Chapter 4

Fuzzy Segmentation

4.1 Introduction.

The segmentation of objects whose color-composition is not common represents a difficult task, due to the illumination and the appropriate threshold selection for each one of the object color-components. In this thesis we propose the Fuzzy C-Means algorithm application for the segmentation of such objects using as a membership criteria to a cluster the Mahalanobis distance. This distance criteria consider also the clusters distribution and have a better results compared with an simple Euclidian distance criteria clasificator. It is chosen, by the characteristics that it represents the face segmentation.

Pattern recognition techniques can be classified into two broad categories: unsupervised techniques and supervised techniques. An unsupervised technique does not use a given set of unclassified data points, whereas a supervised technique uses a dataset with known classifications. These two types of techniques are complementary. For example, unsupervised clustering can be used to produce classification information needed by a supervised pattern recognition technique. In this section, we first introduce the basics of unsupervised clustering. The fuzzy C-Means algorithm (FCM) [69], which is the best known unsupervised fuzzy clustering algorithm is then described in detail.

4.2 Unsupervised Clustering.

Unsupervised clustering is motivated by the need to find interesting patterns or groupings in a given set of data.

In the area of pattern recognition an image processing, unsupervised clustering is often used to perform the task of "segmenting" the images (i.e., partitioning pixel on an image into regions that correspond to different objects or different faces of objects in the images). This is because image segmentation can be viewed as kind of data clustering problem where each datum is

described by a set of image features (e.g., intensity, color, texture, etc) of each pixel.

Conventional clustering algorithms find a "hard partition" of given dataset based on cearting criteria that evaluate the goodness of a partition. By hard partition we mean that each datum belongs to exactly one cluster of the partition. More formally, we can define the concept hard partition as follows.

Definition 1. Let *X* be a set of datum and x_1 be an element of *X*. A partition P={C_1,C_2,...,C_L} of X is hard if and only if

$i) \forall x_i \in X \; \exists C_j \in P$ such that $x_i \in C_j$.

$ii) \forall x_i \in X \; x_i \in C_j \Rightarrow x_i \notin C_i$ where $k \neq j, C_j \in P$.

The first condition in the definition assures that the partition covers all data points in *X*, the second condition assures that all clusters in the partition are mutually exclusive.

In many real-world clustering problems, however, some data points partially belong to multiple clusters, rather than a single cluster exclusively. For example, a pixel in a magnetic resonance image may correspond to mixture of a different types of issues.

A soft clustering algorithms finds a soft partition of a given dataset based on certain criteria. In soft partition, a datum can partially belong to multiple clusters. We formally define this concept below.

Definition 2. Let *X* be a set a data, and x_1 be an element of *X*. A partition P={C_1,C_2,...,C_L} of *X* is soft if and only if the following two condition hold

$i) \forall x_i \in X \; \forall C_j \in P \; 0 \leq \mu_{C_j}(x_i) \leq 1$.

$ii) \forall x_i \in X \; \exists C_j \in P$ such that $\mu_{C_j}(x_i) > 0$.

where $\mu_{C_j}(x_i)$ enotes the degree to which x_1 belongs to cluster C_j.

A type of soft clustering of special interest is one that ensures the membership degree of a point x in all clusters adding up to one, i.e.,

$$\sum_j \mu_{C_j}(x_i) = 1 \quad \forall x_i \in X \tag{4.1}$$

A soft partition that satisfies this additional condition is called a constrained soft partition. The fuzzy c-means algorithm, which is best known as fuzzy clustering algorithm, produces a constrained soft partition.

A constrained soft partition can also be generated by a probabilistic clustering algorithm (e.g., maximum likelihood estimators). Even thought both fuzzy c-means and probabilistic clustering produce a partition of similar properties, the clustering criteria underlyng these algorithms are very different. While we focus our discution an fuzzy clustering in this section, we should point out that probabilistic clustering has also found successful real-world applications. Fuzzy clustering and probabilistic clustering are two different approaches to the problem of clustering.

