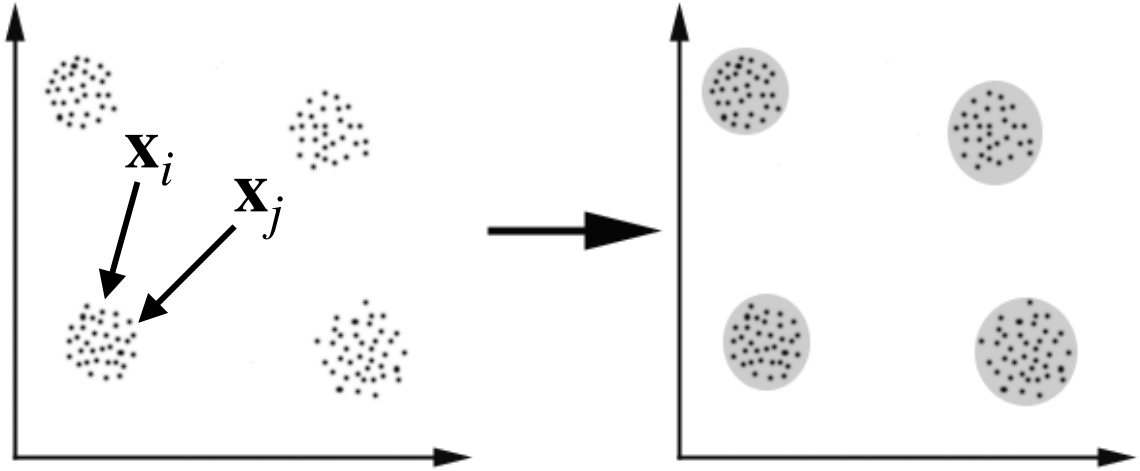
https://www.photoshopessentials.com/basics/selections/magic-wand-tool/

Segmentation as Clustering



- Each pixel has 2 spatial coordinates and 1 gray level or 3 color components.
- Segmentation can be understood as clustering in
  - 1D space (G);
  - 3D space (x,y,G), (H,S,V), (L,a,b);
  - 5D space (x,y,L,a,b).
- Ideally each cluster should be as compact as possible.

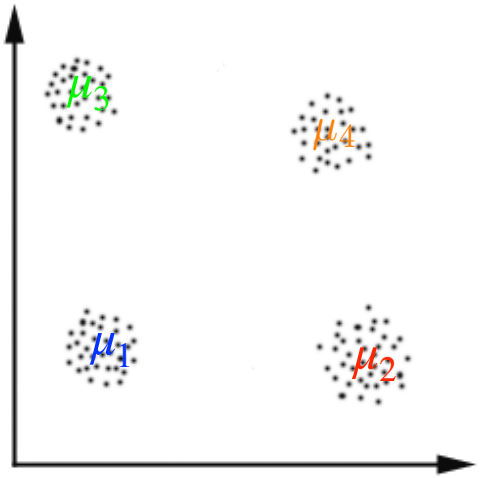
K-Means Clustering



Given a set of input samples:

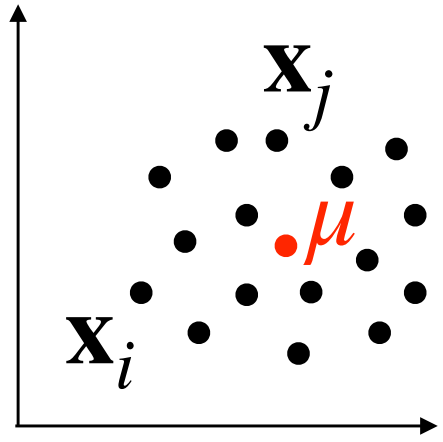
- Group the samples into K clusters.
- K is assumed to be known/given.
- In the example above, each sample is a point in 2D.
- In images, samples can be points in 1D, 3D, or 5D.

K-Means Clusters



- Cluster k is formed by the points $\{\mathbf{x}_{i_1^k}, \ldots, \mathbf{x}_{i_{n^k}^k}\}$.
- μ_k is the **center of gravity** of cluster k.

The mean of points $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, $\mathbf{x}_i \in \mathbb{R}^D$ is

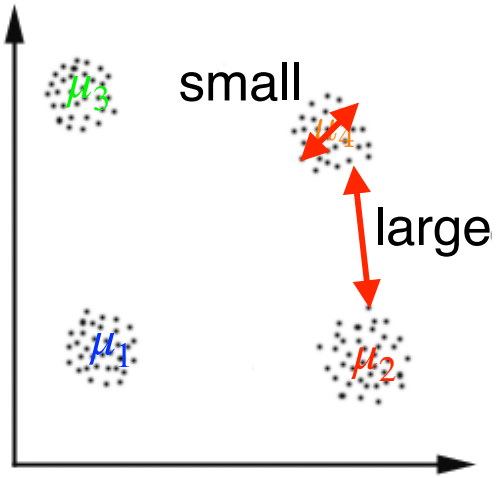


In 2D

$$\mu = \frac{1}{N}\sum_{i=1}^{N} \mathbf{x}_i \,, \mu \in \mathbb{R}^D$$

- If the $\mathbf{x}_i$ were physical points of equal mass, m would be their center of gravity.
- This applies in any dimension.

Formalization



- Cluster k is formed by the points $\{\mathbf{x}_{i_1^k}, \ldots, \mathbf{x}_{i_{n^k}^k}\}$.
- μ_k is the **center of gravity** of cluster k.

- The distances between the points within a cluster should be small.
- The distances across clusters should be large.
- This can be encoded via the distance to cluster centers $\{\mu_1, \ldots, \mu_K\}$:

$$\text{—> Minimize } \sum_{k=1}^{K} \sum_{j=1}^{n_k} (\mathbf{x}_{i_j^k} - \mu_k)^2$$

where $\{\mathbf{x}_{i_1^k}, \ldots, \mathbf{x}_{i_{n^k}^k}\}$ are the n^k samples that belong to cluster k.

Difficult Minimization Problem

Minimize

$$\sum_{k=1}^{K} \sum_{j=1}^{n_k} (\mathbf{x}_{i_j^k} - \mu_k)^2$$

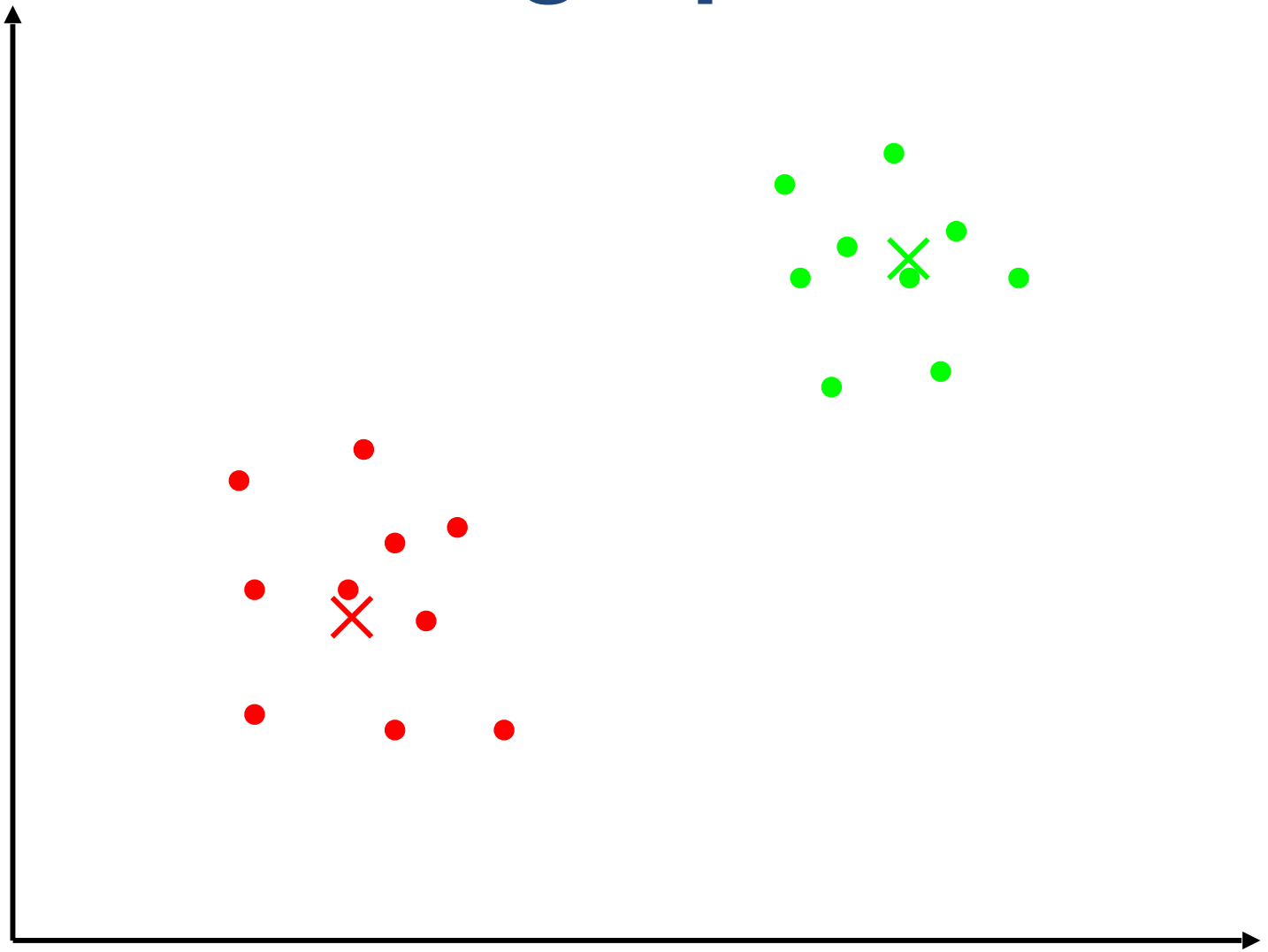
but:

- We don't know what points belong to what cluster.
- We don't know the center of gravity of the clusters.

Simple Solution to the Problem

1. Initialize $\{\mu_1, \ldots, \mu_K\}$, randomly if need be.
2. Until convergence
   - 2.1. Assign each point $\mathbf{x}_i$ to the nearest center μ_k
   - 2.2. Update each center μ_k given the points assigned to it

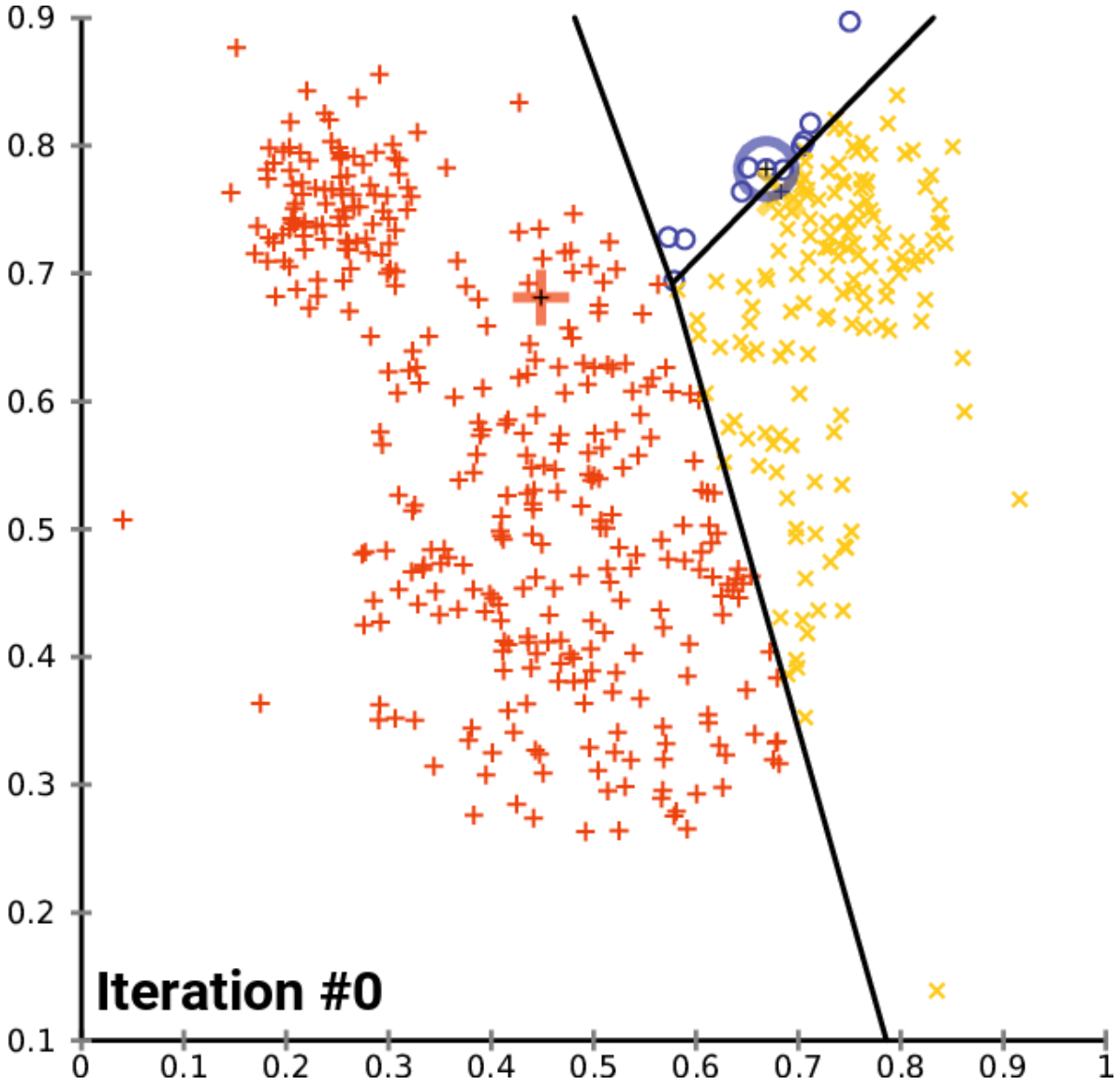
Alternating Optimization



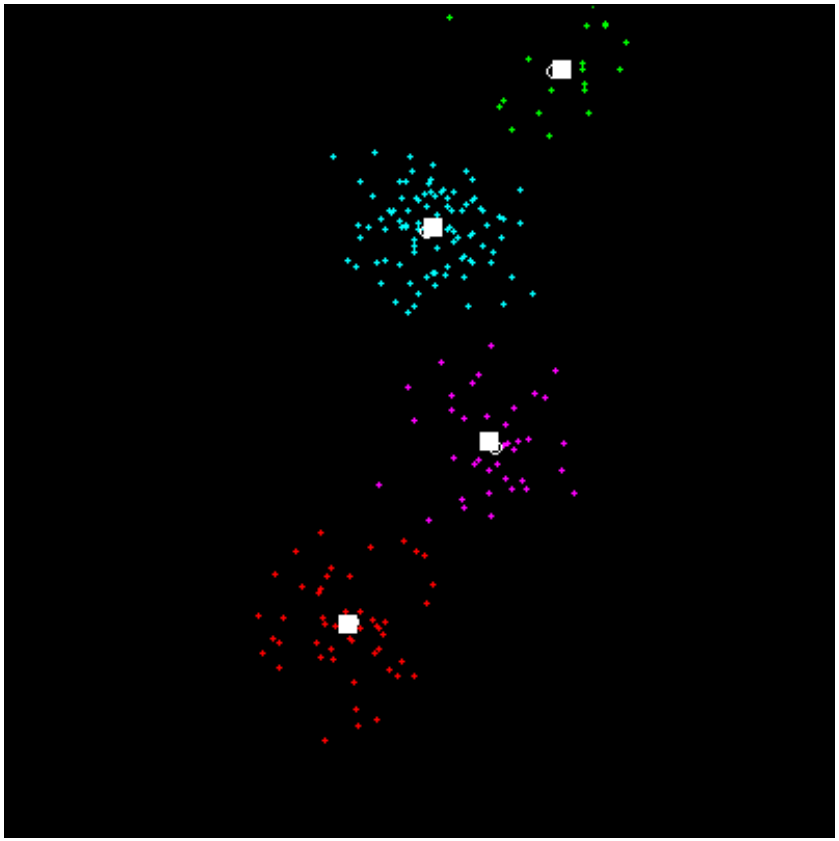
Demo

- Initialize
- Associate point to centers
- Recompute centers

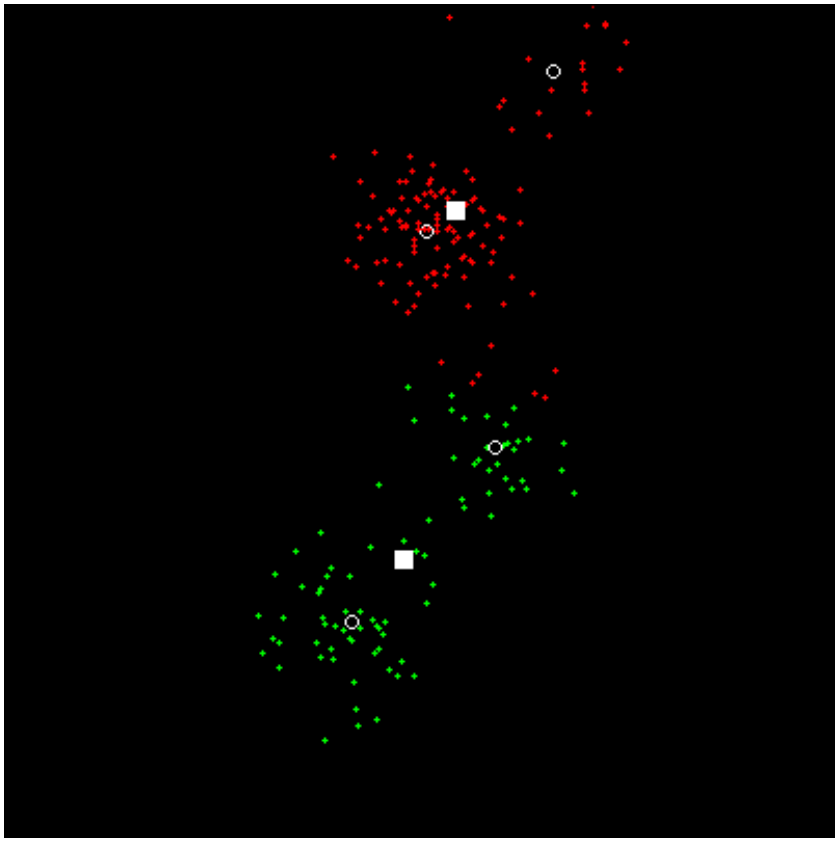
Three Classes



Initial Conditions Matter



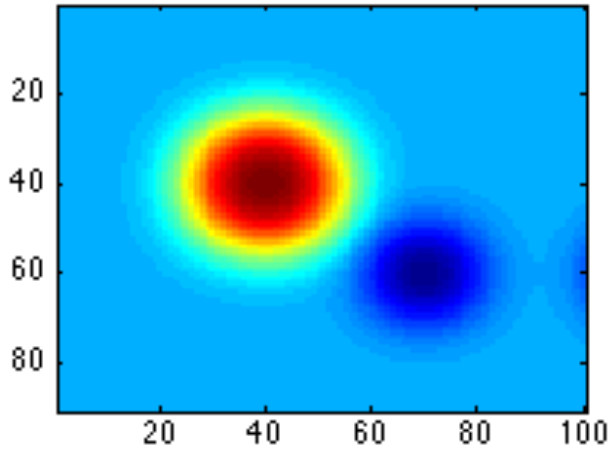
Initially, the points are assigned to the clusters at random —> Success.

