上海交通大学

计算机视觉

教师：赵旭

班级：AI4701

2024 春

18. 图像分割

Grouping in vision

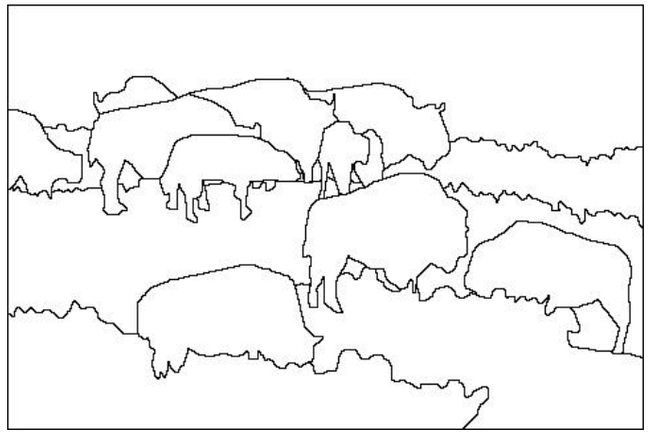
- Goals:
  - Gather features that belong together
  - Obtain an intermediate representation that compactly describes key image (video) parts
- Top down vs. bottom up segmentation
  - Top down: pixels belong together because they are from the same object
  - Bottom up: pixels belong together because they look similar

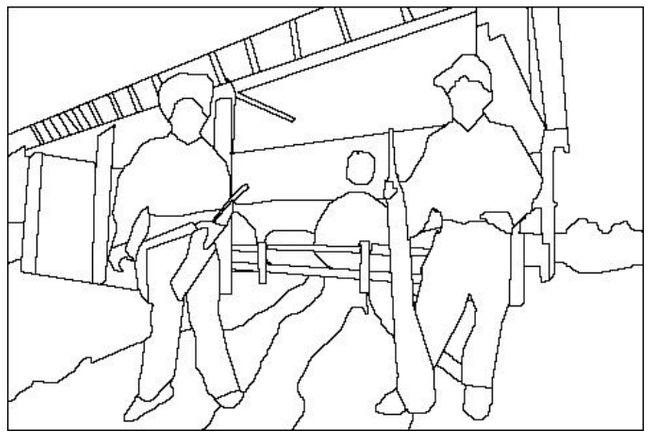
The goals of segmentation

Separate image into coherent “objects”

image

human segmentation





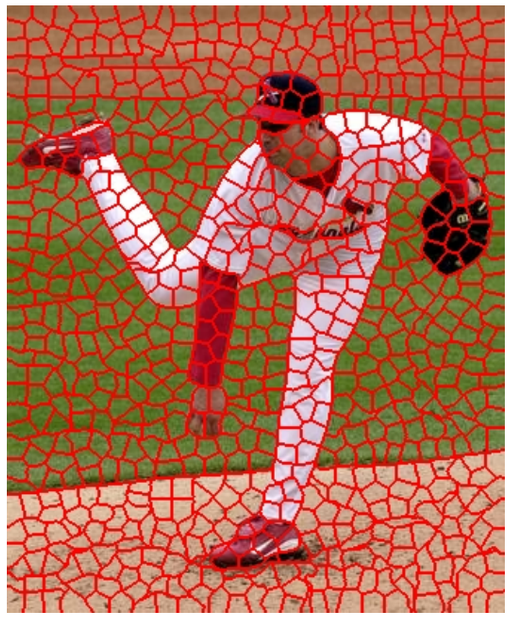
Source: Lana Lazebnik

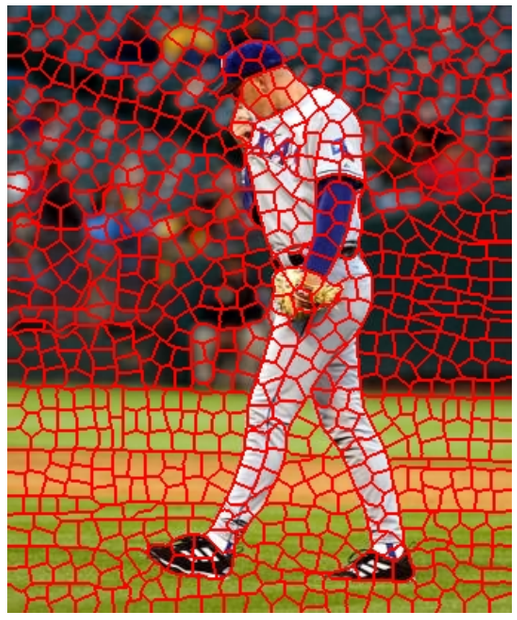
The goals of segmentation

Separate image into coherent “objects”

Group together similar-looking pixels for efficiency of further processing

“superpixels”





X. Ren and J. Malik. **Learning a classification model for segmentation.** ICCV 2003.

Source: Lana Lazebnik

Bottom-up segmentation via clustering

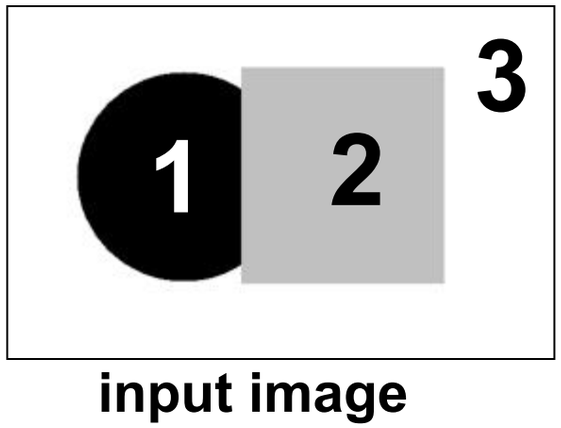
- Algorithms:
  - Mode finding and mean shift: k-means, EM, mean-shift
  - Graph-based: normalized cuts
- Features: color, texture, …
  - Quantization for texture summaries

Clustering

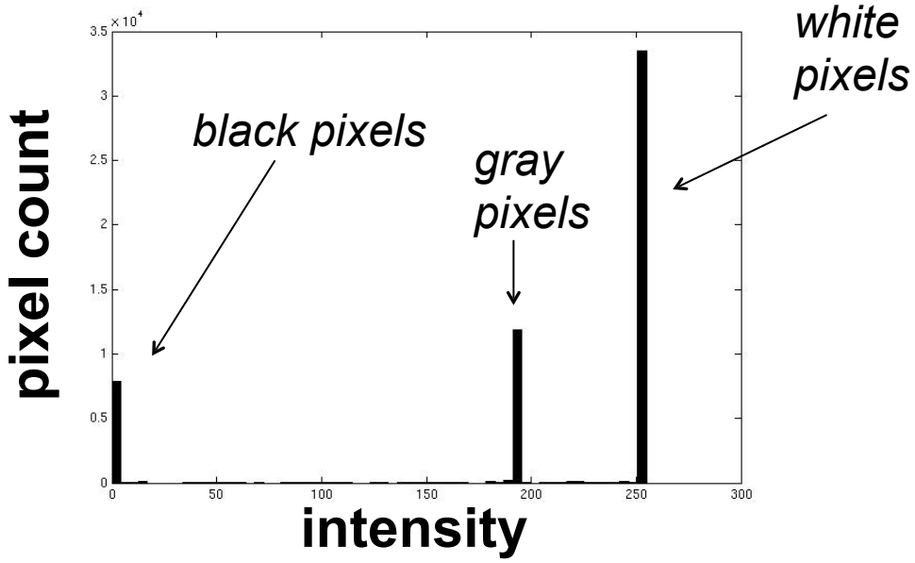
- Clustering algorithms:
  - Unsupervised learning
  - Detect patterns in unlabeled data
    - E.g. group emails or search results
    - E.g. find categories of customers
    - E.g. group pixels into regions
- Useful when don't know what you're looking for
- Requires data, but no labels

Slide credit: Dan Klein

Image segmentation: toy example

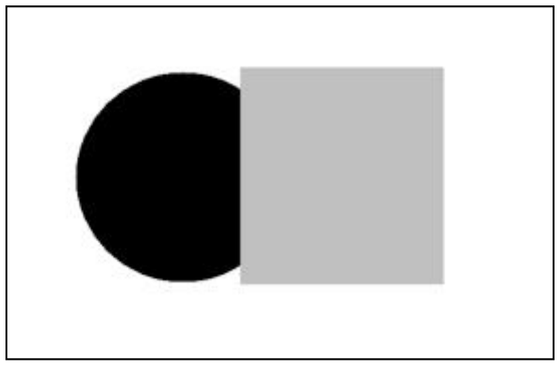


input image

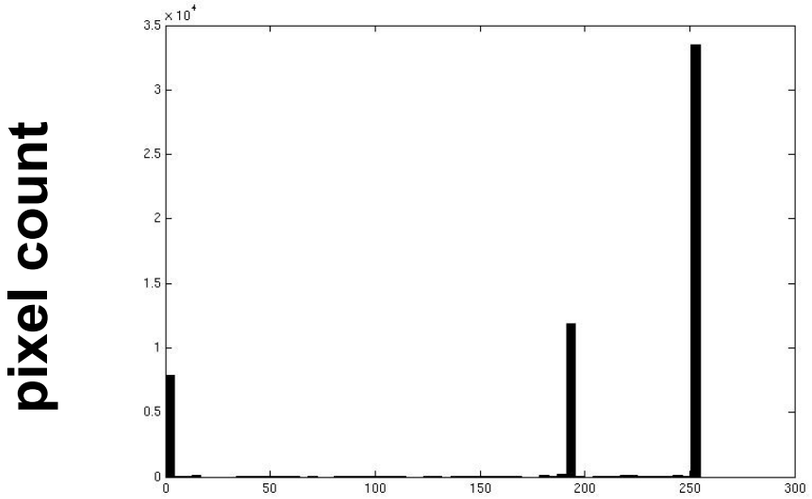


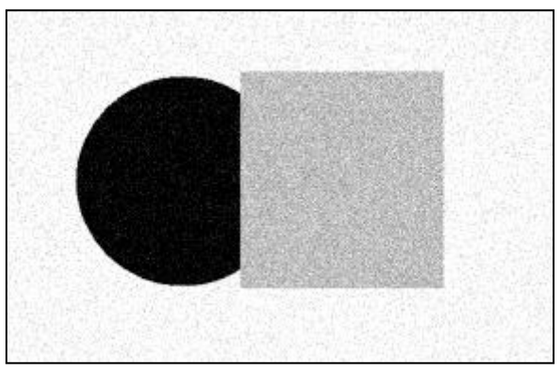
- These intensities define the three groups.
- We could label every pixel in the image according to which of these primary intensities it is.
  - i.e., *segment* the image based on the intensity feature.
- What if the image isn't quite so simple?

Slide credit: Kristen Grauman

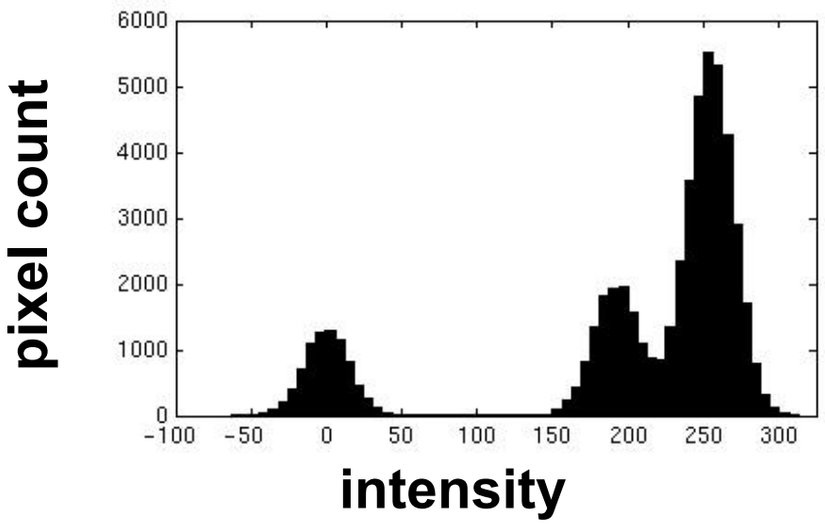


input image

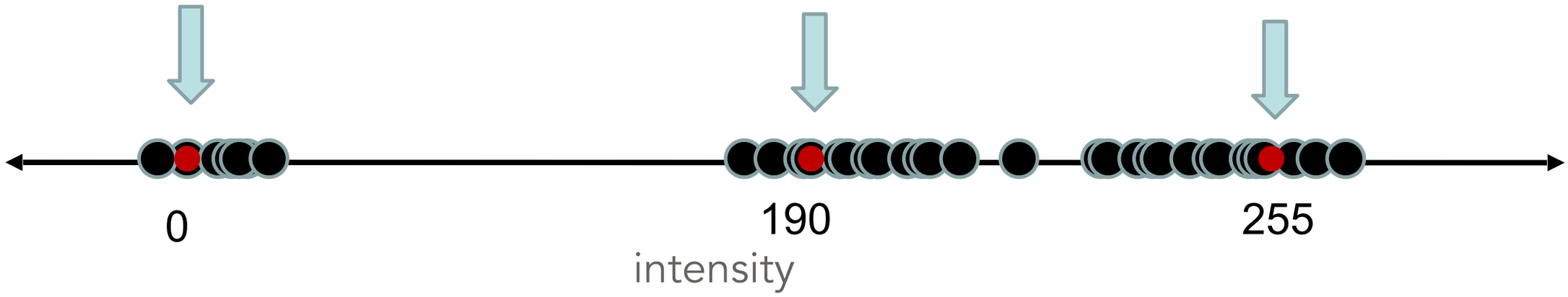




input image



Now how to determine the three main intensities that define our groups?
We need to cluster.

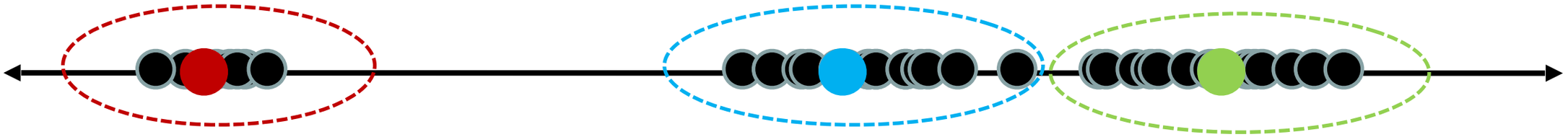


- Goal: choose three “centers” as the representative intensities, and label every pixel according to which of these centers it is nearest to.
- Best cluster centers are those that minimize SSD between all points and their nearest cluster center c_i:

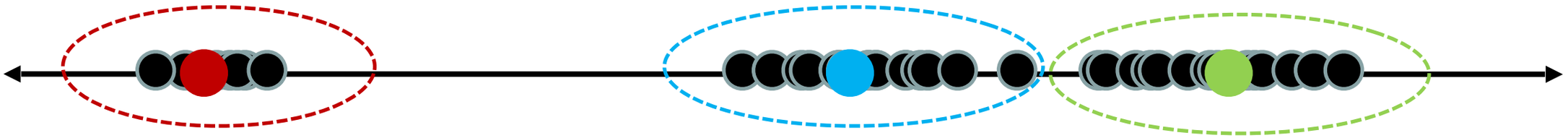
$$\sum_{\text{clusters } i} \sum_{\text{points p in cluster } i} \|p - c_i\|^2$$

Clustering

- With this objective, it is a "chicken and egg" problem:
  - If we knew the **cluster centers**, we could allocate points to groups by assigning each to its closest center.

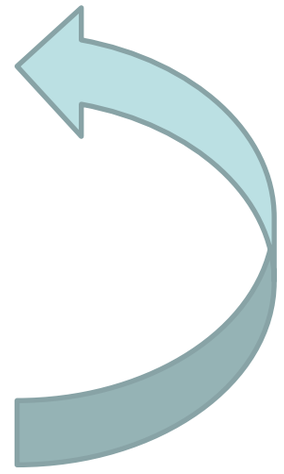


- If we knew the **group memberships**, we could get the centers by computing the mean per group.



K-means clustering

- Basic idea: randomly initialize the *k* cluster centers, and iterate between the two steps we just saw.
  1. Randomly initialize the cluster centers, $c_1, \ldots, c_K$
  2. Given cluster centers, determine points in each cluster
     - For each point p, find the closest c_i. Put p into cluster i
  3. Given points in each cluster, solve for c_i
     - Set c_i to be the mean of points in cluster i
  4. If c_i have changed, repeat Step 2



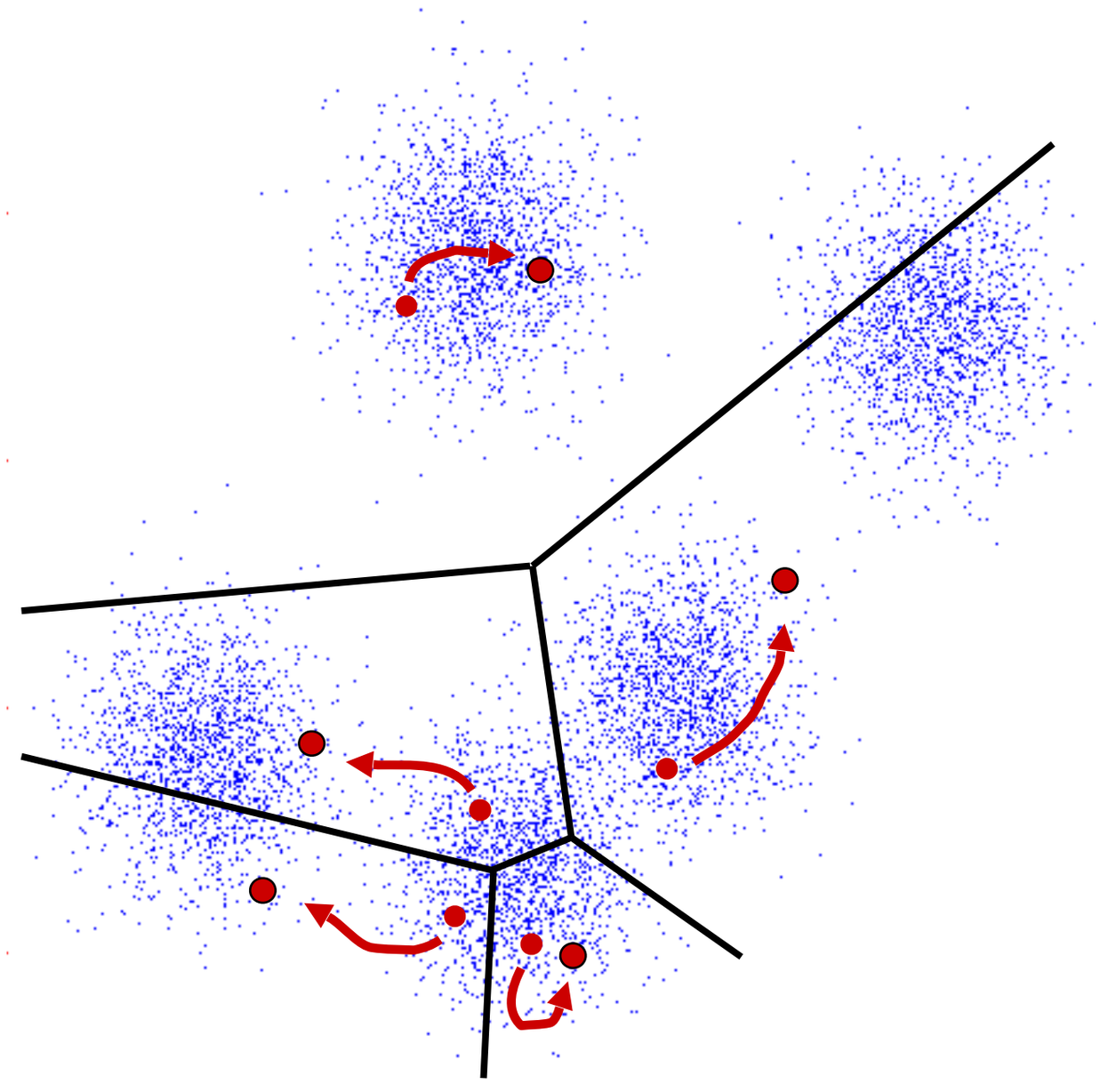
- Properties
  - Will always converge to *some* solution
  - Can be a "local minimum"
    - does not always find the global minimum of objective function:

$$\sum_{\text{clusters } i} \quad \sum_{\text{points p in cluster } i} \|p - c_i\|^2$$

Source: Steve Seitz

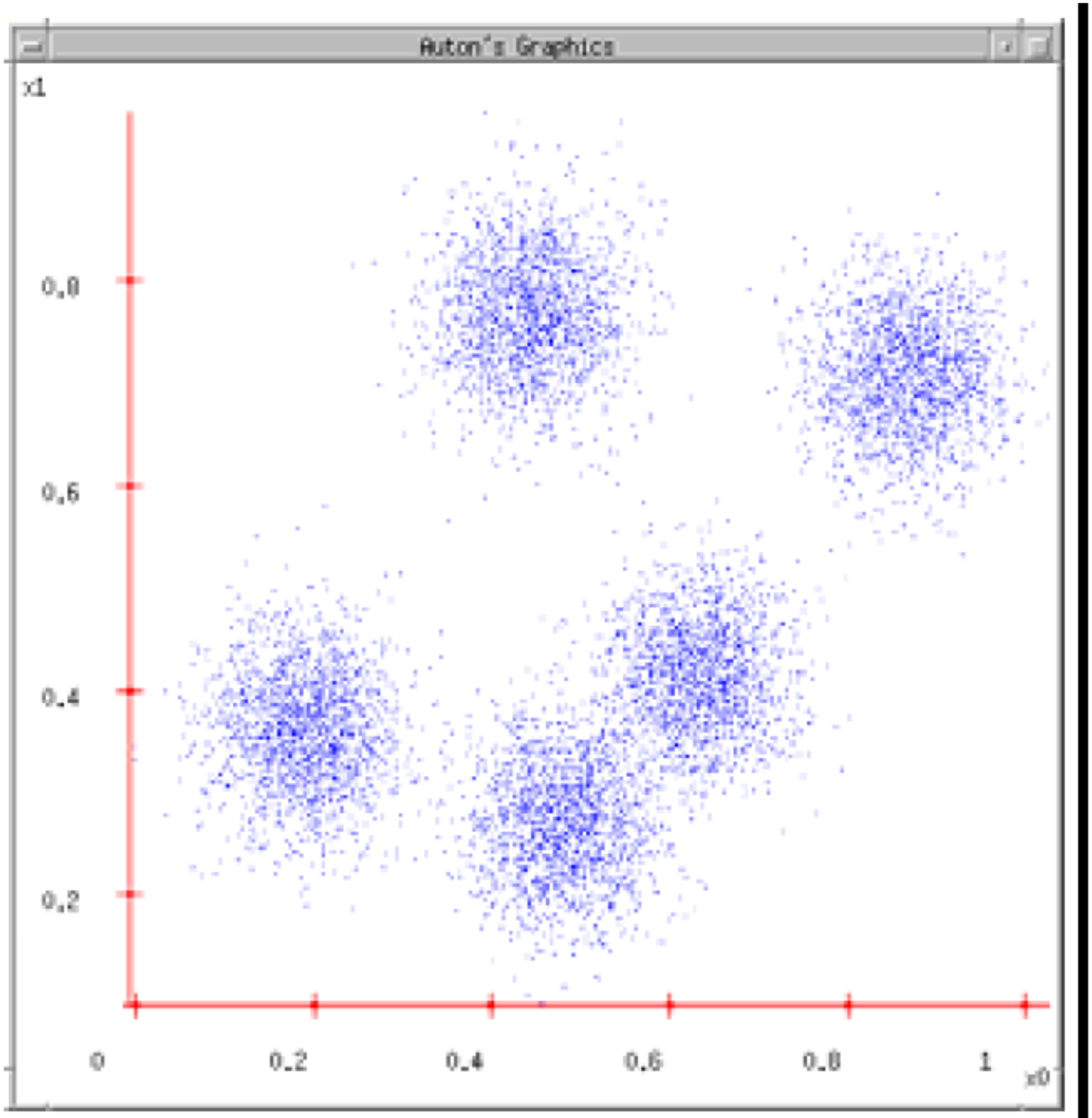
K-Means

- An iterative clustering algorithm
  - Pick K random points as cluster centers (means)
  - Alternate:
    - Assign data instances to closest mean
    - Assign each mean to the average of its assigned points
- Stop when no points' assignments change



K-means

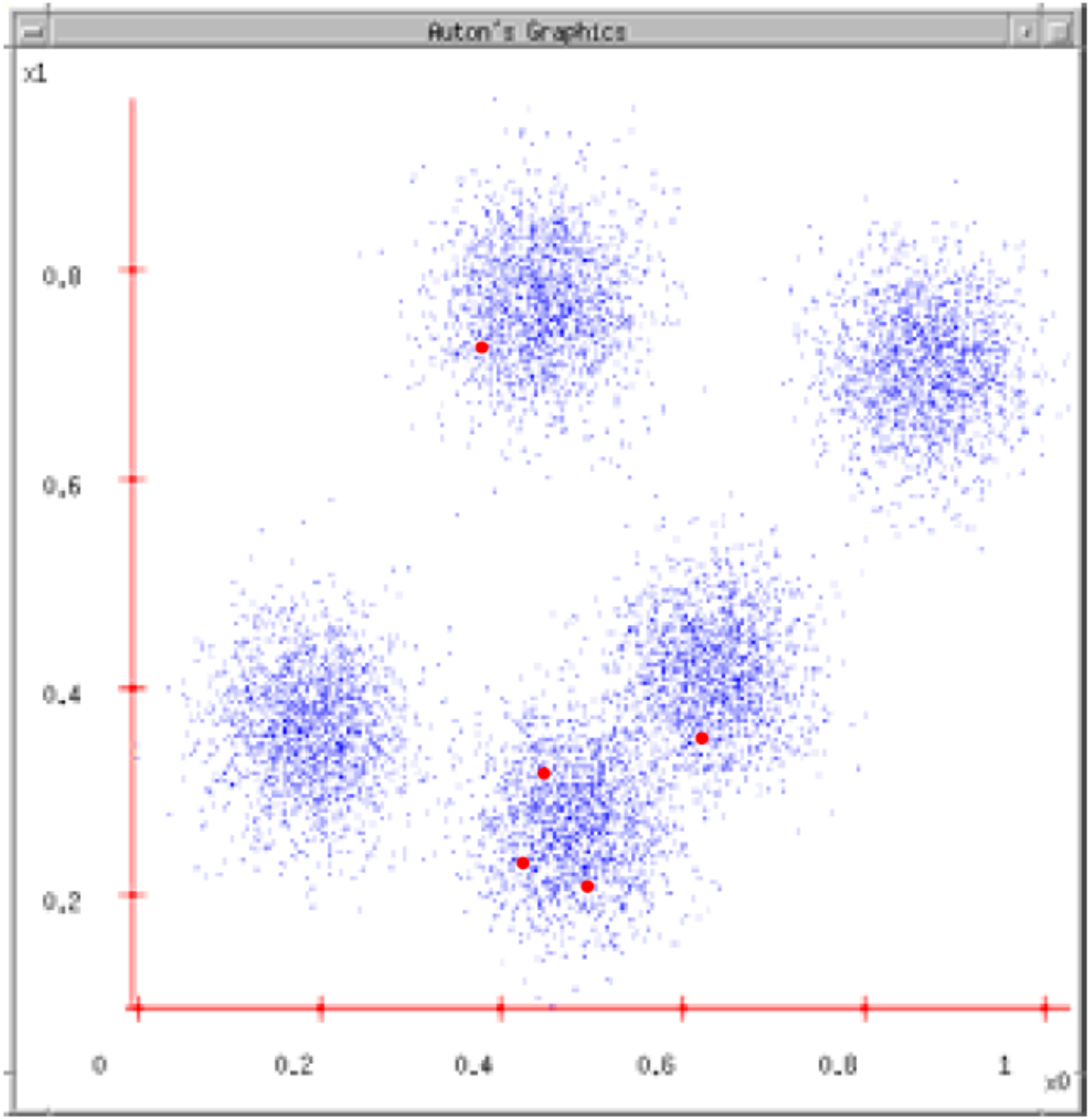
1. Ask user how many clusters they'd like. *(e.g. k=5)*



Slide credit Andrew Moore

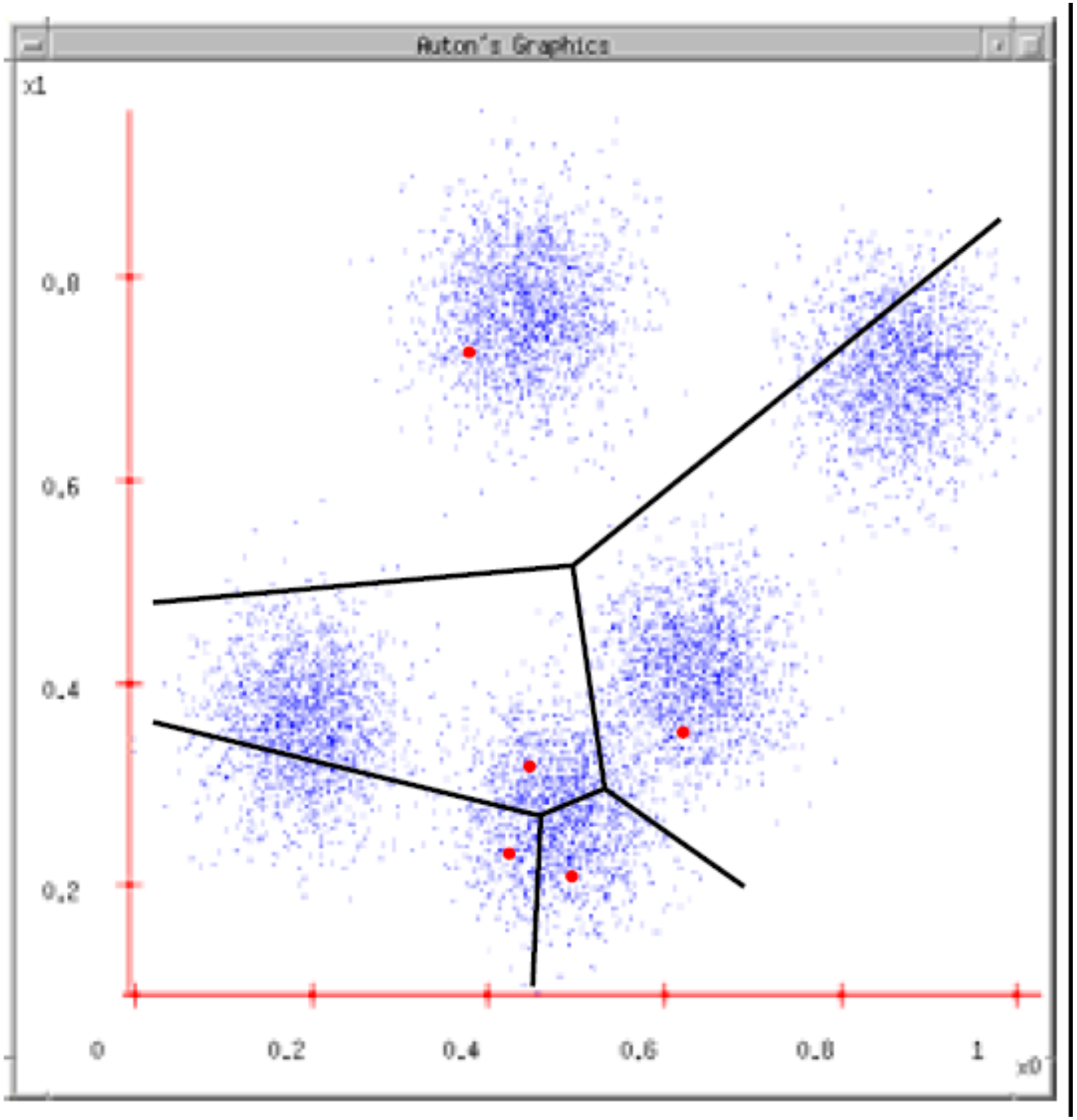
K-means

1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations



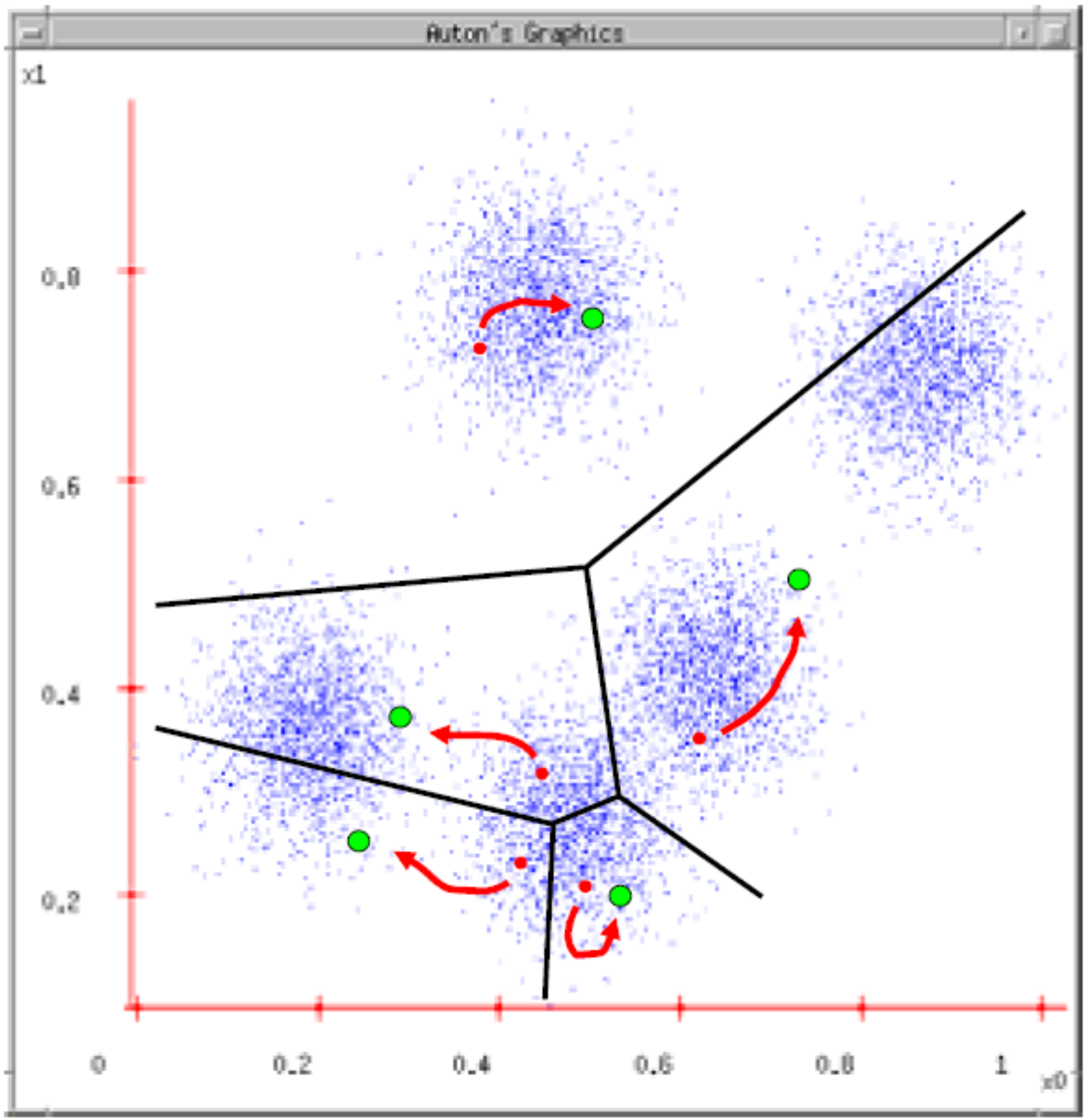
K-means

1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations
3. Each datapoint finds out which Center it's closest to. (Thus each Center "owns" a set of datapoints)



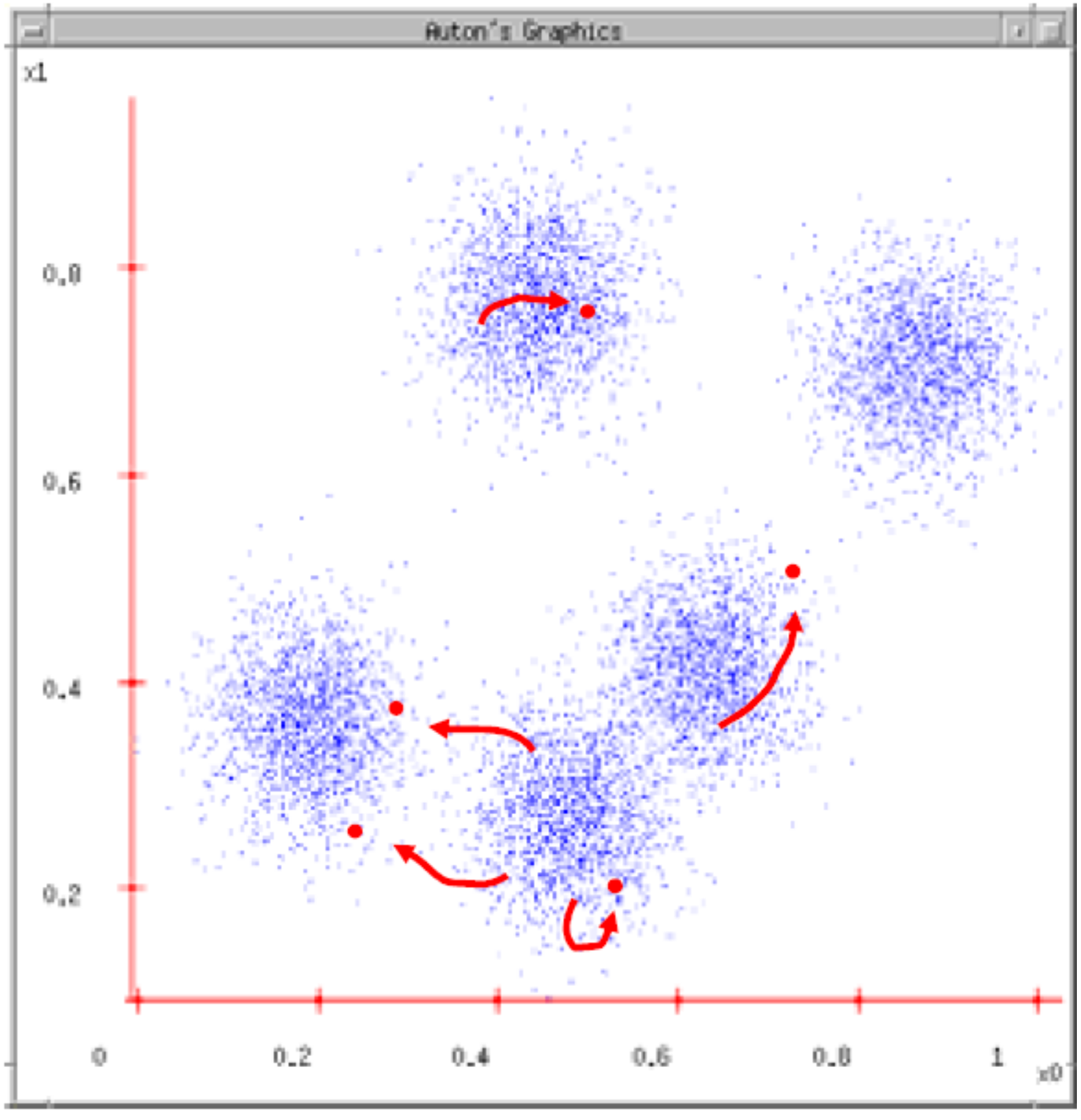
K-means

1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations
3. Each datapoint finds out which Center it's closest to.
4. Each Center finds the centroid of the points it owns



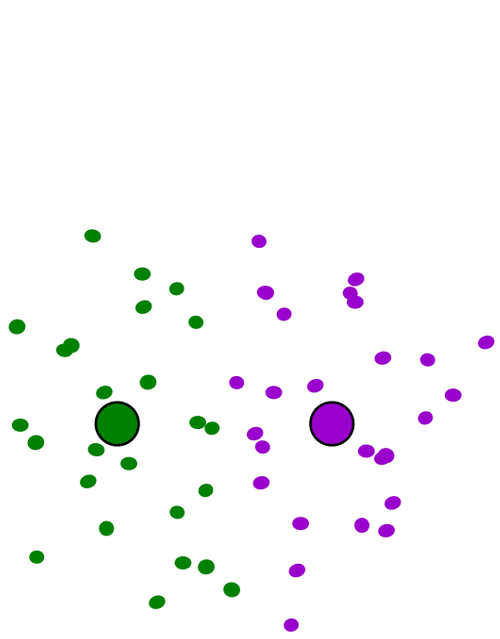
K-means

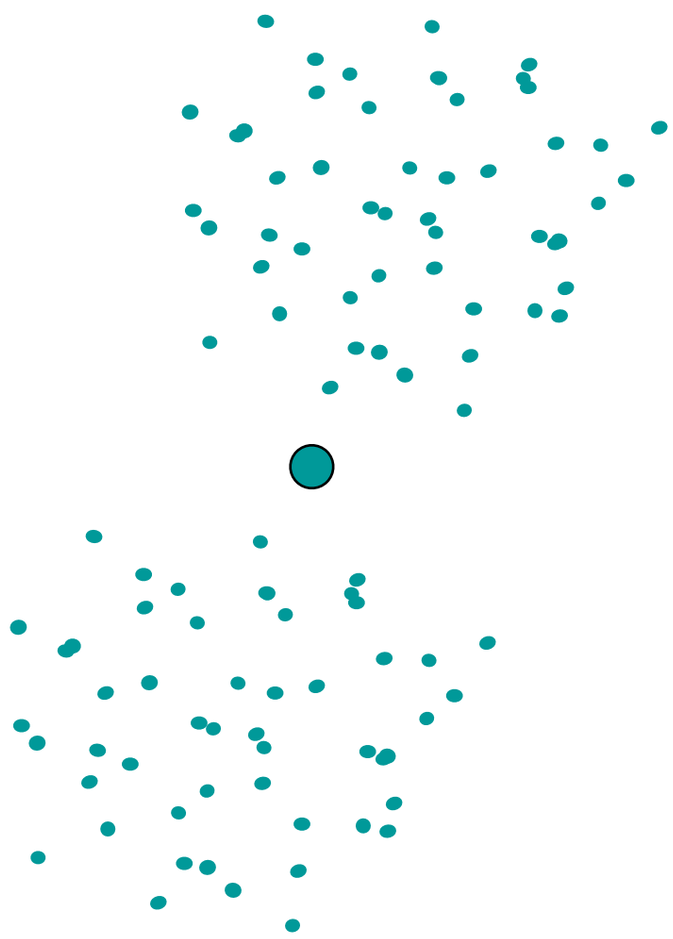
1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations
3. Each datapoint finds out which Center it's closest to.
4. Each Center finds the centroid of the points it owns...
5. ...and jumps there
6. ...Repeat until terminated!



