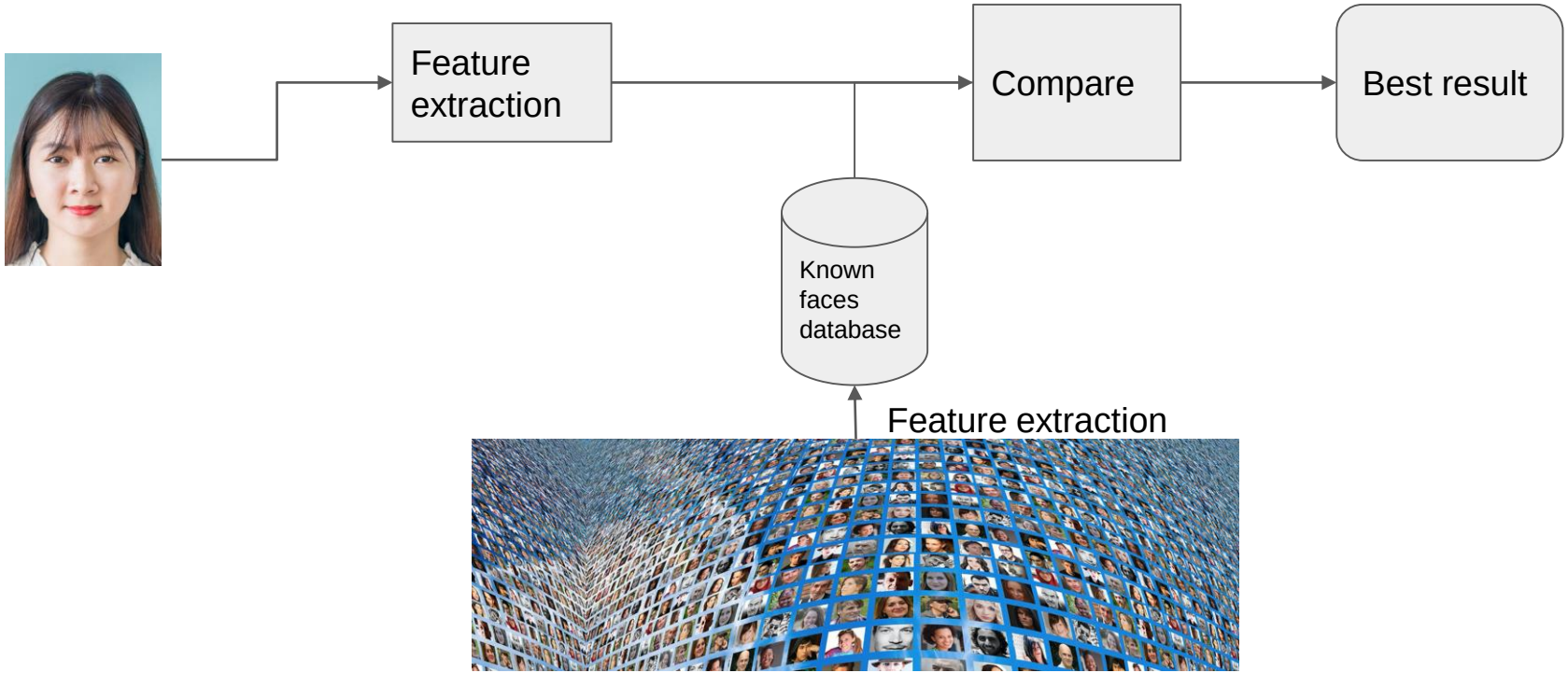
HOW TO SEARCH FOR AN IMAGE IN MILLIONS OF RECORDS IN JUST 0.1S

Content

- Why do we need to extract some features for images?
- How to organize photo storage for easy search?

