Generación de código mediante HDL Coder para procesamiento y clasificación de imágenes biomédicas

Hermenegildo Fabregat Marcos

MÁSTER EN INGENIERÍA INFORMÁTICA. FACULTAD DE INFORMÁTICA
UNIVERSIDAD COMPLUTENSE DE MADRID

Trabajo Fin Máster en Ingeniería informática

5 de julio 2017

Calificación obtenida: 8

Directores:

Alberto del Barrio García
Guillermo Botella Juan

Generación de código mediante HDL Coder para procesamiento y clasificación de imágenes biomédicas

Hermenegildo Fabregat Marcos

MÁSTER EN INGENIERÍA INFORMÁTICA. FACULTAD DE INFORMÁTICA
UNIVERSIDAD COMPLUTENSE DE MADRID

Trabajo Fin Máster en Ingeniería informática

5 de julio 2017

Calificación obtenida: 8

Directores:

Alberto del Barrio García

Guillermo Botella Juan

Autorización de difusión

Hermenegildo Fabregat Marcos

5 de julio 2017

El abajo firmante, matriculado en el Máster en Ingeniería Informática de la Facultad de Informática, autoriza a la Universidad Complutense de Madrid (UCM) a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a su autor el presente Trabajo Fin de Máster: *Generación de código mediante HDL Coder para procesamiento y clasificación de imágenes biomédicas*, realizado durante el curso académico 2016-2017 bajo la dirección de Guillermo Botella Juan y Alberto del Barrio García, ambos miembros del equipo docente de la facultad de Informática de la Universidad Complutense de Madrid, y a la Biblioteca de la UCM a depositarlo en el Archivo Institucional E-Prints Complutense con el objeto de incrementar la difusión, uso e impacto del trabajo en Internet y garantizar su preservación y acceso a largo plazo.

Firmado: Hermenegildo Fabregat Marcos

Resumen en castellano

Durante la época de la crisis del *software* se concibió la idea de *software* como arte. Esta idea estaba vinculada a la complejidad que suponía realizar ciertos desarrollos y mantener el código generado. Esta situación cambió con la aparición de las técnicas de ingeniería del *software* que conocemos actualmente, haciendo posible a día de hoy concebir la expresión de generación automática de código y los entornos asociados a esta forma de trabajo. Este proyecto se basa en la utilización de herramientas de generación de código de forma automática con el objetivo de implementar un sistema para el tratamiento y procesado de imágenes microscópicas y en ultima instancia, portar este sistema a un entorno de bajo consumo energético. El dispositivo con el que se ha realizado la experimentación ha sido la FPGA Cyclone V de Altera.

Diferentes publicaciones han abordado los problemas que en este documento se tratan, desde la mejora de la calidad de las imágenes ruidosas hasta el reconocimiento de formas. Sin embargo, en este trabajo de fin de master estas tareas se llevan a cabo mediante el uso de un marco de trabajo basado en Matlab y *Simulink*, sumando a estos el *toolbox* HDL Coder en combinación con el entorno Quartus II.

Palabras clave

Procesamiento de imágenes, *HDL Coder*, *Simulink*, FPGA.

Abstract

During the software crisis period, the concept of software as art was a fact, considering that developing and supporting software were both very difficult tasks. This situation changed when current well-known software engineering techniques appeared, making it possible to even automatically generate code by using certain tools. This project is about the use of tools for automatic code generation to implement an image processing and enhancing system. The target hardware will be a low-end FPGA: the Cyclone V by Altera.

Many publications have tackled the aforementioned problems to improve the quality of noisy images and recognising shapes. Nevertheless in this MSc. Thesis these tasks will be performed by using a framework based on Matlab, Simulink as well as the HDL Coder toolbox in combination with the Quartus II environment.

Keywords

Digital image processing, HDL Coder, Simulink, FPGA.

Agradecimientos

El presente trabajo fue realizado con la supervisión académica de Alberto del Barrio García y Guillermo Botella Juan. Me gustaría aprovechar la oportunidad para agradecerles su apoyo y colaboración.

Por otro lado, este trabajo ha sido posible gracias a la implicación de Angel Sierra (Matlab, *University program*), al aporte del *Intel® FPGA University Program* y al proyecto DESCARTES (Santander UCM, referencia: PR26/16-20B-1).

En última instancia, me gustaría agradecer el apoyo de familiares y amigos.

¿Por qué esta magnífica tecnología científica, que ahorra trabajo y nos hace la vida mas fácil, nos aporta tan poca felicidad? La repuesta es está, simplemente porque aún no hemos aprendido a usarla con acierto.

(Albert Einstein, 1879-1955)

Índice general

Índice de figuras

Índice de tablas

Índice de *reports*

Capítulo 1

Introducción

1.1. Motivación

Desde la primera publicación de *W.K. Pratt's Digital Image Processing* en 1978, avances en algoritmos y métodos de procesado de imágenes han transformado radicalmente el coste tecnológico de llevar a la práctica numerosas aplicaciones de procesado de imágenes. Este trabajo gira alrededor de diversos campos: *la generación de código automático*, *el reconocimiento de patrones* y *el procesamiento digital de imágenes*.

La generación de código de forma automática viene de la mano de la denominada crisis del *software* y en muchas ocasiones de entornos de diseño asistido de *software*. Este concepto tomó gran importancia por la intervención de Friedrich L. Bauer en una conferencia elaborada por la OTAN (Organización del Tratado del Atlántico Norte) en Múnich entorno al año 1968[26]. En esta conferencia se comentaron muchas dificultades a la hora de realizar *software*, dificultades que hacían que éste se entendiera como arte y no como algo que no produjera ambigüedad alguna. Una de las conclusiones de esta conferencia es que el *software* debía de ser sostenible y para ello debía de ser sobre todo legible. A raíz de esta conferencia se creó una nueva rama en ingeniería, la denominada ingeniería del *software*. Esta rama es la encargada de dar un enfoque sistemático a la generación de programas informáticos para así garantizar la sostenibilidad de los mismos. Tal fue el éxito de la conferencia realizada en Múnich, que los organizadores solicitaron y obtuvieron patrocinio de la OTAN

para una segunda conferencia. Ésta se celebró un año más tarde en Roma. Actualmente, si bien se siguen discutiendo muchos de los temas comentados en esta primera conferencia de la OTAN, estamos ante un tipo de soluciones diferentes a las comentadas, soluciones entre las que podemos encontrar las basadas en entornos de generación automática de código y diseño de *software* asistido. Si bien en cierto modo no existe relación aparente entre conceptos como ingeniería del *software* y generación automática de código, éstos son conceptos muy ligados por el aporte de una metodología sistemática a la generación de *software*.

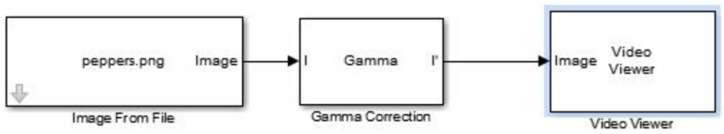


Figura 1.1: Ejemplo de sistema con *Simulink*.

Suponiendo que en la mayoría de los casos, un programa *software* se puede definir como un conjunto de instrucciones verificables una a una y en subconjuntos, los entornos de diseño de *software* asistido se basan en la idea de simplificar lo máximo posible el proceso de mantenimiento de código y para ello otorgan al usuario una interfaz simple con la cual es capaz de crear métodos complejos mediante el uso de funciones ya verificadas. Estos entornos se suelen encontrar con soluciones para la generación de código de forma automática como es el caso de *Simulink*[37] y *HDL Coder*[36]. La combinación de *Simulink* y *HDL Coder*, ambas herramientas de la empresa *Mathworks*, es capaz de generar código HDL a partir de modelos basados en métodos y funciones prediseñadas y verificadas. En la figura 1.1 se puede ver un ejemplo de modelo *Simulink* exportable como código HDL. El esquema mostrado indica un flujo en el que se carga una imagen desde un fichero y se visualiza ésta tras aplicar una corrección del contraste. Este trabajo trata del uso de estas herramientas para la restauración y procesado de imágenes mediante el uso de una FPGA.

Si nos fijamos en el amplio contexto de aplicación de las técnicas de procesamiento de imá-

genes tenemos que son muchos los avances que se han producido en las diferentes implementaciones de éstas. Desde la primera publicación de W.K. Pratt's Digital Image Processing en 1978[24], avances en algoritmos y métodos de procesado de imágenes han transformado ostensiblemente el coste tecnológico de llevar a la práctica numerosas aplicaciones. Un ejemplo de aplicación de estas técnicas lo podemos encontrar en la figura 1.2, donde podemos ver la aplicación de varios filtros y la aplicación de técnicas basadas en umbrales.

Figura 1.2: Procesamiento de imágenes, Lena.

Aunque el auge en el estudio del procesamiento digital de imágenes se puede decir que está fechado cuando el término *digital* empezó a tomar valor en la sociedad, uno de los pilares en los que se fundamentan casi todas las técnicas para la mejora o tratamiento de imágenes son los estudios de Jean-Baptiste-Joseph Fourier (1768-1830), fechados mucho antes de la denominada *era digital*. Extraído este fragmento de uno de sus teorema, éste viene a decir lo siguiente:

> "La variación de la irradiancia o brillantez de una imagen, medida a lo largo de una dirección cualquiera es una función que se puede representar mediante el teorema de Fourier, con una suma de distribuciones senoidales de varias frecuencias."

Una vez establecida una de las principales bases del procesamiento de imágenes, son muchos y muy importantes los estudios que han seguido esta línea, estudios que van desde el mero procesamiento de una imagen para su mejora o reconstrucción hasta la extracción o inferencia de información a partir de una imagen. Si miramos dentro del mundo del reconocimiento de patrones, en éste hay un gran hueco para el procesamiento digital de imágenes, e.g. la detección de rostros[21] o el reconocimiento de patrones en texto manuscritos[16].
En los siguientes apartados de este capítulo se van a desarrollar los distintos objetivos del proyecto y la estructura del documento.

1.2. Objetivos y estado del arte

Como ya se ha comentado, son muchos y variados los avances que se han conseguido en el procesamiento digital de imágenes. Es común decir que el procesado de imágenes puede ser visto desde dos puntos de vista o categorías:

1. **Restauración de imágenes.** La restauración de imágenes tiene el objetivo de recuperar una imagen degradada haciendo uso de modelos de funciones de degradado y de la imagen original (salvo en los casos de reconstrucción ciega). Estos modelos son muy rigurosos pero requieren de mucha parametrización, hecho que hace que en la mayoría de casos tiendan a generar soluciones locales para un problema concreto.

2. **Mejora de imágenes.** Los algoritmos de mejora de la calidad de una imagen normalmente hacen uso de métricas de calidad subjetivas para producir una imagen visualmente más correcta, e.g. *peak-signal-noise-ratio* (PSNR). Este tipo de aproximaciones son computacionálmente más viables que algoritmos de deconvolución ciega.

Con el objetivo de obtener imágenes de mejor calidad, los investigadores han desarrollado sistemas para aumentar imágenes manteniendo una alta resolución. En otras palabras, estos sistemas son capaces de diferenciar entre dos puntos pequeños y similares desde un punto

de vista lejano[20]. Aunque este proyecto se sitúa en similares lineas de trabajo, guarda significativas diferencias ya que el actual proyecto no utiliza el concepto de super-resolución y sí incluye técnicas de aprendizaje automático y procesamiento en *hardware*.
También existen recientes avances en lo que se refiere a captura[27] y procesado[23] que permiten mejorar la extracción de tanta información como sea posible de las imágenes obtenidas de un microscopio. Trasladar el conocimiento de científicos expertos en una materia ha sido desde siempre una de las principales inquietudes de las diferentes comunidades de ingenieros. Cuanto más especializada es la tarea que se intenta reproducir más valor adquiere este reto debido a las dificultades que entraña incluso para los diferentes expertos en la materia, e.g identificación y clasificación de bacterias en imágenes microscópicas (figura 1.3).

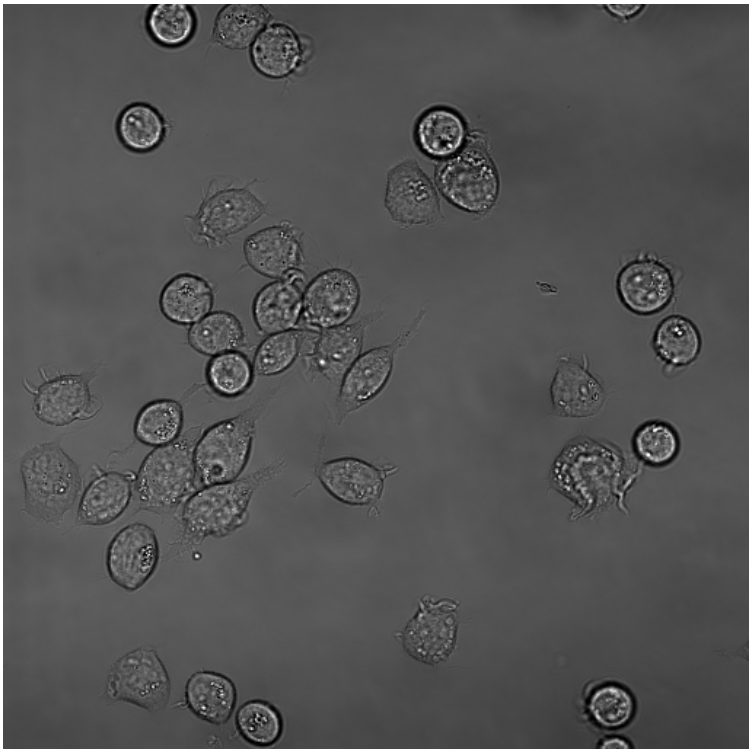


Figura 1.3: Imagen sin procesar obtenida mediante microscopio óptico.[1]

La segmentación de imágenes de células microscópicas constituye un reto donde normalmente las imágenes están corruptas, emborronadas o contienen mucho ruido[13]. Automatizar

[1]https://goo.gl/UOdtR5

este tipo de tareas supone un gran aporte para los diferentes grupos de expertos ya que simplificar una tarea rutinaria con altos requisitos técnicos da lugar a una reducción significativa del esfuerzo que supone su consecución. Existen diferentes estudios que respaldan la idea de que aplicar las aproximaciones clásicas generales de segmentación de imágenes al ámbito biomédico no es la solución más acertada[19,32]. Esta idea no es algo que haya aparecido con la aplicación de técnicas de *machine learning* a este ámbito, es algo demostrado que no existe una solución general para todos los escenarios de aprendizaje automático. Para afrontar los retos antes mencionados pueden ser aplicadas diferentes técnicas de procesado de imágenes, bien sea para enfatizar características valoradas como importantes o para corregir posibles irregularidades presentes en la imagen. Una de las técnicas más típicas es la aplicación de un filtro mediano para la eliminación del ruido denominado *ruido de sal y pimienta*[11,25]. Existen otros tipos de irregularidades que también hacen que sean válidas las técnicas de deconvolución de señales aplicadas a imágenes. Estas técnicas tratan de restaurar una imagen emborronada mediante la obtención y aplicación de una matriz correctora a la imagen para restablecer el punto de vista[29]. Como ya se ha comentado, además de las técnicas para la reducción de irregularidades, existen otras técnicas tales como la binarización o umbralización[4], el filtrado morfológico[38] y la diferencia de Gausianas[15,31] que facilitan pistas e información relevante para la etapa de extracción de patrones.

Como ya se ha comentado, una vez eliminados los nombrados efectos no deseados y centrándonos más en el contexto del actual proyecto, grupos de expertos tales como biólogos necesitan de la automatización de ciertos procesos como pueden ser identificar ciertas formas, movimientos, etc. para así dirigir y validar sus experimentos. La automatización de este tipo procesos es una tarea critica para ellos. Es por esto que resulta interesante la utilización de algoritmos de reconocimiento de patrones[18,34,39].

El problema de aplicar métodos de este tipo a imágenes biológicas reside en la gran cantidad de tiempo de proceso que requieren en comparación con otro tipo de imágenes. Esto es debido a que éstas suelen ser imágenes de muy alta resolución. Es aquí donde cobran mucha

importancia los métodos de aceleración *hardware* como son el uso de GPUs[17] o FPGAs[3].
La simulación *Hardware-in-the-loop* (HIL) es una técnica usada para el desarrollo y comprobación de sistemas embebidos en tiempo real. Actualmente, los estudios y simulaciones basadas en técnicas *Hardware-in-the-Loop* (*HIL*) ofrecen una buena relación precisión-coste[2,22,28]. Por un lado éstas son normalmente más precisas que soluciones basadas únicamente en *software* y por otro lado, son más baratas que soluciones estrictamente *hardware*.
Acercándonos un poco más a la temática del proyecto, simulaciones del tipo *FPGA-in-the-loop* (*FiL*)[8] son un ejemplo de HIL. Estas proveen la capacidad de utilizar *software* para medir y calibrar los diseños *hardware* para cualquier lenguaje de descripción de *hardware* (HDL).
Aunque son muchos los aportes y estudios que aplican *FPGA-in-the-loop*[14,22] y la herramienta *HDL Coder* como piedra angular, este trabajo se sitúa sobre la linea del trabajo de los autores *J. C. T. Hai et al.*[9] en el cual plantean el uso de una *FPGA* y de un entorno de generación automática de código para el procesado de vídeo e imagen. En este caso ellos prueban un algoritmo para la estimación de la altura de un sujeto mediante un método de detección de bordes.
El conjunto de técnicas que se han implementado y simulado en este proyecto abarca desde técnicas de mejora de la calidad de imágenes hasta técnicas de reconocimiento y clasificación de formas. En resumen, el *pipeline* generado mediante *Simulink* tiene como objetivo procesar imágenes obtenidas mediante microscopio haciendo uso de una *FPGA Altera Cyclone V* (herramienta de bajo coste) y la técnica *FPGA-in-the-Loop*. Lo que se persigue en este proyecto es ayudar a biólogos en el procesamiento de imágenes obtenidas mediante microscopio.

1.3. Metodología y plan de trabajo

La metodología seguida durante el desarrollo del proyecto ha sido guiada en función de los resultados. Tanto el *software* utilizado para la generación del código HDL como la *FP-*

GA han impuesto limitaciones al desarrollo del proyecto. En un principio, se han intentado desarrollos complejos e iterativos en los que hubiera un alto desempeño algorítmico por parte de la *FPGA*. Esto ha sido imposible de llevar a cabo debido a limitaciones de memoria impuestas por el modelo de *FPGA* utilizado. Este hecho se comentará en profundidad en siguientes capítulos del documento.

El proyecto resultante sigue el esquema abstracto de la figura 1.4, donde se puede apreciar que todas las fases del proyecto basadas en *Simulink* y con posibilidades dentro de *HDL Coder* tienen tanto una versión estrictamente *software* como una versión *hardware*.

