REPUBLIC OF TURKEY

ALTINBAS UNIVERSITY

Institute of Graduate Studies

Electrical and Computer Engineering

IMAGE SEGMENTATION USING ASSOCIATED RULE DATAMINING TECHNIQUE

Husham Salman Moasi AL-ABBOODI

Master's Thesis

Supervisor

Asst. Prof. Dr. Abdullahi Abdu IBRAHIM

Istanbul, 2021

IMAGE SEGMENTATION USING ASSOCIATED RULE DATAMINING TECHNIQUE

Husham Salman Moasi AL-ABBOODI

Electrical and Computer Engineering

Master's Thesis

ALTINBAŞ UNIVERSITY

2021

The thesis titled IMAGE SEGMENTATION USING ASSOCIATED RULE DATA MINING TECHNIQUE prepared by HUSHAM SALMAN MOASI AL-ABBOODI and submitted on 25/11/2021 has been **accepted unanimously** for the degree of Master's Science in Electrical and Computer Engineering.

Asst. Prof. Dr. Abdullahi Abdu IBRAHIM

Supervisor

Thesis Defense Jury Members:

Asst. Prof. Dr. Abdullahi Abdu IBRAHIM	Faculty of Engineering and Architecture, Altinbas University	_____
Asst. Prof. Dr. Sefer KURNAZ	Faculty of Engineering and Architecture, Altinbas University	_____
Assoc. Prof. Dr. A. Mohammed ABUBAKAR	Faculty of Business Antalya Science University	_____

I hereby declare that this thesis meets all format and submission requirements of a Master of Science

Submission date of the thesis to Institute of Graduate Studies: ___/___/___

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Husham Salman Moasi AL-ABBOODI

DEDICATION

To my beloved wife, Nadia, who encouraged me to urge myself to the limits. I am wholeheartedly grateful for everything she has done to me; her positive energy kept me focus on my goal whenever I felt desperate. Without her love, encouragement, and support I would not be walking on my path to accomplish my dream, to gain master's degree. I owe much to my beloved father and mother; I thank them for their love. I deeply express my gratitude for their unbending support.

ACKNOWLEDGEMENT

First and foremost, I would like to thank my supervisor Asst. Prof. Dr. Abdullahi Abdu IBRAHIM for guiding and helping me along the way in writing this dissertation. Discussing my progress, problems, and ideas with my supervisor Asst. Prof. Dr. Abdullahi Abdu IBRAHIM a couple of times every week helped me tremendously in understanding the logic behind the research. It made me better realize the technical need for this research work.

ABSTRACT

IMAGE SEGMENTATION USING ASSOCIATED RULE DATAMINING TECHNIQUE

AL-ABBOODI, Husham Salman Moasi,

M.S., Electrical and Computer Engineering, Altınbaş University,

Supervisor: Asst. Prof. Dr. Abdullahi Abdu IBRAHIM

Date: 11/2021

Pages: 56

Smart grids are electric grids that are composed of multiple power sources and devices connected to each other to provide better reliability in power generation and power management, modern developments of the smart grid aim at either improving the control of power sources and loads connected to the smart grid by developing a specialized software/hardware, or by improving the communication within the parts of the smart grid and the central control. In this paper we aim at improving both sides of the smart grid system (communication and control), we propose a fuzzy logic-based controller for renewable energy and fossil fuel sources in a grid and an internet of things-based monitoring system which oversees the state of the smart grid, faults that occur in the grid, and how the fuzzy controller overcomes those faults, all in which provide an extra layer of support to the smart grid.

Keywords: Photovoltaic, Solar Energy, Fuzzy Logic, Smart Grid, Internet of Thinks IoT.

TABLE OF CONTENTS

Pages

LIST OF TABLES

Pages

LIST OF FIGURES

Pages

LIST OF ABBREVIATIONS

MSE	:	Mean Square Error
NN	:	Neural Network
PSTNR	:	Peak Signal-To-Noise Ratio
QoS	:	Quality Of Service
TCP	:	Transmission Control Protocol

1. INTRODUCTION

1.1 BACKGROUND

Image analysis allows us to extract information from them, and within this discipline, segmentation allows the identification of their constituent parts. Image segmentation has applications in pattern recognition and traffic control systems, among others. If we take image segmentation to the field of medical images, the applications range from the detection of tumors and other pathologies to the measurement of volumes in tissues [2]:

Figure 1.1: Smart Grid Concept.

1.2 MOTIVATION

The advancement of computer technology has considerably transformed the characteristics of information processing, in different aspects. In the area of medicine, the alternatives for data capture and analysis have evolved notably until now they allow the development of three-dimensional image treatment applications, generated from the information coming from capture devices, such as tomography equipment. computed or magnetic resonance imaging. This type of tool can offer assistance to professionals and improve the characteristics of the diagnosis, by facilitating the analysis of information directly on the three-dimensional data, in a process of non-

invasive characteristics. In addition, due to the evolution produced in the performance of low-cost computers, it has become possible to experiment with solutions that were previously restricted to large equipment, due to the computational demand of the algorithms involved.

1.3 PROBLEM STATEMENT

One aspect of interest consists of the ability to detect certain structures of interest within the image with similar characteristics, through a process known as segmentation. This constitutes an essential stage that conditions the results of image processing in various types of advanced applications. For example, in the case of medicine, it is possible to improve different types of practices, such as planning surgeries and radiotherapy treatments, which can be performed more effectively and with less risk. [1], [2]:

Figure 1.2: Communication Based Smart Grid.

1.4 CONTRIBUTION

There are various image segmentation techniques, and in this work a medical image segmentation procedure based on the metaheuristics of Ant Colony Algorithms is proposed. The algorithms of this metaheuristic mimic the behavior of ants during their search for food, since they always

produce optimal routes between the food source and the nest. This behavior was implemented using artificial ants in order to perform specific image processing tasks.

This procedure was applied to Brain Magnetic Resonance images - seeking the extraction of the segments corresponding to Gray Matter, White Matter and Cerebrospinal Fluid- and the segmentation obtained was of a higher quality than that of the currently existing algorithms for this task..

1.5 THESIS ORGANIZATION

The thesis is structured as follows: Section 2 is where we review some of the previous work and implementation on smart grids. Section 3 is where we give a background about all the components. Section 4 is where we explain our model in details and implementation of that model in details, Section 5 is where we simulate our model and record the results obtained. Section 6 is where we conclude our work and put a future scope into perspective.

1.6 STATE OF THE ART

Typically, segmentation requires time and effort for data preparation by specialized operators and this imposes limitations on medical practice. Even today, manual segmentation processes are still often used through the careful selection of points on the image slices, which constitutes a slow, subjective and little repeatable work [3]. Different computational methods capable of carrying out the segmentation of an image have been proposed; however, there are still no definitive solutions or generally applicable algorithms. Within the different possibilities, thresholding [4] is one of the simplest and most rudimentary algorithms, initially proposed for flat images, capable of isolating objects from the background based on the intensity values of the elements of the image. However, due to its characteristics, this technique is generally not adapted to images in which there are anatomically different areas with similar intensities, which frequently occurs in medical images. Based on the concept of thresholding, the marching cubes or marching tetrahedra algorithms [5] constitute a segmentation option generally used for image rendering, which allow the generation of iso-surfaces from a specified threshold value. The basic idea is to determine if the elements of the volume are crossed by the considered level surface, from the evaluation of the value of the vertices of each voxel with respect to the threshold. These schemes often lead to several ambiguous situations that can generate voids in the resulting surface and also have the inherent disadvantages

of threshold-based methods. On the other hand, approaches based on edge detection allow the delimitation of boundaries between components, characterized by points of high gradient value [6]. These types of methods are appropriate for images with well-defined features, but they can produce unsatisfactory results due to their sensitivity to noise or lack of local adaptability. This is decisive in the case of images from tomographic reconstruction, in which the different tissues do not necessarily have a uniform intensity and the edges are not always clearly defined from the intensities of the voxels. Recently, another segmentation option has become popular. based on active contours, known as snakes, which are based on the evolution of an elastic curve located on the object to segment [7]. This snake is modified from its initial shape and location as a result of the action of internal and external forces that, on the one hand, shape its elasticity and, on the other, tend to bring it closer to the image, respectively. Although satisfactory results from its application have been reported, it is a computationally intensive scheme that poses some drawbacks, as described in [8], mainly when the objects to be segmented have very irregular shapes. Segmentation approaches based on the growth of regions seek to delimit homogeneous areas of the image, through the progressive inclusion of elements that meet certain criteria of similarity [9], [10]. In this work, a segmentation method based on a region growth algorithm is proposed, applied directly to the three-dimensional digital image formed from the integration of the successive cross sections of the original object. In addition, criteria have been established that allow controlling such incorporation, in order to prevent leakage of the filling towards adjacent regions. Due to its characteristics, this segmentation scheme takes into account the local characteristics of the elements of the image and is quite stable with respect to the problems in which the mentioned techniques usually fail..

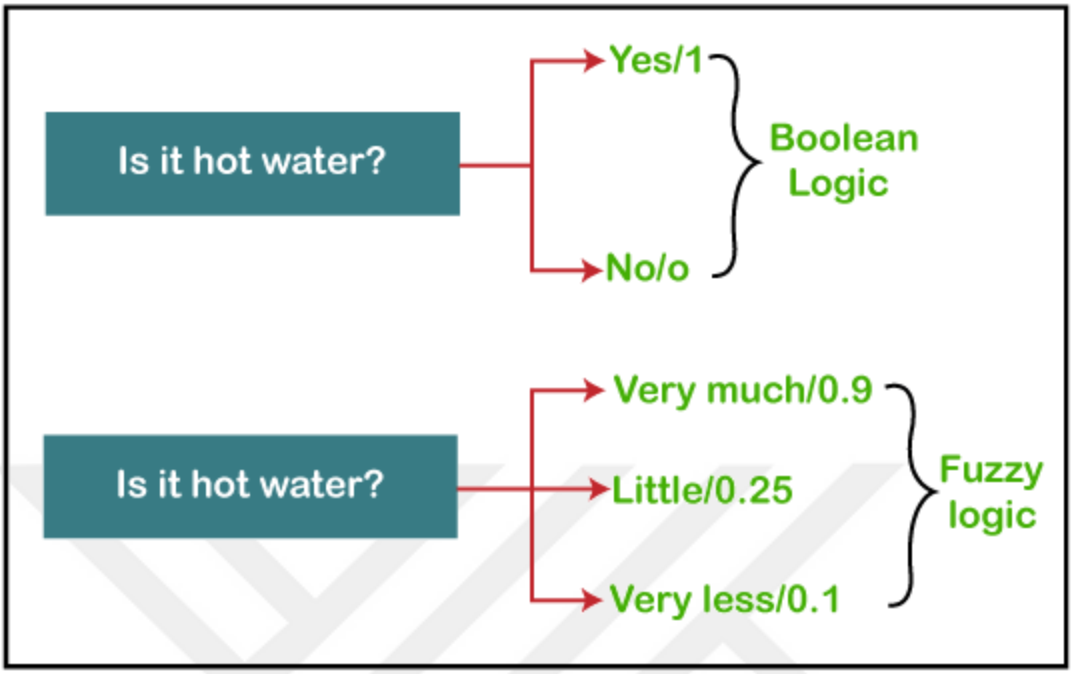


Figure 1.3: Alternatives Consider The Neighborhood Of Neurons As Fuzzy Sets To Offer More Flexibility To The Neural Model.

Other alternatives consider the neighborhood of neurons as fuzzy sets to offer more flexibility to the neural model (Luque Baena et al., 2008), or adapt the neighborhood of each neuron to the advancement of learning (Palomo and López-Rubio, 2016). In addition, other types of divergence measures such as the Euclidean distance can be considered (López-Rubio et al., 2014). Self-organizing maps are also considered as an alternative solution for modeling the background and several works based on this type of unsupervised learning can be found in the literature. In particular, several of them model each pixel using a self-organizing map (Maddalena and Petrosino, 2008; López-Rubio et al., 2011a; Zhao et al., 2015; Singh et al., 2010), thus maintaining a topology between neurons he considers that each prototype can learn according to the distance regarding the neuron that wins. Setting up the topology ahead of time is required; in most cases, a mesh topology is used. This model limits the data distribution to a topology that is dissimilar to the one specified previously. Competitive learning methods have neurons that do not have a relationship with each other. That non-relationship provides greater flexibility to the model. Although the Luque et al. (2008b) and Subudhi et al. (2015) proposed neural networks models help to generate the basic mask that contains the foreground objects, they do not offer any improvements for the neural background model. the use of convolutional neural networks is applied to object

detection in video sequences, specifically for background subtraction (Zhang et al., 2015; Chen et al., 2016b) (Braham and Droogenbroeck, 2016). Many of these proposals require a training phase to build the model, whereas this proposal does not. This proposal doesn't need training data, nor are there any restrictions to what kind of objects can be detected, because it is using an unsupervised methodology. Labeling multiple objects can only be done for individual experiments, but not for motion detection in thousands of IP surveillance cameras across cities, buildings, and roads. Another technique for automatic data labeling (that doesn't improve on traditional techniques) has also been proposed (Bianco et al., 2017). (Braham and Droogenbroeck, 2016). The following discussion should help you to determine which of the deep learning algorithms discussed in this chapter are appropriate for the behavior analysis stage of video surveillance systems, which are outside the scope of this chapter.

2. MATERIALS AND METHODS

In this chapter we review some of the essential components that we used to implement our model; we give a brief introduction to the component.

