Mean Shift Tracking

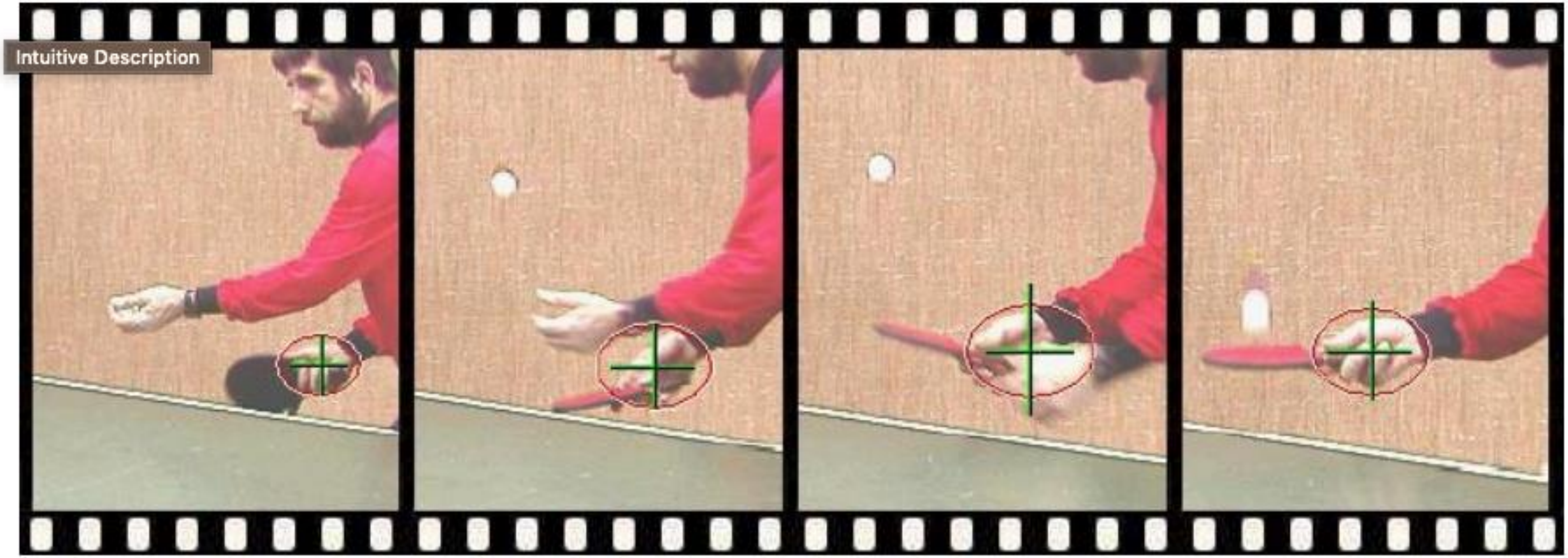
Computer Vision (CS0029)

Mean Shift

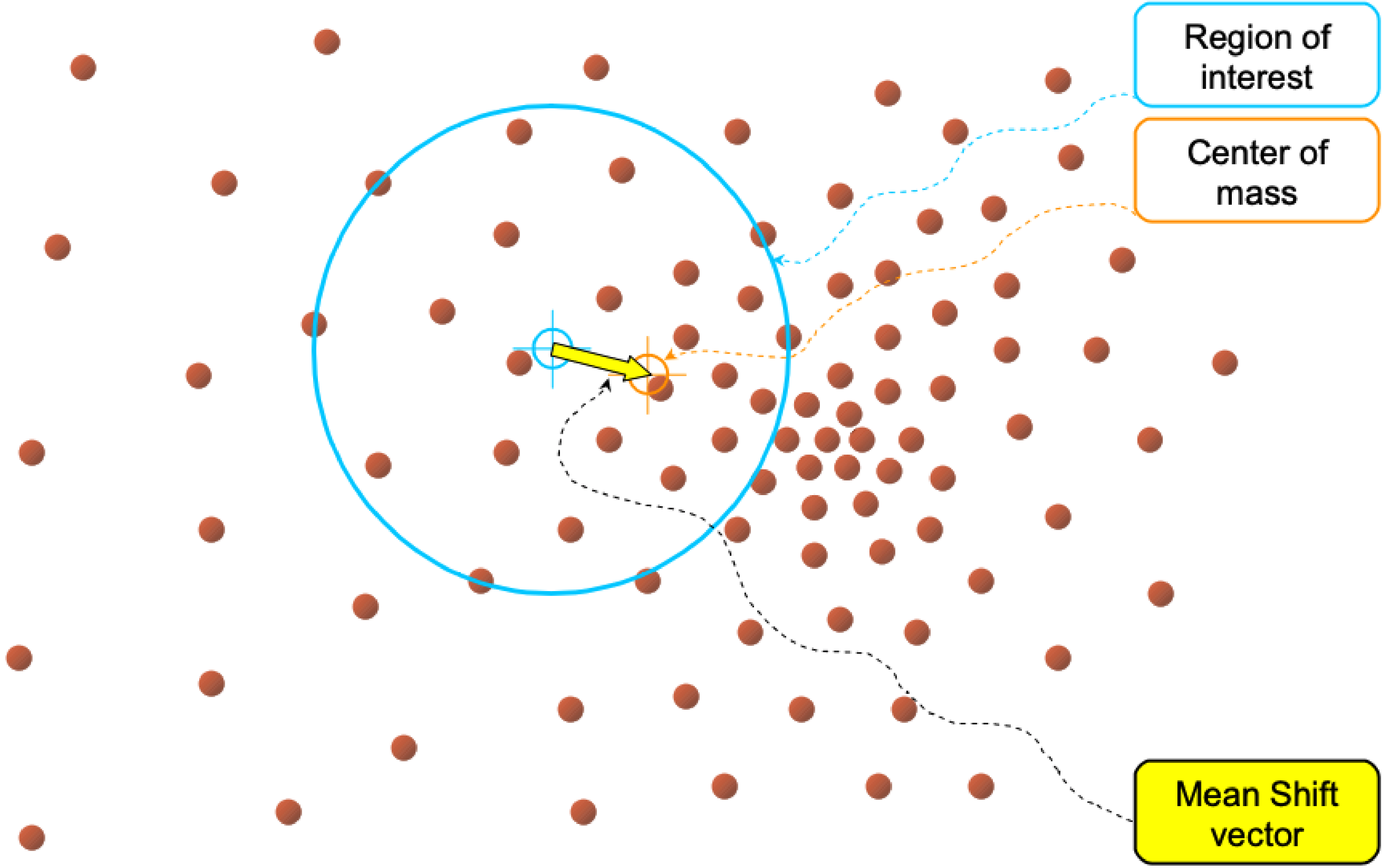
- Real-time non-rigid object tracking

- Mean-shift tracking
  - Represent objects using color histogram
  - Maximization of similarity function using mean shift

