

Computer Vision

Image Segmentation

Dr. Mrinmoy Ghorai

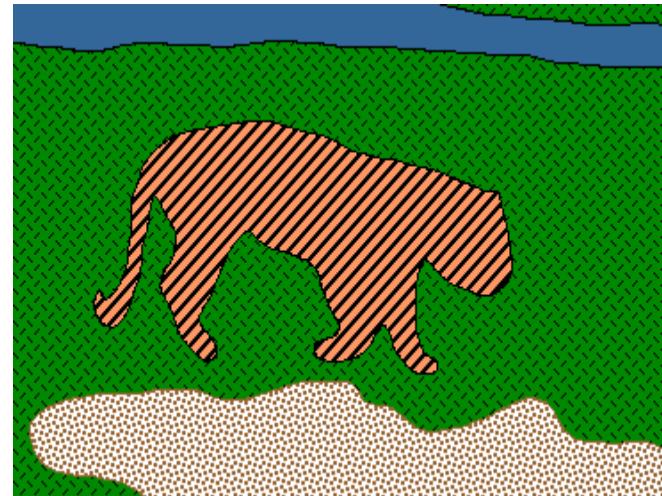
**Indian Institute of Information Technology
Sri City, Chittoor**



Today's class

Image Segmentation

Group pixels into meaningful or perceptually similar regions



Today's class

- Segmentation and grouping
 - Clustering (k-means, mean-shift)
 - Graph (graph cuts)
- Semantic Segmentation
 - Sliding Window
 - FCN
 - DeconNet
- Instance Segmentation
 - Mask-RCNN

Segmentation using clustering

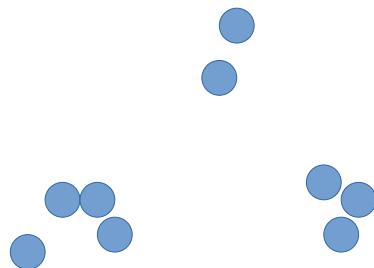
- Kmeans
- Mean-shift

K-means algorithm

$$\operatorname{argmin}_{S, \mu_i, i=1..K} \sum_{i=1}^K \sum_{x \in S_i} \|x - \mu_i\|^2$$

Partition the data into K sets $S = \{S_1, S_2, \dots S_K\}$ with corresponding centers μ_i

Partition such that variance in each partition is as low as possible

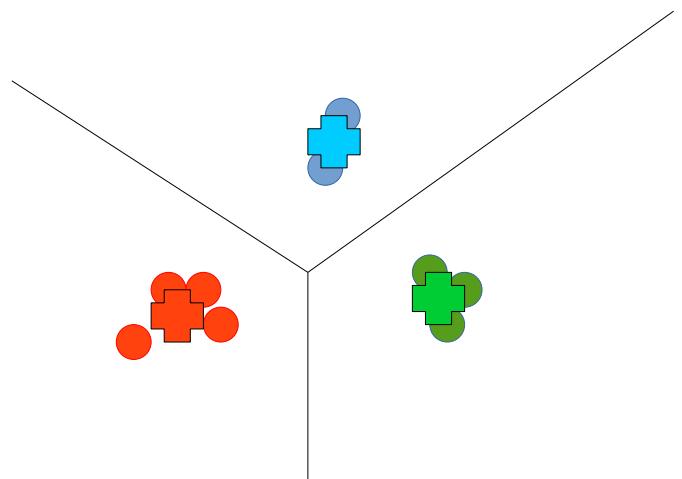


K-means algorithm

$$\operatorname{argmin}_{S, \mu_i, i=1..K} \sum_{i=1}^K \sum_{x \in S_i} \|x - \mu_i\|^2$$

Partition the data into K sets $S = \{S_1, S_2, \dots S_K\}$ with corresponding centers μ_i

Partition such that **variance** in each partition is **as low as possible**



K-means algorithm

1. Initialize K centers μ_i (usually randomly)

K-means algorithm

1. Initialize K centers μ_i (usually randomly)

2. Assign each point x to its nearest center:

$$S^t = \operatorname{argmin}_S \sum_{i=1}^K \sum_{x \in S_i} \|x - \mu_i\|^2$$

K-means algorithm

1. Initialize K centers μ_i (usually randomly)

2. Assign each point x to its nearest center:

$$S^t = \operatorname{argmin}_S \sum_{i=1}^K \sum_{x \in S_i} \|x - \mu_i\|^2$$

3. Update cluster centers as the mean of its members

K-means algorithm

1. Initialize K centers μ_i (usually randomly)

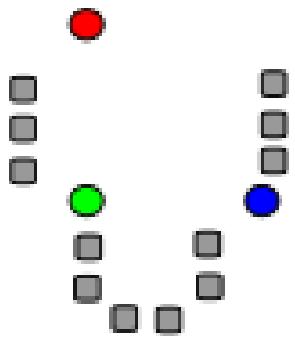
2. Assign each point x to its nearest center:

$$S^t = \operatorname{argmin}_S \sum_{i=1}^K \sum_{x \in S_i} \|x - \mu_i\|^2$$

3. Update cluster centers as the mean of its members

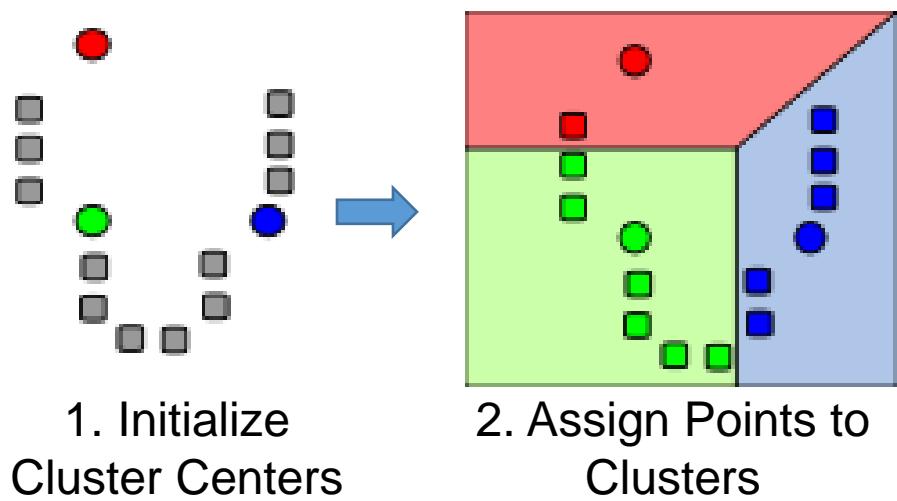
4. Repeat 2-3 until convergence ($t = t+1$)

K-means clustering

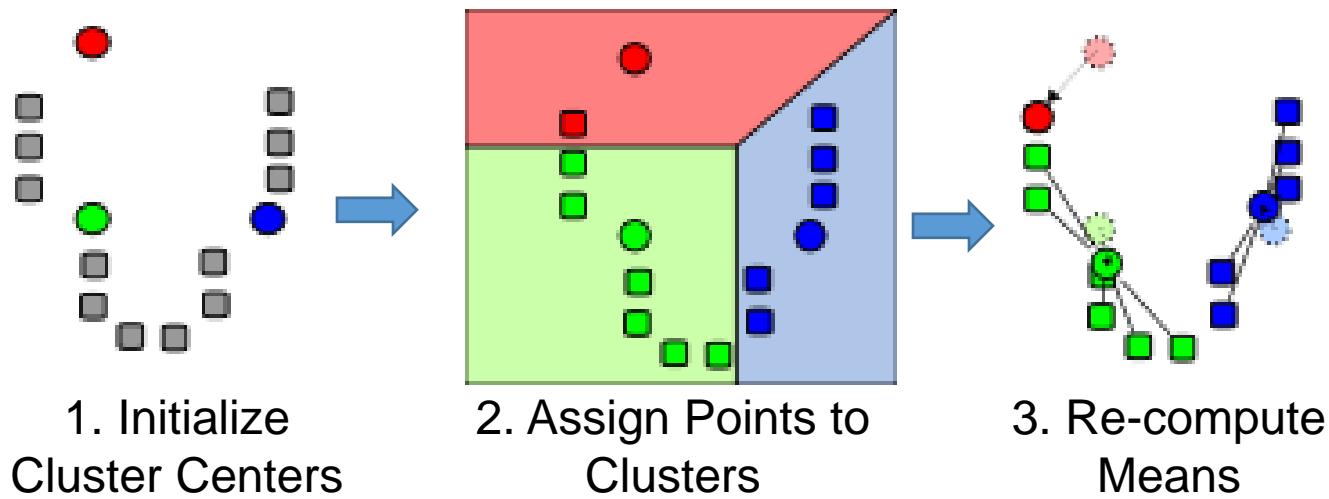


1. Initialize
Cluster Centers

K-means clustering



K-means clustering



K-means clustering

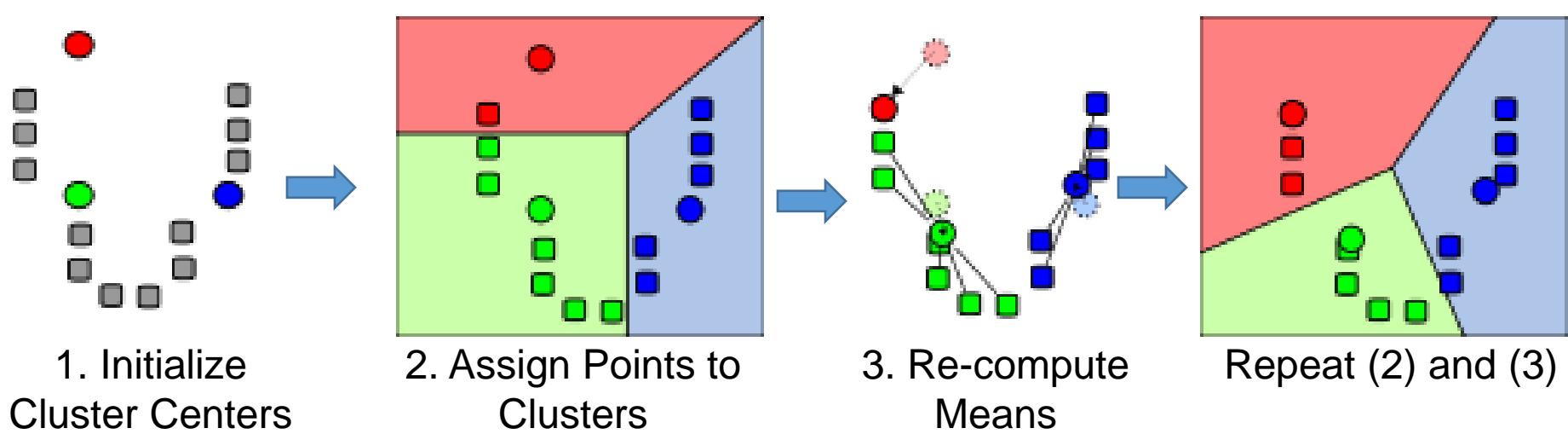


Image Segmentation by K-Means

- Select a value of K
- Select a feature vector for every pixel (color, texture, position, or combination of these etc.)
- Define a similarity measure between feature vectors (Usually Euclidean Distance).
- Apply K-Means Algorithm.
- Merge any components of size less than some threshold to an adjacent component that is most similar to it.

K-means clustering using intensity alone and color alone

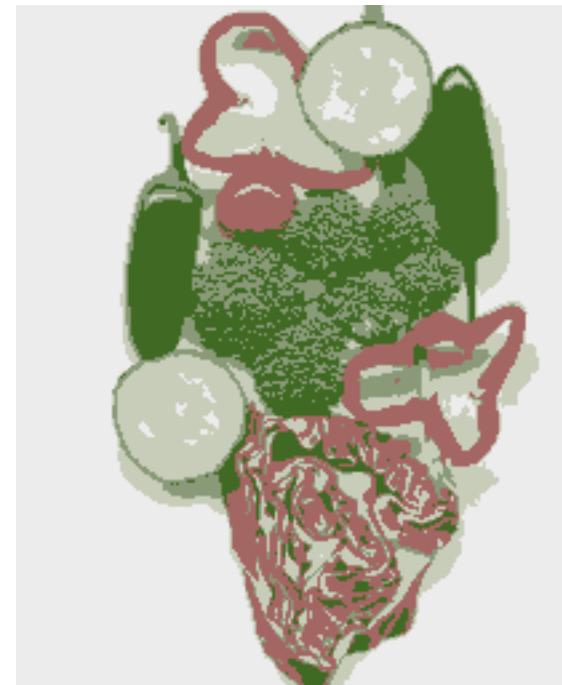
Image



Clusters on intensity



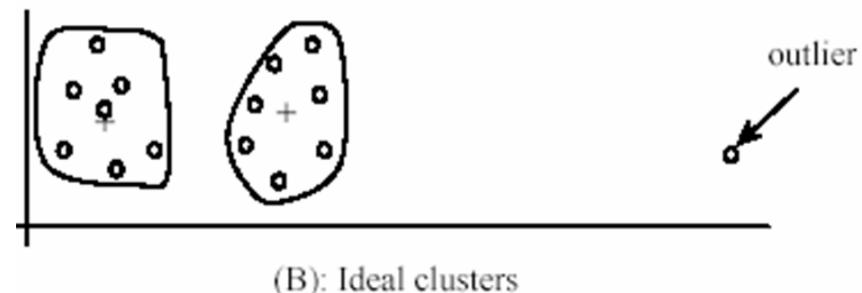
Clusters on color



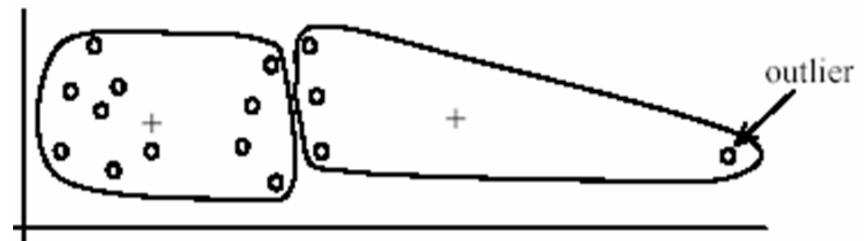
There is no requirement that clusters be spatially localized and they're not

K-Means pros and cons

- Pros
 - Simple and fast
 - Easy to implement
- Cons
 - Need to choose K
 - Sensitive to outliers
- Usage
 - Unsupervised clustering
 - Rarely used for pixel segmentation

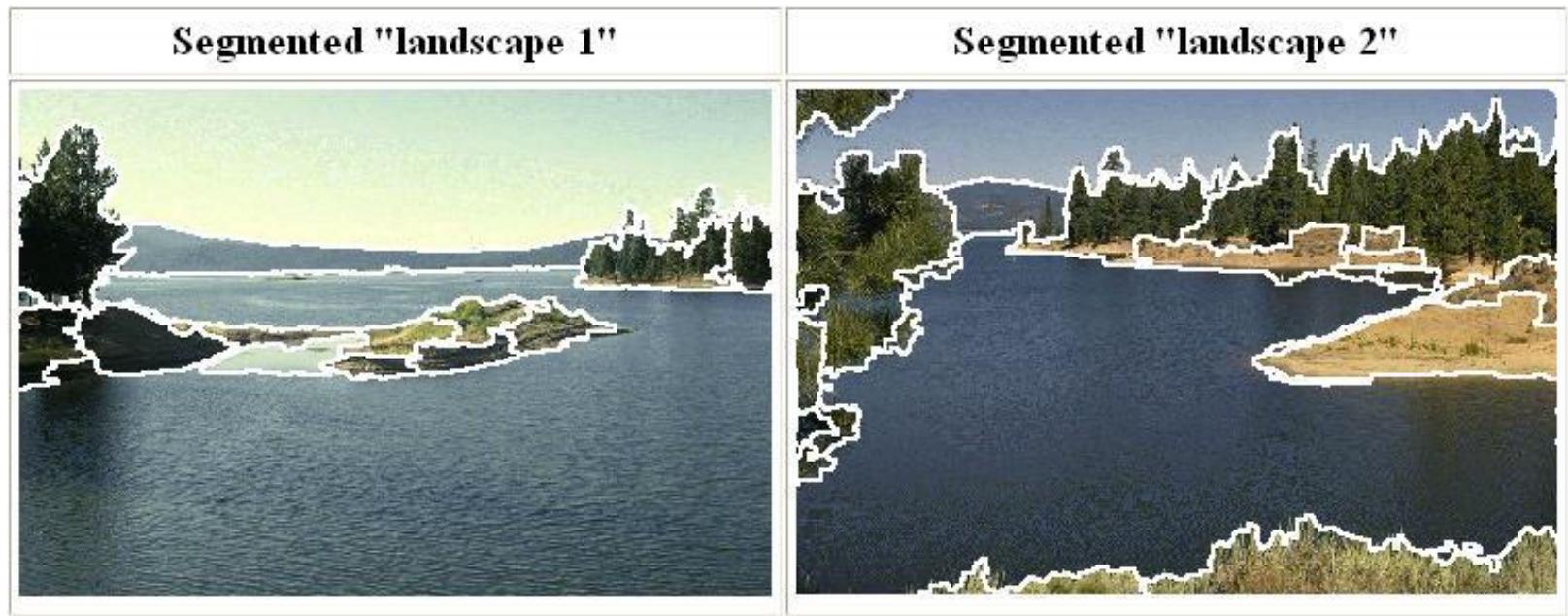


(B): Ideal clusters



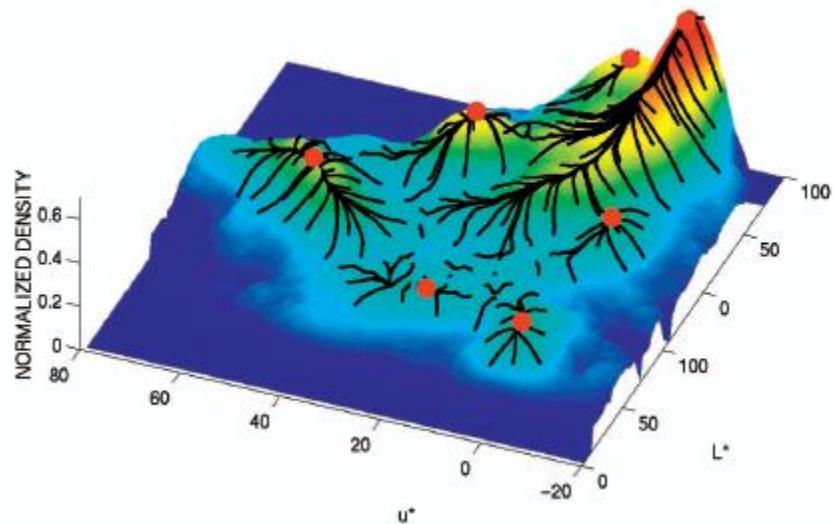
Mean shift segmentation

- Versatile technique for clustering-based segmentation



Mean shift algorithm

- Find *modes* of the non-parametric density
- Locate the **maxima** of a density function