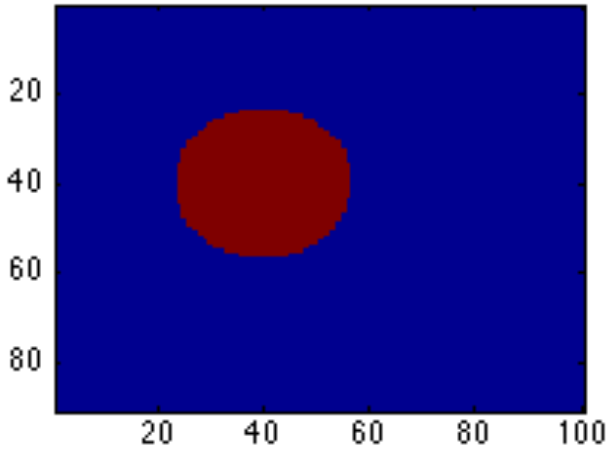


Initially, the points are assigned to the closest cluster —> Failure.

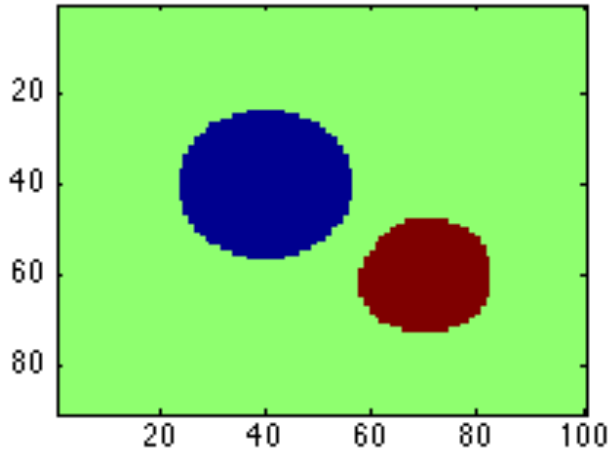
—> In practice try several different random initialization and keep the one that yields the best result in term of the sum of square distances.

GRAY-LEVEL ONLY (1D)

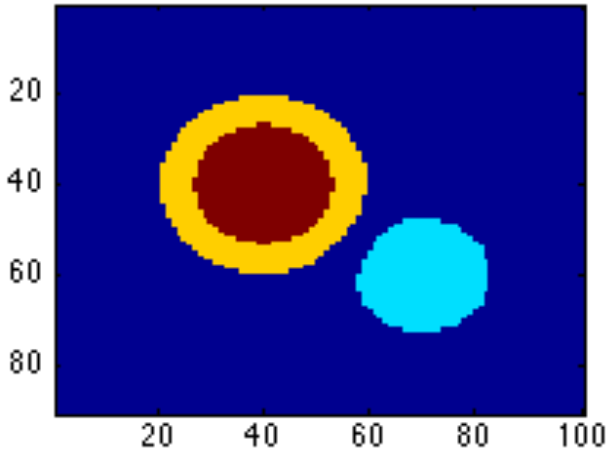




K=2

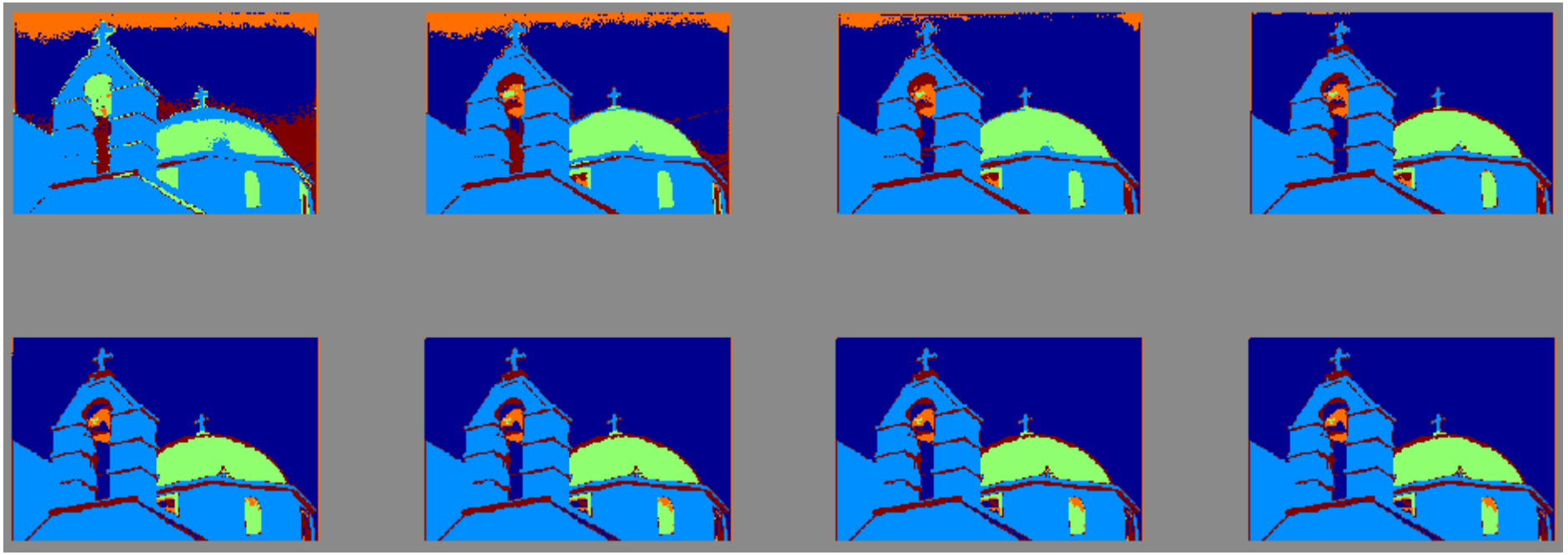


K=3



K=4

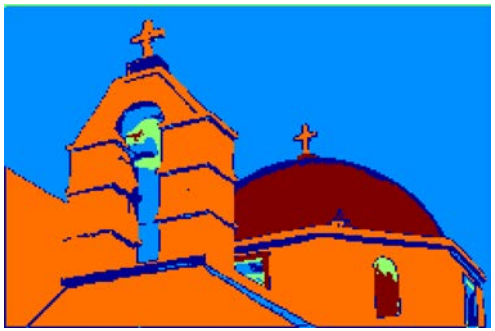
Color Only (3D)



8 iterations for k=5

Color Only (3D)

Different Initializations for k=5

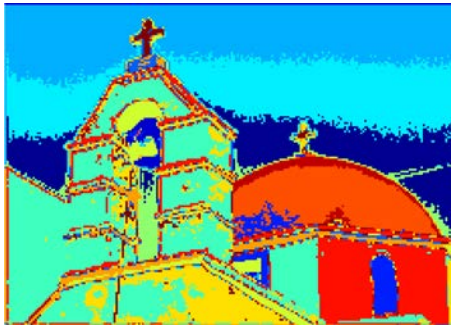



Different values of k


K=3

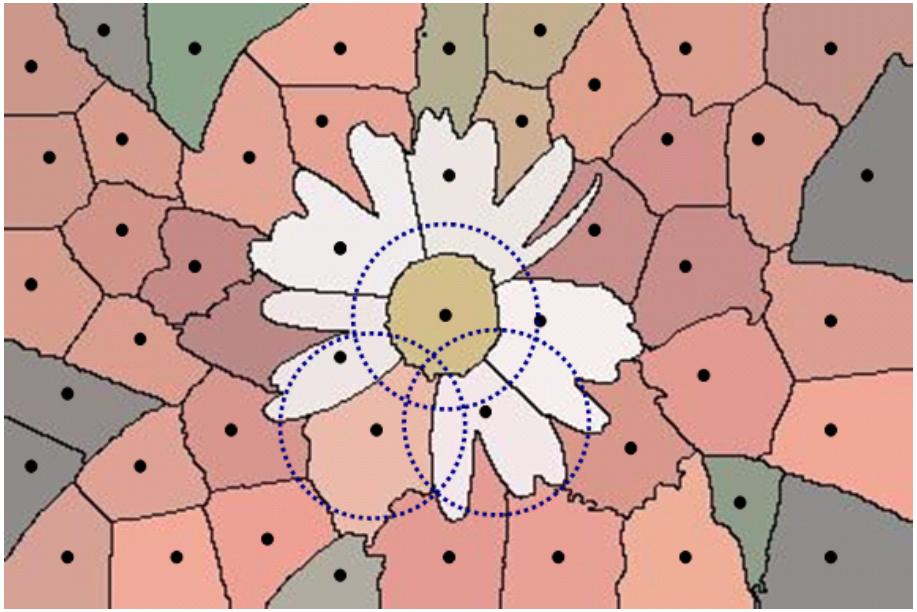

K=5


K=8


K=15

- Different results for different initializations.
- For k well chosen, the results are good for this image because it features bright and distinctive colors.

XY + Color (5D)



$$E(\mathcal{C}_1, \ldots, \mathcal{C}_k, \mathbf{c}_1, \ldots, \mathbf{c}_k) = \sum_j \sum_{i \in \mathcal{C}_j} d(\mathbf{x}_i, \mathbf{c}_j)^2$$

$$\mathbf{x} = \begin{bmatrix} u \\ v \\ L \\ a \\ b \end{bmatrix} \quad \text{or} \quad \begin{bmatrix} u \\ v \\ I \\ 0 \\ 0 \end{bmatrix}$$

$$d(\mathbf{x}, \mathbf{c})^2 = \frac{(\mathbf{x}[0] - \mathbf{c}[0])^2 + (\mathbf{x}[1] - \mathbf{c}[1])^2}{h_s^2} + \frac{(\mathbf{x}[2] - \mathbf{c}[2])^2 + (\mathbf{x}[3] - \mathbf{c}[3])^2 + (\mathbf{x}[4] - \mathbf{c}[4])^2}{h_r^2}$$

Run K-Means algorithm with regularly spaced seeds on a grid and using a distance that is a weighted sum of distances in image space and in gray level/color space.

—> Superpixels

Achanta et al. PAMI'12

SLIC Superpixels

1024x1024

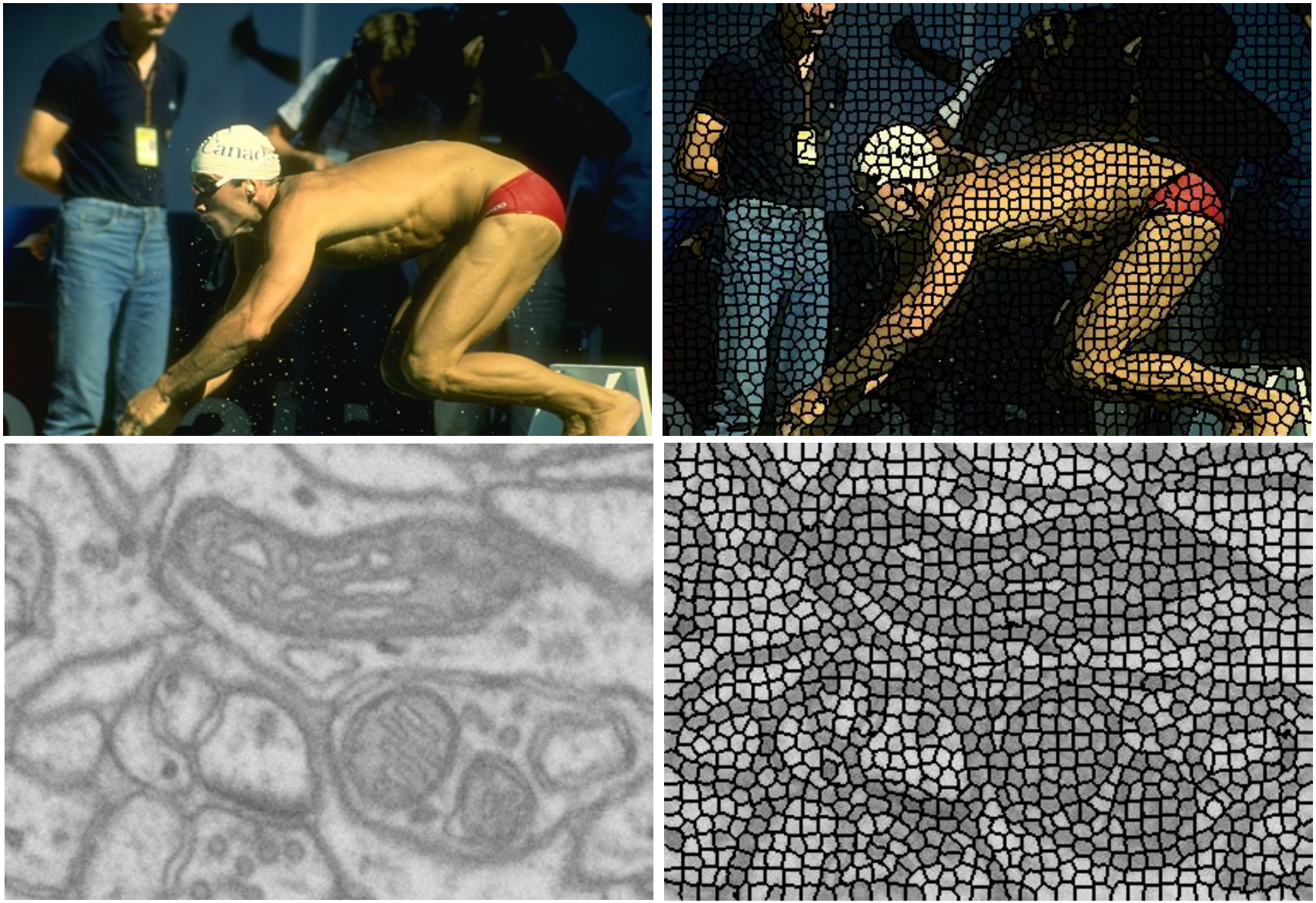
256x256

256x256

64x64

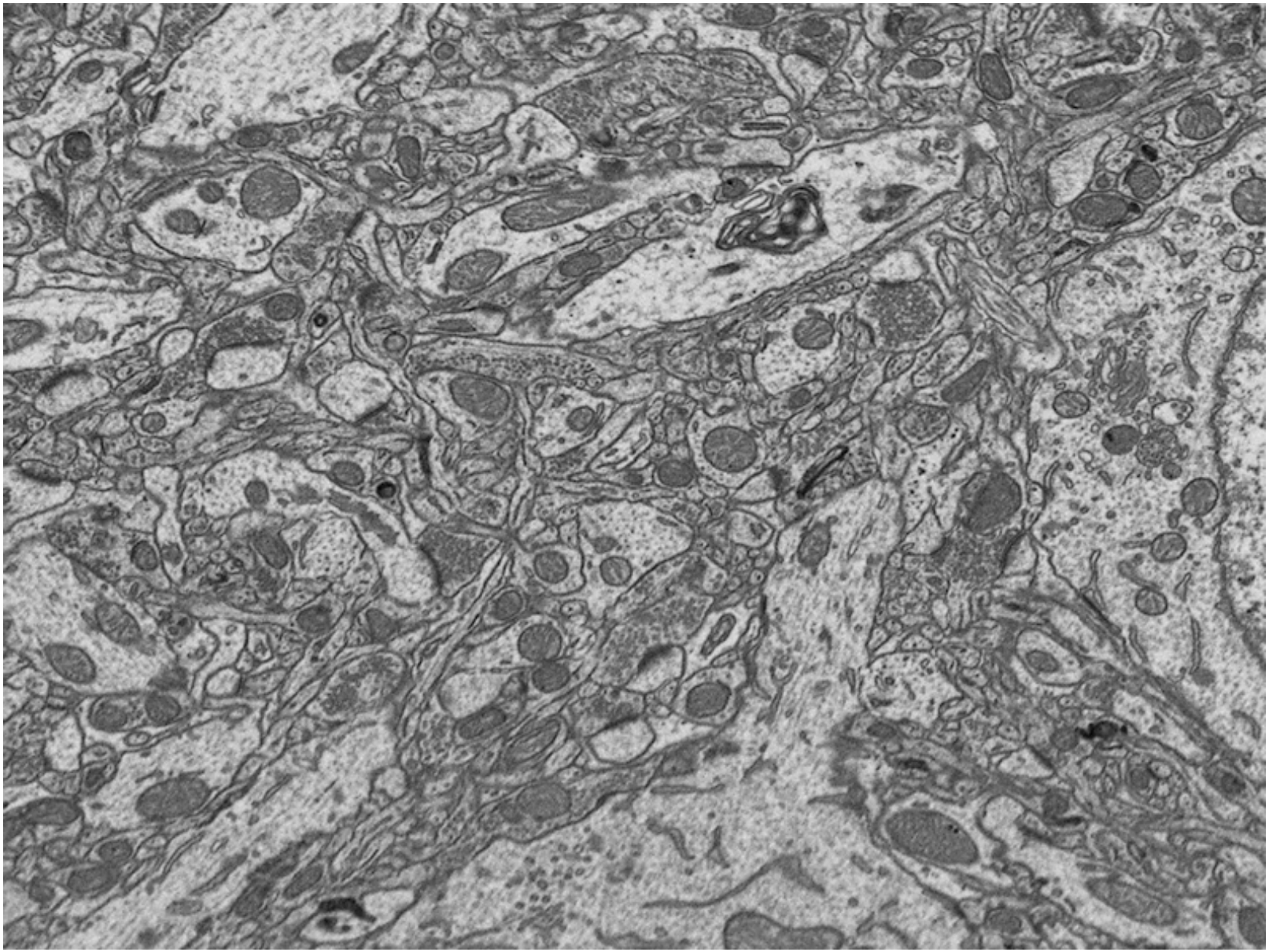
- Superpixel segmentations with centers on a 64x64, 256x256, and 1024x1024 grid.
- Can be used to describe the image in terms of a set of small regions.

Color or Black and White

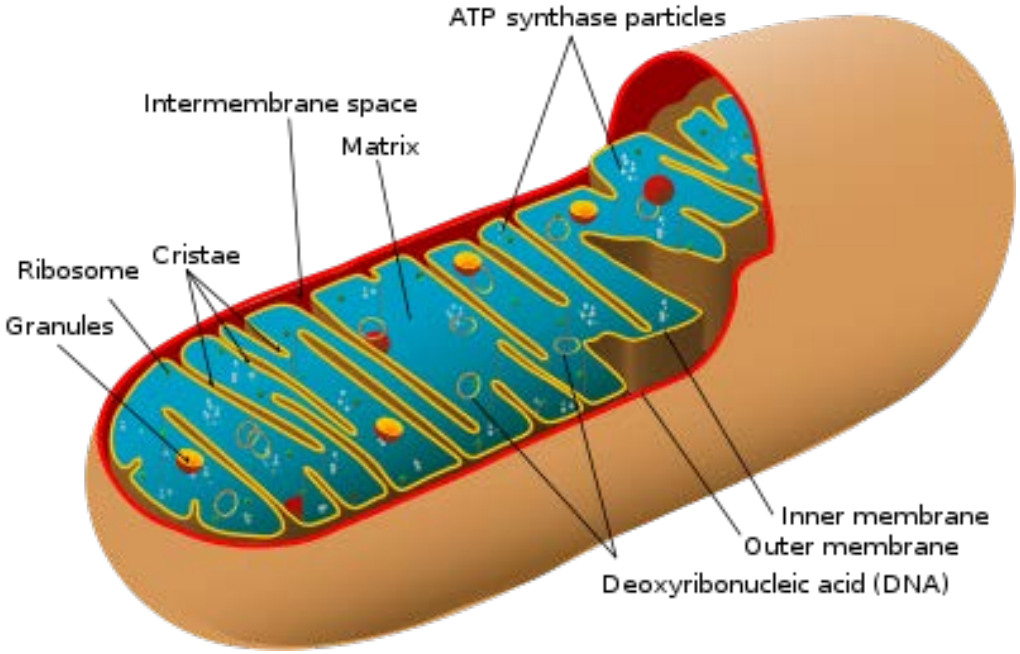


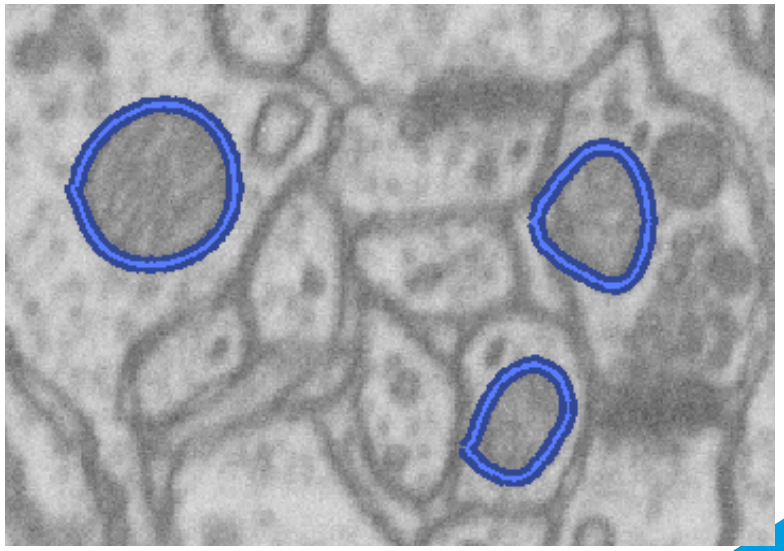
It works in both cases.

