COMPUTER VISION 2018 - 2019

> CLUSTERING

UTRECHT UNIVERSITY

RONALD POPPE

OUTLINE

Concept

Clustering

Appearance models

Voxel-based labeling

Assignment

CLUSTERING

CONCEPT

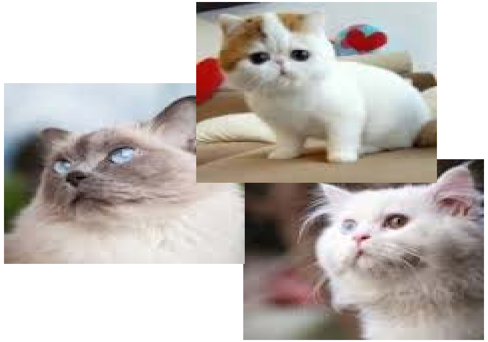
Clustering is the process of grouping similar items, items that belong together:

- Images
- Pixels
- Viewpoints
- Voxels
- Colors
- Etc.

What is similar?

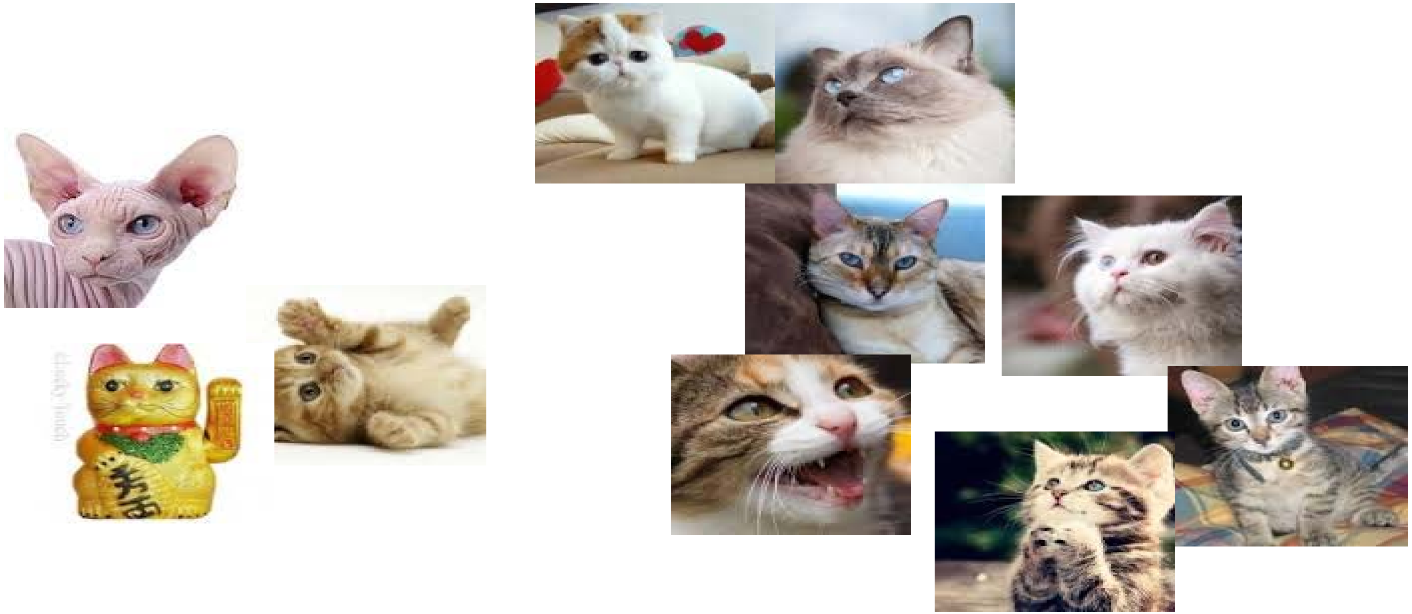
EXAMPLE

Cat photos clustered by species



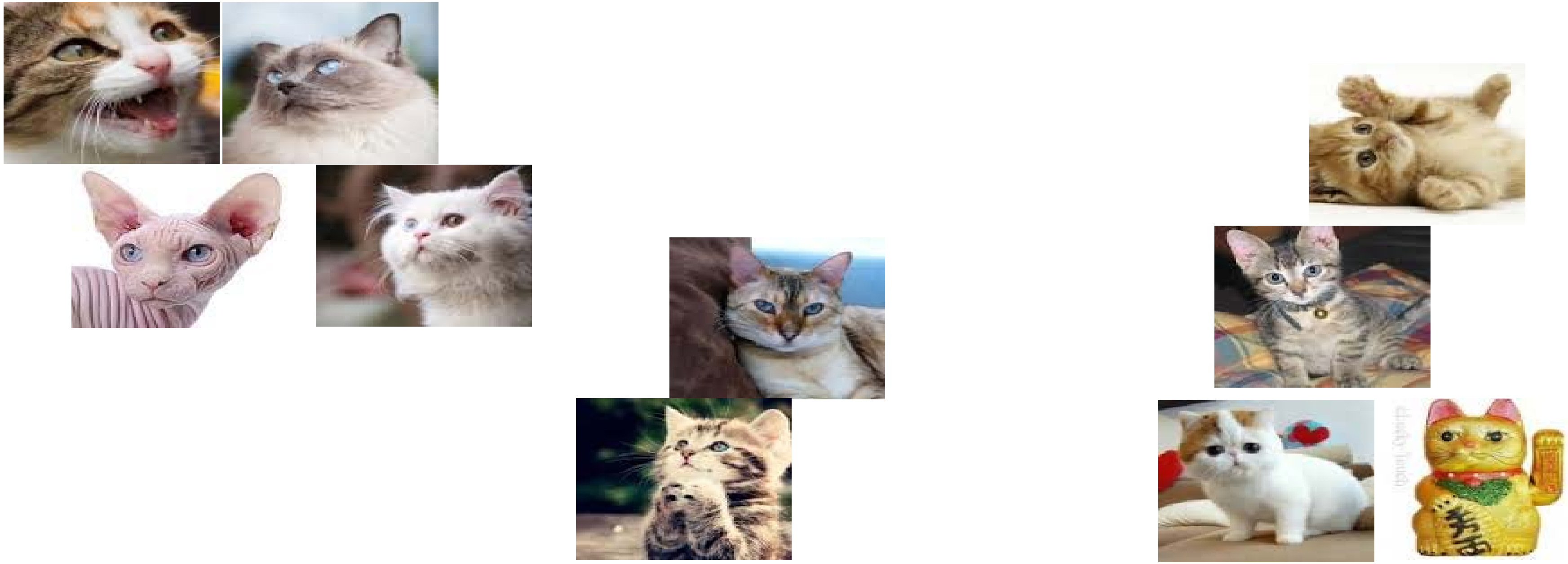
EXAMPLE[2]