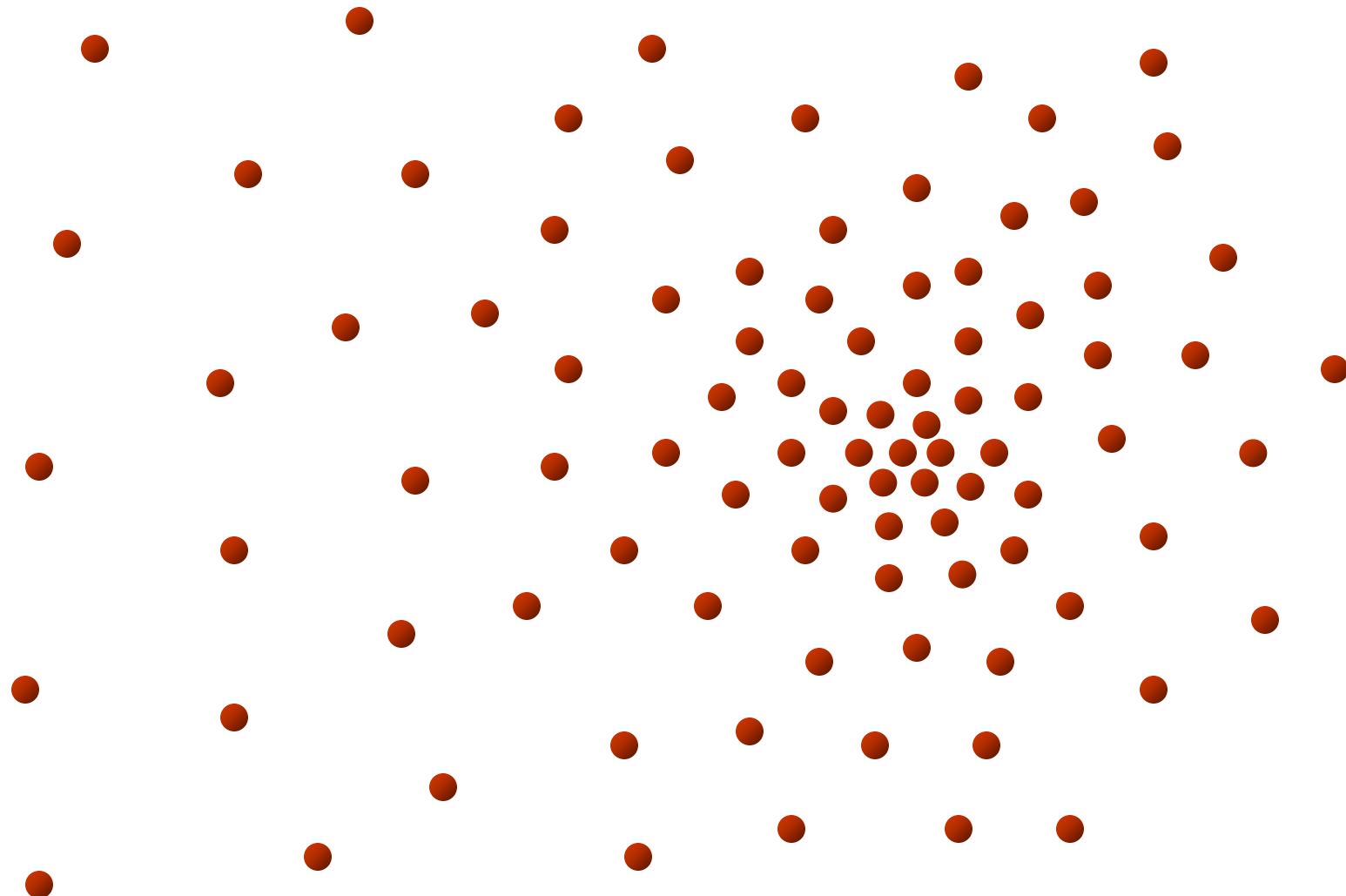


Mean-Shift Algorithm

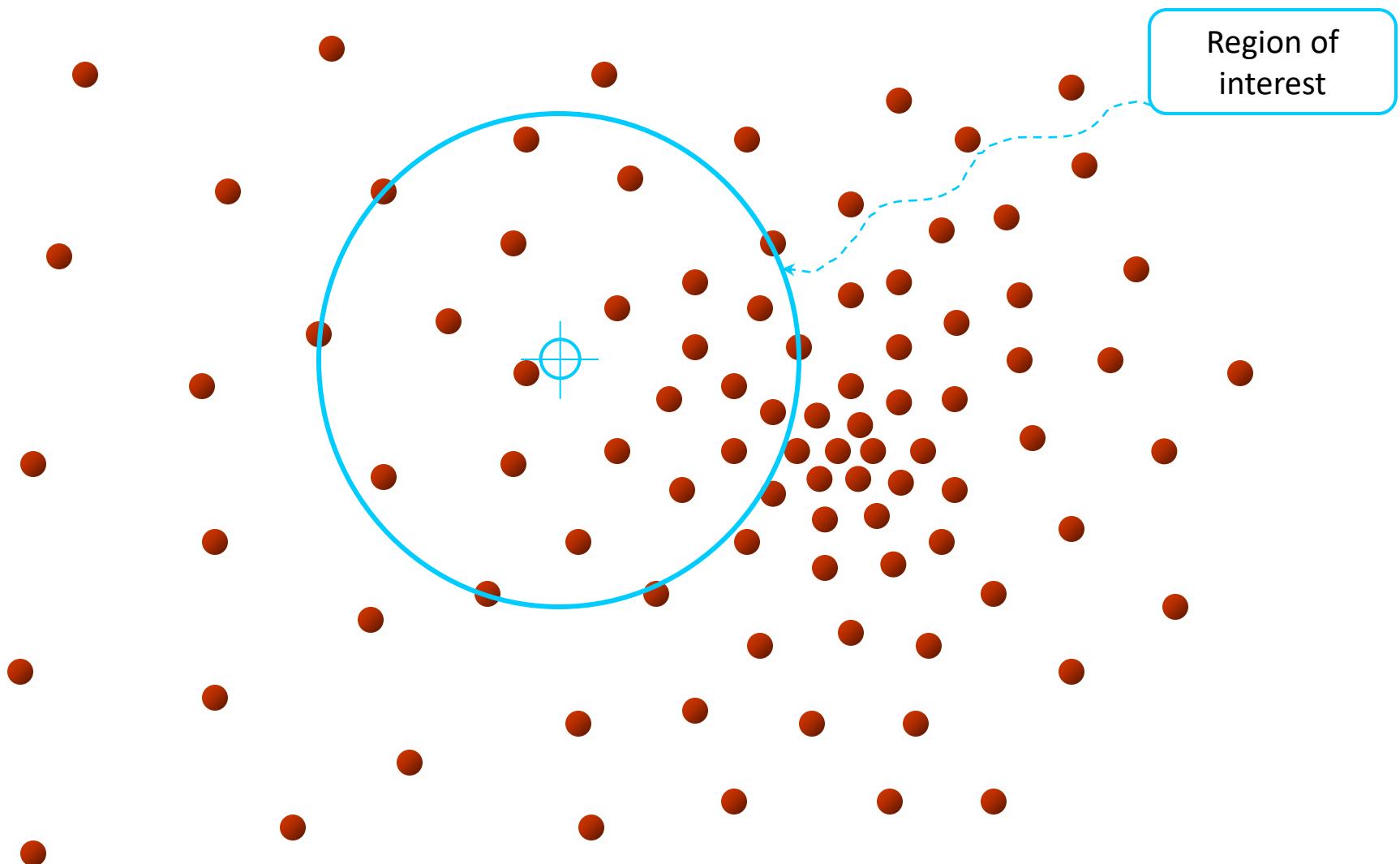
- Iterative Mode Search

1. Initialize random seed, and window W
2. Calculate center of mass (the “mean”) of W : $\sum_{x \in W} x H(x)$
3. Shift the search window to the mean
4. Repeat Step 2 until convergence