Electron Microscopy

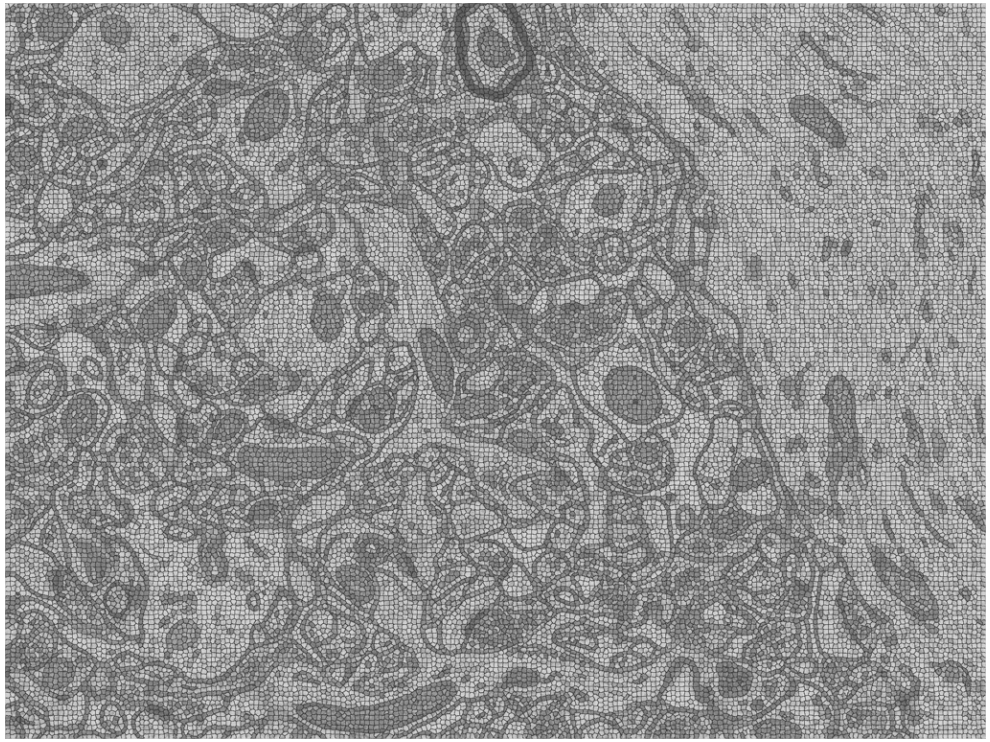


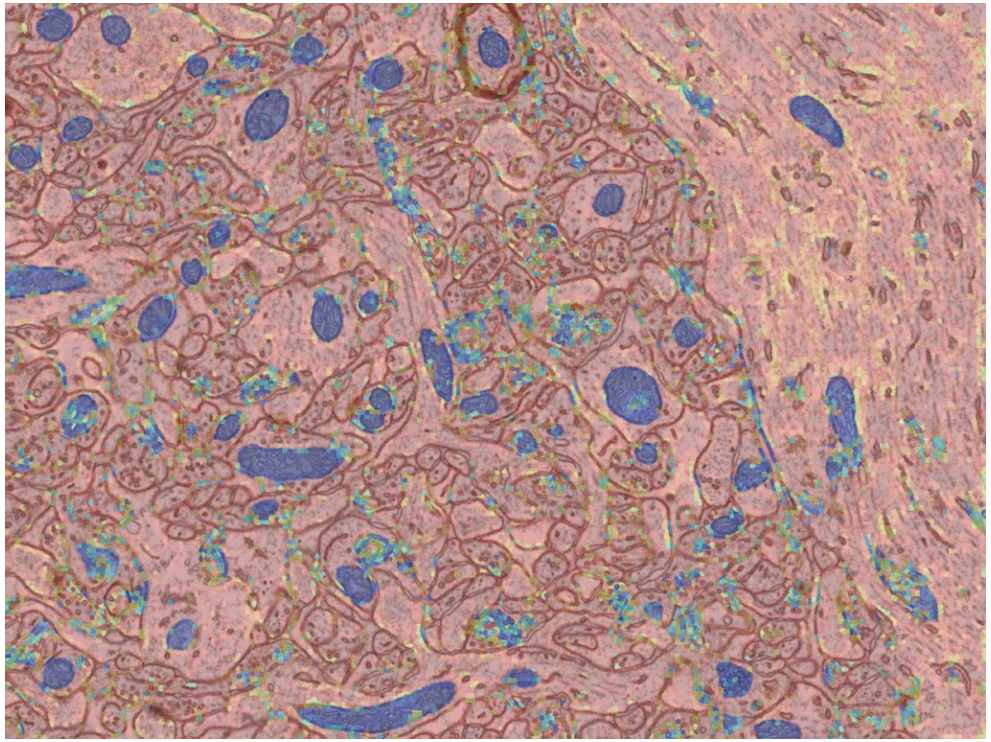
Mitochondria Segmentation





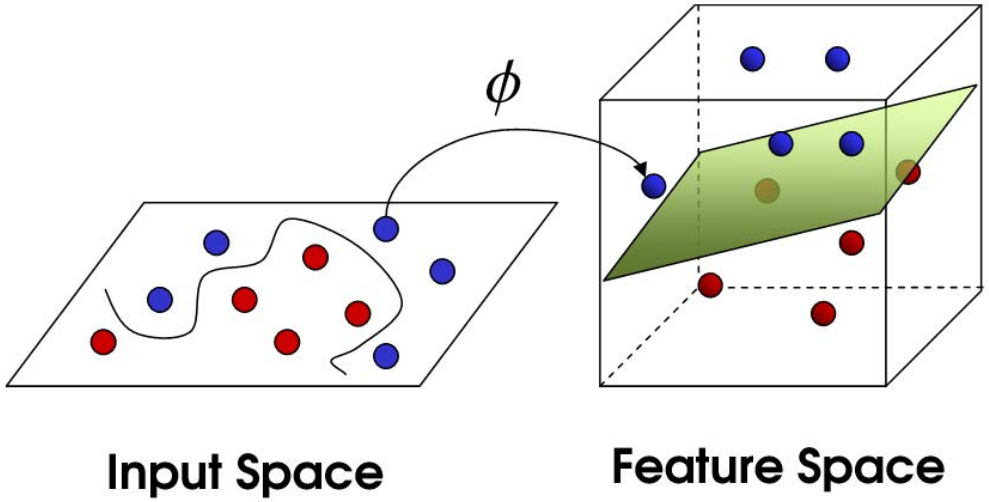
Assigning Probabilities

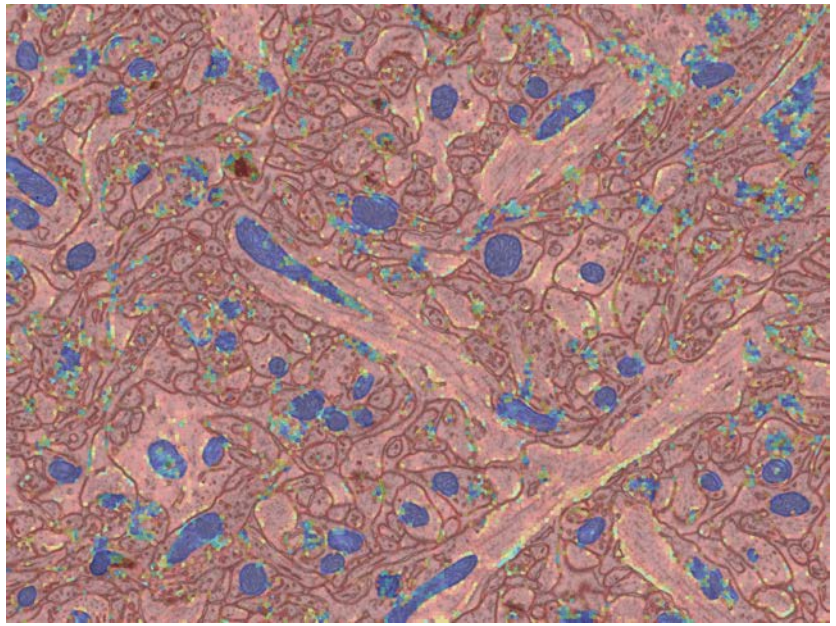




- Compute image statistics for each superpixel.
- Train a classifier to assign a probability to be within a mitochondria.
- Can be used to produce segmentations using the graph-based techniques we will describe next.

SVM Classification



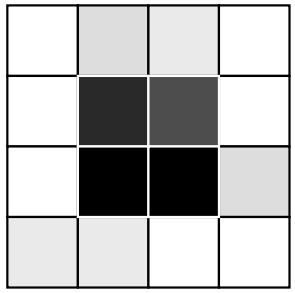


- The features incorporate the filter responses among other things.
- The probability of a superpixel belonging to a mitochondria is estimated from the SVM output.

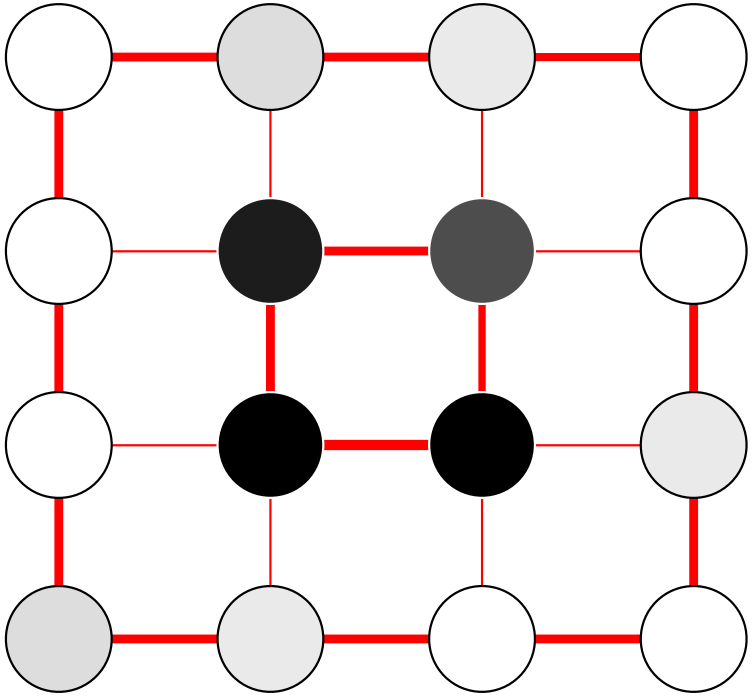
But how we can turn this into a segmentation?

Images as Graphs

An image I(x,y) is equivalent to a graph G(V,E)



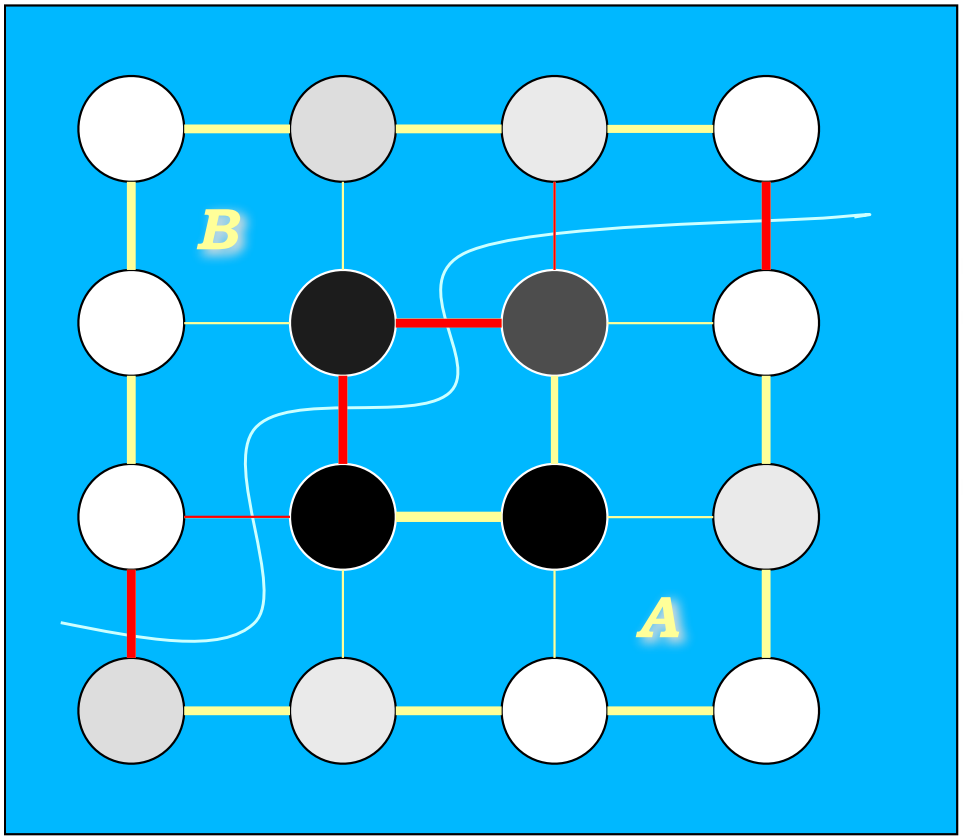
Original image I(x,y)



Graph G(V,E)

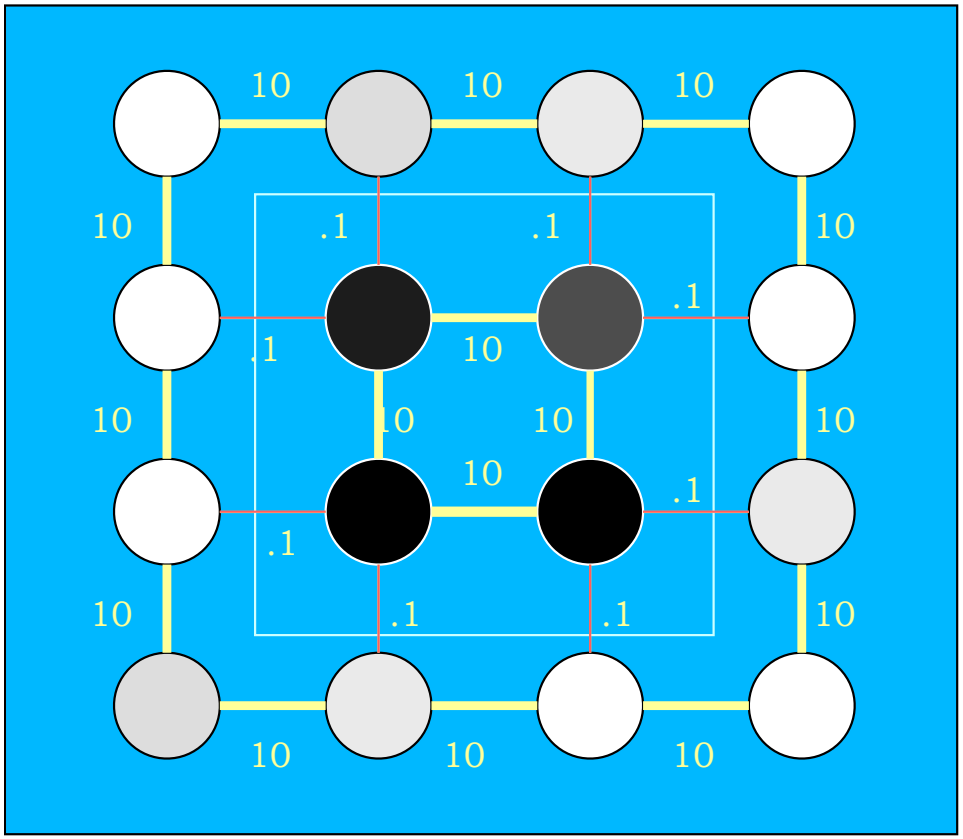
- V is a set of vertices or nodes that represent individual pixels.
- E is a set of edges linking neighboring nodes together. The weight or strength of the edge is proportional to the similarity between the vertices it joins together.

Graph-Cut



A cut through a graph is defined as the total weight of the links that must be removed to divide it into two separate components.

Min-Cut

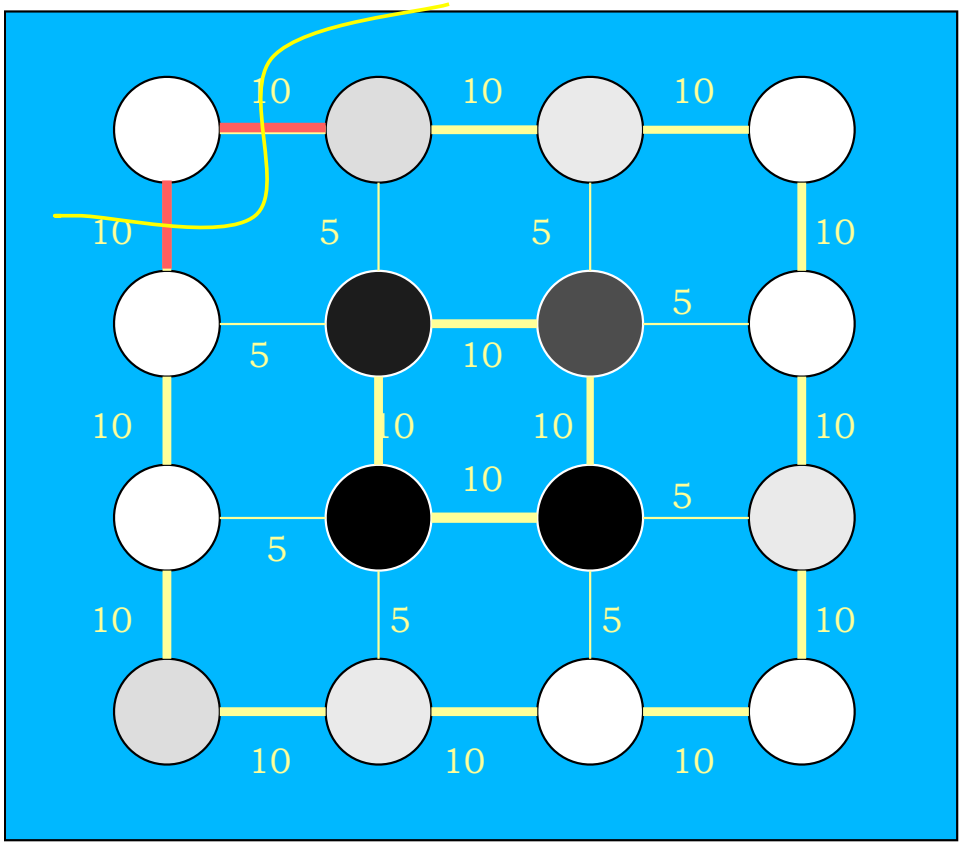


- Find the cut through the graph that has the overall minimum weight, which can be done effectively.

