UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola d'Enginyeria de Barcelona Est

Bachelor's Thesis

Master's degree in Interdisciplinary and Innovative Engineering

CHARACTERIZATION OF BRAIN DEVELOPMENT IN PRETERM CHILDREN USING ULTRASOUND IMAGES

Author: David Rabanaque Rodríguez
Director: Raul Benitez Iglesias
Co-Director: Christian Mata

Convocation: September 2021

Abstract

The most important period for human's brain development is the fetal phase. During these forty weeks period, important morphological changes take place in the human brain, including a huge increase in the brain surface following the development of sulci and gyri.

In preterm newborns these changes occur in an extrauterine environment, and an impaired brain development has been shown in this population at term equivalent age. A normalized atlas of brain maturation with cerebral ultrasound may allow the clinicians to assess these changes weekly from birth to term equivalent age.

Based on the images of the different babies provided by two clinical researchers, this study proposes a web application implemented with python and its different libraries, including Dash, and accessible through docker that allows direct access to the designed app and its database. In this way, a tool is provided that allows a first definition of the different sulci to be made manually to finally pass them through an algorithm with the aim of improving precision and being able to export both the image and the coordinates obtained from it.

Resumen

El período más importante para el desarrollo del cerebro humano es la fase fetal. Durante este período de cuarenta semanas, se producen importantes cambios morfológicos en el cerebro humano, incluido un gran aumento en la superficie del cerebro a raíz del desarrollo de surcos y circunvoluciones.

En los recién nacidos prematuros, estos cambios se producen en un entorno extrauterino y se ha demostrado un deterioro del desarrollo cerebral en esta población a la edad equivalente a término. Un atlas normalizado de maduración cerebral con ecografía cerebral puede permitir a los médicos evaluar estos cambios semanalmente desde el nacimiento hasta la edad equivalente a término.

A partir de las imágenes de los diferentes bebés proporcionados por dos investigadores clínicos, este estudio propone una aplicación web implementada con Python y sus diferentes bibliotecas, incluida Dash, y accesible a través de Docker que permite el acceso directo a la aplicación diseñada y su base de datos. De esta forma, se proporciona una herramienta que permite realizar una primera definición de las diferentes ranuras de forma manual para finalmente pasarlas por un algoritmo con el objetivo de mejorar la precisión y poder exportar tanto la imagen como las coordenadas obtenidas de la misma.

Resum

El període més important per al desenvolupament del cervell humà és la fase fetal. Durant aquest període de quaranta setmanes, es produeixen canvis morfològics importants al cervell humà, incloent un enorme augment de la superfície cerebral després del desenvolupament dels solcs i circumvolucions.

En els nadons prematurs, aquests canvis es produeixen en un entorn extrauterí i s'ha demostrat un deteriorament del desenvolupament cerebral en aquesta població a una edat equivalent al terme. Un atles normalitzat de maduració cerebral amb ultrasons cerebrals pot permetre als clínics avaluar aquests canvis setmanalment des del naixement fins a una edat equivalent al terme.

Basat en les imatges dels diferents nadons proporcionats per dos investigadors clínics, aquest estudi proposa una aplicació web implementada amb Python i les seves diferents biblioteques, inclòs Dash, i accessible a través de Docker que permet accedir directament a l'aplicació dissenyada i a la seva base de dades. D'aquesta manera, es proporciona una eina que permet fer una primera definició de les diferents ranures manualment per passar-les finalment per un algorisme amb l'objectiu de millorar la precisió i poder exportar tant la imatge com les coordenades que se n'obtenen.

Acknowledgments

First of all, I would like to thank my tutors, Raul Benitez and Christain Mata, who explained to me what new learnings the Interdisciplinary Master and innovative engineering could bring me in my knowledge as an engineer, in addition to offering encouragement during the two years of study and a Master's final project in which I have been guided and they have been contributing their experience in computer vision to achieve the objective of the project.

Also thank Thais Agut and Nuria Carreras for having offered me a project that has given me more knowledge and experience in the Python programming language acquired throughout the master's degree and which has been applied in a real case to a study at Hospital Sant Joan of Déu.

Also thank all the teachers that I have had throughout the master's degree, who have taught me and helped me understand the knowledge, both familiar to the degree studied and outside of it. In addition, I would like to thank my colleagues and friends that I have made over these two years, with whom I have shared good times and gone through some of the difficulties that have arisen.

Finally, I would like to thank my parents and sister, who have encouraged me and helped me to get up in the most complicated moments, both in life and in the master's degree, and that without them it would not have been possible to reach this point in my life.

Index

1. Introduction and Project Framework

This chapter proceeds to explain the origins and collaboration, as well as the motivation and the final objectives to be achieved in order to facilitate the segmentation of the sulci for each of the ultrasounds performed on the premature babies that are part of the base data and project.

In the second part of this section, we proceed to explain all the work necessary to complete the project and everything that is required to complete the work satisfactorily to meet its expectations and success; in addition to listing the objectives to be met at the end of the Final Master's Thesis.

1.1. Project origin and motivation

The project arose at the beginning of the second semester of the first year of the Interdisciplinary and Innovative engineering master's degree, where two clinical researchers from the neonatal unit of Sant Joan de Déu hospital, Thais Agut and Nuria Carreras, contacted the CREB UPC Research Institute, specifically with Raul Benitez and Christian Mata, to propose a related project in the field of medicine and computer vision. It is part of a brain maturation project (PI18-00110).

Figure 1. Logo of the Universitat Politècnica de Catalunya (UPC) and the Sant Joan de Déu hospital.

The origin of this comes from the desire to make an atlas of brain sulci and present it in an article, in which the evolution of premature babies over a period of weeks could be shown and can be visualized in a chain, indicating and bordering the shape of each groove in the different ultrasounds of the baby. Thus, obtaining a groove annotation system for the ultrasounds that they had.

Raúl saw that this project that had been proposed could be a topic for a final master's thesis (TFM), since the student who could contribute the knowledge acquired throughout the master's degree and within the medical specialization to help them create an algorithm that could define the sulci in brain ultrasounds collected from different premature babies during different weeks.

This project poses a difficulty when defining a structure, in this case a groove, in an ultrasound since it is an imaging test that uses sound waves (ultrasound) to create images of organs, tissues and structures

inside the body that does not use radiation, but that provides a problem with a special type of noise called Speckle. This noise, which consists of a granular pattern, causes a significant degradation in image quality, thus increasing the difficulty of discriminating fine details in images during a diagnostic examination.

So, the main motivation of the project has been to find a way, using different techniques, to define the sulci of each plane, discriminating the noise that appears in them, which increases the difficulty of processing images through segmentation for the detection of sulci. In addition, to facilitate the interaction between the doctors and the algorithm, it has been decided to make a Dash application, easy and intuitive to use, so that the results are obtained once the area where the different sulci are located in the ultrasound has been indicated in the corresponding plane.

On a personal level, this project has been a challenge for me, since on the one hand it has allowed me to apply and put into practice a large part of the knowledge acquired in python and computer vision that I have been learning throughout the Interdisciplinary and innovative engineering master in the field of health.

In addition, it has provided me with an additional knowledge and experience when working within multidisciplinary teams, formed in this case by four more people, two of them with computer vision knowledge, my tutors Raul Benitez and Christian Mata, and others two more people with knowledge of paediatrics and health, Thais Agut and Nuria Carreras. In this way, it has allowed me to acquire knowledge of how and how to work with people from different fields and coordinate with them.

So this final master's thesis has been useful to me, since although its main objectives are educational and to demonstrate the knowledge acquired, it has also helped me to understand and understand what it is like to work with people from different fields and the ways in which they You have to adapt one to understand part of your knowledge, in this case of a person's brain, and how to explain to other people the computer work that is being done so that they can see it and comment on the results, to confirm if the work you are on the right track or you have to take another direction.

Finally, I think that it is a work that once completed and presented can be continued for other health cases that are related to ultrasounds (images used) and also for the project for which it has been carried out, since it has only been adapted for detect brain sulci, but new functions could also be added to the application to try to detect other anatomies in the brain, such as parts of the skull, etc; expanding the functions of the developed application.

1.1. Project Scope

This project arises from the need to create an atlas of the evolution that occurs in the brain sulci of a premature babies as the weeks progress. This project is carried out jointly with the Sant Joan de Déu Hospital, who have provided the images of different babies throughout the week and of the different coronal and sagittal planes.

So, the needs we want to cover on our part, is to use the knowledge acquired in computer vision and perform different methods to observe which of them best applies to the images used (ultrasound) to distinguish and define in them the different sulci to be analysed. This process is carried out through a segmentation of these through the processed images, thus obtaining the desired result in order to make an atlas that allows in the future, to know the gestational age in which a baby is and if it is evolving correctly.

The people who are in the hospital do not have knowledge of programming and computer vision so that in case of obtaining more images, a new analysis is made to incorporate this new segmentation in the image base. So, a simple way is that the people who work in the hospital, can do this through a dash application that only by starting, they can access and do a first analysis themselves to later apply the algorithm on the given data and obtain the segmentation for the images to be analysed.

As mentioned in the previous paragraph, the app is based on semi-automatic learning, that is, having a shortage of images since the quality of these (due to the noise that appears in the ultrasounds) is not entirely good from the beginning, it has not been possible to create and learn a neural network to obtain an automatic segmentation, which requires a pre-analysis.

1.2. Project Goals

To comply with the points proposed in the scope defined in section 4.1., It is necessary to define objectives that must be met throughout the project so that it is complete. These points are defined:

- Data pre-processing: To carry out the project it is necessary to have a set of images that allows the analysis and pre-processing of each one of them. This point is important since the images provided by the Sant Joan de Déu Hospital contain a part of information that is not necessary to include in the computer vision part and therefore they will be cut and renamed. In addition, the images in different folders are organized to create a database and thus make it easier to access each of them.
- Computer vision methodology: The part that has spent the most time and effort. In this part, different methodologies have been tested (threshold, edge, median, structures, etc.) to find

the technique or the set of them that allow, through a semi-automatic process, to meet the main objective of the doctors, which consists of segmenting the sulci of the brain.

- App Design: It was decided to carry out this last objective to facilitate the professionals who are in the hospital the way to make and define furrows, thus covering a need they have to carry out their work. So, it was decided to do a graphic interface with a python extension that allows you to convert your scripts to production grade applications for this project. Thus, anyone in the hospital is able to use the algorithm through an application in a simple way and from a personal computer, from which the area to be analysed can be selected and the sulci defined.

2. Theorical Framework

To better understand what you want to look for to meet your goals, it is necessary to understand some concepts in relation to preterm birth. Therefore, below we proceed to explain what they are and their brain maturation, the planes to be analysed, the sulci that are identified in each of them and their evolution over the weeks.

2.1. Brain development in preterm newborns

Preterm is defined as babies born alive before 37 weeks of pregnancy are completed. There are sub-categories of preterm birth, based on gestational age [1]:

- Extremely preterm (less than 28 weeks)
- Very preterm (28 to 32 weeks)
- Moderate to late preterm (32 to 36 weeks).

The reasons of preterm birth are multiple, and fetal, maternal and placental factors interact. Some risk factors for preterm birth include chorioamnionitis (infection), maternal hypertensive disorders, genetic factors, altered nutritional status, multiple gestation, toxic exposure, among others. However, the exact cause of a preterm birth is not always known.

This project, focuses on the evaluation of brain maturation of premature babies through ultrasound scans. These images obtained allow us to observe the brain from two different planes that will be explained later, but first of all, it is necessary to explain what the brain is.

The brain is an organ that fills most of the cranial cavity. It is divided into two hemispheres and four main cerebral lobes can be differentiated in each hemisphere. Brain lobes are located and named as follows: the frontal lobes are found in the frontmost region of the cerebral cortex. The parietal lobes and temporal lobes lie behind the frontal lobes, with the parietal lobes located above the temporal lobes. Finally, the occipital lobes are located in the posterior region of the brain

Each of these are responsible for several important functions in the human being. For example, the frontal lobes are vital for motor control, thinking, and reasoning; The parietal lobes process sensory information, while the occipital lobes are the main centers for visual processing. Finally, the temporal lobes are important for the production of language and speech, as well as for the processing of memory and emotion.

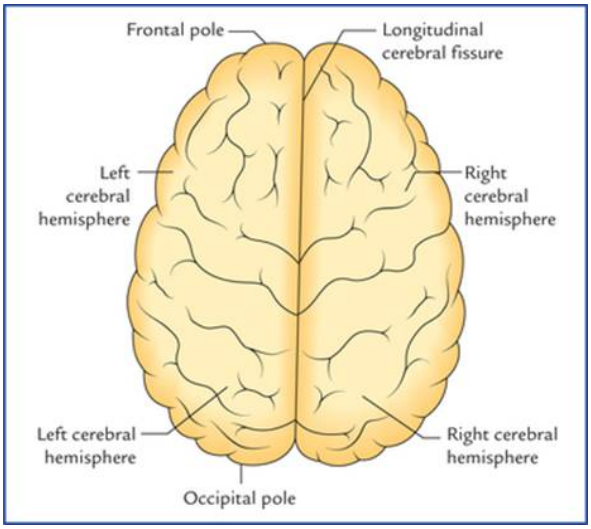

Figure 2.1.1. Superior view of the cerebrum. [2] [4]

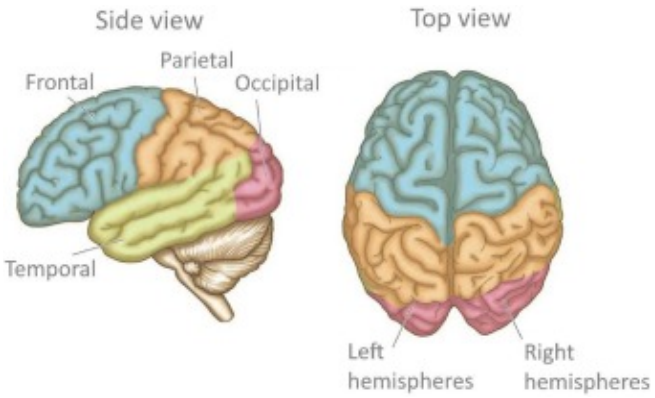

Figure 2.1.2. Lobes and hemispheres of the brain.[7]

The cells of the brain are neurons, and these are grouped within it, forming two different tissues: grey matter and white matter.

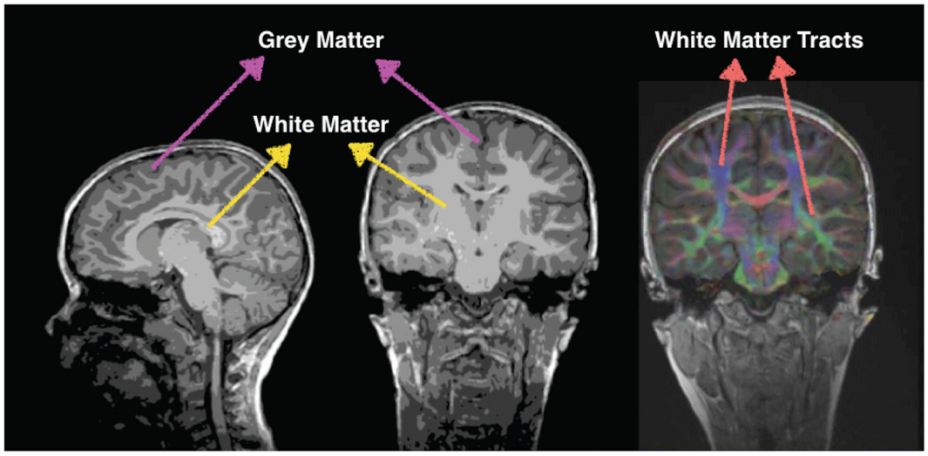

Figure 2.1.3. Child's brain where grey and white matter are shown. [3]

In summary, grey matter is primarily found on the surface of each cerebral hemisphere (known as the cerebral cortex) and in the central structures (known deep nuclei). On the contrary, the white matter is found in the deep parts of the brain, which through the glial cells and myelinated axons that form it, connect the various areas of grey matter.

2.2. Common maturation patterns across babies and evolution in time

When referring to the maturation of the human brain, it is referring to complex changes at specific stages of development. One of the main maturational processes in the human brain is cortical gyration. It is the process whereby the cerebral cortex folds, forming sulci and gyri, in order to increase the cerebral surface with a minor increase in volume. This process take place during the last trimester of gestation and it follows a specific sequence that allow to date a specific brain in base to its gyration status.

Normal brain maturation in fetuses have been described using MRI techniques (figure) as weel as fetal ultrasound (Pistorius 2010. Grade and symmetry of normal fetal cortical development: a longitudinal two- and three-dimensional ultrasound study.) Cerebral ultrasound is the most used neuroimaging technique in the neonatal units worldwide, as it may be performed at the bedside, even in instable patients, and may be repeated a long time, in a cost-effective manner. Cerebral sulci can be easily depicted with ultrasound as they appear as hyperechogenic (white) lines following the cerebral surface.

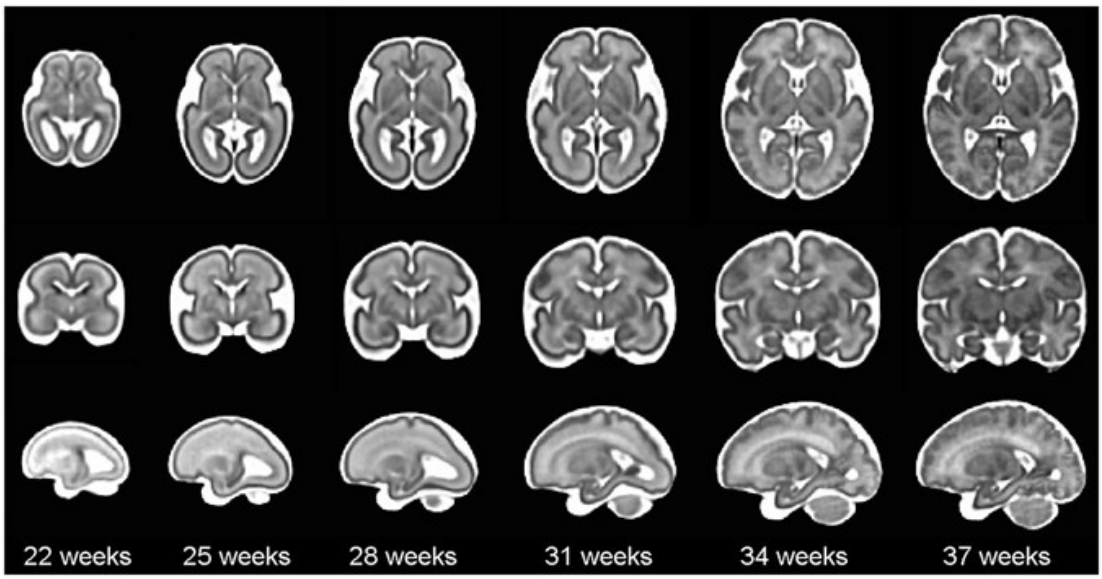

Figure 2.2.1. The spatio-temporal fetal brain magnetic resonance atlas (CRL fetal brain atlas) at six representative gestational ages: 22, 25, 28, 31, 34 and 37 weeks. Axial, coronal, and sagittal views of the atlas have been shown for each age point. [5]

1.2. Planes and sulci of the Brain

For a better understanding of what is being analysed, we proceed to explain the different anatomical planes obtained from the ultrasounds carried out in Sant Joan de Déu and which have been saved in a database to be used later.

First of all, it must be borne in mind that both the relative position and the direction of brain structures are described with special words used in the medical world. For example, the frontal lobe can be said to be "rostral" to the occipital lobe.

Like the human body, the brain is a three-dimensional structure and therefore, any area of the brain can be located within the three planes formed thanks to three axes:

- r plane, formed by the y and z axes.
- q plane, formed by the x and y axes.
- p plane, formed by the x and z axes.

