CS 1674/2074: Grouping: segmentation

PhD. Nils Murrugarra-Llerena
nem177@pitt.edu

[Motivation] Grouping

Finding a Crowd at a Concert

Imagine you're at a large outdoor music festival and you need to find your friends.

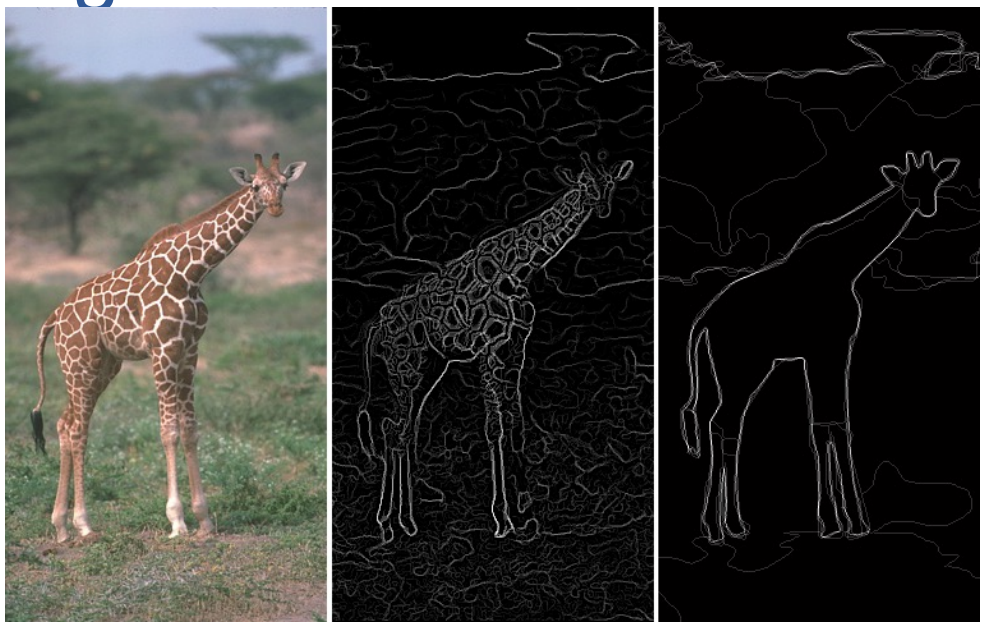
What would you do?

Your eyes don't process the scene one person at a time. Instead, you automatically group people together. You might spot a cluster of people wearing the *same band t-shirt*, or a *tight-knit group all dancing together in a circle*. You instantly recognize these as distinct groups, not just random individuals.

Plan for today

- Segments
  - Find which pixels form a consistent region
  - Clustering (e.g. K-means)

Edges vs Segments



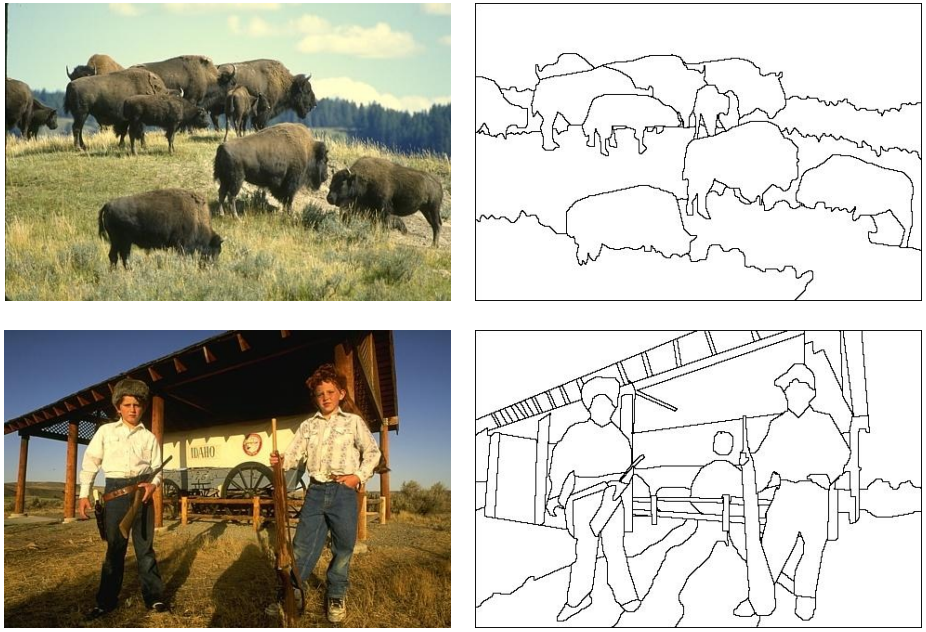
- Edges: More low-level; don't need to be closed
- Segments: Ideally one segment for each semantic group/object; should include closed contours

Figure adapted from J. Hays

The goals of segmentation

- Separate image into coherent “objects”

image | human segmentation

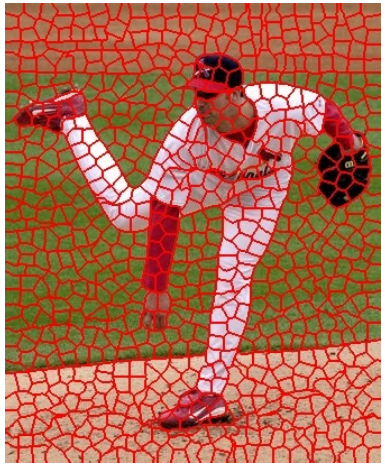


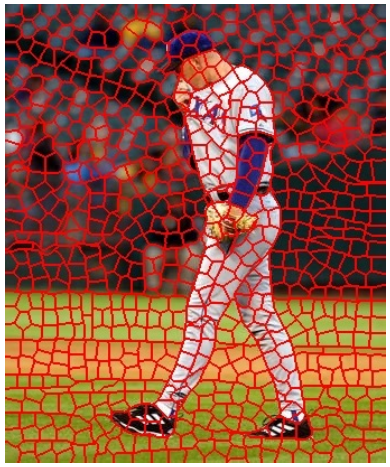
Source: L. Lazebnik

The goals of segmentation

- Separate image into coherent “objects”
- Group together similar-looking pixels for efficiency of further processing

“superpixels”





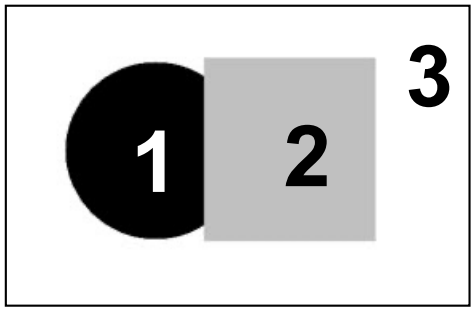
X. Ren and J. Malik. Learning a classification model for segmentation. ICCV 2003.