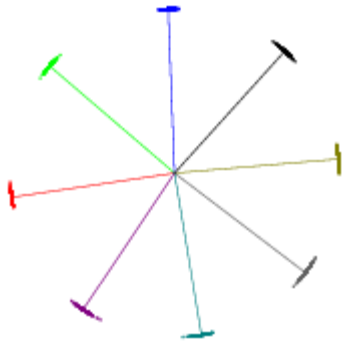
Purpose

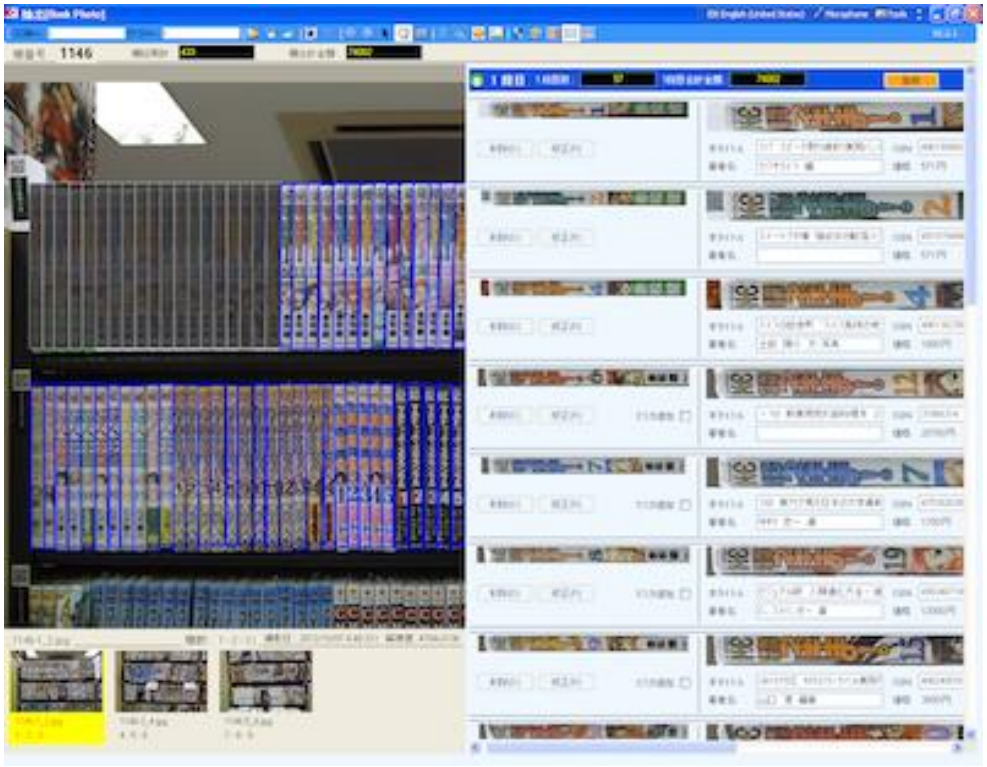
- Search an image in 100M records in real time with popular server



Feature extraction

- Using Convolution neural Network and one shot learning to extract face feature as 512D vector with length = 1
- The feature extracted by neural network will be as below: (sample on 8 identities with 2D features)





The system automatically identify and extract the book back-covers from the images of bookshelves. It then uses to perform matching on the images of back-covers to identify the books.

Feature extraction

This automatic stocktaking system is used in Japan bookstores to perform stocktaking on their book inventory.

It can search through several hundred thousand images in a database in a few seconds.The recognition accuracy has currently reached 92%.