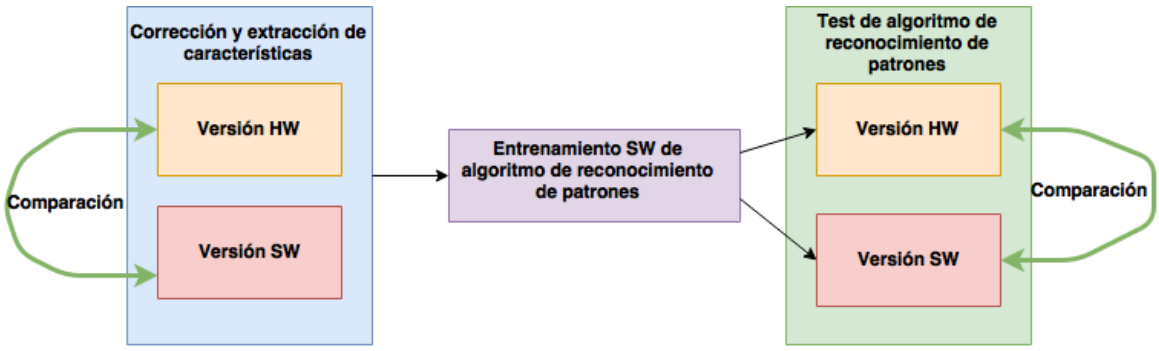


Figura 1.4: Esquema abstracto del proyecto.

Por las limitaciones inherentes al uso de operaciones de coma flotante en entornos *hardware* se ha realizado un estudio de precisión para comprobar la desviación de los resultados *hardware* en comparación con la versión *software*. Además del estudio de precisión, se han realizado comparativas de rendimiento para comprobar las diferencias en cuanto a tiempos de ejecución entre las dos versiones.

Es interesante destacar en este apartado que tanto en la versión del *HDL Coder* como en la versión *software* se utilizan las funciones ofrecidas por *Simulink* para garantizar la correcta equidad a la hora de realizar la comparativa.

1.4. Estructura del documento

El documento está organizado de la siguiente forma:

Capítulo 2. Este proyecto se basa en herramientas de computación científica tanto para la realización de las distintas simulaciones *software* como para la generación de código automático. Para la ejecución del código generado se ha utilizado una *FPGA Altera Cyclone V*. En este capítulo se comentarán las herramientas utilizadas y el estudio teórico alrededor de este proyecto. Ademas se van a comentar las características destacables de los modelos *Simulink* diseñados.

Capítulo 3. En este capítulo se van a mostrar las comparativas de los resultados obtenidos a nivel precisión y en cuanto a rendimiento para los distintos modelos generados, tanto en procesamiento digital de imágenes como algoritmos de aprendizaje automático. Este capítulo está dividido en tantas fases como fases tiene el proyecto.

Capítulo 4. En esta parte del documento figuran las conclusiones obtenidas tras la realización del proyecto y las posibles líneas de trabajo futuro.

Chapter 1

Introduction

1.1. Motivation

Since the first publication of *W.K. Pratt's Digital Image Processing* in 1976, advances in algorithms and in methods of image processing have substantially transformed the technological cost of carrying into effect numerous applications of image processing. This project turns around various fields: automatic generation of code, pattern recognition and digital image processing.

Automatic generation of code comes hand in hand with the software crisis and in many occasions from environments of assisted software design. This concept became important through Friedrich L. Bauer's intervention at a Conference organized by NATO in Munich around 1968[26]. At this conference a great number of difficulties at the time of carrying out software were commented and such difficulties made the same to be regarded as an art and not as something that might give rise to ambiguity. One of the conclusions of the said conference was that software must be sustainable and therefore it must be above all legible. From that conference, a new branch in engineering was created, the so called "software engineering", mainly intended to provide a systematic approach to the generation of computer programs in order to guarantee the sustainability of same, although in a certain sense, software engineering and automatic code generation are very well connected concepts by providing a systematic methodology to the software generation. The conference held in

Munich was so successful that its organizer requested and obtained NATO's patronage for the second conference which actually took place one year later in Rome. Although many of the issues arising in the first NATO conference are still being discussed, we are currently facing types of solutions which are different from previous ones and which solutions we can find in environments of automatic code generation and in assisted software design.

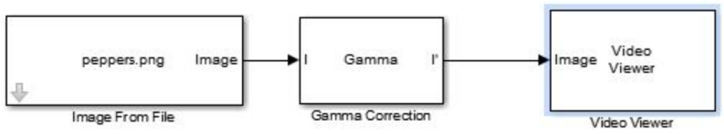


Figure 1.1: Simulink system example.

Assuming that in most cases, a software program can be defined as a set of instructions which can be verified one by one and in sub-settings, environments of assisted software design are based on the idea of simplifying as much as possible the code maintenance process and for such a purpose the user is empowered with a simple user interface with which he will be able to create complex methods by means of using already verified functions. Such environments are usually found with solutions for the code generation in an automatic manner as it happens in the case of *Simulink*[37] and HDL Coder[36]. Adding Simulink to HDL Coder, both tools belonging to Mathworks company, it is possible to generate HDL code based on the composition of certain methods which are already implemented and verified. In figure 1.1 we can see an example of a Simulink model which is exportable as HDL code. This project deals with the use of such tools for the restoration and processing of images by using an FPGA.

If we observe the ample context of application of image processing techniques we see the great number of advances which have taken place in the various implementations of such techniques since the first publication of W.K. Pratt's Digital Image Processing way back in 1978[24]. Advances in algorithms and in methods of image processing have substantially trans-

formed the technological cost of carrying into practice numerous applications. An example of application of these techniques may be found in 1.2 where we can see the application of several filters and the application of techniques based on thresholdings and binarizations.

Figure 1.2: Image processing, Lena.

Although the increase in the study digital image processing may be dated when the term digital began to acquire value in our society, one of the pillars which supports almost all techniques for the improvement or treatment of images are the studies by Jean-Baptiste-Joseph Fourier (1768-1830) many years before the so called "digital era". Excerpting this quotation from the same theorem which reads as follows:

> "Variation of irradiance or brightness of an image being measured along any direction, is a function which can be represented by means of Fourier's theorem with an addition of senoidal distributions of various frequencies."

Once of the main bases of image processing is established, there are many and very important studies which have followed this path, studies which range from the mere crude image processing up to the extraction or generation of information from an image. If we look

deeper into the world of pattern recognition, there is a big slot for recognition of patterns within images, for example: face detection[21] or pattern recognition in written texts[16].
In the following paragraphs of this chapter the different objetives of the project and the structure of the document are going to be developed.

1.2. Objetives and state of the art

As set out hereinbefore, there are many and varied advances made in digital image processing. It is usual to state that image processing may be seen from points of view or categories:

1. **Image restoration.**The aim of image restoration is to recover a degraded image making use of models of degrading functions and of the original image (save in the cases of blind reconstruction). Such models are very rigorous but require much parametrization, which fact makes that in most of the instances they tend to generate local solutions for a specific problem.

2. **Image enhancement.** Algorithms of image quality improvements usually make use of subjective quality metrics to produce an image which is more correct visually, i.e. *peak-signal-noise-ratio* (PSNR). This type of approach is computationally more viable than algorithms of blind deconvolution.

With the objective of obtaining better quality images, researchers have developed systems to enlarge images maintaining a high resolution. In other words, such systems are capable of differentiating two small similar points from a very distant point of view[20]. Although this project stands on similar working guidelines, there are significant differences because this present project does not make use of the super-resolution concept and it does include machine learning techniques and hardware processing.
There also exist recent advances in capture[27] and processing[23] which allow the improvement of extraction of as much information as possible from images obtained with a microscope.

One of the major objectives that different communities of engineers have always tried is the transfer of knowledge of scientific experts in certain fields. The more specialized the task is for reproduction the higher its value due to the difficulties involved even for experts in such fields. For example, identification and classification of bacteria in microscopic images, See Figure 1.3.-. Image segmentation of microscopic cells is a sheer challenge as images are usually corrupt, blurred or contain a lot of noise (figure 1.3).

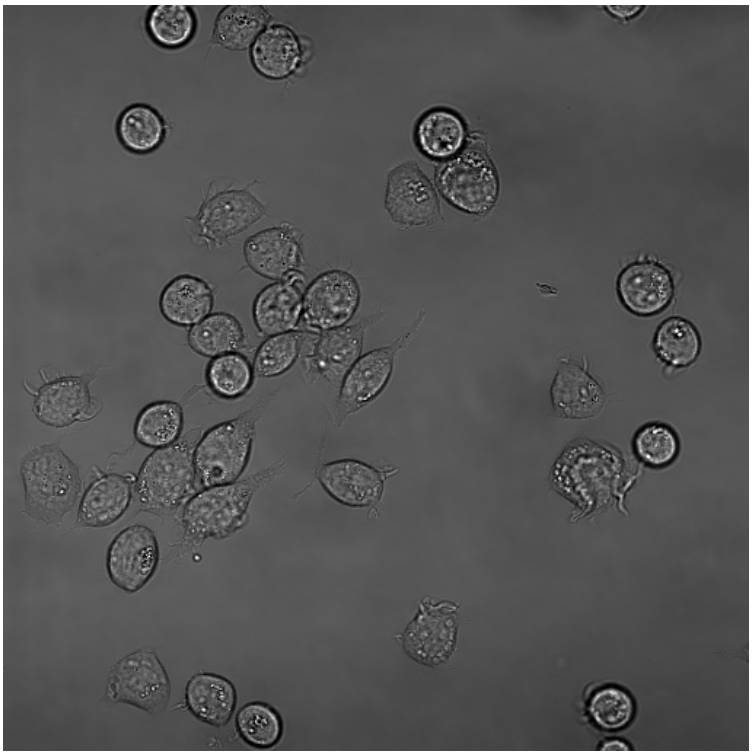


Figure 1.3: Unprocessed Image obtained making use of an optic microscopy.[1]

Image segmentation of microscopic cells is a challenge as images are usually corrupt, blurred or they contain a lot of noise[13]. Automatization of such tasks implies a great advance and support to the different groups of experts. Their task is simplified through high technical requirements thus reducing their effort to achieve their goal or objectives. There are different studies which support the idea that applying the typical approximations of image segmentation from the general environment to the biomedic environment is not the best

[1]https://goo.gl/UOdtR5

solution[19,32]. This idea is not something that has appeared with the application of machine learning techniques in this environment. It is well demonstrated that there is no solution which may be used as a "Swiss Army pocket knife" for any problem of machine learning. Different techniques of image processing may be applied in order to confront the above-mentioned challenges, whether to emphasize characteristics being valued as important or to amend possible irregularities being present in the image. For example, if the environment where the images are captured is hostile, it is possible that irregularities may appear. One of the most typical techniques is the application of a medium filter to eliminate noise known as "salt and pepper"[11,25]. There are other types of irregularities which make also valid techniques of deconvolution of signs applied to images. Such techniques try to restore a blurred image by obtaining and applying a correcting matrix to the image in order to re-establish the point of view[29]. As already mentioned, besides techniques for the reduction of irregularities, there are other techniques such as binarization or thresholding[4], morphological filters[38] and the difference of Gaussians[15,31] which make the extraction of patterns from a group or cluster of images a much easier task.

Once the so called non desired effects are eliminated and focusing on the context of this present project, biologists need automatization of certain processes, identification of certain forms, movements, etc. and thus to direct and validate their experiments. Automatization of this process is a critical task that they encounter. Therefore making use of algorithms of pattern recognition is very interesting[18,34,39].

The problem of applying this type of methods to biological images lies in the large amount of process time that they require in comparison with other types of images. Thus is due to the great amount of information that they contain as being images of a very high resolution and here the methods of hardware acceleration such as the use of GPUs[17] or FPGAs[3] are of great importance.

HIL or Hardware-in-the-Loop simulation is a technique used for the development and checking of systems wich are embedded in real time. Currently the studies and simulations based

upon techniques such as HIL or Hardware-in-the-Loop offer a good precision-cost relationship[2,22,28]. On one hand they are usually more precise than solutions based solely on software and on the other hand, they are cheaper than strict hardware solutions.

Focusing a little more on the subject of this project, simulations of the FPGA-in-the-Loop (FIL)[8] type are an example of HIL. They provide the capacity of using software to measure and calibrate hardware designs for any language of hardware description (HDL).

Although there are many contributions and studies which apply FPGA-in-the-Loop[14,22] as well as the tool HDL Coder as corner stones, this project is in line with the work of authors such as *J. C. T. Hai et al.*[9] who propose the use of an FPGA and an automatic generation code environment for the process of both video and image. In this case they use an algorithm for estimation of height of a subject based upon a method of edge detection.

The various techniques which have been implemented and simulated in this project range from techniques of image quality improvement up to recognition and shape classification techniques. The pipeline so generated aims at processing images obtained by means of a microscope making use of an FPGA Altera Cyclone V and the FPGA-in-the-Loop technique to offer help to biologists in image processing obtained with microscopy.

1.3. Methodology and work plan

The methodology followed during the project development has been evolutive and guided according to result functioning. Both the software used for the HDL code generation as that of the FPGA, have imposed limitations in the project development. Initially more complex and iterative developments have been tried with a high algorithmic performance by the FPGA, however memory limitations imposed by the FPGA model which have made it almost impossible.

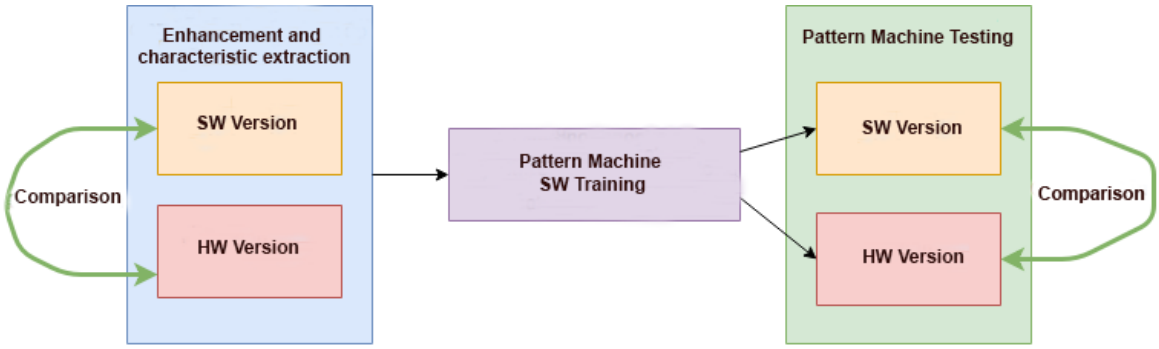


Figure 1.4: Abstract sketch of the project.

The resulting project follows the abstract sketch of figure 1.4 where we can see that all phases of the project based on Simulink and with possibilities within HDL Coder have both a strictly software version as well as a hardware version.

Due to limitations inherent to the use of hardware in operations of "floating coma" a precision study has been carried out to double check the deviation of the hardware results in comparison with the software version. Besides the precision study performance, comparisons have been carried out in order to double check the time differences between both versions.

It is interesting to remark in this respect and in view of the sketch of the above figure that both the HDL Coder version as well as the software version make use of functions offered by Simulink in order to guarantee the correct equity at the time of carrying out the aforesaid comparison.

1.4. Document structure

The document is organized in the following manner:

Chapter 2. This project is based on tools of scientific computation both for the carrying out of various software simulations as well as for the automatic code generation. For the execution of the generated code an FPGA Altera Cyclone V has been used. In this chapter we are going to set out the major features of the tools which have been used and each one of the Simulink systems and sub-systems forming the software and hardware network of this project will be set out.

Chapter 3. In this chapter we are going to show the comparison of the results obtained at a precision level and also in the performance in the different generated models both in the digital image process and in algorithms of automatic learning. This chapter is divided into as many phases as the project has.

Chapter 4. Conclusion and possible future work on the current design.

Capítulo 2

Procesamiento de imágenes biomédicas

Como ya se ha comentado en anteriores apartados del documento, el objetivo final de este proyecto es el reconocimiento de patrones dentro de imágenes biomédicas. Las imágenes representan células distribuidas en una superficie de 1000x1000 píxeles codificada en escala de grises, e.g en la figura 2.1 se pueden observar una imagen del *dataset*. En esta imagen además de poder observar tanto el ruido como el emborronamiento existente, se puede apreciar la complejidad de la tarea de clasificación de las células según la morfología de éstas.

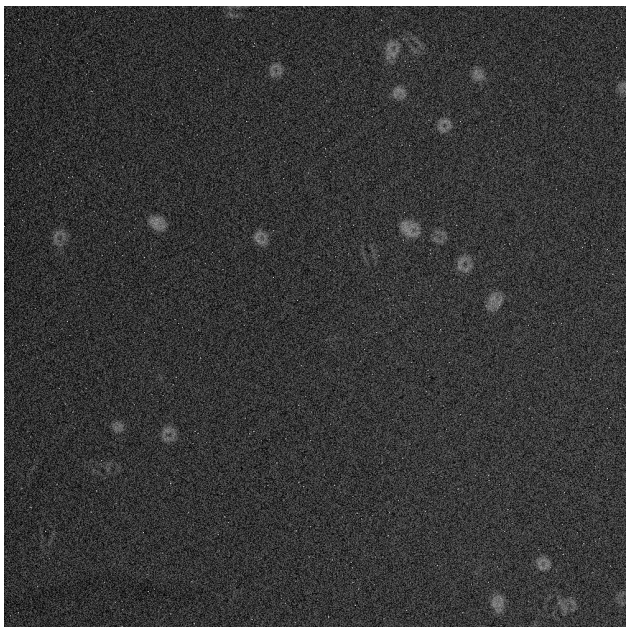


Figura 2.1: Ejemplo de dos imágenes del *dataset*.

Para lograr esto se han utilizado además de técnicas de restauración y mejora de imágenes, técnicas de inteligencia artificial (en las figuras 2.2 y 2.3 se puede observar el esquema metodológico utilizado).

- Fase I y II. Filtro mediano
- Fase I. Diferencia gausiana
- Fase II. Binarización
- Fase III. Dilatación
- Fase III. Erosión
- Fase IV. Redes neuronales

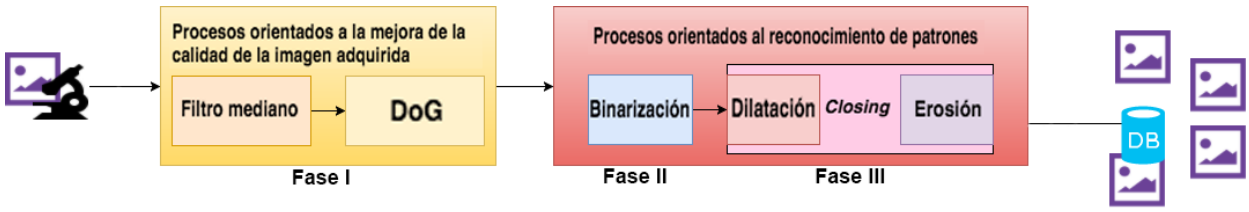


Figura 2.2: Primera parte de la metodología.