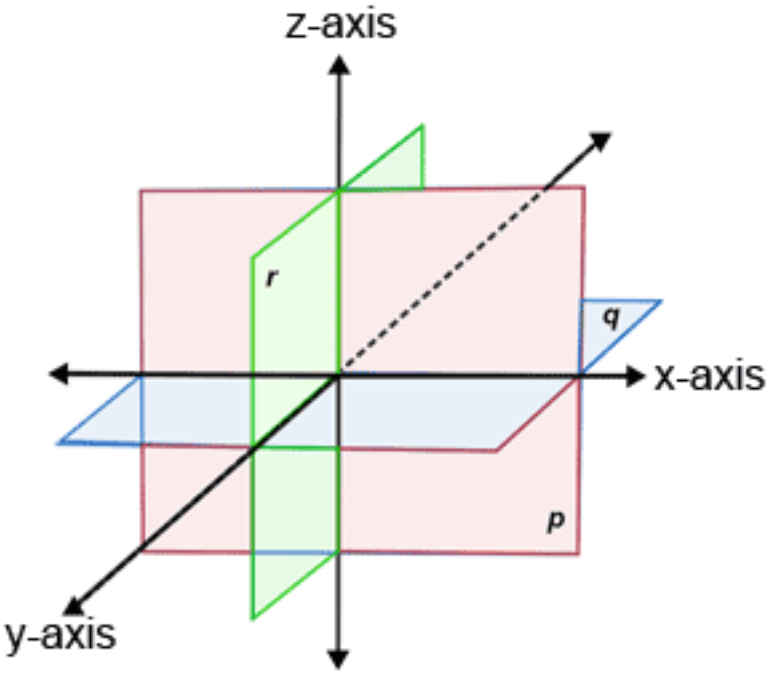

Figure 2.3.1. Axis and planes of the three dimensions.

If the ears are placed on the y-axis and the nose remains on the x-axis, the brain can be cut in three different planes called the coronal or frontal plane (r-plane), horizontal plane (q-plane) and sagittal (p- flat). In figure 2.3.1., the three planes are shown in the same order as defined:

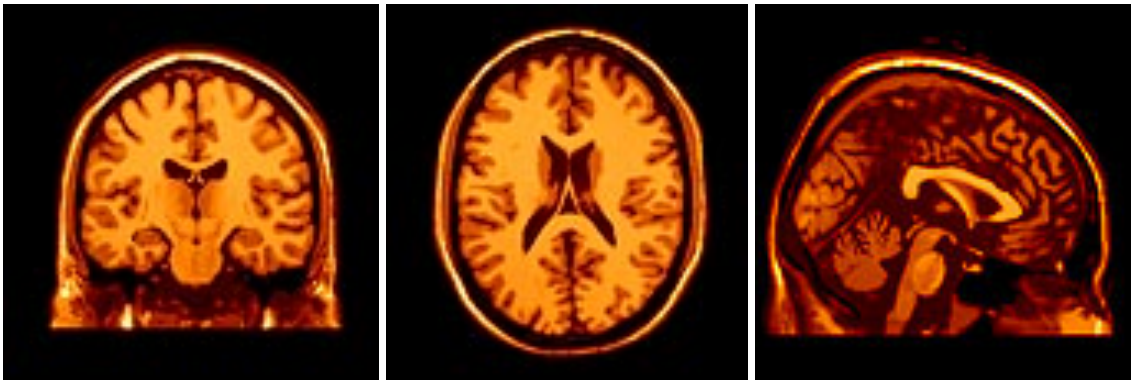

Figure 2.3.1. Planes of the brain are called from left to right: coronal or frontal plane, horizontal and sagittal plane.[6]

These plans provide information on both the internal and external structure of the brain explained in the previous two sections (internal and external structure).

Regarding the external part or surface of the brain, a very convoluted appearance is observed, formed of sulci (sulci or depressions) and gyri (ridges or elevations).

This project does not carry out an analysis of all the existing sulci of the brain, but rather a small analysis of a certain type of them will be carried out to observe and carry out the computer vision processes. These sulci are known as main cerebral sulci, whose shape and position are fairly constant in all cases, and they are as follows:

- Lateral groove (of Sylvius): This groove is the most conspicuous of all the brain sulci and has a stem and three branches. It appears in the lateral surface of the brain and separates the frontoparietal lobes superiorly, from the temporal lobe inferiorly.
- Central sulcus (of Rolando): This sulcus begins by cutting the superomedial edge of the hemisphere, runs sinuously from bottom to front at an angle of 70 ° and ends just above the posterior branch of the lateral sulcus.
- Parieto-occipital groove is present on the medial surface of the hemisphere. It dived the parietal lobe superiorly, from the occipital lobe inferiorly.
- Calcarine sulcus: This groove is present on the medial surface of the cerebral hemisphere. It traverses the occipital lobe.

It should be noted that there are more sulci on the surface of the brain apart from those mentioned, as can be seen in the figure 2.3.2.; as in the medial surface of the cerebral hemisphere, as is the case of Cingulate sulcus or Parieto-occipital sulcus, among others.

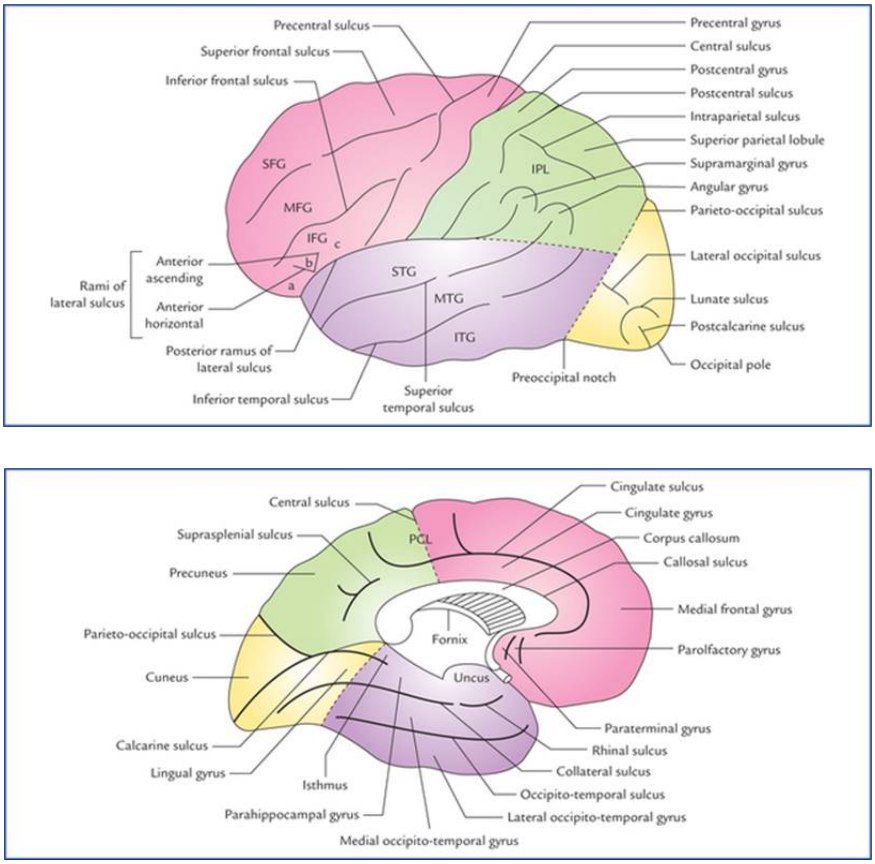

Figure 2.3.2. Sulci on the surface of a person's developed brain. [4]

3. State of the Art

Based on the theoretical framework explained in the previous section, we now proceed to explore and carry out a literary review to address recent works related to this field. For this reason, in this section of the state of the art, the published works are classified and ordered chronologically to check whether or not there really is an application that performs an identical or similar task to that of the present project.

This collection of data and information is related to the part of ultrasound and the different segmentation methods that are carried out on this type of image to show which points of the documents share and/or differ with respect to the objective of this work.

3.1. Literature review

In this section of the state of the art, we proceed to give a brief explanation of other projects that share certain similarities with this project. For this, different papers from previous years have been searched and the authors will be indicated, the year in which it was published and when it was carried out.

To explain chronologically, we begin to explain older projects and advances until we reach the most current ones. So, we will begin by explaining the work carried out by Judith G. Thomas and Richard Alan Peters, in 1991, in which a method that is designed to obtain an automatic measurement of the length of the femur through fetal ultrasound images is presented, in order to estimate gestation age through a comparison with a typical growth chart. In this case, as in the previous ones, real-time ultrasound scans have also been used and have been used for sonographers to indicate the end points of the femur using a device similar to a mouse. It should be noted that the measures, as indicated in the document, have been subjective and inconsistent.

The learning applied in this case has an automatic focus, taking advantage of the a priori knowledge of the general range of femoral size and shape through the use of morphological operators, which process images based on the characteristics of the shape. Thus, morphological operators have been used, on the one hand, to eliminate the noise that appears in the image caused by the probe and to refine the shape of the femur, eliminating artifacts. Once this is done, a single pixel wide femur skeleton is created, thus joining the end points of the femur and finally proceeding with the calculation of the distance between those endpoints.

As seen, machine learning in ultrasound has been carried out in the early 90's. If you go a little further in time, you can find more articles. An example of this is the article written by S. Poonguzhali and G.

Ravindran, in 2006, in which it is proposed to use ultrasound to visualize the structures of the internal organs of the human body. In some cases, noise makes it difficult to identify the boundaries of abnormal regions in the image, and segmentation is used for these. In this case, manual methods are not used, which require a lot of attention from the sonographer and where little precision is usually produced and whose process is very time consuming; Instead, automatic targeting is used to help the physician identify a precise location of abnormal regions.

The segmentation process described is a fully automated technique for massive segmentation in ultrasound images using the region growth technique. This method is unaffected by speckled noise and preserves spatial information. To work properly, the seeding point has been automatically selected from the abnormal region based on the characteristics of the texture, such as the co-occurrence characteristics and the run duration characteristics. Once this is done, we proceed with the threshold to control the growth process of the region, it is automatically selected and the algorithm for the growth of the region is adopted based on the magnitude of the gradient.

This method has been used in different abdominal masses, obtaining as a result that the proposed region growth algorithm is efficient in the selection of the planting point and in the segmentation of the region of interest without manual intervention.

In 2015, Rosa-María Menchón-Lara and José-Luis Sancho-Gómez proposed a fully automatic ultrasound segmentation method applied to atherosclerosis, which is responsible for a large proportion of cardiovascular diseases (CVD). The purpose of this is to make an early diagnosis of this disease to prevent patients from suffering more serious pathologies (heart attacks and strokes) by evaluating the intima-media thickness (IMT) of the common carotid artery (ACC) in ultrasound images in B mode. This type of imaging is considered the most useful tool for the investigation of preclinical atherosclerosis and is usually measured manually by radiologists.

This article proposes an automatic segmentation technique that is based on machine learning and statistical pattern recognition. To achieve this, pixels are classified using artificial neural networks to identify IMT boundaries. As a classification strategy, the concepts of Auto-Encoders (AE) and Deep Learning have been used. This segmentation pattern is tested on a set of 55 longitudinal ultrasound images of the CCA by comparing automatic segmentation with four manual tracings.

Another project that shares characteristics with this is the one carried out by Jen-wei Kuo, Jonathan Mamou, Orlando Aristizábal, Xuan Zhao, Jeffrey A. Ketterling and Yao Wang, a group of researchers who in 2016 proposed a completely automatic segmentation method, called “Nested graphic slice to segment images (2D or 3D)”, which contains several objects with a nested structure. Compared to other chart-based methods developed for multiple regions, this one can work well for nested objects without the need for manual seed selection or initial regions. Even for those cases where different

objects have similar intensity distributions and some of the object's boundaries are missing within magnetic resonance imaging (MRI) and high frequency ultrasound (HFU) they have become the most common imaging modalities for in vivo imaging. of the developing mouse embryo.

The tests performed with this method showed promising results in the separation of the ventricles of the brain, the head and the region of the uterus in the images obtained by high frequency ultrasound.

In 2017, Fausto Milletari, Seyed-Ahmad Ahmadi, Christine Kroll, Annika Plate, Verena Rozanski, Juliana Maiostre, Johannes Levin, Olaf Dietrich, Birgit Ertl-Wagner, Kai Bötzel and Nassir Navab, presented their project in which a focus is made on the segmentation process taking advantage of the abstraction capabilities of convolutional neural networks (CNN). In this case, it is based on the Hough voting method, which employs a strategy that automatically locates and segments the anatomies of interest. This approach not only uses CNN ranking results, but also implements voting by exploiting the characteristics produced by the deepest part of the network.

One of the main characteristics that the work demonstrates is that the learning obtained in the neural network is robust, multiregional, flexible and easily adapts to different modalities. Large and different amounts of data and different dimensionalities (2D, 2.5D and 3D) have been used to train them, in addition to important computational resources.

In the same year, another was also published in which a segmentation is carried out to obtain a personalized muscle diagnosis (neck pain / back injury, work disorders, myopathies, neuropathies) based on the wide availability of ultrasound.

In the paper, presented by Ryan J. Cunningham, Peter J. Harding, and Ian D. Loram, automated muscle segmentation methods are presented that would allow clinicians to study, target, and monitor deep cervical muscles using ultrasound. This process consists of segmenting five bilateral cervical muscles and the spine only by ultrasound, in real time. In the case of labelling the muscle segments, a new multimodal registration method was developed, which consists of the annotation of magnetic resonance images and the registration of shapes in ultrasound images matched by magnetic resonance imaging, approximating the deformation of the tissue. Polynomial regression is then applied to transform the annotations and textures into a half space, before using shape statistics to generate a texture dictionary. For the segmentation part, the test images were compared to the dictionary textures giving an initial segmentation, and then a custom model was actively used to refine the fit. Using ultrasound alone, in invisible participants, our technique currently segments a single image at at0.45s with an accuracy of better than 86% (Jaccard index). That is, in this work an approach that is generally applicable to segment, extrapolate and visualize the deep muscle structure and analyse statistical characteristics in real time has been proposed.

During 2019, other projects were also presented with certain characteristics equal to the present project. One of them was carried out by the authors Yuan Xua, Yuxin Wanga, Jie Yuana, Qian Chengb, Xueding Wangb and Paul L. Carsonc; based on a segmentation performed by machine learning. The other, whose investigators are Xin Yang, Lequan Yu, Shengli Li, Huaxuan Wen, Dandan Luo, Cheng Bian, Jing Qin, Dong Ni and Pheng-Ann Heng, based their work on automated segmentation in volumetric ultrasound.

In the first one, it explains a new method of breast analysis using ultrasound in order to make a segmentation in these images that helps them to discriminate the different tissues to obtain help in locating tumours in that area. This segmentation is based on a convolutional neural network (CNN), trained to be able to learn functions capable of defining discriminatory characteristics, thus obtaining encouraging results where, in many cases, they exceed the predefined and hand-made sets of functions.

However, manual semantic analysis of breast ultrasound images is necessary, which depends on a theoretical basis and the clinical experience of radiologists, resulting in a subjective and variable interpretation according to the observer. This is labour-intensive and time-consuming for large-scale clinical ultrasound, where the properties of the tissues must also be taken into account and consequently the effects it produces on the propagation of the wave.

In the case of the second one, the images used are volumetric ultrasound, which is emerging as a viable imaging modality for routine prenatal examinations as a method of accurate monitoring of maternal and fetal health. However, poor image quality, low contrast, boundary ambiguity, and complex anatomical shapes conspire toward a major lack of efficient tools for segmentation. It makes 3D ultrasound difficult to interpret and makes it difficult to generalize 3D ultrasound in obstetrics.

The objective presented is to analyse the problem of semantic segmentation in prenatal ultrasound volumes proposing three objectives: The first of them, a fully automatic framework is proposed to simultaneously segment multiple anatomical structures of intense clinical interest, including the fetus, the gestational sac and the placenta, which produces an arduous and little studied challenge. For the second, a composite architecture for dense labelling is proposed, in which a custom fully convolutional three-dimensional network explores the concurrency of spatial intensity for initial labelling, while a multidirectional recurrent neural network (RNN) encodes spatially sequentially for the second. ambiguity of combat limits. for significant refinement. Finally, a hierarchical deep monitoring mechanism is introduced to drive the flow of information within an RNN and adjust the latent sequence hierarchy on fine scales and further improve the segmentation results. All of this is extensively verified on large internal data sets, this method illustrates very high segmentation performance, thus promising a breakthrough in prenatal ultrasound exams.

Finally, reference is made to a work presented in 2020 by Jeya Maria Jose Valanarasu, Rajeev Yasarla, Puyang Wang, Ilker Hacihaliloglu and Vishal M. Patel; who proposed an application that uses images similar to those used in the present project, but in this case an automatic segmentation of the ultrasound anatomical landmarks is carried out, since these points play an important role in the treatment of premature babies of very low weight due to increased risk of developing intraventricular haemorrhage (IVH) or other complications.

As the availability of annotated data is limited for the development of an automatic segmentation method for this task, an image synthesis method is proposed that uses a multi-scale self-attention generator to synthesize images of various segmentation masks. Thus, demonstrating that the method can synthesize high quality images for each manipulated segmentation tag with qualitative and quantitative improvements over the most recent synthesis methods.

For the segmentation task, a novel method is proposed, called Confidence Guided Brain Anatomy Segmentation Network (CBAS), which performs segmentation and confidence maps using estimates at different scales. In addition, another technique is presented, which guides CBAS to learn the weights based on the confidence measure in the estimate. The experiments carried out with these methods demonstrate that the proposed method for the synthesis and segmentation tasks achieves significant improvements, in particular, we show that the new synthesis framework can be used to generate realistic images that can be used to improve the performance of a segmentation algorithm.

3.2. Table of recent publications

After the literary review that has been previously explained, a chronological table has been created showing the works published during the last decade (Table 1). This table is ordered by year, authors, explain the characteristics and limitations of each of the works.

Title	Authors	Year of publication	Features	Limitations
Automatic Segmentation of Ultrasound Images Using Morphological Operators [8]	Judith G. Thomas and Richard Alan Peters	1991	Ultrasounds. Artifact Removal (Noise).	Method used to obtain a measurement between two end points. Machine learning.
A Complete Automatic Region Growing Method for Segmentation of Masses on Ultrasound Images [9]	S. Poonguzhali, G.Ravindran	2006	Ultrasounds. Segmentation.	Automatic selection. A region growth algorithm based on the magnitude of the gradient is used.
Fully automatic segmentation of ultrasound common carotid artery images based on machine learning [10]	Rosa-MaríaMenchón-Lara n et al.	2015	Ultrasounds. Segmentation.	Application intended for atherosclerosis. Automatic. Neural networks to define the limits.

Table 1.3.1 Publications and Features

Nested Graph Cut for Automatic Segmentation of High-Frequency Ultrasound Images of the Mouse Embryo [11]	Kuo et al.	2016	Ultrasounds. Segmentation.	Cerebral ventricles. Machine learning. No need for manual selection of seed or starting regions.
Hough-CNN: Deep learning for segmentation of deep brain regions in MRI and ultrasound [12]	Milletari et al.	2017	Segmentation.	Convolutional Neural Network (CNN), which analyses and segments automatically.
Real-Time Ultrasound Segmentation, Analysis and Visualisation of Deep Cervical Muscle Structure [13]	Ryan J. Cunningham, Peter J. Harding and Ian D. Loram	2017	Ultrasounds. Segmentation.	Muscular zone. Automated methods.
Medical breast ultrasound image segmentation by machine learning [14]	Yuan Xua et al.	2019	Ultrasounds. Segmentation.	Mammary tissues analysed area. Convolutional Neural Network (CNN) to segment images (machine learning)

Table 1.3.2 Publications and Features

Towards Automated Semantic Segmentation in Prenatal Volumetric Ultrasound [15]	Xin Yang et al.	2019	Ultrasounds. Segmentation. Prenatal Babies Images.	3D images. Automatic. Multidirectional Recurrent Neural Network (RNN). Supervision.
Learning to Segment Brain Anatomy From 2D Ultrasound with Less Data [16]	Jeya Maria et al.	2020	Ultrasounds. Segmentation. Limited number of images.	Automatic Employs an image synthesis method. Training to improve the segmentation of the Confidence Guided Brain Anatomy Segmentation Network (CBAS).

Table 1.3.3 Publications and Features

Once each of the different works has been described and explained and the points where it resembles and differs from the project presented in this final master's work have been pointed out, we proceed to explain whether, through the search that has been carried out at this point Whether or not there is a project that meets the objectives defined in section 1.3.

Thus, once the characteristics of each of them have been identified, it can be concluded that there is no method and/or application that performs a segmentation of the different brain sulci, taking into account the limited number of ultrasounds, compared to other studies analysed, of each one of the cuts to be analysed during the different weeks for each of the babies. This causes a problem when using a method such as machine learning for groove segmentation, since to obtain good results with this process, a greater number of images are required for each shot and week available.

4. Materials and Methods

4.1. Experimental data

For the completion of the final master's work, the paediatric section of the Sant Joan de Déu Hospital has provided ultrasound scans of different babies, approximately 70. The different images of each baby consisted of ultrasounds performed in different planes, both coronal (c1, c2 , c3, c4 and c5) as sagittal (s1, s2l, s2r, s3l, s3r, s4l and s4r) for a range of 8 weeks (between 24 and 32). The weeks could vary since it depended on the consent of the parents and in some cases, it was accepted to obtain images every week in which the baby was hospitalized and in other cases, only some of them.

