# Face Recognition Using Clustering

Carmine D'Alessandro
*Student*
*Università degli Studi di Salerno*
*Email: c.dalessandro12@studenti.unisa.it*

Lucio Giordano
*Student*
*Università degli Studi di Salerno*
*Email: l.giordano73@studenti.unisa.it*

Ruslana Lishchynska
*Student*
*Università degli Studi di Salerno*
*Email: r.lishchynska@studenti.unisa.it*

Giulia Sellitto
*Student*
*Università degli Studi di Salerno*
*Email: g.sellitto21@studenti.unisa.it*

***Abstract*—Face recognition has gotten a huge attention in the last decades due to its possibilities in many areas of today's life. The task of facial recognition is the key to the recognition process of the individuals according to the presented pattern in forensics, services and systems of control, and other similar systems. This project dedicated to theme listed above using clustering, in particular K-Means.**

**K-Means is one of the most popular method of clustering. It aims to partition *n* observations into *k* clusters in which each observation belongs to the cluster with the nearest mean. K-Means clustering is a type of unsupervised learning, which is used when you have unlabeled data (i.e., data without defined categories or groups). The goal of this algorithm is to find groups in the data, with the number of groups represented by the variable *K*. The algorithm works iteratively to assign each data point to one of *K* groups based on the features that are provided. Data points are clustered based on feature similarity. The project that will be described in this whitepaper has these main objectives: detection of faces and features from the given dataset with videos, usage of K-Means clustering on the extracted features and, accordingly, the main task - face recognition.**

## 1. Introduction

The aim of our project is creating a system of facial recognition using the unsupervised learning technique, clustering. It was a brand-new topic for us but we decided to accept the challenge and tried to figure out a good strategy to solve this task. Our major concernes were about video-analysis, feature extractions, clustering method and optimization in case of bad results. After solving these problems we started optimizing our solution, providing output analysis.

## 2. Dataset Analysis

The dataset is called *GOTCHA*. It contains videos of 62 people. Those people are filmed in different contexts. Those contexts are so divided: in three of them the subjects are watching directly the camera, so they are called *collaborative*. In three of them the subjects are trying to avoid watching the camera , so they are called *non collaborative*. There is one last folder called *3D* in which the subjects are not moving and they are filmed frontally from their left to their right. For matter of privacy the dataset is private and protected under a contract of non-distribution.

## 3. Our Work

### 3.1. Face Detection and Feature Extraction

We divided our work in various steps. Starting from the videos we had to decide how to use that videos for face clustering. The first thing we did was dividing the videos in frames. After framing the videos we could finally start working on the images, where we detected the faces. We used *dlib* library that helped us working with the faces. After the detection, we had to chose how to use those faces for discriminating the different people.

**3.1.1. A First Approach: 68-points Detector.** We decided to use the *68_points shape predictor*, that, taking an image as input, tries to detect 68 main points on them and then we used those points to discriminate them, by calculating the eucledian distance between all the points. Later this method has been optimized by normalizing every distance value, diving it for the length of the diagonal of the detected face boundary box. After various improvements of this method, in parallel with improvements done on the K-Means, we understood that this model could hardly go over 65% of precision in validation and testing, then we decided to change the way of discriminating the different people.

**3.1.2. Evolution: best approach.** The new method doesn't use the *68_points shape predictor* but uses a python library called *face_recognition*. It is built on *dlib* and allows us to detect the faces in the image either with *Convolutional Neural Network* or with *Histogram oriented gradients*. We

propose an algorithm based on the *HOG* because after some evaluations we understood that we preferred a bit lower precision in detection for a much faster method, considering we would like to try making this system *Real Time*. Using *Face_recognition* we could detect the faces in the image and encode them as a 128-d vector, that we used to discriminate the different people.

**3.1.3. Output.** Both of the methods previously proposed, calculate a discriminant vector for each face in input and return them as row of a matrix used as input of the K-Means.

## 3.2. K-Means

K-Means is an algorithm to perform unsupervised learning, in particular clustering, over a dataset. K-Means clustering aims to partition *n* observations into *k* clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster.
The algorithm firstly initialize the means with random values $m_1, m_2, ..., m_k$. Then the algorithm proceeds by alternating between two steps.

