THE HONG KONG POLYTECHNIC UNIVERSITY
香港理工大學

Image Retrieval – COMP4423 Computer Vision

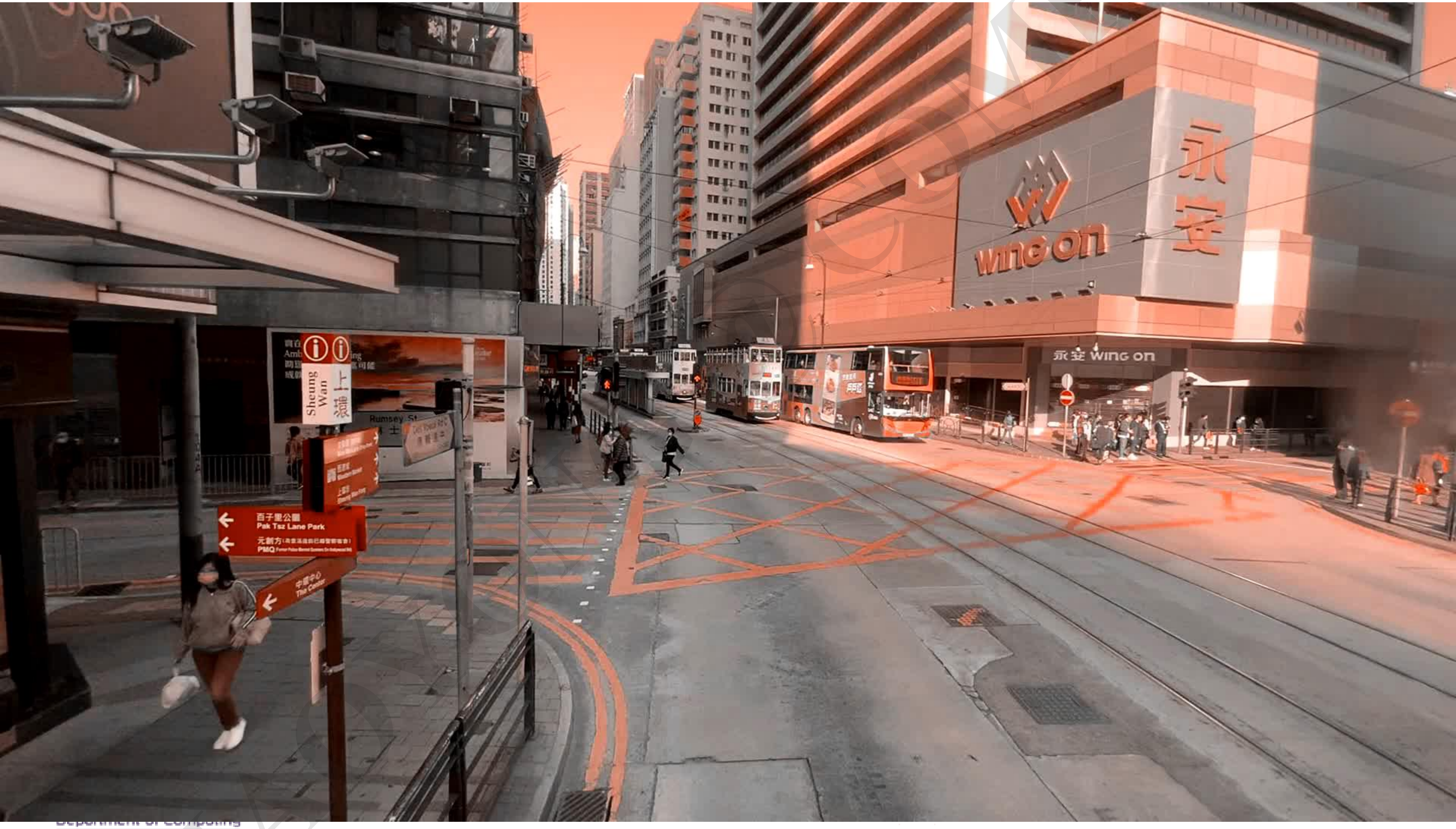
Xiaoyong Wei (魏驍勇)

x1wei@polyu.edu.hk

Department of Computing
電子計算學系

Opening Minds • Shaping the Future
啟迪思維・成就未來

New Toy


Sheung Wan
上環
Rumsey St
百子里公園
Pak Tsz Lane Park
元創方
PMQ
中環中心
The Center
永安
wing on
永安 wing on

New Toy

永安
wing on
永安 wing on
Ambassador Banking
2842 6008
Wing On
Ngan House
ubon Bank
富邦銀行
Jetco
BANK OF CHINA

Outline

- >Clustering
- >K-Means
- >Content-based image retrieval (CBIR)
- >Bag of Visual Words (BoVW)

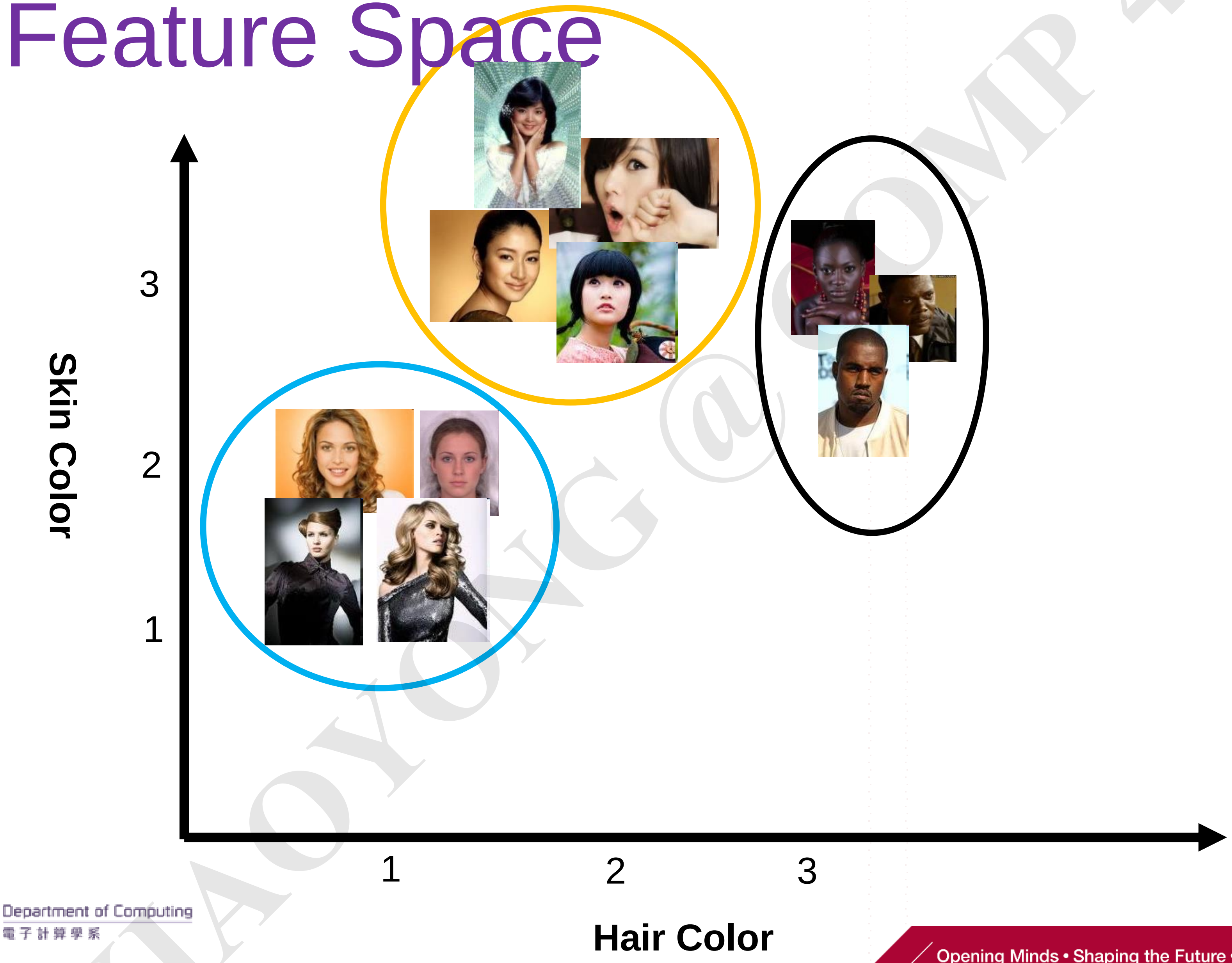
In Feature Extraction, we teach the computers to represent the “content” of the images.

How can we use these “content”?

Recall where we started

How do you group them?

Feature Space



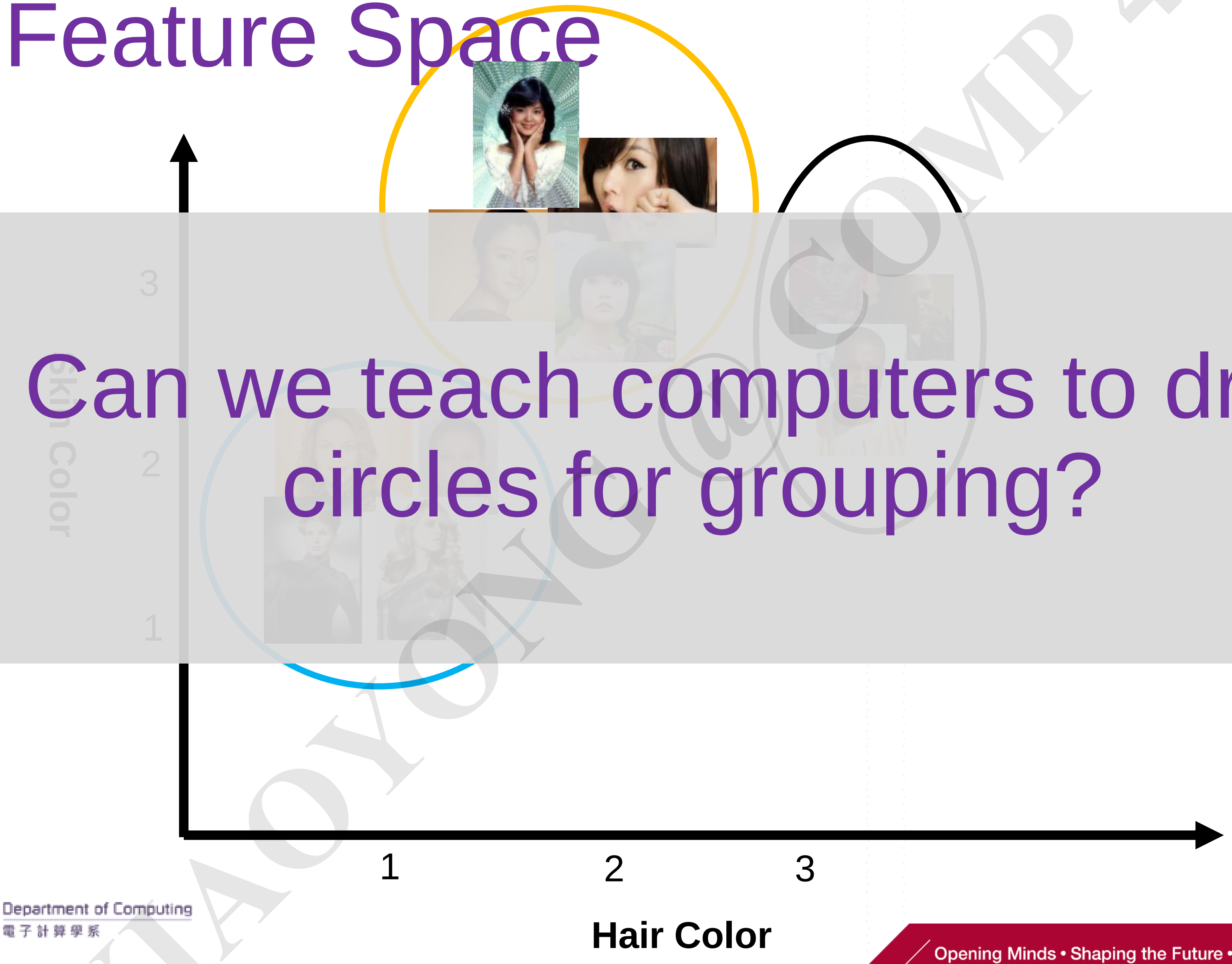
Images become “numbers” (feature vectors) in the **feature space** after the feature extraction.

To use the “content” is to play those “numbers”. This holds to nearly everything we’re going to learn in the rest of this course.

In a more general sense, this applies to texts, audios, videos, and a wide range of other media/information.

We’re all trying to represent things as “numbers” in the feature space, making them “readable” for computers.

Feature Space



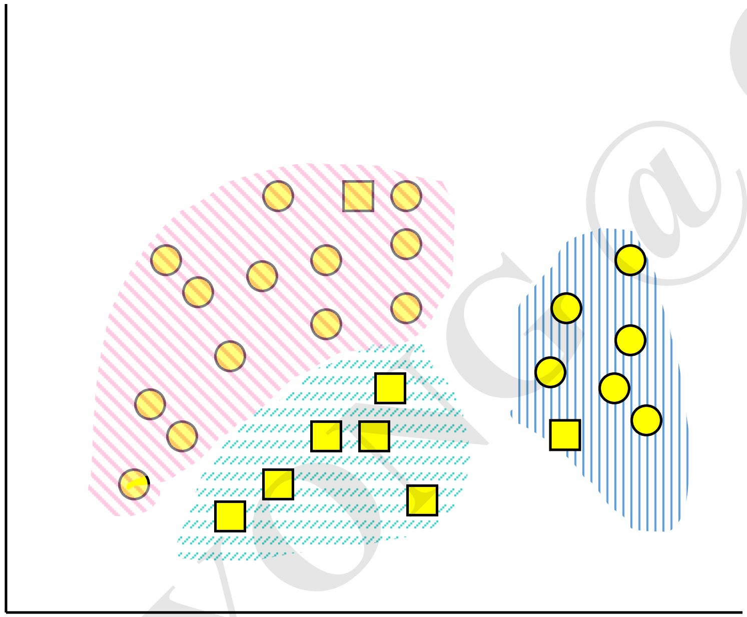
Can we teach computers to draw circles for grouping?

We can answer the question now.

To group the images is to make the circles as far as possible from each other, while the images inside the same circle as close as possible.

This is the idea of **clustering**: to maximize the **inter-cluster** distance while to minimize the **intra-cluster** distance.