As it has not been possible to capture for each baby all the images of each week and plane, the number available is lower than expected.

4.2. Software, programming language and libraries

In this section we proceed to explain the programs when writing code, the programming languages on which the project is based and the libraries used throughout the project and that provide the programmer with complex functions already defined and that can be used in the defined code.

4.2.1. Software

They are used to program the algorithms and functions through lines of code, in addition to allowing you to see the results once they are executed. Throughout the project, two different programs have been used, Google Colab and Jupyter, since both being for the same language, each one offers its advantages:

- Google Colab: It has been used to test and test the different methods applied to the images since the capacity of these, depending on which operations, occupied too much and could not be worked locally. So, they worked remotely using Google servers and PCs.
- Jupyter: It has been used to develop and test the application made for doctors once its design and functionalities have been defined. In this way, it is possible to know if both the application and the designed algorithm could work from a local computer.

4.2.2. Programming Language

The programming language used throughout the project for the different techniques and designed functions, neural network and application, has been Python. This is a multiparadigm programming language capable of partially supporting object orientation, imperative programming and, to a lesser extent, functional programming. In order for it to be used in any interface and thus obtain the different results of the methods, it is necessary to import and install libraries.

4.2.3. Libraries

As mentioned in point "5.2.2. Programming Language", libraries or programming libraries, are a set of functional implementations, coded in a programming language, which offers a well-defined interface for the functionality that is invoked within it. To carry out and complete this project, several of these developed libraries have been used, which are shown and defined, indicating the field to which it belongs within Python, by means of the table presented below:

Library	Definition and functions
matplotlib	Library used to generate graphs from data contained in lists or arrays in the Python programming language and its mathematical extension NumPy.
scikit-image	This library is a collection of algorithms for image processing
OpenCV	Free computer vision library originally developed by Intel.
NumPy	It is a library for the Python programming language that supports creating large multidimensional arrays and vectors, along with a large collection of high-level mathematical functions to operate on them.
PIL	Adds support for opening, manipulating, and saving many different images file formats.

pandas	It is a software library written as a NumPy extension for data manipulation and analysis for the Python programming language. In particular, it offers data structures and operations to manipulate number tables and time series.
keras	Open-Source Neural Networks library written in Python, specially designed to allow experimentation in a more or less short time with Deep Learning networks.
Scikit-learn	Free software machine learning library for the Python programming language. It includes several algorithms for classification, regression and group analysis, including support vector machines, random forests, Gradient boosting, K-means and DBSCAN.
scipy	Library that contains modules for optimization, linear algebra, integration, interpolation, special functions, FFT, signal and image processing, resolution of ODEs and other tasks for science and engineering.
Dash	It is a productive Python framework for creating web analytic applications that allows a user without programming knowledge to perform tasks already defined through interaction with it through buttons or sliders.

Table 4.2.3.1. Python libraries used

5. Image processing workflow

This section explains the different methods analysed to see and compare which of them helps to define and differentiate the sulci of the noise that accompanies the image. In addition to a function, which contains Python, that allows you to identify and save properties of structures that appear in a black and white image. Another point that will be seen is the application of the slic function to define the contour of the groove and finally we will proceed to explain Dash and docker, which allows you to create a user interface so that you can interact with the algorithm easily through a container. The diagram that follows shows in a general way the different blocks covered in the project.

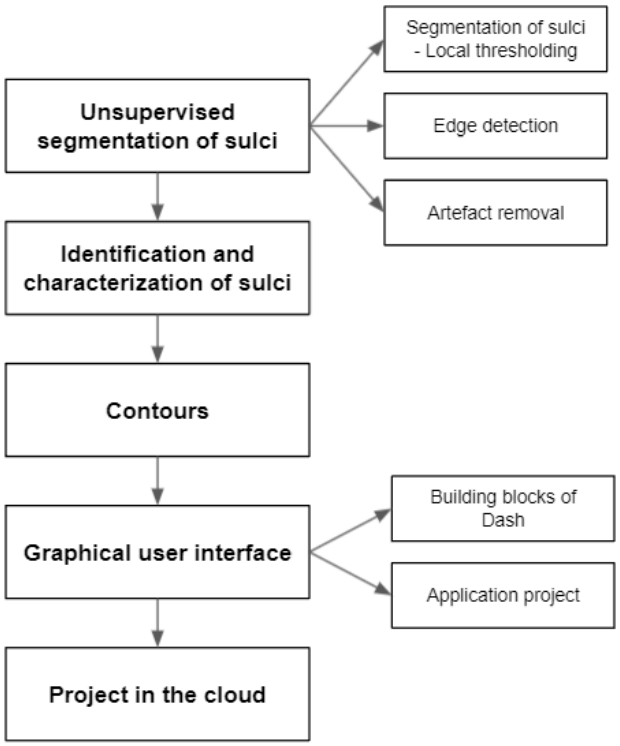

Figure 5.1. Block Diagram of Procedure Performed

Before beginning with the explanation of the methods used, there will be a brief introduction of what computer vision is and how it works.

5.1. Computer vision

Computer vision is an interdisciplinary scientific field whose objective is applied to how computers are able to make a high-level understanding through the digital images or videos that are provided to them. If it is done from an engineering perspective, what is desired is to understand and automate tasks that, already in itself, the human visual system can perform to acquire, process, analyse and understand digital images in order to extract from high dimensions with the aim of producing numerical or symbolic information.

Currently, computer vision is on the rise since it is a field that has been able to surpass humans in some tasks related to the detection and labelling of objects, driving its growth due to the large amount of information that is generated and that is used to train and improve it. All this is possible thanks to the great computing power available to analyse the information that is entered and the algorithms that have been improved or created to achieve a higher precision rate, which makes them have a greater reaction. to visual inputs compared to humans.

The open question that arises is, how exactly do our brains work and how can we approach that with our own algorithms? There is no clear answer to this question, as there are very few complete and functional theories of how the brain and eyes process images. So even if an assumption is made that neural networks "mimic the way the brain works," no one knows whether this is true or not.

What is known is that computer vision has to do with pattern recognition. In this way, computers are trained to understand visual data by feeding it with a large number of images and, if possible, labelled, with the aim of subjecting them to various software techniques or algorithms that can find patterns of all the elements that are used. relate to those labels.

5.2. Unsupervised segmentation of sulci

In a supervised system there are example inputs and desired outputs given by a "master", whose objective is to originate a learning of a general rule that maps the inputs to the outputs. For this project, on the other hand, an unsupervised system is carried out, which does not have data and images that allow defining outputs that depend on the input, since for this to work correctly it would be necessary a large number of images, which are not has, and therefore the direction has been taken to create an unsupervised system.

In this case, which is an unsupervised system, it has been necessary for a person with knowledge to be able to indicate the areas in which the sulci to be segmented are located so that the system can apply

the defined method and perform more precisely the segmentation of the desired structure. That is, the input already has defined regions in which these are found, thus being pre-defined, before passing it through the defined algorithm which provides the different segmentations as output according to the number of regions that have been defined.

Having explained what type of system is used, it is necessary to find a method that provides adequate results for the interests of the doctors, for whom the final application has been designed. So next, we proceed to explain the two methods used in the images to obtain a clearer definition of the sulci and facilitate the segmentation function used. These methods are defined as intensity and edge and both are used in grayscale images, dividing and separating structures according to the tonality of the analysed pixel with those around it. [17]

5.2.1. Segmentation of sulci

In a grayscale image, where black and white are the extremes, it is understood that in the image there is greater intensity when the pixel value is closer to 255. This value comes from the fact that each pixel is based on a value 8-bit binary, which if the conversion is made from binary to decimal, can only take values between 0 and 255.

In this section we proceed to explain what the intensity method consists of in order to improve and facilitate the detection of structures, taking advantage of the grey scale and that because the parts of the brain, such as bones and furrows, have a more intense tone and close to white and that on the contrary, both the brain tissue and the brain fluid are dark, allowing this to be a good option and obtain good results. To highlight and define the structures in the image above the rest, it is necessary to use a function that delimits these surfaces, and for this case the threshold has been used.

The threshold is a technique which consists of assigning one value or another of pixels according to the result obtained from the comparison of the value of the pixel according to the defined threshold. In the case where the value of the established threshold is greater than the analysed pixel, this is established as a value of 0; otherwise, if it is lower, the maximum value is set, which is generally 255. The thresholds are used in the segmentation technique, which are used to separate objects (considered as foreground) from their background. A threshold is a value that has two regions on each side, that is, below the threshold or above the threshold.

In the field of Computer Vision, this technique is done a lot in grayscale images and therefore, in the case that colour images are analysed, a conversion is necessary. This is done since it is easier to

determine and interpret a threshold in a single dimension, the grayscale, than in a colour image in which there are 3 dimensions (RGB).

The library used for this case is the scikit-image, which contains a large number of filters, such as the threshold mean, which returns a threshold value based on the average of the grey scale values of the image.; Otsu that the threshold value is based on the Otsu method; local threshold in which the threshold mask based on the local neighbourhood of pixels is calculated; or others like triangle, yen, etc.

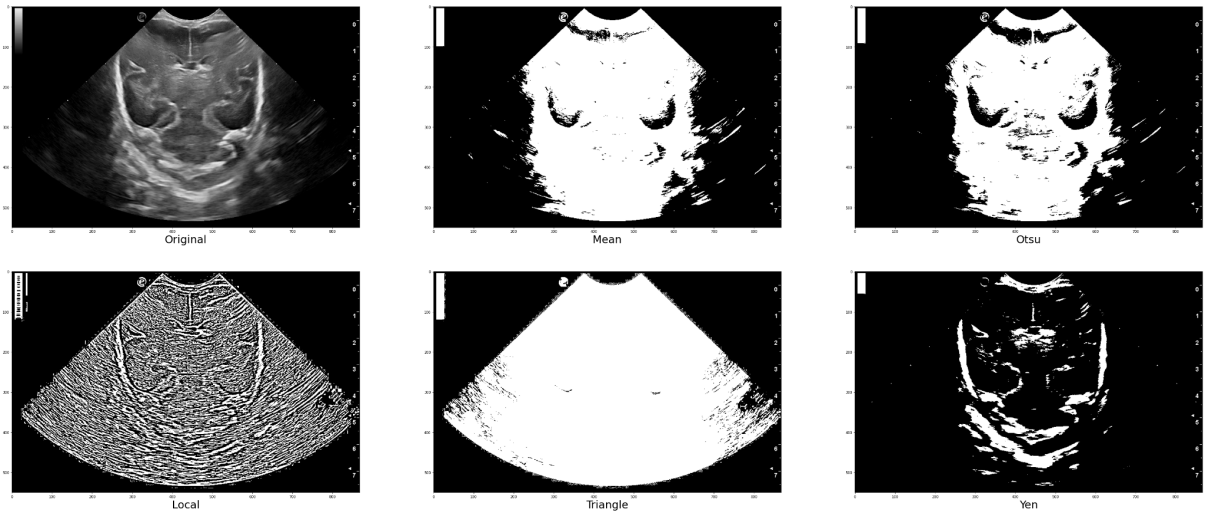

Figure 5.2.1.1. Filters applied to an ultrasound (Original). As an example, the figure shows, from left to right, the result obtained from the Mean, Otsu, Local, Triangle and Yen filters. [18]

As can be seen, many of the thresholds selected for the pixel to take as a value of 0 or 255 are not adequate. This is due to the main precision problem that occurs in ultrasound and is the noise called Speckle. This causes the threshold value to differ and therefore produces that in the areas where there is noise, they are considered with a value of 255 (white) when in reality, this value should only have this value would be the skull bones and the undulations like the furrows. So, with this drawback in the images, it can be concluded that in the field of computer vision this is a degree of difficulty when defining the sulci that appear.

It is worth mentioning that of the thresholds that have been tested in intensities, the one that has given the best result is the case of the local threshold. As has been observed in figure x, if it is observed that there are areas where, if attention is paid, parts of the surface are observed where the sulci have been defined, but these are surrounded by other small segments that have been created due to image noise. This function is performed on a pixel and takes into account those around it. This means that you can select the number of pixels to take into account when choosing how large the surface to be analysed

is to choose the threshold value. This method is suitable and produces better results in those images where there is a large variation in intensity in the background, but as a negative point, it is a slow process since it requires calculating the threshold value by means of a function (mean, standard deviation, etc.) for each pixel of the image, taking into account the number of neighbourhood pixels that have been chosen.

The function used in python to perform most of the filters, including the local Threshold, belongs to the skimage.filters library and is the following:

$$threshold\_local(image, block\_size, method, param);$$

Where the first parameter corresponds to the image to which the filter is applied, the second one, block_size, corresponds to the odd size of the pixel neighbourhood that is used to calculate the threshold value; the third value to introduce is used to determine the method used and finally param, which is used to specify sigma for the "Gaussian" method or the function object for the "generic" method, taking the flat matrix of the local neighbourhood as a single argument and returning the calculated threshold for the center pixel.

The figure that follows shows different images in which a local threshold of different areas has been made, in which it can be seen how the areas with greater and intensity differ and are separated from other areas where there is not or there is no groove or bone. In this example, the neighbourhood pixel area that gives the best solution is 27x27 pixels.

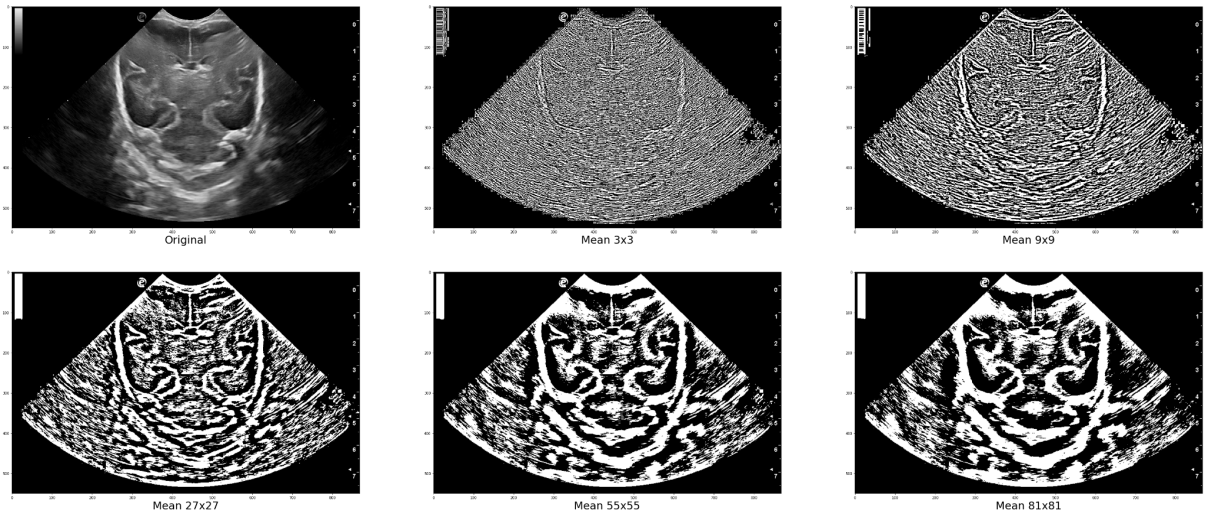

Figure 5.2.1.2. Local Filter applied to an ultrasound (Original) with different sizes of surface to analyse. As an example, the figure shows, from left to right, the result obtained from the Mean 3x3, 9x9, 27x27, 55x55 and 81x81 pixels.

It should be noted that when a neighbourhood pixel value to be analysed is chosen, as the pixel being analysed is in the center and since there are the same number of rows and columns on one side and on the other, it is necessary that this value is odd, otherwise you get an error.

As mentioned above, different functions can be applied to this filter. If it is taken into account that they are binary images with values 1 and 0, where the first of this corresponds to white and the second to black, it is understood that logical operations can be applied to it. That is, two local thresholds are performed where two different functions are used, for example mean and standard deviation, thus obtaining a pixel value for each of them and finally applying the truth tables according to the chosen fundamental binary operation.

$$Image \ = \ (Local\ Threshold\ 1)\ \&\ (Local\ Threshold\ 2);$$

$$Image \ = \ (Local\ Threshold\ 1)\ |\ (Local\ Threshold\ 2);$$

The following figure shows three images, of which two of them contain the results obtained after applying a logical operation to two local thresholds, of which one of them has a mean function applied and the other a standard deviation. Of these two Boolean operations that are shown, the one that provides the best result is the AND, since in it you can see the structures in addition to having a small separation between them and the noise present in the image. In the case of OR, the segments to be analysed are totally lost and therefore it is not an option to take into account since no possible result can be obtained.

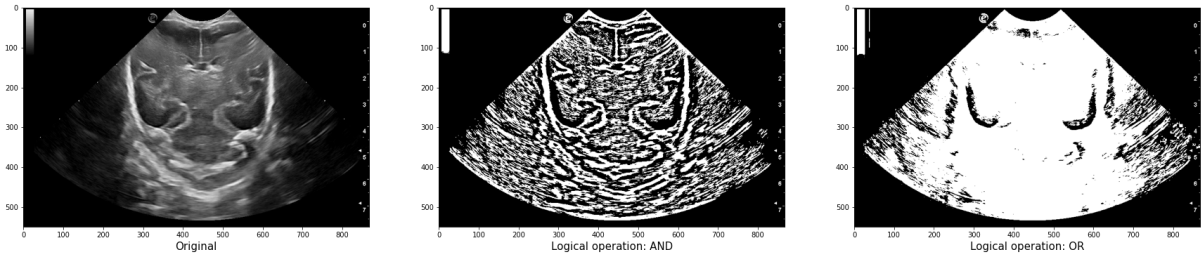

Figure 5.2.1.3. Logical operation applied two images resulting from the local ultrasound threshold (Original) using "AND" and "OR" as operator.

An easy way to see if the result obtained corresponds, is by superimposing both images and comparing it with the original, to see if it matches and the pixels that belong to a structure have a value close to 255. In the figure 5.2.1.4., the noise is still reflected as irregular and small structures, but instead the structures are highlighted and well defined, having regular and continuous contours. Although the objective of this method was to improve the visibility of the structures and it can be concluded that the objective of having well-defined sulci has not really been achieved, it is true that other functions can be used that together with this method, allows eliminating much of it and define the regions of interest.

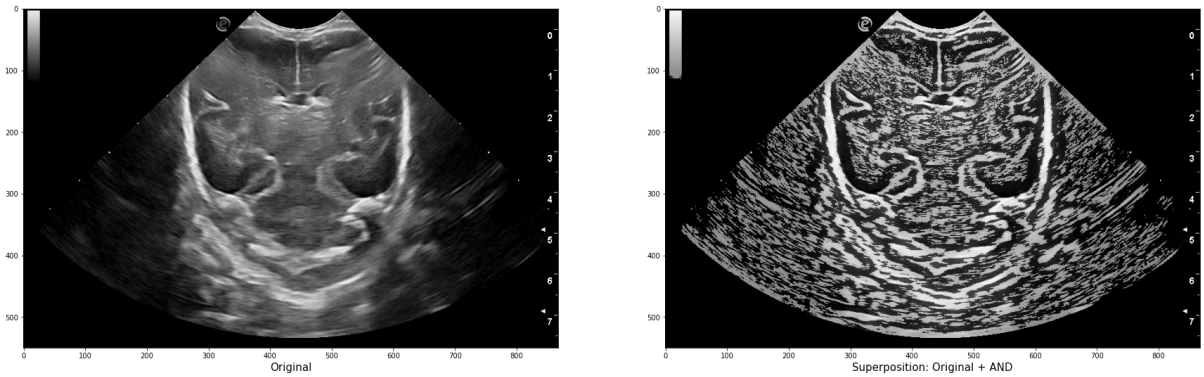

Figure 5.2.1.4. Comparison between the original image and the superposition of the original image with the result obtained from the local threshold applied to the edge detection.

5.2.2. Edge detection

Within the methodology used in intensity, there are also functions that detect the edges that appear in the image. Although the result is still a grayscale image, this method is based on the definition of lines that can be more or less large, depending on the difference in intensity on the analysed surface and marking the pixels with a white hue when the difference of values between them is greater and black, when it is less. In this way, it defines the contours as a separation line between two different intensities, obtaining as a result, an image full of lines from the separation of intensities found by the function in the original image, which can be considered as the contours of the structure. that you are looking for.

It must be taken into account that in a grayscale image, the greater the difference in intensity between structures and the background, the easier it is to find these separations. In this case, in ultrasound there is the problem of background noise, which blurs the background (whose hue should be black), with a grey hue and thus increasing the difficulty of finding the exact pixel where the separation occurs.