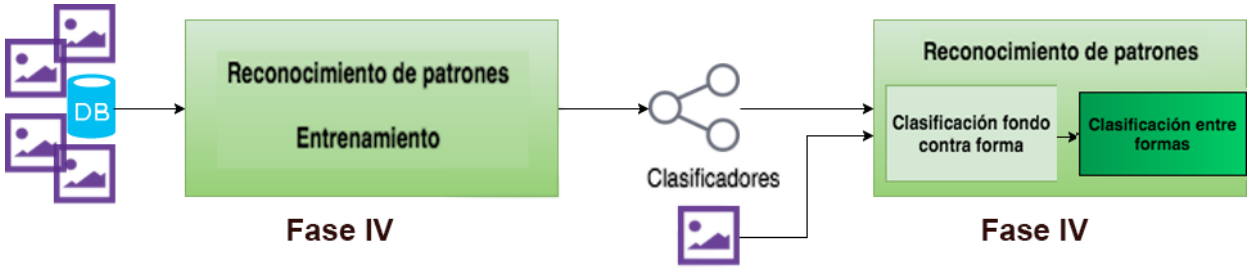


Figura 2.3: Segunda parte de la metodología.

A lo largo de este capítulo se mostrarán en primera instancia las herramientas que han sido necesarias para la realización de este proyecto y por último, se presentarán todas las fases en las que se ha dividido el mismo y en las cuales se aplican las diferentes técnicas comentadas.

De igual forma, en cada uno de los apartados se comentarán los fundamentos teóricos de estas técnicas.

2.1. Herramientas utilizadas

FPGA Altera Cyclone V Para las simulaciones *hardware* se ha hecho uso de la Cyclone V SE 5CSEMA5F31C6N la cual según datos comerciales de la empresa *Altera*, tiene las siguientes características:

Tabla 2.1: Especificaciones FPGA Altera Cyclone V - 5CSEMA5F31C6N.

64MB SDRAM (16-bit data bus)	87 Variable-precision DSP blocks
Clock x4 (50MHz, single-ended)	174 18 bit x 19 bit multipliers
32,075 Adaptive logic modules (ALMs)	480 MLABs (Kb)
397 M10K memory blocks	3,970 M10K memory (Kb)
6 FPGA PLLs	3 HPS PLLs

Simulink *Simulink* es una herramienta de desarrollo asistido de modelos, simulaciones y análisis de sistemas dinámicos multidominio. Esta herramienta ofrece una simbiosis con el resto de entornos de Mathworks y programas o *scripts* Matlab. *Simulink* es ampliamente utilizado en sistemas de control automático y procesamiento digital de señales. Se han podido modelar con facilidad los distintos sistemas del proyecto gracias a la gran cantidad de funciones prediseñadas para el tratamiento de imágenes que incorpora *Simulink*.

HDL Coder HDL Coder, es un *toolbox* de Matlab que genera código VHDL y Verilog sintetizable. Esta herramienta puede ser utilizada para la programación de sistemas tales como una FPGA, como es el caso que nos ocupa. La herramienta trabaja directamente con funciones Matlab y modelos de *Simulink*. Ésta convierte los diseños Matlab en diseños Verilog y HDL equivalentes. HDL Coder convierte automáticamente operaciones basadas en punto flotante en operaciones en punto fijo. En el presente, HDL Coder soporta más de 200 bloques de *Simulink*. Haciendo uso de estos bloques, uno puede

diseñar sistemas complejos en un corto periodo de tiempo. En el contexto del proyecto actual, se ha recomendado su uso y no el de otros sistemas, por su alta compatibilidad con diferentes fabricantes y modelos de FPGAs.

Quartus II Quartus II es una herramienta *software* producida por Altera para el análisis y síntesis de diseños realizados en HDL. La Edición Web es una versión gratuita de Quartus II que puede ser descargada desde la web. Esta edición permite la compilación y la programación de un número limitado de dispositivos Altera. La familia de FPGAs de bajo coste Cyclone, está soportada por esta edición. Esta herramienta se ha utilizado para la síntesis del código generado por el HDL Coder de Mathworks.

2.2. Fase I. Eliminación de ruido y Énfasis de bordes

2.2.1. Fundamento teórico

Son muchos los estudios que avalan el uso de un filtro mediano para la eliminación de ruido del tipo *salt & pepper*, e.g. *The Component Median Filter for Noise Removal in Digital Images* de *Harish and M.R.Gowtham*[5] en el cual se muestran los fundamentos matemáticos de este filtro (ecuación 2.1) además de los tipos de métricas para el análisis del posible ruido existente en una imagen.

$$FILTRO\,MEDIANO(X_1 \cdots X_N) = \frac{1}{N} \times \sum_{i=1}^{N} x_i \tag{2.1}$$

Ecuación matemática del filtro mediano.

En resumidas cuentas, el filtro mediano consiste en dada una máscara de $N \times N$ píxeles igualar el valor de los elementos de esta mascara $(X_1 \cdots X_N)$ al valor medio de la misma. El calculo de la mediana supone además un criterio de ordenación de los valores basado en el valor central del conjunto que divide éste en dos partes iguales.

Figura 2.4: Eliminación de ruido mediante filtro mediano con mascara de 3×3.

En la figura 2.4 se muestra a la izquierda una imagen con ruido del tipo *salt & pepper* y a la derecha la misma imagen tras aplicar filtro mediano y como se puede ver, este método resulta muy potente aun siendo dependiente del tamaño escogido para la máscara. Un ejemplo del efecto que tiene el tamaño de la máscara en el resultado lo podemos encontrar en la comparativa presentada en la figura 2.5. En esta se puede observar como la imagen de la derecha (máscara de tamaño 5×5) se encuentra más borrosa debido al efecto normalizador de este filtro.

Figura 2.5: Comparativa de aplicación de filtro mediano: máscaras de 3×3 y 5×5.

Después de haber reducido el ruido existente en la imagen, se ha aplicado un filtro basado en una diferencia de gausianas (ecuación 2.2) para enfatizar los bordes de las figuras existentes. La tarea realizada tiene como objetivo facilitar la determinación de la morfología de éstas. En resumidas cuentas esta función se traduce como la diferencia entre una imagen convolucionada con la varianza σ_1^2 y la misma imagen convolucionada con la varianza σ_2^2, teniendo en cuenta que $\sigma_2^2 > \sigma_1^2$.

$$DoG = G_{\sigma_1}(x,y) - G_{\sigma_2}(x,y) = \frac{1}{\sqrt{2\pi}} \left(\frac{1}{\sigma_1} e^{-(x^2+y^2)/2\sigma_1^2} - \frac{1}{\sigma_2} e^{-(x^2+y^2)/2\sigma_2^2} \right) \tag{2.2}$$

Ecuación matemática diferencia de gausianas.

Un ejemplo de aplicación de esta técnica la podemos ver en la figura 2.6, donde a la izquierda podemos ver los efectos de la aplicación de un filtro gausiano en una imagen cualquiera y a la derecha los efectos, tanto positivos como negativos de aplicar diferencias de gausianas.

Figura 2.6: Ejemplo de aplicación de diferencia de gausianas: Aplicación de filtro gausiano (izquierda) y aplicación de diferencia de gausianas (derecha).

2.2.2. Aplicación práctica

Dentro de la fase I, se ha configurado el filtro mediano con una máscara de tamaño 7×7 debido a la alta resolución de las imágenes del *dataset*. Para el filtro gausiano se han utilizado las gausianas mostradas en la ecuación 2.3.

$$G_{\sigma_1} = \begin{pmatrix} -1{,}0000 & -1{,}0000 & -1{,}0000 \\ -1{,}0000 & 9{,}8000 & -1{,}0000 \\ -1{,}0000 & -1{,}0000 & -1{,}0000 \end{pmatrix} G_{\sigma_2} = \begin{pmatrix} 0{,}1667 & 0{,}6667 & 0{,}1667 \\ 0{,}6667 & -3{,}3333 & 0{,}6667 \\ 0{,}1667 & 0{,}6667 & 0{,}1667 \end{pmatrix} \tag{2.3}$$

Diferencia de gausianas utilizada.

Como se puede observar, tanto los métodos mostrados hasta el momento como los que aparecerán a continuación, son métodos parametrizados según el escenario objeto de estudio. La codificación de estos algoritmos en *Simulink* ha sido realmente rápida ya que se han utilizado los bloques *Median Filter* e *Image Filter* de *Simulink* para generar esta parte del sistema (figura 2.7).

Figura 2.7: Esquema de eliminación de ruido y énfasis de bordes.

2.3. Fase II. Binarización y reducción de ruido

2.3.1. Fundamento teórico

Una vez llegados a este punto del *pipeline*, el proyecto tiene como tarea generar información útil para una posterior etapa de *machine learning*. El primer paso es generar una

imagen binaria con el único fin de hacer énfasis en la morfología de las formas presentes en la imagen. Para esto se ha utilizado la técnica denominada como binarización, siendo ésta un resultado derivado de la técnica de umbralización. Como se puede observar en la ecuación 2.4, esta técnica es dependiente del estudio del valor umbral.

$$BIN(X_1 \cdots X_N, UM) = \begin{pmatrix} \begin{cases} 255, & \text{if } X_{1,1} \geq UM \\ 0, & \text{otherwise} \end{cases} & \cdots & \cdots \\ \cdots & \cdots & \cdots \\ \cdots & \cdots & \begin{cases} 255, & \text{if } X_{N,N} \geq UM \\ 0, & \text{otherwise} \end{cases} \end{pmatrix} \tag{2.4}$$

Ecuación matemática del filtro de binarización.

En resumidas cuentas esta técnica consiste en la generación de una imagen binaria con el fin de filtrar y reducir el dominio de los datos de entrada.
Una vez llegados a este punto es necesario destacar en el hecho de que haciendo uso de esta técnica se produce una gran perdida de información relativa a la imagen, siendo ésta aún mayor en el caso de seleccionar un valor umbral incorrecto. Aunque existen estudios que presentan técnicas para el calculo automático del valor umbral[1,12], por temas de simplificación se ha optado por la selección manual del valor umbral mediante el estudio de los histogramas de las imágenes del *dataset* ya que éste nos permite conocer la frecuencia relativa de aparición de cada uno de los posibles niveles de intensidad dentro de la imagen en cuestión.

2.3.2. Aplicación práctica

En la figura 2.8 podemos ver el histograma con la distribución media de los valores en el conjunto de imágenes del dataset.

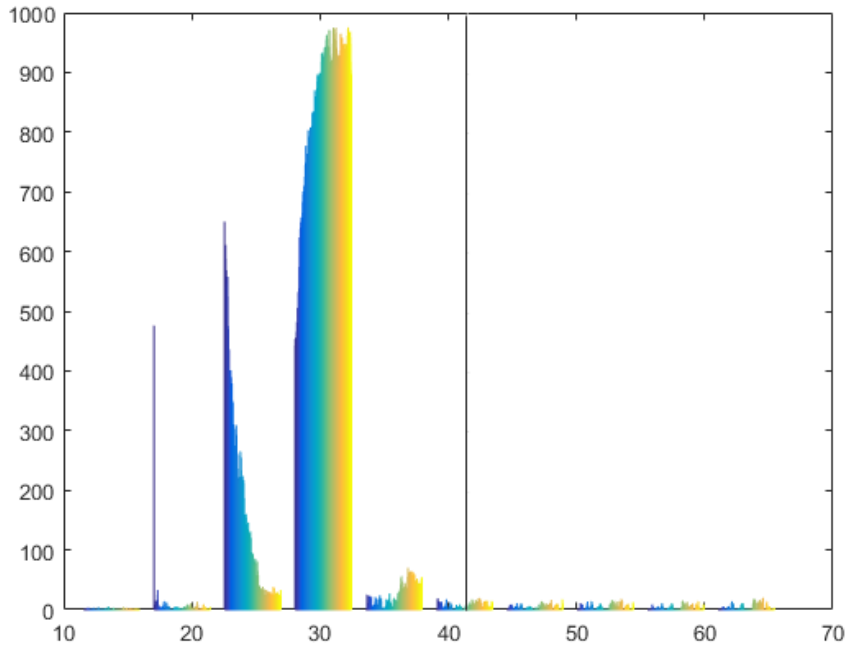


Figura 2.8: Histograma con la frecuencia media de los valores dentro de las imágenes del *dataset*.

Si asumimos que la mayoría de información presente en las imágenes se corresponde con el fondo, es comprensible deducir que el mismo esté representado en el histograma por los puntos mayor acumulación de píxeles. Aunque existe información posiblemente relevante entre los puntos de más acumulación, se han asumido estos valores como posible ruido o formas no enfocadas por el microscopio.

Por último, la codificación del algoritmo de binarización se ha planteado mediante el uso de un bloque comparador y un multiplicador (figura 2.9). El esquema completo de esta fase se puede observar en la figura 2.10. La selección del valor umbral se ha realizado mediante un estudio de los histogramas de las imágenes del *dataset*, habiéndose probado diferentes valores con el fin de obtener el valor mínimo mediante el cual se obtuviera la menor cantidad de fondo.

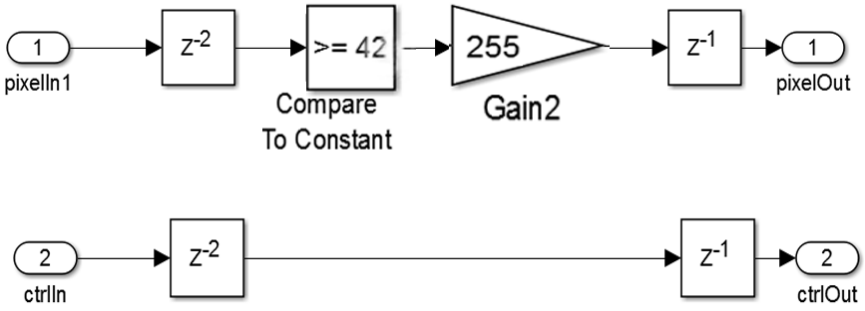


Figura 2.9: Esquema *Simulink* - Esquema de binarización.

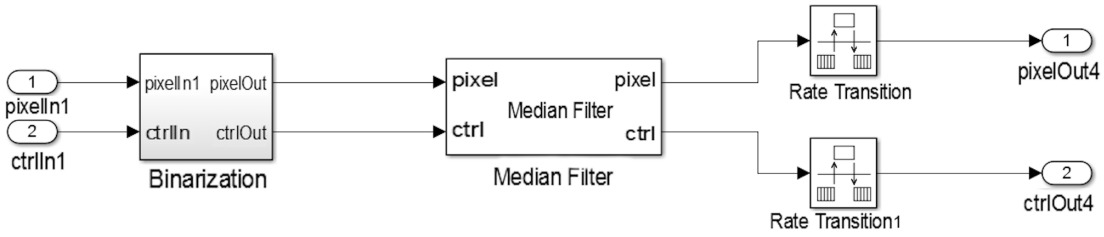


Figura 2.10: Esquema *Simulink* - Fase II: Binarización y reducción de ruido.

2.4. Fase III. *Closing*: Rellenado de regiones

2.4.1. Fundamento teórico

El procesamiento morfológico de imágenes es un campo de investigación muy importante. Éste se puede ver reflejado a día de hoy en aplicaciones tan dispares como pueden ser aplicaciones para el procesamiento de imágenes biomédicas y aplicaciones para el reconocimiento de patrones en textos manuscritos. En este proyecto se ha utilizado la técnica de *closing* para enfatizar la morfología de las células representadas en las imágenes anteriormente binarizadas. Si observamos la figura 2.11 donde se muestra el efecto de la aplicación de esta técnica, podemos ver que el resultado es el cierre o énfasis de la morfología de la figura, es decir esta técnica consigue rellenar los huecos existentes generando así una estructura más uniforme.

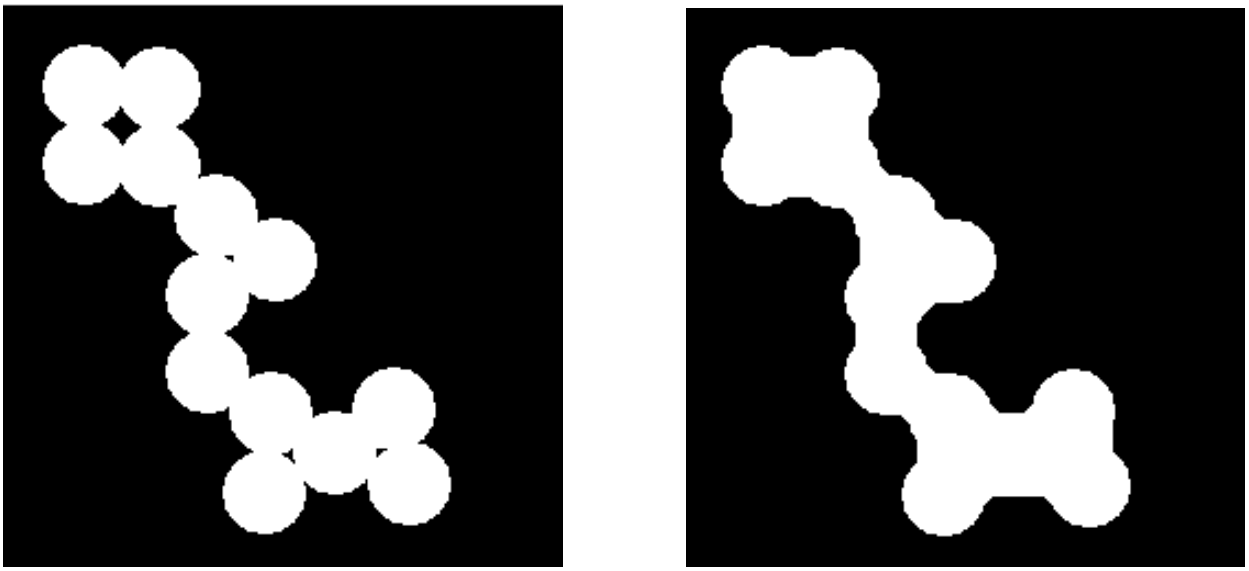


Figura 2.11: Imagen original y aplicación de técnica *closing*.[1]

Para conservar la forma característica de la figura que se quiere procesar, se genera un patrón, en este caso con forma de disco.