Think in the feature space

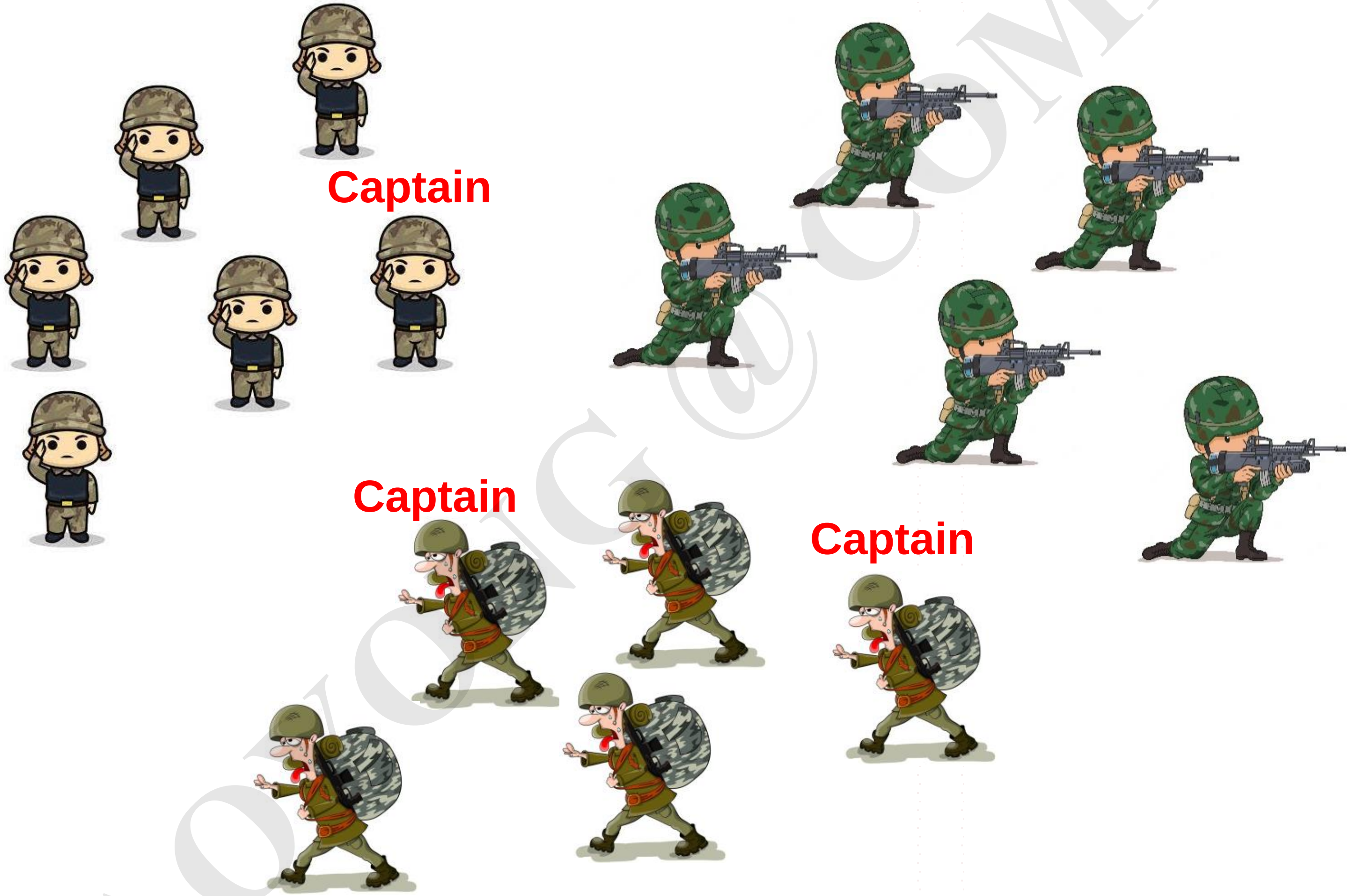


Think in the feature space

> K-Means

- 1: Pick a number (K) of cluster centers (at random)
- 2: Assign every item to its nearest cluster center (e.g. using Euclidean distance)
- 3: Move each cluster center to the **mean** of its assigned items
- 4: Repeat steps 2,3 until convergence (change in cluster assignments less than a threshold)

1. Captains assigned randomly

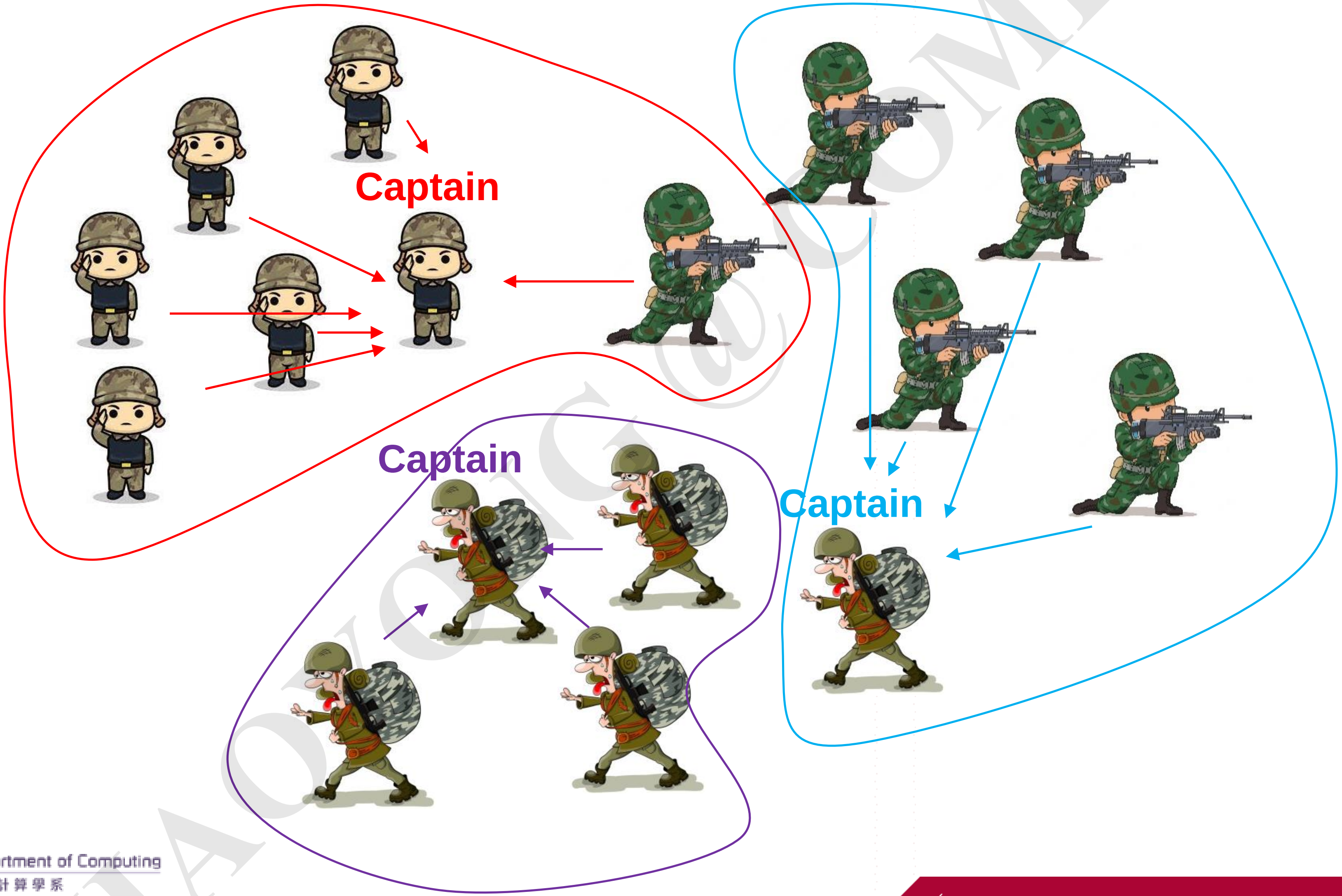


Captain

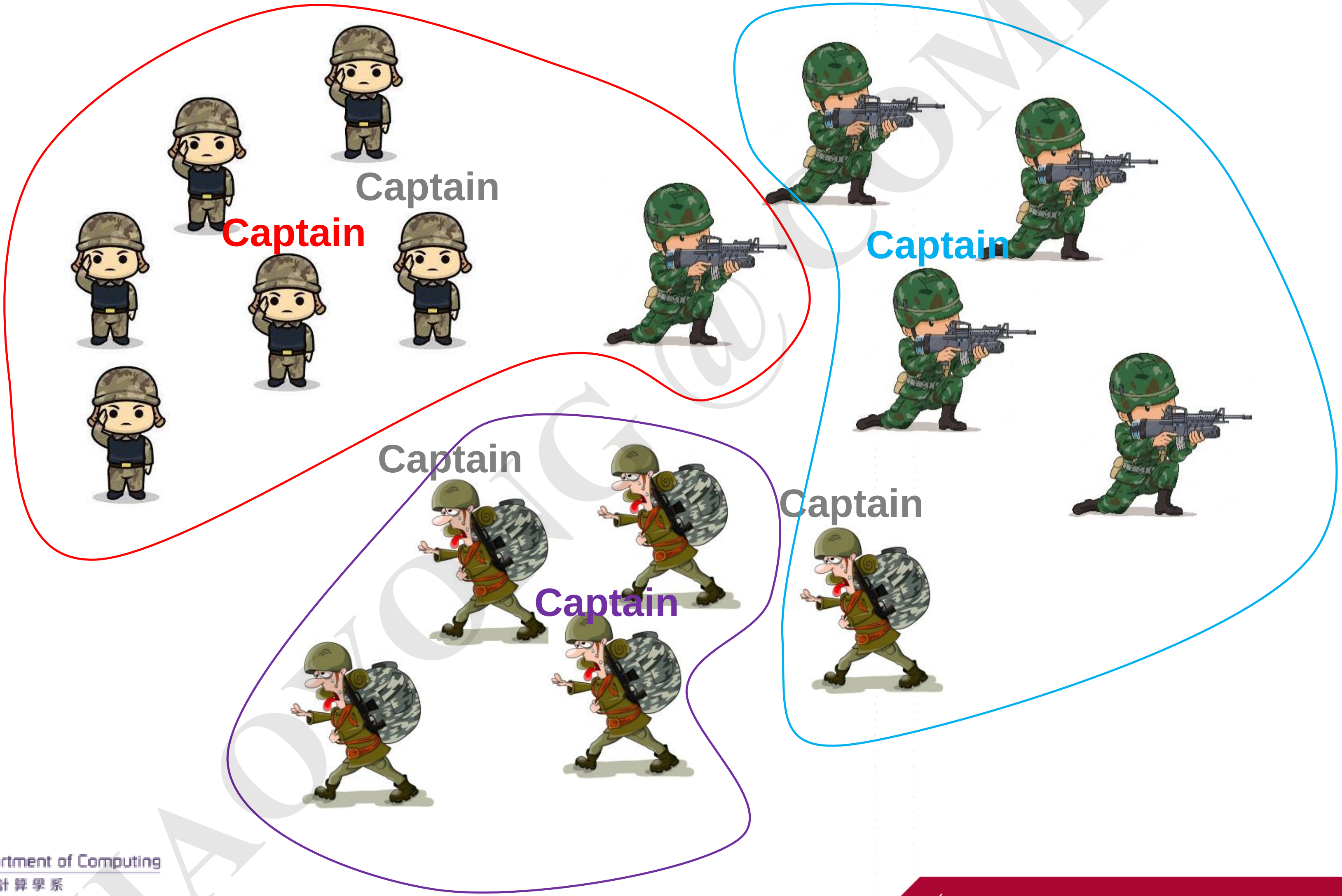
Captain

Captain

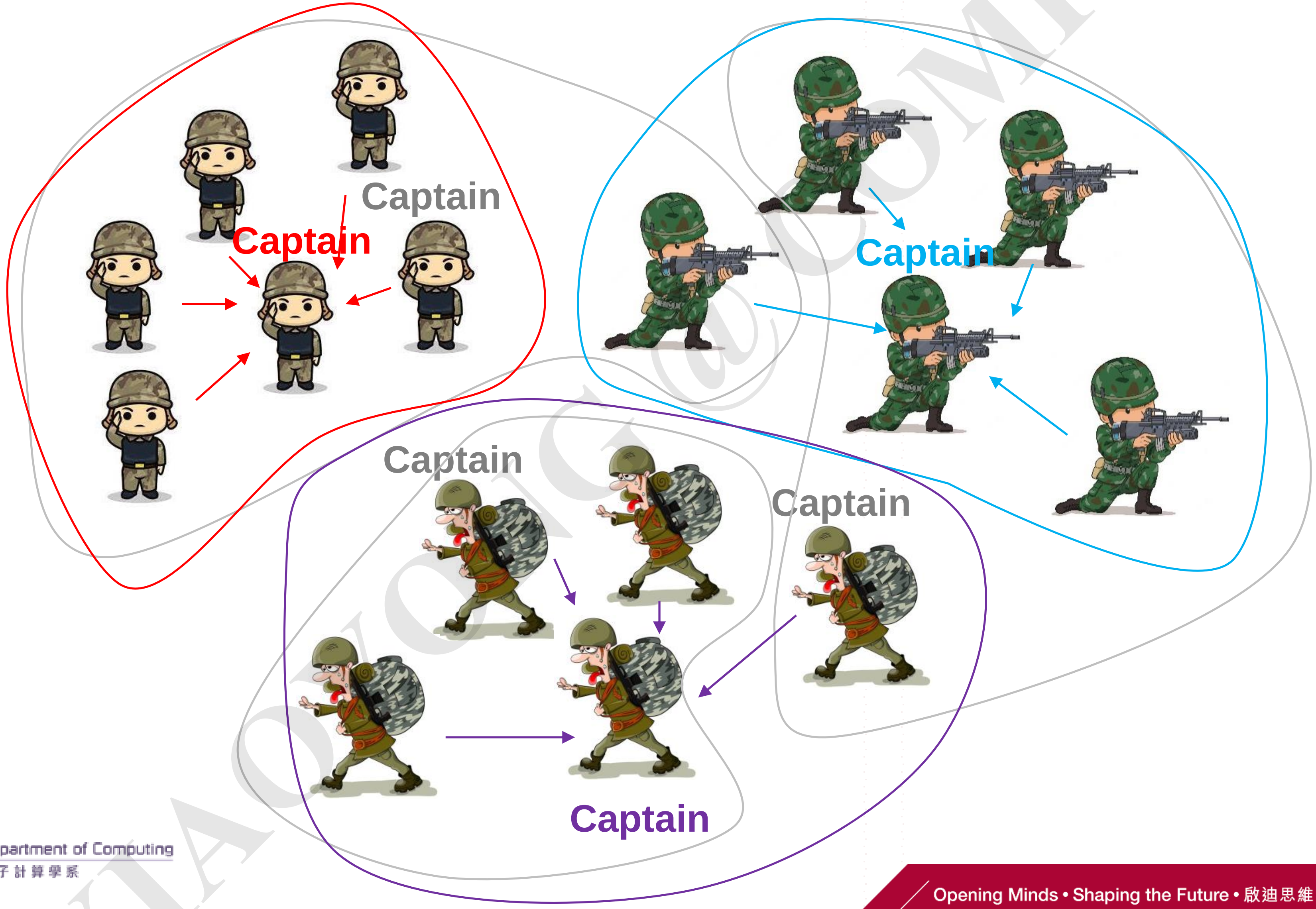
2. Soldiers report to the nearest captains