2.1 SEGMENTATION

The discipline of image analysis consists of the extraction of information from graphic representations. Human beings - through the sense of vision - perform image analysis in everyday tasks such as identifying objects from a distance or reading a text, analysis actions that we could qualify as qualitative. In addition to the qualitative tasks that human vision performs efficiently, there are quantitative image analysis tasks. For example, in [1] it is mentioned that there are diagnostic techniques for atrophic rhinitis in pigs by calculating areas in cross-sectional images of the nose of this animal. It is in these tasks that human vision does not perform well - proof of this is the optical illusions that make it look like large objects that are not - so the need arises to use computers to obtain this information. Returning to the example of atrophic rhinitis, before performing the area calculation it is necessary to refine the images obtained to remove imperfections and make the area calculation more accurate. This pre-processing work, if done manually, requires considerable time and effort, which on the contrary can be done quickly and efficiently by a computer and a good algorithm. This is another aspect where the use of computers is beneficial for image analysis. The image analysis process involves subjecting the image to a series of consecutive and logical stages to obtain the desired information, one of these stages being segmentation. The segmentation process consists of dividing the image into regions, where each region corresponds -for example- with different objects or with different parts of the same object. This stage is critical because it represents a semantic leap from computational units -such as pixels- towards elements closer to reality such as objects and their constituent parts. Successful segmentation is necessary to make the rest of the image analysis stages simpler. Segmentation is especially important in the field of medical imaging, where it is common to use computers and automatic segmentation techniques to delineate anatomical structures, since performing these procedures manually is tedious and requires considerable time [9]. These techniques have to deal with the fact that image segmentation is a computationally expensive procedure, so response times and their optimization must be taken into consideration when designing a procedure of this nature

[10]. Given its importance, several procedures have been defined to face this task, largely because the characteristics and nature of the image to be segmented influence the selection of the algorithm. Segmentation techniques can be grouped into threshold-based techniques, edge detection-based techniques, region-based techniques, and dividing transformation techniques. Likewise, the performance of these techniques can be optimized through the use of advanced tools such as evolutionary algorithms, neural networks and fuzzy logic [2]. However, there is still no generally accepted procedure that produces quality segmentations for multiple application domains [8]. In this work, the use of a non-traditional method for the segmentation of medical images - emphasizing brain magnetic resonance - based on artificial ant colonies is proposed. Natural ant colonies show a complex behavior in contrast to the simple behavior that each ant shows as an individual, and this is especially visible in the food search procedure: The ants, when roaming the terrain in search of food, leave pheromone traces , which in turn serve as a guide to the rest of the members of the colony in their search for food. Both Marco Dorigo [5] and Chialvo and Millonas [6] defined new computational paradigms establishing an analogy with this behavior.

Images require segmentation within their process, either in early stages -a pre-processing level- or in later stages [2]. Raut et al. [2] makes an extensive list of applications that fall into this category, among which we can mention: object recognition, pattern recognition, traffic control systems and location of objects in satellite images. Image segmentation has become an important tool for disease diagnosis [10]. There are multiple modalities of medical imaging - MRIs, CT scans and mammograms to name a few - and computers are often used for their analysis when you have a large number of images or they are large. Segmentation algorithms are currently applied to measure tissue volumes, to detect tumors and pathologies, and to study anatomical structures [2]. The ACO algorithms have been applied successfully to a great diversity of problems, showing results similar to the best algorithms designed to face these problems or in certain cases being the best approximation for it [25]. There are currently ACO algorithms for sequential ordering problems, classification problems, vehicle routing problems, and even image analysis applications like those shown in Chapter 3, which led us to select metaheuristics as the approach to take..

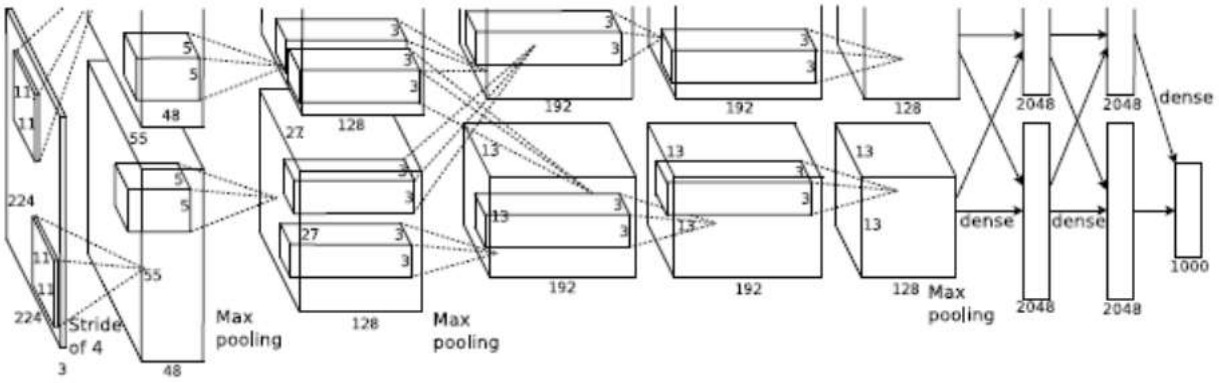


Figure 2.1: Flowchart For Work Algorithm.

2.1.1 Threshold-based techniques

This set of techniques establishes thresholds so that the image to be segmented can be divided into groups of pixels, where a group of pixels has associated values lower than the threshold while another group has higher values. In [8], these techniques are in turn classified into two groups.

2.1.2 Segmentation

Object extraction is an important preliminary task in video surveillance and human movement analysis. For public safety, video surveillance is an important technology, such as in transit networks, city centers, schools, and hospitals. Video surveillance can also be used to keep track of the patterns of human behavior, like traffic lights and road crossings (Maddalena and Petrosino, 2008). As an example, scenes are commonly captured using a static camera, which compresses the video information to create a single frame. The task in this instance is to identify objects that are arriving, as their arrival may present an opportunity to derive a large number of characteristics that could be used to carry out classification and recognition operations later in the process. To be sure, many people assume that a scene that lacks important objects such as people or objects can be modeled with a statistical background model. Background subtraction is a typical approach to distinguishing interesting objects from the background. This process uses a non-reference background model to remove the current image. When the observed color value of a pixel is sufficiently captured by the model, the pixel is defined as a background. For example, in some circumstances, the pixel is classified as the foreground. Nevertheless, backgrounds with difficult circumstances, such as lighting changes, moving trees, water, video displays, rotating fans,

shadows, camouflage (foreground objects are similar to the background), and changes occasional from the true background (for example removing an object from the scene), all prove problematic in applications such as transportation, logistics, surveillance, telecommunications, the media, retail, manufacturing, law enforcement, defense, real estate, among others. We will not be able to fix these problems with static background models only. Dynamic background problems can be solved by employing a wide range of solutions. Many of them have a heavy computational cost, which keeps them from being used in real-time applications. Even when a visual analysis is used to evaluate the detected events, there may be an issue with the prior process's ability to handle the resulting data. To maintain the ritual of the incoming video data stream, many systems use real-time processing. Neural networks have been designed to model the continuous background, and two examples of these solutions are provided in the papers of Maddalena and Petrosino (2008) and López-Rubio et al. (2011a). We describe a self-organizing map model as a special case of neural networks in this chapter. The Network for Emotional Literacy™ was specifically designed to handle variable input distributions, with a two-phased learning strategy that gets the most active neuron to perform a more intense action. It's due to how neurons adapt to changes in the input distribution, not the number of remaining neurons, which learn at a much slower pace. The algorithm's design aims to optimize a cost function that considers these two goals: being cost-effective and cost-minimizing. thus, the proposal's original contribution is twofold: A neural network model designed to handle varying input distributions is proposed. The proposed learning model is used to solve one of the main problems in current video surveillance systems: background modeling. Previous background models of self-organizing maps differ substantially from the proposal. (López-Rubio et al., 2011a).

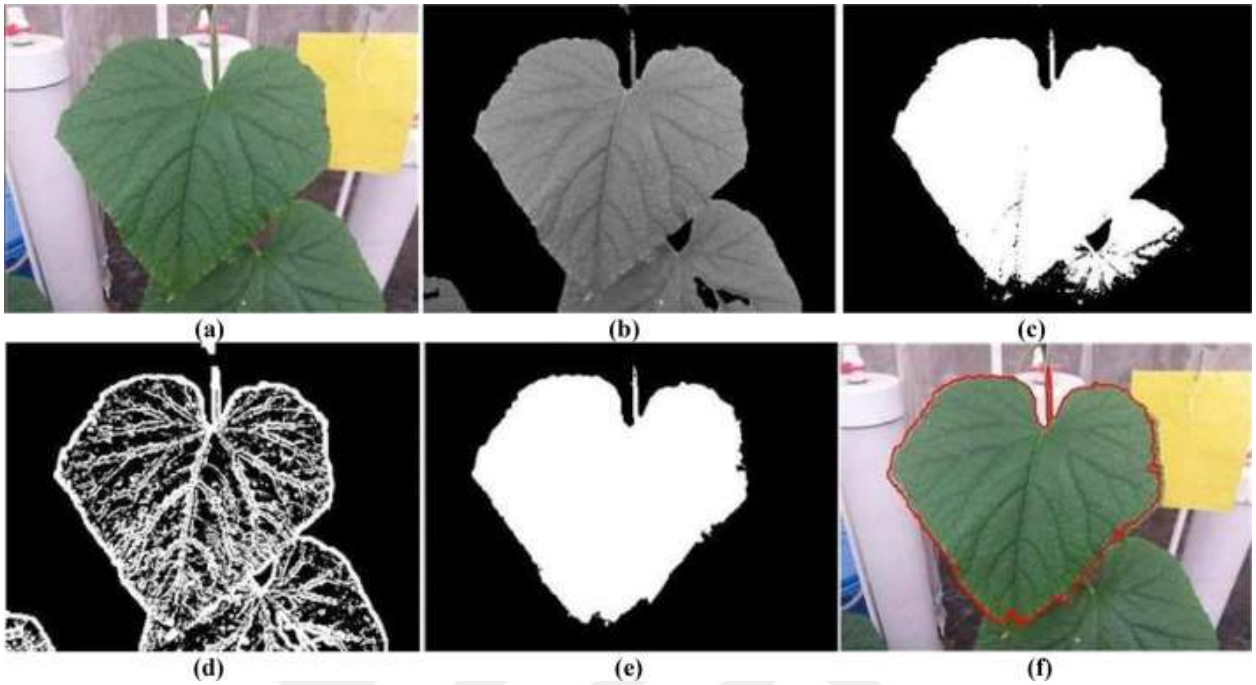


Figure 2.2: A- Original Image, B-Covert Image, C- Segmentation Image, D- More Classification, F-Edge.

Also, a standard self-organizing map topology does not solve the problem of neurons ceasing to represent input, also known as dead neurons. The concept is that instead of moving all neurons in the network toward the middle of the input distribution, a new term is added to the cost function. Thus, the input distribution that changes over time dictates how all of the neurons within the neural network are structured. The other ART-type background models differ from this model in two ways. Also, there is no need for a surveillance parameter in the new proposal as the nodes themselves will do the surveillance instead. This new approach, in contrast, requires that the background of the scene be modeled with all the neurons used. Thus, there is no longer a need for a procedure to make the decision of which neurons are used to model the background.

Self-organizing maps is just a special case of a neural network. For applications that process non-stationary input distributions, it is useful. It's difficult to successfully implement a competitive neural network in this kind of situation because, even if a neuron sets its prototype in the middle of a group, the group could disappear over time. It is therefore possible that the neuron loses the ability to represent any current input sample, causing its death, which is called neuronal death. If this occurs, the neuron's prototype will not win the competition, and it will not learn. To circumvent this issue, it is proposed that all neuron prototypes gradually move in the general input

distribution's direction. By implementing this concept, all the prototypes stay in the entrance space, allowing them to better represent a group. Instead of adding the squared Euclidean distance penalty to the cost function of the kmeans algorithm, we use a term which causes a small increase in the kmeans standard's cost function. When a neuron doesn't win a race for a long time, it gradually increases in prominence on the input side, where it has a better chance of getting samples to help it win. Dimension D denotes the number of samples in the input space, or x is an element of the real plane. This means that in this particular case, D = 3. In traditional competitive learning, N neurons, each with a vector of N elements, are used. The vectors of these neurons store prototypes; in this case, the set {1, ..., N}. the following is considered to be the best cost function: [11].

$$\mathcal{E} = \frac{1}{2}\sum_{j=1}^{M}\left\|\mathbf{x}_j - \mathbf{w}_{Winner(\mathbf{x}_j)}\right\|^2 + \frac{\lambda}{2}\sum_{j=1}^{M}\sum_{i=1}^{N}\left\|\mathbf{x}_j - \mathbf{w}_i\right\|^2 \tag{2.1}$$

The first term in equation is Mean Square Error (Mean Squared Error Ueda and Nakano 1994). (3.1). When this occurs, it alerts all possible escape routes. In equation (3.1), the second term serves the role of shifting all of the prototypes to a nearby minimum of the Mean Square Error, which is located near the centers of the input distributions. It makes finding alternative group configurations easy because of the two cost terms that use two different objectives. A variety of competitive learning neural networks, such as those developed by Menéndez et al. (2014), can be used for partitioning groups. That is, each sample will only belong to one group. Grouping methods allow the division or grouping of groups, whereas the number of neurons is fixed to ensure real-time functioning (Chira et al., 2014). Neuron I is associated with the group of input samples as follows:

$$C_i = \left\{\mathbf{x}_j \in \mathbb{R}^D \mid i = Winner\left(\mathbf{x}_j\right)\right\} \tag{2.2}$$