The fuzzy c-means algorithm generalizes a hard clustering algorithm called the c-means algorithm, which was introduced in the ISODATA clustering method. The (hard) c-means algorithm aims to identify compact, well-separated cluster. Figure 4.1 shows a two-dimensional dataset containing compact well separated clusters. In contrast, the dataset shown in the figure 4.2 contain clusters that are not compact and well separated. Informally, a compact cluster has a ball-like shape. The center of the ball is called the prototype of the cluster. A set of cluster are well separated when any two points in a cluster are closer than the shortest distance between two clusters in different clusters. Figure 4.3 shows two clusters that are not well separated because there are points in C_2 that are closer to a point in C_1 than point in C_2. We formally define well separated clusters bellow.

Definition 3. A partition P={C_1,C_2,...,C_k} of de dataset *X* has compact separated cluster if and only if any two points in a cluster are closer than the distance between two points in different cluster, i.e, $\forall x, y \in C_P \; d(x,y) < d(z,w)$ where $z \in C_q, w \in C_r, j \neq k$, and *d* denotes a distance measure.

Assuming that a dataset contains c compact, well-separated clusters, the goal of hard c-means algorithm is twofold:

1. To find the centers of these clusters, and
2. To determine the clusters (i.e., labels) of each point in the dataset.

In fact, the second goal can easily be achieved once we accomplish the first goal, based on the assumption that clusters are compact and well separated. Given cluster centers, a point in the dataset belongs to cluster whose center is closest, i.e.,

$$x_i \in C_j \text{ if } |x_i - v_j| < |x_i - v_k| \quad k = 1, 2...., c, \; k \neq j \tag{4.2}$$

where v_j denotes the center of the cluster C_j.

In order to archive the first goal (i.e., finding the cluster centers), we need to establish a criterion that can be used to search for these cluster centers. One such criteria is the sum of the distance between points in each cluster and their center.

$$J(P,V) = \sum_{j=1}^{c} \sum_{x_i \in C_j} |x_i - v_j|^2 \tag{4.3}$$

where *V* is a vector of cluster center to be identified. This criterion is useful because a set of true cluster centers will give a minimal *J* value for a given database. Based on these observations, the hard c-means algorithm tries to find the clusters centers *V* than minimize *J*. However, J is also a function of partition P, which is determined by the cluster centers *V* according to equation 4.1. Therefore, the hard c-means algorithm (HCM) [70] searches for the true cluster center by iterating the following two step:

1. Calculating the current partition based on the current cluster.

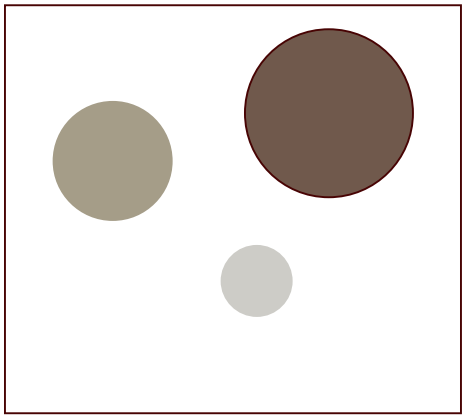

Figure 4.1: An Example of compact well separated clusters.

Figure 4.2: An example of two clusters that are not compact and well separated.

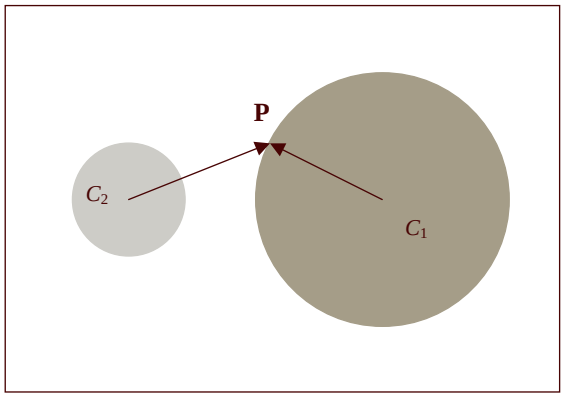

Figure 4.3: Two clusters that are compact, but not well separated.