Mean shift

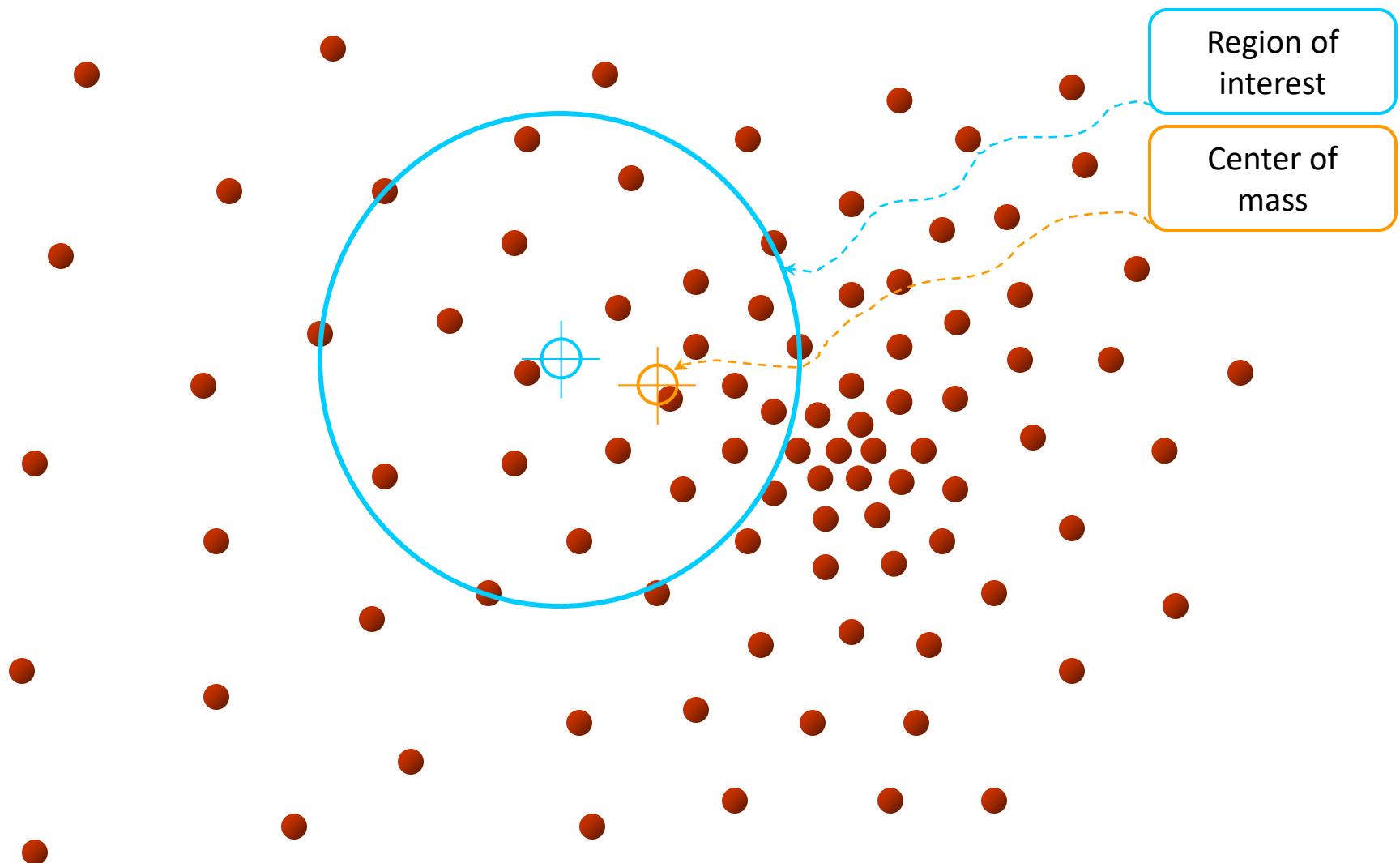


Mean shift

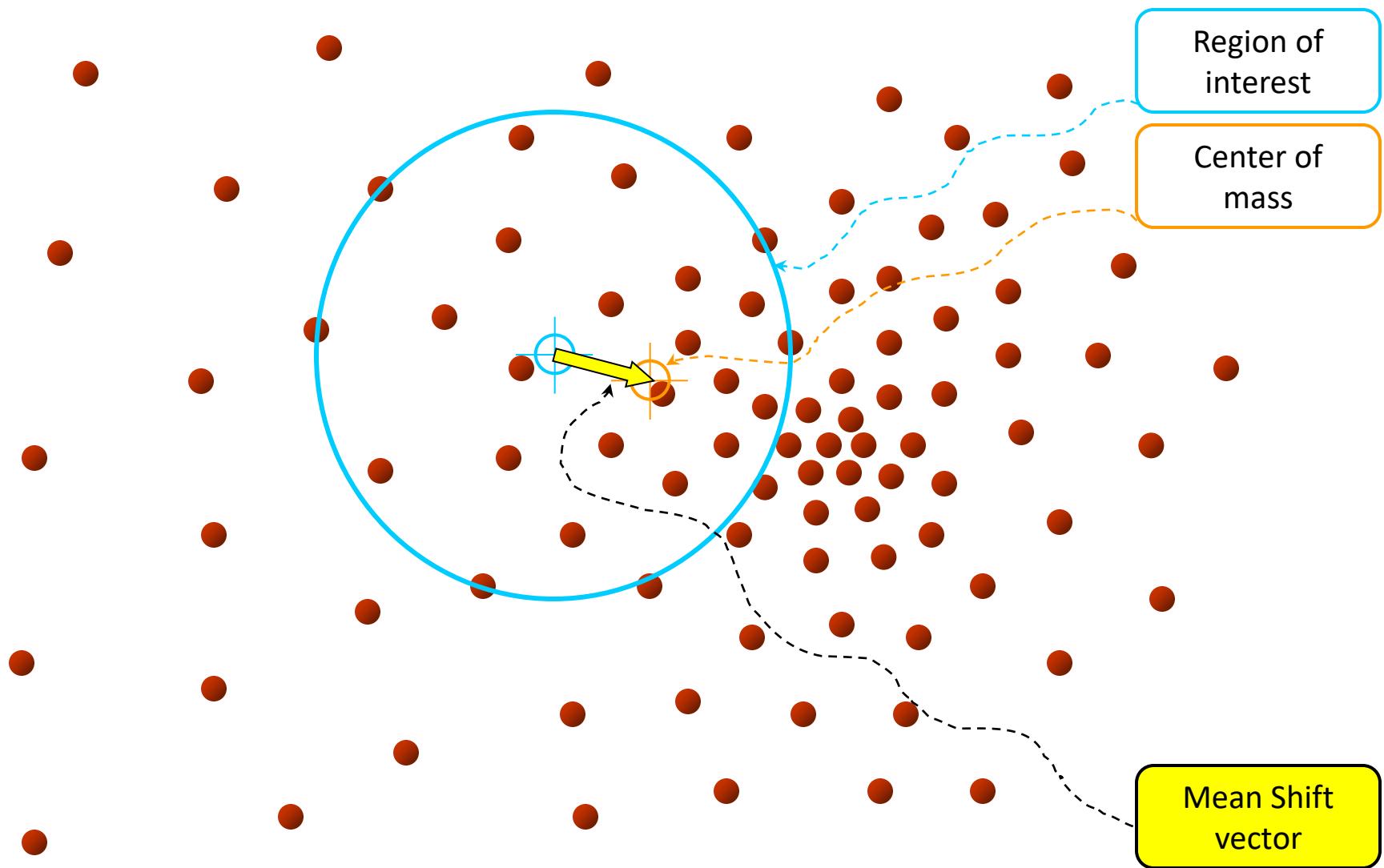


Region of
interest

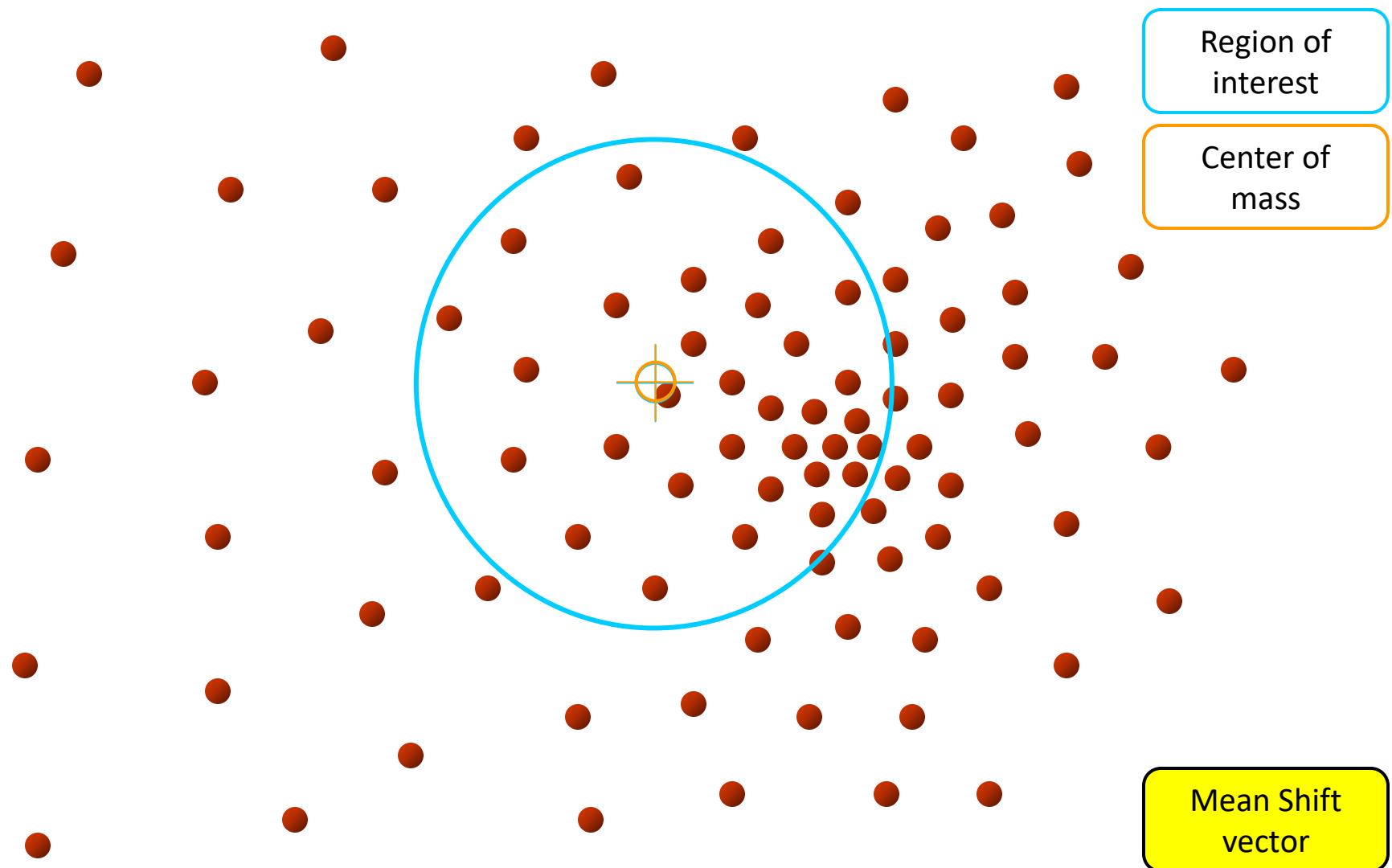
Mean shift



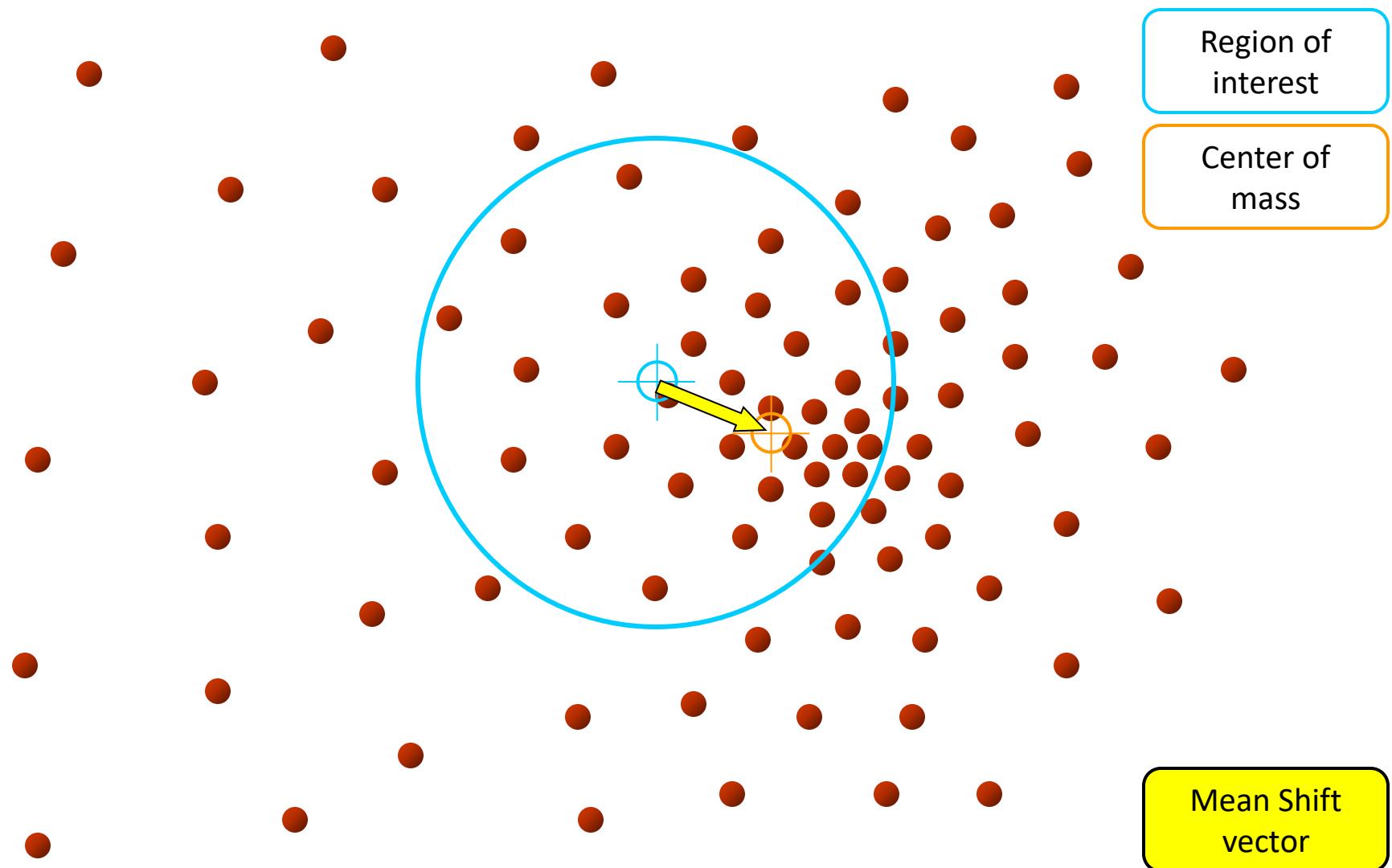
Mean shift



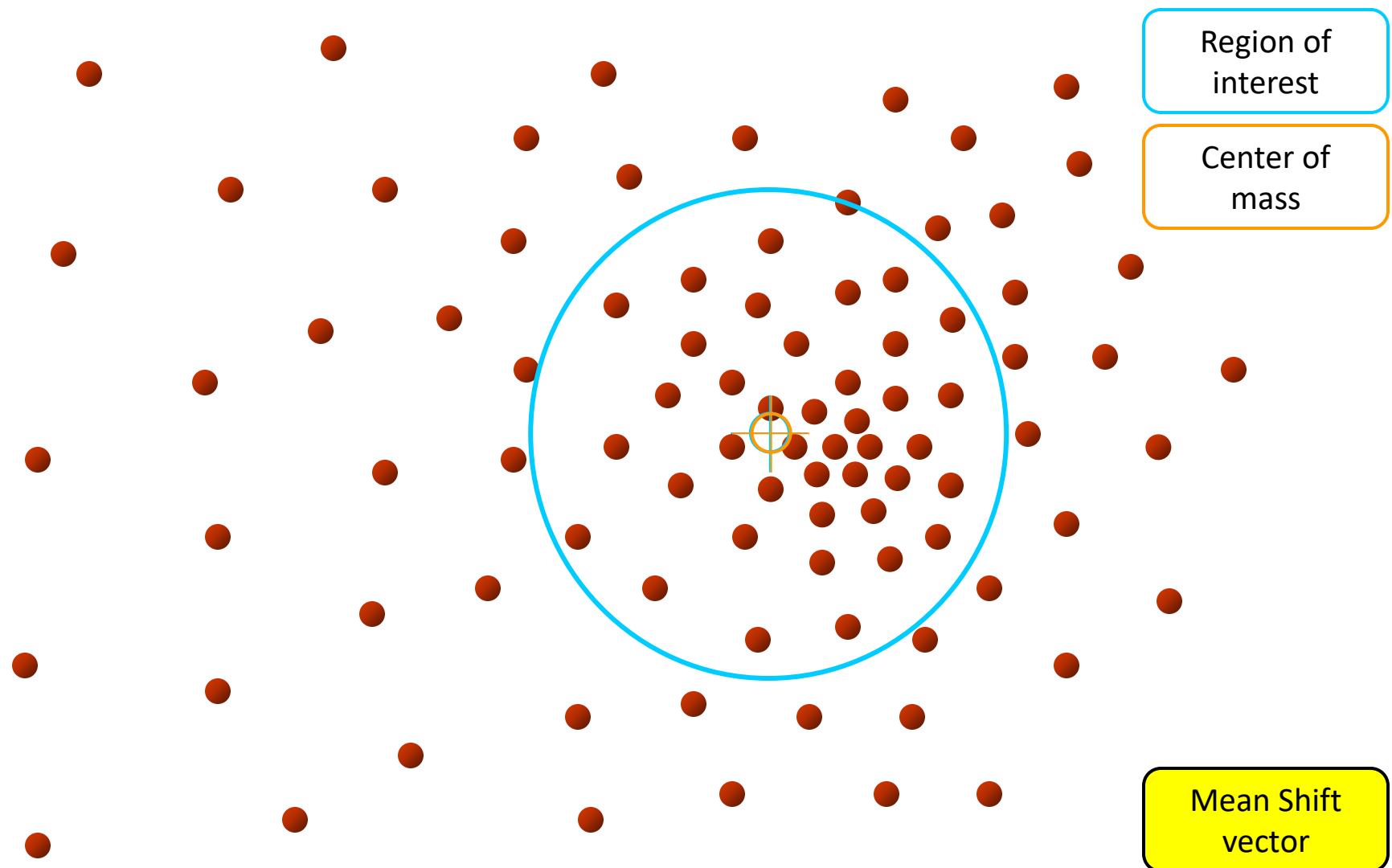
Mean shift



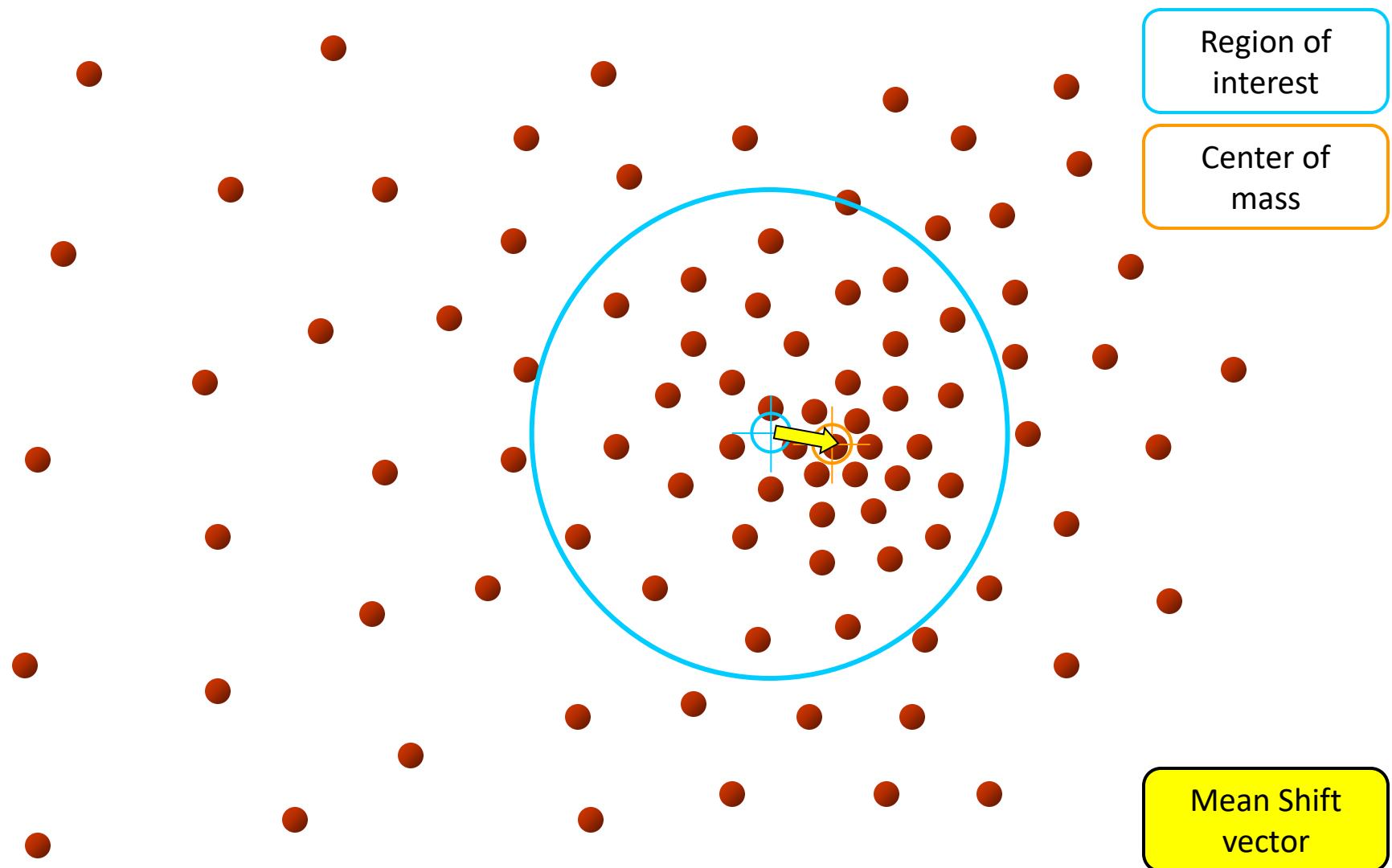
Mean shift



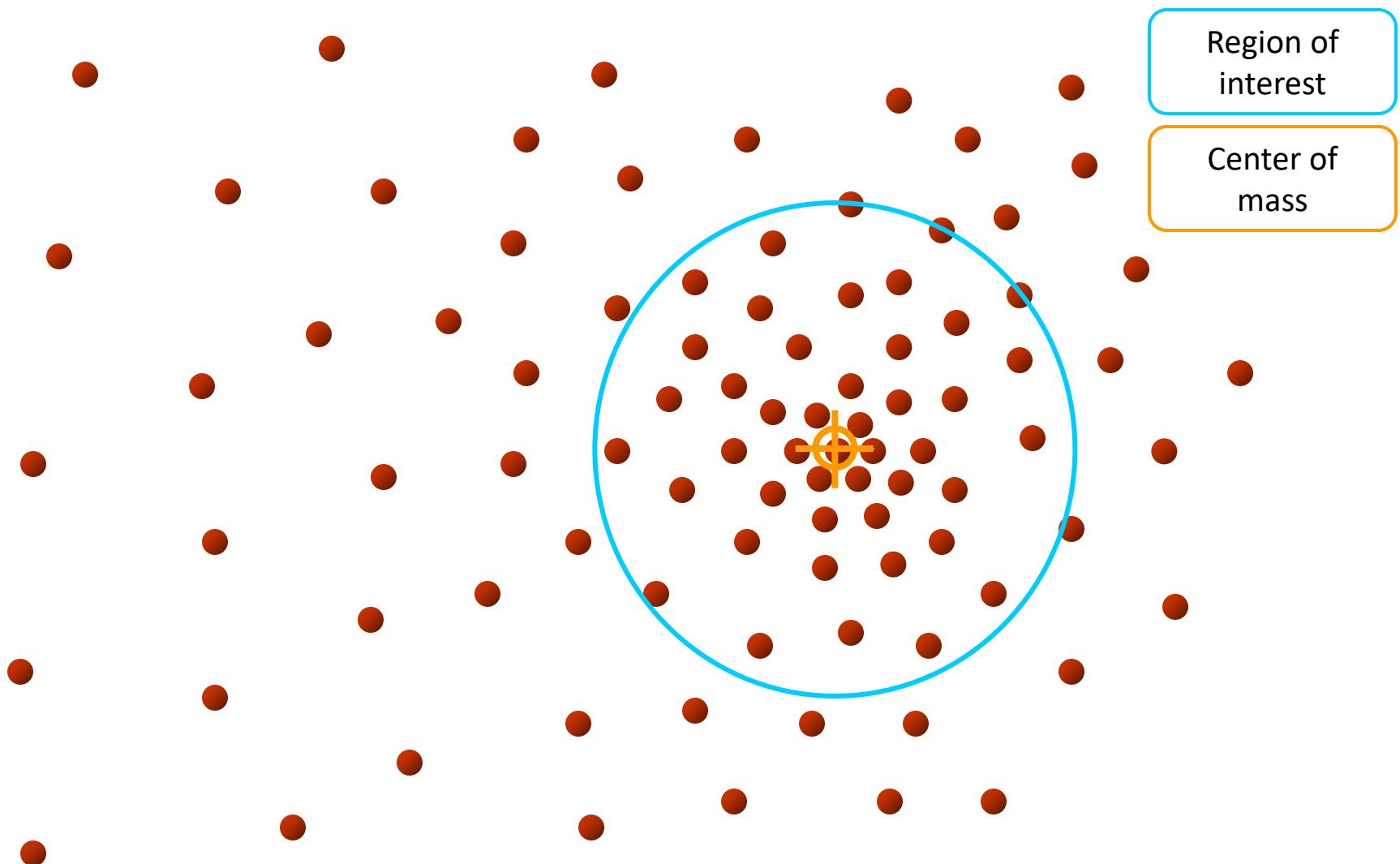
Mean shift



Mean shift



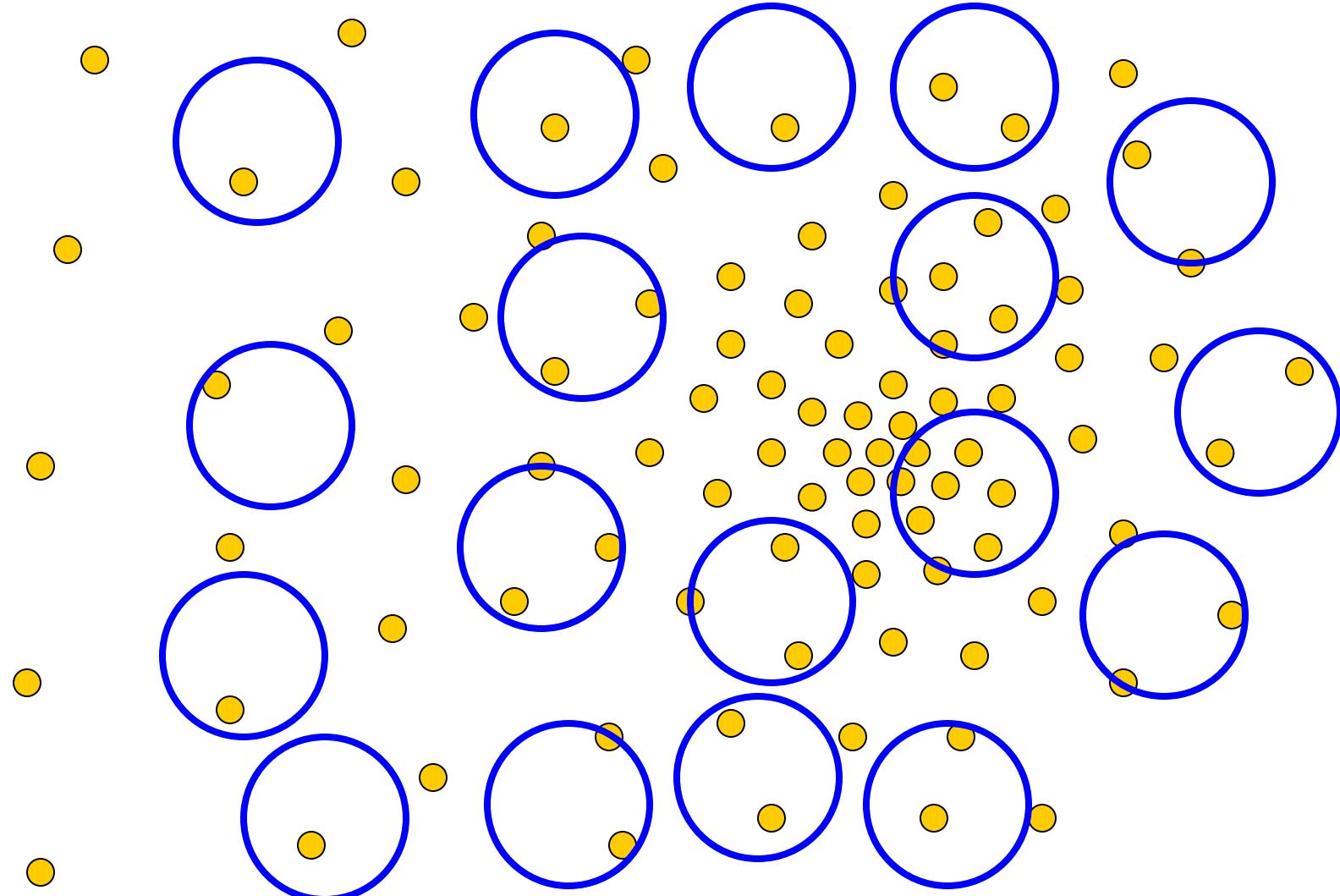
Mean shift



Region of
interest

Center of
mass

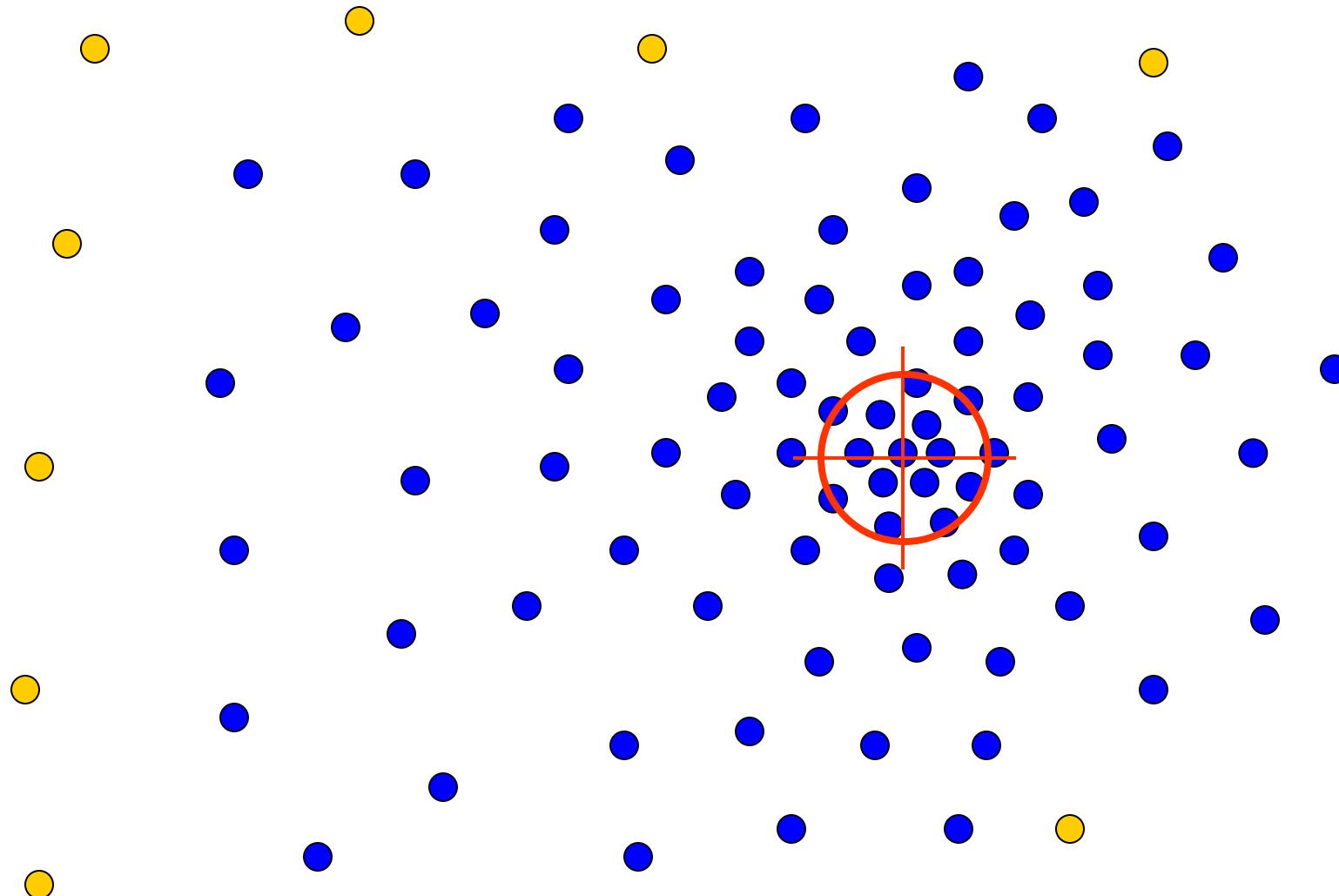
Real Modality Analysis



Tessellate the space with windows

Run the procedure in parallel

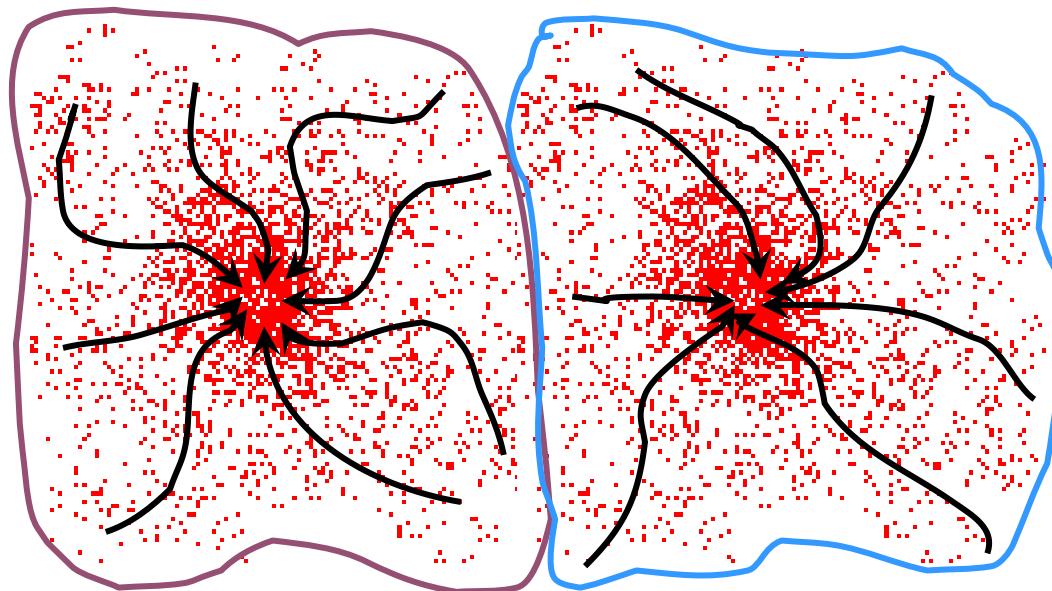
Real Modality Analysis



The **blue** data points were traversed by the windows towards the mode.

Attraction basin

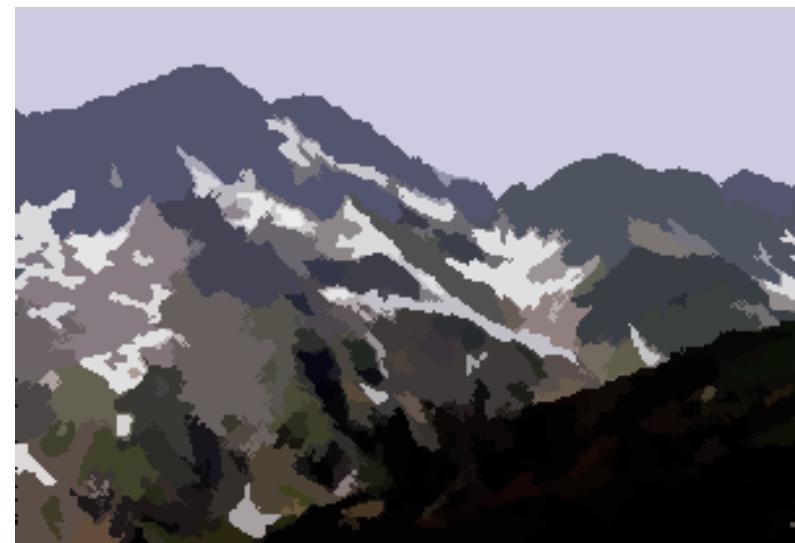
- **Attraction basin:** the region for which all points belong to the same mode
- **Cluster:** all data points in the attraction basin of a mode

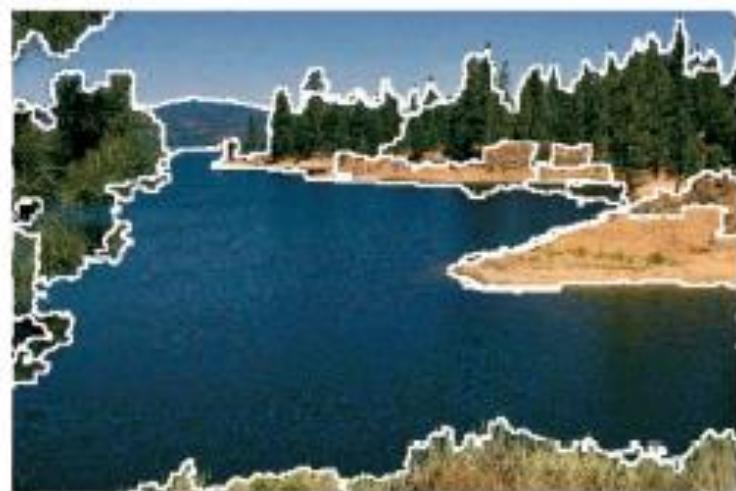


Segmentation by Mean Shift

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

Mean shift segmentation results





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

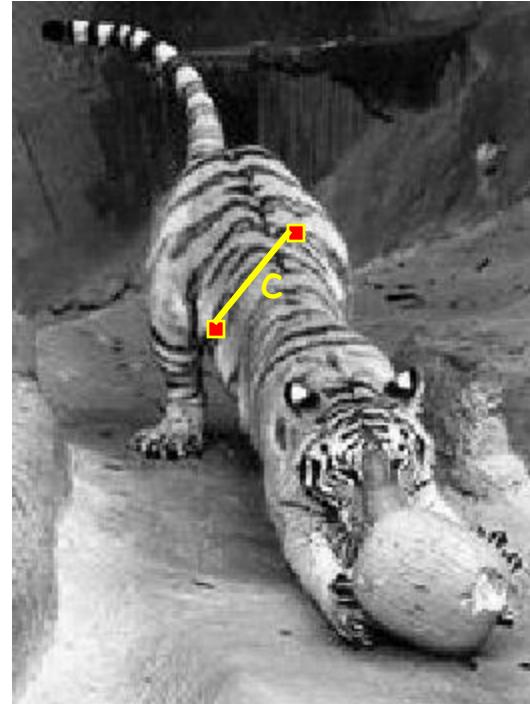
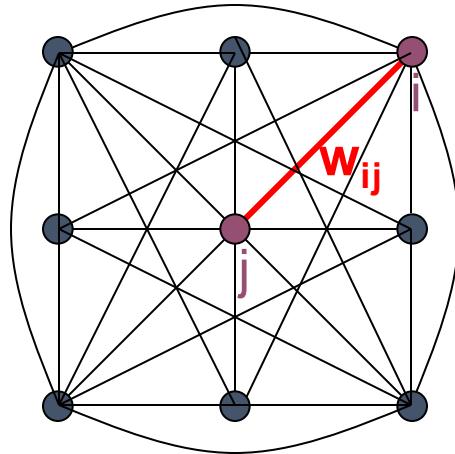
Mean shift pros and cons

- Pros
 - Good general-purpose segmentation
 - Flexible in number and shape of regions
