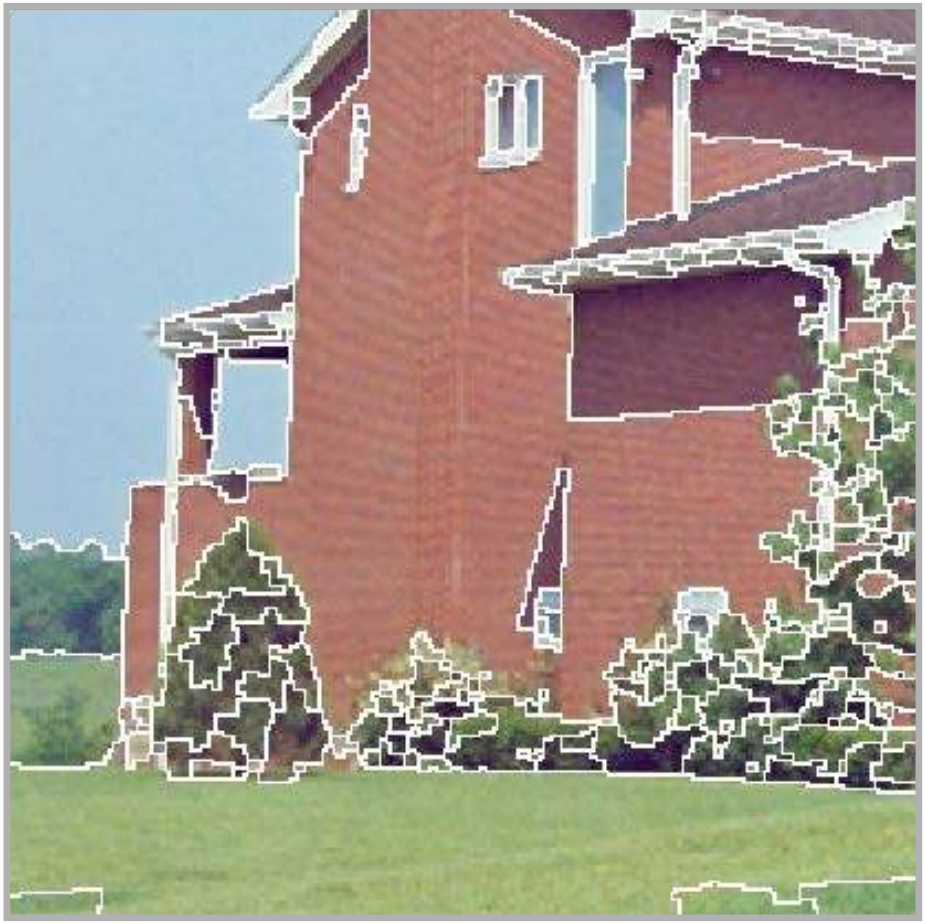
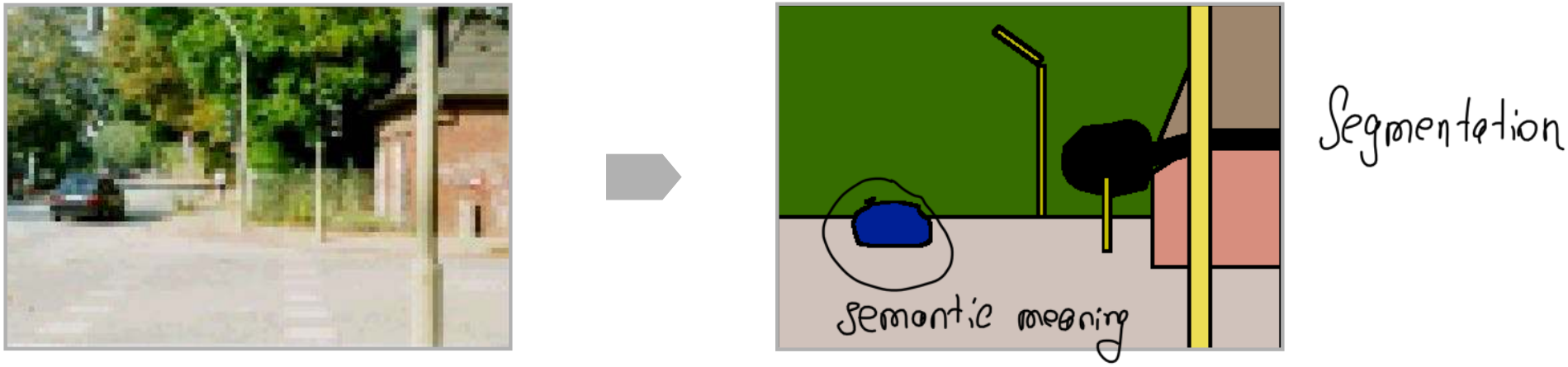
Regions

→ correspond to meaningful semantic object



Pascal Fua
IC-CVLab

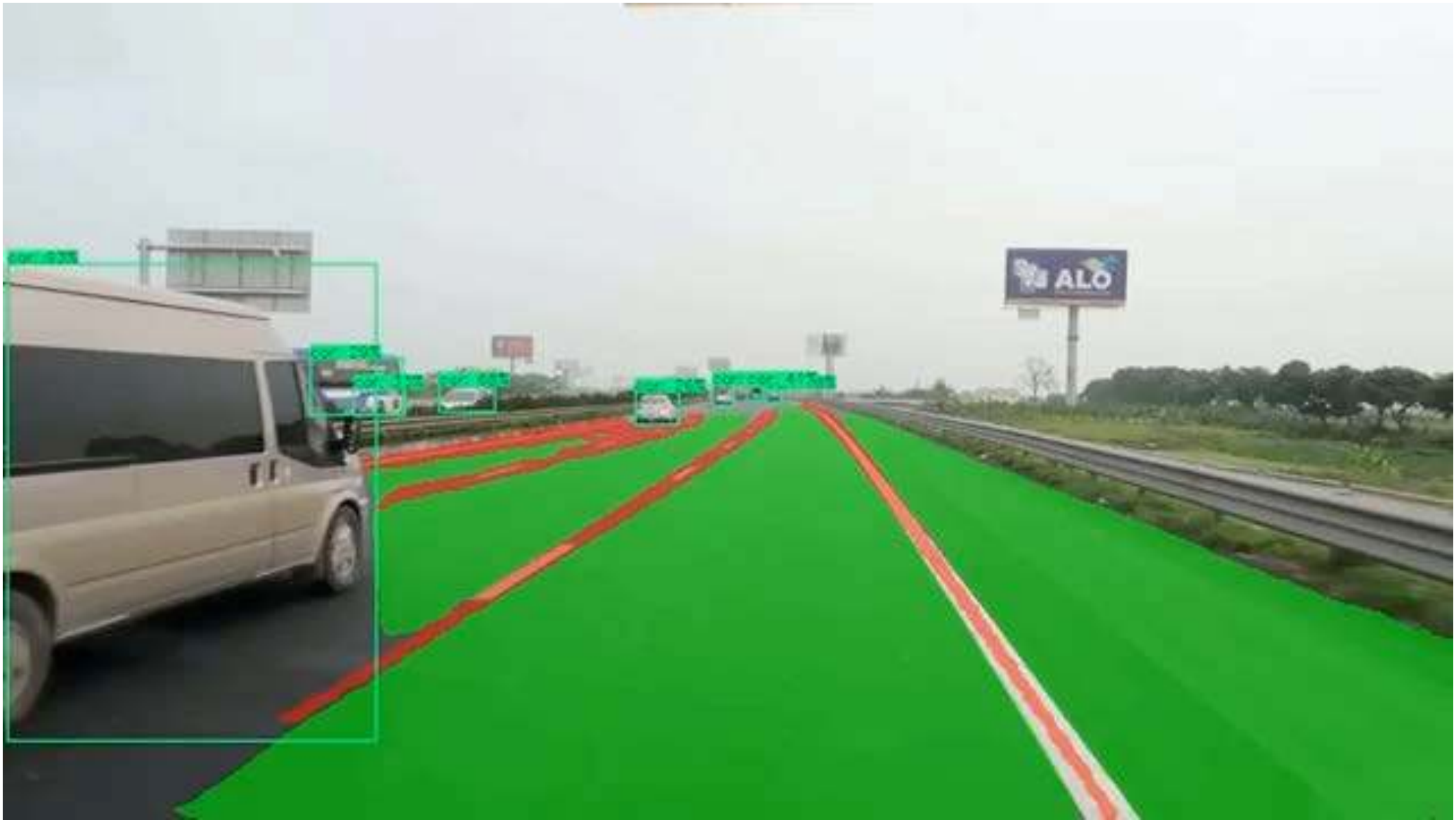
Region Segmentation



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

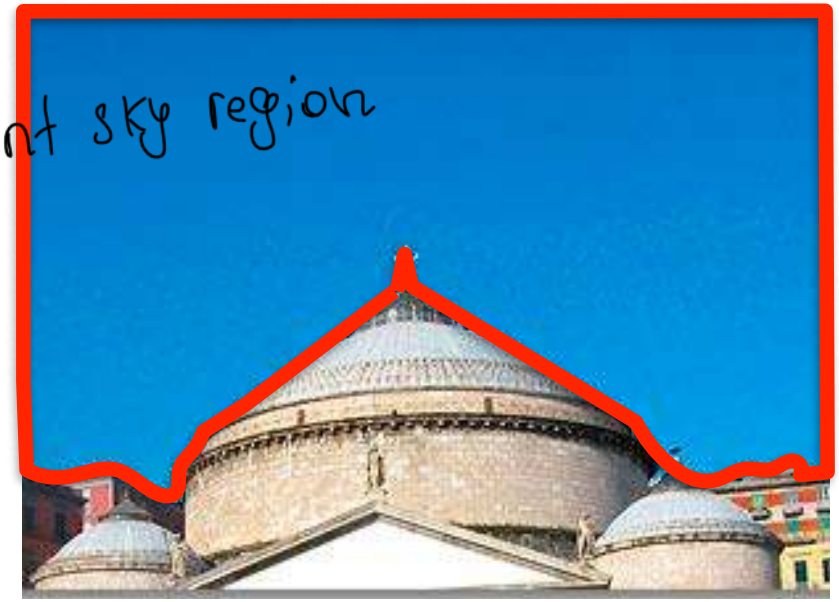


Vu et al. , ArXiv'22

Applications: Photoshop

I find this blue sky too bland!

I prefer this one.

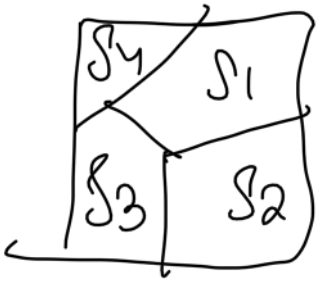


Find the sky region.

Replace it.

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

In Theory



Look for an image partition such that:

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

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

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

homogenously uniform in some sense

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

where H measures homogeneity.

Complex Gray Level Variations

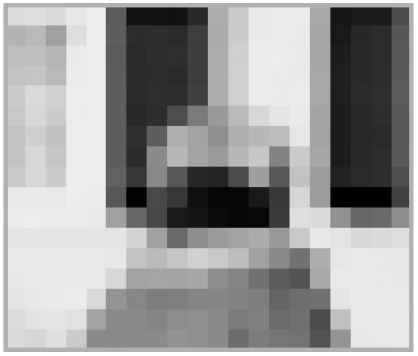
do not correspond to face

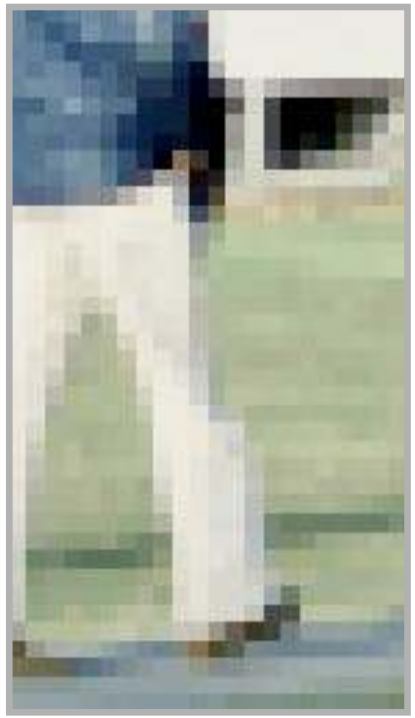
does not work

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

Context is Essential

What is it?

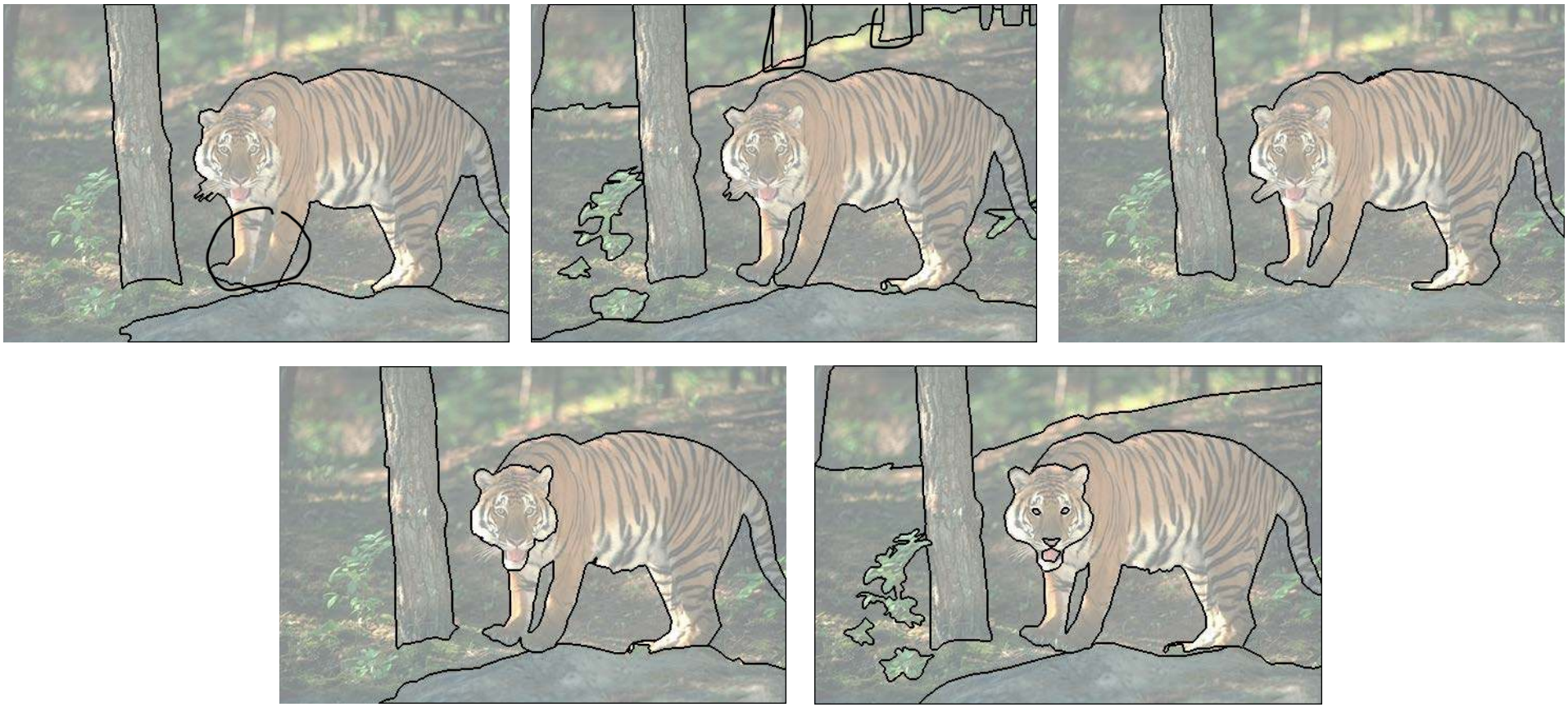




Without context, it's hard to segment

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.

purpose?

No absolute definition on segmentation

Homogeneous or Not?

In terms of color → ✗

statistics → ✓

(orange with stripes)

texture →

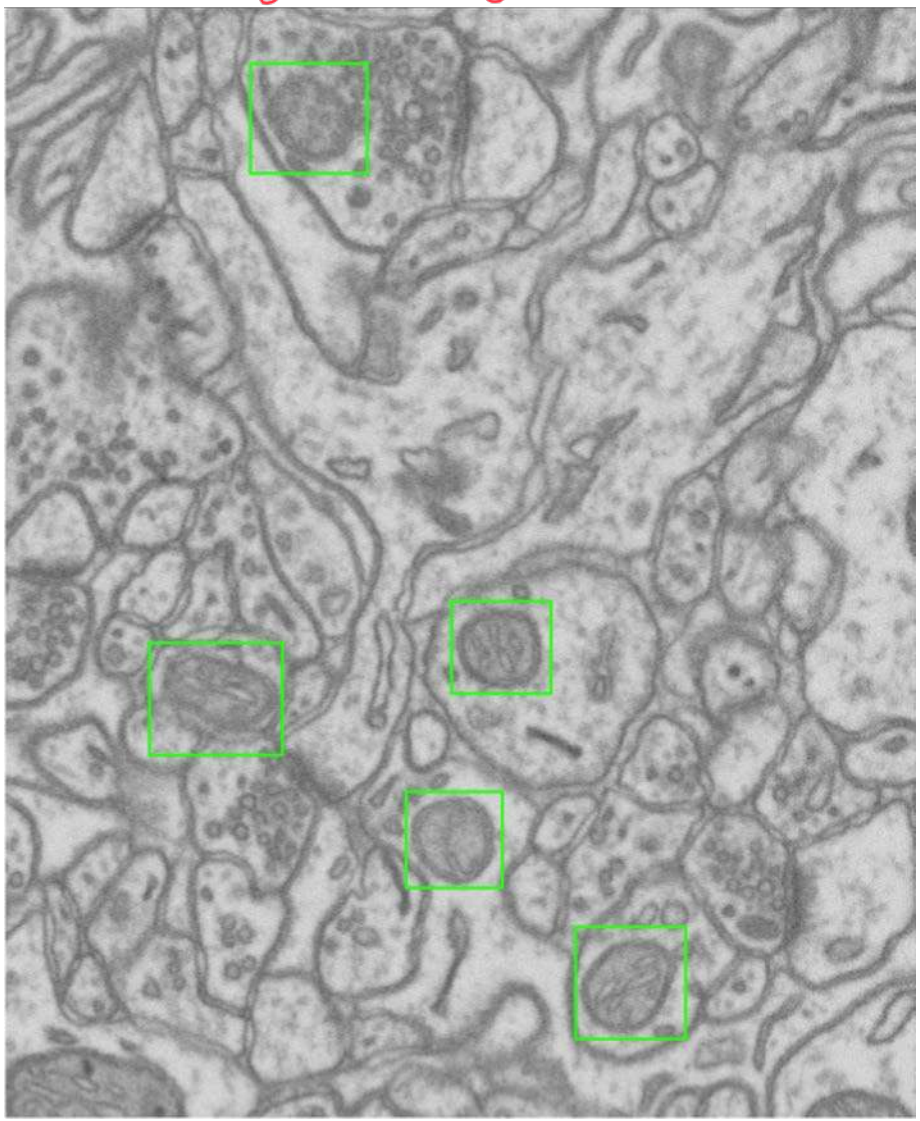
orientations of gradients

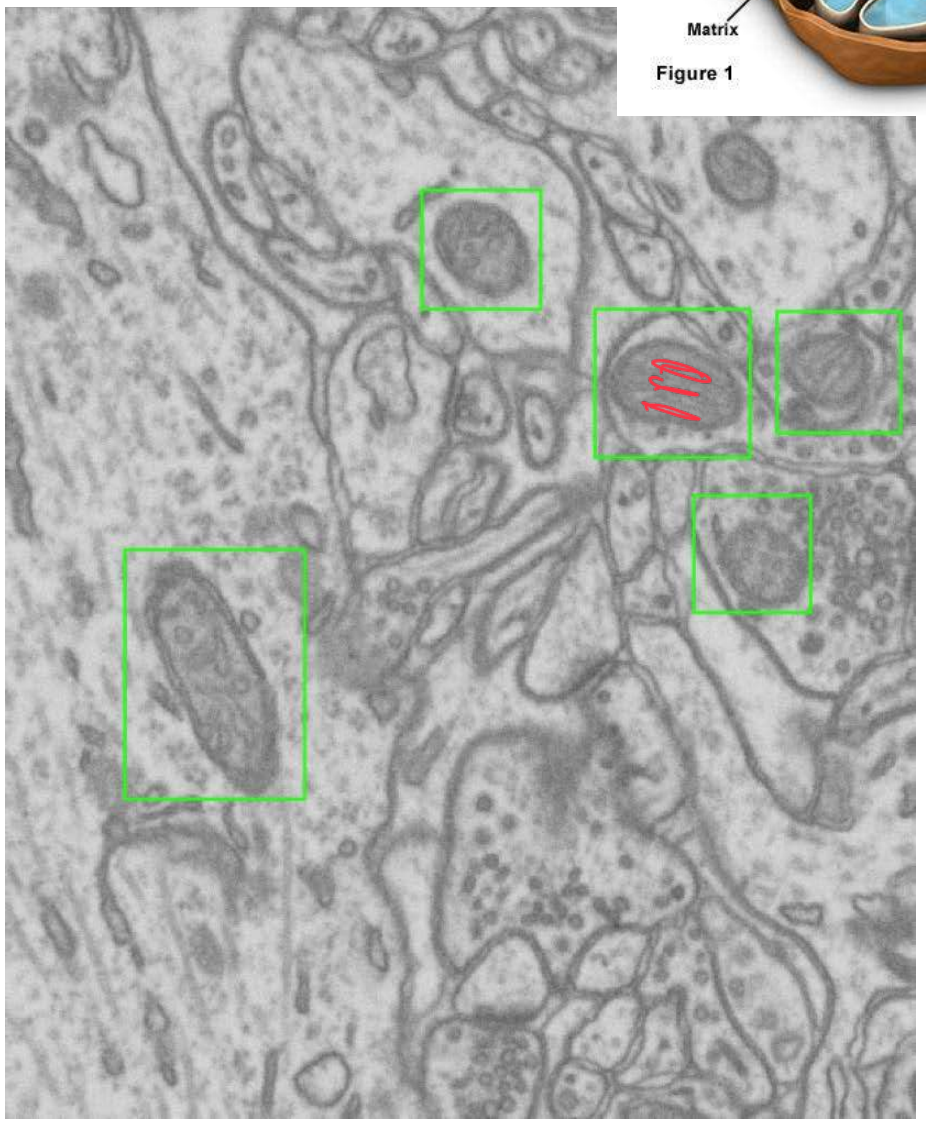
What is homogeneous in some parts of these images are the statistical properties, not the actual pixel values.

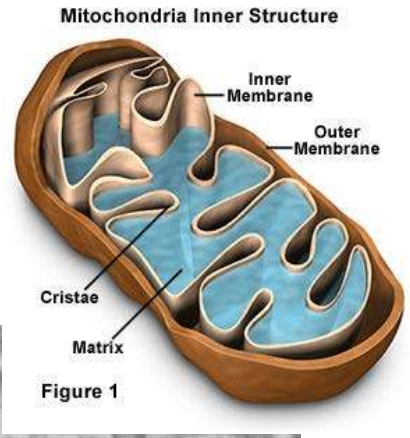
Mitochondria

homogenous, segment ✓

textual information







In Theory

Merge:
- Start with a partition that satisfies Eq. 3.
- Satisfy Eq. 4 by merging regions.

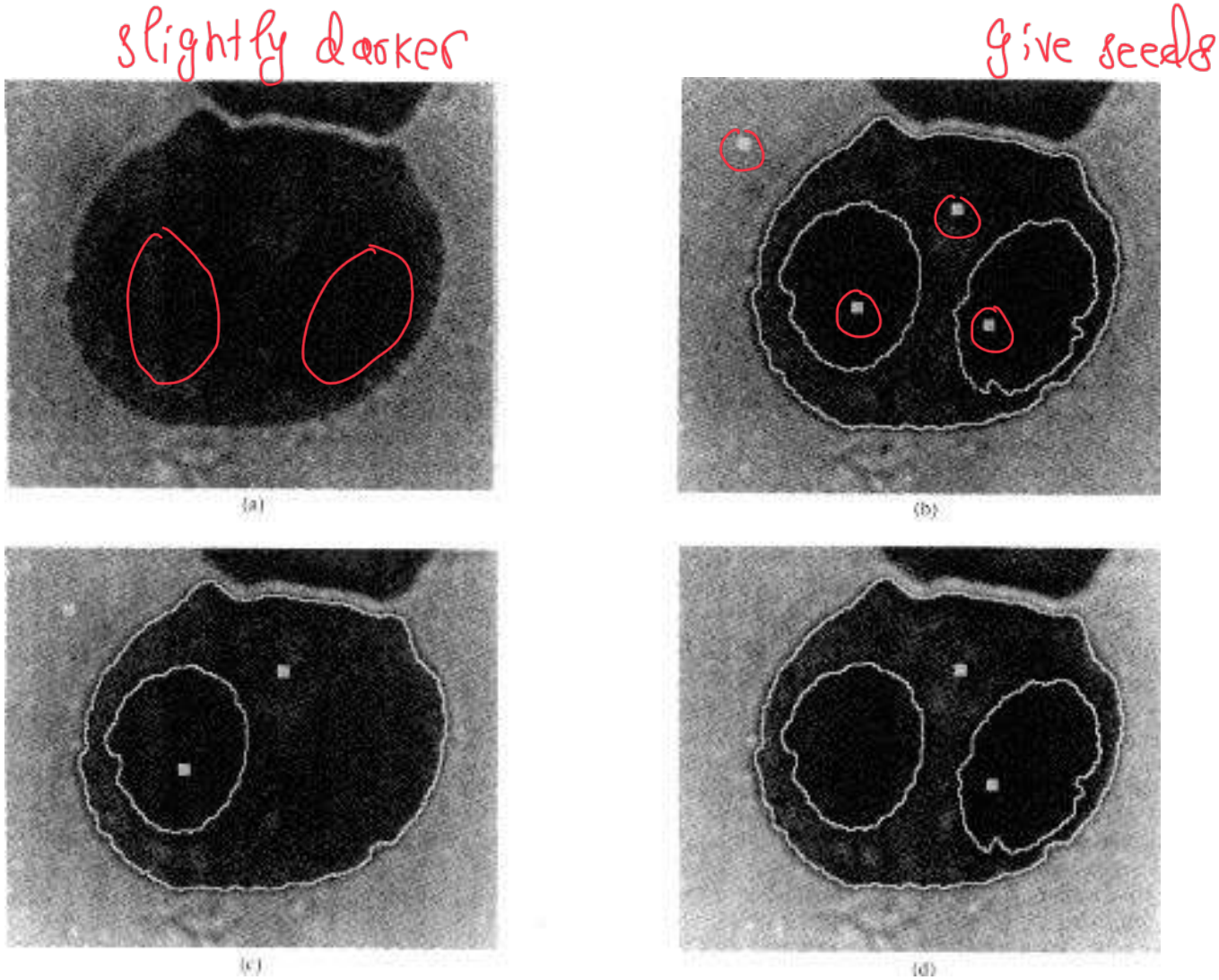
Split:
- Start with a partition that satisfies Eq. 4.
- Split regions until they all satisfy Eq. 3.

Homogeneity:
- Uniform gray-level or color statistics.
- Regions to which a parametric surface can be fitted.

From Simple to Complex Algorithms

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

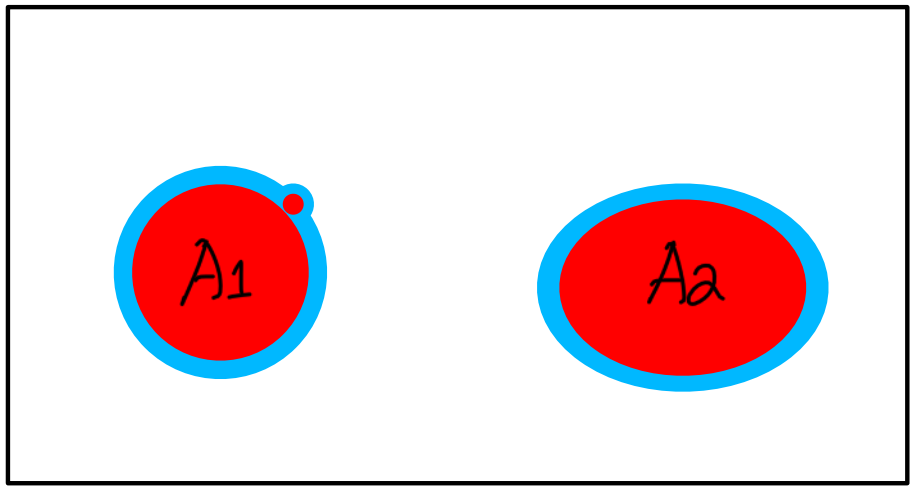
Region Growing



Interactive Segmentation of a Cell

Region Growing

progressively adding neighbours



- Labeled pixels.
- Unlabeled pixels.

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

seed regions

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

unlabeled pixels

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

immediate

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

take minimum value of these 2

grey-level of x

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.

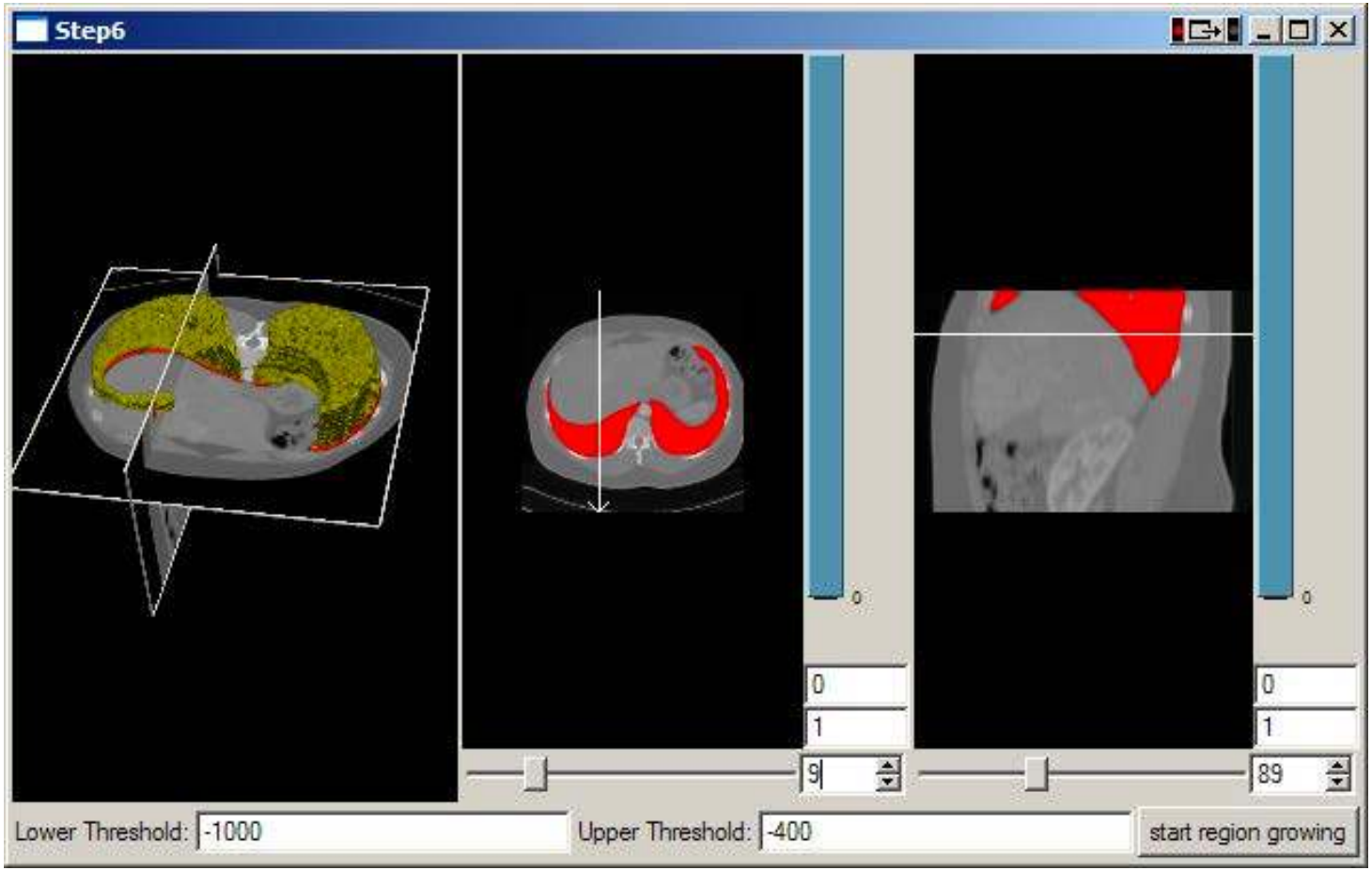
// T sorted list

→ fast

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

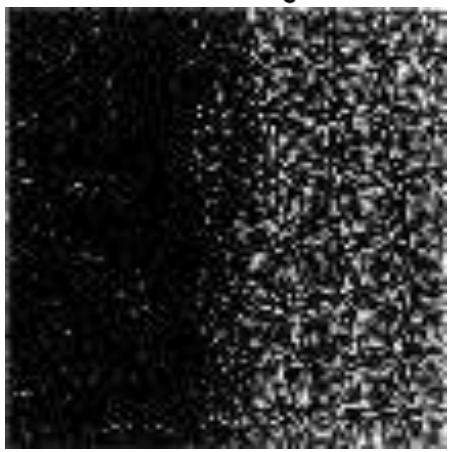


Medical Imaging Interaction Toolkit

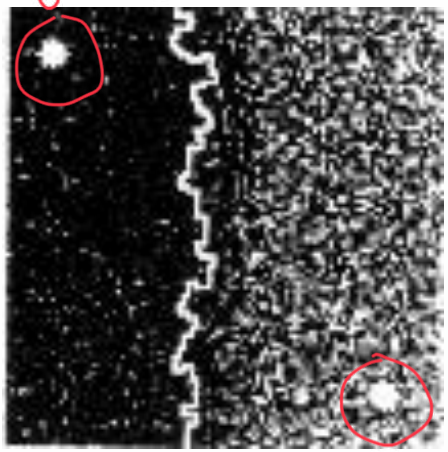
Limitations

Synthetic image

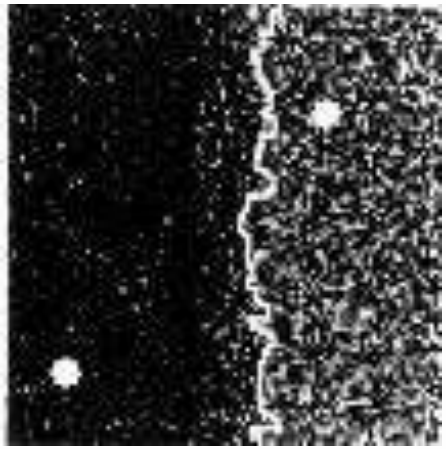
seeds



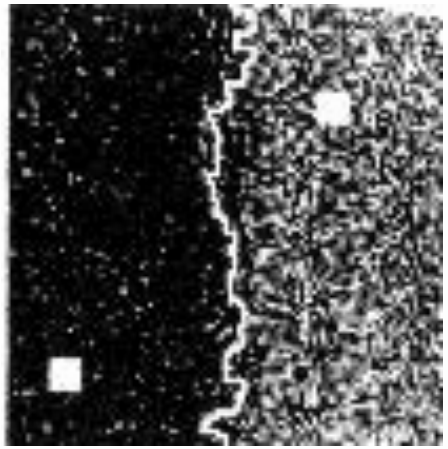
Original image



Result given two seeds.