2. Modifying the current cluster centers using a gradient decent method to minimize the J function.

The cycle terminates when the difference between cluster centers in two cycles is smaller than a threshold. This means that the algorithm has converged to a local minimum of *J*.

4.3 Fuzzy c-Means Algorithm.

The fuzzy C-Means algorithm (FCM) generalizes the hard c-mans algorithm to allow a point to partially belong to multiple clusters. Therefore, it produces a soft partition for a given dataset. In fact, it produces a constrained soft partition. To do this, the objective function J_1 of hard c-means has been extended in two ways:

The fuzzy membership degrees in clusters were incorporated into the formula, and

An additional parameter m was introduced as a weight exponent in the fuzzy membership.

The extended objective function [71], denoted J_m, is:

$$J_m(P,V) = \sum_{j=1}^{c} \sum_{x_i \in C_j} (\mu_{C_i}(x_k))^m \left|x_k - v_i\right|^2 \tag{4.4}$$

where P is fuzzy partition of the dataset X formed by C_1,C_2,...,C_k. The parameter *m* is a weight that determines the degree to which partial members of a clusters affect the clustering result.

Like hard c-means, fuzzy c-means also tries to find a good partition by searching for prototypes vi that minimize the objective function J_m. Unlike

Algorithm 2 FCM algorithm

FCM (*X*, *c*, *m*, *e*)
X: an unlabeled data set.
c: the number the clusters.
m: the parameter in the objective function.
e: a threshold for the convergence criteria.
Initialize prototype V=$\{v_1,v_2,...,v_c\}$
Repeat
$V^{\mathrm{Previous}} \leftarrow V$
Compute membership functions using equations 3.
Update the prototype, v_i in V using equation 2.
Until $\sum_{i=1}^{c} \left|v_i^{\mathrm{Previous}} - v_i\right| \leq e$

hard c-means, however, the fuzzy c-means algorithm also needs to search for membership functions μ_{C_i} that minimize J_m. To accomplish these two objectives, a necessary condition for local minimum of J_m was derived from J_m. This condition, which is formally stated below, serves as the foundation of the fuzzy c-means algorithm.

Theorem 1. Fuzzy c-means theorem. A constrained fuzzy partition $\{C_1,C_2,...,C_k\}$ can be a local minimum of objective function J_m only if the following conditions are satisfied:

$$\mu_{C_i}(x) = \frac{1}{\sum_{j=1}^{k} \left(\frac{|x-v_i|^2}{|x-v_j|^2}\right)^{\frac{1}{m-1}}} \quad 1 \leq i \leq k, x \in X \tag{4.5}$$

$$v_i = \frac{\sum_{x \in X} (\mu_{C_i}(x))^m x}{\sum_{x \in X}^{n} (\mu_{C_i}(x))^m} \quad 1 \leq i \leq k \tag{4.6}$$

Based on this theorem, FCM updates the prototypes and the membership function iteratively using equations 4.2 and 4.3 until a convergence criterion is reached. The Process is explained in algorithm 2.

Suppose we are given a dataset of six points, each of which has two features F_1 and F_2. We list the dataset in table 4.1. Assuming that we want to use FCM to partition the dataset into two clusters (i.e., the parameter *c*=2), suppose we set the parameter m in FCM at 2, and the initial prototypes to v_1=(5,5) and v_2=(10,10).

The initial membership functions of the two clusters are calculated using equation 4.2.

$$\mu_{C_1}(x_1) = \frac{1}{\sum_{j=1}^{2} \left(\frac{|x_1-v_1|}{|x_1-v_j|}\right)^2}$$

	F_1	F_2
x_1	2	12
x_2	4	9
x_3	7	13
x_4	11	5
x_5	12	7
x_6	14	7

Table 4.1: Dataset values.

