UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

PROGRAMA DE MAESTRÍA Y DOCTORADO EN INGENIERÍA
INGENIERÍA ELÉCTRICA – SISTEMAS ELECTRÓNICOS

DESARROLLO DE UN SISTEMA EN FPGA PARA ENSAMBLE ROBOTIZADO GUIADO POR VISIÓN.

TESIS
QUE PARA OPTAR POR EL GRADO DE:
MAESTRO EN INGENIERÍA

PRESENTA:
ING. ADRIAN JIMÉNEZ HERNÁNDEZ

TUTOR PRINCIPAL
DR. JUAN MARIO PEÑA CABRERA, IIMAS

MÉXICO, D.F. NOVIEMBRE 2015

JURADO ASIGNADO:

Presidente:	Dra. Elsi Violeta Mejía Uriarte
Secretario:	Dr. Saúl De La Rosa Nieves
Vocal:	Dr. Juan Mario Peña Cabrera
1er. Suplente:	M.I. Jesús Álvarez Castillo
2do. Suplente:	Dr. Jorge Prado Molina

Lugar o lugares donde se realizó la tesis:

Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas
Universidad Nacional Autónoma de México

TUTOR DE TESIS:

Dr. Juan Mario Peña Cabrera

FIRMA

"Un viaje de mil millas comienza con el primer paso."

Lao-tsé

Agradecimientos

Al Dr. Juan Mario Peña Cabrera por el tiempo, los consejos y conocimientos compartidos, que fomentaron la realización de este trabajo.

A la Ing. Martha P. Lara Mendoza, por las ideas, sugerencias y el apoyo que impulsaron el logro de esta meta.

A los doctores y maestros por el tiempo dedicado a la revisión de esta tesis.

A la Coordinación de Estudios de Posgrado de la UNAM (CEP-UNAM) por la beca parcial de maestría otorgada.

Al Consejo Nacional de Ciencia y Tecnología (CONACYT) por la beca de maestría otorgada.

A mi familia por ser el punto de partida de mi vida y por todo su incondicional apoyo.

Resumen

En este trabajo de tesis se desarrolla un sistema de visión artificial en un dispositivo lógico programable o por su nombre en inglés field programmable gate array (FPGA), para la localización y el ensamble por inserción (peg in hole) de piezas con formas geométricas simples. El componente central es la tarjeta de desarrollo Nexys 4, donde se escribe en lenguaje descriptor de hardware (HDL) el comportamiento de todo el sistema. Se utiliza una cámara que captura y transmite las imágenes a la tarjeta, donde se aplican técnicas para realzar la información, encontrar características morfológicas de las piezas, así como la ubicación y el reconocimiento de éstas. Posteriormente con estos datos es posible generar las instrucciones que se enviarán al robot para realizar el ensamble. En las pruebas se utilizaron objetos de juguete y el ensamble se realizó satisfactoriamente en el 50 % de los casos mediante la información del área y centroide; el cálculo de este último en las piezas, mostró una precisión de 6 milímetros.

Contenido

Índice de Figuras

Índice de Tablas

Nomenclatura

Siglas / Abreviaturas

BOF Boundary Object Function.

BRAM Block Random Access Memory.

CIF Common Intermediate Format.

CLB Configurable Logic Block.

CMOS Complementary metal-oxide semiconductor.

CNC Computer Numerical Control.

DSP Digital signal processor.

FPGA Field programmable gate array.

FPS Fotogramas Por Segundo.

GDL Grados De Libertad.

LC Logic Cell.

LUT Lookup table.

Peg in Hole Método de ensamble por inserción.

SCARA Selective Compliance Assembly Robot Arm.

SCCB Serial Camera Control Bus.

SoPC System On a Programmable Chip.

UART Universal Asynchronous Receiver-Transmitter.

VGA Video Graphics Array.

VHDL VHSIC Hardware Description Language.

VHSIC Very High Speed Integrated Circuit.