Non-Rigid Object Tracking

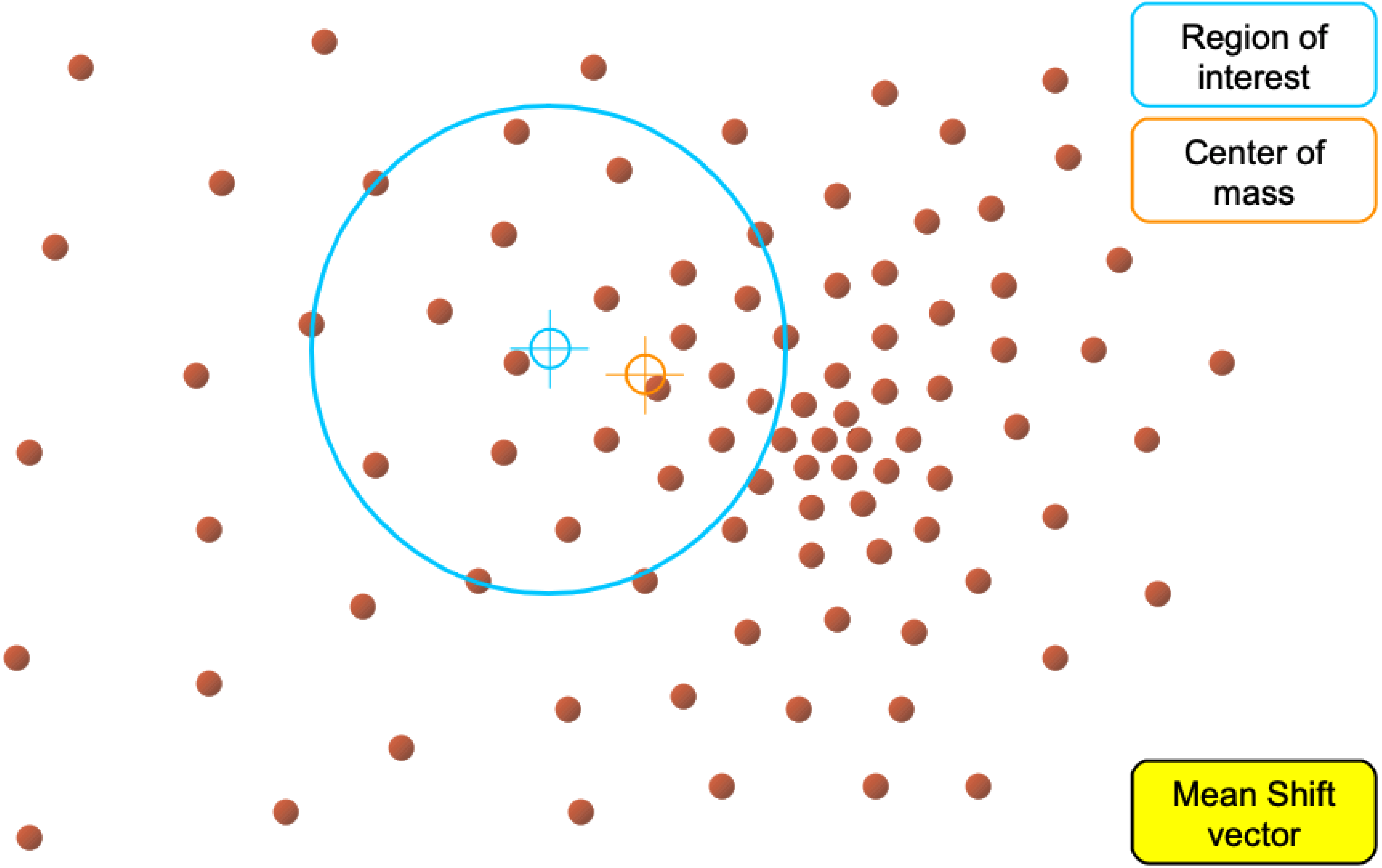


Intuitive Description



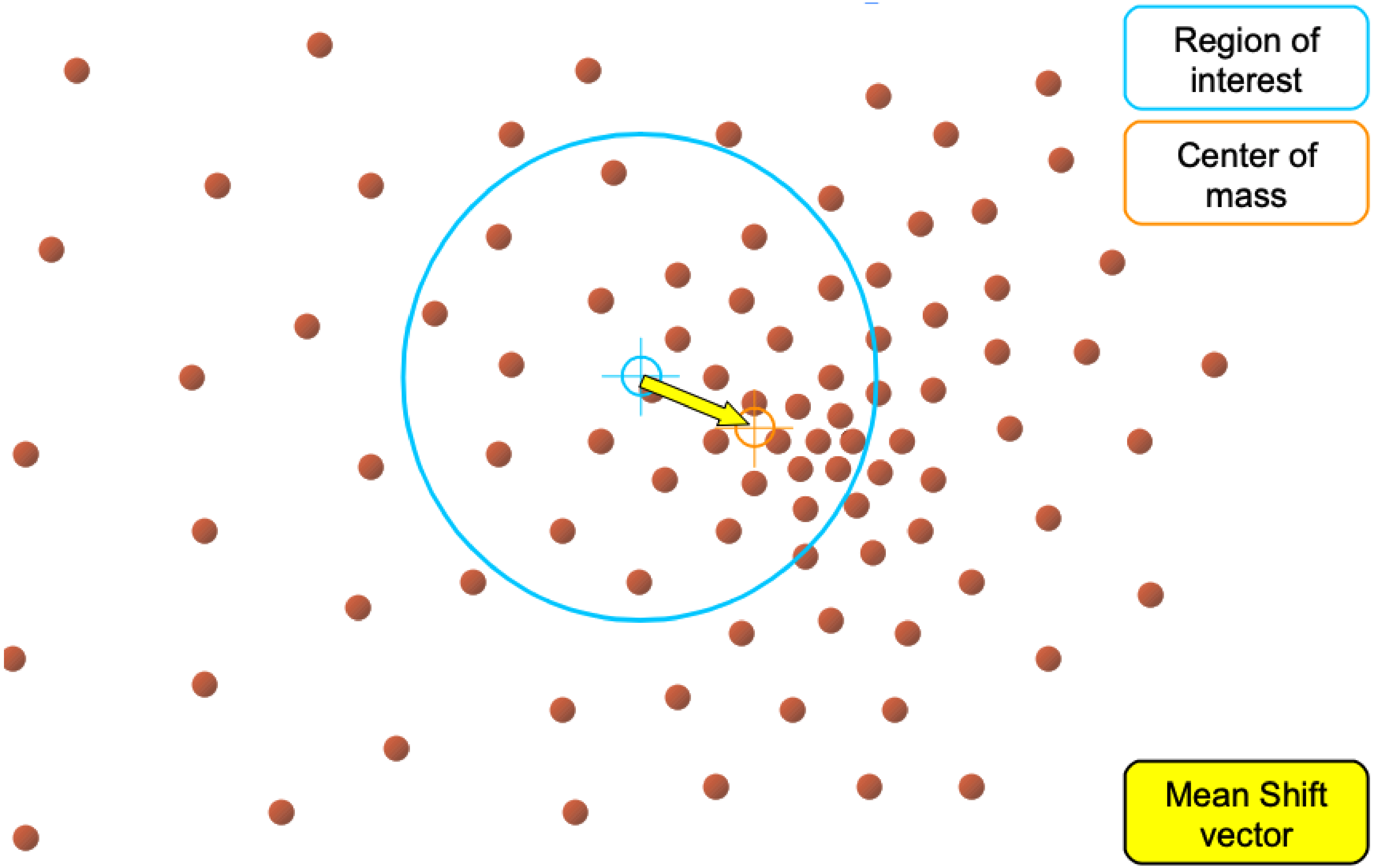
Objective : Find the densest region

Intuitive Description



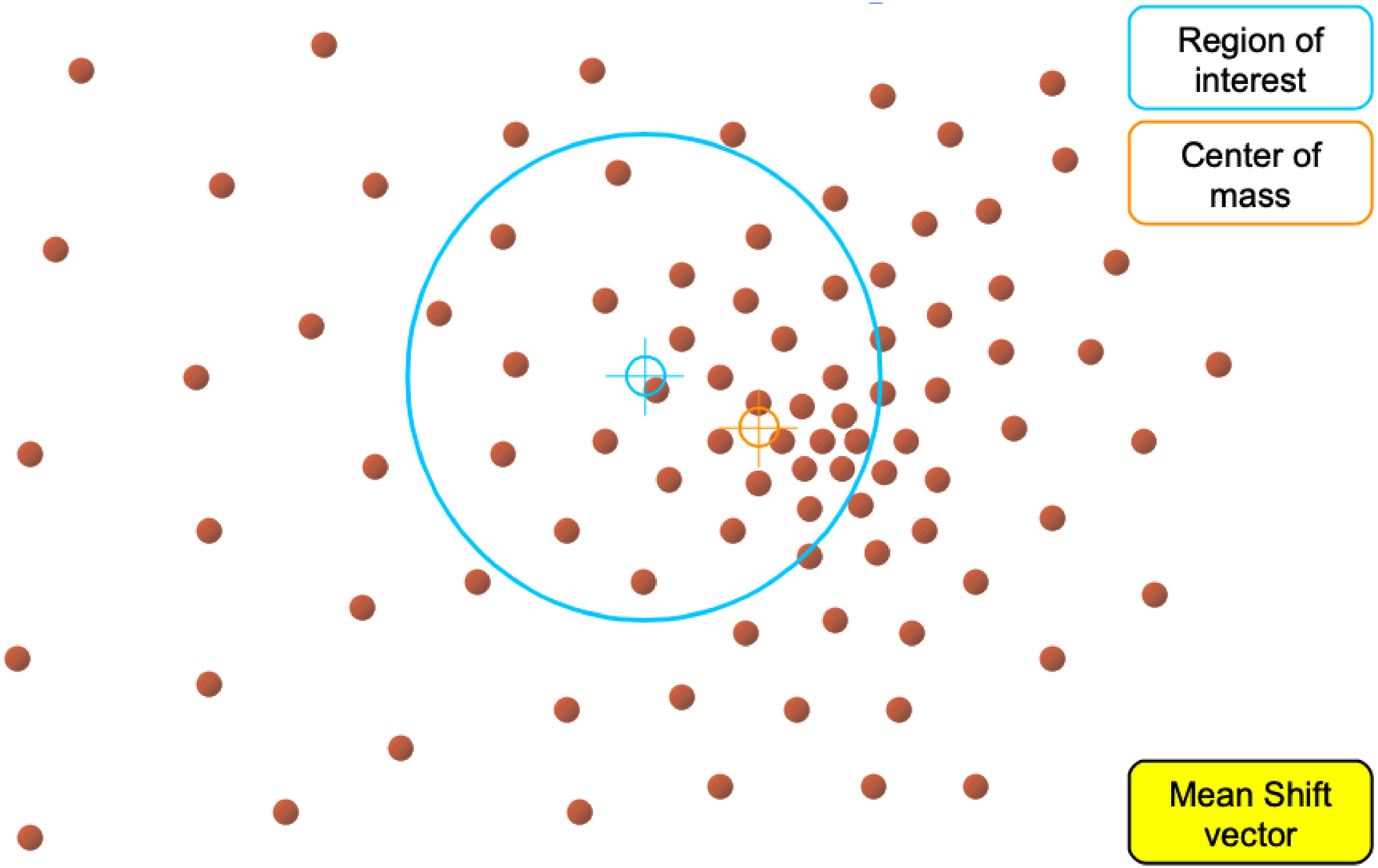
Objective : Find the densest region

Intuitive Description



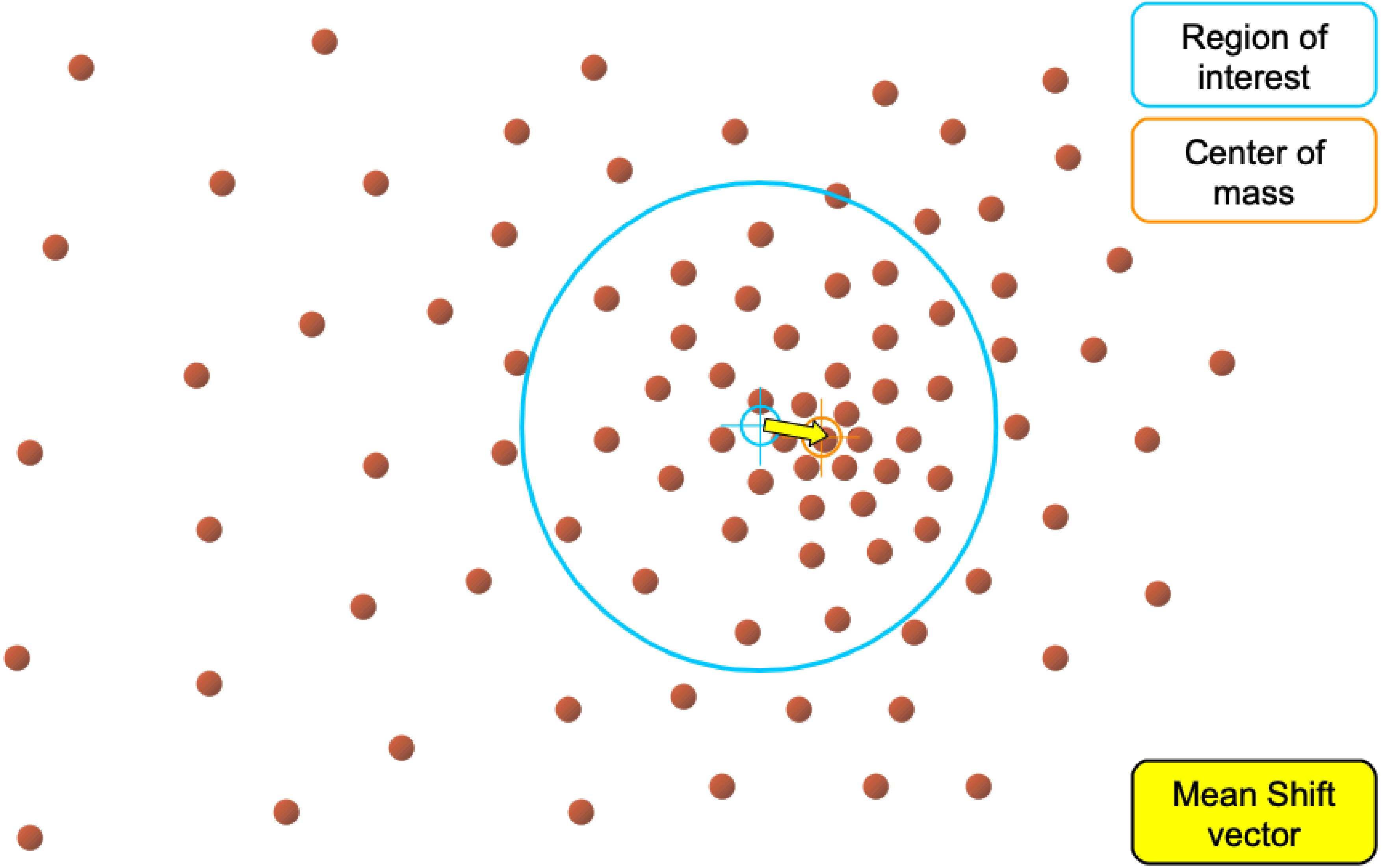
Objective : Find the densest region