Feature extraction

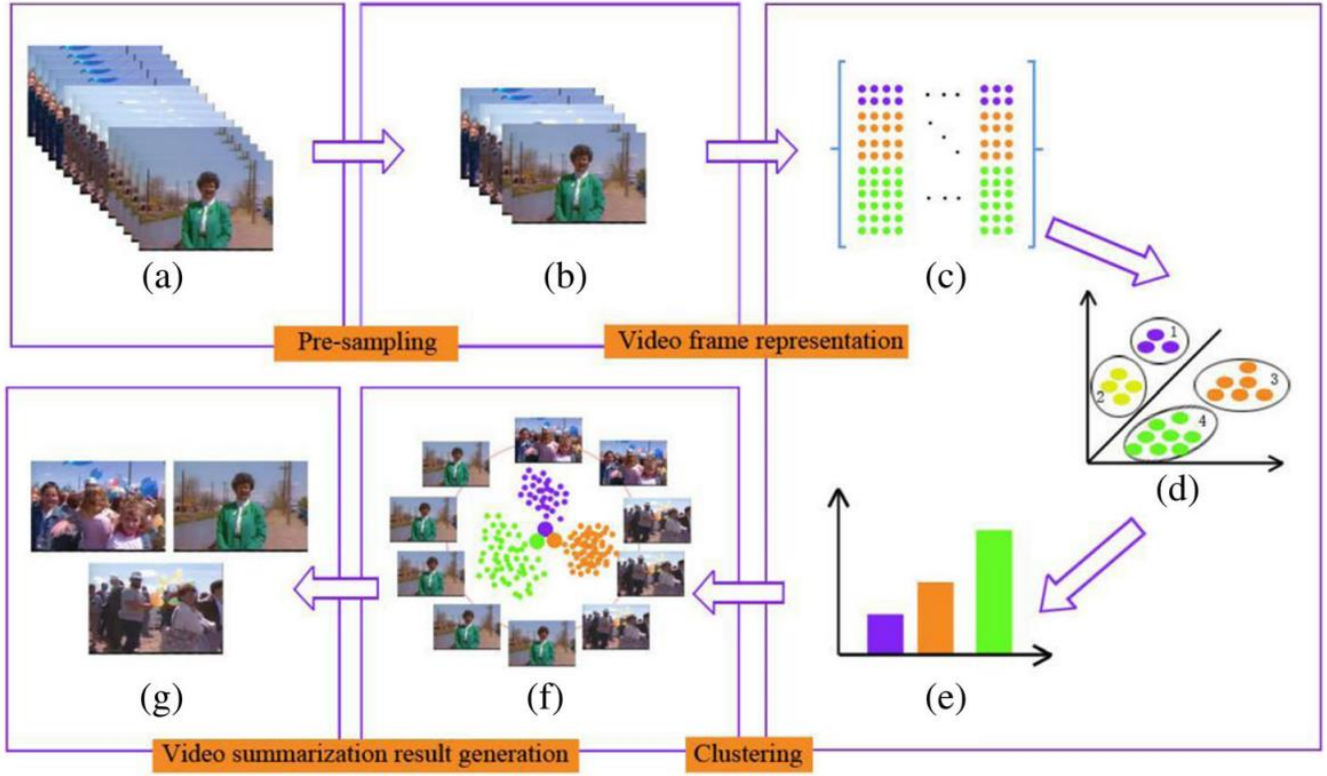


Image searching - compare

- Compare 2 features: using euclidean distance

$$d(\mathbf{p}, \mathbf{q}) = d(\mathbf{q}, \mathbf{p}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2}$$

$$= \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}.$$

- In case, the number known image database is small(example 1000 records), compare each features in database with input image's feature can get the most similar images
- But, known images database contain a hundred million of records→ compare sequentially will take a huge computation cost

Image searching: binary tree

- The idea of this solution is using k-mean with k=2 to clustering known image to 2 cluster. Then for each cluster, continue clustering to 2 sub cluster. We can do this clustering process until each cluster contain only 1 image.