 - Robust to outliers
- Cons
 - Have to choose kernel size in advance
 - Not suitable for high-dimensional features
- When to use it
 - Multiple segmentations
 - Tracking, clustering, filtering applications
 - Comaniciu et al., [Real-Time Tracking of Non-Rigid Objects using Mean Shift](#), CVPR 2000. Best Paper Award

Graph-based Segmentation

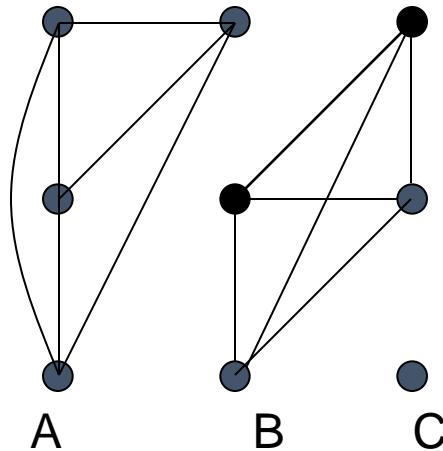
- Min-cut
- Normalized-cut

Images as graphs



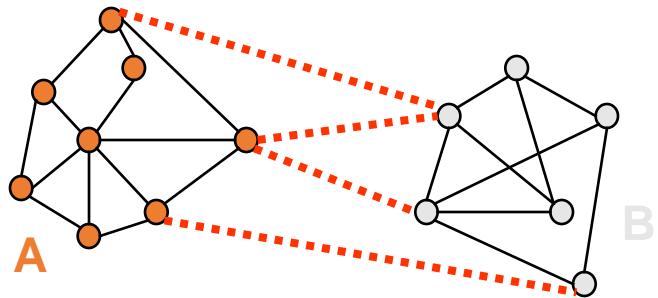
- *Fully-connected* graph
 - node for every pixel
 - link between *every* pair of pixels, \mathbf{p}, \mathbf{q}
 - similarity w_{ij} for each link

Segmentation by Graph Cuts



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

Cuts in a graph

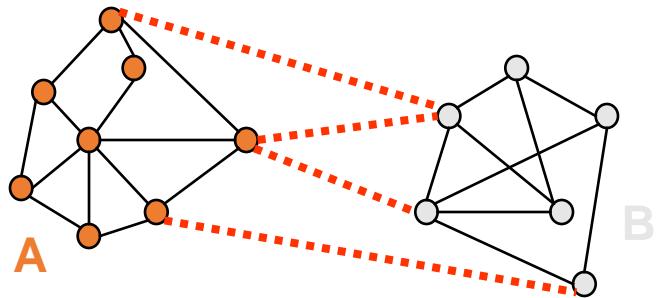


- **Link Cut**

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

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

Cuts in a graph

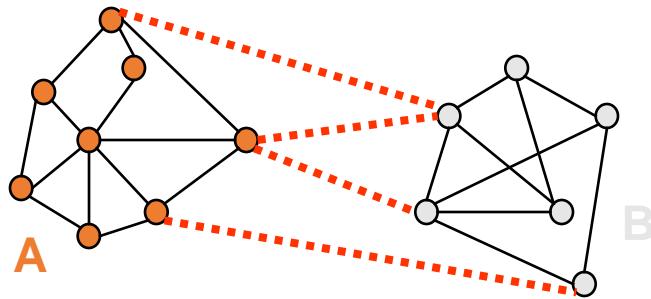


- Link Cut
 - set of links whose removal makes a graph disconnected
 - cost of a cut:

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

How to find the set of links to remove?

Cuts in a graph



- **Link Cut**

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

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

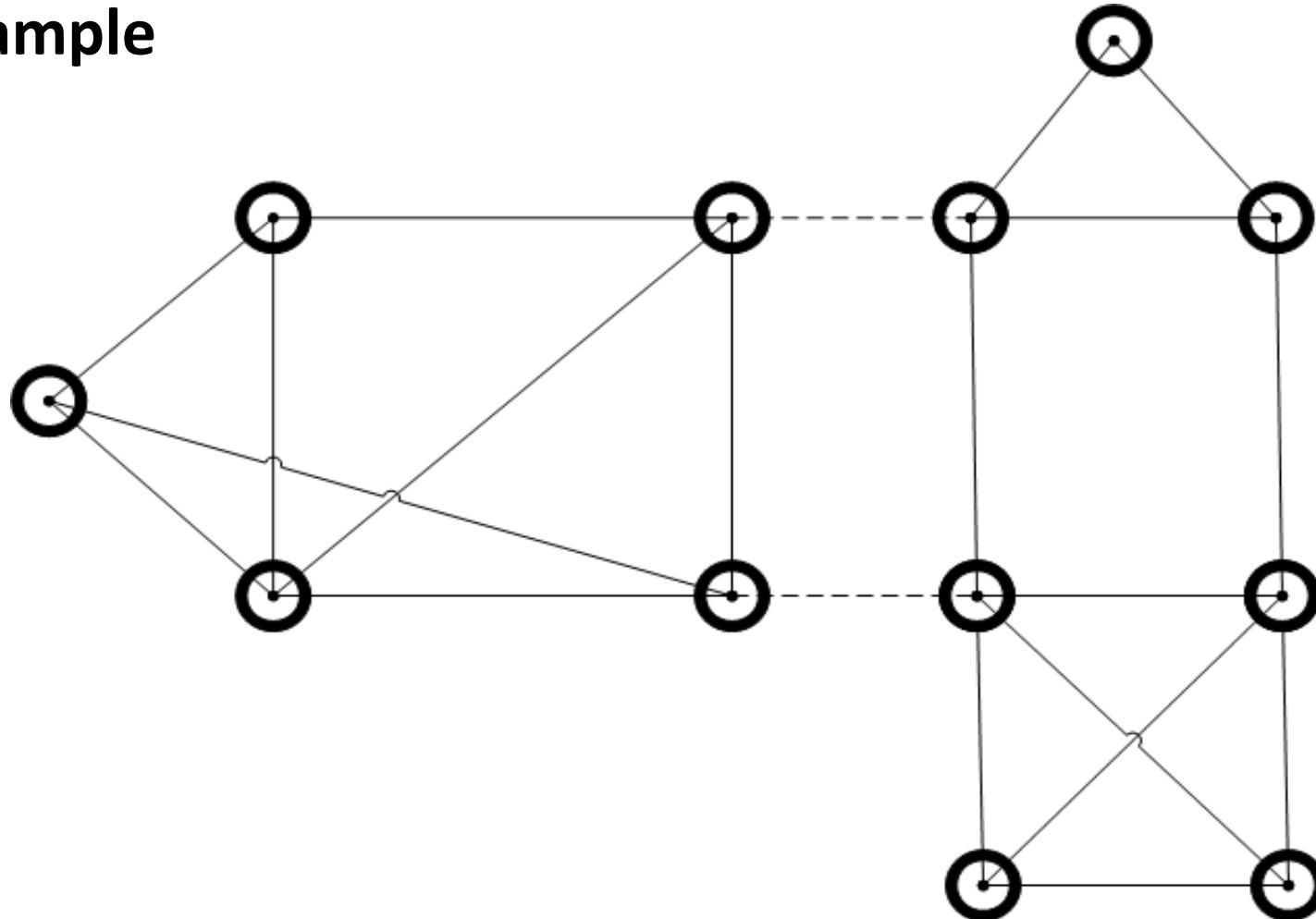
How to find the set of links to remove?