- It should be the subset of edges of least weight that can be removed to partition the graph.

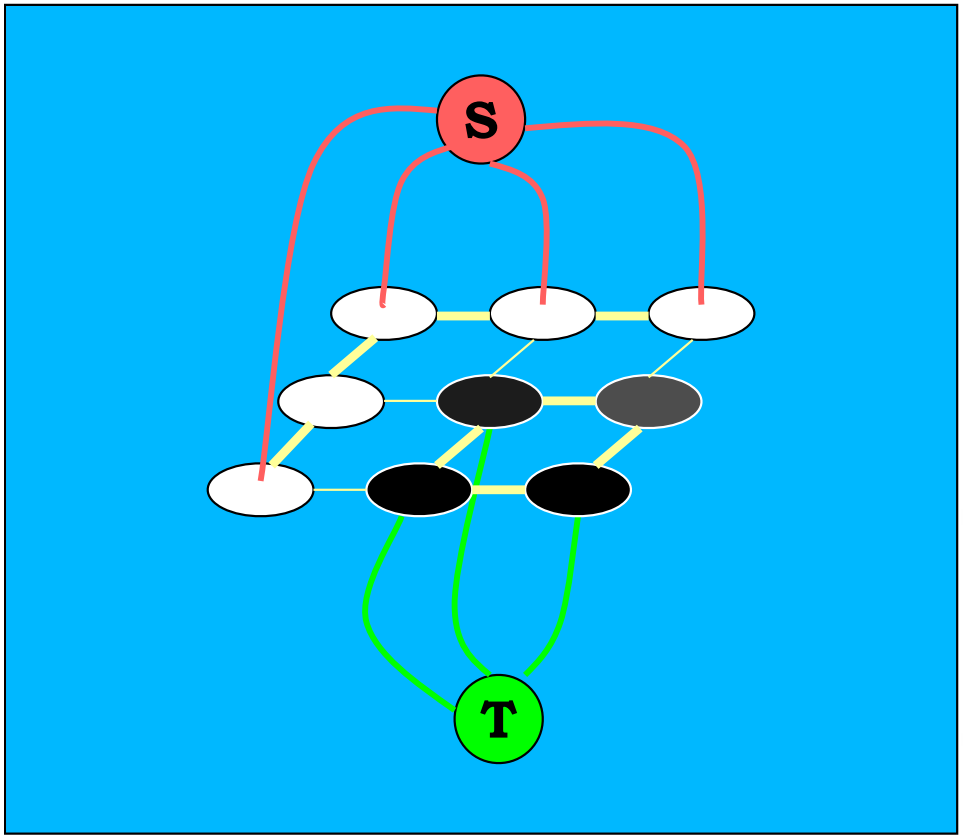
- Since weight encodes similarity, this should be equivalent to partitioning the graph along the boundary of least similarity.

Trivial Cut



- Has a preference for shortcuts, which may sometime result in trivial solutions.
- Must be constrained to avoid them.

ST Min-Cut



- Introduce two special nodes called source (S) and sink (T)

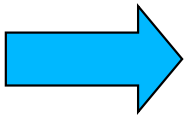
- S and T are linked to some image nodes by links of very large weight that will never be selected in a cut.

- Find the minimum cut separating the source from the sink.

--> The problem becomes deciding how to connect S and T to the image nodes.

Interactive ST Min-Cut

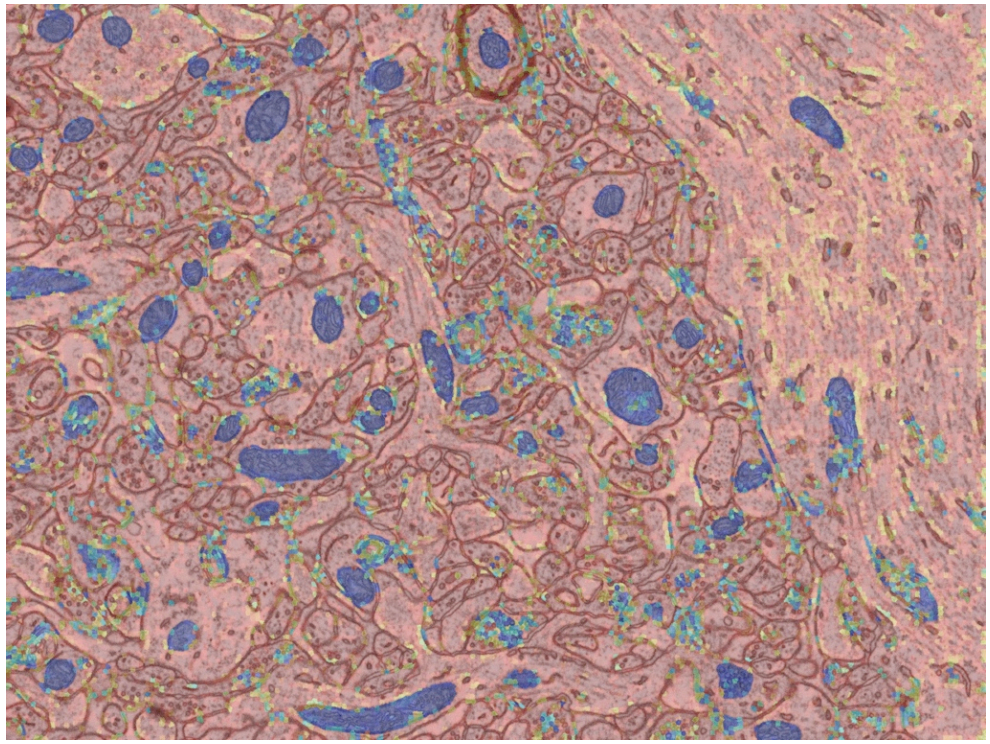
User-selected pixels connected to S (red) and pixels connected to T (cyan)

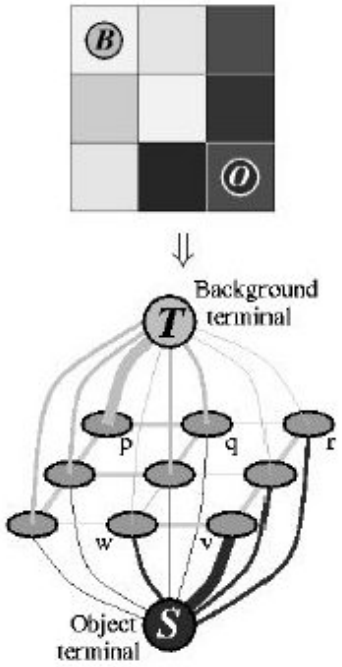


Minimum S-T cut

--> If we have a good initial 'guess' to tell us how to link the source and sink to the image, we will get an optimal segmentation.

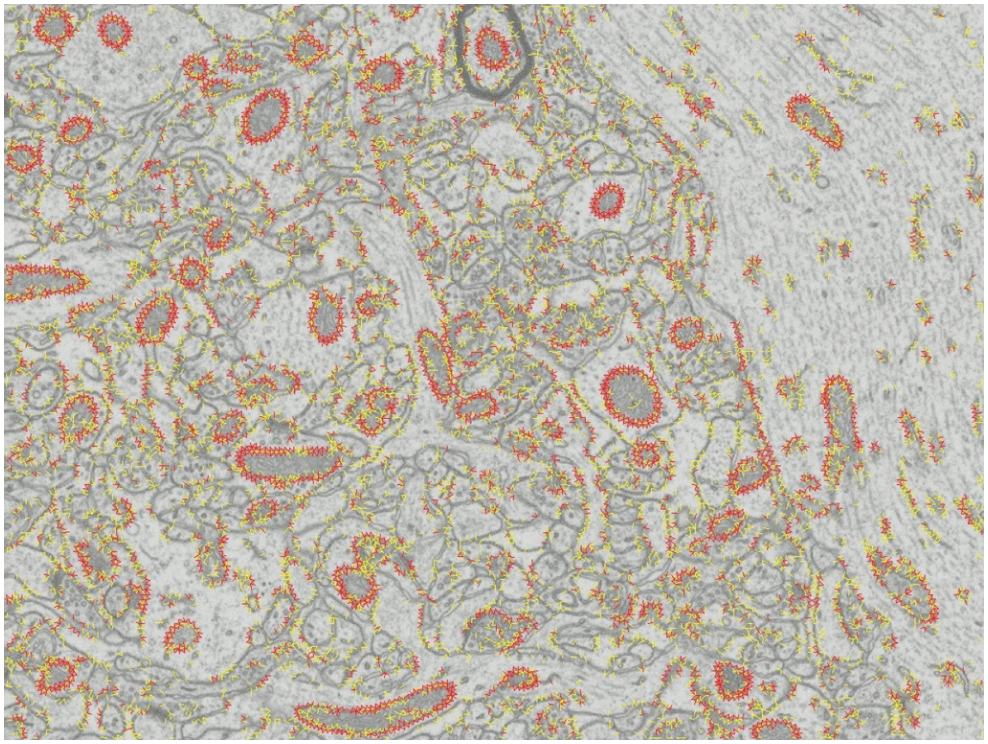
Back To Mitochondria

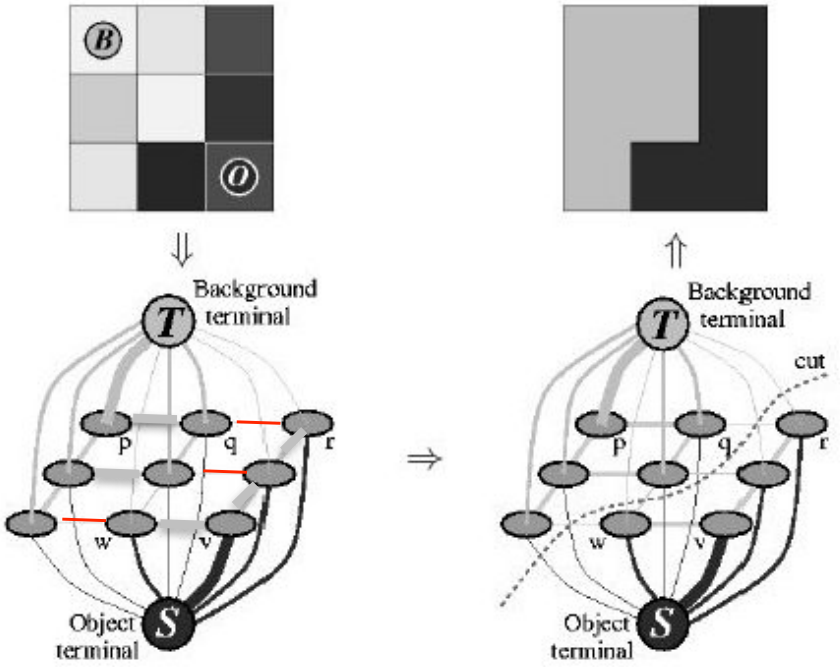




- A high probability of being a mitochondria can be represented by a strong edge connecting a supervoxel to the source and a weak one to the sink.
- And conversely for a low probability.

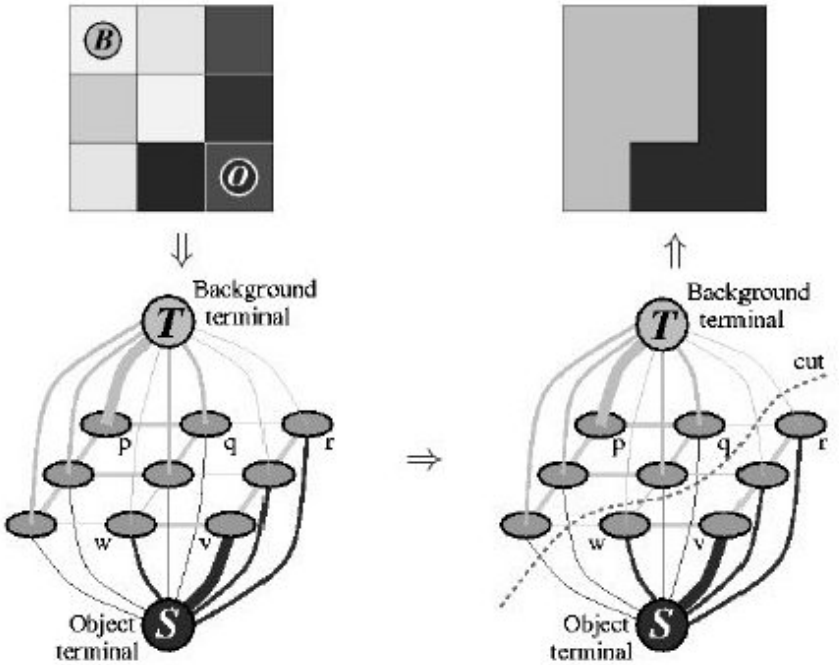
Back To Mitochondria





Another classifier can be trained to assign a high-weight to edges connecting supervoxels belonging to the same class and a low one to others.

Minimizing a Loss Function

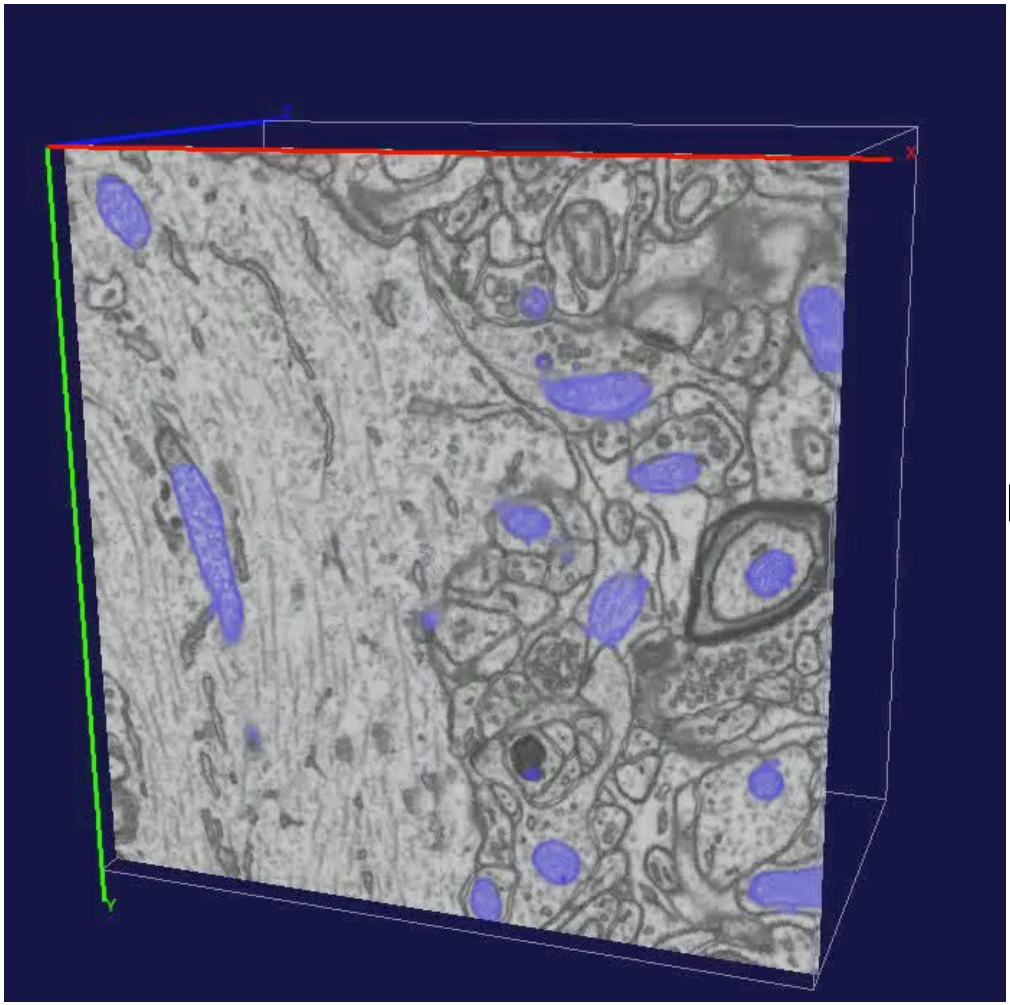


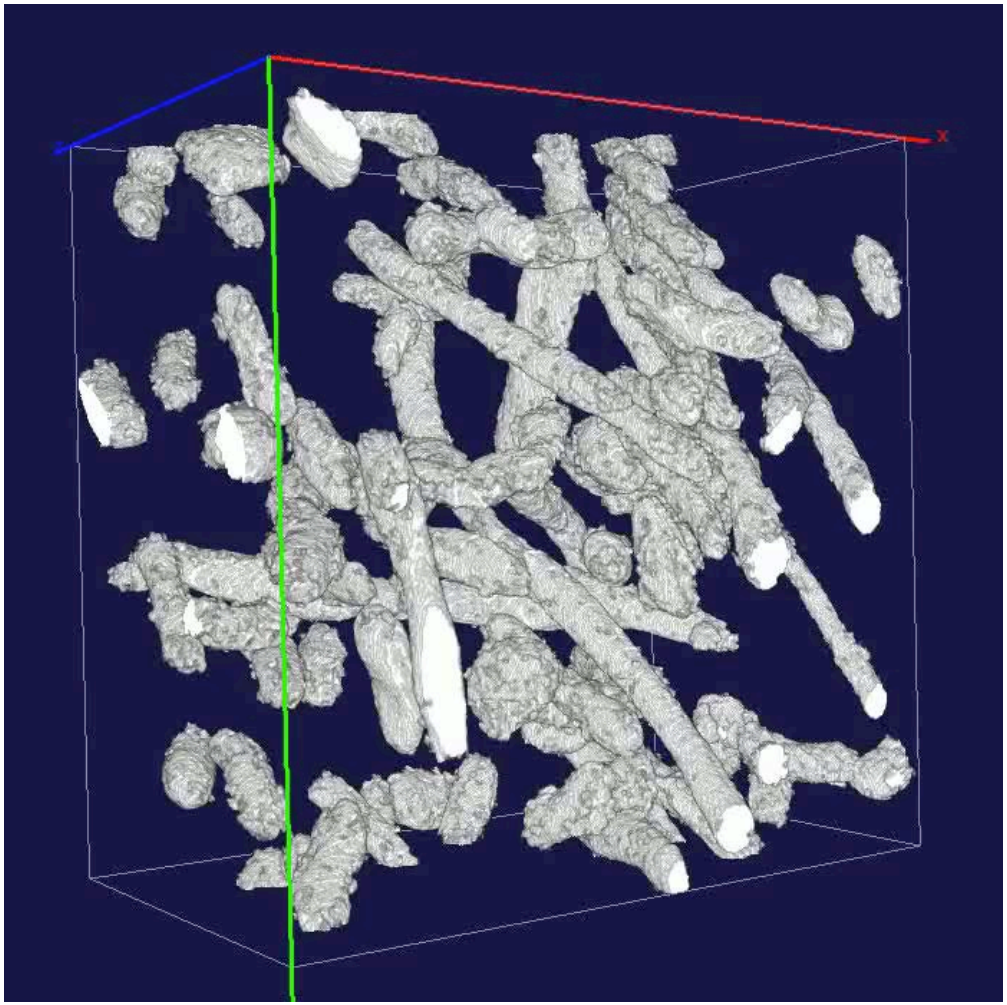
In probabilistic terms, this amounts to minimizing

$$E(y|x,\lambda) = \sum_i \underbrace{\psi(y_i|x_i)}_{\text{unary term}} + \lambda \sum_{(i,j)\in\mathcal{E}} \underbrace{\phi(y_i,y_j|x_i,x_j)}_{\text{pairwise term}} \ ,$$