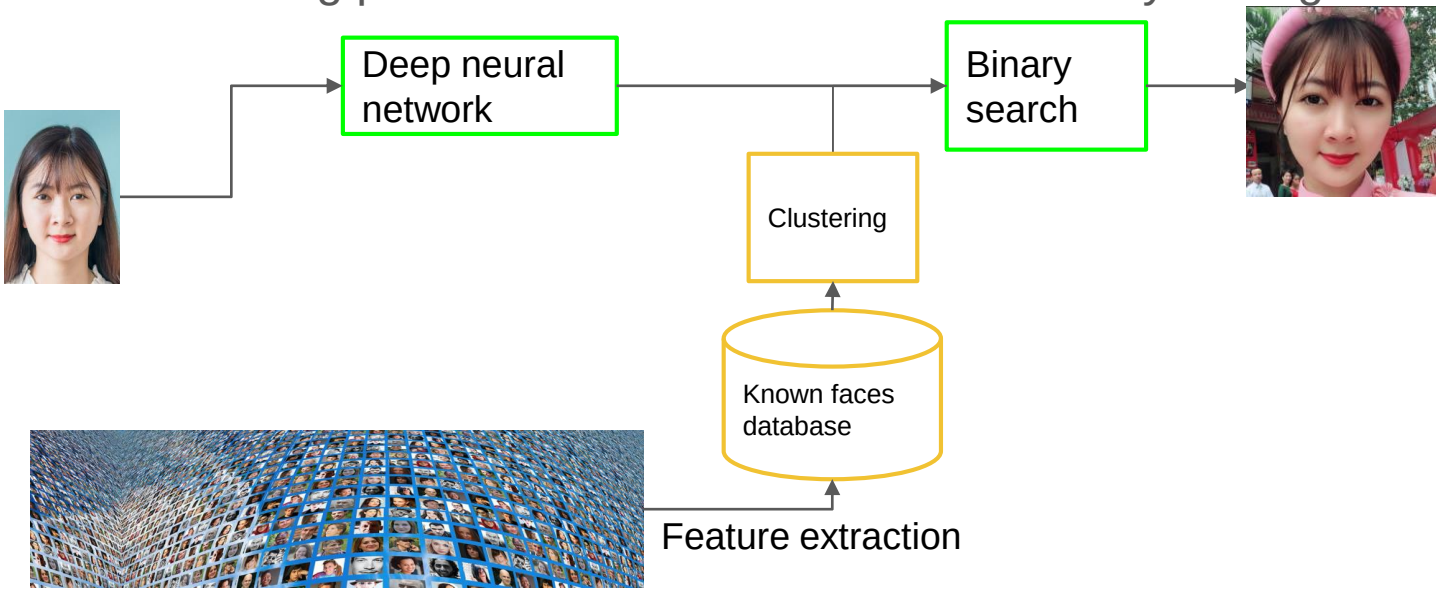
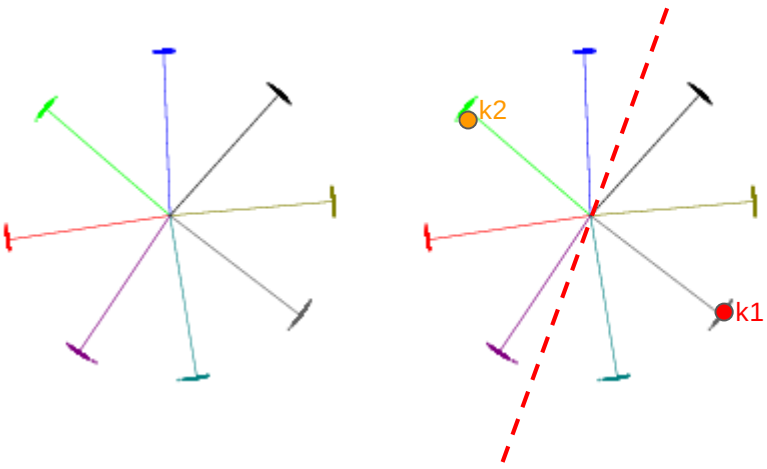
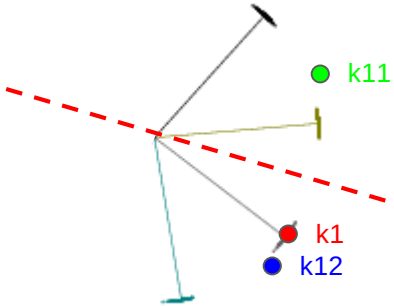


Image searching: binary tree

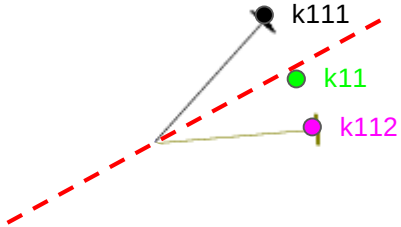
- Example on 8 identities with 2D features