Source: L. Lazebnik

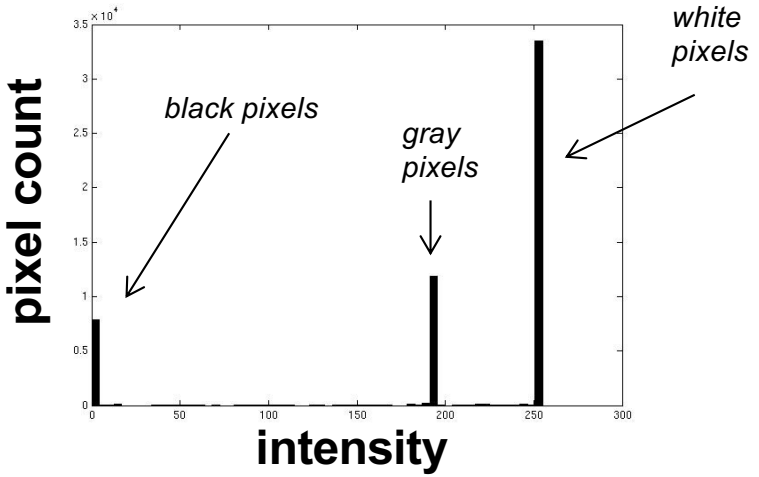
Similarity

Slide: K. Grauman

Image Segmentation: Toy Example



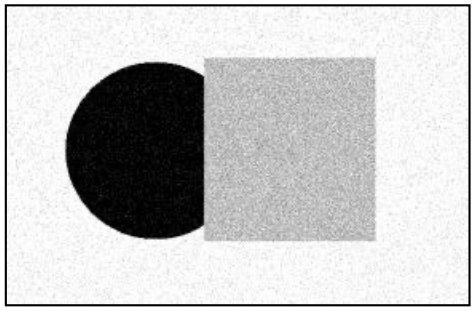
input image



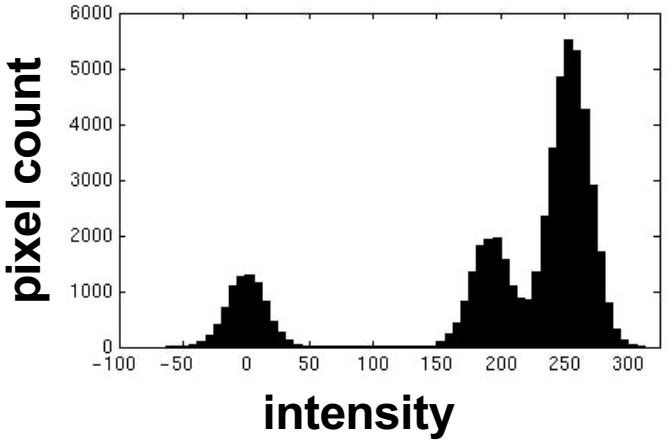
- These intensities define the three groups.
- We could label every pixel in the image according to which of these primary intensities it is.
  - i.e., *segment* the image based on the intensity feature.
- What if the image isn't quite so simple?

Source: K. Grauman

Image Segmentation: Toy Example



input image

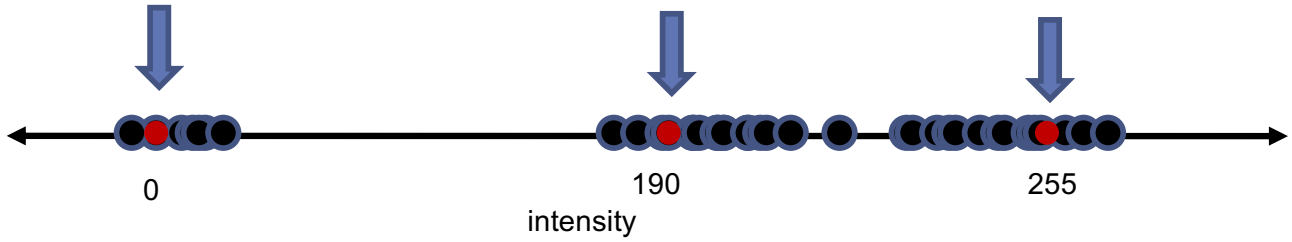


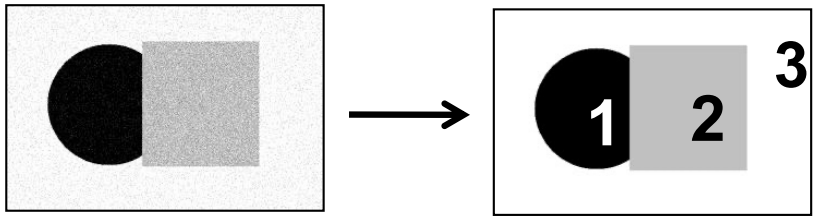
- Now how to determine the three main intensities that define our groups?

- We need to ***cluster***.