Capítulo 1

Introducción

En los procesos de ensamble y control de calidad, hay una gran necesidad de sistemas robóticos avanzados capaces de detectar, reconocer y sujetar objetos, con la finalidad de realizar ensambles en ambientes no estructurados y con objetos posicionados aleatoriamente. Normalmente, los robots de ensamble tradicionales, son programados para recoger una pieza de la misma posición todo el tiempo, pero si la pieza está ligeramente fuera de lugar el robot no podrá sujetar la pieza. De forma que se pueden tener ventajas significativas cuando se acoplan los sistemas de visión a estos robots, ya que gracias al reconocimiento avanzado con el que cuentan pueden ajustar las coordenadas del lugar donde se espera que el robot recoja la pieza y donde realmente se encuentra ésta. Los sistemas de ensamble robótico basados en visión, se han consolidado hasta el punto en el que pueden ser aplicados satisfactoriamente en las tareas de ensamble robótico y control de calidad; sin embargo, a la vez que los sistemas robóticos avanzados se vuelven más populares y utilizados en los ambientes de ensamble industriales, crece la necesidad de un funcionamiento más confiable con la menor cantidad posible de tiempo desperdiciado.

La economía de hoy en día se ha concentrado más en los productos personalizados que en la fabricación en masa; por lo que este tipo de economía es más dependiente de los sistemas que sean fáciles de utilizar, que sean altamente adaptables y que permitan lotes pequeños de productos hechos a la medida [1]. Muchos sistemas de ensamble, requieren el mantenimiento de expertos constantemente, y son menos seguros debido a su complejidad, esta es una razón del porque se requiere simplificar la parte mecánica y de programación en los procesos operados por robots. Implementar los sistemas de visión en las operaciones de ensamble y control de calidad es una tarea de muchas facetas que demanda de un vasto conocimiento en diversas

áreas y es común que requiera de soluciones a problemas específicos. Comúnmente un proceso de planeación y desarrollo de una operación de ensamble basada en visión artificial se divide en varias etapas y tareas, como la detección, el reconocimiento, el agarre, la medición, la detección de fallas, etc. Así como también se divide en la selección de componentes y en las condiciones del área de trabajo, como la cámara, las lentes, la iluminación, el equipo de cómputo, etc [2].

El control visual depende de la visión por computadora para manipular las acciones de un robot, sin embargo, los algoritmos de control visual, en especial los de procesamiento de imágenes son muy demandantes en cuestión de cómputo y transferencia de datos; si estas dos tareas no se realizan con un buen desempeño, el control visual se verá ralentizado y podría fallar en el cumplimiento de sus deberes. Para solucionar el problema de velocidad, se han utilizado estaciones de cómputo de alto rendimiento, pero estas pueden llegar a ser grandes, estorbosas, caras o demandar un alto consumo de energía. Una propuesta de trabajo para evitar las grandes estaciones de cómputo es por medio de los sistemas embebidos, los cuales son ligeros, pequeños y de bajo consumo de energía. Muchos trabajos de investigación en la actualidad han optado por utilizar tarjetas de desarrollo con procesadores de señales dedicados o con el uso de FPGAs, dependiendo de los algoritmos de imágenes y de las aplicaciones donde se vayan a utilizar [3] [4].

1.1 Objetivos

Desarrollo de un sistema de visión artificial en FPGA con el fin de ensamblar por medio de inserción (peg in hole), piezas geométricas sencillas desde el punto de vista parte/contraparte. El sistema deberá reconocer y localizar la pieza dentro de un campo de trabajo, así como su respectivo lugar de ensamble; para posteriormente agarrarla, trasladarla e insertarla utilizando un brazo robot.

Objetivos particulares

- Realizar el análisis de las imágenes para encontrar la posición de las piezas.
- Definir el comportamiento del robot con tal de llevar a cabo las funciones mencionadas anteriormente.

- Implementar el control de la adquisición y los algoritmos de procesamiento de las imágenes, al igual que las secuencias de movimiento para el manipulador, en el FPGA.
- Validar el funcionamiento general del sistema.

1.2 Antecedentes

1.2.1 Ensamble industrial

El ensamble manual de piezas es caro ya que precisa altos niveles de calidad, porque necesita trabajadores altamente cualificados, requiere de inspección y verificación para compensar errores humanos, toma más tiempo, y además, puede ser difícil y tedioso. El ensamble manual es el punto más débil de una cadena de producción debido a que esta actividad toma una parte del costo y del tiempo total de producción. Tomando en cuenta lo anterior y aunado a los recortes de precios de los robots, las mejoras en el desempeño de estos y en los sistemas de visión; es posible entender el crecimiento del ensamble robotizado [2].