Intuitive Description



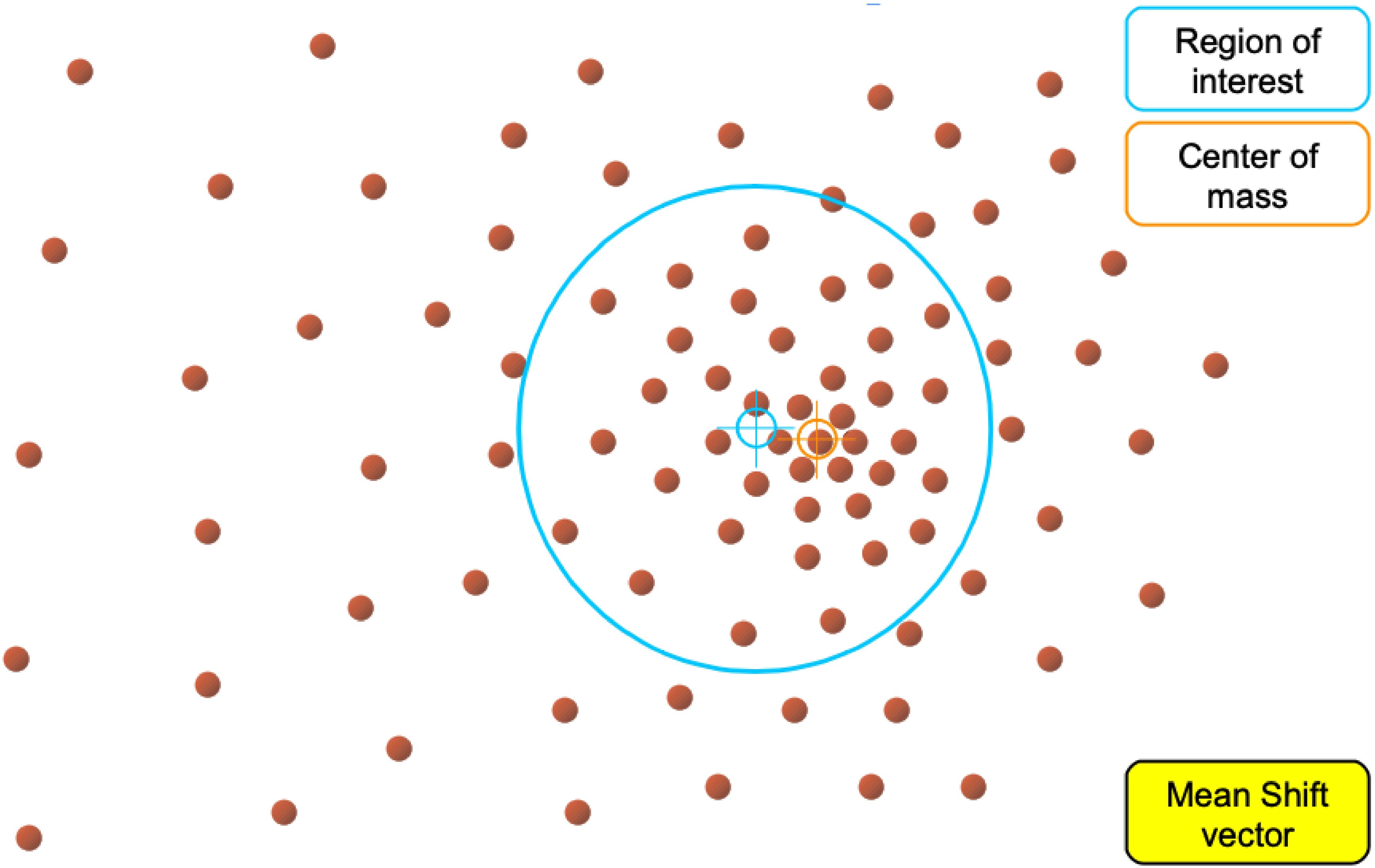
Objective : Find the densest region

Intuitive Description



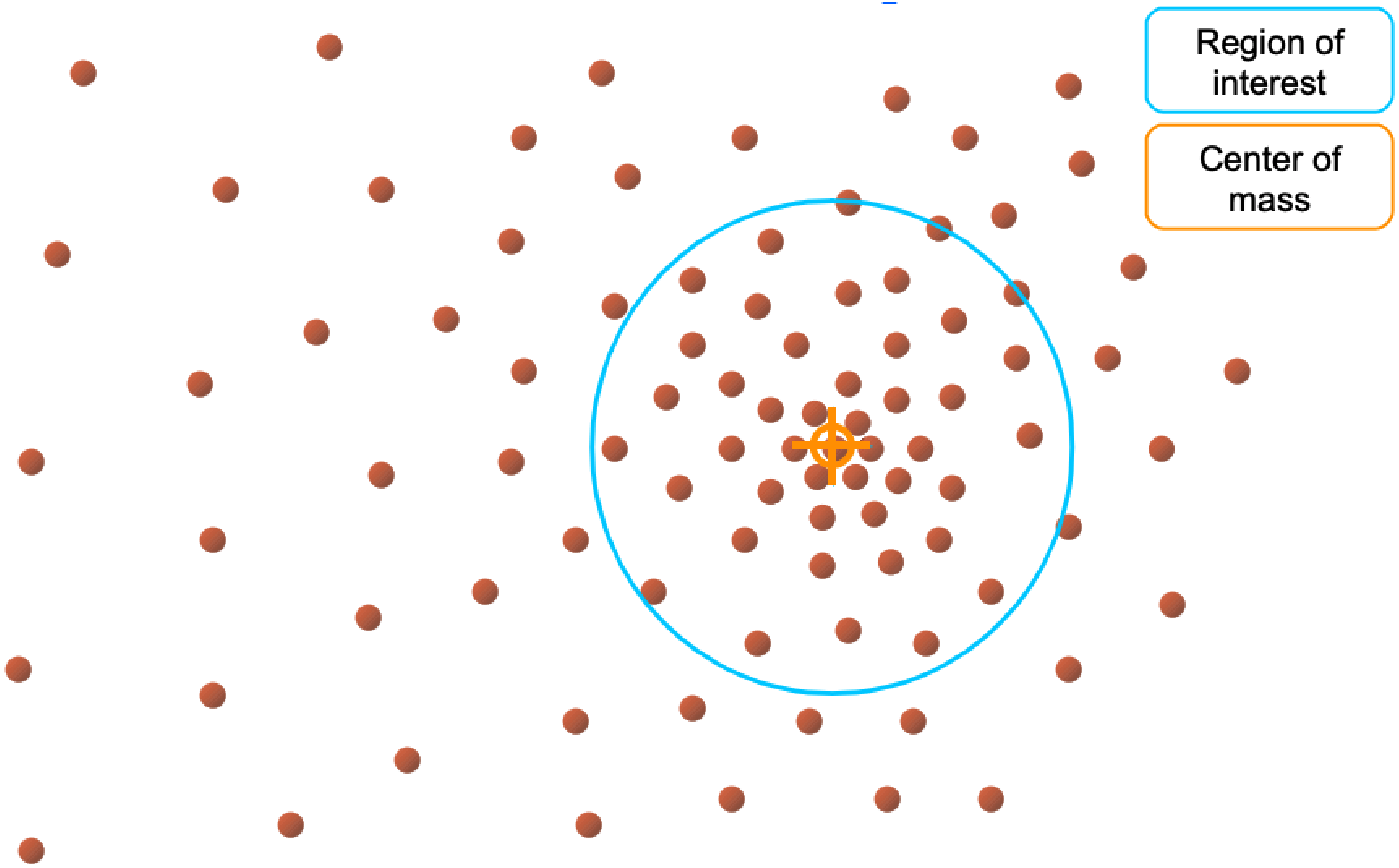
Objective : Find the densest region

Intuitive Description



Objective : Find the densest region

Intuitive Description

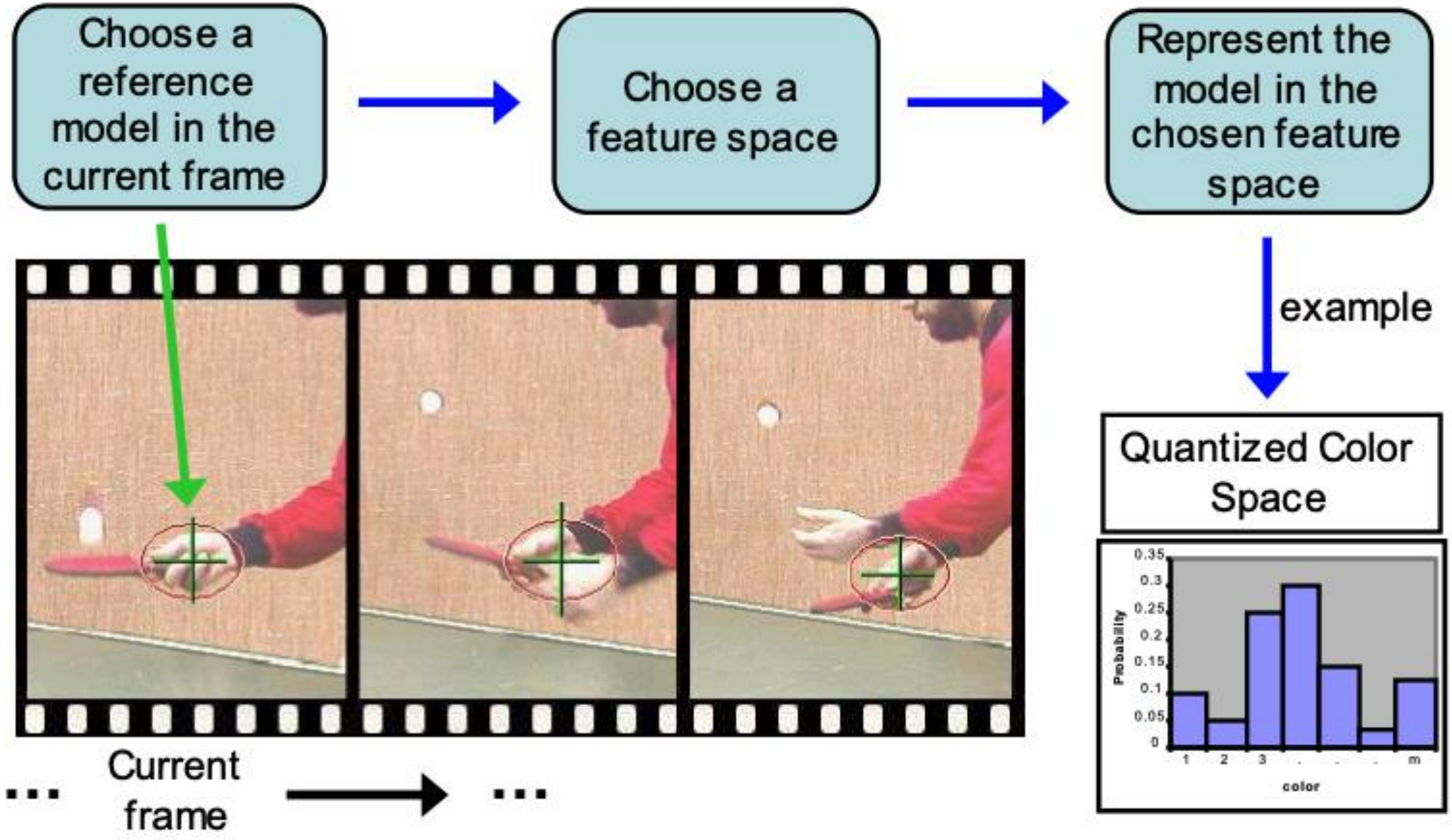


Objective : Find the densest region