2.4.2. Aplicación práctica

Aunque lo ideal hubiera sido generar una única fase para la aplicación de esta técnica, debido a las restricciones de memoria de la FPGA se ha tenido que dividir en dos fases. Esto no ha resultado problemático ya que esta operación también se puede ver como la aplicación de una erosión y una dilatación (en este orden) como se ve en la ecuación 2.5 (dilatación: $\oplus$ y erosión: $\ominus$). En ésta, mientras que el término A representa una imagen, el término B representa un kernel a aplicar sobre ésta.

$$A \bullet B = (A \oplus B) \ominus B \tag{2.5}$$

Ecuación matemática del filtro morfológico *closing*.

[1] https://es.mathworks.com/help/images/ref/imclose.html

Fase III. Dilatación y Erosión

En morfología matemática la suma de Minkowski[35] sirve para definir las operaciones de dilatación y erosión. Estas operaciones son las dos operaciones morfológicas más básicas. El principal efecto de la aplicación sobre una imagen binaria de la operación de dilatación es el crecimiento gradual de los límites de las regiones de los píxeles del primer plano, aumentando así el grosor de estas regiones y volviendo más pequeños los huecos en su interior. La operación dilatación es la complementaria de la erosión *i.e.* dilatar los píxeles del primer plano es equivalente a erosionar los píxeles del fondo de la imagen, e.g. mientras que en la figura 2.15 se puede observar el efecto de la operación dilatación, en la figura 2.14 se puede observar el efecto de la operación erosión sobre la misma imagen.

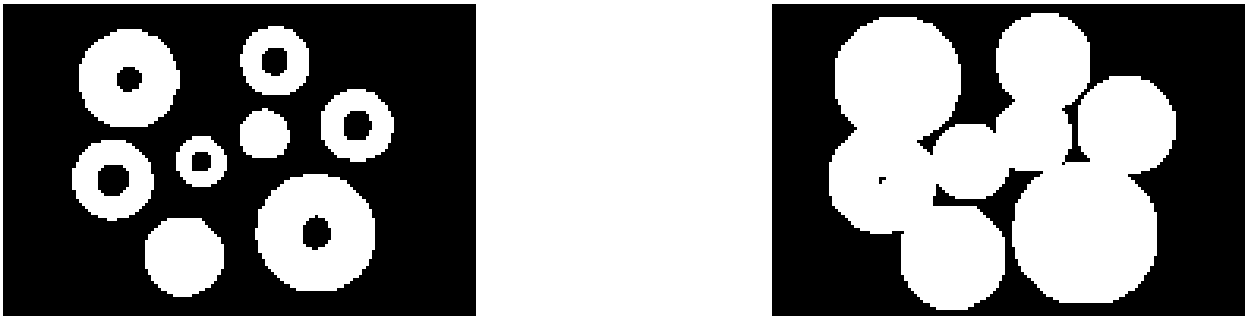


Figura 2.12: Imagen original y aplicación de operación dilatación.[2]

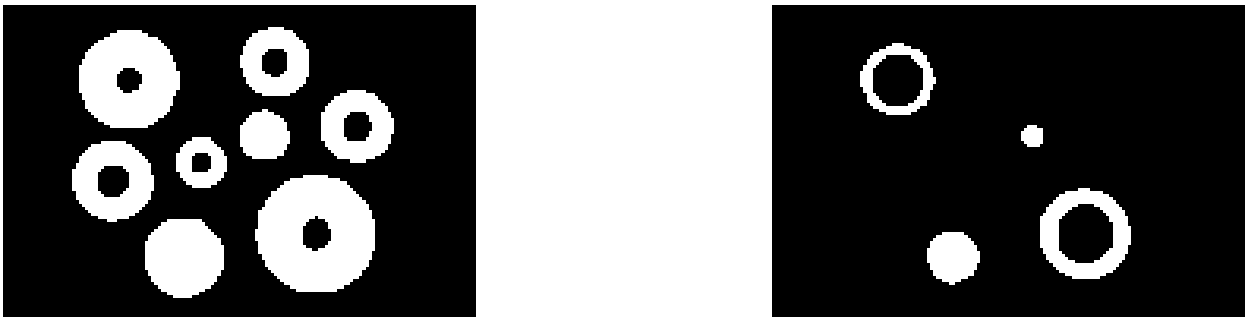


Figura 2.13: Imagen original y aplicación de técnica erosión.[3]

[2]https://reference.wolfram.com/language/ref/Dilation.html
[3]https://reference.wolfram.com/language/ref/Erosion.html

$$D(A,B) = A \oplus B = \bigcup_{\beta \in B} (A + \beta) \tag{2.6}$$

Ecuación matemática de la operación *dilatación*.

$$D(A,B) = A \ominus (-B) = \bigcap_{\beta \in B} (A - \beta) \tag{2.7}$$

Ecuación matemática de la operación *erosión*.

Para la codificación de las operaciones dilatación y erosión se han utilizado los bloques *Grayscale dilation* (figura 2.15) y *Grayscale erosion* (figura 2.14) de *Simulink* con un patrón o kernel en forma de disco.

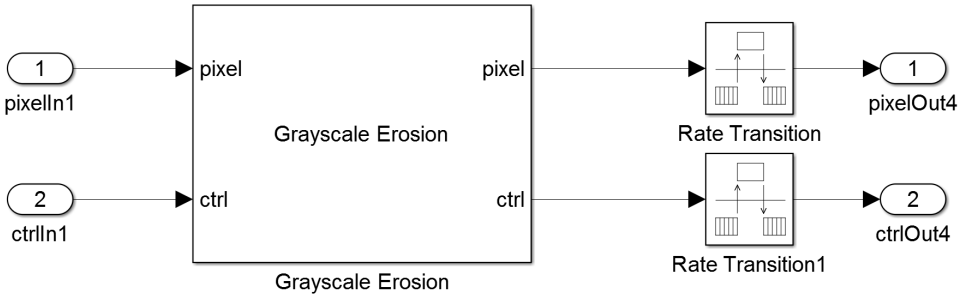


Figura 2.14: Esquema *Simulink* - Operación morfológica: erosión.

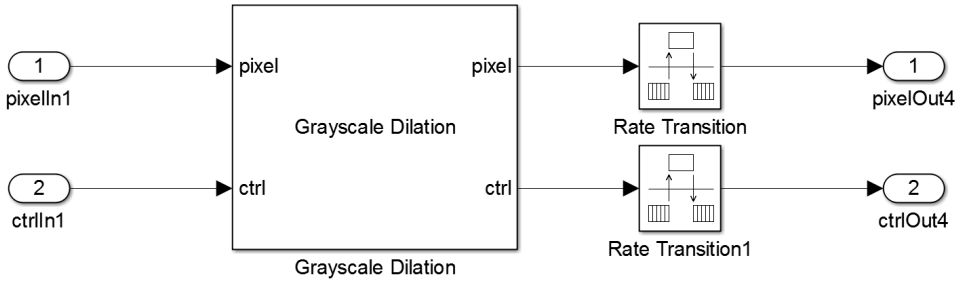


Figura 2.15: Esquema *Simulink* - Operación morfológica: dilatación.

2.5. Fase IV. Reconocimiento de patrones

Debido a las limitaciones de memoria impuestas por la FPGA, se han tomado varias decisiones de diseño para obtener una simplificación de la tarea que nos ocupa. Esta fase del proyecto se ha dividido en dos tareas de clasificación que toman como entrada partes de la imagen para determinar una decisión en el contexto de un problema de clasificación binaria.

- Identificación de partes de la imagen que no forman parte de ninguna célula.
- Clasificación de partes de una figura.

Existen muchos algoritmos orientados al reconocimiento de patrones, desde arboles de decisión hasta máquinas de soporte vectorial, pasando por redes neuronales y sistemas basados en reglas. En este proyecto se ha hecho uso de redes neuronales para dar una posible respuesta a la problemática del reconocimiento de patrones en imágenes de este tipo (motivados en parte por los buenos resultados obtenidos en estudios similares[19,30]).

2.5.1. Redes neuronales. Fundamentos teóricos

Las redes neuronales son un modelo computacional de aprendizaje y con la idea inicial de emular el procesamiento biológico de información. Se basa en el concepto de neurona y en la interconexión de las posibles neuronas entre sí, formando un malla similar a lo que ocurre en un cerebro. Un ejemplo de lo que vendría a ser el concepto de red neuronal artificial lo podemos encontrar en la figura 2.16 en la que podemos ver una distribución de neuronas (nodos rojos) distribuidas en tres tipos de capas, capa de entrada, capas ocultas y capa de salida (Perceptron multicapa).

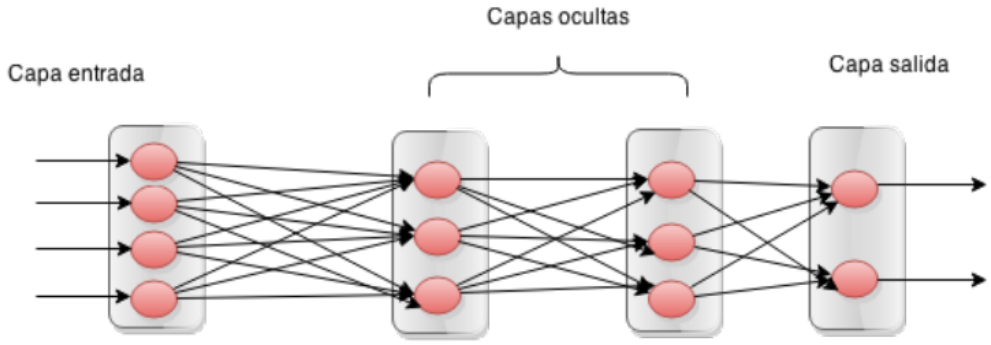


Figura 2.16: Modelo teórico de red neuronal.

Mientras que la capa de entrada tiene tantas neuronas como parámetros tiene una instancia en el modelo de datos que se va a utilizar en la red, la capa de salida tiene tantas neuronas como numero de conjuntos posibles o clases en las que dividir las diferentes instancias. Si tenemos en cuenta la ecuación 2.8 la cual define el comportamiento de una neurona en la red, las capas ocultas además de añadir complejidad al modelo, dotan de un gran poder de expresividad a las neuronas ya que estas procesan la salida de otras neuronas de capas anteriores y a su vez otorgan la entrada a las neuronas de capas siguientes.

$$Y = f(\sum_{i=1}^{n} w_i * x_i) \tag{2.8}$$

Ecuación matemática de una neurona artificial.

En esta ecuación se muestra como una neurona procesa todo estimulo representado como x_i proveniente de las capas anteriores y genera un estimulo de salida Y que se transmite a las neuronas de las capas siguientes. A cada estimulo recibido por la neurona se le asocia un peso w_i y el procesamiento del conjunto de estímulos se traduce como el sumatorio de los productos obtenidos entre estos estímulos y sus pesos asociados.
Al resultado de la aplicación de la ecuación 2.8 hay que añadir la aplicación de una función de activación, siendo las más comunes la gaussiana y la sigmoidal o sigmoide (ecuación 2.9).

$$sigmoide(Y) = \frac{1}{1 + e^{-Y}} \tag{2.9}$$

Función de activación: sigmoide.

Una vez visto los principales fundamentos del modelo de red neuronal, el entrenamiento de este modelo de aprendizaje es el proceso de establecimiento de los vectores de pesos dentro de cada neurona.

2.5.2. Redes neuronales. Aplicación práctica

Como ya se ha comentado, esta parte del proyecto está dividida en dos fases que aunque hacen uso del mismo algoritmo están diferenciadas por la finalidad que cumplen. En este apartado del documento se van a comentar los aspectos generales que comparten estas dos fases, detallando los aspectos específicos en sus respectivos apartados.

Para estos apartados y con el objetivo de simplificar los modelos resultantes se han tomado decisiones tales como la generación de los pesos asociados a las neuronas mediante *software*[4] y el uso de los valores de los píxeles como entrada del modelo. Una vez se han generado los distintos vectores de pesos, éstos se han trasladado al diseño de red neuronal implementado con *Simulink*. Si recapitulamos en lo que se refiere al modelo teórico de red neuronal tenemos que las diferencias entre los posibles modelos vienen dadas por el número de capas ocultas, el número de neuronas por capa y el valor de los pesos asociados a cada neurona. Un ejemplo de la disposición de las neuronas a lo largo de las capas en los modelos de *Simulink* diseñados lo podemos encontrar en la figura 2.17, donde mostramos el diseño de una red neuronal con los vectores de pesos codificados en forma de constantes.

[4]Los diseños de la fase de entrenamiento de cada fase se comentan en sus respectivos apartados.

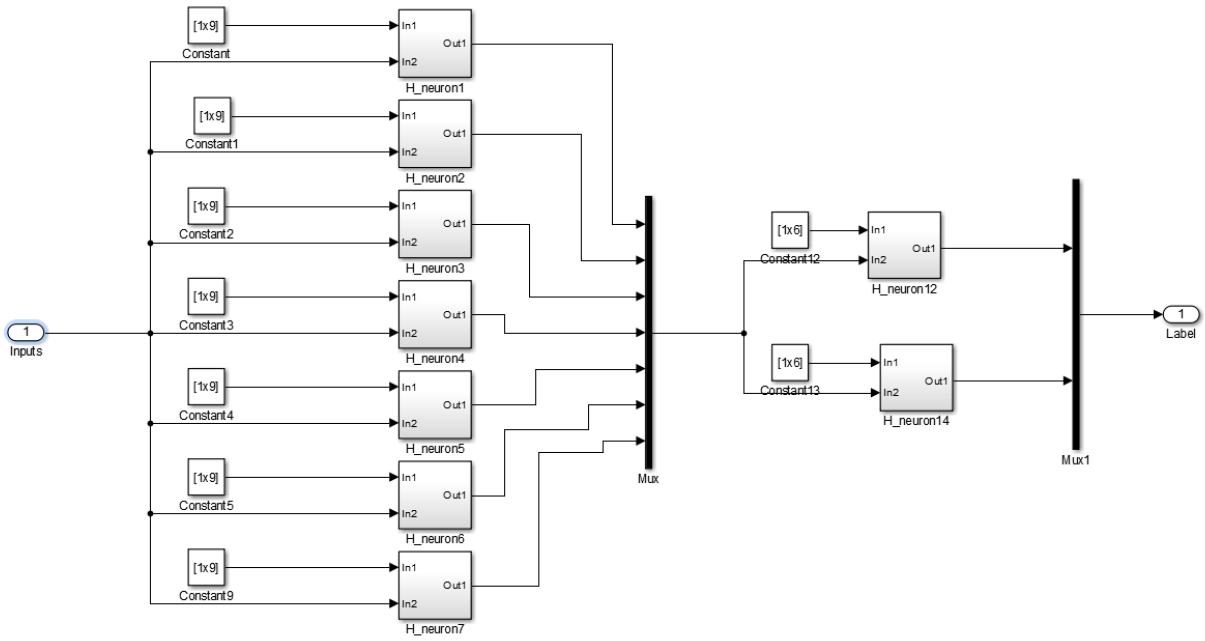


Figura 2.17: Esquema *Simulink* - Red neuronal.

Para los modelos implementados en estas fases se ha hecho uso de los modelos de neurona y de función sigmoide que se muestran en las figuras 2.18 y 2.19[10]. Como función de activación se ha utilizado una aproximación de la función sigmoide debido a la dificultad de sintetizar funciones exponenciales (ecuación 2.10).

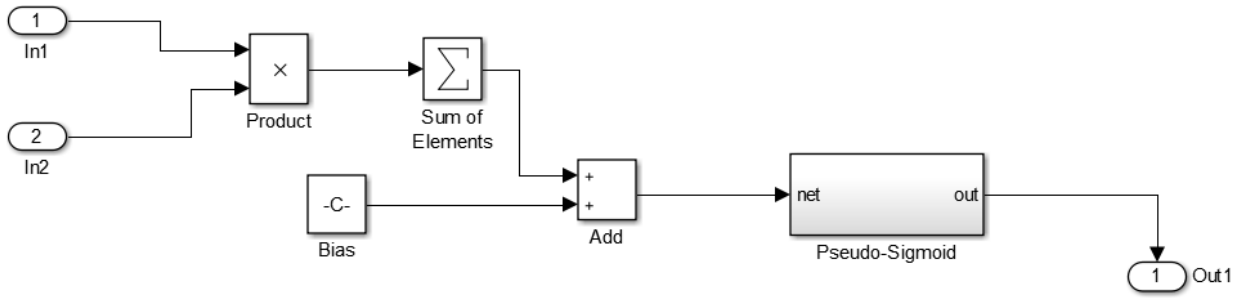


Figura 2.18: Esquema *Simulink* - Neurona.

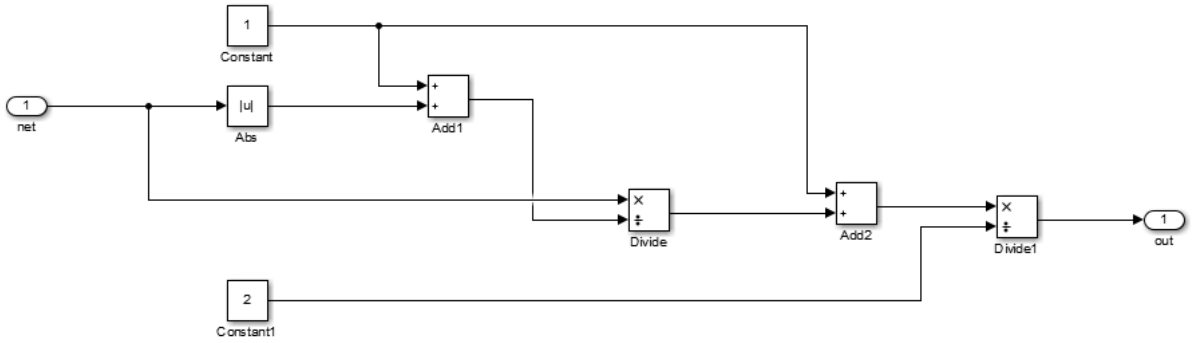


Figura 2.19: Esquema *Simulink* - Función sigmoide.

$$sigmoide(Y) = \frac{1}{2}\left(\frac{Y}{1+|Y|}+1\right) \tag{2.10}$$

Función de activación: sigmoide (aproximación).

Las figuras 2.20 y 2.21 muestran algunas de las instancias que conforman el *dataset* que se ha utilizado para entrenar y validar las redes neuronales resultado de esta parte del proyecto. Debido a que no se ha podido encontrar un *dataset* etiquetado y validado por expertos, es importante destacar que aunque se ha intentado seguir un criterio claro en la determinación de la clase asociada a la diferentes instancias, los experimentos asociados a este *dataset* son especulativos.