El ensamble industrial forma parte de los procesos de producción y puede definirse como una serie de tareas ordenadas en las cuales piezas, partes y componentes se manipulan físicamente hasta emparejarse, unirse o insertarse para formar un objeto especificado, esta operación forma parte de los procesos de producción y juega un papel muy importante en la competitividad de la industria en general. El ensamble como parte de los sistemas de producción, implica el manejo de partes y sub ensambles, que se han fabricado en diferentes tiempos y posiblemente en lugares separados; por eso, las tareas de ensamble resultan de requerir unir piezas individuales, sub ensambles y sustancias en un ensamble final de mayor complejidad [5].

La parte central en el problema del ensamble es la etapa del ajuste entre las partes de este, como en el caso del ensamble *peg in hole*, es decir, aquellos ensambles donde se inserta una figura dentro de otra con la misma forma [6]. Si bien el objetivo de insertar una pieza dentro de otra es una acción trivial y simple de realizar para el ser humano; para una máquina es una tarea que representa numerosos obstáculos que superar y requiere de varios tipos de sensores y estrategias para cumplir con el objetivo.

En estas operaciones de ensamble en las que el robot entra en contacto físico con el medio ambiente, existen fuerzas de interacción que al igual que las medidas de posición deben controlarse; por lo que se necesita un control de la posición y uno de fuerza-torque ya que de

lo contrario alguna pieza puede dañarse cuando el robot ejerza mucha presión al realizar el ensamble.

Para llevar a cabo un ensamble por inserción con un manipulador se necesita la posición precisa de la cavidad donde se insertará la pieza, si esta no es circular se necesita también la información de la orientación. Comúnmente se utiliza un sensor de fuerza-torque para conocer tanto la posición como la orientación de la cavidad; con esta técnica se han realizado varios trabajos de investigación [7] [8]. Otras investigaciones buscan imitar el comportamiento humano al realizar un ensamble de inserción, el cual se puede describir como colocar la pieza en contacto y cerca de la cavidad, para después comenzar a deslizar la pieza contra la superficie hasta encontrar la cavidad y terminar el ensamble [9]. Incluso otras investigaciones han realizado ensambles de inserción en piezas circulares cuya holgura es de apenas $6\mu m$ y la repetibilidad del manipulador es de $10\mu m$ por medio de una técnica conocida como *Passive Alignment Principle* [10].

El ensamble de piezas con formas geométricas no regulares se ha realizado usando un sistema de visión y sensores de fuerza torque, en estos se utilizó la información geométrica obtenida de los modelos CAD de las piezas para determinar puntos clave en el contorno de estas y con la ayuda de la cámara se determinó la posición y orientación de las piezas, finalmente con el sensor de fuerza torque se controló la aplicación de esta en el ensamble; y así esa técnica logro un 100 % de éxito en las tareas de ensamble pero no es viable en ensambles donde el error de orientación entre las piezas es grande [11] [12].

Actualmente las tareas de ensamble tipo *peg in hole* utilizan los sistemas de visión para poder ubicar las piezas y su lugar de ensamble, y además se apoyan de los sensores de fuerza-torque cuando no es posible utilizar las cámaras como método sensorial.

1.2.2 Sistemas de visión

Los robots industriales actuales, realizan tareas de manejo y ensamble de partes con una velocidad y precisión excelente. Sin embargo, comparado con los operadores humanos, los robots se ven entorpecidos por su falta de percepción sensorial al realizar tareas más complejas en un entorno poco estructurado [13]. Actualmente se está estudiando el uso de la visión artificial para proveer de información a los robots con tal de tener un agarre confiable de los objetos y realizar las tareas de ensamble.

La visión artificial es la aplicación de la visión por computadora y estudios relacionados a la solución de problemas en la industria, entre ellos la automatización. Algunas aplicaciones incluyen sistemas de seguridad, control e inspección visual o cualquier otra actividad donde la visión juegue un rol importante. Los primeros robots inteligentes asistidos por visión aparecieron en Japón en 1970 en el laboratorio electrónico de MITI y en el de investigación central de Hitachi; cada uno era controlado por una minicomputadora cuyas capacidades eran limitadas comparadas con las actuales[14]. La primera aplicación satisfactoria del procesamiento de imágenes en la automatización industrial fue la búsqueda de defectos en las tarjetas de circuito impreso en 1972, y en 1973 fue desarrollado un sistema automático asistido por visión para unir cables en el ensamble de un transistor; el cual posteriormente fue ampliado para el ensamble de circuitos integrados [14].