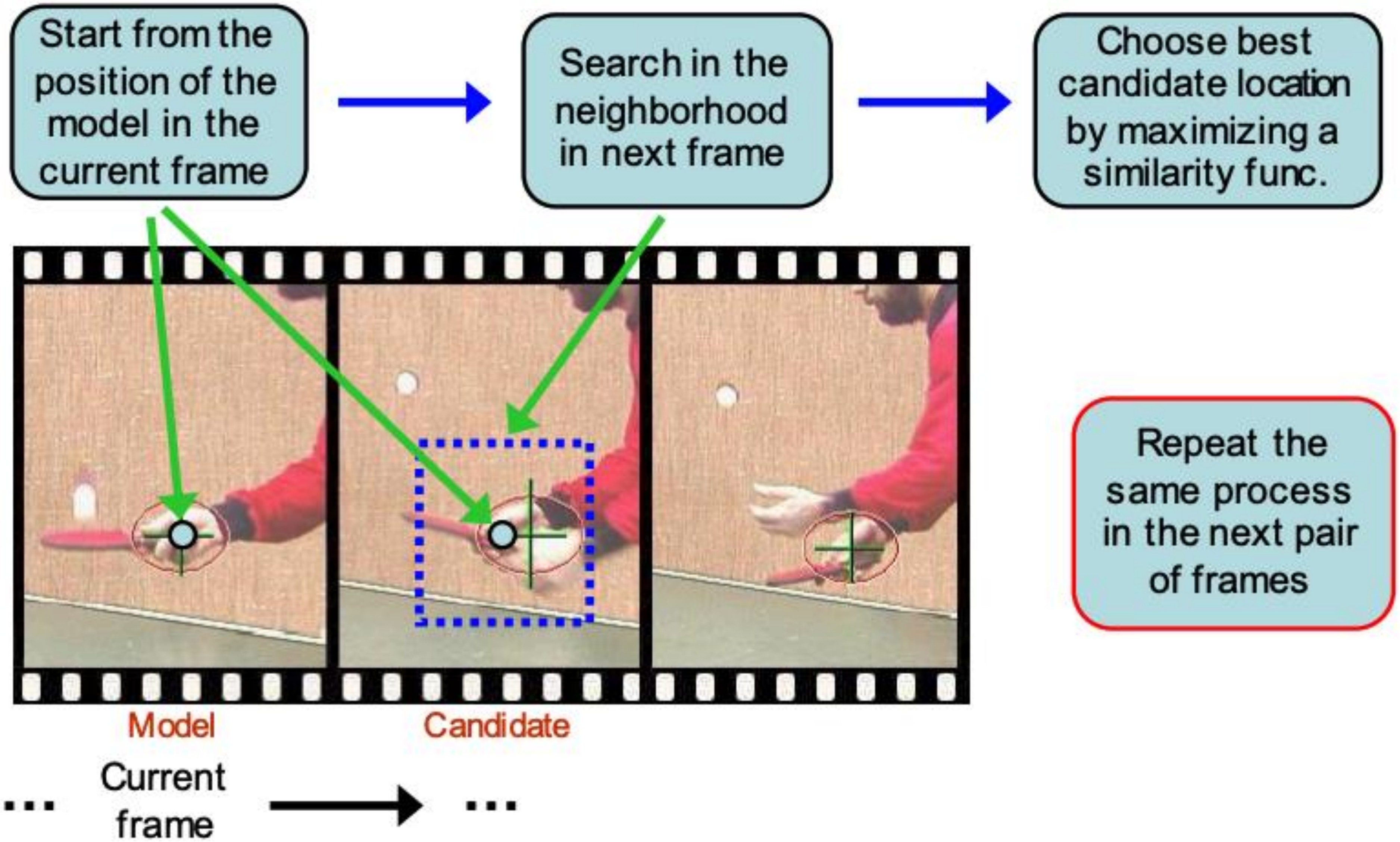
Algorithm

- Obtain a statistical distribution q for current appearance of the object of interest
- Get next video image
- For a candidate at location $\boldsymbol{y} = (x, y)$ in the new image, obtain a statistical distribution $p(\boldsymbol{y})$
- Search the neighborhood of $\boldsymbol{y}$ in the new image