3. Re-election of captains

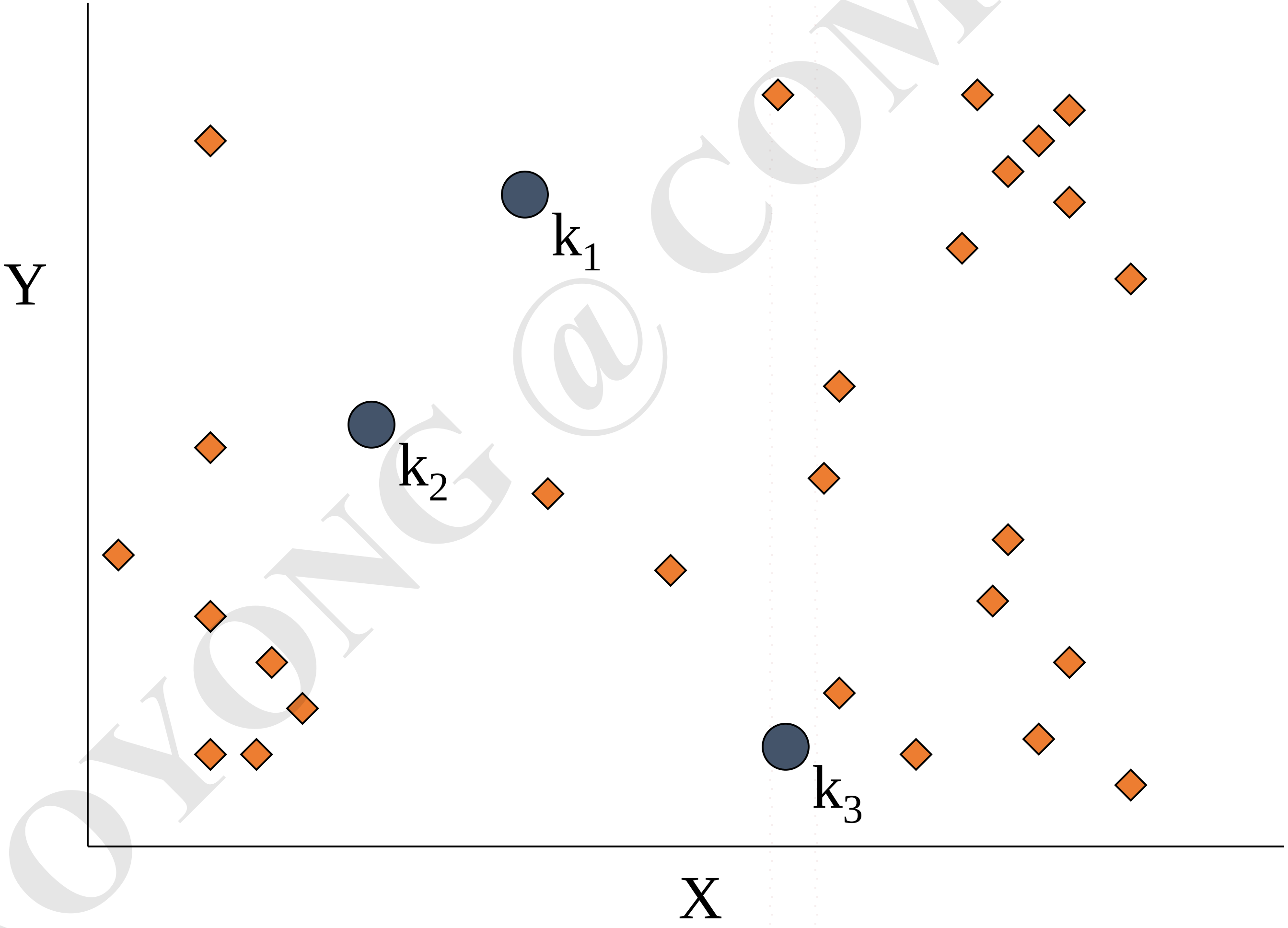


4. Report again



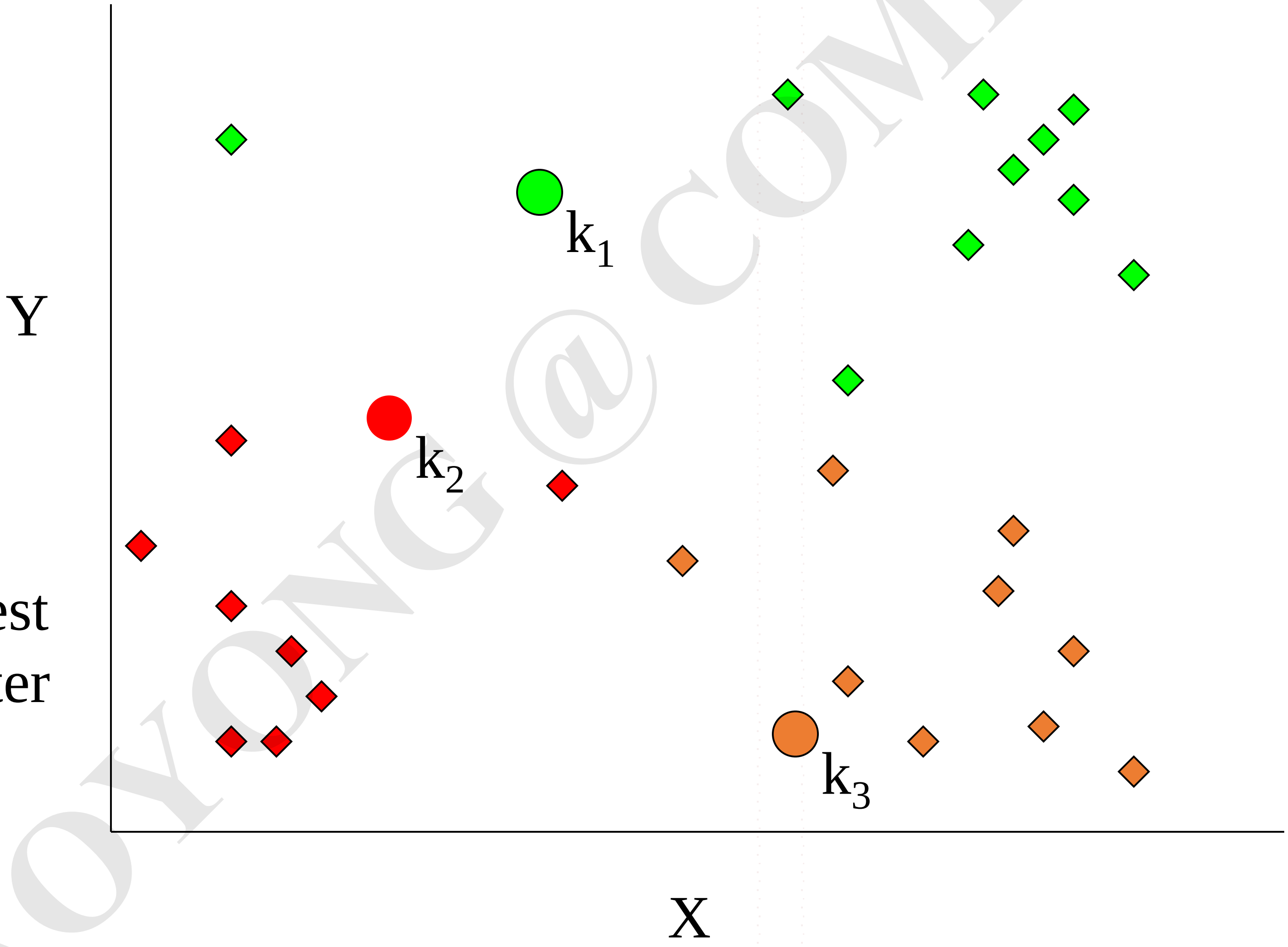
K-Means Step 1

Pick 3 initial cluster centers (randomly)



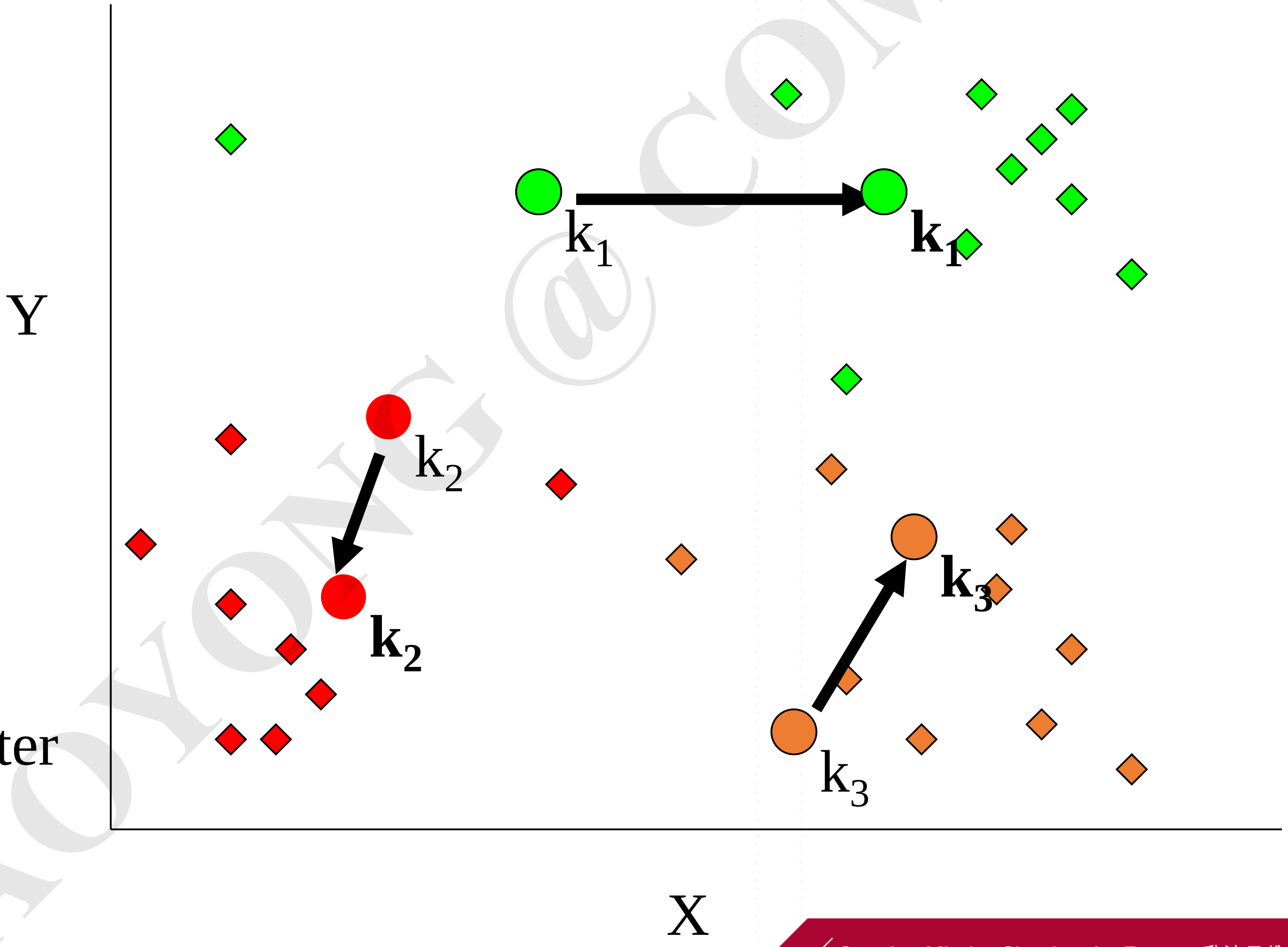
K-Means Step 2

Assign each point to the nearest cluster center



K-Means Step 3

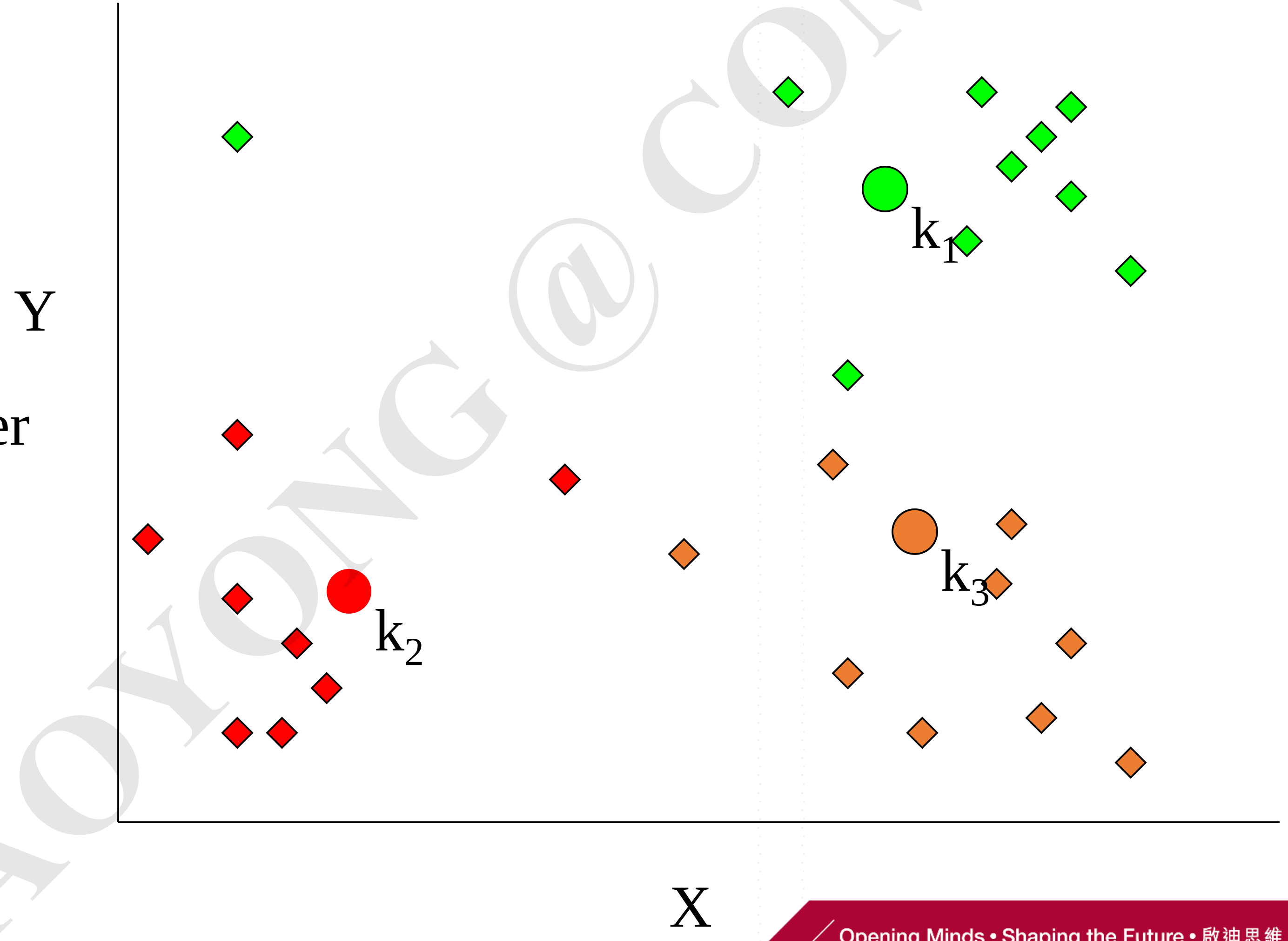
Move
each cluster
center
to the **mean**
of each cluster