Source: K. Grauman

Image Segmentation: Toy Example





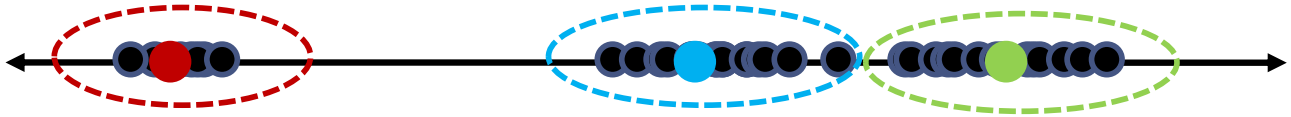
- Goal: choose three "centers" as the representative intensities, and label every pixel according to which of these centers it is nearest to.
- Best cluster centers are those that minimize *sum of squared differences* (SSD) between all points and their nearest cluster center c_i:

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

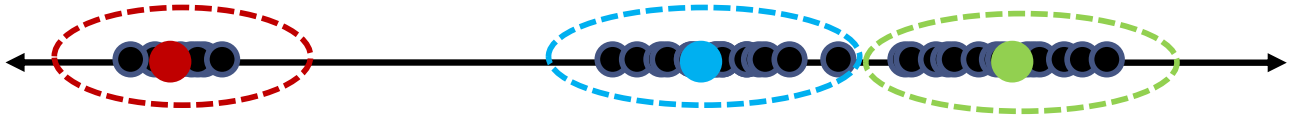
Source: K. Grauman

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.



Source: K. Grauman

K-means clustering

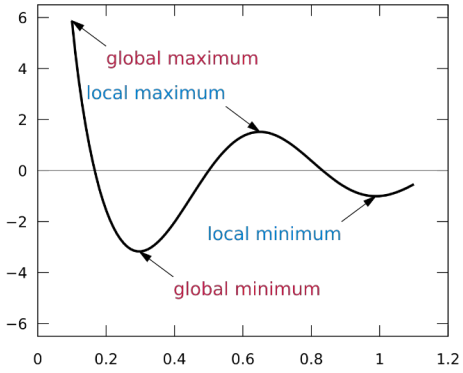
- **Basic idea**: randomly initialize the *k* cluster centers, and iterate between the two steps we just saw.

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" of objective:

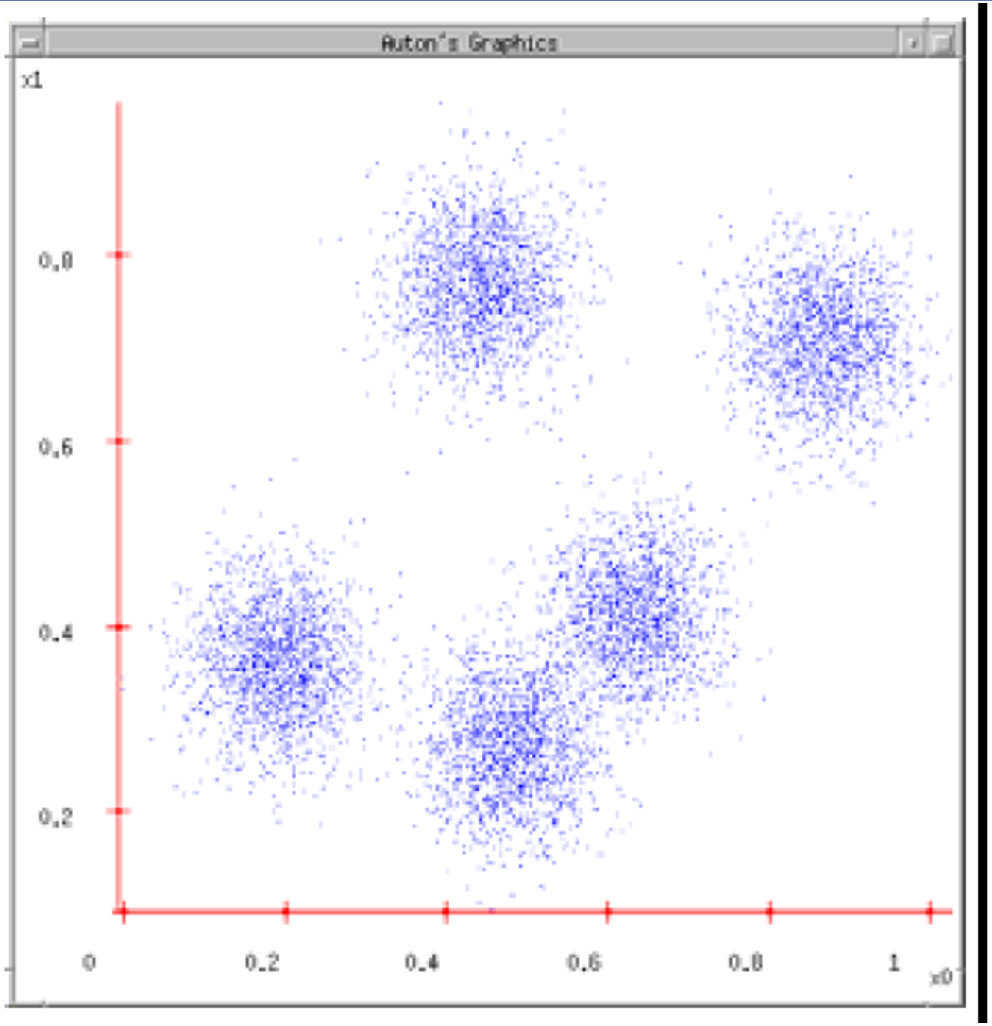
$$\sum_{\text{clusters } i} \sum_{\text{points p in cluster } i} \|p - c_i\|^2$$