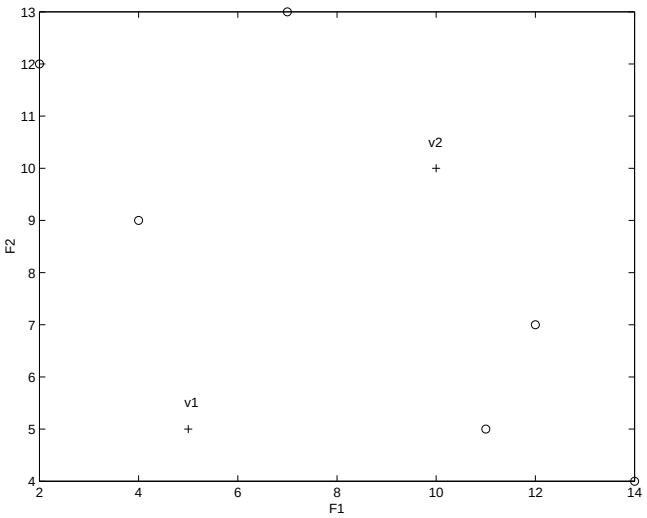

Figure 4.4: Dataset graphical representation.

$$|x_1 - v_1|^2 = 3^2 + 7^2 = 58$$

$$|x_1 - v_2|^2 = 8^2 + 2^2 = 68$$

$$\mu_{C_1}(x_1) = \frac{1}{\frac{58}{58} + \frac{58}{68}} = 0.5397$$

Similarly, we obtain the following

$$\mu_{C_2}(x_1) = \frac{1}{\frac{68}{58} + \frac{68}{68}} = 0.4603$$

$$\mu_{C_1}(x_2) = \frac{1}{\frac{17}{17} + \frac{17}{37}} = 0.6852$$

$$\mu_{C_2}(x_2) = \frac{1}{\frac{37}{17} + \frac{37}{37}} = 0.3148$$

$$\mu_{C_1}(x_3) = \frac{1}{\frac{68}{68} + \frac{68}{18}} = 0.2093$$

$$\mu_{C_2}(x_3) = \frac{1}{\frac{18}{68} + \frac{18}{18}} = 0.7907$$

$$\mu_{C_1}(x_4) = \frac{1}{\frac{36}{36} + \frac{36}{26}} = 0.4194$$

$$\mu_{C_2}(x_4) = \frac{1}{\frac{26}{36} + \frac{26}{26}} = 0.5806$$

$$\mu_{C_1}(x_5) = \frac{1}{\frac{53}{53} + \frac{53}{13}} = 0.197$$

$$\mu_{C_2}(x_5) = \frac{1}{\frac{13}{53} + \frac{13}{13}} = 0.803$$

$$\mu_{C_1}(x_6) = \frac{1}{\frac{82}{82} + \frac{82}{52}} = 0.3881$$

$$\mu_{C_2}(x_6) = \frac{1}{\frac{52}{82} + \frac{52}{52}} = 0.6119$$

Therefore, using these initial prototypes of the two clusters, membership function indicated that x_1 and x_2 are more in the first cluster, while the remaining points in the dataset are more in the second cluster.

The FCM algorithm then updates the prototypes according to equation 4.3.

$$v_1 = \frac{\sum_{k=1}^{6} (\mu_{C_1}(x_k))^2 x_k}{\sum_{k=1}^{6} (\mu_{C_1}(x_k))^2}$$

$$= \left(\frac{7.2761}{1.0979}, \frac{10.044}{1.0979} \right)$$

= (6.6273, 9.1484)

$$v_2 = \frac{\sum_{k=1}^{6} (\mu_{C_2}(x_k))^2 x_k}{\sum_{k=1}^{6} (\mu_{C_2}(x_k))^2}$$

= (9.7374, 8.4887)

The updated prototype v_1, as is shown in fig 5, is moved closer to the center of the cluster formed by x_1, x_2 and x_3; while the updated prototype v_2 is moved closer to the cluster formed by x_4, x_5 and x_6.

We wish to make a few important points regarding the FCM algorithm:

- FCM is guaranteed to converge for *m*>1. This important convergence theorem was established in 1980 [72].

- FCM finds a local minimum (or saddle point) of the objective function J_m. This is because the FCM theorem (theorem 1) is derived from the condition that the gradient of the objective function J_m should be 0 at an FCM solution, which is satisfied by all local minima and saddle points.