Result given two different seeds.



Result given larger seeds.

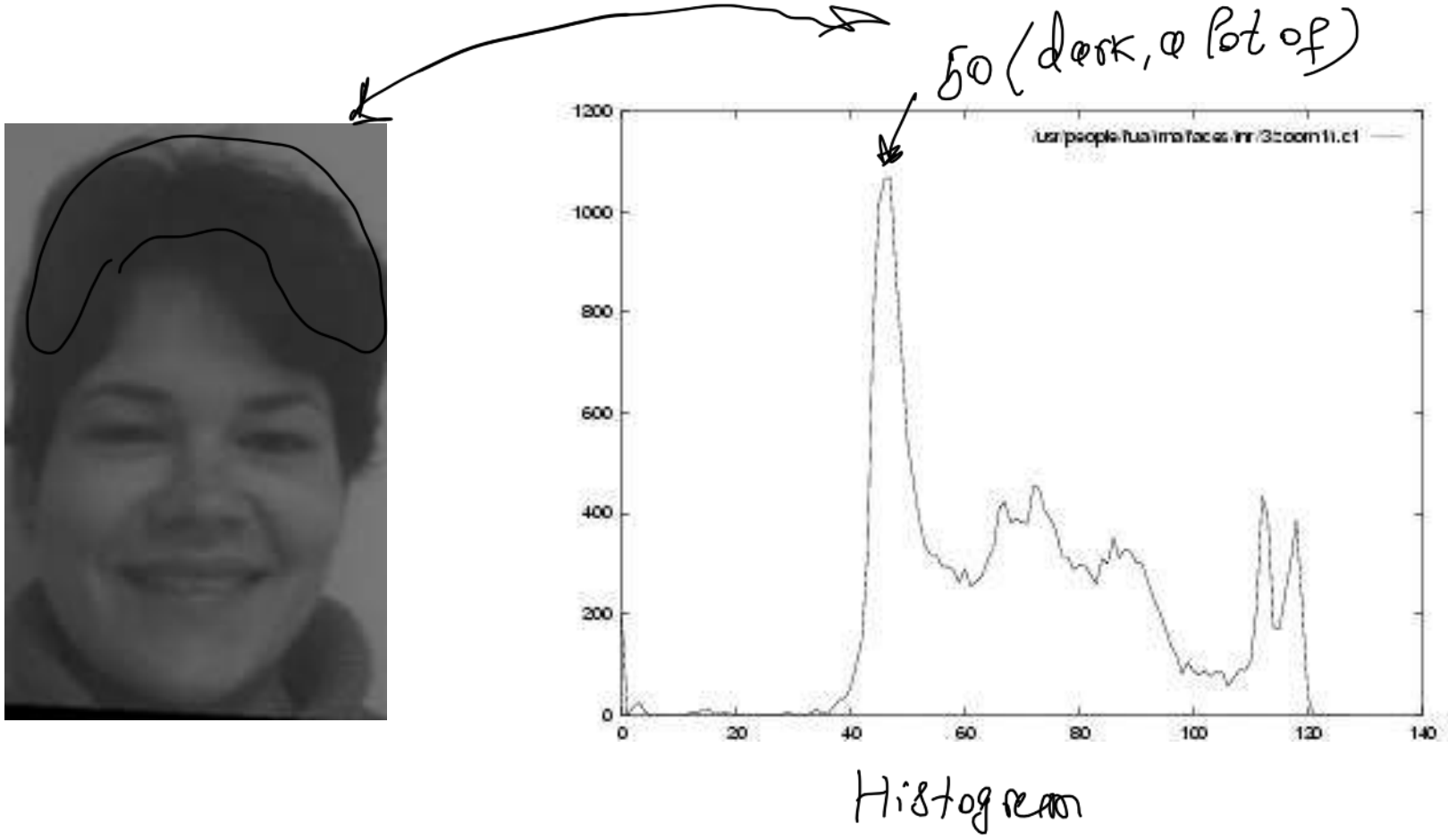
Greedy Algorithm

Statistics become more reliable

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

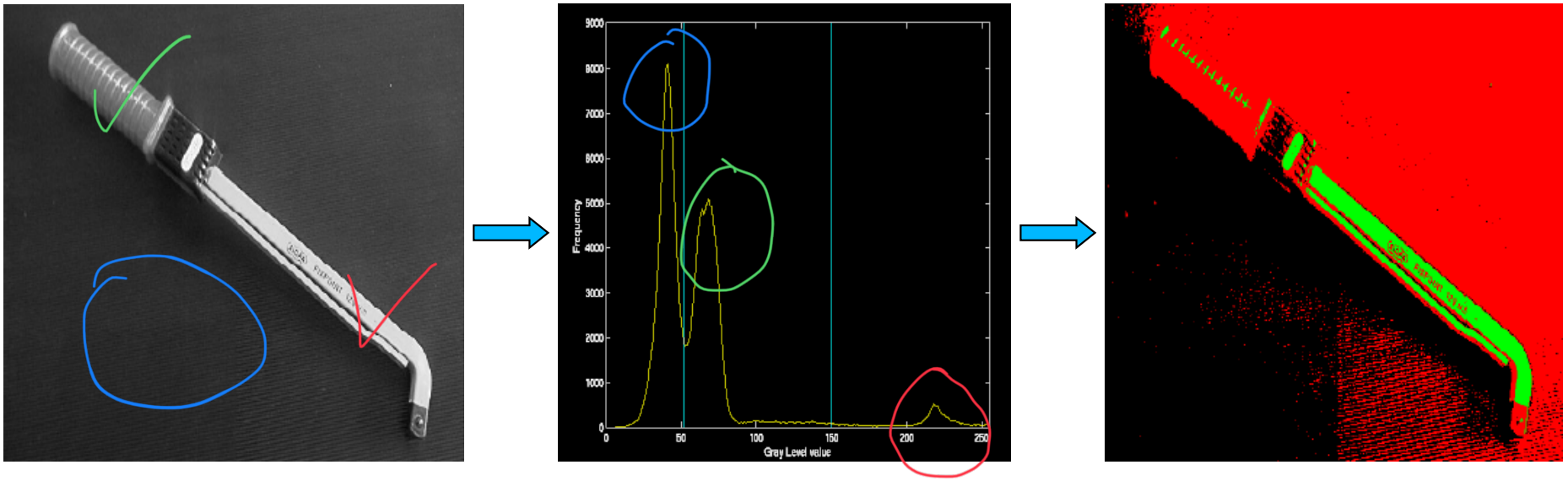
↳ not very robust

Image Histogram



Number of pixels that have a given gray level.

Histogram Splitting

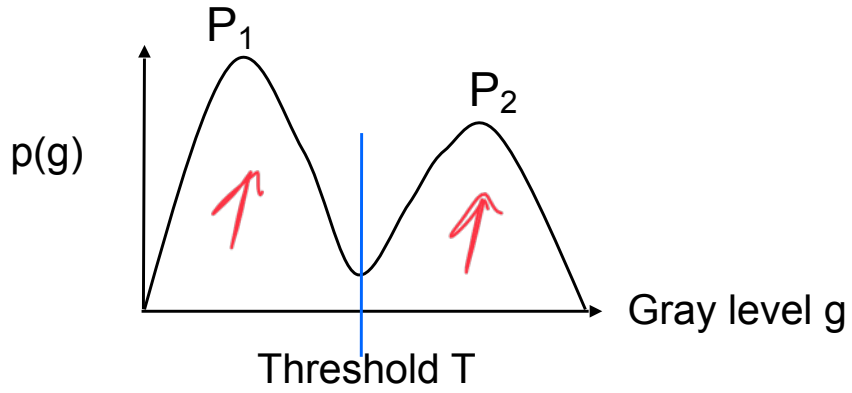


- Groups of similar pixels appear as bumps in the brightness histogram (Blue vertical lines)
- Split the histogram at local-minima
- Label pixels according to which bump they belong to → connected components give regions

—> Cannot stop there, must go on.

not considering geometric proximity (Pixels a Pot)

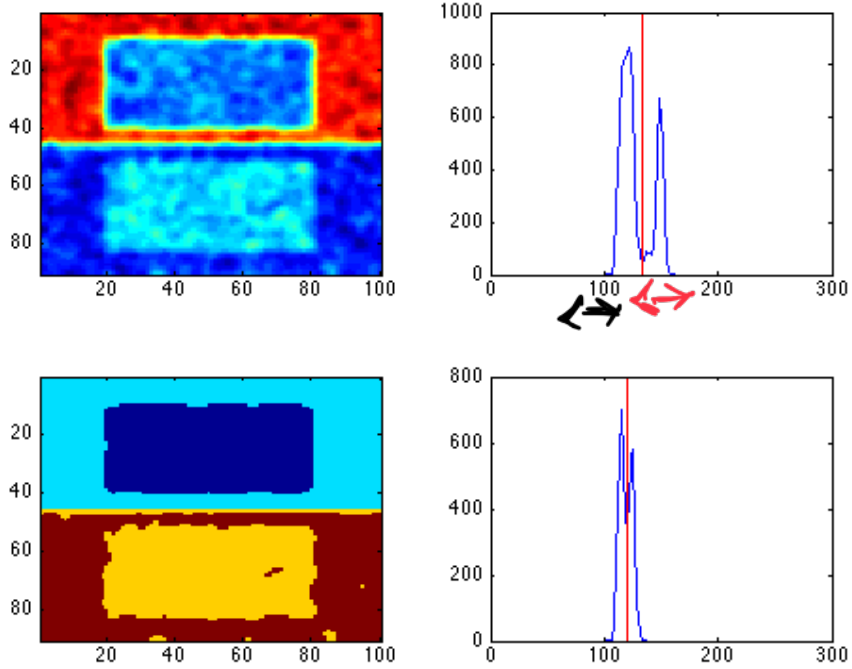
Recursive Splitting



- Compute image histogram.
- Smooth histogram. ~ so that you don't have too many local minima
- Look for peaks separated by deep valleys.
- Group pixels into connected regions. ✓
- Smooth these regions. (within seperate images)
- Iterate.

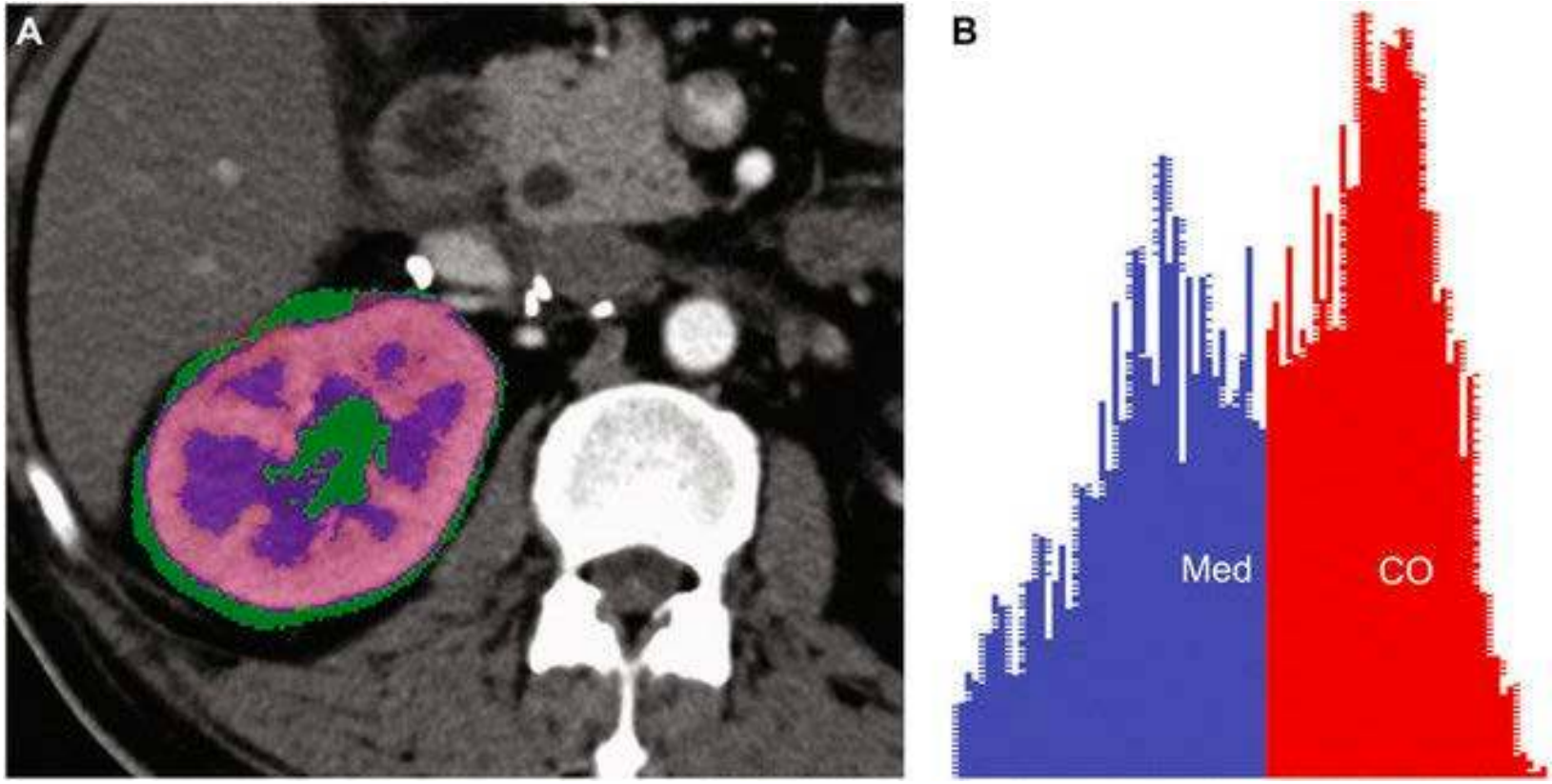
Recursive Splitting

- 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.



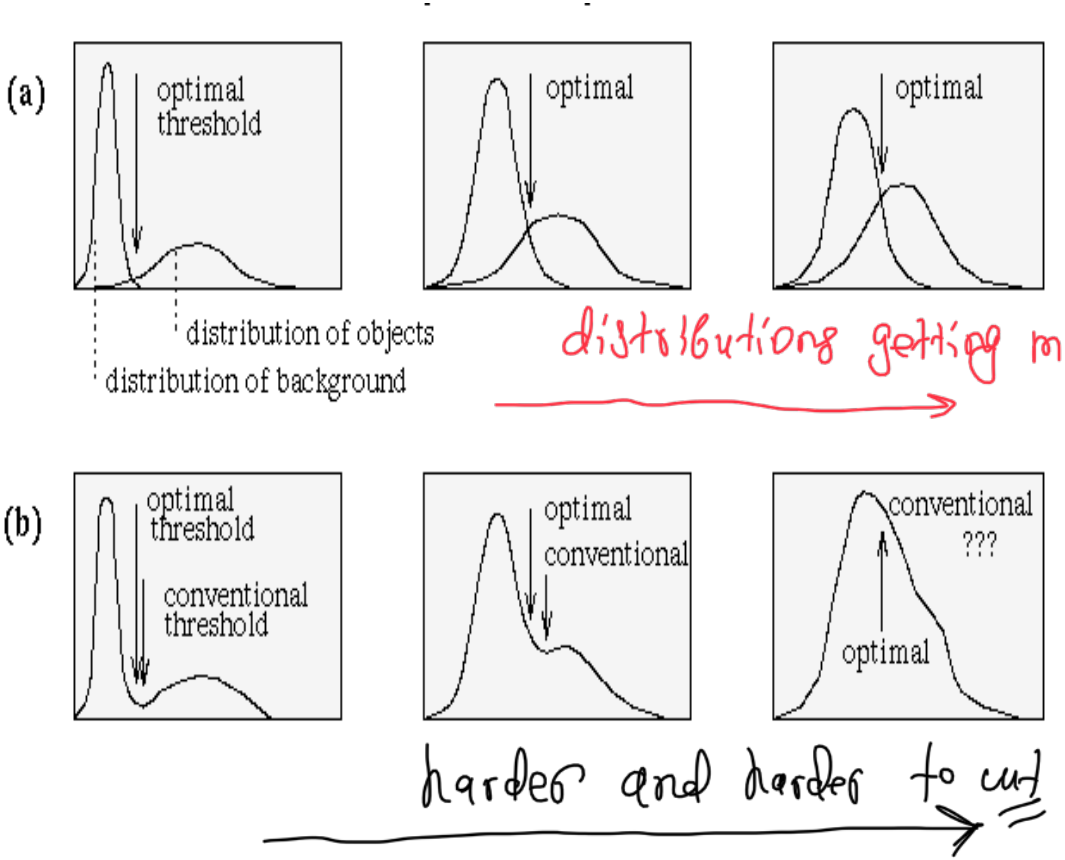
-> Decisions can be deferred until enough information becomes available.

Medical Application



It has its applications but .....

Finding Thresholds is Hard



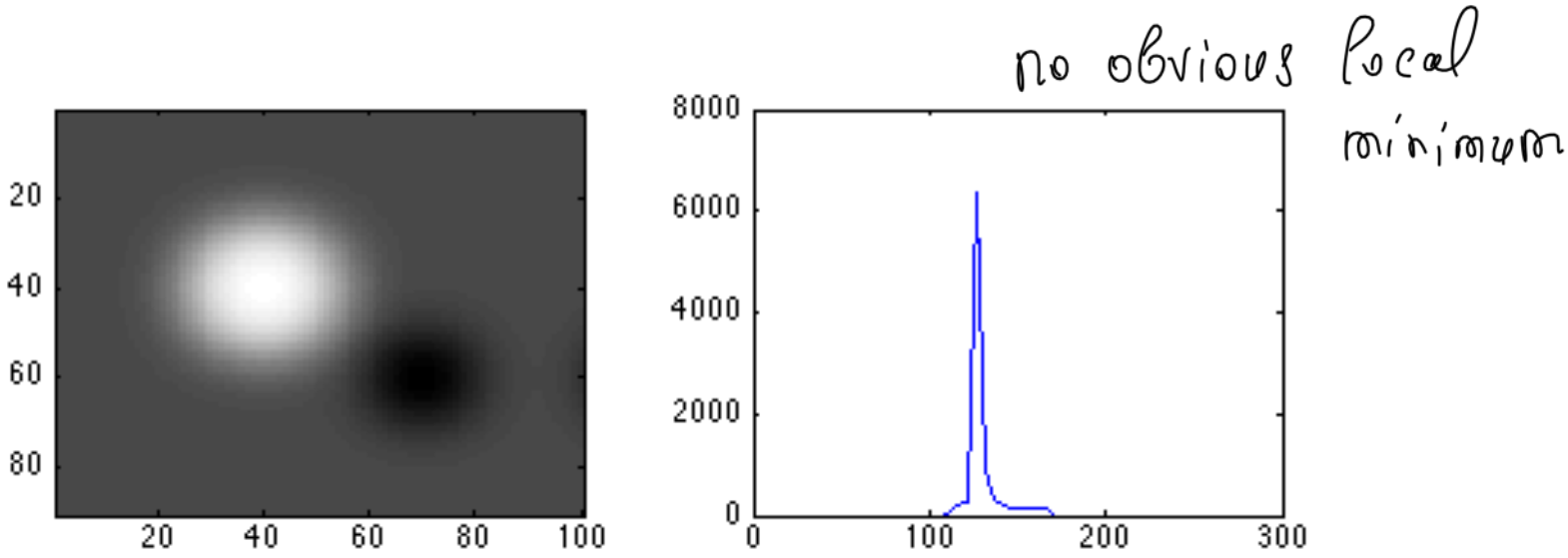
Probability distributions

Corresponding histograms

—> Choosing optimal thresholds is a difficult optimization problem.

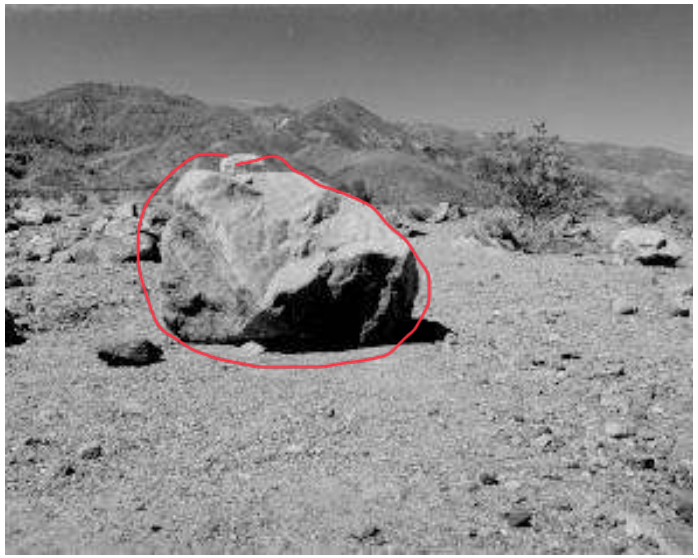
Stop split histogram → when you don't have local minima

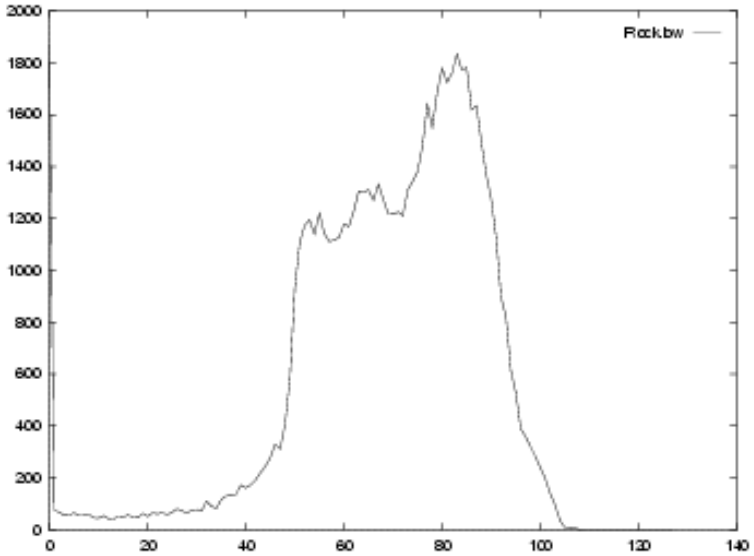
No Obvious Threshold



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

No Valid Threshold at All

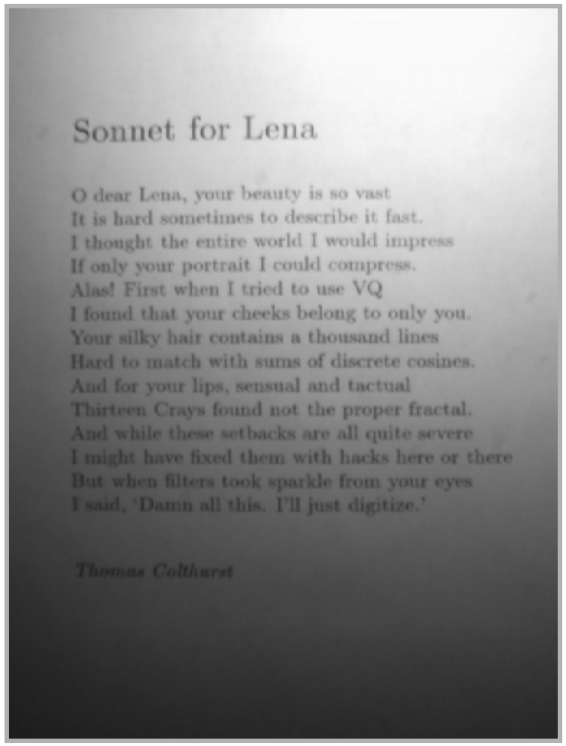


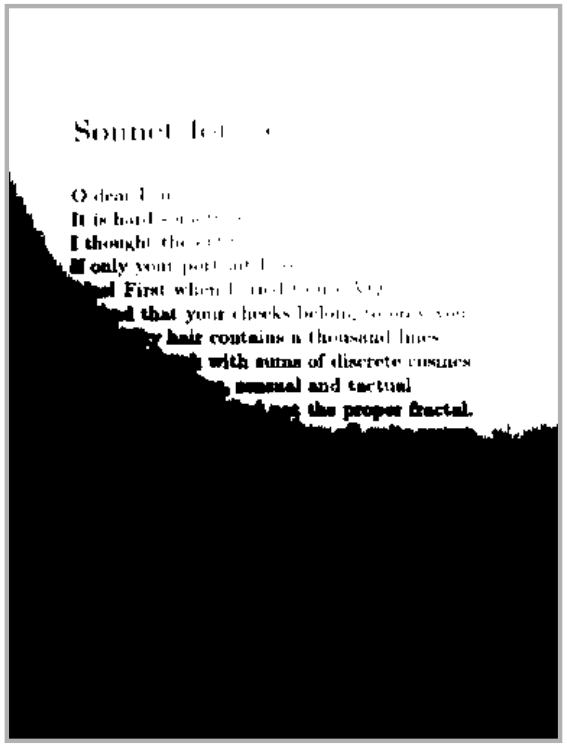


hard to find threshold that would find the stone's boundary

In this image, no threshold can generates meaningful results.

No Global Threshold



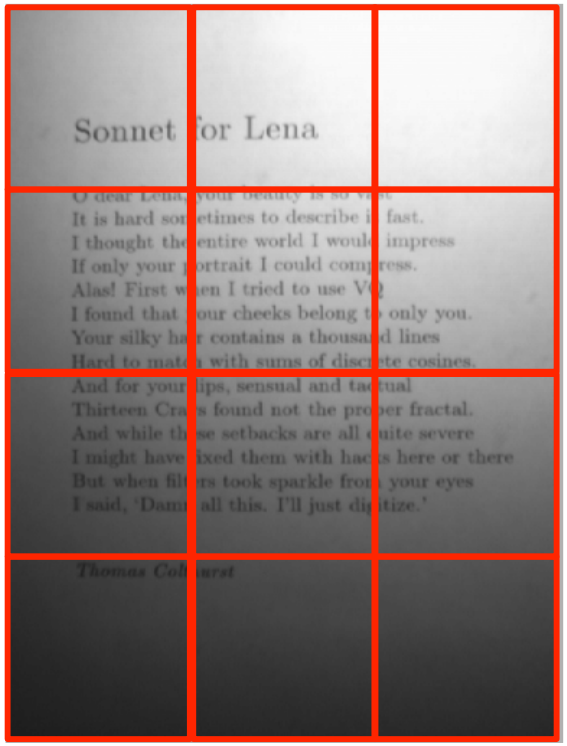


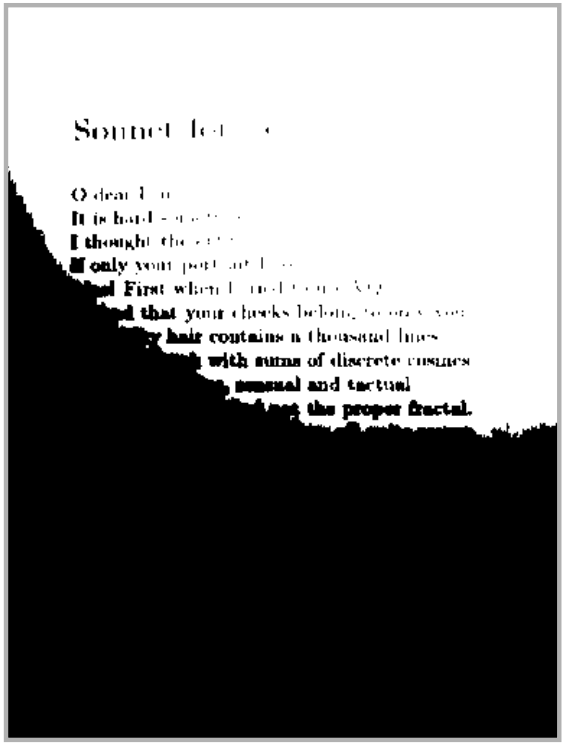
→ with split

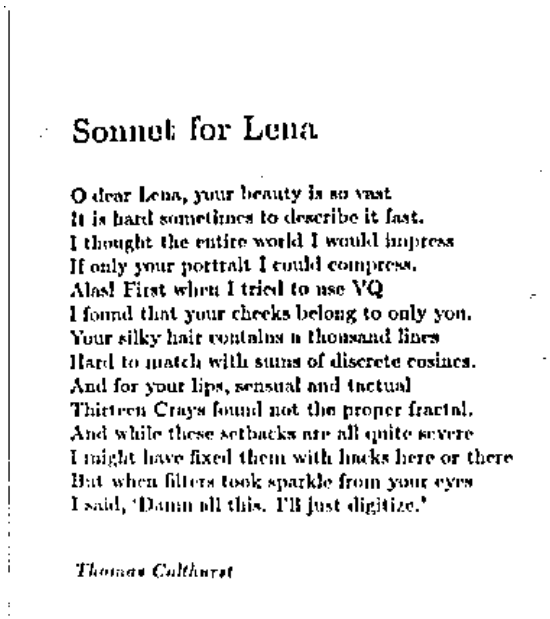
Because of the complex illumination:

- 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





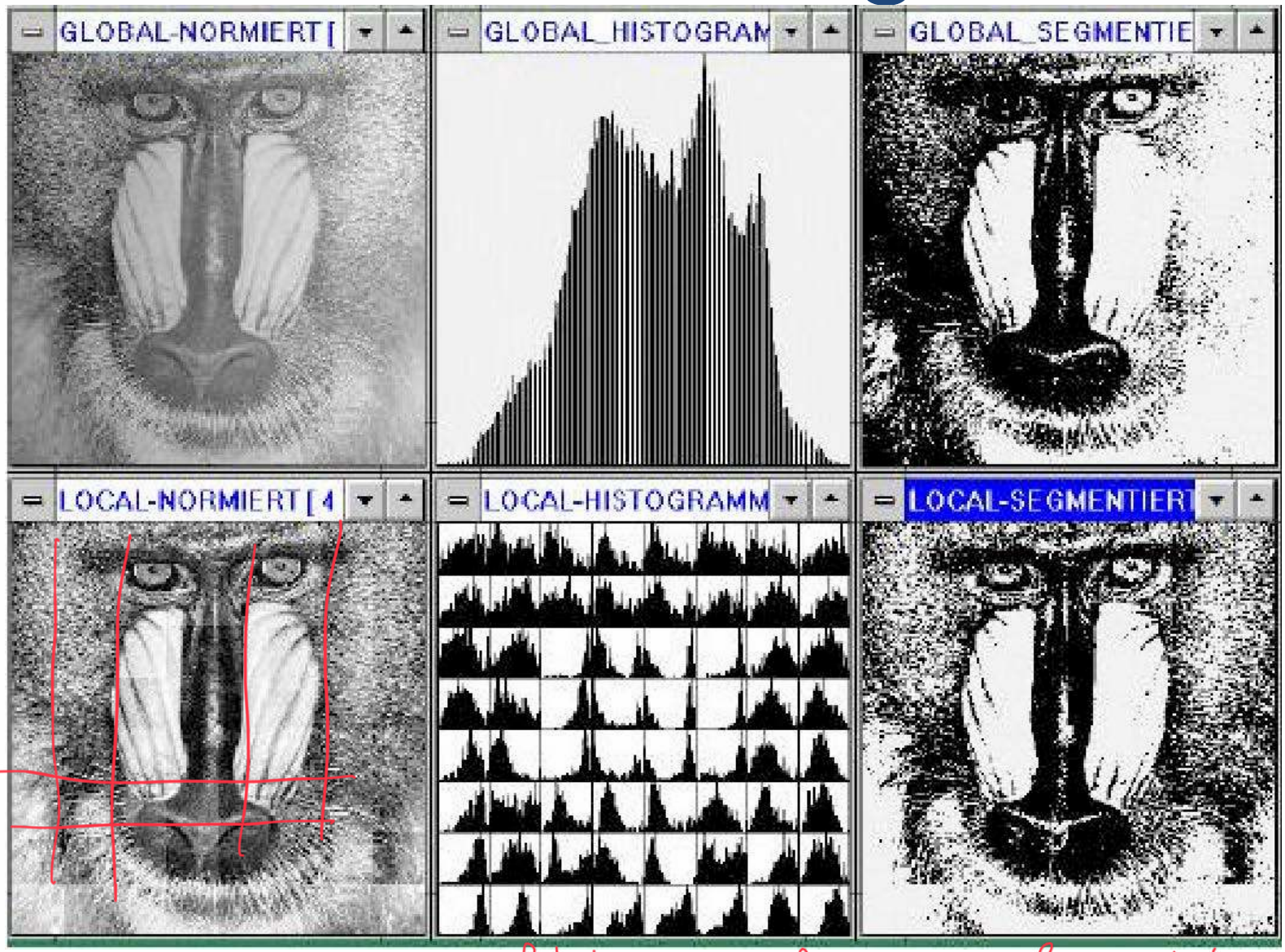


Global Local

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

Use Local Histograms

Local information
⇒ proximity, geometry matter!



teorically not meaningful

plot histograms indiv.