2.2 BACKGROUND MODELLING

In the previous section, you learned about a neural network model that can be used to solve the foreground detection problem. Separating the background and foreground objects that appear in a video sequence is necessary to tackle this problem. The background pattern is acquired at each pixel to serve this purpose. The color background that appears at a pixel is supposed to represent the properties of the pixel's background color. So as long as a new frame of the video is taken, they should be updated to reflect that. There is no supervisor to provide information on which pixels correspond to the background in a given frame, and so unsupervised learning can introduce inaccuracies. Background colors can also change due to the addition of new shadows, as well as because of light sources and various events such as the movement of trees, giving the background a more dynamic look. A competitive neural network is linked to each pixel to learn its color information. By assessing how well the background model represents the current observed pixel's color, it is possible to determine whether the pixel belongs to the foreground. This is calculated with the error quantization qn, r ∈ R, using the current observed color xn, r ∈ RD, as input and with a competing neural network, represented by the competing neural network for the pixel: [16].

$$q_{n,\mathbf{r}} = \min_{i \in \{1,\ldots,N\}} \left\| \mathbf{x}_{n,\mathbf{r}} - \mathbf{w}_{i,n,\mathbf{r}} \right\| \tag{2.3}$$

For the prototype to appear at time step n of the ith neuron of the competitive neural network associated with the pixel with r coordinates, Wi, N, and R are all set in the R D zone. A filtered image with elements q˜n, r is obtained by processing the image formed by the quantization errors qn, r with a filter of the median size of 5 × 5 pixels. Reducing noise in the quantization image is the purpose of this operation. The pixel is claimed to be located in the foreground object, so it must satisfy these conditions: First, q>n; second, r must be greater than a system threshold T, where T must be greater than 0. This is done to allow objects in the foreground to have a distinct color from those learned by prototypes. Training samples from the first 100 frames of the video are used to initialize the neural network that corresponds to each pixel. When using the technique for foreground detection, the learning process should be disabled for the pixels that are marked as foreground. The patterns of the pixels that do not belong to the background are protected from

being affected by learning of the samples that do. To disable learning for the foreground pixels, a Boolean parameter L of the algorithm is defined that is set to true. For those experiments, thus, the foreground detection is replaced Single-pixel prototypes can remain in a specific color scheme for a previous foreground object even if that object is removed. When the video sequence begins with some foreground objects and all of the pixels' prototypes are initialized to foreground colors, this phenomenon can occur. To avoid this problem, a parameter of the algorithm, known as the Z-frame time limit, is imposed. All of the pixel's neurons are reset to the color of the currently observed pixel if the pixel has been constantly declared as being in the foreground for a total of Z frames. To ensure that the background model can recover from the problems mentioned above, this procedure is followed. In all the experiments, the parameter has been set to be equal to 1000. This has given favorable results. The higher value of this Z parameter may produce better performance in situations where the foreground objects are not moving. L has no impact on this procedure; that is, L has no bearing on this procedure. That is, regardless of the value of L, the pixel reset is executed. While computational complexity is only in the O (NDNumRowsNumCols) range, background modeling needs N times D-dimensional pixel values, which are obtained by summing together all of the D-dimensional pixel values found in each video frame. In other words, it's like every time the size of the variables changes, the problem changes linearly..

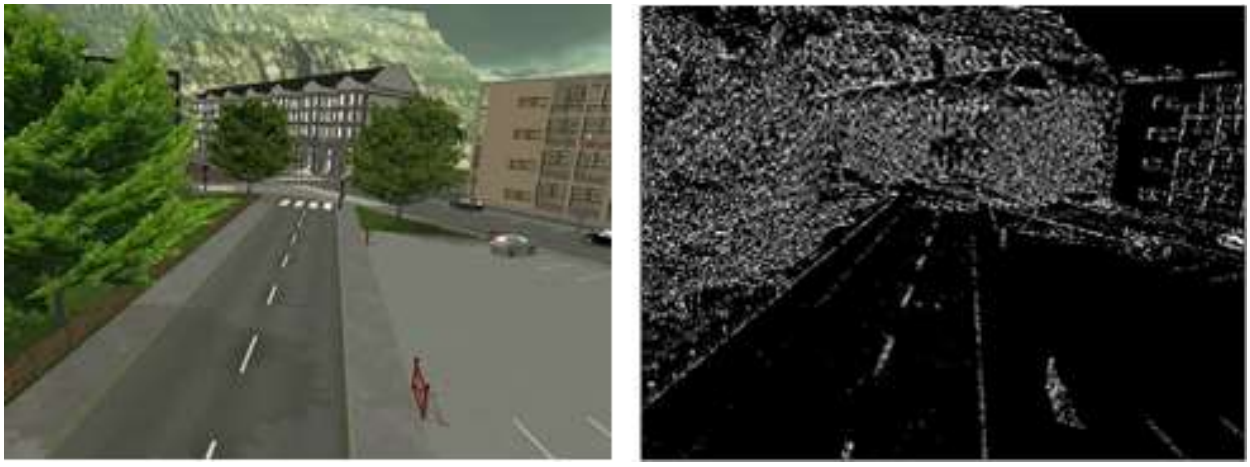


Figure 2.3: D-Dimensional Pixel Values Found In Each Video Frame.

2.2.1 Foreground Detection

Computer vision system design is hindered by the detection of objects in the foreground. The algorithms used to solve this problem must be able to handle all of the difficult situations that exist

in videos from the real world. Some drawbacks include lighting changes, which cause objects in the background to cast shadows on the foreground. Trees, waves, and other objects also move in the background, adding to the list of drawbacks. Object extraction is an important preliminary task in video surveillance and human movement analysis. For effective management in public transportation and services, use of video surveillance is essential (Luque-Baena et al., 2015b; Kamijo et al., 2000; Cheng and Hsu, 2011). For safety in public transport and public services, the need for video surveillance is critical (Geronimo et al., 2010; Räty, 2010). For this type of application, the background of a scene without objects of interest is assumed to show the common activities of people. Background subtraction processes involve subtracting the current image from the reference background model to detect foreground objects. In real-world use, however, modeling the background poses a challenge due to the previously mentioned shortcomings. Simple and static background models cannot solve these problems. Furthermore, for example, a Gaussian distribution (Wren et al., 1997) or a self-organizing neural network may be used to model the background of a general scene (Maddalena and Petrosino, 2008). While there are several approaches to modeling video sequences in the literature, the techniques include Gaussian mixtures and probabilistic neural networks, as seen in Subsection 3.2 in Chapter 3. To obtain an accurate and consistent segmentation mask, foreground detection methods can be problematic and typically require post-processing that revises and improves the initial mask. Mostly based on the neighborhoods and similarities of adjacent pixels, we want to filter out spurious values that appear in the foreground mask. The most popular method involves applying morphological operators, though there are other techniques in which temporal filters are used (Luque et al., 2008b). Their use in system composition is well known, most recently as a technique that complements other standard ones (like a mixture of Gaussians), and then the end result improves from the mix (Zhao et al., 2012; Baf et al., 2008). The model described in this chapter implements a probabilistic self-organizing map incorporating pixel characteristics that offer a diverse range of features. A global scene lighting change is included in the foreground output mask in order to enhance the appearance of the self-organizing map (SOM).. [19].

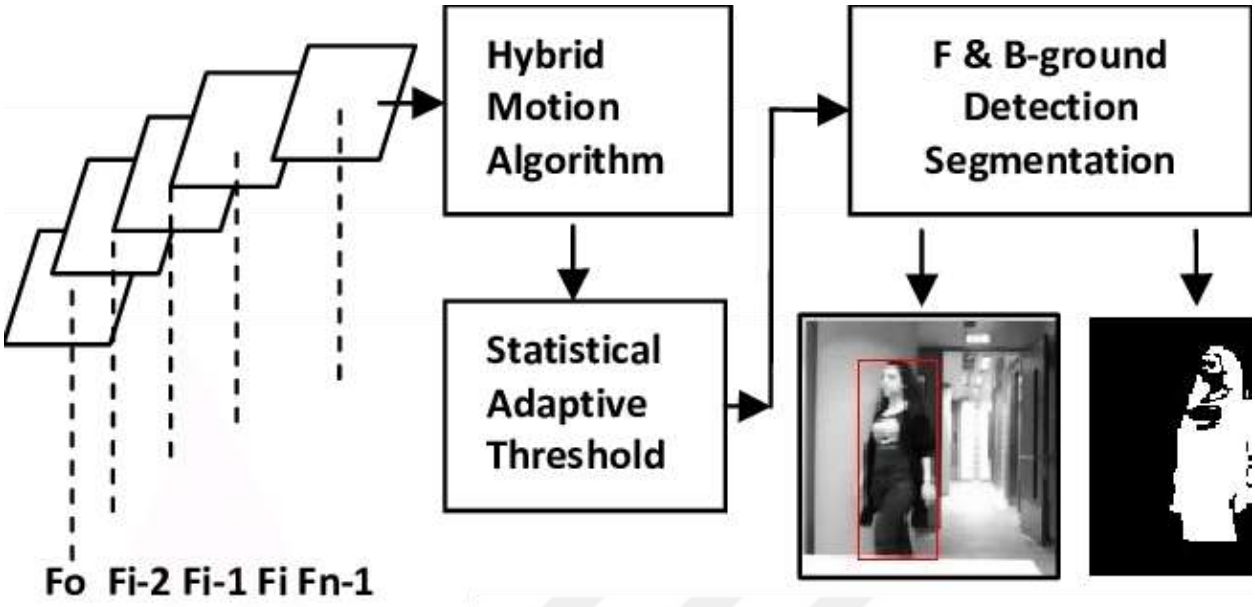


Figure 2.4: The Model Described in This Chapter Implements A Probabilistic Self-Organizing Map Incorporating Pixel Characteristics That Offer A Diverse Range Of Features.

2.3 DETECTION OF MOVING OBJECTS

Motion detection is the practice of determining when something has changed location or location has changed. Using mechanical or electronic methods, motion detection can be achieved, but the use of electronic sensors is standard. Inactive motion sensors are also called passive sensors. Passive sensors do not emit any energy, and are the most widely used type. When compared to background objects at room temperature, humans emit radiation in the mid-infrared wavelength range and thus are sensitive to the temperature of the skin. While active sensors emit a signal, such as light, microwave, or sound, into the environment and detect any change in behavior, passive sensors do not make any sounds or send signals into the environment. New detection techniques currently using digital cameras capable of recording video are being employed. When using motion detection software, the output of a camera can be used to determine if there is motion. Most supervisory software algorithms use motion detection. Algorithms can activate cameras and begin video recording when they detect movement. Another name for this process is activity detection. García et al. (2014) and Gómez et al. (2015) describe a more advanced motion detection video surveillance system that can analyze the type of movement and thus decide whether or not to raise an alarm. A form of artificial neural network that is capable of learning without supervision (Kohonen, 1982). This can be attributed to SOMs having being utilized in the identification of

knowledge, data mining, visualization of large data sets, and two-dimensional data mapping since its inception (Yin, 2008; Kohonen, 2013). It is keeping track of how the input data is related to one another while also preserving the overall structure. A wide range of applications have long been found for this neural network (Samuel Kaski and Kohonen, 1998; Oja et al., 2003). especially, in the computer vision realm, it's been applied to tasks like color quantification (Dekker, 1994; Papamarkos, 1999; Xiao et al., 2012; Palomo and Domínguez, 2014), and image segmentation (Bhandarkar et al., 1997; Dong and Xie, 2005; Maddalena and Petrosino, 2008; Lacerda and Mello, 2013). While batch learning has better performance for escaping local minima, incremental learning provides better performance for color quantification (Chang et al., 2005). As shown in subsection 3.2, the method has previously been applied to detect foreground objects in video sequences. However, these proposals require a SOM for each pixel of the frame of the video. This is why microcontroller chips are unable to perform the task at hand; they do not have computational resources available. Every one of these plans requires a significant amount of computation, which makes it difficult for computer vision systems to perform (Casanova et al., 2013). The resulting systems are extremely expensive and complex to produce on a large scale. To obtain a lower-cost and simpler motion sensors, the proposed strategy in this chapter is to change the strategy. An inexpensive, small, and flexible hardware device is called a microcontroller board. This is a characteristic feature of many crucial technologies, including those that exist within embedded systems (Marwedel, 2006), real-time systems (Kopetz, 1997) and wireless sensor networks (Wang et al., 2010). (Yick et al., 2008 ; Sengupta et al., 2013). While they have less hardware resources and are severely underpowered, they are incapable of dealing with more complex tasks. Though these devices are capable of incorporating learning schemes, recent advances in the computing power of microcontrollers and a shift in programming paradigms enable the inclusion of learning schemes on the chips. These devices adjust their behavior based on the data detected. Due to their low power consumption and low cost, microcontrollers are frequently used in motion detection systems. Helping those with kinetic challenges is enhanced by input devices designed specifically for them, such as microcontroller-based input devices that measure movement in real time (Papadimitriou et al., 2006). a prototype PCB design includes a microcontroller integrated to detect movement and proximity (Dobrzynski et al., 2012). The eight photodiodes are used as light sensors in this prototype. A significant drawback of using solar power plants is that they use large amounts of energy to estimate the movement of clouds (Fung

et al., 2014). The management of the solar panels is optimized for maximum electricity output, because it uses a built-in microcontroller to approximate the motion vectors of the clouds. Finally, with low-power motion detection systems, microcontrollers, and wireless communication devices, it is possible to achieve street lighting with low power consumption (Adnan et al., 2015). The street lights turn on whenever someone is nearby. This chapter details the application of the SOM (self-organizing machine) on an Arduino DUE (Arduino Due-based) board, as well as the implementation of a self-learning motion detection algorithm for device decisions in all conditions, avoiding non-online computing and communication with other devices. Popular, inexpensive, and with a microcontroller on a single board, the Arduino DUE is highly used, and open source for easy project development (Lian et al., 2013; Cela et al. ., 2013; Ortega-Zamorano et al., 2015; Kornuta et al., 2012). It has also been proposed to use fixed-point representation for data used in Arduino programming, in order to attain a more rapid system with fewer hardware resources. In this type of device, the SOM can be used..