- *Assignment step*: Assign each observation to the cluster whose mean has the least squared Euclidean Distance, this is intuitively the "nearest" mean.
- *Update step*: Calculate the new means to be the centroids of the observations in the new clusters.

The algorithm has converged when the assignments no longer change.
There are many initialization methods, we decided to use the random one.

## 3.3. Labels Matching

The clustering algorithm builds clusters of data and labels them in a way that is different from the original labels we got from the videos dataset. In order to perform a proper recognition of the subjects and check the goodness of our system, we needed to match the labels generated by the algorithm to the original ones. We modeled this new problem as a linear programming problem, representing it as showed in Figure 1.
We have a bipartite graph in which the first group of nodes $S = \{s_0, s_1, ..., s_n\}$ represents the original labels associated with the subjects and provideded by the dataset. The second group of nodes $C = \{c_0, c_1, ..., c_n\}$ represents the clusters labels generated by the algorithm. A generic edge $(s_i, c_j)$ represents the fact that subjects labeled $i$ are put in the $j$ cluster and it has a weight, $p_{(i,j)}$ which is the probability that the matching is actually the right one. Our objective is finding the right set of edges to obtain the matching between labels. As working in a python environment, we used *PuLP* library, which is built to model and solve linear programming problems. *PuLP* solver implements the *Branch and Cut* variation of the *Simplex* algorithm, specialized in solving integer linear programming problems in an efficient way.

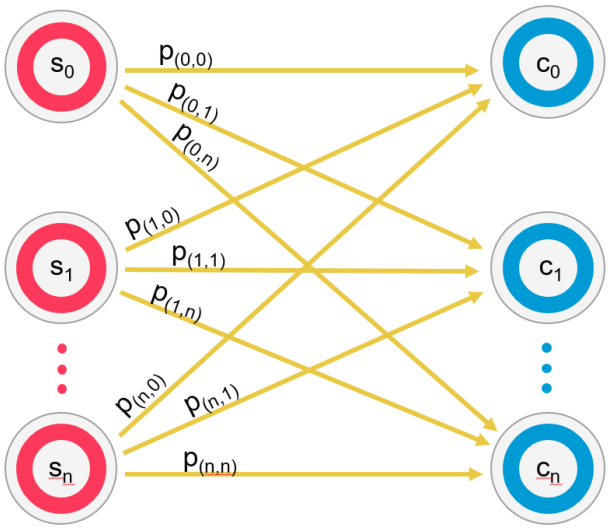


Figure 1. An intuitive representation of the labels matching problem.

## 3.4. Optimization

A crucial part of our work was about optimization. There were many possible different approaches, we chose one based on the sub-clustering of the data. During the process, each data cluster is further divided into a few different ones to perform a better classification of data and avoid wrong cluster intersections.
Finding the right number of clusters in which classify data is a common problem related to clustering. Known to be a *NP-Hard* problem, we tried to extimate the right number calculating the internal variance of the sub-clusters in advance (choosing the best one) without doing all the computation and testing. The method used to extimate the quality of the sub-clustering is the *Elbow method*.

## 3.5. Validation and Testing

For the validation task, we used the same data used in training to check how well the clustering performs. In this part we only wanted to see how much the training was effective. We took the labels provided as output by the model prediction and compared them to the original labels provided by the dataset, returning the overall percentage of matchings as shown in Figure 2.
In the figure *validation* is the actual validation done on the sublabels, *test over itself* is a test done on the training label itself. They return a different value because the validation is done on the sub-labels, the test over itself on the originary ones. For the testing we extracted new additional frames from the videos, different from the previous ones. In this way we could ensure that results were not distorted by the fact that the model had already "seen" the data. Using an unsupervised learning method, testing had to be performed paying attention to the existing matching between the original and predicted labels. We performed testing

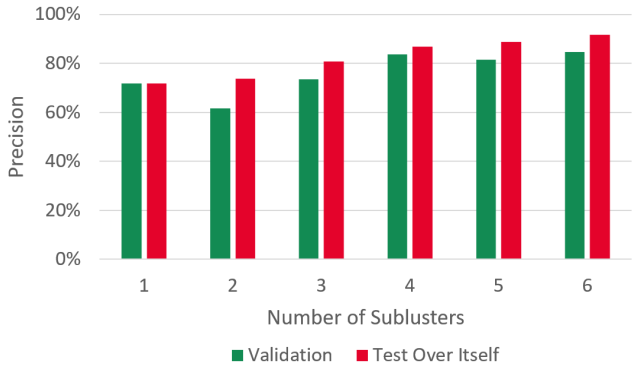


Figure 2. Main statistics.