Segment them

- 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. ↳ spatial information between regions ⇒ not at all!
- Thresholds are hard to find.
- Some boundaries will not be found because the gray levels on both side belong to the same histogram peak. (rock example)

Using Color

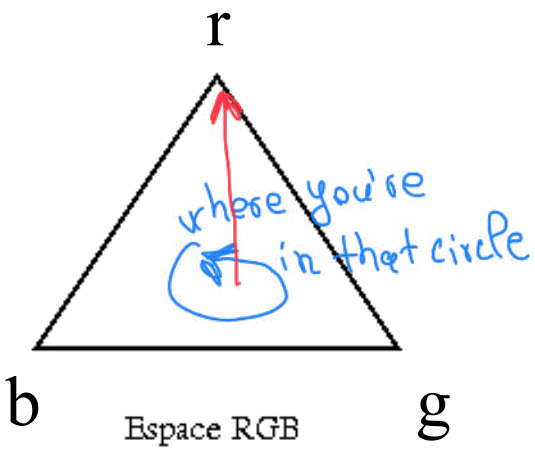
RGB

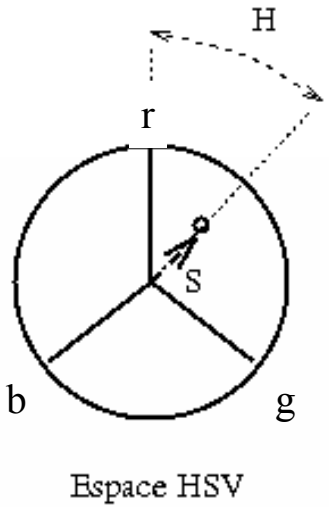
R

B

G

Color Space





$$V = R + G + B$$

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

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

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

Gray-level

- Value

defines color

- Hue

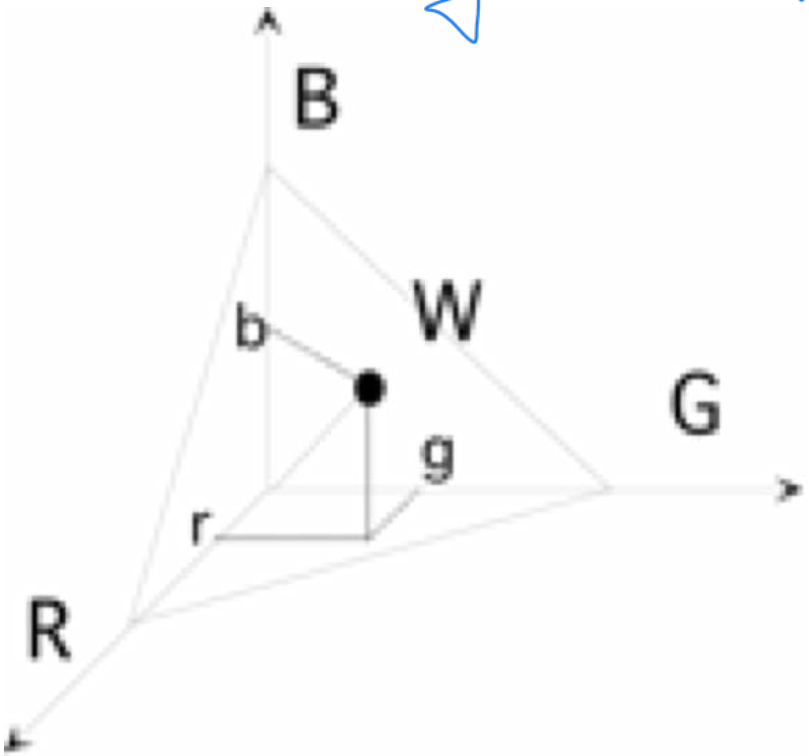
- Saturation

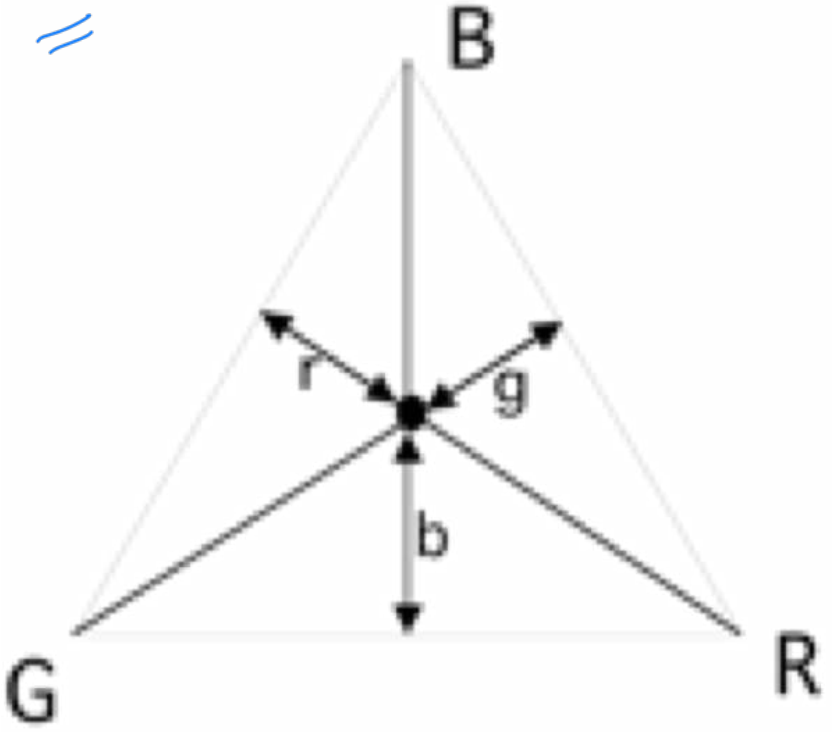
R=G=B (In the center)

↑ fully saturated color

RGB Chromaticity Diagram

Chromaticity of the pixel → color

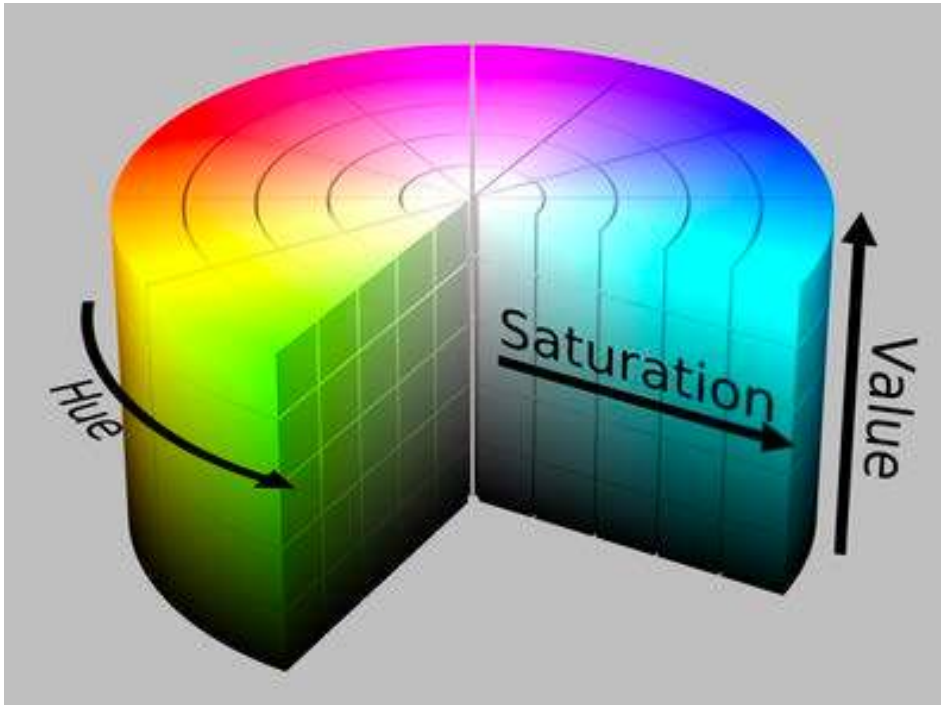




The Maxwell triangle involves projecting the colors in RGB space onto R+G+B=1 plane.

→ Chromaticity becomes independent from luminance.

HSV Space



Hue / Saturation / Value

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

→ angle

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

$1 - \frac{3\min(r,g,b)}{I}$

→ How close you're to the boundary of triangle

$$V = R + G + B$$

→ how bright you are

HSV Images

RGB

Value

R+G+B
gray-level image

saturation's low

How close we're to the pure color?

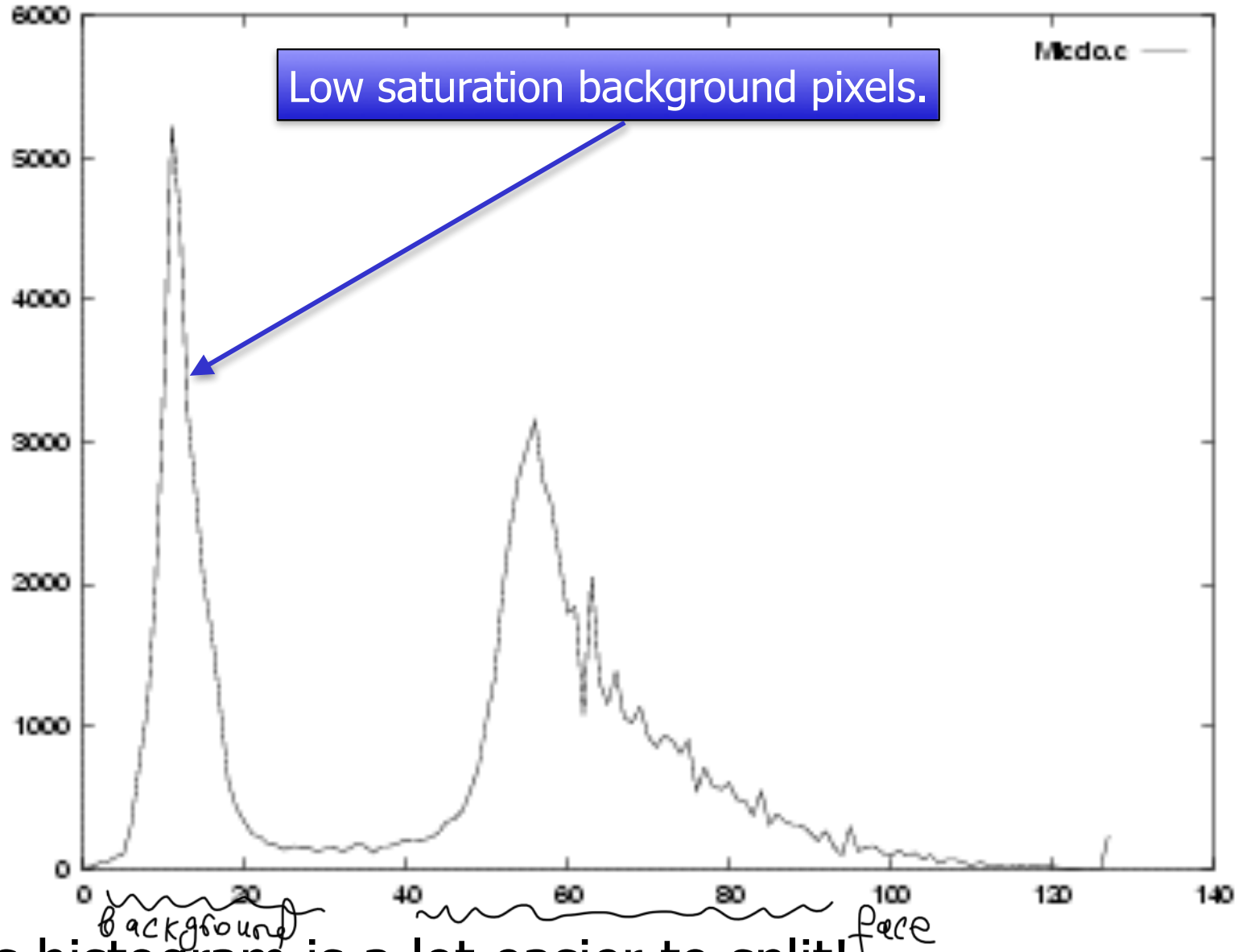
Saturation

what kind of color we're dealing with?

Hue

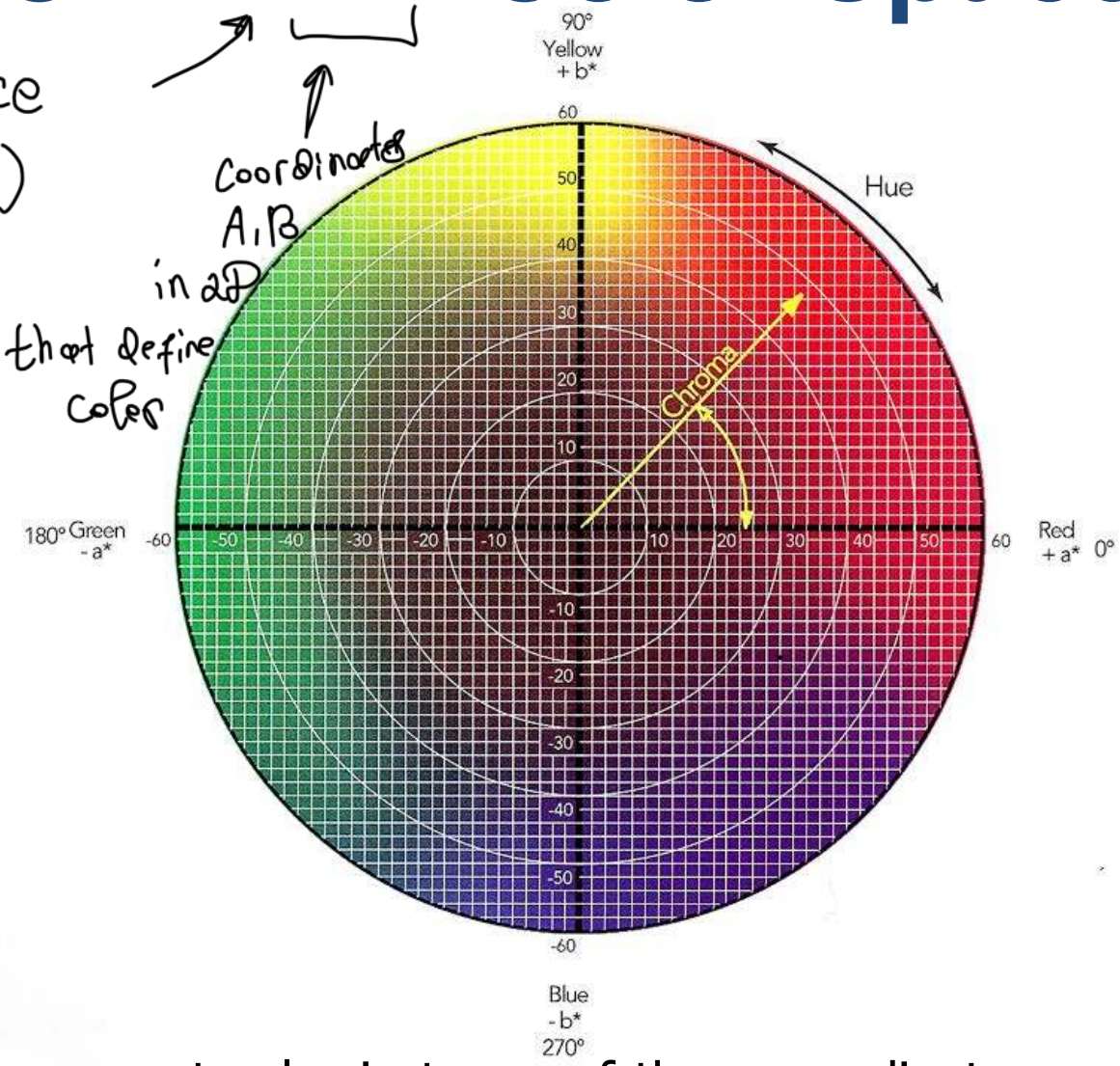
- Skin is saturated
- Lips is even more saturated

Saturation Histogram



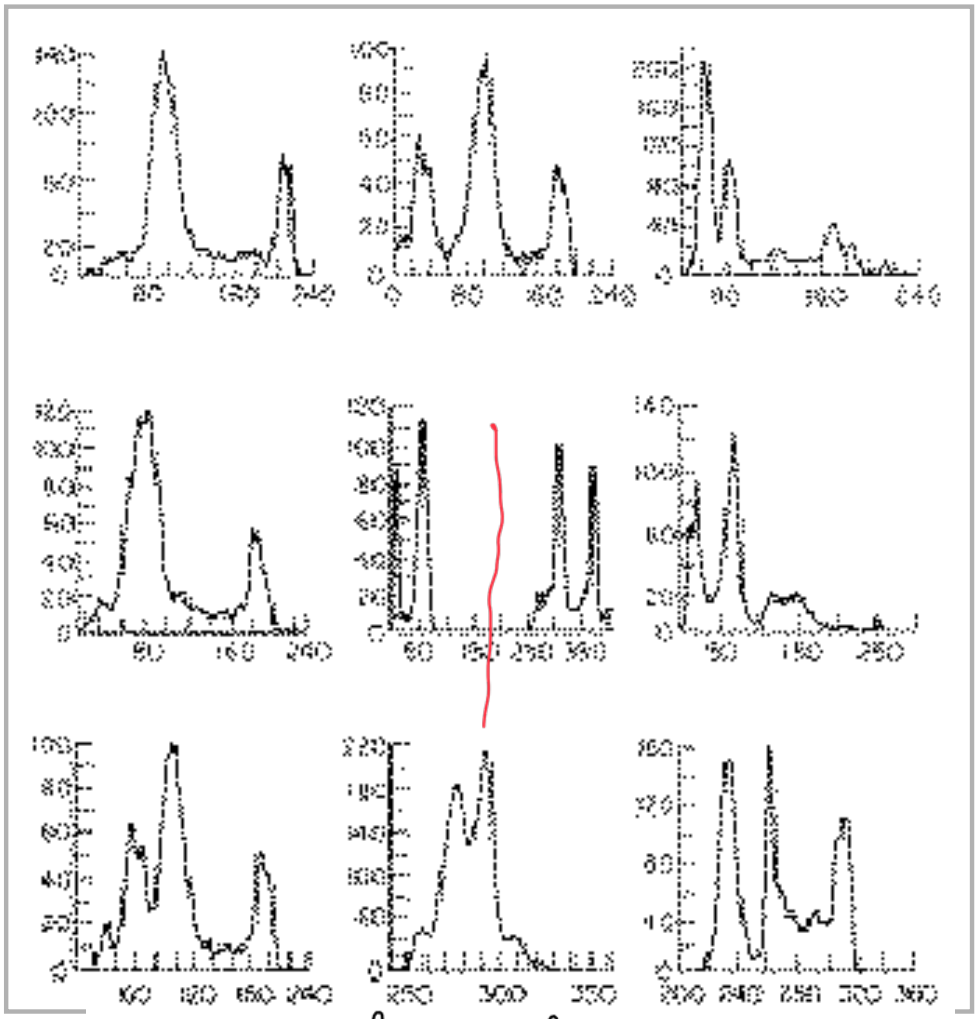
- This histogram is a lot easier to split!
- It makes it easy to segment the head from the background.

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

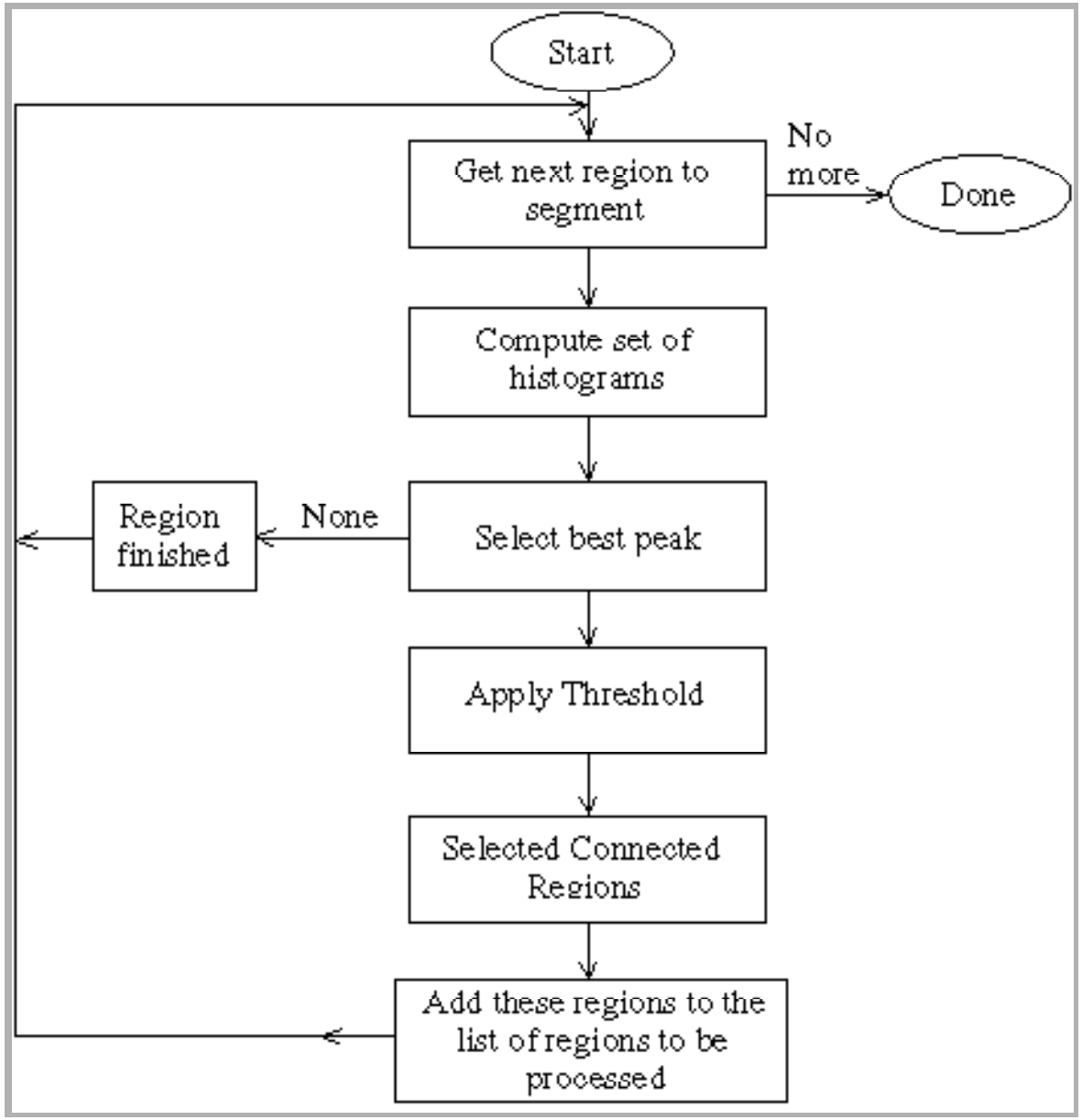


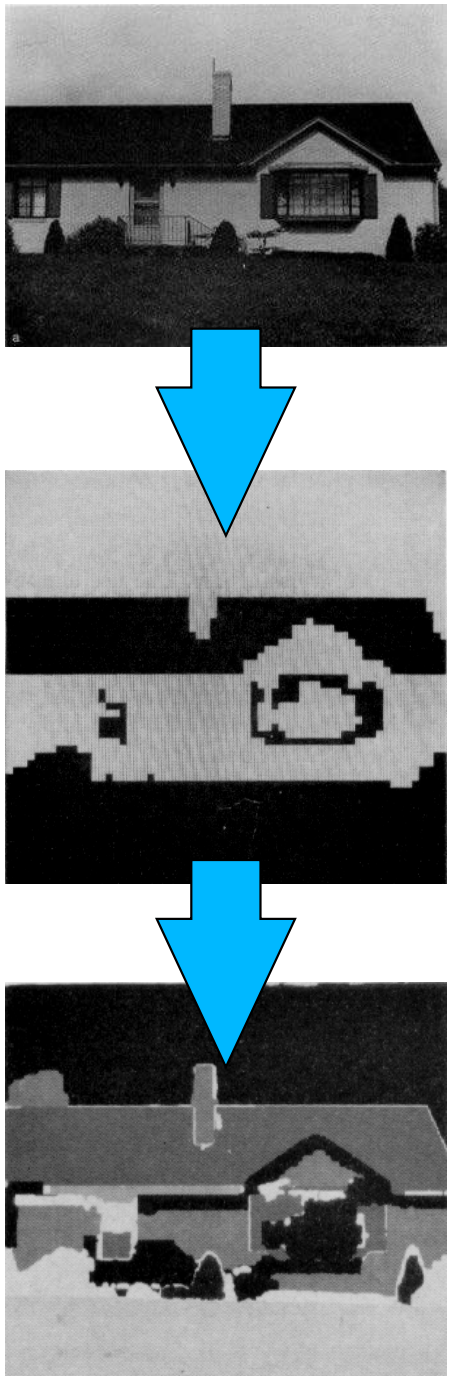
not only traditional but also HSV histograms (or LAB)

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

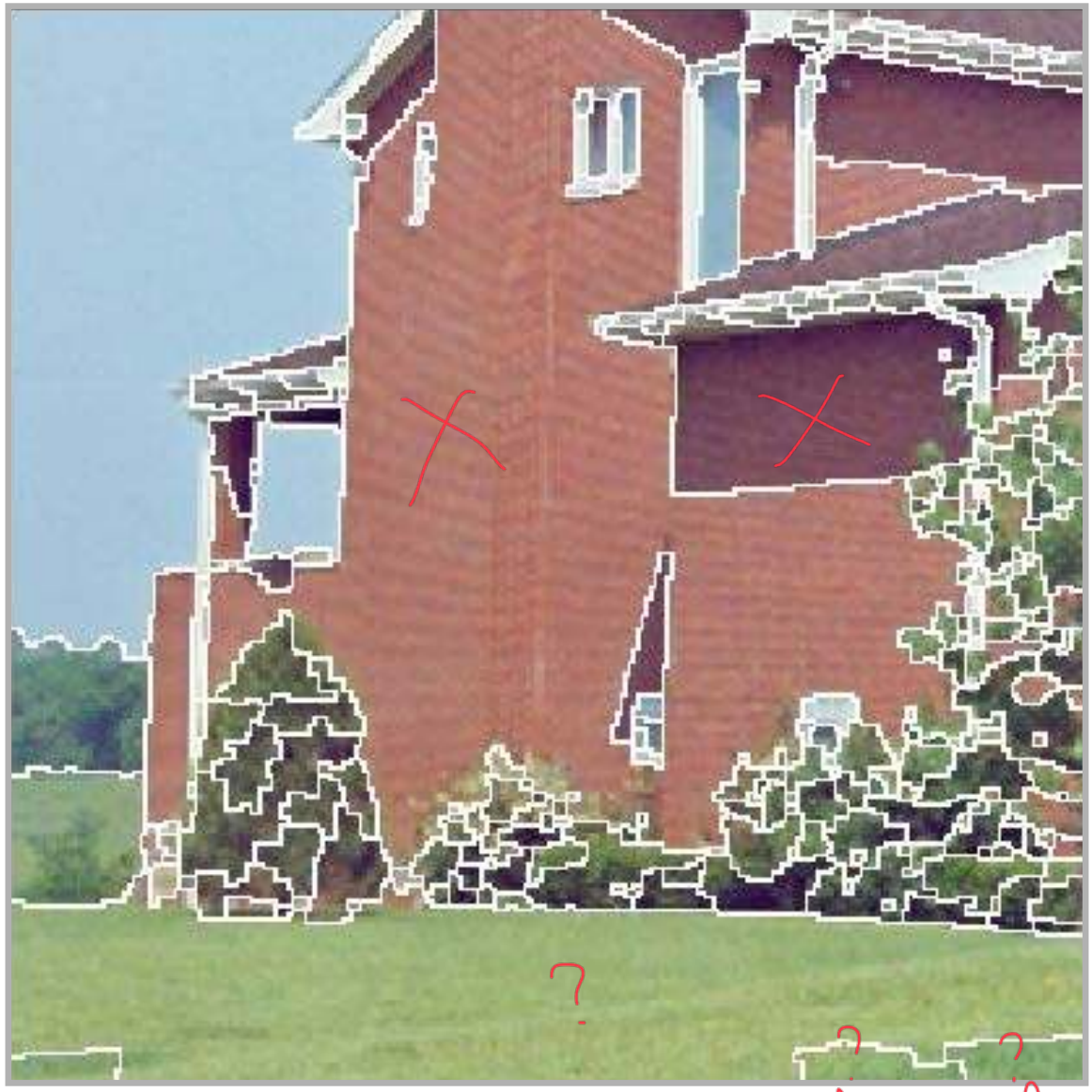
use the one that's easier to segment

Recursive Algorithm





Early Color Segmentation

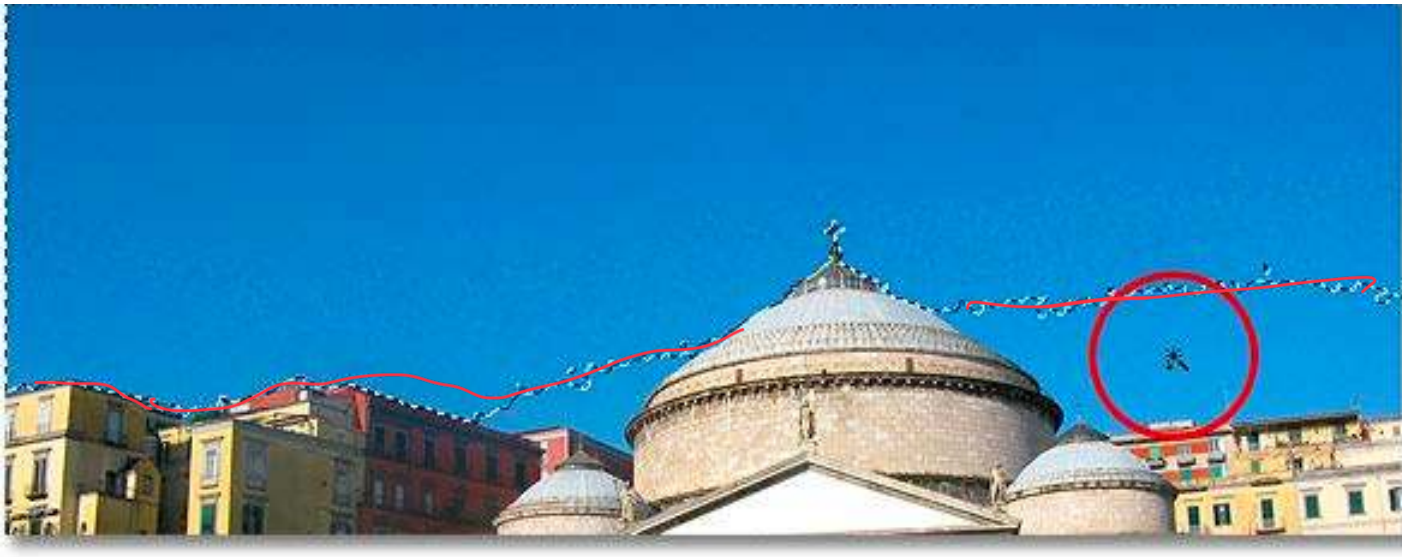


Ohlander & Price, CGIP'78

More Recent Color Segmentation

I find this blue sky too bland!