  - Find new "better matching" location $\boldsymbol{y}_{new}$ using mean shift where the distribution $p(\boldsymbol{y}_{new})$ is the most similar to q

General Framework: Target Representation

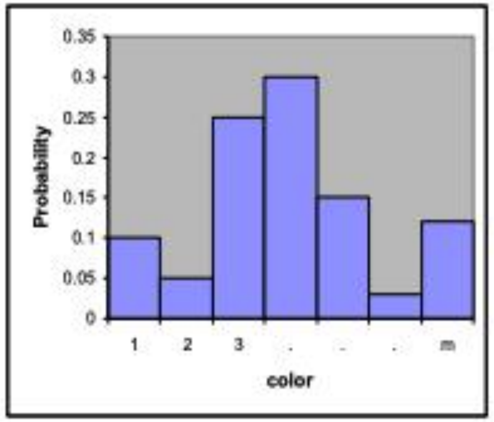


General Framework: Target Localization



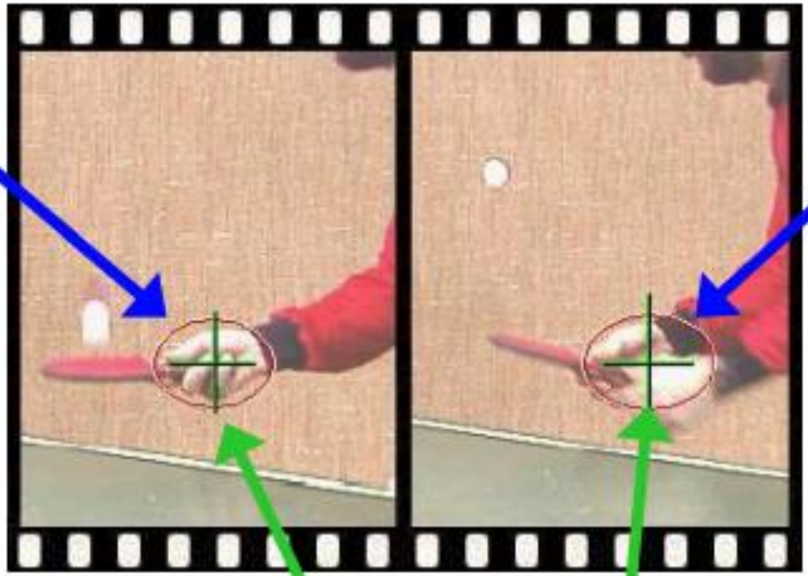
PDF Representation as a m-bin Histograms

Target Model
(centered at **0**)