K-Means getting stuck

- A local optimum:





Slide credit: Dan Klein

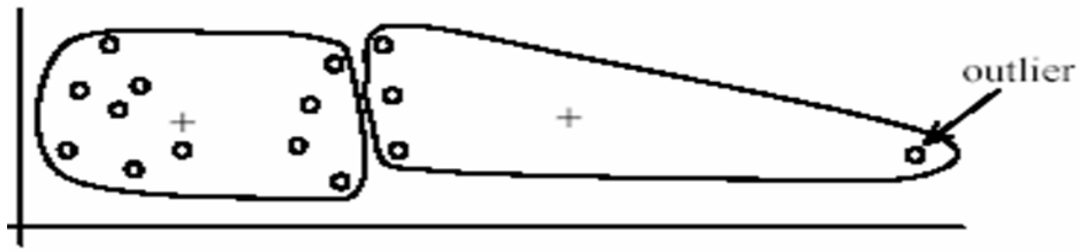
K-means: pros and cons

Pros

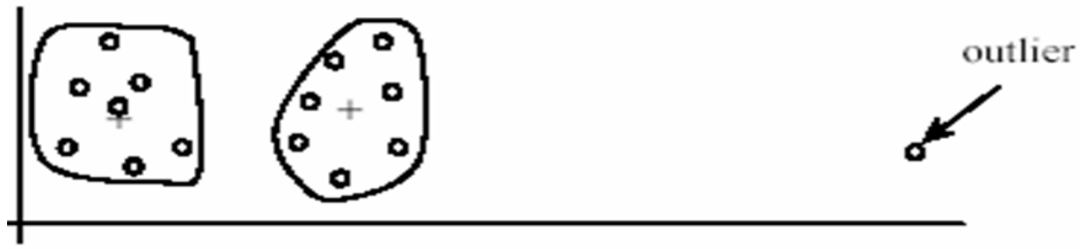
- Simple, fast to compute
- Converges to local minimum of within-cluster squared error

Cons/issues

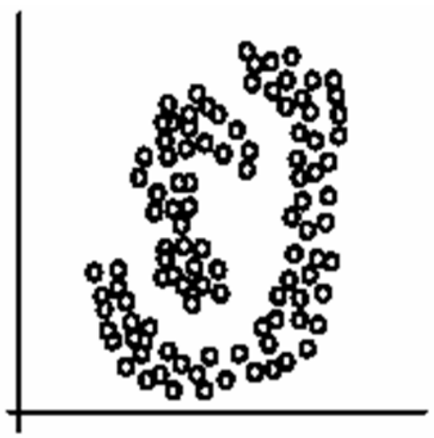
- Setting k?
- Sensitive to initial centers
- Sensitive to outliers
- Detects spherical clusters
- Assuming means can be computed



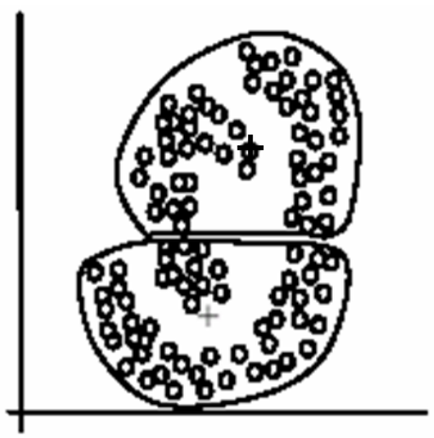
(A): Undesirable clusters



(B): Ideal clusters



(A): Two natural clusters



(B): *k*-means clusters

Probabilistic clustering

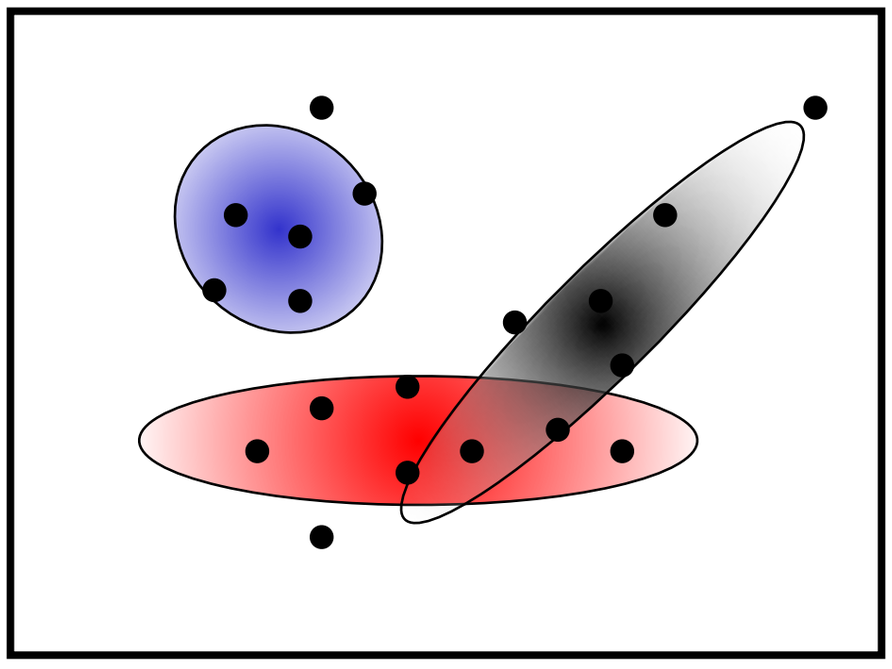
Basic questions

- what's the probability that a point **x** is in cluster m?
- what's the shape of each cluster?

K-means doesn't answer these questions

Probabilistic clustering (basic idea)

- Treat each cluster as a Gaussian density function

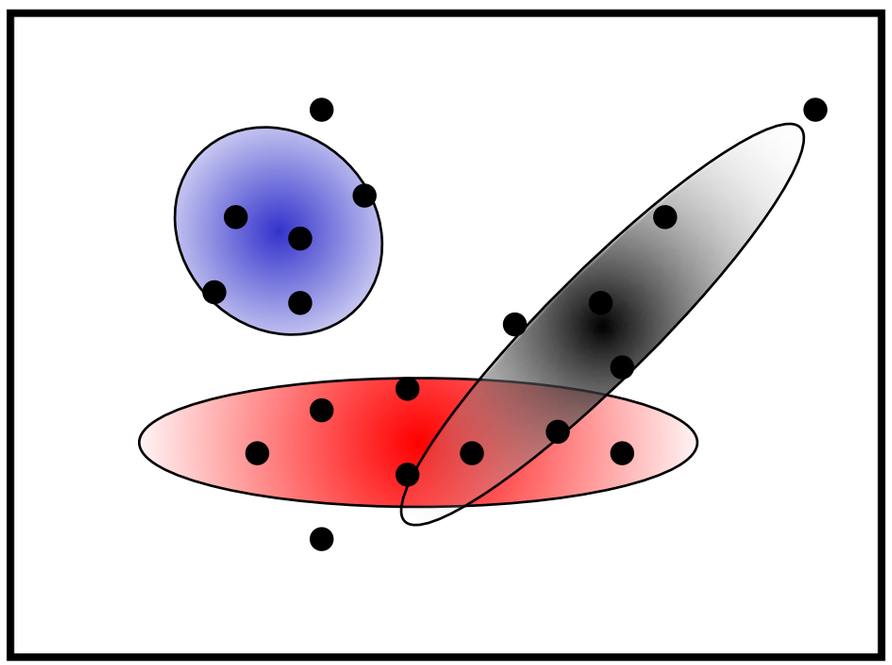


Slide credit: Steve Seitz

Expectation Maximization (EM)

A probabilistic variant of K-means:

- E step: "soft assignment" of points to clusters
  - estimate probability that a point is in a cluster
- M step: update cluster parameters
  - mean and variance info (covariance matrix)
- maximizes the likelihood of the points given the clusters



Slide credit: Steve Seitz

Segmentation as clustering

Depending on what we choose as the *feature space*, we can group pixels in different ways.

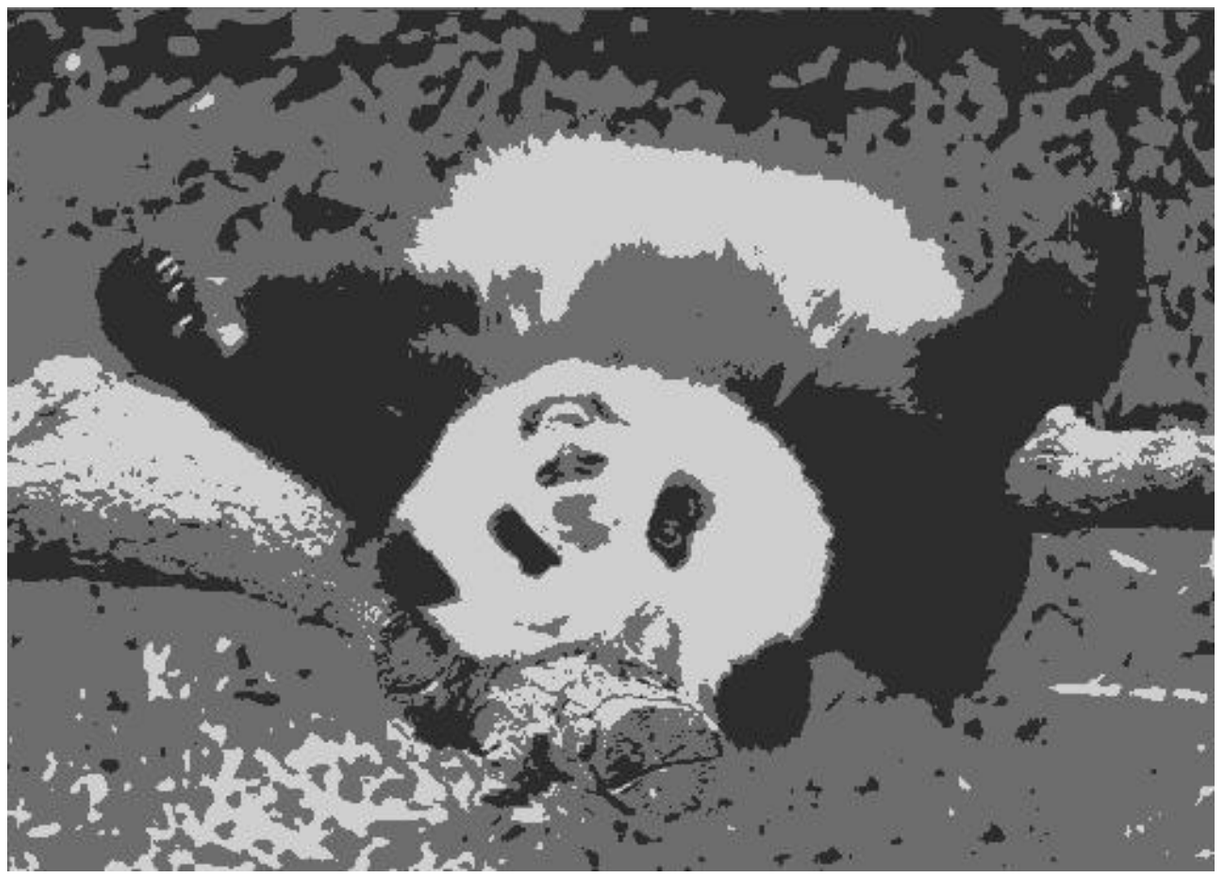
Grouping pixels based on **intensity** similarity

Feature space: intensity value (1-d)

Slide credit: Kristen Grauman

K=2

K=3



quantization of the feature space; segmentation label map

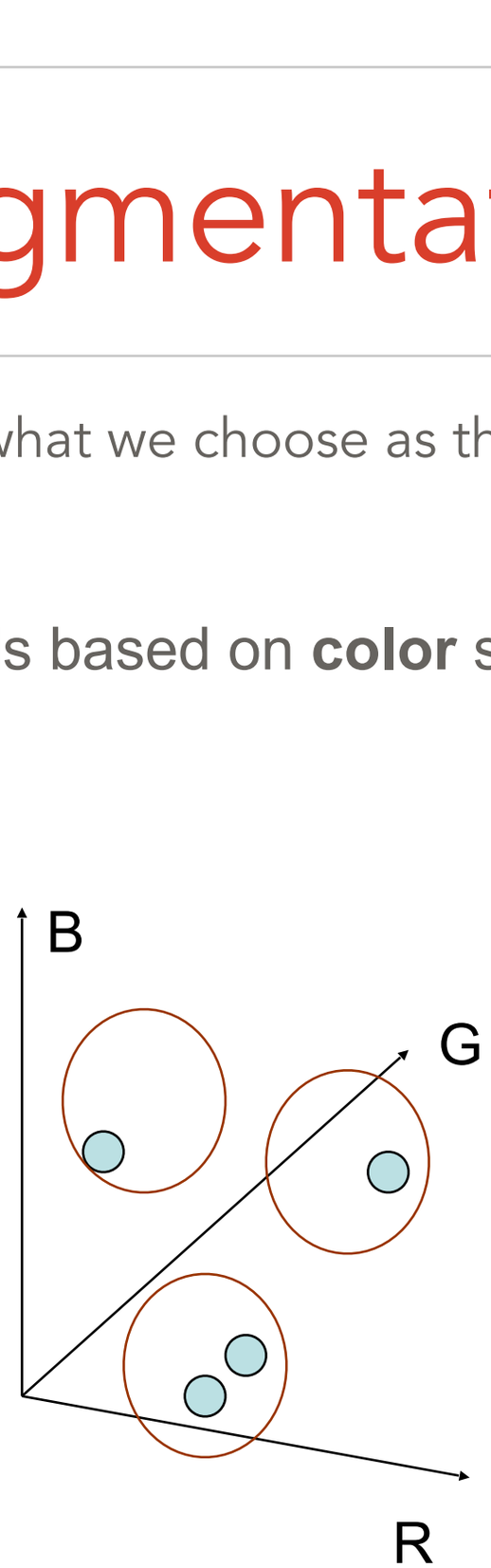
```
img_as_col = double(im(:));
cluster_membs = kmeans(img_as_col, K);

labelim = zeros(size(im));
for i=1:k
   inds = find(cluster_membs==i);
   meanval = mean(img_as_column(inds));
   labelim(inds) = meanval;
end
```

Slide credit: Kristen Grauman

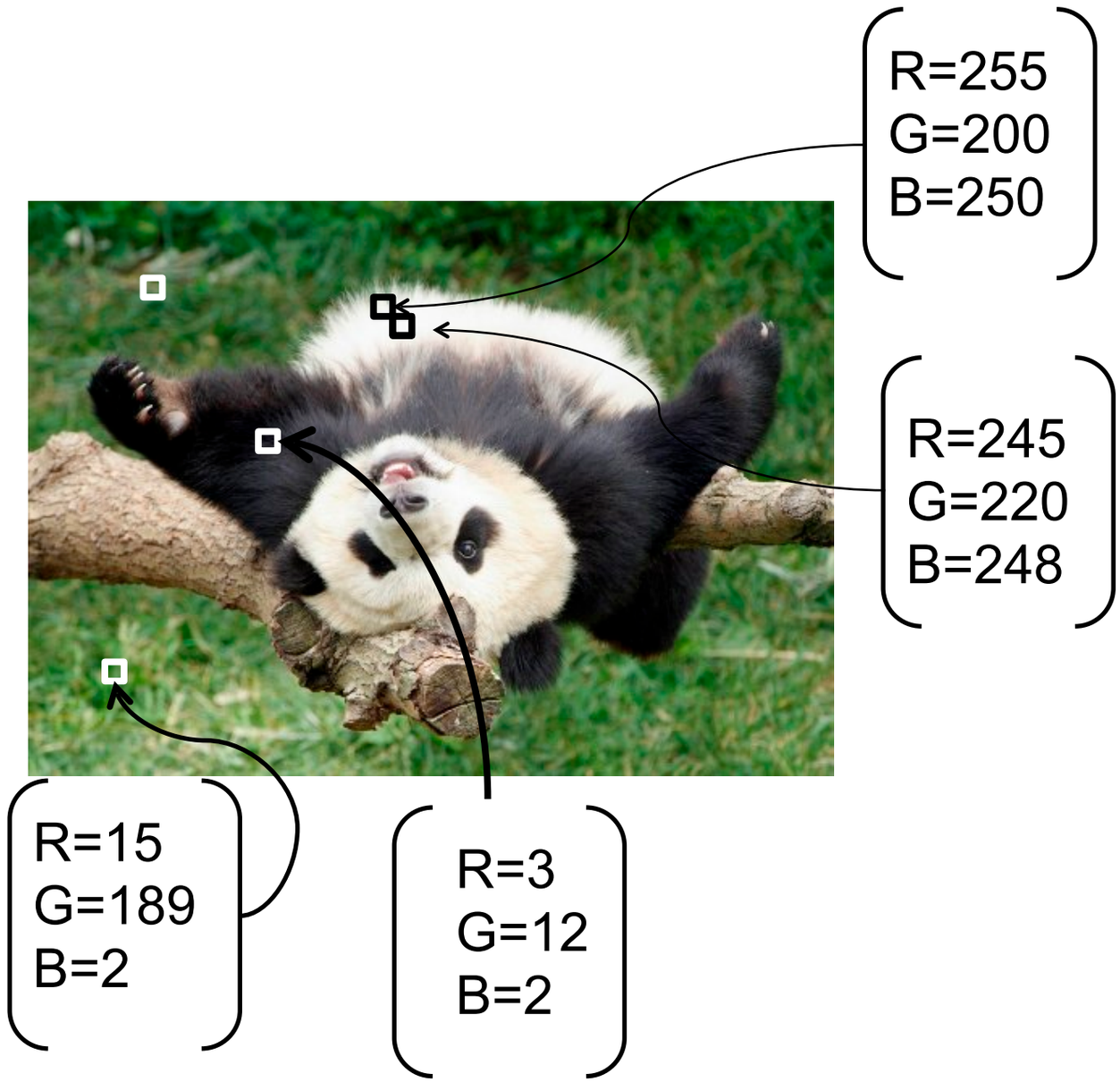
Segmentation as clustering

Depending on what we choose as the *feature space*, we can group pixels in different ways.

Grouping pixels based on **color** similarity



Feature space: color value (3-d)



Segmentation as clustering

Depending on what we choose as the *feature space*, we can group pixels in different ways.

Grouping pixels based on **intensity** similarity

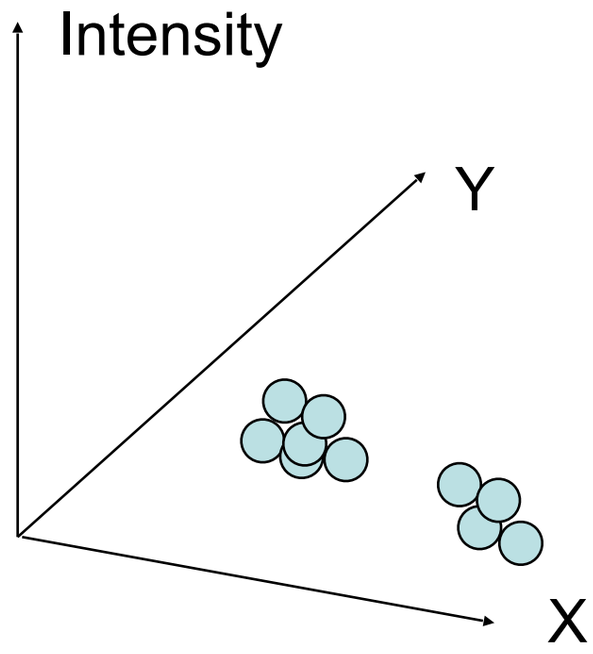
Clusters based on intensity similarity don't have to be spatially coherent.

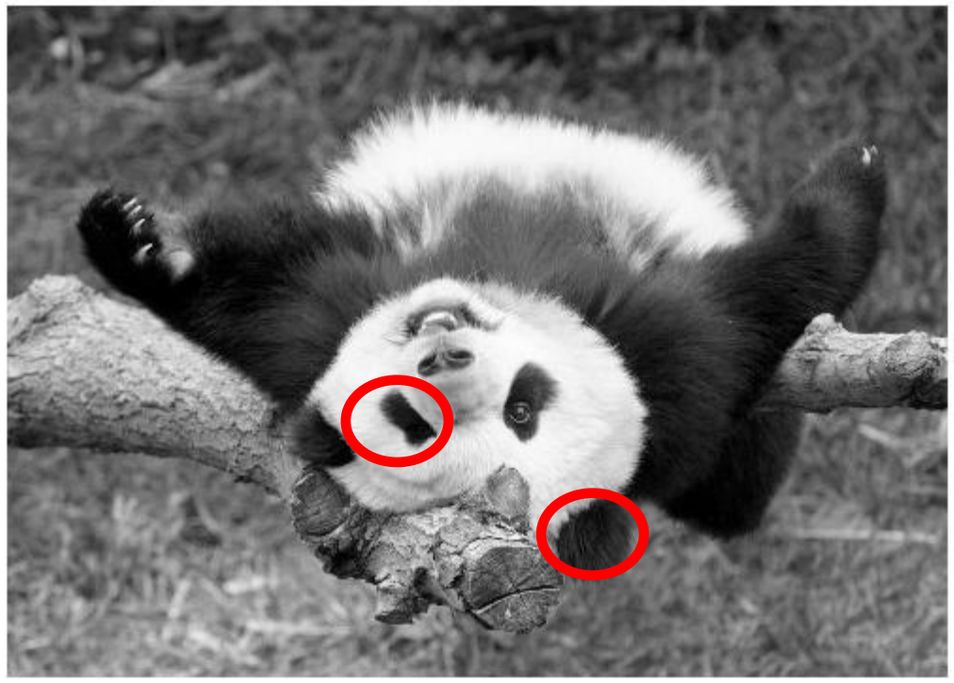
Slide credit: Kristen Grauman

Segmentation as clustering

Depending on what we choose as the *feature space*, we can group pixels in different ways.

Grouping pixels based on **intensity+position** similarity





Both regions are black, but if we also include **position (x,y),** then we could group the two into distinct segments; way to encode both similarity & proximity.

Slide credit: Kristen Grauman

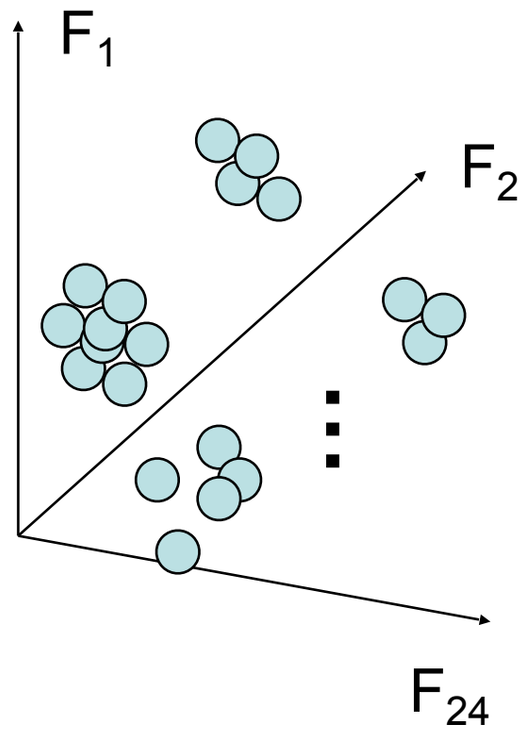
Segmentation as clustering

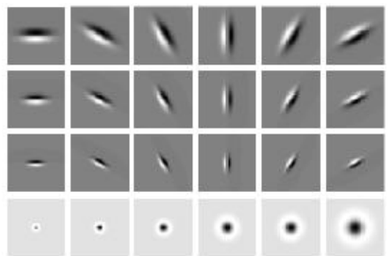
- Color, brightness, position alone are not enough to distinguish all regions…

Segmentation as clustering

Depending on what we choose as the *feature space*, we can group pixels in different ways.

Grouping pixels based on **texture** similarity



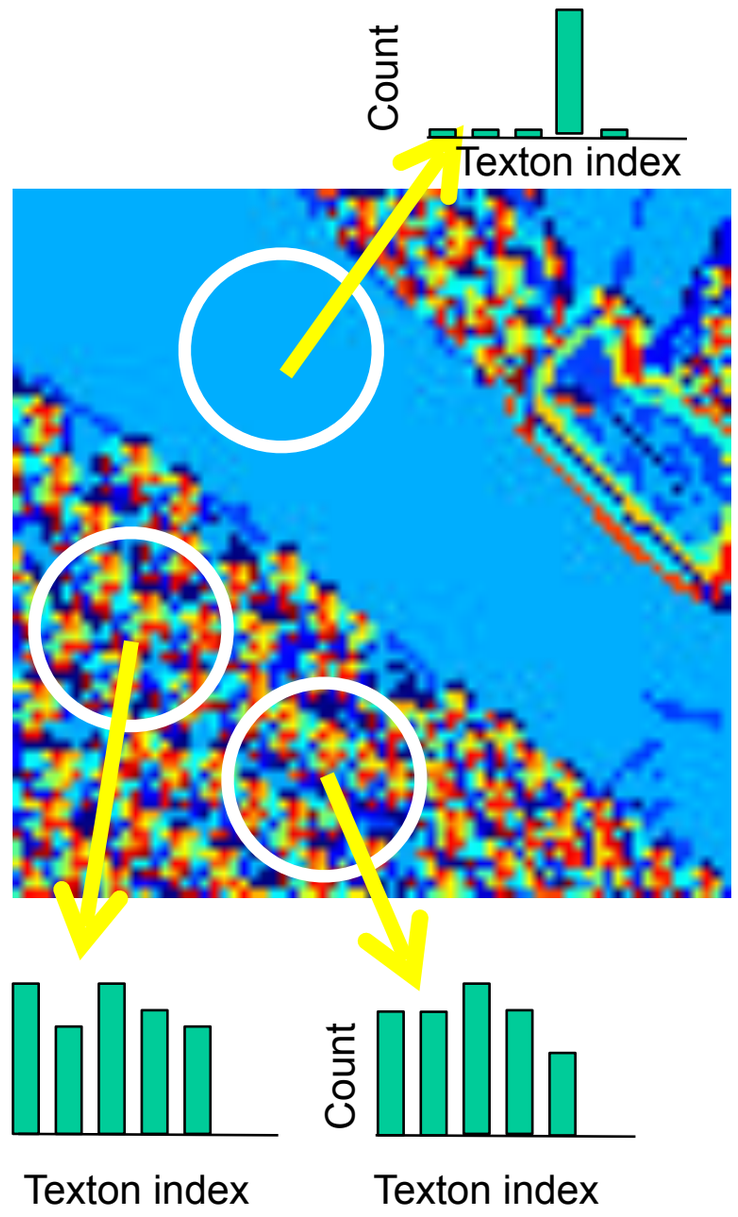


Filter bank of 24 filters

Feature space: filter bank responses (e.g., 24-d)

Segmentation with texture features

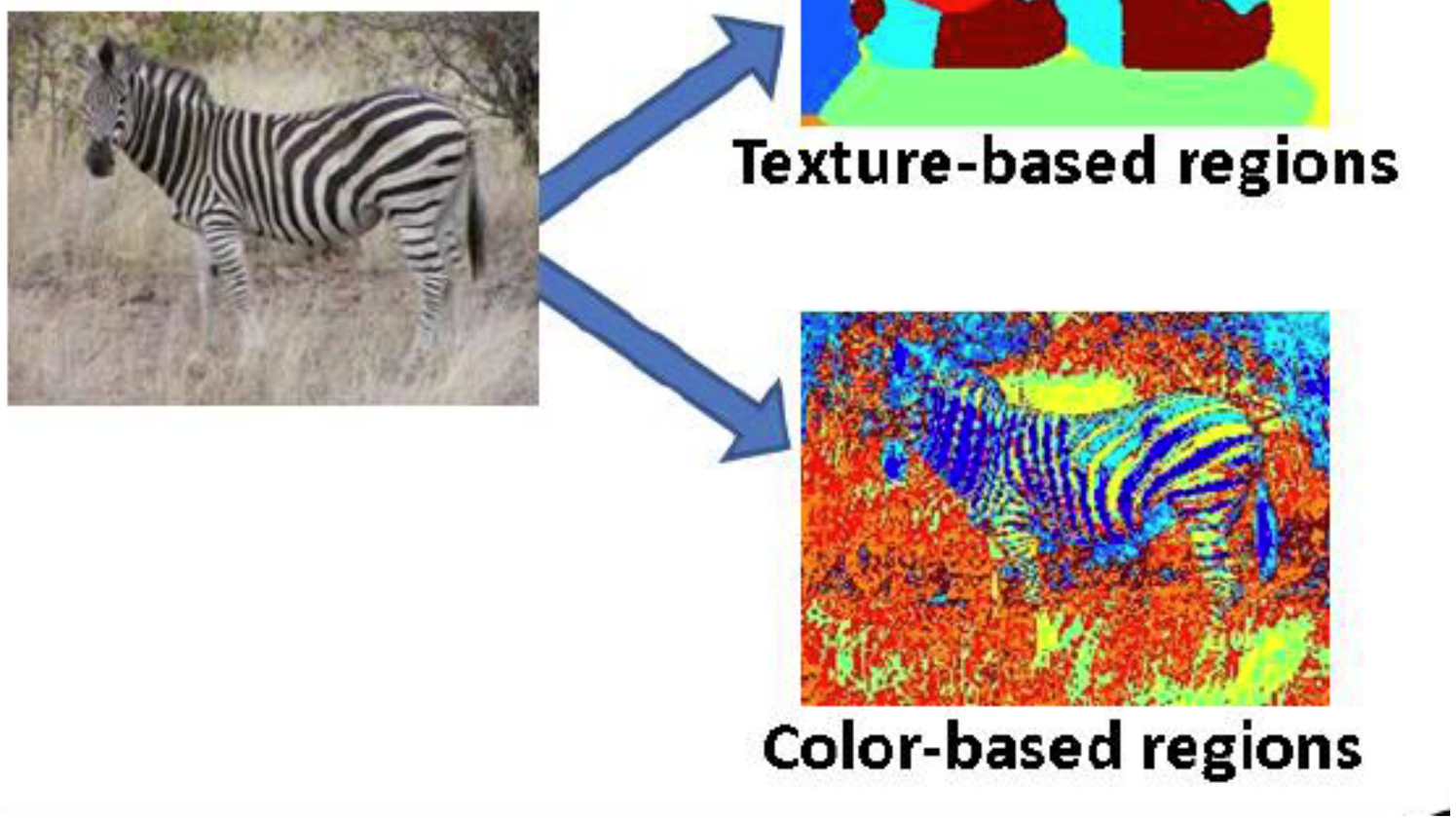
- Find "textons" by **clustering** vectors of filter bank outputs
- Describe texture in a window based on *texton histogram*



Malik, Belongie, Leung and Shi. IJCV 2001.