Cat photos clustered by color similarity



EXAMPLE[3]

Cat photos clustered by relative viewpoint



DISTANCE FUNCTION

Clustering depends on our similarity measure

We operationalize similarity with a distance function $d(\mathbf{x1}, \mathbf{x2})$

- $\mathbf{x1}$ and $\mathbf{x2}$ are feature vectors corresponding to data samples
- d can be any arbitrary distance function

In general:

- More similar items have lower distance, $d(\mathbf{x1},\mathbf{x1}) = 0$
- Not always a maximum value for d

DISTANCE FUNCTION[2]

Feature vectors *x1* and *x2* consist of "*n* numbers that say something about a data sample"

- $\boldsymbol{x1}, \boldsymbol{x2} \in \mathbb{R}^n$, with *n* the dimensionality of the vectors

In our example, when *n=1*:

- ***x1*** and ***x2*** can be percentages of cat visibility

When *n*>1

- ***x1*** and ***x2*** can be a mean RGB color values (*n=3*)
- ***x1*** and ***x2*** can be a vector of pixel values (*n=3 * width * height*)

DISTANCE FUNCTION[3]

We have seen some distance functions *d* before:

- Euclidian distance
- Mahalanobis
- Manhattan distance

There are many more…

DISTANCE FUNCTION[4]

Distance functions can be combined:

- E.g. color and viewpoint

Range of distance values is important

- Different metrics can be weighted differently

Similarity functions *s* are conceptually the same as distance functions but

- Larger values correspond to more similar data points
- Range might be set to [0 – 1], with $s(\mathbf{x1}, \mathbf{x1}) == 1$

CLUSTERING ALGORITHMS

CLUSTERING ALGORITHMS

There are many algorithms. They vary in:

- Offline (determine once) vs. online (change over time)
- Computational performance (speed and memory)
- Number of clusters known in advance
- Etc.

We focus on K-means:

- Assumes that the number of clusters is known
- Is relatively fast

K-MEANS

Basic concept. We have:

- N data points in n-dimensional space
- K clusters with a center in n-dimensional space
- Process: simultaneously find the locations of the K clusters and the labeling of data points
- Goal: minimize total distance between data points and closest cluster center

Optimal solution is computationally expensive but:

- Heuristic algorithms exist that can do it fast(er)
- Solutions is not guaranteed to be best, but often quite good

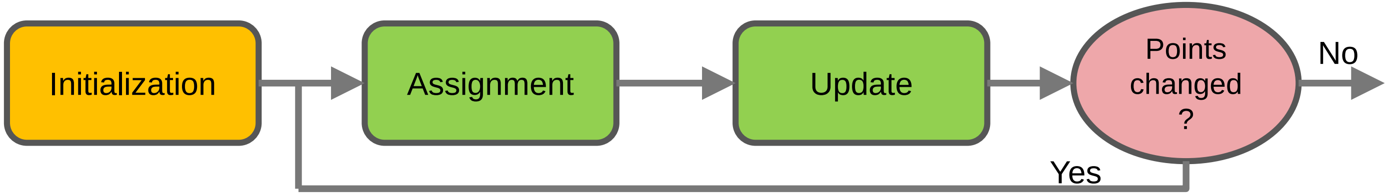
K-MEANS^2

We look at Expectation-Maximization/Minimization (EM)

- Iteratively loops through assignment and update steps
- Commonly used in computer vision algorithms

After initialization, the algorithm consists of two iterative steps:

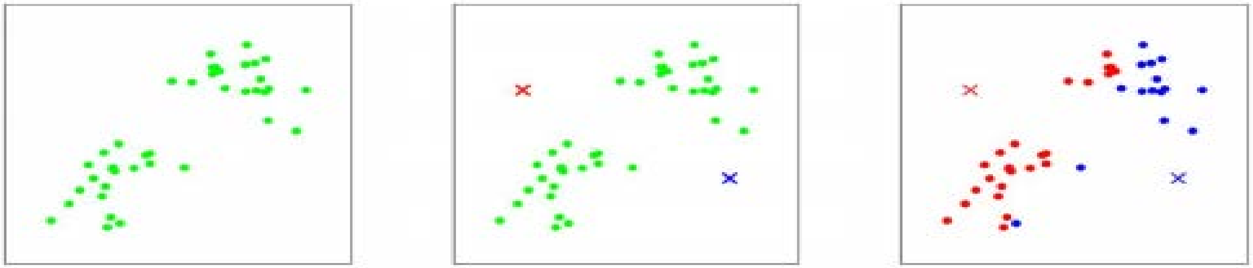
- Assignment: given the cluster centers, label all data points
- Update: move cluster centers to minimize total distance



K-MEANS[3]

Assignment:

- For each data point *i*, label it with the closest cluster center *c*
- Find the cluster center with smallest distance to data point
- Distance depends on application