- The result of applying FCM to a given dataset depends not only on the choice of parameters m and c, but also on the choice of initial prototypes.

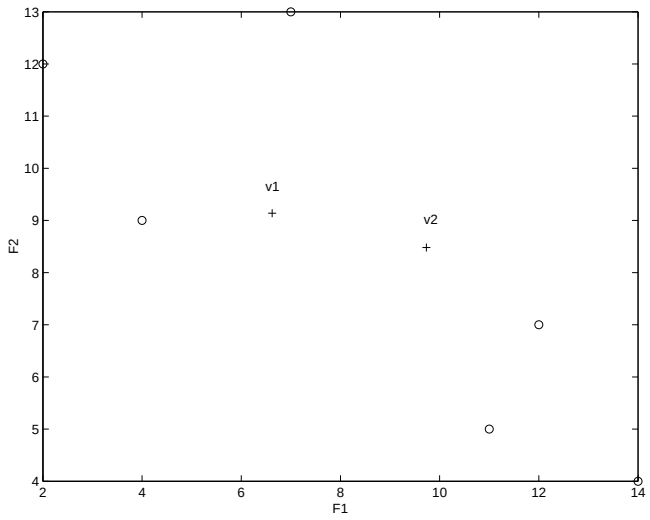

Figure 4.5: Prototype updating.

4.4 Mahalanobis distance.

The Mahalanobis distance is based on correlations between variables by which different patterns can be identified and analysed. It is a useful way of determining similarity of an unknown sample set to a known one. It differs from Euclidean distance in that it takes into account the correlations of the data set.

Formally, the Mahalanobis distance from a group of values with mean $\mu = (\mu_1, \mu_2, \mu_3, ..., \mu_p)$ and covariance matrix Σ for a multivariate vector

$x = (x_1, x_2, x_3, ..., x_p)$ is defined as:

$$D_M(x) = \sqrt{(x-\mu)'\Sigma^{-1}(x-\mu)} \tag{4.7}$$

Mahalanobis distance can also be defined as dissimilarity measure between two random vectors $\vec{x}$ and $\vec{y}$ of the same distribution with the covariance matrix Σ:

$$d(\vec{x}, \vec{y}) = \sqrt{(\vec{x}-\vec{y})'\Sigma^{-1}(\vec{x}-\vec{y})} \tag{4.8}$$

If the covariance matrix is the identity matrix then it is the same as Euclidean distance. If covariance matrix is diagonal, then it is called *normalized Euclidean distance*:

$$d(\vec{x}, \vec{y}) = \sqrt{\sum_{i=1}^{p} \frac{(x_i - y_i)^2}{\sigma_i^2}} \tag{4.9}$$

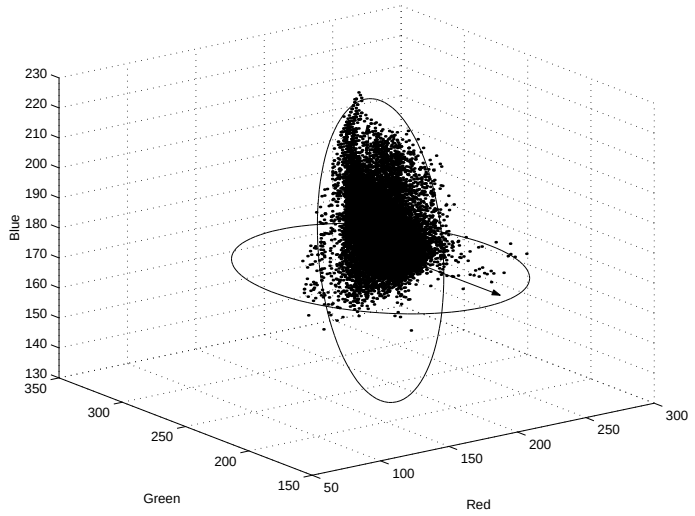

Figure 4.6: Example of a cluster data distribution used to implement color segmentation.

where σ_i is the standard deviation of the x_i over the sample set. Thus, the Mahalanobis distance is used as a memberschip criteria considering the (4.9) equation. The figure 4.6 shows a example of a cluster data distribution used to implement color segmentation.