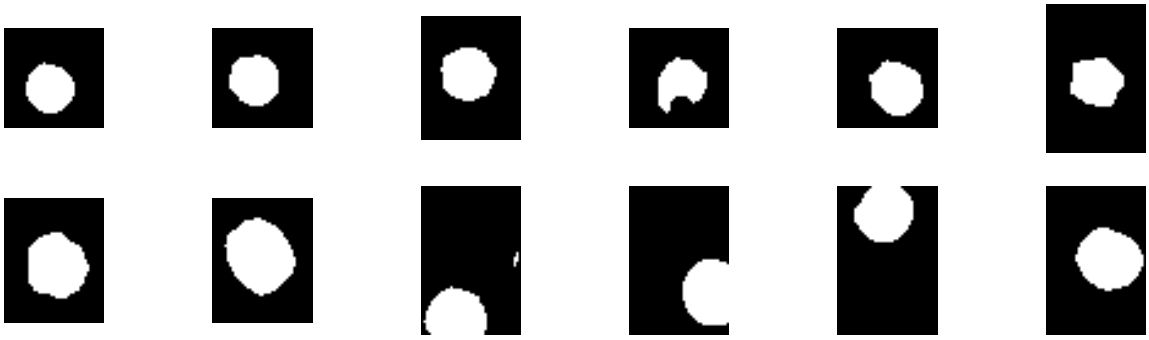


Figura 2.20: Dataset de formas: Figuras circulares.

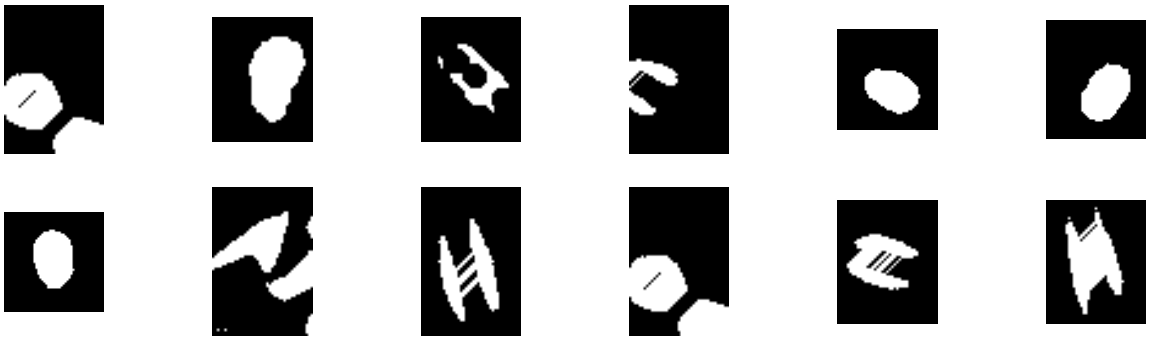


Figura 2.21: Dataset de formas: Figuras con forma de elipse.

Reconocimiento de formas

Este modelo esta diseñado para la clasificación de ventanas de una imagen en las clases *Fondo* y *Parte de forma*. Partiendo de que las imágenes a clasificar son imágenes en blanco y negro, a grandes rasgos el resultado de este sistema se podría ver como diferenciar entre algo totalmente negro o algo parcialmente blanco, hecho que no sería del todo cierto. De asumir esto estaríamos otorgando a los algoritmos de las anteriores fases una confianza absoluta y anularíamos la posibilidad de un margen de error por parte de estos. Aunque resulta necesaria su aplicación es fácil suponer que este clasificador tendrá un alto nivel de acierto. Siguiendo el esquema que se muestra en la figura 2.22 este modelo está entrenado con ventanas o conjuntos de píxeles de una imagen con información asociada a la distribución de los valores de los píxeles dentro de ella. Aunque se ha probado con diferentes tamaños de ventana, el que se ha seleccionado finalmente ha sido el de una ventana de 3x3 píxeles por los buenos resultados obtenidos en cuanto a precisión y ocupación de la FPGA. Durante la evaluación de este modelo y del tamaño de ventana a utilizar ha sido de gran importancia minimizar el espacio ocupado por este diseño, es por esto por lo que se ha escogido el mínimo tamaño de ventana que diera respuesta al problema planteado.

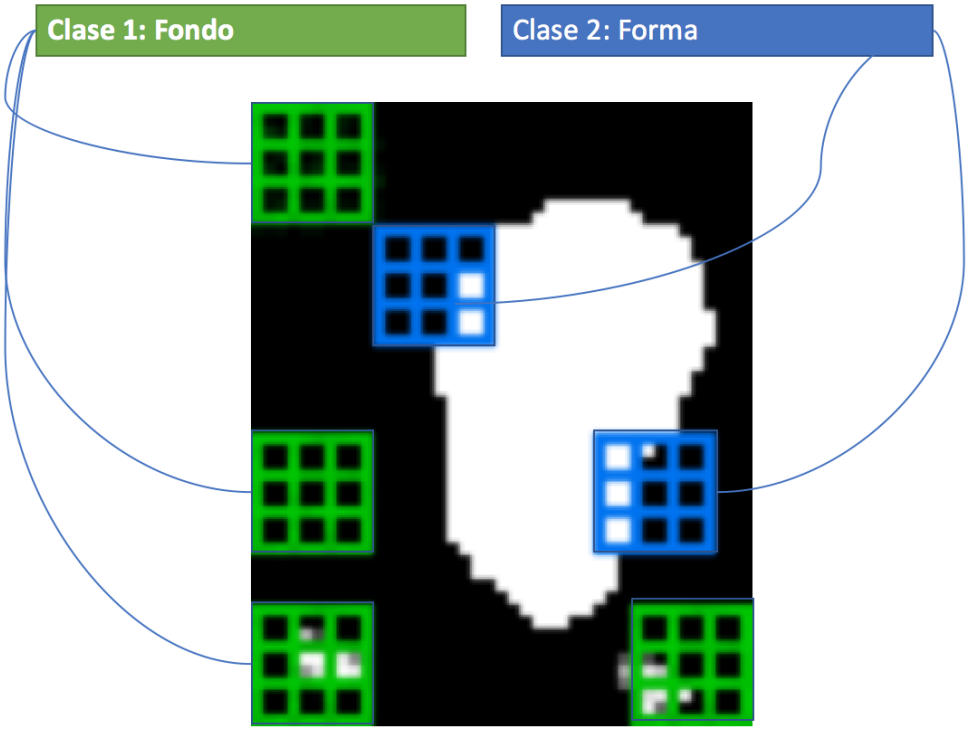


Figura 2.22: Datos de entrada de la red neuronal.

El funcionamiento de un sistema basado en este clasificador viene dado a continuación:

Algorithm 1 Reconocimiento de formas dentro de una imágen

Require: Imagen de tamaño X,Y. Ventana de tamaño N,M.

IM_r = Matriz de ceros de tamaño X,Y
for $x = 0$ hasta X **do**
  for $y = 0$ hasta Y **do**
    Obtener el conjunto de píxeles $C_{x,y}$ de tamaño $N * M$ alrededor del $Pixel_{x,y}$.
    $IM_r(x, y)$ = red_neuronal($C_{x,y}$)
  end for
end for
return IM_r

En resumen, si la distribución de los píxeles a clasificar no es similar a una ya vista por el modelo, bien sea por que contiene poca información o por que no tiene información acerca de esta distribución, éste lo asumiría como fondo. En conclusión, esta red neuronal está entrenada para diferenciar las posibles formas de lo que se podría considerar fondo,

error (ruido) o parte de una forma no delimitada.

Clasificación de formas

Una vez tenemos identificadas dentro de la imagen las zonas que son formas y las que no lo son, si utilizamos esta información para seleccionar las zonas de interés podemos reducir bastante el esfuerzo que supone clasificar las formas según su forma.

De igual modo que antes, en este clasificador también se ha hecho uso de un entrenamiento basado en ventanas. Se ha probado con diferentes tamaños y se ha acabado seleccionando un tamaño de 6x6. Si bien en el anterior caso el tamaño de la ventana era un factor casi sin importancia, pasando a priorizar los datos de ocupación de la FPGA, en este caso resulta de bastante importancia ya que según se reduce el tamaño de la ventana la precisión del clasificador disminuye y al revés si lo que hacemos es aumentarlo. Al final se ha seleccionado este tamaño de ventana y no uno más grande por limitaciones inherentes al modelo de FPGA utilizado.

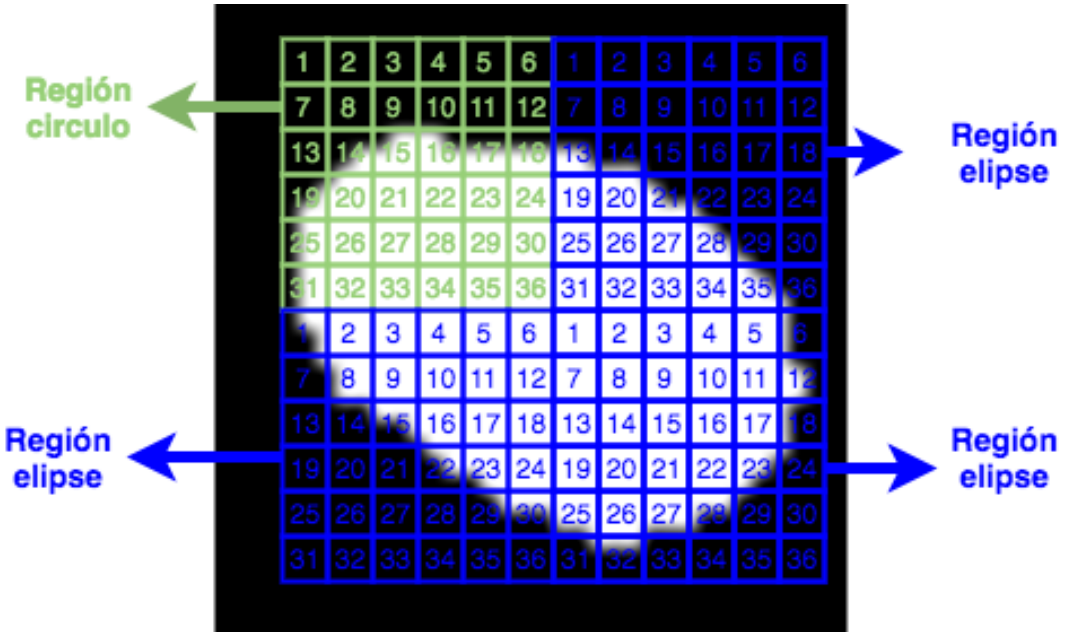


Figura 2.23: Clasificación de las regiones de una figura.

Este algoritmo de clasificación tiene como objetivo decidir si un segmento de una forma es parte de un tipo de forma o de otro (circulo o elipse). Si pensamos en una clasificación ideal, es decir con una precisión del 100 %, estaríamos ante un clasificador que para todo píxel perteneciente a una figura del tipo X, éste determina que es el del tipo X. Como no se han conseguido resultados que indiquen la posibilidad de nada parecido, se ha propuesto la clasificación de una figura en función de un algoritmo de votación basado en las distintas clasificaciones hechas sobre las ventanas que la forman. La figura 2.23 es una muestra de la indecisión que se produce al clasificar una instancia del *dataset*. En este ejemplo se muestra un figura dividida en 4 ventanas de un tamaño de 6x6 píxeles. Cada una de las ventanas se ha clasificado como parte de un circulo o como parte de elipse dando como resultado una clasificación donde el clasificador no es capaz de determinar con total seguridad que tipo de figura es. Sin embargo, si lo vemos como un problema de votación, esta figura se clasificaría correctamente como elipse.

Capítulo 3

Experimentación

En este capítulo se van a comentar las distintas pruebas realizadas sobre la solución planteada con el fin de optimizar y validar los diferentes diseños que la forman. En la figura 3.1 se puede observar el flujo seguido durante la experimentación y relacionado con el uso de las diferentes herramientas. Todas las partes que forman este proyecto tienen unas fuertes bases teóricas que han sido probadas en el contexto que nos ocupa haciendo uso de Matlab. Gracias a esto se han podido validar posteriormente los modelos *Simulink* con mucha facilidad.

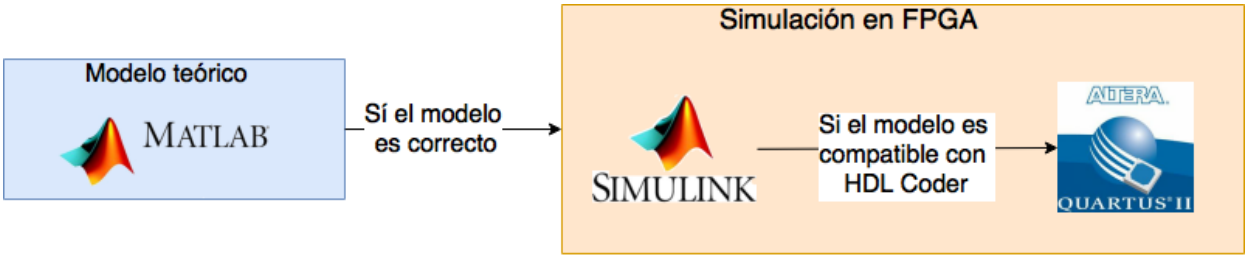


Figura 3.1: Flujo seguido durante la experimentación.

Todos los modelos que han sido probados con Matlab y han dado buenos resultados, se han portado a la FPGA, primero mediante la codificación en *Simulink* y luego mediante el paso a HDL con HDL Coder. Es necesario destacar que todos los modelos ejecutados en la FPGA han pasado diferentes filtros de compatibilidad.

- Compatibilidad con *Simulink* de funciones Matlab utilizadas.

- Compatibilidad con HDL Coder de funciones *Simulink* utilizadas.
- Compatibilidad con especificaciones de la FPGA de los modelos HDL generados con HDL Coder.

Como ya se ha comentado, este trabajo tiene dos vertientes. En la parte de procesamiento de imágenes han sido necesarias dos tipos de métricas para su evaluación, unas que midan la calidad de una imagen reconstruida y otras que midan la diferencia estructural entre dos imágenes. En la figura 3.2 se muestra el modelo genérico que se ha construido para realizar las comparativas entre las soluciones *hardware* y las soluciones *software*.

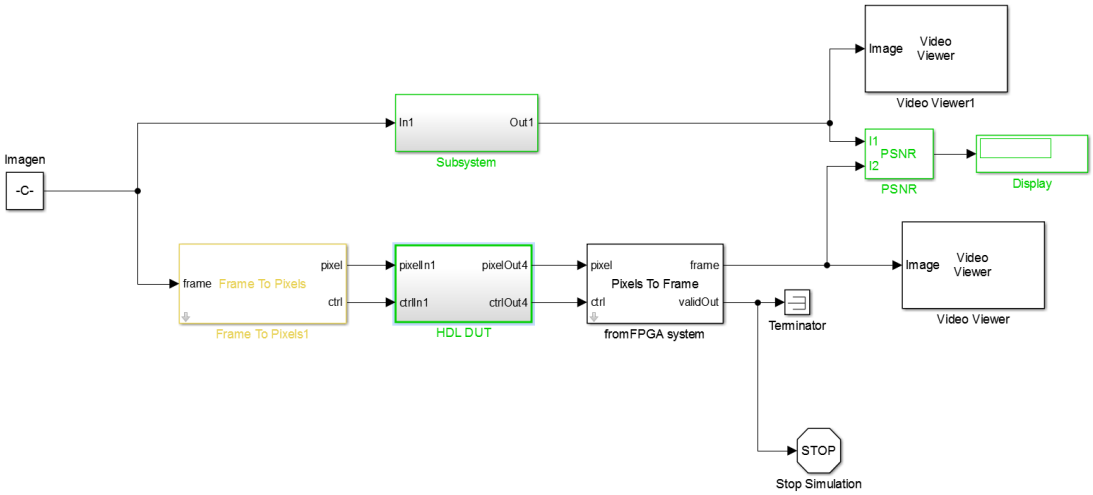


Figura 3.2: Esquema *Simulink* - Esquema abstracto de *Simulink* aplicado a todas las fases de reparación de imágenes.

Este modelo se ha modificado de forma que permitiera validar los distintos diseños en función de estas métricas.

Además de las diferentes métricas de calidad y de los tiempos de ejecución obtenidos, para la evaluación de la calidad de los modelos generados por el HDL Coder también se han utilizado algunos datos de ocupación de los recursos de la FPGA:

LAB Un LAB o *logic array block*, está formado por un array de celdas lógicas. Cualquier señal que esta disponible en alguna de las celdas lógicas del LAB esta disponible para todo el LAB.

ALM Cada ALM contiene varios recursos operacionales basados en *Lookup tables*.

DSP Un DSP es un procesador con una arquitectura optimizada para el tratamiento digital de señales.

PLL Un PLL o *Phase Locked Loop* es un circuito que permite controlar la frecuencia y fase de una señal mediante el uso de una señal externa.

A lo largo de este capítulo se va a mostrar toda la experimentación realizada sobre los distintos modelos que conforman la solución.

3.1. Filtro mediano y DoG

En esta sección se van a comentar los resultados obtenidos en la fase de aplicación del filtro mediano y de la diferencia de gausianas. Ambas técnicas están destinadas a mejorar la calidad de la imagen inicial, estando destinado el filtro mediano a corregir el posible ruido de partida y la diferencia de gausianas a enfatizar los bordes de la imagen. Para el estudio de la aplicación de estas técnicas se ha pensado en las métricas PSNR (*peak-signal noise ratio*) y SSIM (structure similarity metric)[33] (como se puede ver en la tabla 3.1[1]). También se ha probado la versión *hardware* de este modelo para diferentes precisiones.

Tabla 3.1: Estudio de precisión: Fase I - Filtro mediano y DoG. Métricas de calidad.

	HW_uint5	HW_uint6	HW_uint7	HW_uint8	SW
PSNR(Im_X, Original) (db)[2,3]	18.64	28.09	23.12	31.47	37.22
SSIM(Im_X,Im_SW)[4,5]	0.25	0.41	0.72	0.89	0.91

[1]Los datos mostrados en esta tabla se corresponden a la media obtenida con 5 imágenes.

[2]El PNSR o *Peak signal-to-noise ratio*, sirve para medir el ratio entre la máxima energía de una señal y la energía en forma de ruido que afecta a la fidelidad de su representación. Cuanto mayor es este valor mejor es la calidad de la señal.

[3]Valor de PSNR obtenido al comparar la imagen IM con precisión X con la imagen original (precisión double).

[4]El indice de similitud estructural o SSIM es un método para medir la similitud entre dos imágenes. Esta métrica se mide entre 0 y 1, siendo 1 el resultado de una comparación entre dos imágenes exactas.

[5]Valor de SSIM obtenido al comparar la imagen IM con precisión X con la imagen obtenida mediante la versión software (precisión double).