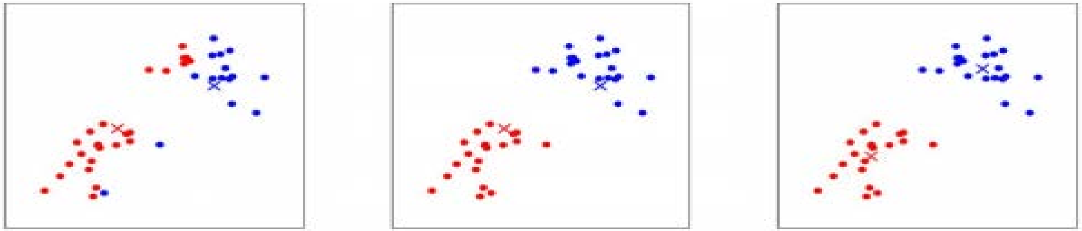


K-MEANS[4]

Update:

- For each cluster label, find all points that have the label
- The new cluster center is the mean of all these points
- So we minimize the total distance to the cluster

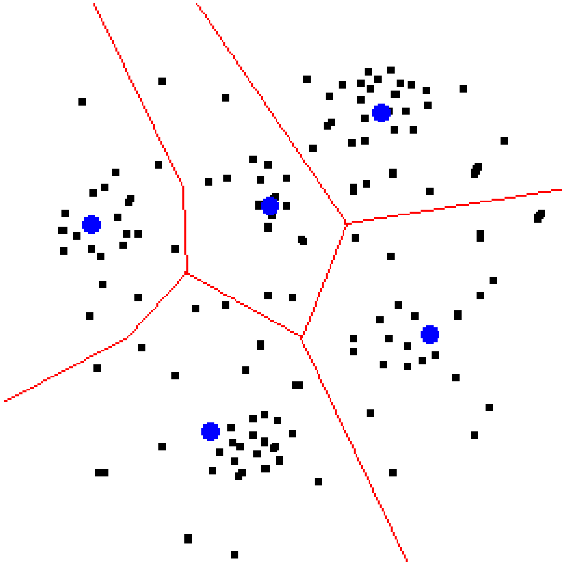
This results in some points being closer than the cluster they are labeled to → go the assignment step



K-MEANS[5]

The results of K-means are:

- A labeling of all points
- Cluster centers
- A partitioning of the space → Voronoi diagram



K-MEANS[6]

Process iterates over assignment/update steps until convergence

- Convergence reached when no data points change label
- Update step will then have same result as in previous step

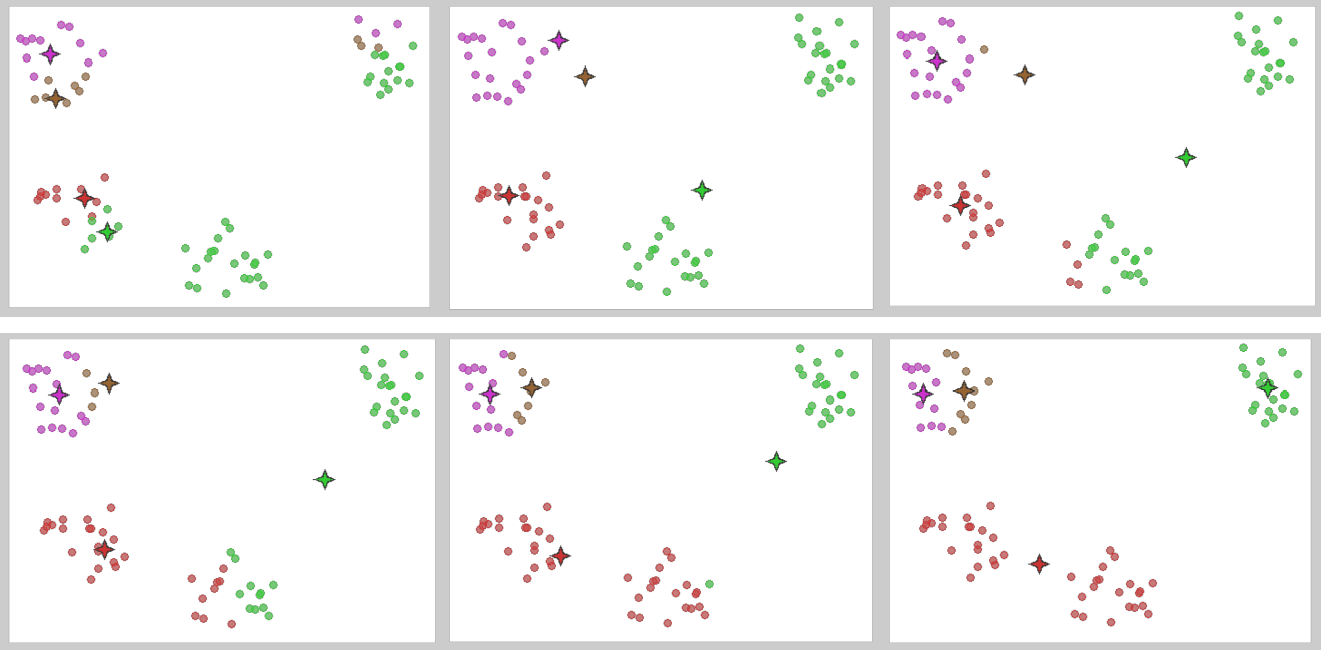
Several options of the initialization of the cluster centers

- Random cluster centers (range of data should be known)
- Random data points are cluster center
- Data points assigned a random cluster label, and then proceed to update step

K-MEANS[7]

Final clustering depends on initialization

- Risk of converging to local minimum



K-MEANS[8]

Online demo (in Matlab): http://www.ncdd.com.br/projetos/k-means-animation.rar

With missing function plota.m:

```
function plota(x,classe,colors)
    n = length(classe);
    figure(1);
    hold on;

    for i = 1 : n
        pos = classe(i);
        plot(x(i,1), x(i,2), '+', 'Color', colors(pos, :), 'MarkerSize',10);
    end
End
```

Or: http://gerardmeier.com/play/cluster-detection/

K-MEANS[9]