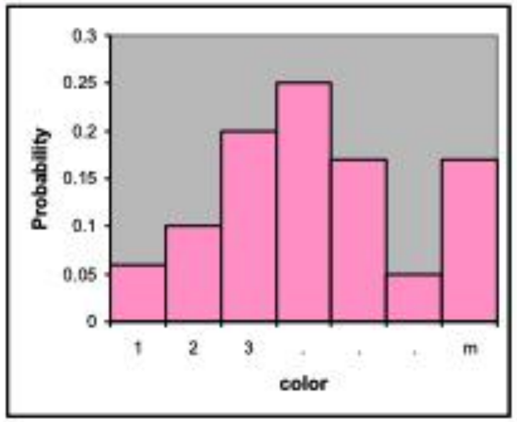


$$\hat{\mathbf{q}} = \left\{ \hat{q}_u \right\}_{u=1\ldots m}$$

$$\sum_{u=1}^{m} \hat{q}_u = 1$$



Candidate
(centered at **y**)



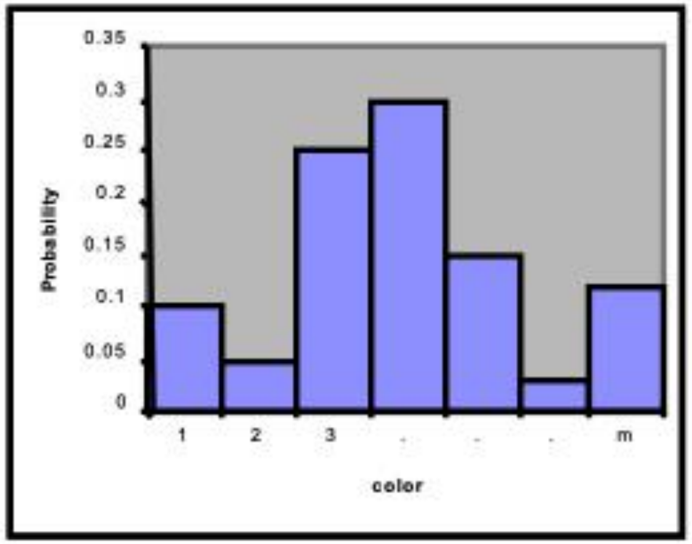
$$\hat{\mathbf{p}}(\mathbf{y}) = \left\{ \hat{p}_u(\mathbf{y}) \right\}_{u=1\ldots m}$$

$$\sum_{u=1}^{m} \hat{p}_u(\mathbf{y}) = 1$$

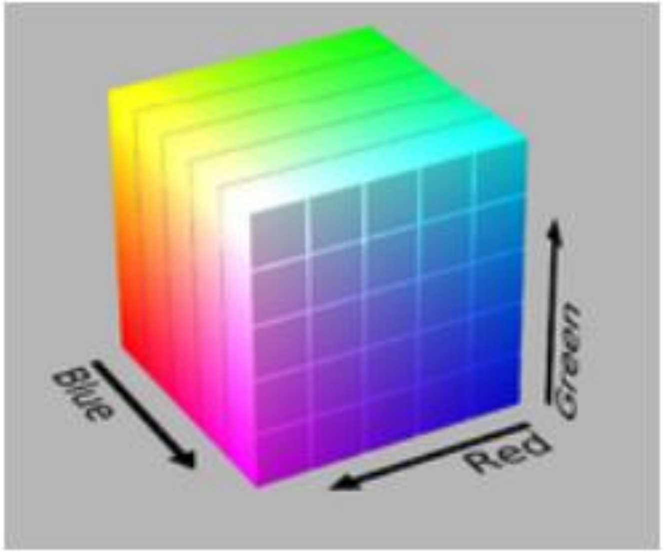
Similarity Function: $\hat{\rho}(\mathbf{y}) = \rho\left[\hat{\mathbf{q}}, \hat{\mathbf{p}}(\mathbf{y})\right]$

Color Histogram/Distribution

1-D quantized color (grayscale)



3-D quantized color (RGB)

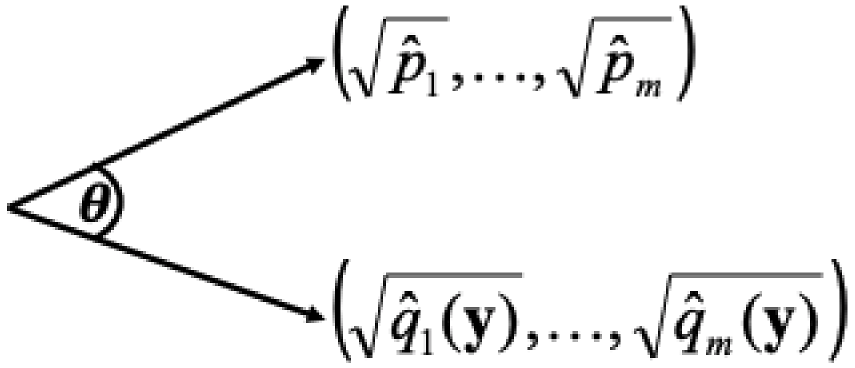


Similarity Function

- Similarity between two discrete distributions estimated using Bhattacharyya Coefficient

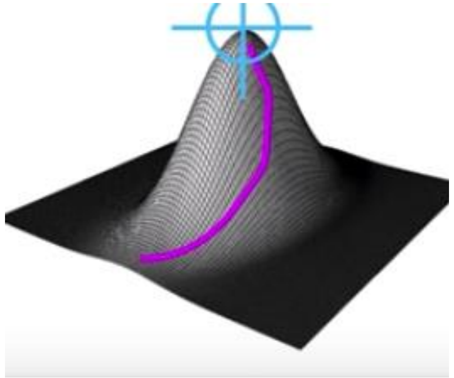
$$\hat{\rho}(\mathbf{y}) \equiv \rho\left[\hat{\mathbf{p}}(\mathbf{y}), \hat{\mathbf{q}}\right] = \sum_{u=1}^{m} \sqrt{\hat{p}_u(\mathbf{y})\hat{q}_u} = \cos\theta$$

- Interpretation: Cosine of angle or (normalized) correlation between m-dimensional unit vectors



$$\sum_{u=1}^{m} \hat{p}_u(\mathbf{y}) = 1 \qquad \sum_{u=1}^{m} \hat{q}_u = 1$$

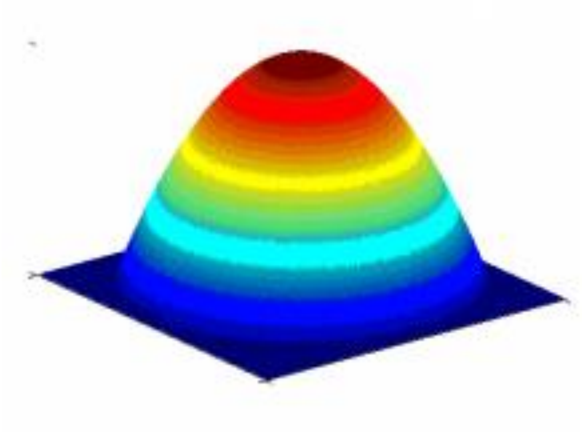
Similarity Function and Problems



- Local maximum of similarity function gives the location of best match in the next frame
- If only color information is used, spatial information is lost (histograms), and it is not always a smooth surface
- Smooth the similarity function by using weighted histograms
    - Weight pixel contributions based on their spatial location (how close to center of the region)

Model

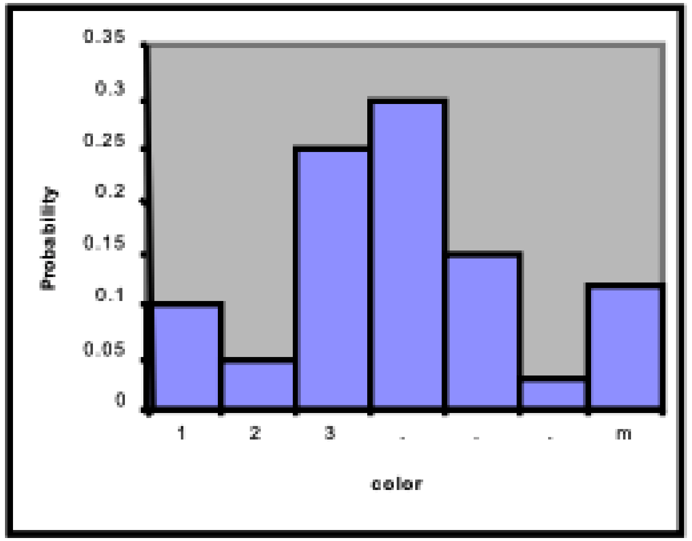
- We need a differentiable, isotropic, monotonically decreasing kernel to assign smaller weights to pixel at periphery of the circular region of radius h