For this case, two functions have been used that are already predefined in the skimage.filters library called Robert and Sobel. Both are defined as discrete differentiation operators that are used to calculate an approximation of the gradient of the intensity function of the image. In this project, the functions do not use all the parameters, thus omitting the optional ones and only using the required ones. So, the functions used have the following structure:

- Edge Robert

$$skimage.filters.roberts(image);$$

- Edge Sobel

$$skimage.filters.sobel(image);$$

Where the "image" parameter that is entered in both is the image to which the filter is applied.

In the figure 5.2.2.1., the result of the Robert and Sobel functions applied to an ultrasound is shown, being able to observe how in the case of having applied Robert to the original image, a darker image is obtained as a result where it is true the edges where this change in intensity occurs are defined, but these have a low intensity due to the discrete differentiation operators used. On the contrary, Sobel emphasizes this separation between the background and the structures that appear in them much more, being able to observe how these separation lines have a greater intensity.

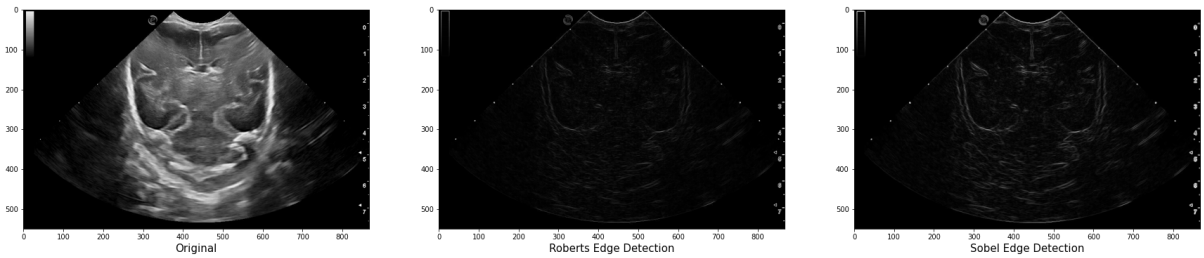

Figure 5.2.2.1. Filters applied to an ultrasound (Original). As an example, the figure shows, from left to right, the result obtained from the Mean, Otsu, Local, Triangle and Yen filters. [18]

It should be mentioned that the filters have been applied directly to the original image, but an image treatment can also be carried out in order to improve the result obtained with the applied filter. To do this, it has been decided to apply the local average as a filter to the image and try to blur the background to try and ensure that it is not detected by the function and does not appear in the image resulting from the filter. On the other hand, if the number of pixels taken into account is too large, structures of interest can be blurred.

This can be seen through the figure 5.2.2.2., where the selem parameter indicates the number of pixels that are taken around and the shape of the surface when calculating the local mean of the pixel to be analysed, obtaining as a result that the greater the pixel value, the more the original image is distorted and the edge function becomes unable to specify the separation between the background and the structures, making the separation line between the two thicker.

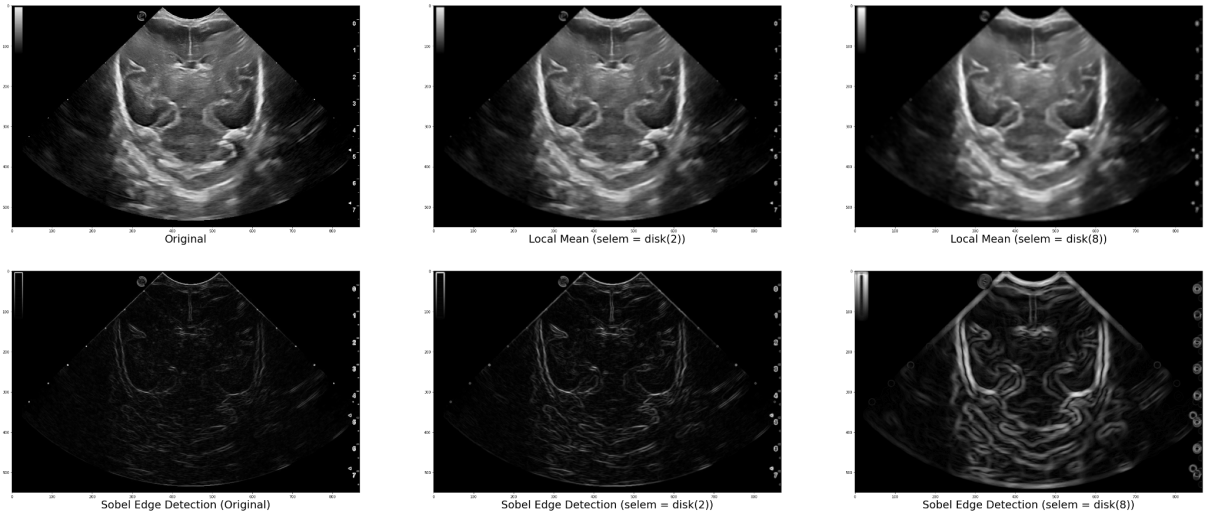

Figure 5.2.2.2. Filters applied to an ultrasound (Original). In the first row the Local Mean is applied with a disk of 2 and 8, on the contrary, in the second row the resulting image is shown when applying the Sobel edge detection to the upper image.

With this, it can be said that care must be taken when choosing the function and the way in which the pre-processing has been done so as not to distort the resulting figure of the original image, since it can be so distorted that when the functions to obtain the results, these are not valid as in the case when a disk of 8 is selected, being able to observe a very thick edge and even appearing in areas where there is no groove or bone.

In addition to pre-processing, other functions can also be applied to improve the results obtained with the applied function, in this case edge. As has been observed, the edges are highlighted but at a low intensity, that is, in the image, the separation lines do not appear with a white hue, but rather with a grey hue. Besides that, there are lines with a value closer to zero (black) that are almost imperceptible. To do this, we have thought about applying the function explained in section 5.2.1 with the aim of eliminating those lines with very low intensity values and highlighting, with the maximum intensity value, those that are perceptible.

The result obtained from what is explained in the previous paragraph can be seen in the figure 5.2.2.3. which shows the process followed to obtain an image where the pixels can only have black and white as values. From left to right you can see the original image to which a local average and a sobel edge detection have been applied, as seen in the second image. Finally, a local threshold identical to that of the intensity section has been applied, thus obtaining an image in which the structures can be seen, which are separated from the background noise, but on the contrary, in this case the structures are not stuffed and only their silhouettes are marked.

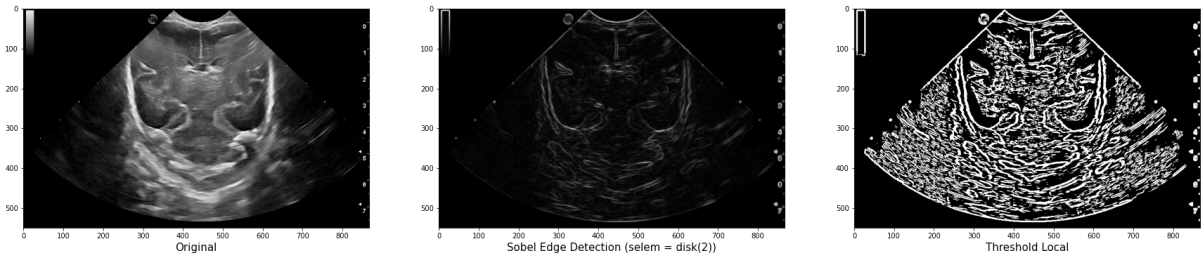

Figure 5.2.2.3. The process of applying the sobel edge detection and a local threshold to the original image is shown.

As has been done in the previous section, in this case we also proceed to show a comparison of the result obtained by superimposing images with respect to the original image to observe which pixels are intensified and see whether or not they belong to structures. The figure 5.2.2.4. shows the comparison of both images where you can see how the noise continues to appear and in addition, the sulci and bones are also shown, but unlike the intensity section, in this case it does not maintain the surface indicating them as a structure, but that marks the outline of each of these.

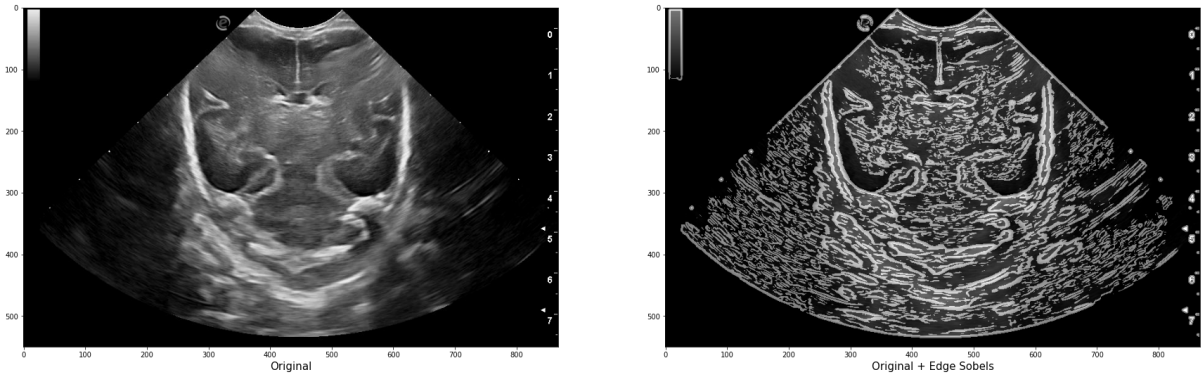

Figure 5.2.2.4. Comparison between the original image and the superposition of the original image with the result obtained from the local threshold applied to the edge detection.

5.2.3. Artefact removal

This last section, which is included in the Unsupervised Segmentation section, explains two mathematical morphology operations, called erosion and dilation, which use structuring elements to achieve, in the first case, the reduction of shapes; and in the second case, the expansion of the shapes in an image where there is only black or white.

Both functions have been used in conjunction with others with the aim of trying to eliminate much of the noise that appears in the treated ultrasounds and maintaining the structures that define the sulci with the least change on them. To achieve this, the functions of the skimage.morphology library called erosion and dilated are used, which receive the same parameters:

- Erosion Function: This function sets one pixel in (i, j) to the minimum over all pixels in the neighbourhood centered on (i, j), allowing bright regions to be shrunk and dark regions enlarged.

$$erosion(image, selem = None, out = None, shift\_x = False, shift\_y = Flase);$$

- Dilation Function: In contrast to erosion, in this case the function sets a pixel in (i, j) to the maximum over all pixels in the neighbourhood centered on (i, j), enlarging the bright regions and shrinking the dark regions.

$$dilation(image, selem = None, out = None, shift\_x = False, shift\_y = Flase);$$

On the one hand, it receives the binarized image to proceed to carry out the method, and on the other hand, the parameter "selem", which defines the neighbourhood expressed as a matrix of ones and zeros. In the figure 5.2.3.1 you can see the image received, which is in the middle and both functions have been applied to it. In the case of the first image, it is observed how the white structures have been eroding and diminishing the surface, causing some of them to disappear. On the contrary, the dilation method makes these increase and overlap each other, thus joining structures and turning them into one.

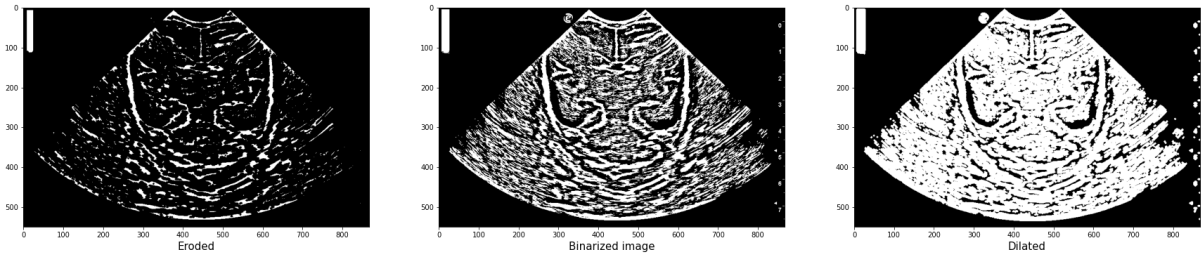

Figure 5.2.3.1. Erosion and dilation applied to a binarized image.

In this case, what you want to do is eliminate and make the large amount of noise disappear from the image and, therefore, the best method that can be used is erosion. The problem with this is that it also affects the bones and sulci, causing them to have less precision and even parts of them may have disappeared.

However, the eroded image can be used to remove the noise from the initial binarized image using the function called reconstruction from the same library skimage.morphology. This consists of a morphological reconstruction of the image, where two images are used, the first one called "seed", which specifies the values to be propagated; and another image named "mask", which provides the maximum value allowed in each pixel.

It should be mentioned that the reconstruction of the image can be done by dilation or erosion. In case dilation is chosen, the high intensity values will replace the close low intensity values but the image used as a mask will limit the diffusion of high intensity values. In contrast, if the method employs erosion, the low intensity values extend from the seed image and are limited by the mask image, which represents the minimum allowed value.

The function used is as follows:

$$reconstruction(seed, mask, ethod =' dilation');$$

Where the parameters seed, mask have been mentioned in the previous paragraph, method is whether dilation or erosion is used in the function; and the last one used, selem, corresponds to the neighbourhood expressed as an n-D matrix of 1 and 0.

The main objective of doing this post process is to eliminate the maximum amount of noise while maintaining the structures that appear in it as accurately as possible. This reconstruction function allows to achieve this using two images: One of them by erosion to the binarized image and using this as a mask and a second, which is the result of another erosion applied to the already eroded image, thus obtaining fewer structures in the image and using it as a mask. In both cases, the selem parameter uses a disk of 2 (selem = disk (2)) and the result of these images can be seen in the figure below.

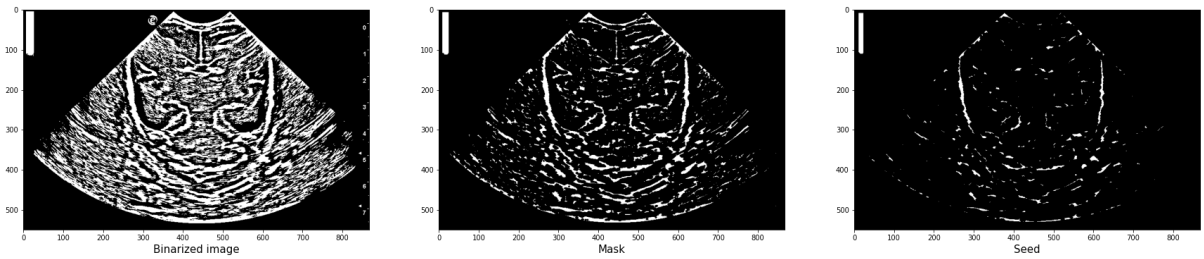

Figure 5.2.3.2. Definition of the mask and seed that are used in the reconstruction function

By using the images defined as seed and mask in the reconstruction function and defining the method parameter as dilation, it is possible to observe as a result (figure 5.2.3.3.) an image in which the structures to be preserved are more or less defined and maintaining the original shape and, in turn, eliminating much of the noise that appears in it, reducing its volume and thus obtaining small structures separated from each other and wrapped in the black background.

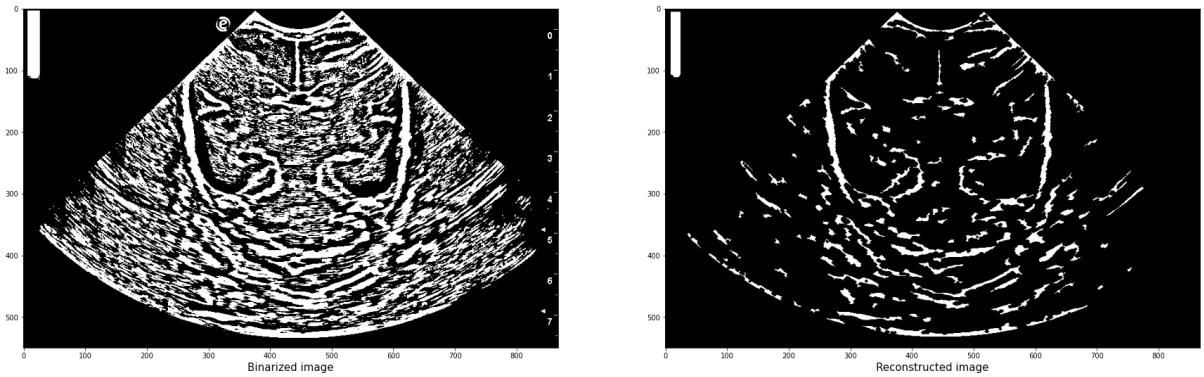

Figure 5.2.3.3. Image obtained by applying the reconstruction function to the binarized image

To better observe if the methods used and explained in this section provide a good result, we have proceeded to superimpose the original image with the one obtained from the reconstruction and compare it with the original, as seen in the figure 5.2.3.4.. In it you can see how much of the noise disappears from the image, being able to observe how structures (the skull bones and sulci) are visible and correctly defined with respect to the original image. It is also observed that thanks to the post-processing done, much of the noise has been eliminated, thus leaving small echoes of them in the dark background of the image and well separated from the important structures.

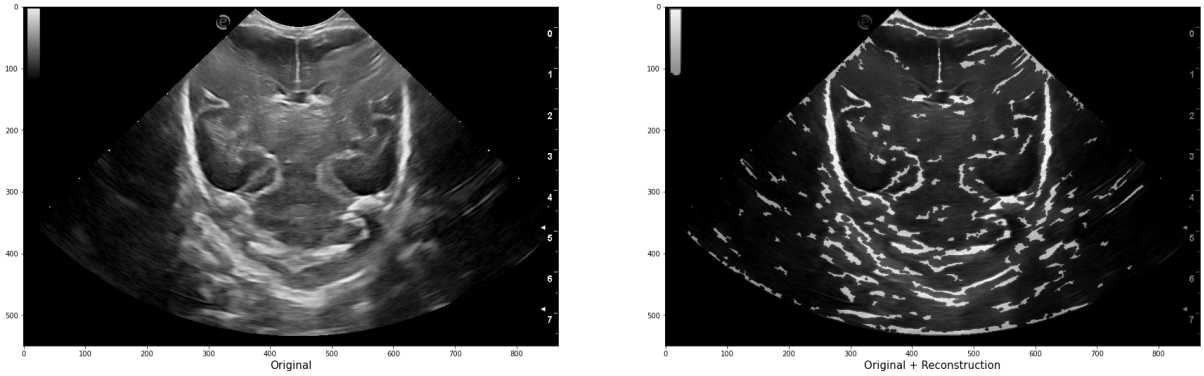

Figure 5.2.3.4. Comparison between the original image and the overlay of the original image with the result obtained from the reconstruction function

Another function analysed is the closing call. This function returns the morphological closure in grayscale of an image. This closure is defined as a dilation followed by erosion capable of removing small dark spots (i.e., "pepper") and connecting small shiny cracks. This allows in many cases to “close” the (dark) spaces between the (bright) features. The function follows the following structure, where the parameters used are the image and the footprint, which allows the neighbourhood to be expressed as a matrix of ones and zeros.

$$closing(image, footprint = None, out = None);$$

An applied example is observed in the figure that follows, where it can be seen how a set of structures that are initially part of the same, are joined together to finally obtain one, being this very similar to the structure that you want to segment.

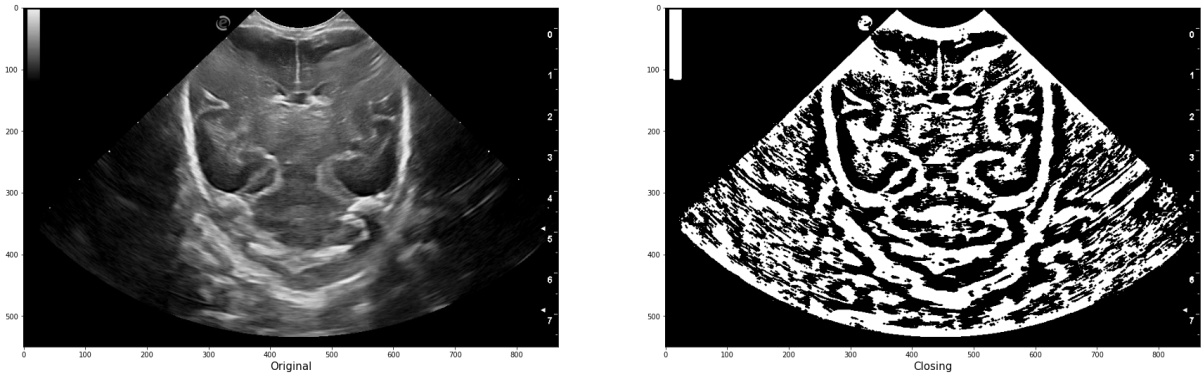

Figure 5.2.3.5. Comparison of the original image with that obtained after applying a threshold and finally the closure function

The problem is that, as has happened in this case, part of the noise that remains in the image and that could not be eliminated by a threshold, is enlarged and can be detrimental when defining the sulci of interest.