One idea: Find minimum cut

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

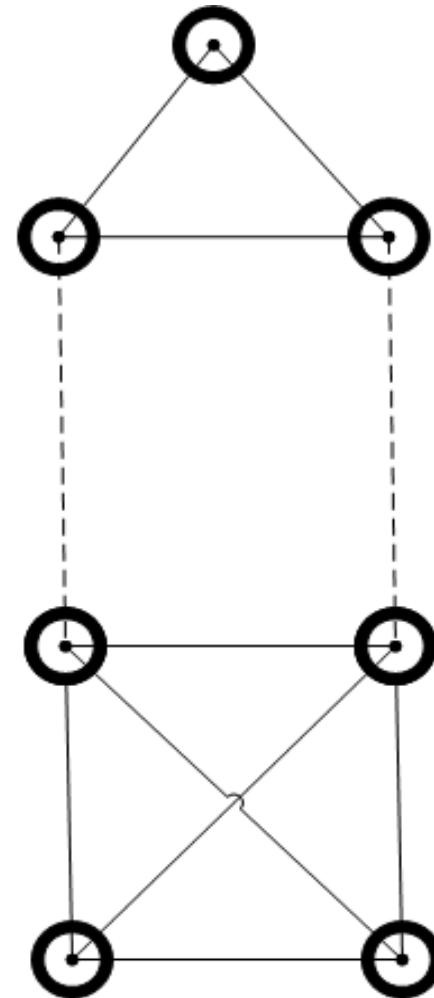
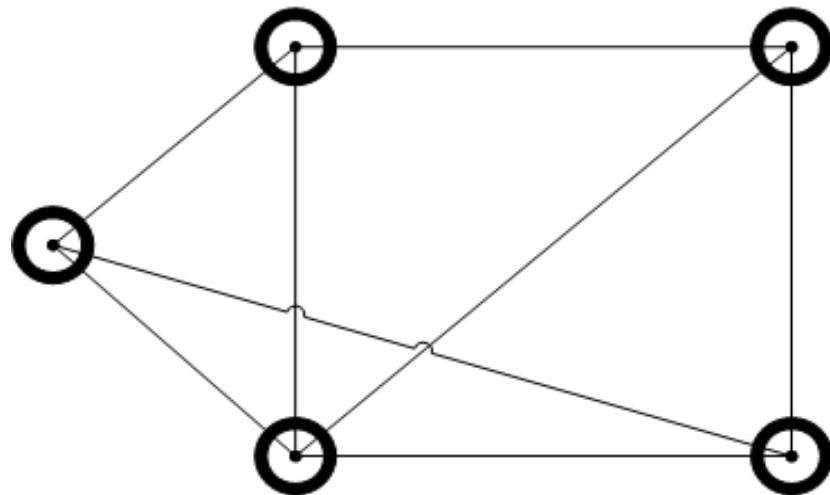
Min Cut method

Example



Min Cut method

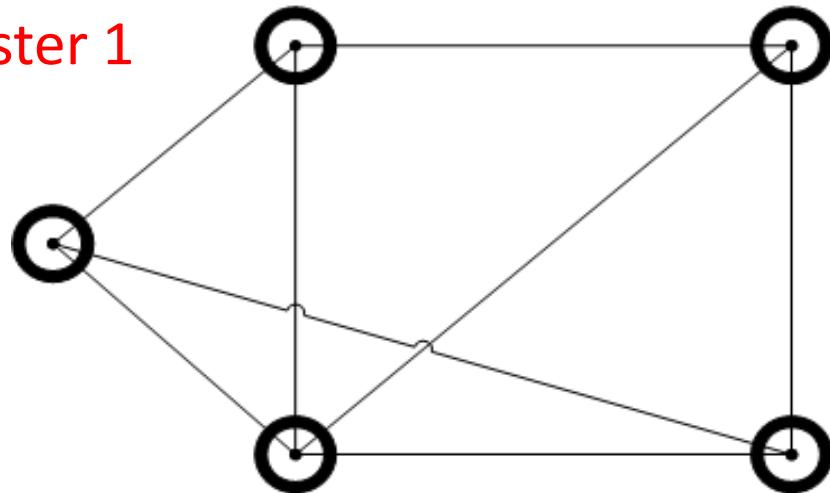
Example Continued



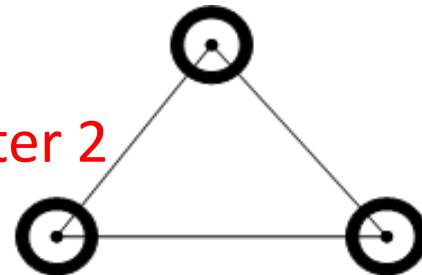
Min Cut method

Example Continued

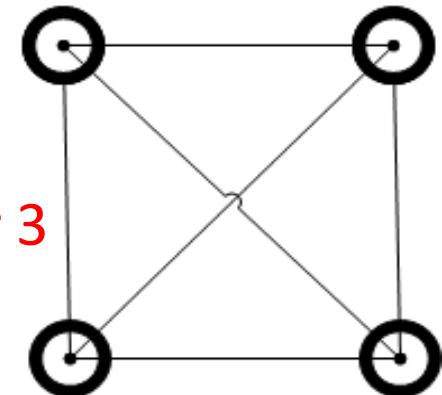
Cluster 1



Cluster 2

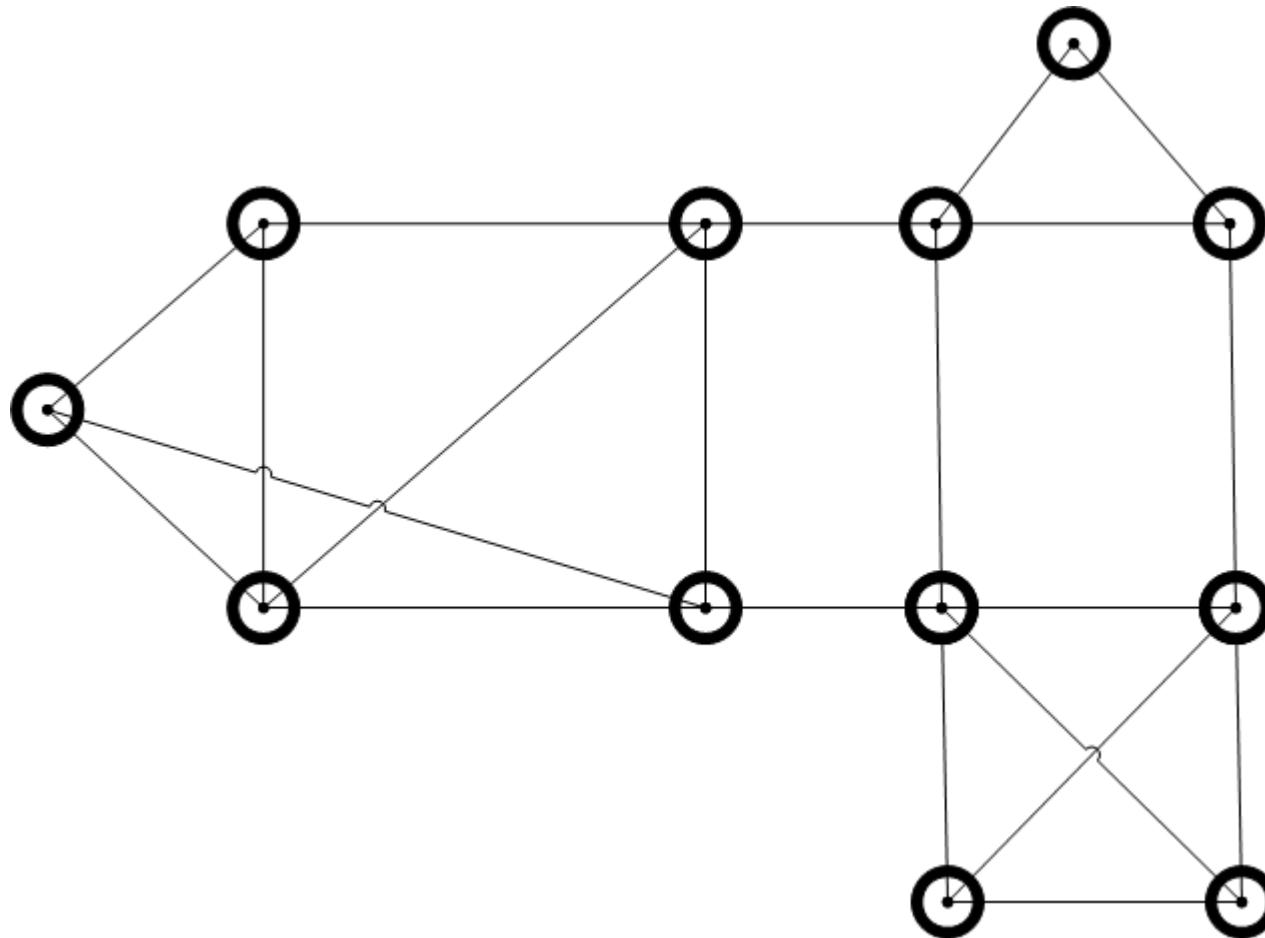


Cluster 3



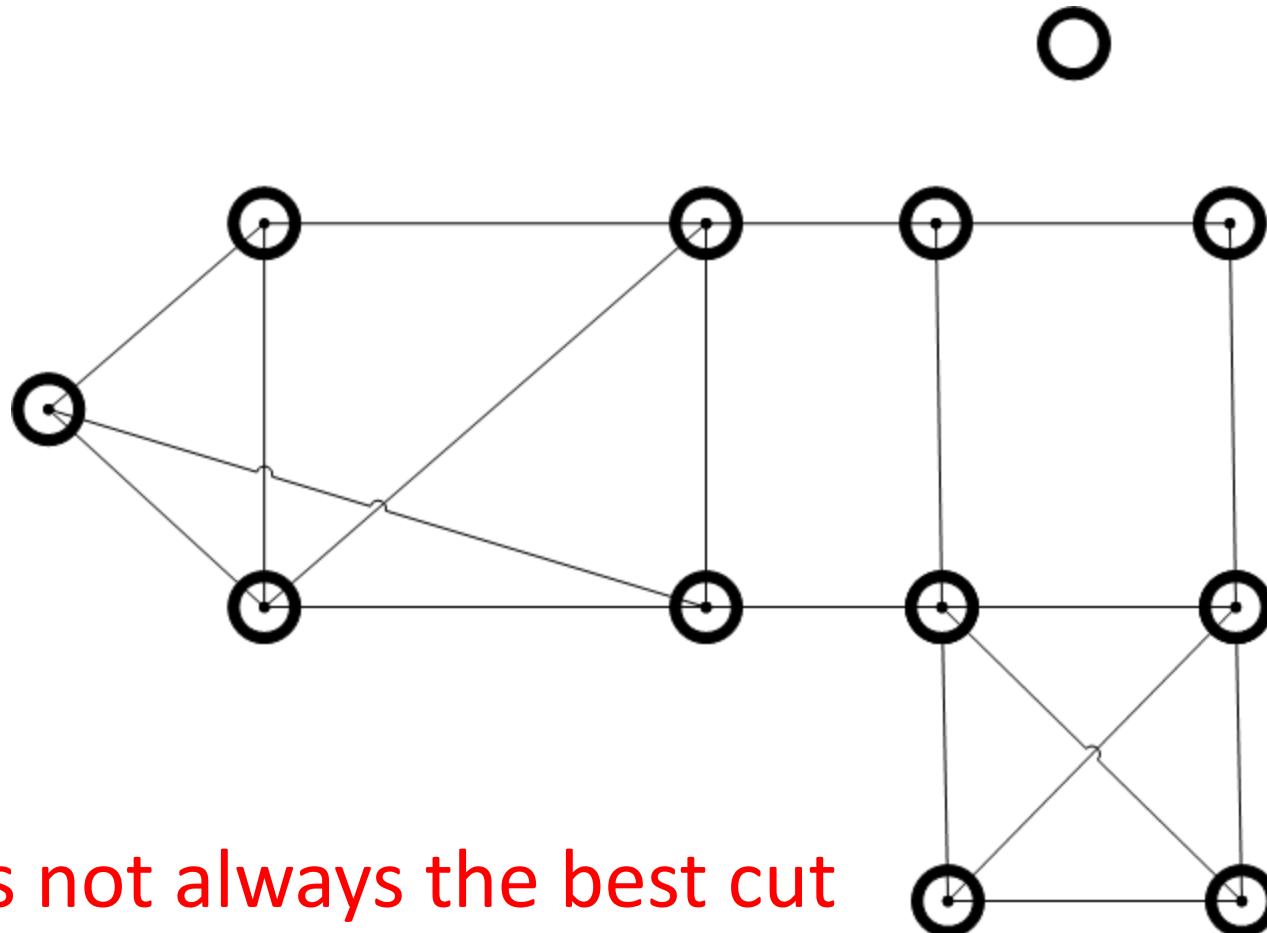
Min Cut method – main problem

Example



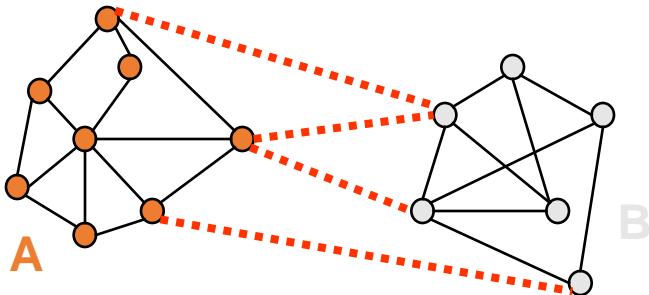
Min Cut method – main problem

Example – Another possible cut



Min cut is not always the best cut

Cuts in a graph



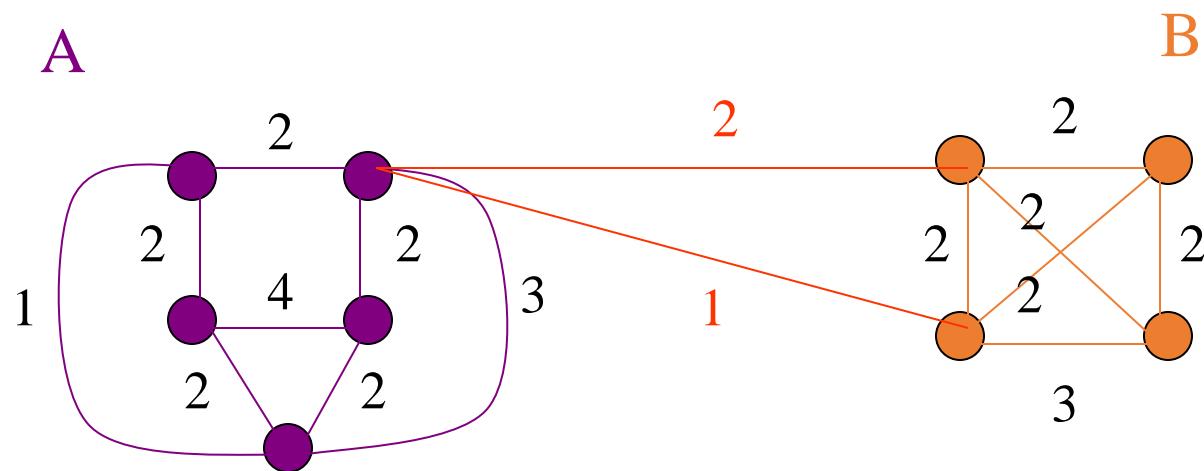
Normalized Cut

- a cut penalizes small segments
- fix by normalizing for size of segments

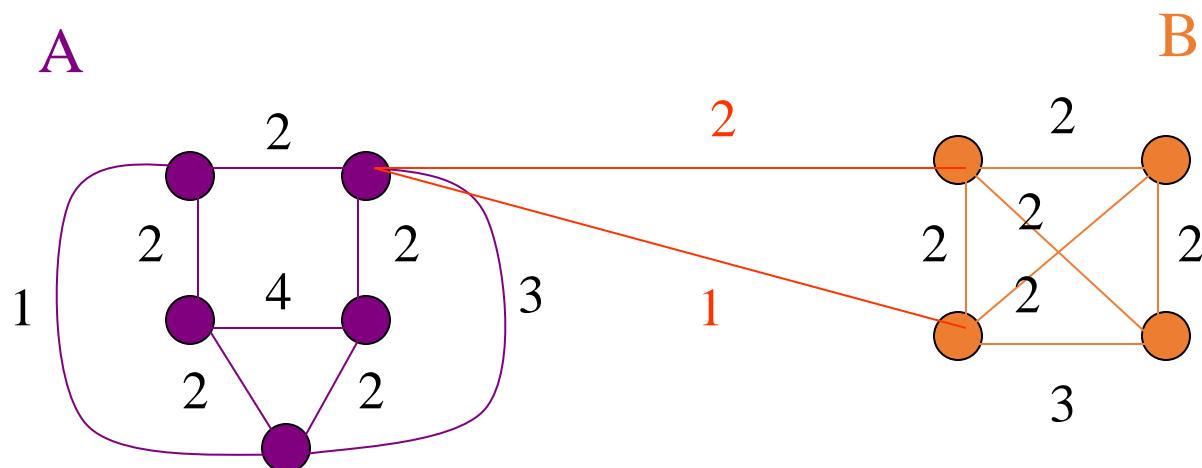
$$Ncut(A, B) = \frac{cut(A, B)}{volume(A)} + \frac{cut(A, B)}{volume(B)}$$