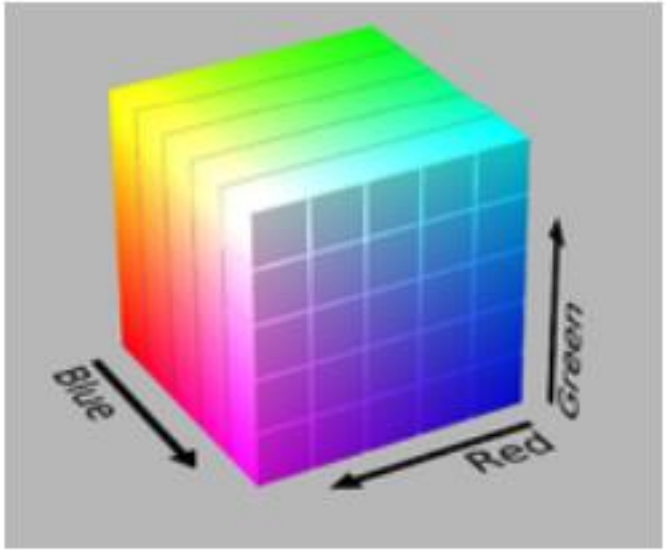


- Epanechnikov profile
  - $k(r) = \begin{cases} 1 - r & if\ r < 1 \\ 0 & otherwise \end{cases}$
  - $r = [\frac{\sqrt{x*x+y*y}}{h}]^2$
    - $\sqrt{x * x + y * y}$ is distance from center (0,0)
    - h is bandwidth size

Weighted Model

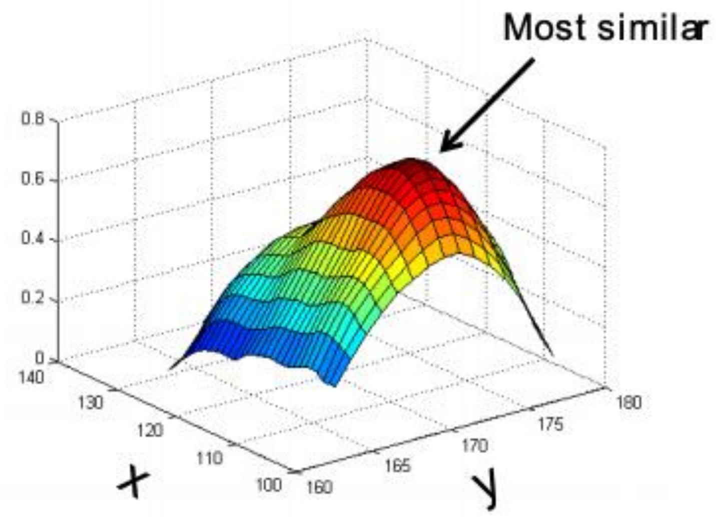
- Probability of feature (color) $u = 1 \ldots m$ in target model is computed as:
  - $\widehat{q_u} = C \sum_{i=1}^{n} k(\left\| \frac{x_0 - x_i}{h} \right\|^2) \, \delta(b(x_i) - u)$
  - $\left\| \frac{x_0 - x_i}{h} \right\|^2$: pixel weight
  - $\delta[b(x_i) - u]$: pixel bin color index is u
  - $\delta(.)$: Kronecher delta function: 1 if input is 0, 0 other wise
  - C is just a normalization content to make q a pdf



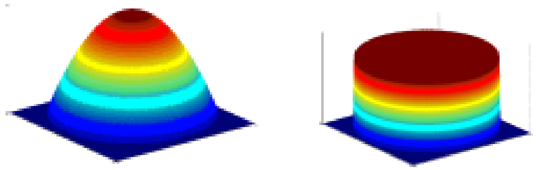


Target Localization

- Similarity function evaluated at different locations



- Why Epanechnikoc profile?
  - Derivative of the profile is constant if we use uniform model



- Simplified Mean Shift Vector
  - $y_1 = \frac{\sum_{i=1}^{n} x_i w_i}{\sum_{i=1}^{n} w_i}$ (x_i: pixel location in valid area)
  - $w_i = \sum_{u=1}^{m} \sqrt{\frac{q_u}{p_u(y_0)}}\, \delta(b(x_i) - u)$

Algorithm

- Generate model $\widehat{q_u}$
- Generate target $\widehat{p_u}$ in current frame (start at previous frame location y_0)
- Compute weights w_i
- Find next best location of the target candidate using $y_1 = \frac{\sum_{i=1}^{n} x_i w_i}{\sum_{i=1}^{n} w_i}$
- If $\|y_1 - y_0\| < \varepsilon$, then stop, otherwise set y_0 = y_1 and go to step 1

- Note: Do not round any values for locations during the iterations

Result

- https://www.youtube.com/watch?v=opE7frnFRqc

Summary

- Algorithm for **non-rigid** object tracking

- Compare the weighted histograms (by patch color and location) of template patch(previous frame) and candidate patch (current frame)
  - Find the target location by searching the most similar candidate patch

- Use mean shift to find the target location in the space of similarity function
  - Mean shift vector equation: $y_1 = \frac{\sum_{i=1}^{n} x_i w_i}{\sum_{i=1}^{n} w_i}$

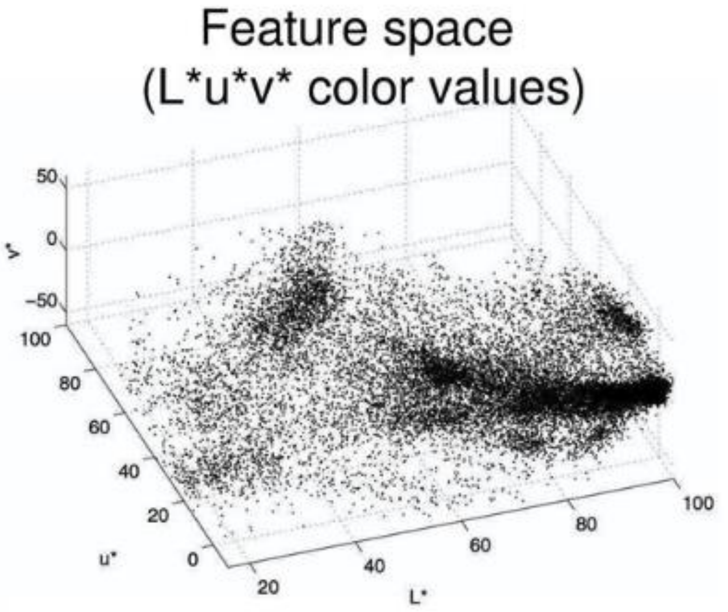
Mean Shift Segmentation

Computer Vision (CS0029)