Replace it.

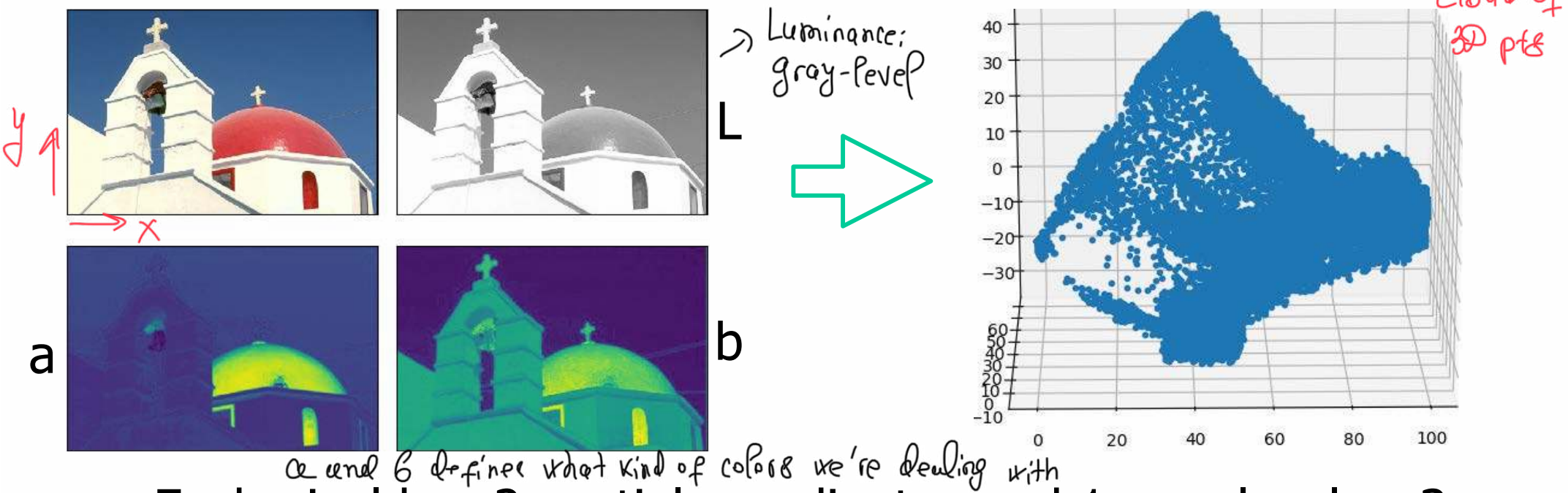


puts boundaries like this

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



a and b define what kind of colors we're dealing with

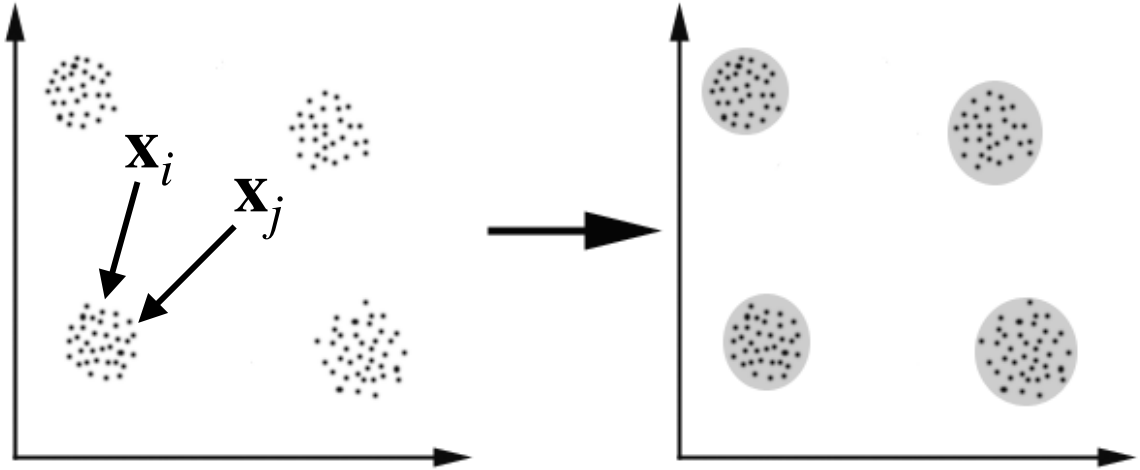
- Each pixel has 2 spatial coordinates and 1 gray level or 3 color components.
- Segmentation can be understood as clustering in
  - 1D space (G);
  - 3D space (x,y,G), (H,S,V), (L,a,b);
  - 5D space (x,y,L,a,b).
- Ideally each cluster should be as compact as possible.

(5)

The main problem with histogram splitting → it completely forgets where the pixels come from

→ complete description of pixel

K-Means Clustering

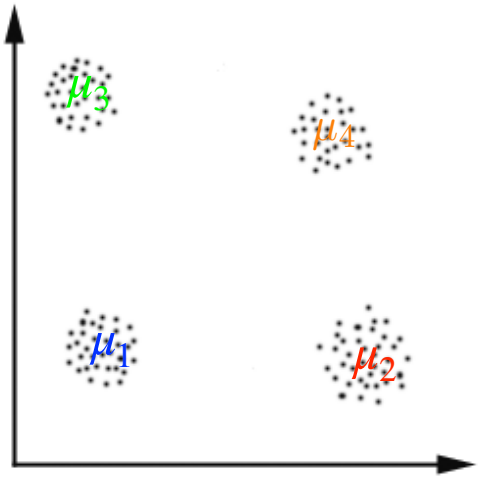


in n-dimension

Given a set of input samples:

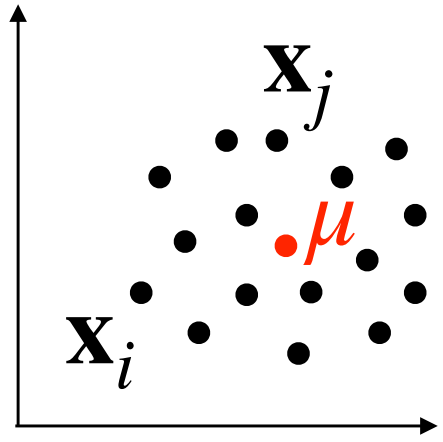
- Group the samples into K clusters.
- K is assumed to be known/given. priori
- In the example above, each sample is a point in 2D.
- In images, samples can be points in 1D, 3D, or 5D.

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

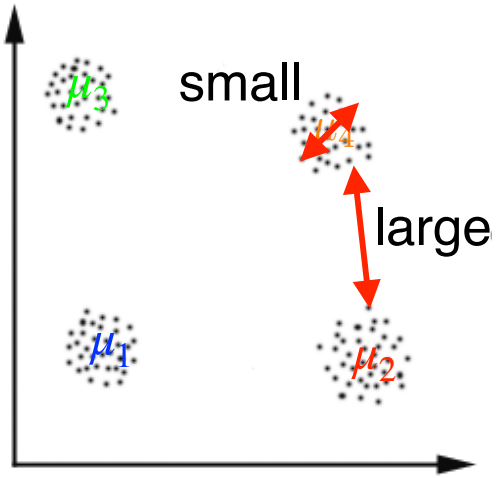


In 2D

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

- If the $\mathbf{x}_i$ were physical points of equal mass, μ 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.

intra distances of the clusters → small

- 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 $\{\mu_1, \ldots, \mu_K\}$:

k-means

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

we want to find

sum over all clusters

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

i) We want to find cluster centers

ii) Assignments to clusters of points to minimize J

Minimize

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

but:

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

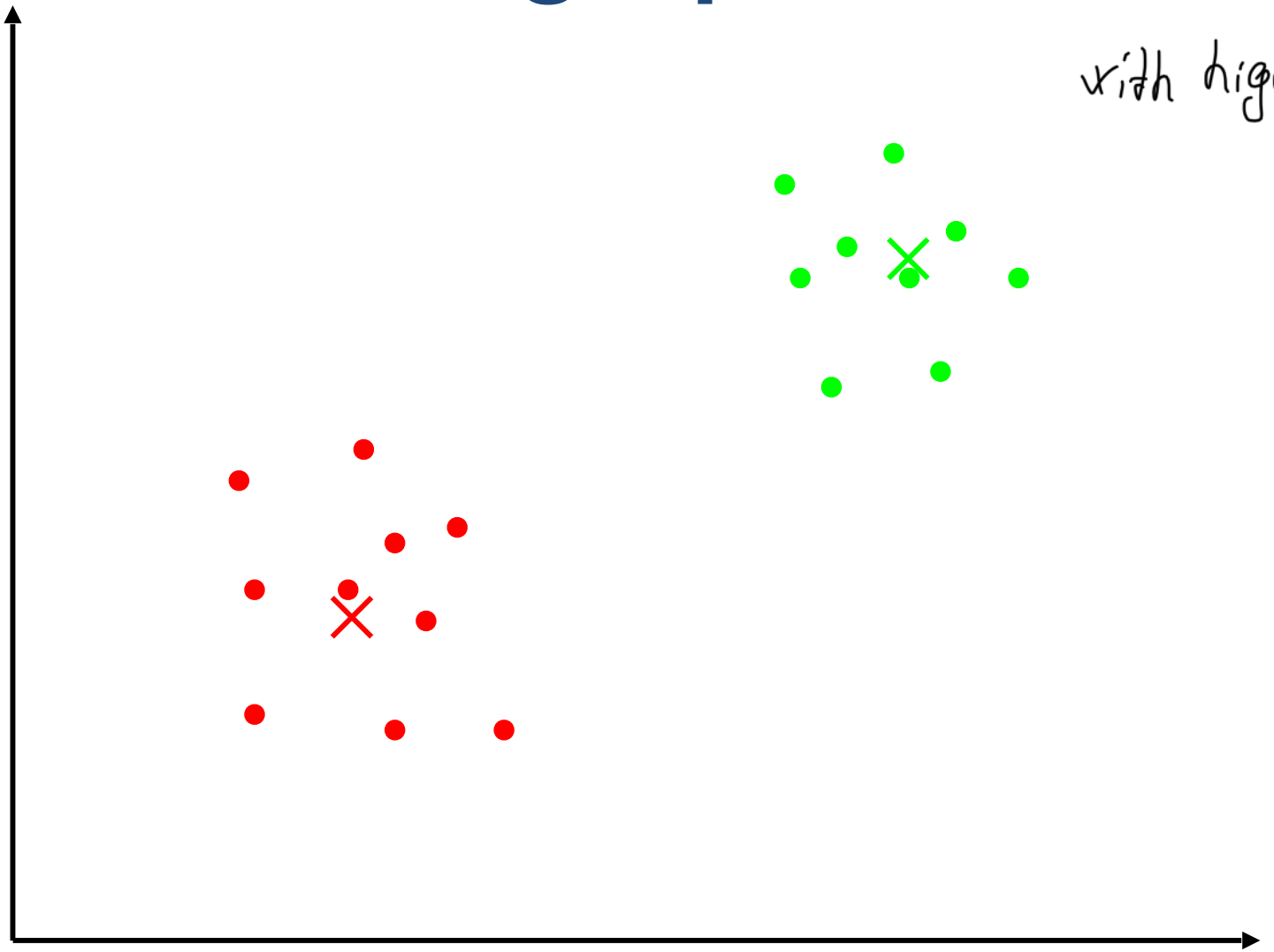
We don't know μ's

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

By computing mean of these pts

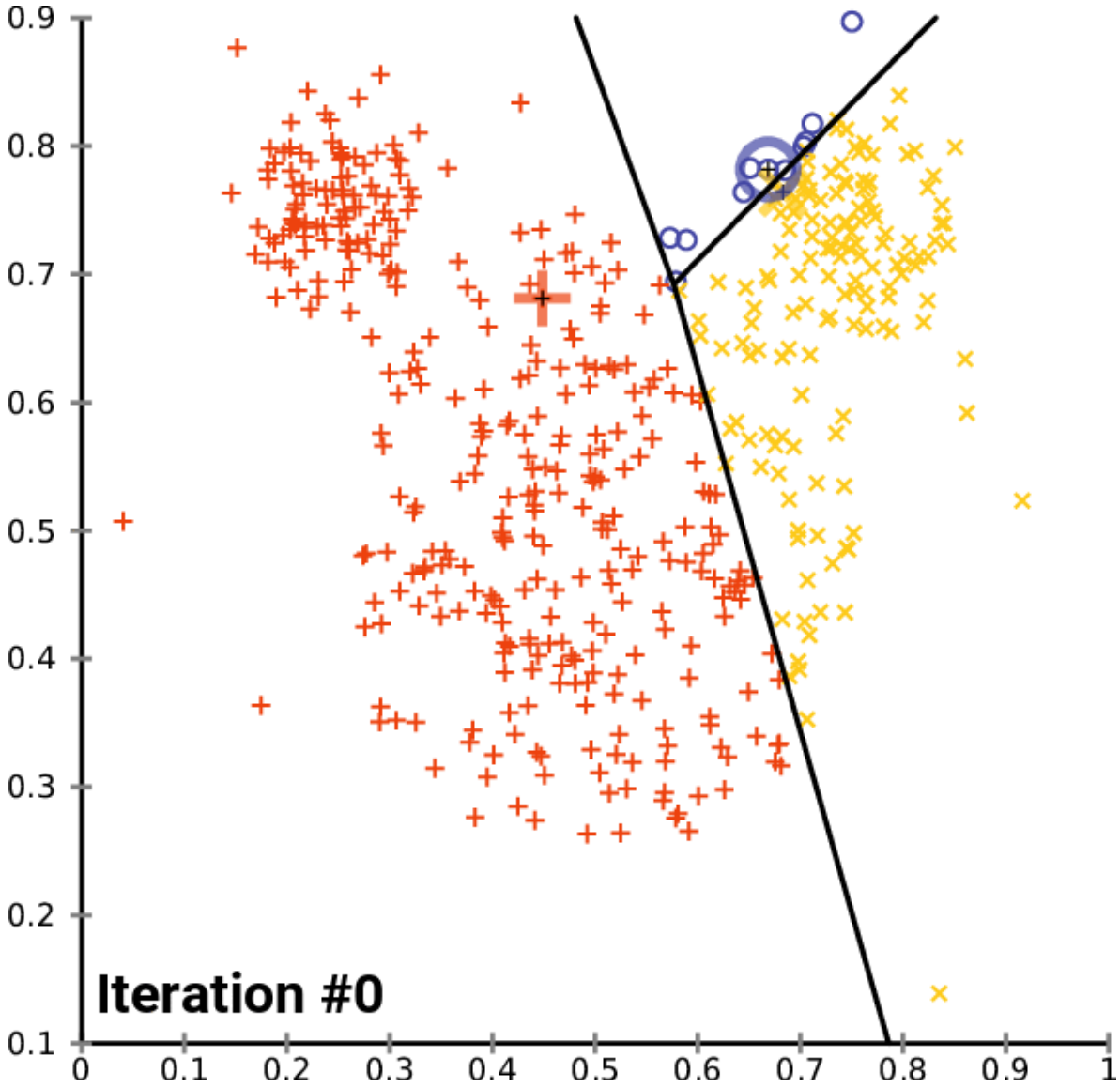
Alternating Optimization



Demo

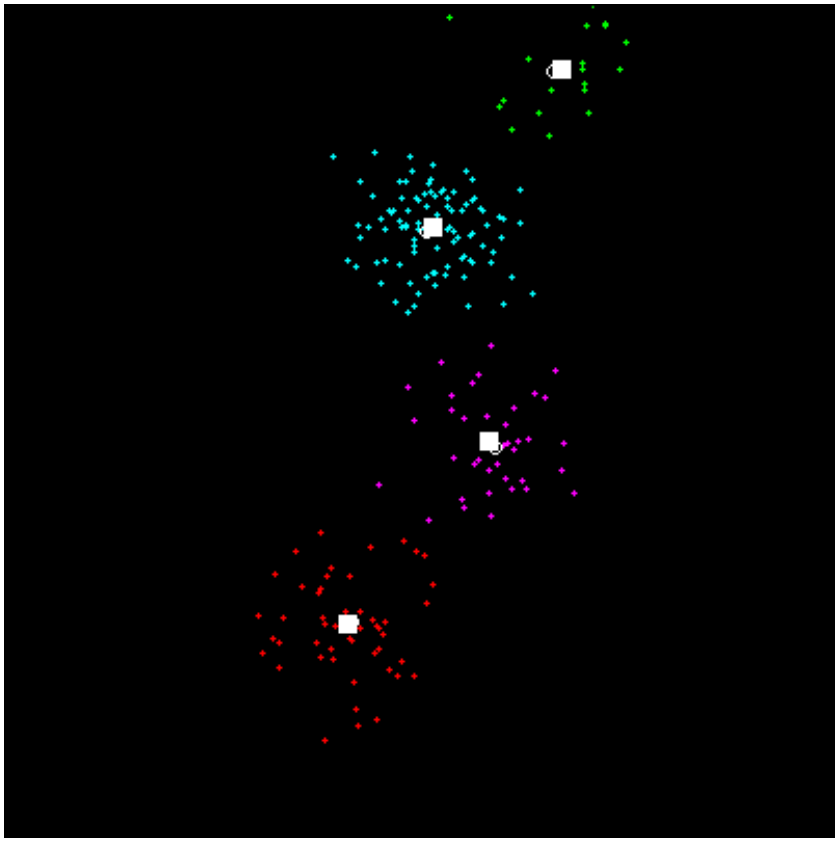
- Initialize
- Associate point to centers
- Recompute centers

Three Classes

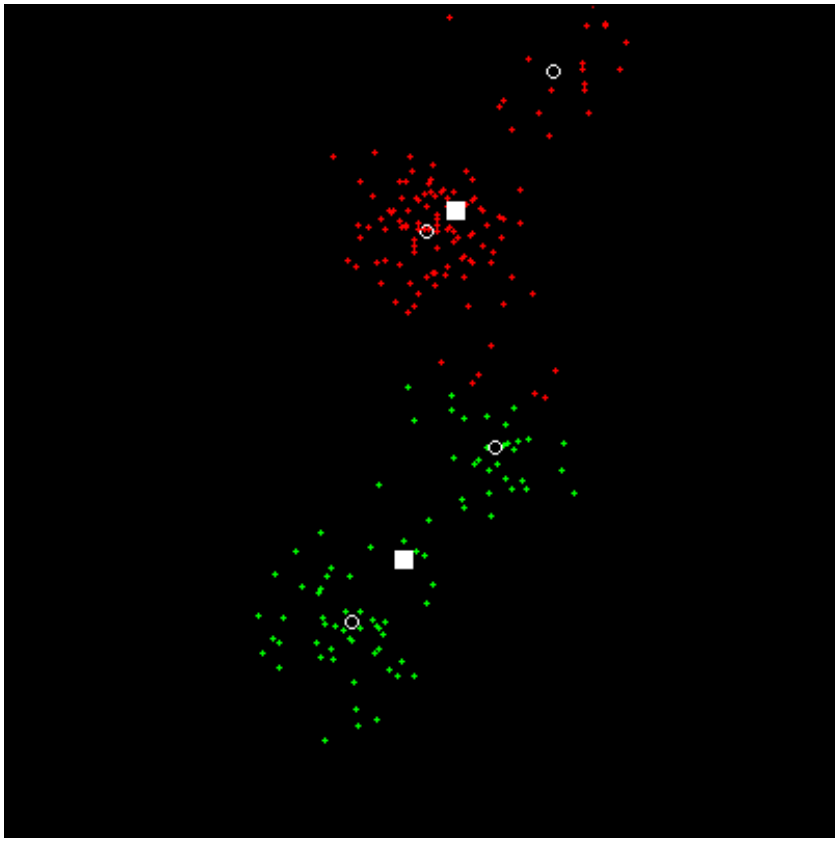


Demo

Initial Conditions Matter



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

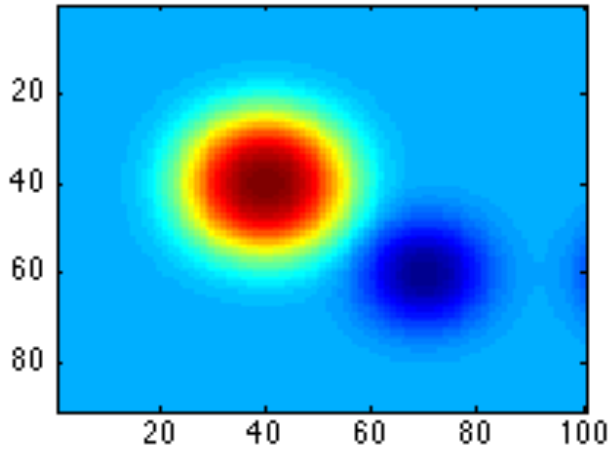


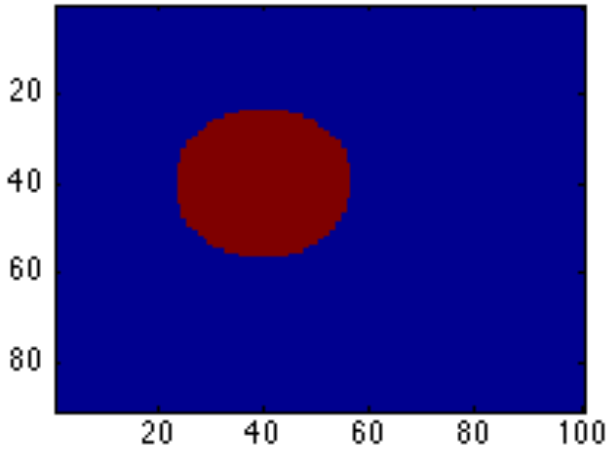
Initially, the points are assigned to the closest cluster —> Failure.

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

our criteria

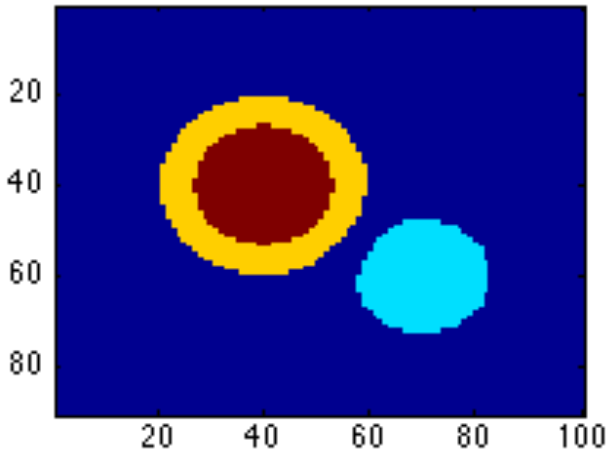
GRAY-LEVEL ONLY (1D)





K=2

K=3



K=4

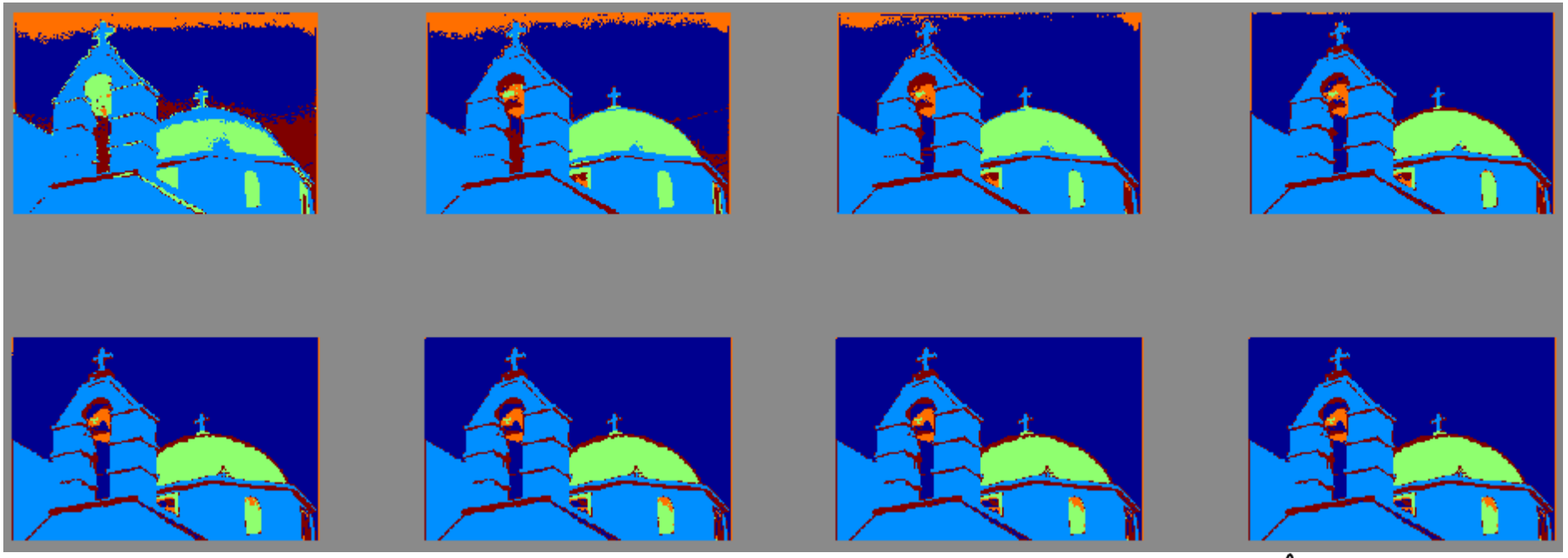
K → # of clusters

K-means worked!

For this picture, histogram splitting failed!

Color Only (3D)

LAB-space



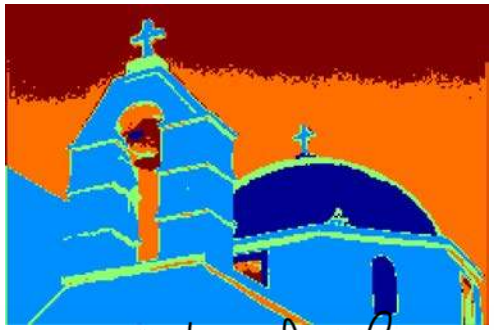
dome, roof, walls ✓

8 iterations for k=5

Color Only (3D)

Different Initializations for k=5


good ✓


not wonderful


good ✓

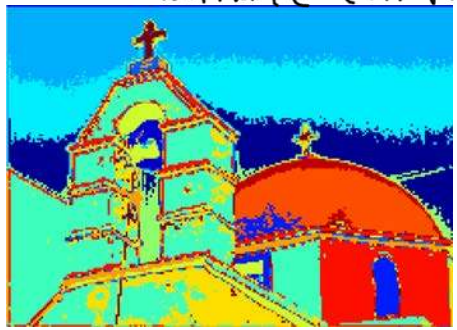

not good
↳ could be eliminated on the basis they don't minimize criteria

Different values of k

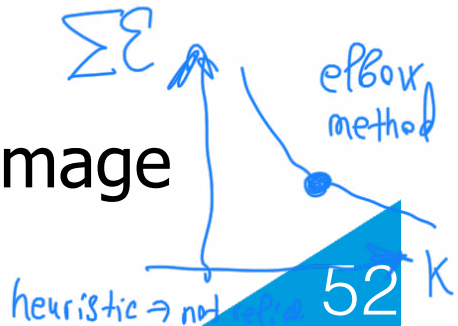

K=3


K=5


K=8
How # clusters?

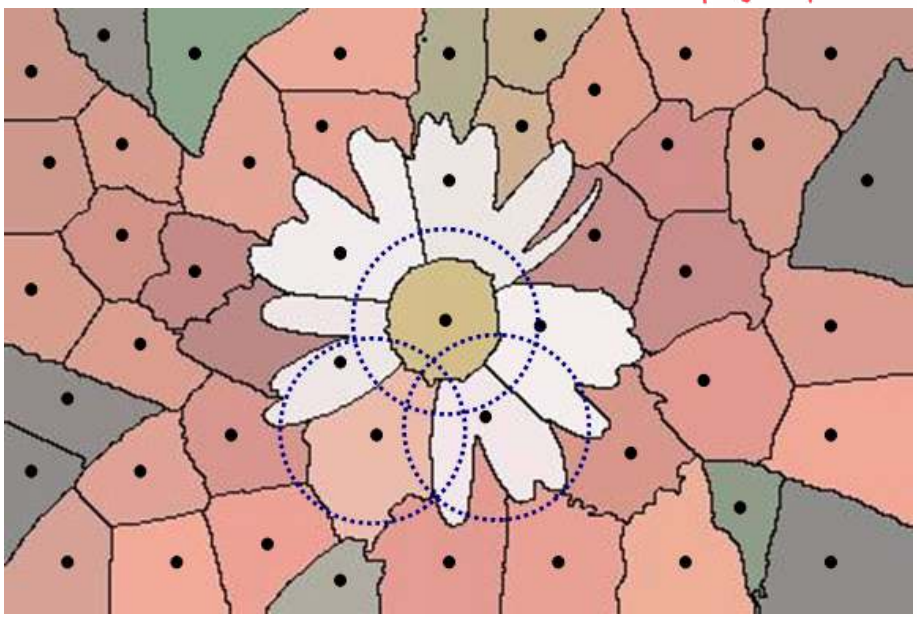

K=15

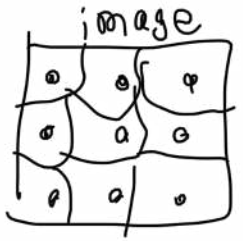
- Different results for different initializations.
- For k well chosen, the results are good for this image because it features bright and distinctive colors.



XY + Color (5D)

Coordinates of pixel





Start with seeds on a regular grid

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

You create 1 pixel for ∀ cluster

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

color → Intensity

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

cluster center

different weights

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.