Slide: Steve Seitz, image: Wikipedia

K-means

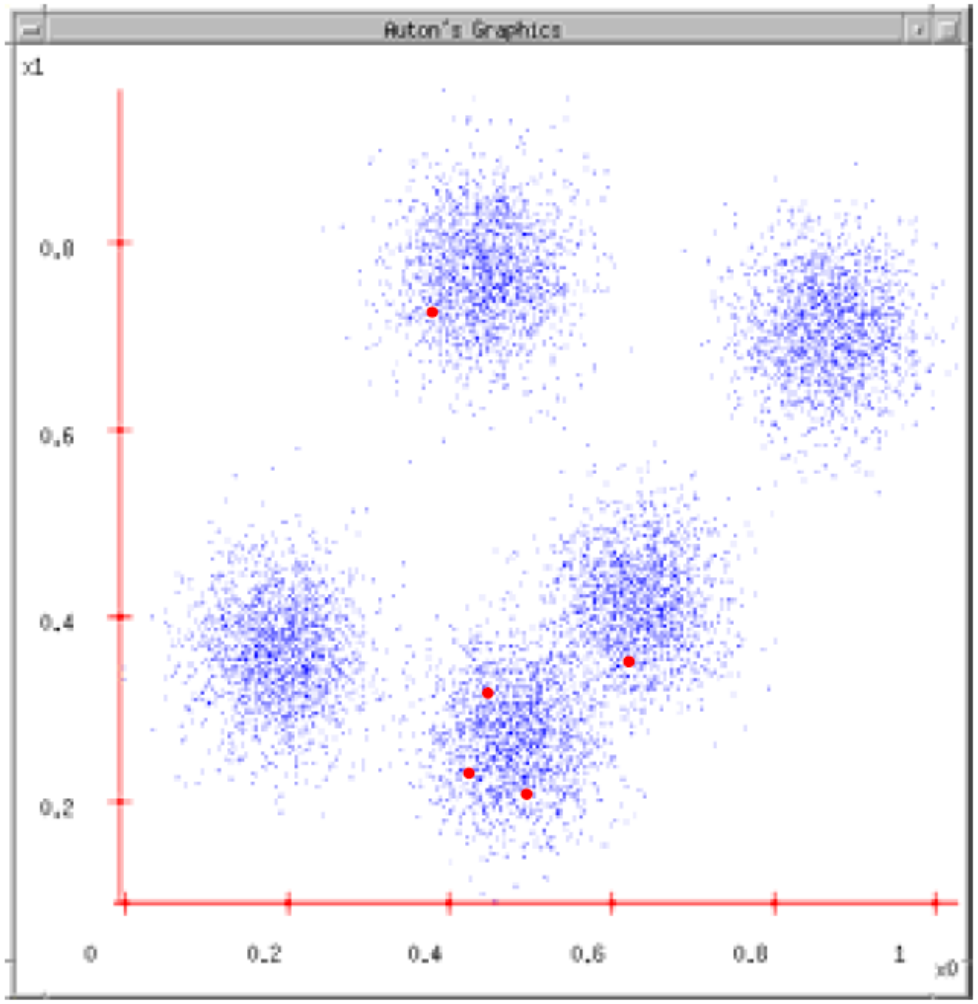
1. Ask user how many clusters they'd like. *(e.g. k=5)*



Source: A. Moore

K-means

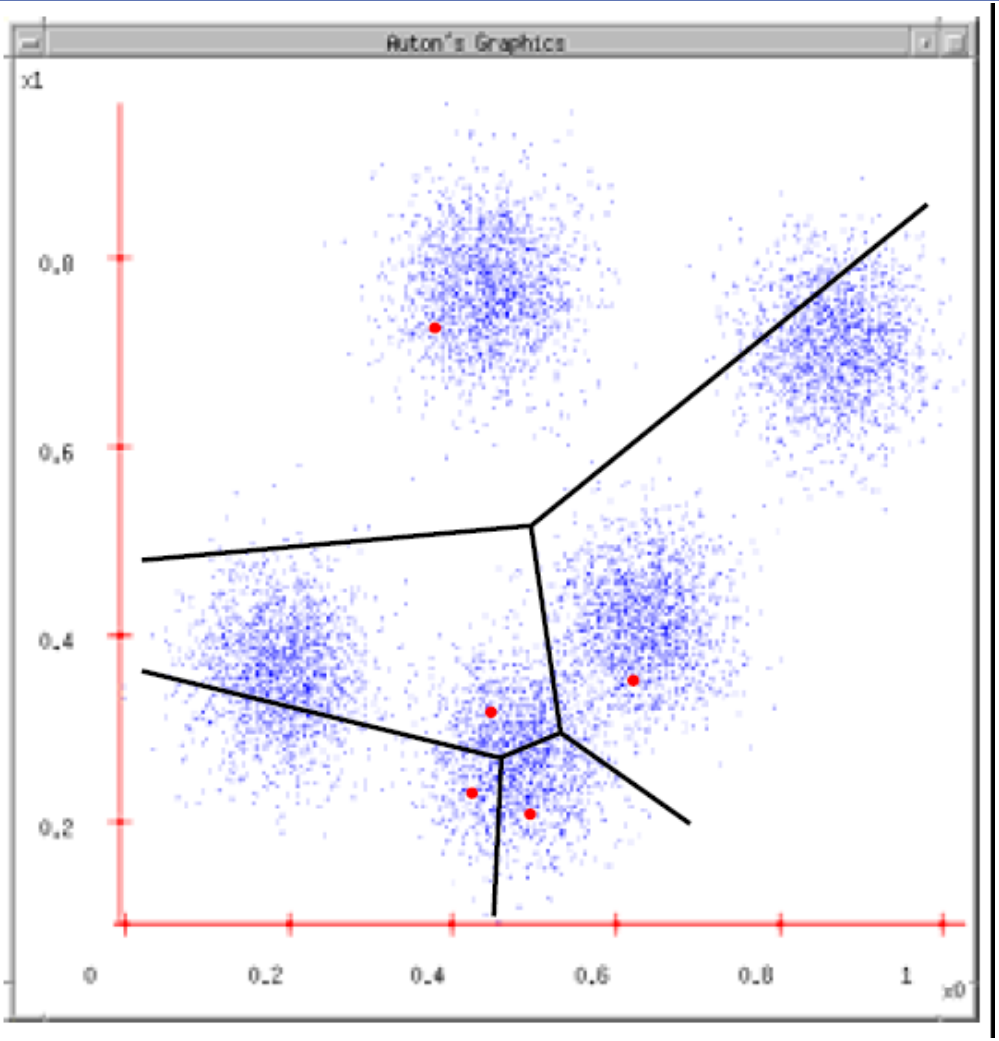
1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations



Source: A. Moore

K-means

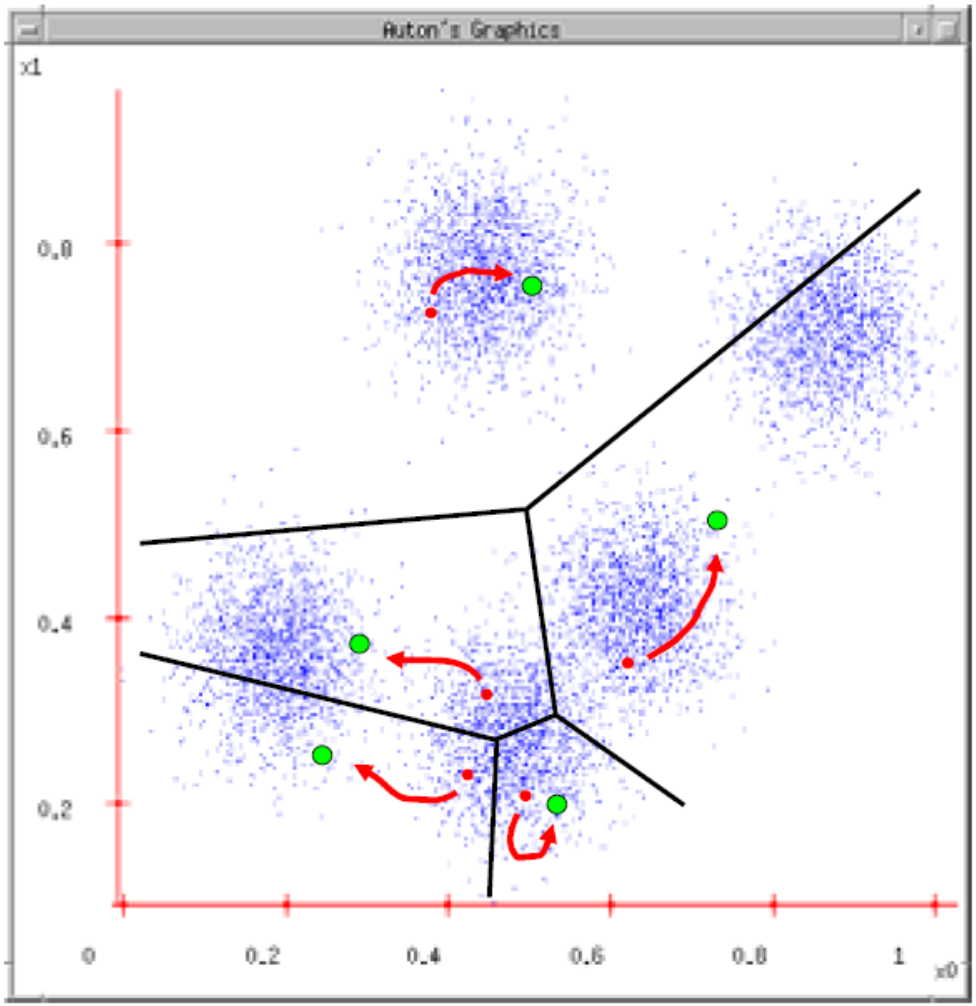
1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations
3. Each datapoint finds out which Center it's closest to. (Thus each Center "owns" a set of datapoints)



Source: A. Moore

K-means

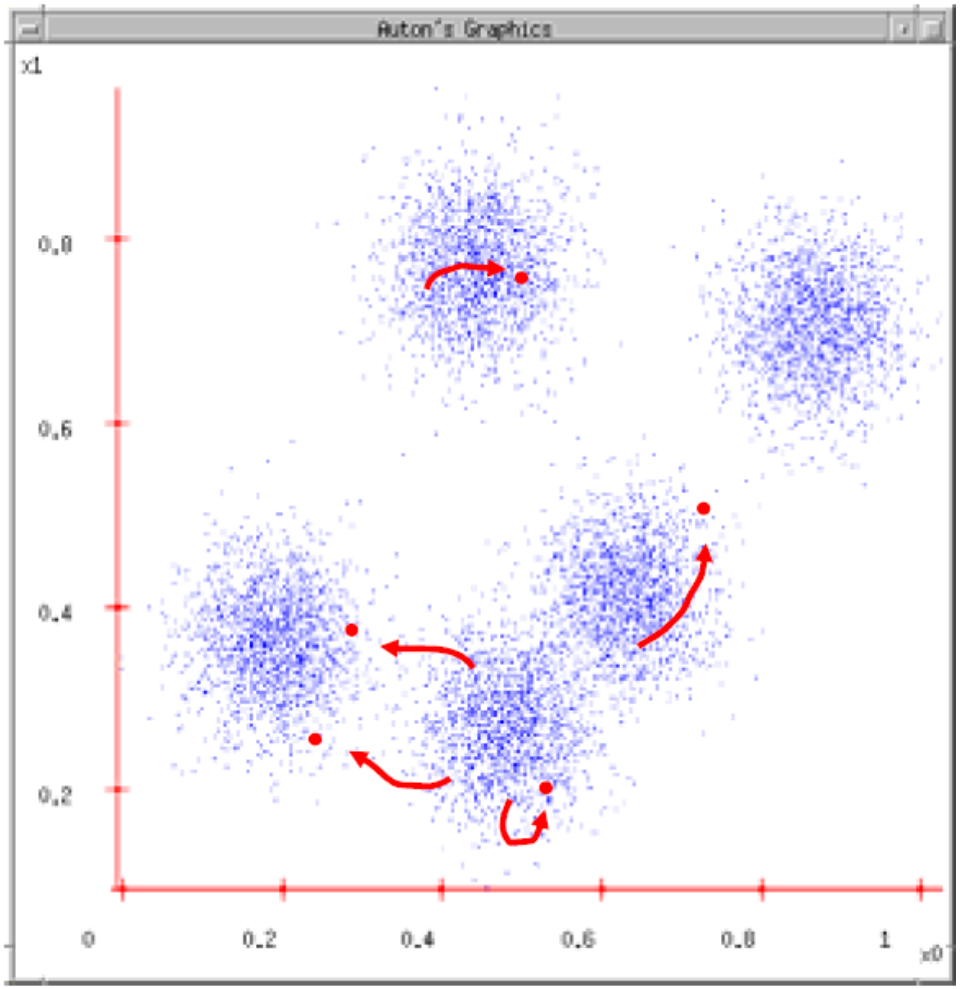
1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations
3. Each datapoint finds out which Center it's closest to.
4. Each Center finds the centroid of the points it owns



Source: A. Moore

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!