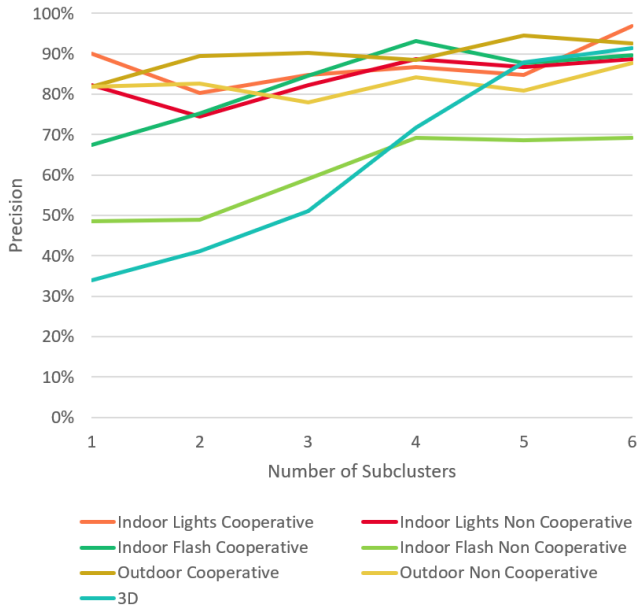


Figure 3. Model precision according to the number of subclusters.

on each environment indipendently. This separation was meant to test the performance in the best way in different situations. The *Elbow* extimation found the optimal number of subclusters as 3. However, knowing the extimation to be quite "conservative", willing to see "what happens", we decided to run the algorithm also for a bigger number of subclusters. Results are shown in Figure 3, for different number of subclusters. We can note that the 3_D videos are the most difficult to work with, maybe because the typology of videos is too different from the other ones. Anyway, it is possible to see our model improving its performances while increasing the number of subclusters, although in a slow way. Regarding the other video types, we can notice that accuracy growth stops after a treshold in a sort of "overfitting of clusters". It is possible to see this kind of overfitting in Figure 4, in which we trained the model only with the videos from *Indoor Lights Cooperative*, but testing with all the test suites.

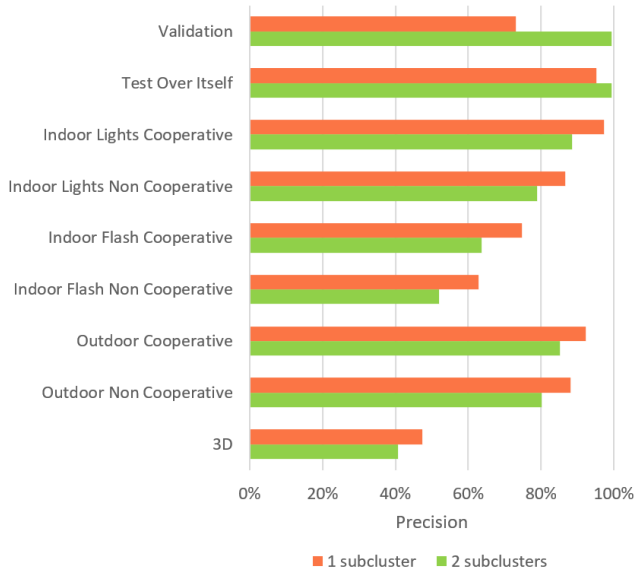


Figure 4. Statistics got training the model only on a single context.

We tried to run the algorithm for an increasing number of subclusters obtaining overfitting already with 2, making the *Elbow* extimation of 1 correct. It is possible to see a drastic reduction of the testing precision. Worst performances are usually obtained on the *Indoor Flash Non Cooperative* videos, in which the light isn't very good and subject do their best to avoid the camera.

## 4. Conclusion

### 4.1. Future Development

As we said, we preferred to use *Histogram Oriented Gradients* instead of *Convolutional Neural Network* for face detection because, even if they are less precise, they are much faster. In fact, this choice is justified by the fact that we want to implement a *Real Time* system in the future. Furthermore, we want to replace K-Means with another algorithm which could be better in terms of accuracy and performance.