→ Snooth region centered around seed

—> Superpixels

Achanta et al. PAMI'12

SLIC Superpixels

1024x1024

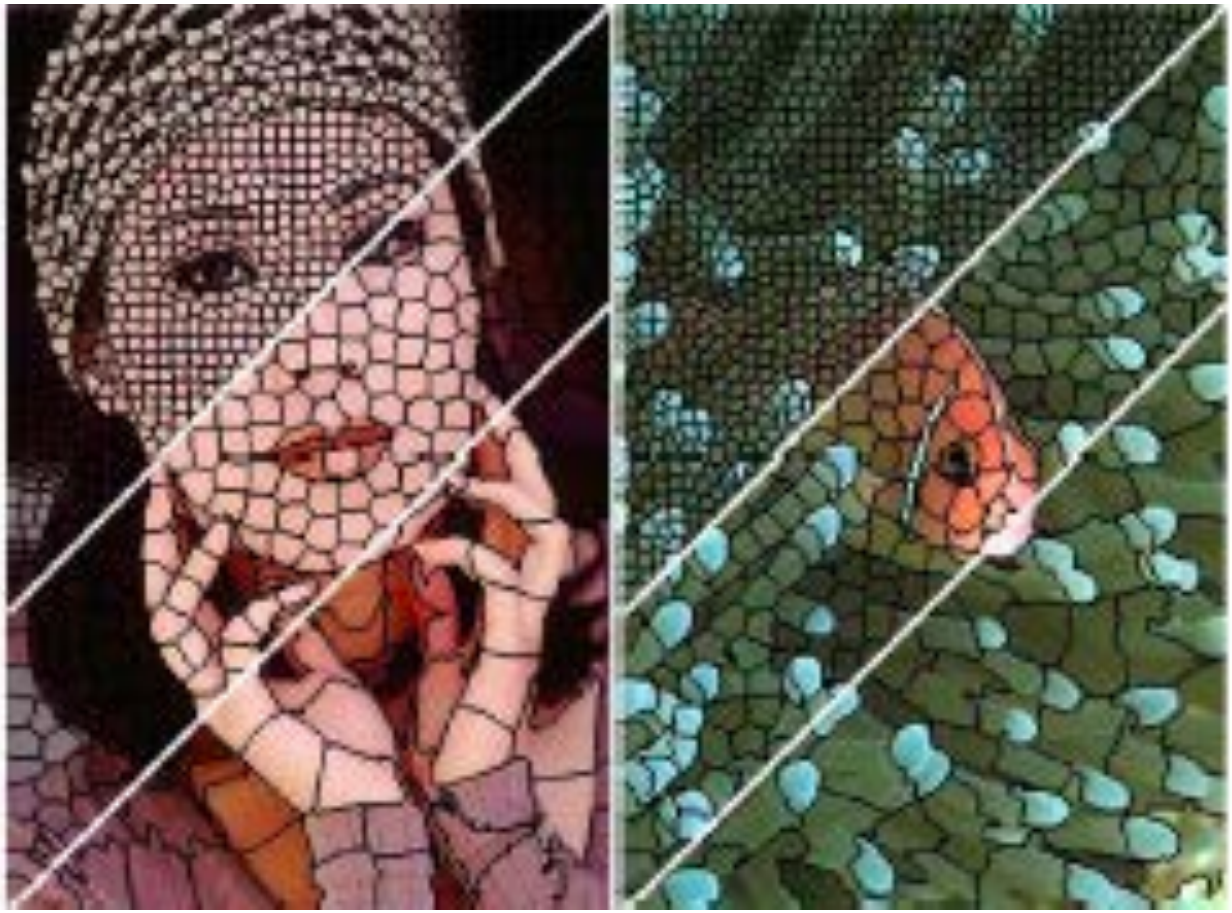
fewer seeds

↓

Bigger regions

Superpixel segmentation

256x256



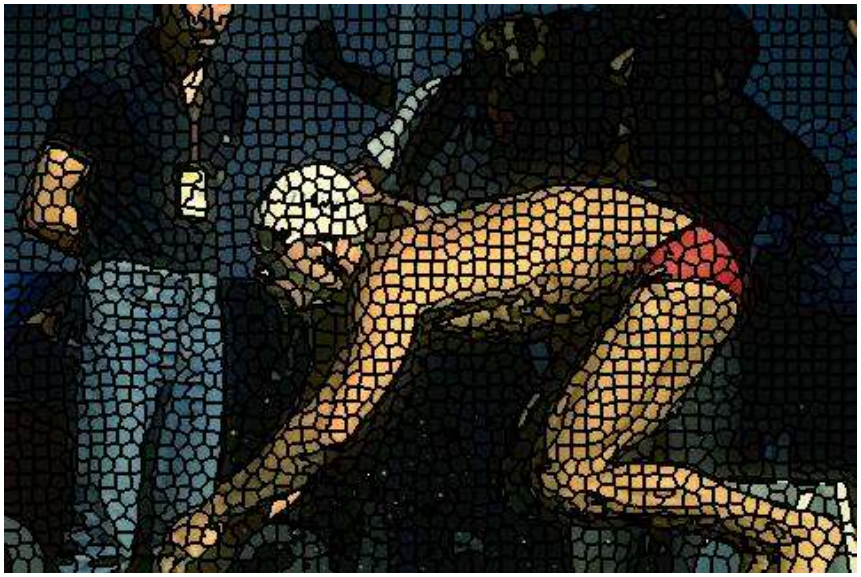
256x256

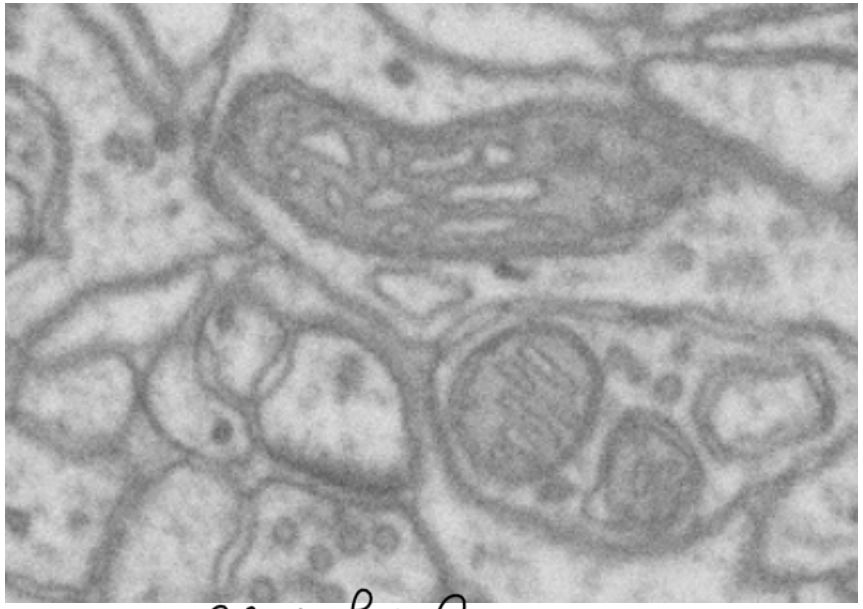
Think about grass issue!

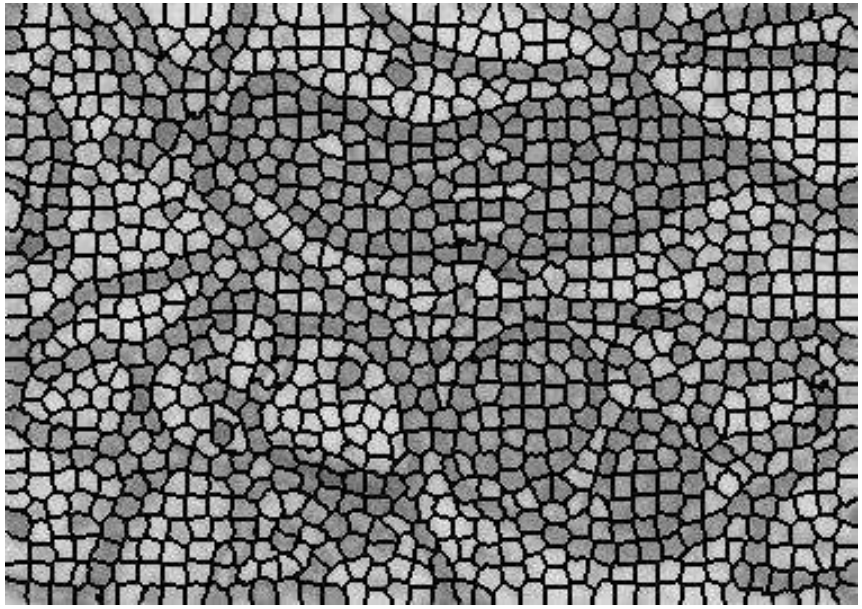
64x64

- 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. ⇝ on which you can compute statistics (Local)

Color or Black and White



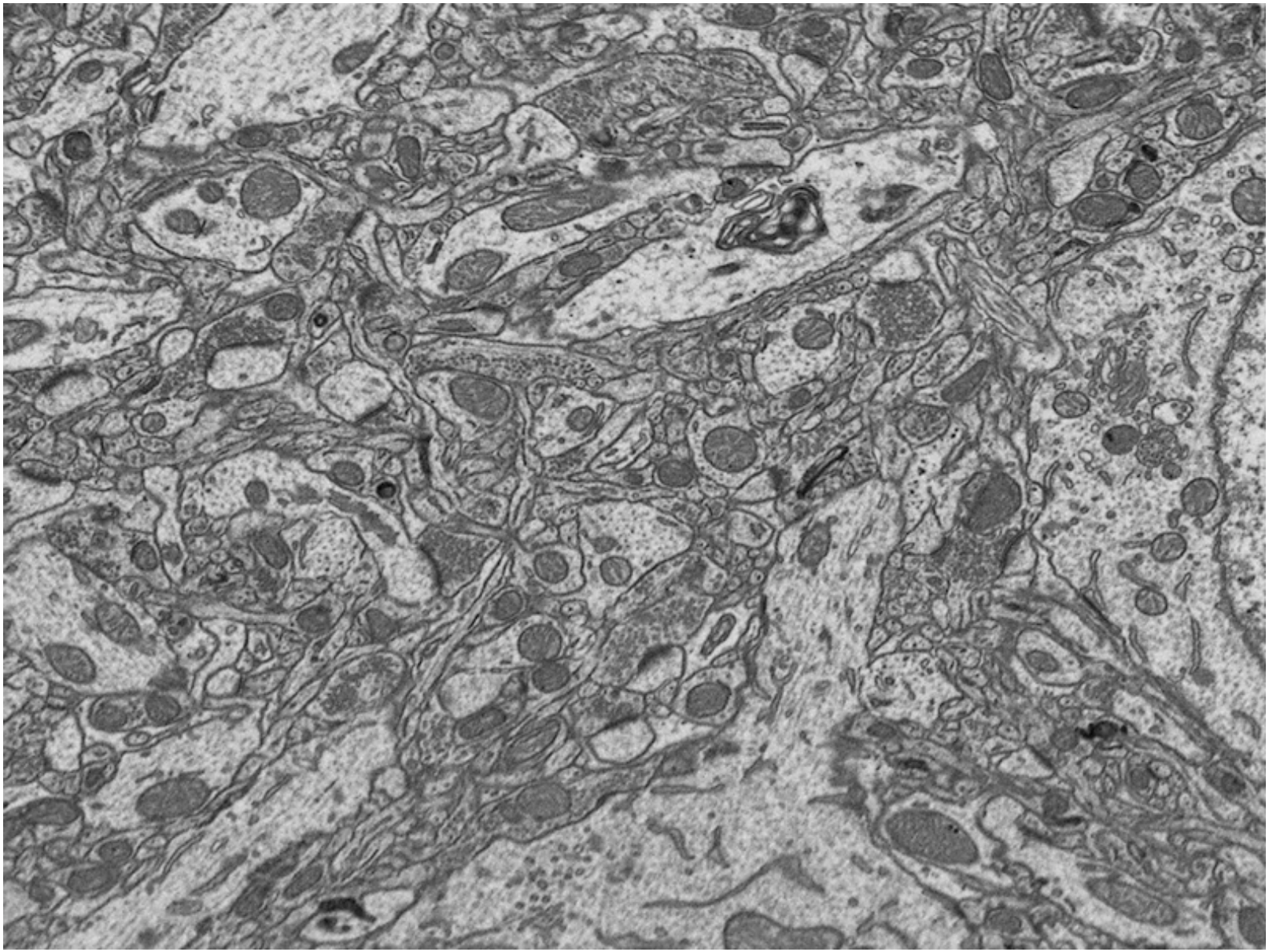




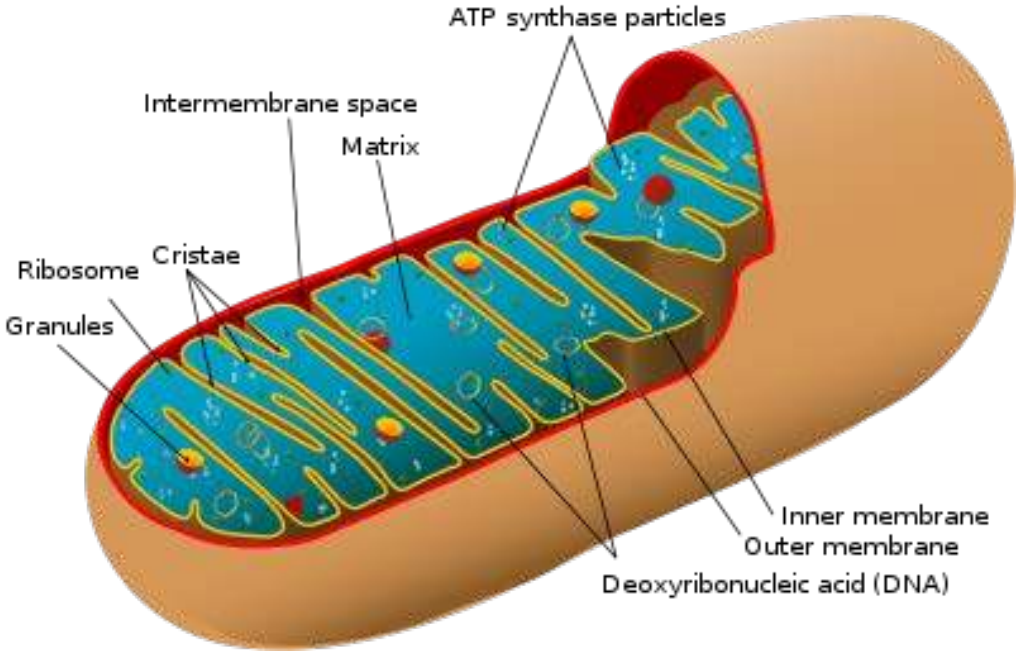
gray - level

It works in both cases.

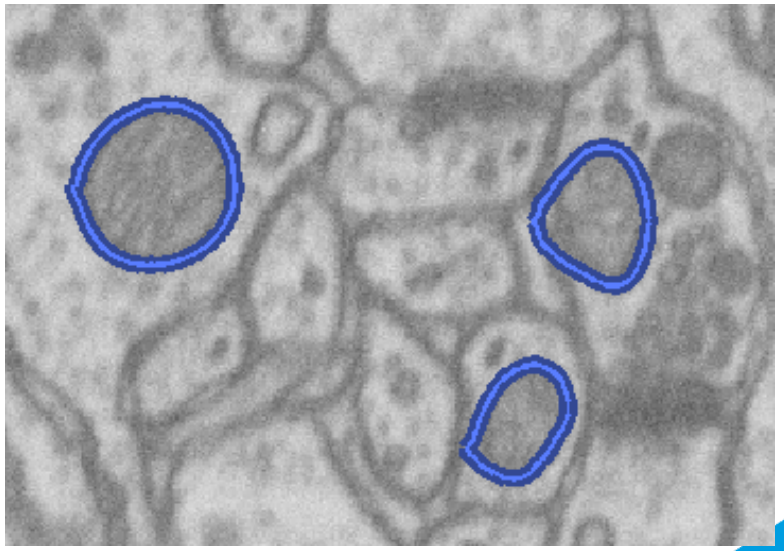
Electron Microscopy



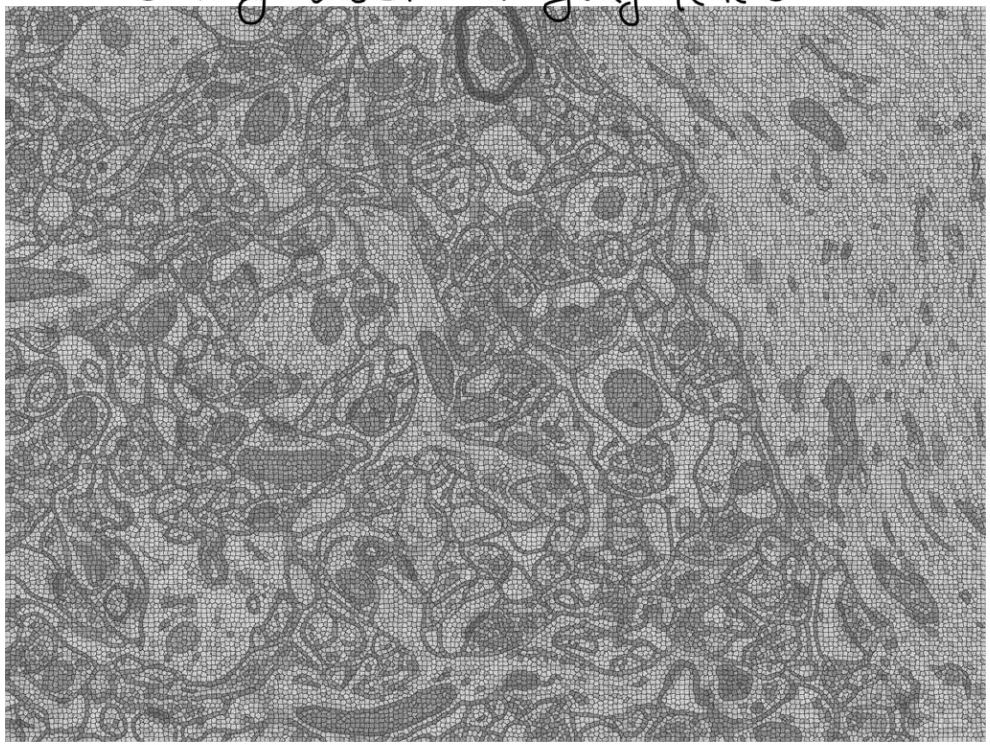
Mitochondria Segmentation

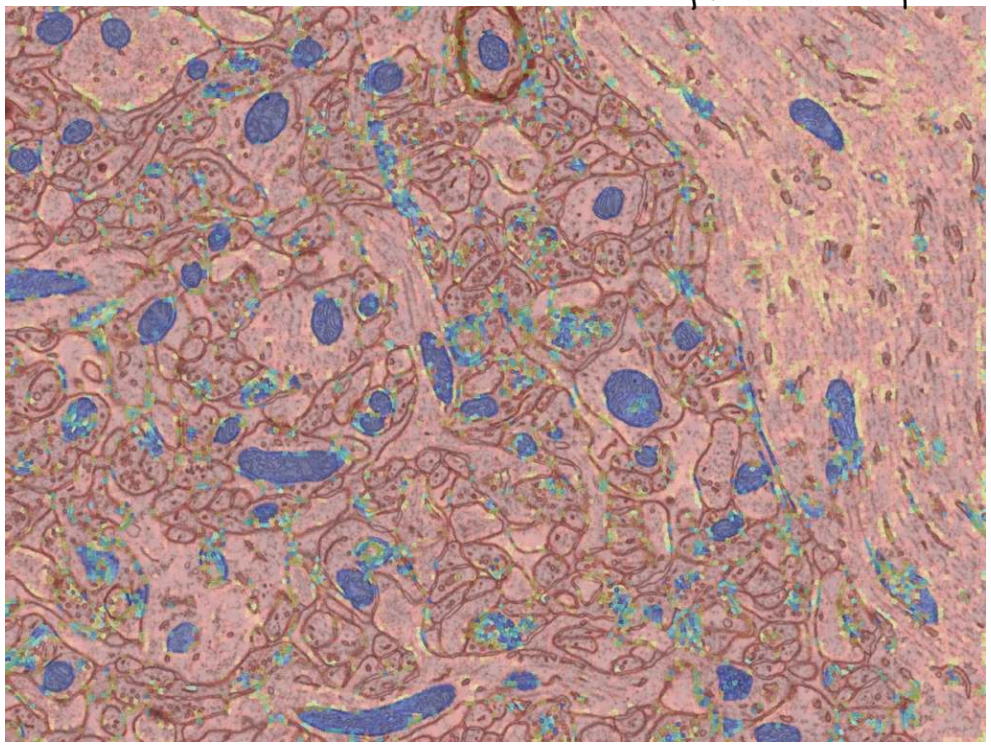


and find boundaries
slice of mitochondria



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.

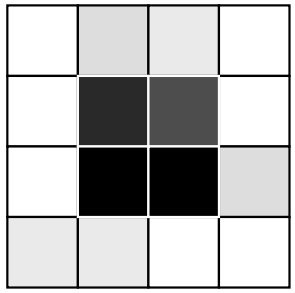
based on image statistics

red → unlikely to be a mitochondria

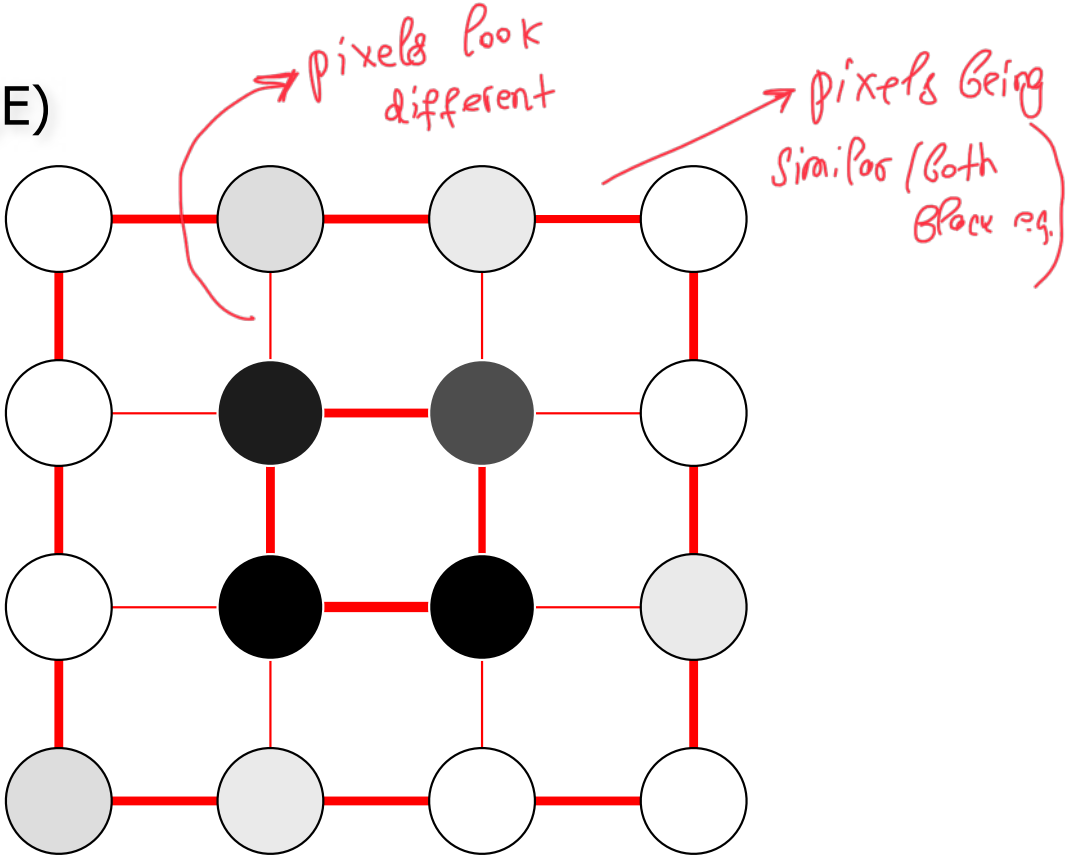
blue → likely

Images as Graphs

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



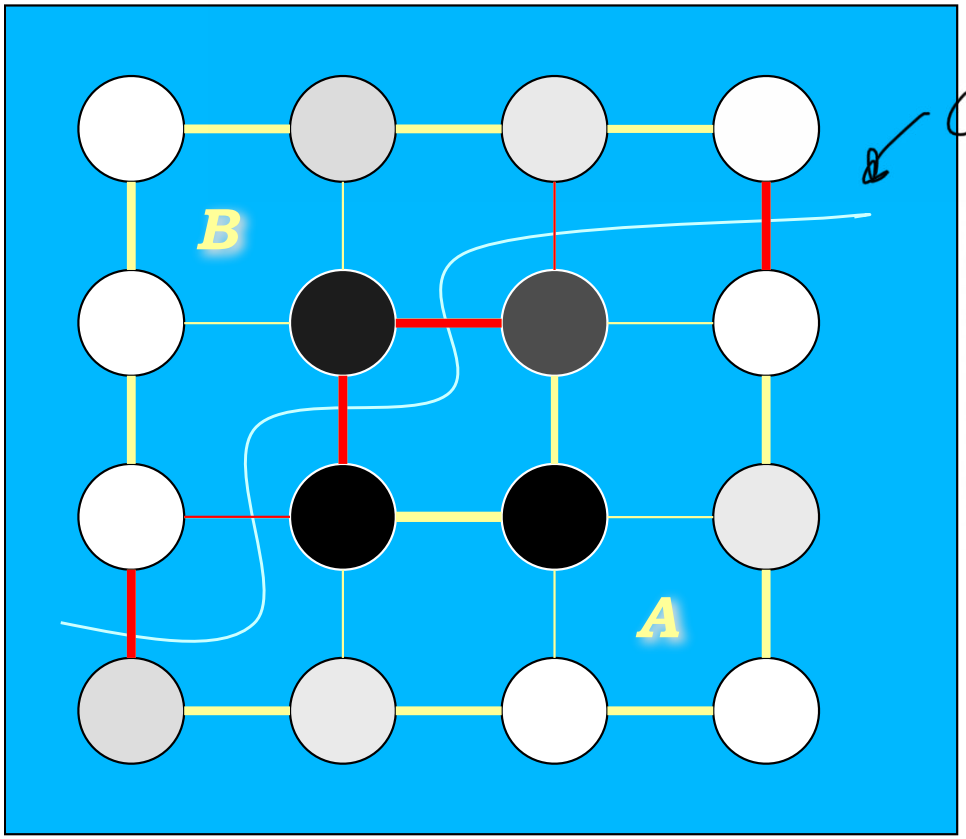
Original image I(x,y)



- 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

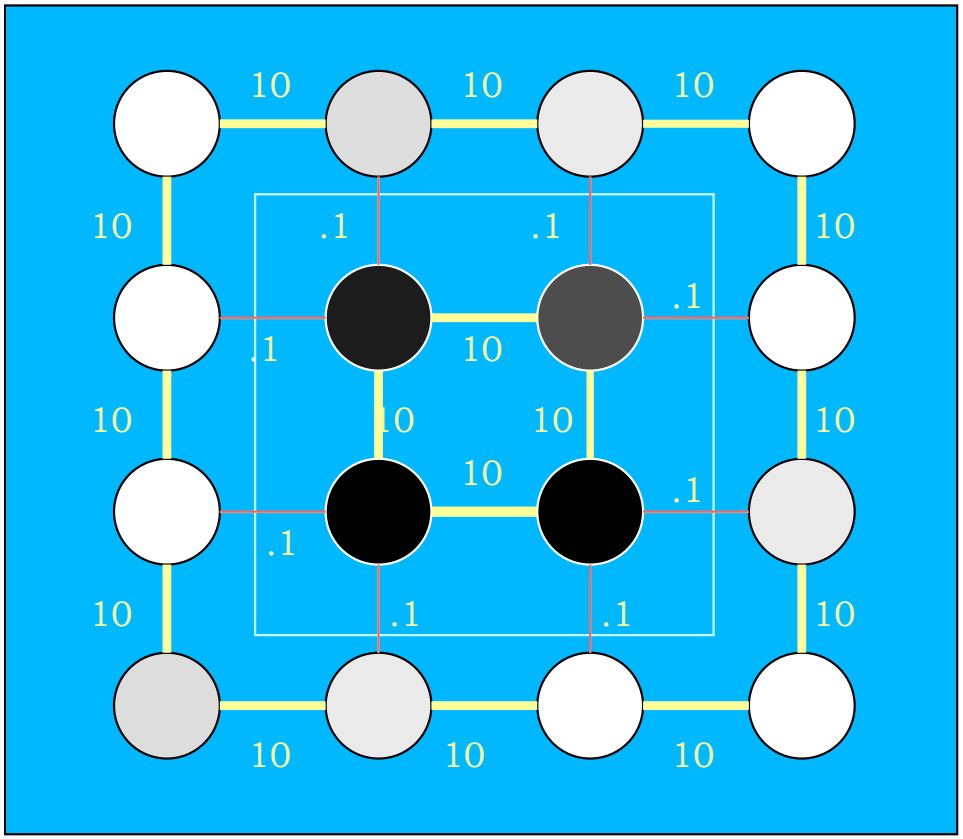
Segment given image into 2 regions



cuts graph into 2 independented connected components

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



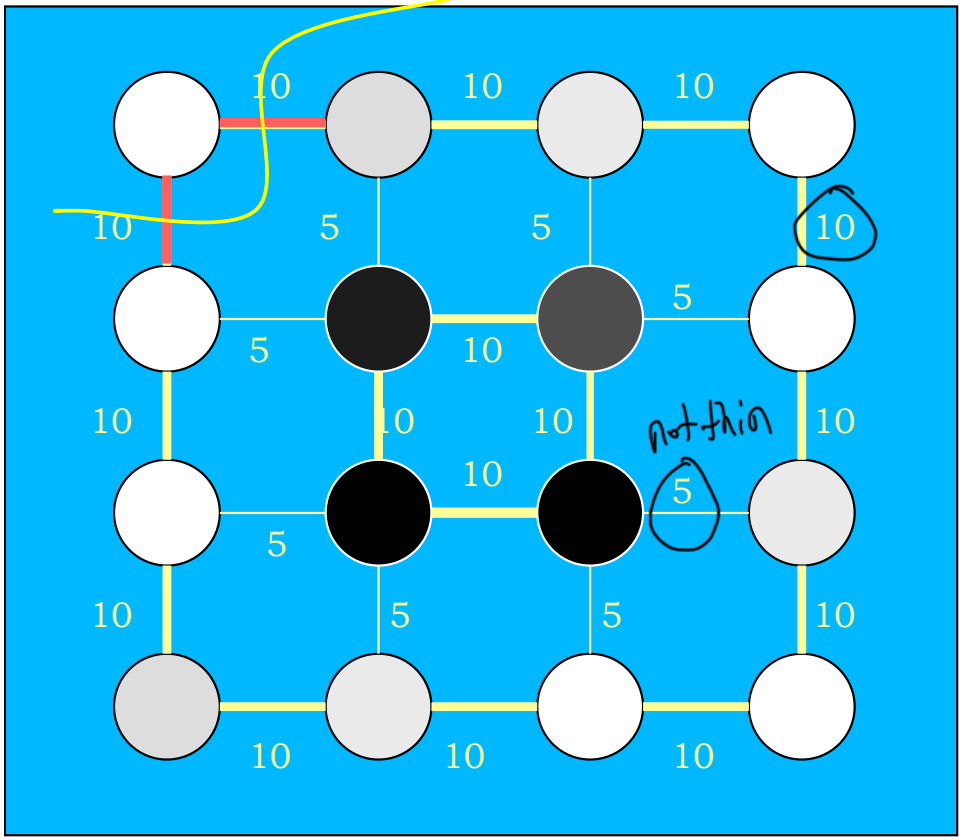
- Find the cut through the graph that has the overall minimum weight, which can be done effectively. → cut edges that are thinnest

- 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.