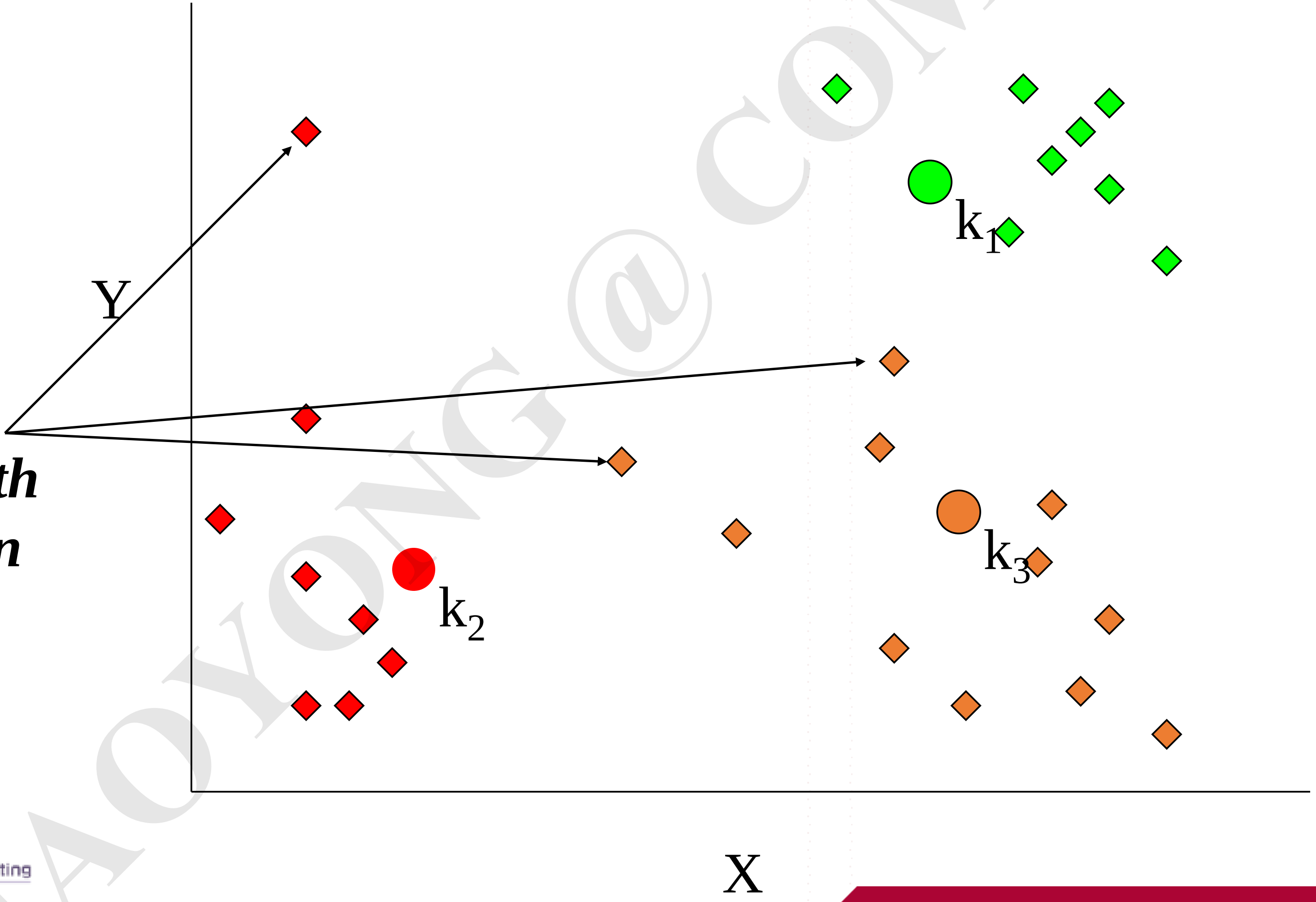
K-Means Step 4

Reassign points to the new (nearest) cluster center

Q: Which points are reassigned?