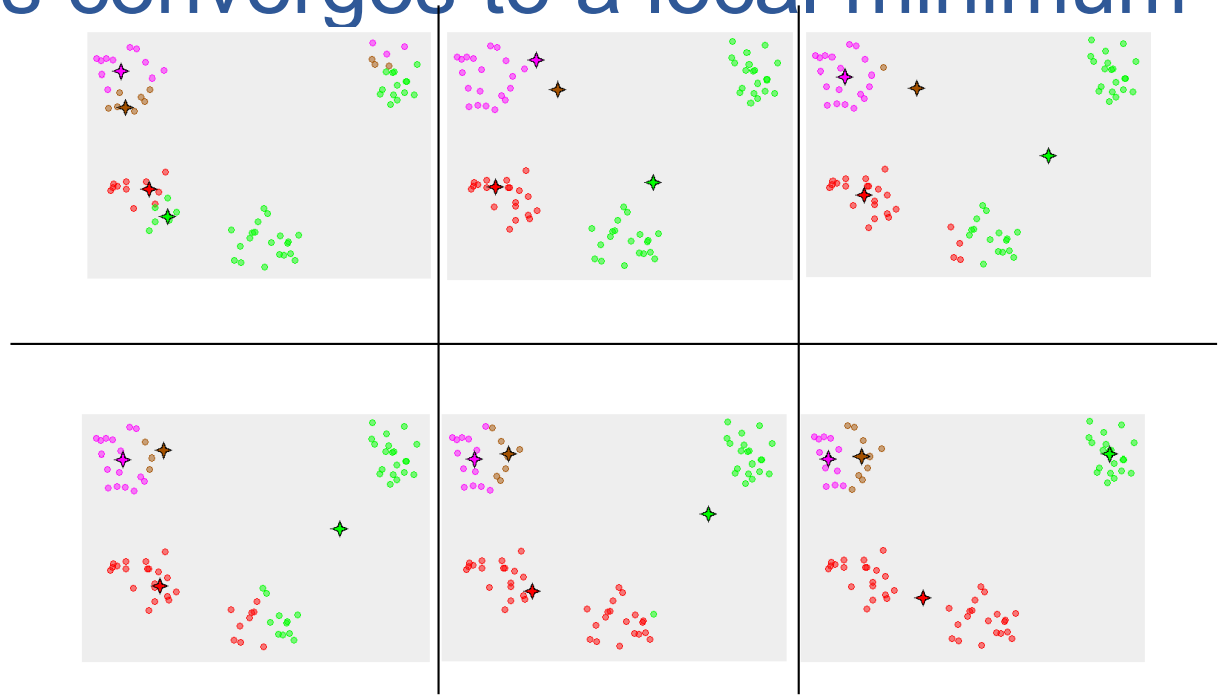
Source: A. Moore

Motivation: Interactive Clustering

https://www.naftaliharris.com/blog/visualizing-k-means-clustering/

Explained Visually: https://setosa.io/ev/

K-means converges to a local minimum



How can I try to fix this problem?

Adapted from James Hays

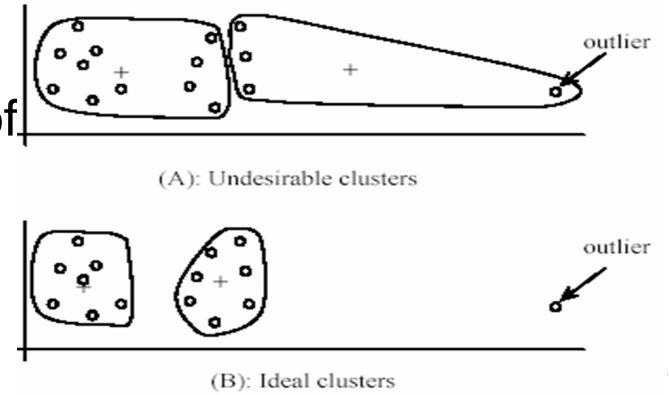
K-means: pros and cons

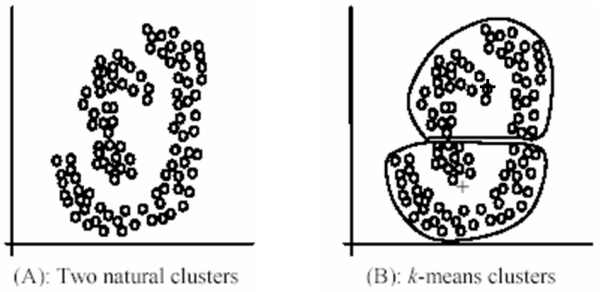
Pros

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

Cons/issues

- Setting k?
  - One way: silhouette coefficient
- Sensitive to initial centers
  - Use heuristics or output of another method
- Sensitive to outliers
- Detects spherical clusters





Adapted from K. Grauman

Segmentation as Clustering

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

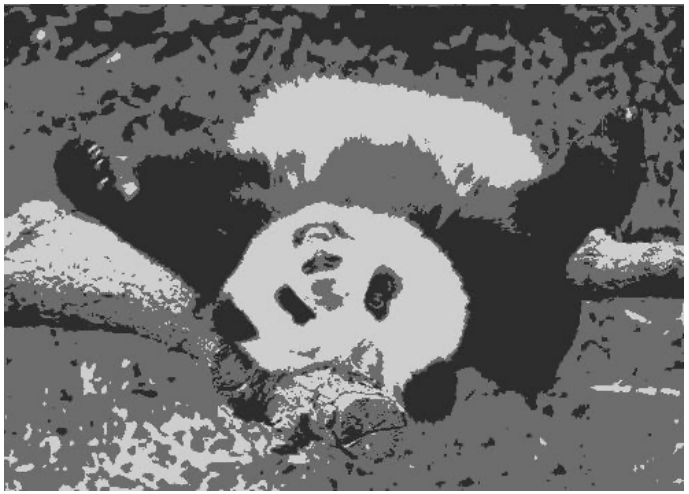
Grouping pixels based on **intensity** similarity

Feature space: intensity value (1-d)

Source: K. Grauman

K=2

K=3



Adapted from K. 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** similarity

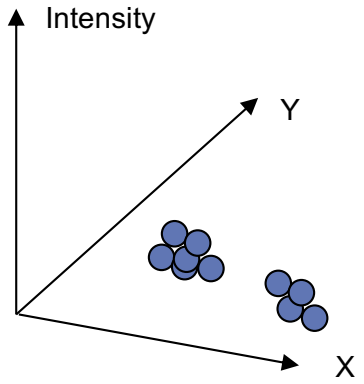
Clusters based on intensity similarity don't have to be spatially coherent.

