Ritsumeikan University
Emergent Systems Lab

Infinite Bayesian Model for Image Segmentation

by

David Esparza Alba

Supervisor: PhD. Tadahiro Taniguchi

Kusatsu, Shiga. Japan. November 5^{th}, 2011

Contents

Chapter 1

Introduction

Machine Learning is a branch of artificial intelligence that estimates unknown data based on certain information. Machine Learning is widely used on supervised learning and unsupervised learning applications.

In supervised learning algorithms for each observed value x we have a target value y, and the goal is to try to estimate the target value y^* for any x^*. One example of supervised learning is classification. Suppose that a bank wants to know if it is feasible or not to give a loan to a certain client depending on the money he earns and his age. Given some data of previous loan decisions it is possible for the bank to know if it is feasible or not to give the loan to the client. Another example of supervised learning is regression. Consider the case we want to know the price of a house that has certain characteristics like size, number of rooms, location, etc. Given a set of house prices it is possible to find a function that estimates the correct price of the house.

In the case of unsupervised learning we only have a set of observed values X and the goal is to know as much as we can of that data set, like how is the data distributed in the space.

In the past, it was difficult for researches to obtain large data sets, they used to worked with small data sets of their own invention to test their algorithms, but in the last few years with the use of Internet large data sets have become available and the use of machine learning techniques have taken an important role to analyze that amount of data. In the following years is expected that the number of machine learning applications continue growing.

The purpose of this research is to use advanced machine learning techniques to

estimate the number of clusters in an image, in order to be capable of identify objects or part of objects that share certain characteristics inside the image.

The content of this project is divided in 5 chapters.

This chapter talks about the importance of machine learning in the present, and a brief explanation about what this project is about.

Chapter 2 contains two important topics for the development of this project. Image Filtering and Image Segmentation. Here we explain some implemented methods and their results.

Chapter 3 is about Non-parametric methods, including a brief explanation of the Dirichlet Process and the Chinese Restaurant Process. Also this chapter contains how a clustering method using Gibbs Sampling was used for image segmentation.

In chapter 4 we talk about two steps based in graph search that helped us to do a better segmentation.

Finally chapter 5 has the conclusions of this project.

Chapter 2

Image Processing

Image processing consists in analyze and manipulate certain characteristics of an image, such as color, texture, intensity, etc. The applications of image processing algorithms are varied, they can go from medical image analysis to identify abnormalities to robotics.

In this chapter we are going to talk about two concepts that were important in the development of this project.

1. Filters
2. Image Segmentation

2.1 Filters

In most of real applications, images present some amount of noise that complicates working with them. Image processing algorithms are very sensitive to noise and for that reason is necessary to eliminate or reduce that noise.

Filters are used to reduce the amount of noise in an image, and there are different kinds of filters, some filters works better than others depending on the kind of noise.

Noise can be defined as a variation of intensity or color in an image. Noise can occur for several reasons, such as luminosity, wrong functionality of the camera, the condition of the environment, etc.

When working with images, we must assume that noise can be presented. Then, the value of a pixel in position (x, y), is given by the following expression:

$$g(x, y) = f(x, y) + \epsilon(x, y)$$

where $f(x, y)$ is the real value of the pixel and $\epsilon(x, y)$ represents the noise in that pixel.

Image with Noise

47	63	121	15
43	89	127	254
10	13	7	45
28	7	35	25

Original Image

45	67	120	12
45	89	128	255
10	15	12	43
27	7	35	26

Noise

2	-4	1	3
-2	0	-1	-1
0	-2	-5	2
1	0	0	-1

Figure 2.1: At the left image with noise f. At the right original image f, and noise ϵ.

There are several kinds of noise, such as:

- Gaussian Noise
- "Salt and Pepper" Noise
- White Noise
- Film Grain
- etc.

We are not going to go deeper into this subject, instead we are going to focus in methods that allow us to reduce the amount of noise in an image.

Filters help us to reduce the amount of noise and also are capable to detect some important features like edges.

The value of the pixels change in the filtering process. These new values are obtained using information of the neighbor pixels. Most of the filters use kernels to

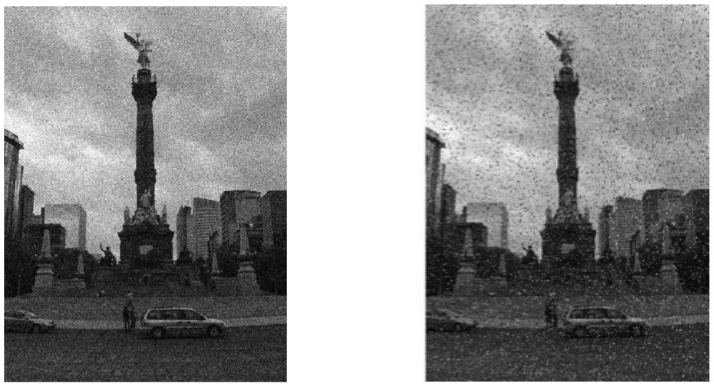


Figure 2.2: At the left image with Gaussian noise. At the right image with "Salt and Pepper" noise.

obtain the new values of the pixels. A kernel is a $k \times l$ matrix that is convolved with the image. The convolution operator is defined as

$$g = f \star h$$

For our case, f represents the image and h the kernel. The resulting image is given by g. Each pixel value of the image is obtained with the following formula

$$g(i,j) = \sum_{l} \sum_{k} f(i+k, j+l)h(k,l)$$

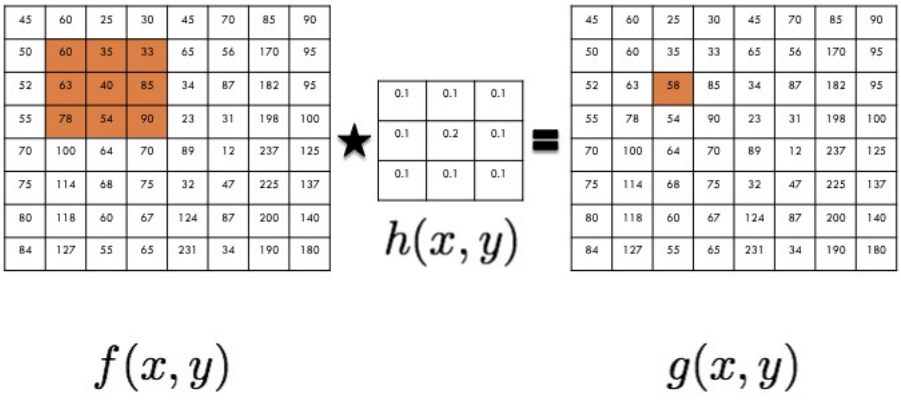


Figure 2.3: Convolution of image f and kernel h

2.1.1 Mean Filter

This filter computes the mean of all elements in the window and assigns that value to the new pixel. A 3×3 kernel is shown in figure 2.4.