- $volume(A)$ = sum of costs of all edges that touch A

Example Normalized Cut



Example Normalized Cut



$$\text{Ncut}(A,B) = \frac{3}{21} + \frac{3}{16}$$

Normalized cuts results



Normalized cuts: Pro and con

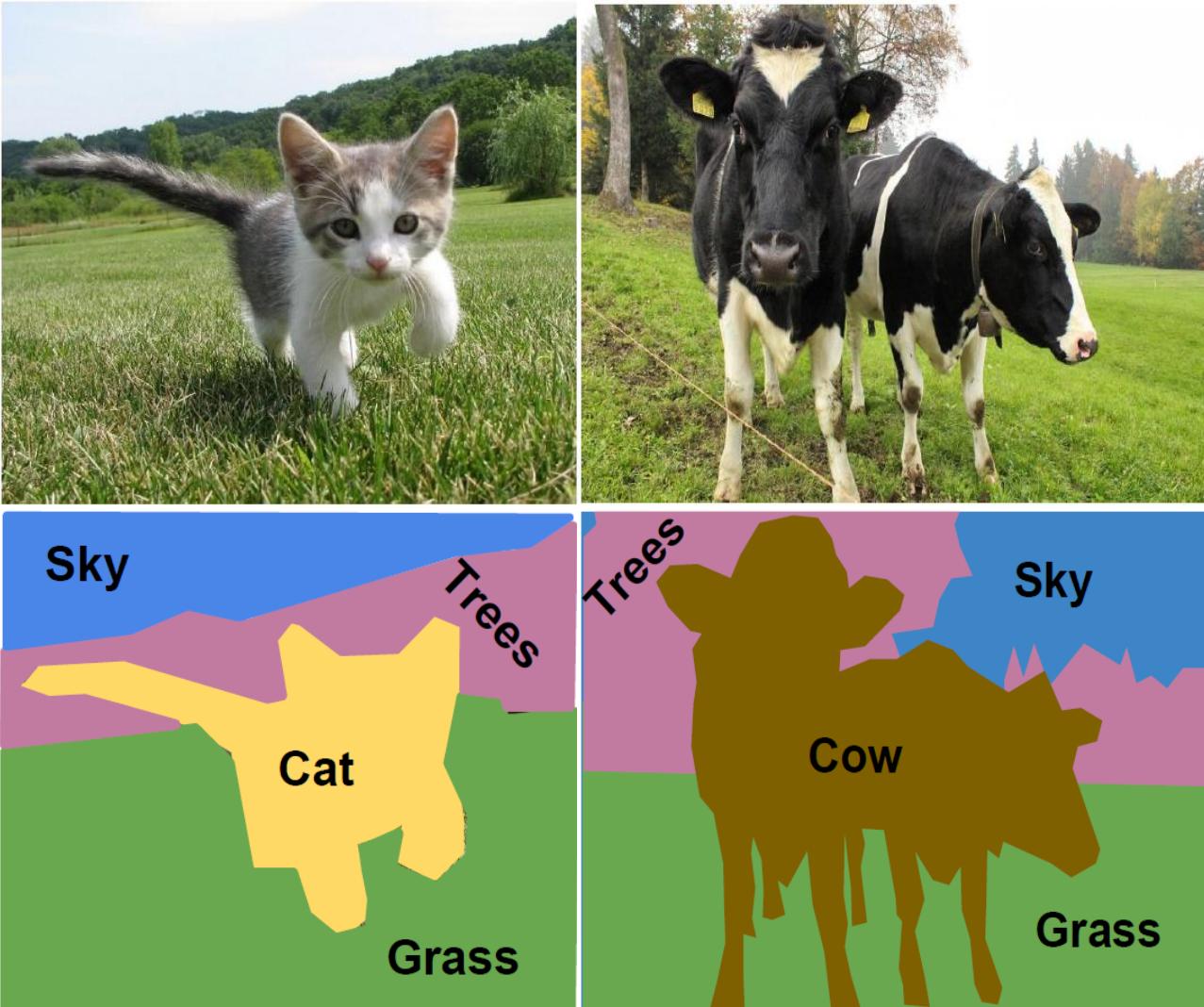
- Pros
 - Generic framework, can be used with many different features and affinity formulations
 - Provides regular segments
- Cons
 - Need to chose number of segments
 - High storage requirement and time complexity

Semantic Segmentation

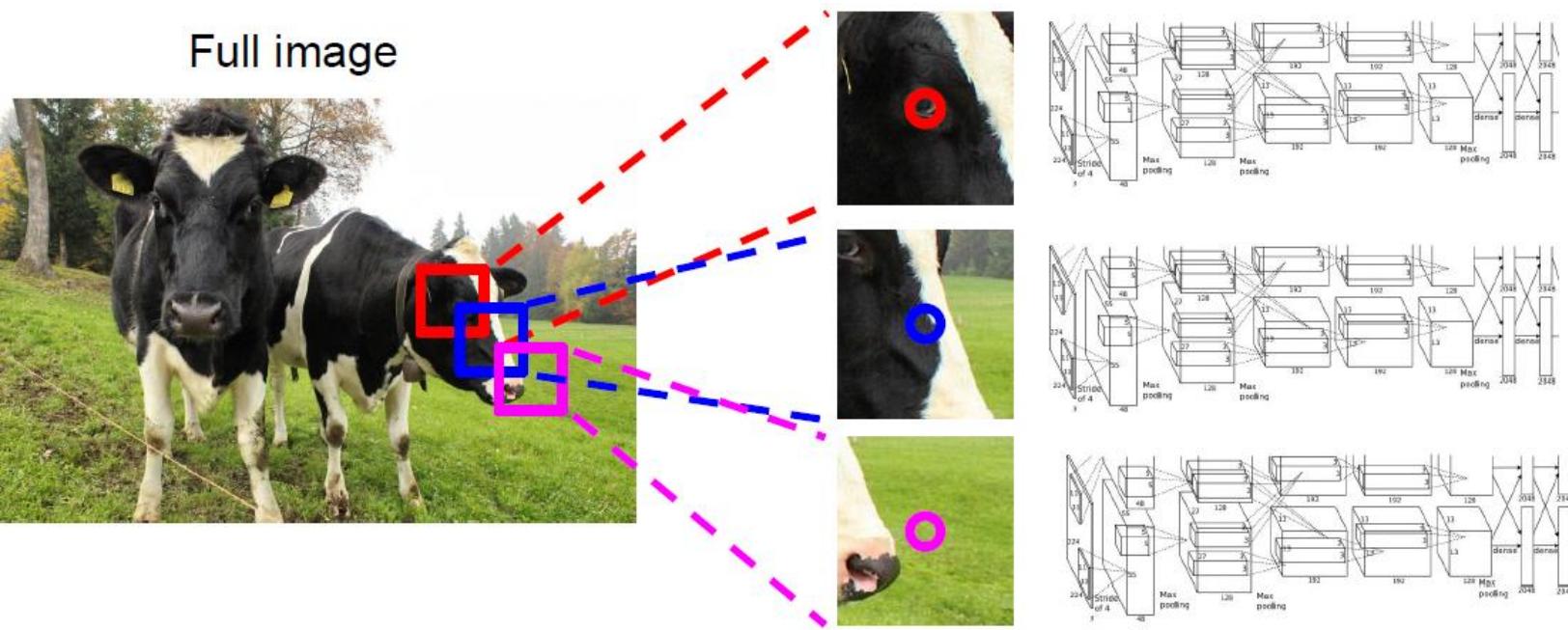
These images are CC0 public domain [source1](#) [source2](#)

Label each pixel in the image with a category label

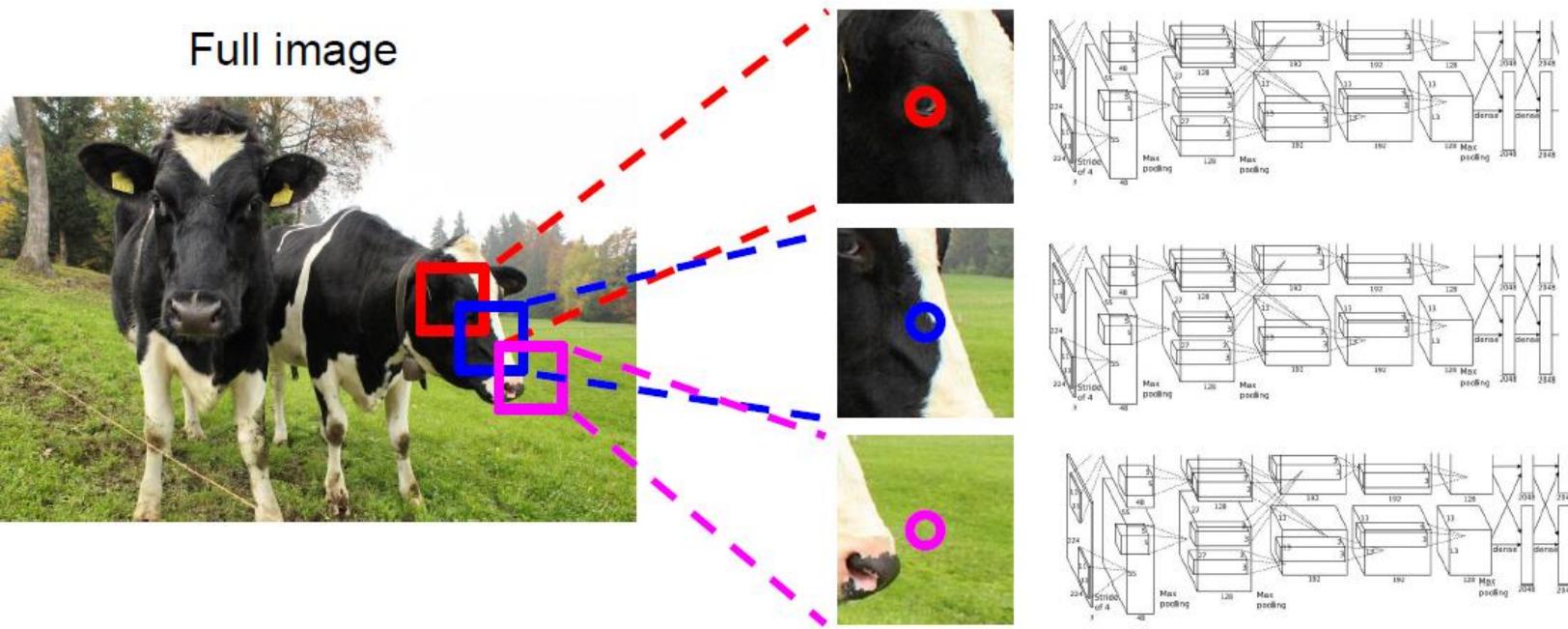
Don't differentiate instances, only care about pixels



Semantic Segmentation: Sliding Window



Semantic Segmentation: Sliding Window



Problem:

Very inefficient!

Not reusing shared features between overlapping patches

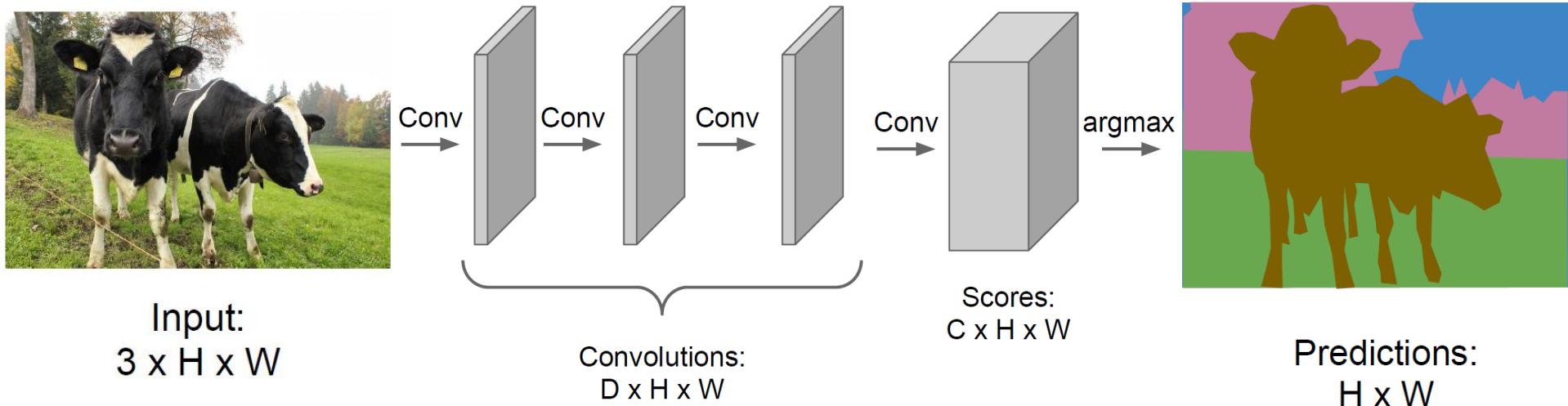
Cow

Cow

Grass

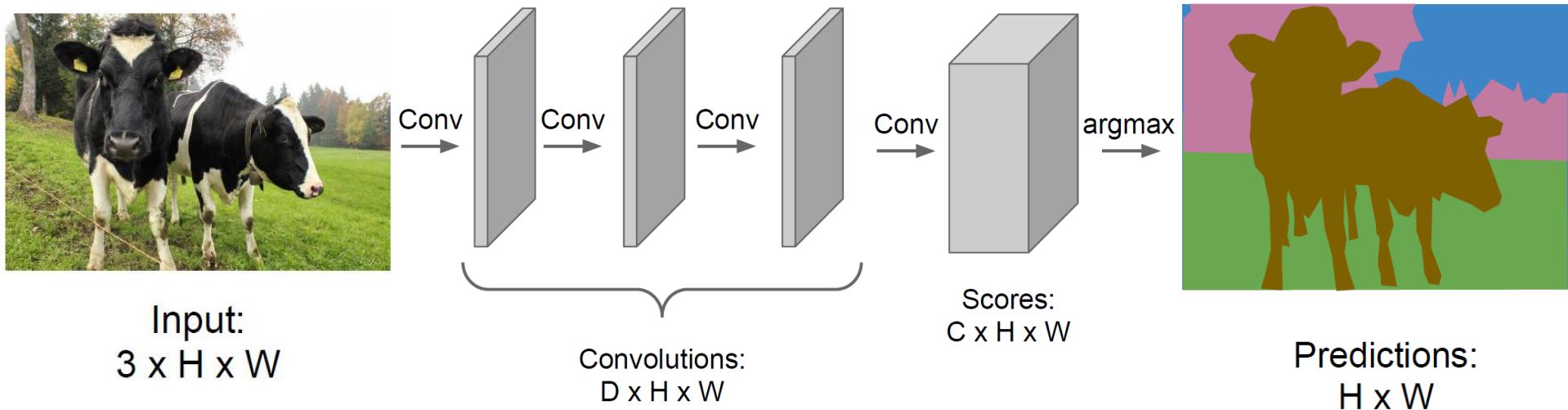
Semantic Segmentation: Fully Convolutional

Design a network as a bunch of convolutional layers
to make predictions for pixels all at once!



Semantic Segmentation: Fully Convolutional

Design a network as a bunch of convolutional layers
to make predictions for pixels all at once!



Problem:

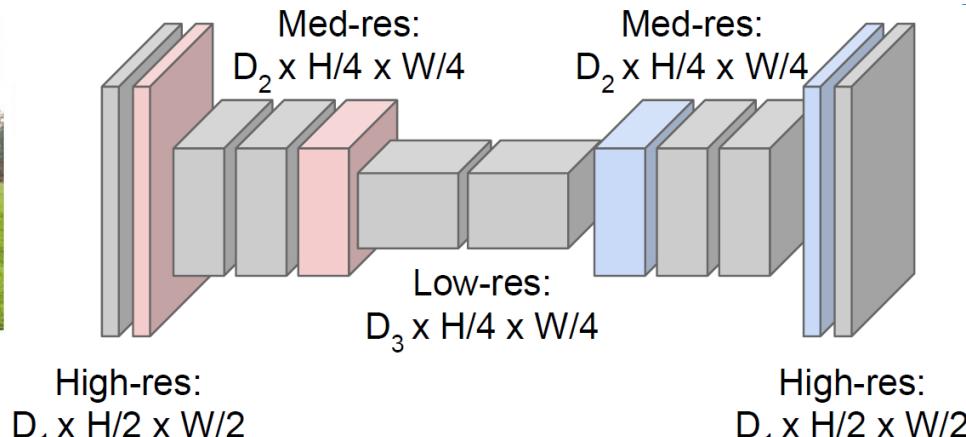
Convolutions at original image resolution
will be very expensive ...

Semantic Segmentation: Fully Convolutional

Design network as a bunch of convolutional layers, with **downsampling** and **upsampling** inside the network!



Input:
 $3 \times H \times W$



Predictions:
 $H \times W$

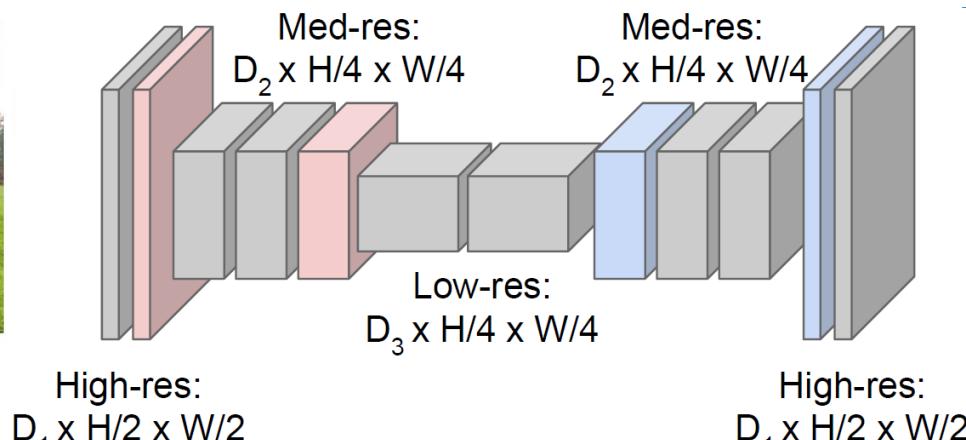
Semantic Segmentation: Fully Convolutional

Downsampling:
Pooling, strided
convolution



Input:
 $3 \times H \times W$

Design network as a bunch of convolutional layers, with **downsampling** and **upsampling** inside the network!