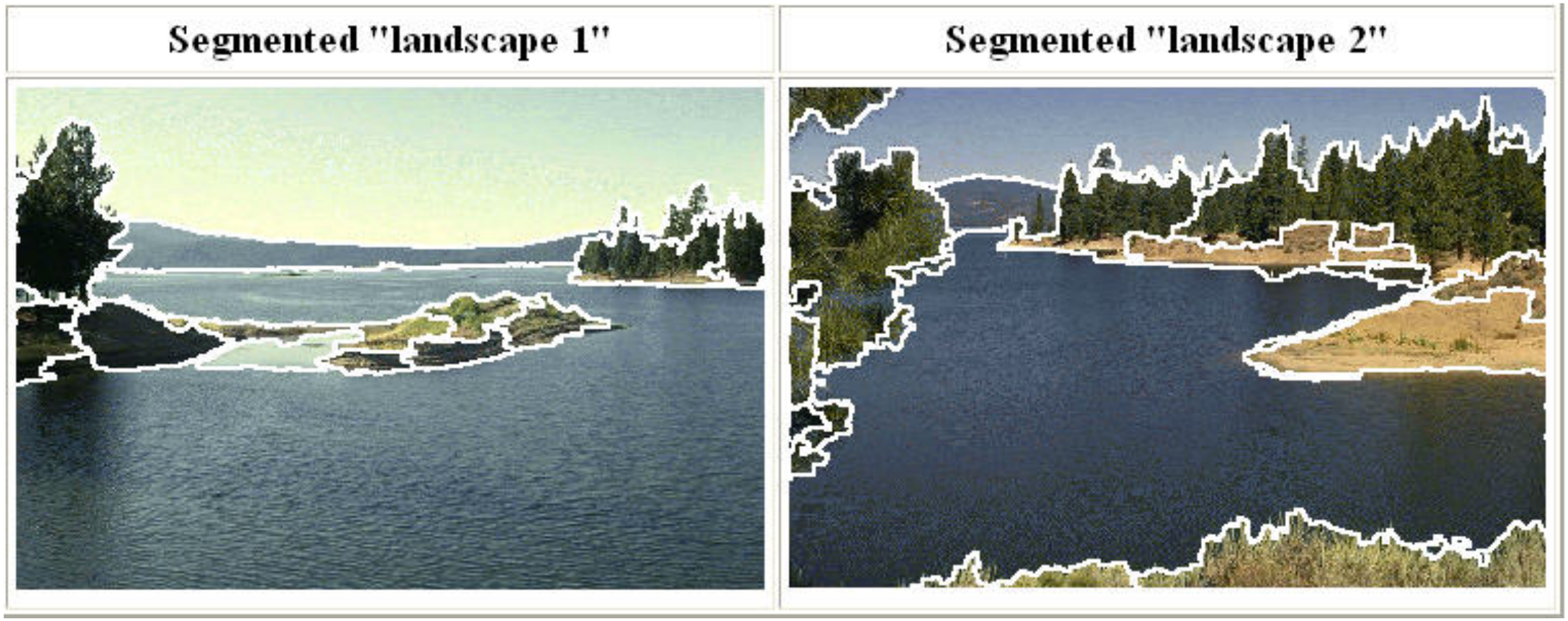
Adapted from Lana Lazebnik

Image segmentation example



Mean shift clustering and segmentation

- An advanced and versatile technique for clustering-based segmentation



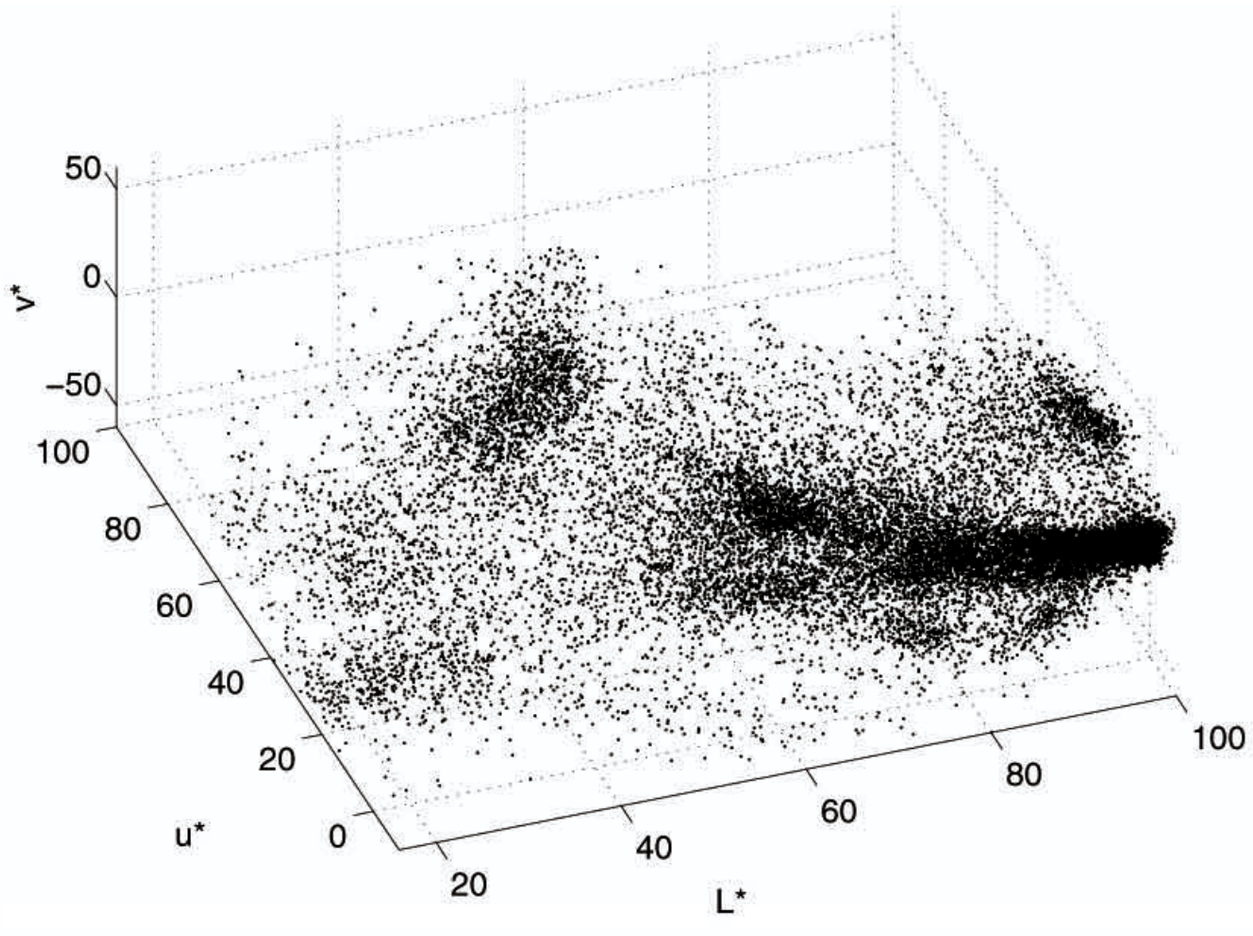
http://www.caip.rutgers.edu/~comanici/MSPAMI/msPamiResults.html

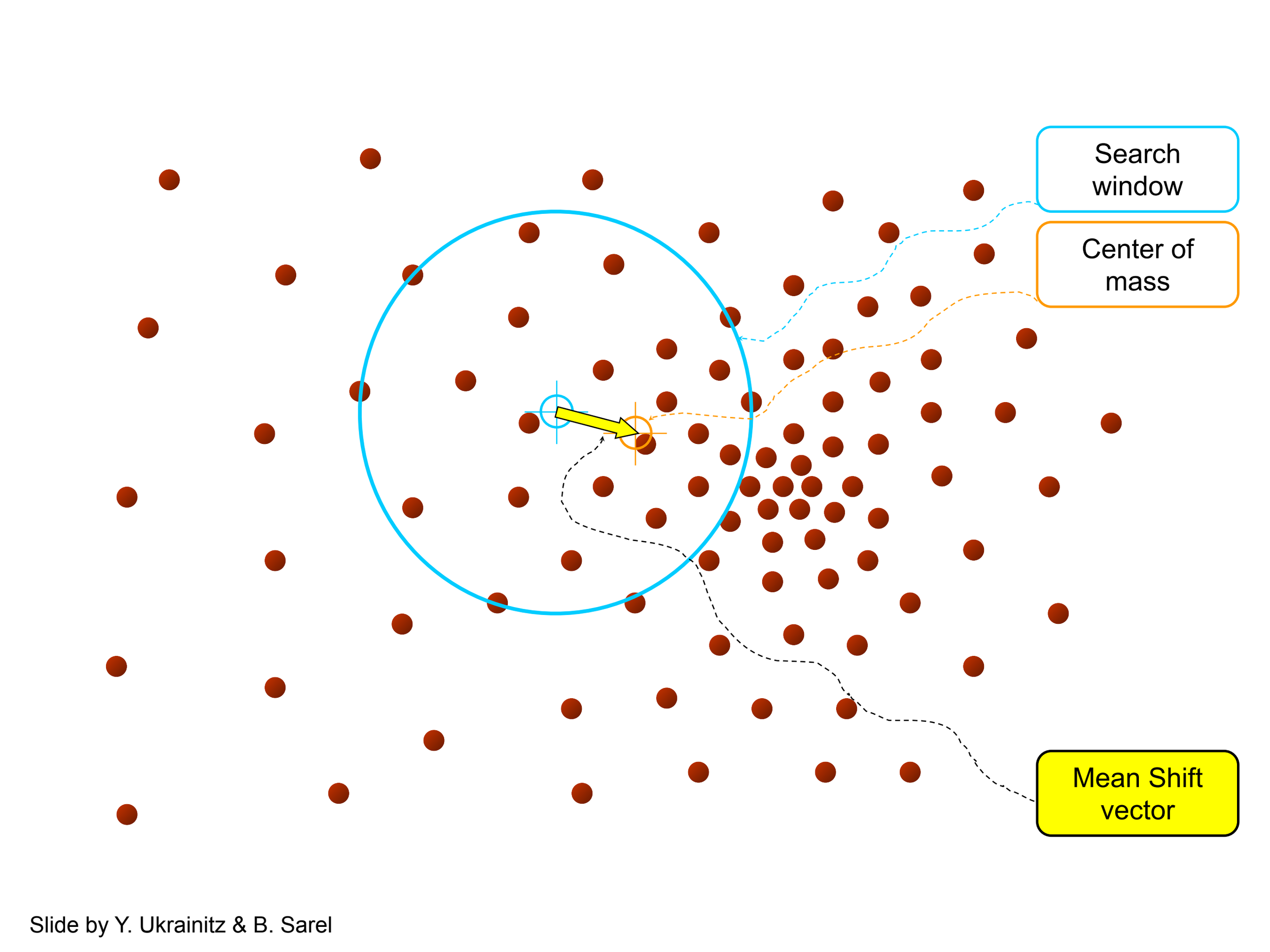
D. Comaniciu and P. Meer, Mean Shift: A Robust Approach toward Feature Space Analysis, PAMI 2002.

Mean shift algorithm

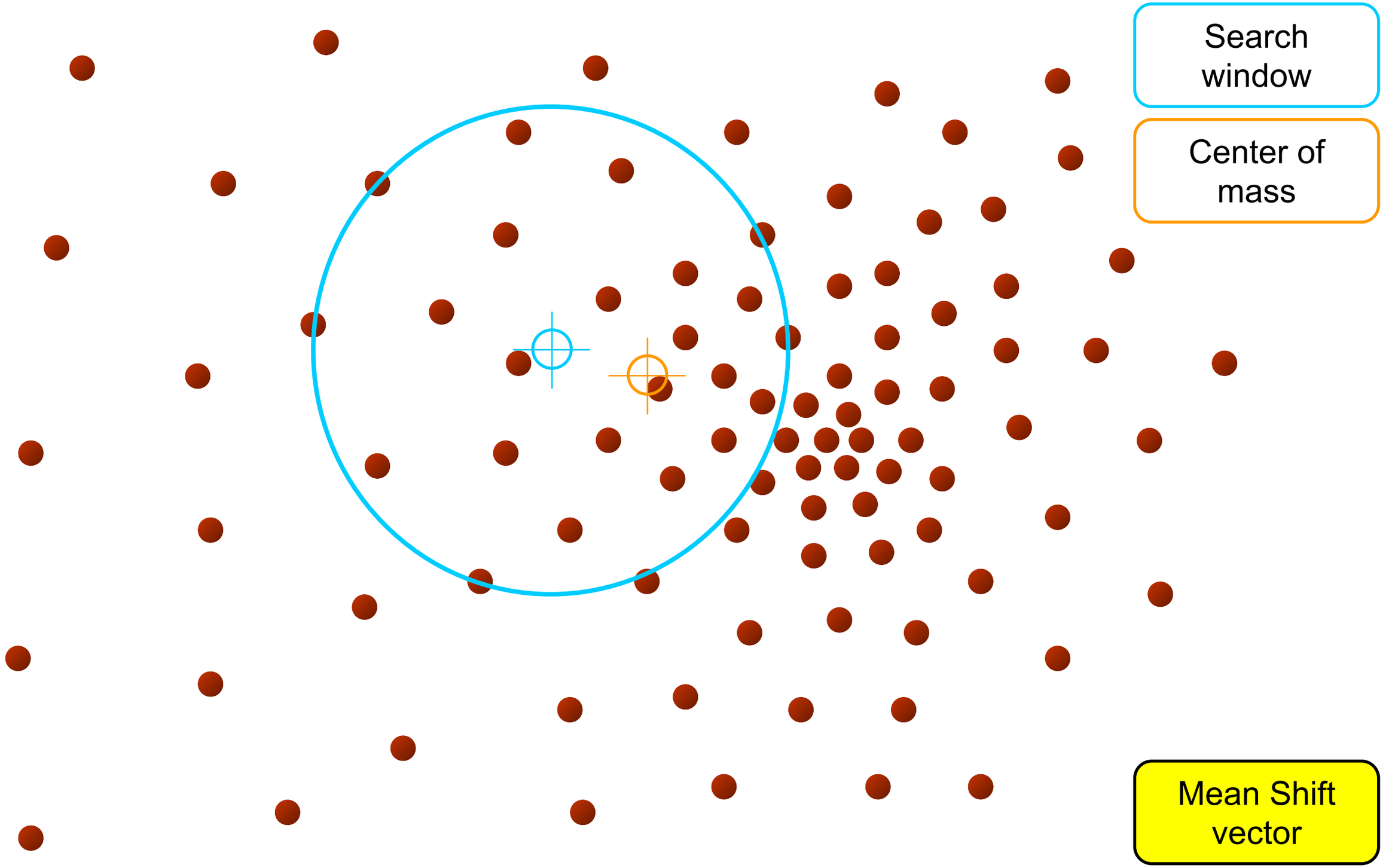
image

Feature space
(L*u*v* color values)

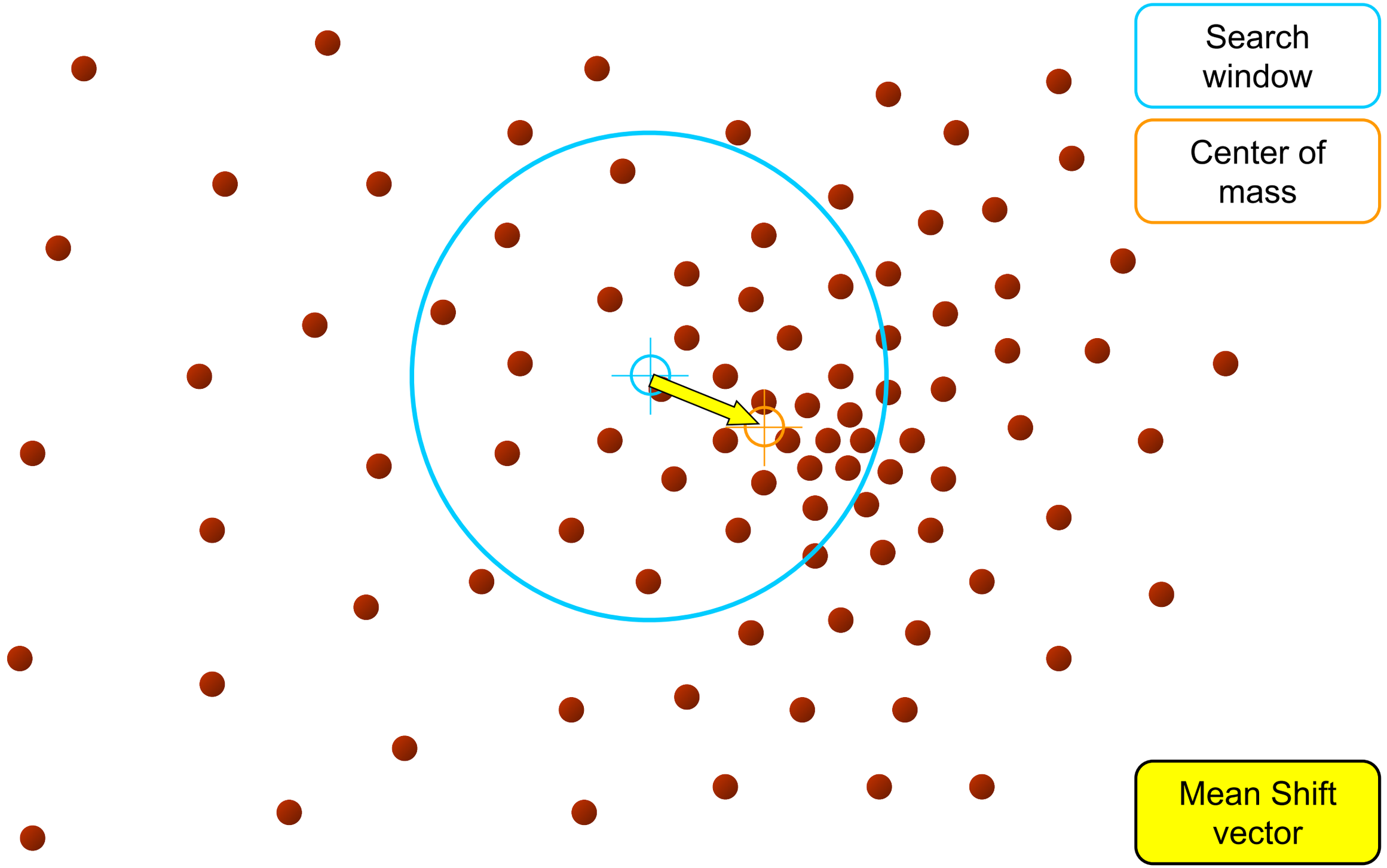




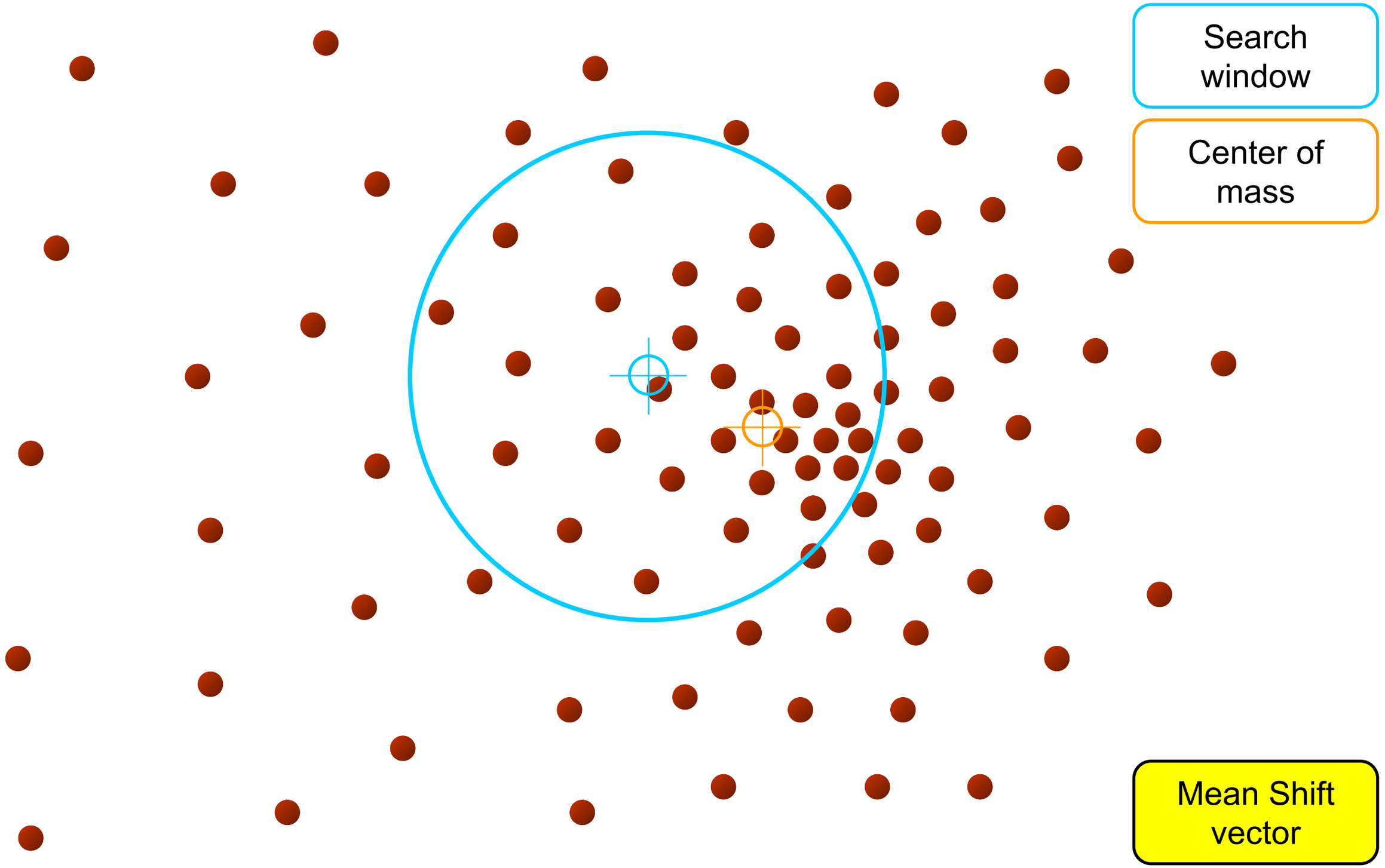
Slide by Y. Ukrainitz & B. Sarel



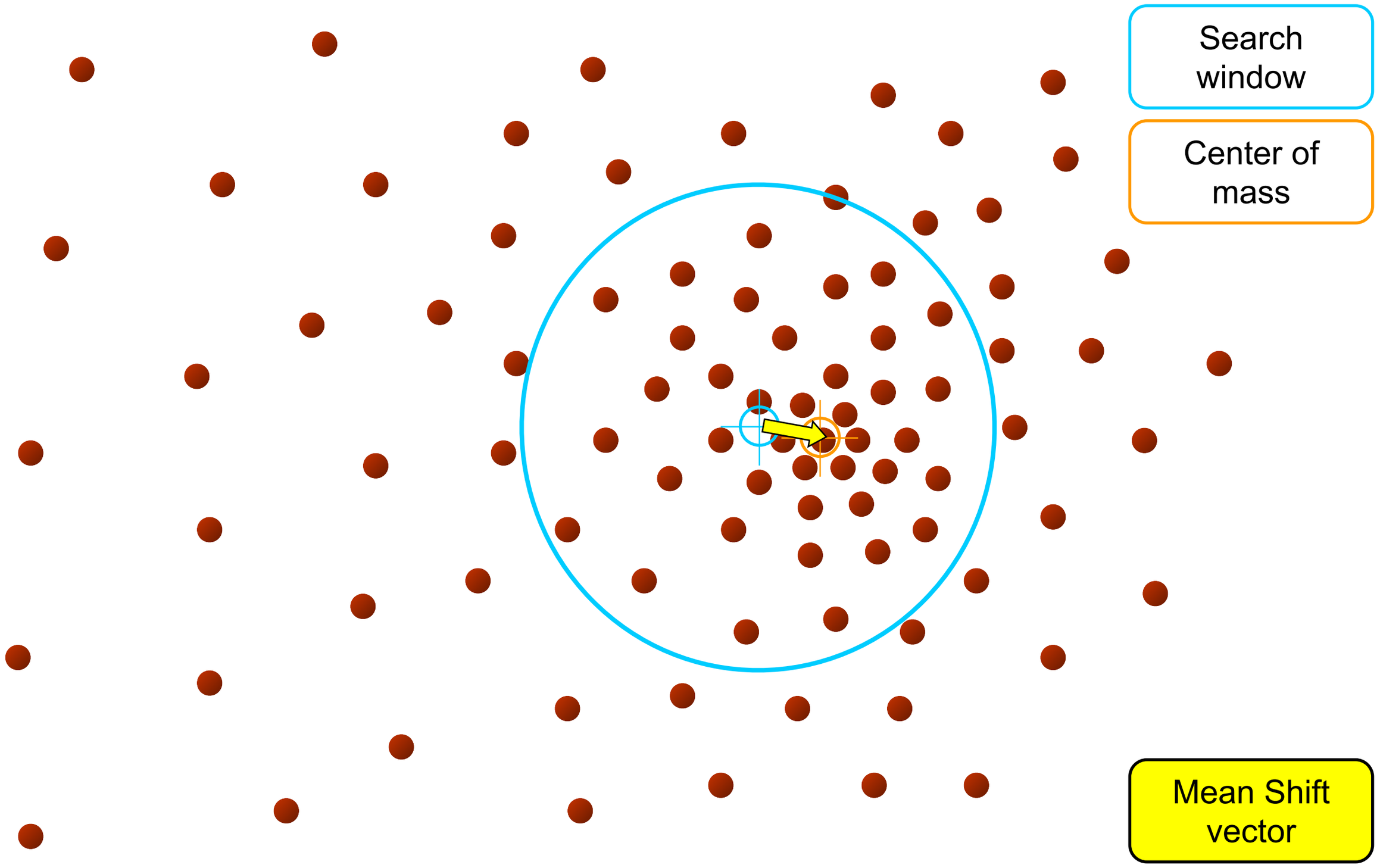
Slide by Y. Ukrainitz & B. Sarel



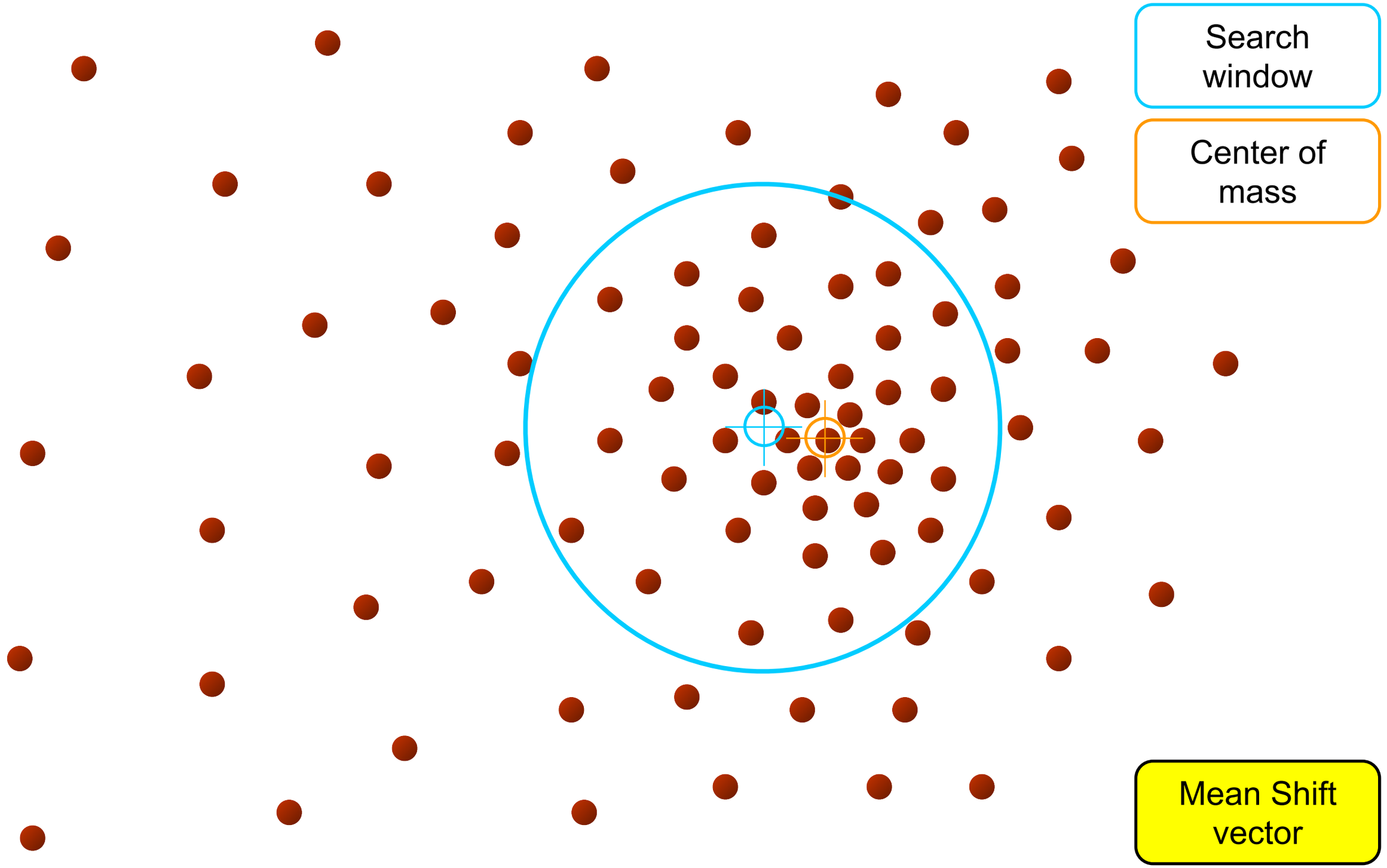
Slide by Y. Ukrainitz & B. Sarel



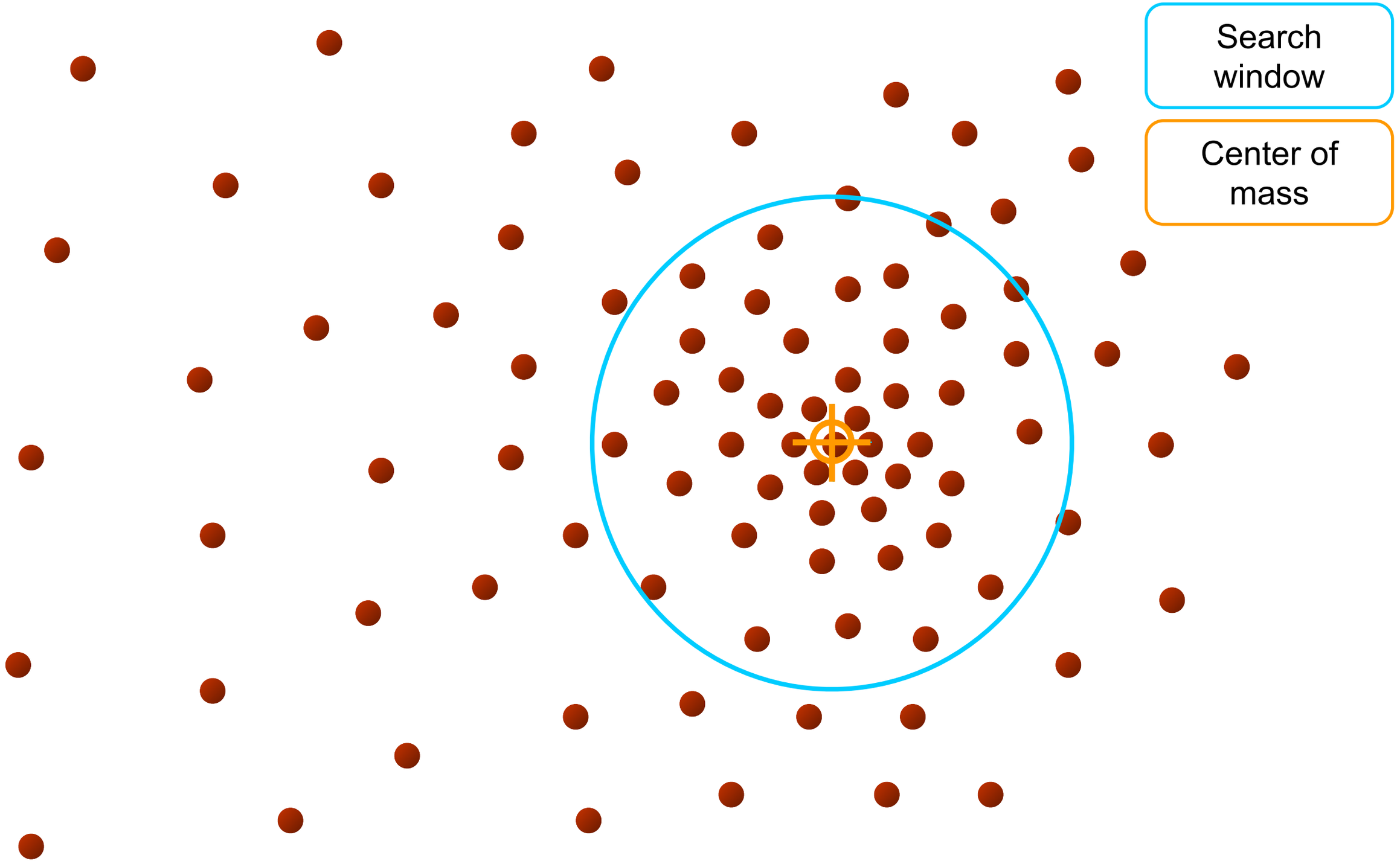
Slide by Y. Ukrainitz & B. Sarel



Slide by Y. Ukrainitz & B. Sarel



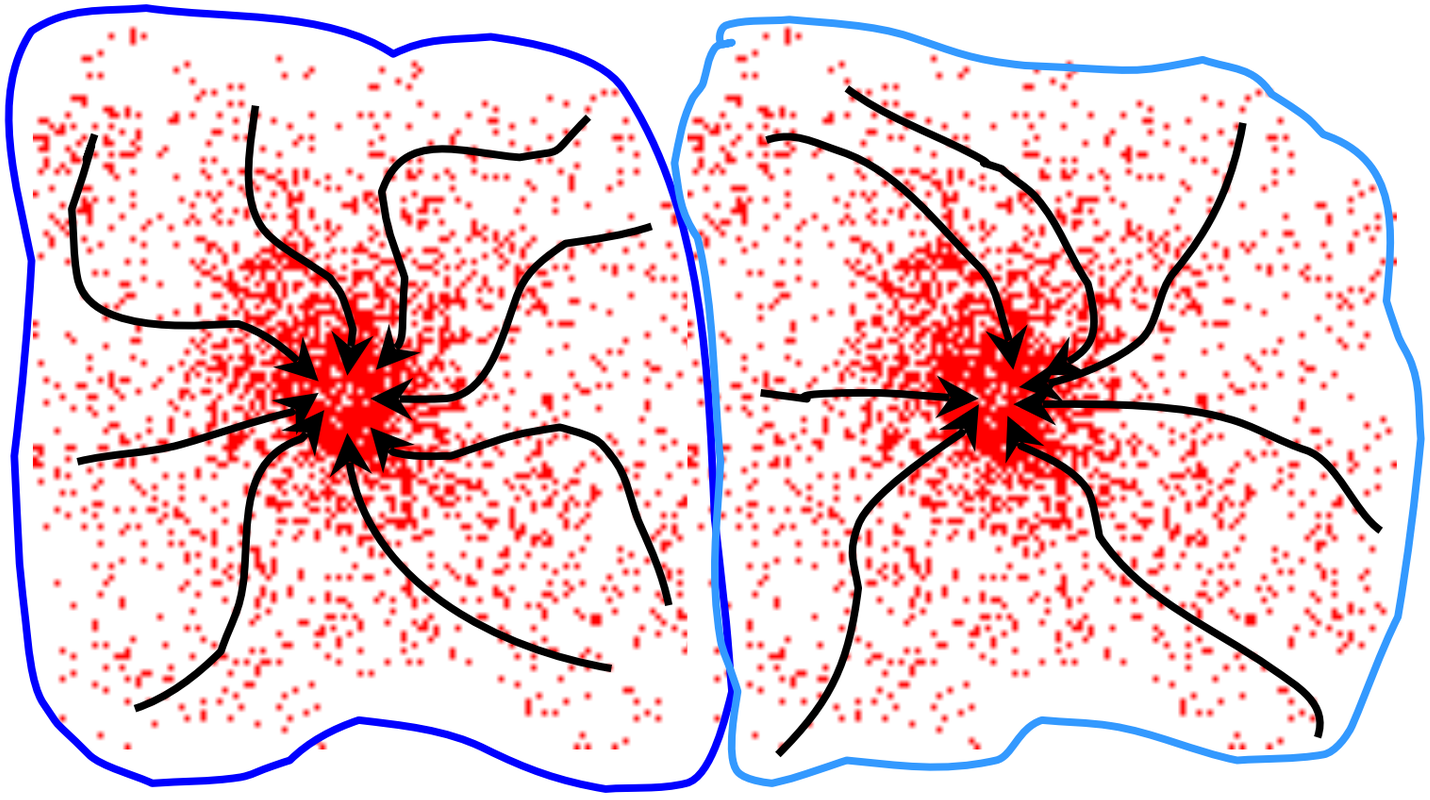
Slide by Y. Ukrainitz & B. Sarel



Slide by Y. Ukrainitz & B. Sarel

Mean shift clustering

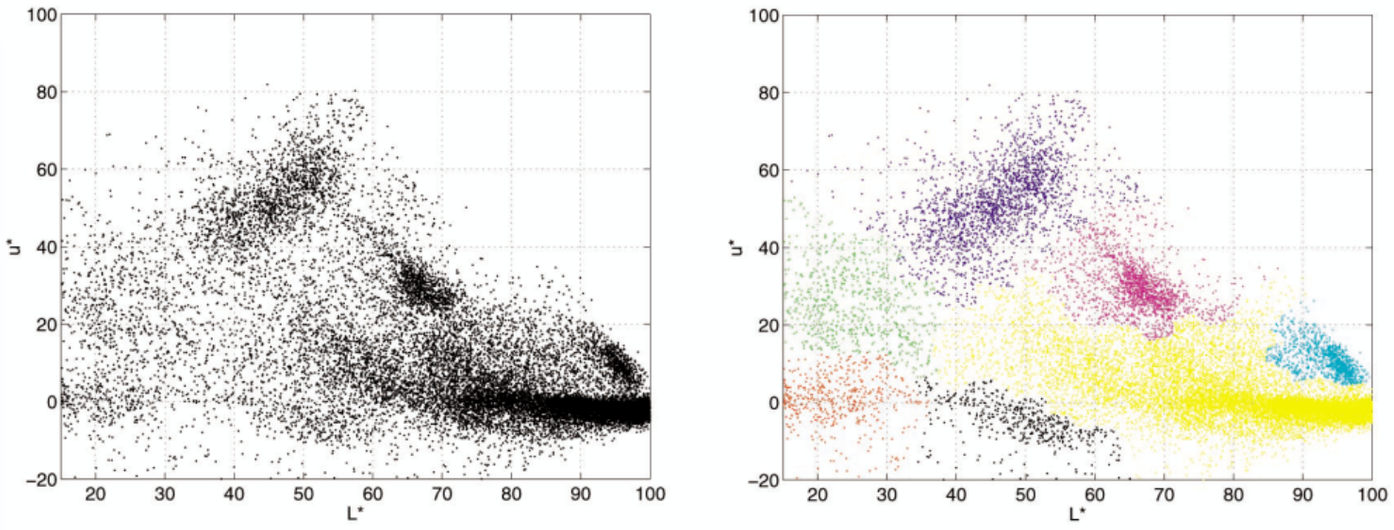
- Cluster: all data points in the attraction basin of a mode
- Attraction basin: the region for which all trajectories lead to the same mode



Slide by Y. Ukrainitz & B. Sarel

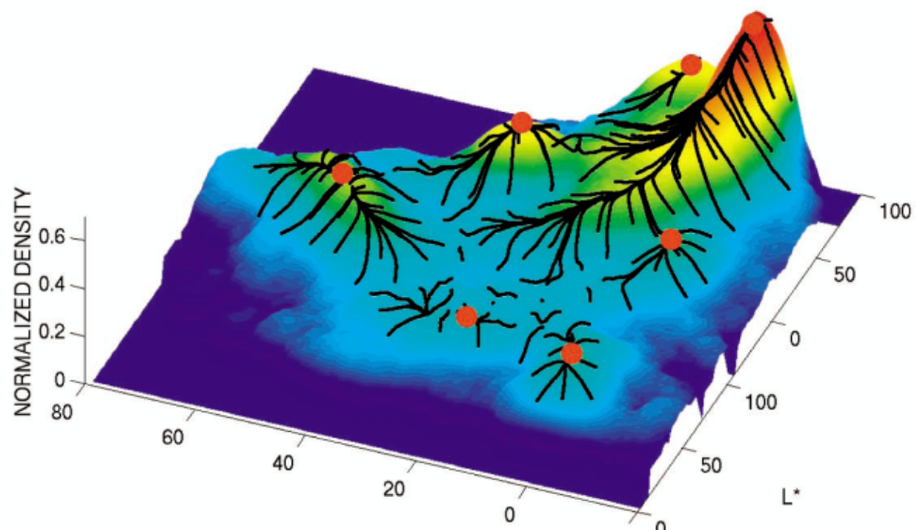
Mean shift clustering/segmentation

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

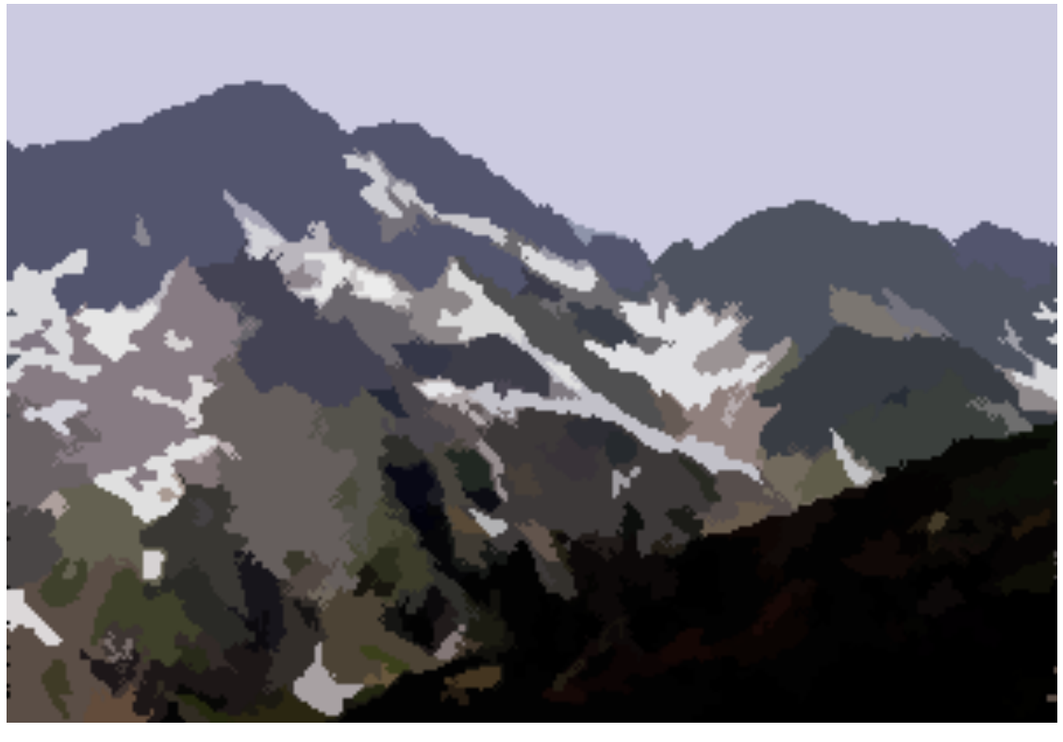


(a)