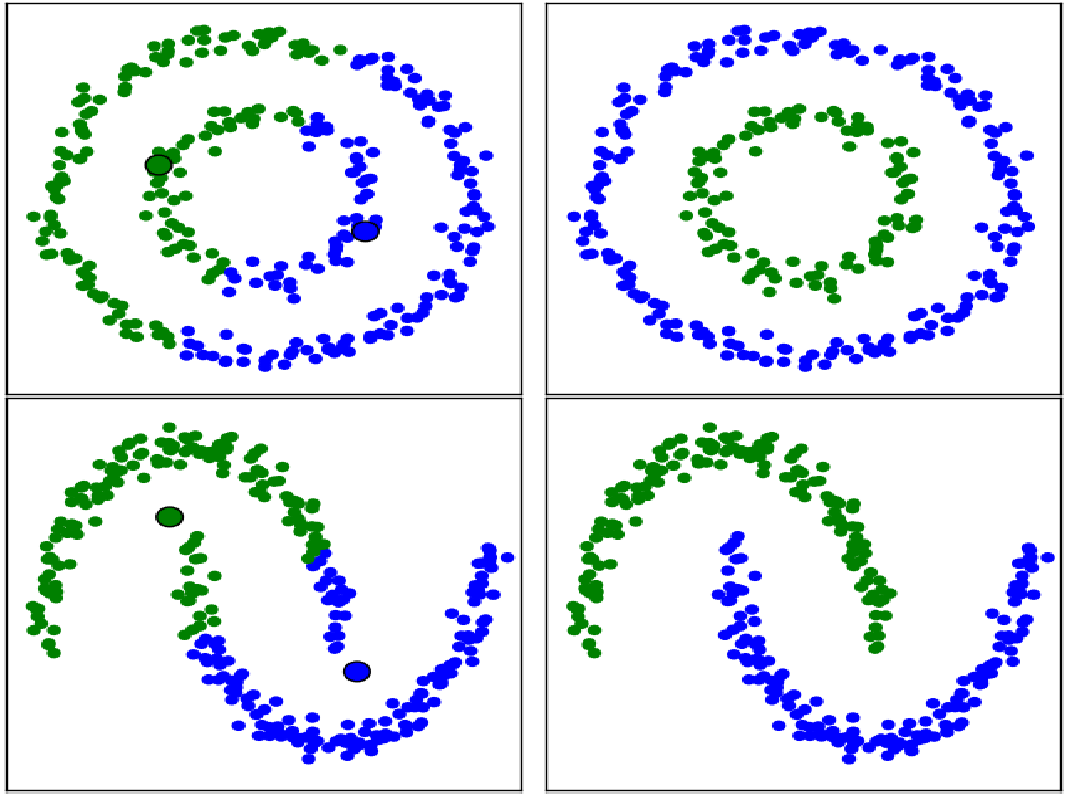
Advantages of K-means:

- Easy to understand
- Quickly converges

Limitations of K-means:

- Might get stuck in local minimum
- Each data point belongs to exactly one cluster, even though clusters might be very close
- Only works for continuous distance functions
- Structure of data points (e.g. connectivity) not taken into account

K-MEANS[10]



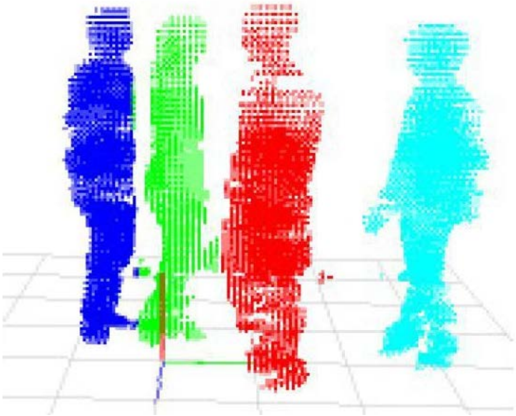
RECAP

Many different ways of clustering

- Some algorithms require number of clusters to be known
- Several distance measures can be used

In our example, we can cluster voxels based on their position

After the break, we look at color models



QUESTIONS?

APPEARANCE MODELS

COLORS AND COLOR SPACES

Colors can be described in different color spaces:

- RGB
- HSV
- CIELab

Most have three channels, 8 bits, so 2^{24} different colors per pixel

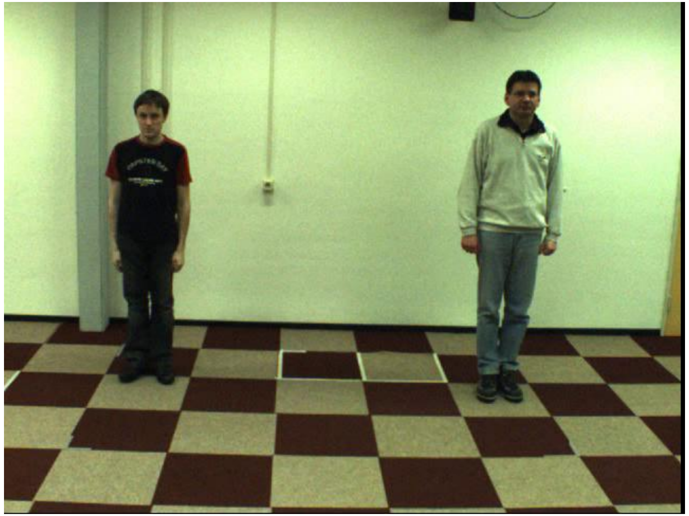
How to describe colors of an object conveniently?

- Single color (mean)
- List of all the colors, and how often they appear (histogram)
- Something in between?

COLORS AND COLOR SPACES[2]

Example:

- Two different subjects
- Segmented using background segmentation as mask



COLORS AND COLOR SPACES[3]

What are their mean RGB colors?

- Right: (71, 82, 61)
- Left: (29, 28, 22)
- What about the red sleeves or the blue jeans?

COLOR HISTOGRAMS

What if we would count how often a color appears?