To finish this point of erosion and dilation, a method that can be used in the case of edge will be explained. As has been observed in this section, the structures were defined by their profile, thus remaining hollow inside. Python provides a library that contains a function which allows filling in these gaps, bearing in mind that if the values entered in the input parameters are too large, the shape of the structure can be lost, thus losing precision. On the contrary, if values are set too small, the function will not be able to join the contours and therefore the defined contours will continue to be separated. An example of this can be seen in the figure that follows, where it can be seen that depending on the function it has been able to detect the contours, these are filled or not.

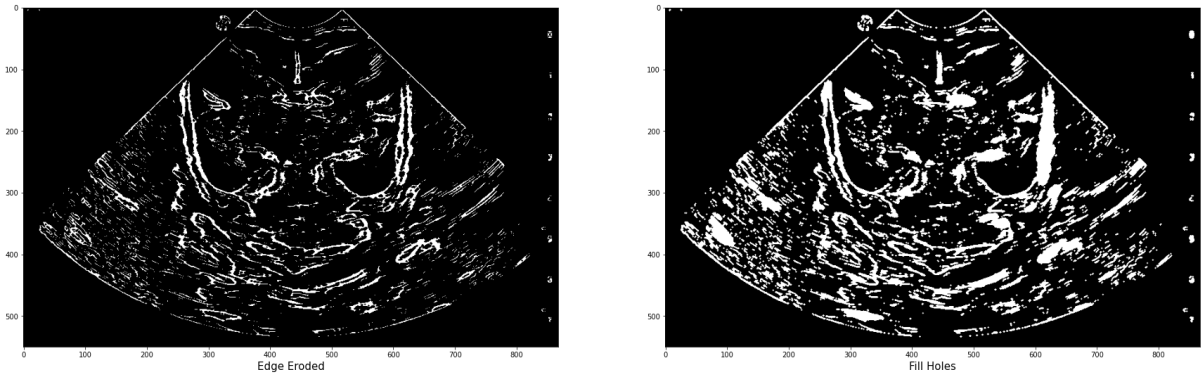

Figure 5.2.3.6. Union and padding of nearby structures defined by the edge detection function.

The result obtained in this case is not entirely adequate, since it can be seen how the noise has become greater, since due to the proximity between them, an expansion has been made and therefore joined. Furthermore, in this case, it can be seen with respect to the previous method that precision is lost when defining the sulci and bones, either because they have become thicker or because the function has not been able to join the contours of the structures because they were too far apart.

5.3. Identification and characterization of sulci

Once the two methods used in Unsupervised Segmentation have been explained, an explanation is given about two algorithms that are defined in one of the Python libraries: skimage, within the measures section (skimage.measures). These functions are called label and regionsprops, in which the first one allows us to identify the structures in the image; and the second makes it possible to obtain certain characteristics on each of the regions defined in the images to be analysed.

For the first of the two functions, label, the image to be labelled (label_image) is entered as parameters and, as optional, the return_num parameter which, if set to True, makes the function return the number of regions found (nregions). Therefore, what you get at the output of this function is an image and an integer.

$$label\_image, nregions\ = label(image, return\_num = True);$$

The image resulting from the label function does not facilitate, as far as the human eye is concerned, the identification of the structures, since the colours used in some cases are very similar between the different detected regions. To solve this and that visually a person can correctly distinguish the different structures, the label2rgb function is used, which returns an RGB image where color-coded labels are painted on the input image, in this case, the one obtained from the label function: label_image; and the original image that is used to reinforce the labels.

$$image\_label\_overlay = label2rgb(label\_image, image);$$

If each of the previous functions is plotted and the images obtained are compared, it can be seen how, in the case of the label function, the structures have been defined on a dark blue colour (the colour assigned to the background region), but the different structures have very similar colours between those that have consecutive identifiers. So, if label2rgb is used to the image obtained by the skimage.measure label function, it can be seen how each structure is defined with a different colour between them and the background, being able to appreciate the shape and surface that occupies each of the 155 structures that appear in the reconstruction image (Figure 5.2.3.3.)

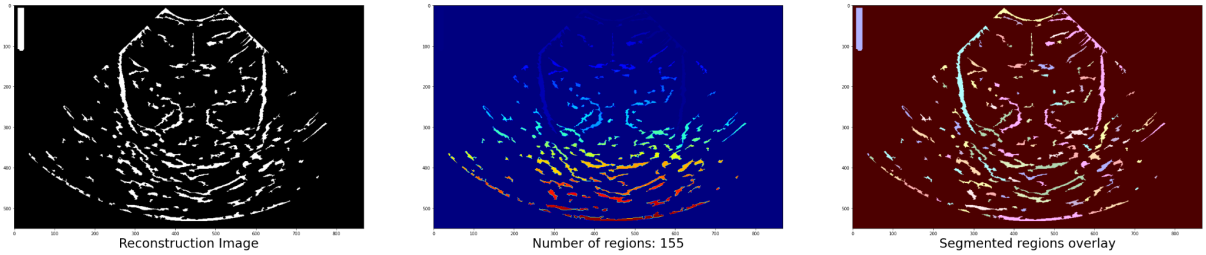

Figure 5.3.1. Definition of the structures that appear in the reconstructed image obtained in section 5.2.3. The first one using the skimage.measure function and the second one, applying the label2rgb to better appreciate the definition.

As mentioned, the label function allows you to recognize structures and label them, but it does not provide information about each region such as the area it occupies, what its centroid is, etc. out of the way, regionprops. It is defined within the skimage.measures library, as an algorithm that returns a list formed by as many elements as structures have been detected in the label_image image and, in each of its positions, characteristics are stored on the structure in the form of dictionary. Some of this information that each position contains is, for example, the area, the coordinates in which it is located, its centroid, etc.

$$props = regionprops(label\_image);$$

To access the characteristics of an element that has been obtained by applying the function, it is necessary to know the position in which it is located, that is, each structure corresponds to a position in the list and, as with vectors, you can access by adding a number between claudators to the variable 'props'. In the case in which you only want to know a characteristic of the selected element, such as the area, it is necessary to enter the attribute as a string, as shown in the example that follows:

$$area = prop[0]['area'];$$

This function has the particularity and unique function of, through a binarized image, to know the number of structures present and some of their characteristics, to condition, through these, if it is worth keeping the label or eliminating it and that, for Therefore, it does not appear in the image. That is, according to one of the properties of the elements, it can be decided whether to eliminate part of those structures or to keep it as shown in figure 5.3.1. In this you can see, to the right of the initial binarized figure, a histogram that shows approximately the number of elements (vertical axis) that a range of areas has. In addition, the limit that defines those elements that are kept in the image is shown in the black line, as can be seen in the figure on the left, in which, for this example, all those structures with an area less than 50.

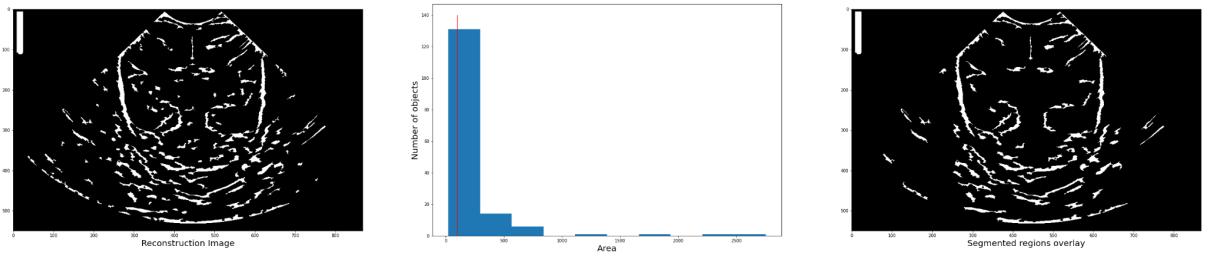

Figure 5.3.1. Definition, by means of the histogram, of the number of structures that have an area and the elimination of those with an area lower than the reference value (red line), in order to obtain a new image (third column) with those structures that meet the condition.

Another feature that can also be obtained from the regionprops function and that has been used in the current project is that of centroids. This property labels the coordinates (x, y) of the central point of each of the structures and to be able to decide whether or not to eliminate a structure according to its position in the image. To better observe this, we proceed to enlarge the reconstructed image with the structures of an area smaller than 50, as shown in figure 5.3.2. In this, the centroids of the structures are shown by adding the maximum and minimum values of the axes for each of the labels, thus obtaining a box that defines the surface on which the structure is located in the image.

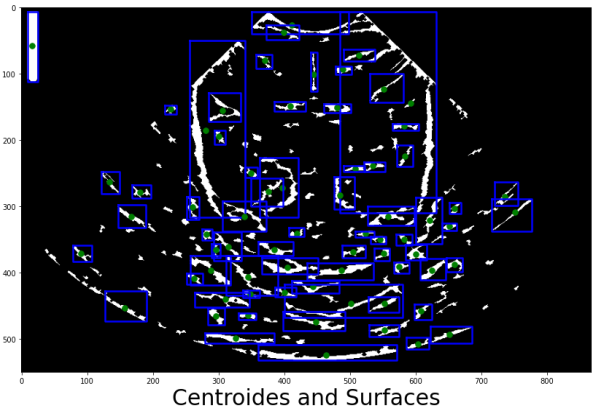

Figure 5.3.2. Definition of the centroids and surfaces where there is a structure.

5.4. Contours

This is one of the most important sections of the project, since it allows to identify the contour of the structure to be analysed. The segmentation process is considered one of the general problems in the field of computer vision and is a process that consists of dividing a digital image into different segments, known as image objects, with the aim of simplifying and changing the representation of the image. image into something more meaningful and easier to analyse. This method is used in order to detect and locate objects and their limits, whether they are straight lines, curves, etc; found in the image. So,

in this way it is possible to assign, to each pixel of the image, a label so that those that have the same category are considered to share the same characteristics and, therefore, are the same structure.

When segmentation is applied to an image, the result that is obtained is a set of segments that collectively cover the entire image or a set of contours extracted from the image in order to detect the edges. The result can be interpreted as a set of regions in which each of its pixels share similarities with respect to some characteristic or property such as colour intensity, and in which the pixels of adjacent regions do not share the same characteristics.

Image 5.4.1 shows an example of segmentation performed in a case of computer vision detection of citrus peel defects using a region-oriented segmentation algorithm. In the figure, you can see how the different colours correspond to different labels detected on the orange peel. In addition, by means of a threshold, those categories that do not comply with a certain characteristic have been eliminated to identify the areas in which the orange colour does not appear.

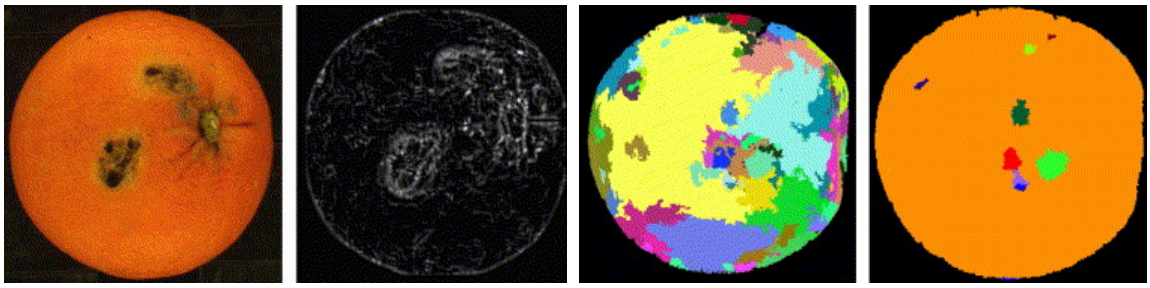

Figure 5.4.1. Example of segmentation carried out on an orange to eliminate categories that do not meet a condition. [19]

It is worth mentioning that the result of the images treated with thresholds are considered as part of the segmentation, since the pixels are being labelled according to intensity and they form a structure. The problem is that the region obtained has a surface that you do not want to visualize, since you only want to define the contour to visualize it on the original image with a colour, for example red, which allows you to easily identify the structure on the rest of the image.

This can be done using the slic function from the skimage.segmentation library, which performs a segmentation of the objects in the image through k-means clustering. In this function, as in the previous ones, the input parameters have to be defined: the image to which the segmentation must be applied, the number of labels by which the image will be initially segmented, the mask (an optional parameter) which is a two-dimensional matrix and which is used to calculate only the super pixels where the mask is True (1), and finally the start label, which indicates the start value of the labels. So, the function used in the code used is as follows:

$$m\_slic = segmentation.slic\left(img, n_{segments} = 2, mask = mask, start\_label = 1\right);$$

If we do not take into account the input parameter corresponding to the image, the most decisive one is the one that allows defining the number of segments that the image has from the beginning, and which will be adjusted to the image structures using K-means : n_segments. In addition, adding a mask made up of the Boolean values False (0) and True (1) makes the calculations that the function has to perform to segment the objects and thus obtain their contour, are facilitated.

An example of how the labelled image looks can be seen in the figure 5.4.2., which shows different examples of grids, in which it can be seen that, as the value of n_segments increase, so does the number of clusters and, therefore, the number of lines (yellow colour) that divide the surface.

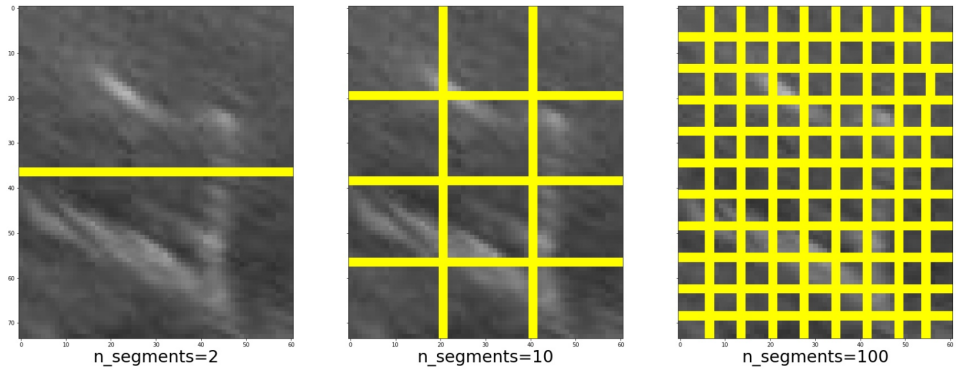

Figure 5.4.2. Definition of the number of n_segments in the image, as it grows, more yellow lines appear.

It should be mentioned that the number of divisions with the value indicated in n_segments does not always correspond, since in many cases an approximate number of labels is taken in the segmented output image.

All this is done with the sole objective of obtaining the contour or contours of the structure that is kept in the image, taking into account the mask. Therefore, we will proceed to explain the process followed to obtain the segmented structure, in this case using as an example an ultrasound of a premature baby with coronal section number 4. In the example shown and explained below, the segmentation is performed in a specific groove called the sagittal, to better describe and appreciate the process. To do this, a process similar to that of the application will be followed where only the image area is analysed once it has been cropped with a rectangular shape, thus obtaining a smaller pixel area and, therefore, the segmentation function, which is based on a local threshold (section 5.2.1 Local threshold), applied

to the selected area and not to the entire original image in order to reduce the number of calculations and focus on the sulci.

So, to focus on a certain structure and reduce the calculations that have to be made to segment, the original image has been cut into a rectangular shape to stay and maintain the desired area. This same step is also carried out to obtain the mask of the same area on the reconstructed image (section 5.2.3.), Since both images are necessary as input parameters of the slic function, as explained previously.

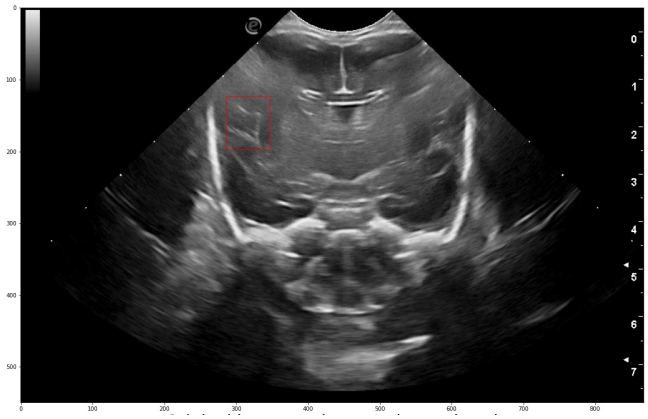

Figure 5.4.3. Definition, by means of a red rectangle, the area in which the segmentation algorithms will be applied.

In figure 5.4.4., The steps to follow and the results obtained from the slic function are shown where, from left to right, the cropped area of the original image can be seen followed by the same area but by the mask. So, once the image and the mask have been obtained, we proceed to define the approximate number of regions that are related to the K-means and which will be moulded by the image and the mask until they fit the structure, being able to see how the Grid (yellow line of the third image), defined by the n_segment parameter, has been shaped and adjusted to the structure in order to obtain the segmentation of the structure (red line of the fourth image). For this example, specifically a value of 10 segments has been used.

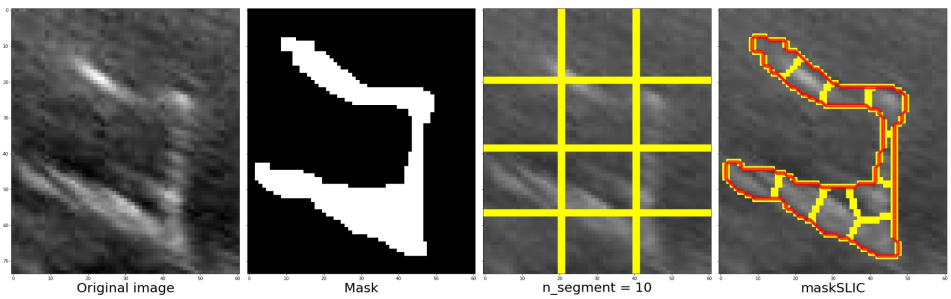

Figure 5.4.4. Steps to be followed once the zone has been defined, defining the mask, the number of segments and finally the contour using the maskSLIC function.

The objective of this method is to extract the contour of the slic function (red line) to transfer it back to the original image and have the shape of the structure to be segmented, defined and marked, in this case a groove. That is, the result of the segmentation is a 2D matrix with the same width and height as the cropped image, and which is made up of zeros and ones, where the zeros correspond to the background and the ones to the outline. Therefore, it is necessary to resize this matrix to the dimensions of the original image but taking into account the position in which it should be for the segmentation to coincide, that is, the obtained contour must be placed in the position of the element selected from the full ultrasound image. So, it is important to save the values of the vertices (maximum and minimum x and y) of the bounding box (shown in figure 5.4.3.) So, using an empty matrix of the same dimension as the original image and these saved values, translate the outline with respect to the original image so that they can be viewed together, as shown in the following part of code:

$$mask\_segmentation = np.zeros(img.shape);$$

$$mask\_segmentation[y\_min: y\_max, x\_min: x\_max] = m\_slic;$$

In the code, mask_segmentation is the matrix to which the contour is to be placed in the desired position, and m_slic is the matrix obtained from the segmentation function which contains the contour. Therefore, the dimensions of the first correspond to the original image (img) while those of the second are those of the selected region. In case it is done correctly, the result obtained can be seen in the following figure, which shows the original image that we have wanted to analyse and the contour of the groove obtained from the m_slic function and that has been moved in the area in the which should be to correctly display the result.

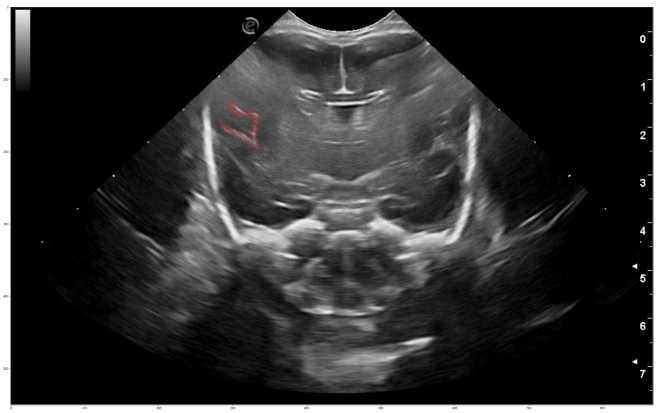

Figure 5.4.5. Sample of the final result when applying the segmentation algorithms explained in the section to a groove.