Upsampling:
??



Predictions:
 $H \times W$

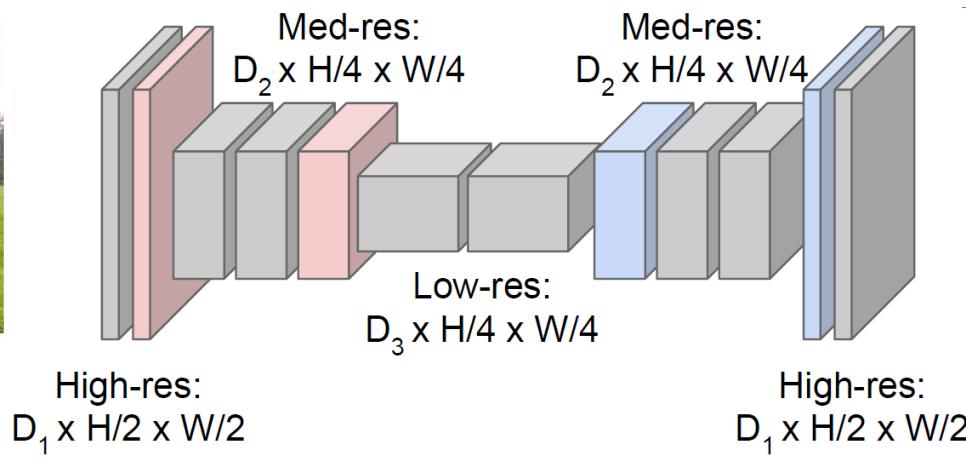
Semantic Segmentation: Fully Convolutional

Downsampling:
Pooling, strided
convolution



Input:
 $3 \times H \times W$

Design network as a bunch of convolutional layers, with **downsampling** and **upsampling** inside the network!



Upsampling:
Unpooling or
strided transpose
convolution



Predictions:
 $H \times W$

Upsampling: Unpooling

Nearest Neighbor

1	2
3	4



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

Input: 2×2

Output: 4×4

Upsampling: Unpooling

Nearest Neighbor

1	2
3	4



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

Input: 2 x 2

Output: 4 x 4

“Bed of Nails”

1	2
3	4



1	0	2	0
0	0	0	0
3	0	4	0
0	0	0	0

Input: 2 x 2

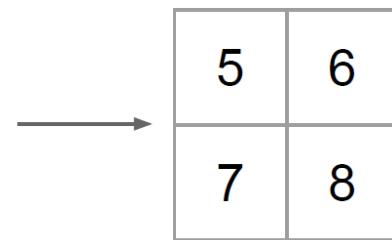
Output: 4 x 4

Upsampling: Max Unpooling

Max Pooling

Remember which element was max!

1	2	6	3
3	5	2	1
1	2	2	1
7	3	4	8



5	6
7	8

Input: 4 x 4

Output: 2 x 2

Upsampling: Max Unpooling

Max Pooling

Remember which element was max!

1	2	6	3
3	5	2	1
1	2	2	1
7	3	4	8

Input: 4 x 4



5	6
7	8

Output: 2 x 2

Max Unpooling

Use positions from pooling layer

1	2
3	4

Input: 2 x 2



0	0	2	0
0	1	0	0
0	0	0	0
3	0	0	4

Output: 4 x 4

Upsampling: Max Unpooling

Max Pooling

Remember which element was max!

1	2	6	3
3	5	2	1
1	2	2	1
7	3	4	8

Input: 4 x 4

5	6
7	8

Output: 2 x 2

Max Unpooling

Use positions from pooling layer

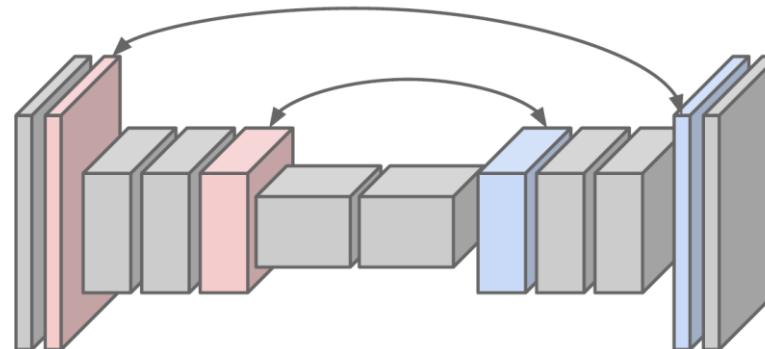
1	2
3	4

Input: 2 x 2

0	0	2	0
0	1	0	0
0	0	0	0
3	0	0	4

Output: 4 x 4

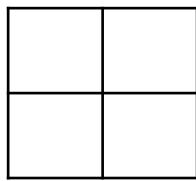
Corresponding pairs of
downsampling and
upsampling layers