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

Figure 2.4: A 3×3 kernel for mean filter

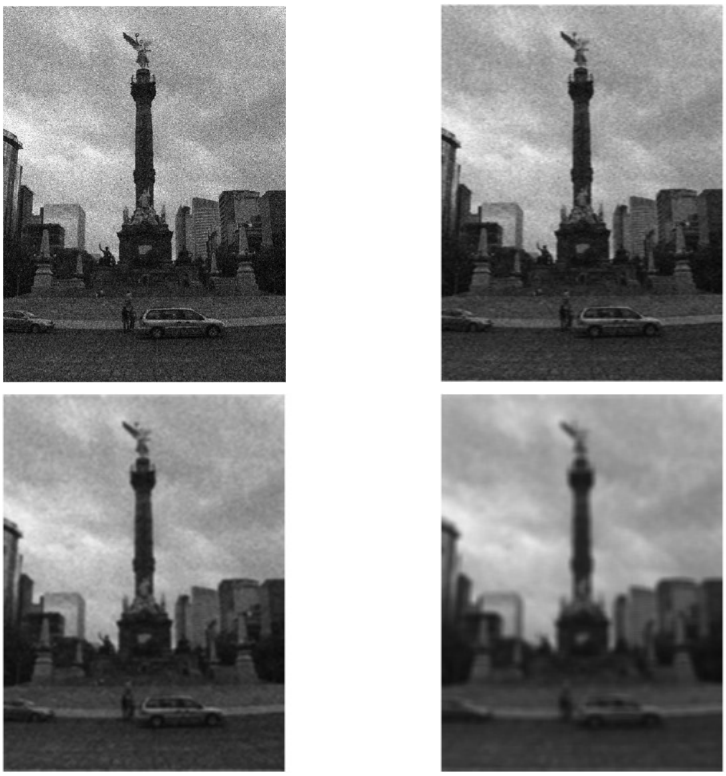


Figure 2.5: At top-left image with Gaussian noise, then the result after applying mean filter of size 3×3, 7×7 and 15×15

2.1.2 Median Filter

The median filter assigns to the new pixel the value of the median of all elements in the window. There is no need of using kernel for this filter.

Be S the set of n elements inside the window sorted according to their value. The new pixel value correspond to the middle element of S, which is obtained by the following expression:

$$g(x,y) = \begin{cases} S_{n/2} & \text{if n is odd} \\ \frac{S_{n/2}+S_{n/2-1}}{2} & \text{if n is even} \end{cases}$$

Consider the following image

45	60	25	30	45	70	85	90
50	60	35	33	65	56	170	95
52	63	40	85	34	87	182	95
55	78	54	90	23	31	198	100
70	100	64	70	89	12	237	125
75	114	68	75	32	47	225	137
80	118	60	67	124	87	200	140
84	127	55	65	231	34	190	180

Figure 2.6: Image to be filtered, the yellow square is called "window"

Here the set of sorted elements is $S = \{33, 45, 40, 54, 60, 63, 78, 85, 90\}$ and according to the formula above, the middle element is 60. Then, the new image will look as follows:

45	60	25	30	45	70	85	90
50	60	35	33	65	56	170	95
52	63	60	85	34	87	182	95
55	78	54	90	23	31	198	100
70	100	64	70	89	12	237	125
75	114	68	75	32	47	225	137
80	118	60	67	124	87	200	140
84	127	55	65	231	34	190	180

Figure 2.7: Image after applying the median filter

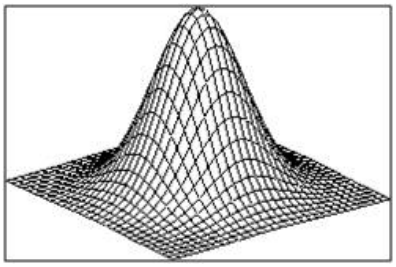
2.1.3 Gaussian Filter

The Gaussian filter convolves the image with a Gaussian function with mean μ and standard deviation σ. This filter uses a Gaussian function as a kernel, where the element at the center has the largest value and the other values decrease symmetrically

Figure 2.8: Median filter with kernel size of 3×3, 5×5 and 9×9

as the distance from the center increases. The element at the center is located at $(0, 0)$, then the kernel is obtained with

$$h(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$



0.00	0.01	0.02	0.01	0.00
0.01	0.06	0.10	0.06	0.01
0.02	0.10	0.16	0.10	0.02
0.01	0.06	0.10	0.06	0.01
0.00	0.01	0.02	0.01	0.00

Figure 2.9: A 5×5 Gaussian kernel with $\sigma = 1$

2.1.4 Prewitt Filter

Prewitt filter makes use of the first derivatives of the image horizontally, vertically and diagonally. The derivative in one direction is given by

Figure 2.10: At top-left image with Gaussian noise, then after Gaussian filter with size and stardard deviation of $5 \times 5, \sigma = 1$, $9 \times 9, \sigma = 3$ and $9 \times 9, \sigma = 2$

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h}$$

Using the information of the nearest neighbors, a value of $h = 1$ is chosen. Then

$$f'(x) = \frac{f(x+1) - f(x-1)}{2}$$

The kernel for the horizontal direction is

-1/2	0	1/2

If we multiply by two the entire kernel and add the kernels in diagonal direction, we obtain the kernel in the x-direction.

$$G_x = \begin{array}{|c|c|c|} \hline -1 & 0 & 1 \\ \hline -1 & 0 & 1 \\ \hline -1 & 0 & 1 \\ \hline \end{array}$$

We do the same in the y-direction to obtain G_y. Then, the new value of the pixel in position (x, y) is given by the magnitude of the gradient.

$$g(x, y) = \sqrt{(f \star G_x)^2 + (f \star G_y)^2}$$

Figure 2.11: Image after applying the Prewitt filter

2.1.5 Sobel Filter

Sobel filter is a weighted version of the Prewitt filter, making emphasis in the horizontal and vertical directions. Here the kernels for the x and y directions are defined by

$$G_x = \begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array} \qquad G_y = \begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline \end{array}$$

The value of the pixel in position (x, y) is obtained with

$$g(x, y) = \sqrt{(f \star G_x)^2 + (f \star G_y)^2}$$

2.1.6 Laplacian Filter

The Laplacian filter uses second order derivatives to obtain the new value of each pixel. The firs order derivative in one dimension is given by

$$f'(x) = \frac{f(x + h) - f(x)}{h}$$

Then, the second derivative would be

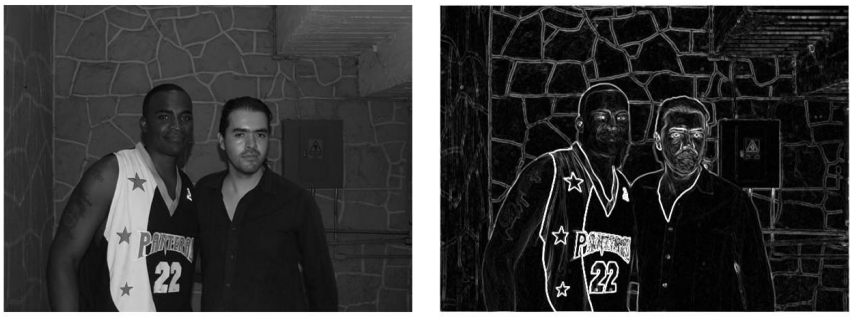


Figure 2.12: Image after applying the Sobel filter

$$f''(x) = \frac{f'(x+h) - f'(x)}{h}$$

Which can expressed as:

$$f''(x) = \frac{\frac{f(x+h)-f(x)}{h} - \frac{f(x)-f(x-h)}{h}}{h}$$

$$= \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}$$

Setting $h = 1$ we have

$$f''(x,y) = f(x+1) - 2f(x) + f(x-1)$$

Then, the kernel in one direction is

1	-2	1

The Laplacian is defined by the sum of the second partial derivatives

$$L(x,y) = \frac{\partial^2 g(x,y)}{\partial x^2} + \frac{\partial^2 g(x,y)}{\partial y^2}$$