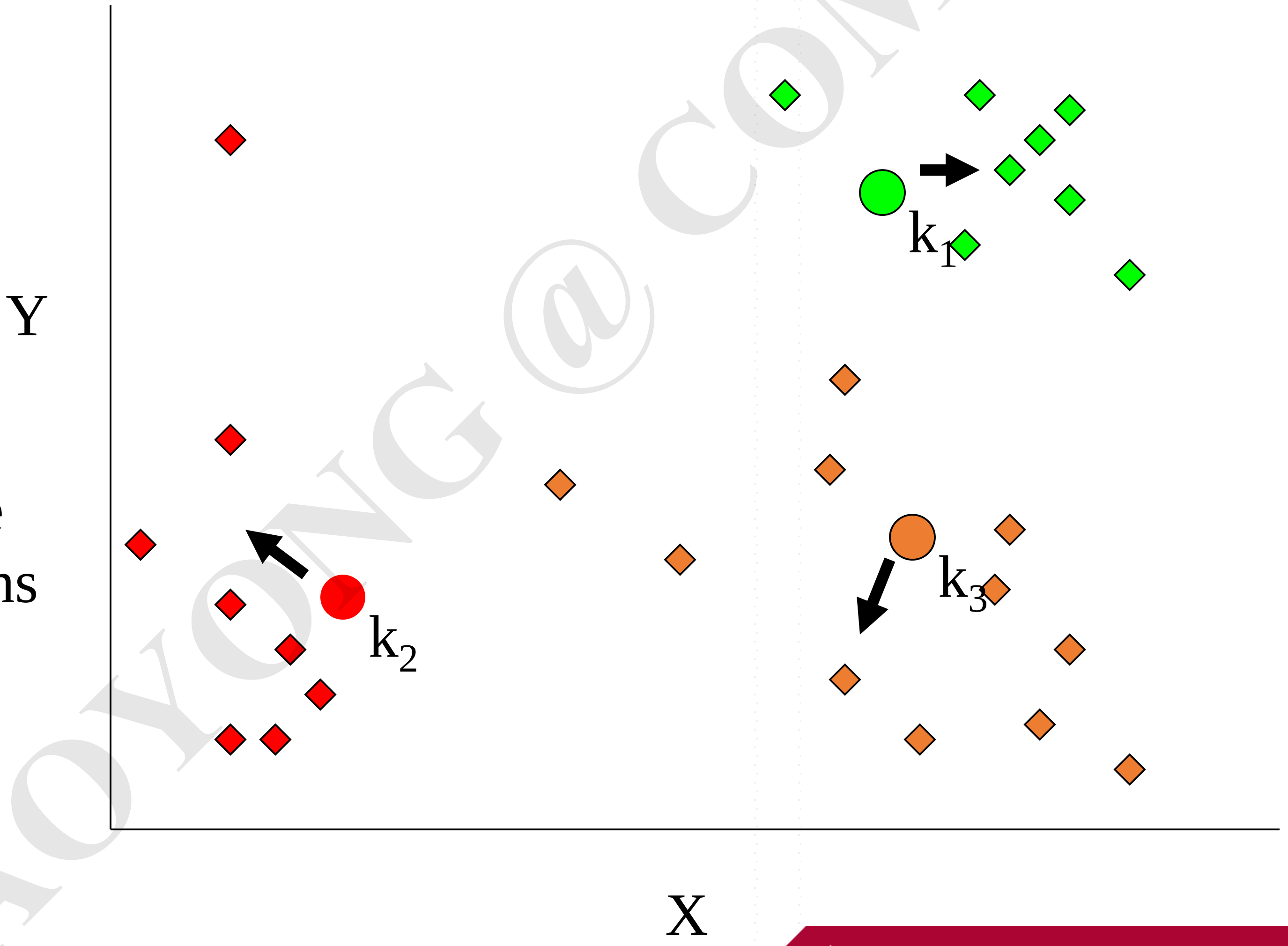


K-Means Step 4 …



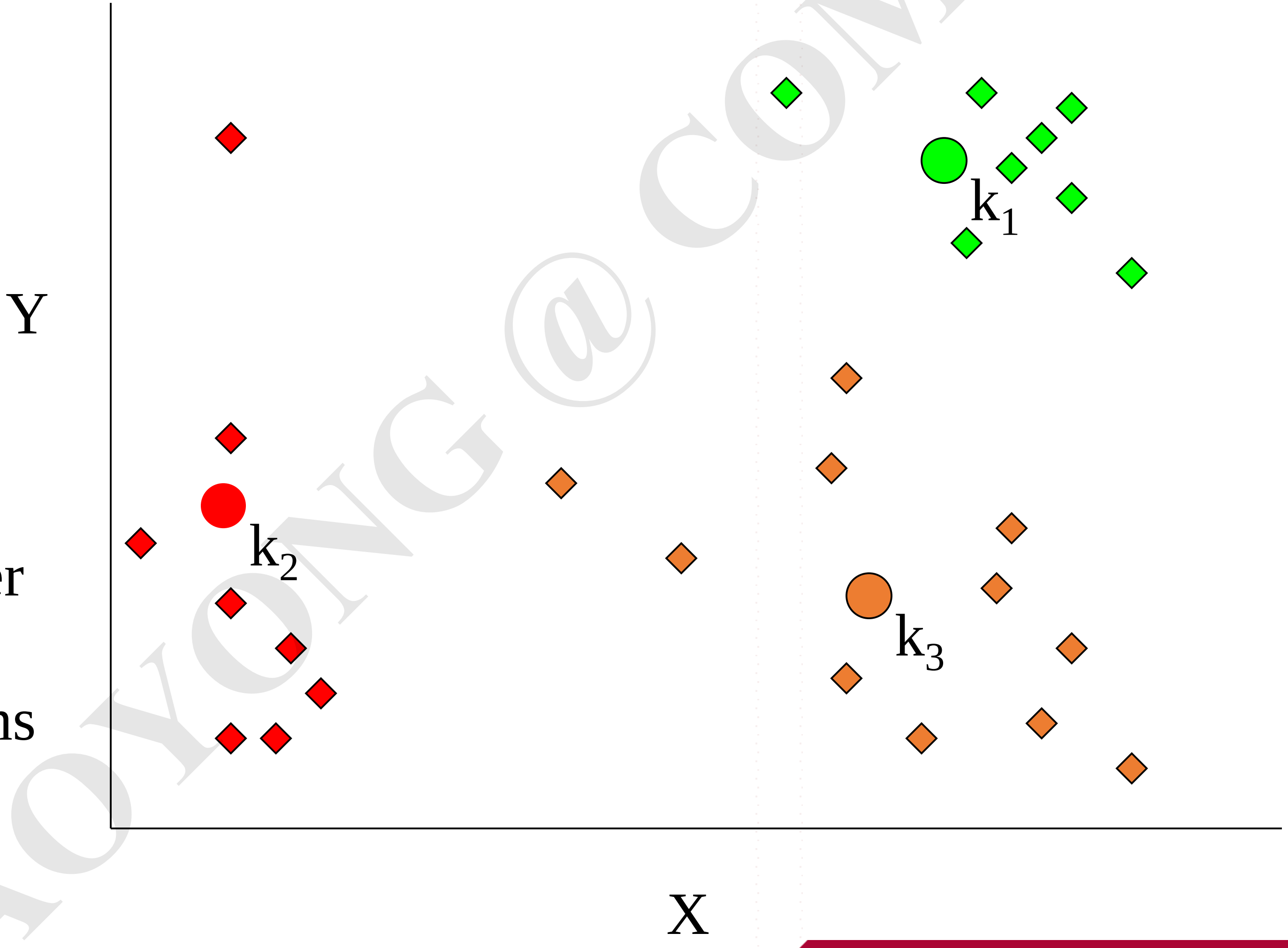
A: three points with animation

K-Means Step 4 …



Re-compute cluster means

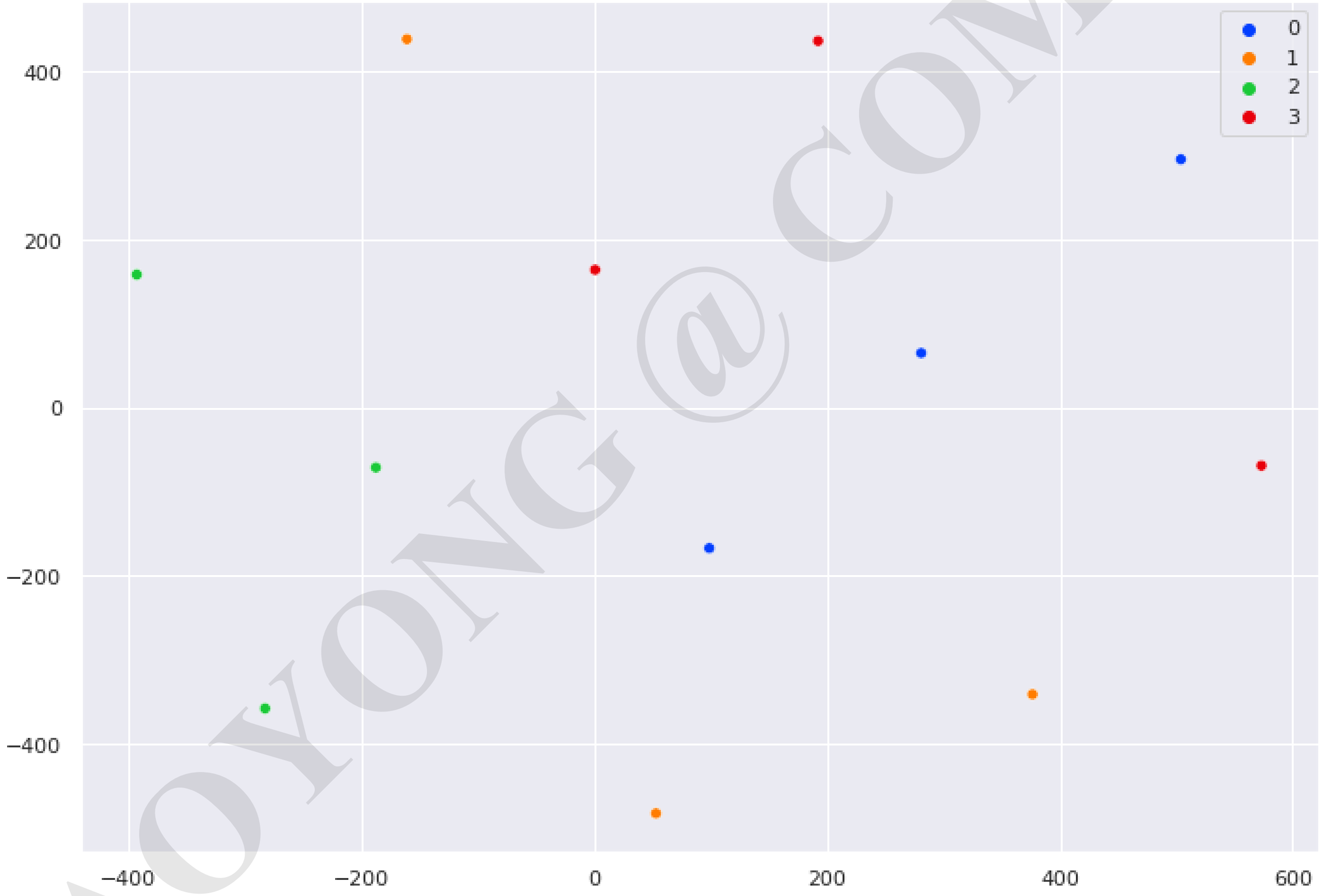
K-Means Step 5



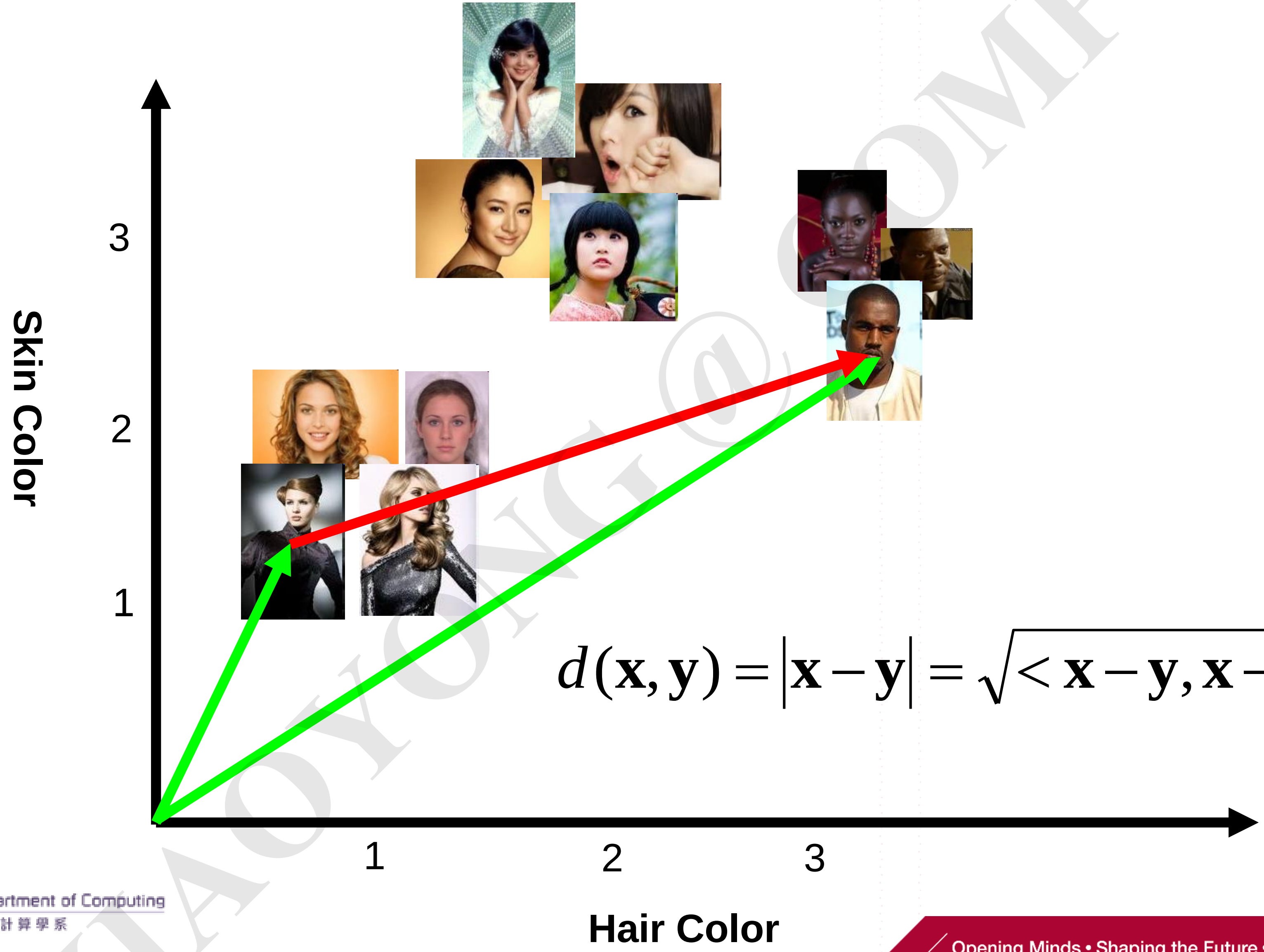
Move cluster centers to cluster means

Let’s take images from IMHere as examples

Let’s take images from IMHere as examples



Metrics – Euclidian Distance



$$d(\mathbf{x},\mathbf{y}) = |\mathbf{x}-\mathbf{y}| = \sqrt{<\mathbf{x}-\mathbf{y},\mathbf{x}-\mathbf{y}>}$$

Clustering is one of the most representative examples of **Unsupervised Learning**.

We will get back later.

Grouping images is fun.

However, that’s not the way we employed in IMHere.

IMHere – Token-based attendance checking

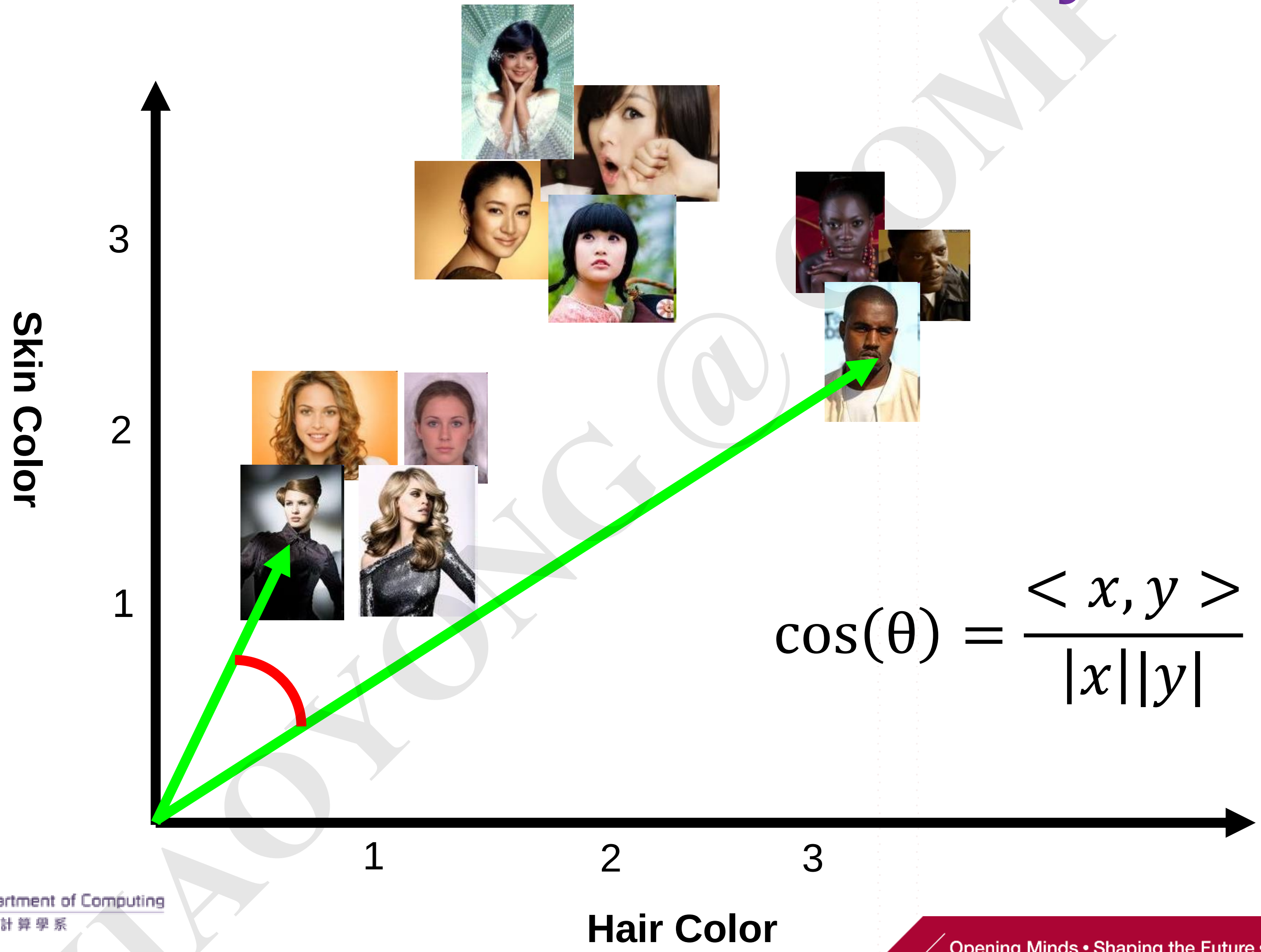
In IMHere, we are **looking for** images that are with **similar content** with the one you uploaded. You are checked-in if those are what you used for registrations.

This is called **Content-Based Image Retrieval (CBIR)**.

General Steps of CBIR

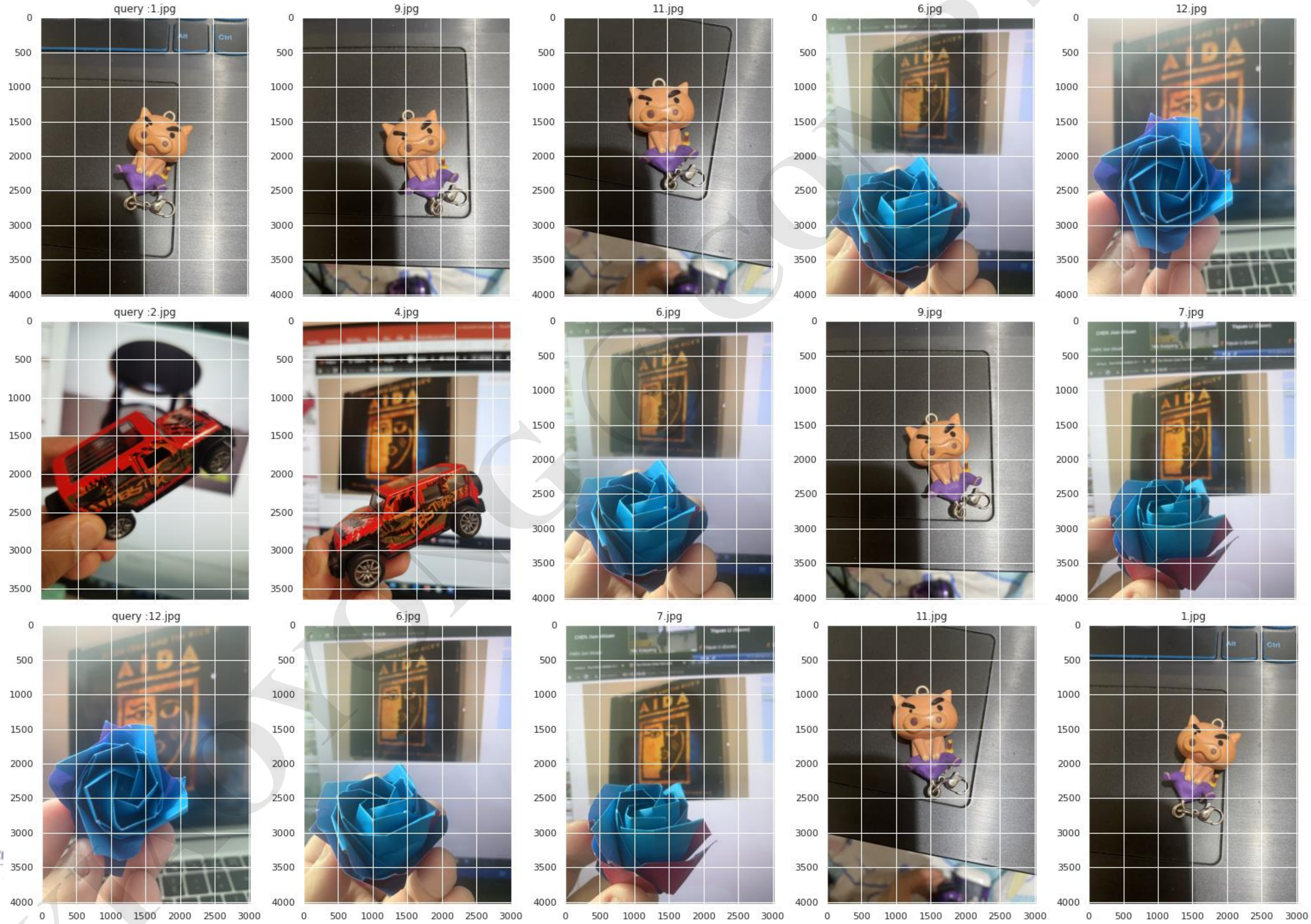
- 0. Extract features vectors of all images on file
- 1. Extract the feature vector for the **query** image
- 2. Compare it to all the (**target**) images on file by calculating the query-target similarities
- 3. Sort the similarities in a descending order with which we generate a **ranked list** of the targets
- 4. Present the ranked list to the searcher

Metrics – Cosine Similarity



$$\cos(\theta) = \frac{<x, y>}{|x||y|}$$

Examples from IMHere



It’s straightforward to compare the query to the targets in an sequential manner (one by one), but it’s not efficient.

Let’s assume we have 1 million targets in IMHere. It may take 27 hours per query (100 milliseconds for each target).

Is there a better way?

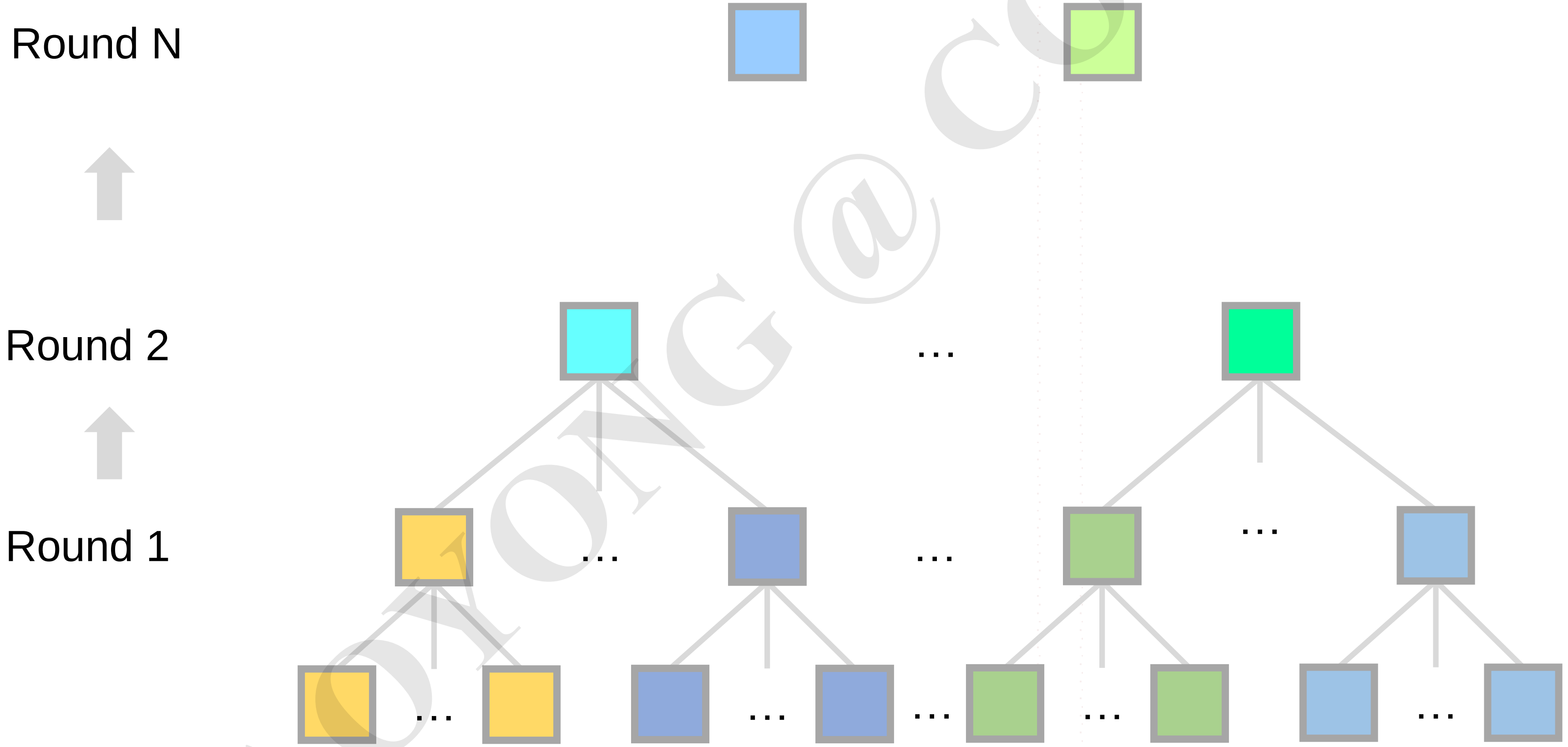
Clustering comes to play!