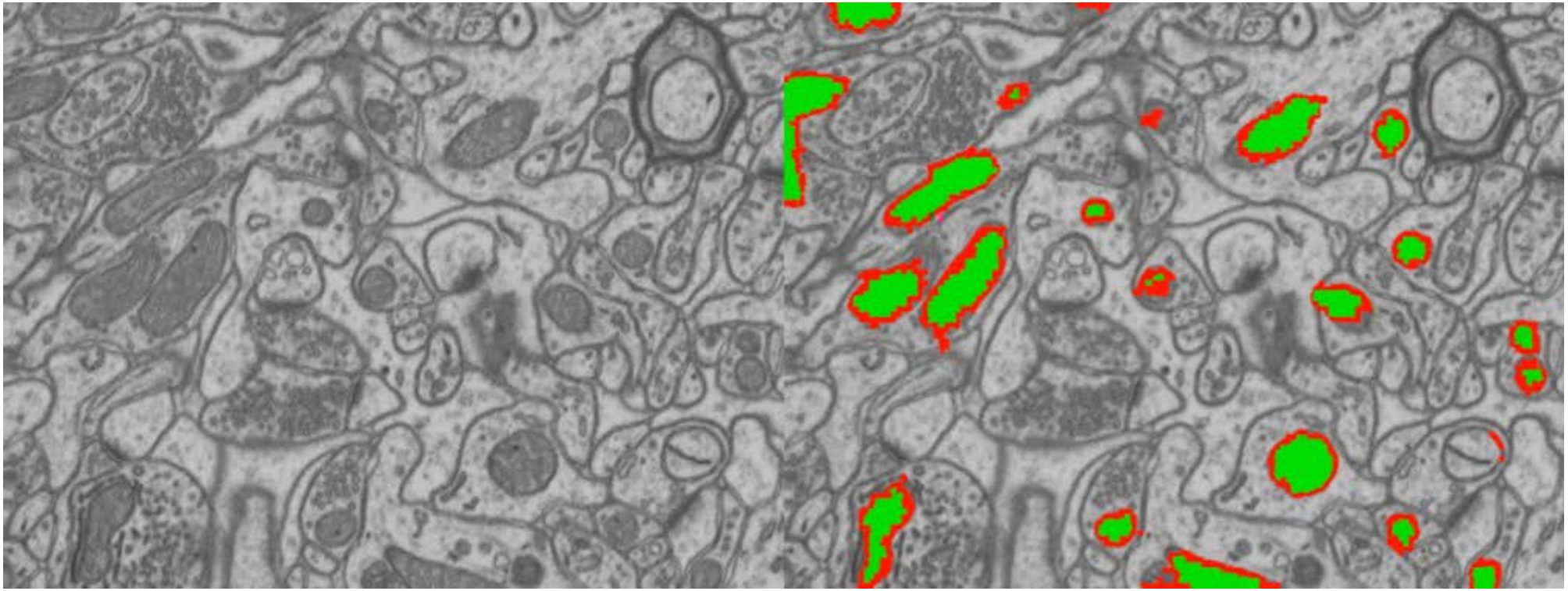
with respect to the set of labels $y = [y_1,\ldots,y_n]$ given the set of supervoxels $[x_1,\ldots,x_n]$, where E is negative log-likelihood of the labels being correct.

3D Mitochondria





Modeling Membranes



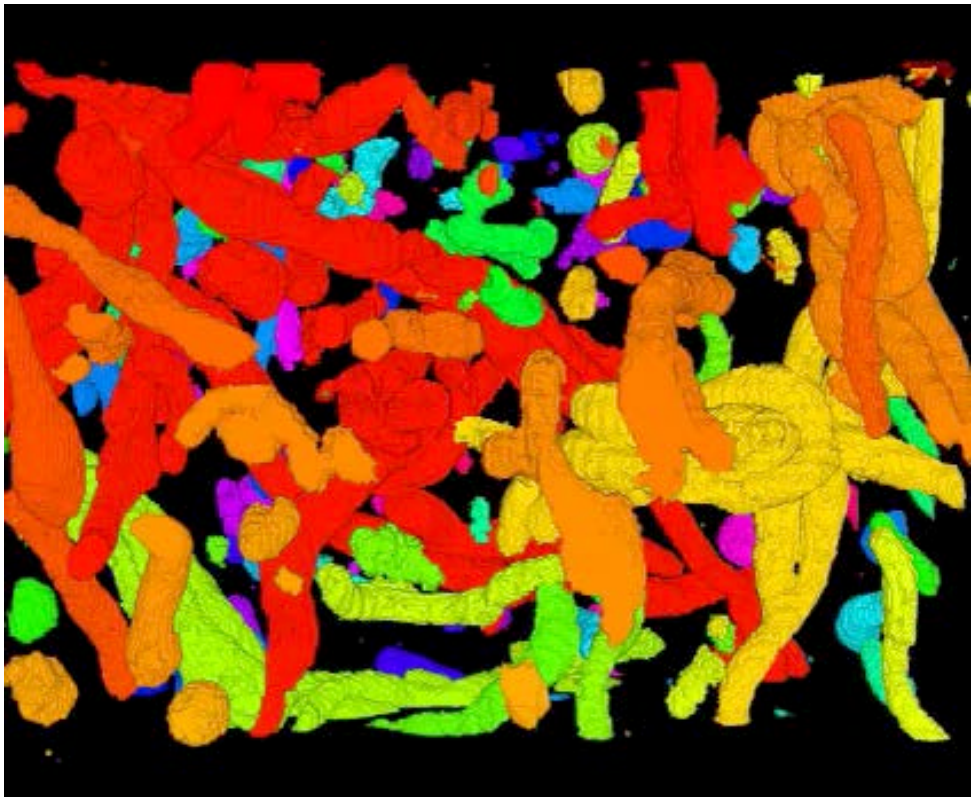
Here we use three classes instead of two:

- Inside
- Membrane
- Everything else

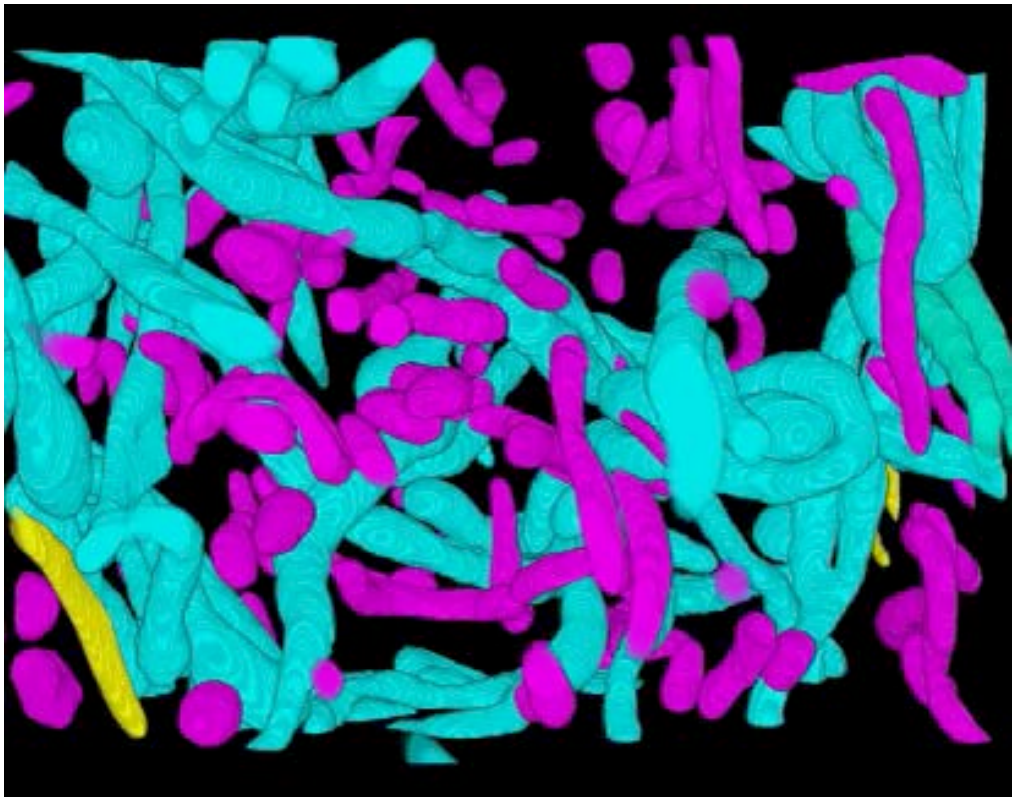
—>Because the inside is fully enclosed by the membranes, we can still find a global optimum.

Speeding up the Analysis Process

3.21 µm × 3.21 µm × 1.08 µm: 53 mitochondria



Automated result



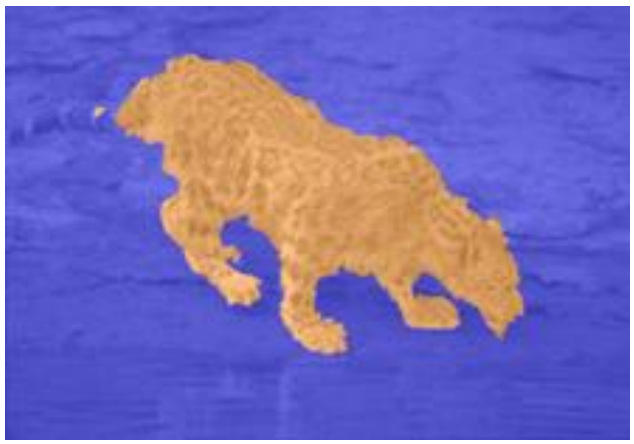
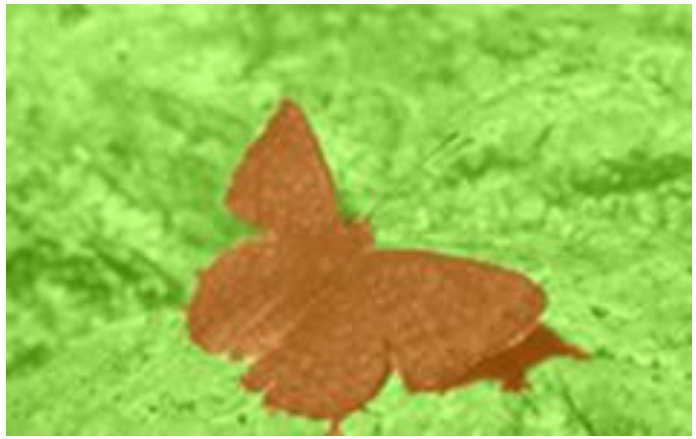
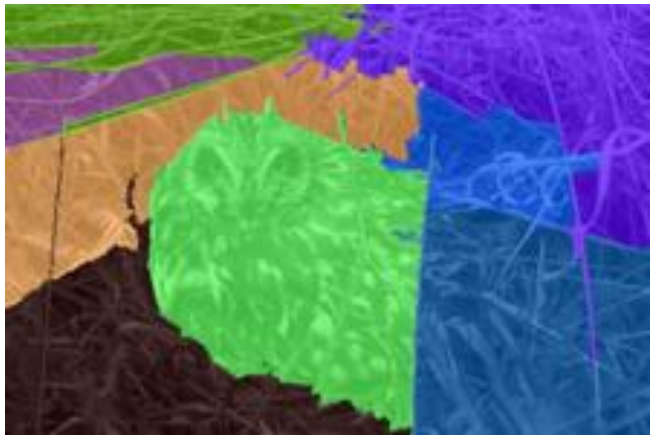
Interactively cleaned-up result

- By hand: 6 hours.
- Semi-automatically: 1.5 hours

—> Substantial time saving for the neuroscientists.

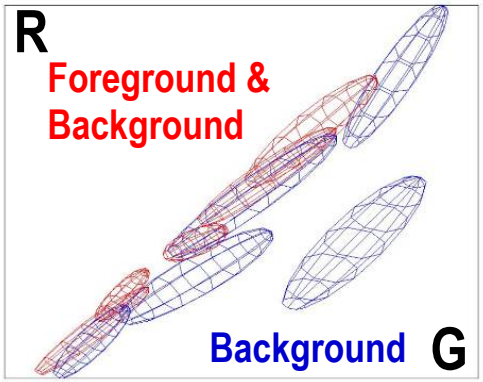
Graph-Cut on Ordinary Images



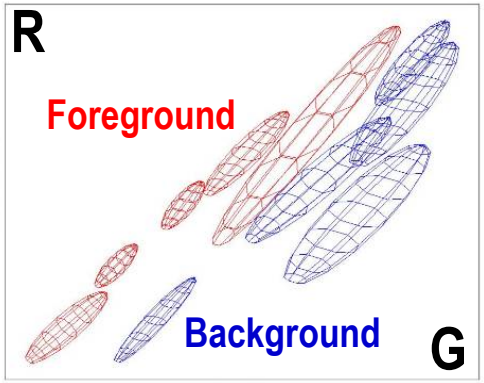





Galun et al., ICCV'03

Interactive Foreground Extraction

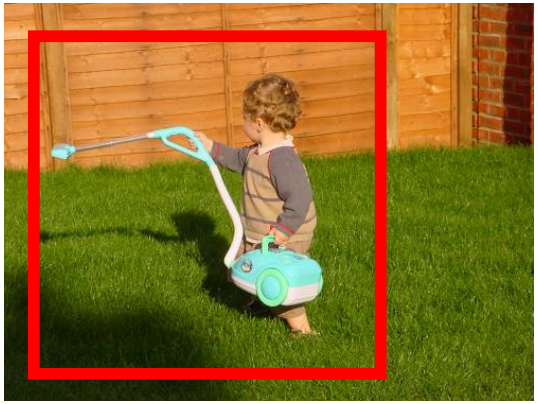


Iterated
graph cut



- K-means to learn color distributions
- Graph cuts to infer the segmentation

Relatively Easy Examples



More Difficult Examples

Camouflage & Low Contrast

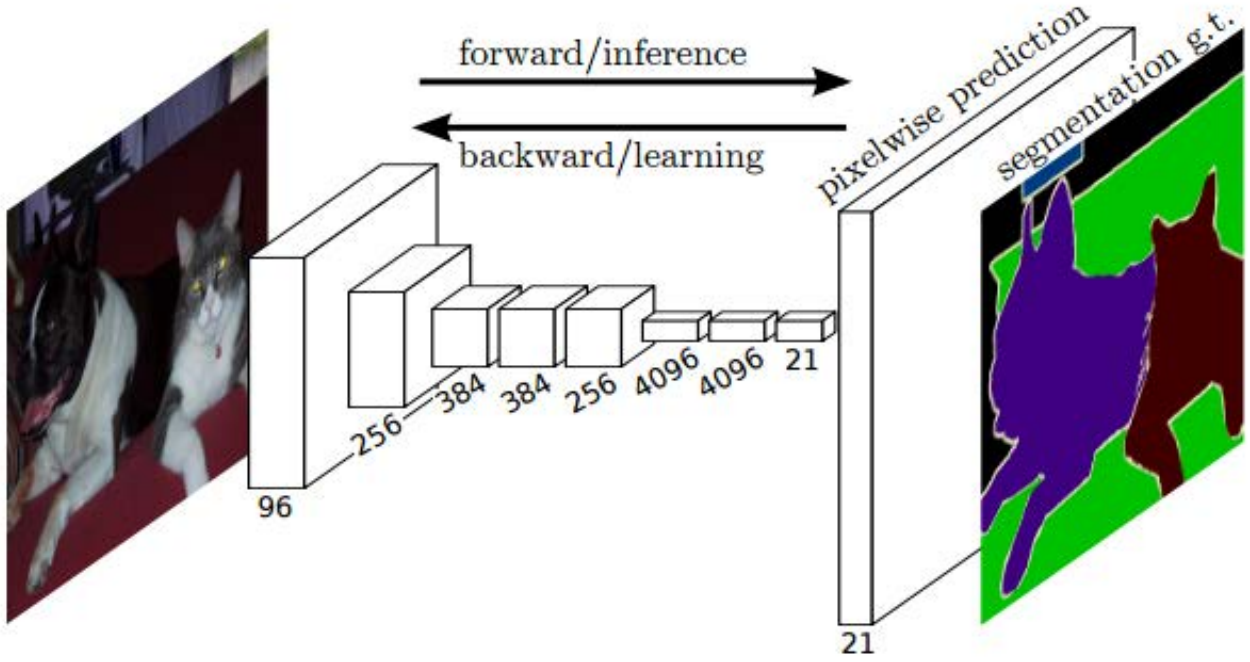
Fine structure

No telepathy

Initial Rectangle

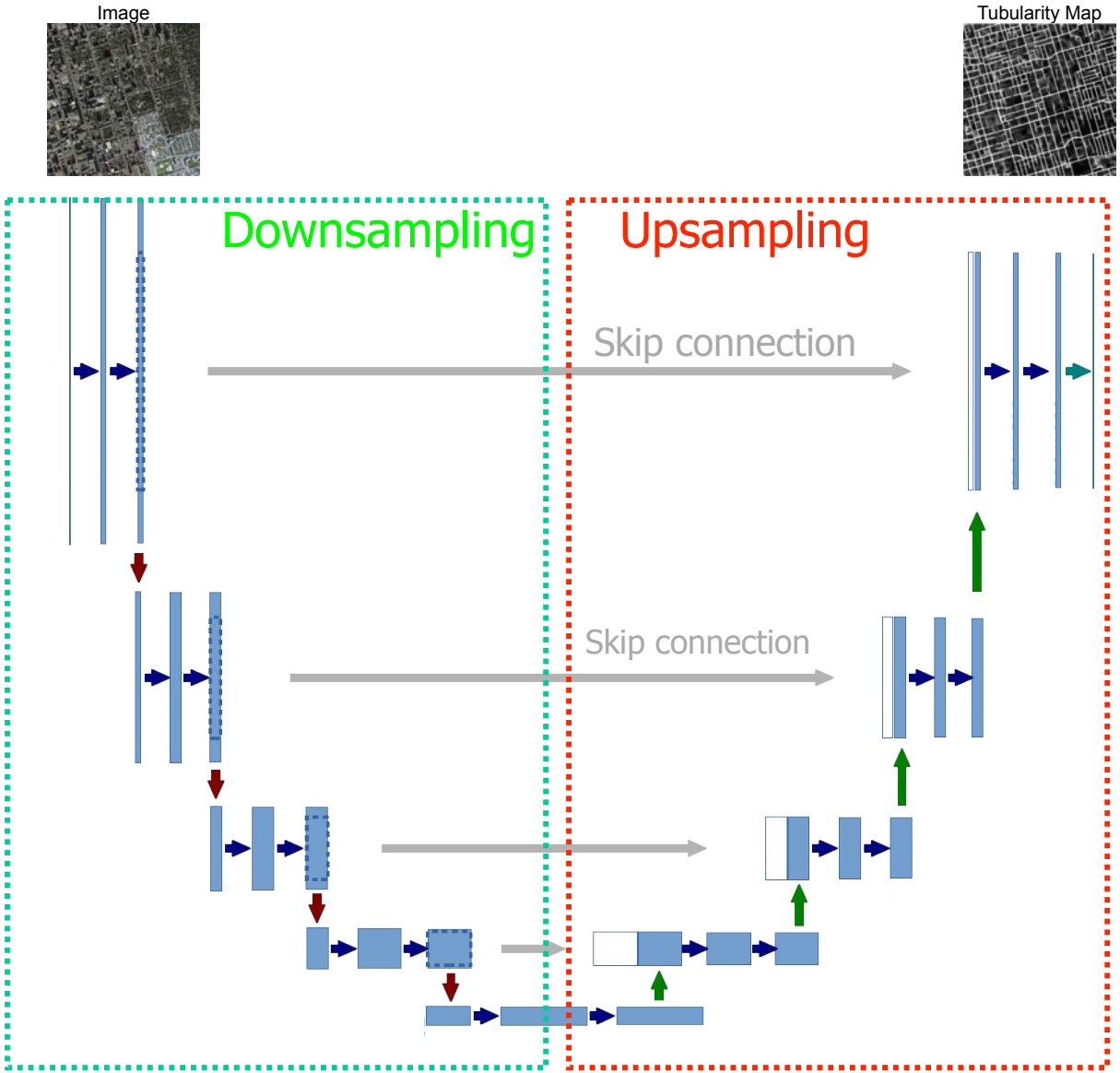
Initial Result

Convolutional Neural Nets



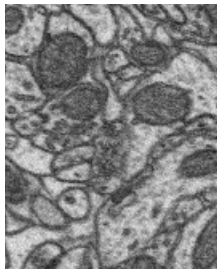
- Connect input layer to output one made of segmentation labels.
- Need layers that both downscale and upscale.
- Connect the lower layers directly to the upper ones.