Adding both kernels we obtain

$$G_{xy} = \begin{array}{|c|c|c|}\hline 0 & 1 & 0 \\ \hline 1 & -4 & 1 \\ \hline 0 & 1 & 0 \\ \hline \end{array}$$

Figure 2.13: Image after applying the Laplacian filter

One possible application for the laplacian filter is "sharpening" an image. The goal is to highlight fine details in the image. Consider the process shown in figure 2.14.

Figure 2.14: Sharpening process

In result, if we add the result of the Laplacian filter to the original image, we can highlight the edges of the image. See image 2.15.

Figure 2.15: Example of sharpening

2.2 Image Segmentation

Image segmentation refers to the act of grouping pixels that share certain characteristics like color, intensity, texture, etc. The main goal of image segmentation is to be able to identify different objects in an image, where each segmented region represents an object or a part of an object.

There a many applications of image segmentation. It can be applied to face recognition systems, object recognition, medical analysis, etc.

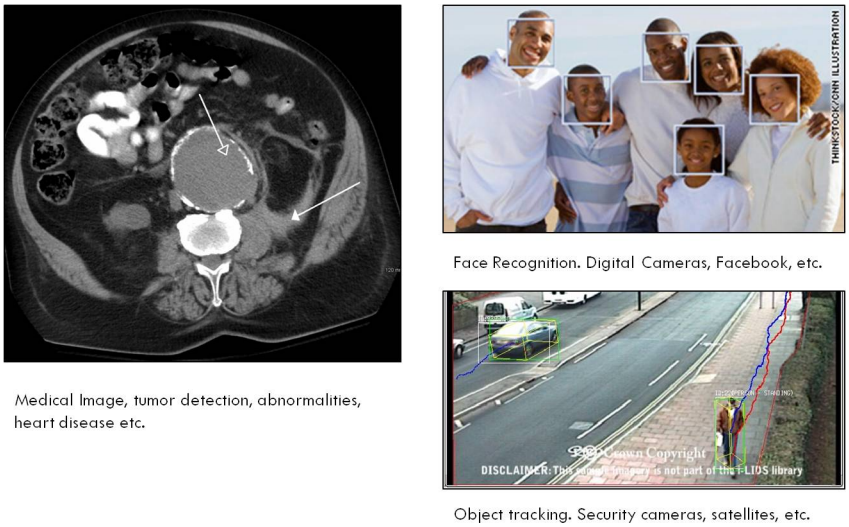


Figure 2.16: Possible application for image segmentation algorithms

Here we are going to talk about a specific method called "Minimum Spanning Tree Segmentation Method" in order to understand more easily what image segmentation is.

2.2.1 MST Segmentation

Minimum Spanning Tree Segmentation (MST Segmentation) is graph based segmentation method. Before the explanation of this method, we have to define some concepts that are important to know.

Tree. Is a non-empty set of vertices and edges, where any two vertices are connected by a single path. A tree of N vertices has $N - 1$ edges.

Weighted Graph. A graph is a weighted graph if the edges have certain cost associated to them.

Spanning Tree. A spanning tree of a connected, undirected graph can be defines as the minimal set of edges that connect all vertices.

Minimum Spanning Tree. In a weighted undirected graph, the MST is the spanning tree of minimum weight.

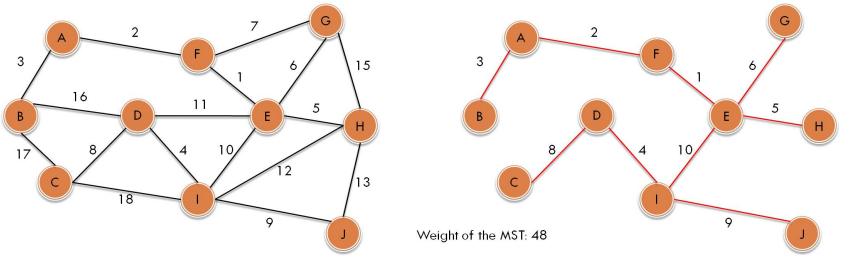


Figure 2.17: MST of graph

In order to find the MST in a graph we used the Krusal's algorithm, which consists of sorting all the edges according to their weight in non-decreasing order, and the start adding from top to bottom those vertices that don't form a cycle.

To find the MST of an image, we have to visualize the image as a undirected graph, where each pixel represents a vertex and there is an edge between each two neighbor pixels, then each pixel has 8 edges associated to it, except those in the border. The weight of and edge e is defined as:

$$e.w = |e.u - e.v|,$$

where $e.w$ represents the weight of edge e, and $e.u$ and $e.v$ are the two vertices connected by e.

The main idea of the segmentation method is to divide and image into multiple MST's. Define $C = \{C_1, ..., C_K\}$ as the set of MST's in the image, and let $Int(C_k)$ be the internal difference of the k^{th} MST, which is defined by

$$Int(C_k) = \max_{e \in C_k} e.w$$

The Minimum Internal Difference between two components C_a and C_b is expressed by

$$MInt(C_a, C_b) = \min(Int(C_a) + \tau(C_a), Int(C_b) + \tau(C_b)),$$

where $\tau(C_k) = \alpha/|C_k|$. Here $|C_k|$ represents the number of elements in component C_k, and α is a constant parameter defined by the user. Algorithm 1 shows the process of the Kruskal's algorithm applied to the image segmentation problem,

where the function $Find_Set(a)$ returns the index of the MST that vertex a belongs to, and $Union_Set(a, b)$ merge two different MST's into a single one.