- Table with 2^{24} rows
- Probably many colors that do not occur at all
- Very similar colors in different rows (“bins”)

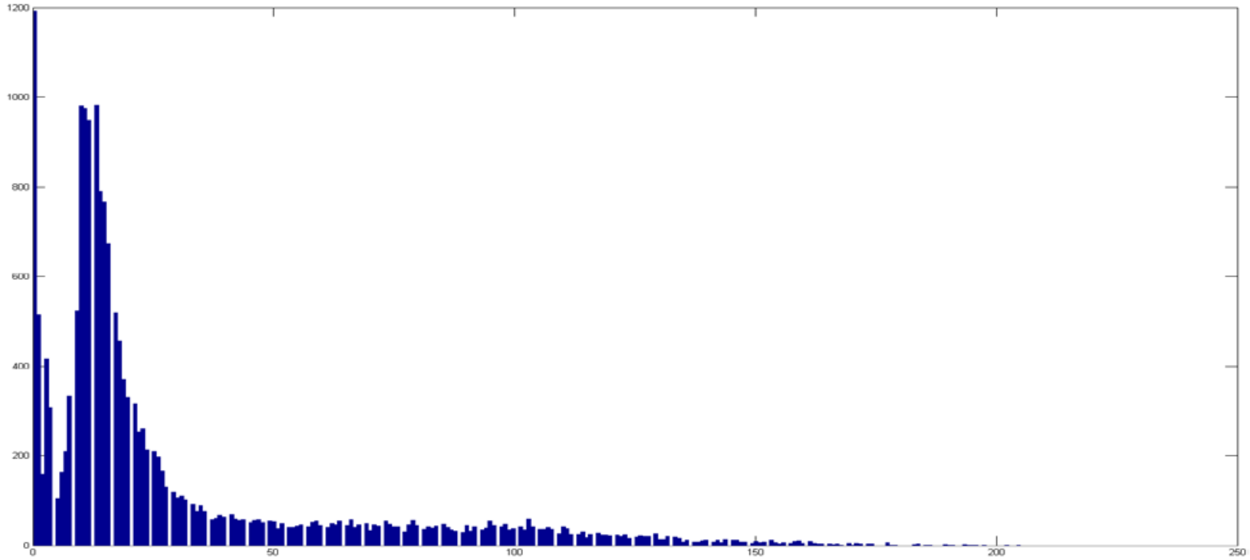
What if we look at a single channel?

- Determine how often a value appears
- Only 2^8 (256) possibilities per channel, so 3 x 256 in total

COLOR HISTOGRAMS[2]

Can be visualized by a histogram

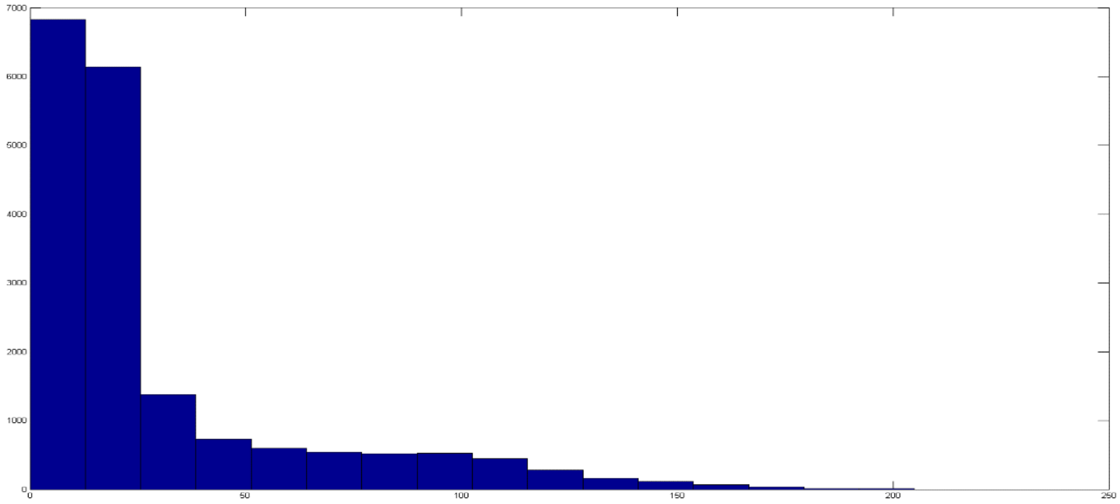
- X-axis corresponds to values
- Y-axis corresponds to number of pixels



COLOR HISTOGRAMS[3]

256 values (bins) is still quite a lot

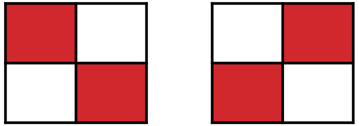
- What about fewer bins?
- Group/cluster neighboring bins
- Typically equidistantly spaced (0-15, 16-31, 32- …)

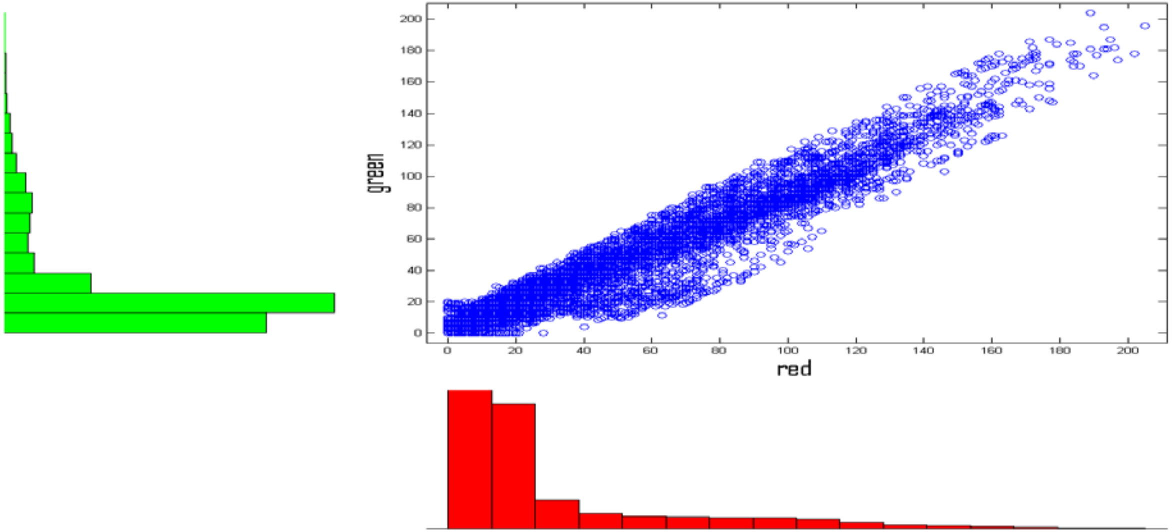


COLOR HISTOGRAMS[4]

What about multiple channels?