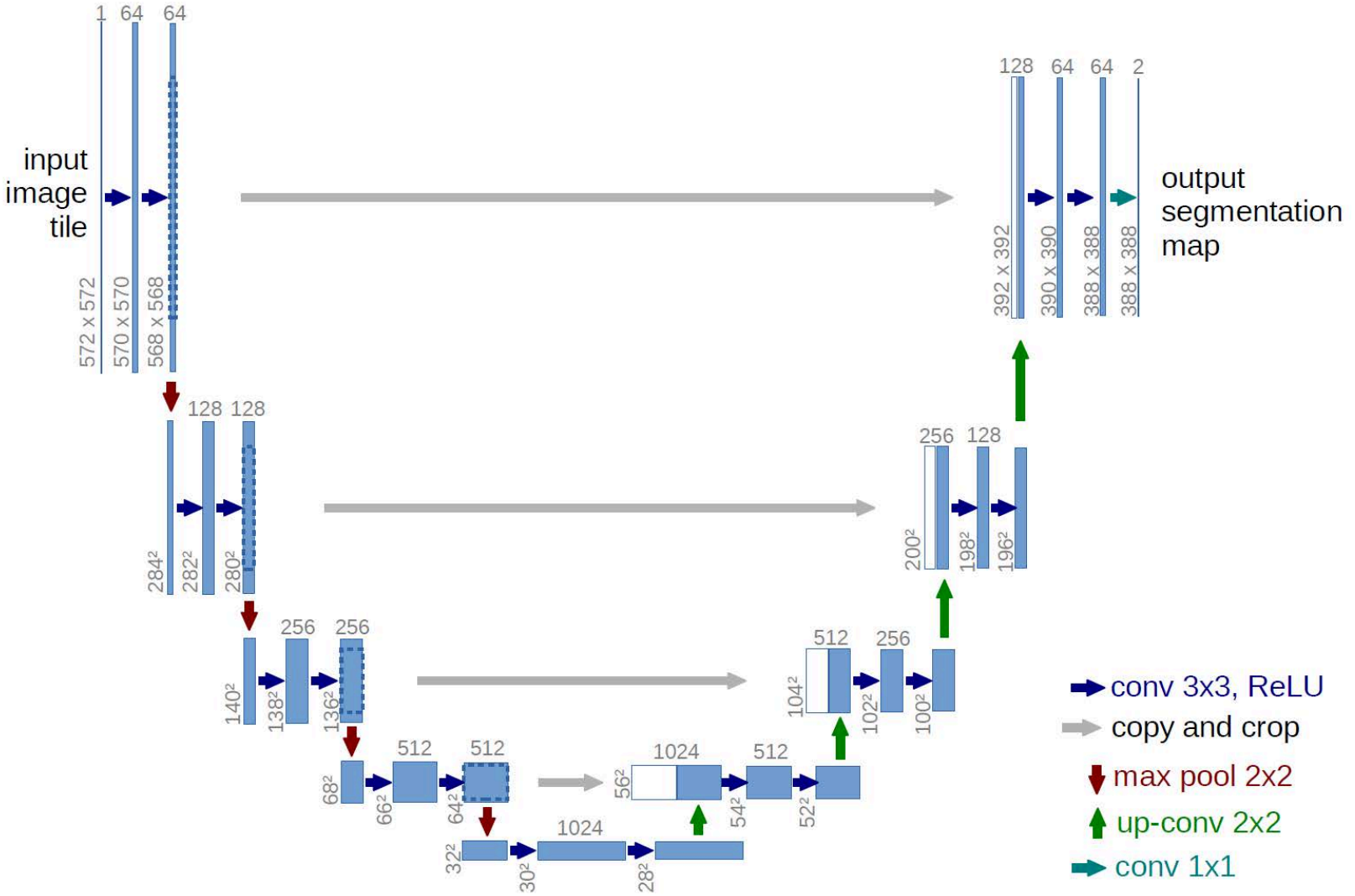
Reminder: U-Net for Delineation



—> Train a U-Net to output a tubularity map.

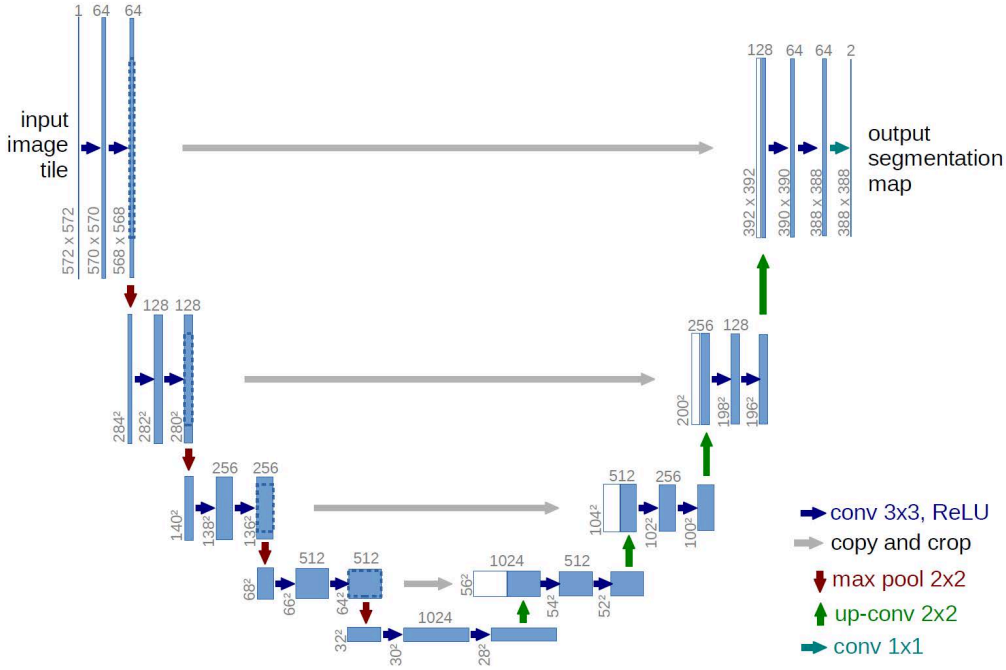
U-Net for Segmentation



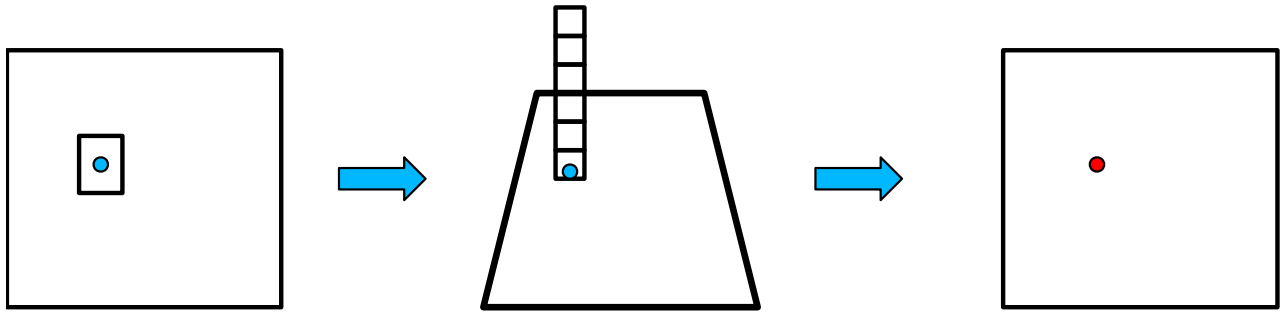


- Same architecture (in more details).
- Train it to produce a segmentation mask instead of a delineation mask.

Potential Interpretation

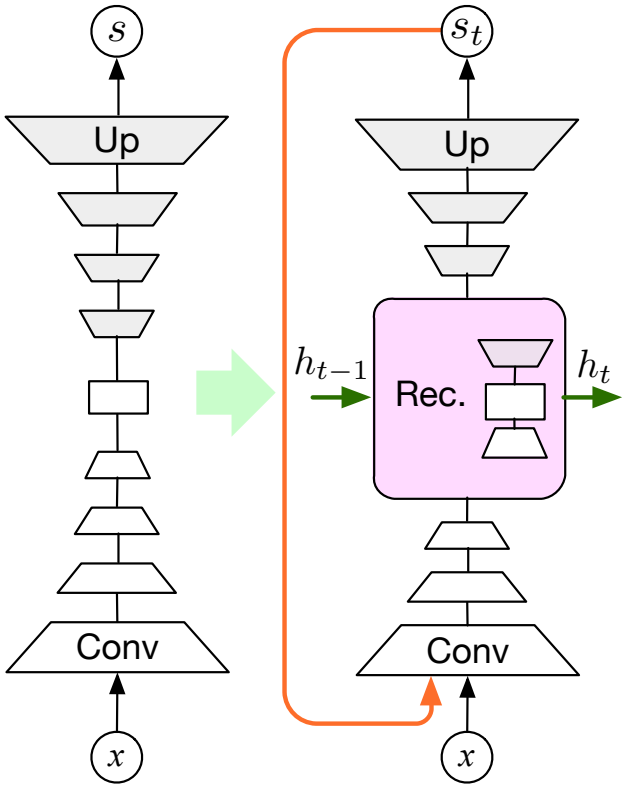


- A key role of the ConvNet is to generate for every output pixel a feature vector containing the output of all the intermediate layers.
- A classifier then assigns a label to the individual pixels.

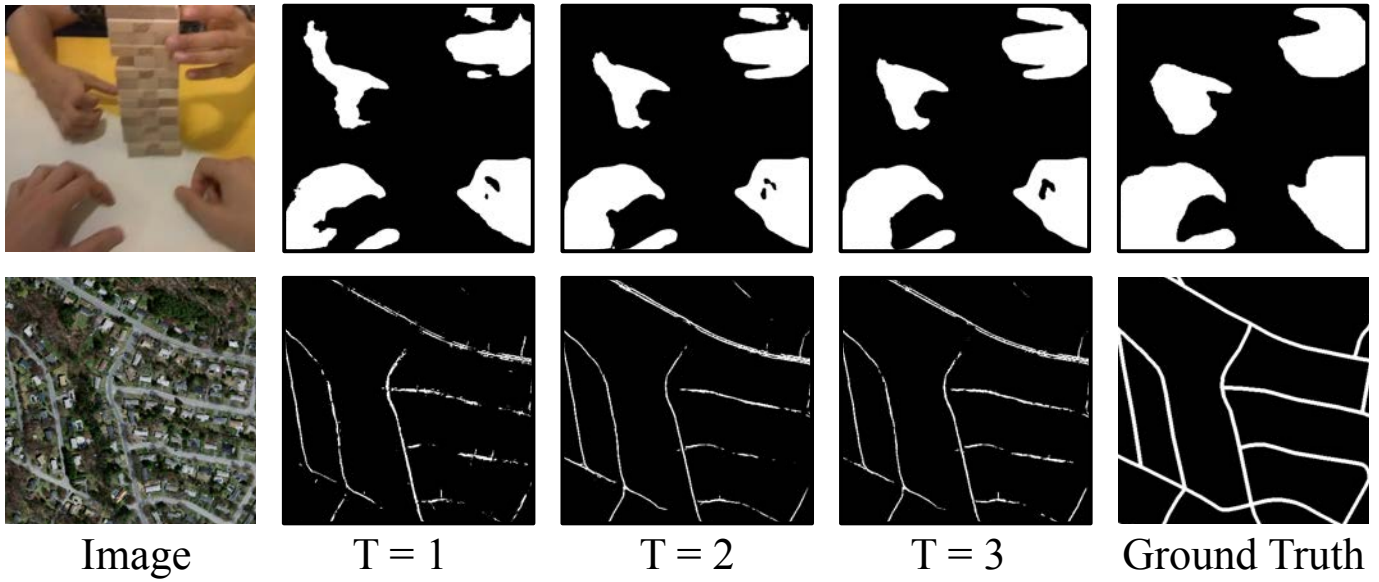


In the Street

Recursive Segmentation

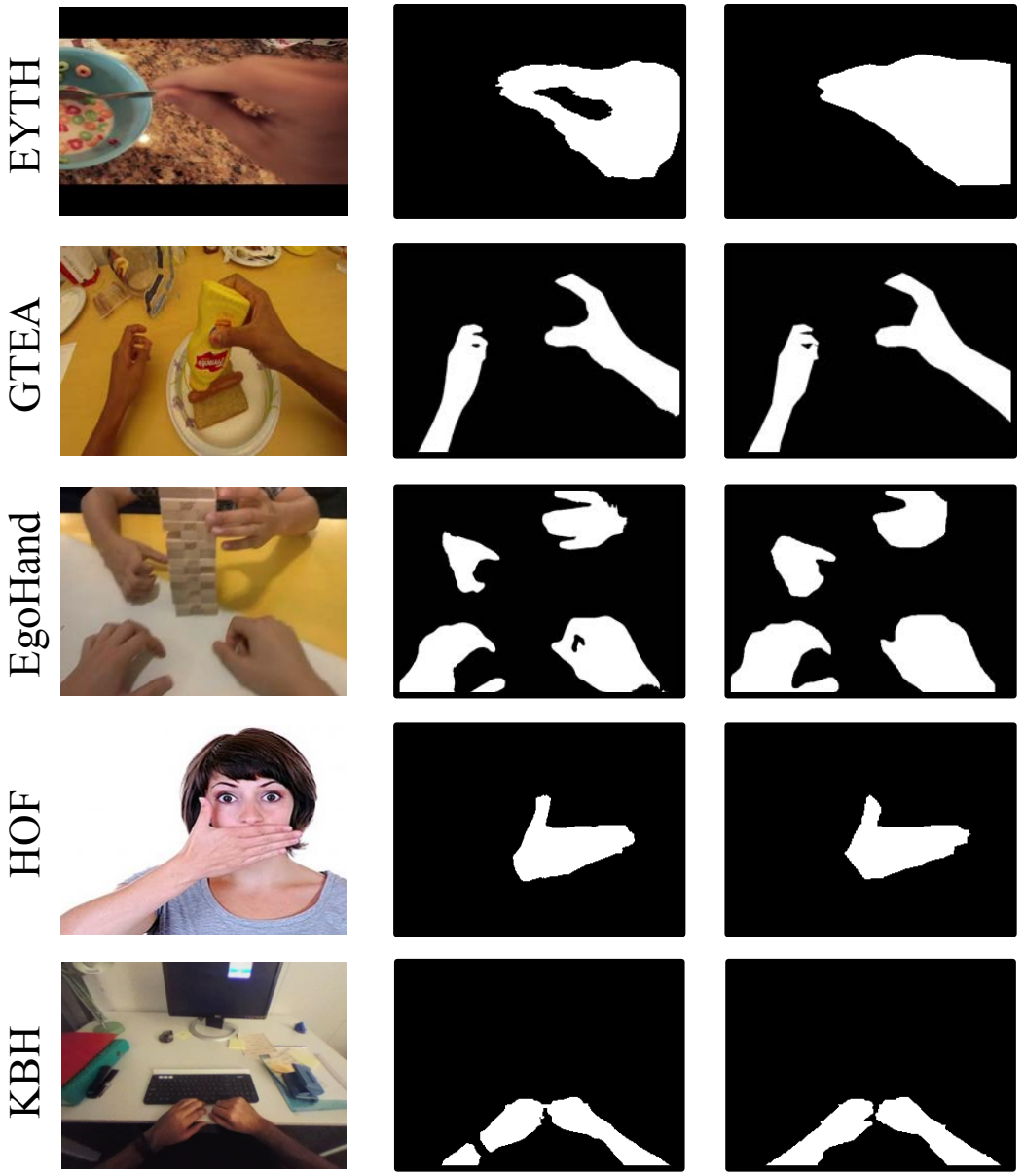


U-Net Rec. U-Net



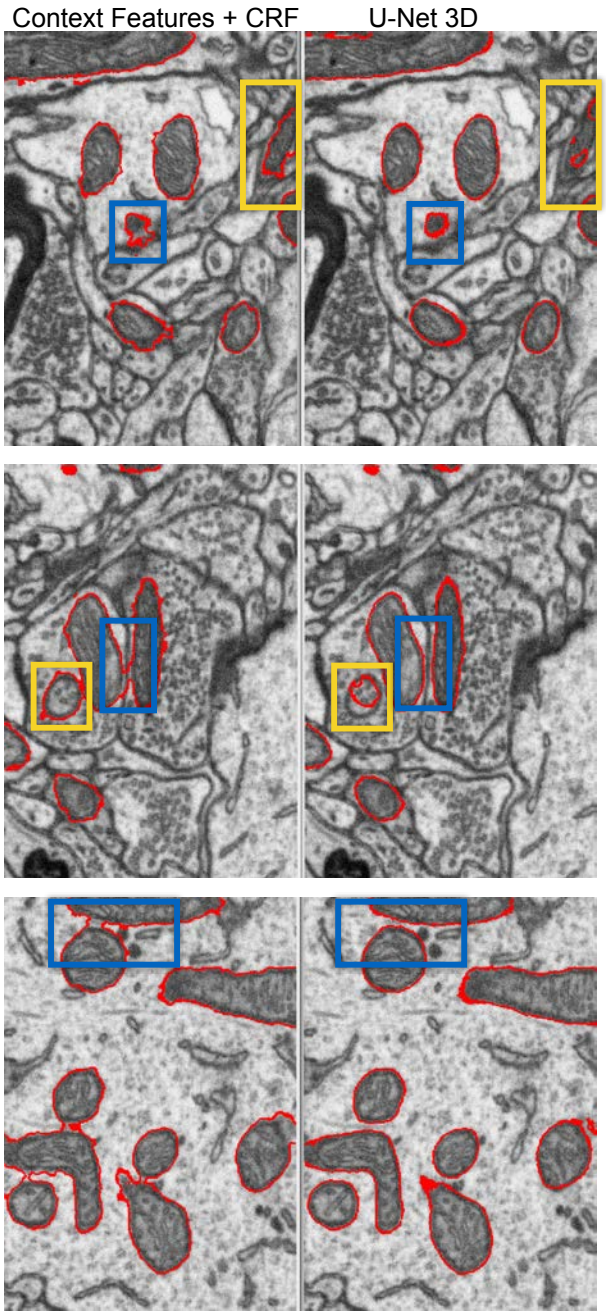
As for delineation, feeding the output back into the network is an effective way to take context into account.