Muchas actividades industriales se han beneficiado de las aplicaciones de la visión artificial en los procesos de manufactura, puesto que mejora la calidad, productividad y además, ofrece una ventaja competitiva a las industrias que la emplean. La visión artificial es ampliamente reconocida como una tecnología de innovación en cuestiones de supervisión y control de calidad, y se utiliza en varios sectores de la industria desde la agricultura, donde se usa para localizar plantíos de granos de maíz e inspeccionar la calidad de estas semillas [15]; hasta su uso en la industria farmacéutica en la cual se utiliza para la inspección en línea del correcto ensamble de hasta 600 jeringas por minuto [16].

Como ya se mencionó, uno de los problemas fundamentales en la manufactura asistida por robots, es la capacidad de localizar e identificar partes o piezas de tal manera que un robot pueda agarrar y manipular estas de forma segura y precisa. En general, estas partes se encuentran en un contenedor o en una cinta transportadora colocadas de forma aleatoria, por lo que son necesarios sistemas avanzados de percepción para identificar y localizar los objetos de interés. Esta tarea de percepción se conoce como *bin-picking* y ha sido un tema ampliamente estudiado debido al gran impacto que representa en la flexibilidad y productividad de las compañías manufactureras. Un ejemplo de estos trabajos es el realizado por Alberto Pretto que presenta un sistema de visión para ubicar los 6 GDL de piezas bidimensionales colocadas en un contenedor de forma aleatoria, el algoritmo para reconocer y ubicar las piezas se basa en los datos del CAD, y el margen de éxito de reconocer la ubicación de las piezas es de al menos 94 % [17].

Varios trabajos se centran en el reconocimiento y ubicación de objetos amontonados dentro de un contenedor, algunos usan múltiples cámaras, otros se centran en figuras complejas, en otros se definen puntos clave en las piezas que sirven de referencia para el reconocimiento, etc.; la mayoría de los trabajos se diferencian en el método empleado para hacer el reconocimiento de

los objetos y en la estrategia utilizada para determinar la posición 3D de estos [18] [19] [20] [21] [22] [23].

Un método fiable para reconocer los objetos es por medio del cálculo de la función frontera (BOF), el cual permite obtener características geométricas de los objetos sin importar, el tamaño, dirección y posición en que estén colocados; este método fue probado en un sistema que realizo el ensamble de piezas tipo *peg in hole* usando una red neuronal artificial encargada de realizar el reconocimiento de objetos [13].

Actualmente existen máquinas comerciales que utilizan el reconocimiento de objetos para identificar el lugar de colocación de estos, ya sea en posiciones fijas o en movimiento sobre una cinta transportadora, un ejemplo son los robots Epson que cuentan con sistemas de visión para asistir en las tareas de ubicación y transporte de objetos; incluso cuentan con un conjunto de operaciones para que el usuario seleccione las que mejor se adapten a su aplicación [24].

La tarea final de integrar un sistema de visión a un proceso industrial, no es nada sencillo, por la diversidad y variabilidad de procesos en la industria que afectan las medidas y por ende los resultados. Al diseñar este tipo de sistemas se deben tener en cuenta estas variaciones, con tal de que se tenga un buen desempeño en la industria [25].

1.2.3 Sistemas implementados en hardware

Cada vez son más los sistemas electrónicos que utilizan cámaras para satisfacer una necesidad, por esto los diseñadores han adicionado capacidades de inteligencia o de análisis a los sistemas de visión con tal de ayudar a identificar personas u objetos. Por ejemplo, en la industria donde se producen partes de maquinaria, se utilizan múltiples sistemas de visión para verificar las piezas a la vez que se están fabricando e identificar aquellas que no cumplen los requerimientos. Actualmente las empresas ya tienen aparatos o dispositivos que cuentan con análisis de vídeo, como es el caso de vehículos tripulados y no tripulados, sistemas de inspección, vídeo vigilancia, en automóviles y hasta en dispositivos personales como los teléfonos inteligentes.