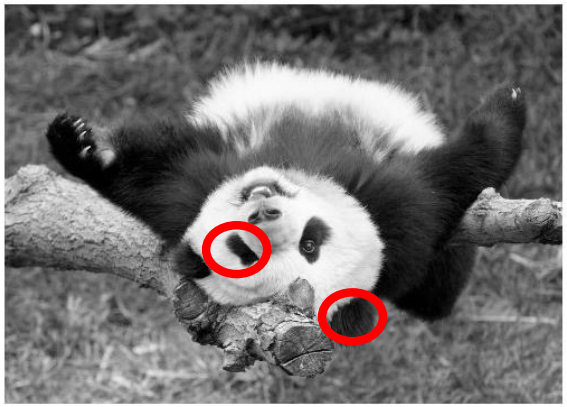
Source: K. Grauman

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





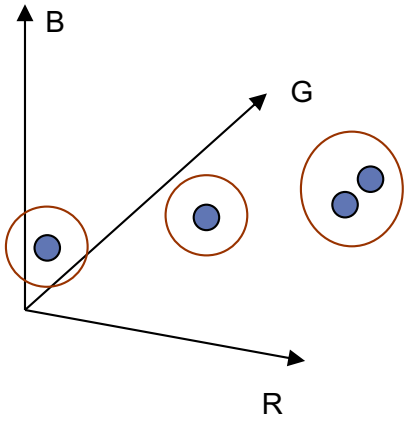
Both regions are black, but if we also include **position (x,y),** then we could group the two into distinct segments; way to encode both similarity & proximity.

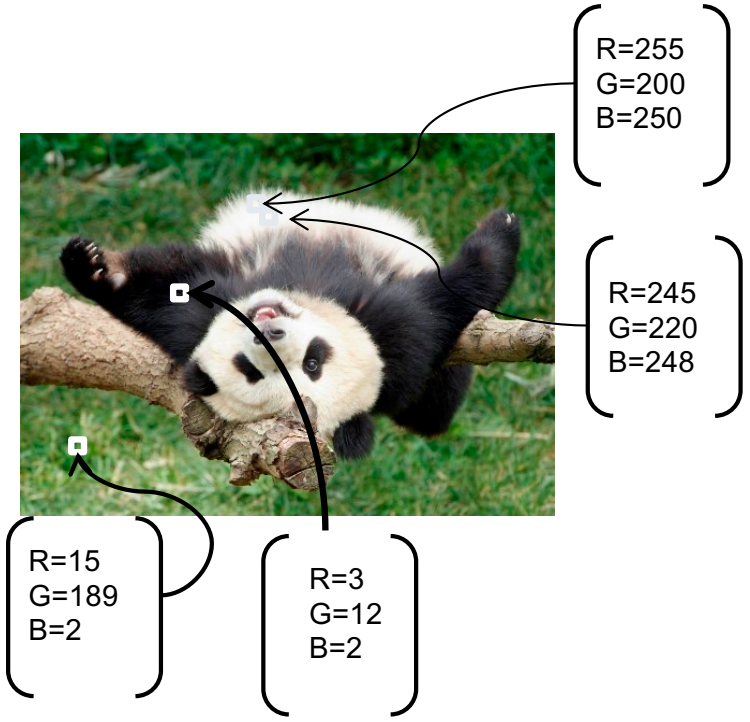
Source: K. Grauman

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



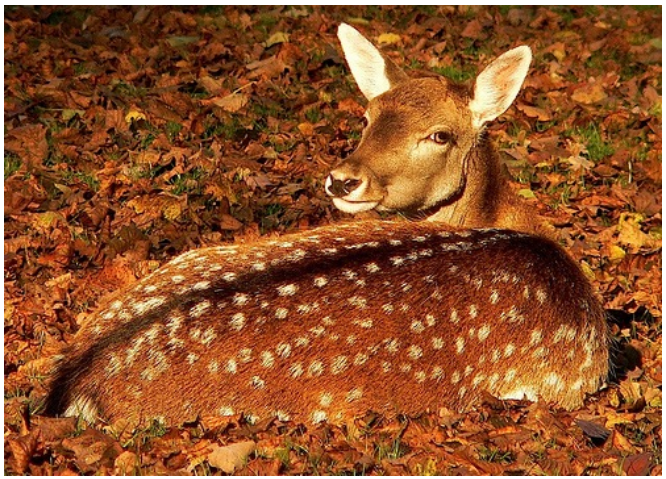


Feature space: color value (3-d)