(b)



Mean shift segmentation results



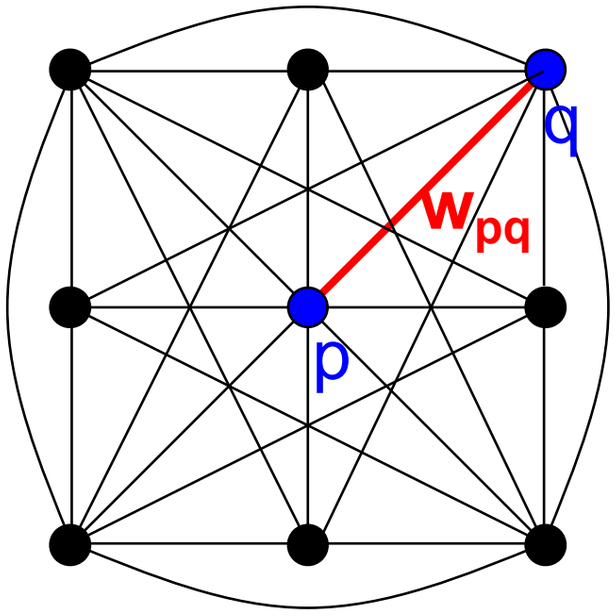
Mean shift segmentation results

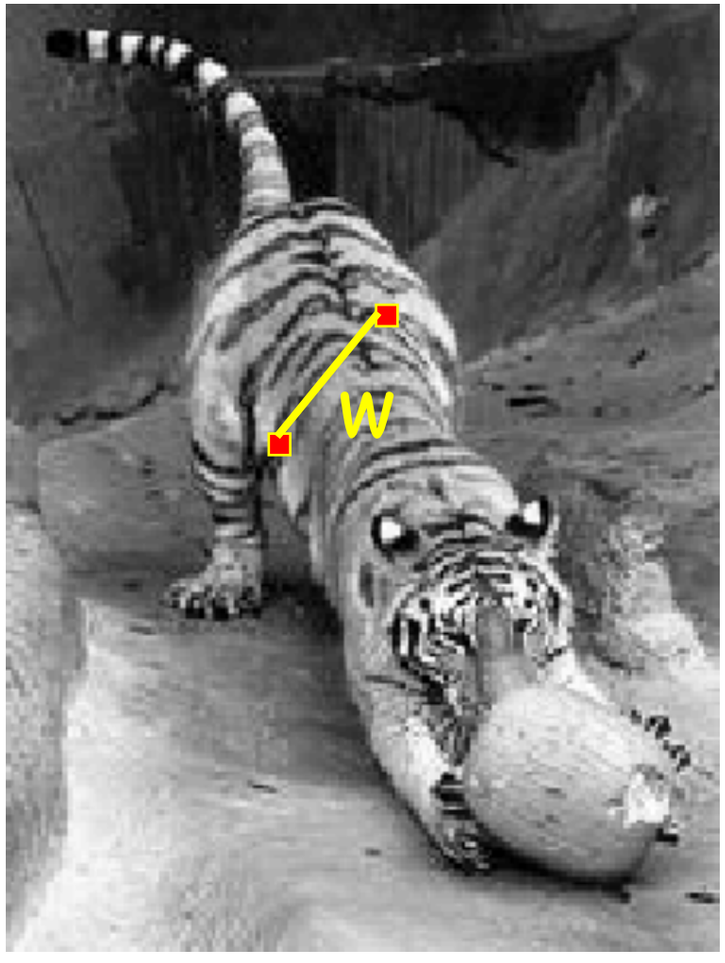
Mean shift segmentation results

Mean shift

- Pros:
  - Does not assume shape on clusters
  - One parameter choice (window size, aka “bandwidth”)
  - Generic technique
  - Find multiple modes
- Cons:
  - Selection of window size
  - Does not scale well with dimension of feature space

Images as graphs



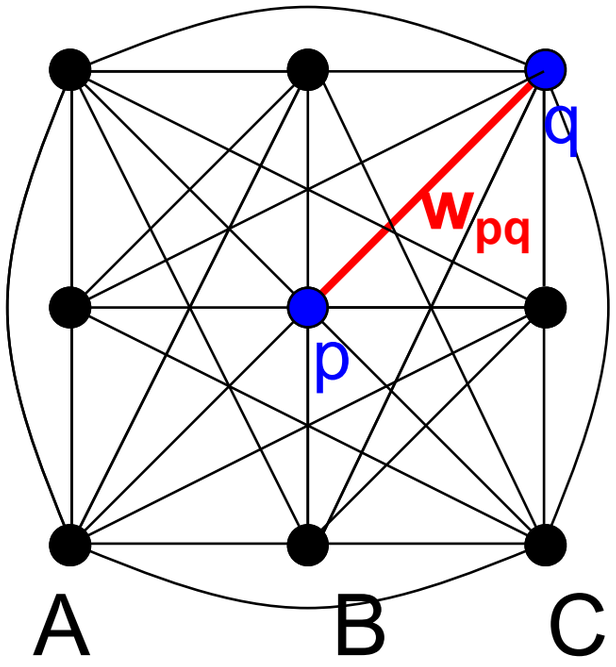


Fully-connected graph

- node (vertex) for every pixel
- link between *every* pair of pixels, **p**,**q**
- affinity weight $\mathbf{w_{pq}}$ for each link (edge)
  - $\mathbf{w_{pq}}$ measures *similarity*
    - similarity is *inversely proportional* to difference (in color and position…)

Source: Steve Seitz

Segmentation by Graph Cuts



Break Graph into Segments

- Want to delete links that cross **between** segments
- Easiest to break links that have low similarity (low weight)
  - similar pixels should be in the same segments
  - dissimilar pixels should be in different segments

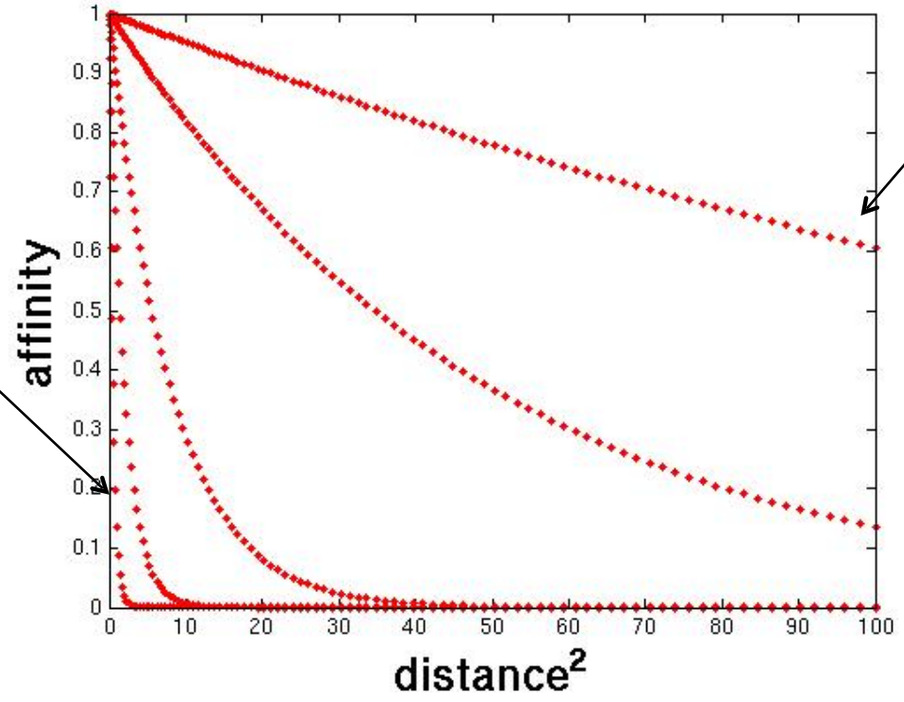
Source: Steve Seitz

Measuring affinity

- One possibility:

$$aff(x,y) = \exp\left\{-\left(\frac{1}{2\sigma_d^2}\right)\left(\|x-y\|^2\right)\right\}$$

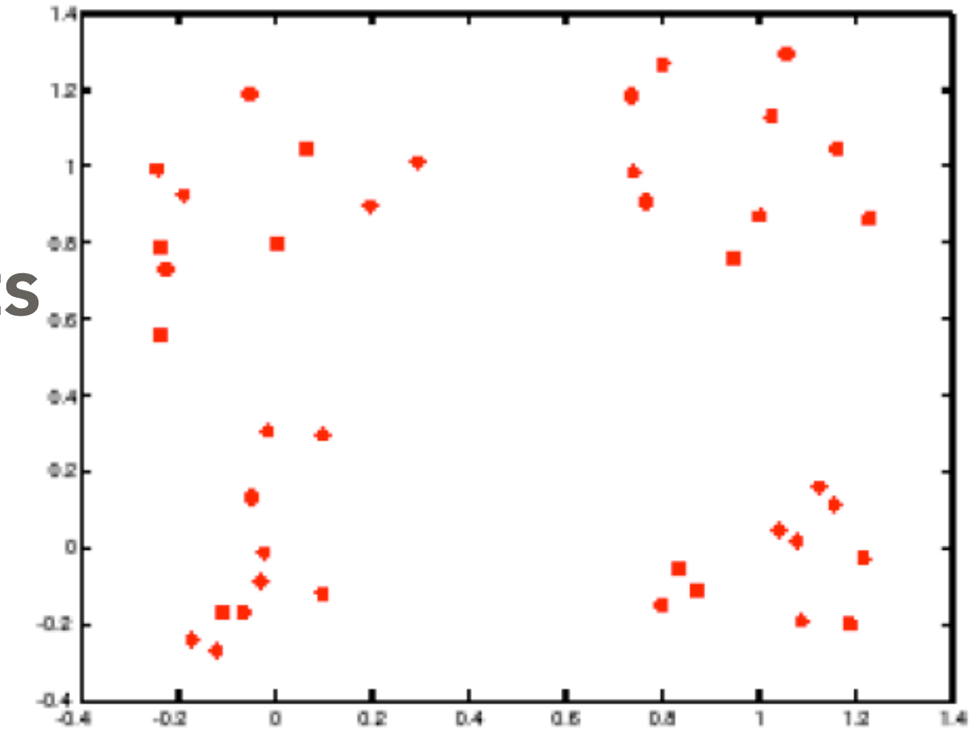
Small sigma: group only nearby points



Large sigma: group distant points

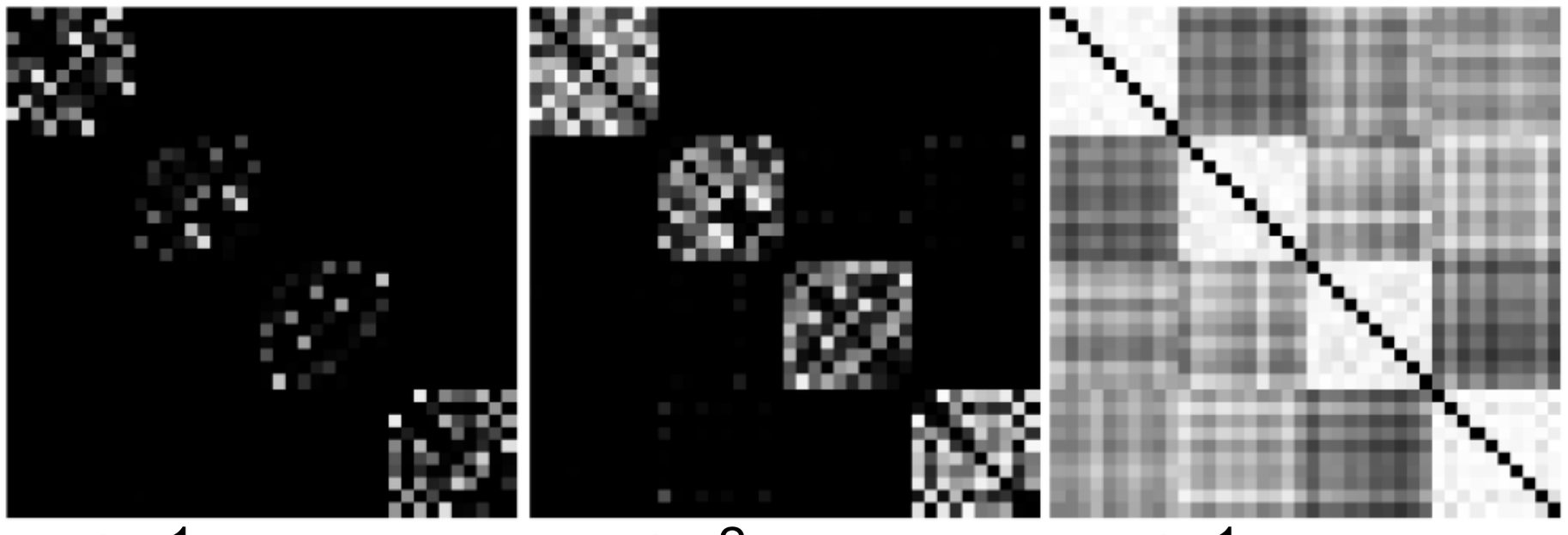
Measuring affinity

Data points



σ=.2

Affinity matrices



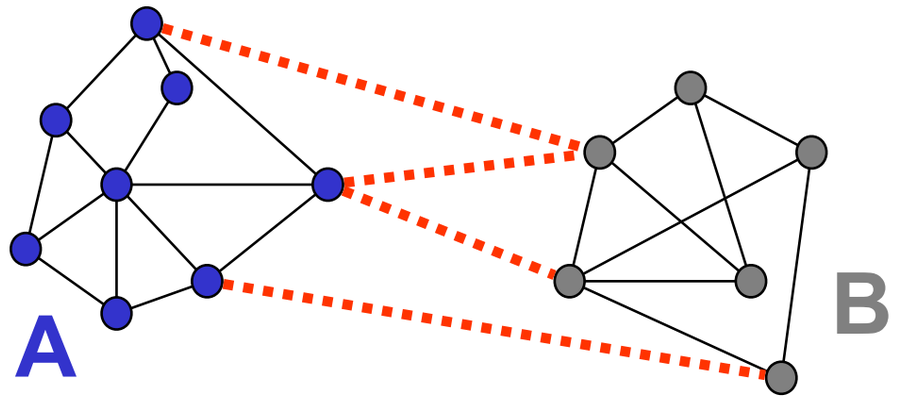
σ=.1 σ=.2 σ=1

Slide credit: Kristen Grauman

Cuts in a graph: Min cut

Link Cut

- set of links whose removal makes a graph disconnected
- cost of a cut:



$$cut(A,B) = \sum_{p \in A, q \in B} w_{p,q}$$

Find minimum cut

- gives you a segmentation
- fast algorithms exist for doing this

Minimum cut

- Problem with minimum cut:

  Weight of cut proportional to number of edges in the cut; tends to produce small, isolated components.

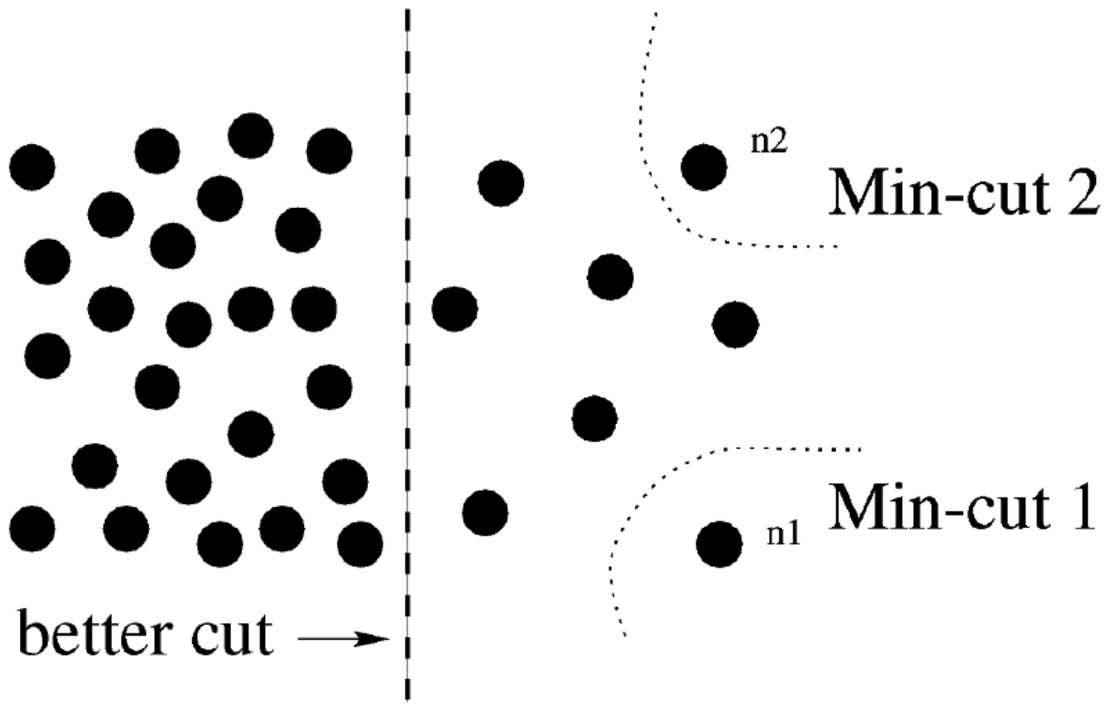
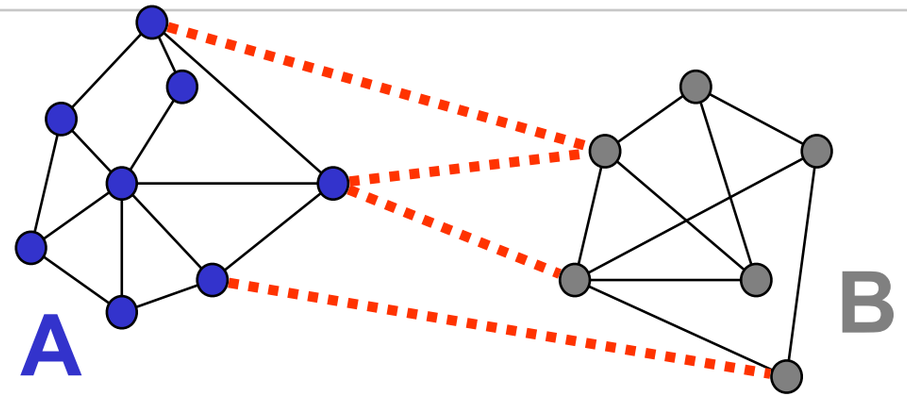


Fig. 1. A case where minimum cut gives a bad partition.

[Shi & Malik, 2000 PAMI]

Cuts in a graph: Normalized cut

Normalized Cut



- fix bias of Min Cut by **normalizing** for size of segments:

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

assoc(A,V) = sum of weights of all edges that touch A

- Ncut value small when we get two clusters with many edges with high weights, and few edges of low weight between them
- Approximate solution for minimizing the Ncut value : generalized eigenvalue problem.

Source: Steve Seitz



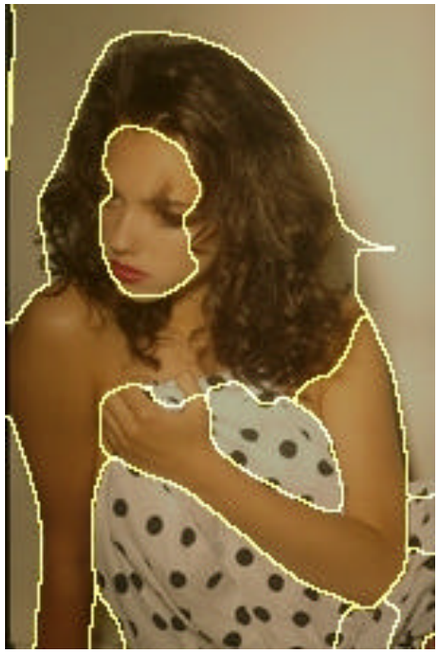


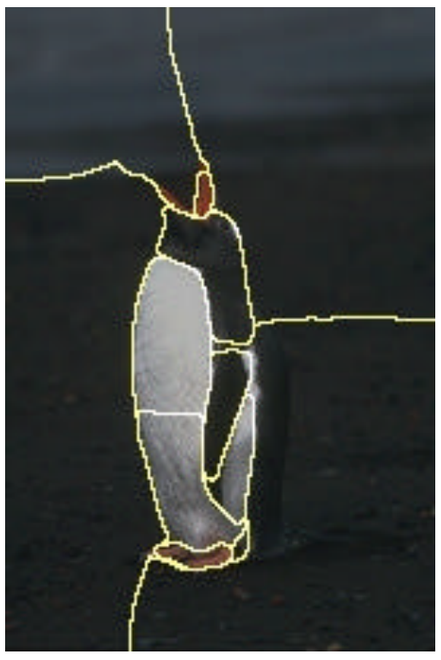
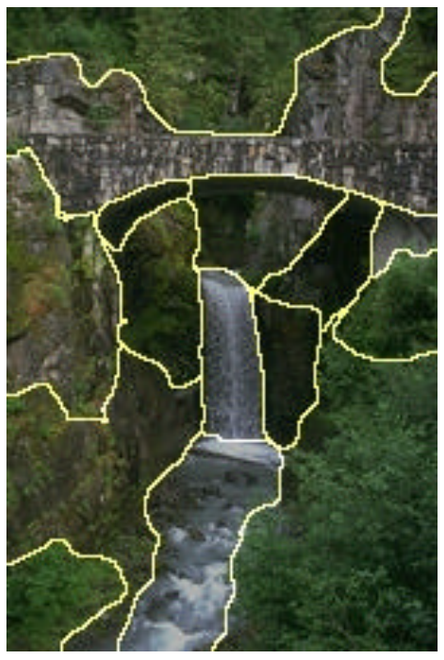


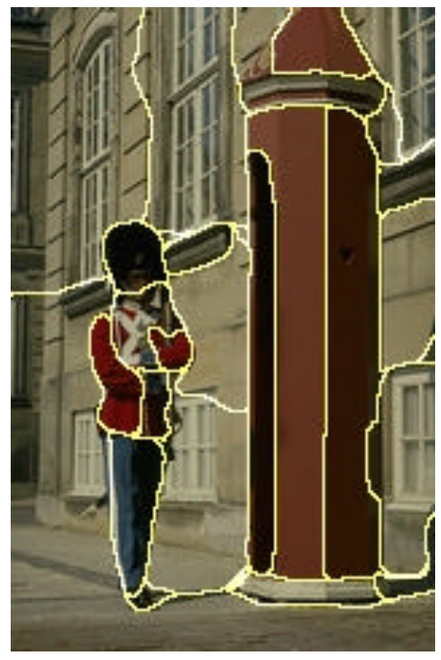










http://www.cs.berkeley.edu/~fowlkes/BSE/

Normalized cuts: pros and cons

Pros:

- Generic framework, flexible to choice of function that computes weights ("affinities") between nodes
- Does not require model of the data distribution

Cons:

- Time complexity can be high
  - Dense, highly connected graphs → many affinity computations
  - Solving eigenvalue problem
- Preference for balanced partitions

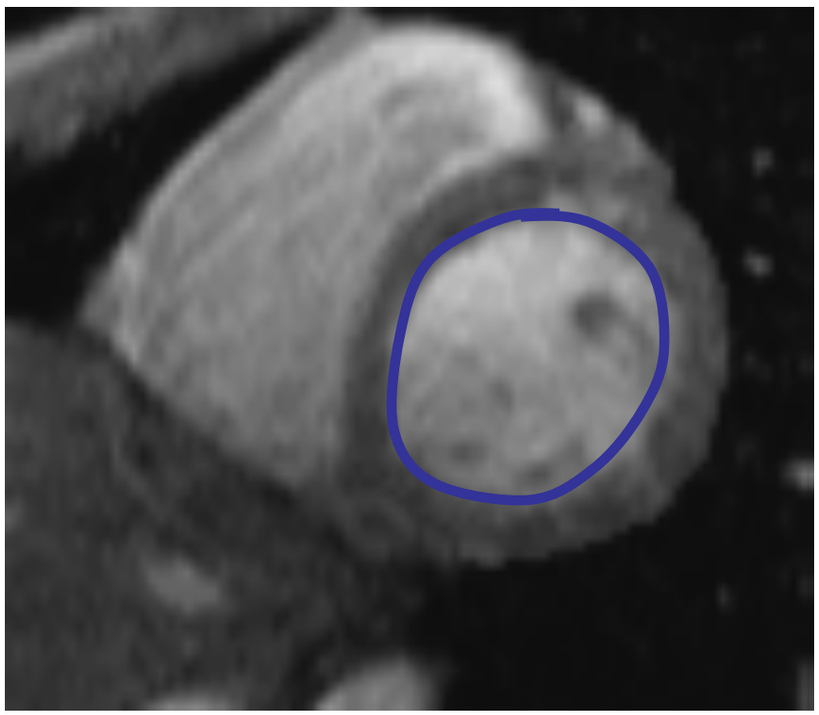
Summary

- Segmentation to find object boundaries or mid-level regions, tokens.
- Bottom-up segmentation via clustering
  - General choices -- features, affinity functions, and clustering algorithms
  - Grouping also useful for quantization, can create new feature summaries
    - Texton histograms for texture within local region
- Example clustering methods
  - K-means
  - Mean shift
  - Graph cut, normalized cuts

Deformable contours

Given: initial contour (model) near desired object

Goal: evolve the contour to fit exact object boundary



Main idea: elastic band is iteratively adjusted so as to

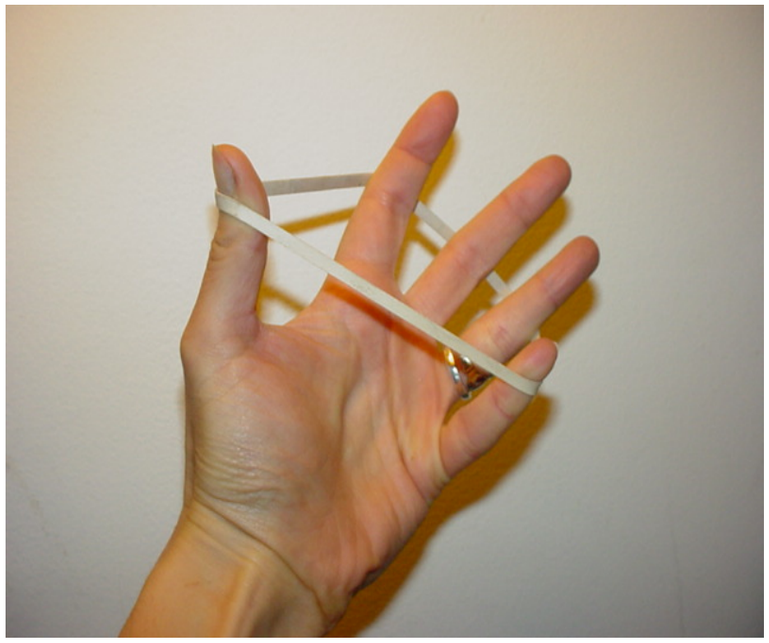
- be near image positions with high gradients, **and**
- satisfy shape "preferences" or contour priors

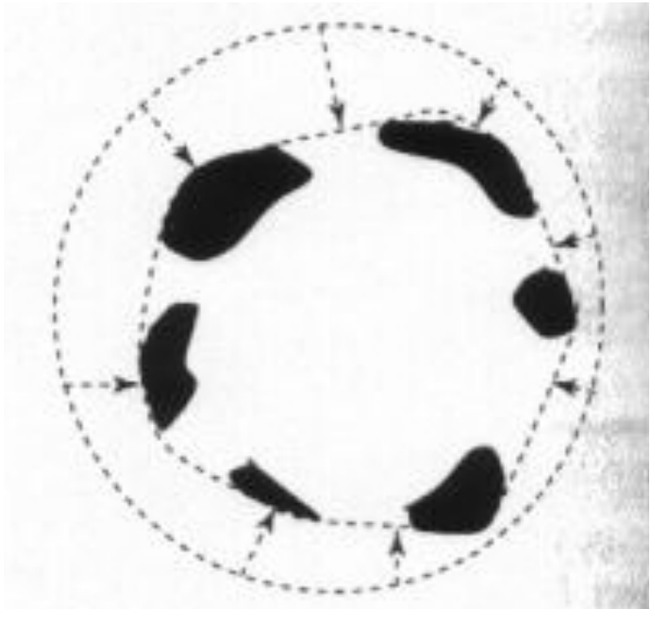
a.k.a. active contours, snakes

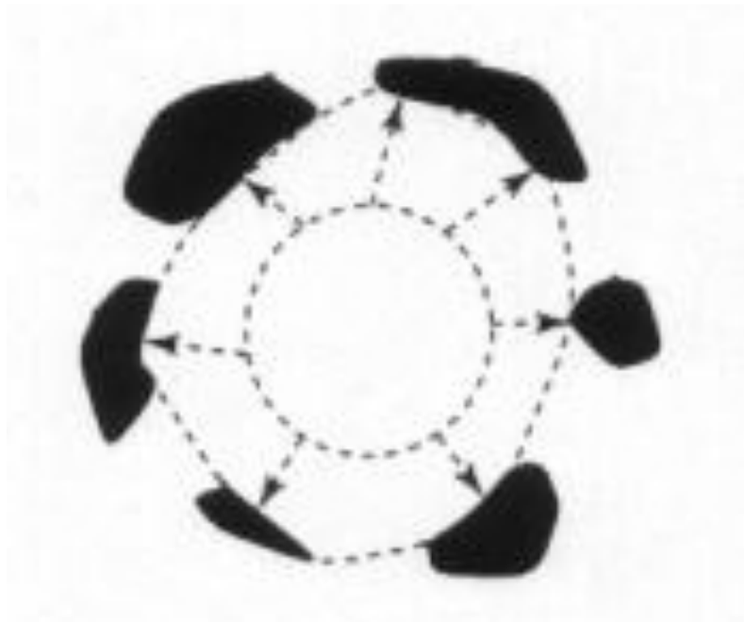
[Snakes: Active contour models, Kass, Witkin, & Terzopoulos, ICCV1987]

Figure credit: Yuri Boykov

Deformable contours: intuition



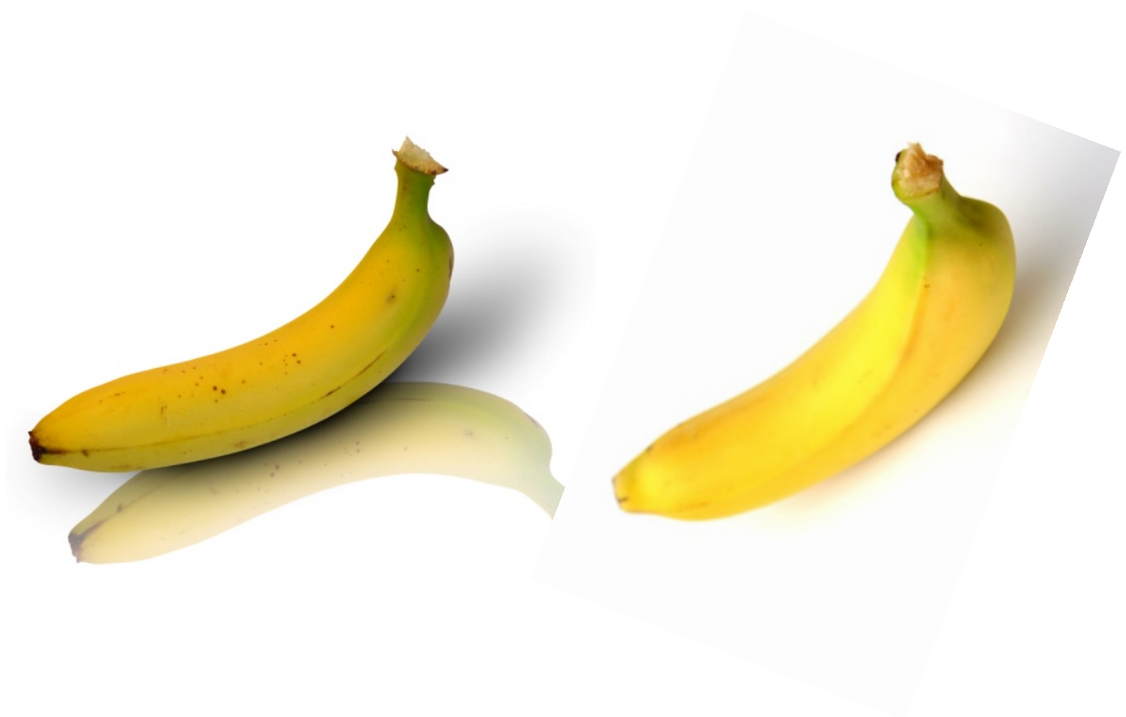


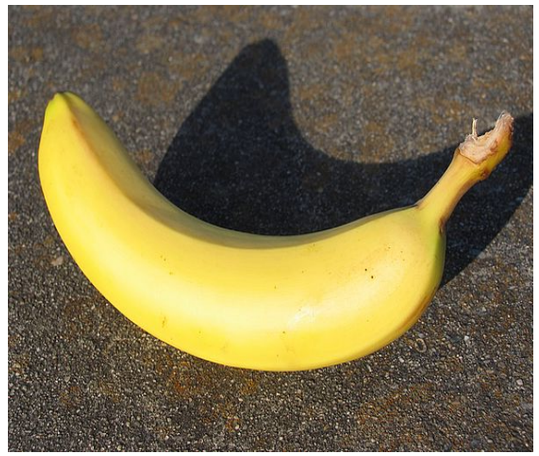


Slide credit: Kristen Grauman

Image from http://www.healthline.com/blogs/exercise_fitness/uploaded_images/HandBand2-795868.JPG

Why do we want to fit deformable shapes?





- Some objects have similar basic form but some variety in the contour shape.

Slide credit: Kristen Grauman

Why do we want to fit deformable shapes?