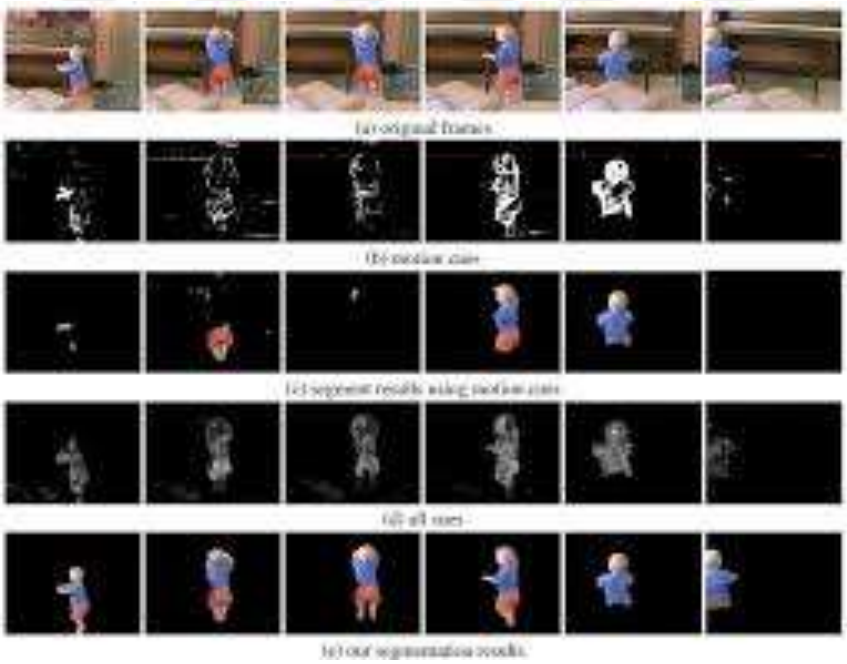


Figure 2.5: Example Dataset.

2.4 DATASET

To increase the speed of the learning process, it has been decided to use a fixed-point representation of the data. Also, the floating-point representation is typically the most common, but it may not always be the most effective. A radical shift occurred in the methods used to represent the type of data. A variable of type “floating” (“float”) or “double” (“double”) that holds a 4- or 8-byte value, respectively, is allocated for the floating-point representation. This means that a variable of type "integer" of size 4 bytes holds the fixed-point representation. With this paradigm shift, there will be radical changes to the way the SOM model is programmed, but the speed of learning will increase and more specific variables will be possible to represent. The figures in Table 5.1 show the microsecond (μs) time it takes to calculate the basic arithmetic operation (+, -, *, /) on the Arduino DUE microcontroller with the listed variable types (integer, float, and double). Newer motion detection techniques typically use a color model for each pixel (Bouwmans, 2014b). This is not feasible because of the microcontroller's hardware constraints. Therefore, the block subdivision approach is suggested, in which each region's color model is learned on an individual basis. To get the corresponding RGB color value for each pixel, it is assumed that the input video frame has Nrow × Ncol pixels, and that each pixel has an RGB color vector yh ∈ [0, 1] 3 whose coordinates are h ∈ {1, ..., Nrow} × { 1, ..., Ncol}. The next step is to divide the input frame into blocks of Nrow Brow pixels, where Nrow is a multiple of Brow, and into blocks of Ncol Bcol pixels, where Ncol is a multiple of Bcol. To summarize color information for each block, it is necessary to describe the color data contained in the pixel's color information y h in a concise manner. the means of the colors of each block are being calculated.

3. PROPOSED METHOD

3.1 NETWORK ARCHITECTURES USED

You can use the specific math library (math.h) to evaluate the exponential functions in the model. Computational time: This procedure uses a microcontroller, and the computational time of this procedure is equal to 58.9 microseconds. To reduce computational time, an approximation for the exponential function has been implemented. With the reduction in time, it is possible to update more blocks of pixels in a given amount of time, which means it is possible to process a greater number of blocks of pixels in total. A lookup table search followed by a linear interpolation has been used to find an approximation. This approach has been the subject of extensive investigation in the past (Ortega-Zamorano et al., 2014a). For equally spaced values of the independent variable, the table displays the values of the exponential function. on the other hand, because you need:

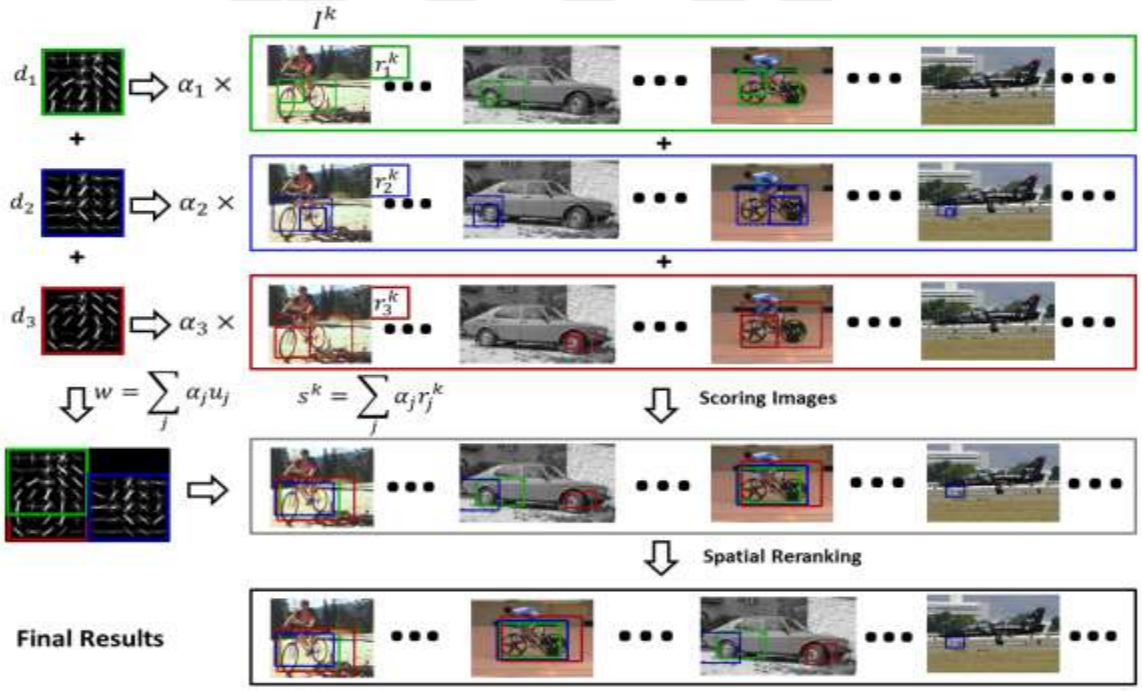


Figure 3.1: Numerous Precisely Determined Values To Perform The Algorithm Correctly.

numerous precisely determined values To perform the algorithm correctly, an interpolation procedure based on two adjacent tabulated values (i.e., lower and larger) is employed in addition to the calculation of the function approximation. This case reduces the computing time to 1,437 nanoseconds, which means a 97.5% reduction compared to “math.h”, the standard library. Memory is needed depending on the accuracy of the approximation when storing table values. Figure 4.2

shows the memory requirement according to the level of accuracy that you require for your approximated method that utilizes the table and linear interpolation. A $5 \cdot 10{-}5$ absolute error has been selected. Because of this, the lookup table requires a memory size of 4 KB. This figure illustrates the absolute error, in terms of standard deviations, involved in calculating the negative exponential function from 0 to 16..

3.2 IMAGE FILTERING

A digital image is a two-dimensional representation of an image in the form of a numerical matrix. In grayscale images, pixels are represented by an integer numeric value that is between 0 and 255. One of the main problems encountered today is the appearance of noise in digital images. The main sources of noise appear during the image acquisition and transmission phases. There are many types of noise and among the best known are some such as Gaussian or impulsive. For the elimination or minimization of noise there are many methods such as the filtering process. They are a set of techniques whose main objective is to obtain, from an initial image with a certain noise, a final one whose result is another image with better quality. The main goals of applying filters are to smooth the image, remove noise, enhancement, and edge detection, in this paper we propose the combining of the powerful set of Fuzzy rules with the genetic algorithm pattering recognition features to filter the image from impulsive noise yet retaining the image features unfiltered thus performing a powerful noise filtering on the noise without and destruction to the image itself.

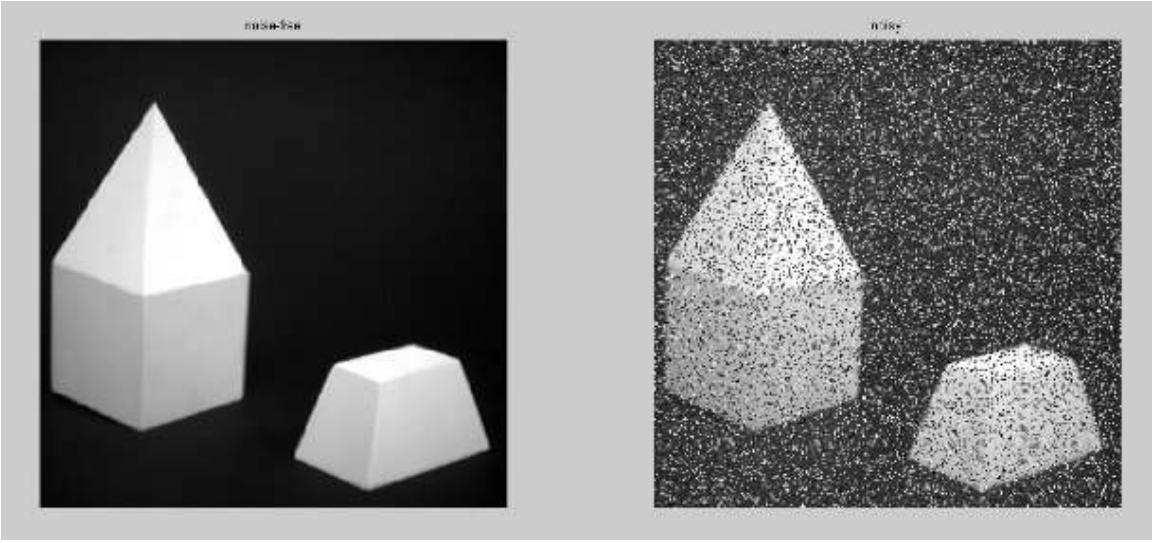


Figure 0.2: The effects of impulsive noise on an image.

The objectives of this project are various. The initial objective is to understand and replicate the FIRE algorithm, developed by Fabrizio Russo in 1998. Other objectives of this project consist of making a series of proposals or modifications on the original proposal. These proposals lie in the optimization of the parameters used by the fuzzy sets. In the original proposal, these parameters are fixed throughout the process, but in the proposals that will be made, they will be modified to be variable in order to give the genetic algorithm greater freedom and try to achieve better results. Finally, it is intended to devise a system that is capable of training with several images with Salt and Pepper noise to obtain a final system that can be used on any other image with this noise and achieve satisfactory results. This system is designed because we believe that it makes no sense to devise an algorithm to treat an image individually. We think that the important thing is to devise a system that can be applied to any image the rest of the paper is organized as follows: we will be explaining the methods and methodology in section 2, while in section 3 we will explain our proposed method, in section 4 we will put our results and in section 5 we will conclude our work.

3.2.1 Fuzzy Logic Filtering

One of the mathematical disciplines with the largest number of researchers today is the so-called fuzzy logic. It uses expressions that are neither totally true nor completely false, that is, it is the logic applied to concepts that can take any value of truth within a set of values. Fuzzy logic allows you to deal with imprecise information such as average height or low temperature. The concept of fuzzy logic was conceived by Lofti A. Zadeh in 1965. Zadeh was dissatisfied with the classic sets that only allowed two options (the belonging or not of an element to said set), and presented fuzzy logic as a way of processing the information allowing partial memberships to some sets. The original intention of the Zadeh was to create a formalism to more efficiently manipulate the imprecision and vagueness of linguistically expressed human reasoning. However, it came as some surprise that the success of fuzzy logic came in the field of automatic process control due to the tremendous interest caused in Japan back in 1987. In 1965 Lotfi Zadeh introduced fuzzy set theory. A fuzzy set A defined on a finite and nonempty universe U = {u1,…, un} is given by: A: {(ui, μA (ui)) | ui ϵ U} where μA: U → [0,1] is such that μA (ui) ϵ [0,1] denotes the degree of membership of the element ui to the set A. Therefore, the closer the value of μA (ui) to 1, the greater the degree of membership of ui in A. When A is a set in the classical sense, its membership function can take only two values 0 and 1, that is, μA (ui) = {1 or 0} when ui does or does not belong to A,

respectively. In the real world, there is a lot of non-perfect knowledge, that is, knowledge that is vague, imprecise, uncertain, ambiguous, inaccurate, or probabilistic in nature. Human reasoning and thinking frequently carries information of this type, probably caused by the inherent inaccuracy of human concepts and reasoning based on experiences similar but not identical to previous experiences. Fuzzy logic is an alternative logic to classical logic that aims to introduce a degree of vagueness in the concepts that qualifies to model what has been described above.fig.1 shows the fuzzy logic workflow:

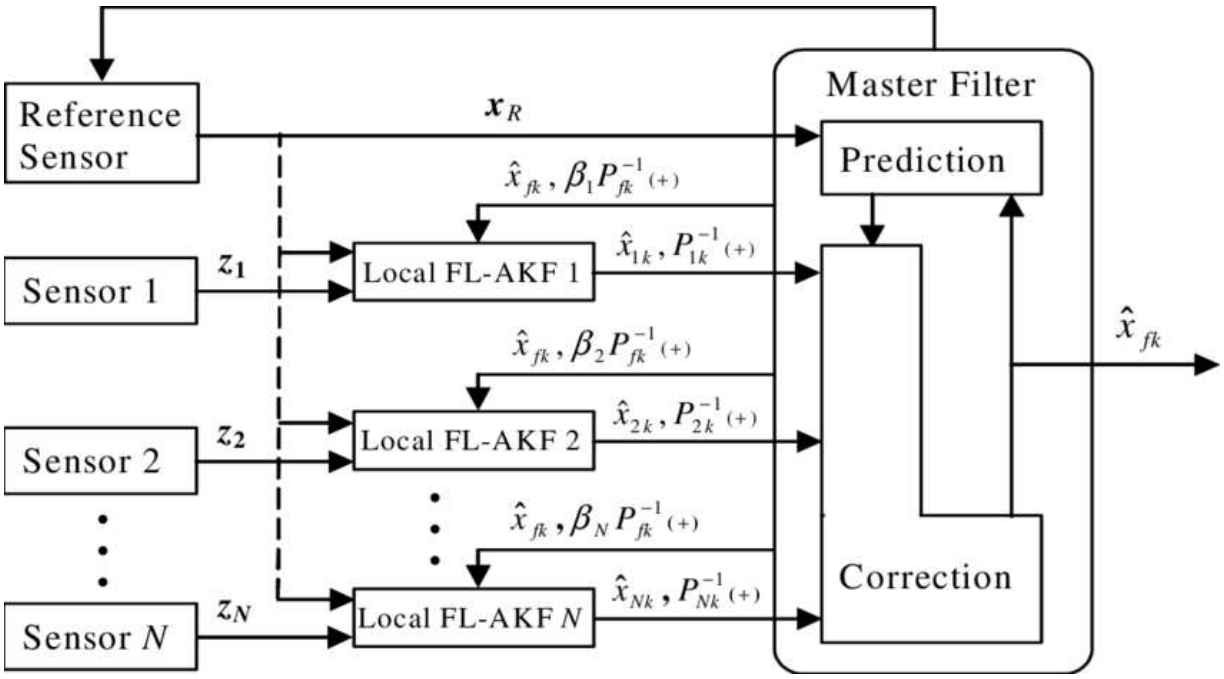


Figure 3.3: Fuzzy logic image filtering workflow.

3.2.2 Genetic Algorithm Filtering

ver the years, humans have managed to increase their ability to interact with previously unthinkable environments and understand the difficulty of many others in which they are not so capable. Therefore, your ability to predict the nature of your environment becomes increasingly important. Among the sciences that have done the most to achieve these achievements, artificial intelligence ranks high. In the eighties, the interest in biology as a source of inspiration for computer science began to be a reality among scientists, who saw how this interest and its subsequent development have led them to achievements never before thought. Achievements that have been achieved with developments such as neural networks, learning or evolutionary

computing and their genetic algorithms. Genetic algorithms mimic nature's evolutionary process to solve search or optimization problems. Specifically, each individual in a genetic algorithm represents a solution to the problem and those best adapted will survive, that is, those who offer a better solution to it. For this, mechanisms of nature such as selection, reproduction or mutation are imitated.

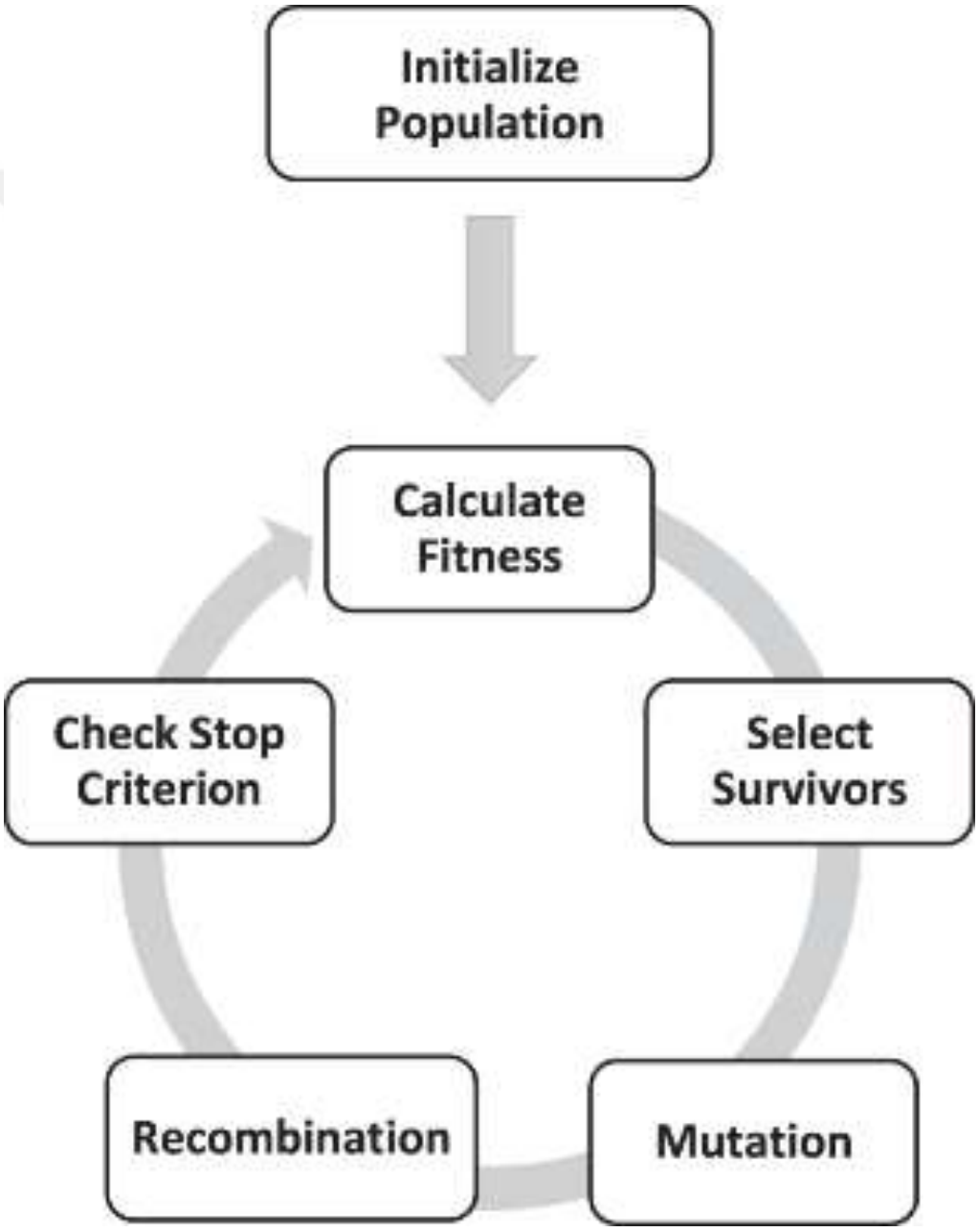


Figure 3.4: Genetic algorithm workflow.

Some of the basic concepts of the different elements and components that make up a genetic algorithm are:

i. Chromosome or individual: Set of genes. A chromosome or individual represents the solution to a given problem in the form of genes, which are those that represent the parameters of the system to be optimized or of the problem to be solved.

ii. Population: Set of individuals that uses the genetic algorithm to solve the problem.

iii. Fitness (Adaptation, quality): Value assigned to each individual based on how far or near the solution is. The higher the fitness value of a chromosome, the better solution it represents.

iv. Fitness function: Function that assigns each individual their fitness. It depends on each specific problem.

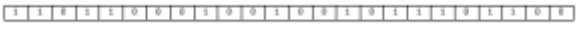
v. Crossing: An operation that takes two individuals and combines their chromosomes to obtain new individuals.

vi. Mutation: An operation that slightly modifies a gene in an individual.

vii. Selection: Operation that chooses individuals to carry out the crossing operation.

3.2.3 Proposed Filtering Scheme

This section will describe certain technical aspects of the configuration in the implementation of the Fuzzy Genetic Filtering algorithm. The process and its mechanics follow exactly the same steps as those described in the FIRE Algorithm section, except for an extension explained later. The representation of a chromosome is defined as a string of Mx8 bits:



This chromosome has been set as the initial one. It has 24 elements so it has 3 8-bit substrings and as described in the rule generation, 24 rules will be generated. The parameters of the fuzzy sets to be fuzzified are fixed throughout the program are:

In the process of adjusting the genetic algorithm, the author does not detail anything about any of its parameters. It does not impose a method of selection of parents, crossing, mutation, etc. Therefore, those used in this and the other proposals will be detailed. All of them are explained in detail in the Preliminary Concepts. When starting the genetic algorithm, the initial population consists of 20 individuals (the imposed initial chromosome and 19 random ones).

The stop condition is determined by two parameters: - A maximum number of iterations = 200 is reached - During 10 generations, the Fitness of the best chromosome does not improve by 0.000001. It is recalled that $FFFFFFFFFFFFF = 1\ MMMMMM$ the selection of parents is carried out using the roulette method. To cross the parents, a point crossing is used. In our case, since the chromosome is made up of M 8-bit strings, a crossing is made for each Substring (in our case 3). Furthermore, the crossover is made with a probability of 0.9. If this does not occur, the descendants will be the parents themselves. The mutation is made on each gene and has a probability of being 0.05. Once the new offspring (20 individuals) are evaluated, the next generation will consist of the 20 best individuals (from between the current generation and the new offspring). That is, the highest level of elitism possible is applied to fully guide the search for the best solutions. Once the genetic algorithm has been completed, the chromosome that represents the best rule base for an initial noisy image will have been obtained. After obtaining the winning chromosome, the initial noise image is cleaned up and then the same rule base is applied again using the previously filtered image. This second image will be the final solution of the algorithm. This extension has been called the best image recovery and cleaning process.

Below are the different images used to carry out the tests. In total, 27 images have been used, of which 7 have been used for training and the rest for testing. All images that have been used are 256x256 pixels in size. The tests have been performed with the same images but with different noise densities (0.1, 0.18 and 0.3). For the development of the tests, the algorithms proposed in the previous chapter have been used: 1. General cleaning algorithm + Implementation of the FIRE algorithm 2. General cleaning algorithm + Adjustment of the LP fuzzy set parameters 3. General cleaning algorithm + Adjustment of the parameters of the fuzzy set LP and LN 4. General cleaning algorithm + Optimization of the parameters of the fuzzy set LP for each input variable Δxjj For all the proposals the configuration of the parameters is identical: - Number of binary strings of Substrings chromosome: M = 3 - Neighborhood window size: 3x3 - Fitness: 1 $MMMMMM$ - Fuzzy

set parameters: $aa0$ = 128, $aa1$ = 120, $aa2$ = 40, bb = 127 yy cc = 0.5. This fuzzy set is always included in all proposals as the initial chromosome. - Initial chromosome

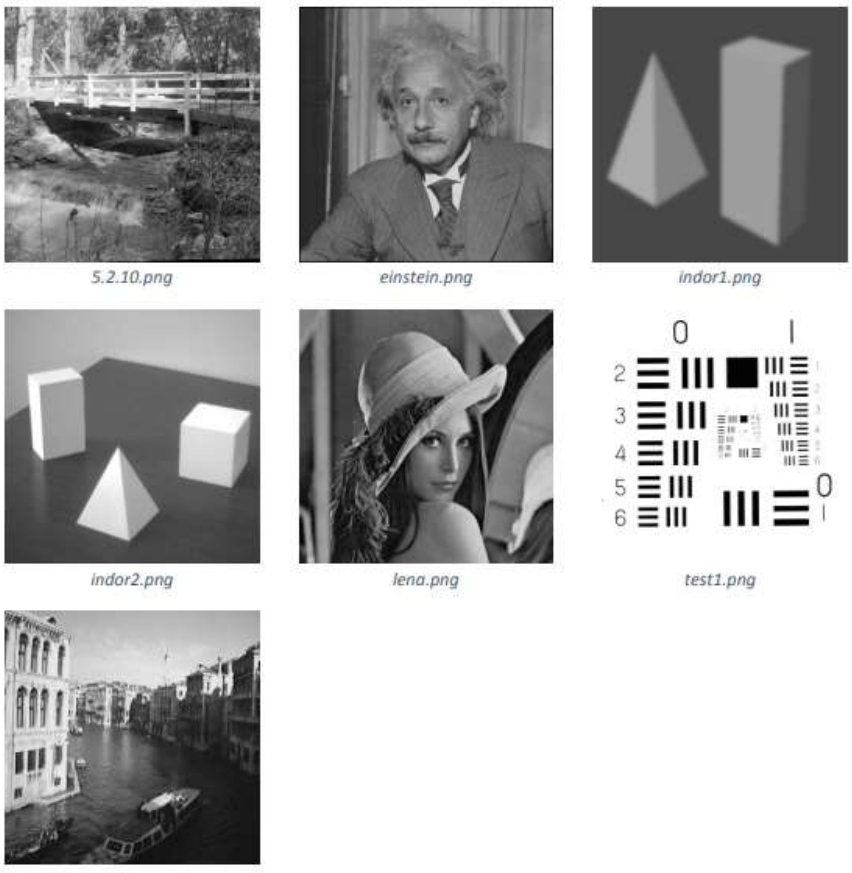


Figure 0.5: the MSE to the number of EPOCHs.

Some of the most common to measure the quality of filtered images are:

Mean Square Error MSE

The lower the MSE value, the more similar the filtered image is to the original image. In the ideal situation where the filtering is perfect, the MSE obtained will be 0.

$$MSE = \frac{1}{MN}\sum_{x=1}^{M}\sum_{y=1}^{N}(f'(x,y) - f(x,y))^2 \tag{4.1}$$

Peak Signal to Noise Ratio (PSNR)

The higher the PSNR value, the more similar the filtered image is to the original image. In the ideal situation in which the filtering is perfect, the PSNR obtained will be + ∞.

$$PSNR = 10 \log_{10} \frac{(L-1)^2}{MSE} \tag{4.2}$$

Table 0.1: PSNR and MSE of the Fuzzy-Genetic, Median, and Gaussian filters.