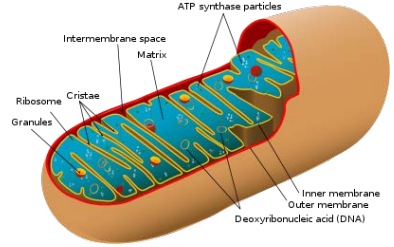
Recursive Hand Segmentation



Wang et al., ICCV'19

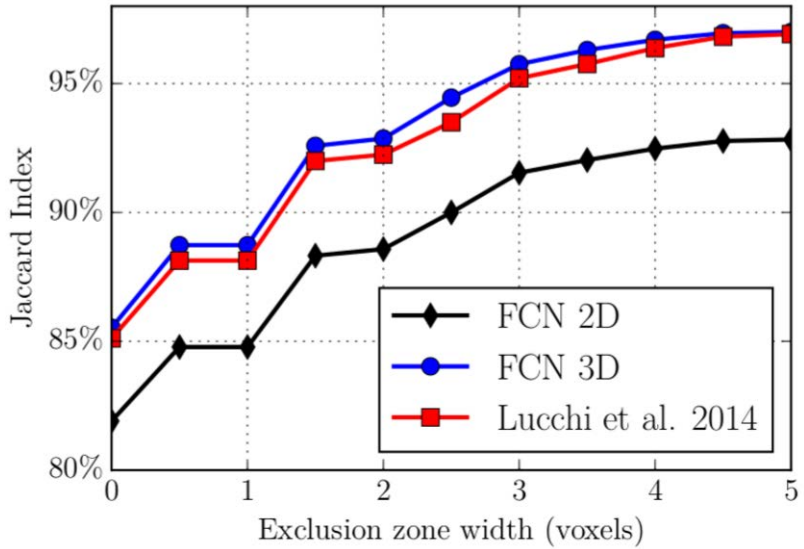
Mitochondria





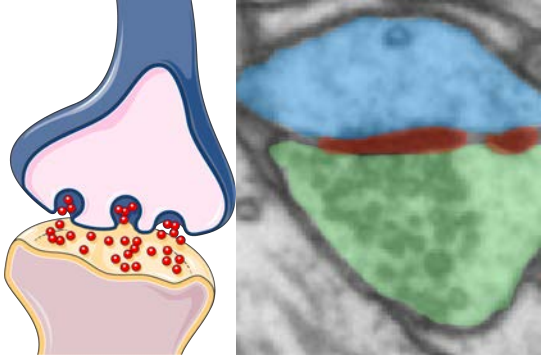
Striatum Mitochondria

Method	Jaccard Index
Context F. + CRF	84.6%
U-Net 2D	82.4%
U-Net 3D	**86.1%**



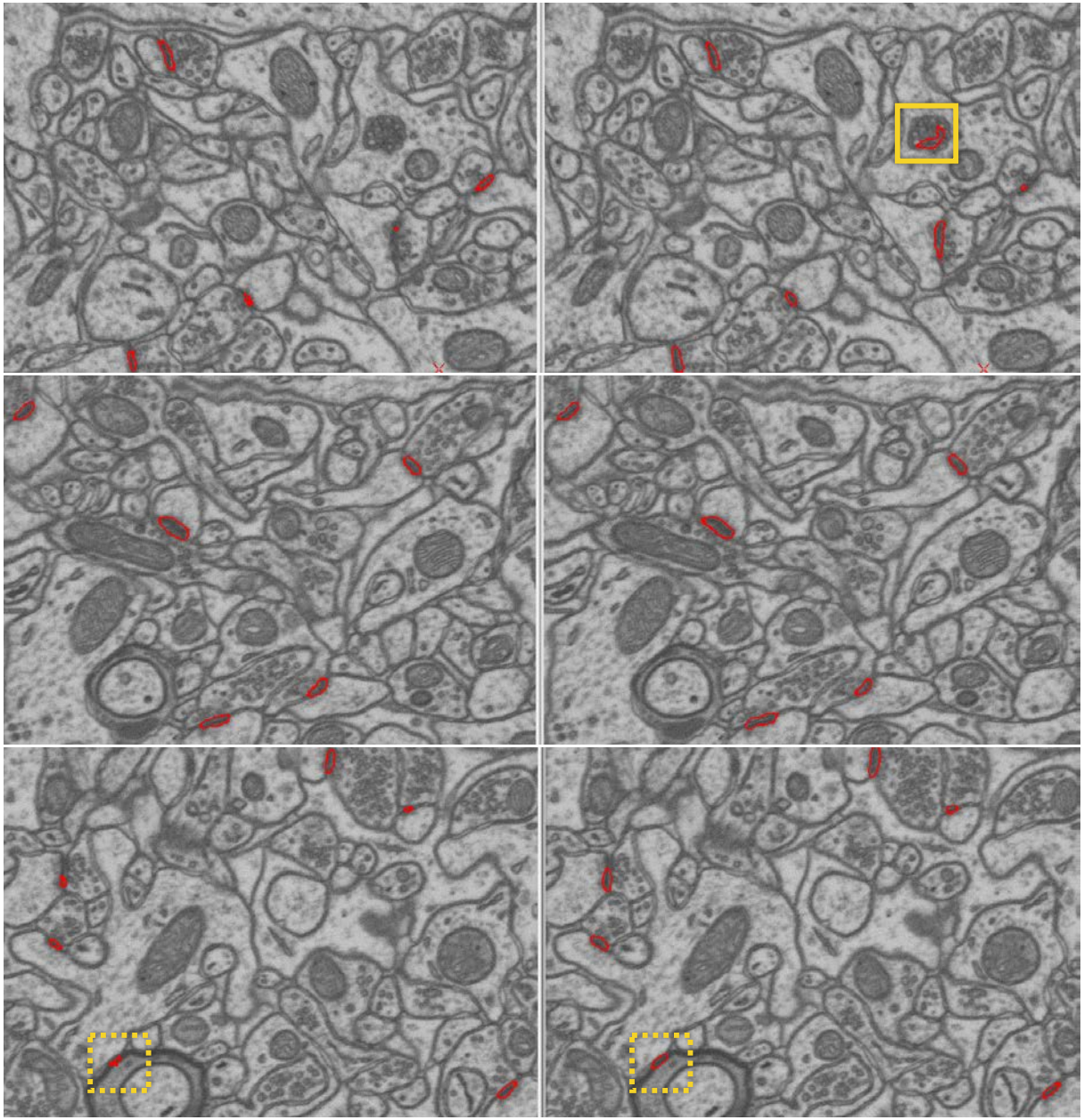
—> Using a ConvNet boosts performance but there is still work to do!

Synapses



Context Features 3D CRF

U-Net 3D

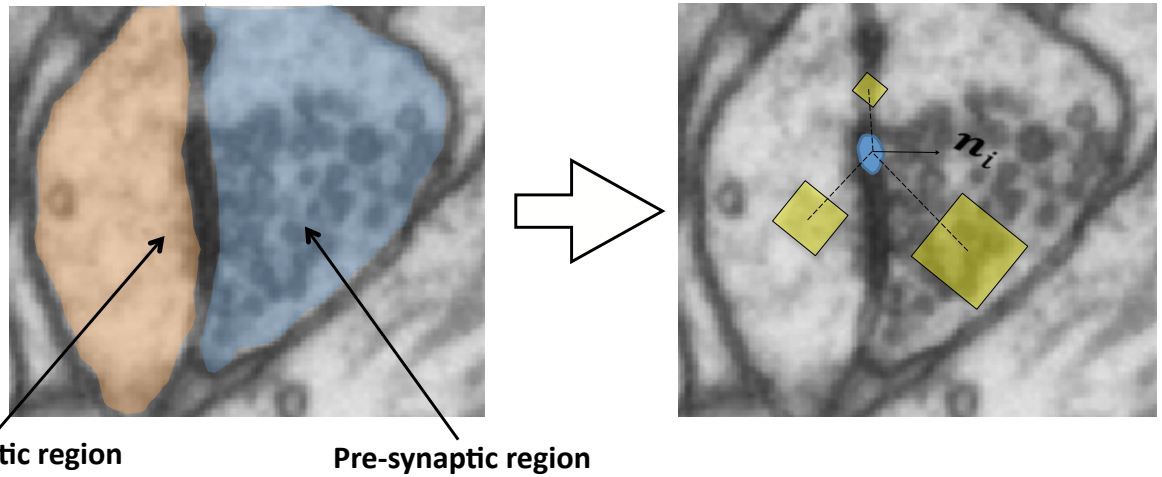


Jaccard Index	Method
66.8%	Context Features 2D
85.2%	Context Features 3D
73.5%	U-Net 2D
77.0%	U-Net 3D

?

Context-Based Features

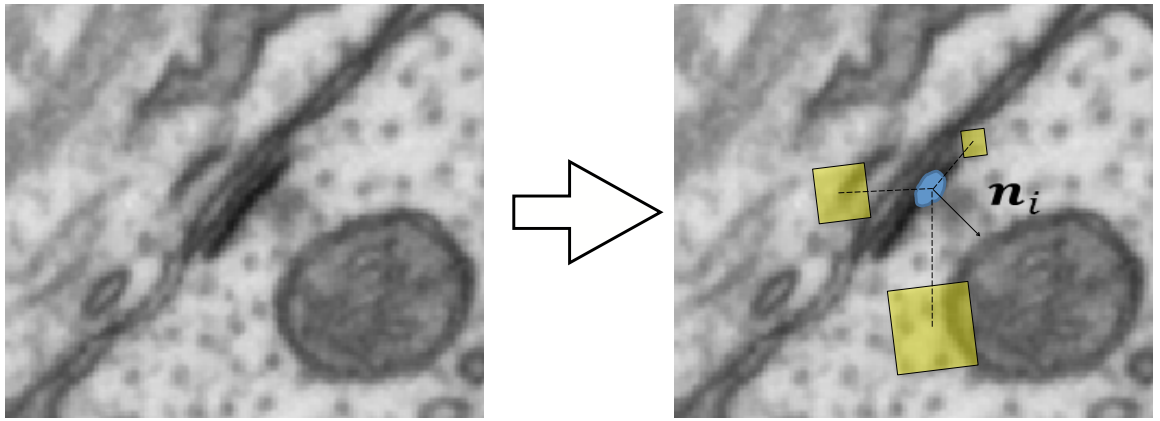
Synapse:



- Dendrite
- No vesicles

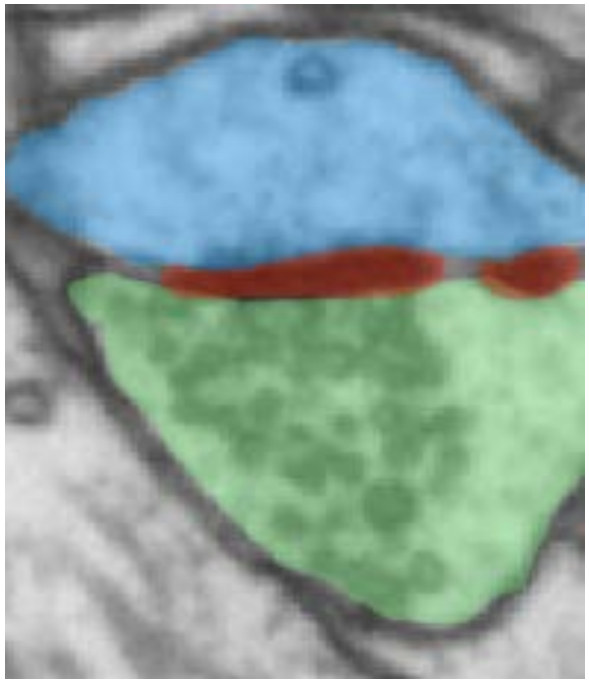
- Axon terminal
- Many vesicles

Non-Synapse:

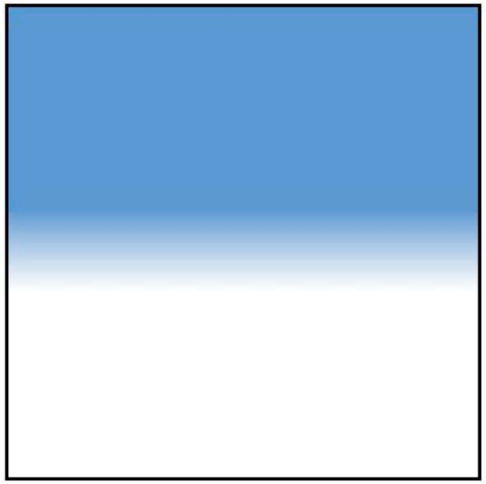


- The old style context-based features explicitly model orientation.
- How can we do that in a deep-learning framework?

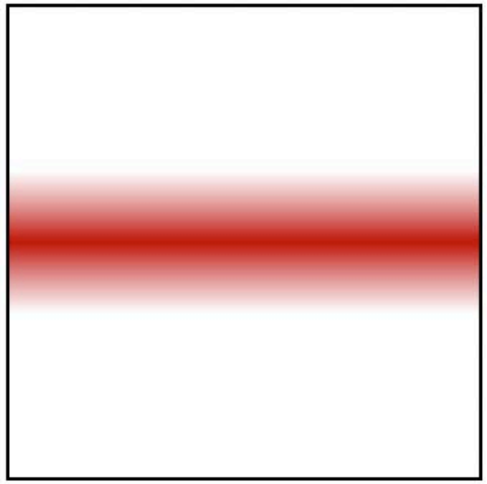
Probabilistic Atlases



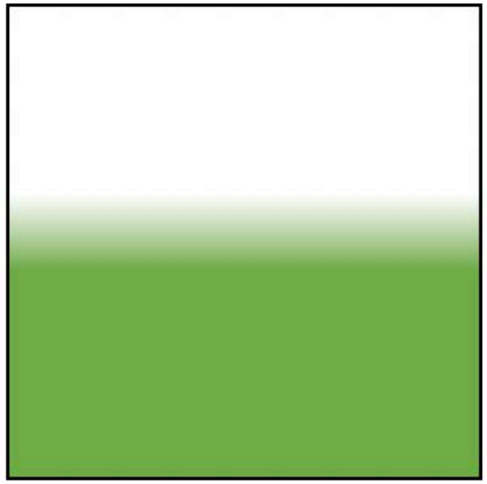
Synapse in canonical orientation



Probability of being a post-synaptic voxel

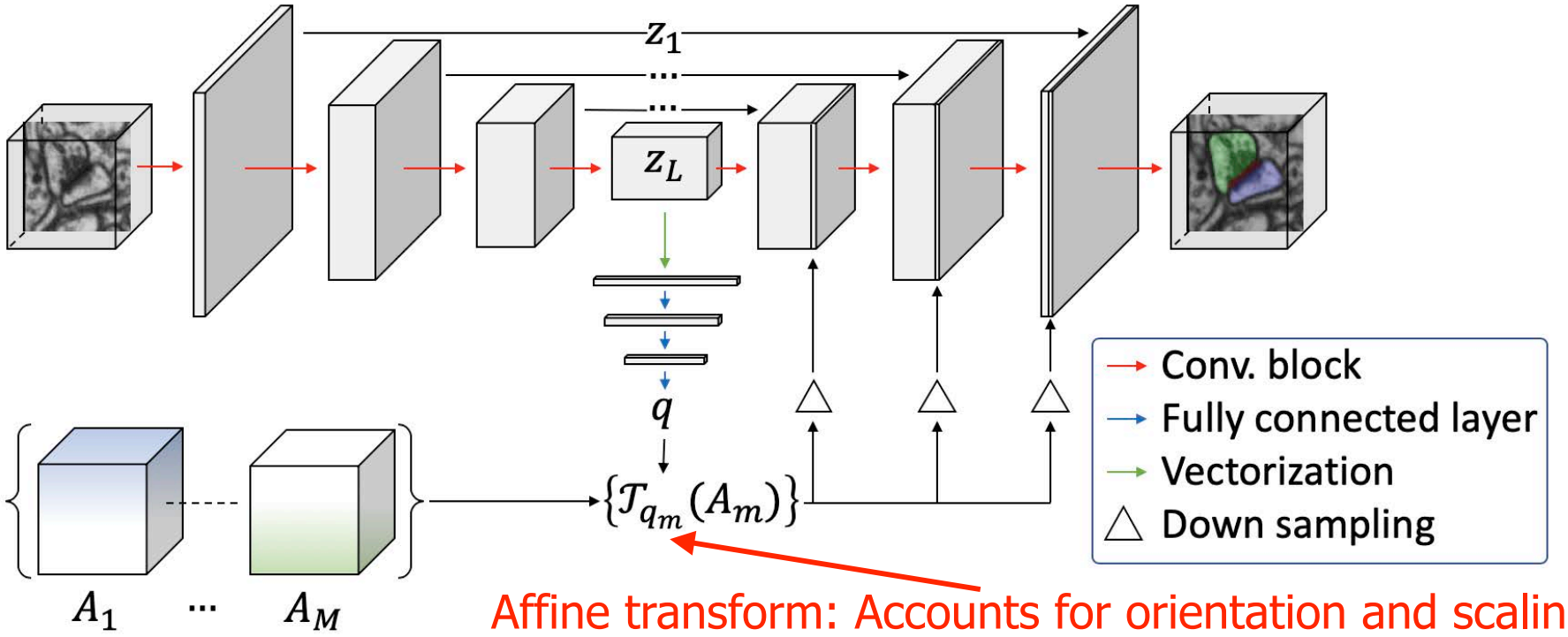


Probability of being a cleft voxel

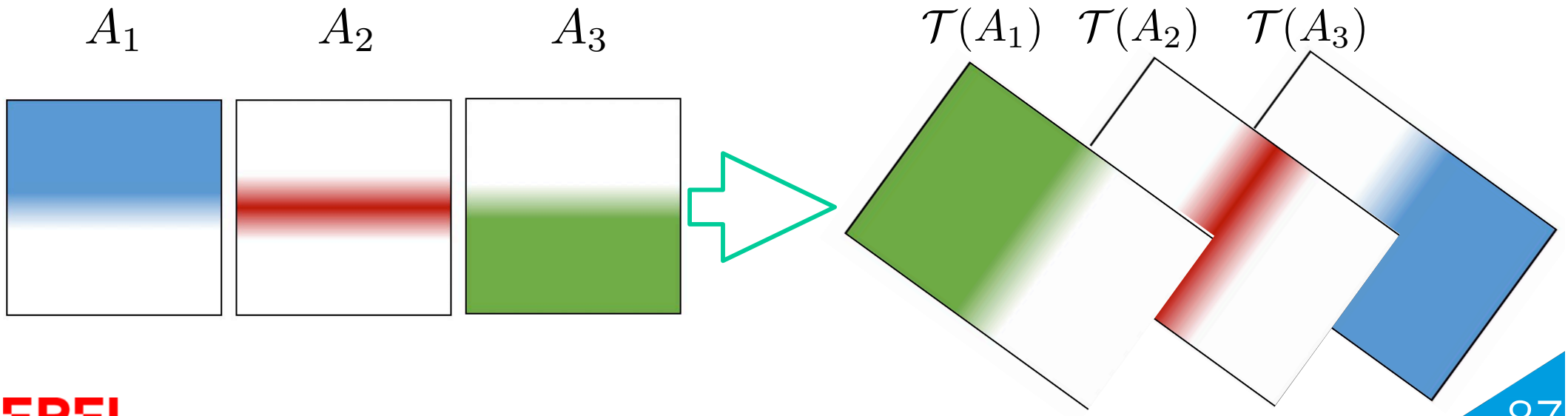


Probability of being a pre-synaptic voxel

U-Net + Atlases



Affine transform: Accounts for orientation and scaling.