- Non-rigid, deformable objects can change their shape over time, e.g. lips, hands…

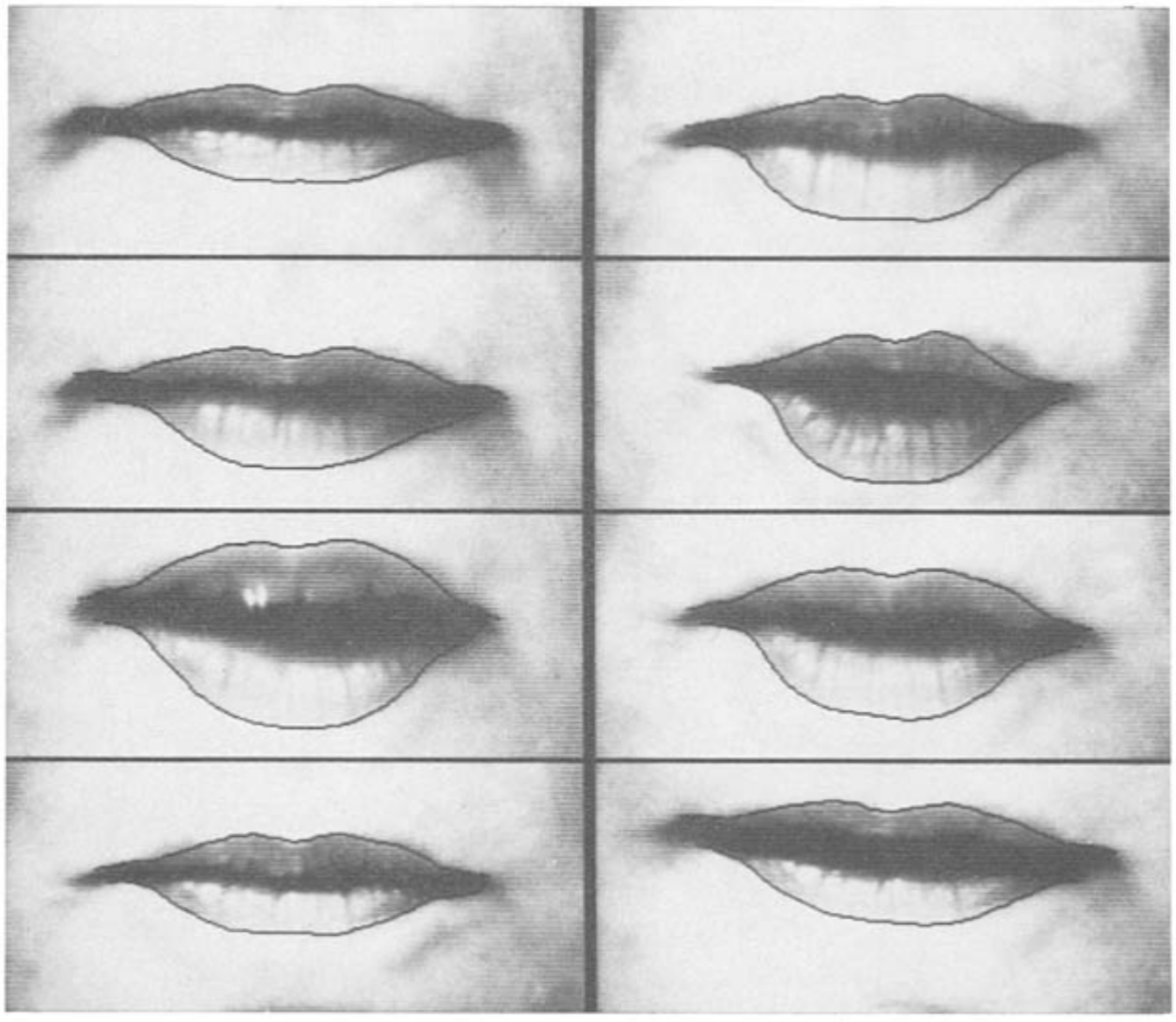
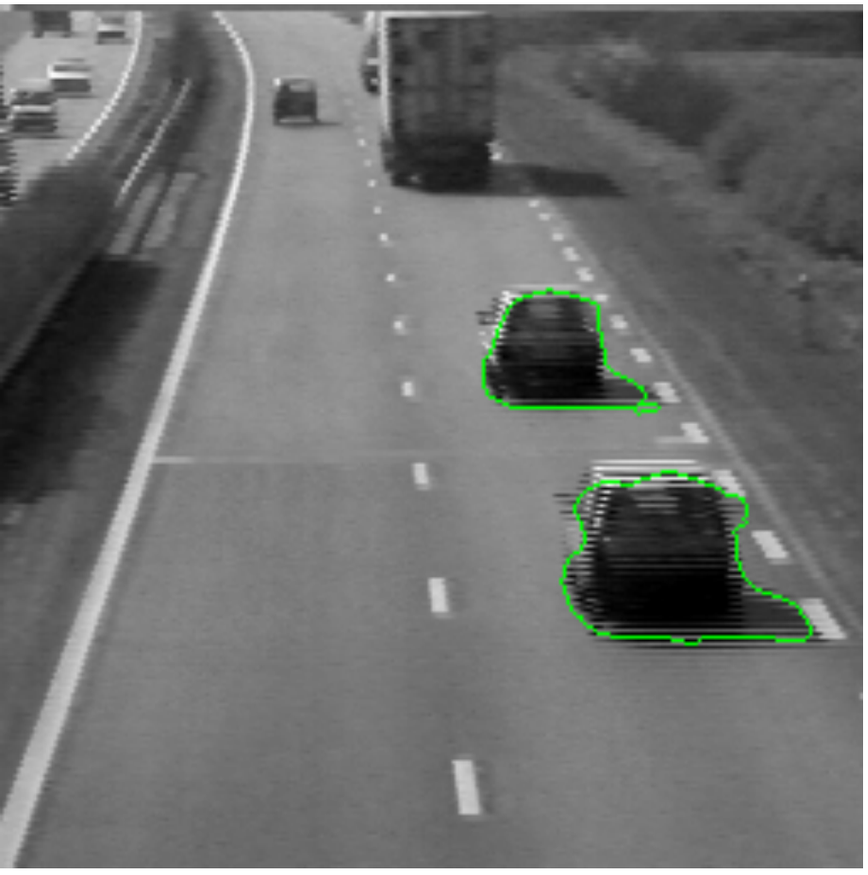
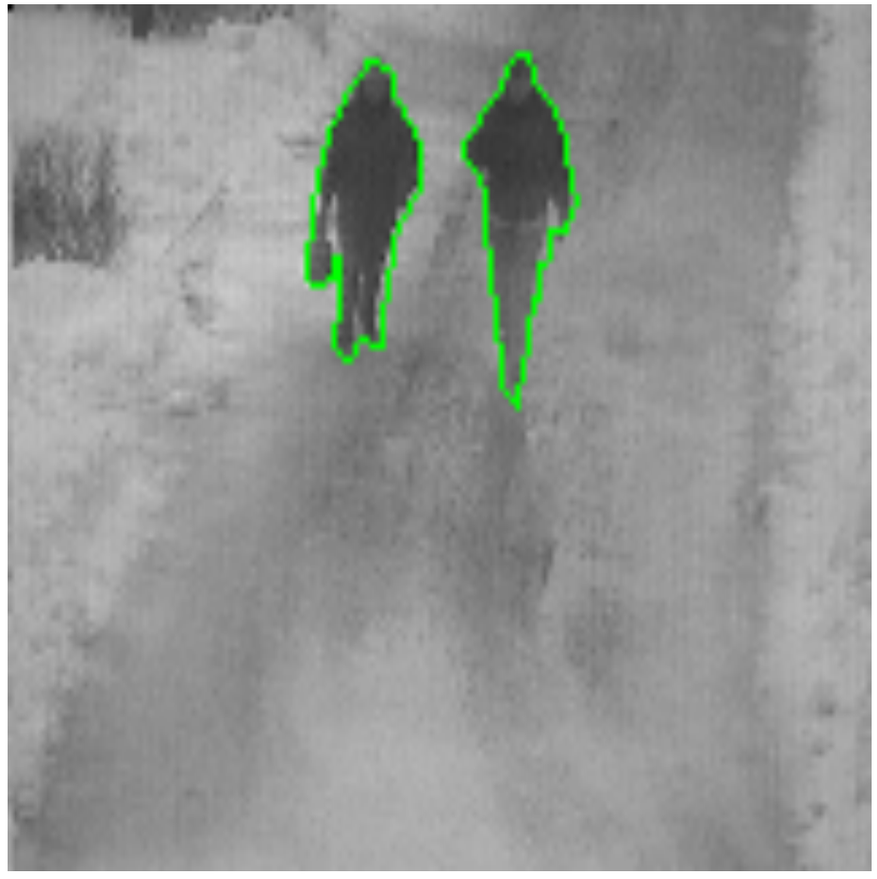


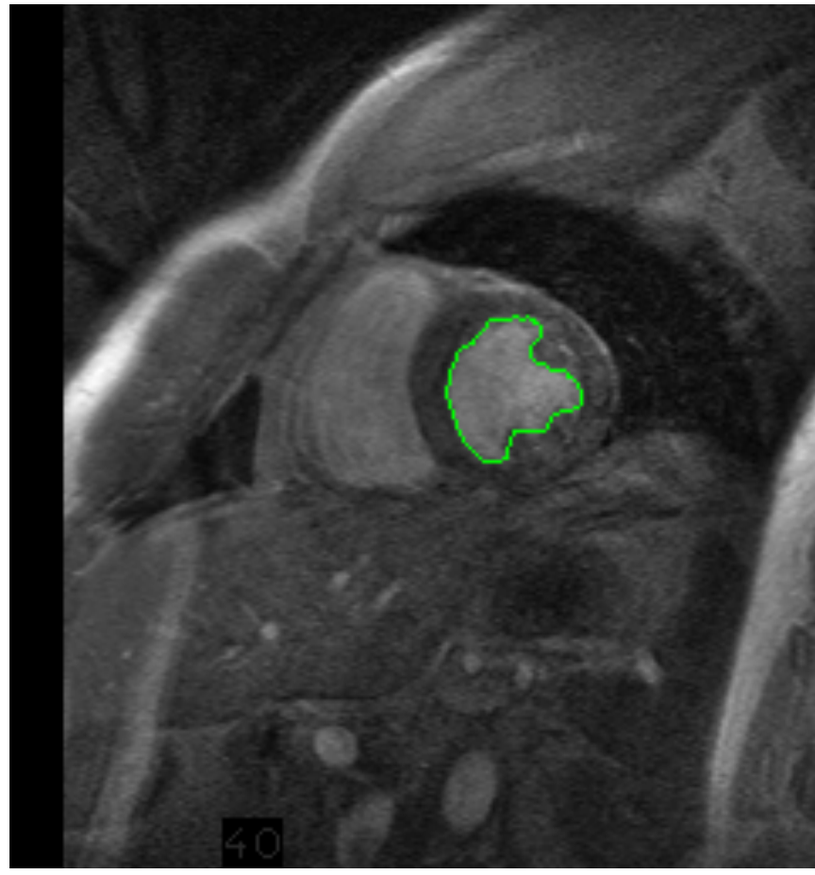
Figure from Kass et al. 1987

Slide credit: Kristen Grauman

Why do we want to fit deformable shapes?







- Non-rigid, deformable objects can change their shape over time.

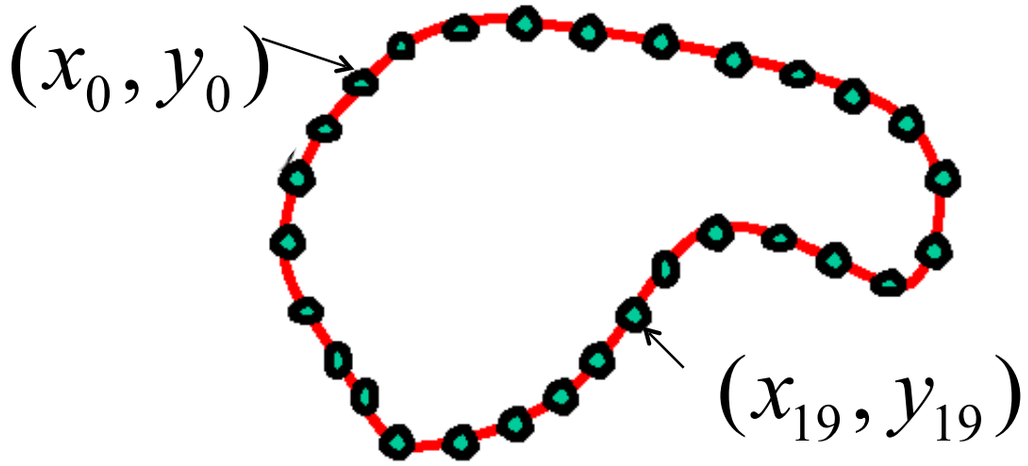
Figure credit: Julien Jomier

Aspects we need to consider

- Representation of the contours
- Defining the energy functions
  - External
  - Internal
- Minimizing the energy function
- Extensions:
  - Tracking
  - Interactive segmentation

Representation

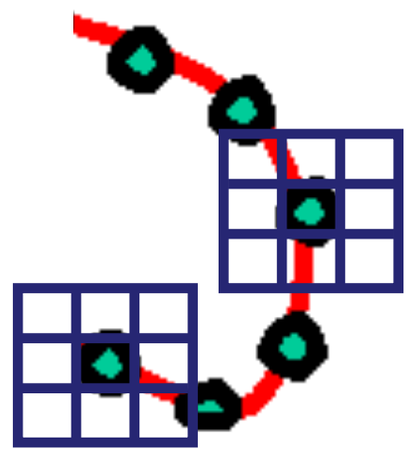
- We'll consider a discrete representation of the contour, consisting of a list of 2d point positions ("vertices").



$$\mathbf{v}_i = (x_i, y_i),$$

for $i = 0, 1, \ldots, n-1$

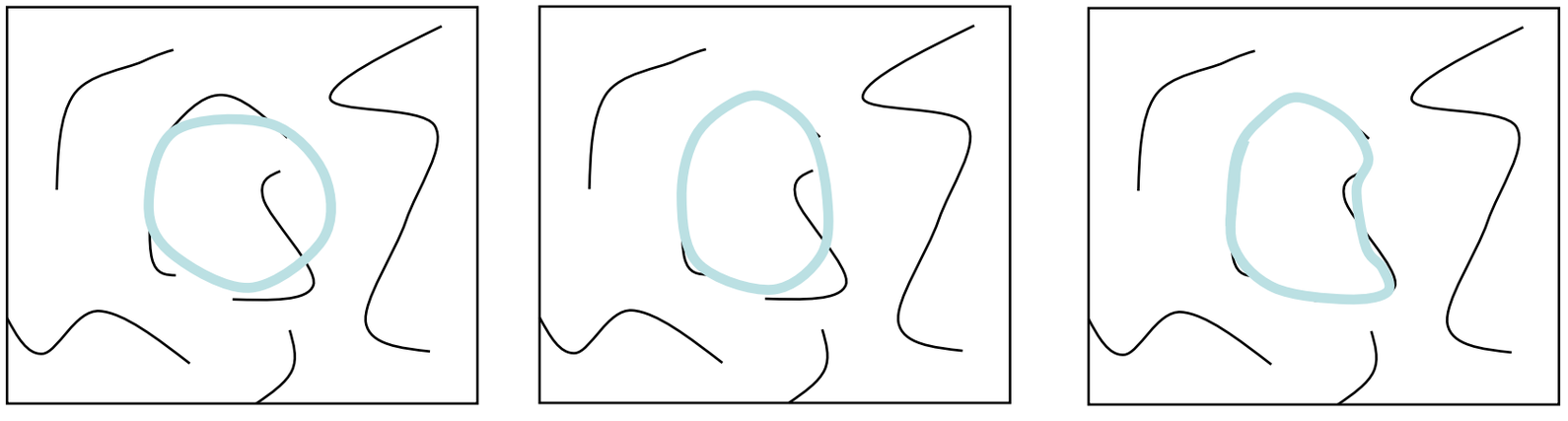
- At each iteration, we'll have the option to move each vertex to another nearby location ("state").



Fitting deformable contours

How should we adjust the current contour to form the new contour at each iteration?

- Define a cost function ("energy" function) that says how good a candidate configuration is.
- Seek next configuration that minimizes that cost function.



Energy function

The total energy (cost) of the current snake is defined as:

$$E_{total} = E_{internal} + E_{external}$$

Internal energy: encourage *prior* shape preferences: e.g., smoothness, elasticity, particular known shape.

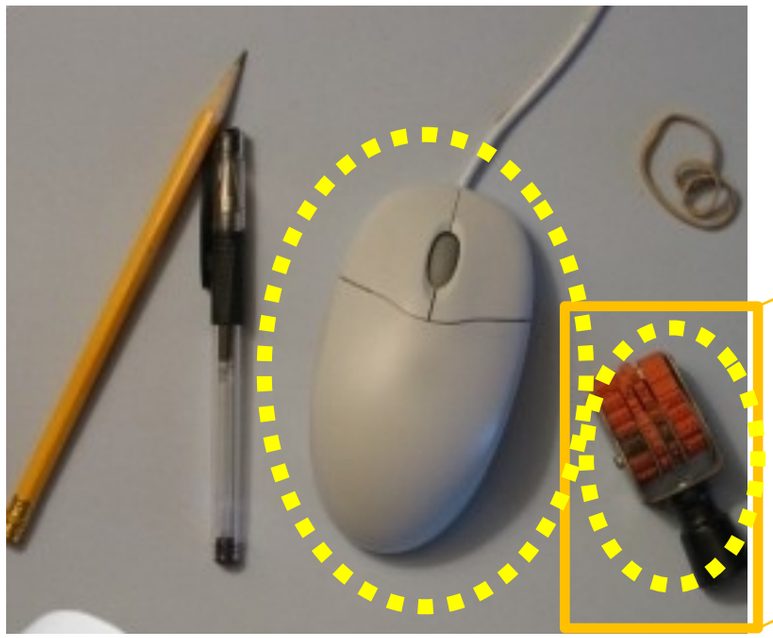
External energy ("image" energy): encourage contour to fit on places where image structures exist, e.g., edges.

A good fit between the current deformable contour and the target shape in the image will yield a **low** value for this cost function.

External energy: intuition

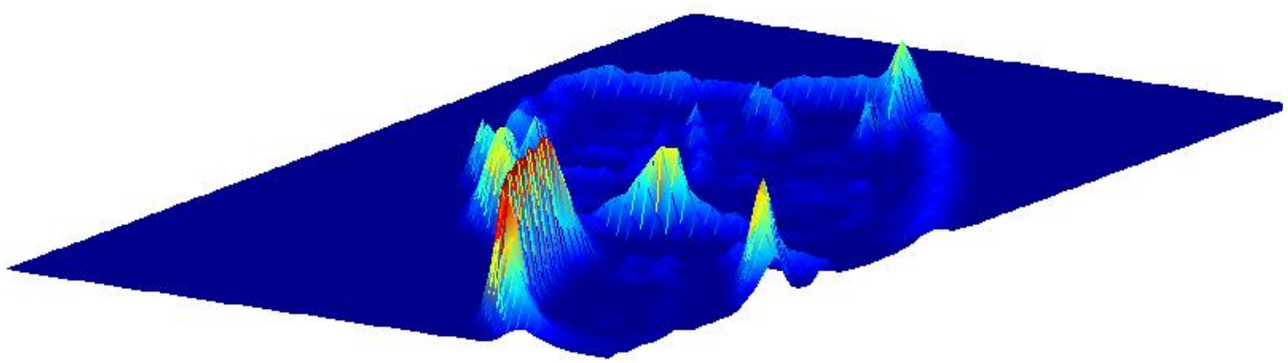
- Measure how well the curve matches the image data
- "Attract" the curve toward different image features
  - Edges, lines, texture gradient, etc.

External image energy



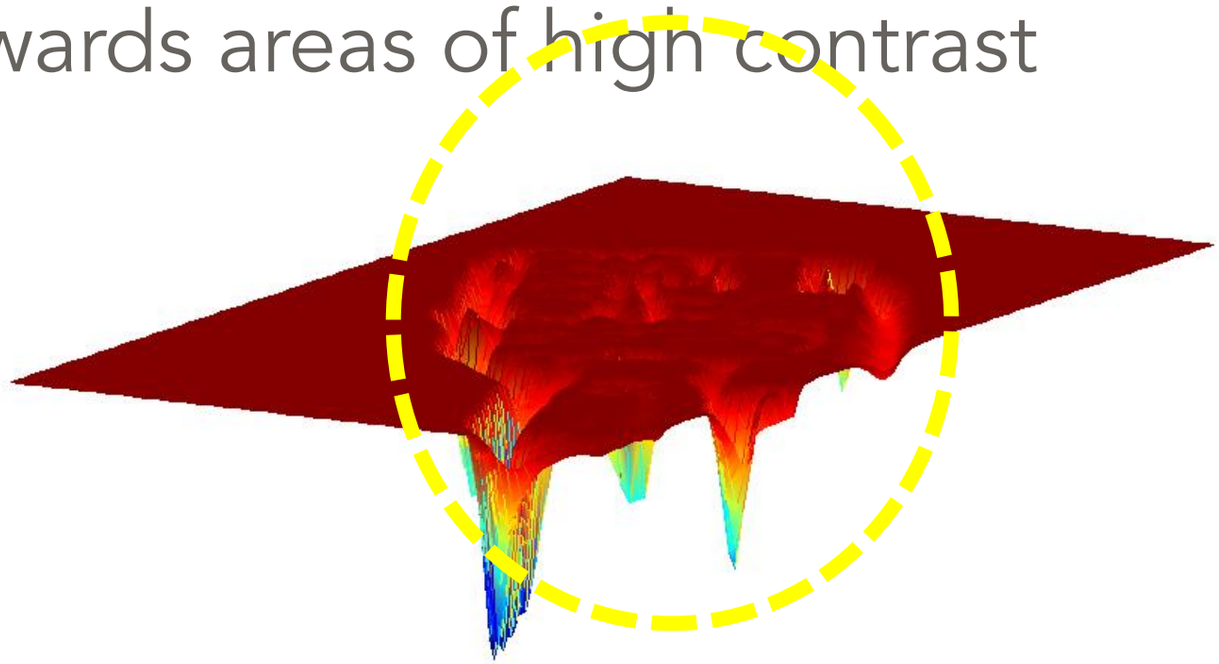
How do edges affect "snap" of rubber band?

Think of external energy from image as gravitational pull towards areas of high contrast



Magnitude of gradient

$$G_x(I)^2 + G_y(I)^2$$

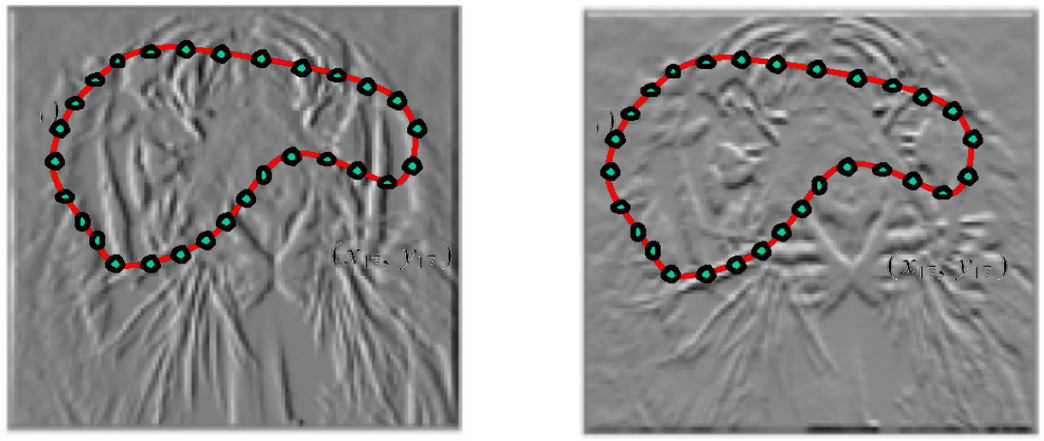


- (Magnitude of gradient)

$$-\left(G_x(I)^2 + G_y(I)^2\right)$$

External image energy

- Gradient images $G_x(x,y)$ and $G_y(x,y)$



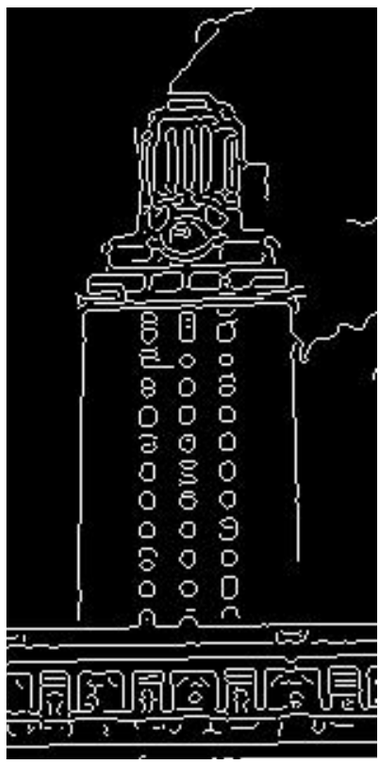
- External energy at a point on the curve is:

$$E_{external}(\mathbf{v}) = -(|G_x(\mathbf{v})|^2 + |G_y(\mathbf{v})|^2)$$

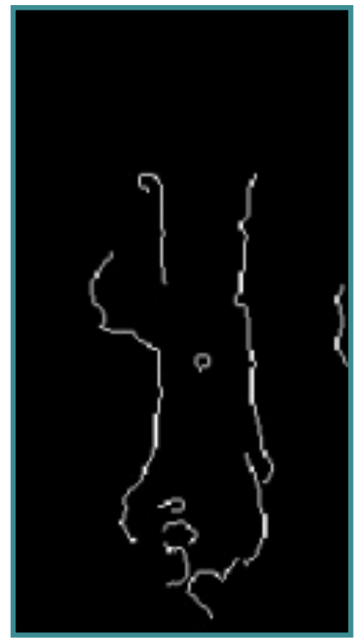
- External energy for the whole curve:

$$E_{external} = -\sum_{i=0}^{n-1} |G_x(x_i, y_i)|^2 + |G_y(x_i, y_i)|^2$$

Internal energy: intuition



What are the underlying boundaries in this fragmented edge image?



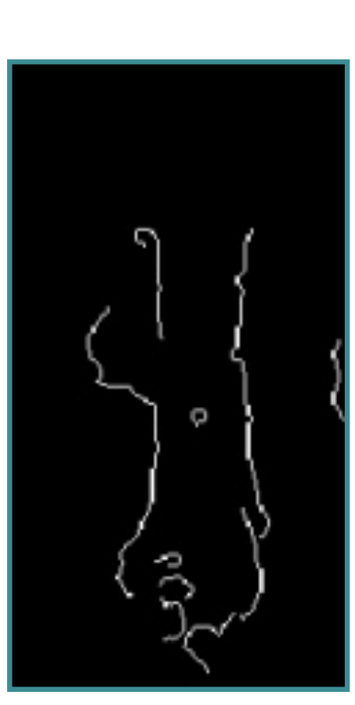
And in this one?

Slide credit: Kristen Grauman

Internal energy: intuition

A priori, we want to favor **smooth** shapes, contours with **low curvature**, contours similar to a **known shape**, etc. to balance what is actually observed (i.e., in the gradient image).





Slide credit: Kristen Grauman

Internal energy

For a *continuous* curve, a common internal energy term is the "bending energy".

At some point *v(s)* on the curve, this is:

$$E_{internal}(\mathbf{v}(s)) = \alpha \left| \frac{d\mathbf{v}}{ds} \right|^2 + \beta \left| \frac{d^2\mathbf{v}}{d^2 s} \right|^2$$

Tension, Elasticity

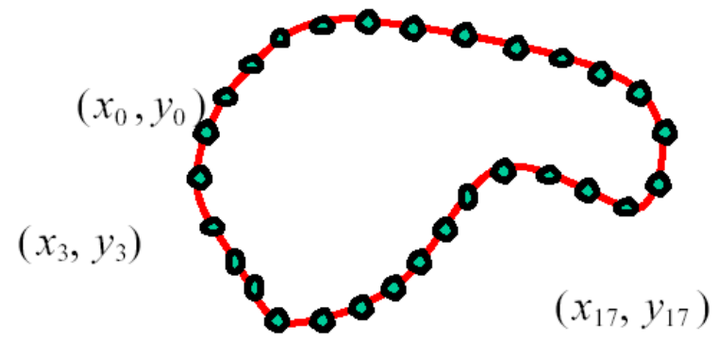
Stiffness, Curvature

Slide credit: Kristen Grauman

Internal energy

- For our discrete representation,

$$\mathbf{v}_i = (x_i, y_i) \qquad i = 0 \ldots n-1$$



$$\frac{d\mathbf{v}}{ds} \approx \mathbf{v}_{i+1} - \mathbf{v}_i \qquad \frac{d^2\mathbf{v}}{ds^2} \approx (\mathbf{v}_{i+1} - \mathbf{v}_i) - (\mathbf{v}_i - \mathbf{v}_{i-1}) = \mathbf{v}_{i+1} - 2\mathbf{v}_i + \mathbf{v}_{i-1}$$

*Note these are derivatives relative to **position**---not spatial image gradients.*

- Internal energy for the whole curve:

$$E_{internal} = \sum_{i=0}^{n-1} \alpha \left\| \mathbf{v}_{i+1} - \mathbf{v}_i \right\|^2 + \beta \left\| \mathbf{v}_{i+1} - 2\mathbf{v}_i + \mathbf{v}_{i-1} \right\|^2$$

*Why do these reflect **tension** and **curvature**?*

Example: compare curvature

$$E_{curvature}(v_i) = \|\mathbf{v}_{i+1} - 2\mathbf{v}_i + \mathbf{v}_{i-1}\|^2$$
$$= (x_{i+1} - 2x_i + x_{i-1})^2 + (y_{i+1} - 2y_i + y_{i-1})^2$$

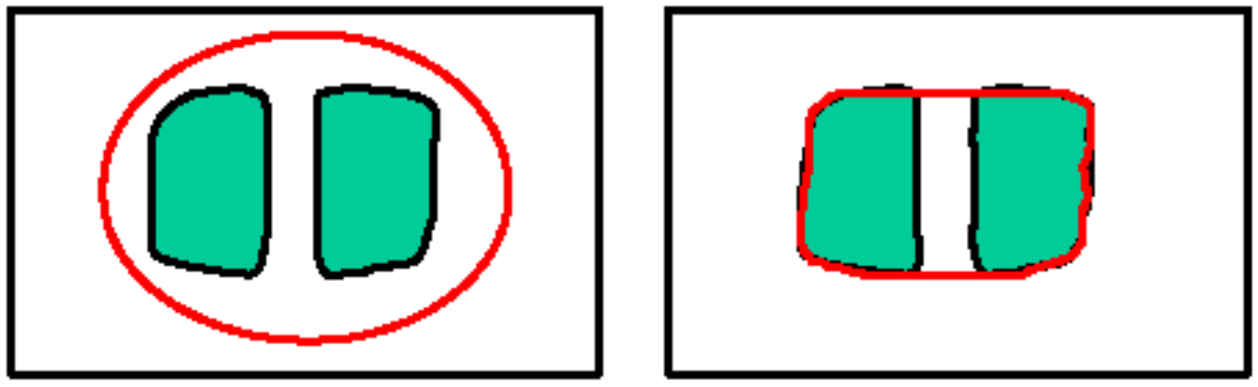
$$(3 - 2(2) + 1)2 + (1 - 2(5) + 1)2$$
$$= (-8)2 = 64$$

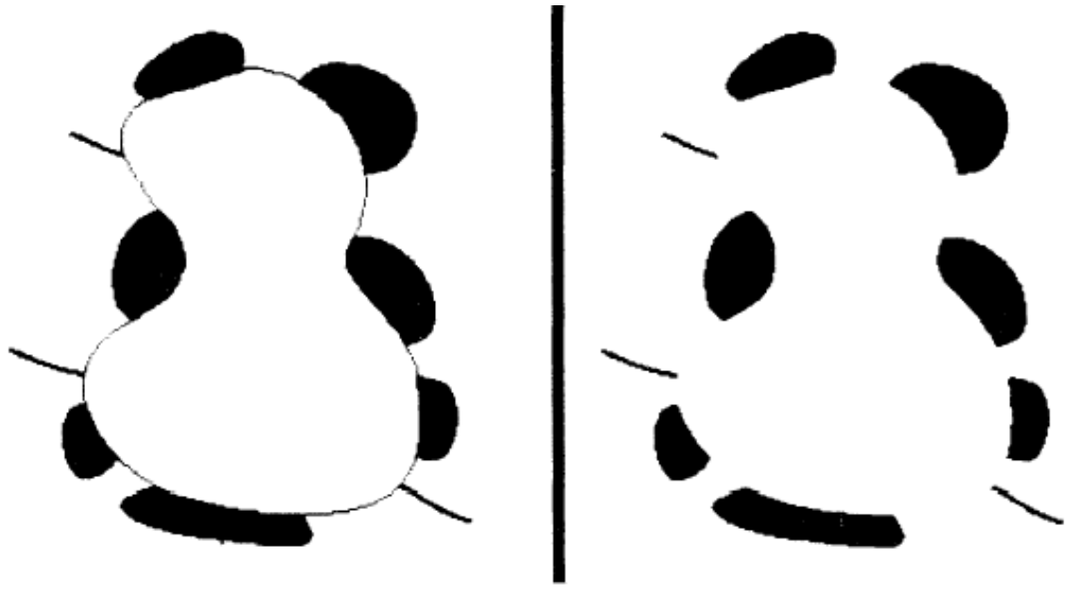
$$(3 - 2(2) + 1)2 + (1 - 2(2) + 1)2$$
$$= (-2)2 = 4$$

Slide credit: Kristen Grauman

Dealing with missing data

- The preferences for low-curvature, smoothness help deal with missing data:



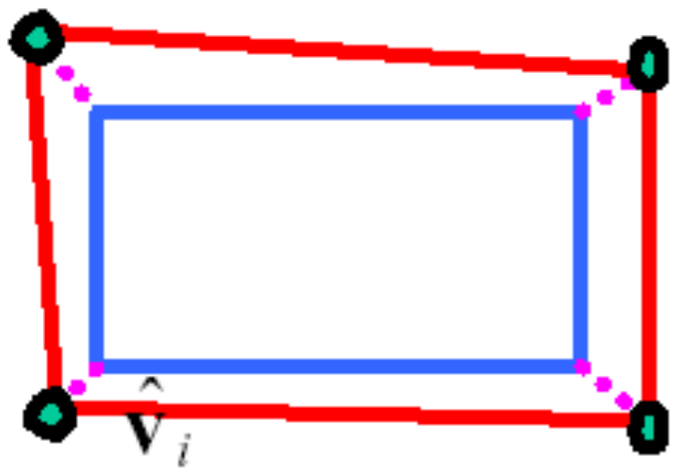


[Figure from Kass et al. 1987]

Illusory contours found!

Extending the internal energy: capture shape prior

- If object is some smooth variation on a known shape, we can use a term that will penalize deviation from that shape:



$$E_{internal} += \alpha \cdot \sum_{i=0}^{n-1} (\mathbf{v}_i - \hat{\mathbf{v}}_i)^2$$

where $\{\hat{\mathbf{v}}_i\}$ are the points of the known shape.

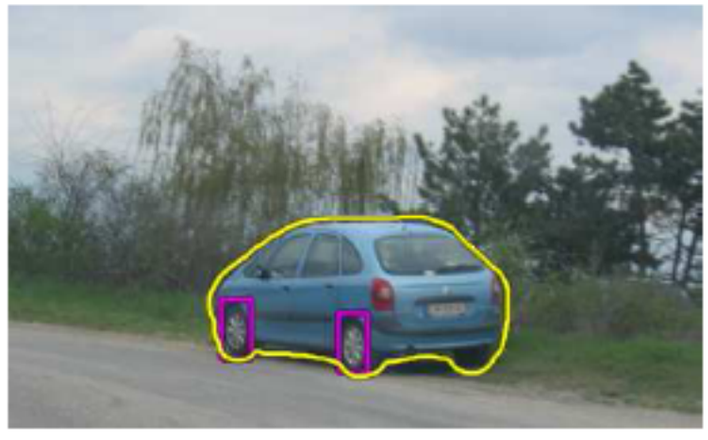


Fig from Y. Boykov

Total energy: function of the weights

$$E_{total} = E_{internal} + \gamma E_{external}$$

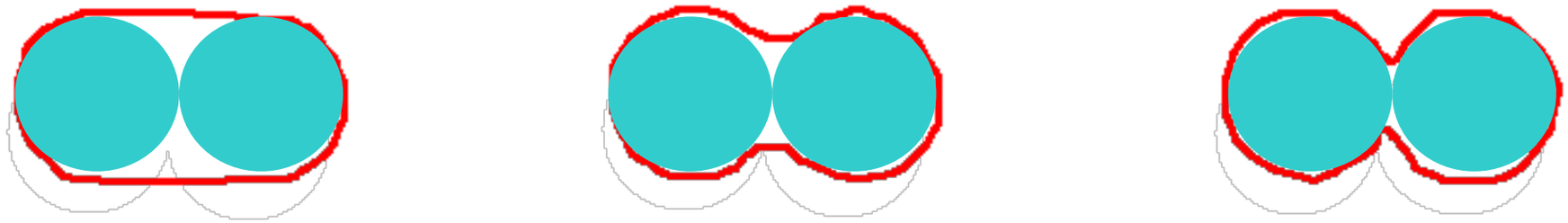
$$E_{external} = -\sum_{i=0}^{n-1} |G_x(x_i, y_i)|^2 + |G_y(x_i, y_i)|^2$$

$$E_{internal} = \sum_{i=0}^{n-1} \alpha \left(\bar{d} - \|\mathbf{v}_{i+1} - \mathbf{v}_i\|\right)^2 + \beta \|\mathbf{v}_{i+1} - 2\mathbf{v}_i + \mathbf{v}_{i-1}\|^2$$

Total energy: function of the weights

e.g., weight controls the penalty for internal elasticity

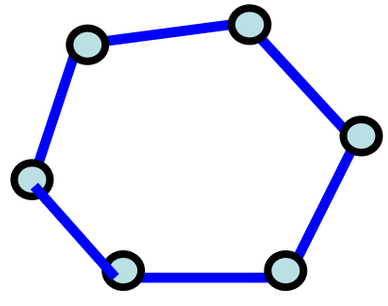
α



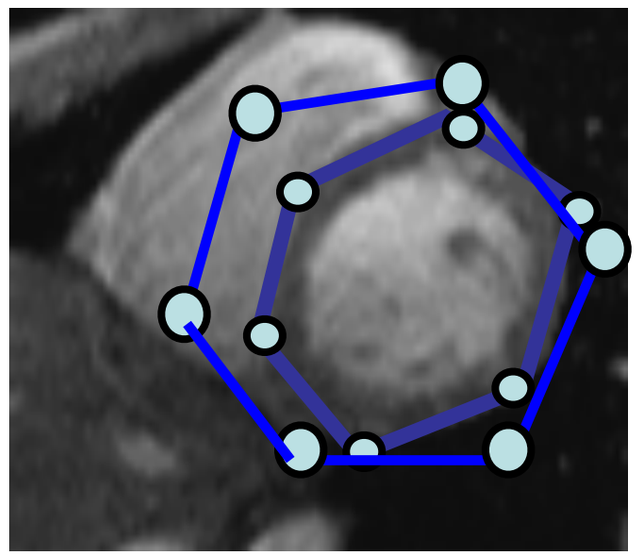
large α medium α small α

Fig from Y. Boykov

Recap: deformable contour



- A simple elastic snake is defined by:
  - A set of *n* points,
  - An internal energy term (tension, bending, plus optional shape prior)
  - An external energy term (gradient-based)



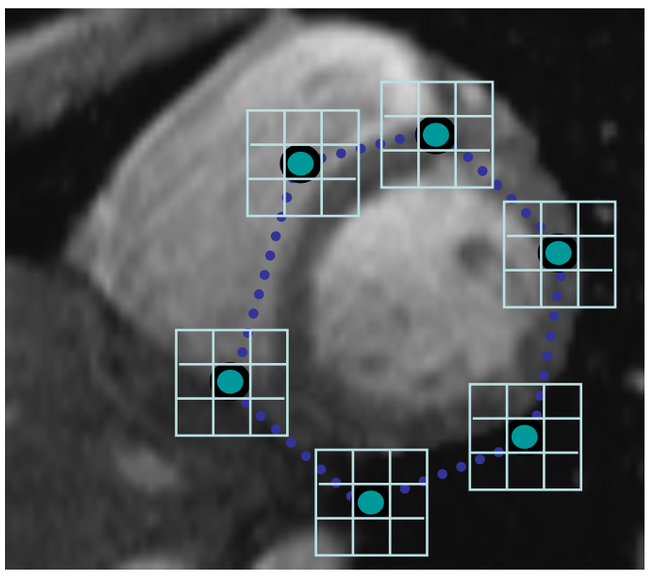
- To use to segment an object:
  - Initialize in the vicinity of the object
  - Modify the points to minimize the total energy

Energy minimization

- Several algorithms have been proposed to fit deformable contours.
- We'll look at two:
  - Greedy search
  - Dynamic programming (for 2d snakes)

Energy minimization: greedy

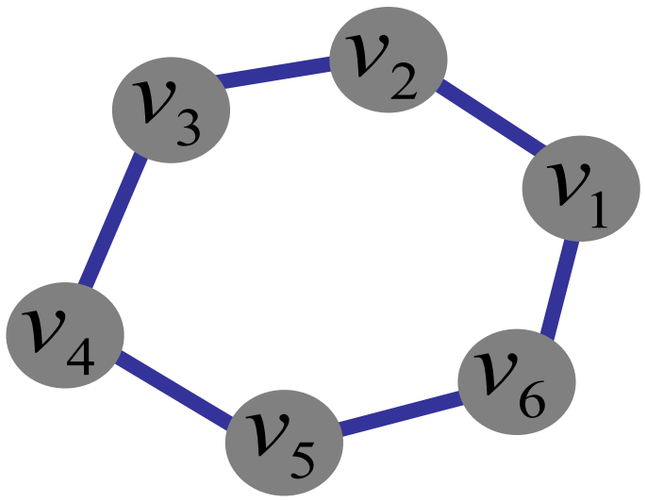
- For each point, search window around it and move to where energy function is minimal
  - Typical window size, e.g., 5 x 5 pixels
- Stop when predefined number of points have not changed in last iteration, or after max number of iterations
- Note:
  - Convergence not guaranteed
  - Need decent initialization

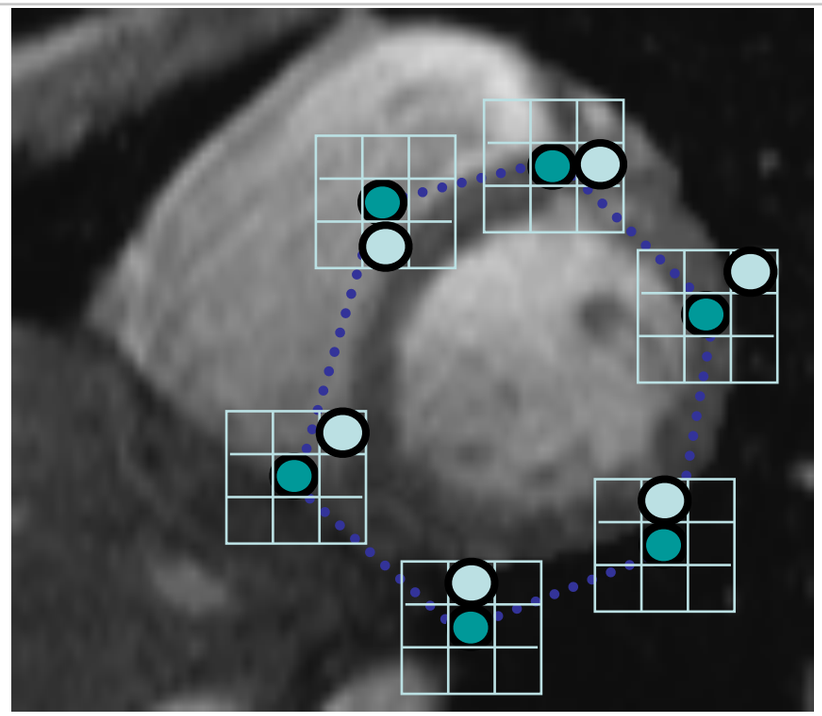


Energy minimization

- Several algorithms have been proposed to fit deformable contours.
- We'll look at two:
  - Greedy search
  - Dynamic programming (for 2d snakes)

Energy minimization: dynamic programming





- With this form of the energy function, we can minimize using dynamic programming, with the *Viterbi* algorithm.
- Iterate until optimal position* for each point is the center of the box, i.e., the snake is optimal in the local search space constrained by boxes.
- *optimal within resolution of considered window

Fig from Y. Boykov
[Amini, Weymouth, Jain, 1990]

Deformable contours: pros and cons

Pros:

- Useful to track and fit non-rigid shapes
- Contour remains connected
- Possible to fill in “subjective” contours
- Flexibility in how energy function is defined, weighted.