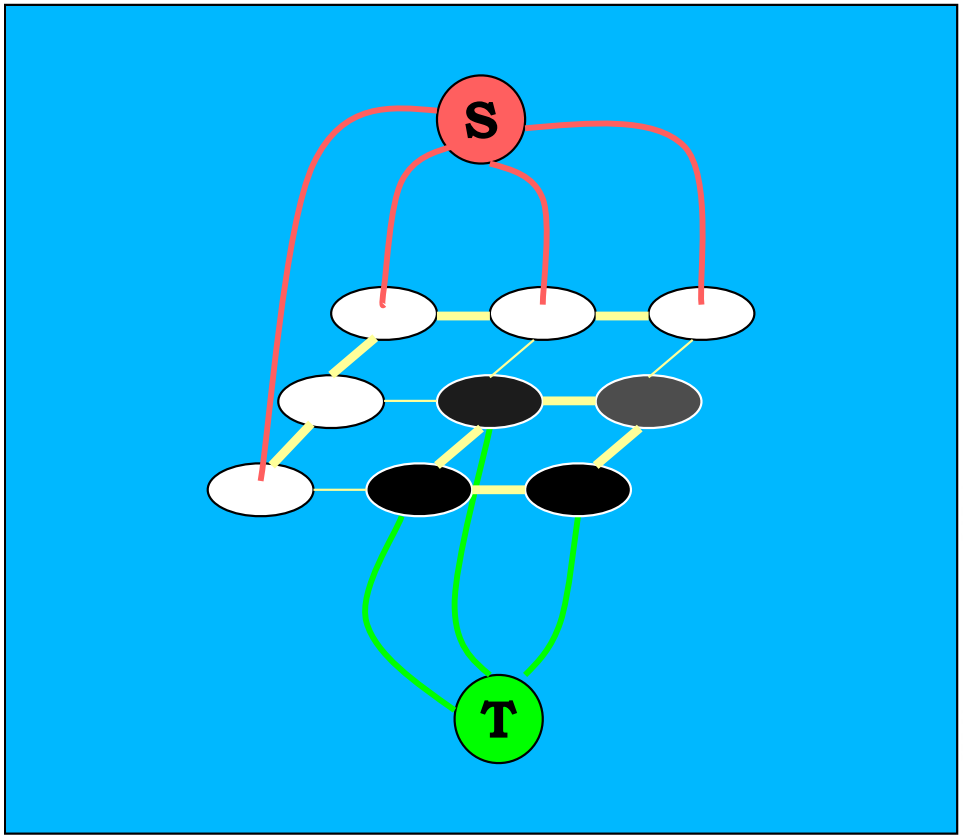
(thin edges)

Trivial Cut



- Has a preference for short cuts, 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.

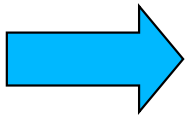
↳ will avoid trivial cut

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

Interactive ST Min-Cut

Human intervention: You had to choose foreground and background pixels

Run graph-cut



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

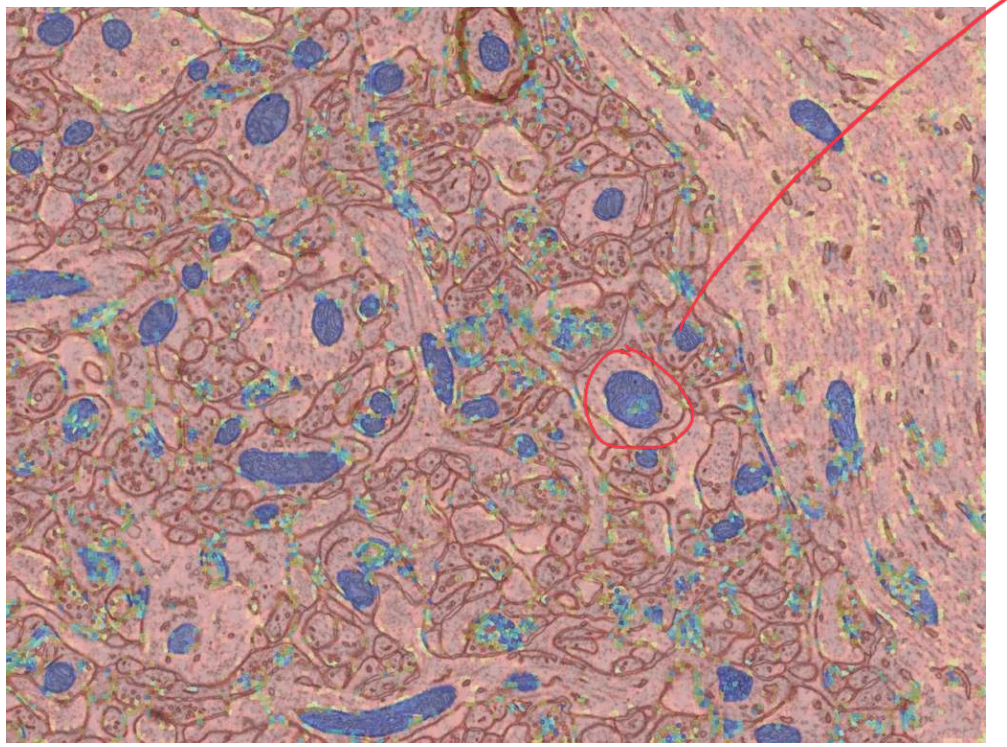
Minimum S-T cut

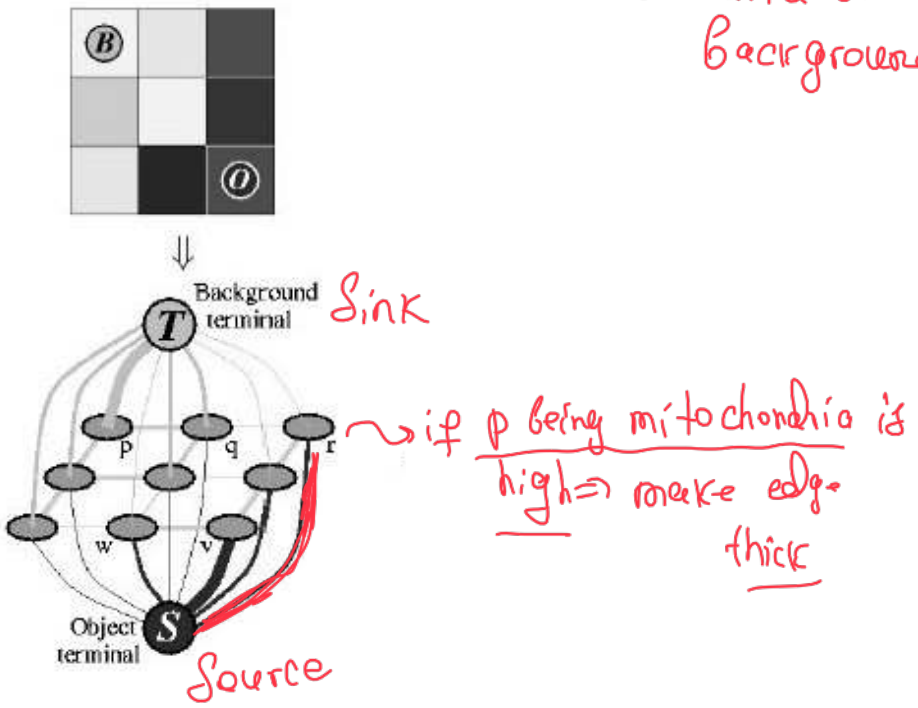
→ background

→ background

--> 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.

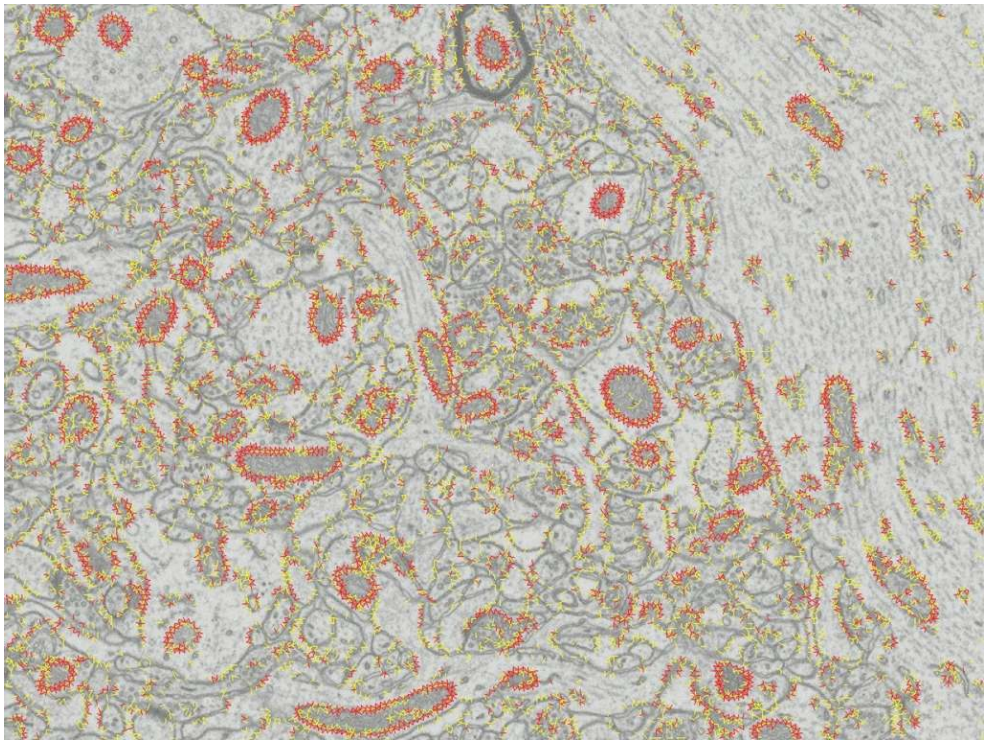
Back To Mitochondria

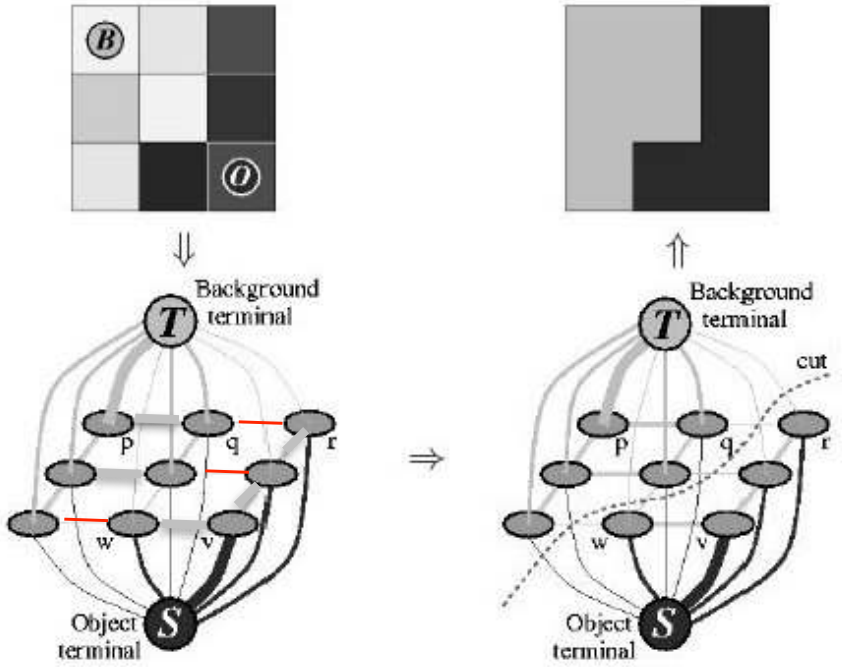




- A high probability of being a mitochondria can be represented by a strong edge connecting a supervoxel to the source and a weak one to the sink.
- And conversely for a low probability.

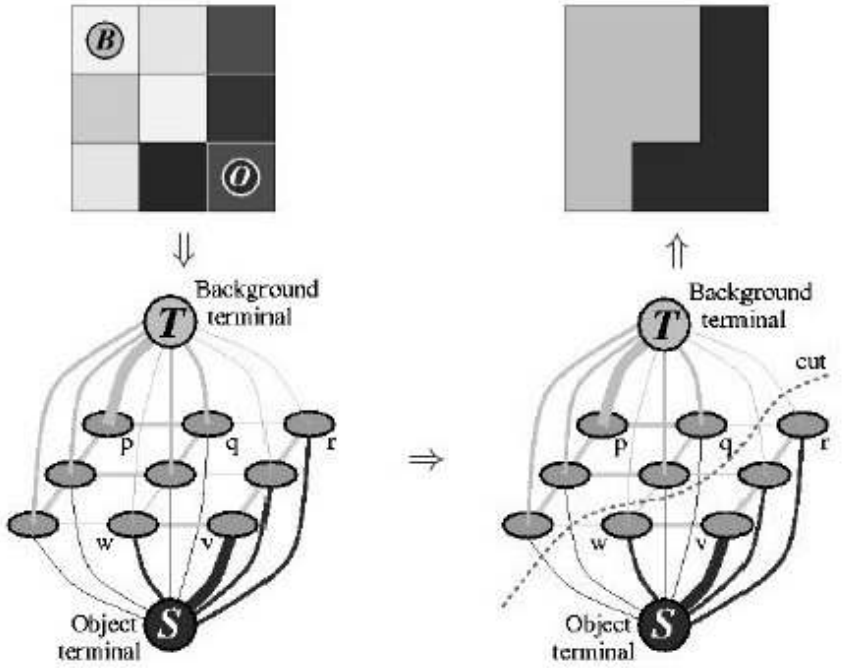
Back To Mitochondria





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



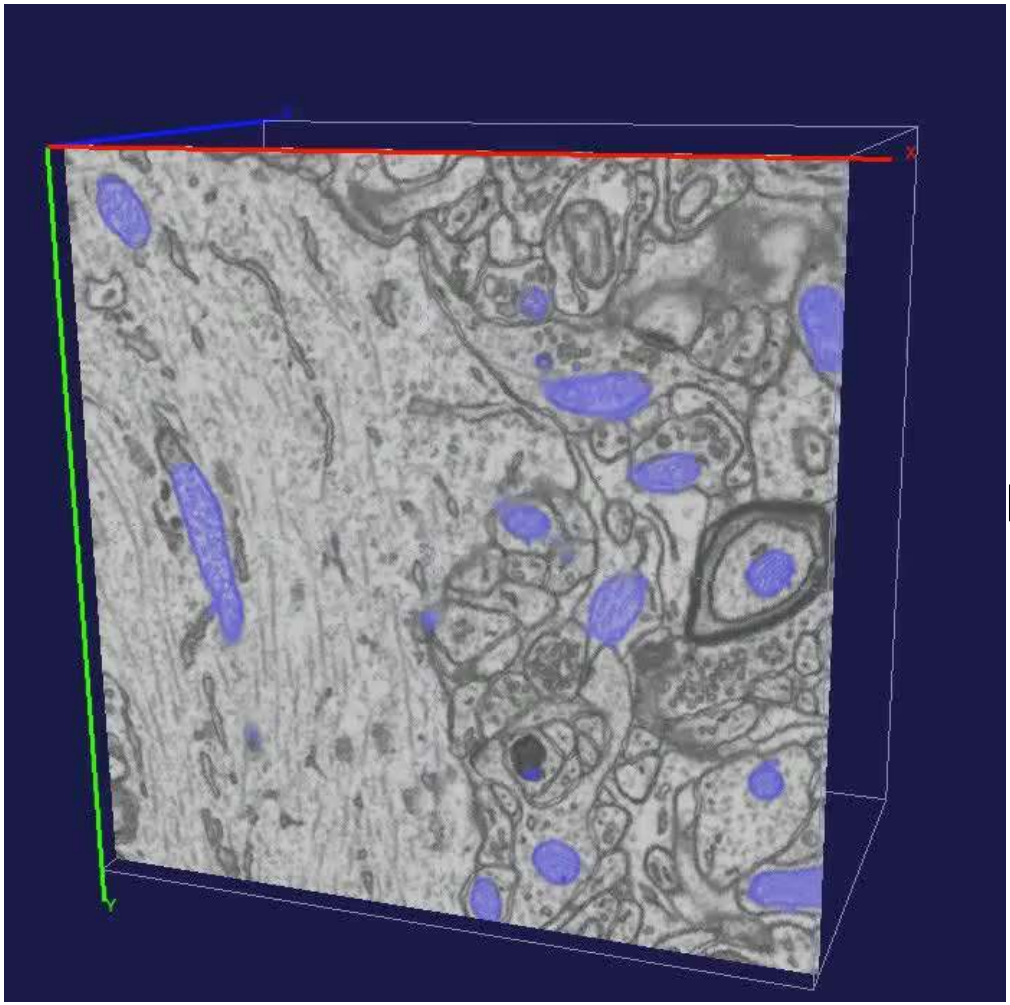
(maximum likelihood)
Solving ML problem
⇓
most likely segmentation given data

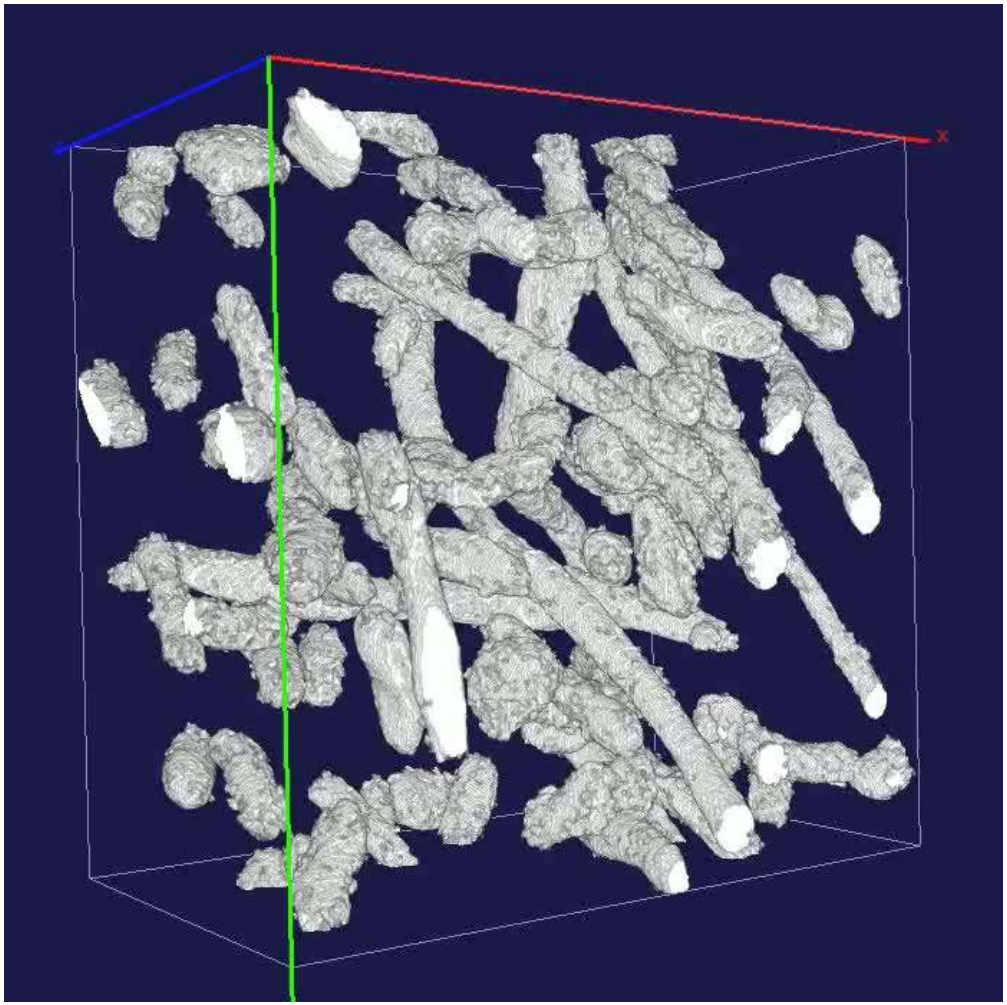
In probabilistic terms, this amounts to minimizing

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

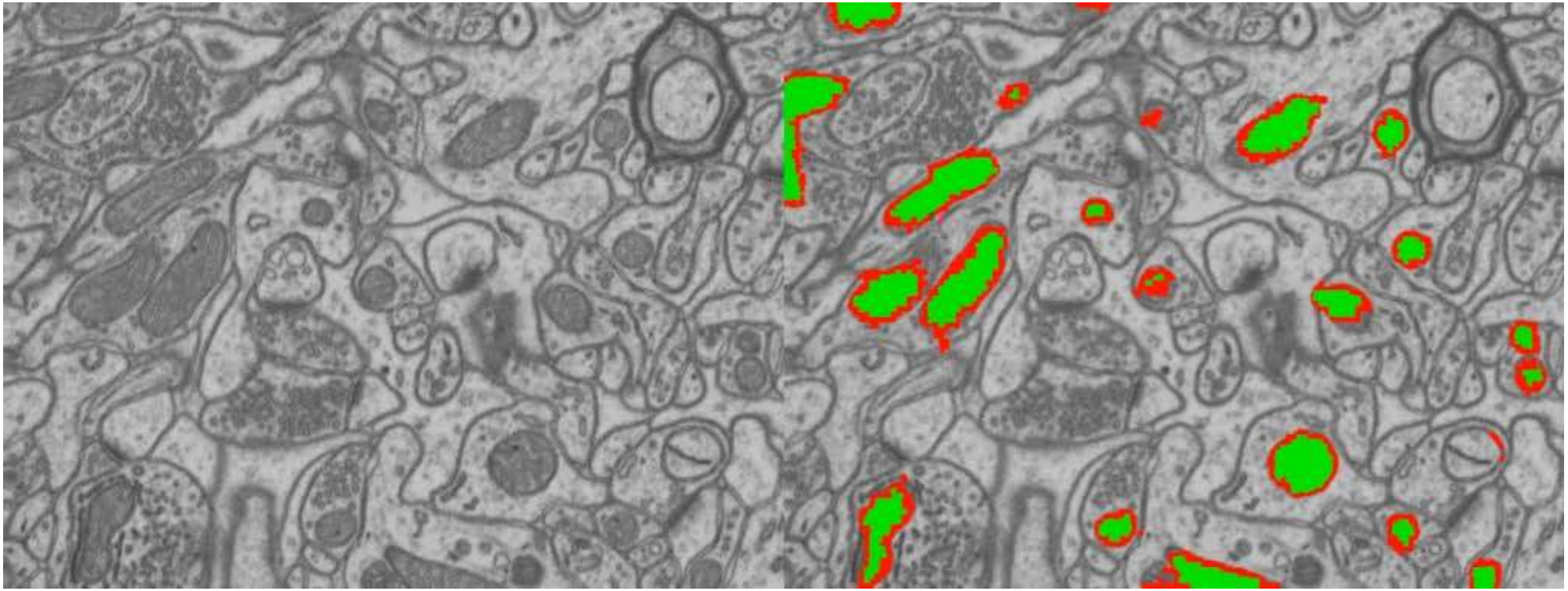
with respect to the set of labels $y = [y_1,\ldots,y_n]$ given the set of supervoxels $[y_1,\ldots,y_n]$, where E is negative log-likelihood of the labels being correct.

3D Mitochondria





Modeling Membranes



Here we use three classes instead of two:

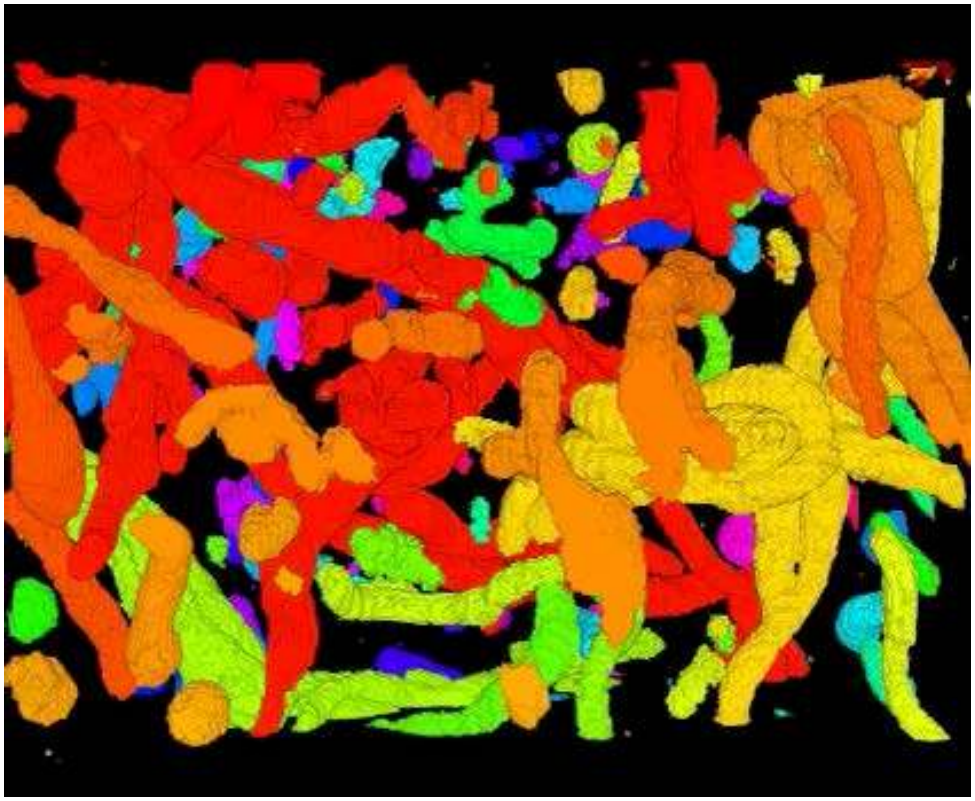
- Inside
- Membrane
- Everything else (Background)

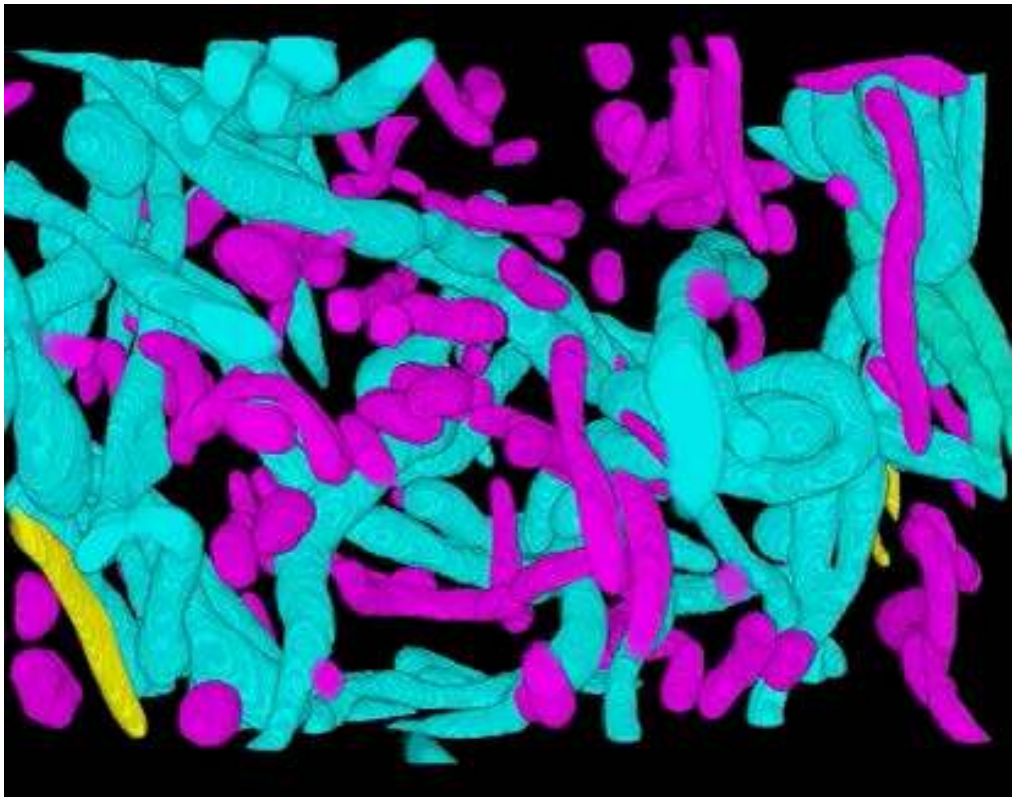
Graph-cut algo works ✓

—>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

Segmenting mitochondria from volumes of images

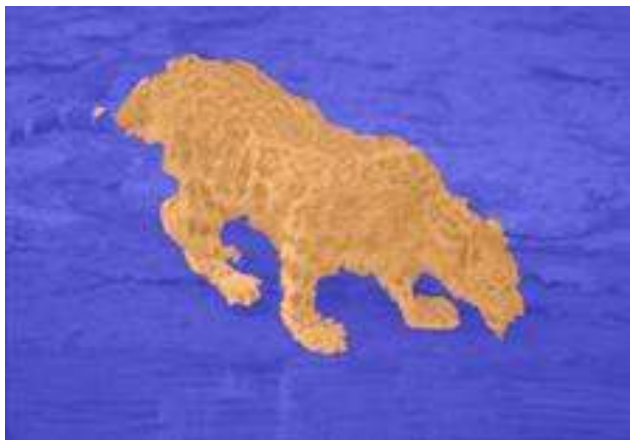
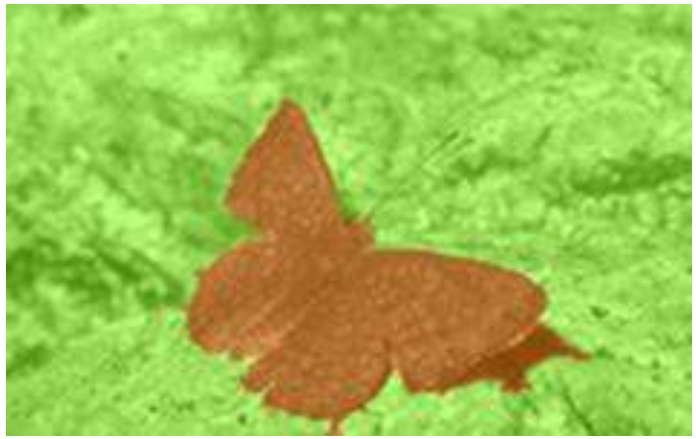
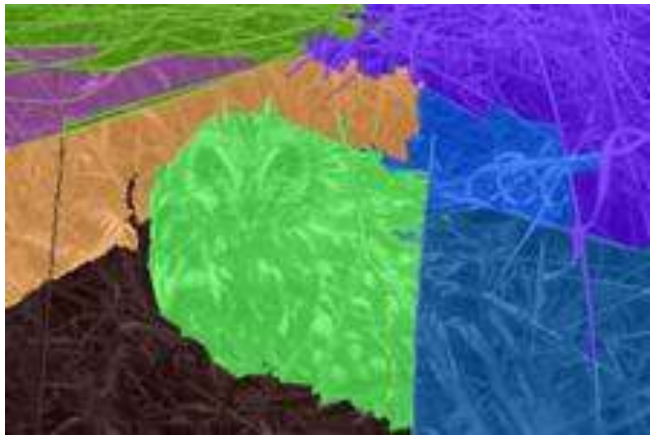
- By hand: 6 hours.
- Semi-automatically: 1.5 hours

—> Substantial time saving for the neuroscientists.