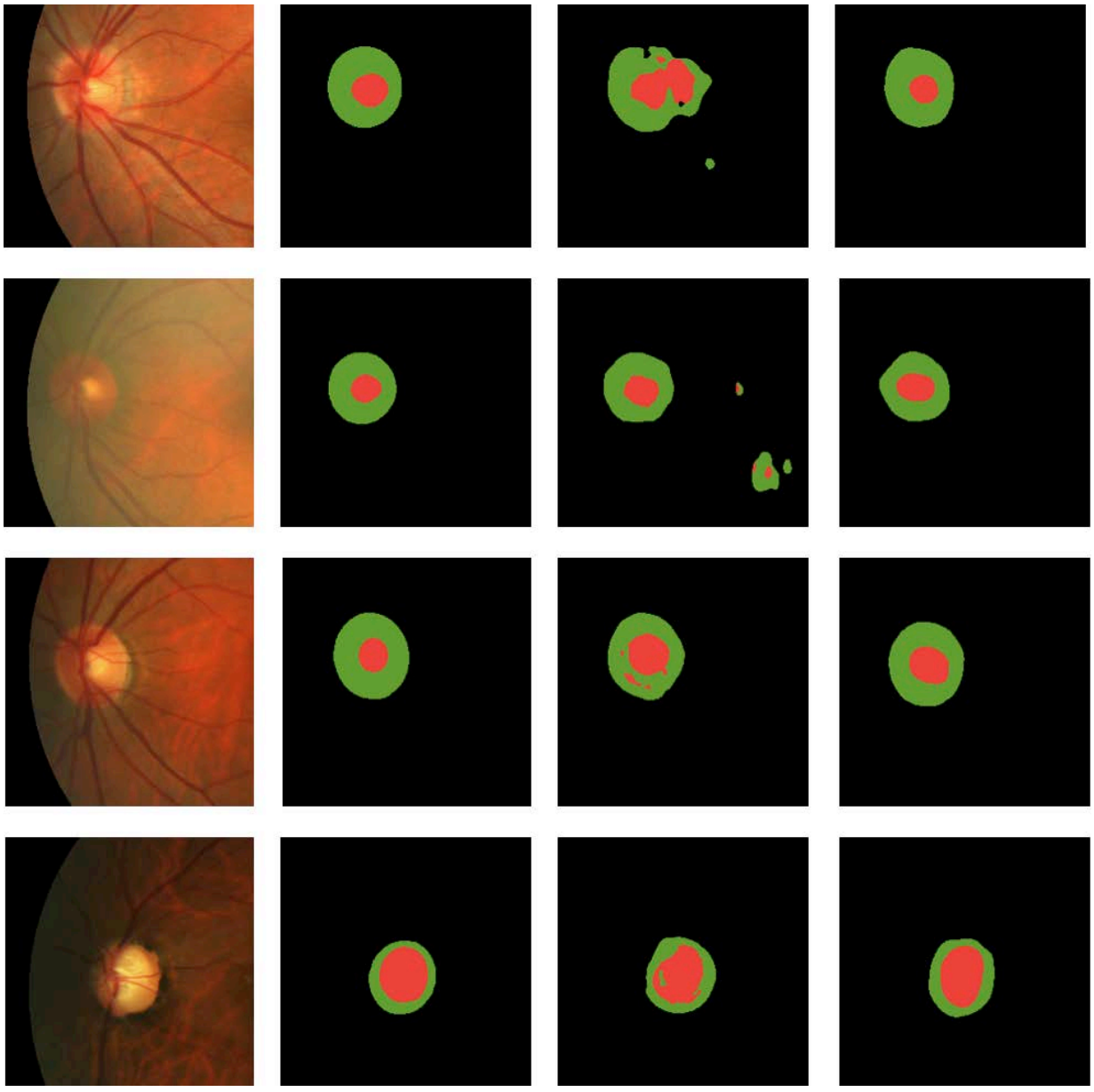


3D Synapses

Synaptic Junction Segmentation

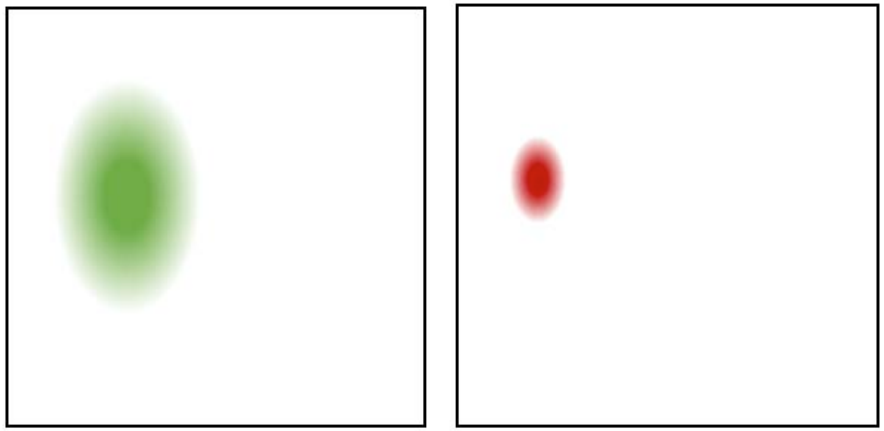
Baseline vs PA-Net

Atlases for Optic Disk Segmentation



Ground truth U-Net PA-Net

Probabilistic Atlases



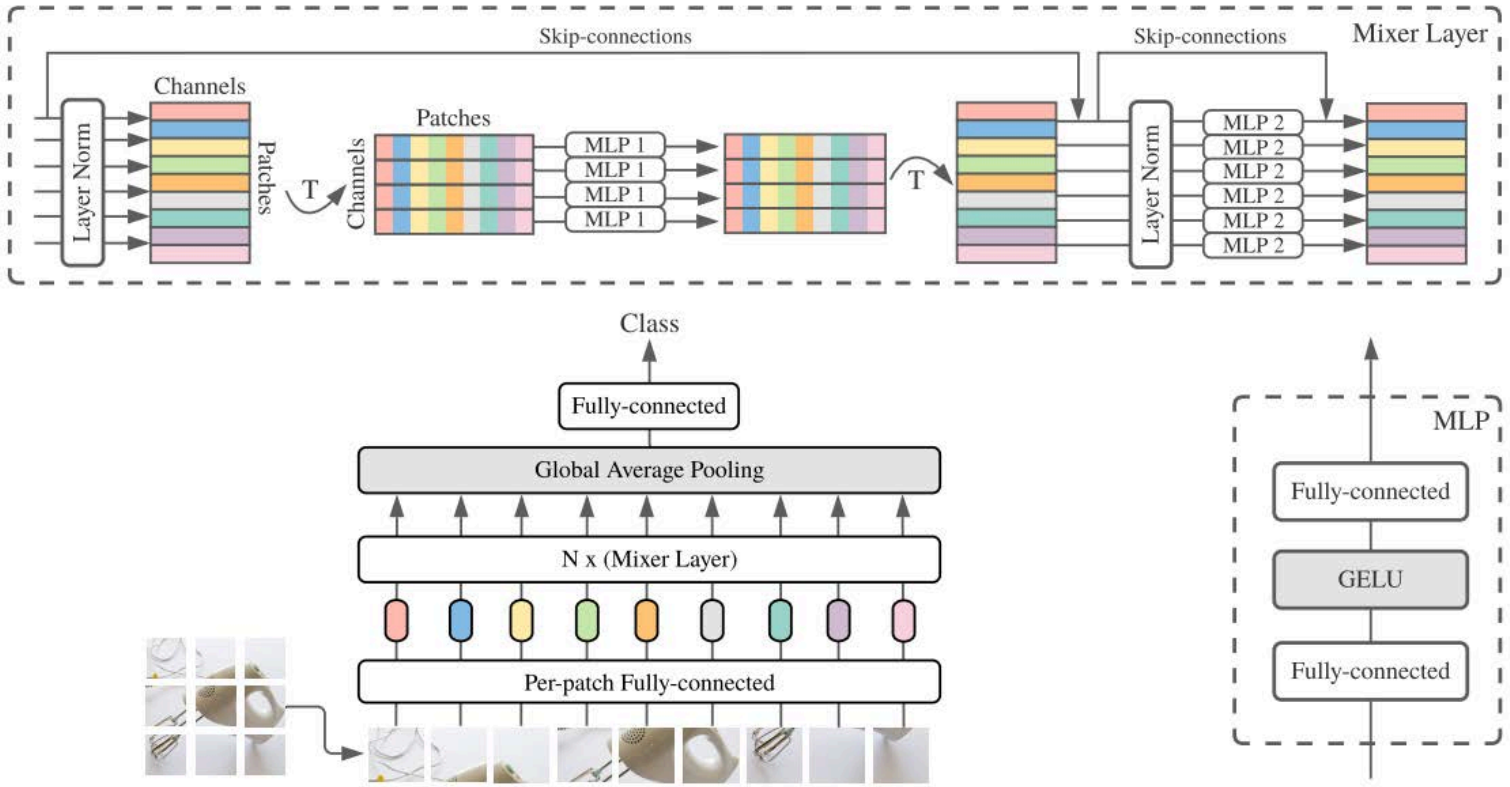
Optic disk Optic nerve

Moral of the Story

- Deep Networks are powerful tools, especially when there is a lot of training data.
- However, modeling your problem properly is still needed to achieve the highest possible level of performance.
- The old techniques often inform our design choices.

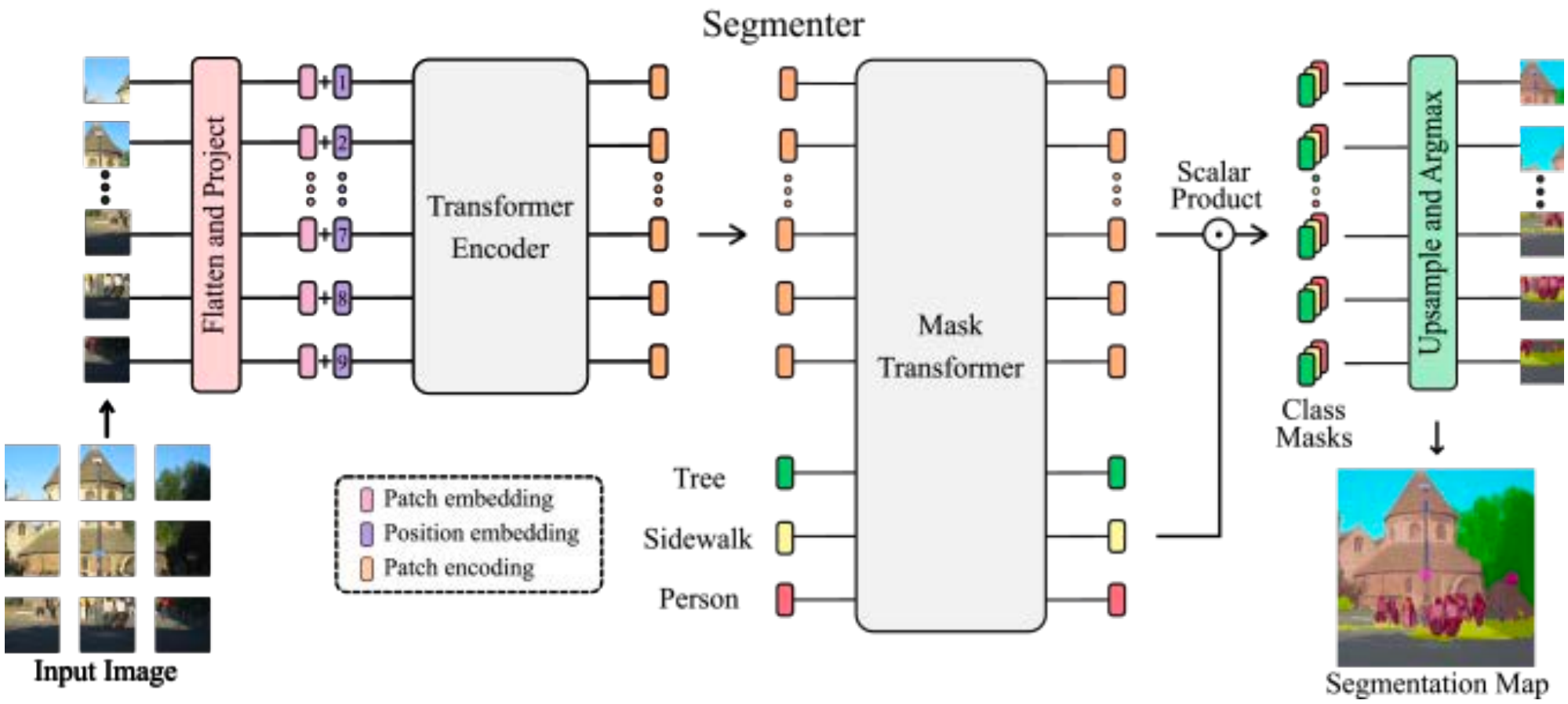
—> Yet another reason why I am still talking about them.

Reminder: Vision Transformers



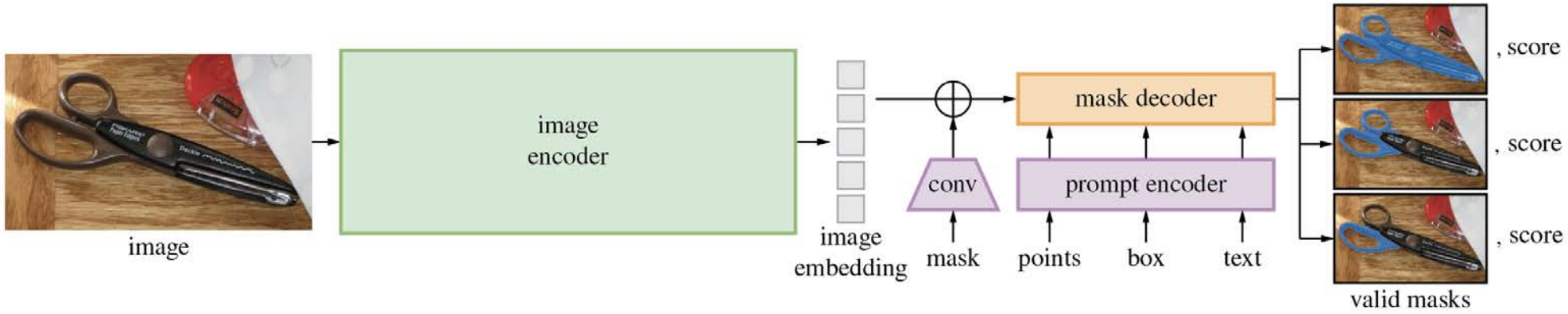
- Break up the images into square patches.
- Transform each patch into a feature vector.
- Feed to a transformer architecture.

Vision Transformers for Segmentation



- Replace the "recognition" machinery by a "mask transformer".
- Pros: Good at modeling long range relationships.
- Cons: Flattening the patches looses some amount of information.

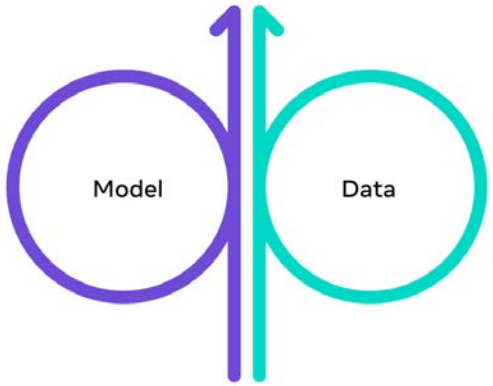
Segment Anything



- Based on a Vision Transformer
- Encoder/Decoder architecture
- Traditional prompt for segmentation
- Class agnostic segmentation

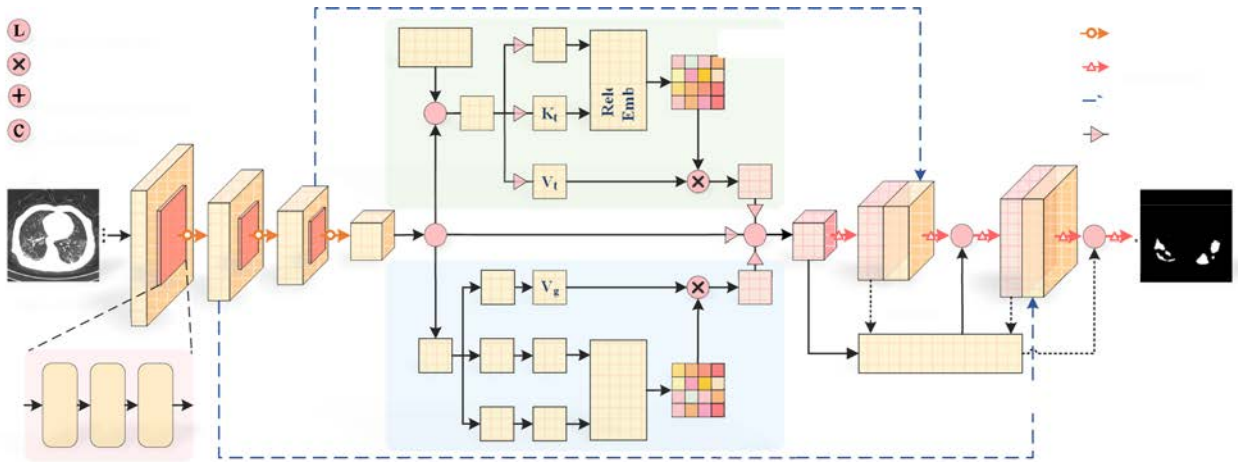
Segment Anything

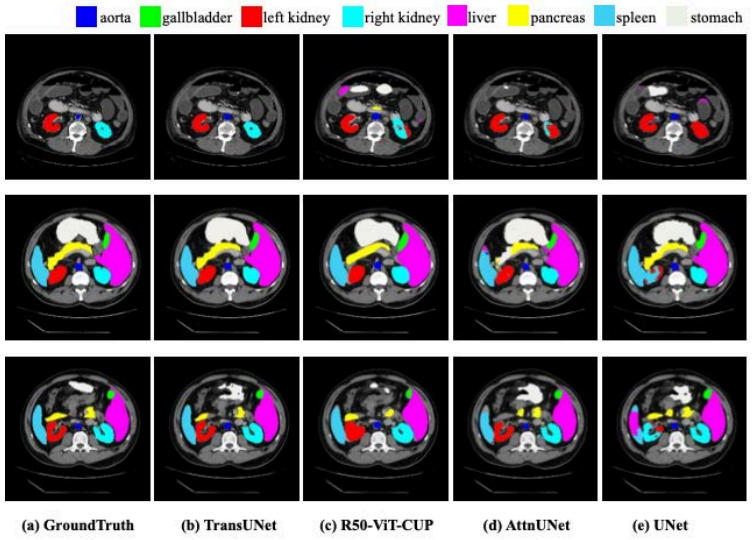
- The secret is in the data



- SA-1B dataset:
  - 11 million images - 1.1 billion masks
- Interactive and automated annotation

U-NET + Transformers





- A CNN produces a low-resolution feature vector.
- A transformer operates on that feature vector.
- The upsampling is similar to that of U-Net

—> Best of both worlds?

In Short

- Local methods can provide valuable information but are inherently limited.
- Domain knowledge, user interaction, and training data can be used to turn this data into usable results:
  - Given enough training data, deep nets deliver the best performance today.
  - It can be further enhanced by introducing domain knowledge.
  - Given smaller amounts of training data, K-Means and graphical models still have their uses.
- Same philosophy as for delineation.

What About the Dog?

Most current algorithms won't "see" it!