Algorithm 1 Kruskal's Algorithm

$E = \{e_1, ..., e_M\}$: set of all edges
$MST \leftarrow$ empty
sort E in non-descending order
$c \leftarrow 0$
$i \leftarrow 0$
while $c < N - 1$ **do**
  $a \leftarrow Find_Set(e_i.u)$
  $b \leftarrow Find_Set(e_i.v)$
  if $a \neq bande_i.w < MInt(a, b)$ **then**
    add e_i to MST
    $Union_Set(a, b)$
    $c \leftarrow c + 1$
  end if
end while

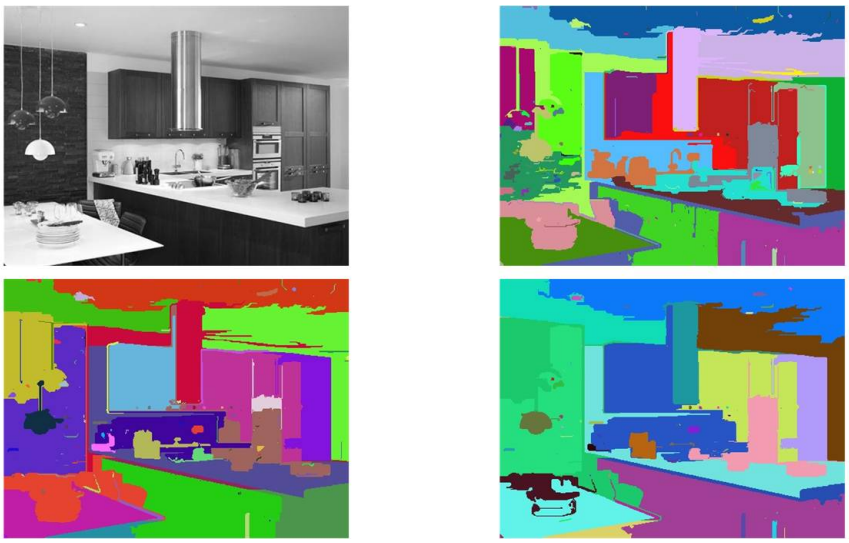


Figure 2.18: At top left the image to be segmented, then the result after applying the MST segmentation method with $\alpha = 10000$, $\alpha = 15000$ and $\alpha = 20000$

Chapter 3

Clustering and Non-parametric Methods

Given a data set of observations $X = \{x1, ..., x_N\}$ and nothing else, the task is to assign each one of these observations to clusters, so that the elements in the same cluster are more similar to each other than those in other clusters.

In other words, what we want to do is to estimate the distribution of the data in the space, in order to understand in a better way the behavior of that data and to predict future events.

Clustering is being used in many applications, one example is social networks, where clustering algorithms are used to identify groups of people. Also is used in companies like Amazon to make suggestions of products to their customers. In this project we used clustering for image segmentation, trying to identify group of pixels with similar color.

Most of clustering algorithms use a fixed number of clusters, but this is a limitation for the algorithm, because, we really don't know how is the data distributed in the space.

Non-parametric methods try estimate the number of clusters in the space. Nowadays these methods are attracting increasing interest. This because the amount of available data is becoming bigger and bigger with the use of the Internet. Following we talk about some clustering algorithms and non-parametric methods and how they were applied into our image segmentation problem.

3.1 Non-parametric Methods

As we said, non-parametric methods try to estimate the number of clusters. There is a confusion about what non-parametric means, it doesn't means the absence of parameters, instead it refers that the number of parameters is unbounded, in can increase infinitely. Here we explain two simple non-parametric methods for density estimation: The histogram method and Kernel Density Estimators. Then we talk about the Dirichlet Process and the Chinese Restaurant Process, which was the method used for our project.

3.1.1 Histogram Method

Consider a continuous variable x. Standard histograms divide x into distinct bins of size Δ_i. Be n_i the number of observations of x falling in bin i. If there is a total of N observations, the probability (height) of each bin is given by

$$p_i = \frac{n_i}{N\Delta_i}$$

where

$$\int p(x)dx = 1$$

For this case consider all bins have the same size Δ and $N = 300$. The observations are drawn from a Gaussian mixture model of three Gaussian functions in one dimension.

If the value if Δ is too small, the height of the bins changes abruptly, resulting in bad density estimation.

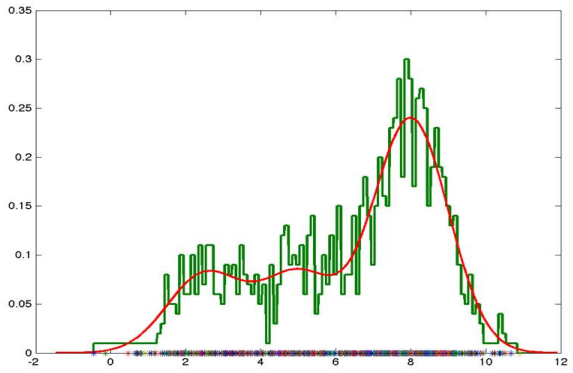


Figure 3.1: Result of the histogram method using $\Delta = 0.1$

On the other hand, if the value of Δ is too big, the height of the bins remains constant for long intervals and the resulting model turns to be quite different from the original.

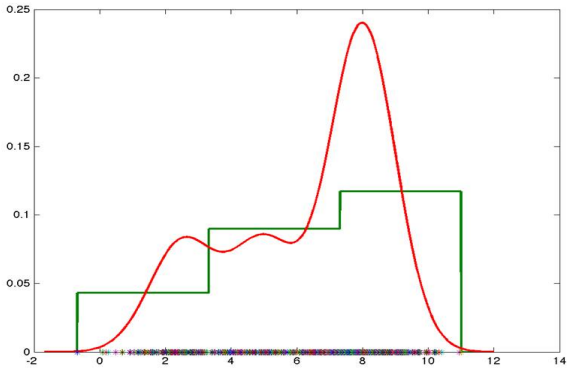


Figure 3.2: Result of the histogram method using $\Delta = 4$

In conclusion, the value of Δ has to be well defined in order to obtain the right model. Then, our problem consists on finding the right value for Δ.

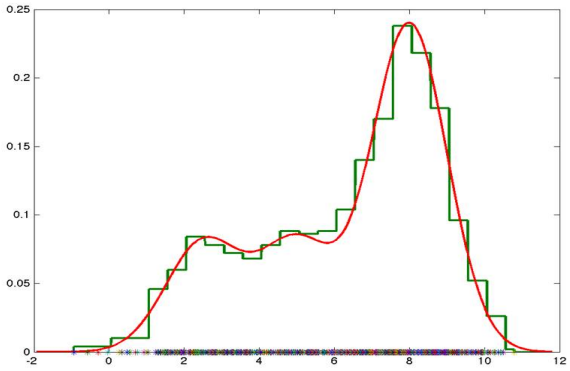


Figure 3.3: Result of the histogram method using $\Delta = 0.5$

3.1.2 Kernel Density Estimators

Consider a region R that contains x in a D-dimensional space. The probability mass associated with this region is given by

$$P = \int_R p(x)dx$$

Suppose we have a collection of N observations, and each one of them has a probability P of falling within R. Then, the expected number of points that lie inside R is expressed by

$$K = NP$$

If the probability density $p(x)$ is roughly constant, then

$$P \approx p(x)V$$
$$p(x) = \frac{K}{NV}$$

where V is the volume of R.

Now we have two choices, we can fix K and obtain V (K-nearest neighbor method), or we can fix V and find K (Kernel Approach).

Kernel Approach

In order to count the number of points falling in region R defined as a small hypercube, it is convenient to define the function:

$$k(u) = \begin{cases} 1 & \text{if } |u_i| \leq 1/2 \text{ for } i = 1, ..., D \\ 0 & \text{otherwise} \end{cases}$$

where $k(i)$ is known as the kernel function, and the value of $k((x - x_n)/h)$ will be 1 if the point x_n lies inside a cube of side h centered at x.