Cons:

- Must have decent initialization near true boundary, may get stuck in local minimum
- Parameters of energy function must be set well based on prior information

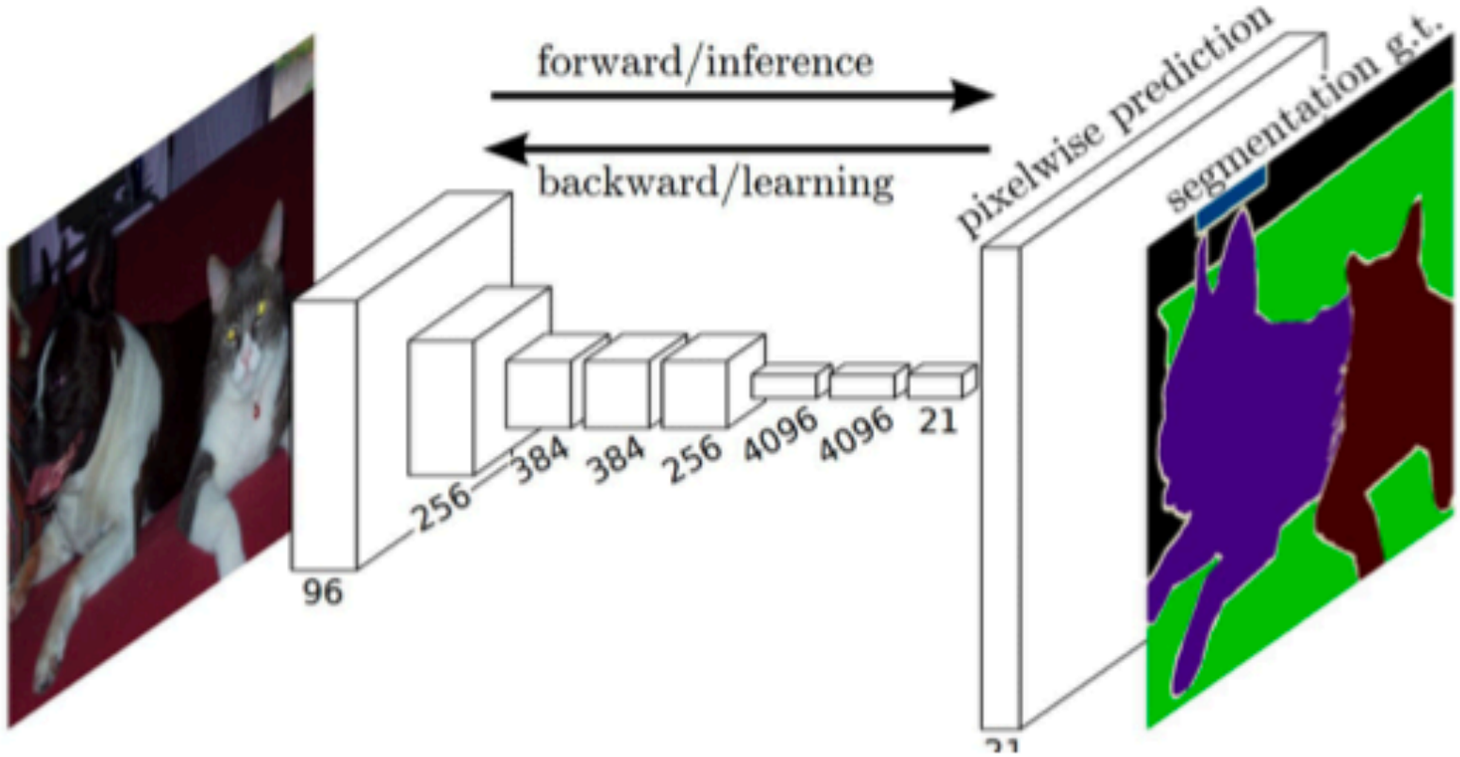
Slide credit: Kristen Grauman

DL-Based Segmentation

- **Methods**
- **Datasets**
- **Evaluation Metrics**
- **Performance**

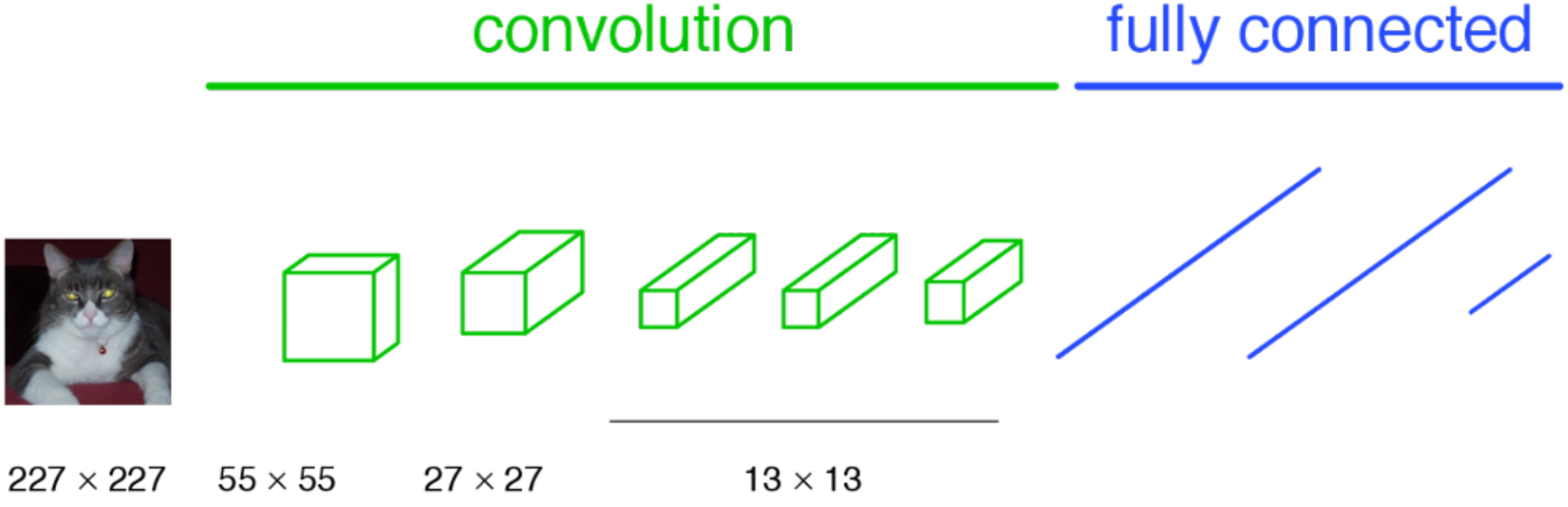
S. MINAEE *et al.*: Image Segmentation Using Deep Learning: A Survey. TPAMI, 2021

1. Fully Convolutional Models

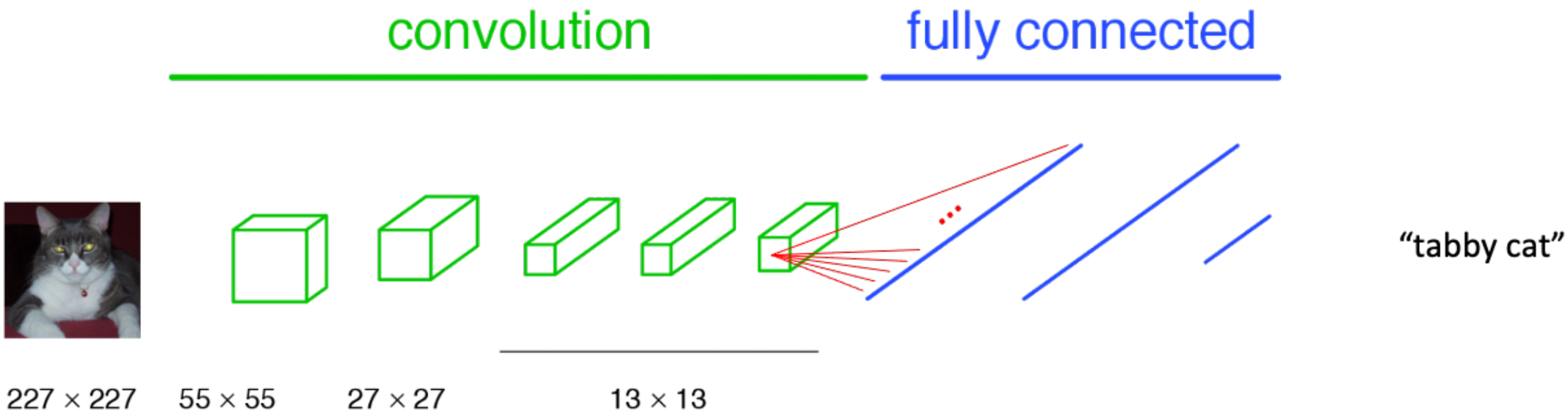


J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3431–3440.

A classification network…



A classification network…



The response of every kernel across all positions are attached densely to the array of perceptrons in the fully-connected layer.

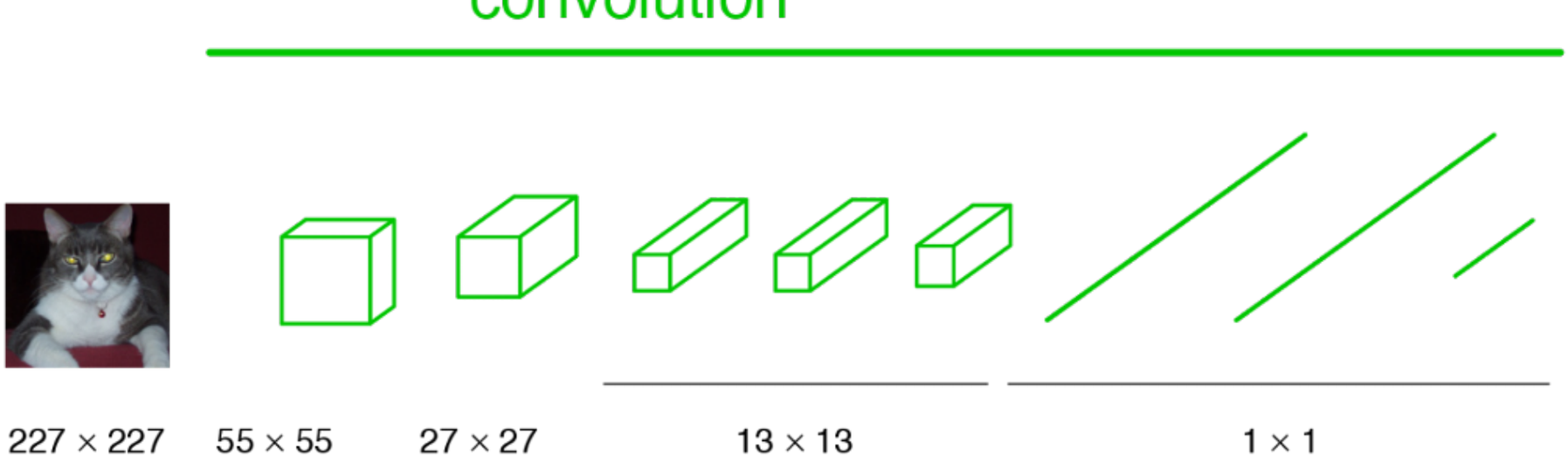
AlexNet: 256 filters over 6x6 response map
Each 2,359,296 response is attached to one of 4096 perceptrons, leading to 37 mil params.

Fully Convolutional Models

- **Problem**
  - **We want a label at every pixel**
  - **Current network gives us a label for the whole image.**
- **Approach**
  - **'Convolutionalize' *all layers* of network, so that we can treat it as one (complex) filter and slide around our full image.**

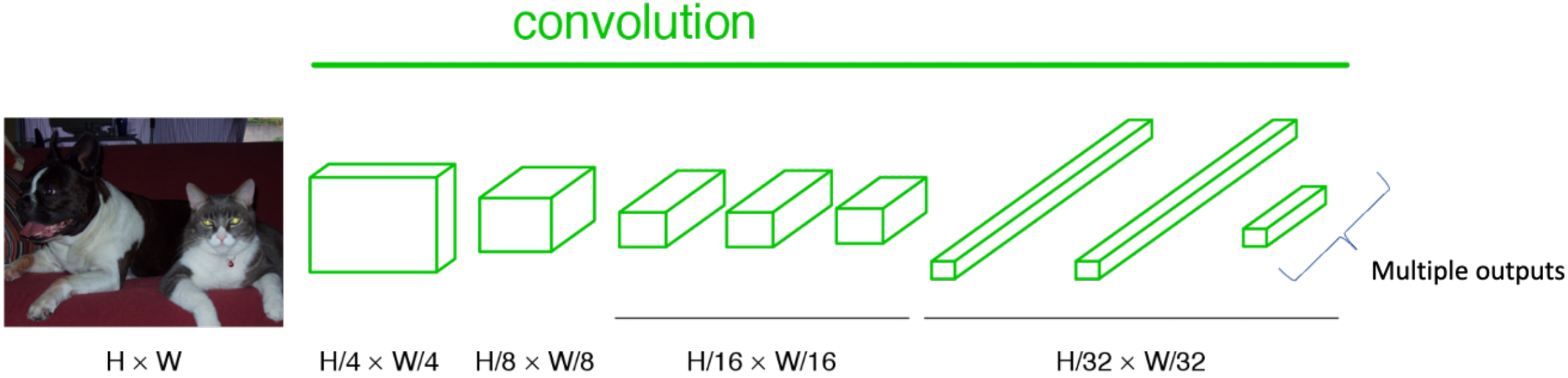
J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3431–3440.

Convolutionalization



1x1 convolution operates across all filters in the previous layer, and is slid across all positions.

Becoming fully convolutional



Arbitrary-sized image

When we turn these operations into a convolution, the 13x13 just becomes another parameter and our output size adjust dynamically.

Now we have a *vector/matrix* output, and our network acts itself like a complex filter.

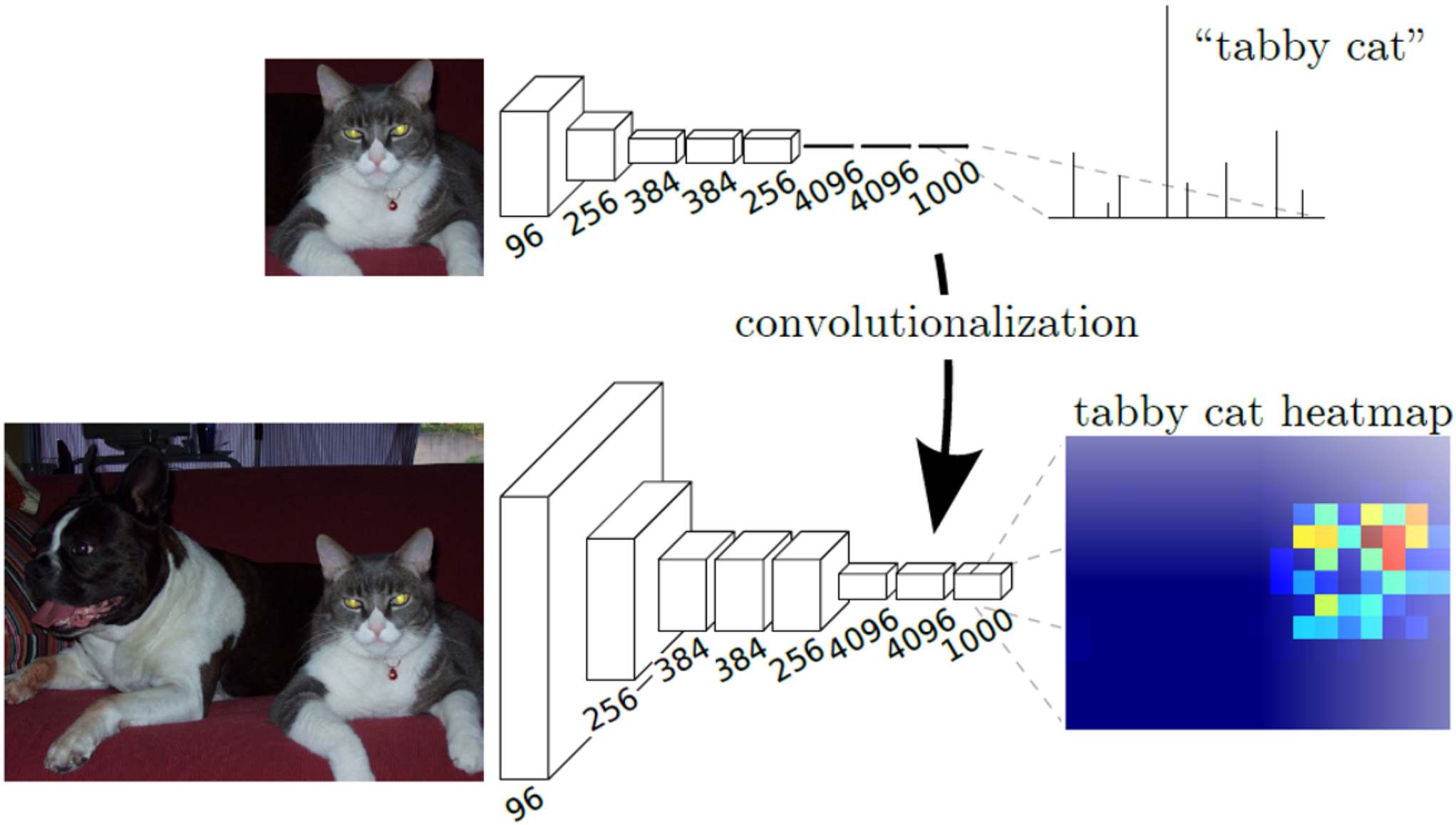
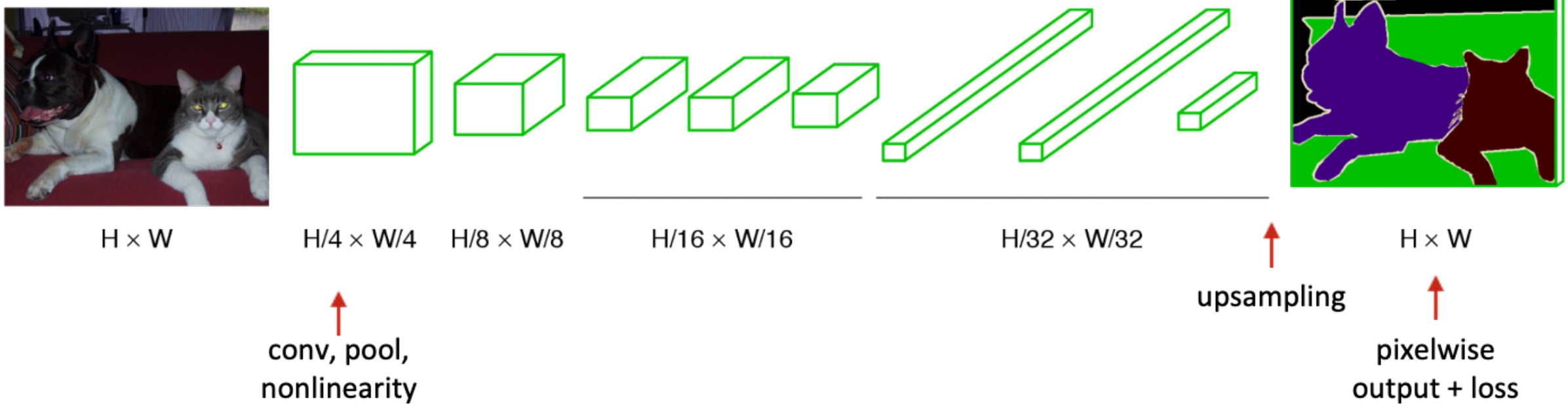
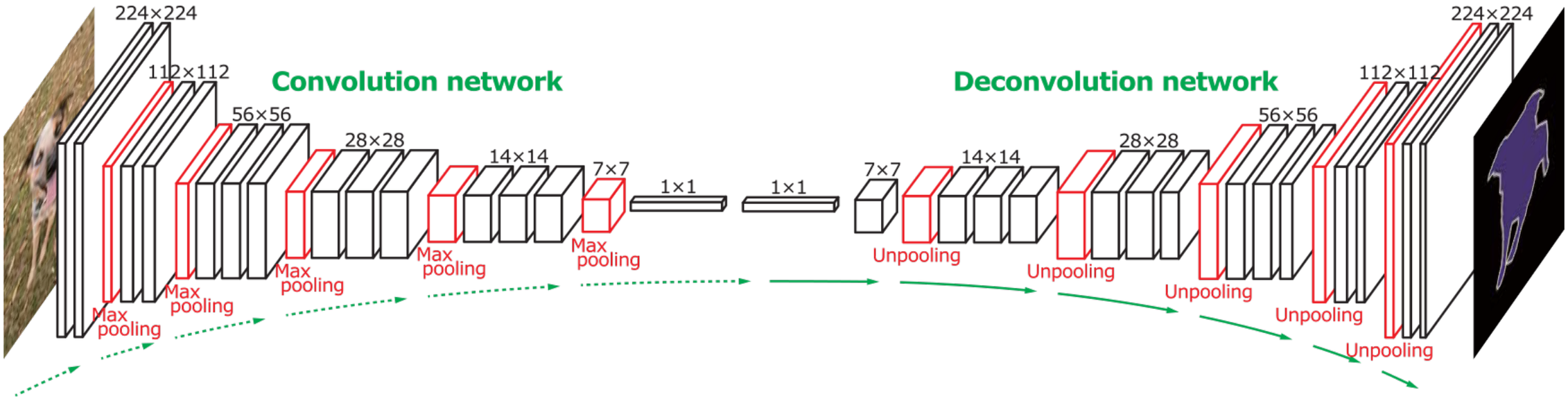


Figure 2. Transforming fully connected layers into convolution layers enables a classification net to output a heatmap. Adding layers and a spatial loss (as in Figure 1) produces an efficient machine for end-to-end dense learning.

End-to-end, pixels-to-pixels network



‘Deconvolution’ networks *learn to upsample*



Often called “deconvolution”, but misnomer.
Not the deconvolution that we saw in deblurring -> that is division in the Fourier domain.

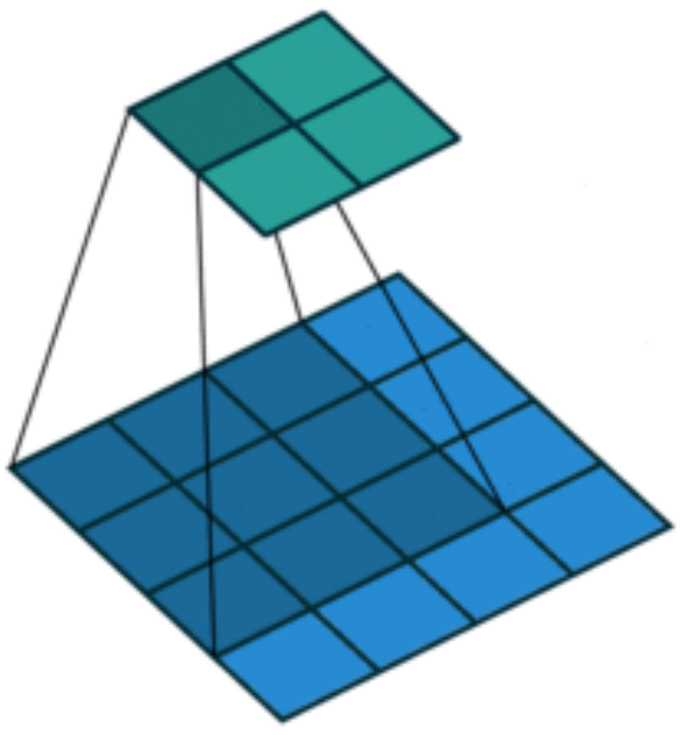
‘Transposed convolution’ is better.

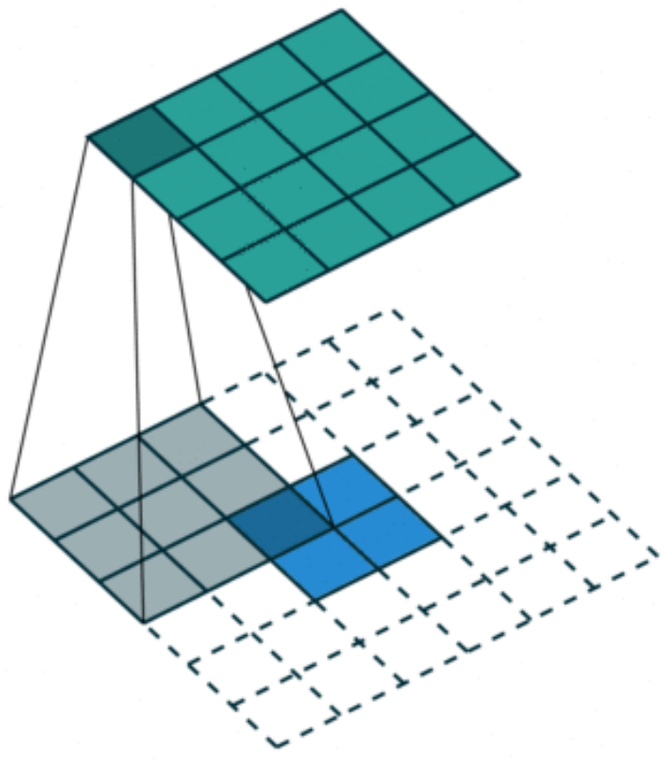
Zeiler et al., Deconvolutional Networks, CVPR 2010
Noh et al., Learning Deconvolution Network for Semantic Segmentation, ICCV 2015

Upsampling with transposed convolution

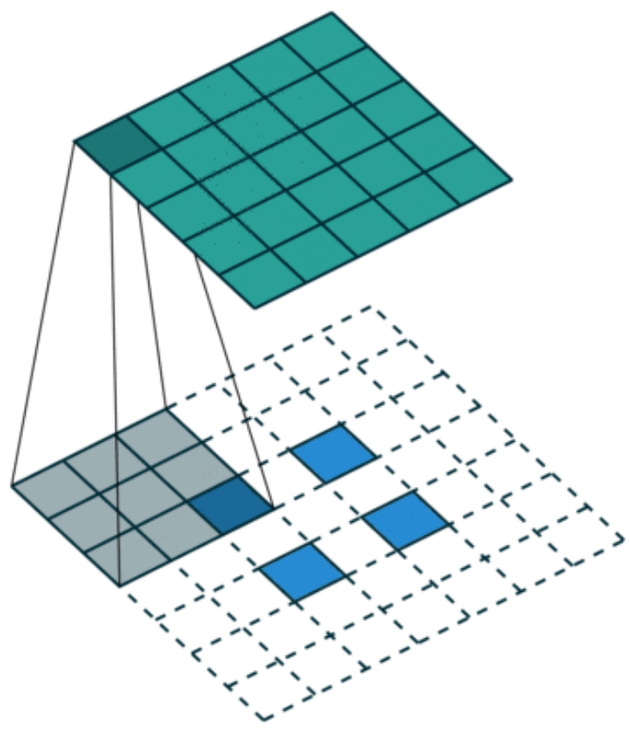
Transposed convolution = padding/striding smaller image then weighted sum of input x filter: 'stamping' kernel

Convolution





2x2, stride 1, 3x3 kernel, upsample to 4x4



2x2, stride 2, 3x3 kernel, upsample to 5x5.

Kernel

1	1	1
1	1	1
1	1	1

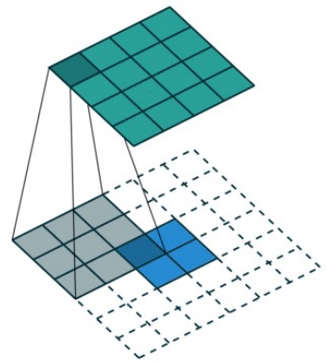
Feature map

1	2
3	4

Padded feature map

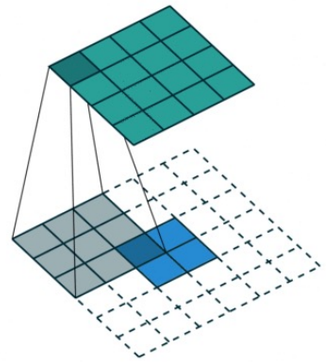
		1	2		
		3	4		

Inspired by andriys



Kernel

1	1	1
1	1	1
1	1	1



Input feature map

1	2
3	4

Output feature map

1	1	1			
1	1	1			
1	1	1			

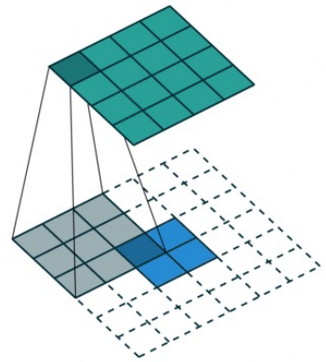
Padded input feature map

		1	2		
		3	4		

Inspired by andriys

Kernel

1	1	1
1	1	1
1	1	1



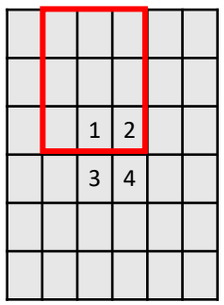
Input feature map

1	2
3	4

Output feature map

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

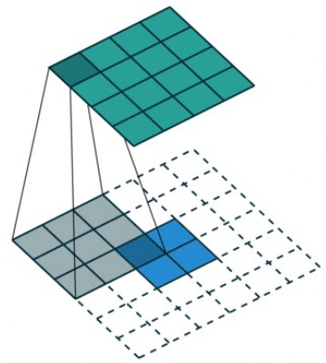
Padded input feature map



Inspired by andriys

Kernel

1	1	1
1	1	1
1	1	1



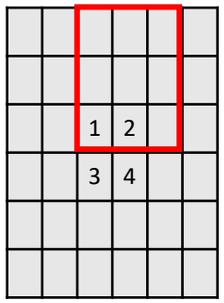
Input feature map

1	2
3	4

Output feature map

1	4	7	6	3	
1	4	7	6	3	
1	4	7	6	3	

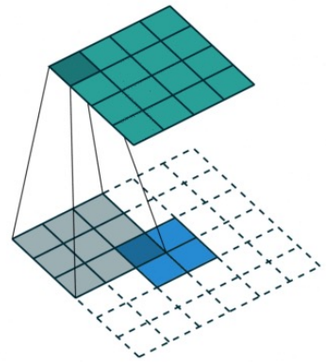
Padded input feature map



Inspired by andriys

Kernel

1	1	1
1	1	1
1	1	1



Input feature map

1	2
3	4

Output feature map

1	4	7	8	5	2
1	4	7	8	5	2
1	4	7	8	5	2

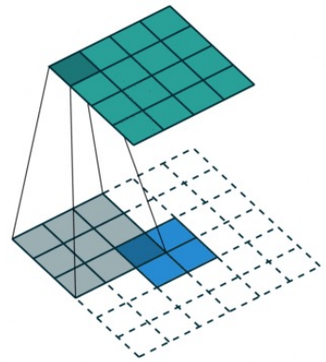
Padded input feature map

		1	2		
		3	4		

Inspired by andriys

Kernel

1	1	1
1	1	1
1	1	1



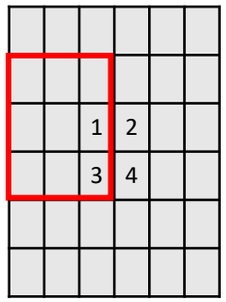
Input feature map

1	2
3	4

Output feature map

1	4	7	8	5	2
5	8	11	8	5	2
5	8	11	8	5	2
4	4	4			

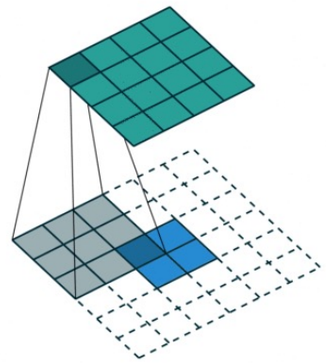
Padded input feature map



Inspired by andriys

Kernel

1	1	1
1	1	1
1	1	1



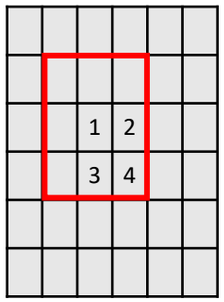
Input feature map

1	2
3	4

Output feature map

1	4	7	8	5	2
5	18	21	18	5	2
5	18	21	18	5	2
4	14	14	10		

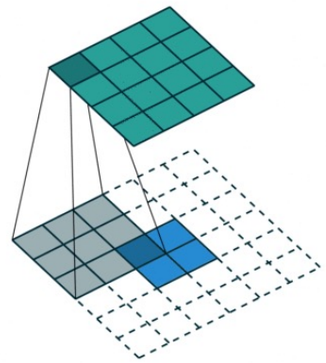
Padded input feature map



Inspired by andriys

Kernel

1	1	1
1	1	1
1	1	1



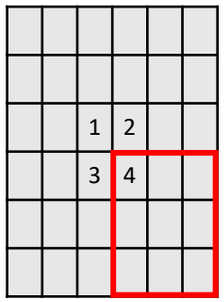
Input feature map

1	2
3	4

Output feature map

1	4	7	8	5	2
5	18	31	34	21	8
9	32	55	60	37	14
11	38	66	64	43	16
7	24	41	44	27	10
3	10	17	18	11	4

Padded input feature map



Inspired by andriys

Kernel

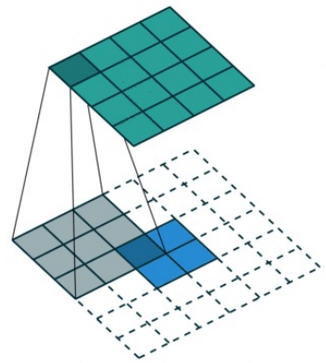
1	1	1
1	1	1
1	1	1

Input feature map

1	2
3	4

Padded input feature map

		1	2		
		3	4		

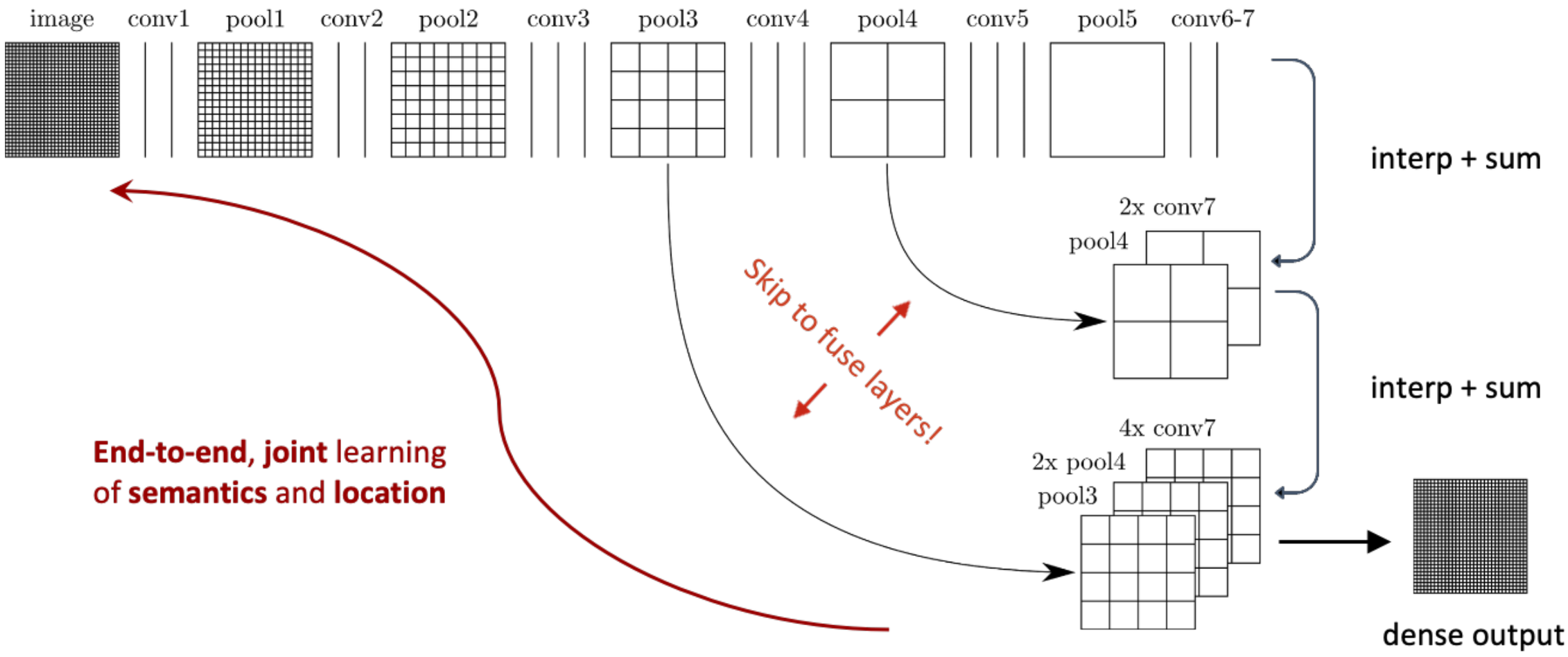


Cropped output feature map

18	31	34	21
32	55	60	37
38	66	64	43
24	41	44	27

Inspired by andriys

Fully Convolutional Models



J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3431–3440.