4.5 Matlab tools.

The Fuzzy Logic Toolbox is equipped with some tools that allow to find clusters in input-output training data. We can use the cluster information to generate a Sugeno-type fuzzy inference system that best models the data behaviour using a minimum number of rules. The rules partition themselves according to the fuzzy qualities associated with each of the data clusters. This type of FIS generation can be accomplished automatically using the command line function, genfis2.

The Fuzzy Logic Toolbox command line function `fcm` starts with an initial guess for the cluster centers, which are intended to mark the mean location of each cluster. The initial guess for these cluster centers is most likely incorrect. Additionally, `fcm` assigns every data point a membership grade for each cluster. By iteratively updating the cluster centers and the membership grades for each data point, `fcm` iteratively moves the cluster centers to the right location within a data set. This iteration is based on minimizing an objective function that represents the distance from any given data point to a cluster center weighted by that data points membership grade.

`fcm` is a command line function whose output is a list of cluster centers and several membership grades for each data point. We can use the information returned by fcm to help we build a fuzzy inference system by creating membership functions to represent the fuzzy qualities of each cluster.

Now, the `fcm` function will be described:

```
[center, U, obj_fcn] = fcm(data, cluster_n)
```

The input arguments of this function are:

data: data set to be clustered; each row is a sample data point.

cluster_n: number of clusters (greater than one).

The output arguments of this function are:

center: matrix of final cluster centers where each row provides the center coordinates.

U: final fuzzy partition matrix (or membership function matrix).

obj_fcn: values of the objective function during iterations.

4.6 Implementation.

To implement the segmentation system it is necessary to use as data an image of the object to be segment (in our case a person face). Each pixel of the image is coded in three components represented respectively with the red, green and blue color.

The next code assign to each pixel its respective color component dataset represented by VP with the `fcm` function format (that means the pixel data is presented in row form). Something that one must not forget is that the image dataset is obteined in integer format but to work with it will be necesary to change it to double format.

```
R=Im(:,:,1);
G=Im(:,:,2);
B=Im(:,:,3);
[m,n]=size(R);
indice=m*n;
erik=0;
for a1=1:m
for an=1:n
data=R(a1,an);
data1=G(a1,an);
```

```
data2=B(a1,an);
num=num+1;
VR(num)=data;
VG(num)=data1;
VB(num)=data2;
end
end
VP=[VR;VG;VB];
VP=double(VP);
```

There is an important parameter in the `fcm` function, this is the cluster number in wich one wants to divide the presented dataset, this parameter should be founded heuristically. For this example its value was 7. If this value is big, then the system generalization is not good enough and if is very small then the neighbor colors can be confused. The matlab code to find the image clusters is:

```
[center,U,of]=fcm(VPT,7);
```

After used this function we have in the variable *center* the clusters centers, which will be used to classify the pixels belonging to the interest class. In order to apply the Mahalanobis distance as a membership criteria, is necessary, first to find the standard desviation founded to the 7 clusters. In our case the interest class is the class that represent the flesh color. In this thesis the classification is achieved calculating the minimum Mahalanobis distance from each pixel to the cluster data. The code in C++ to achieve that in real time is:

```
for(int i=1;i<=sizeImage;i++)
{
b=*pBuffer;
pBuffer++;
g=*pBuffer;
pBuffer++;
r=*pBuffer;
pBuffer++;
dist=sqrt(((abs(r-176.1448)*abs(r-176.1448))+(abs(g-115.1489)...
...*abs(g-115.1489))+(abs(b-20.4083)*abs(b-20.4083)))/10.83);
if (dist<45)
temp1=255;
else
temp1=0;
```

```
pBuffer--;
pBuffer--;
pBuffer--;
*pBuffer=temp1;
pBuffer++;
*pBuffer=temp1;
pBuffer++;
*pBuffer=temp1;
pBuffer++;
}
pBuffer=pixel;
```

The previous code considers that sizeImage is the image size and also that the flesh color class centroid values are 176.1448 for red, 115.1489 for green and 20.4083 for blue, also a cluster standard desviation of 10.83 and a similarity criteria minor to 45.