Para la realización de este estudio se ha tomado como *baseline* o marca a batir el resultado obtenido mediante un modelo *software*. Como se puede observar en esta tabla, para una precisión de *unsigned integer* con una longitud de palabra de 8 bits (*uint8*) se obtienen resultados bastante similares en lo que se refiere a SSIM. No obstante, la diferencia de ruido con la versión *software* es bastante notable. Otro dato relevante de la anterior tabla es que conforme se baja la precisión de los datos de entrada, se produce una caída casi lineal en el estudio de la calidad.

La elección del modelo con precisión *uint8* viene guiada, además de por los anteriores datos, por el estudio de aprovechamiento de recursos mostrado en la tabla 3.2 [6]. En este modelo es destacable el hecho de que la reducción de la precisión no es un factor que modifique lo suficiente los tiempos de procesamiento como para asumir la perdida de calidad.

Tabla 3.2: Estudio de precisión: Fase I - Filtro mediano y DoG. Tiempo y aprovechamiento de area.

	Total block memory bits	Logic usage (in ALMs)	Logic array blocks (LABs)	Execution time (HW / SW)	Transfer time (HW)
uint5	155,903 / 4,065,280 (3.83 %)	17,848 / 32,070 (55.65 %)	3,207 / 3,207 (100 %)	3.36 / 3.45	0.20
uint6	164,376 / 4,065,280 (4.04 %)	18,401 / 32,070 (57.37 %)	3,207 / 3,207 (100 %)	3.65 / 3.45	0.20
uint7	172,849 / 4,065,280 (4.25 %)	18,537 / 32,070 (57.80 %)	3,207 / 3,207 (100 %)	3.83 / 3.45	0.22
uint8	181,322 / 4,065,280 (4.46 %)	19,238 / 32,070 (59.98 %)	3,207 / 3,207 (100 %)	4.05 / 3.45	0.23

Además se puede observar que si bien es cierto que conforme se reduce la precisión se reducen ciertos parámetros de ocupación de la FPGA, existe un cuello de botella en el número de logic array blocks (LABs) ocupados y que no está influenciado por la precisión utilizada.

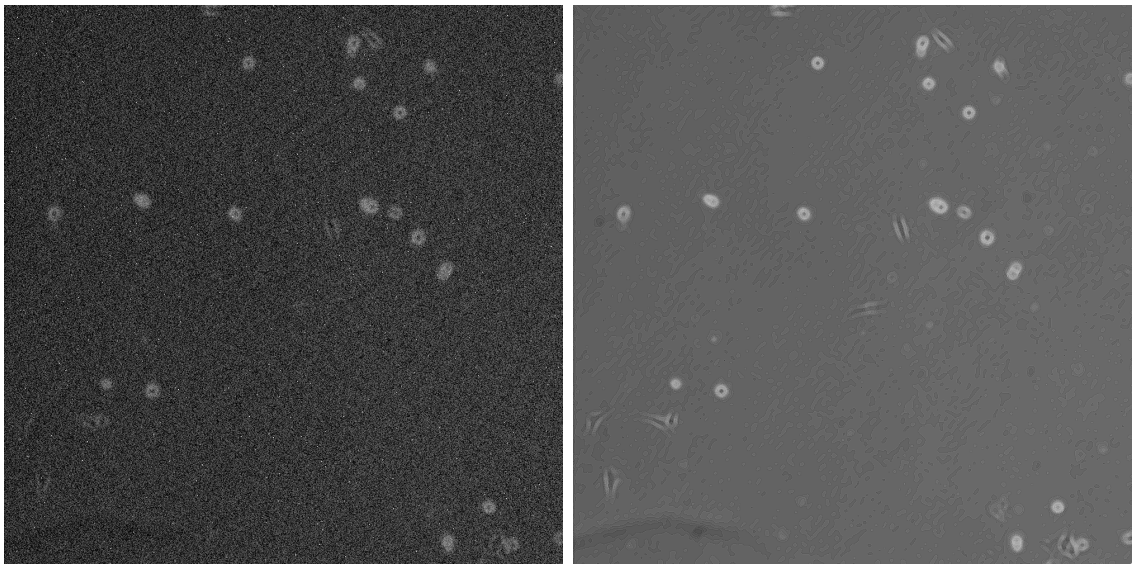
[6] Los tiempos de ejecución mostrados en esta tabla no incluyen los tiempos de envío y recepción (expuestos en la columna *transfer*)

Report 1. *Resource report*: Fase I - Filtro mediano y DoG (uint8)

***Resource report* - Simulink** **Unit**	**Type**
1	Multipliers
72	Adders and Subtractors
1,307	Registers
16	RAMs
1,123	Multiplexers
46	I/O Bits
***Resource report* - Quartus** **Unit**	**Type**
13,919	Total register
181,322 / 4,065,280 (4.46 %)	Total block memory bits
19,238 / 32,070 (59.98 %)	Logic usage (in ALMs)
0 / 87 (0 %)	Total DSP Blocks
1 / 6 (16.66 %)	Total PLLs
3,207 / 3,207 (100 %)	Total LABs

A la vista de estos resultados y del informe de aprovechamiento de recursos (*report* 1) obtenido mediante *Simulink* y *Quartus*, en el cual se puede ver un bajo nivel de aprovechamiento de muchos de los recursos de la FPGA, se han investigado las directivas o configuraciones posibles del software *Quartus II* para la optimización de las fases de *synthesis* y *fitter* con el fin de reducir el número de LABs utilizados y dar la posibilidad así de aumentar la precisión. La conclusión de esta parte del experimento es que estas directivas no son aplicables al modelo de FPGA utilizado y además, tampoco sería posible su uso con la licencia actual.

El resultado de esta fase del proyecto con la configuración seleccionada se muestra en la figura 3.3 donde se puede ver una clara reducción del ruido.



(a) Imagen original. (b) Resultado fase I

Figura 3.3: Comparativa: Original / Resultado Fase I (uint8)

3.2. Binarización y reducción de ruido

Desde esta parte del proyecto, el objetivo no es tanto obtener una imagen de gran calidad sino generar una imagen de la cual extraer posibles atributos para un futuro entrenamiento de un clasificador. En esta sección se van a comentar los resultados obtenidos tras la aplicación de un filtro de binarización y tras la corrección del posible ruido generado tras la aplicación del filtro.

Como se ha comentado acerca de la técnica de umbralización, ésta genera una imagen totalmente diferente a la inicial tan solo guardando la morfología de las figuras presentes. Debido a esto se ha utilizado como criterio de validación el número de formas identificadas, descartando así el uso de las métricas PSNR y SSIM. Para la realización de esta fase del estudio se ha tenido que modificar el umbral del filtro de binarización según la precisión utilizada, intentando llegar a una solución compromiso entre el número de artefactos y formas concretas identificadas. Los errores encontrados en este estudio van desde la aparición de más figuras que las existentes (artefactos) hasta la delimitación del área de una figura de forma errónea

(dando lugar a artefactos más complejos). Aunque errores tales como la identificación de más figuras se pueden subsanar en fases posteriores, errores en la determinación del área de una célula son difíciles de reparar y para el problema que nos ocupa este hecho resulta de gran importancia.

Tabla 3.3: Estudio de precisión: Fase II - Binarización y Filtro mediano. Métricas de calidad.

	Threshold	*Precision*	*Recall*
uint5	6	0.823	0.7
uint6	10	0.894	0.85
uint7	20	0.90	0.95
uint8	42	1	1

Como se puede observar en la tabla anterior, aunque los efectos comentados en cuanto a la aparición de artefactos se producen en los casos uint7, uint6 y uint5 (parte de las formas identificadas se corresponden a artefactos), en el caso de uint7 y uint6 estos artefactos no interfieren en la identificación de otras formas. Estos son pequeños fragmentos o partes de formas que debido al umbral seleccionado están incompletos. No obstante el estudio realizado muestra que de haber seleccionado un umbral menor se habrían modificado los contornos de las figuras identificadas e identificado además más artefactos. A la vista de los datos de la anterior tabla los casos más relevantes son los de una precisión mayor o igual a 7 bits ya que son los que mejor se aproximan al valor máximo de cobertura y mejor delimitan el contorno de una forma.

Tabla 3.4: Estudio de precisión: Fase II - Binarización y Filtro mediano. Tiempo y aprovechamiento de area.

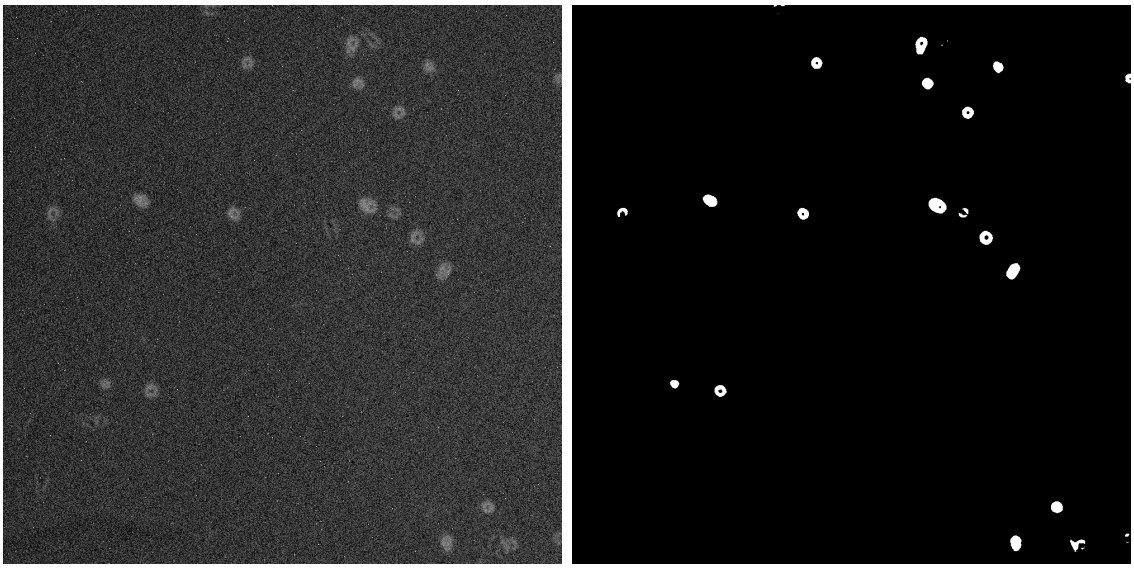
	Total block memory bits	Logic usage (in ALMs)	Logic array blocks (LABs)	Execution time (HW / SW)	Transfer time (HW)
uint5	145,353 / 4,065,280 (3.57 %)	14,147 / 32,070 (44.11 %)	2,642 / 3,207 (82.38 %)	3.67 / 1.70	0.20
uint6	151,768 / 4,065,280 (3.73 %)	14,561 / 32,070 (45.40 %)	2,714 / 3,207 (84.62 %)	3.75 / 1.70	0.20
uint7	158,155 / 4,065,280 (3.89 %)	15,073 / 32,070 (47.00 %)	2,816 / 3,207 (87.80 %)	4.03 / 1.70	0.22
uint8	164,542 / 4,065,280 (4.04 %)	15,496 / 32,070 (48.31 %)	2,852 / 3,207 (88.93 %)	3.96 / 1.70	0.23

Además, como se puede ver en la tabla 3.4 [7] en la que se muestra el estudio del apro-

[7]Los tiempos de ejecución mostrados en esta tabla no incluyen los tiempos de envío y recepción (ex-

vechamiento de recursos de la FPGA, aunque sigue sin ser de importancia la relación entre la precisión y el tiempo, si que existe una pequeña relación entre esta y el número de LABs utilizados (parámetro que en el anterior modelo ha resultado limitante como un cuello de botella).

Aunque podemos determinarla como la peor solución si miramos los parámetros de ocupación de la FPGA y los tiempos de proceso, se ha escogido como mejor opción dentro del conjunto de soluciones la versión de *uint8* ya que es la que consigue mejores resultados en cuanto al número de imágenes identificadas (figura 3.4 y *report* 2).



(a) Imagen original. (b) Resultado fase II.

Figura 3.4: Comparativa: Original / Resultado Fase II (uint8).

En lo que se refiere a número de LABs, la holgura existente en este modelo a dado pie a pensar en la importancia de la aplicación de cada uno de los bloques que forman los distintos modelos. Esto a dado lugar al planteamiento de un flujo alternativo de experimentación.

puestos en la columna *transfer*)

Report 2. ***Resource report*: Fase II - Binarización y Filtro mediano (uint8)**

***Resource report* - Simulink** **Unit**	**Type**
0	Multipliers
40	Adders and Subtractors
1,067	Registers
12	RAMs
977	Multiplexers
30	I/O Bits
***Resource report* - Quartus** **Unit**	**Type**
11,115	Total register
164,542 / 4,065,280 (4.04 %)	Total block memory bits
15,496 / 32,070 (48.31 %)	Logic usage (in ALMs)
0 / 87 (0 %)	Total DSP Blocks
1 / 6 (16.66 %)	Total PLLs
2,852 / 3,207 (88.93 %)	Total LABs

Esta línea de experimentación parte de la idea de simplificar el actual flujo de trabajo con el objetivo de incluir todas las partes vistas hasta ahora en único modelo. Como se puede observar en la figura 3.5 la alternativa que se plantea con este modelo es suprimir la aplicación del filtro mediano tras la aplicación de la binarización.

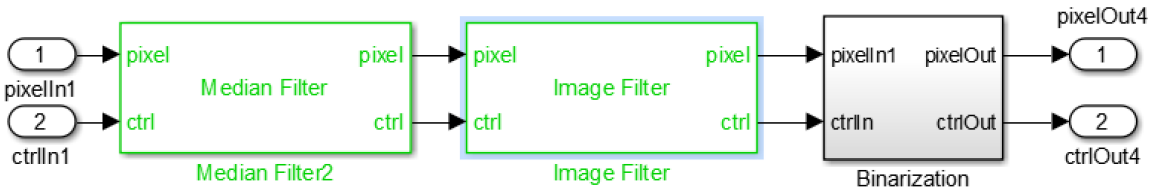


Figura 3.5: Esquema *Simulink* - Flujo alternativo.

Se ha utilizado PSNR y SSIM para la evaluación de este modelo, utilizando como imagen de referencia la imagen obtenida en las anteriores pruebas. Aunque se han obtenido valores entorno al 0.9 en SSIM y datos de ocupación del 100 % de LABs, esta parte del estudio se

ha descartado al obtener valores de PSNR indicativos de una alta disminución de la calidad conforme a los umbrales validados por el comité MPEG (*Moving Picture Experts Group*).

3.3. *Closing*: Rellenado de regiones

Como se puede ver en la tabla 3.5 [8] para la evaluación de esta parte del proyecto se han comparado los tiempos obtenidos con una versión software y otra hardware de los algoritmos de dilatación y erosión. Los datos mostrados en esta tabla indican que no existe una relación directa entre la precisión de los datos de entrada y el tiempo de ejecución, por lo tanto para la evaluación de la fases de reconocimiento de patrones se ha utilizado la mejor versión obtenida en la fase anterior.

Tabla 3.5: Estudio de precisión: Fase III - Rellenado de regiones.

Precisión	Dilation - *Execution time* (HW / SW)	Erosion - *Execution time* (HW / SW)	Transfer time (HW)
uint5	3.87 / 4.23	3.91 / 5.88	0.20
uint6	4.18 / 4.23	4.77 / 5.88	0.20
uint7	3.75 / 4.23	3.98 / 5.88	0.22
uint8	4.30 / 4.23	3.97 / 5.88	0.23

En cuanto al aprovechamiento de recursos en los diferentes experimentos probados sobre esta fase, no se han obtenido diferencias significativas en los indicadores de ocupación de la FPGA estudiados. Como se puede observar en el *report* 3 el cuello de botella asociado al número de LABs vuelve a ser un factor importante a tener en cuenta ya que según los datos se necesitaría una FPGA con alrededor de 5,000 LABs para procesar la operación de *Closing* en único modelo.

[8] Los tiempos de ejecución mostrados en esta tabla no incluyen los tiempos de envío y recepción (expuestos en la columna *transfer*)

Report 3. *Resource report*: Fase III - Rellenado de regiones

***Resource report* - Simulink** **Unit - Dilatación**	**Type**	**Unit - Erosión**
0	Multipliers	0
40	Adders and Subtractors	40
492	Registers	497
12	RAMs	12
219	Multiplexers	219
18	I/O Bits	18
***Resource report* - Quartus** **Unit - Dilatación**	**Type**	**Unit - Erosión**
8347	Total register	8533
125,832 / 4,065,280 (3.09 %)	Total block memory bits	163,496 / 4,065,280 (4.02 %)
11,649 / 32,070 (36.32 %)	Logic usage (in ALMs)	11,736 / 32,070 (36.59 %)
0 / 87 (0 %)	Total DSP Blocks	0 / 87 (0 %)
1 / 6 (16.66 %)	Total PLLs	1 / 6 (16.66 %)
2,360 / 3,207 (73.58 %)	Total LABS	2,393 / 3,207 (74.61 %)

3.4. Reconocimiento de patrones

En esta sección se van a comentar los resultados obtenidos en los estudios realizados en relación a la parte de reconocimiento de patrones. Aunque se han podido obtener unos datos de validación de los resultados, es necesario tener en cuenta que estos son parcialmente especulativos ya que para la realización de este proyecto no se ha podido contar con un corpus etiquetado por un experto.

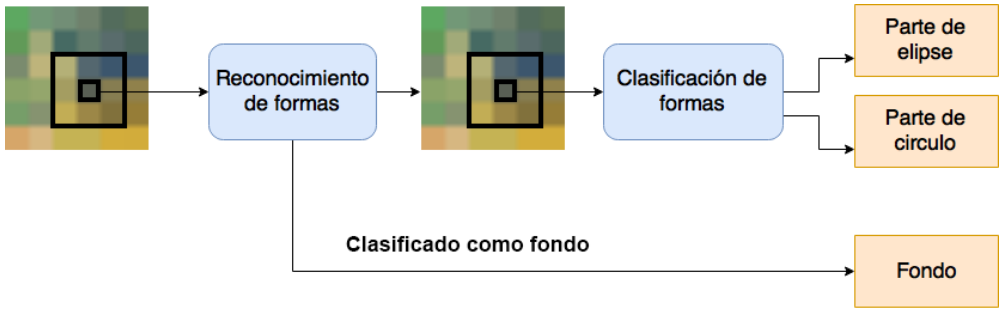


Figura 3.6: Resumen esquema reconocimiento de patrones.

Como se ha comentado, se ha dividido el proceso de identificación morfológico de una

célula en dos partes (tal y como se muestra en la figura 3.6). La primera es el reconocimiento o identificación de formas (es posible hacer la traducción del término "forma" a "célula" asumiendo como correctas las anteriores fases del proyecto). La segunda y última fase es la clasificación de las distintas células identificadas.