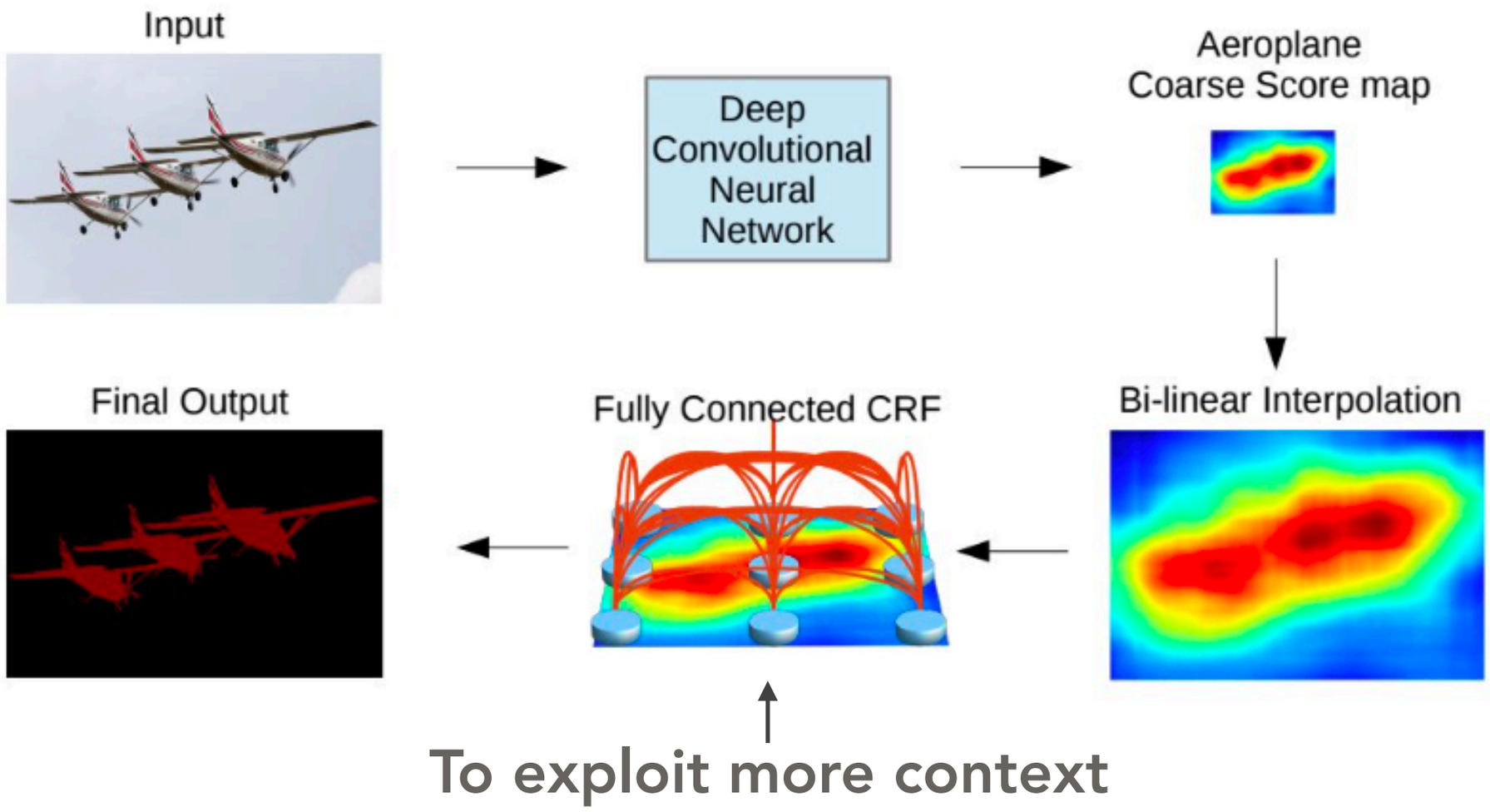
Fully Convolutional Models

- **Pro**
  - **End-to-end manner on variable-sized images**
- **Con**
  - **It is too computationally expensive for real-time inference, it does not account for global context information in an efficient manner, and it is not easily generalizable to 3D images.**

J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3431–3440.

2. CNNs With Graphical Models

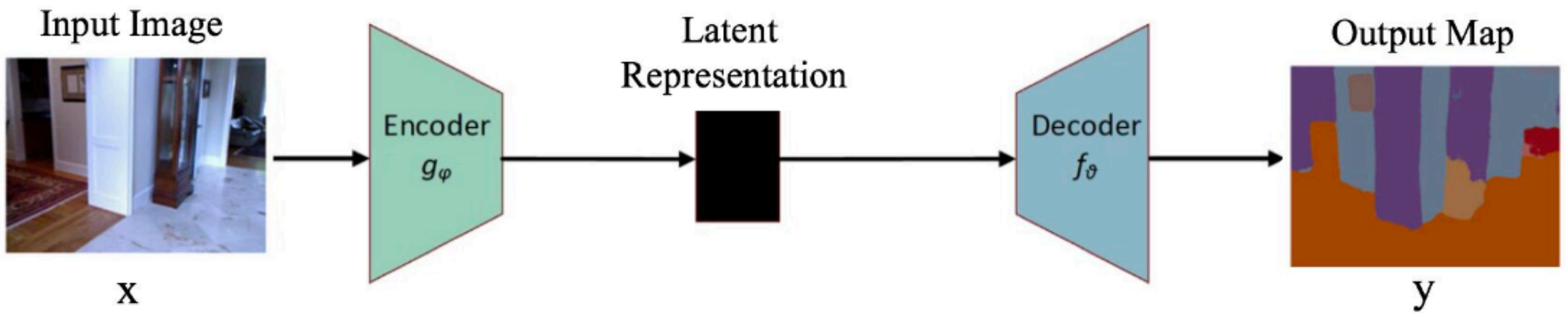
Incorporate into DL architectures probabilistic graphical models, such as Conditional Random Fields (CRFs) and Markov Random Fields (MRFs).



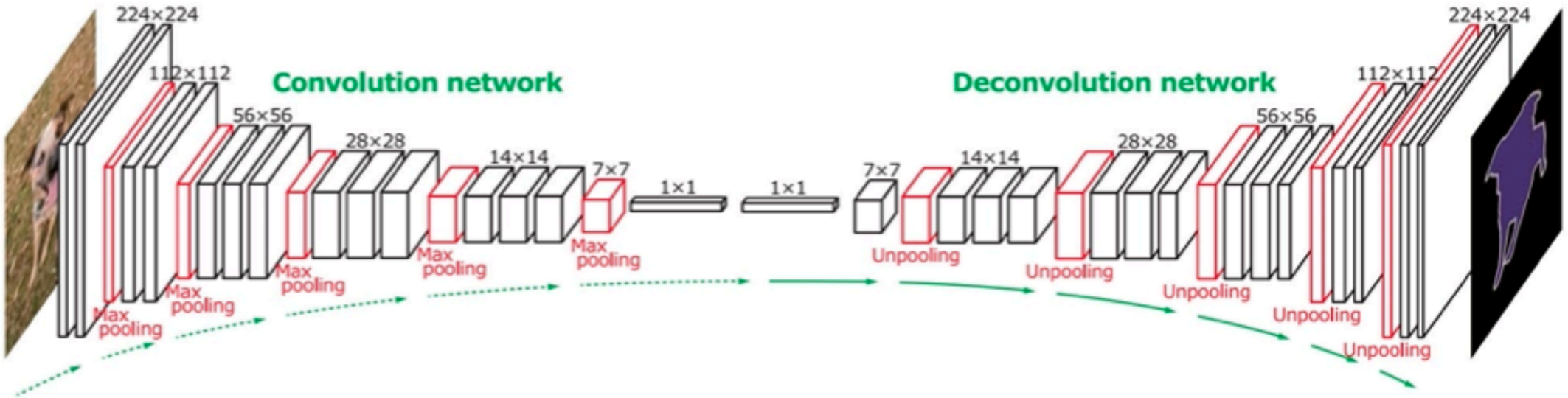
A. G. Schwing and R. Urtasun, "Fully connected deep structured networks," *arXiv preprint arXiv:1503.02351*, 2015.

3. Encoder-Decoder Based Models

- **Most of the popular DL-based segmentation models use some kind of encoder-decoder architecture.**
- **Two categories**
  - **General Image Segmentation Networks**
  - **Medical and Biomedical Image Segmentation**



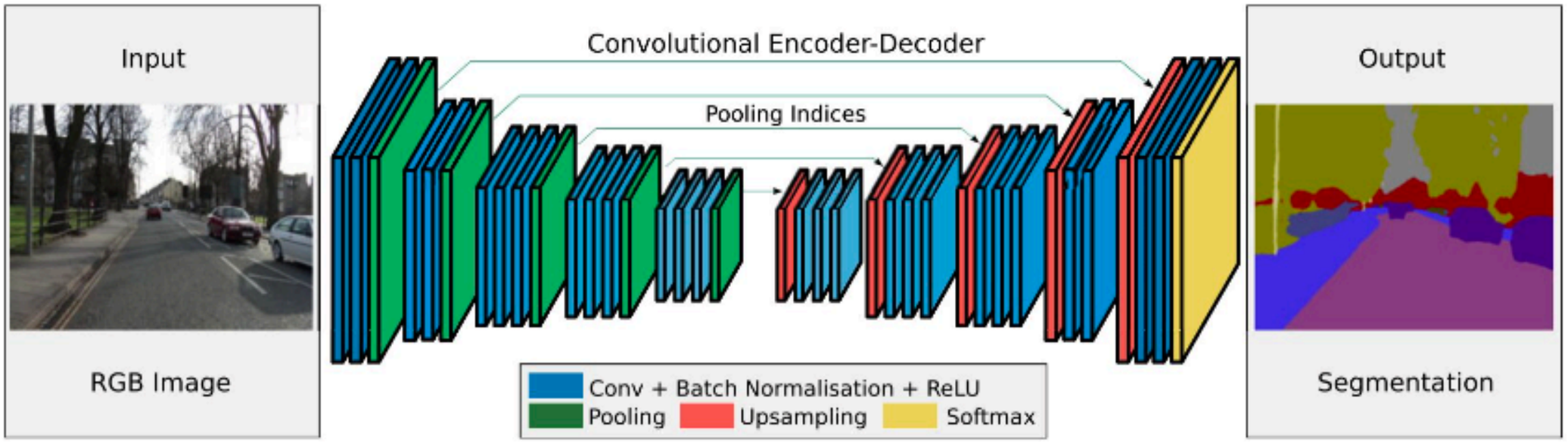
Encoder-Decoder Based Models: General Image Segmentation



Encoder **Decoder**

H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," in *IEEE International Conference on Computer Vision*, 2015, pp. 1520–1528.

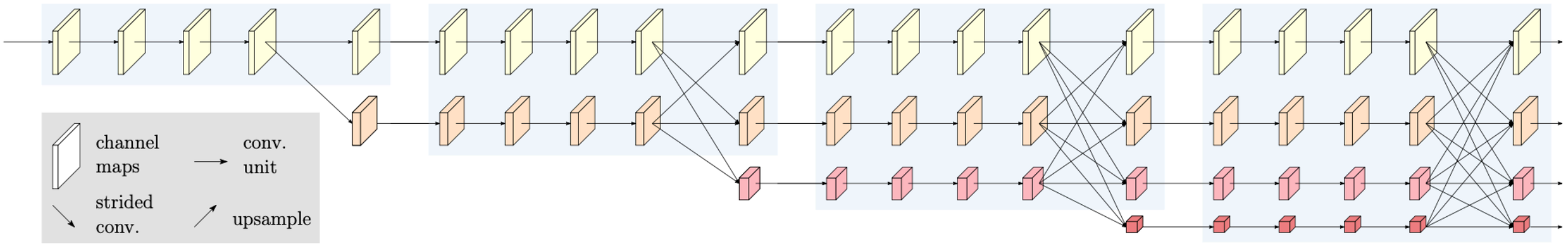
Encoder-Decoder Based Models: General Image Segmentation



- **SegNet: the main novelty**
  - the decoder upsamples its lower-resolution input feature map(s) using pooling indices computed in the max-pooling step of the corresponding encoder to perform nonlinear up-sampling.

V. Badrinarayanan, A. Kendall, and R. Cipolla, "SegNet: A deep convolutional encoder-decoder architecture for image segmenta- tion," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.

Encoder-Decoder Based Models: General Image Segmentation

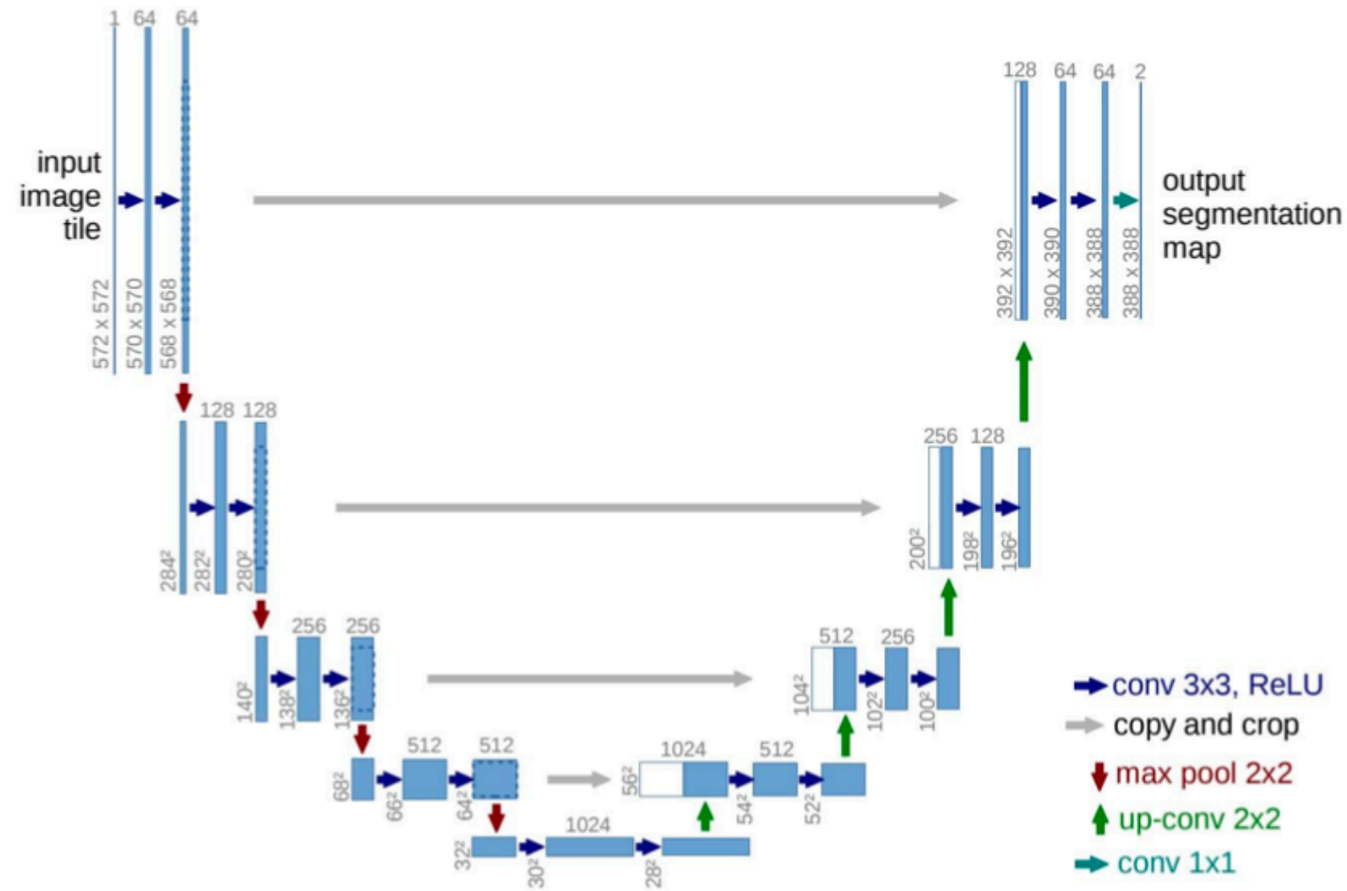


- **HRNet**
  - **To compensate the loss of resolution through the encoding process.**
  - **Maintains high-resolution representations through the encoding process by connecting the high-to-low resolution convolution streams in parallel and repeatedly exchanging the information across resolutions.**

Y. Yuan, X. Chen, and J. Wang, "Object-contextual representations for semantic segmentation," *arXiv preprint arXiv:1909.11065*, 2019.

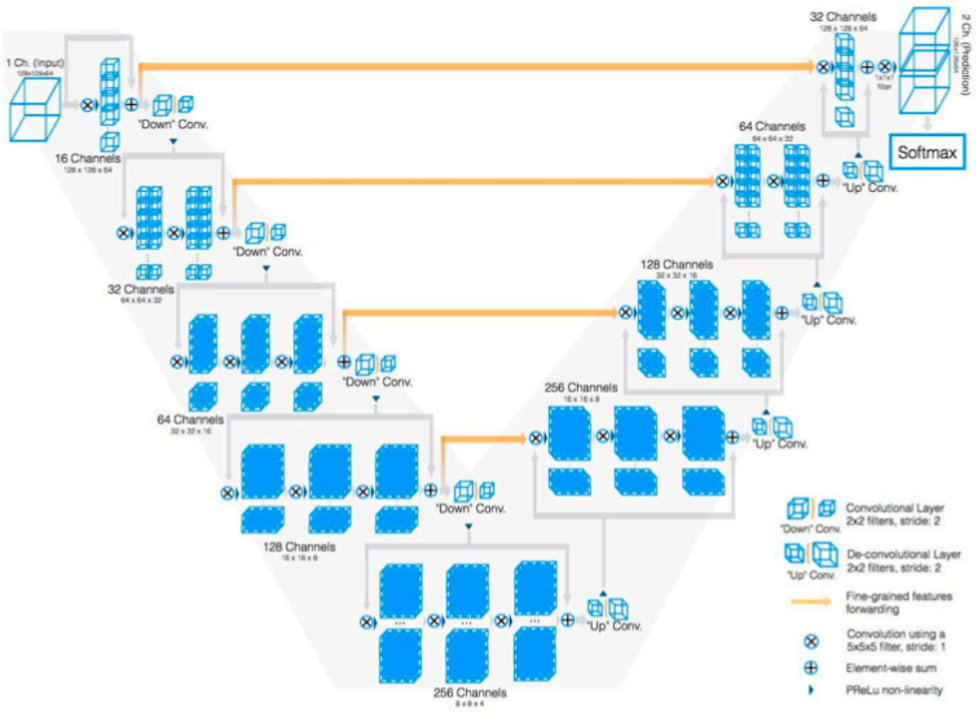
Encoder-Decoder Based Models: Medical Image Segmentation

- **U-Net**
  - **Two parts**
    - **a contracting path to capture context**
    - **a symmetric expanding path that enables precise localization**
  - **The U-Net training strategy relies on the use of data augmentation to learn effectively from very few annotated images.**



O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2015, pp. 234–241.

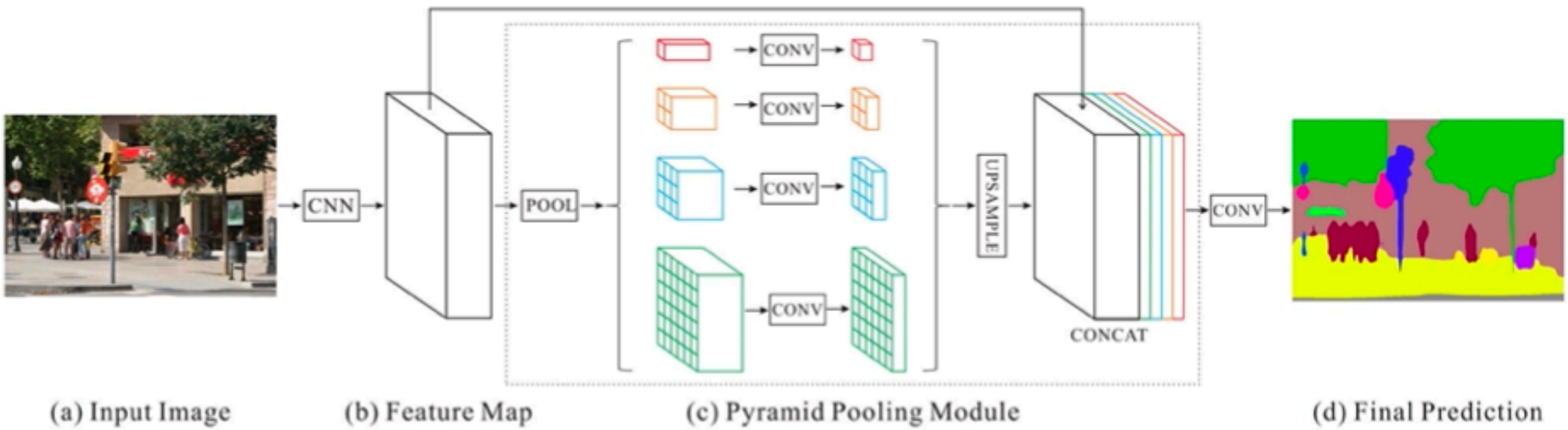
Encoder-Decoder Based Models: Medical Image Segmentation



- **V-Net**
  - **3D medical image segmentation**
  - **new loss function based on the Dice coefficient, enabling the model to deal with strong imbalance between the foreground and background voxels.**

F. Milletari, N. Navab, and S.-A. Ahmadi, "V-Net: Fully convo- lutional neural networks for volumetric medical image segmen- tation," in *International Conference on 3D Vision*. IEEE, 2016, pp. 565–571.

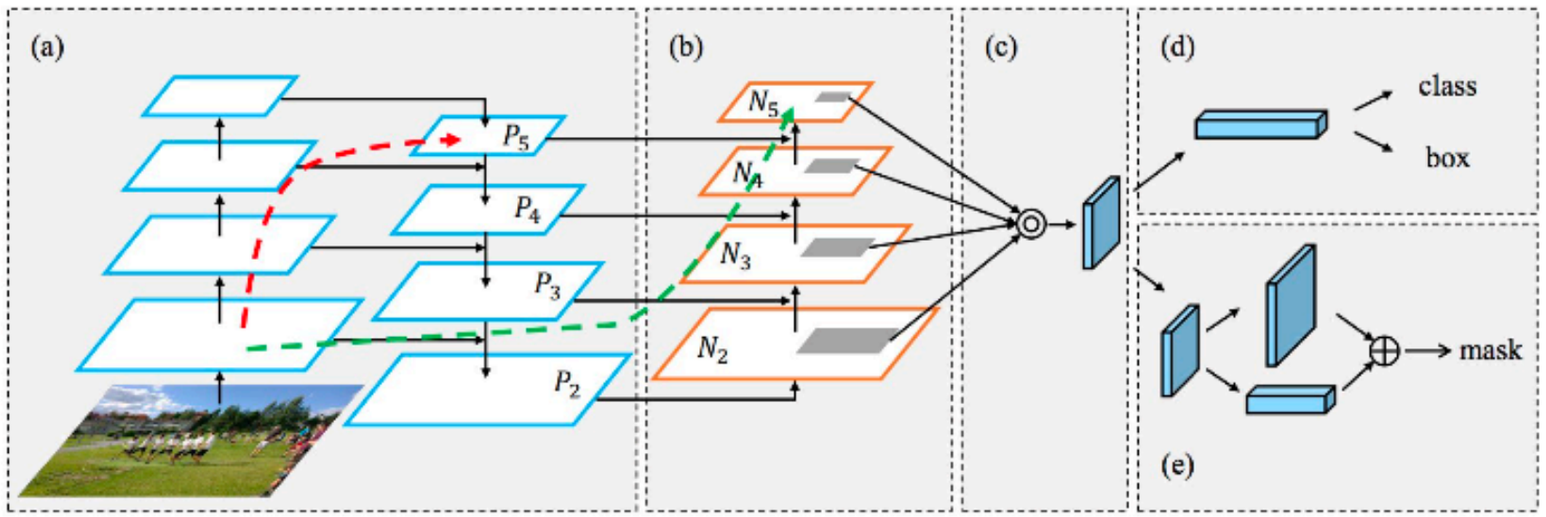
4. Multiscale and Pyramid Network Based Models



- **Pyramid Scene Parsing Network (PSPN)**
  - **multiscale network to better learn the global context representation of a scene.**
  - **The feature maps are pooled at four different scales, each one corresponding to a pyramid level, and processed by a 1×1 convolutional layer to reduce their dimensions**

H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2881–2890.

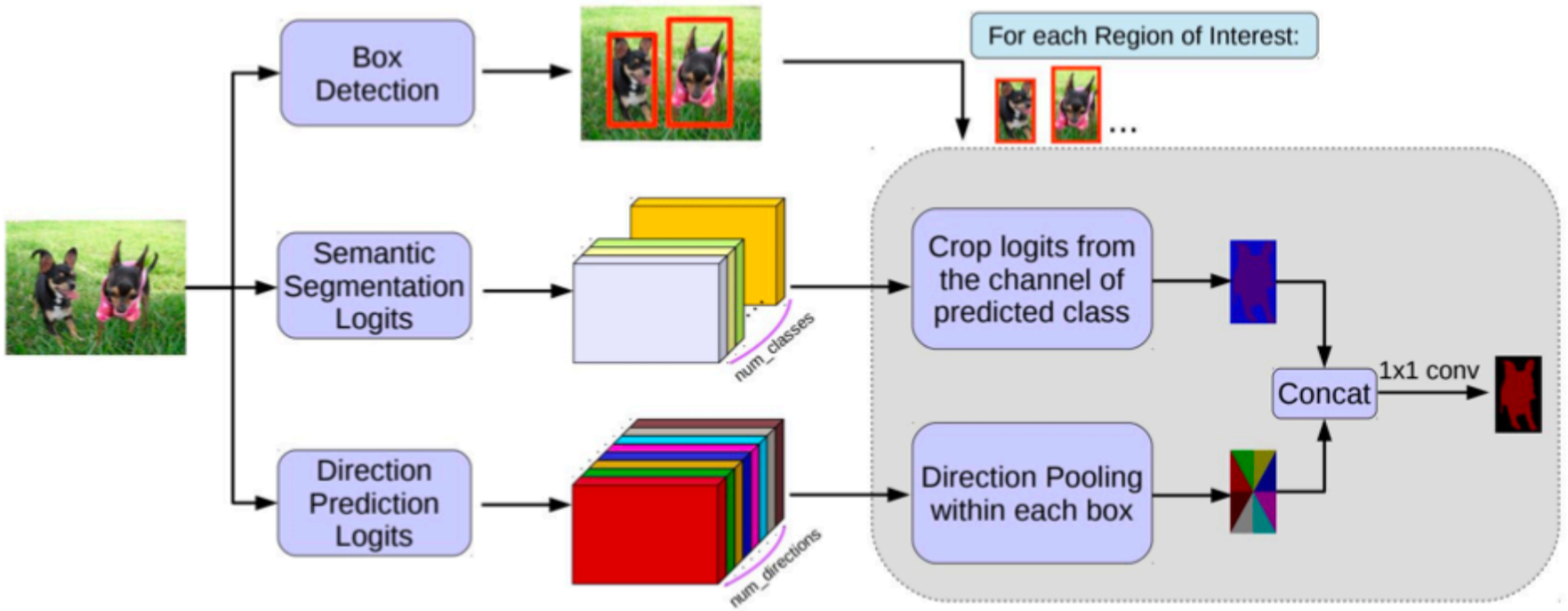
5. R-CNN Based Models



- **Path Aggregation Network (PANet)**
  - **based on the Mask R-CNN and FPN models.**
  - **The feature extractor of the network uses an FPN backbone with a new augmented bottom-up pathway.**
  - **Each stage of this third pathway takes as input the feature maps of the previous stage and processes them with a 3 × 3 convolutional layer. A lateral connection adds the output to the same-stage feature maps of the top-down pathway and these feed the next stage.**

S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path aggregation network for instance segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8759–8768.

R-CNN Based Models

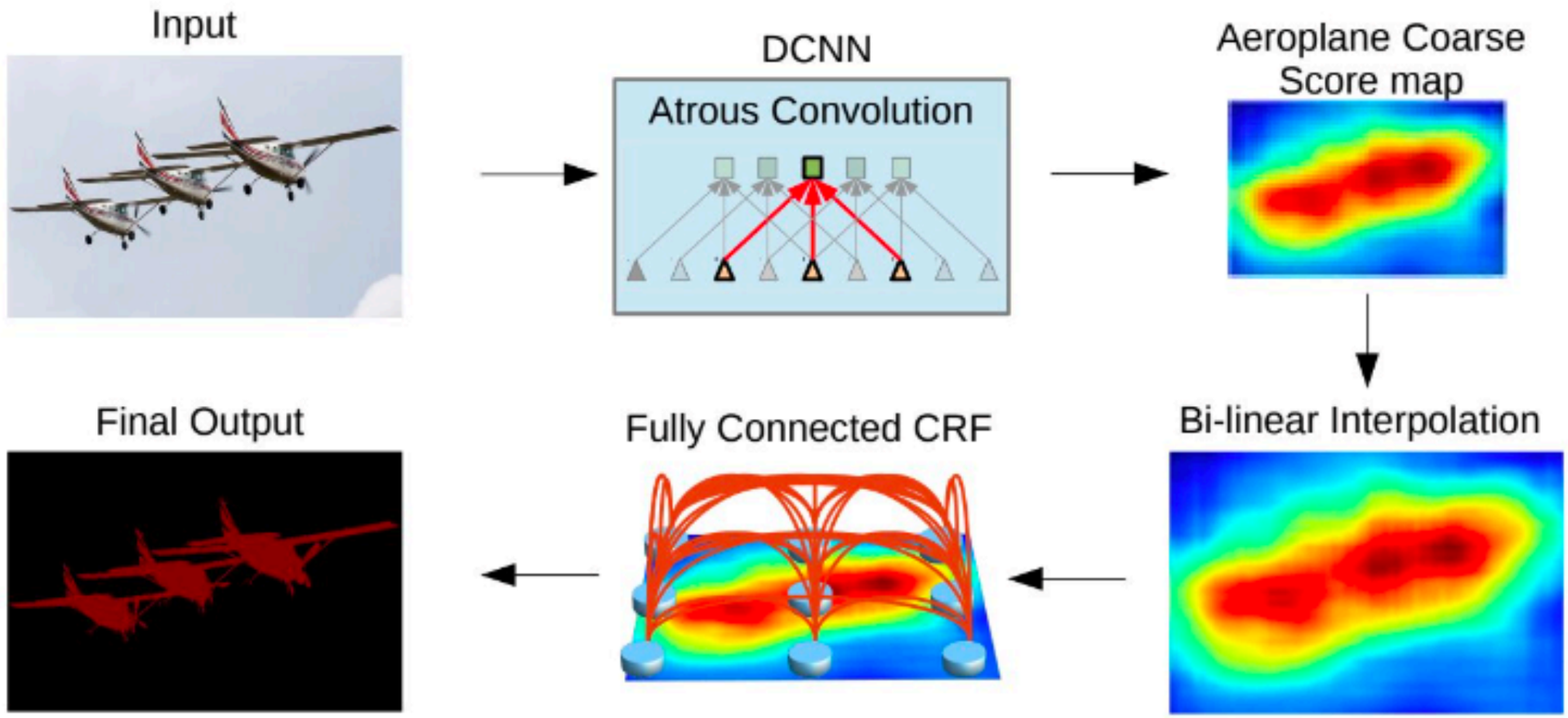


- **MaskLab**
  - **refine object detection with semantic and direction features based on Faster R-CNN.**
  - **Within each region of interest, MaskLab performs foreground/background segmentation by combining semantic and direction prediction.**

L.-C. Chen, A. Hermans, G. Papandreou, F. Schroff, P. Wang, and H. Adam, "Masklab: Instance segmentation by refining object detection with semantic and direction features," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4013–4022.

6. Dilated Convolutional Models

- **Typical method: DeepLab (v2)**



L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "DeepLab: Semantic image segmentation with deep convo- lutional nets, atrous convolution, and fully connected crfs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 4, pp. 834–848, 2017.

Dilated Convolutional Models

- **Typical method: DeepLab**
- **To deal with three challenges:**

(1) **reduced feature resolution**

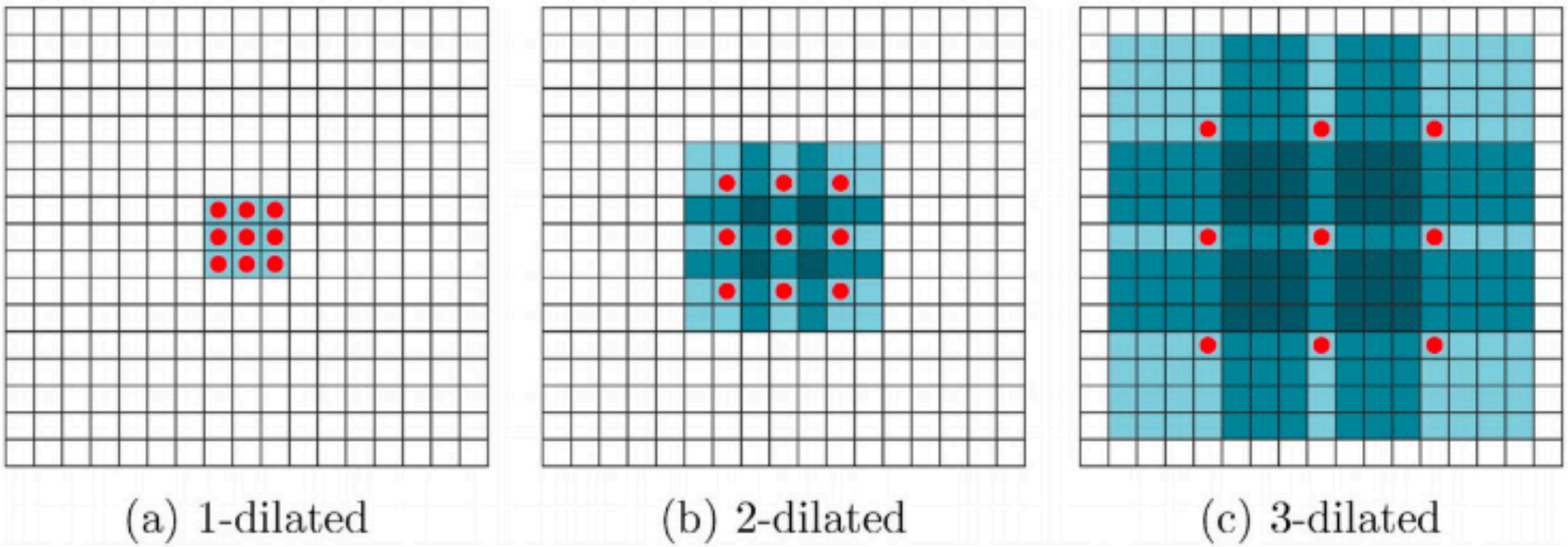
(2) **existence of objects at multiple scales**

(3) **reduced localization accuracy due to DCNN invariance.**

L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "DeepLab: Semantic image segmentation with deep convo- lutional nets, atrous convolution, and fully connected crfs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 4, pp. 834–848, 2017.

Dilated Convolutional Models

(1) reduced feature resolution: dilated convolution

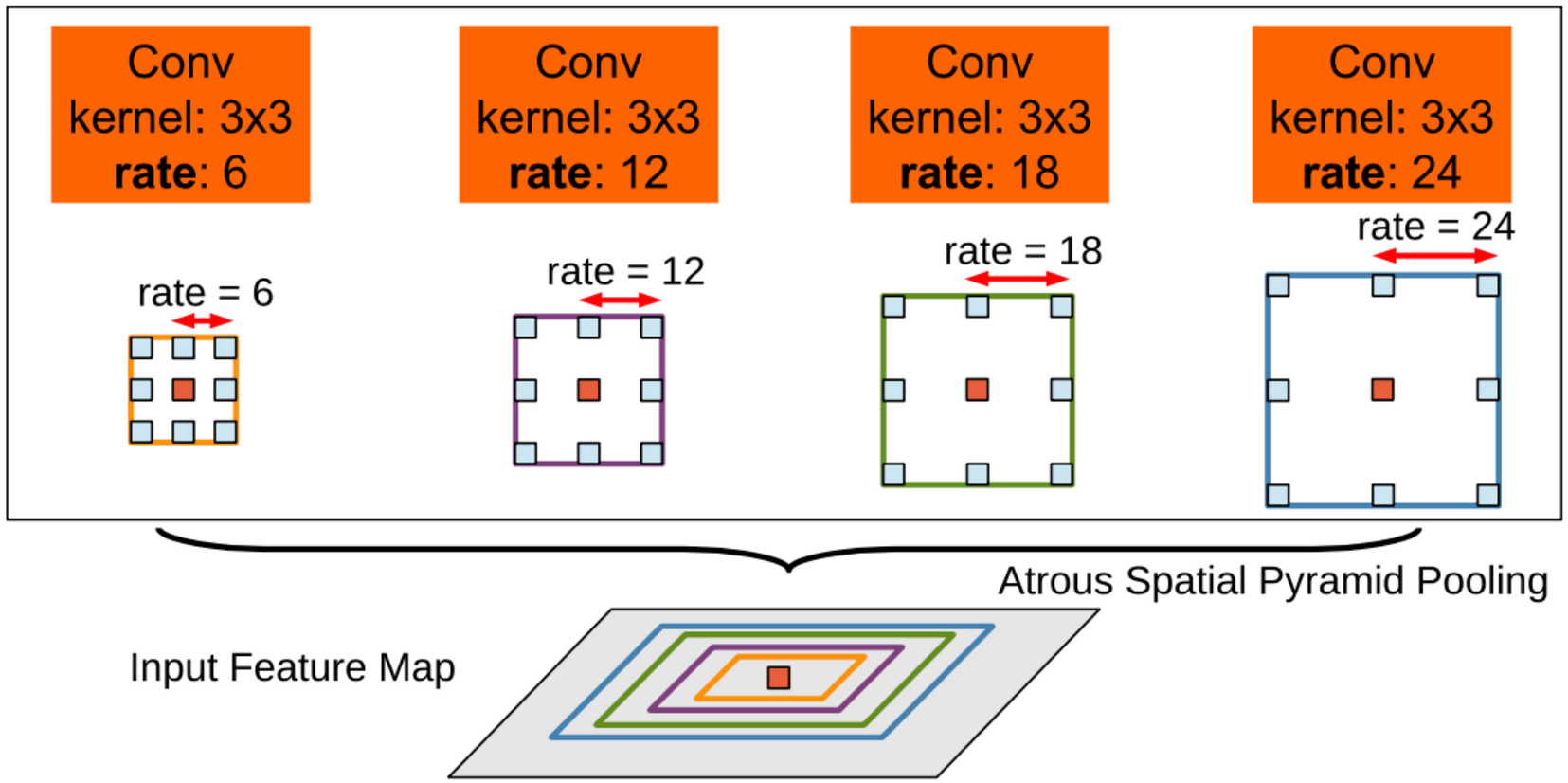


(a) 1-dilated (b) 2-dilated (c) 3-dilated

L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "DeepLab: Semantic image segmentation with deep convo- lutional nets, atrous convolution, and fully connected crfs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 4, pp. 834–848, 2017.

Dilated Convolutional Models

(2) **existence of objects at multiple scales : "atrous spatial pyramid pooling" (ASPP)**



L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "DeepLab: Semantic image segmentation with deep convo- lutional nets, atrous convolution, and fully connected crfs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 4, pp. 834–848, 2017.