Indexing: group the images as clusters and pick one from each cluster as its representative

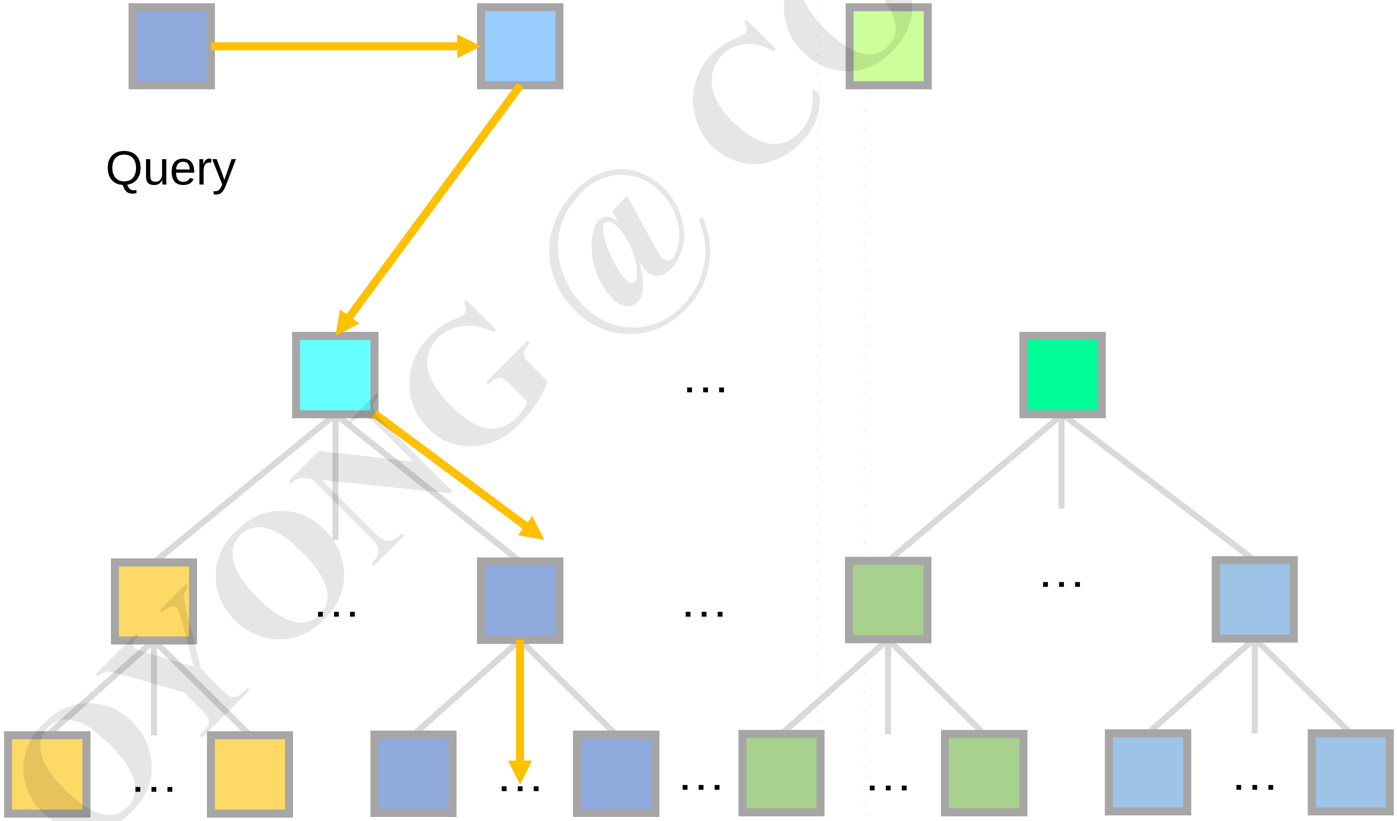
Coarse: compare to the representatives only and find top-k

Fine: compare to the member images of the top-k clusters

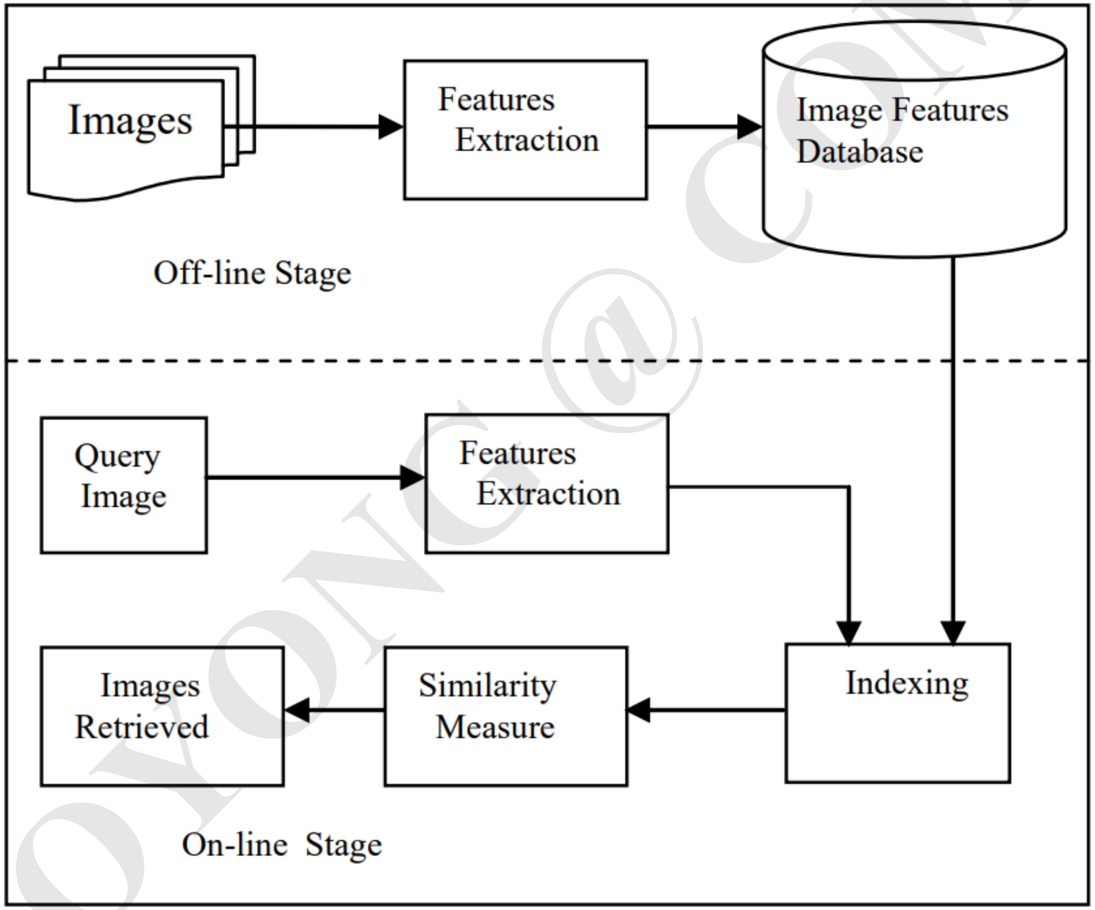
Index the images as a tree



Search from Coarse to Fine



A typical CBIR system



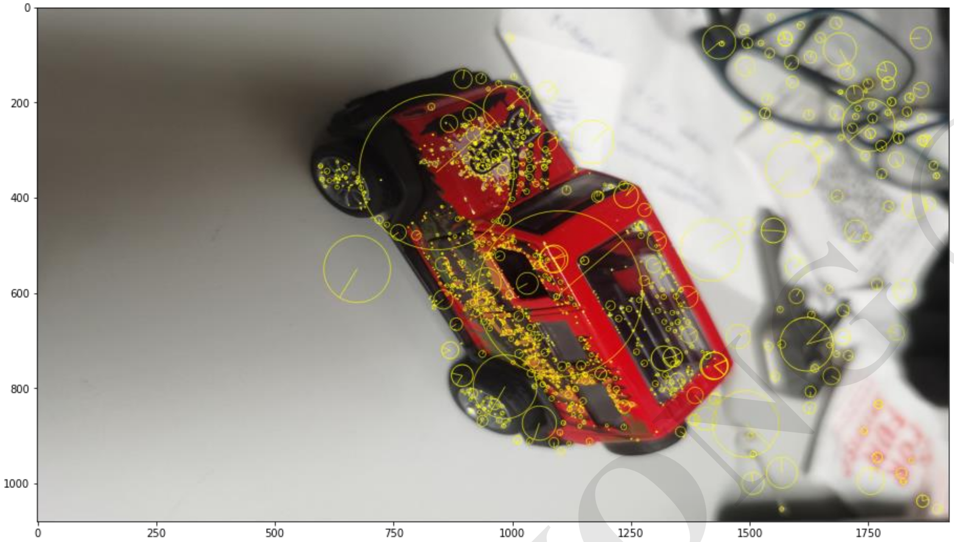
Alkhawlani M, Elmogy M, El Bakry H. Text-based, content-based, and semantic-based image retrievals: A survey[J]. Int. J. Comput. Inf. Technol, 2015, 4(01): 58-66.

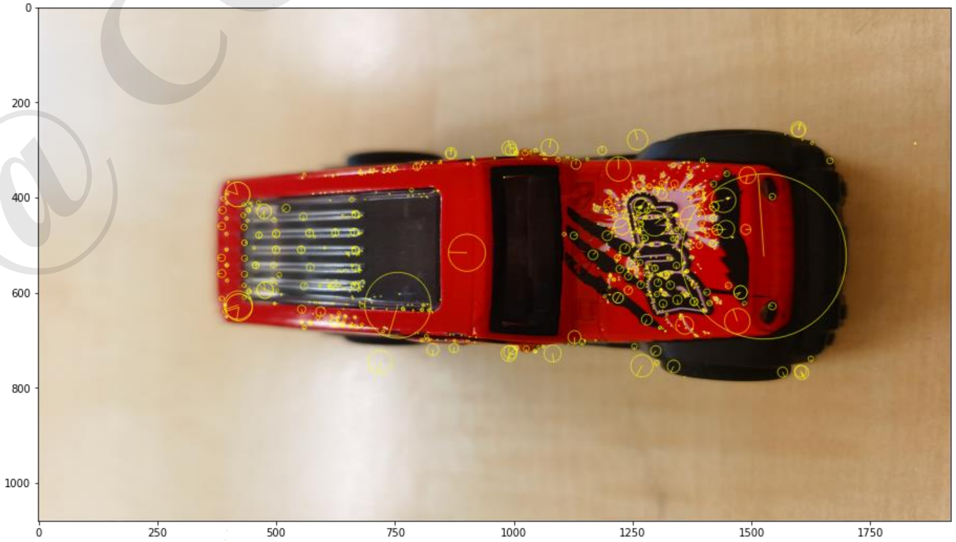
Do you notice?

What we introduced are based on an assumption of ***one-feature-per-image***.

How can we deal with images with multiple feature vectors?

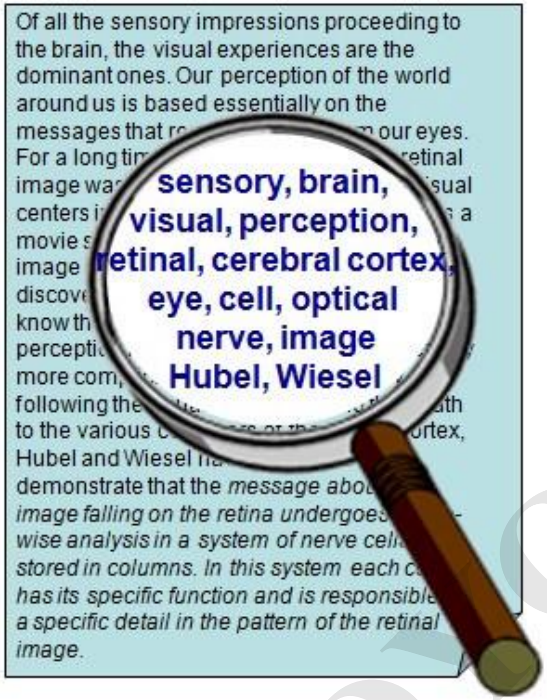
Multiple (Local) Feature Vectors

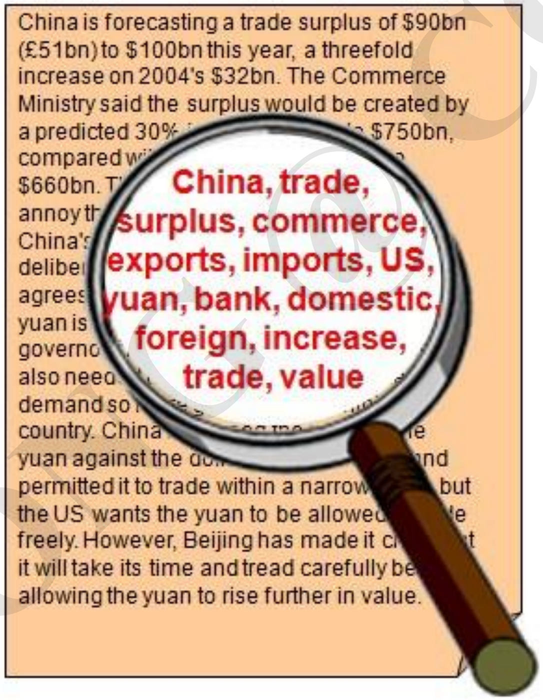


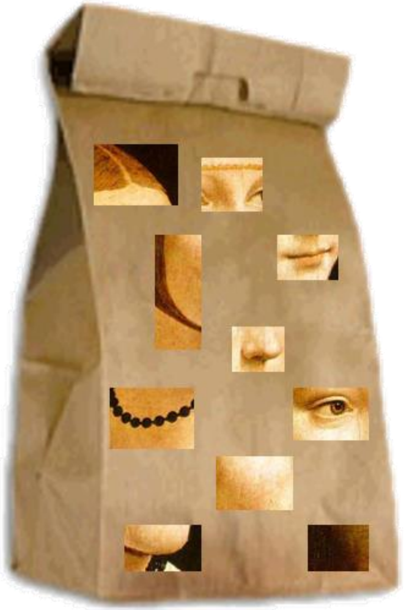


Bag of Visual Words (BoVW)