Method	PSNR	MSE	Noise ratio
Gaussian	31.96	2.51	25%
Median	32.85	2.13	25%
Fuzzy-Genetic	59.24	0.039	25%

3.3 SEGMENTATION USING TRANSFER LEARNING

In many applications, neural networks are usually trained from scratch, and rely solely on the training data to adjust their parameters. However, as more and more networks are trained for various tasks, it is reasonable to look for methods that avoid having to start from scratch and can instead build on the results of previously trained networks [49] . In the case of object detection, since the networks tend to be very deep, and therefore the number of parameters to be adjusted is very high, for efficient training a base is required very extensive training data, as well as a very high training time. Therefore, it is usual to start from a network previously trained for a similar purpose instead of initializing the weights and biases randomly. Object detection networks that have been trained with large data sets and to detect a wide variety of objects have the ability to distinguish and identify multiple shapes, which means that starting from the parameters of a pre-trained network is much more convenient than training it from scratch in many cases, especially when there is not a large number of training examples. The first objective of this chapter is to compare two different initializations. The Tensorflow object detection model zoo repository [28] offers the possibility to download several pretrained models with different databases. The neural network that will be compared in this case is a Faster-RCNN [21] with the convolutional network Resnet 101 [33] as a feature extractor. The two models differ in the databases with which they have been trained, one has been trained with the COCO data and the other with the KITTI data.

These networks will be retrained to perform the role of detecting people, or pedestrians if speaking in terms related to traffic. For this, a subset of 1000 images extracted from the data set of [50] has been used, in which a total of 8466 pedestrians appear. For the evaluation of the models, 446 images containing 3474 annotated pedestrians were used, from a different set, also from [50]. In the final part of this chapter, we will train an object detection network starting from scratch, the same architecture as the one compared in the previous section. In this case, we will do a training with a greater number of epochs, since the initialization is random and the convolutional module will have to learn the characteristics from scratch. The performance of this network will be compared with the two previously analyzed to be able to observe in greater detail the importance of the transfer of learning in very deep networks.

3.3.1 Object Detection Using Transfer Learning

As explained in the previous section, the databases used to train the two initializations of parameters to be compared are COCO and KITTI:

• The database COCO1 is a large set that contains more than 200,000 images distributed in 80 classes of objects that represent scenes from the real world [51]. It is a large enough database so that a well-trained network with this data set is able to learn quality visual characteristics for the recognition and detection of objects in images.

• KITTI2 is a database compiled by different sensors, such as cameras, stereo pairs or LiDARs, placed in a car [14]. Therefore, it contains 2D and 3D data. The data set for object detection and object orientation estimation consists of 7481 training images and 7518 evaluation images, comprising a total of 80,256 labeled objects. To make the comparison between the two models, they have been retrained under identical specifications:

• Image size: 640 × 480 pixels

• Mini-batch size: 2 images

• N´ Number of steps: 200,000

• Learning speed: 0.0002

• Initial aspect ratios of the anchors: [0.25, 0.33, 0.5, 1.0, 2.0]

• Initial scales of the anchors: [0.25, 0.5, 1.0, 2.0]

• Maximum number of regions proposed: 100

• Dropout regularization.

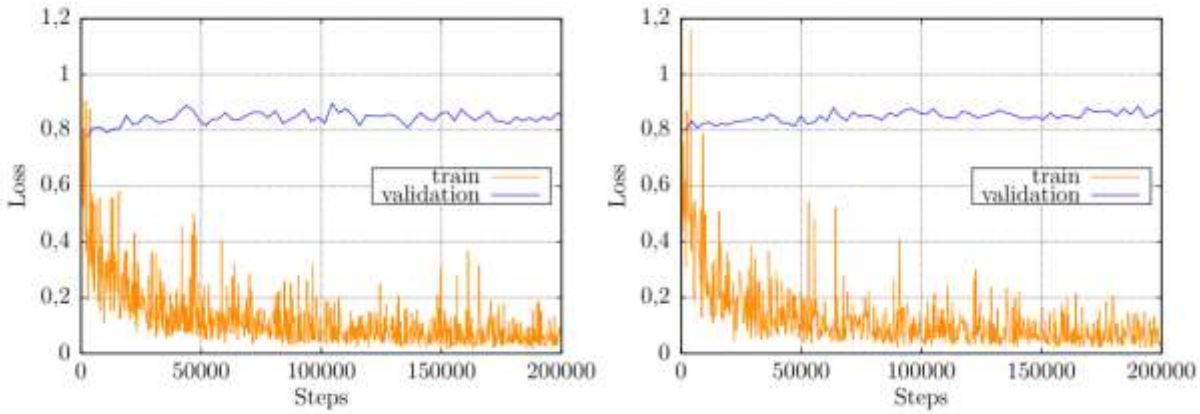


Figure 3.6: Although The Cost On The Training Data Is Low.

As can be seen, although the cost on the training data is low (this is logical, since the parameters are adjusted for this to occur), the cost applied to the evaluation examples does not vary much, it even starts to rise slightly. It can also be seen that the values are very similar for both initializations of weights. To evaluate the results of the detectors in greater detail, the following section will explain the metrics used.

To measure the performance of the models there are two important questions, the first one refers to the speed of inference, that is, how long does it take for the model to receive an image at the input, process it and return the detections. The second question is more focused on the ability of the model to make good detections. This section will focus on the second question, for that two measures have been used, that of precision:

$$Precision = \frac{TP}{TP + FP} \tag{4.3}$$

The detection model returns not only detections but also probabilities that such detection is correct. By setting different thresholds, a precision and completeness curve can be constructed, calculating the values of each one for a given threshold:

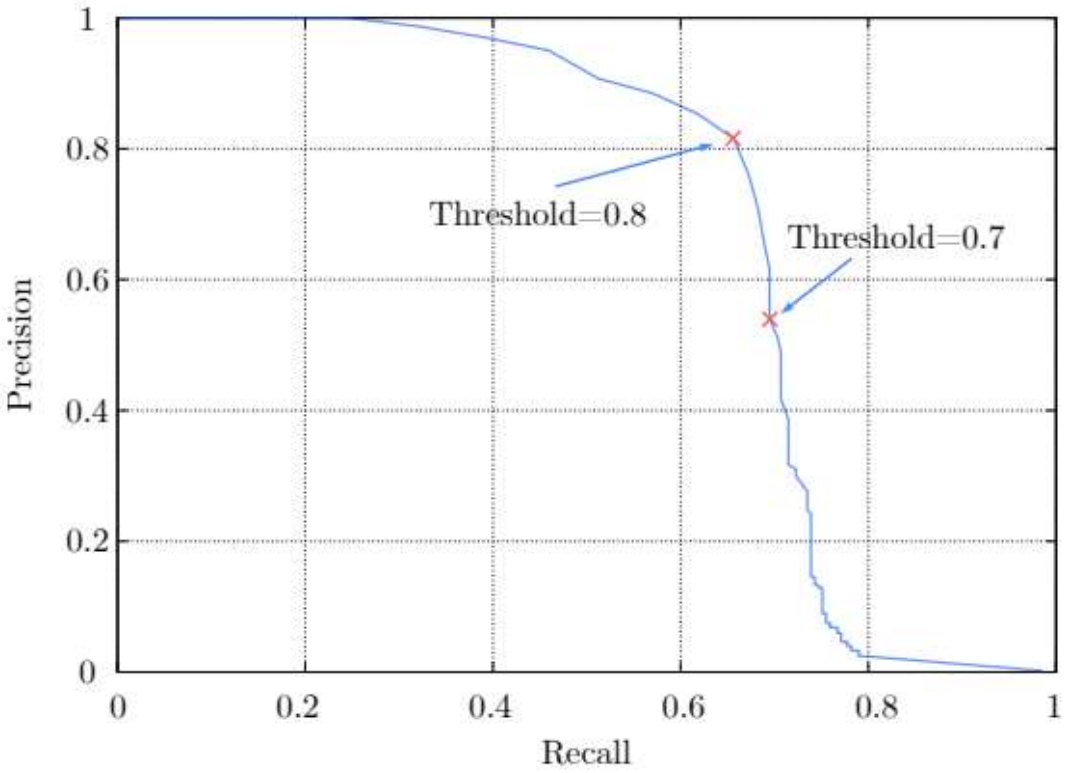


Figure 0.7: The area under the precision and completeness curve is known as the mean precision or Average Precision.

The area under the precision and completeness curve is known as the mean precision or Average Precision. In the particular case of pedestrian detection there is only one class to predict, but generally object detectors tend to have multiple classes. The mean precision is calculated for each class and the mean precision (mAP) provides a more general view of how good the detector is for all classes. For a total number of N classes:

$$mAP = \frac{1}{N} \sum_{class=1}^{N} AP_{class} \tag{4.4}$$

The criterion used to determine if a detection is good, that is, if it is a true positive, has to do with the IoU (Equation 3.31). If a detection with a given class has an IoU greater than a threshold with a ground-truth of that class, then it will be considered as good, otherwise it will be considered as a false positive. The measure of the mean precision or mAP for a single threshold value of IoU = 0.5 is known as the PASCAL mAP. There is another measure, the COCO mAP, and it is also commonly used to evaluate object detectors. This last metric is calculated by making the average of ten values of mAP, obtained for the values of IoU=[0.5, 0.55, . . ., 0.95].

In this section the models will be evaluated, for this purpose the evaluation data set will be used exclusively. For each step of the training, a mAP value of both PASCAL and COCO is obtained. The following graphs show the evolution of these measures throughout the training:

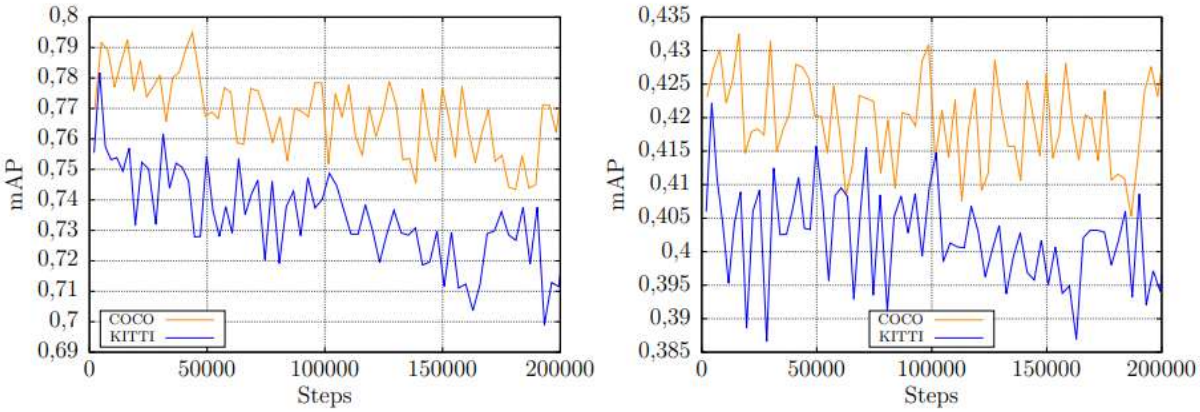


Figure 0.8: The Value Of The Map In The Large And Medium Sizes Oscillates.

As can be seen in the curves of Figure 4.4, while the value of the mAP in the large and medium sizes oscillates, but does not show a downward trend, in the small pedestrians there does appear a negative slope, quite Remarkable for the model initialized with KITTI weights. The following images show a few significant examples of pedestrian detections in busy scenes:

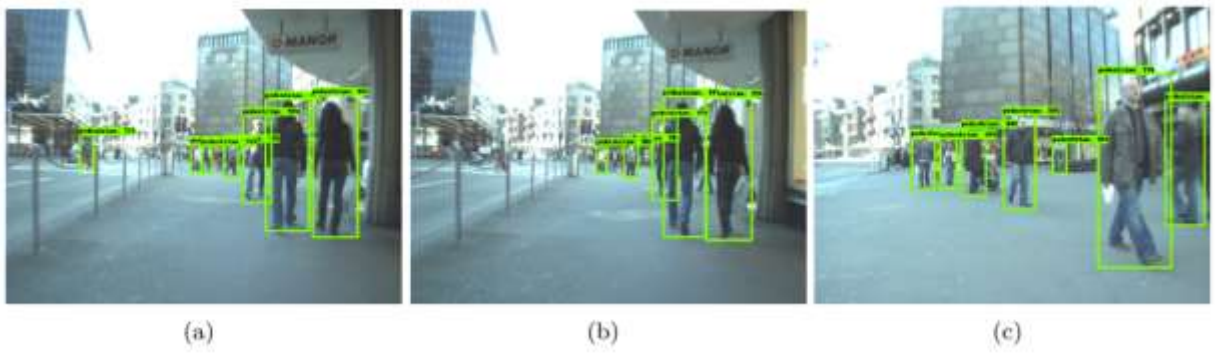


Figure 3.9: Comparing The Two Networks Trained Identically And With Different Initializations Face To Face.