4.7 Results.

The obtained results using the Mahalanobis fuzzy C-Means as a segmentation method is quite good for objects whose colors are not trivial. A fast training is an important advantage obtained with the use of Fuzzy C-Means matlab tools as well as the easy change of its parameters. This allows to experiment with different operation conditions like changing the class number until the system robustness is satisfied.

The figure 4.7 shows the cluster distribution obtained by training the `fcm` function. The figure 4.8 shows the obatined clusters centers. While the figure 4.9 shows a sequence and their respective segmentation using the following cluster center values for the class flesh color: red=176.1448, green=115.1489 and blue =20.4083.

The figure 4.10 (left), shows the result of implement the color-face segmentation, using only the euclidian distance as a cluster membership criteria and figure 9 (right) the result using the Mahalanobis distance as a cluster membership criteria.

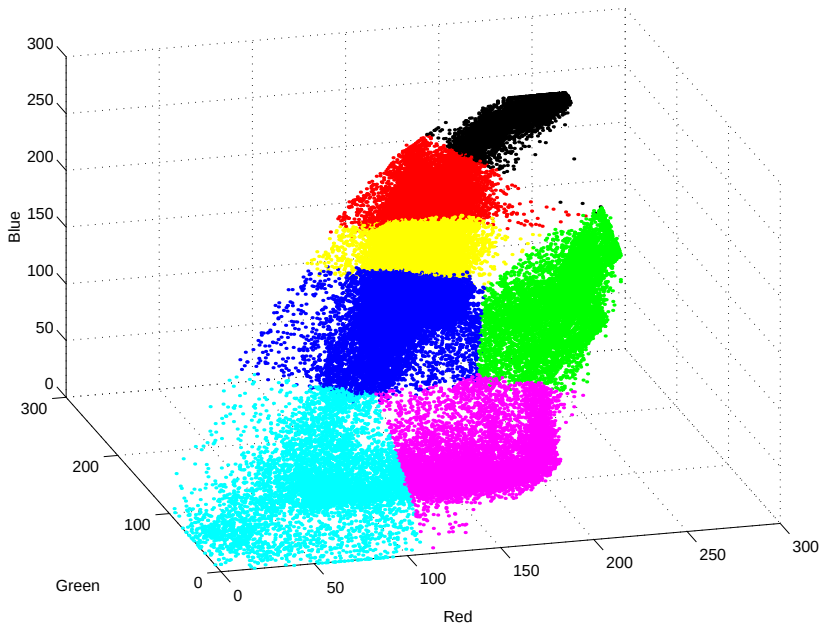

Figure 4.7: Cluster distribution.