Adapted from K. Grauman

Segmentation as Clustering

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

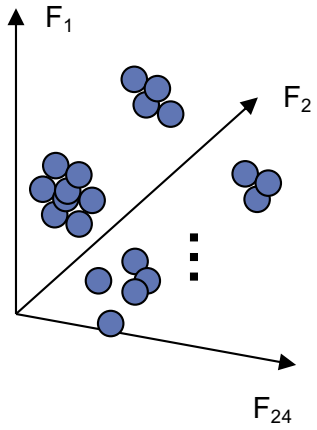


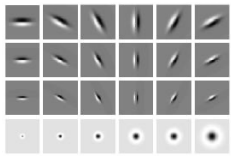
Source: L. Lazebnik

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



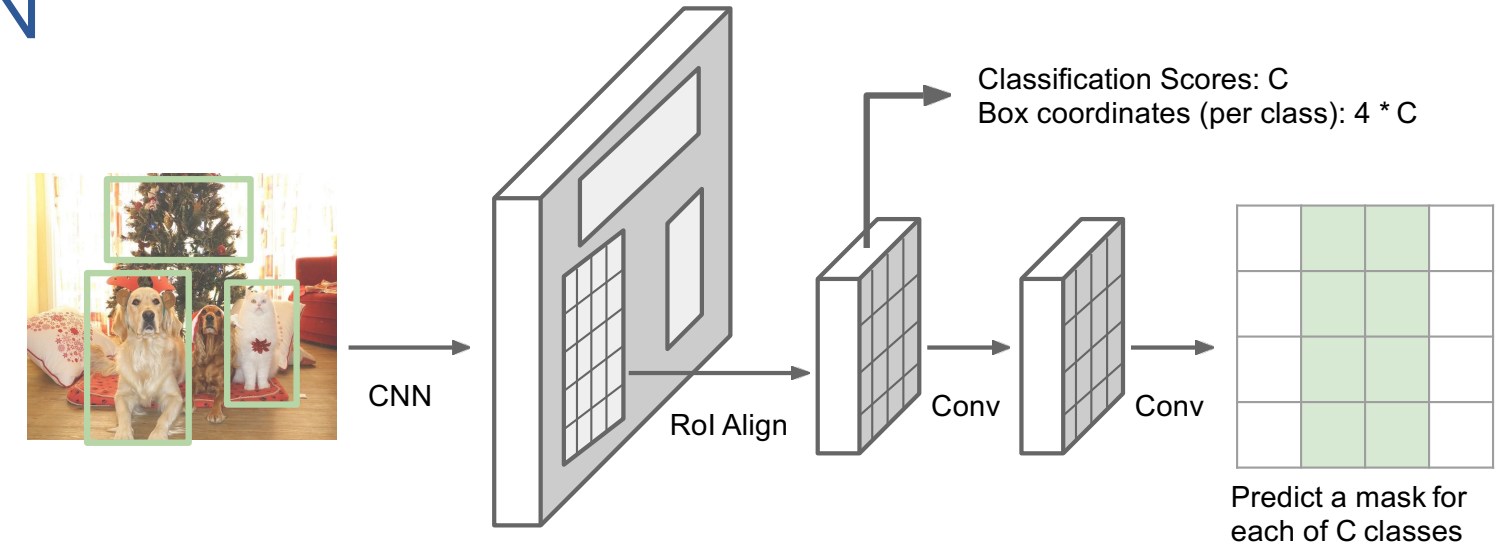


Filter bank of 24 filters

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

Source: K. Grauman

State-of-the-art (instance) segmentation: Mask R-CNN

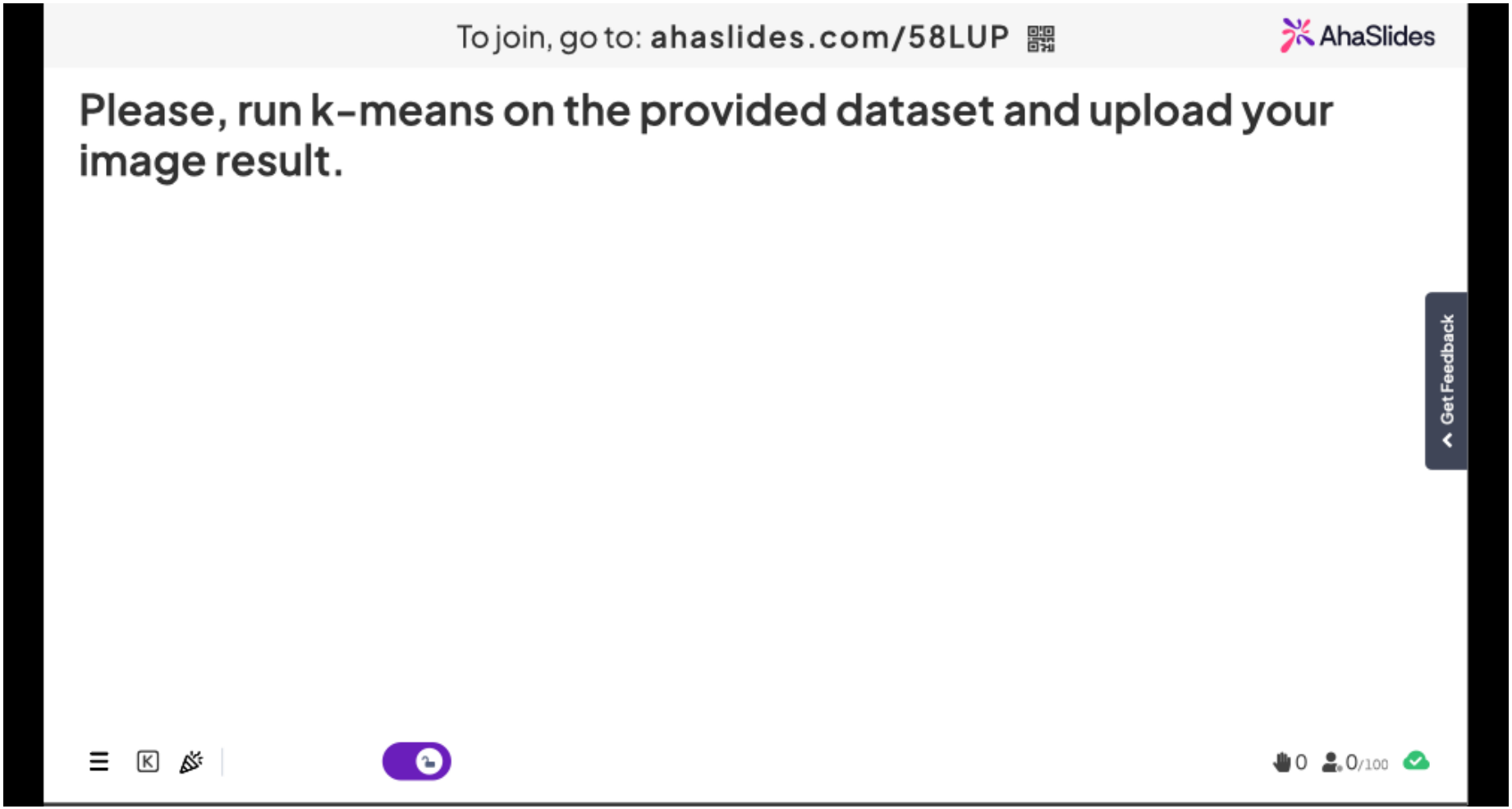


He et al, "Mask R-CNN", ICCV 2017; slide adapted from Justin Johnson

Lab 4: k-Means

Duration: 40 min

Use JPEG, PNG and GIF files less than 15 MB [ahaslides]


To join, go to: ahaslides.com/58LUP
AhaSlides
Please, run k-means on the provided dataset and upload your image result.
Get Feedback
0
0/100

Summary

- Segments: use clustering (e.g. K-means) to group pixels by intensity, texture, etc.