3.4.1. Reconocimiento de formas

Una matriz de confusión es una herramienta que permite la visualización del desempeño de un algoritmo que se emplea en aprendizaje supervisado. Cada fila de la matriz representa el número de predicciones de cada clase (*Output Class*), mientras que cada columna representa a las instancias en la clase real (*Target Class*).

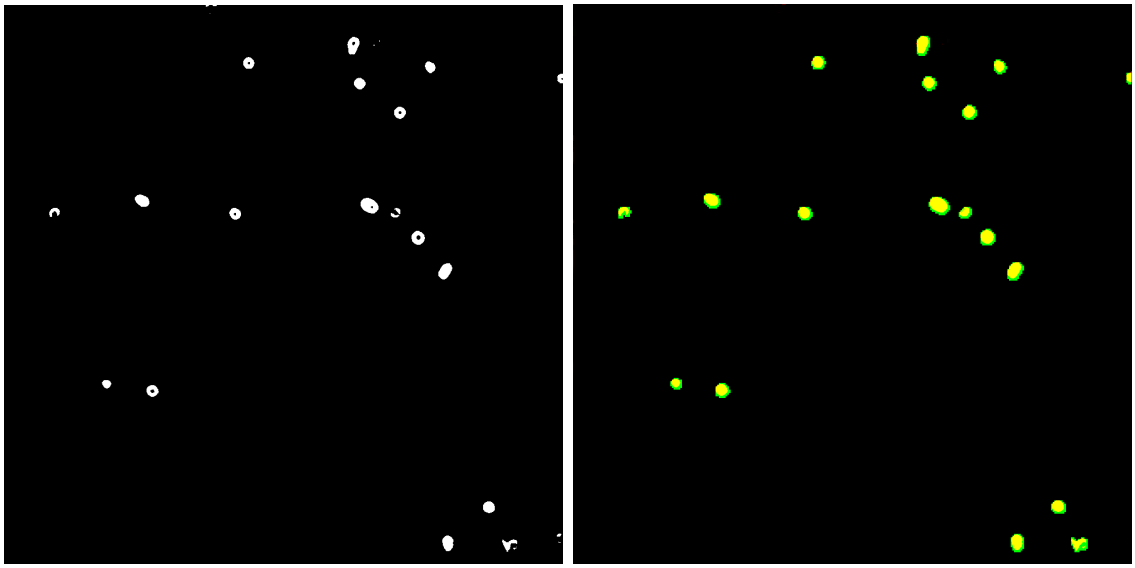
Tabla 3.6: Matriz de confusión: Identificación de formas.

A la vista de los datos mostrados en la matriz de confusión (figura 3.6), se puede observar que estamos ante un problema de clasificación desbalanceado ya que existen muchas instancias de la clase considerada como fondo (clase 1) y pocas de la clase considerada como forma (clase 2). Examinando las imágenes utilizadas, este hecho era algo obvio.

Tabla 3.7: Estudio de precisión: Fase IV - Identificación de formas.

	Software version	*Hardware version*
Execution time (hours)	12.31	9.46

Si nos fijamos en los tiempos obtenidos (tabla 3.7), podemos observar que en ambos casos estos son muy altos. Este hecho es debido a que se ha utilizado un algoritmo que realiza tantas llamadas a la red neuronal como píxeles hay en una imagen, todo esto con el fin de obtener los datos de rendimiento de la FPGA ante un algoritmo con alto coste computacional. Si nos fijamos en las diferencias obtenidas, se produce una notable mejora al hacer uso de la FPGA.



(a) Resultado fase III. (b) Identificación de formas.

Figura 3.7: Comparativa: Identificación de formas.

En la figura 3.7 podemos encontrar un ejemplo de funcionamiento del reconocedor de formas. En esta podemos observar en amarillo los píxeles que el sistema ha identificado como formas (coincidiendo con la realidad) y en verde los píxeles que el sistema ha reconocido de forma añadida.

Mientras que los datos muestran una clasificación de lo que se ha considerado como forma prácticamente perfecta, estos muestran que las características utilizadas para la definición de una instancia en el caso de la clase fondo (clase 1) no son del todo perfectas. Si volvemos a lo comentado sobre la figura 3.7 se puede intuir que el error obtenido se concentra en los limites de las células.

Report 4. *Resource report*: Fase IV - Identificación de formas

Neural network configuration	
Unit	**Type**
1x9	Neural network inputs
2	Output layer neurons
1	Hidden layers
6	Hidden layer neurons
Si	Enabled Bias
***Resource report* - Simulink**	
Unit	**Type**
66	Multipliers
89	Adders and Subtractors
0	Registers
0	RAMs
56	Multiplexers
122	I/O Bits
***Resource report* - Quartus**	
Unit	**Type**
1563	Total register
113,296 / 4,065,280 (2.78 %)	Total block memory bits
16,479 / 32,070 (51.38 %)	Logic usage (in ALMs)
66 / 87 (75.86 %)	Total DSP Blocks
1 / 6 (16.66 %)	Total PLLs
2,088 / 3,207 (65.10 %)	Total LABs

Como se puede observar en el *report* 4 y en lo que se refiere a los datos de aprovechamiento de la FPGA por parte del diseño, tenemos un reparto más equitativo del consumo de recursos. Todavía es destacable el consumo de LABs pero ahora cobra algo más importancia los datos referentes al consumo de DSPs en comparación con otros modelos. Las posibilidades de crecimiento del modelo actual de red neuronal son bastante interesantes. Durante la realización de esta fase se ha visto que el número de LABs necesarios está fuertemente relacionado con la cantidad de datos de entrada. Aunque en el diseño actual sería sencillo incrementar el número de neuronas, debido al reducido coste asociado a los datos de entrada, a la vista de los resultados obtenidos no es necesario aumentar la complejidad del modelo diseñado.

3.4.2. Clasificación de formas

Tal y como se ha comentado, la clasificación de formas tiene dos partes aplicadas secuencialmente, una basada en redes neuronales y otra basada en un algoritmo de votación por mayoría. A continuación se muestran los datos que se han obtenido en cada una de las partes (primero en los resultados de la red neuronal - tabla 3.8 - y después al aplicar la votación para clasificar una figura concreta - tabla 3.10). La matriz de confusión relativa a la clasificación de patrones dentro de una imagen (tabla 3.8) muestra que se produce un error muy grande al realizar una clasificación en dos clases. Tras evaluar estos datos y constatar que los datos utilizados no eran lo suficientemente relevantes como para hacer una clasificación únicamente basado en estos, se lanzó una hipótesis con el de mejorar la respuesta a esta confusión. Esta hipótesis alegaba que las posibles discrepancias dentro del algoritmo podían venir de la clasificación de las zonas interiores de una célula. Este hecho es de importancia si tenemos en cuenta que las ventanas utilizadas como entrada en este experimento son de un tamaño muy inferior al de una célula (6x6).

Tabla 3.8: Matriz de confusión: Resultado de clasificación de patrones dentro de una imagen.

	Class 1 (Part of Circle)	*Class 2* (Part of Ellipse)	*Classification overall*	*Accuracy*
Class 1 (Part of Circle)	3151	1646	4797	65.68 %
Class 2 (Part of Ellipse)	2549	3670	6219	59.01 %
Truth overall	5700	5316	11016	
Recall	55.28 %	69.03 %		
Kappa	0.24	*Overall accuracy*	61.91 %	

Tabla 3.9: Estudio de precisión: Fase IV - Clasificación entre tipos de forma.

	Software version	*Hardware version*
Execution time (minutes)	6.05	4.86

En lo que se refiere a los tiempos de ejecución obtenidos (tabla 3.9), éstos no son tan elevados como en el anterior caso. Esto es debido a que el número de píxeles a tratar es mucho menor en este caso. En cuanto a la diferencia de tiempos, como podemos observar, vuelven a destacar los tiempos obtenidos por la versión *hardware*, pero no tanto como era de esperar.

Tabla 3.10: Matriz de confusión: Resultado de aplicar sistema de votación a la clasificación realizada por la red neuronal para la determinación de la forma de una célula.

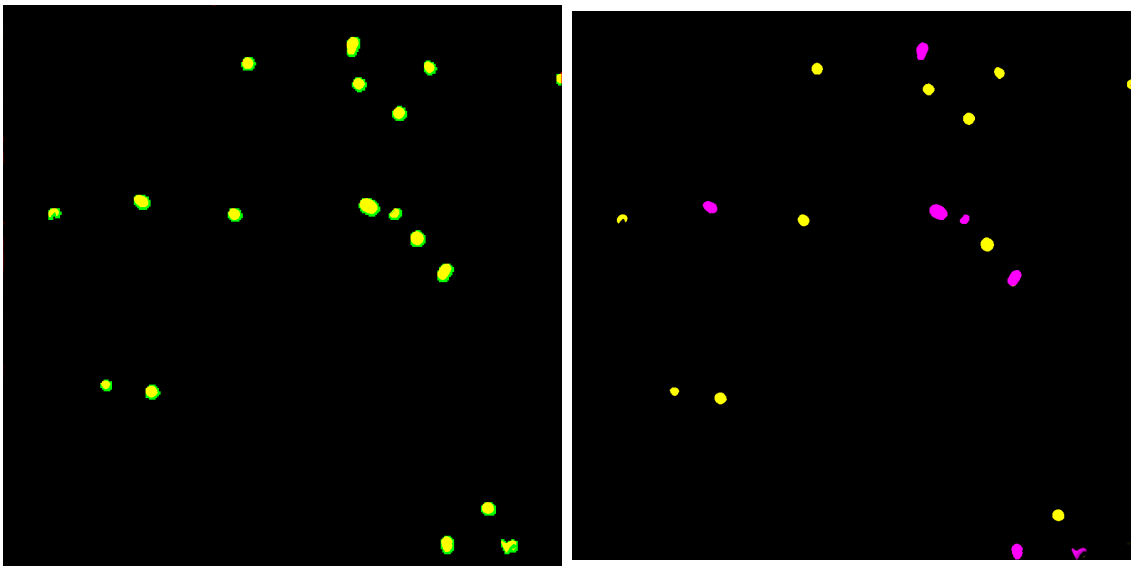
	Class 1 (Circle)	*Class 2* (Ellipse)	*Classification overall*	*Accuracy*
Class 1 (Circle)	46	18	64	71.87 %
Class 2 (Ellipse)	8	30	38	78.94 %
Truth overall	54	48	102	
Recall	85.18 %	62.5 %		
Kappa	0.48	*Overall accuracy*	74.51 %	

Con el fin de obtener una mejor clasificación e intentar demostrar la anterior hipótesis se planteó un diseño de sistema de votación en el cual tuvieran más importancia los píxeles que marcaran el limite de una figura o célula. Los datos de esta clasificación son los mostrados en la tabla 3.10. Como se observa, esta clasificación mejora las expectativas obtenidas haciendo uso únicamente de la red neuronal, no obstante añade una capa adicional de complejidad a tener en cuenta.

Report 5. *Resource report*: Fase IV - Clasificación de formas

Neural network configuration	
Unit	**Type**
1x36	Neural network inputs
2	Output layer neurons
1	Hidden layers
7	Hidden layer neurons
Si	Enabled Bias
***Resource report* - Simulink**	
Unit	**Type**
266	Multipliers
293	Adders and Subtractors
0	Registers
0	RAMs
68	Multiplexers
340	I/O Bits
***Resource report* - Quartus**	
Unit	**Type**
5562	Total register
113,296 / 4,065,280 (2.78 %)	Total block memory bits
25,004 / 32,070 (77.96 %)	Logic usage (in ALMs)
87 / 87 (100 %)	Total DSP Blocks
1 / 6 (16.66 %)	Total PLLs
3,207 / 3,207 (100 %)	Total LABs

Los datos mostrados en el *report* 5 indican aumentos destacables en el uso de elementos LABs y DSPs debido al crecimiento del tamaño del vector de entrada con respecto al anterior modelo de red neuronal. Estos datos dan pie a pensar que con el actual diseño no hay posibilidad de aumentar la complejidad de la red neuronal incrementando el número de neuronas por capa.



(a) Identificación de formas.

(b) Clasificación de formas.

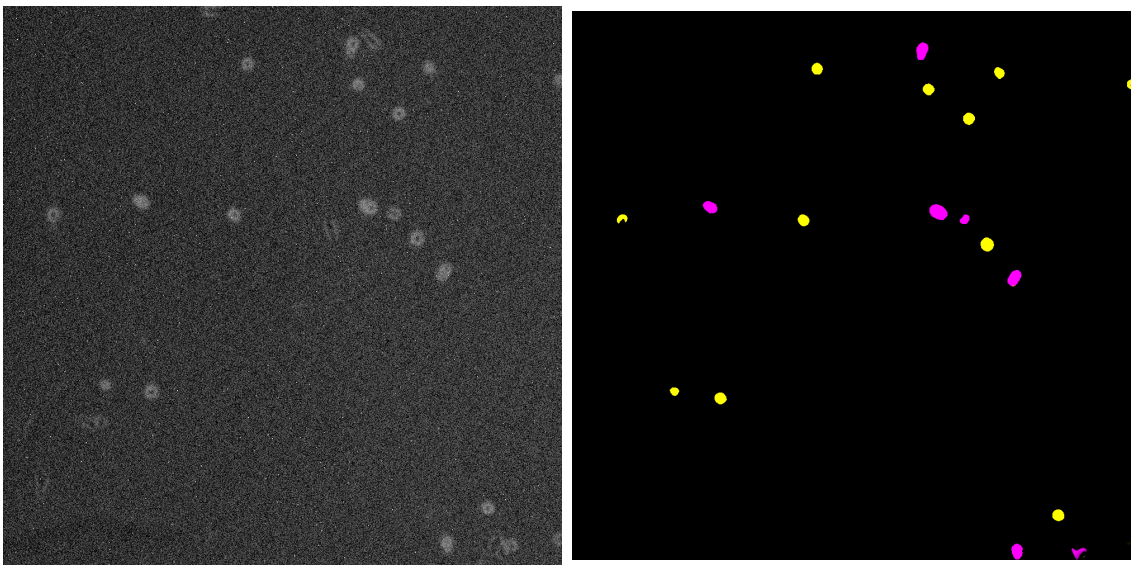
Figura 3.8: Comparativa: Clasificación de formas.

En la figura 3.8, se puede observar en el margen derecho el resultado de clasificar las figuras identificadas en la imagen de la izquierda (figura 3.8a y figura 3.8b). Las figuras clasificadas como elipses tienen color lila y las clasificadas como círculos tienen color amarillo. Al haber identificado algunos artefactos como formas, este segundo clasificador identifica estos artefactos como elementos dentro de las clases estudiadas.

Capítulo 4

Conclusiones y trabajo futuro

Si resumimos lo logrado en este proyecto tenemos por un lado un sistema de procesamiento de imágenes y por otro lado un sistema encargado de reconocer y clasificar patrones dentro de imágenes procesadas. El sistema de procesamiento de imágenes está enfocado en la mejora y la extracción de características para su posterior paso como datos de entrada al sistema reconocedor de patrones. Este último sistema está basado en dos clasificadores encargados de identificar y clasificar partes de una imagen en función de la morfología del objeto que describen.



(a) Imagen original. (b) Resultado fase IV

Figura 4.1: Resultado experimentación: versión original / versión clasificada

Son muchas las posibles conclusiones que se pueden obtener de un trabajo como éste. Si comenzamos evaluando los resultados obtenidos de forma global, tal y como se muestra en la figura 4.3, tenemos un sistema que es capaz de procesar imágenes y clasificar ciertos patrones dentro de estas (en el caso de identificación de formas se han tenido mejores resultados que posteriormente en la clasificación de formas). Un dato importante a tener cuenta a la hora de evaluar la relevancia de estos resultados es que no se han podido validar con ningún experto las decisiones tomadas tanto en la creación del corpus como en la implementación de los diferentes diseños. Este hecho hace que aún siendo de interés los resultados obtenidos a nivel de precisión, estos tienen ciertas componentes especulativas que hay que tener en cuenta.

Por otro lado y como ya se ha comentado, debido tanto a las limitaciones propias de la FPGA como a las inherentes al tipo de licencia utilizada del software *Quartus*, no se ha podido obtener un proyecto basado en único modelo. Básicamente esto es debido a que el total de *logic array blocks* necesarios para obtener la solución diseñada es demasiado alto, unos 10,420 *logic array blocks*. Si analizamos ahora los resultados intermedios obtenidos, nos encontramos que todos los modelos tienen un cuello de botella en lo que se refiere al uso de LABs, lo que hace imposible acoplarlos entre ellos con el fin de tener un solución basada en un número menor de modelos. El número de modelos diferentes hace que los tiempos de llenado y vaciado del bus de comunicaciones pasen de ser poco relevantes a ser un factor diferenciador (como se puede observar en las tablas 3.5, 3.4 y 3.2). En cuanto al diseño de un único modelo *Simulink*, como posible trabajo futuro y asumiendo como invariante el flujo de trabajo planteado, se puede barajar la posibilidad de trabajar con un modelo de FPGA con mejores especificaciones. Por otro lado y suponiendo el uso de *Quartus*, se puede plantear el uso de un CPLD como el MAX V que permita la optimización de la fase de *synthesis* y de *fitter* para la obtención de un modelo basado en la ocupación de la mínima área posible. Si observamos los *reports* obtenidos, estos muestran un bajo porcentaje de ocupación de elementos del tipo ALM en la mayoría de los modelos planteados. De conseguir permitir el

uso compartido de elementos lógicos o de aumentar el uso de ALMs como alternativa a los *logic array blocks*, sería interesante la realización de un estudio similar al presente.

Si procedemos a plantear cambios sobre la actual solución, podemos encontrar propuestas que van desde la reorganización de los modelos hasta la simplificación de la solución. Un ejemplo lo podemos encontrar si nos fijamos en la problemática existente con las restricciones de diseño. Aunque en la fase II se ha planteado una posible simplificación del modelo, se podría contemplar como otra posibilidad el hecho de centrar la atención en cuestionar la aplicación del algoritmo de *Closing*. En cierto modo, esto es debido a que su aplicación tiene un bajo nivel de impacto en el resultado final (véase figura 4.4) en comparación con otras partes del proyecto.