Graph-Cut on Ordinary Images

↳ not only in neuroscience



sink and source → should have semantic meaning (fg and bg)

Galun et al., ICCV'03

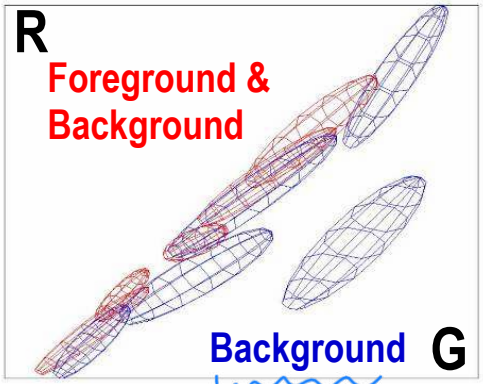
Interactive Foreground Extraction

K-means: not good by itself

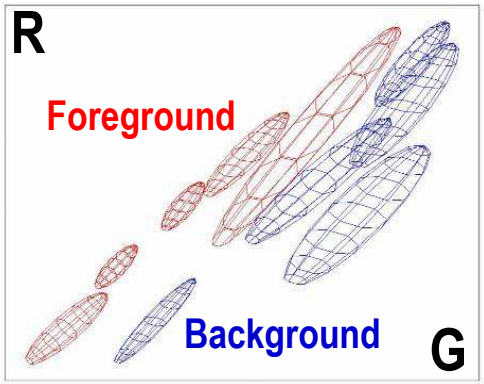
with combination →

→ gives good results

Automatic segmentation of flowers



Iterated graph cut

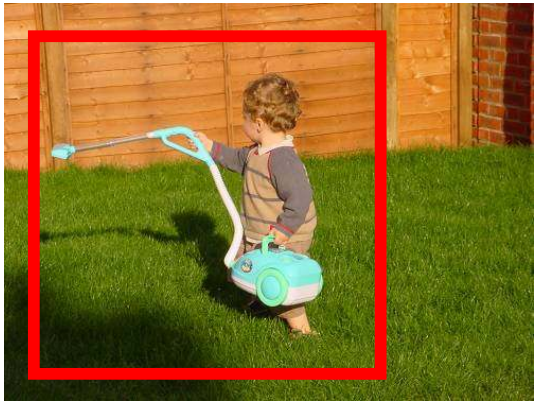


outside of the box

- K-means to learn color distributions of fg and bg
- Graph cuts to infer the segmentation

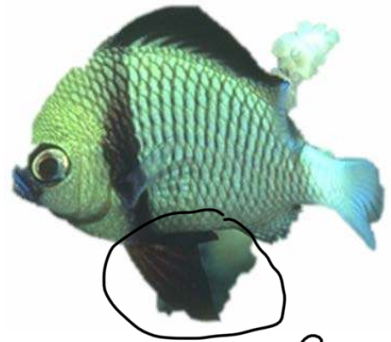
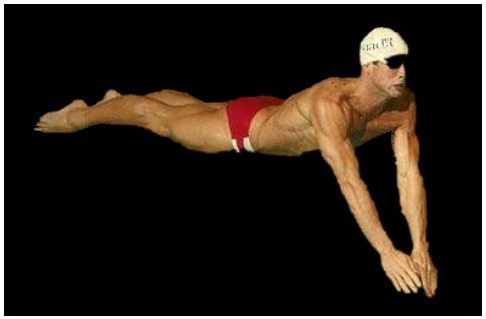
Use K-means to assign image pixels either to fg or bg, based on how similar their colors are to the k-means distribution

Relatively Easy Examples

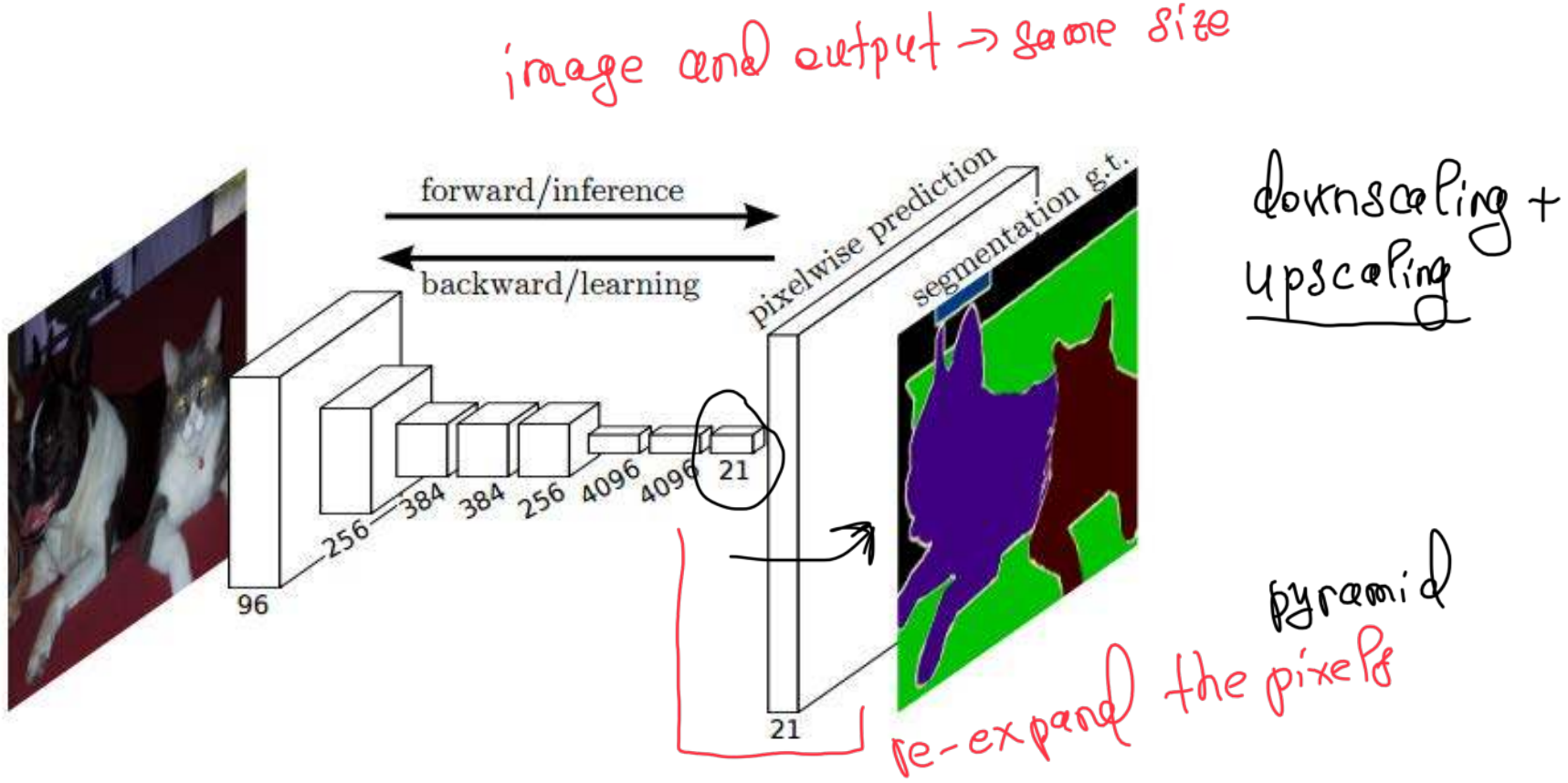


More Difficult Examples

Including cases where failing

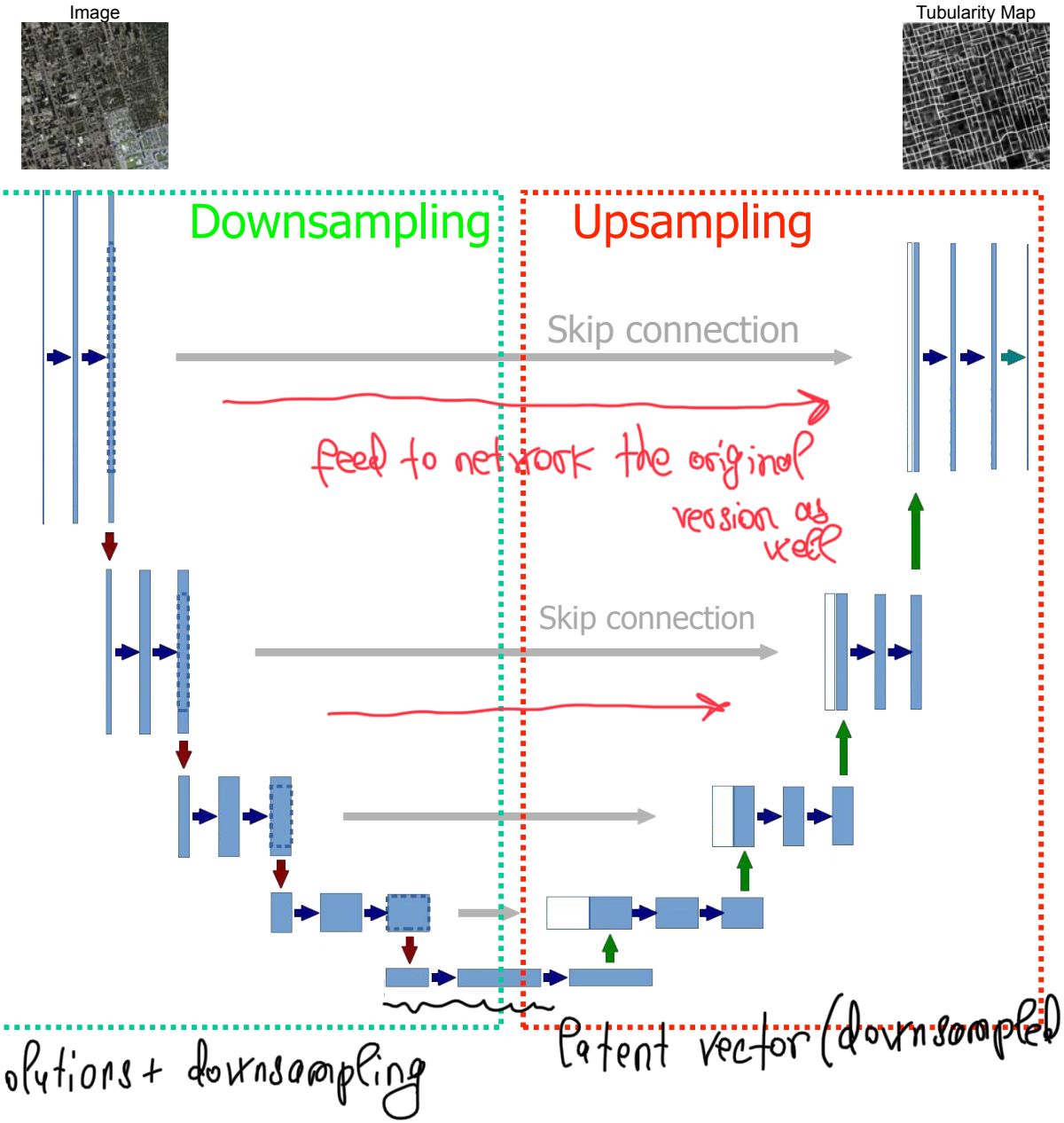
	Camouflage & Low Contrast	Troubling! Fine structure	are looking at! Semantics of where you No telepathy
Initial Rectangle			
Initial Result	 Camouflage actually		

Convolutional Neural Nets



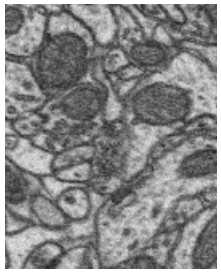
- Connect input layer to output one made of segmentation labels.
- Need layers that both downscale and upscale.
- Connect the lower layers directly to the upper ones. ★

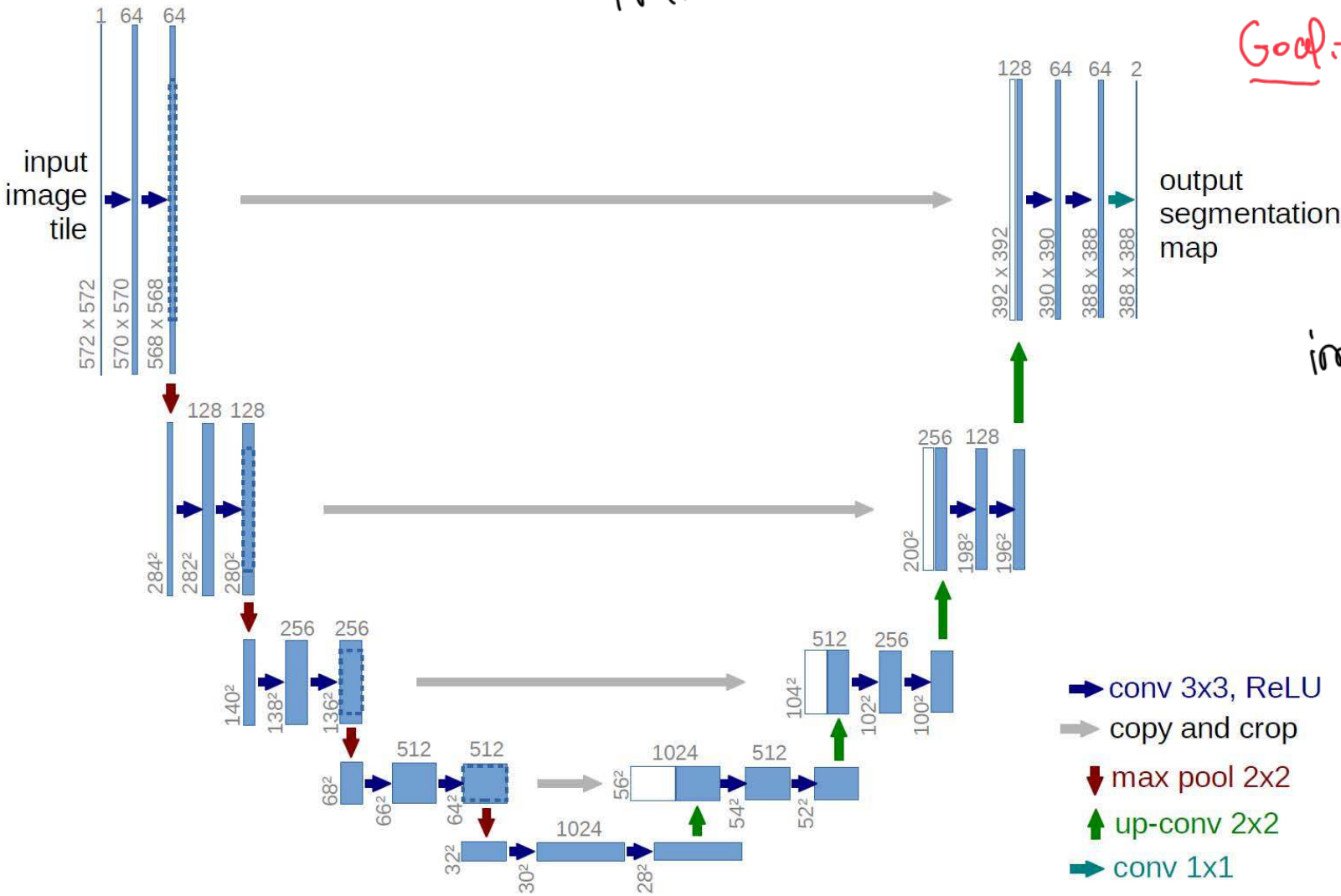
Reminder: U-Net for Delineation



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

U-Net for Segmentation





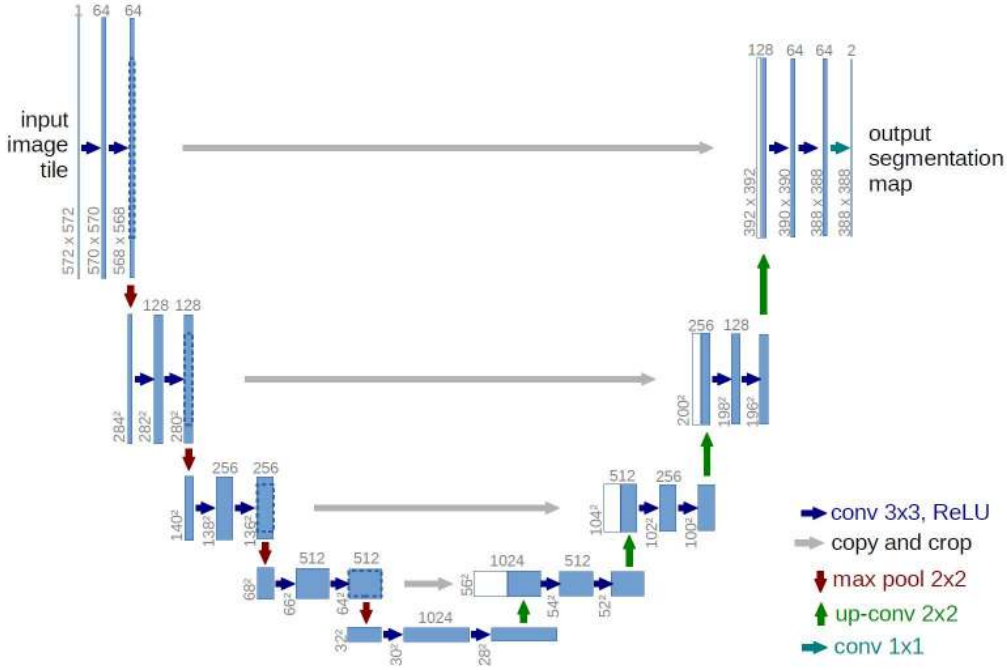
Train with SGD, or Adam

Goal: finding mitochon- -dria

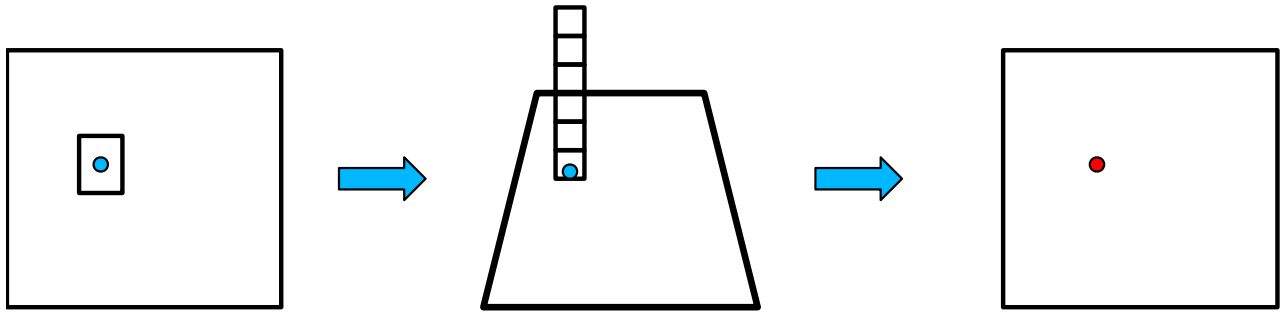
image with zeros and ones

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

Potential Interpretation

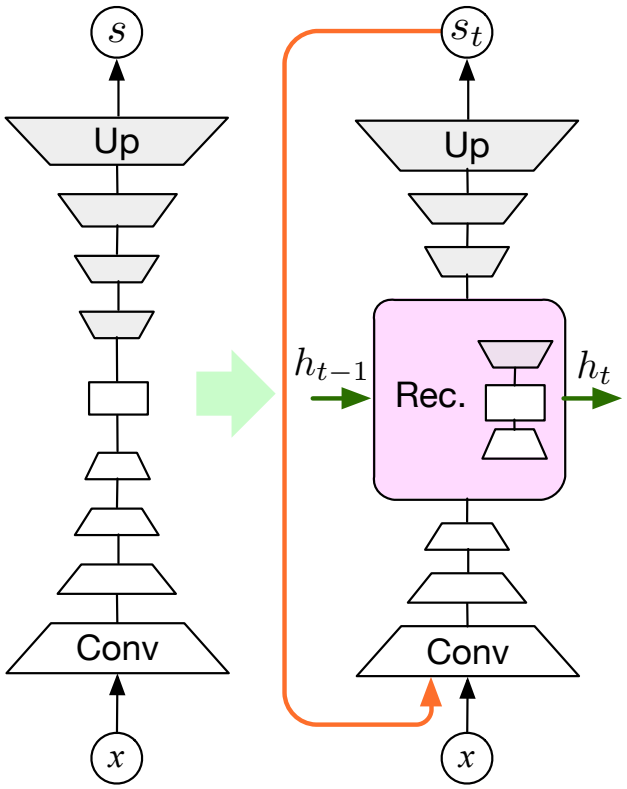


- A key role of the ConvNet is to generate for every output pixel a feature vector containing the output of all the intermediate layers.
- A classifier then assigns a label to the individual pixels.



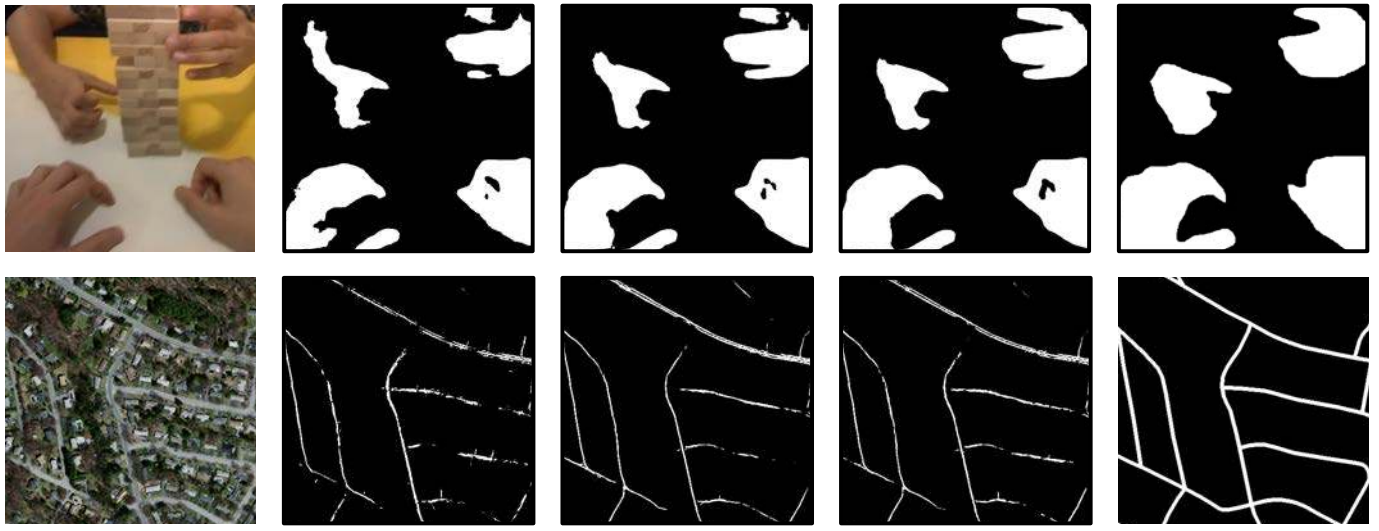
In the Street

Recursive Segmentation



U-Net Rec. U-Net

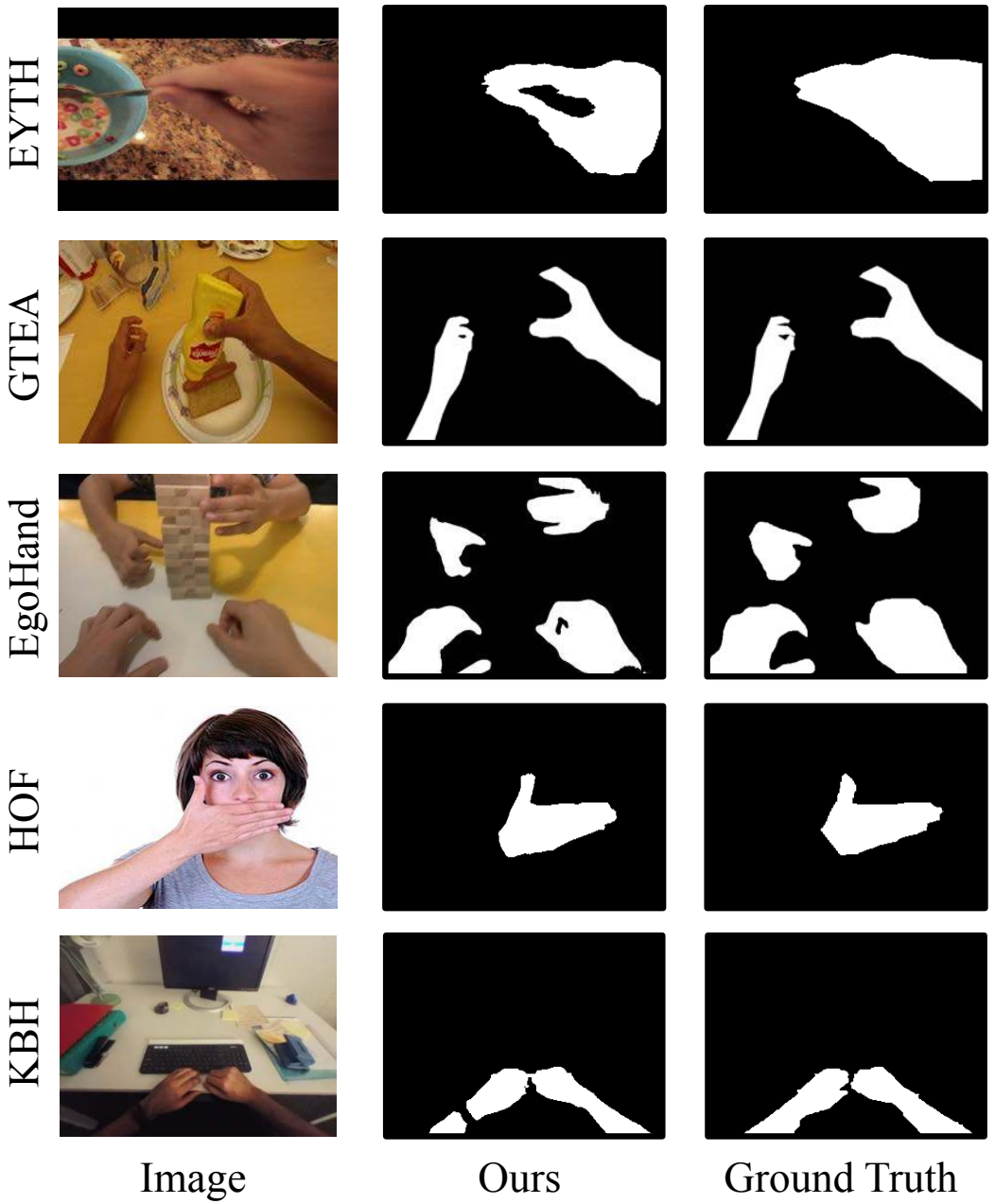
feed recursively to the network image and prev result



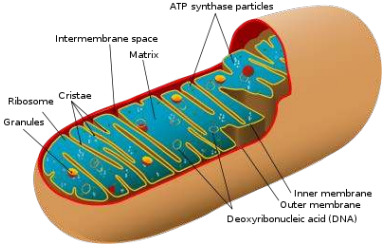
Hand/Road segmentation

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

Recursive Hand Segmentation

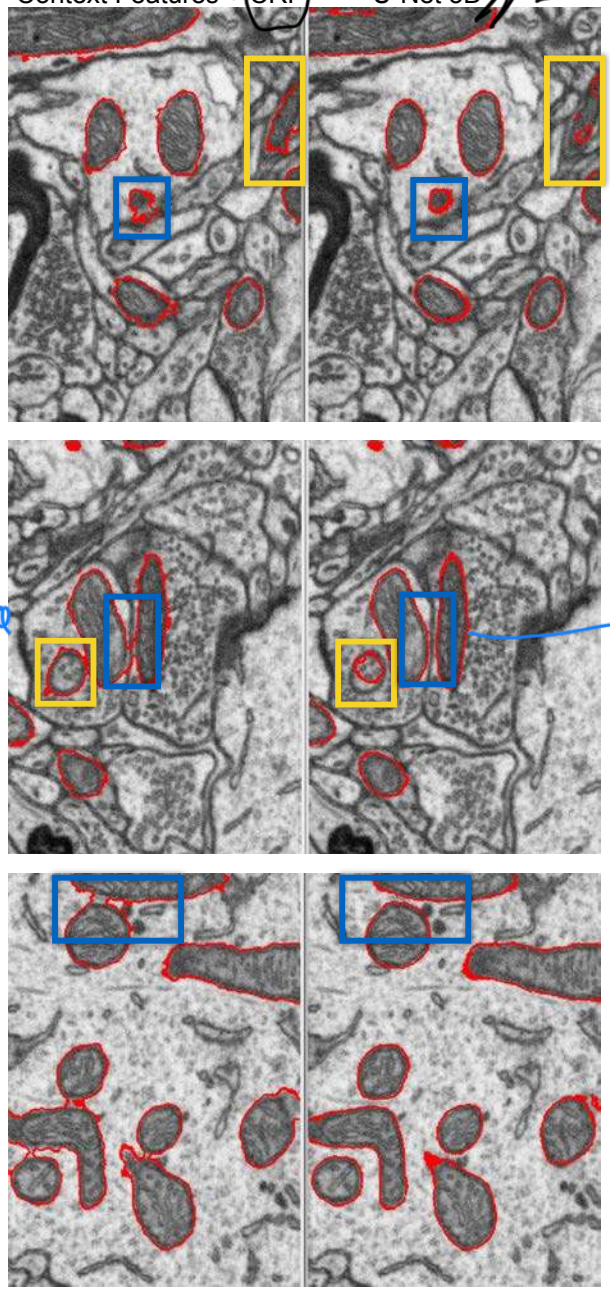


Mitochondria



on superpixels → graph-based techniques

Context Features + CRF | U-Net 3D → stacks of img's



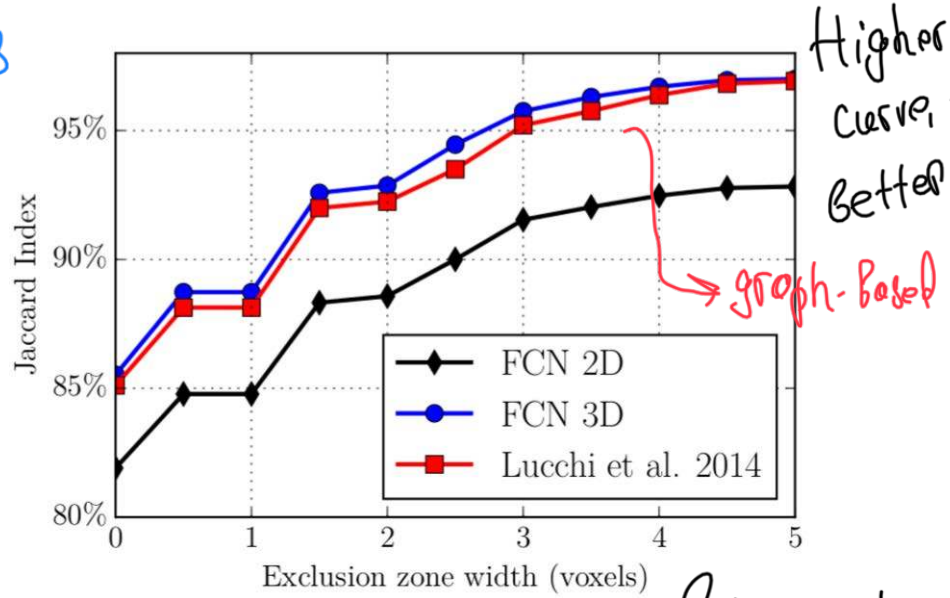
old algorithm works better (finds sth that's super small)

merges 2 mitochondria

U-Net does better to find boundaries

Striatum Mitochondria

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



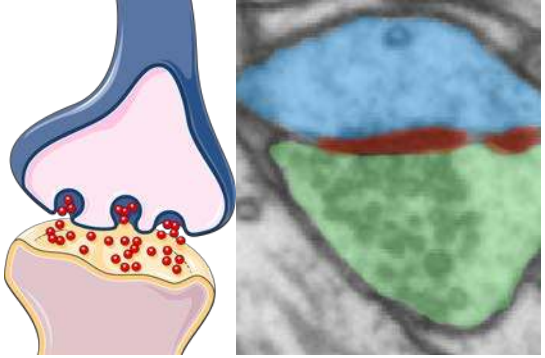
Higher curve, the better

graph-based

3D → gives you spatial information

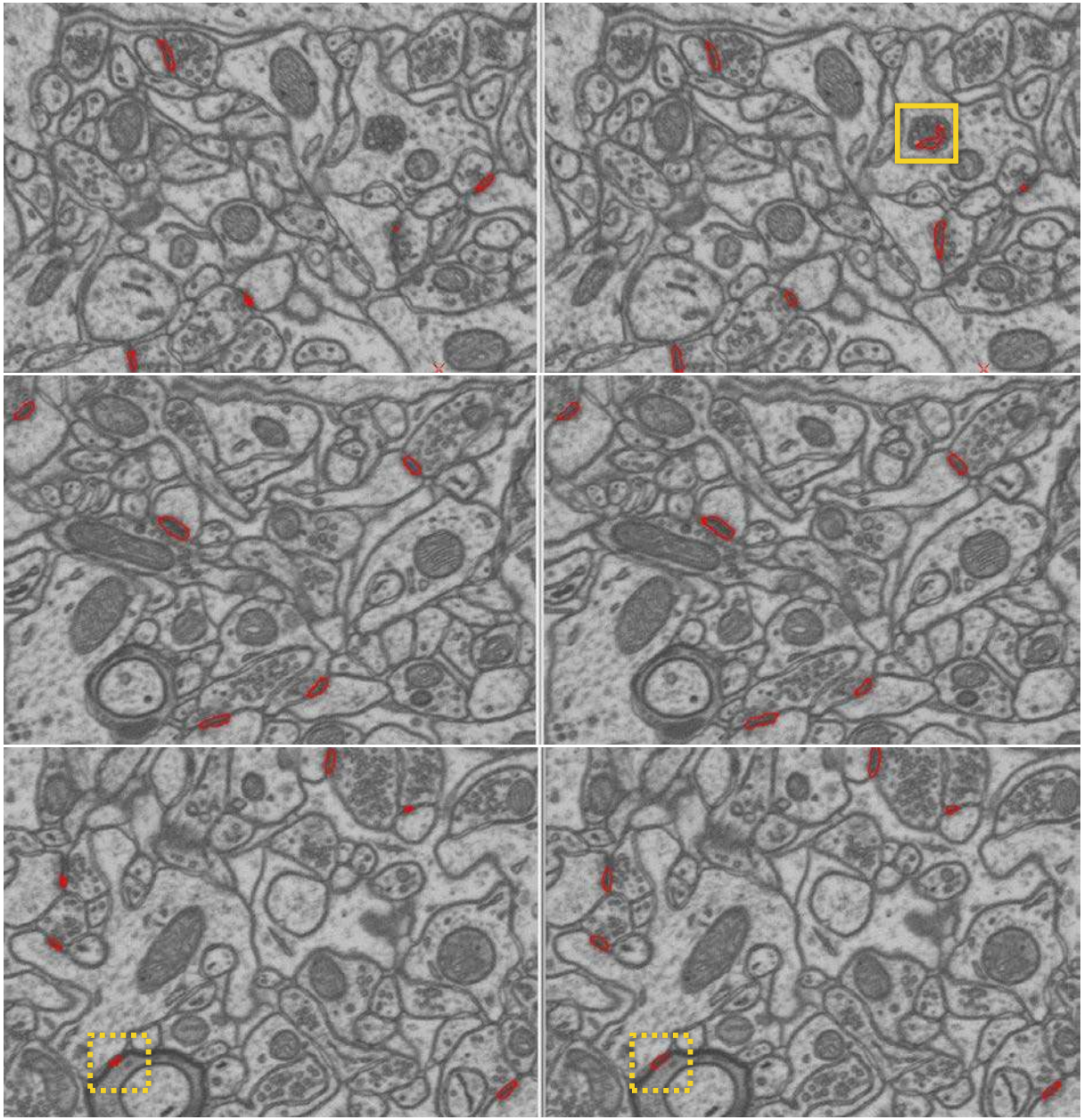
—> Using a ConvNet boosts performance but there is still work to do!