BoW vs. BoVW

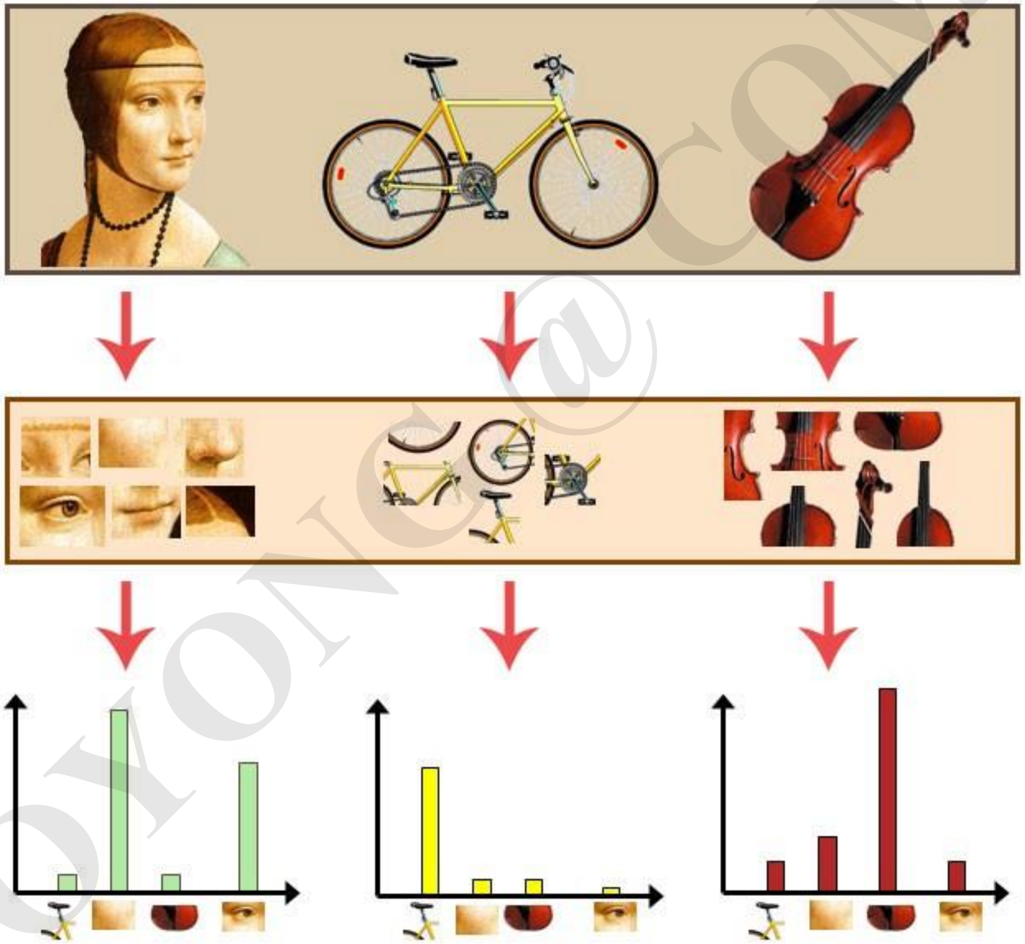




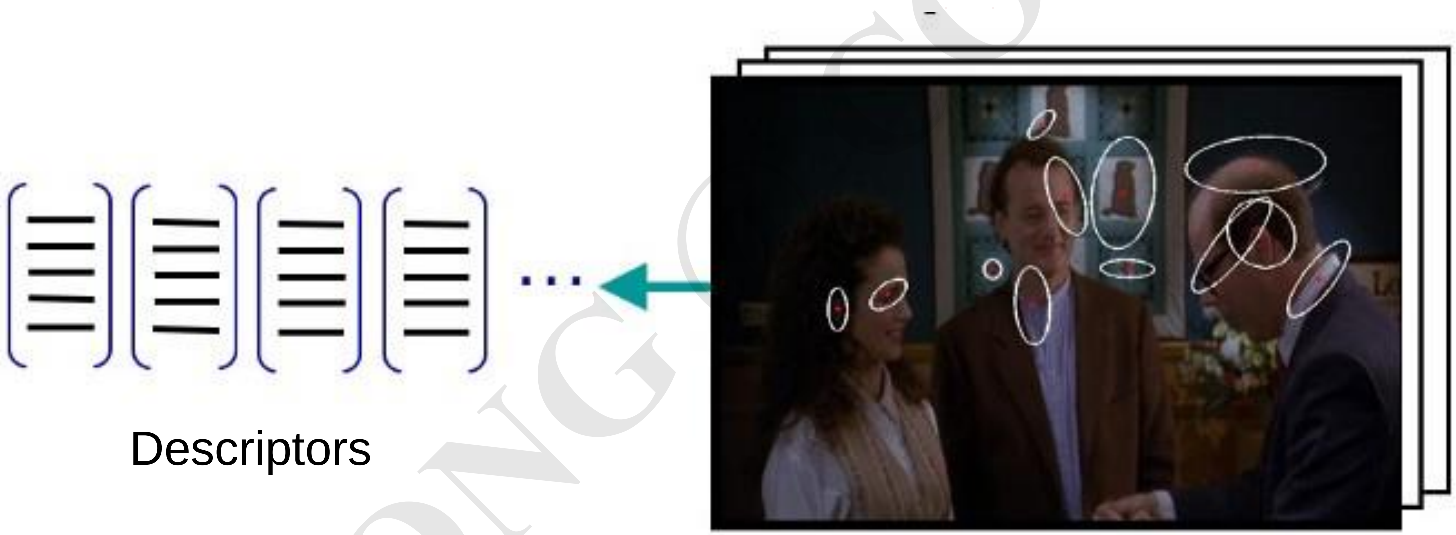


Bag of Visual Words in a Nutshell - The art of choosing important features - Bethea Davida

Visual words and bags



Step 1: Visual Descriptors Extraction



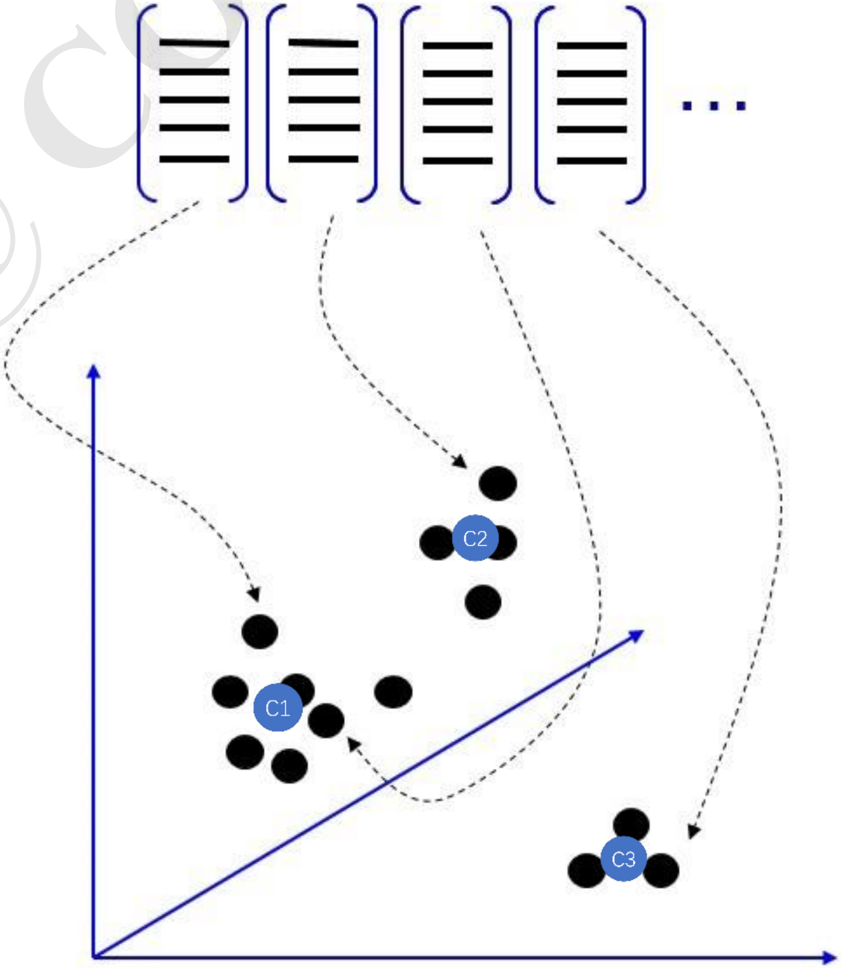
Descriptors

Step 2: Dictionary (CodeBook)

Group the descriptors (from all images) using clustering

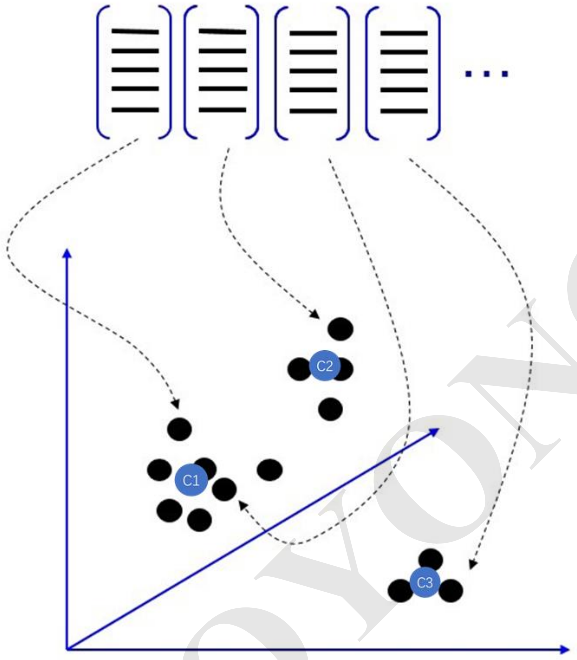
Pick one from each cluster as the representative and put them together to construct a dictionary (codebook)

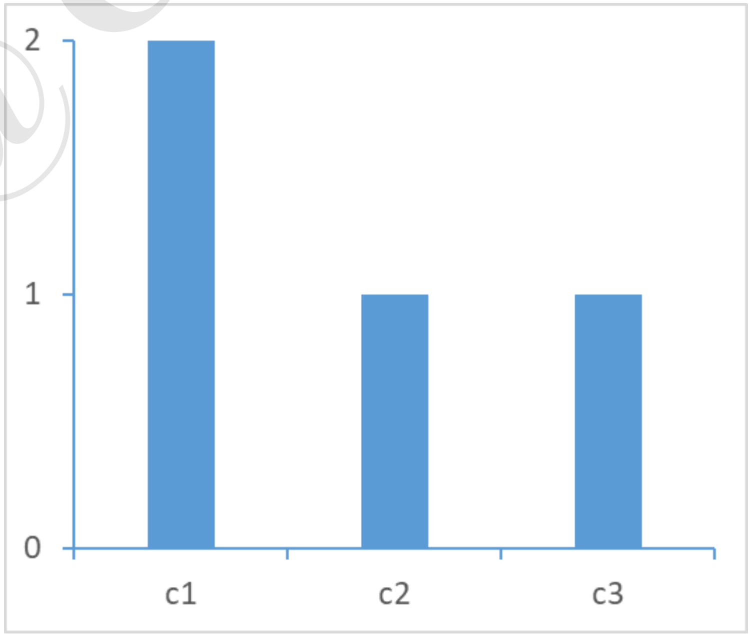
Descriptors in a dictionary are then used as visual words



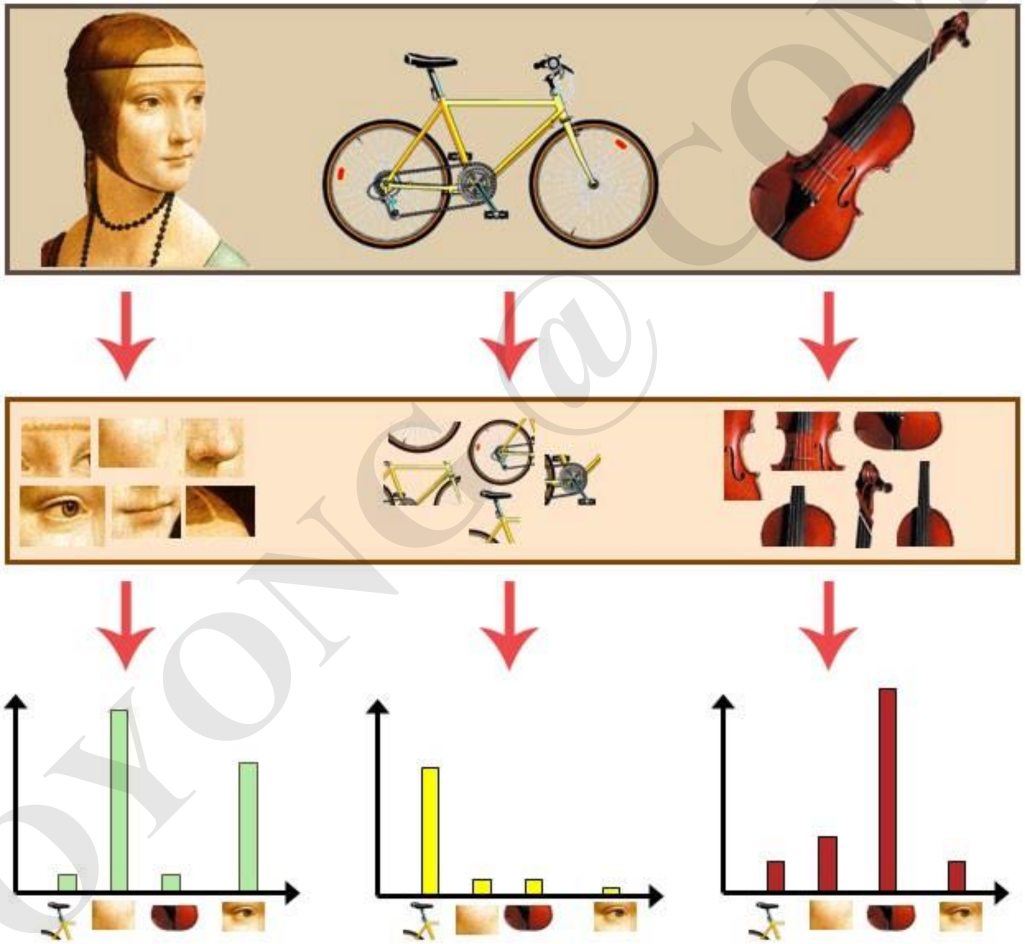
Step 3: Count words in a bag

For each visual word in the dictionary, calculate its frequency in the bag and construct a histogram as the feature vector