The number of points lying inside the cube is

$$K = \sum_{n=1}^{N} k\left(\frac{x - x_n}{h}\right),$$

then

$$p(x) = \frac{1}{N}\sum_{n=1}^{N} \frac{1}{h^D} k\left(\frac{x - x_n}{h}\right),$$

where $V = h^D$ is the volume of the hypercube of side h in dimension D.

Then we have N cubes centered in the N data points $(x_1, ..., x_n)$. If we use a one dimensional Gaussian as a kernel to get a smoother density model, we obtain:

$$p(x) = \frac{1}{N}\sum_{n=1}^{N} \frac{1}{(2\pi h^2)^{1/2}} e\left\{-\frac{(x - x_n)^2}{2h^2}\right\},$$

where h represents the standard deviation of the Gaussian function. The division by N makes the density to stay correctly normalized.

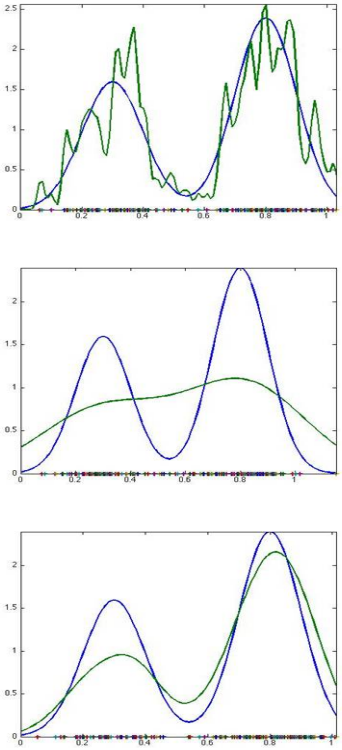


Figure 3.4: Estimation of density by Kernel Approach using $N = 200$ and $h = 0.01$, $h = 0.2$ and $h = 0.08$ respectively

K-nearest Neighbors

The goal is to determine the volume V for a fixed number of points K. Consider a small sphere centered on the point x at which we want to estimate the density $p(x)$, then

$$p(x) = \frac{K}{NV}$$

Allow the radius of the sphere to grow until it contains precisely K data points. So now the value of K will govern the degree of smoothing. See figure 3.5.

3.1.3 Dirichlet Process

The Dirichlet distribution is defined as:

$$Dir(q|\alpha) = \frac{\Gamma\left(\sum_{i=1}^{m} \alpha_i\right)}{\prod_{i=1}^{m} \Gamma(\alpha_i)} \prod_{i=1}^{m} q_i^{\alpha_i - 1},$$

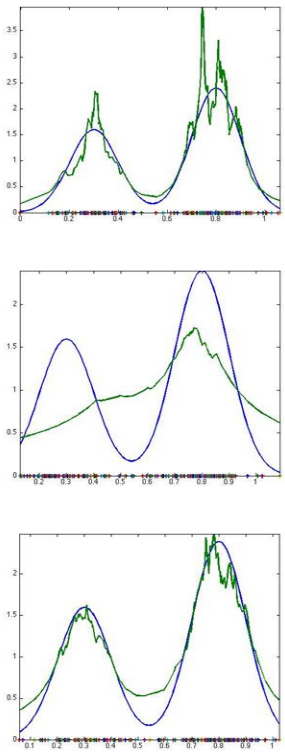


Figure 3.5: Estimation of density by the K-nearest Neighbor method using $N = 200$ with $K = 15$, $K = 100$ and $K = 30$ respectively

where $q = (q_1, ..., q_m)$, $0 \leq q_i \leq 1$ and $\sum_{i=1}^{m} q_i = 1$. The mean and variance of the distribution is given by

$$\mathbb{E}[q_k] = \frac{\alpha_k}{\widehat{\alpha}},$$

$$var[q_k] = \frac{\alpha_k(\widehat{\alpha} - \alpha_k)}{\widehat{\alpha}^2(\widehat{\alpha} + 1)},$$

where

$$\widehat{\alpha} = \sum_{i=1}^{m} \alpha_i$$

Is important to mention that the Dirichlet distribution is a conjugate prior of the multinomial distribution, then

$$
\begin{aligned}
posterior &\propto likelihood \times prior \\
p(q|z,\alpha) &\propto p(z|q)p(q|\alpha) \\
&\propto (q_1^{z_1},...,q_m^{z_m})(q_1^{\alpha_1-1},...,q_m^{\alpha_m-1}) \\
&\propto (q_1^{\alpha_1+z_1-1},...,q_m^{\alpha_m+z_m-1}) \\
&\propto Dir(\alpha_1+z_1,...,\alpha_m+z_m)
\end{aligned}
$$

3.1.4 Chinese Restaurant Process

Imagine a restaurant with K tables, where $K \to \infty$. Customers walk in and choose a table following a Dirichlet distribution $Dir(\alpha_1/K + N_1, ..., \alpha_K/K + N_K)$, where N_k is the number of people sitting at table k.

The N^{th} customer will choose an occupied table with probability

$$p(q_k) = \frac{N_k}{N-1+\alpha},$$

where N is the number of people already in the restaurant.

Now suppose that the N^{th} customer wants to eat in a new table alone. What is the probability of choosing a new table? Well, if there are L occupied tables, the number of empty tables would be $Q = K - L$. Then, the probability of choosing an empty table is given by

$$
\begin{aligned}
p(q_{new}) &= \sum_{k=1}^{Q} \lim_{K\to\infty} \frac{\alpha/K}{N-1+\alpha} \\
&= \frac{\alpha}{N-1+\alpha} \sum_{k=1}^{Q} \lim_{K\to\infty} \frac{1}{K} \\
&= \frac{\alpha}{N-1+\alpha} \lim_{K\to\infty} \frac{Q}{K} \\
&= \frac{\alpha}{N-1+\alpha} \lim_{K\to\infty} \frac{K-L}{K} \\
&= \frac{\alpha}{N-1+\alpha}
\end{aligned}
$$

3.2 Clustering

Clustering refers to identify group of observations that share similar characteristics, for this specific case we want to identify groups of pixels with similar color. To do this we can visualize an image as a 3-dimensional space, where each one of the colors

rgb (Red, Green and Blue) represent one dimension.