Synapses



Context Features 3D CRF

U-Net 3D



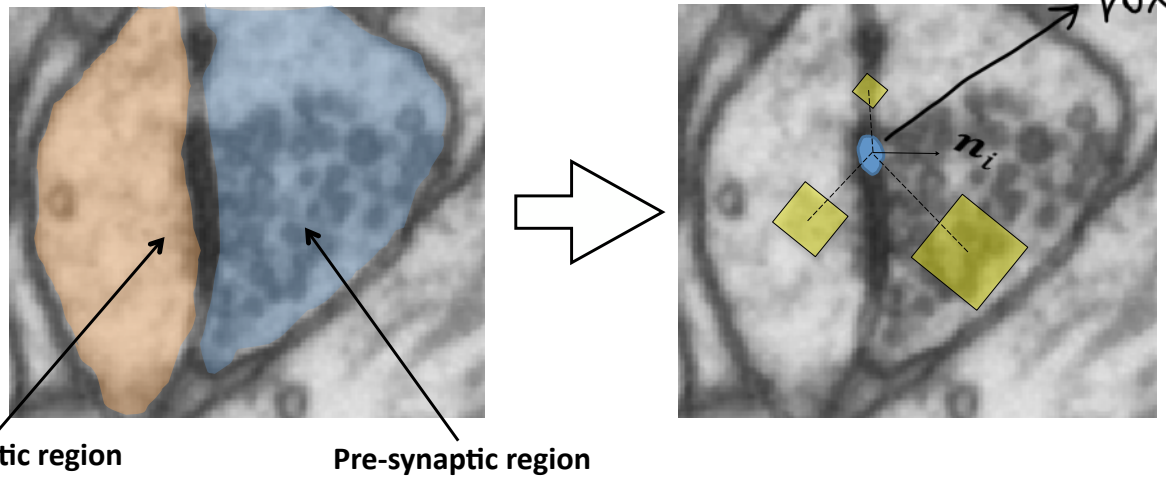
Jaccard Index	Method
66.8%	Context Features 2D
85.2%	Context Features 3D
73.5%	U-Net 2D
77.0%	U-Net 3D

on this specific example

?

Context-Based Features

Synapse:



Post-synaptic region

- Dendrite
- No vesicles

Pre-synaptic region

- Axon terminal
- Many vesicles

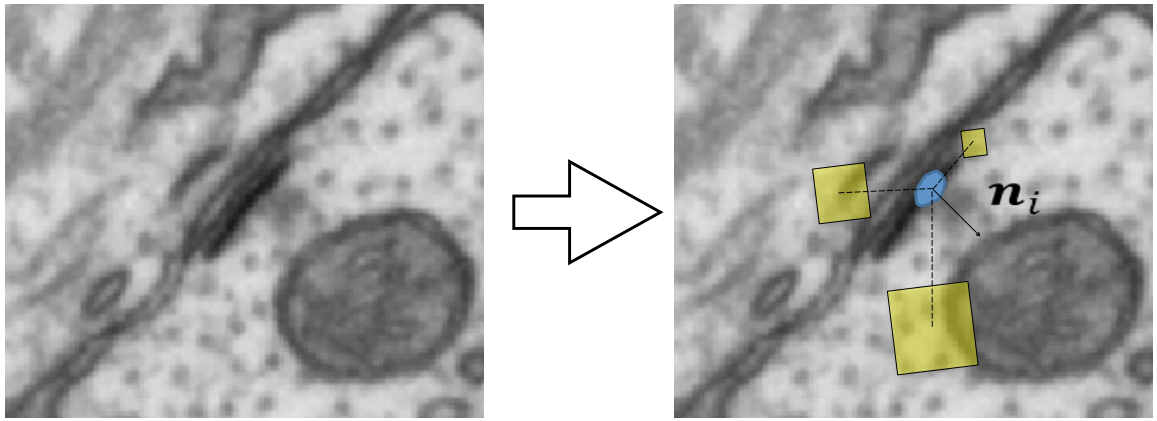
voxel: pixel in 3D

i) Estimate direction of the edge (orientation)

ii) Compute local stats of those boxes

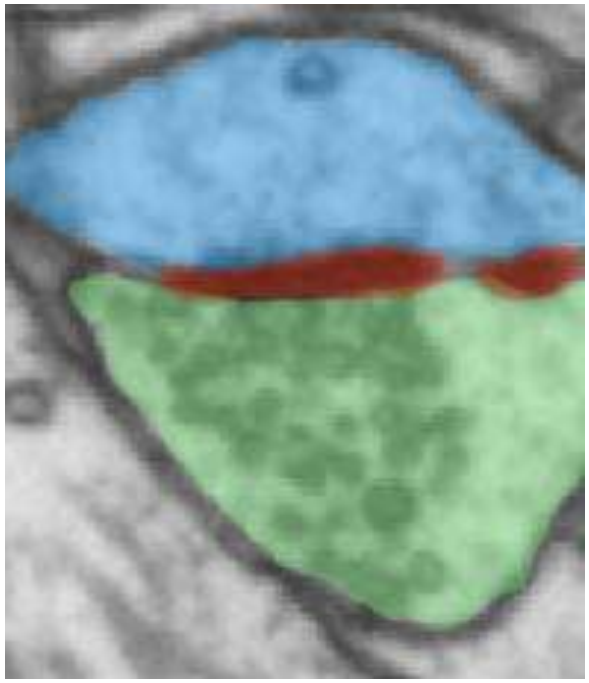
iii) Use AdaBoost algo to choose subset of boxes

Non-Synapse:

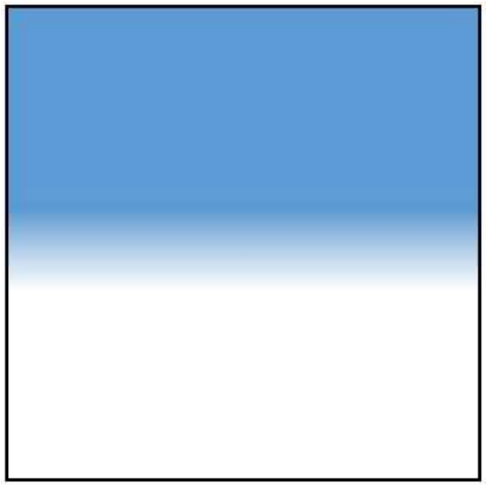


- The old style context-based features explicitly model orientation.
- How can we do that in a deep-learning framework?

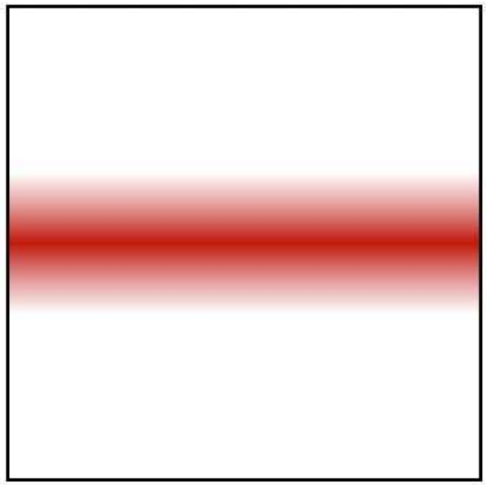
Probabilistic Atlases



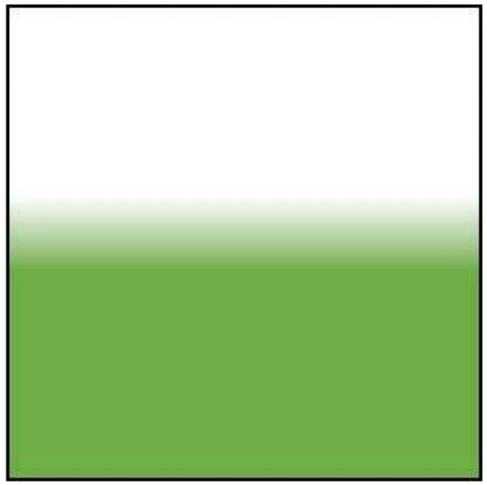
Synapse in canonical orientation



Probability of being a post-synaptic voxel

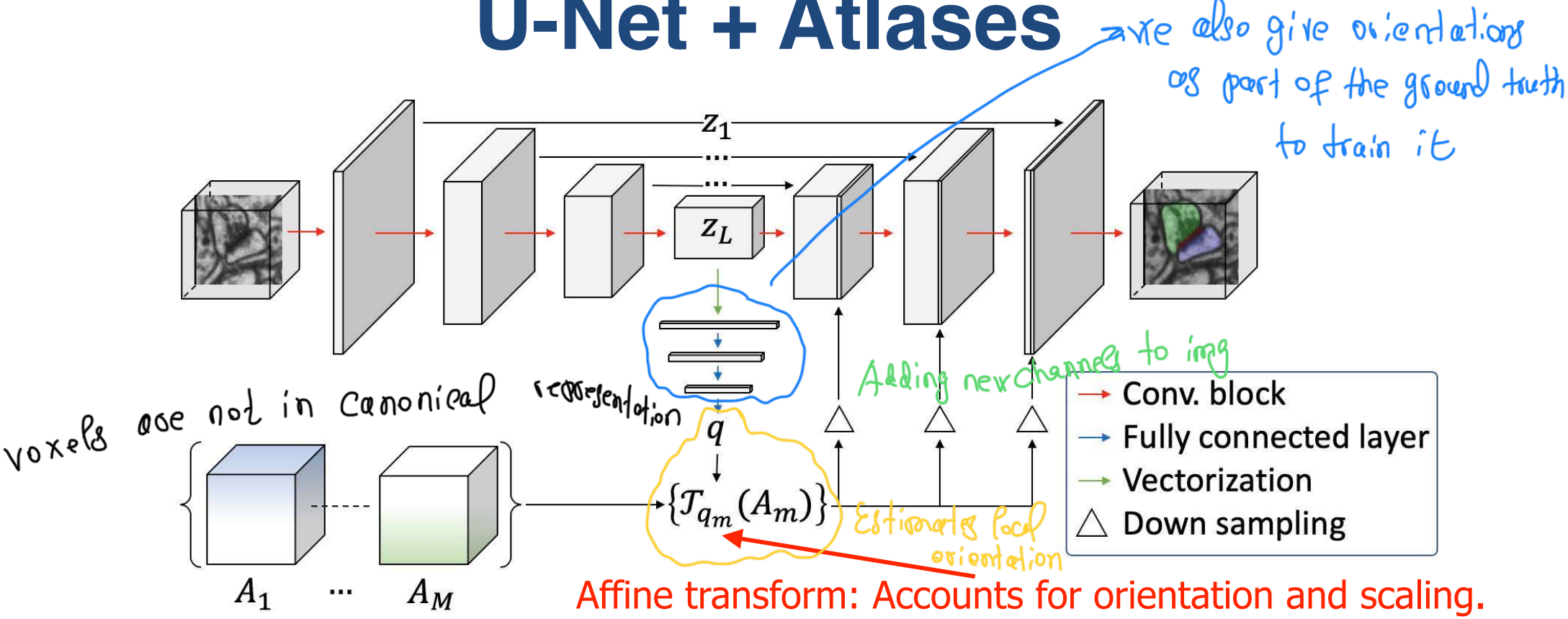


Probability of being a cleft voxel

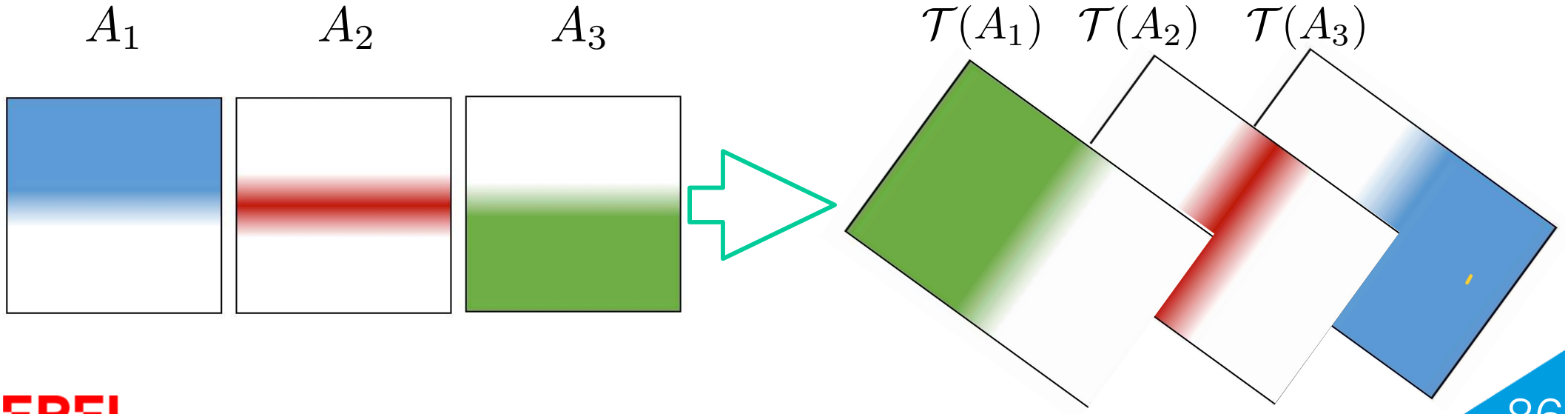


Probability of being a pre-synaptic voxel

U-Net + Atlases



Affine transform: Accounts for orientation and scaling.



3D Synapses

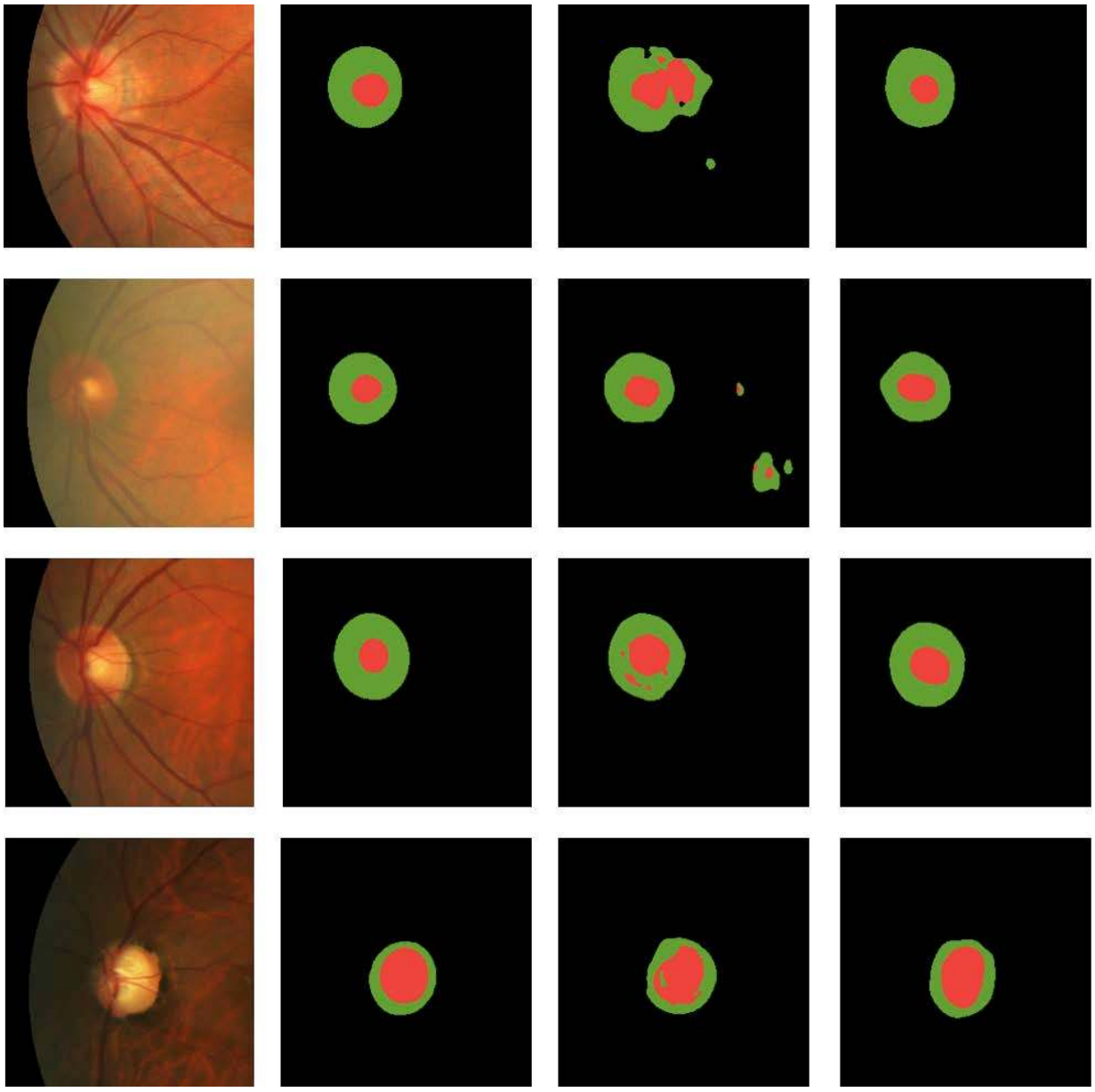
Synaptic Junction Segmentation

Baseline vs PA-Net

Probabilistic - APLS

U-Net

Atlases for Optic Disk Segmentation

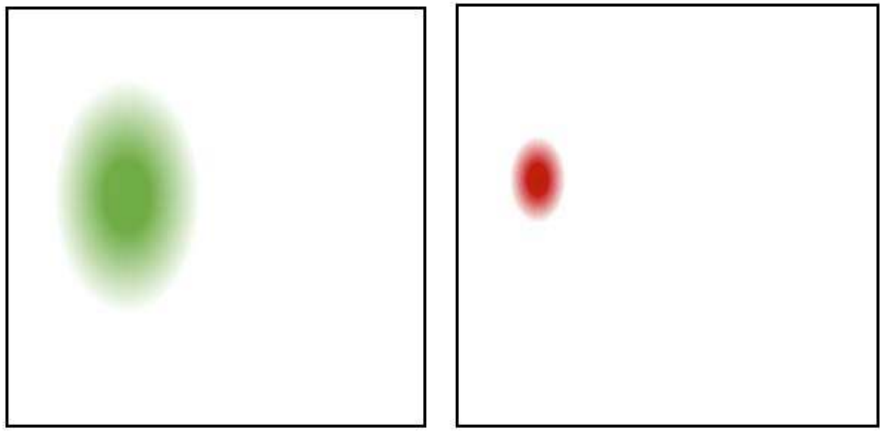


Ground truth U-Net PA-Net

Retina of patients

optic nerves

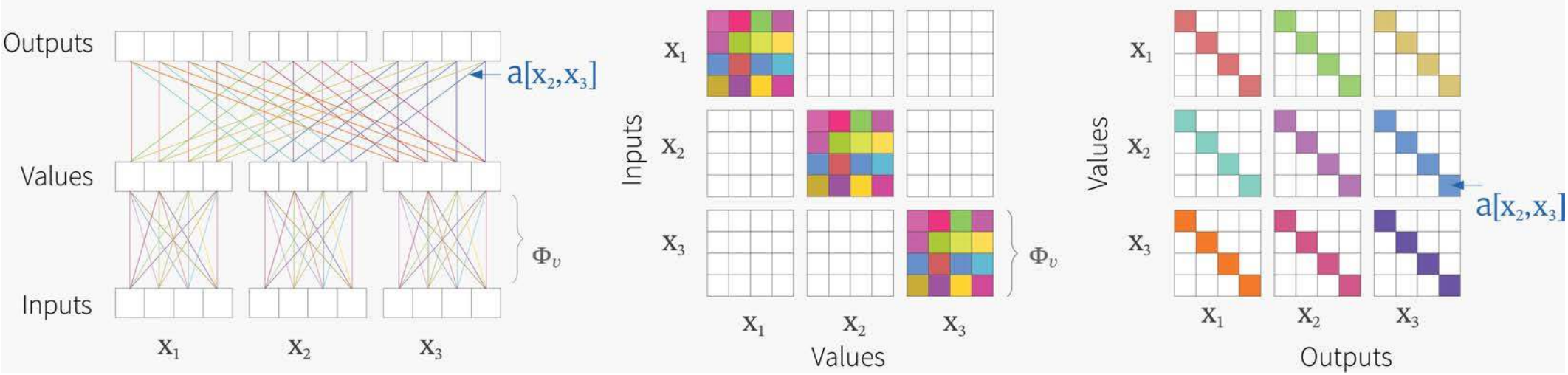
Probabilistic Atlases



Optic disk Optic nerve

fg, outer (inner) disk

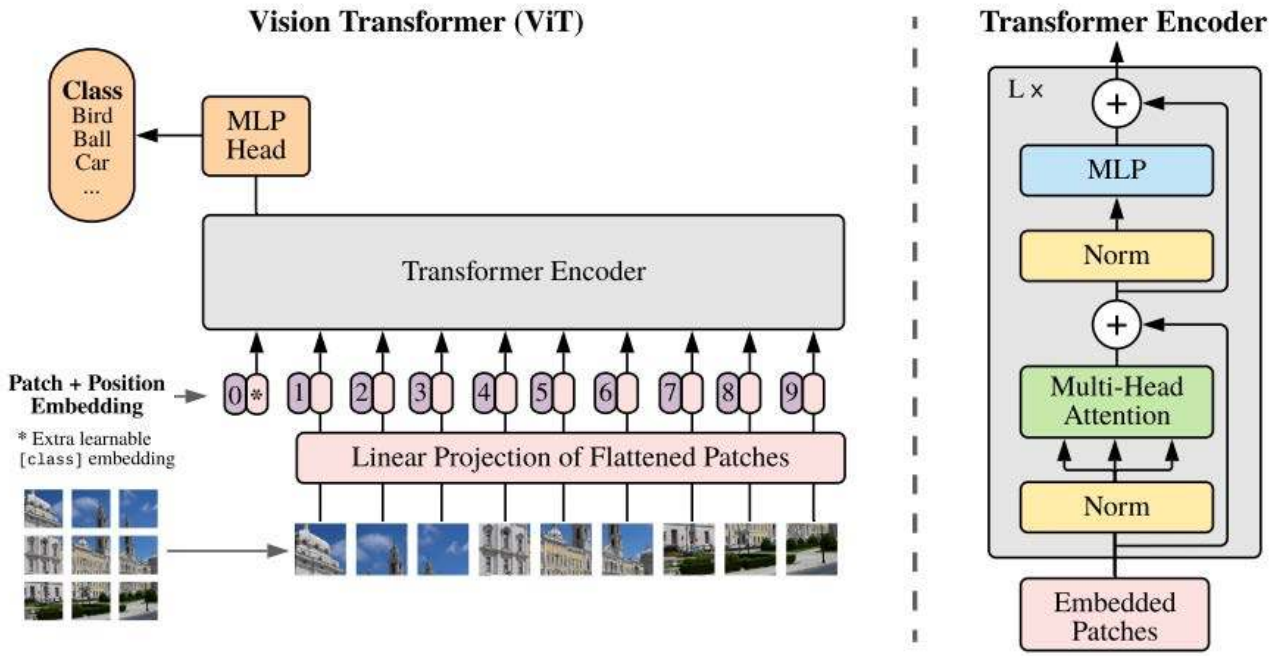
Optional: Transformers in NLP



Given $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N]$,

- compute SelfAttention($\mathbf{X}$) using far fewer weights than if we used a fully connected network.
- In NLP the $\mathbf{x}_n$ are embedding for words.
- In Vision they can be taken to be image patches.

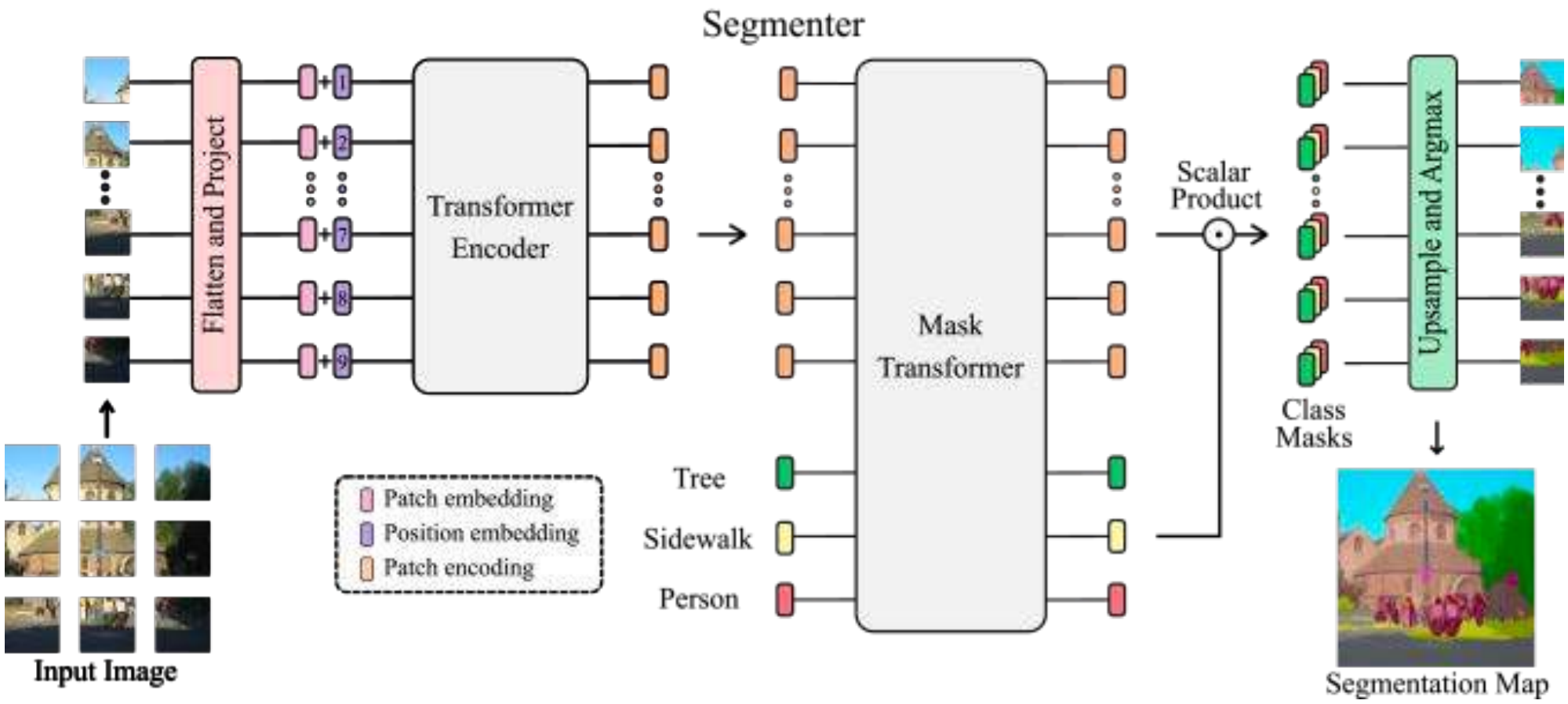
Vision Transformers for Recognition



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

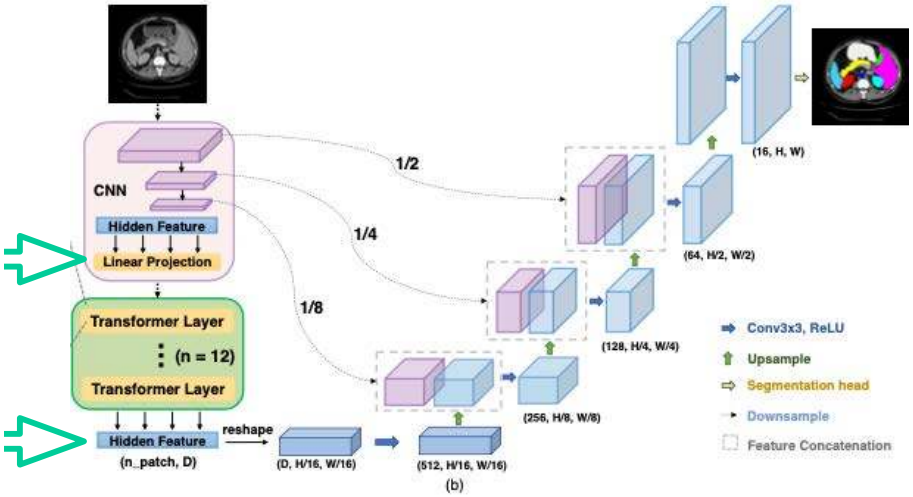
Dosovitskiy et al., ICLR'21

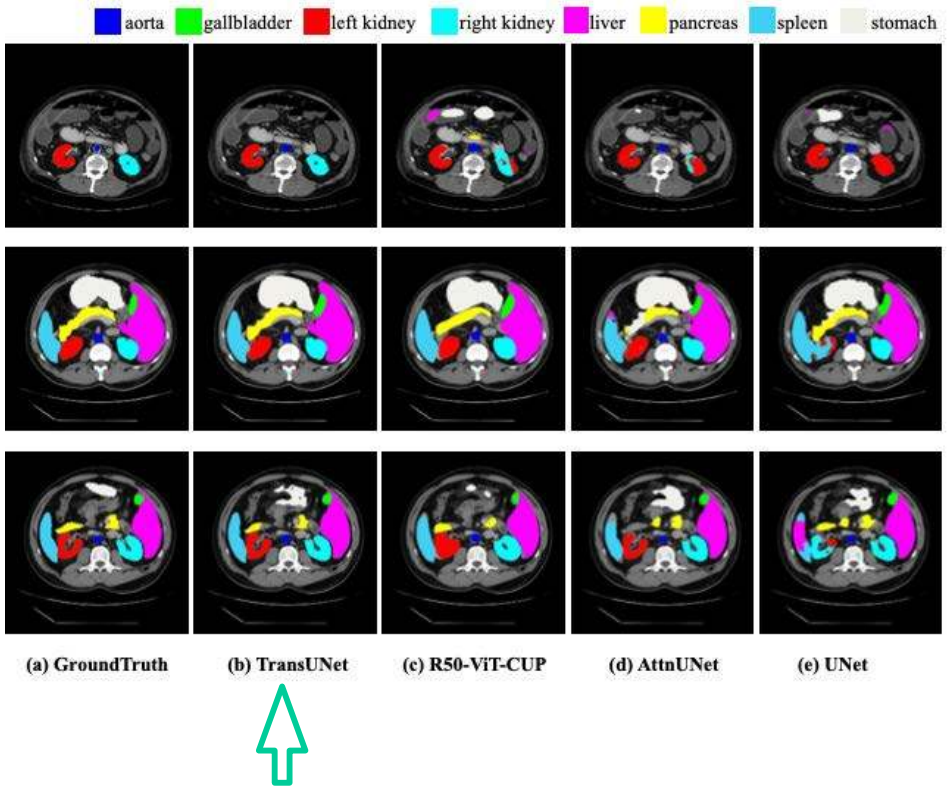
Vision Transformers for Segmentation



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

U-NET + Transformers





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

—> Best of both worlds?

Moral of the Story

- Deep Networks are powerful tools, especially when there is enough 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.

In Short

- Local methods can provide valuable information but are inherently limited.

You need context!

- 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.

if you have limited data

old-fashioned techniques

- Same philosophy as for delineation.

What About the Dog?

Most current algorithms won't "see" it!