- Independent channels: 3 x 16 bins
- Not clear how colors are correlated

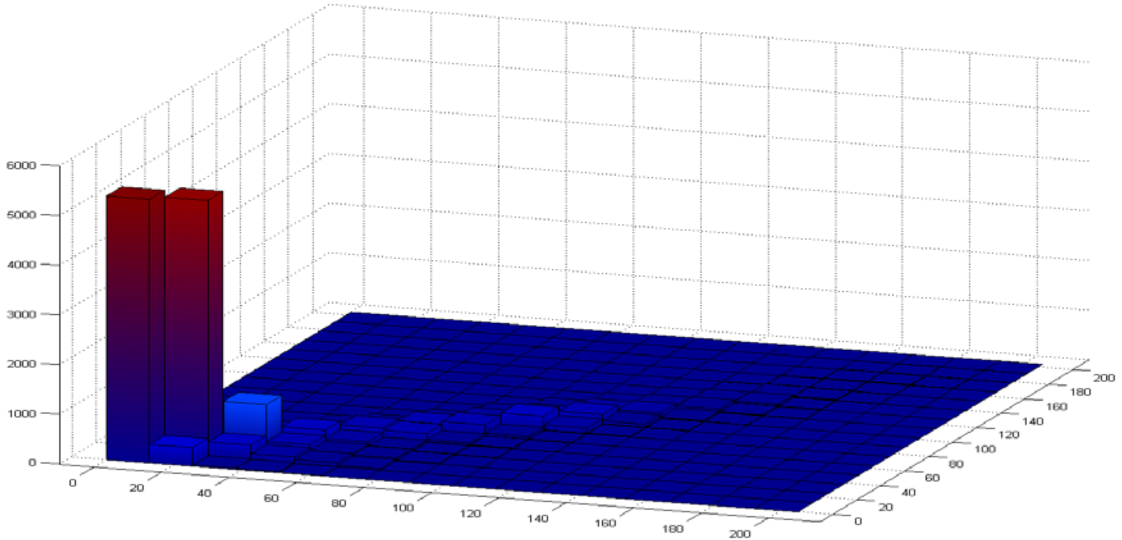




COLOR HISTOGRAMS[5]

What about dependent channels?

- 16 x 16 x 16 bins
- Correlation can be modeled, but more bins needed



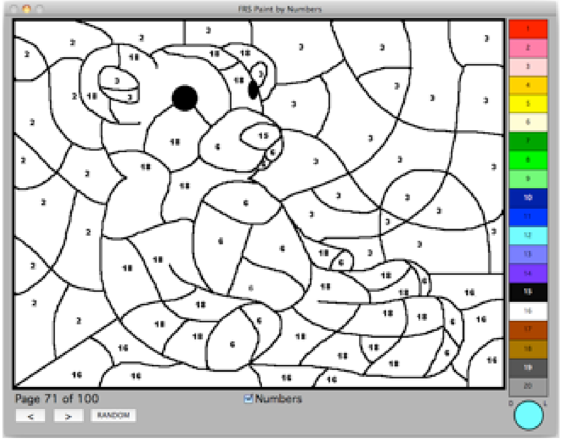
COLOR HISTOGRAMS[6]

Instead of equidistantly sampled bins

- Use only common or discriminative colors

We can find those by clustering

- Colors correspond to cluster centers
- All cluster centers together are termed a codebook
- Can be considered a table of all relevant colors



COLOR HISTOGRAMS[7]

If we want to compare images using histograms

- We need a distance function between two histograms
- Histograms depend on the number of elements in each bin

Normalization is required

- Bins together sum up to one
- Divide all bins by total sum of bins

Histograms can be compared in many ways

- Chi-squared (χ^2) is convenient

COLOR HISTOGRAMS[8]

Chi-squared distance between histograms *x1* and *x2* with *n* bins:

$$\chi^2(\boldsymbol{x1}, \boldsymbol{x2}) = \sum_{1}^{n} \frac{(\boldsymbol{x1}_i - \boldsymbol{x2}_i)^2}{(\boldsymbol{x1}_i + \boldsymbol{x2}_i)}$$

We can now compare histograms of any kind

- As long as bins correspond to the same colors/clusters
- They have the same number of bins
- Histograms are normalized

GAUSSIAN MIXTURE MODELS

So we can model a set of colored pixels by:

- Their mean color (*n=3*)
- A histogram of colors ($n = 2^{24}$, or much lower)

What about a middle way?

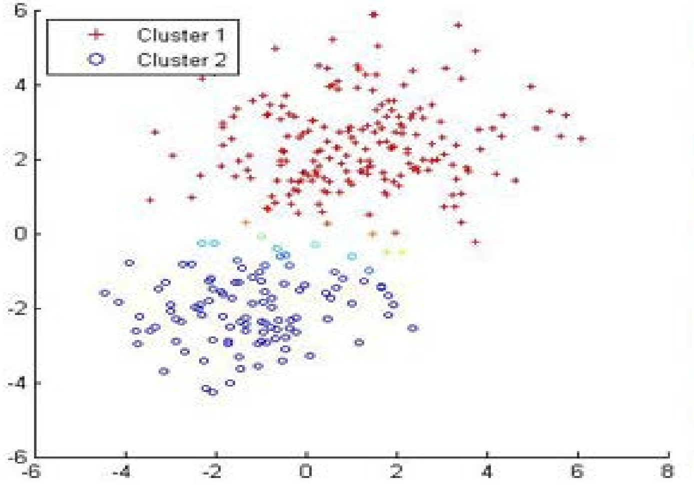
- Model colors as a multi-modal distribution
- Multi-modal means there are several “peaks”