Usualmente, los diseñadores de estos sistemas implementaban los algoritmos de procesamiento de imágenes en DSP, microprocesadores o múltiples procesadores; sin embargo, conforme la calidad de vídeo avanza y se utilizan imágenes con mejor resolución, mayores velocidades y algoritmos más complejos, es más difícil usar las mismas herramientas de implementación. El uso de procesadores digitales de señales o DSP para aplicaciones de vídeo tiene sus ventajas

tales como la programación en lenguaje C, tasas de reloj relativamente altas, bibliotecas optimizadas que permiten el desarrollo y la depuración más rápida. Sin embargo, estos están limitados a la cantidad de instrucciones que pueden ejecutar en paralelo, tienen un número fijo de multiplicadores y acumuladores, y sus puertos de entrada y salida son limitados [26].

Por su parte los FPGAs son circuitos que se componen de un gran número de compuertas lógicas programables, de forma que se pueden implementar circuitos digitales con lenguaje descriptor de hardware y como los elementos lógicos pueden ejecutarse en paralelo ofrecen un procesamiento de alta velocidad. Esta tecnología fue inventada por Xilinx en 1984, y desde entonces ha estado evolucionando hasta convertirse en parte importante de las industrias de medición, aeroespacial, de procesamiento de imágenes, de consumo, de comunicaciones y de procesamiento de señales. Una de las principales razones por la cual los FPGAs se han vuelto tan populares, es la cantidad de instrucciones por segundo que son capaces de ejecutar. Por ejemplo, en un DSP la implementación de un filtro de 256 etapas usando una unidad de multiplicación y acumulación convencional (MAC), tardaría 256 ciclos de reloj; en un FPGA, usando celdas para procesamiento digital de señales embebidas el filtro se ejecutaría en 1 ciclo de reloj, independientemente de la longitud del mismo. Otra ventaja presente en los FPGAs es la capacidad de personalizar la arquitectura en función del comportamiento deseado para el sistema, es decir, optimizar los recursos o aumentar el desempeño [27].

Los algoritmos de visión juegan un rol fundamental en el procesamiento de vídeo, que se requiere en la mayoría de las aplicaciones en tiempo real, como en trabajos donde se utilizan algoritmos *Harris* para detectar las esquinas de los objetos [28]. Asimismo, otras investigaciones en tareas de detección de bordes han demostrado tener un alto desempeño y un bajo consumo de energía al utilizar FPGAs, además de poder utilizarse en dispositivos portables [29]. Otros proyectos han desarrollado un robot móvil con una cámara montada en un brazo manipulador, con el fin de reconocer objetos en el piso de una fábrica y acercarlos a los operadores. El principal trabajo del FPGA, es el análisis de las imágenes y el reconocimiento de objetos, que se llevan a cabo en una red neuronal, la cual a su vez se beneficia del procesamiento en paralelo. Con este proyecto se muestran las ventajas de usar estos dispositivos programables, pues ofrecen sistemas de alto desempeño, con una buena eficiencia energética, pequeños y aptos para aplicaciones móviles [30].

La inspección visual es una aplicación del procesamiento de imágenes muy importante que se usa en todos los procesos de fabricación; por ejemplo en las tarjetas de circuito impreso dónde se captura una imagen de esta para encontrar fallas en la soldadura, se aplican algoritmos de procesamiento de imágenes y los resultados se comparan con los valores preestablecidos de

otra sin defectos, y por tanto se puede saber si la tarjeta tiene o no defectos; dichos algoritmos se implementan en hardware para reducir tiempos de procesamiento y costos, en comparación con sistemas de inspección actuales [31].

Los FPGAs cada vez son más utilizados en instrumentos o dispositivos que necesitan atender múltiples tareas, como en el caso de sistemas capaces de controlar el movimiento y trayectoria de un robot, adquirir y procesar señales e imágenes, mantener constante comunicación con una computadora, y todo esto en tiempo real [32].