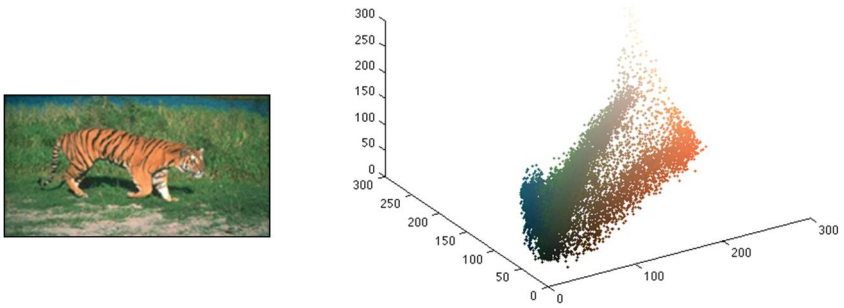


Figure 3.6: Distribution of pixels in a 3-dimensional space (rgb space)

We need to associate each pixel to a specific cluster (hard clustering). Each cluster is represented by a Gaussian distribution with mean μ_k and precision matrix Λ_k. Also each Gaussian has a mixing proportion parameter π_k associated to it, in order to maintain the property that the integral over all the space is equal to 1. Given this, the probability density function is defined by

$$f(x_i|\theta) = \sum_{j=1}^{K} \pi_j \mathcal{N}(x_i|\mu_j, \Lambda_j^{-1})$$

Because we want to do a hard clustering we can rewrite the last equation in the following form

$$f(x_i|\theta) = \prod_{j=1}^{K} \left[\pi_j \mathcal{N}(x|\mu_j, \Lambda_j^{-1})\right]^{c_{ij}},$$

where $c_{ij} = 1$ if pixel i is associated to cluster j, otherwise $c_{ij} = 0$.

Now the problem consists on estimating the values of the parameters of each Gaussian, in order to do this we used a technique called "Gibbs Sampling" that requires the conditional posterior distribution of each parameter given the values of the other parameters. Let's start with the finite case, for which the graphical model is shown in figure 3.7

The joint probability of this model is given by

$$P(X, C, \pi, \mu, \Lambda) = p(X|X, \mu, \Lambda)p(\mu|\Lambda)p(\Lambda)p(C|\pi)p(\pi)$$

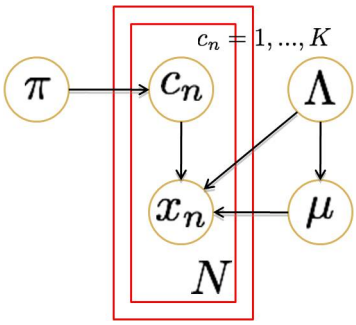


Figure 3.7: Graphical model for the finite case

According to our graphical model, the posterior distributions for the parameters are obtained with

$$
\begin{aligned}
q(\pi) &\sim p(C|\pi)p(\pi) \\
q(C) &\sim p(X|C,\mu,\Lambda)p(C|\pi) \\
q(\Lambda) &\sim p(X|C,\mu,\Lambda)p(\Lambda) \\
q(\mu) &\sim p(X|C,\mu,\Lambda)p(\mu|\Lambda)
\end{aligned}
$$

where $p(C|\pi), p(\pi),)p(\mu|\Lambda$ and $p(\Lambda)$ are the prior distributions. Is important to mention that all prior distributions are conjugate priors of the parameter we are looking for.

$$
\begin{aligned}
p(\pi) &\sim Dir(\alpha_0/K, ..., \alpha_0/K) \\
p(C|\pi) &= Mult(\pi_1, ..., \pi_K) \\
p(\mu_j|\Lambda_j) &\sim \mathcal{N}(\mu_j|m_0, (\beta_0\Lambda_j)^{-1}) \\
p(\Lambda_j) &\sim \mathcal{W}(\Lambda_j|W_0, v_0)
\end{aligned}
$$

The values $m_0, \lambda_0, \beta_0, W_0$ and v_0 are the hyper parameters of the model and were selected empirically. In each iteration new samples of the posterior distributions must be obtained, but before this, is important to define some useful statistics:

$$
\begin{aligned}
N_j &= \sum_{i=1}^{N} c_{ij} \\
\overline{x}_j &= \frac{1}{N_j} \sum_{i=1}^{N} c_{ij} x_i \\
S_j &= \frac{1}{N_j} \sum_{i=1}^{N} c_{ij} (x_i - \overline{x}_j)(x_i - \overline{x}_j)^T
\end{aligned}
$$

As we are using conjugate prior distributions we are sure that the posterior distributions will be of the same family of the prior distributions, but with different parameters. These new parameters are given by

$$
\begin{aligned}
\alpha_j &= \alpha_0/K + N_j \\
m_j &= \frac{\beta_0 m_0 + N_j \overline{x}_j}{\beta_0 + N_j} \\
\beta_j &= \beta_0 + N_j \\
W_j^{-1} &= W_0^{-1} + N_j S_j + \frac{\beta_0 N_j}{\beta_0 + N_j} (\overline{x}_j - m_0)(\overline{x}_j - m_0)^T \\
v_j &= v_0 + N_j
\end{aligned}
$$

For the infinite case, the prior distribution for the variable C is still a multinomial distribution, but with the difference that there is not need to know the variable π. Using the Chinese Restaurant Process, the prior distribution is expressed by

$$
p(c_{ij} = 1) = \begin{cases} \frac{N_{-i,j}}{N-1+\alpha_0} & \text{For clusters where } N_{-i,j} > 0 \\ \frac{\alpha_0}{N-1+\alpha_0} & \text{For the rest of the clusters} \end{cases}
$$

The probability of assigning certain pixel to an already existing cluster is proportional to the number of pixels associated to it. On the other hand, the probability of assigning the same pixel to a new cluster is proportional to the concentration parameter α_0. Then, we can say that pixels tends to join to the "most popular cluster".

Consider a data set of $N = 300$ observations using the following values for the hyper parameters: $\alpha_0 = 3.5, v_0 = D, \beta_0 = 1, W_0 = 0.01 \times D \times I, m_0 = \overline{x}$. The data set consists of several points in a 2-dimensional space drawn from three Gaussian functions as shown in figure 3.8.

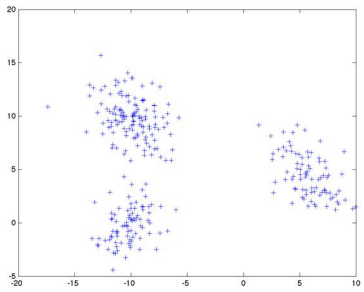


Figure 3.8: Test data set distribution

Figures 3.9 and 3.10 show how the number of clusters in the first iterations is greater than the real number of clusters, but as the number of iterations increases, the number of clusters decreases until it oscillate around the real value. At the end it can exist more clusters than the expected, but the mixing proportion of the those extra cluster will be almost zero. Also is important to notice that the creation of new clusters depends on the chosen value of the concentration parameter α_0.

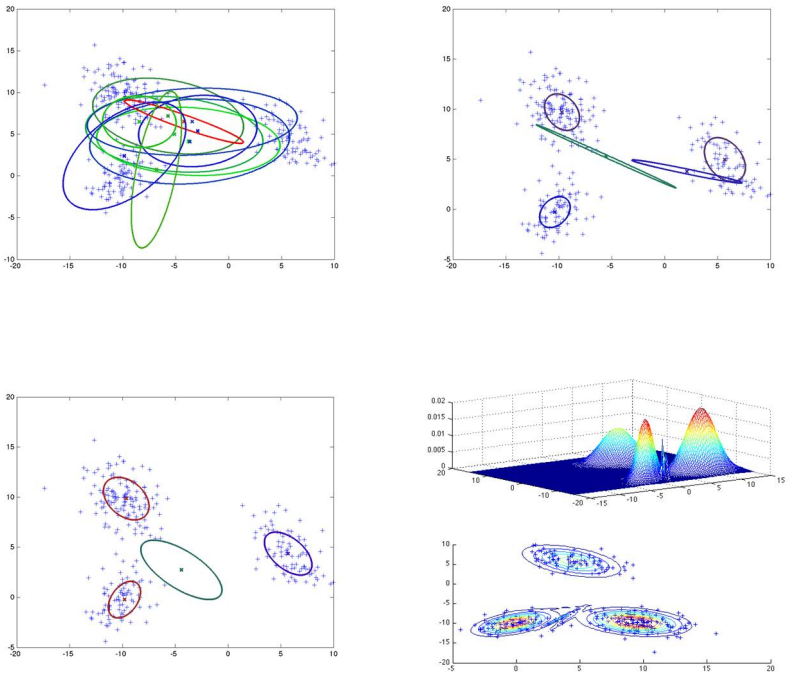


Figure 3.9: Infinite model after 1 iteration with 9 clusters. After 25 iterations with 5 clusters. After 30 iterations with 4 clusters. Final probability density function.