5.5. Graphical user interface

When working with a programming language, in this case Python, the objective to be obtained is a set of algorithms that, when working together, allows obtaining a result according to the input or inputs it has. Examples of these are the application of a filter to an input image or the definition of structures of a colour image converting it to grayscale, applying a threshold and finally detecting objects that have remained in the reconstructed image. as it has been part of the project.

From the point of view of an engineer, it is easy to get the file or folder where the code is located and run it to obtain the final result and analyse it. But if it has to be executed by a person who does not have programming knowledge, as in this case, it is necessary to create an interface that facilitates the execution and interaction with the code in a simple way and without the need to see code.

To achieve this, the Dash library has been used, which allows creating interactive applications that are opened through a browser without the need to have knowledge of HTML, CSS and Javascript to create interactive panels. The advantages of Dash are that there are libraries that work together with it, such as dash-html-components, which allows you to compose the design of the structures as if you were writing with HTML or using an HTML template engine; or dash-core-components, which contains a basic set of components such as buttons, plots, Drag and drop, etc. These two libraries contain elements that allow you to build and define the application with its corresponding functions only with the Python programming language.

In figure 5.5.1., an example of the previous libraries is shown, in which it is observed how two columns have been integrated into it, the first with a single graph and the second with two, each of them depending on their respective dropdowns, sliders, and radio items. In addition, you can select the region of points you want to analyse to show the results through the graphics on the right. So, it is a case where you are allowed to interact with your code.

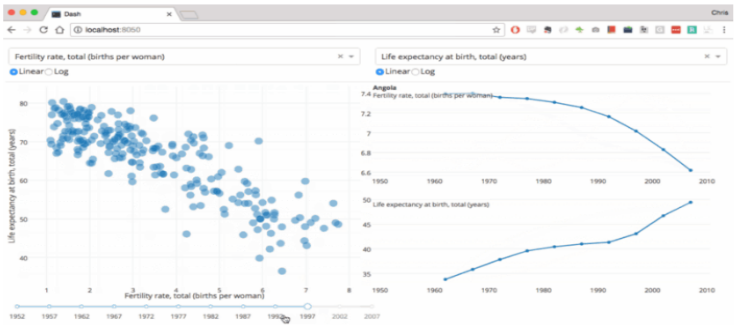

Figure 5.5.1. Dash application example [20]

5.5.1. Building blocks of Dash

One of the most important points that must be taken into account when building an application are the necessary blocks for it to work, perform the calculations and interact with it. In this case, the two blocks that make up the applications made using the dash libraries are layout and appCallback.

In the case of the layout, it describes how the application is, that is, the appearance it has; elements such as graphics, drop-down menus, etc are defined; as well as the location in which it is located, the size of each one of them, the colour, etc. In addition, this block creates and designs HTML content from the dash-html-components library, which allows you to create headings, paragraphs, images, etc. The dash-core-components library is also used to enter graphics, drop-down menus, buttons, sliders, etc. In both cases, only Python is used as a programming language to define and place it where the user or client wants it.

On the other hand, there is the block called appCallback, which is used so that the application has interactivity with the person who uses it by performing, for example, a click on a button and thus executing and performing the activities and operations necessary to obtain a result and display it through another element, such as a graph or table, for example.

So, you can differentiate two parts within the dash, one in which the aesthetic part of the application is defined, and another more related to the functional part and that depends on the interacting elements. For this to work correctly and both blocks are related, it is important that a parameter called id is defined for each element of the layout, which must be unique for each one and that allows identifying and obtaining the information that the desired element maintains, such as values, tables, graphs, etc; In addition to being able to return the result obtained from the function that has been applied and show it through another element or the same element of the app, in the latter case producing the exchange of the data that is sent with those returned by the appCallback.

For this change of values and information shown through the application to work, it is necessary to define the inputs and outputs of the appCallback, which are composed of the id component and its property as an argument when it is defined. In the case of inputs, they are used to define the components, the change in value of which is triggered in the callback. In the case of output, it is necessary to define the components that will be updated within the design when, when the set of operations and functions is performed, it returns one or more objects, such as values, tables, data, etc.

5.5.2. Application project

The set of functions that have been explained in the previous sections are used to define structures in an ultrasound, determine their centroid and define the segmentation to obtain, as a result, the same

image but with the outlines of the desired structures, trying to this is the most faithful and adjusted to the real one. So that this can be used by people in the medical field, it has been chosen to make an application through dash, so that they themselves can define the areas and obtain the results without the need for a person with programming knowledge.

For an easy compression of where the image being analysed is located and its corresponding data and separate it from the resulting image with the values obtained from the functions defined in the appCallback, it has been decided to separate the layout by means of two main rows, such as and as can be seen in the figure 5.5.2.1.. The first consists of the original ultrasound and a table that contains the areas to be analysed and that have been selected by the user; while the second row contains the same image to which the obtained contours have been added, as well as another table in which the coordinates obtained from the segmentation are shown. So, the main part of the layout is made up of 4 different cards, of which two of them correspond to the first row (each in a different column) and a second row with the same number of cards and separated in the same way. It is worth mentioning that the first column of both rows contains the corresponding images and in the second, the table with the coordinates for either the first case (Table 1) those indicated by the medical staff, as well as the second (Table 2) those obtained as a result of the functions.

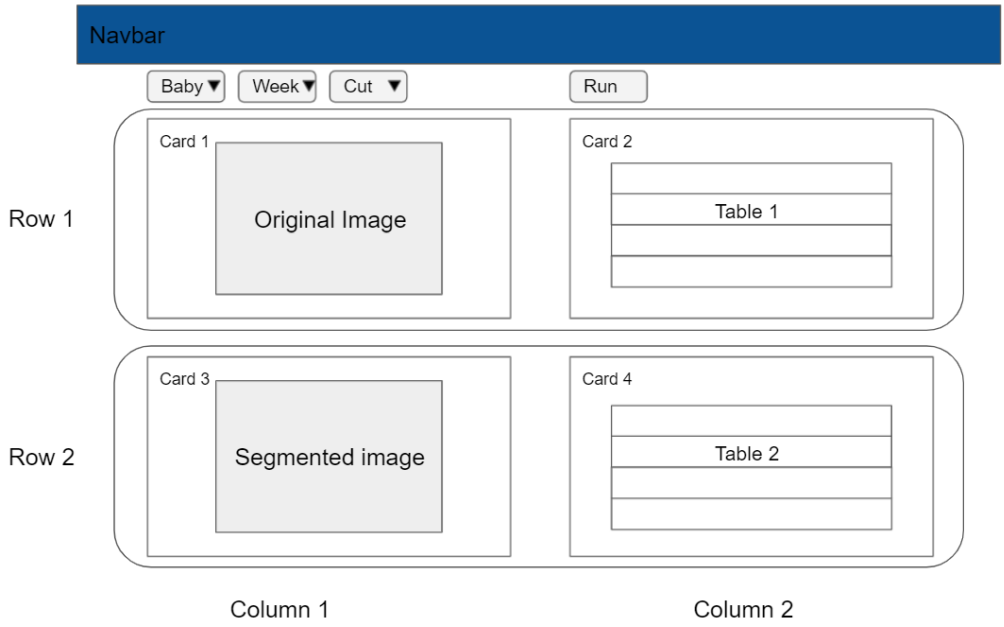

Figure 5.5.2.1. Design of the project application.

To make the application interactive and allow image changes and run the segmentation code, two different types of buttons have been added. For the first case, three dropdown buttons have been used that allow the modification of the image that is displayed, being able to choose the baby through

its identifier defined by the hospital, the week in which the ultrasound was performed and finally the brain cut. For the part of the execution of the functions and that the result of the contours is shown in the bottom row, a button called run is used to start the execution.

5.6. Project in the cloud

In this last point, we proceed to explain Docker, a software application that allows the developer to quickly create, run and scale their applications by creating containers. This is one of the most used mechanisms when deploying software, since it allows a technology based on a language and an operating system used by the developer, such as dash in the case of this project, to be available to anyone. person thanks to its ease of use.

Figure 5.6.1. Docker's Logo.

To properly understand what the docker application is, you have to start talking about containers. In software, they are considered as a standard unit that packages the code and all its dependencies so that the application runs quickly and reliably from one computing environment to another. For each container, it is necessary to create an image, which is defined as a lightweight, independent and executable software package that includes everything necessary for the application to work such as its code, runtime, system tools, libraries of the system and settings. The main characteristic is that the containers are always isolated from the software in their environment and ensure that it works uniformly despite the differences between development and staging, that is, that containers allow the virtualization of the operating system or environment rather than the hardware of the machine on which it is being used. So, this makes them more portable and efficient and therefore more useful in cases like this project.

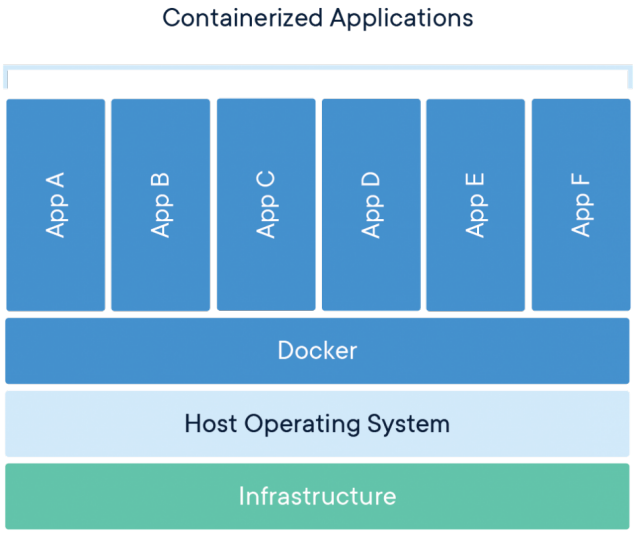

Figure 5.6.2. How works Docker.

In order for the image to work correctly, it is necessary to create a document, called Dockerfile, which contains some commands that are executed when it is called through the console that opens inside the folder from which it is going to be created. the picture. For this, it is necessary that the programming language is defined in it, together with the version used, the address of the image, the libraries to be installed in the container, the port used between the dash application and the host machine, and the command is also included to copy everything that is in the folder

To do this, it is necessary to have the docker application installed on the computer and use the following command lines in the console within the folder of the software that you want to create the image:

$ docker build --tag app_segmentation .
$ docker tag app_segmentation user/app_segmentation:app_segmentation

Once this is complete, the image can be accessed from the developer's computer through the docker application without the need to run the document or file containing the software. The problem is that the image is found locally and, if you want to use it on other computers, it is necessary to perform a push so that the image is saved in a repository, as shown in the next line of command:

$ docker push user/app_segmentation:app_segmentation

Once this is done, the following lines appear, which indicate everything that is being loaded into the repository so that it can be downloaded by anyone with access.

The push refers to repository [docker.io/user/app_segmentation]
5fdc0d51fdb3: Pushed
63df04689c9d: Pushed
1eb91e823e20: Pushed
3cc89c4f3133: Pushed
0d30d0f872bf: Pushed
1c0caf8b7cd7: Mounted from user/app_segmentation
cd6b2a9ae627: Mounted from user/app_segmentation
84c97f2e3099: Mounted from user/app_segmentation
b0cb6a43f300: Mounted from user/app_segmentation
4b4c002ee6ca: Mounted from user/app_segmentation
cdc9dae211b4: Mounted from user/app_segmentation
7095af798ace: Mounted from user/app_segmentation
fe6a4fdbedc0: Mounted from user/app_segmentation
e4d0e810d54a: Mounted from user/app_segmentation
4e006334a6fd: Mounted from user/app_segmentation
app_segmentation: digest: sha256:22c6fa7bcec7c759888fc4d008f0f0f9baeb387d4c0b57a3565683dd5809752d size: 3483

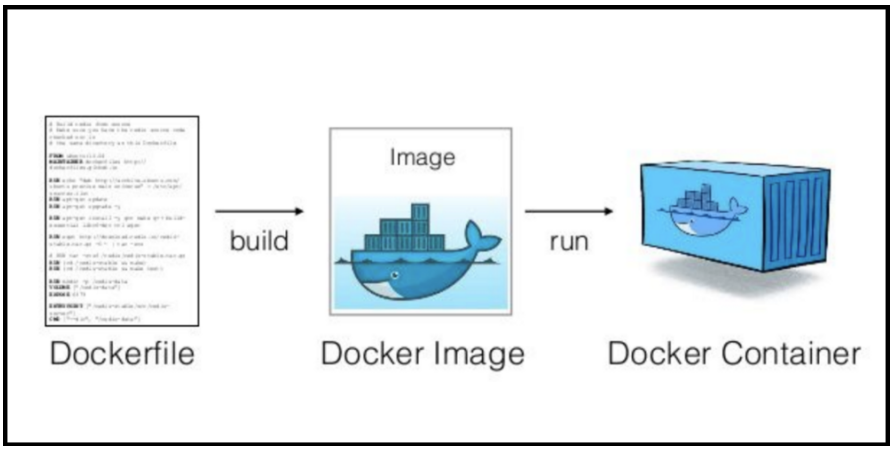

Figure 5.6.2. Process of creating a Docker Container.

For this to work on another computer as an image, it is necessary that it has the docker application installed, since it is the one that will allow the user to run the software and, through the console, execute a pull from the repository to download the image, such as and as shown in the following command line:

*docker pull user/app_segmentation:app_segmentation_*

6. Results

In the previous section, concepts, processes and libraries that have been used to achieve a desired objective for each of them have been explained, but that used together and in a certain order, allow obtaining the final result of the project, which is a model application capable of performing a semiautomatic segmentation of the sulci that appear in the ultrasound of the section to be analysed. So, in summary, this section presents the result of the application obtained by including the local threshold, segmentation and dash functions; in addition to the process that follows to obtain the segmentation of the sulci identified in an image by the user.

At this point we proceed to explain the structure and design of the application that has been defined through the dash library and how is the interaction between it and the user who uses it so that there is a correct use of the functions in the moment in which the user wishes to apply them and, in this way, obtain a final result, being capable of being modified by the user.

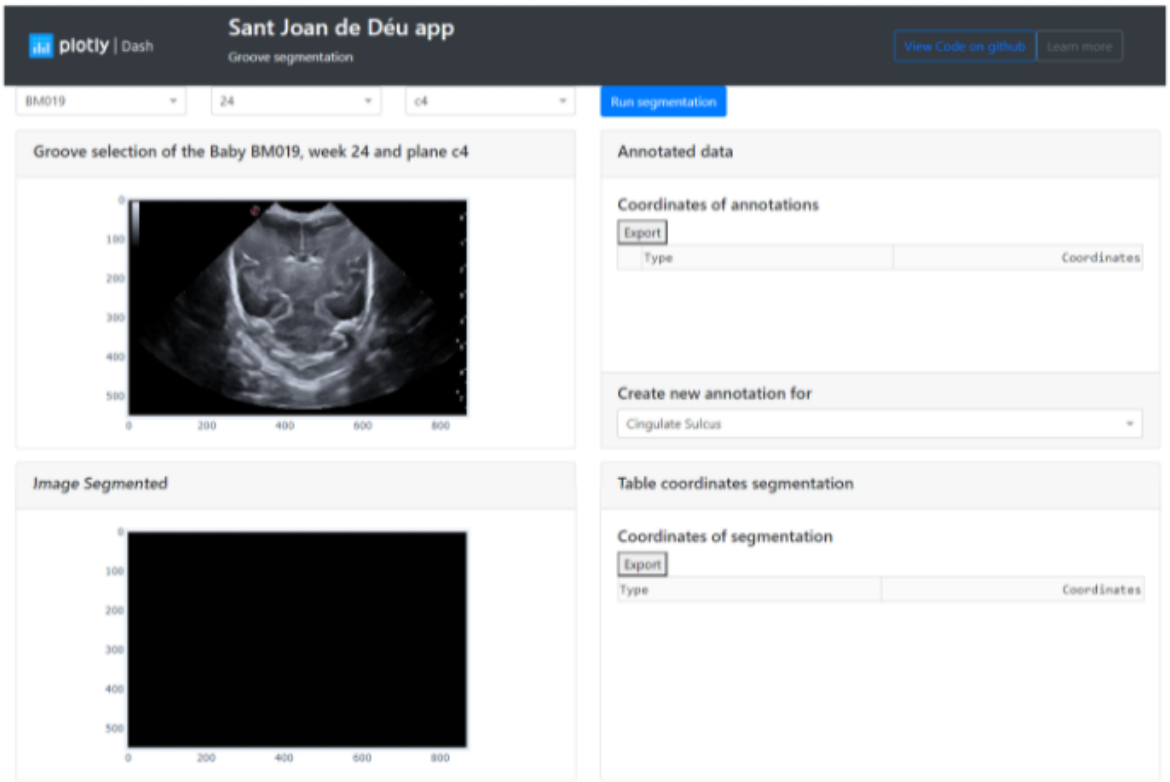

Figure 6.1. Semiautomatic groove detection application.

First, it begins by explaining how the application is displayed through the monitor, being able to see that the interface is simple and easy to interpret to search and display the desired information. As shown in figure 6.1., The app layout finally consists of three rows and two columns. In the first row, there are three dropdowns in the first column that allow you to choose the baby, the week and the cut from the database, and a fourth button in the second column (blue colour) that when clicked allows you to obtain the result of segmentation based on what is selected by the user in the chosen image. In the second and third rows a similar structure is observed, since in both cases, in the first column, a card with its respective header and a body in which the figure is plotted is observed, and, in the second column, it is observed You have defined another card with a header and a body where the table of coordinates is located. The structural difference between these two rows is in the fact that, in the second row of the application, specifically in the second column, there is also a footer with a dropdown to choose what type of groove to select in the original image. In summary, it can be seen how the application consists of two main parts: a first where the image to be segmented is selected and the areas of the sulci are defined and, a second in which the results of the algorithms are shown once information is entered.

Having said this, we proceed to explain in more detail the two parts that exist and the differences between them, since in one, the participation of the user is necessary to be able to carry out semi-automatic learning, while in the other the results of so that the user can modify the resulting segmentation. The first part consists of the buttons located in the first row and the two cards that are in the next row. In it, the user can select the baby (identifier), the week and the court that he wants to show in the first card (left column). Therefore, the corresponding image is displayed if it exists in the database, and if it does not, the displayed image is a black image that has been defined. In this way, it allows the person who wants to carry out the segmentation to know whether or not it is found in the database depending on whether it is displayed or not.

In the event that the image exists, the user can choose the groove that he wants to define in the image by means of the dropdown in the second card (second column of the second row), in which the name of all the sulci can be found. that you want to define in the images of the medical study for which the application of this project has been developed. Once the name of the groove has been selected, the user goes to the image card and proceeds to define the area in which the groove is located using the draw closed freeform tool. This process is repeated as many times as the sulci you want to define, obtaining the image of the cut with segments of different colour according to the groove that has been chosen, thus defining the area in which it is located. At the same time, the modification of the table is observed, since as a new groove is added, it appears in a new row of the table with the characteristics of the new segment, that is, the name and the coordinates.

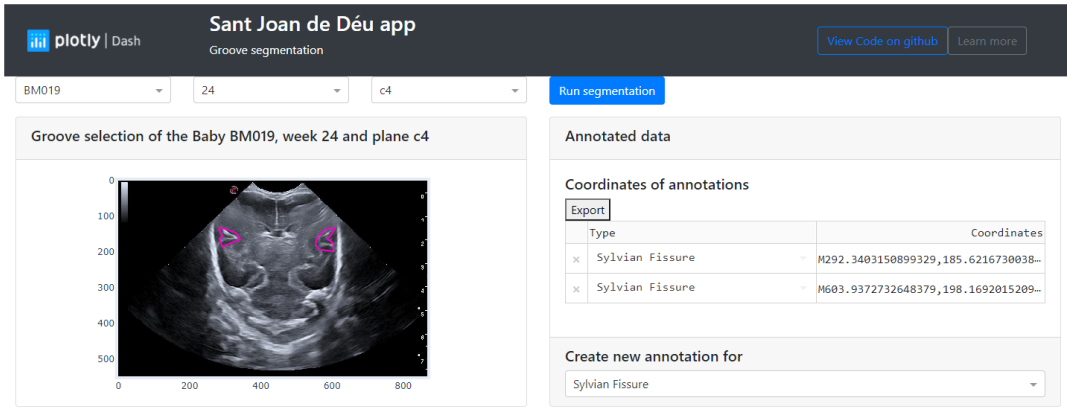

Figure 6.2. Manual selection of the area in which the sulci are located, in this case Sylvian Fissure

In case the segment selection is not correct or has been defined incorrectly, two different actions can be taken to correct the error. The first and easiest of these is to delete the segment from the table using the cross that appears in the first column and redefine it if necessary. The second consists of selecting the segment manually so that the vertices by which it is formed are displayed and being able to select one of them to move it and, in this way, improve the definition of the groove by modifying the value of the coordinate, both in the image and in the table, as shown in the figure below.