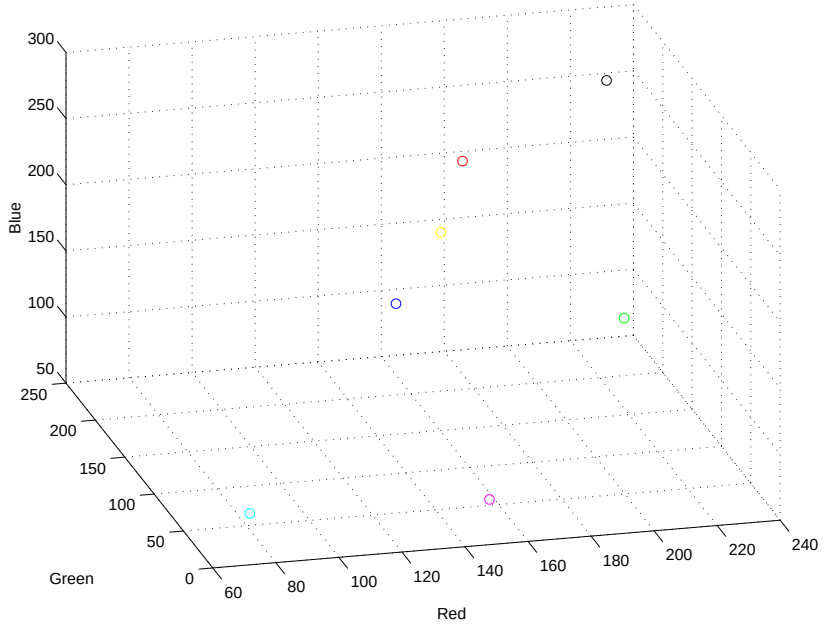

Figure 4.8: Clusters centers.

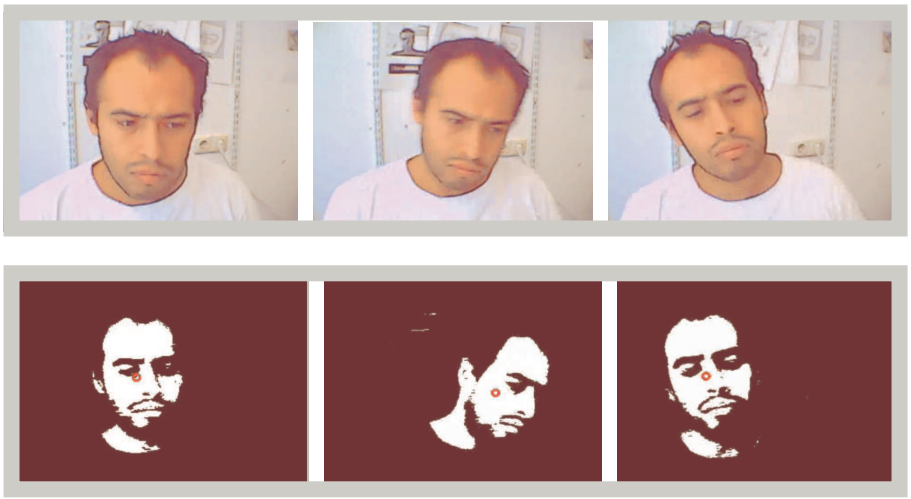

Figure 4.9: (Up) Original images, (Down) Segmented images.

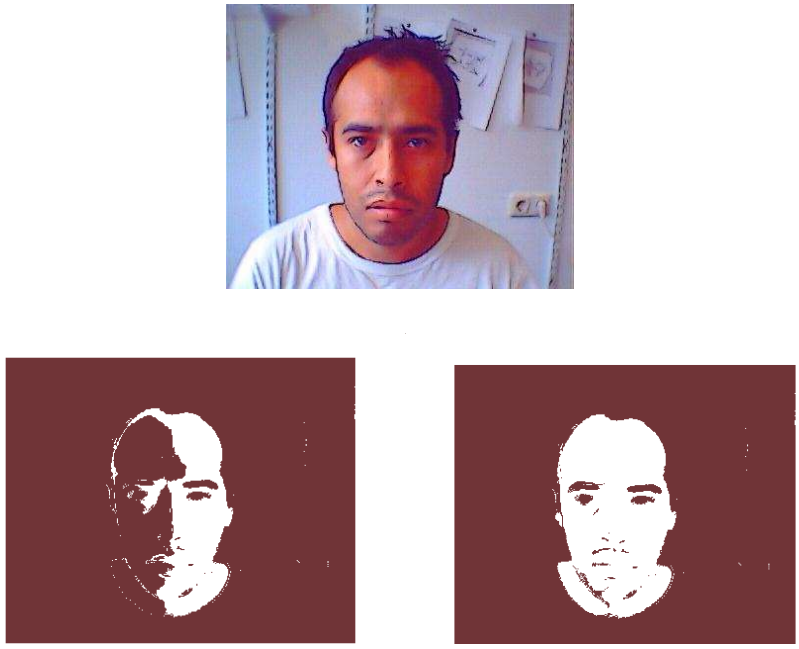

Figure 4.10: (Up) Original ,(left) Euclidian distance, (rigth) Mahalanobis distance.