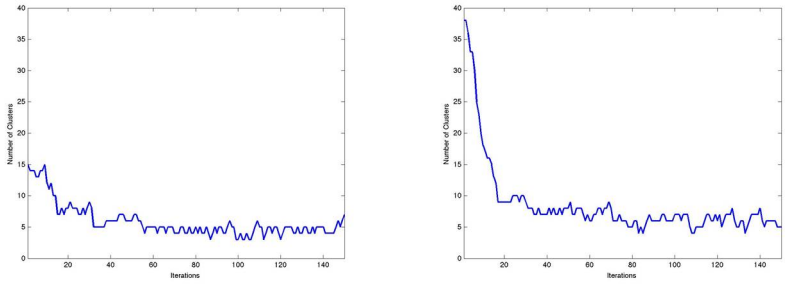


Figure 3.10: Plot iterations vs. number of clusters, where the number clusters decreases while iterations increases until the number of clusters oscillate around the expected value depending on the value of α_0

Figure 3.11 shows the result of applying the clustering algorithm in an image, where we can identify the different clusters founded, then we assigned to each pixel the value of the mean μ of the cluster they belong to. Then the resulting image can be used as a map to extract the pixels associated to a specific cluster from the original image as shown in figure 3.12.

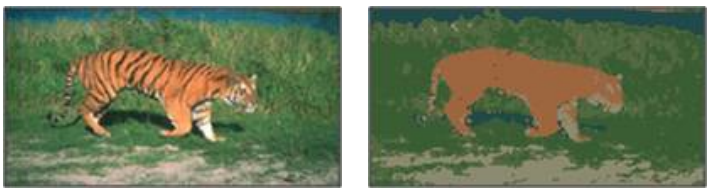


Figure 3.11: At the left the original image. At the right the resulting image after the clustering process.

Figure 3.12: Pixels associated to the same cluster extracted from the original image

Chapter 4

Split and Merge

Once we have identified the clusters in the image, the last part of the segmentation algorithm consists on two steps. The Split step and the Merge step. These two steps helped us to perform a better segmentation of the image. In this chapter we are going to explain in detail how these two steps work.

4.1 Split Step

The goal of this step is to isolate group of pixels that are connected and belong to the same cluster. For this, a Breadth First Search was implemented and for each pixel the algorithm compare the current pixel with each one of its 8 neighbors, and if a neighbor pixel and the current pixel are associated to the same cluster, then they are considered part of the same object.

Figure 4.1 shows how an image looks after the split step, where each object founded is represented by a color, for this image in particular 507 objects were founded.

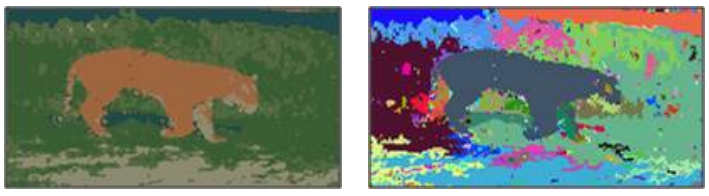


Figure 4.1: At left image resulting after the clustering process. At the right the image after the split step containing 507 objects.

4.2 Merge Step

The merge step has the task of remove small objects in such a way that those objects with a number of pixels bellow of a certain threshold value (TH) are eliminated. The pixels of the removed objects are then associated to the most similar (in color) neighbor not removed object.

The implementation of the merge step consists in a Breadth First Search along the image, if the current pixel is part of an object that has to be removed, then the search tries to find between the neighbors which is the object with the most similar color. Once located this neighbor object, the pixels of the current object take the same value of the neighbor object.

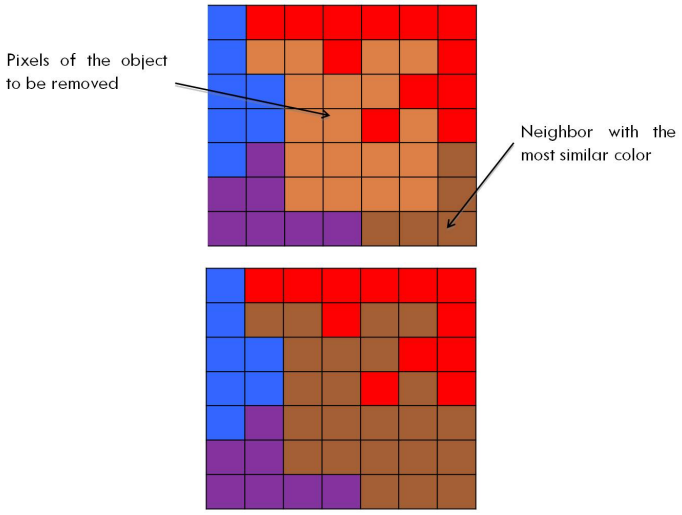


Figure 4.2: The image at top shows the object to be removed and the closest neighbor with more similar color. At bottom the resulting object after the merge process.

Figures 4.3 and 4.4 shows for two examples how if the threshold value (TH) increases, the number of objects decreases very fast, indicating that there are many objects with few pixels associated to them.

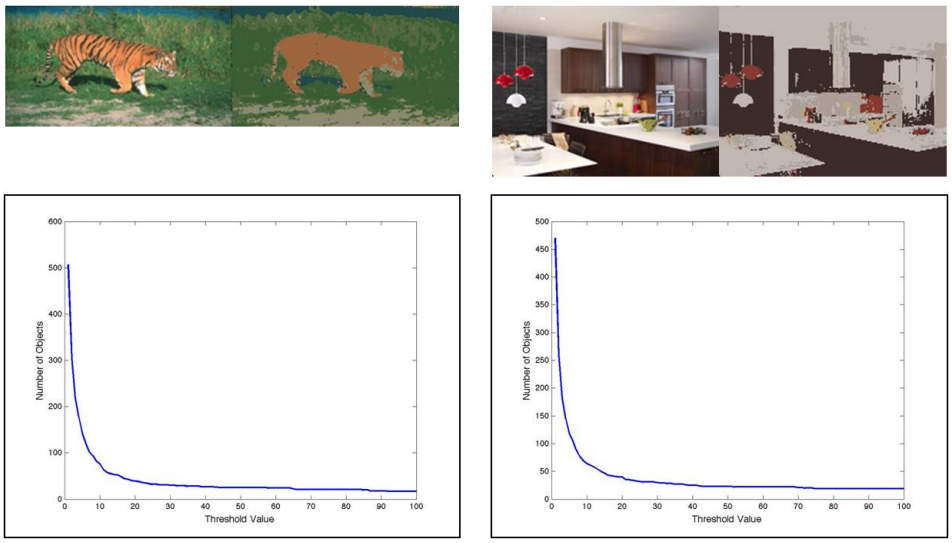


Figure 4.3: Plot "iterations vs number of objects", if the number of iterations increases, the number of objects decreases. Both images has many small objects that are removed for small values of TH.