Dilated Convolutional Models

(3) **reduced localization accuracy due to DCNN invariance: CNN + CRF (post-processing)**

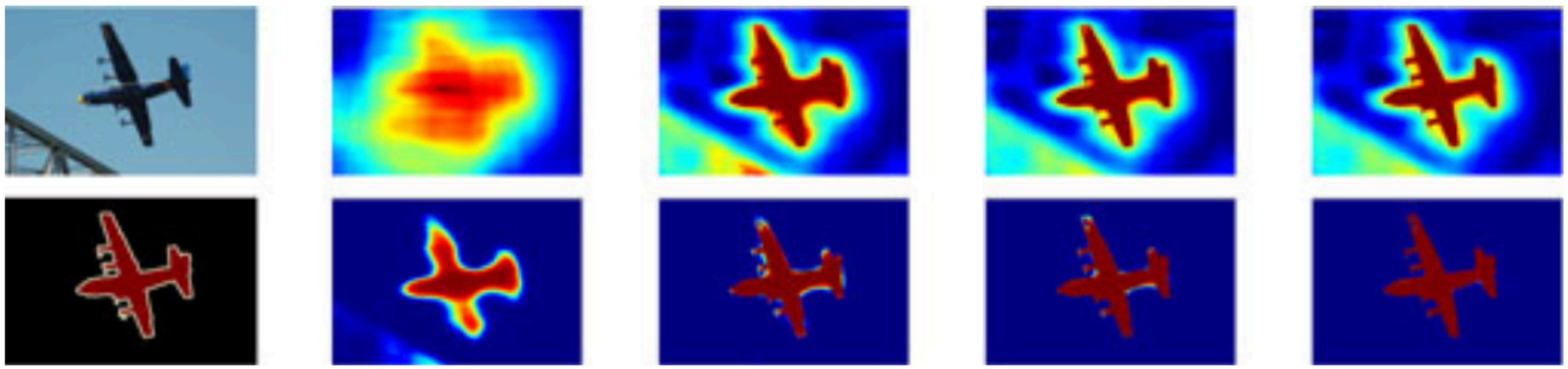
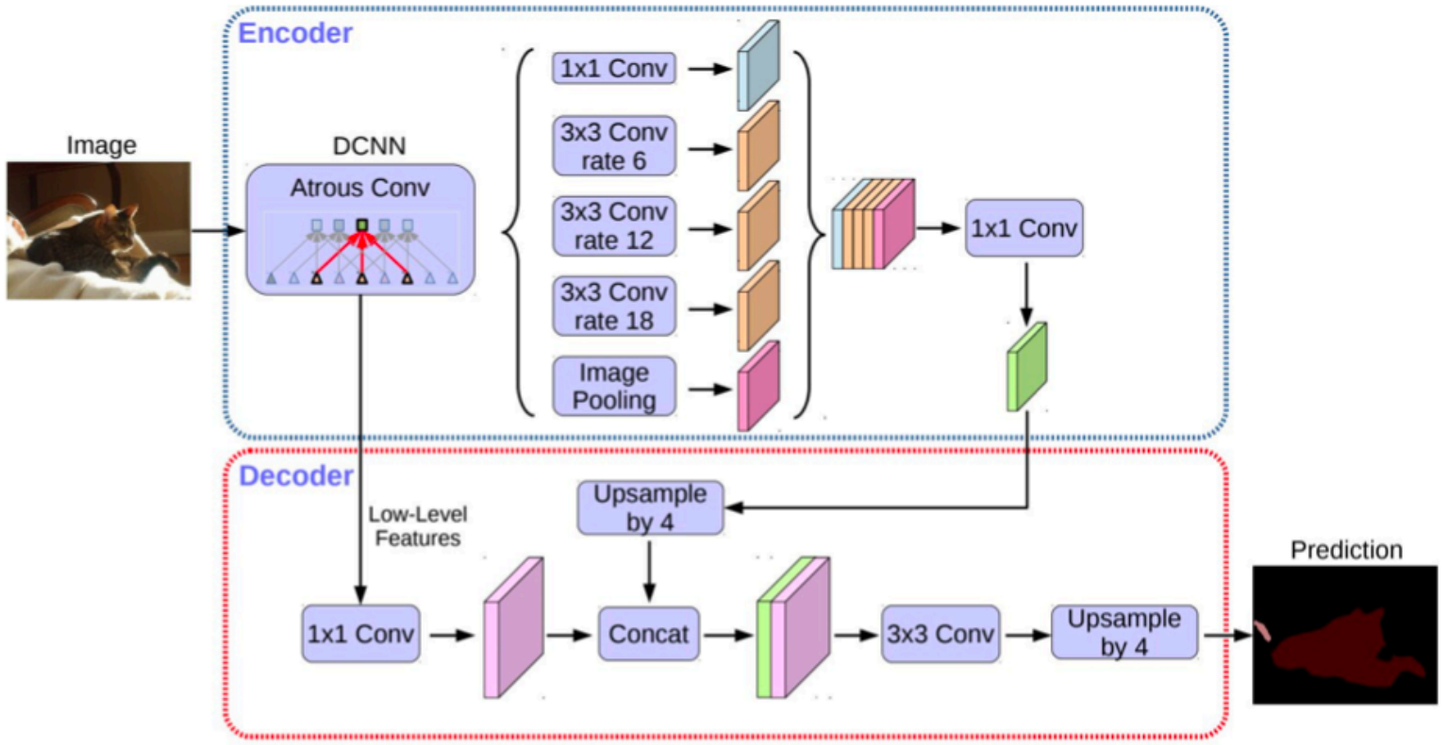


Fig. 5. Score map (input before softmax function) and belief map (output of softmax function) for Aeroplane. We show the score (1st row) and belief (2nd row) maps after each mean field iteration. The output of last DCNN layer is used as input to the mean field inference.

L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "DeepLab: Semantic image segmentation with deep convo- lutional nets, atrous convolution, and fully connected crfs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 4, pp. 834–848, 2017.

Dilated Convolutional Models

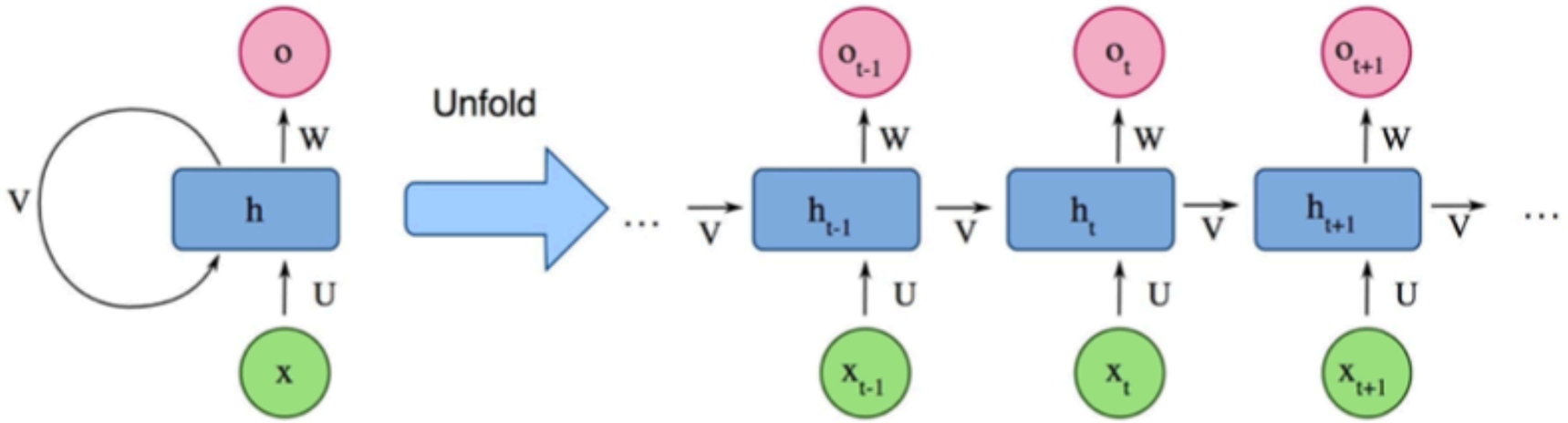


- **DeepLab v3**
  - **combines cascaded and parallel modules of dilated convolutions.**

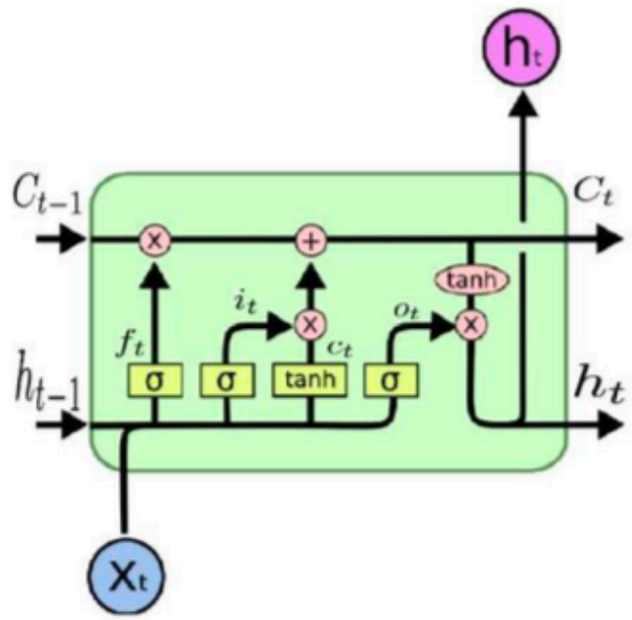
L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *European Conference on Computer Vision*, 2018, pp. 801–818.

7. RNN Based Models

Using RNNs, pixels may be linked together and processed sequentially to model global contexts and improve semantic segmentation

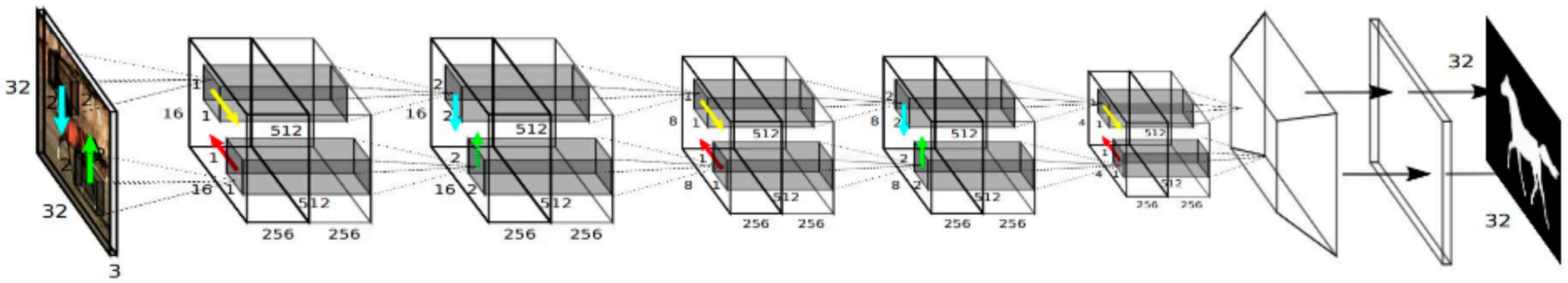


RNN



LSTM

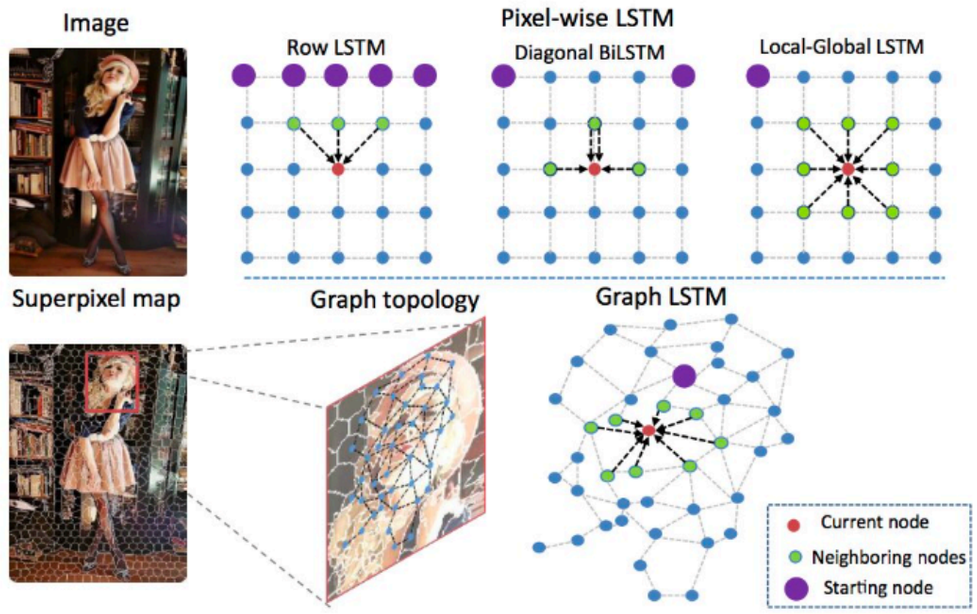
RNN Based Models



- **ReSeg**
  - **Each ReNet layer is composed of four RNNs that sweep the image horizontally and vertically in both directions, encoding patches/activations, and providing relevant global information.**

F. Visin, M. Ciccone, A. Romero, K. Kastner, K. Cho, Y. Bengio, M. Matteucci, and A. Courville, "ReSeg: A recurrent neural network-based model for semantic segmentation," in *IEEE Con- ference on Computer Vision and Pattern Recognition Workshops*, 2016, pp. 41–48.

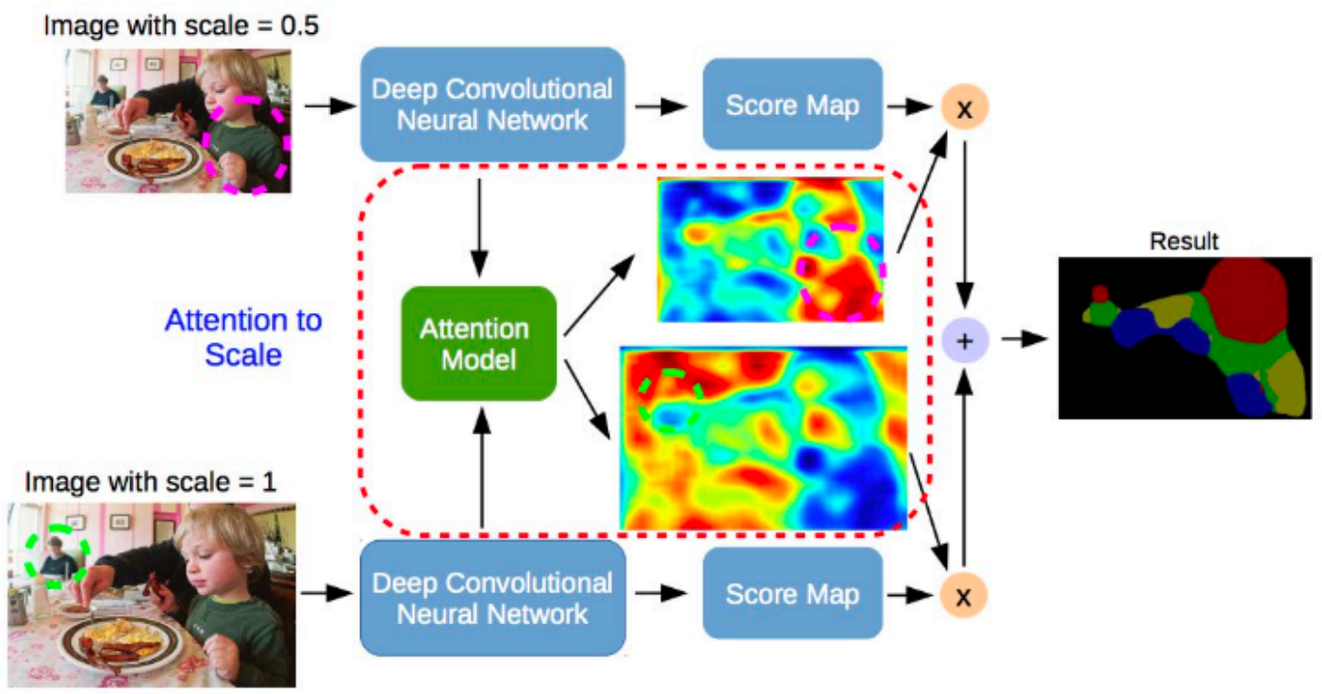
RNN Based Models



- Graph-LSTM network for semantic segmentation
  - convolutional layers are augmented by graph-LSTM layers built on super-pixel maps, which provide a more global structural context.

X. Liang, X. Shen, J. Feng, L. Lin, and S. Yan, "Semantic object parsing with graph LSTM," in *European Conference on Computer Vision*. Springer, 2016, pp. 125–143.

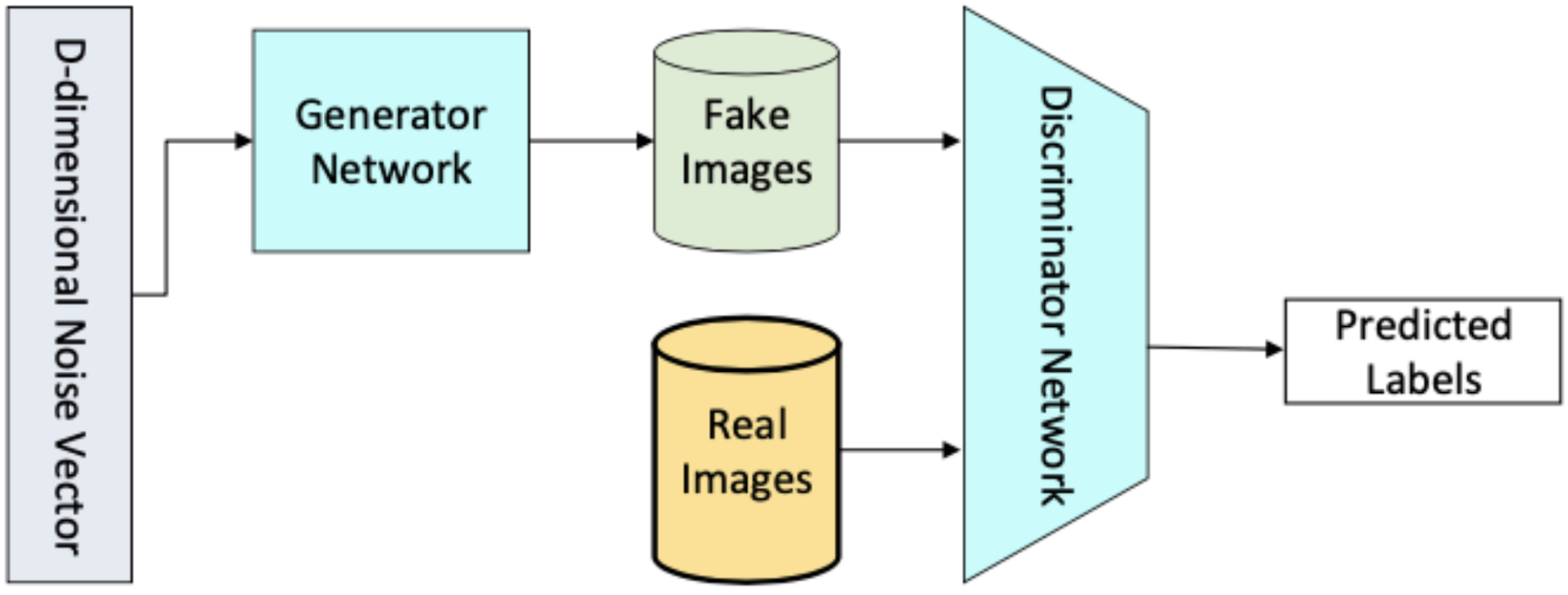
8. Attention-Based Models



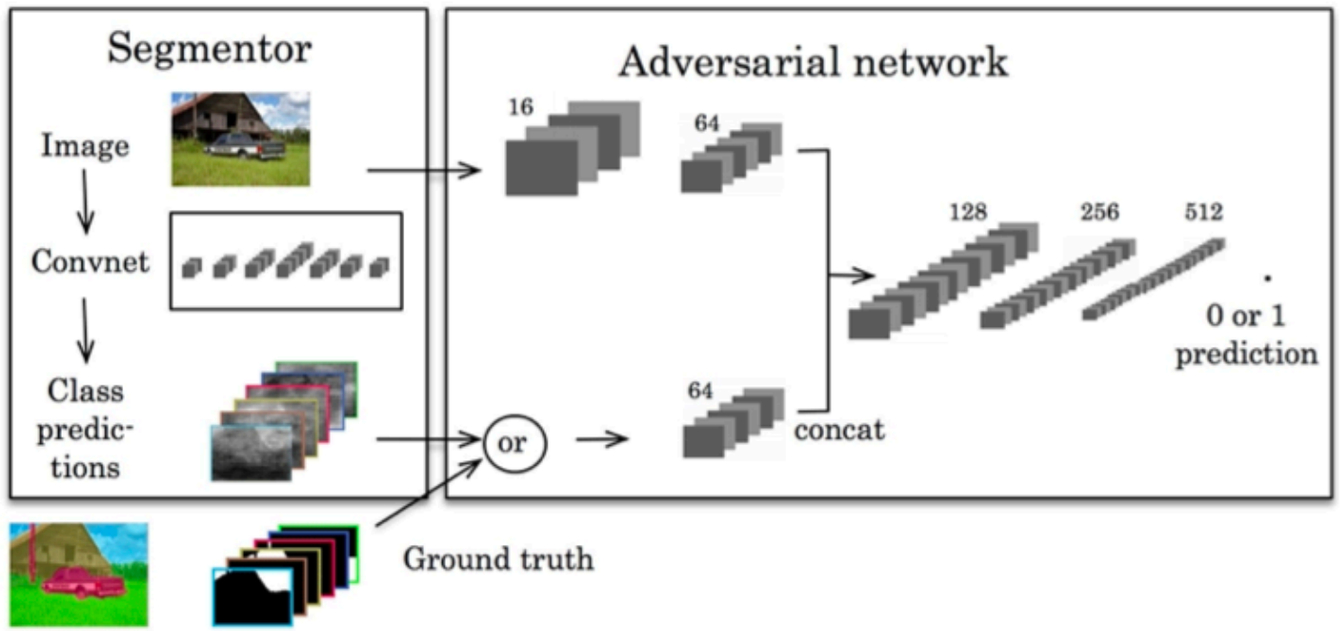
- **Attention mechanism that learns to softly weight multiscale features at each pixel location**

L.-C. Chen, Y. Yang, J. Wang, W. Xu, and A. L. Yuille, "Attention to scale: Scale-aware semantic image segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 3640–3649.

9. Generative Adversarial Networks (GANs)



Generative Models and Adversarial Training



- **Adversarial network that discriminates between ground-truth segmentation maps and those generated by the segmentation network is trained.**

P. Luc, C. Couprie, S. Chintala, and J. Verbeek, "Semantic segmentation using adversarial networks," *arXiv preprint arXiv:1611.08408*, 2016.

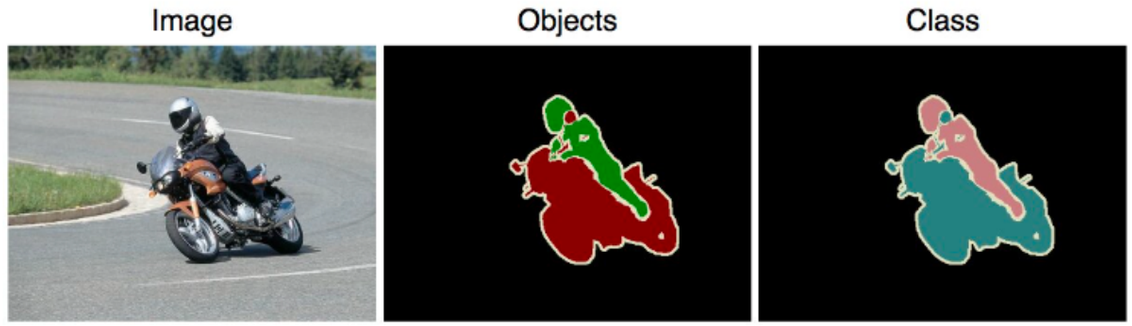
10. CNN Models With Active Contour Models

- **One approach is to formulate new loss functions that are inspired by ACM principles.**
- **To utilize the ACM merely as a post-processor of the output of an FCN.**
- **Truly end-to-end backpropagation trainable, fully-integrated FCN-ACM combination.**

X. Chen, B. M. Williams, S. R. Vallabhaneni, G. Czanner, R. Williams, and Y. Zheng, "Learning active contour models for medical image segmentation," in IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 11 632–11 640.

Datasets - 2D Image Datasets

- **PASCAL Visual Object Classes (VOC)**



- **5 tasks—classification, segmentation, detection, action recognition, and person layout.**
- **For the segmentation task, there are 21 labeled object classes and pixels are labeled as background if they do not belong to any of these classes. The dataset is divided into two sets, training and validation, with 1,464 and 1,449 images, respectively.**

M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zis- serman, "The PASCAL visual object classes (VOC) challenge," *International Journal of Computer Vision*, vol. 88, pp. 303–338, 2010. http://host.robots.ox.ac.uk/pascal/VOC/voc2012/.

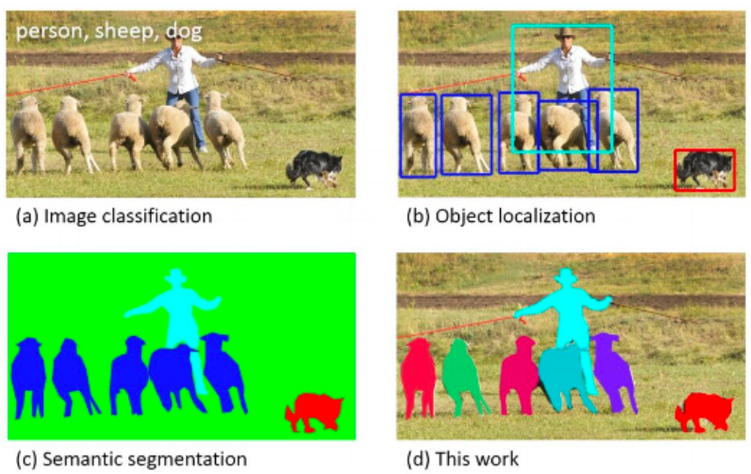
Datasets - 2D Image Datasets

- **PASCAL Context: is an extension of the PASCAL VOC 2010 detection challenge.**
- **It contains more than 400 classes (including the original 20 classes plus backgrounds from PASCAL VOC segmentation), in three categories (objects, stuff, and hybrids). Many of the object categories of this dataset are too sparse and; therefore, a subset of 59 classes is usually selected for use.**

R. Mottaghi, X. Chen, X. Liu, N.-G. Cho, S.-W. Lee, S. Fidler, R. Urtasun, and A. Yuille, "The role of context for object detection and semantic segmentation in the wild," in IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 891–898.

Datasets - 2D Image Datasets

- **Microsoft Common Objects in Context (MS COCO) a large-scale object detection, segmentation, and captioning dataset.**
- **COCO contains photos of 91 object types, with a total of 2.5 million labeled instances in 328K images.**



T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dolla´r, and C. L. Zitnick, "Microsoft COCO: Common objects in context," in *European Conference on Computer Vision*. Springer, 2014.

Datasets - 2D Image Datasets

- **Cityscapes is a large database with a focus on semantic understanding of urban street scenes.**
- **It contains a diverse set of stereo video sequences recorded in street scenes from 50 cities, with high quality pixel-level annotation of 5K frames, in addition to a set of 20K weakly annotated frames.**
- **It includes semantic and dense pixel annotations of 30 classes, grouped into 8 categories—flat surfaces, humans, vehicles, constructions, objects, nature, sky, and void.**

M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Be- nenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 3213–3223.

Datasets - 2D Image Datasets

- **ADE20K / MIT Scene Parsing (SceneParse150)**
- **SiftFlow**
- **Stanford Background**
- **Berkeley Segmentation Dataset (BSD)**
- **Youtube-Objects**
- **CamVid**
- **KITTI**
- **Semantic Boundaries Dataset (SBD)** [162],
- **PASCAL Part** [163],
- **SYNTHIA** [164]
- **Adobe's Portrait Segmentation** [165].

Datasets - 2.5D Datasets

- **ScanNet: RGB-D video dataset containing 2.5 million views in more than 1,500 scans, annotated with 3D camera poses, surface reconstructions, and instance level semantic segmentations.**
- **Stanford 2D-3D: a variety of mutually registered 2D, 2.5D, and 3D modalities, with instance level semantic and geometric annotations, acquired from 6 indoor areas. It contains over 70,000 RGB images, along with the corresponding depths, surface normals, semantic annotations, as well as global XYZ images, camera information, and registered raw and semantically annotated 3D meshes and point clouds.**

Datasets - 2.5D Datasets

- **NYU-Depth V2: consists of video sequences from a variety of indoor scenes, recorded by the RGB and depth cameras of the Microsoft Kinect. It includes 1,449 densely labeled RGB and depth image pairs of more than 450 scenes taken from 3 cities. Each object is labeled with a class and instance number (e.g., cup1, cup2, cup3, etc.). It also contains 407,024 unlabeled frames.**
- **SUN-3D: is a large RGB-D video dataset that contains 415 sequences captured from 254 different spaces in 41 different buildings; 8 sequences are annotated and more will be annotated in the future. Each annotated frame provides the semantic segmentation of the objects in the scene as well as information about the camera pose.**

Evaluation Metrics

- **Pixel accuracy: the ratio of properly classified pixels divided by the total number of pixels.**
- **Mean Pixel Accuracy (MPA) : the ratio of correct pixels is computed in a per-class manner and then averaged over the total number of classes.**
- **Intersection over Union (IoU)**, or the **Jaccard Index: the area of intersection between the predicted segmentation map A and the ground truth map B, divided by the area of the union between the two maps, and ranges between 0 and 1**
- **Precision / Recall / F1 score**
- **Dice coefficient**

Quantitative Performance of DL-Based Models

TABLE 1
Accuracies of segmentation models on the PASCAL VOC test set

Method	Backbone	mIoU
FCN [30]	VGG-16	62.2
CRF-RNN [38]	-	72.0
CRF-RNN* [38]	-	74.7
BoxSup* [119]	-	75.1
Piecewise* [39]	-	78.0
DPN* [40]	-	77.5
DeepLab-CRF [76]	ResNet-101	79.7
GCN* [120]	ResNet-152	82.2
Dynamic Routing [142]	-	84.0
RefineNet [117]	ResNet-152	84.2
Wide ResNet [121]	WideResNet-38	84.9
PSPNet [54]	ResNet-101	85.4
DeeplabV3 [13]	ResNet-101	85.7
PSANet [98]	ResNet-101	85.7
EncNet [116]	ResNet-101	85.9
DFN* [99]	ResNet-101	86.2
Exfuse [122]	ResNet-101	86.2
SDN* [43]	DenseNet-161	86.6
DIS [125]	ResNet-101	86.8
APC-Net* [58]	ResNet-101	87.1
EMANet [95]	ResNet-101	87.7
DeeplabV3+ [81]	Xception-71	87.8
Exfuse [122]	ResNeXt-131	87.9
MSCI [59]	ResNet-152	88.0
EMANet [95]	ResNet-152	88.2
DeeplabV3+* [81]	Xception-71	89.0
EfficientNet+NAS-FPN [137]	-	90.5

* Models pre-trained on other datasets (MS-COCO, ImageNet, etc.).

TABLE 2
Accuracies of segmentation models on the Cityscapes dataset

Method	Backbone	mIoU
SegNet [25]	-	57.0
FCN-8s [30]	-	65.3
DPN [40]	-	66.8
Dilation10 [77]	-	67.1
DeeplabV2 [76]	ResNet-101	70.4
RefineNet [117]	ResNet-101	73.6
FoveaNet [126]	ResNet-101	74.1
Ladder DenseNet [127]	Ladder DenseNet-169	73.7
GCN [120]	ResNet-101	76.9
DUC-HDC [78]	ResNet-101	77.6
Wide ResNet [121]	WideResNet-38	78.4
PSPNet [54]	ResNet-101	85.4
BiSeNet [128]	ResNet-101	78.9
DFN [99]	ResNet-101	79.3
PSANet [98]	ResNet-101	80.1
DenseASPP [79]	DenseNet-161	80.6
Dynamic Routing [142]	-	80.7
SPGNet [129]	2xResNet-50	81.1
DANet [91]	ResNet-101	81.5
CCNet [96]	ResNet-101	81.4
DeeplabV3 [13]	ResNet-101	81.3
IPC [141]	ResNet-101	81.8
AC-Net [131]	ResNet-101	82.3
OCR [42]	ResNet-101	82.4
ResNeSt200 [93]	ResNeSt-200	82.7
GS-CNN [130]	WideResNet	82.8
HA-Net [94]	ResNext-101	83.2
HRNetV2+OCR [42]	HRNetV2-W48	83.7
Hierarchical MSA [139]	HRNet-OCR	85.1

Quantitative Performance of DL-Based Models

TABLE 3
Accuracies of segmentation models on the MS COCO stuff dataset

Method	Backbone	mIoU
RefineNet [117]	ResNet-101	33.6
CCN [57]	Ladder DenseNet-101	35.7
DANet [91]	ResNet-50	37.9
DSSPN [132]	ResNet-101	37.3
EMA-Net [95]	ResNet-50	37.5
SGR [133]	ResNet-101	39.1
OCR [42]	ResNet-101	39.5
DANet [91]	ResNet-101	39.7
EMA-Net [95]	ResNet-50	39.9
AC-Net [131]	ResNet-101	40.1
OCR [42]	HRNetV2-W48	40.5

TABLE 4
Accuracies of segmentation models on the ADE20k validation dataset

Method	Backbone	mIoU
FCN [30]	-	29.39
DilatedNet [77]	-	32.31
CascadeNet [134]	-	34.90
RefineNet [117]	ResNet-152	40.7
PSPNet [54]	ResNet-101	43.29
PSPNet [54]	ResNet-269	44.94
EncNet [116]	ResNet-101	44.64
SAC [135]	ResNet-101	44.3
PSANet [98]	ResNet-101	43.70
UperNet [136]	ResNet-101	42.66
DSSPN [132]	ResNet-101	43.68
DM-Net [56]	ResNet-101	45.50
AC-Net [131]	ResNet-101	45.90
ResNeSt-101 [93]	ResNeSt-101	46.91
ResNeSt-200 [93]	ResNeSt-200	48.36

Quantitative Performance of DL-Based Models

TABLE 5
Instance segmentation model performance on COCO test-dev 2017

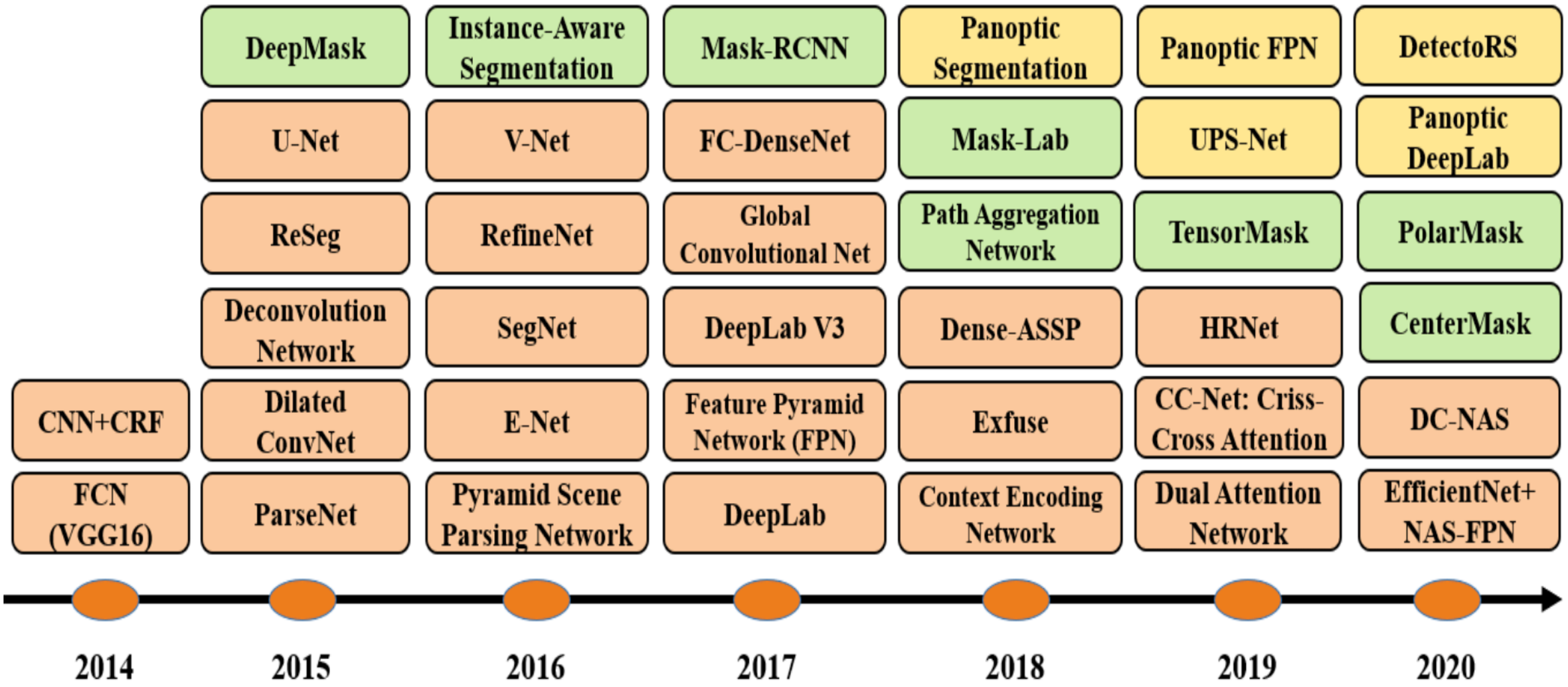
Method	Backbone	FPS	AP
YOLACT-550 [74]	R-101-FPN	33.5	29.8
YOLACT-700 [74]	R-101-FPN	23.8	31.2
RetinaMask [172]	R-101-FPN	10.2	34.7
TensorMask [67]	R-101-FPN	2.6	37.1
SharpMask [173]	R-101-FPN	8.0	37.4
Mask-RCNN [62]	R-101-FPN	10.6	37.9
CenterMask [72]	R-101-FPN	13.2	38.3

TABLE 7
Segmentation model performance on the NYUD-v2 and SUN-RGBD

	NYUD-v2		SUN-RGBD	
Method	m-Acc	m-IoU	m-Acc	m-IoU
Mutex [177]	-	31.5	-	-
MS-CNN [178]	45.1	34.1	-	-
FCN [30]	46.1	34.0	-	-
Joint-Seg [179]	52.3	39.2	-	-
SegNet [25]	-	-	44.76	31.84
Structured Net [39]	53.6	40.6	53.4	42.3
B-SegNet [180]	-	-	45.9	30.7
3D-GNN [181]	55.7	43.1	57.0	45.9
LSD-Net [46]	60.7	45.9	58.0	-
RefineNet [117]	58.9	46.5	58.5	45.9
D-aware CNN [182]	61.1	48.4	53.5	42.0
MTI-Net [183]	62.9	49	-	-
RDFNet [184]	62.8	50.1	60.1	47.7
ESANet-R34-NBt1D [140]	-	50.3	-	48.17
G-Aware Net [185]	68.7	59.6	74.9	54.5

Challenges and Opportunities

- More Challenging Datasets
- Combining DL and Earlier Segmentation Models
- Interpretable Deep Models
- Weakly-Supervised and Unsupervised Learning
- Real-time Models for Various Applications
- Memory Efficient Models
- Applications



Timeline of representative DL-based image segmentation algorithms. Orange, green, and yellow blocks indicate semantic, instance, and panoptic segmentation algorithms, respectively.