1.3 Organización de la tesis

La exposición de este trabajo se reparte en cinco capítulos: En el primero se ha presentado el objetivo de esta tesis, así como algunos de los trabajos realizados sobre visión artificial orientados hacia aplicaciones industriales, y en especial al ensamble de piezas. También se ha presentado una introducción muy general respecto a los temas a tratar. En el capítulo dos, se mostrarán de manera breve y resumida, los fundamentos teóricos que dan sustento a los temas abordados en la tesis. Se explican los tipos de robots y sus principales componentes. En el área de la visión artificial, se abordan temas del procesamiento de imágenes así como los conceptos básicos de ésta. En cuanto al tema de los FPGAs, se presenta una breve introducción a su composición y la descripción de su comportamiento. Para el tercer capítulo se explica a detalle el diseño del sistema desarrollado tomando como base conceptos mencionados en el capítulo dos; asimismo se menciona su funcionamiento y todas las herramientas utilizadas. Las pruebas realizadas para determinar el funcionamiento de todo el sistema se comentan en el capítulo cuatro, además de los resultados obtenidos. El apartado cinco está compuesto por la conclusión del trabajo y las propuestas para mejorar el sistema. Ya por último están los apéndices, donde se incluye información útil para replicar las pruebas, ya que se anexa el código de Matlab para implementar los mismos algoritmos que en el FPGA.

Capítulo 2

Marco teórico

2.1 Celdas de manufactura

El avance en los métodos de manufactura a través de la historia ha jugado un papel vital en el desarrollo de la tecnología actual, la cual permite el crecimiento en la economía de las naciones y por tanto, una mejora en el nivel de vida de su población [33]. Dentro del ambiente de la producción de bienes, muchas de las piezas producidas tienen ciertas semejanzas en su forma y método de manufactura; esto da lugar a una clasificación en grupos o familias de piezas similares entre sí y, por ende, a un proceso productivo más eficiente. Para explotar completamente las similitudes en las partes de una familia, la producción debe organizarse en Celdas de Manufactura que contengan una o varias máquinas especializadas en elaborar las partes. Dichas celdas son comúnmente clasificadas según el número de máquinas y el grado de automatización que tengan; siendo las Celdas de Manufactura Flexible las que presentan el mayor nivel de automatización [34]. Estas celdas se definen como un grupo de estaciones de trabajo, usualmente máquinas de control numérico por computadora (CNC), interconectadas por un sistema automático de manejo, transporte y almacenamiento de materiales y controladas por un equipo de cómputo integrado, ver Figura 2.1 [33]. Las celdas de manufactura flexible pueden estar atendidas por uno o dos operadores en función del número de máquinas en la celda. A su vez estas celdas se pueden atender por un robot para la carga, descarga y transferencia de piezas de trabajo entre las estaciones. Por lo que surge la necesidad de implementar robots manipuladores dotados con modos sensoriales para cubrir estas funciones.

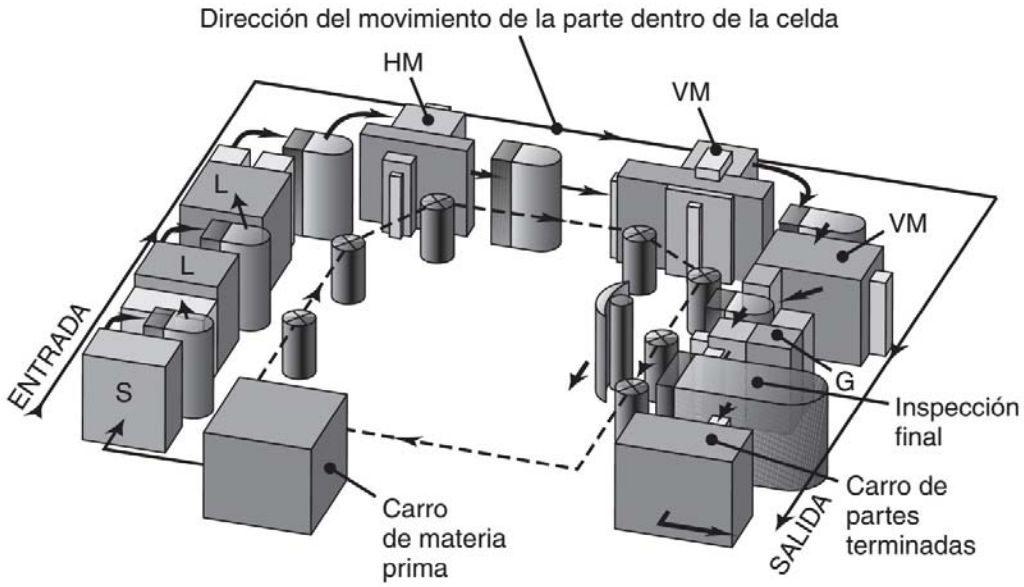


Figura 2.1 Ejemplo de una celda flexible de manufactura que muestra diversas máquinas, herramientas y una estación de inspección. Sierra (S), torno (L), fresadora horizontal(HM), fresadora vertical(VM), rectificadora (G), y posiciones de los trabajadores (X). [34]