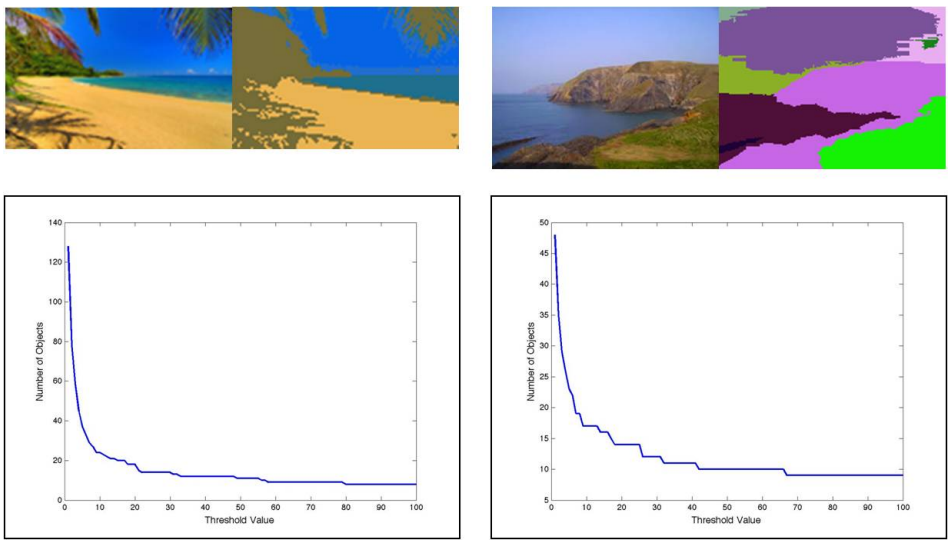


Figure 4.4: The image at the right has less objects, but many of them have a large number of pixels associated, then it is needed a greater value for TH to remove those objects.

Figures 4.5 and 4.6 show the result of applying the image segmentation method on different test images. We can see how the percentage of objects removed is considerable, that means that after the split process there exists a lot of small objects with few pixels associated to it. This works fine if we want to identify the most relevant objects in an image, but it will be difficult if we try to find small objects.

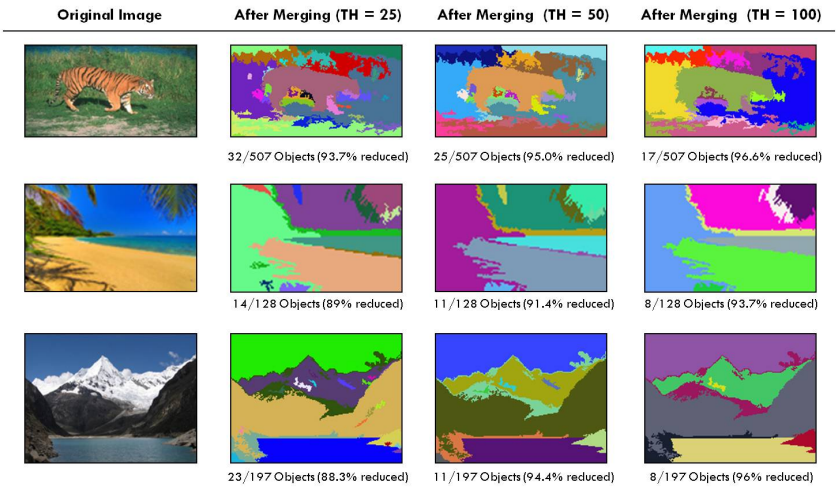


Figure 4.5: Different test images after the merge step for threshold values of $TH = 25$, $TH = 50$ and $TH = 100$

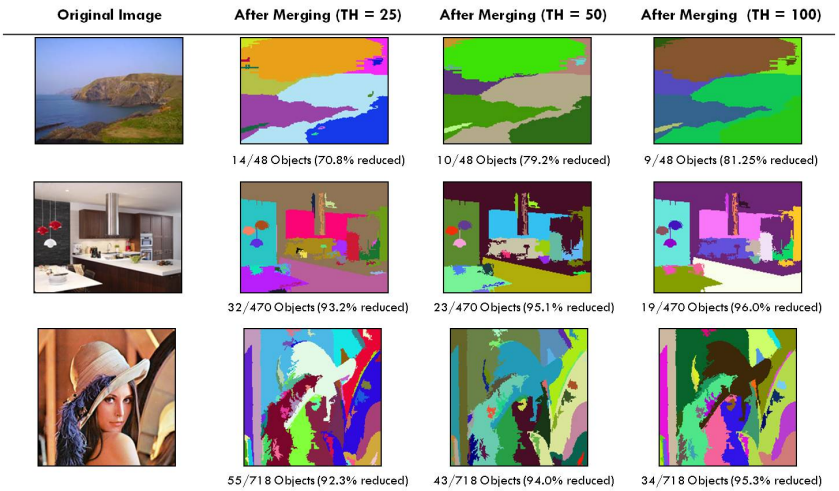


Figure 4.6: Different test images after the merge step for threshold values of $TH = 25$, $TH = 50$ and $TH = 100$

Chapter 5

Conclusion

In this research project were studied different algorithms about image processing, like filters, edge detection algorithms and image segmentation methods, but, the true propose of this research was to study some classic and also modern machine learning techniques. Machine learning is a branch of artificial intelligence that has been gaining a lot of attention in the last years. Because of the amount of available data in the web, more and more machine learning methods and applications have been developed and this tendency is expected to continue in the next years.

Machine Learning has been used in many companies such as Amazon, Facebook, Google, etc. It can be applied in many fields like computer vision, robotics, making decision systems, and more. In this project we presented an unsupervised learning algorithm to identify clusters in an image, using an infinite model.

Image segmentation can be used in different applications, including computer vision, object recognition, object tracking, etc. The reason of choosing image segmentation as the application of this research was to get involved in the image processing field and because the data sets can be obtained easily.

The results in chapter 4 show that the proposed method works fine for identifying large group of pixels connected of the same color, but is difficult to identify small objects inside an image.

The Split and Merge steps helped to do a better segmentation of the images, but these methods can be improved and take in consideration other properties besides color, such as texture, position, etc.

Also another thing to be improved is the complexity time of the clustering al-

gorithm, calculate the statistics needed for obtain the parameters of the posterior distributions consumes a lot of time, and make it difficult to work with bigger images.