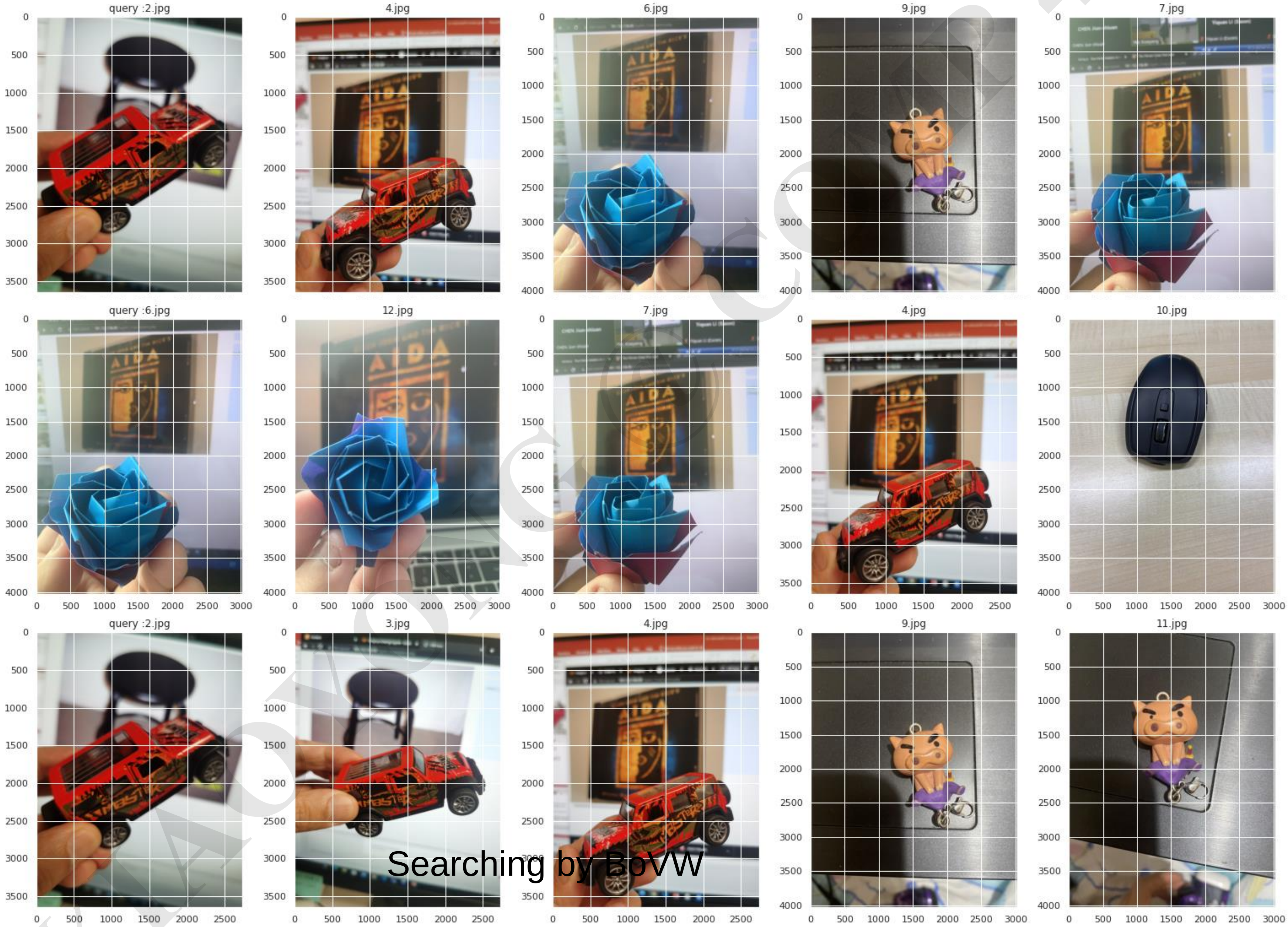




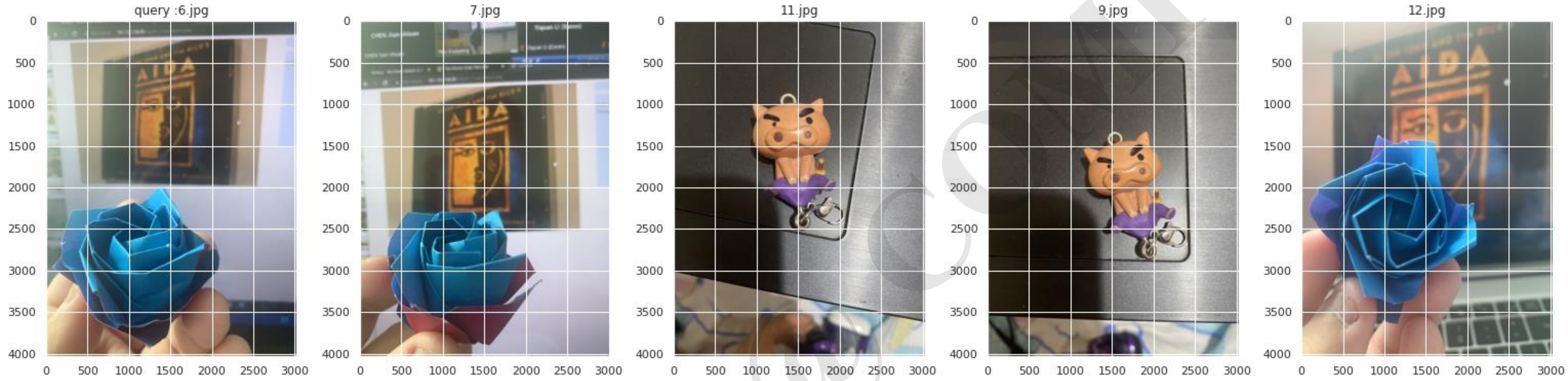
Visual words and bags



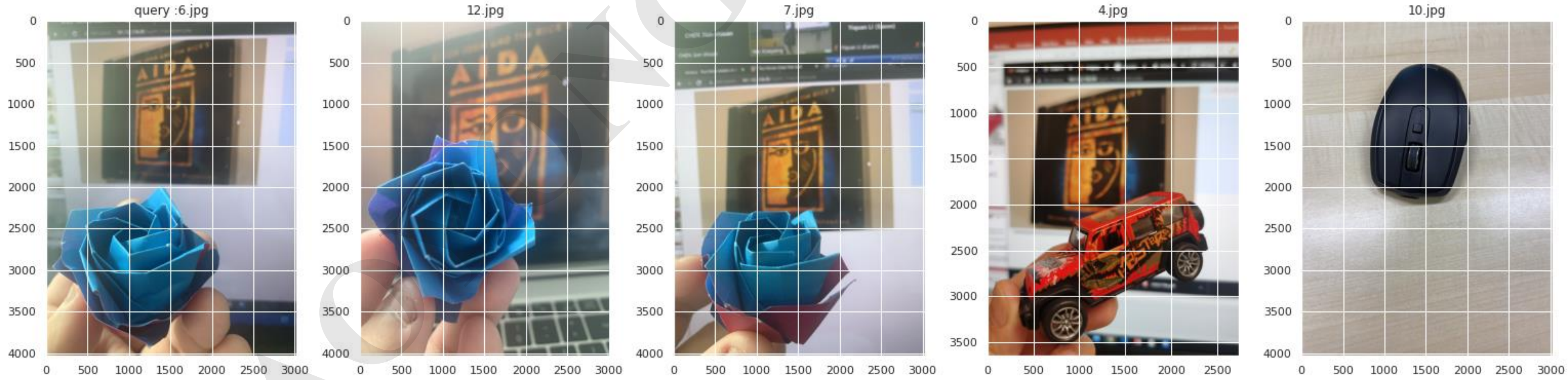
Examples from IMHere



Examples from IMHere



Searching by Histograms



Searching by BoVW

CBIR is not the only solution

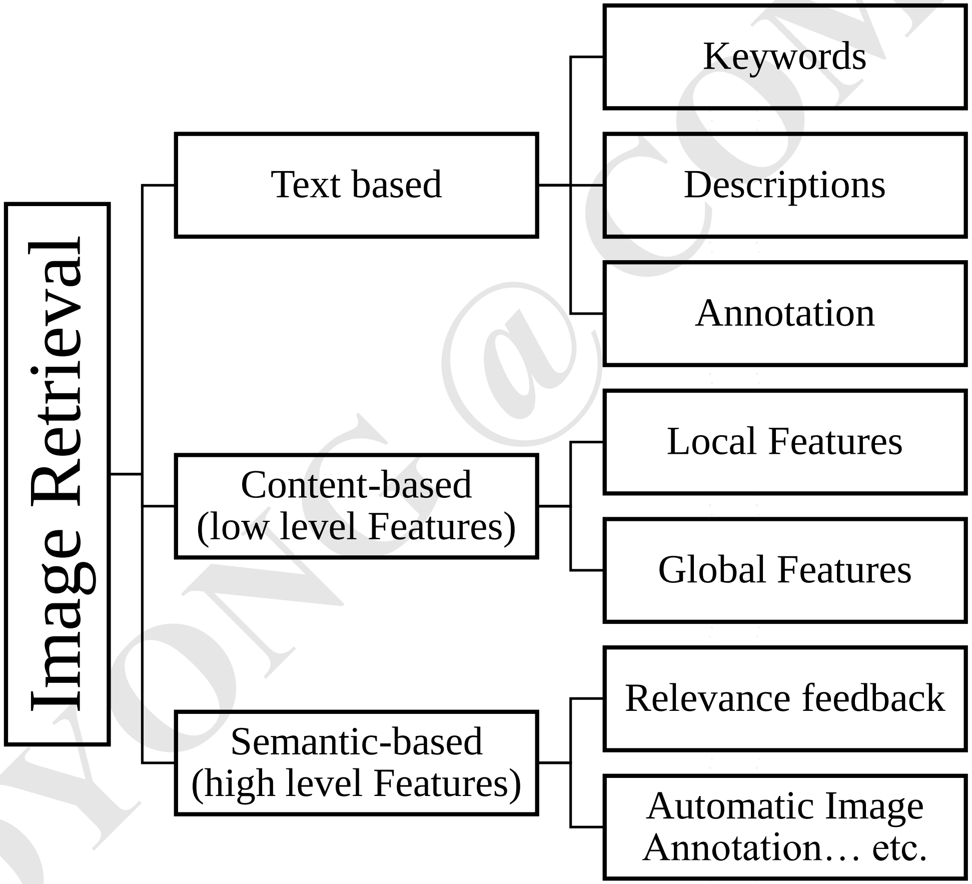


Image Retrieval
Text based
Keywords
Descriptions
Annotation
Content-based (low level Features)
Local Features
Global Features
Semantic-based (high level Features)
Relevance feedback
Automatic Image Annotation… etc.

Alkhawlani M , Elmogy M , Bakry H E . Text-based, Content-based, and Semantic-based Image Retrievals: A Survey. 2015.

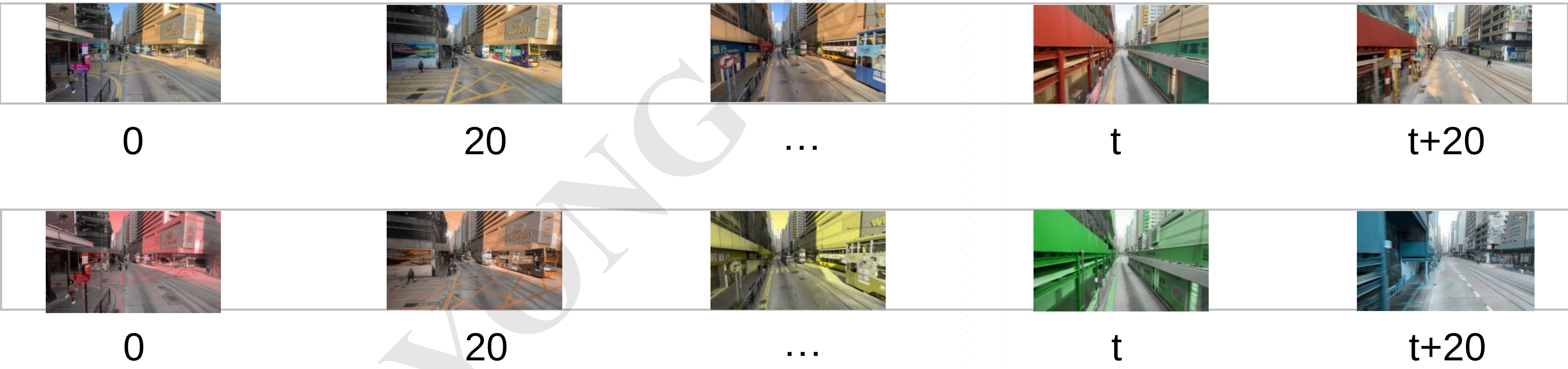
The New Toy

> The question to answer: is BoVW able to find the nearest keyframes?

Keyframes selected using a fixed interval of 20 frames

The New Toy

- The question to answer: is BoVW able to find the nearest keyframes?



Keyframe tones modified

The New Toy

- The question to answer: is BoVW able to find the nearest keyframes?

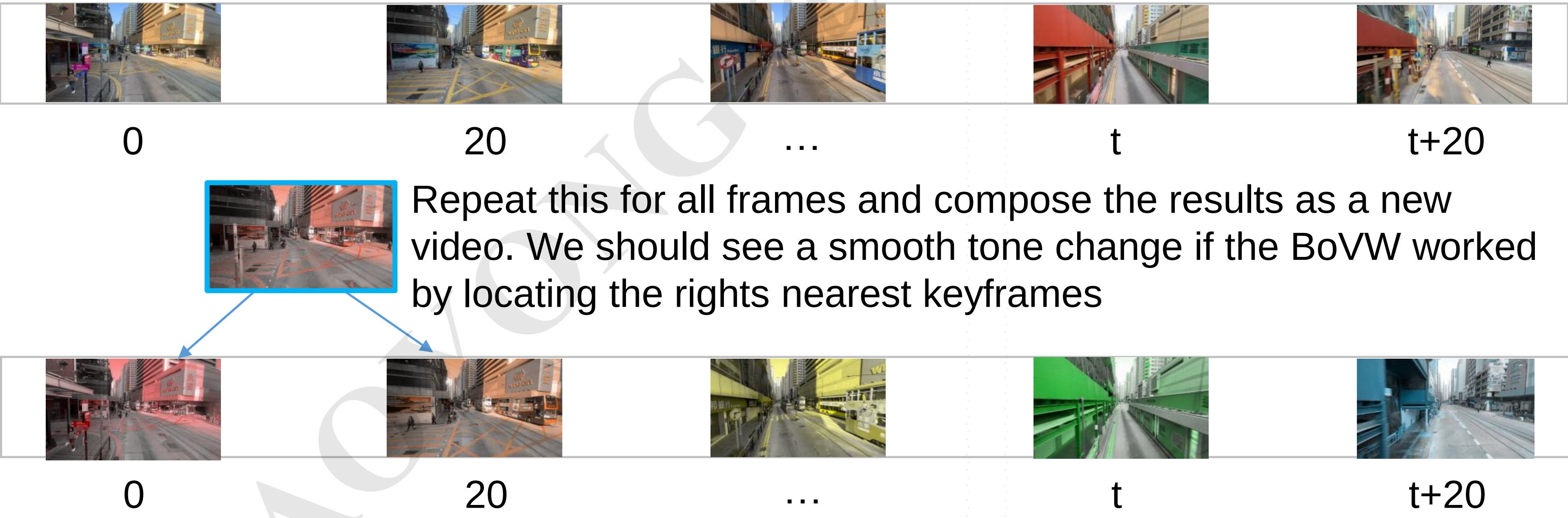
0 20 … t t+20

Will the **nearest keyframes** of a query **frame in the feature space** be the same as its **nearest keyframes in the video?**

0 20 … t t+20

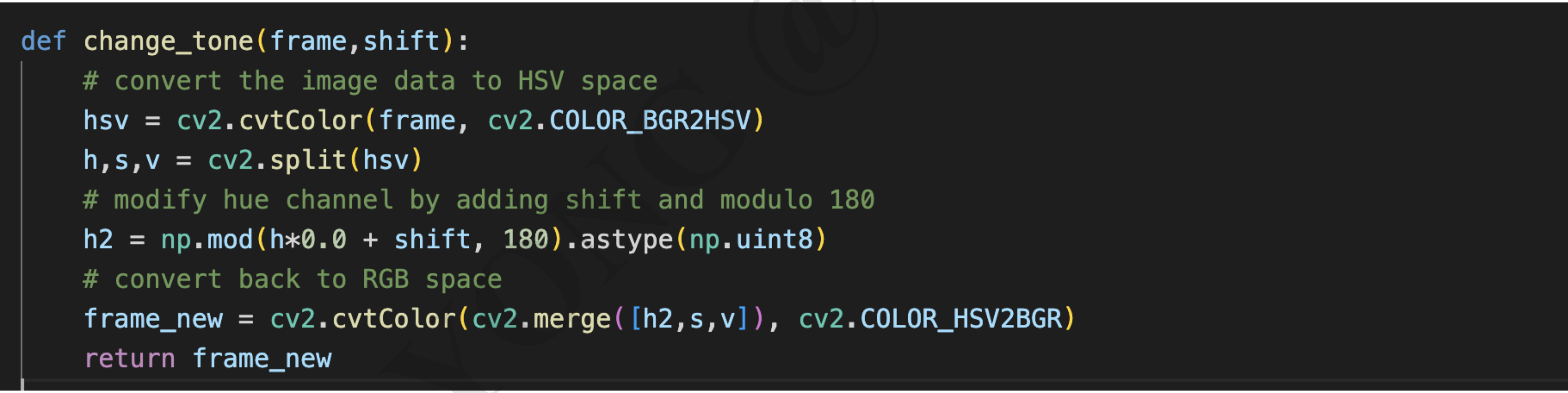
The New Toy

> The question to answer: is BoVW able to find the nearest keyframes?



Repeat this for all frames and compose the results as a new video. We should see a smooth tone change if the BoVW worked by locating the rights nearest keyframes

Code to modify the tone of an image (nparry in BRG)



```
def change_tone(frame,shift):
    # convert the image data to HSV space
    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
    h,s,v = cv2.split(hsv)
    # modify hue channel by adding shift and modulo 180
    h2 = np.mod(h*0.0 + shift, 180).astype(np.uint8)
    # convert back to RGB space
    frame_new = cv2.cvtColor(cv2.merge([h2,s,v]), cv2.COLOR_HSV2BGR)
    return frame_new
```

Please take it as a challenge and we will release the sample code later. In fact, you will have all the necessary code by completing the tasks of our next tutorial.

Thank you!