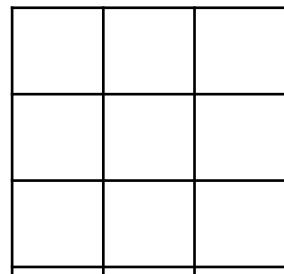
Slide credit: cs231n

Upsampling: Transpose Convolution

3 x 3 transpose convolution, stride 2 pad 1



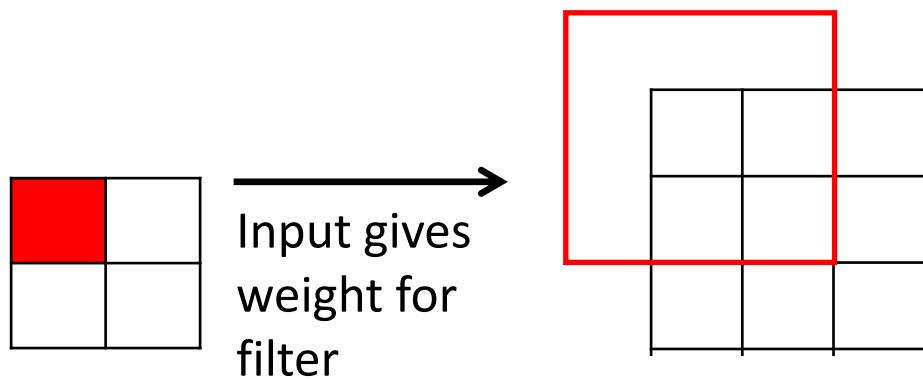
Input: 2 x 2



Output: 3 x 3

Upsampling: Transpose Convolution

3 x 3 transpose convolution, stride 2 pad 1

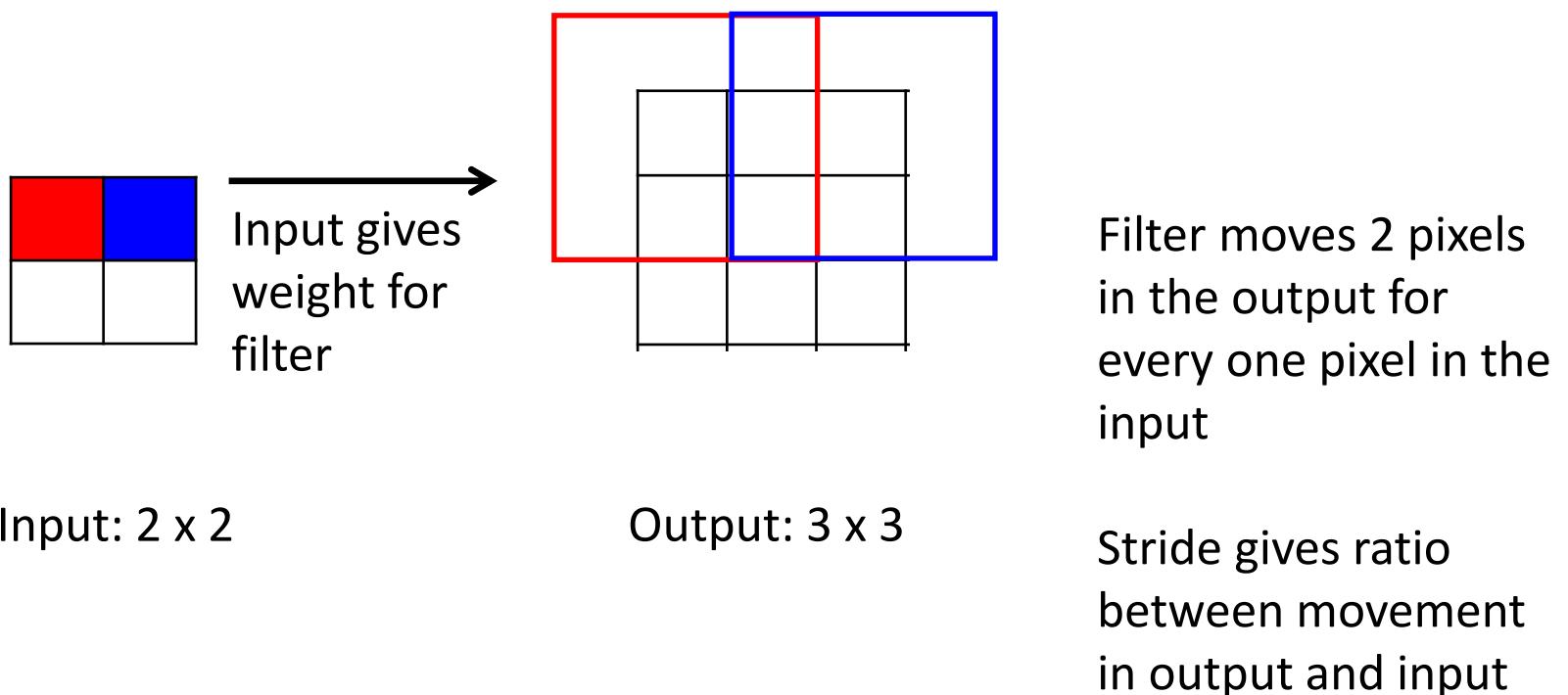


Input: 2 x 2

Output: 3 x 3

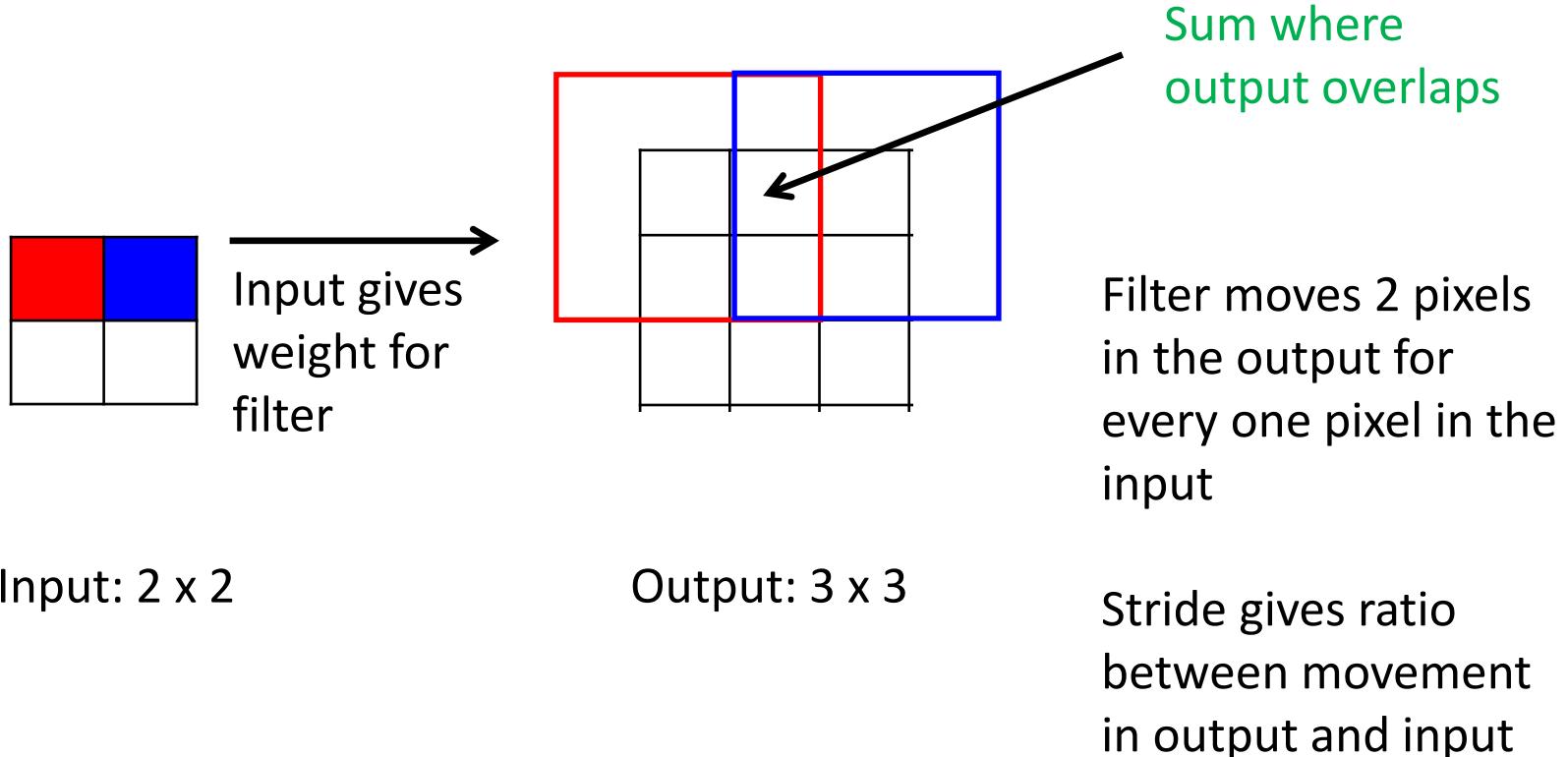
Upsampling: Transpose Convolution

3 x 3 transpose convolution, stride 2 pad 1



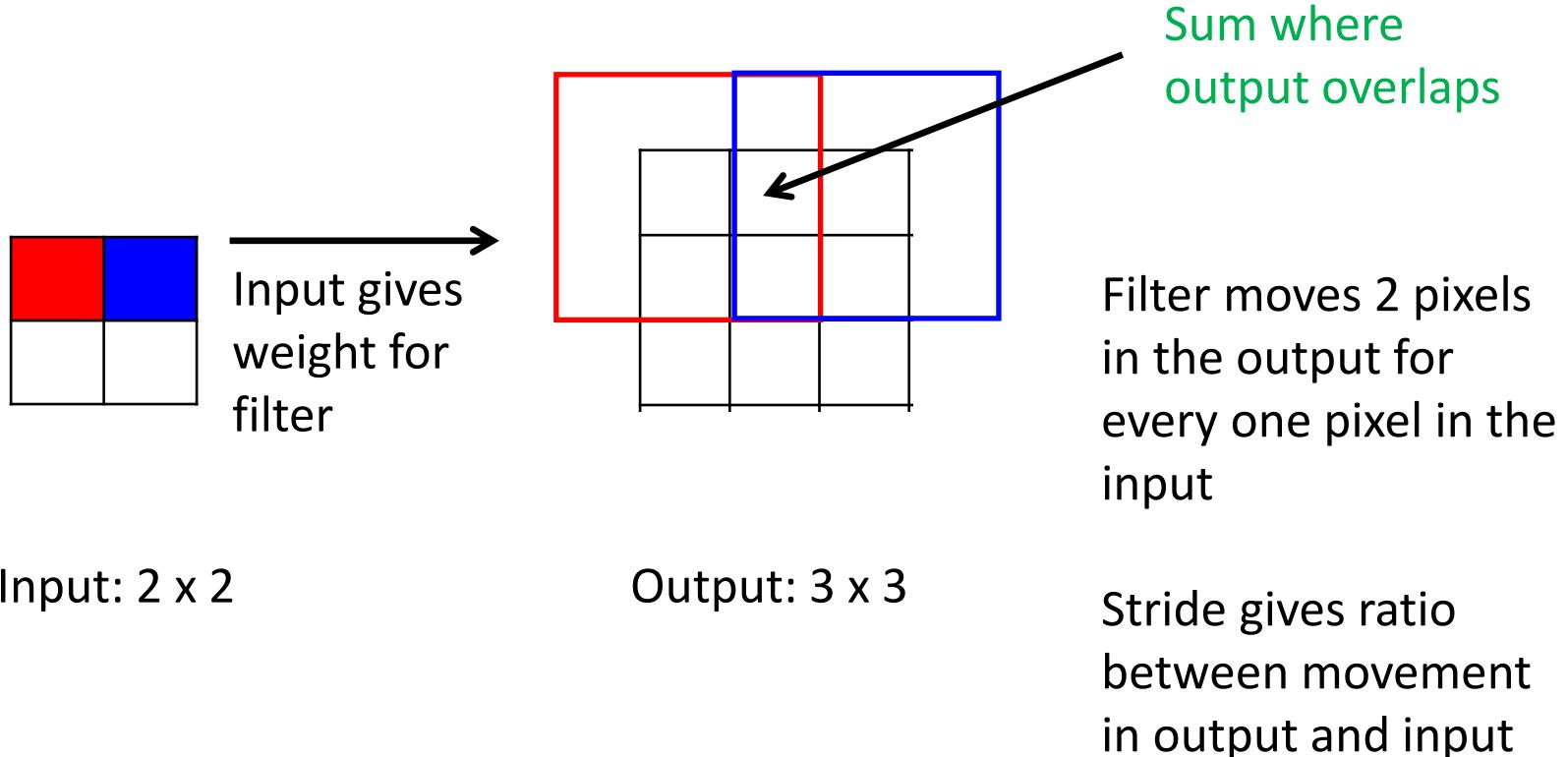
Upsampling: Transpose Convolution

3 x 3 transpose convolution, stride 2 pad 1



Upsampling: Transpose Convolution

3 x 3 transpose convolution, stride 2 pad 1

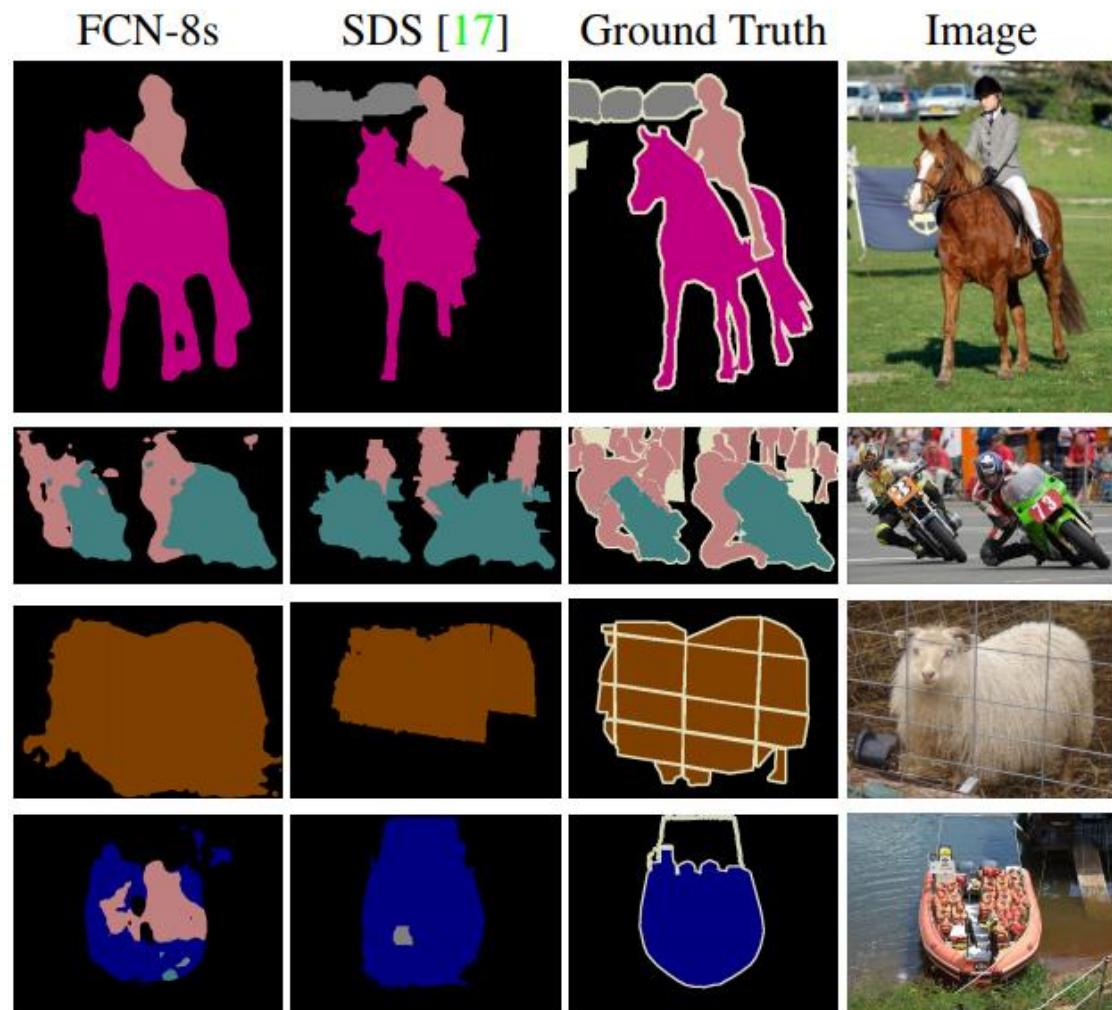


Actual output: 5*5

Need to crop 3*3 centrally

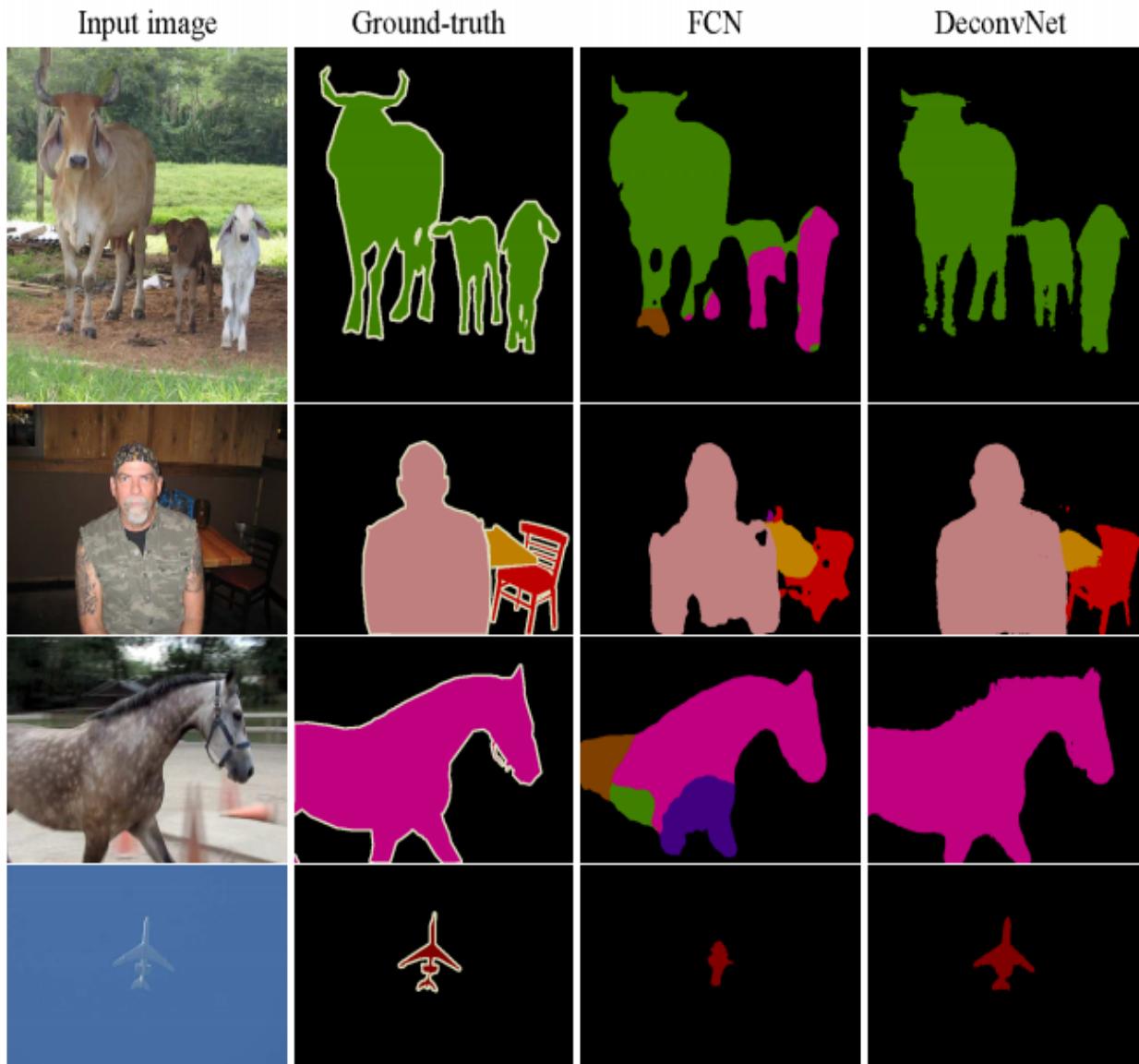
“Fully convolutional (FCN)” results

- Takes advantage of pre-training from classification
- Applied to objects and scenes (NYUd v2)
- But feature pooling reduces spatial sensitivity and resolution

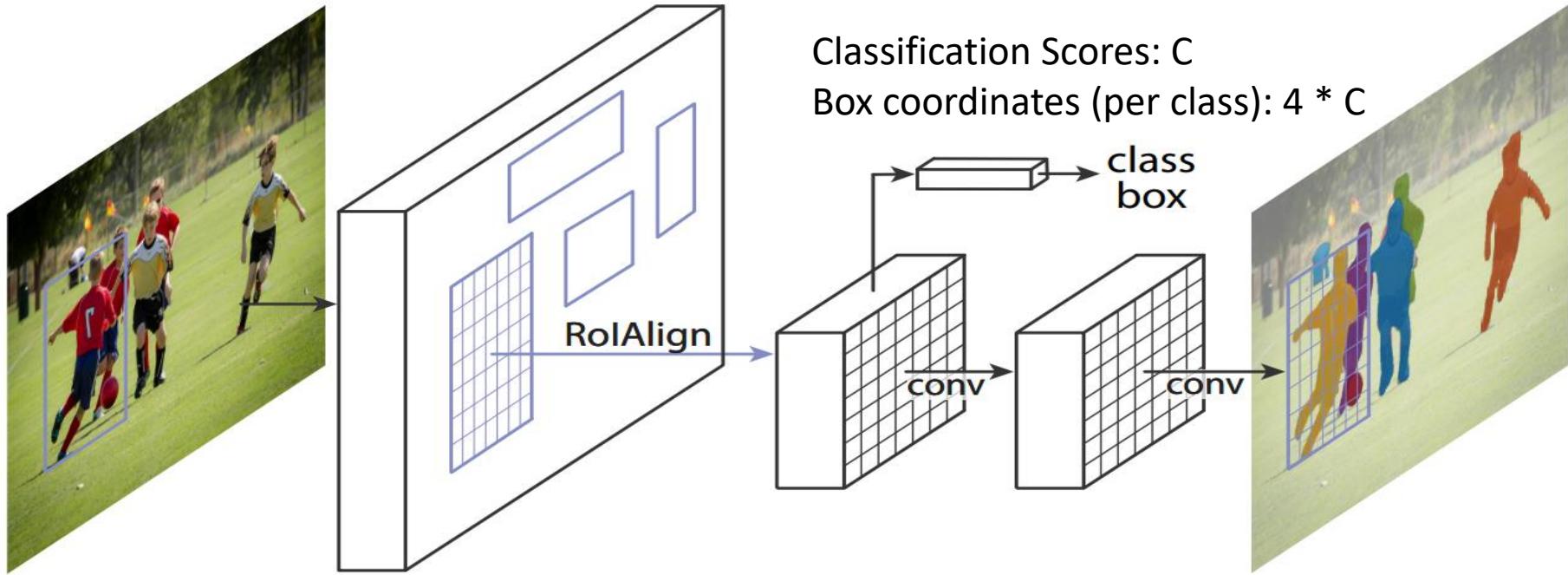


“DeconNet” results

- A deep deconvolution network
- Pixel-wise prediction



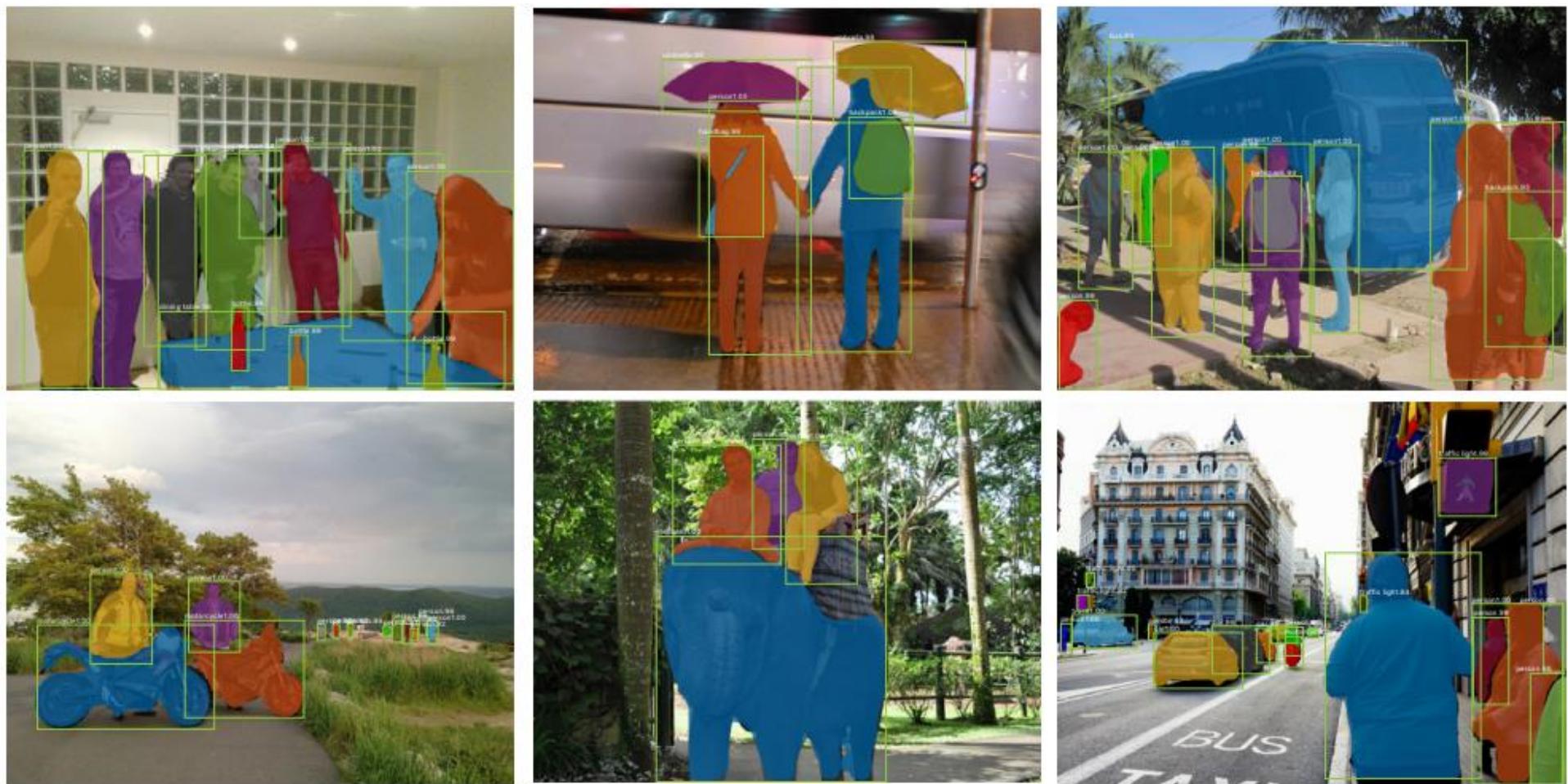
Instance Segmentation: Mask R-CNN (He et al. ICCV 2017)



Mask R-CNN - extends Faster R-CNN by adding a branch for predicting an object mask in parallel with the existing branch for bounding box recognition.

Mask R-CNN is simple to train and adds only a small overhead to Faster R-CNN, running at 5 fps.

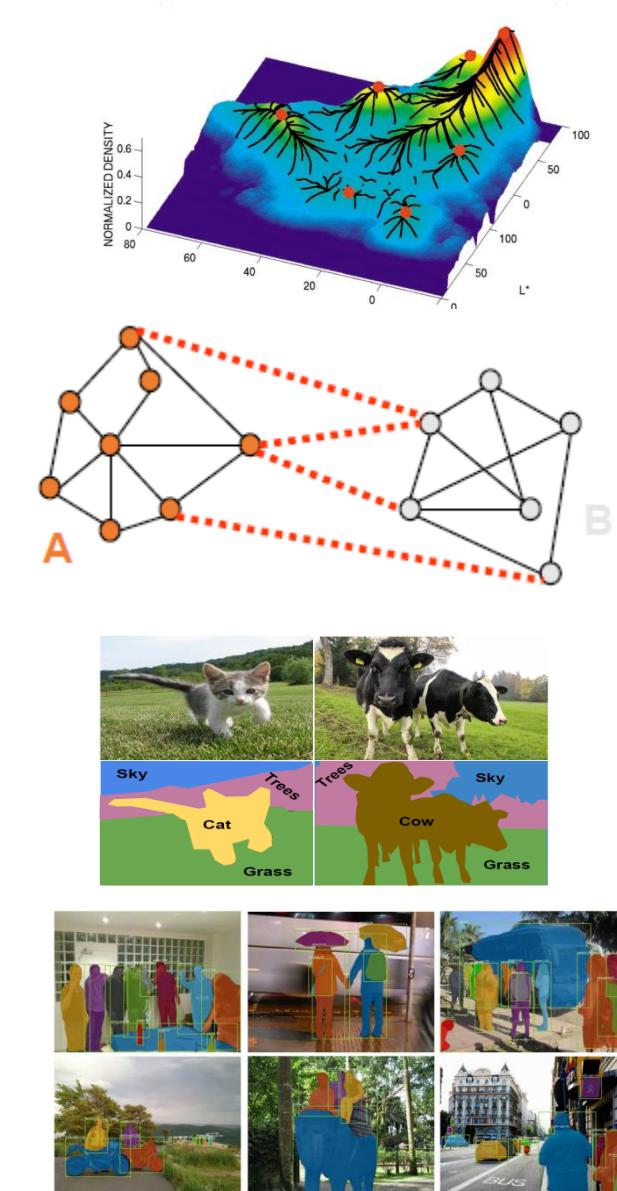
Instance Segmentation: Mask R-CNN (He et al. ICCV 2017)



Mask R-CNN results on the COCO test set. These results are based on ResNet-101, achieving a mask AP of 35.7 and running at 5 fps. Masks are shown in color, and bounding box, category, and confidences are also shown.

Things to remember

- Mean-shift segmentation
 - Good general-purpose segmentation method
 - Generally useful clustering, tracking technique
- Graph Cut Segmentation
 - Normalized graph cut is better
 - High storage requirement and time complexity
- Semantic Segmentation
 - Sliding window inefficient
 - Fully Convolutional: Convolutions at original image resolution is very expensive
 - Fully Convolutional: Down sampling and Up sampling
 - Unsampling usning Unpooling and Transpose Convolution
 - DeconvNet: deeper network
- Instance Segmentation
 - Mak-RCNN: Extension of faster RCNN



Acknowledgements

- Thanks to the following researchers for making their teaching/research material online
 - Forsyth
 - Steve Seitz
 - Noah Snavely
 - J.B. Huang
 - Derek Hoiem
 - J. Hays
 - J. Johnson
 - R. Girshick
 - S. Lazebnik
 - K. Grauman
 - Antonio Torralba
 - Rob Fergus
 - Leibe
 - And many more

Next class

- Projective Geometry and Camera Models