1st clustering

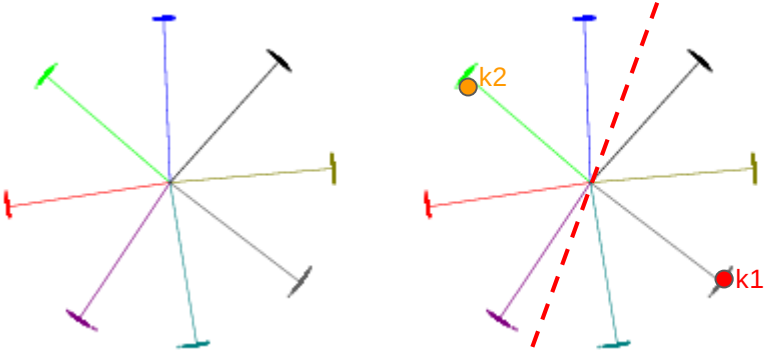


2nd clustering

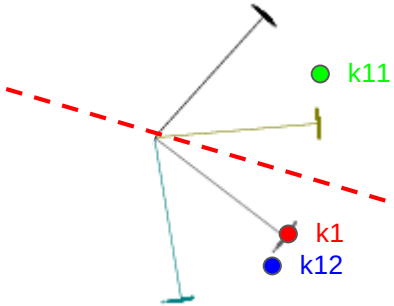


3rd clustering

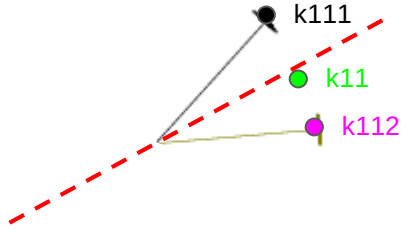
Image searching: binary tree



1st clustering



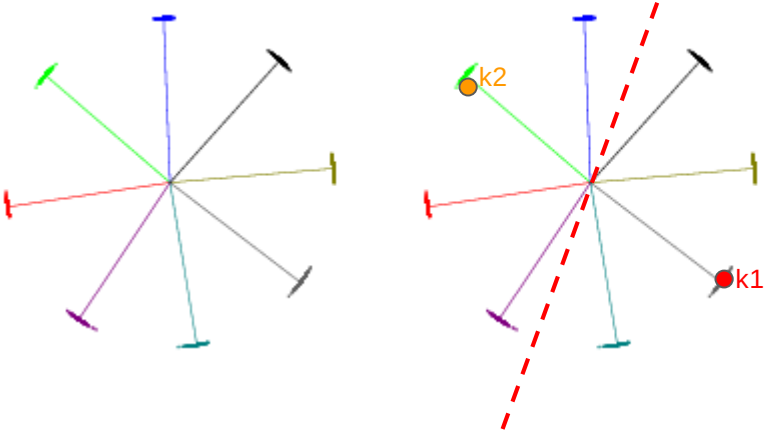
2nd clustering



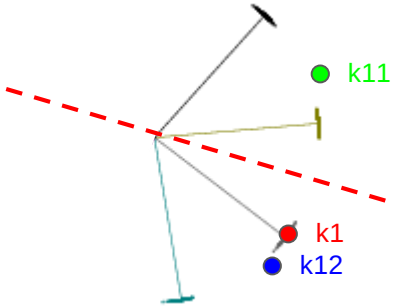
3rd clustering

- To compare input face(with feature vector f) with known database:
  - Compare f with cluster kernel k1, k2 -> example we got d(f, k1) < d(f,k2)
  - Compare f with cluster kernel k11, k12 → example we got k11
  - Compare f with cluster kernel k111, k112 → we can got the most similar features vector

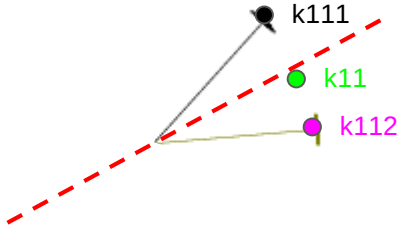
Image searching: binary tree



1st clustering



2nd clustering



3rd clustering

- By this solution(k-mean clustering):
  - The clustering doing in offline mode
  - The complexity is O(logn) instead of O(n) for sequential searching