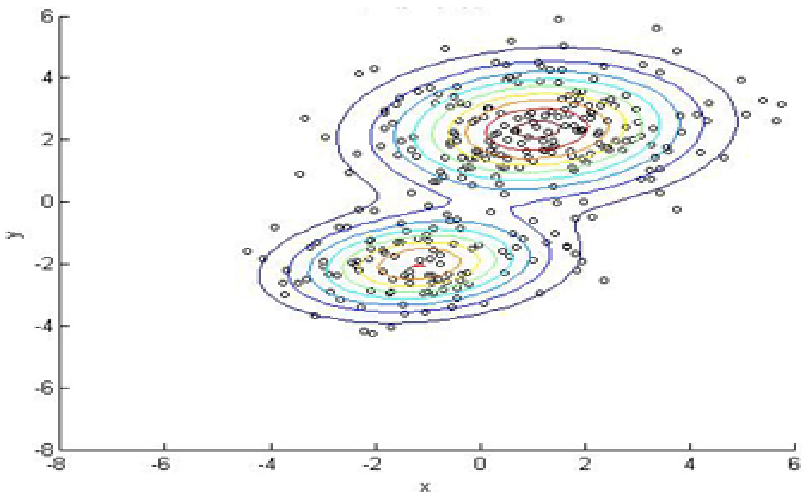
When each distribution is considered a normal distribution:

- Gaussian mixture models additionally model the (co)variance

GAUSSIAN MIXTURE MODELS[2]

- More efficient than a histogram (only mean and (co)variance)
- Less precise than a histogram
- Can be in any number of dimensions/channels

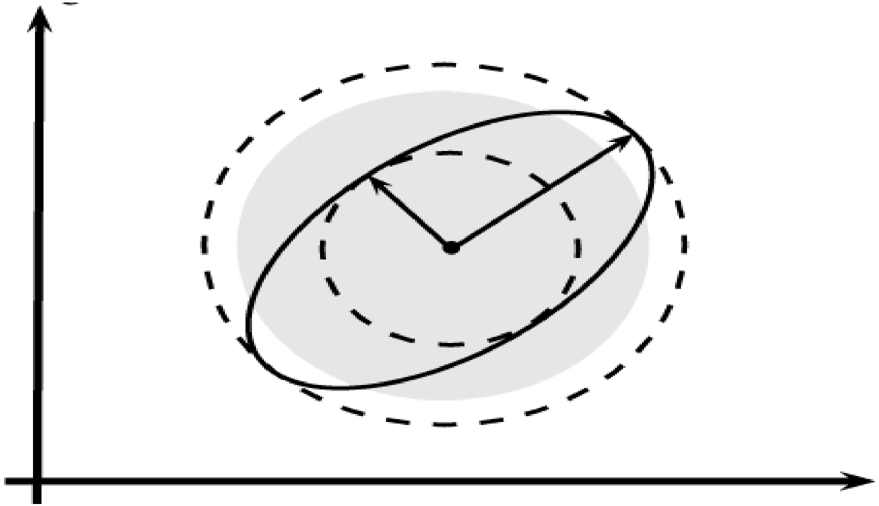




GAUSSIAN MIXTURE MODELS[3]

Each Gaussian (cluster) has a:

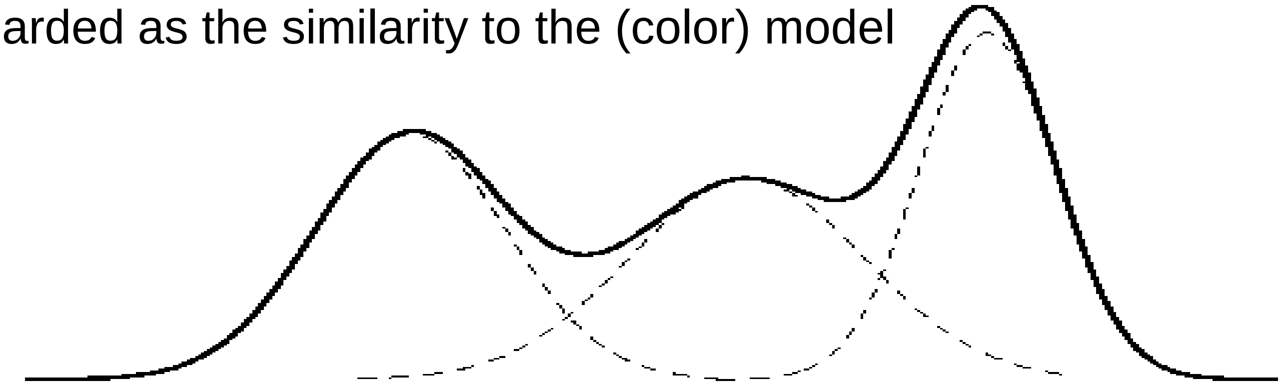
- Mean: determines the location of the cluster
- (Co)variance matrix: determines how peaked the cluster is, and in which direction it is oriented



GAUSSIAN MIXTURE MODELS[4]

All Gaussians together form a probability density function (PDF)

- Each Gaussian can have a weight that determines importance, e.g. higher for clusters corresponding to colors that occur often
- Weights sum up to one
- Value under the function determines how likely that point is generated by the Gaussian mixture model
- It can also be regarded as the similarity to the (color) model



GAUSSIAN MIXTURE MODELS[5]

Advantages of Gaussian mixture models:

- Compact representation of range of colors
- Smooth distance function (missing colors are “interpolated”)
- Distance can be calculated efficiently (Mahalanobis)

Disadvantage:

- When colors do not form a nice area, many Gaussians might be needed

RECAP

A set of colors can be modeled as:

- Single mean
- Gaussian mixture model
- Histogram

Each of these can be determined in (a subset of) 1, 2 or 3 channels

Each of these can be determined in different color spaces

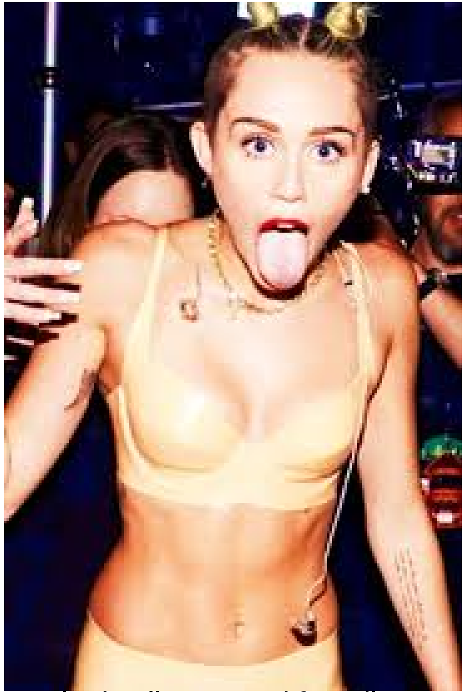
QUESTIONS?

QUESTIONS FOR YOU!

What kind of color model would you use for…

GMM (2 Gaussians) will do for Lady Gaga

Histogram with many bins for Nicki Minaj



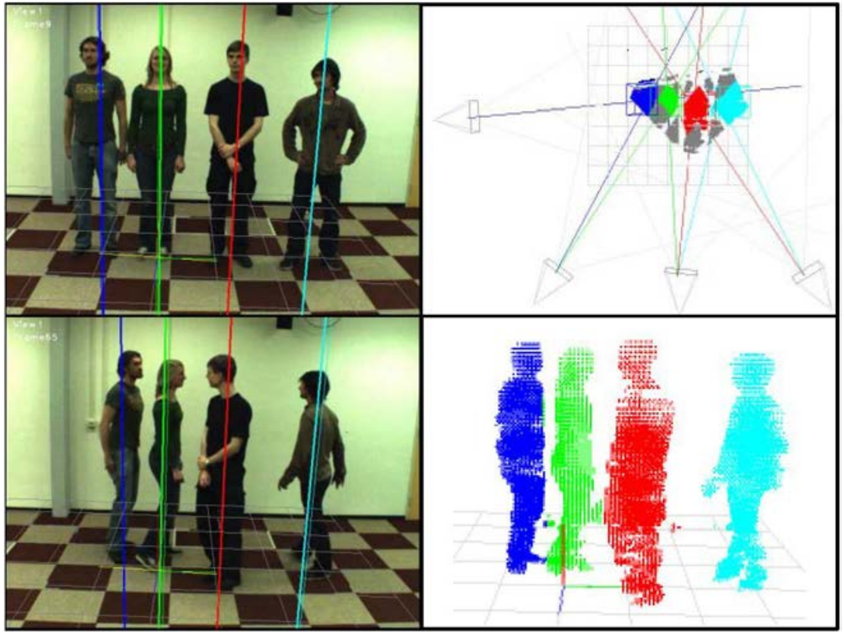
Mean color is all you need for Miley Cyrus

VOXEL-BASED LABELING

VOXEL-BASED LABELING

Determine which voxel belongs to which object/subject

- Color indicates the subject identity



FIRST ANALYSIS

All similarly labeled voxels form a cluster

In 3D, these voxels are close together

- Height does not play a role → ignore it?
- Cluster voxels based on distance to each other?

In an image, voxels can be projected to the different subjects

- Each subject might be wearing differently colored clothing
- Cluster pixels based on colors?

We can take into account both cues

CONCEPT

For our voxel-based tracking, we use two types of input:

- The voxel model
- The 2D images from each view

We'll use everything we've learned in this lecture:

- Clustering of voxels (3D)
- Color models (image)

CONCEPT^2

Initialization (offline):

1. Cluster voxels into persons using only voxel locations
2. Make a color model for each person

For each next frame (online):

3. Cluster these voxels to form the new person locations
4. Label voxels to persons using color models

INITIALIZATION

When clustering the voxels into persons:

- Choose a frame where everyone is visible and well-seperated
- The number of clusters is equal to the number of subjects
- Run a clustering algorithm based on the location of the voxel
- Ignore the height!
- Check if clusters are not really close to each other (or if the distance of clusters to center is large)

Output:

- Cluster centers corresponding to the 2D location of each person (on the ground plane)
- A label of each voxel to which person it belongs

INITIALIZATION^2

Next step is to make a color model for each person

We need to know which pixels belong to which person:

- Project the labeled voxels to view (frontal view)
- Make sure that the pixels are visible (occlusion!)

Construct the color model (per view or all views together):

- Histogram
- GMM
- Mean color (will not give you the full points)

INITIALIZATION^3

You might want to make a smart color model:

- Use only the lower/upper part?
- Discard dark pixels?
- Etc.

ONLINE PROCESSING

The online processing deals with labeling the voxels in subsequent frames

We will use color cues!

- Normally, you would track the position, based on the previous position and an estimate of the movement
- We call this tracking

ONLINE PROCESSING[2]

First, cluster the voxels. Each cluster now (hopefully) corresponds to a person, but we do not know which (because of the random initialization).

Then, find out which cluster belongs to which person:

- Project voxels of one cluster to one (or more) 2D image(s) (occlusion!)
- Determine the color of the pixels
- Use a suitable measure for distance between pixels and model

ONLINE PROCESSING[3]

Determine the position of each subject:

- Outliers have quite a large impact on the estimation
- Optionally: iteratively filter out outliers
- Optionally: if you can improve the tracks, you can earn additional points

ASSIGNMENT

ASSIGNMENT

Assignment 2 due Sunday 23:00

Assignment 3:

- You will track people based on voxels and color models
- Builds on Assignment 2 and the topics of this lecture

If you get stuck:

- Ask Breixo: b.solinofernandez@students.uu.nl
- Join Slack: https://join.slack.com/t/infomcv2019/signup

NEXT LECTURE

Image features

- How to describe images in a convenient way

Next Tuesday 13:15-15:00, BESTUURS-LIEREGG

QUESTIONS?