In general, comparing the two networks trained identically and with different initializations face to face, the highest performance is offered by the network pre-trained with COCO. This phenomenon is not very surprising since it is a considerably larger data set than KITTI. In the superimposed curves of Figures 4.3 and 4.4, as well as in Table 4.1, the network pretrained with COCO shows the best values in all cases. Still, the difference from KITTI initialization is not very

great. On the other hand, Figure 4.1 shows the value of the cost function during training in the two models, both for the training set and for the evaluation set. The value of the training cost and the evaluation cost is very similar at the beginning of the training, this may be logical, since both databases (those that have been used to train and evaluate) have similar characteristics , have been captured with the same camera and both are subsets of the same data set. Throughout the training the cost value on the training data decreases, as is to be expected, since the parameters are adjusted precisely so that this happens. However, the cost curve on the evaluation data hardly changes in the two models, which may be indicative that, although the model learns to better classify the training data set, it is not capable of generalizing this learning to external data, even getting slightly worse (Figure 4.3), giving signs of a small overfitting to the training data. It is also significant that in the comparison of the mAP for different sizes (Figure 4.4), a considerable difference appears between the decrease in the mAP for small pedestrians if the COCO curve and the KITTI curve are compared. Taking into account that in Figure 4.3, at the end of the training the difference in mAP values between the two models is greater than that which exists at the beginning, the decrease in small pedestrians may be a cause of this happening. Furthermore, Figures 4.5 and 4.6 show examples of how the pretrained COCO model is more effective in classifying small pedestrians than the KITTI model. In the training database [50], pedestrians with a pixel height size less than 48 pixels are not labeled. This feature of the database can lead to lower performance when classifying small objects. Finally, putting into context the results obtained with the state of the art (Table 2.2 b), although the databases used to evaluate the models are not the same, the values of mAP with IoU = 0.5 obtained for the two trained models show a performance very similar to that of the Faster-RCNN in the “Easy” mode”

4. CONCLUSION

A new unsupervised learning model has been presented, which can adapt to input distributions that vary over time. It has been integrated into a computer vision system for the detection of foreground objects in video sequences. According to the qualitative and quantitative results offered, the proposed learning model has been successfully applied to unstable distributions, which are common in computer vision applications. In this sense, a neural model with two learning mechanisms has been developed, one of them to follow the general flow of the input distribution and the other to adjust the neurons to specific groups of the input. Experimental results have been shown in detail, using different sets of videos and comparing with state-of-the-art methods from a qualitative and quantitative point of view. The proposal has obtained good results in a comparison with several state-of-the-art algorithms for the background modeling problem, tested on a set of videos that belong to one of the most recent and used public comparison standards. In particular, the results have demonstrated the effectiveness and robustness of the competitive learning proposal to manage situations without special problems and the appearance of shadows. it can be said that the objectives of this work have been successfully met. It has been understood and managed to replicate the original FIRE proposal. In addition, new proposals for improvement have been devised and a general training-test system has been created that serves to treat any image a posteriori. The results for algorithms 2, 3 and 4 (the new proposals) have not been as expected because it is assumed that they should improve algorithm 1 (since algorithm 1 should be a particular case of the proposals). This is due to the fact that the rule base that represents the winning chromosome of algorithm 1 is of better quality than those obtained with the proposals. The explanation for all this is that by adding more parameters to our chromosome, its search space is much greater and consequently, it makes it lose itself in it. For all these proposals, a study has been carried out which shows the quality of the proposals that have been prepared and which demonstrate their success. After carrying out the experimental study and based on the results obtained in the tests for each of the noise levels, the following lessons learned can be highlighted: For a low noise level (0.1), the proposal that works best is the Algorithm 1 (General cleaning algorithm + original FIRE algorithm). Although it is the best method in 16 of the 27 images, algorithm 3 (General cleaning algorithm + Adjustment of the LP and LN fuzzy set parameters) has a fairly similar ranking mean. If algorithm 1 has an average of 1.67, algorithm 3 has 1.85. Despite

this, the reality is that algorithm 1 is the best method overall. Algorithms 2 and 4 obtain quite similar results between them but very discrete compared to algorithms 1 and 3. Although a priori it might be thought that adding more parameters to the chromosome in order to grant more freedom of learning to the genetic algorithm, does not a better noise removal pattern is obtained.

REFERENCES

[1] N. M. Borden, S. E. Forseen y C. Stefan, Imaging Anatomy of the Human Brain: A Comprehensive Atlas Including Adjacent Structures. Springer Publishing Company, 2015, isbn: 9781936287741 (vid. págs. 3, 4).

[2] J. E. Hall, Guyton and Hall Textbook of Medical Physiology, ép. Guyton Physiology Series. Elsevier, 2016, isbn: 9781455770052 (vid. pág. 3).

[3] E. C. Holland, "Progenitor cells and glioma formation.", eng, Current opinion in neurology, vol. 14, n.o 6, págs. 683-688, dic. de 2001, issn: 1350-7540 (Print) (vid. pág. 5).

[4] S. Ferguson y M. S. Lesniak, "Percival Bailey and the classification of brain tumors.", eng, Neurosurgical focus, vol. 18, n.o 4, e7, abr. de 2005, issn: 1092-0684 (Electronic) (vid. pág. 5).

[5] D. N. Louis, A. Perry, G. Reifenberger, A. von Deimling, D. Figarella-Branger, W. K. Cavenee, H. Ohgaki, O. D. Wiestler, P. Kleihues y D. W. Ellison, "The 2016 World Health Organization Classification of Tumors of the Central Nervous System: a summary.", eng, Acta neuropathologica, vol. 131, n.o 6, págs. 803-820, jun. de 2016, issn: 1432-0533 (Electronic). doi: 10.1007/s00401-016-1545-1 (vid. págs. 5, 6).

[6] J. W. Kernohan y R. F. Mabon, "A simplified classification of the gliomas.", eng, Proceedings of the staff meetings. Mayo Clinic, vol. 24, n.o 3, págs. 71-75, feb. de 1949, issn: 0092-699X (Print) (vid. pág. 5).

[7] C. Daumas-Duport, B. Scheithauer, J. O'Fallon y P. Kelly, "Grading of astrocytomas. A simple and reproducible method.", eng, Cancer, vol. 62, n.o 10, págs. 2152-2165, nov. de 1988, issn: 0008-543X (Print) (vid. pág. 5).

[8] B. W. S. (P. Kleihues P. C. Burger, Histological Typing of Tumours of the Central Nervous System, 2.a ed., ép. WHO. World Health Organization. International Histological Classification of Tumours. Springer-Verlag Berlin Heidelberg, 1993, pág. 114, isbn: 978-3- 540-56971-8. doi: 10.1007/978-3-642-84988-6 (vid. pág. 5).

[9] J. J. Plascak y J. L. Fisher, “Area-Based Socioeconomic Position and Adult Glioma: A Hierarchical Analysis of Surveillance Epidemiology and End Results Data”, PLoS ONE, vol. 8, n.o 4, J. S. Castresana, ed., e60910, abr. de 2013, issn: 1932-6203. doi: 10.1371/ journal.pone.0060910 (vid. pág. 6).

[10] H. Ohgaki y P. Kleihues, “Epidemiology and etiology of gliomas”, Acta Neuropathologica, vol. 109, n.o 1, págs. 93-108, ene. de 2005, issn: 1432-0533. doi: 10.1007/s00401-005- 0991-y (vid. págs. 6, 11).

[11] S. Antoni, I. Soerjomataram, B. Møller, F. Bray y J. Ferlay, “An assessment of GLOBOCAN methods for deriving national estimates of cancer incidence”, Bulletin of the World Health Organization, vol. 94, n.o 3, págs. 174-184, mar. de 2016, issn: 0042-9686. doi: 10.2471/ BLT.15.164384 (vid. pág. 6).

[12] B. H. Menze, A. Jakab, S. Bauer, J. Kalpathy-Cramer, K. Farahani, J. Kirby, Y. Burren, N. Porz, J. Slotboom, R. Wiest, L. Lanczi, E. Gerstner, M. A. Weber, T. Arbel, B. B. Avants, N. Ayache, P. Buendia, D. L. Collins, N. Cordier, J. J. Corso, A. Criminisi, T. Das, H. Delingette, Ç. Demiralp, C. R. Durst, M. Dojat, S. Doyle, J. Festa, F. Forbes, E. Geremia, B. Glocker, P. Golland, X. Guo, A. Hamamci, K. M. Iftekharuddin, R. Jena, N. M. John, E. Konukoglu, D. Lashkari, J. A. Mariz, R. Meier, S. Pereira, D. Precup, S. J. Price, T. R. Raviv, S. M. S. Reza, M. Ryan, D. Sarikaya, L. Schwartz, H. C. Shin, J. Shotton, C. A. Silva, N. Sousa, N. K. Subbanna, G. Szekely, T. J. Taylor, O. M. Thomas, N. J. Tustison, G. Unal, F. Vasseur, M. Wintermark, D. H. Ye, L. Zhao, B. Zhao, D. Zikic, M. Prastawa, M. Reyes y K. V. Leemput, “The Multimodal Brain Tumor Image Segmentation Benchmark (BRATS)”, IEEE Transactions on Medical Imaging, vol. 34, n.o 10, págs. 1993-2024, 2015, issn: 0278-0062 VO - 34. doi: 10.1109/TMI.2014.2377694 (vid. págs. 6, 11, 13, 17, 18, 20).

[14] R. A. y L. M., “Burden and cost of neurological diseases: a European North–South comparison”, Acta Neurologica Scandinavica, vol. 132, n.o 1, págs. 16-22, doi: 10.1111/ane.12339 (vid. pág. 7).

[15] A. R. Giovagnoli, R. F. Meneses, A. Silvani, I. Milanesi, L. Fariselli, A. Salmaggi y A. Boiardi, “Quality of life and brain tumors: what beyond the clinical burden?”, Journal of

Neurology, vol. 261, n.o 5, págs. 894-904, mayo de 2014, issn: 1432-1459. doi: 10.1007/ s00415-014-7273-3 (vid. pág. 7).

[16] E. Alpaydin y F. Bach, Introduction to Machine Learning. MIT Press, 2009, isbn: 9780262303262 (vid. pág. 7).

[17] G. Chartrand, P. M. Cheng, E. Vorontsov, M. Drozdzal, S. Turcotte, C. J. Pal, S. Kadoury y A. Tang, "Deep Learning: A Primer for Radiologists", RadioGraphics, vol. 37, n.o 7, págs. 2113-2131, 2017. doi: 10.1148/rg.2017170077 (vid. págs. 7, 8).

[18] V. K. Ho, J. C. Reijneveld, R. H. Enting, H. P. Bienfait, P. Robe, B. G. Baumert y O. Visser, "Changing incidence and improved survival of gliomas", European Journal of Cancer, vol. 50, n.o 13, págs. 2309-2318, sep. de 2014, issn: 0959-8049. doi: 10.1016/J.EJCA.2014. 05.019 (vid. pág. 11).

[19] MICCAI BraTS 2017: People | Section for Biomedical Image Analysis (SBIA) | Perelman School of Medicine at the University of Pennsylvania (vid. pág. 11).

[20] E. Eisenhauer, P. Therasse, J. Bogaerts, L. Schwartz, D. Sargent, R. Ford, J. Dancey, S. Arbuck, S. Gwyther, M. Mooney, L. Rubinstein, L. Shankar, L. Dodd, R. Kaplan, D. Lacombe y J. Verweij, "New response evaluation criteria in solid tumours: Revised RECIST guideline (version 1.1)", European Journal of Cancer, vol. 45, n.o 2, págs. 228-247, ene. de 2009, issn: 0959-8049. doi: 10.1016/J.EJCA.2008.10.026 (vid. pág. 11).

[21] P. Y. Wen, D. R. Macdonald, D. A. Reardon, T. F. Cloughesy, A. G. Sorensen, E. Galanis, J. Degroot, W. Wick, M. R. Gilbert, A. B. Lassman, C. Tsien, T. Mikkelsen, E. T. Wong, M. C. Chamberlain, R. Stupp, K. R. Lamborn, M. A. Vogelbaum, M. J. van den Bent y S. M. Chang, "Updated response assessment criteria for high-grade gliomas: response assessment in neuro-oncology working group.", eng, Journal of clinical oncology : official journal of the American Society of Clinical Oncology, vol. 28, n.o 11, págs. 1963-1972, abr. de 2010, issn: 1527-7755 (Electronic). doi: 10.1200/JCO.2009.26.3541 (vid. pág. 11).

[22] D. Hanahan y R. A. Weinberg, “Hallmarks of cancer: the next generation.”, eng, Cell, vol. 144, n.o 5, págs. 646-674, mar. de 2011, issn: 1097-4172 (Electronic). doi: 10.1016/ j.cell.2011.02.013 (vid. pág. 14).

[23] J. Juan-Albarracín, E. Fuster-Garcia, J. V. Manjón, M. Robles, F. Aparici, L. MartíBonmatí y J. M. García-Gómez, “Automated Glioblastoma Segmentation Based on a Multiparametric Structured Unsupervised Classification”, PLOS ONE, vol. 10, n.o 5, e0125143, mayo de 2015. doi: 10.1371/journal.pone.0125143 (vid. pág. 14).

[24] J. Juan-Albarracín, E. Fuster-Garcia, A. Pérez-Girbés, F. Aparici-Robles, Á. AlberichBayarri, A. Revert-Ventura, L. Martí-Bonmatí y J. M. García-Gómez, “Glioblastoma: Vascular Habitats Detected at Preoperative Dynamic Susceptibility-weighted Contrastenhanced Perfusion MR Imaging Predict Survival”, Radiology, vol. 287, n.o 3, págs. 944-954, 2018. doi: 10.1148/radiol.2017170845 (vid. pág. 14).

[25] L. Jan, L. Teresa, I. A. J., S. A. W., H. Arend, V. Dirk, S. J. A. K. y V. H. Sabine, “Nosologic imaging of the brain: segmentation and classification using MRI and MRSI”, NMR in Biomedicine, vol. 22, n.o 4, págs. 374-390, 2008. doi: 10.1002/nbm.1347 (vid. pág. 14).