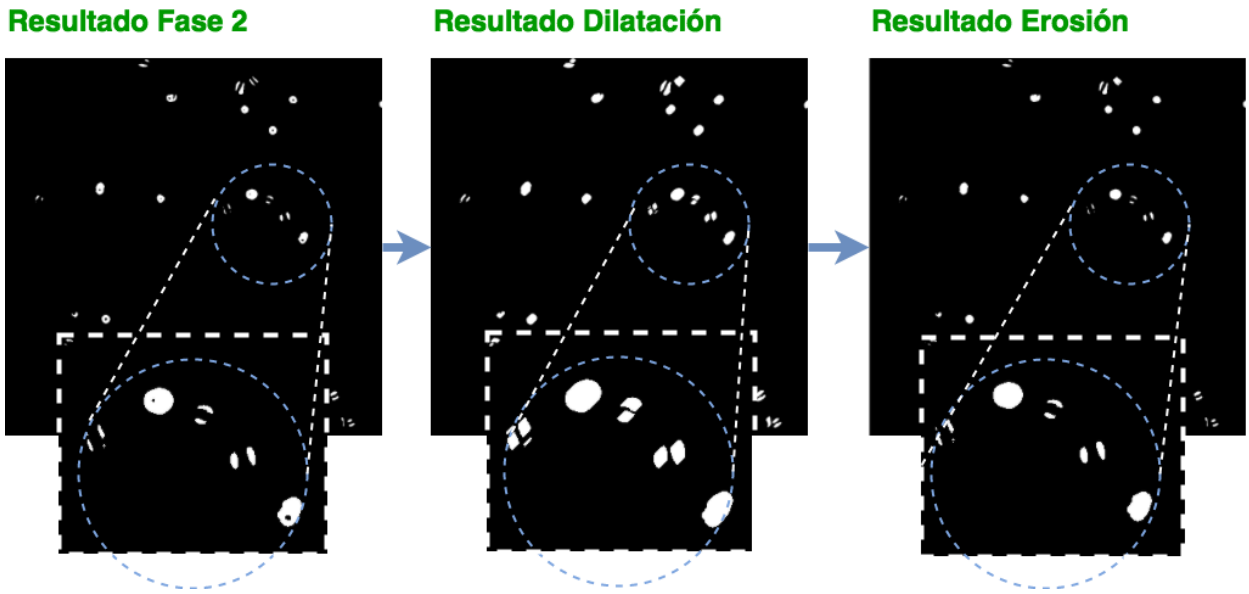


Figura 4.2: Comparativa: Resultado Fase II ->Dilatación ->Erosión.

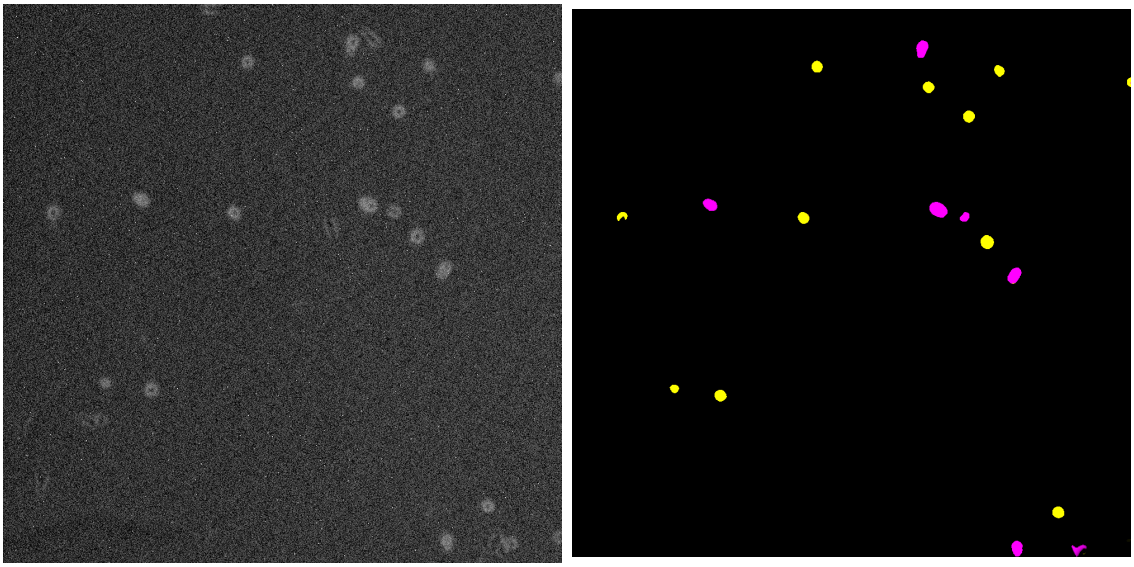
Por último, en lo que se refiere al modelo de aprendizaje automático planteado, aunque existen muchos estudios que tratan acerca de reconocimiento de patrones en imágenes, el número de estos disminuye conforme lo hace el dominio de los datos de entrada (en ciertos dominios resulta una ventaja trabajar con una escala RGB). Aún así, existen algunos proyectos en el campo del aprendizaje automático sobre imágenes que obtienen resultados

bastante interesantes en el proceso de segmentación de imágenes microscópicas. Algunos de estos proyectos se fundamentan en el uso del cálculo de la superficie asociada a la célula como característica discriminante o ya más cercano a este proyecto, el uso del patrón descrito por la disposición de la membrana de la célula y ciertos atributos de esta[6].

Los resultados asociados a este proyecto han sido aceptados para el *Summer Simulation Multiconference 2017*[7].

Conclusions and future work

If we summarize the achievements of this project, we have on one side, an image processing system being focused on the improvement and extraction of characteristics and, on the other side, a pattern recognition system based upon two classifying parts of an image in accordance with the morphology of the object which they describe.



(a) Original image. (b) Phase IV result.

Figura 4.3: Experiment result: original version / classified version.

A project like this one could have several possible conclusions. Should we start valuating the results being obtained in a global form as shown in figure 4.3, we have a system which is capable of image processing and of classifying certain patterns within the said images (in the case of identification of shapes, better results have been obtained than in classification of forms afterwards). Due to the fact that decisions made both in the corpus creation as in the implementation of the different designs have not been validated with an expert and even though the results obtained at a precision level in this experiment are of certain interest,

these result have certain speculative components which should be taken into account.

On the other hand, as already stated, due to the FPGA limitations, due to the limitations of the used licence of Quartus software and due to the limitations of the designed solution, no solution based on a sole model has been obtained. Basically this is due to the fact that the total number of logic array blocks being needed to obtain the same is much too high: some 10,420 logic array blocks.

If we analyse now the intermediate results being obtained, we have had to divide the solution into different designs due to the bottle neck existing in the implemented designs as far as the use of LABs is concerned. This fact makes the filling and emptying ties of the communication bus change from being little relevant to becoming a differentiating factor (tables 3.5, 3.4 and 3.2). As possible future work and assuming that the planned flow is invariable, with regard to the design of an only Simulink model we can contemplate the possibility of working with an FPGA model with better specifications. On the other hand and assuming the use of Quartus, the use of an CPLD as MAX V can be contemplated which may allow optimization of the synthesis and fitter phases in order to obtain a model based on the occupation of the minimum possible area whether allowing the shared use of logical elements or increasing the use of ALMs thus allowing the reduction of the use of logic array blocks as the reports being obtained show a low percentage of occupation of elements of the ALM type.

Should we introduce changes in the current solution we can find proposals ranging from the reorganization of models to the simplification of the solution. For example if we look at the existing problems with the design restrictions due to the number of LABs and although in phase 2 a possible simplification has been contemplated, it could be accepted as another possibility the fact of putting our scope in questioning the application of "Closing" algorithm due to its low impact level in the final result (figure 4.4).

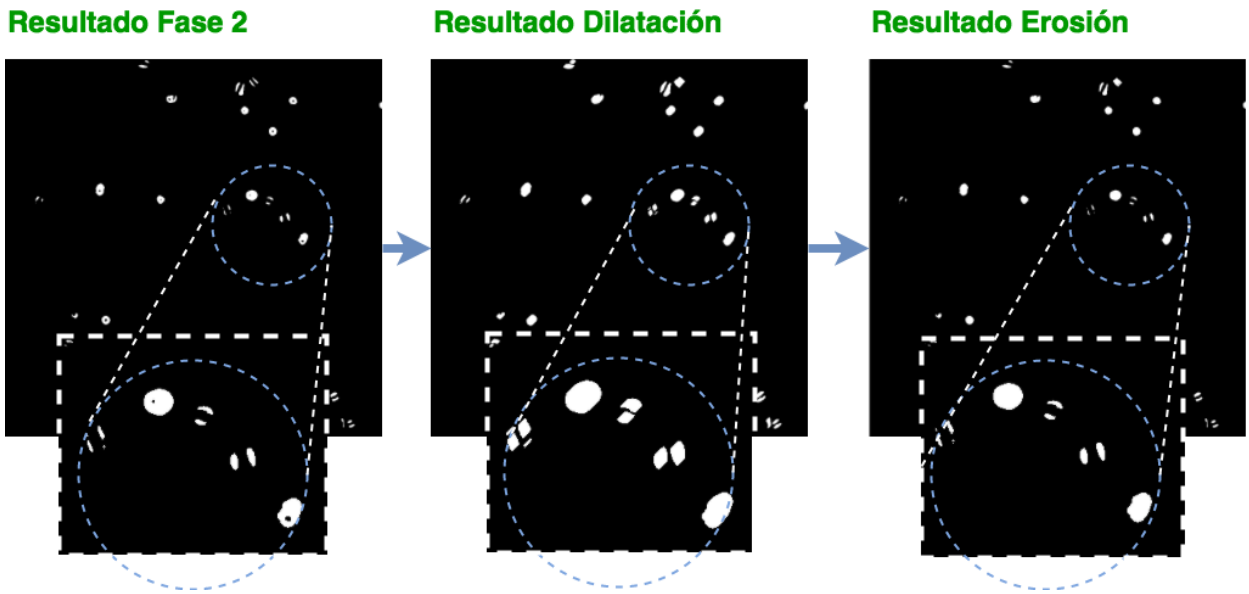


Figura 4.4: Comparison: Phase II result ->Dilation ->Erosion.

Finally, as far as the machine learning model is concerned, although there are numerous studies which deal with image pattern recognition, the number of studies diminish in accordance with the diminution of the input data domain (in certain domains it is an advantage to work with an RGB scale). Even so there exist several approximations in this field which obtain quite interesting results in the cell segmentation process, for example, the use, as a discerning characteristic, of the calculus of surface associated to the cell or as shown in this project the use of the pattern described by the disposition of the membrane of the cell and some attributes of it[6].

Certain results of this project have been accepted to be present in the Summer Simulation Multiconference 2017[7].

Bibliografía

[1] Ahmed S Abutaleb. Automatic thresholding of gray-level pictures using two-dimensional entropy. *Computer vision, graphics, and image processing*, 47(1):22–32, 1989.

[2] Jesús Martín Alonso and Alberto A. Del Barrio. A distributed hw-sw platform for fireworks. In *Proceedings of the Summer Computer Simulation Conference*, SCSC '16, pages 17:1–17:7, San Diego, CA, USA, 2016. Society for Computer Simulation International.

[3] M. Bromberger, P. Bastian, J. P. Bergeest, C. Conrad, V. Heuveline, K. Rohr, and W. Karl. Fpga-accelerated richardson-lucy deconvolution for 3d image data. In *2016 IEEE 13th International Symposium on Biomedical Imaging (ISBI)*, pages 132–135, April 2016.

[4] Nabendu Chaki, Soharab Hossain Shaikh, and Khalid Saeed. *Exploring Image Binarization Techniques*. Springer Publishing Company, Incorporated, 2014.

[5] James C Church, Yixin Chen, and Stephen V Rice. A spatial median filter for noise removal in digital images. In *Southeastcon, 2008. IEEE*, pages 618–623. IEEE, 2008.

[6] Sotiris Dimopoulos, Christian E. Mayer, Fabian Rudolf, and Joerg Stelling. Accurate cell segmentation in microscopy images using membrane patterns. *Bioinformatics*, 30(18):2644, 2014.

[7] Hermenegildo Fabregat, Alberto A. Del Barrio, and Guillermo Botella. Simulation and implementation of a low-cost platform to improve the quality of biological images. In *Part of the 2017 Summer Simulation Multiconference, SummerSim 2017*, 2017.

[8] Lingkan Gong and Oliver Diessel. *Functional Verification of Dynamically Reconfigurable FPGA-based Systems*. Springer, 2014.

[9] J. C. T. Hai, O. C. Pun, and T. W. Haw. Accelerating video and image processing design for fpga using hdl coder and simulink. In *2015 IEEE Conference on Sustainable Utilization And Development In Engineering and Technology (CSUDET)*, pages 1–5, Oct 2015.

[10] Thamer M Jamel and Ban M Khammas. Implementation of a sigmoid activation function for neural network using fpga. In *inProceed-ings of the 13th Scientific Conference of Al-Ma?moon University College, Baghdad, Iraq*, 2012.

[11] Bhat Jasra, Aniqa Yaqoob, and Sanjay Kumar Dubey. Removal of high density salt and pepper noise using bpann-modified median filter technique. In *Cloud System and Big Data Engineering (Confluence), 2016 6th International Conference*, pages 78–82. IEEE, 2016.

[12] Liu Jianzhuang, Li Wenqing, and Tian Yupeng. Automatic thresholding of gray-level pictures using two-dimension otsu method. In *Circuits and Systems, 1991. Conference Proceedings, China., 1991 International Conference on*, pages 325–327. IEEE, 1991.

[13] N. N. Kachouie, P. Fieguth, and E. Jervis. Stem-cell localization: A deconvolution problem. In *2007 29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 5525–5528, Aug 2007.

[14] Shahram Karimi, Philippe Poure, Yves Berviller, and Shahrokh Saadate. A design methodology for power electronics digital control based on an fpga in the loop prototyping. In *Electronics, Circuits and Systems, 2007. ICECS 2007. 14th IEEE International Conference on*, pages 701–704. IEEE, 2007.

[15] Y. Lv, G. Jiang, M. Yu, H. Xu, F. Shao, and S. Liu. Difference of gaussian statistical

features based blind image quality assessment: A deep learning approach. In *2015 IEEE International Conference on Image Processing (ICIP)*, pages 2344–2348, Sept 2015.

[16] J. Mantas. An overview of character recognition methodologies. *Pattern Recognition*, 19(6):425 – 430, 1986.

[17] T. Mazanec, A. He?mánek, and J. Kamenický. Blind image deconvolution algorithm on nvidia cuda platform. In *13th IEEE Symposium on Design and Diagnostics of Electronic Circuits and Systems*, pages 125–126, April 2010.

[18] Anke Meyer-Baese. *Pattern recognition for medical imaging*. Academic Press, 2004.

[19] Amaleena Mohamad, Noorain A Jusoh, Lei Win Shoon, et al. Bacteria identification from microscopic morphology: a survey. *International Journal on Soft Computing, Artificial Intelligence and Applications (IJSCAI)*, 3(2):1–12, 2014.

[20] A.J. NELSON and S.T. HESS. Localization microscopy: mapping cellular dynamics with single molecules. *Journal of Microscopy*, 254(1):1–8, 2014.

[21] E. Osuna, R. Freund, and F. Girosit. Training support vector machines: an application to face detection. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 130–136, Jun 1997.

[22] Carlos Paiz, Christopher Pohl, Rafael Radkowski, Jens Hagemeyer, Mario Porrmann, and Ulrich Rückert. FPGA-in-the-Loop-Simulations for Dynamically Reconfigurable Applications. In *Proceedings of the 2009 International Conference on Field-Programmable Technology (FPT'09)*, pages 372–375, 2009.

[23] David Perez-Guaita, Dean Andrew, Philip Heraud, James Beeson, David Anderson, Jack Richards, and Bayden R. Wood. High resolution ftir imaging provides automated discrimination and detection of single malaria parasite infected erythrocytes on glass. *Faraday Discuss.*, 187:341–352, 2016.

[24] WK Pratt. Digital image processing. 1978. *Publisher John Wiley & Sons*.

[25] J Jezebel Priestley, T Anusuya, R Pratheepa, and V Elamaran. Salt and pepper noise reduction with a novel approach of noise models using median filter. In *Computational Intelligence and Computing Research (ICCIC), 2014 IEEE International Conference on*, pages 1–4. IEEE, 2014.

[26] Brian Randell. The 1968/69 nato software engineering reports. *History of Software Engineering*, page 37, 1996.

[27] Matthias Rieckher. Light sheet microscopy to measure protein dynamics. *Journal of Cellular Physiology*, 232(1):27–35, 2017.

[28] José L. Risco-Martín, Saurabh Mittal, Juan Carlos Fabero, Pedro Malagón, and José L. Ayala. Real-time hardware/software co-design using devs-based transparent m&s framework. In *Proceedings of the Summer Computer Simulation Conference*, SCSC '16, pages 45:1–45:8, San Diego, CA, USA, 2016. Society for Computer Simulation International.

[29] Pablo Ruiz, Xu Zhou, Javier Mateos, Rafael Molina, and Aggelos K Katsaggelos. Variational bayesian blind image deconvolution: A review. *Digital Signal Processing*, 47:116–127, 2015.

[30] Riries Rulaningtyas, Andriyan Bayu Suksmono, and Tati LR Mengko. Automatic classification of tuberculosis bacteria using neural network. In *Electrical Engineering and Informatics (ICEEI), 2011 International Conference on*, pages 1–4. IEEE, 2011.

[31] E. T. Scott and S. S. Hemami. Image utility estimation using difference-of-gaussian scale space. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 101–105, Sept 2016.

[32] S Kamaledin Setarehdan and Sameer Singh. *Advanced algorithmic approaches to medical image segmentation: state-of-the-art applications in cardiology, neurology, mammography and pathology*. Springer Science & Business Media, 2012.

[33] K Silpa and S Aruna Mastani. Comparison of image quality metrics. In *International Journal of Engineering Research and Technology*, volume 1. ESRSA Publications, 2012.

[34] Violet Snell, W Christmas, and Josef Kittler. Texture and shape in fluorescence pattern identification for auto-immune disease diagnosis. In *Pattern Recognition (ICPR), 2012 21st International Conference on*, pages 3750–3753. IEEE, 2012.

[35] María Teresa Taranilla, Gustavo Kavka, Edilma Olinda Gagliardi, and Gregorio Hernández Peñalver. Una operación entre polígonos: Sumas de minkowski. In *VIII Congreso Argentino de Ciencias de la Computación*, 2002.

[36] Inc. ©1994-2017 The MathWorks. Generate verilog and vhdl code for fpga and asic designs, 2017.

[37] Inc. ©1994-2017 The MathWorks. Simulink and model based design, 2017.

[38] L. Vincent. Morphological grayscale reconstruction in image analysis: applications and efficient algorithms. *IEEE Transactions on Image Processing*, 2(2):176–201, Apr 1993.

[39] L. Zhang, Q. Wang, and J. Qi. Research based on fuzzy algorithm of cancer cells in pleural fluid microscopic images recognition. In *2006 International Conference on Intelligent Information Hiding and Multimedia*, pages 211–214, Dec 2006.