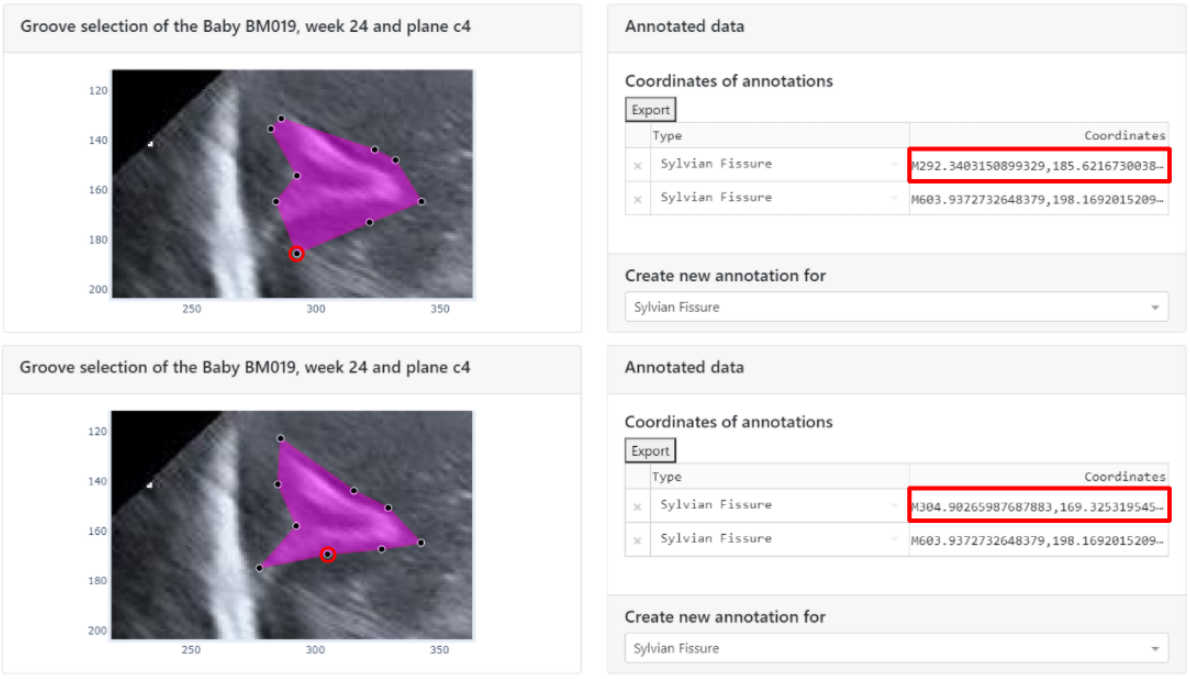

Figure 6.3. Movement of the vertices of the manually defined segment and changing the coordinates of the corresponding groove in the table.

Once the section of the groove selection has been explained, we proceed to explain the part in which the result of the algorithms is obtained once the blue "Run segmentation" button has been clicked. To obtain the desired result, it is necessary to go through a set of functions explained in the Procedures Performed section. This process consists of 5 steps in which it is defined as follows:

- The first of all of them is the rectangular cropping of the image, taking the maximum and minimum values of both axes to create a new image to which the algorithms will be applied.
- Once there is a new image where the groove is located, a local threshold is made to try to eliminate the noise that appears and the closure is applied in order to keep the sulci and bones in those where the structure is divided into more little ones and these are part of it.
- As a third step, we proceed to identify the structures, know the centroid and the area of each one of them to maintain the structures whose centroid is within the manual segmentation and eliminate the rest. Once the structures that meet the condition have been identified, we proceed to analyse which of these contains a larger surface area, in order to eliminate the rest and obtain an image with a black background where only a single structure appears.
- With the image obtained in the previous point, before performing a new segmentation using the Slic algorithm that allows to quickly and easily obtain the outline of the structure, the image is multiplied by a black and white one where there is a structure defined by the manual segmentation. This multiplication of images has the objective of eliminating those parts of the structure that are outside the manual segmentation that the user has not marked as part of it.
- Finally, so that the contour can be shown correctly in the original image, we proceed to adjust the coordinates of the rectangle cropped with the segmentation to where it had originally been cropped in the original image. This is done using transposition of the image coordinates with the help of the maximum and minimum values of the x and y axes obtained in the first section when cropping.

The diagram in figure 6.4 shows the steps to follow both in the application and in the implemented algorithm. Being able to see the actions to be carried out for the algorithm to run and display the results on the screen.

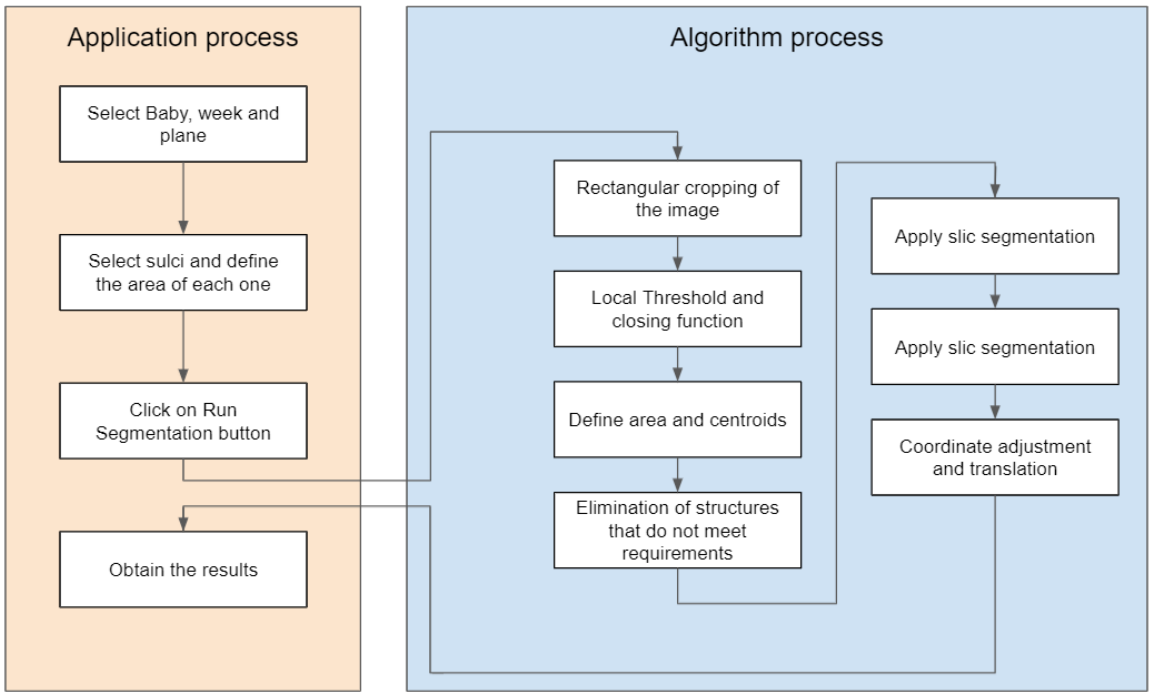

Figure 6.4. Application operation diagram

This process is done as many times as sulci have been defined, obtaining as a result an image (figure 6.5.) in which the same baby, week and cut as the one chosen in the first part is shown, but in this case the segmentation obtained It is not manual, but the one obtained by the algorithm defined above. So, you can see how the manual segmentation has been modified by one that is closer to the shape of the groove that has been selected. In addition to the image, the coordinates (in the horizontal and vertical axes) of each of the vertices that constitute the segment are returned for each of the sulci defined in the annotation table.

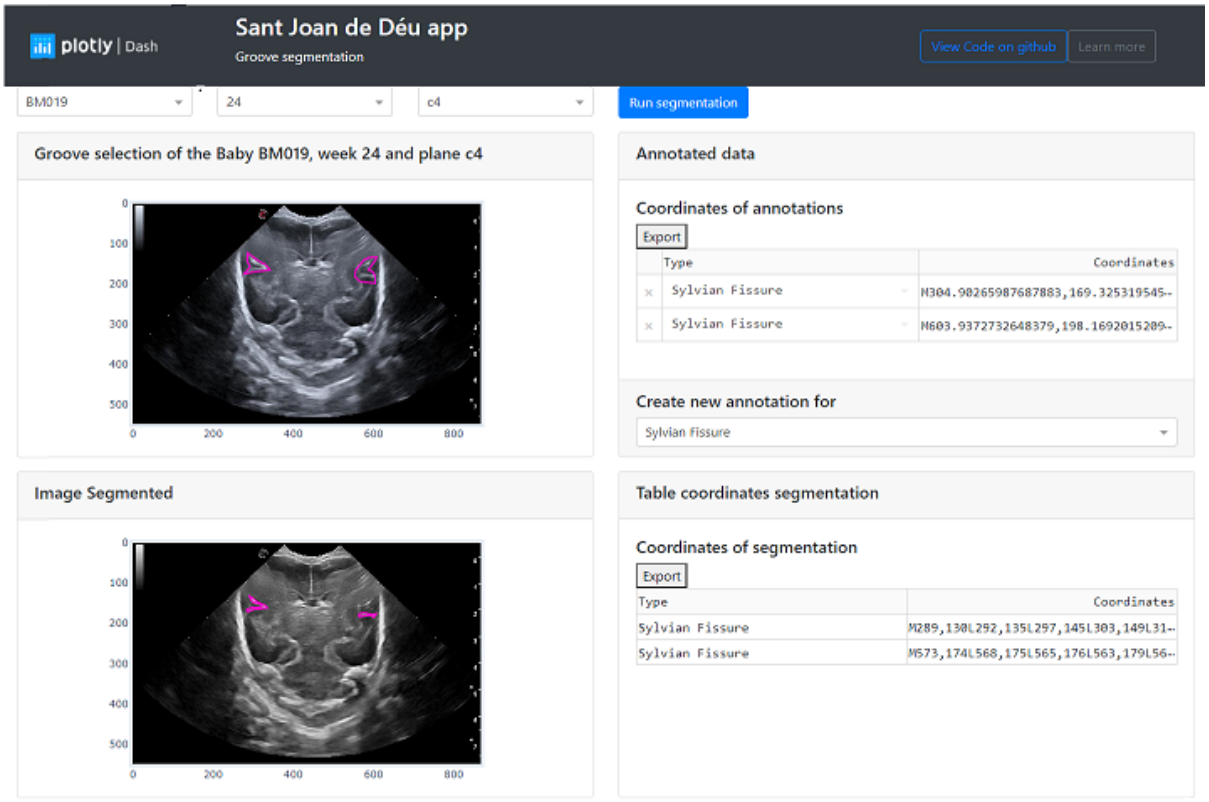

Figure 6.5. Result, in the last row, of the segmentation carried out by the application algorithms for each of the sulci defined in the first row.

Sometimes the segmentation algorithm does not work correctly due to different factors present in the image. An example of this is the fact that the local threshold may not correctly separate the groove from the ultrasound background noise or may separate it into various structures and therefore only a segmented part is shown (due to the third step explained previously). To solve this error, the user is allowed to make modifications to the segmented image to be able to adjust the vertices, as in the case of the second example, in which several of the vertices of the segment can be moved and adjusted to correctly define the groove. An example of the latter can be seen in the figure 6.6., which shows how the result of the segmentation does not correspond to what is desired and, therefore, it is necessary to adjust the result, being able to observe as part of the vertices of the tops have been placed in new positions in the image to correctly define the groove.

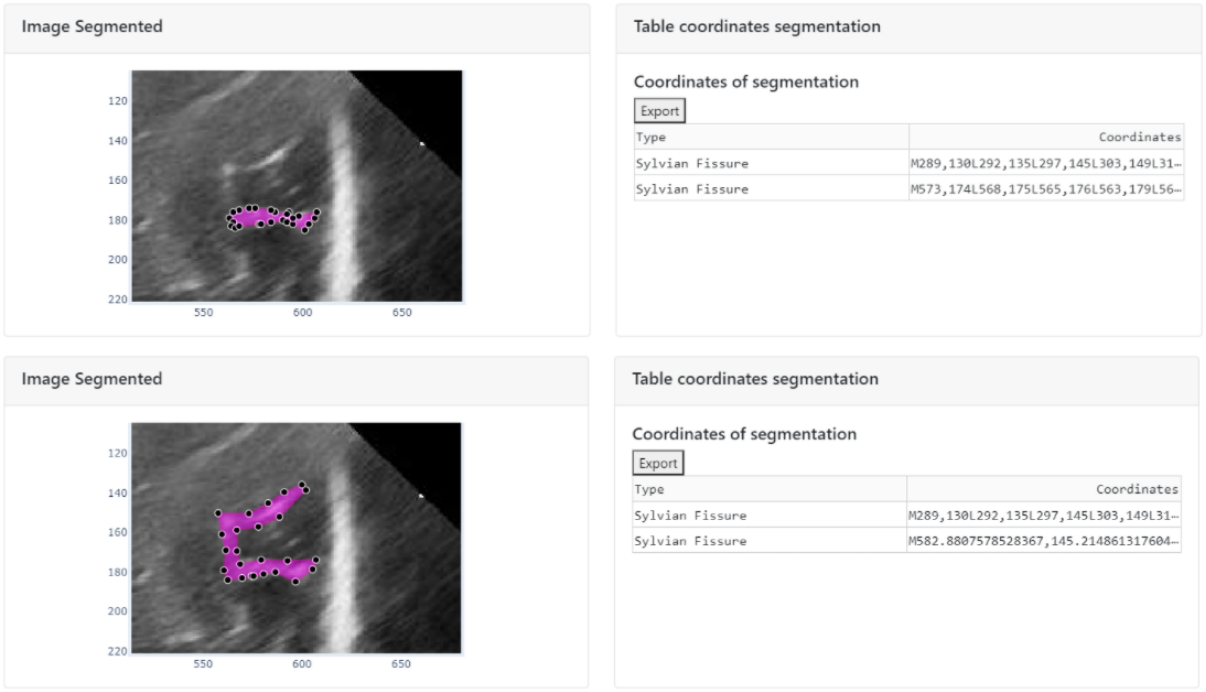

Figure 6.6. Movement of the vertices of the segment defined by the algorithm and change of the coordinates of the corresponding groove in the table.

It is worth mentioning that the modifications of the vectors that are made in the image will be reflected in the table through the modifications in the displayed values of the selected coordinates. As a last point, the data defined in the table can be exported to an excel document using the "Export" button, which appears at the top of both tables, and which once clicked, proceeds to download the file from the browser.

To finish this section of results, figure 6.7 is shown, which contains several examples made with the application of segmentation in the Sylvian groove. These examples refer to babies born between 24 and 32 weeks gestation. That is, each row represents a baby born in a given week (from top to bottom the weeks are: 24, 26, 28, 30, 32) and the image in the first column represents the segmentation performed by the user and, in the case of the second, the one carried out by the algorithm. It can be seen that in images where the groove is moderately defined, that is, in which noise does not affect its visibility much, the result provided by the algorithm is good and therefore, it is not necessary to move the vertices to correct the contour obtained by the algorithm. On the contrary, in those structures where the noise is too great and produces that the perception of the structure is almost null, it is appreciated how the algorithm hesitates to define it and therefore an irregular contour is obtained and, most likely, it does not coincide. with the original.

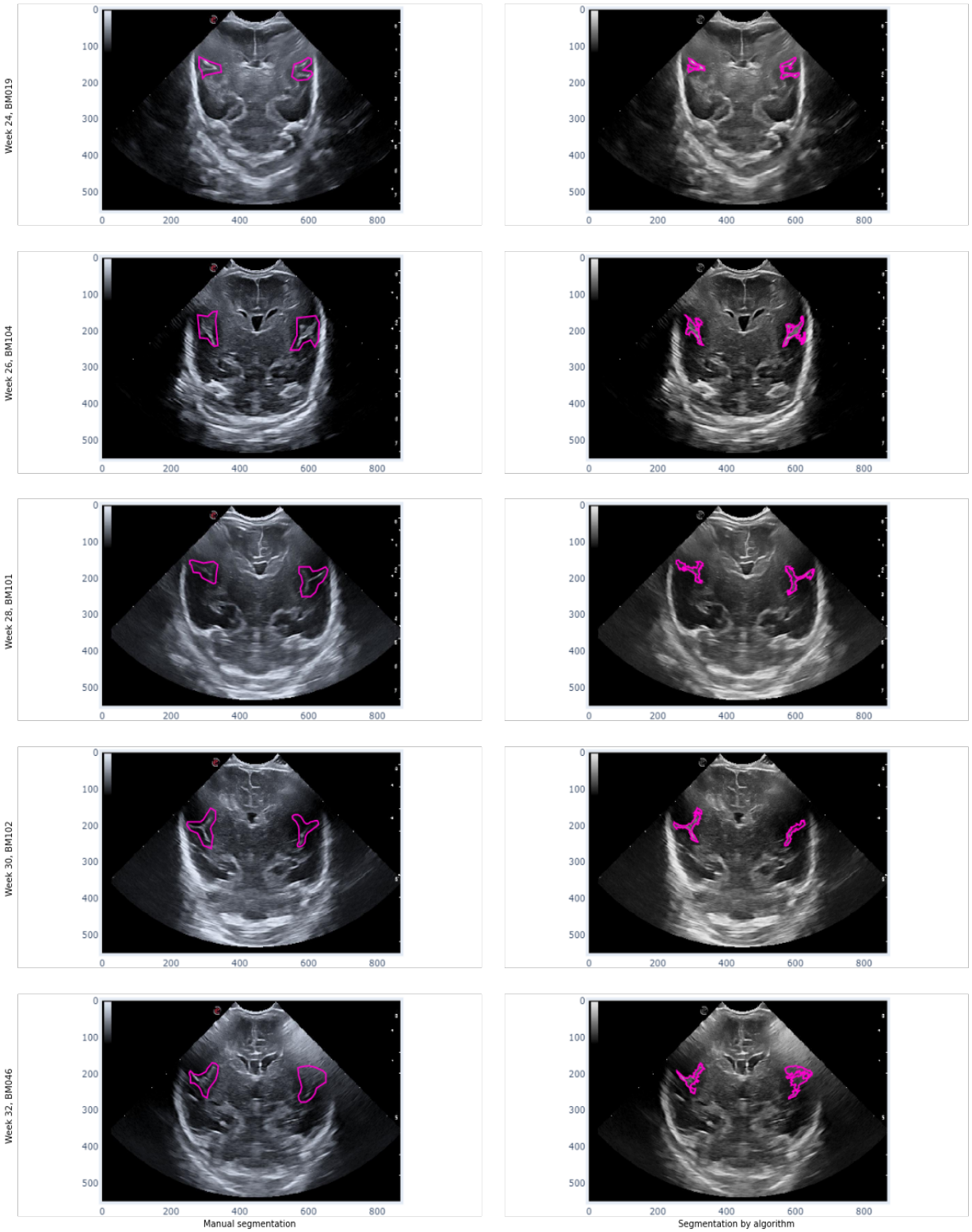

Figure 6.7. Examples of segmentation on the sulcus of sylvian, made for different babies born between 24 and 32 weeks of gestation.

Below is a table with all the available cases, at the time of publication of the document, in which there is at least one image (green colour) of a cut in the weeks in which the ultrasound was done for each one of the babies.

Baby\Week	24	25	26	27	28	29	30	31	32
BM001									
BM002									
BM008									
BM009									
BM013									
BM014									
BM017									
BM018									
BM019									
BM022									
BM028									
BM039									
BM040									
BM041									
BM046									
BM050									
BM051									
BM053									

Table 6.2.1. Cuts available for each baby in the weeks where the box is green

Baby\Week	24	25	26	27	28	29	30	31	32
BM054									
BM055									
BM056									
BM057									
BM058									
BM060									
BM062									
BM073									
BM074									
BM078									
BM079									
BM080									
BM085									
BM087									
BM091									
BM098									
BM101									
BM102									
BM103									
BM104									
BM105									

Table 6.2.2. Cuts available for each baby in the weeks where the box is green.

Baby\Week	24	25	26	27	28	29	30	31	32
BM106									
BM107									
BM108									
BM109									
BM113									
BM114									
BM115									
BM119									
BM120									
BM125									
BM126									
BM128									
BM133									
BM138									
BM148									
BM150									
BM157									
BM159									
BM160									
BM161									
BM162									

Table 6.2.3. Cuts available for each baby in the weeks where the box is green.