[27] B. H. Menze, K. Van Leemput, D. Lashkari, T. Riklin-Raviv, E. Geremia, E. Alberts, P. Gruber, S. Wegener, M.-A. Weber, G. Szekely, N. Ayache y P. Golland, “A Generative Probabilistic Model and Discriminative Extensions for Brain Lesion Segmentation–With Application to Tumor and Stroke.”, eng, IEEE transactions on medical imaging, vol. 35, n.o 4, págs. 933-946, abr. de 2016, issn: 1558-254X (Electronic). doi: 10.1109/TMI.2015. 2502596 (vid. pág. 14).

[28] K. Kamnitsas, W. Bai, E. Ferrante, S. G. McDonagh, M. Sinclair, N. Pawlowski, M. Rajchl, M. C. H. Lee, B. Kainz, D. Rueckert y B. Glocker, “Ensembles of Multiple Models and Architectures for Robust Brain Tumour Segmentation”, CoRR, vol. abs/1711.0, 2017. arXiv: 1711.01468 (vid. págs. 15, 56).

[29] K. Kamnitsas, E. Ferrante, S. Parisot, C. Ledig, A. Nori, A. Criminisi, D. Rueckert y B. Glocker, DeepMedic for Brain Tumor Segmentation. abr. de 2016, págs. 138-149, isbn: 978-3-319-55523-2. doi: 10.1007/978-3-319-55524-9_14 (vid. pág. 15).

[30] J. Long, E. Shelhamer y T. Darrell, “Fully Convolutional Networks for Semantic Segmentation”, CoRR, vol. abs/1411.4, 2014 (vid. pág. 15).

[31] O. Ronneberger, P. Fischer y T. Brox, “U-Net: Convolutional Networks for Biomedical Image Segmentation”, CoRR, vol. abs/1505.0, 2015 (vid. pág. 15).

[32] G. Wang, W. Li, S. Ourselin y T. Vercauteren, “Automatic Brain Tumor Segmentation using Cascaded Anisotropic Convolutional Neural Networks”, CoRR, vol. abs/1709.0, 2017. arXiv: 1709.00382 (vid. págs. 15, 56).

[33] F. Isensee, P. Kickingereder, W. Wick, M. Bendszus y K. H. Maier-Hein, “Brain Tumor Segmentation and Radiomics Survival Prediction: Contribution to the BRATS 2017 Challenge”, CoRR, vol. abs/1802.1, 2018 (vid. págs. 16, 56).

[34] S. Bakas, H. Akbari, A. Sotiras, M. Bilello, M. Rozycki, J. S. Kirby, J. B. Freymann, K. Farahani y C. Davatzikos, “Advancing The Cancer Genome Atlas glioma MRI collections with expert segmentation labels and radiomic features.”, eng, Scientific data, vol. 4, pág. 170 117, sep. de 2017, issn: 2052-4463 (Electronic). doi: 10.1038/sdata.2017.117 (vid. pág. 17).

[35] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. J. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Józefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. G. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. A. Tucker, V. Vanhoucke, V. Vasudevan, F. B. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu y X. Zheng, “TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems”, CoRR, vol. abs/1603.0, 2016. arXiv: 1603.04467 (vid. pág. 21).

[36] K. He, X. Zhang, S. Ren y J. Sun, “Deep Residual Learning for Image Recognition”, CoRR, vol. abs/1512.0, 2015 (vid. pág. 26).

[37] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever y R. Salakhutdinov, “Dropout: A Simple Way to Prevent Neural Networks from Overfitting”, Journal of Machine Learning Research, vol. 15, págs. 1929-1958, 2014 (vid. pág. 34).

[38] D. P. Kingma y J. Ba, “Adam: A Method for Stochastic Optimization”, CoRR, vol. abs/1412.6, 2014 (vid. pág. 39).

[39] N. J. Tustison, B. B. Avants, P. A. Cook, Y. Zheng, A. Egan, P. A. Yushkevich y J. C. Gee, “N4ITK: Improved N3 Bias Correction”, IEEE transactions on medical imaging, vol. 29, n.o 6, págs. 1310-1320, jun. de 2010, issn: 0278-0062. doi: 10.1109/TMI.2010.2046908 (vid. pág. 41).

[40] Y. Benjamini e Y. Hochberg, “Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing”, Journal of the Royal Statistical Society. Series B (Methodological), vol. 57, n.o 1, págs. 289-300, 1995, issn: 00359246 (vid. pág. 43).

[41] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville e Y. Bengio, “Generative Adversarial Networks”, ArXiv e-prints, jun. de 2014. arXiv: 1406. 2661 [stat.ML] (vid. pág. 60).

APPENEDX

Codes for Image Segmentation

```
classdef Network <handle

    properties
        layers={}
        loss=[]
        loss_prime=[]
    end

    methods
        %% add layer
        function obj = add(obj,layer)
            obj.layers{end+1}= layer;
        end

        %% use method
        function obj = use(obj, loss, loss_prime)
           obj.loss = loss;
           obj.loss_prime = loss_prime;
        end


        %% Predite
        function result = predict(obj, input_data)
            samples = size(input_data,1);
            result =[];

            for i=1:samples
                output = input_data(i,:);
                for j=1:length(obj.layers)
                    layer = obj.layers{j};
                    output =
layer.forward_propagation(output);
                end
                result(i,:) = output;
            end
            return
        end

        %% Training the network
        function history = fit(obj, x_train,
y_train,x_val, y_val, epochs, learning_rate)
            m = size(x_train,1);

            for i= 1: epochs
                err =0;
                for j=1:m
                    output = x_train(j,:);
```

```
                            for k=1:length(obj.layers)
                                layer = obj.layers{k};
                                output =
layer.forward_propagation(output);
                            end

                            % compute loss (for display purpose
only)
                            err = err + obj.loss(y_train(j),
output);

                            % backward propagation
                            error = obj.loss_prime(y_train(j),
output);
                            for k = length(obj.layers):-1:1
                                layer = obj.layers{k};
                                error =
layer.backward_propagation(error, learning_rate);
                            end

                            % calculate average error on all
samples
                            err= err/m;

                        end
                        %% training, val error  'tanh, -1/1'
                        pre_tr_y = obj.predict(x_train);
                        train_accuracy(i) =
sum(sign(pre_tr_y)==y_train)/numel(y_train);

                        pre_val_y = obj.predict(x_val);
                        val_accuracy(i) =
sum(sign(pre_val_y)==y_val)/numel(y_val);

                        printString =
['epoch',num2str(i),'/',num2str(epochs),'--------------
','error:  ',num2str(err), '     train_accuracy:
',num2str(train_accuracy(i)), '     val_accuracy:
',num2str(val_accuracy(i)) ];
                        disp(printString)
                    end
                    history(1,:)= val_accuracy(:);
                    history(2,:)= train_accuracy(:);
                end




            end
        end
```

Codes for the feature extraction

```
      classdef ActLayer <Layer

          properties
              activation
              activation_prime
          end

          methods
              function obj =
ActLayer(activation,activation_prime)
                  obj.activation = activation;
                  obj.activation_prime = activation_prime;
              end

              function output =
forward_propagation(obj,input_data)
                  obj.input = input_data;
                  obj.output = obj.activation(obj.input);
                  output = obj.output;
              end

              function input_error = backward_propagation(obj,
output_error, learning_rate)
                  input_error =
obj.activation_prime(obj.input).*output_error;
              end

          end
      end


      classdef FCLayer < Layer

          properties

              weights
              bias
          end

          methods
              function obj = FCLayer(input_size,output_size)

                  obj.weights = rand(input_size,output_size) -
0.5;
                  obj.bias = rand(1,output_size) -0.5;
                  obj.input = [];
                  obj.output = [];
```

```
            end

            function output =
forward_propagation(obj,input_data)
                obj.input = input_data;
                obj.output = obj.input*obj.weights +obj.bias;
                output = obj.output;

            end

            function input_error =
backward_propagation(obj,output_error, learning_rate)

                input_error = output_error*obj.weights';
                weights_error = obj.input'*output_error;
                % dBias = output_error

                % update parameters
                obj.weights = obj.weights - learning_rate *
weights_error;
                obj.bias = obj.bias - learning_rate *
output_error;

            end
        end
    end
    function varargout = BrainMain(varargin)

    gui_Singleton = 1;
    gui_State = struct('gui_Name',       mfilename, ...
                       'gui_Singleton',  gui_Singleton, ...
                       'gui_OpeningFcn',
@BrainMain_OpeningFcn, ...
                       'gui_OutputFcn',  @BrainMain_OutputFcn,
...
                       'gui_LayoutFcn',  [] , ...
                       'gui_Callback',   []);
    if nargin && ischar(varargin{1})
        gui_State.gui_Callback = str2func(varargin{1});
    end

    if nargout
        [varargout{1:nargout}] = gui_mainfcn(gui_State,
varargin{:});
    else
        gui_mainfcn(gui_State, varargin{:});
    end

    function BrainMain_OpeningFcn(hObject, eventdata, handles,
varargin)
    handles.output = hObject;
```

```
      guidata(hObject, handles);



      function varargout = BrainMain_OutputFcn(hObject,
eventdata, handles)
      varargout{1} = handles.output;


      function pushbutton1_Callback(hObject, eventdata, handles)
      global brainImg
      [filename pathname] =
uigetfile({'*.jpg';'*.tif';'*.gif';'*.png'},'Select An Image');
      brainImg = imread([pathname filename]);
      axes(handles.axes1);
      imshow(brainImg);
      axis off
      [m n c] = size(brainImg);
      if c == 3
          brainImg  = rgb2gray(brainImg);
      end

      % --- Executes on button press in pushbutton2.
      function pushbutton2_Callback(hObject, eventdata, handles)
      % hObject    handle to pushbutton2 (see GCBO)
      % eventdata  reserved - to be defined in a future version
of MATLAB
      % handles    structure with handles and user data (see
GUIDATA)
      global brainImg
      [ brainImg ] = Preprocess( brainImg );
      axes(handles.axes2);
      imshow(brainImg);
      axis off


      % --- Executes on button press in pushbutton3.
      function pushbutton3_Callback(hObject, eventdata, handles)
      % hObject    handle to pushbutton3 (see GCBO)
      % eventdata  reserved - to be defined in a future version
of MATLAB
      % handles    structure with handles and user data (see
GUIDATA)
      global brainImg
      %******************Segmentation**********************
      brainImg = imresize(brainImg,[256 256]);
      noclus = 4;
      data = im2double(brainImg);
      data = data(:);
      %&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
```

```
      tic
      [center,U,obj_fcn] = clusterpixel(data,noclus);
      %&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

      figure(2),
      plot(obj_fcn);
      title('Objective
Function','fontsize',15,'fontname','monotype corsiva');
      xlabel('Iteration','fontsize',15,'fontname','monotype
corsiva');
      ylabel('Cost Function','fontsize',15,'fontname','monotype
corsiva');
      maxU = max(U);
      index1 = find(U(1,:) == maxU);
      index2 = find(U(2,:) == maxU);
      index3 = find(U(3,:) == maxU);
      index4 = find(U(4,:) == maxU);
      fcmImage(1:length(data))=0;
      fcmImage(index1)= 1;
      fcmImage(index2)= 0.8;
      fcmImage(index3)= 0.6;
      fcmImage(index4)= 0.4;
      imagNew = reshape(fcmImage,256,256);
      segtime = toc;
      axes(handles.axes3);
      imshow(imagNew,[]);
       figure(3),
       imshow(imagNew,[]);
       title('Segmented Image');
      colormap(jet)
      axis off
      % impixelinfo;
      % set(handles.text4,'String',segtime);


      % --- Executes on button press in pushbutton4.
      function pushbutton4_Callback(hObject, eventdata, handles)
      % hObject    handle to pushbutton4 (see GCBO)
      % eventdata  reserved - to be defined in a future version
of MATLAB
      % handles    structure with handles and user data (see
GUIDATA)
      global  brainImg TestImgFea imagNew
      GLCM_mat = graycomatrix(imagNew,'Offset',[2 0;0 2]);

           GLCMstruct = Computefea(GLCM_mat,0);

           v1=GLCMstruct.contr(1);

           v2=GLCMstruct.corrm(1);
```

```
          v3=GLCMstruct.cprom(1);

          v4=GLCMstruct.cshad(1);

          v5=GLCMstruct.dissi(1);

          v6=GLCMstruct.energ(1);

          v7=GLCMstruct.entro(1);

          v8=GLCMstruct.homom1(1);

          v9=GLCMstruct.homop(1);

          v10=GLCMstruct.maxpr(1);

          v11=GLCMstruct.sosvh(1);

          v12=GLCMstruct.autoc(1);

          TestImgFea = [v1,v2,v3,v4,v5,v6,v7,v8];
     set(handles.uitable1,'Data',TestImgFea);
     set(handles.uitable1, 'ColumnName', {'Contrast',
'Correlation',....

'Dissimilarity','Energy','Entropy','Homogeneity','Homop','Max.P
rob',.....
             });
         set(handles.uitable1, 'RowName', {'Value'});


     % --- Executes on button press in pushbutton5.

     % --- Executes when figure1 is resized.
     function figure1_SizeChangedFcn(hObject, eventdata,
handles)
     % hObject    handle to figure1 (see GCBO)
     % eventdata  reserved - to be defined in a future version
of MATLAB
     % handles    structure with handles and user data (see
GUIDATA)


     % --- If Enable == 'on', executes on mouse press in 5
pixel border.
     % --- Otherwise, executes on mouse press in 5 pixel border
or over pushbutton1.
     function pushbutton1_ButtonDownFcn(hObject, eventdata,
handles)
     % hObject    handle to pushbutton1 (see GCBO)
```

```
     % eventdata  reserved - to be defined in a future version
of MATLAB
     % handles    structure with handles and user data (see
GUIDATA)
```