Segmentation

- Group similar color (same objects)
- Find modes (peaks) in the feature space

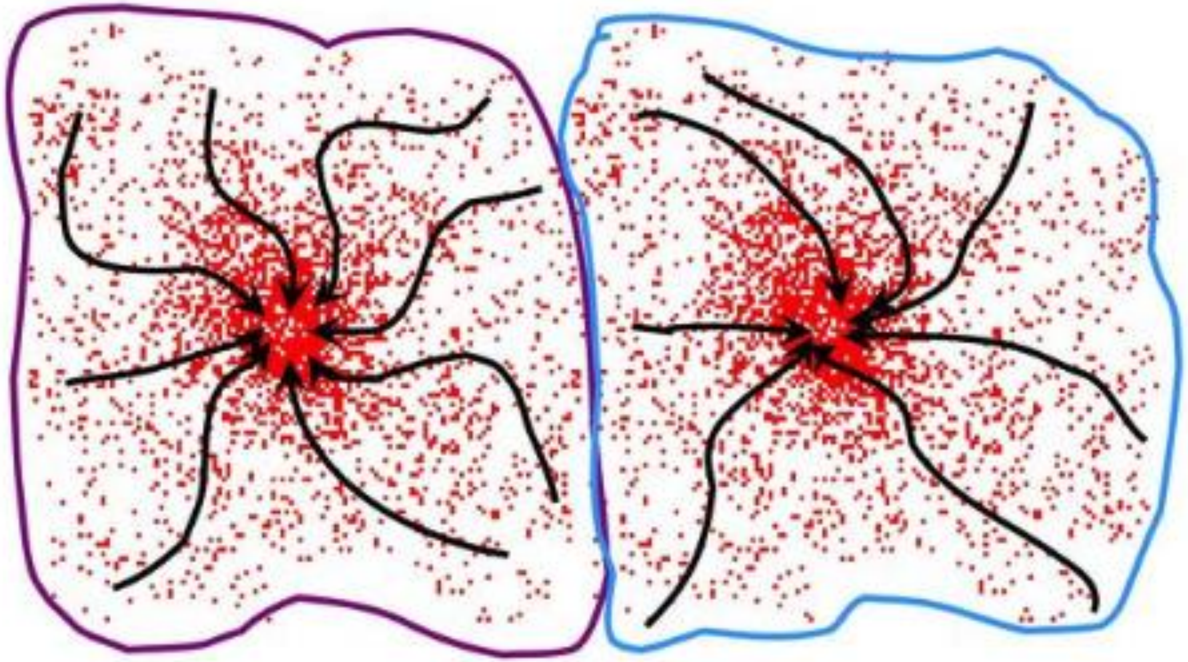
- K-mean clustering
  - Cluster pixels by similar color
  - You have to give the K (number of clusters)



Mean Shift Clustering

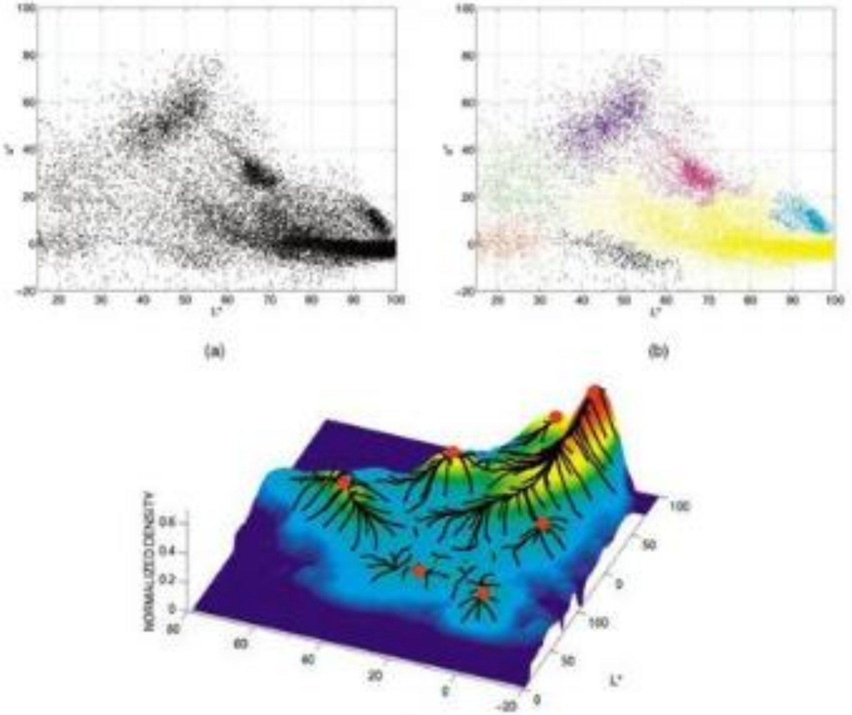
- Cluster: all data points in the attraction basin of a mode

- Attraction basin: the region for which all trajectories lead to the same mode

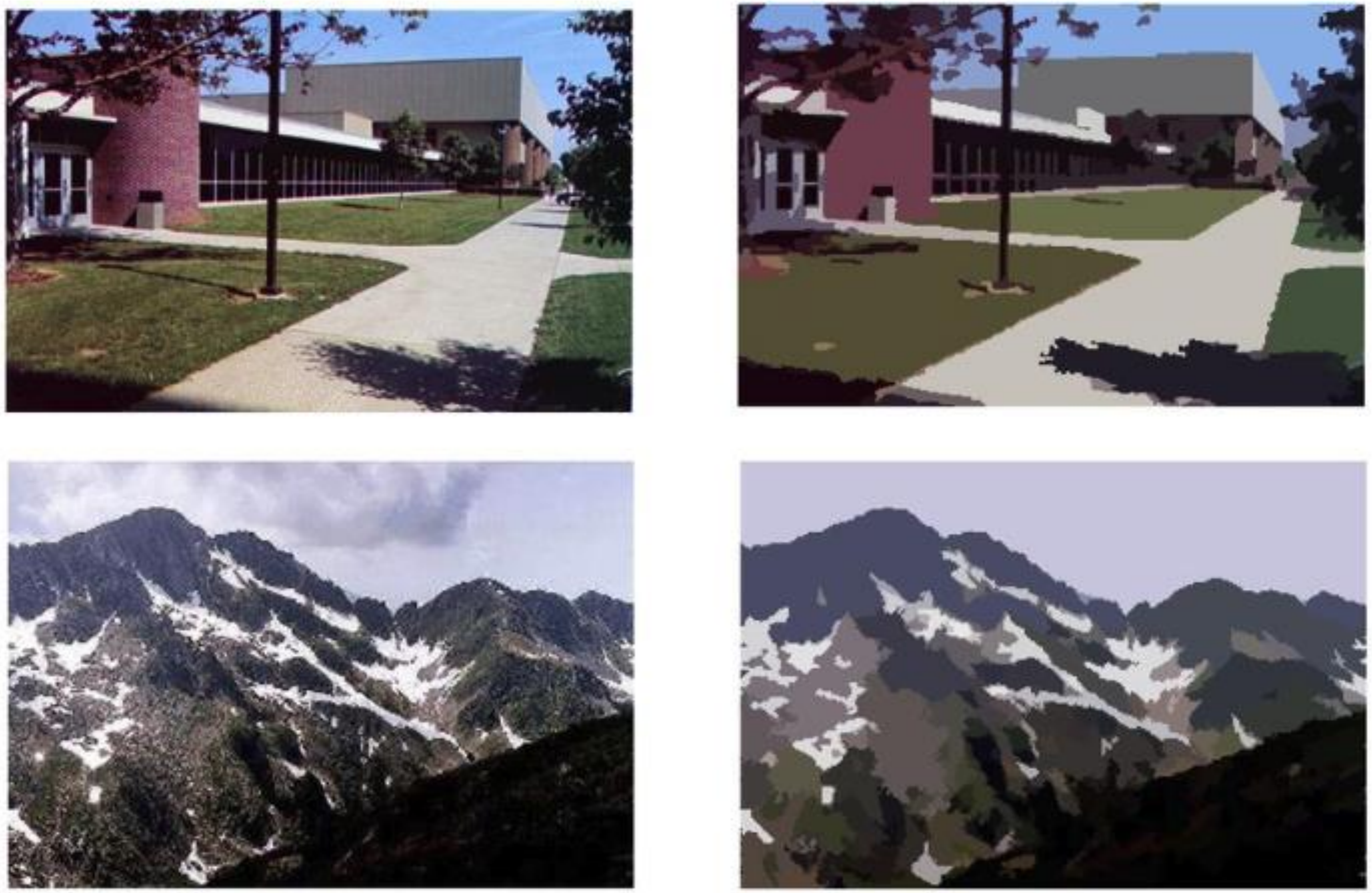


Mean Shift Segmentation

- Initialize windows at individual feature space
  - We may use information from L,U,V, and maybe X,Y
- 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 Segmentation Results

Procs and Cons of Mean Shift Segmentation

- Procs
  - Does not assume spherical clusters
  - Just a single parameter (window size)
  - Find variable number of modes
  - Robust to outliers
- Cons
  - Output depends on window size
  - Computationally expensive