Baby\Week	24	25	26	27	28	29	30	31	32
BM166									
BM171									
BM172									
BM173									
BM176									
BM179									
BM180									
BM184									
BM185									
BM187									

Table 6.2.4. Cuts available for each baby in the weeks where the box is green.

7. Project Schedule

In this section we proceed to describe the Gantt chart that shows how the different parts of the project have been distributed and which is shown in the figure (...). As can be seen, the total time used is one year, starting the first meetings and gathering information in mid-September 2020 and ending with the delivery of the document on September 30, 2021.

The project can be divided into different areas, which can be divided according to the tasks and processes carried out. In the first one, it focuses on defining the objectives of the project that you want to present at the end of this and on collecting the data to learn the concepts of different subjects (brain and structure, computer vision and filters, etc.) and how to relate them to see how to get the result.

The second is more intended for the practice of different functions and algorithms of the different libraries that python contains, in addition to understanding how dash works to define how the application. This part contains a large part of programming, but where it will be applied with greater intensity and duration is in the third phase, in which we proceed to define both the design and the different callbacks of the functions defined in the application so that everything works correctly, correcting those errors that were coming out during the development process.

Finally, the information and results obtained during the previous stages are written and compiled, including the conclusions obtained and points to improve for further work.

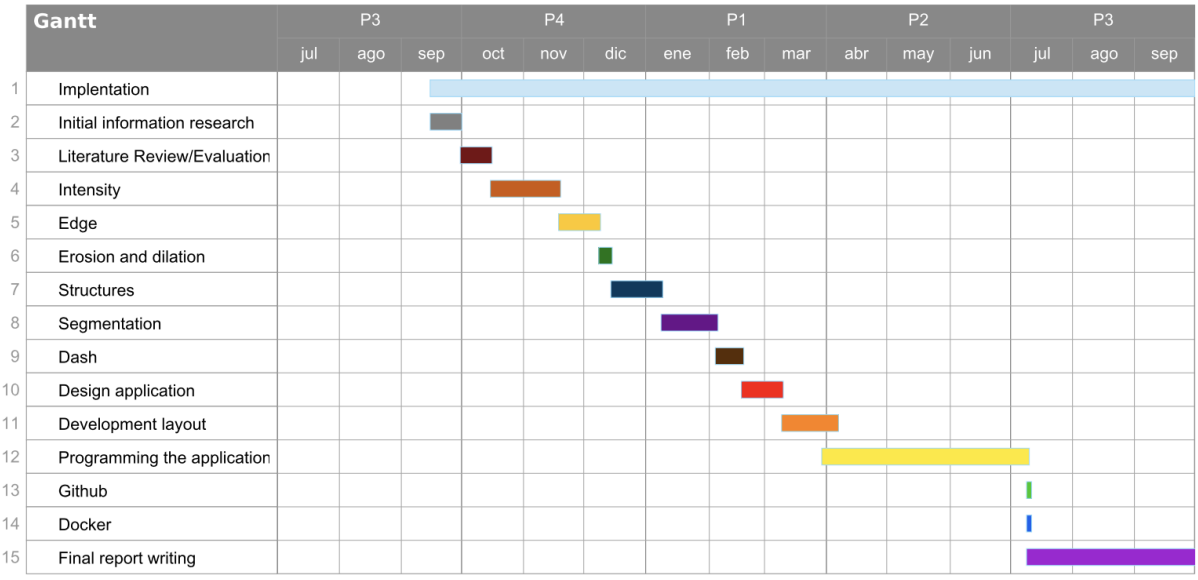

Figure 7.1. Schedule of the project

8. Environmental impact

This chapter proceeds to explain the environmental impact that has occurred throughout the project, from beginning to end, in an approximate way, thus obtaining the amount of CO2 produced.

It should be noted that the project has been carried out in the Sars-covid 19 period and that, therefore, all the work has been carried out electronically and that there have been no emissions from public transport.

Having said this, we proceed to take into account the time used from the moment in which the application begins to be programmed and designed using the Python programming language and its different libraries, until the writing of this project using Word and its subsequent delivery. It should be mentioned that, throughout the project, a MacBook Pro of 13 ” with two USB-C ports has been used, powered by a source that consumes a power of 61 watts. It is known, thanks to the schedule in the previous section, that the project has lasted approximately one year and that during the academic period 40 hours a week have been worked (periods ranging from mid-September to mid-December and from February to June, both included), while during the holiday period (considered Christmas and the months of July and August) it has been reduced to about 20 hours, approximately. Therefore, if the calculations are made, the amount of watts consumed throughout the project has been approximately 63.44 KWh.

$$Electricity\ consumption\ = 61W * 1.040h\ = 63.44Wh\ = 63{,}44\ KWh\ ;$$

This value is obtained by considering that the computer was connected to the electrical network during working hours.

Finally, and taking into account the values according to the National Markets and Competition Commission (CNMC), as of April 16, 2021, it is assumed that the emissions emitted are 0.15kg of CO2 for each KWh. So, as a consequence, throughout this project a total of 9,516 kg has been emitted during the development of the project.

$$CO2\ emission\ =\ 63{,}44\ KWh\ *\ 0{,}15Kg/KWh\ =\ 9{,}516\ CO2\ kg$$

The use to be made of the application designed by users does not fall within the development process and, therefore, has not been taken into account when calculating the energy used and, as a consequence, the emissions. Finally, it is worth mentioning that during the programming and drafting of the document, the only energy used is electricity.

9. Conclusions and future work

The creation of a brain atlas during the growth and development of premature babies after a certain week made this project possible to propose as a Final Master's Thesis. With this objective proposed by Nuria and Thais, the design, creation and development of an application based on computer vision has been carried out in order to define those structures that are important for the work they wish to carry out.

In order to properly understand what it is that you want to analyse, it has been necessary to obtain some knowledge about what the human brain is like, how it is composed, what shape and sulci it has and, finally, how it develops during the first weeks of life of the premature baby, since the furrows are forming and changing throughout the incubation period. In addition, it is also necessary to understand the characteristics of ultrasound and how it affects the noise that appears in it, since it causes complications when correctly identifying and defining the shape of the sulci and that, therefore, must be taken into account when it comes to the segmentation process.

One of the limitations of this project is the small number of images that have been provided and that have made the decision to go from an automatic application, which through a neural network was able to determine which sulci appear, to an application semi-automatic. So, in order to obtain an acceptable result with a limited number of images, a method has been designed for this project that facilitates the segmentation carried out in the algorithm used in the application. With which, it has been decided to focus the process on a semi-automatic segmentation in which it begins with the help of a person in charge of carrying out a pre-process. This action consists of indicating the area of the image in which each of the visible sulci is located and that can be distinguished so that, once this task is carried out, the designed function/algorithm is applied and a segmentation of each brain groove indicated above is obtained.

A point to highlight in the project has been to define an algorithm that allows defining the sulci that appear in the image and that allows distinguishing with respect to the noise generated by the capture of the images. This process has taken a long period of research and application of different techniques and functions on the images to finally select and determine which of them will be part of the application's algorithm.

Another of the great difficulties that this project has entailed has been when developing the application. In the design part, seen in point 5.5.2. Application project, only the structure that it must have through the screen has been defined, but the difficulty has come when writing the code, since I had no knowledge about Dash. It should be noted that the layout part was not a big setback once we understood how it works, but the part of relating the components with their respective appCallback

made its difficulty increase and these became more complicated when more input parameters are defined.

This code used in the application can be found through a GitHub repository, which has been specifically created so that it can be downloaded and executed using the necessary python commands. The other option, more recommended and without the need to install python and the libraries through the console, is using Docker, which allows you to open the repository of the designed app through the Docker Hub application.

It can be said that this first version of the application presented in the project meets the objectives set at the beginning by the interested parties in a satisfactory way, fulfilling the desire to be able to carry out a semi-automatic segmentation and to be able to correct those segments that have not been set correctly manually. In addition to being a tool that consists of a simple and intuitive handling for people who do not have programming knowledge and is easy to access, since it is opened through a browser.

9.1. Future works

However, it is true that during the final development of the application, improvements have been seen that could include new functionalities that in the future could help medical personnel to be able to perform more functions, but as the first version it is considered efficient and ready for use. These improvements that have been coming out and have been seen at the end of the project are defined through the following points:

- One thing that has been observed is that the application is loaded to the docker with a database already defined, but in case you want to add more images of one or more babies, you have to save and re-upload everything. So, an improvement for a future version would be to define a database dynamically without the need for it to be recompiled using components that allow this function and that can be found in the dash library.
- Another point is that it is possible to decide to which row / rows of the table that are defined in the manual segmentation is applied or not the segmentation algorithm in those cases in which the result of this is perceived or has not come out as expected. So, when the function is executed, the sulci selected by the user are ignored by the algorithm and the coordinates defined in the manual segmentation are kept in the second table (Coordinates of the segmentation obtained by the algorithm), since it can be considered that they are better defined.
- As mentioned in the previous point, there are cases in which the segmentation has not gone as expected. Therefore, and taking into account that this has been a first version, it is important

to bear in mind that the segmentation algorithm designed can be improved to better determine these sulci that have not been correctly defined.

- Throughout the project, the number of images that we have is not large enough to use a neural network that can be trained and allows to identify and point out the location of the points. In the event that in the future more images of premature babies could be obtained, thus increasing the database, a more intelligent algorithm (neural network) could be designed, which allows to identify the cut and label each of the sulci that appear.
- Finally, the image can be pre-processed before segmenting the image to better define the selected area or improve its definition by applying a filter on the displayed image. This could be done through a new dropdown button that allows the person who is using it to apply an algorithm to see if it produces an improvement in the areas that have doubts to define or select the groove.

10. Budget

In this last chapter, the project budget is presented, which has been broken down into different categories, according to the phases that have been explained in the Project Schedule section and the hours that have been dedicated to each of them. In addition to this, it is important to know the salary of an engineer with a Master's license and only the electrical energy used throughout the project by the computer, which in this case has been a MacBook Pro of 13" and which had already been purchased before starting work. Finally, part of the software used has no cost and, therefore, cannot be added to the total cost of the project.

The following table shows, in a broken down, the cost of each of the phases and the total value of the project

Task	Hours	Cost (€/h)	Total
Documentation and Literature Review			
Initial information research	480	25	12.000,00 €
Literature Review/Evaluation	480	25	12.000,00 €
Design and Code			
Intensity	200	25	5.000,00 €
Edge	120	25	3.000,00 €
Erosion and dilation	140	25	3.500,00 €
Structures	80	25	2.000,00 €
Segmentation	80	25	2.000,00 €
Dash	600	25	15.000,00 €
Design application	160	25	4.000,00 €
Development layout	600	25	15.000,00 €
Programming the applications callbacks	40	25	1.000,00 €
Github	4	25	100,00 €
Docker	4	25	100,00 €
Final report writing	320	25	8.000,00 €
Total	1.040		82.700,00 €

Table 10.1. Budget of the project

11. Bibliography

[1] Preterm birth, 2018. World Health Organization. [online] Available at: https://www.who.int/news-room/fact-sheets/detail/preterm-birth

[2] Neupsy Key. 2017. Cerebrum. [online] Available at: <https://neupsykey.com/cerebrum/>

[3] Van Eimeren, L., Niogi, S., McCandliss, B., Holloway, I. and Ansari, D., 2021. White matter microstructures underlying mathematical abilities in children. [online] Pubmed.gob. Available at: <https://pubmed.ncbi.nlm.nih.gov/18596611/>

[4] Doctorlib. n.d. Textbook of Clinical Neuroanatomy, 2 ed. 12. Cerebrum. [online] Available at: <https://doctorlib.info/anatomy/textbook-clinical-neuroanatomy/12.html>

[5] Gholipour, A., Rollins, C., Velasco-Annis, C., Ouaalam, A., Akhondi-Asl, A., Afacan, O., Ortinau, C., Clancy, S., Limperopoulos, C., Yang, E., Estroff, J. and Warfield, S. A normative spatiotemporal MRI atlas of the fetal brain for automatic segmentation and analysis of early brain growth. [online] Scientific Reports. Available at: <https://www.nature.com/articles/s41598-017-00525-w>

[6] Faculty.washington.edu. n.d. Neuroscience for Kids - Directions/Planes. [online] Available at: <https://faculty.washington.edu/chudler/slice.html>

[7] Lim, S., Nisar, H., Thee, K. and Yap, V., 2018. A novel method for tracking and analysis of EEG activation across brain lobes. [online] Sciencedirect. Available at: <https://www.sciencedirect.com/science/article/abs/pii/S1746809417301301>

[8] G., J., Alan Peter II, R. and Jeanty, P., 1991. Automatic segmentation of ultrasound images using morphological operators. [online] Ieeexplore.ieee.org. Available at: <https://ieeexplore.ieee.org/document/79476> [Accessed 27 September 2021].

[9] S., P. and Ravindran, R., 2021. A complete automatic region growing method for segmentation of masses on ultrasound images. [online] Ieeexplore.ieee.org. Available at: <https://ieeexplore.ieee.org/document/4155869?arnumber=4155869>

[10] Menchón-Lara, R. and Sancho-Gómez, J., 2015. Fully automatic segmentation of ultrasound common carotid artery images based on machine learning. [online] ScienceDirect. Available at: <https://www.sciencedirect.com/science/article/abs/pii/S0925231214013034>

[11] Jen-wei Kuo, Jonathan Mamou, Orlando Aristizábal, Xuan Zhao, Jeffrey A. Ketterling, and Yao Wang., 2015. Nested Graph Cut for Automatic Segmentation of High-Frequency Ultrasound Images of the Mouse Embryo. [online] Ieeexplore.ieee.org. Available at: <https://ieeexplore.ieee.org/document/7247715>

[12] Milletari, F., Ahmadi, S., Kroll, C., Plate, A., Rozanski, V., Maiostre, J., Levin, J., Dietrich, O., Ertl-Wagner, B., Bötzel, K. and Navab, N., 2017. Hough-CNN: Deep learning for segmentation of deep brain regions in MRI and ultrasound. [online] ScienceDirect. Available at: <https://www.sciencedirect.com/science/article/abs/pii/S1077314217300620>

[13] Cunningham, R., Harding, P. and Loram, I., 2017. Real-Time Ultrasound Segmentation, Analysis and Visualisation of Deep Cervical Muscle Structure. [online] ScienceDirect. Available at: <https://www.sciencedirect.com/science/article/abs/pii/S1077314217300620>

[14] Xu, Y., Wang, Y., Yuan, J., Cheng, Q., Wang, X. and Carson, P., 2019. Medical breast ultrasound image segmentation by machine learning. [online] ScienceDirect. Available at: <https://www.sciencedirect.com/science/article/abs/pii/S0041624X18302257>

[15] Xin Yang , Lequan Yu , Shengli Li, Huaxuan Wen, Dandan Luo, Cheng Bian, Jing Qin , Dong Ni, and Pheng-Ann Heng., 2018. Towards Automated Semantic Segmentation in Prenatal Volumetric Ultrasound. [online] Ieeexplore.ieee.org. Available at: <https://ieeexplore.ieee.org/document/8418398>

[16] Jeya Maria Jose Valanarasu, Rajeev Yasarla, Puyang Wang,, Ilker Hacihaliloglu and Vishal M. Patel., 2020. Learning to Segment Brain Anatomy From 2D Ultrasound With Less Data. [online] Ieeexplore.ieee.org. Available at: <https://ieeexplore.ieee.org/document/9113237>

[17] Edeza, T., 2021. Image Processing with Python — Unsupervised Learning for Image Segmentation. [online] towardsdatascience. Available at: <https://towardsdatascience.com/image-processing-with-python-unsupervised-learning-for-image-segmentation-90ebd23d91a4>

[18] GeeksforGeeks. 2021. Python | Thresholding techniques using OpenCV | Set-1 (Simple Thresholding) - GeeksforGeeks. [online] Available at: <https://www.geeksforgeeks.org/python-thresholding-techniques-using-opencv-set-1-simple-thresholding/>

[19] Blasco, J., Aleixos, N. and Moltó, E., 2007. Computer vision detection of peel defects in citrus by means of a region oriented segmentation algorithm. [online] Sciencedirect. Available at: <https://www.sciencedirect.com/science/article/abs/pii/S0260877406007242>

[20] Stack Overflow. n.d. 'plotly-dash' tag wiki. [online] Available at: <https://stackoverflow.com/tags/plotly-dash/info>

[21] Rabanaque, D., Benitez, R. & Mata, C. (2021). Sant Joan de Déu App (Version 1), GitHub [Software] Available at: https://github.com/Derther/app_TFM

[22] Rabanaque, D., Benitez, R. & Mata, C. (2021). Sant Joan de Déu App (Version 1), DockerHub [online] Available at: https://hub.docker.com/repository/docker/derther/app_segmentation

12. Annex A

This section includes all the files and instructions used to build the Docker repository. To do this, it has been divided into two parts: Annex A1 and A2. In the first one, the Dockerfile.txt file necessary to build the image and which must be included in the project folder is exposed. Annex A2 shows the necessary instructions to build the image and create the container through the device's console, as well as other commands of interest.

12.1. Dockerfile

```
FROM python:3

#Workdir on the image (unix System)

WORKDIR /users/david/Desktop/app

# Different python libraries to install

COPY requirements.txt ./

RUN pip install --no-cache-dir -r requirements.txt

RUN apt-get update && apt-get install -y python3-opencv

RUN pip install opencv-python

#Copy everything in the folder to the work directory

COPY . .

# Comunication port between the dash application and the host machine

EXPOSE 8050

#Execute with python the application

CMD ["python", "./app_segmentation_docker.py"]

#Build instruction to launch from terminal

# docker build . -t "dicom-viewer"

# test intructions -- DO NOT UNCOMMENT

#CMD export DISPLAY =":0"

#-e DISPLAY=$DISPLAY \

#-v /tmp/.X11-unix:/tmp/.X11-unix:rw
```

12.2. Instructions to run docker

LOGIN

$ docker login

$ docker logout

BUILD A LOCAL IMAGE

Go to the folder of the main project and execute:

$ docker build --tag dicom-viewer .

RUN DOCKER CONTAINER

$ docker run --publish 8050:8050 dicom-viewer

NOTE: Important to add in your docker application

if __name__ == "__main__":

app.run_server(host='0.0.0.0', port=8050, debug=DEBUG)

PUSH IMAGE TO HUB (Remote repository)

$ docker logout

$ docker tag dicom-viewer user_name/dicom-viewer:dicom-viewer

$ docker login

$ docker push user_name/dicom-viewer:dicom-viewer

NOTE: Change the name of the hub user

COMMAND TO LIST IMAGES

$ docker images

COMMAND TO LIST CONTAINERS

$ docker ps -a

STOP ALL DOCKER IMAGES RUNNING

$ docker system prune

13. Annex B

At this point we proceed to show the GitHub repository, in which are the files necessary to make the application work. Keep in mind that, for the application to run, you must have a Python interpreter installed and follow the instructions in the README file.

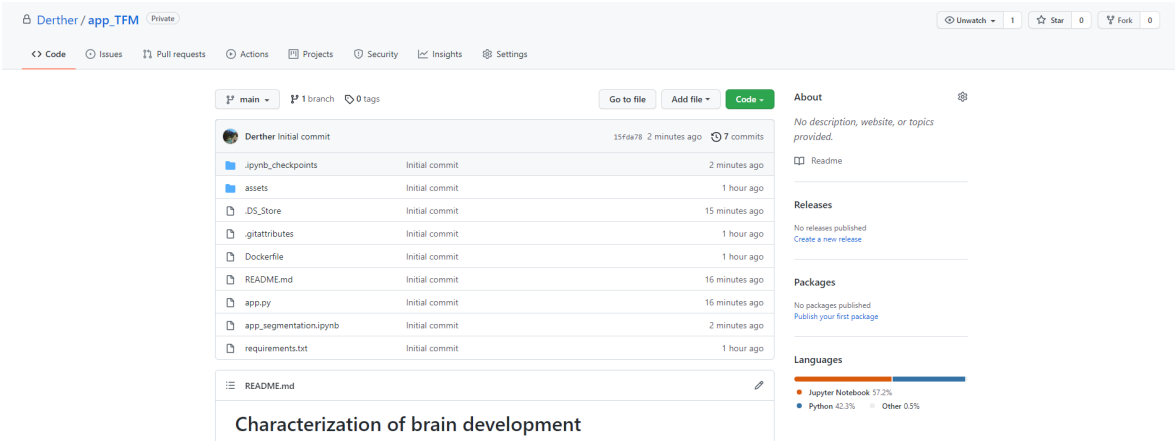

Figure 13.1. Screenshot of the Characterization of brain development repository on GitHub [21]