2.1.1 Robots manipuladores

Aunque el término robot se utiliza para nombrar a virtualmente cualquier sistema que sea más o menos autónomo, durante este trabajo se utilizará para nombrar solamente a los manipuladores industriales, ver Figura 2.2; que en esencia son un brazo mecánico que opera bajo un control por computadora y están compuestos por los siguientes elementos [35]:

- Efector final. Es la parte que realiza la tarea requerida en función de la aplicación, esta se encuentra unida al último eslabón del brazo y puede ser desde una simple pinza hasta una mano antropomórfica.
- Actuadores. Mueven la estructura del manipulador y entre los más comunes están los servomotores, motores a pasos y los cilindros hidráulicos o neumáticos.
- Sensores. Recolectan información del estado interno del robot y pueden ser de fuerza, distancia, proximidad, posición, visión, etcétera.
- Controlador. Es el encargado de manipular los movimientos de los actuadores y coordinarlos con base en la realimentación de los sensores.

- Procesador. Se encarga de hacer los cálculos de movimientos y velocidades, así como de supervisar las acciones coordinadas del controlador y los sensores.

(a) Kuka robotics modelo KR 5-2 [36]

(b) ABB modelo IRB 4400 [37]

(c) Robot Epson modelo C4 [24]

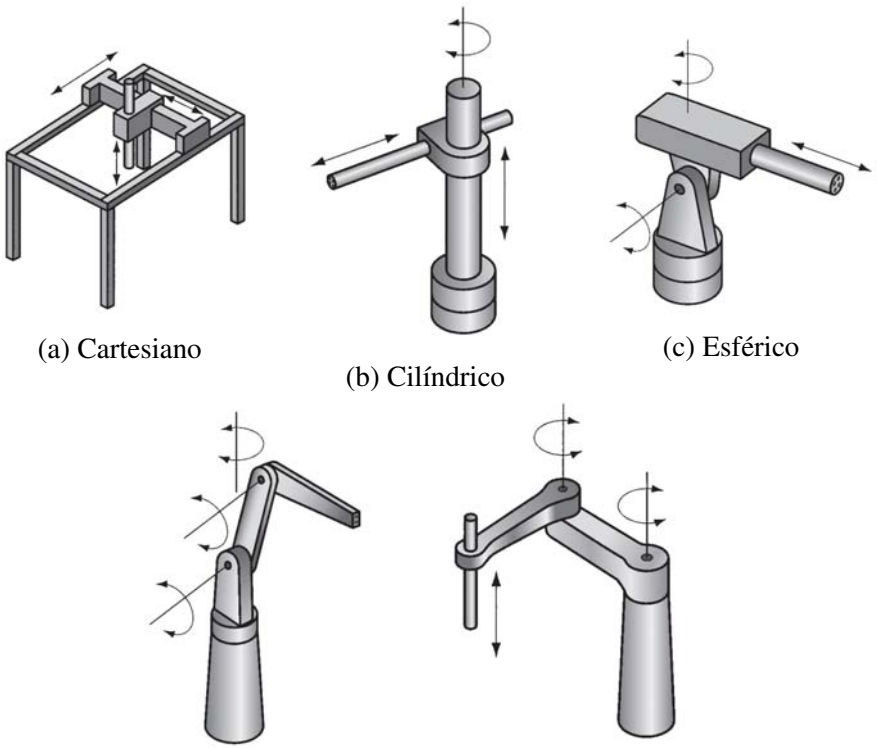
Figura 2.2 Ejemplo de robots manipuladores de 6 GDL.

Grados de libertad. En un manipulador robótico, el número de articulaciones determina el número de Grados de Libertad (GDL). En el espacio tridimensional, un objeto rígido tiene seis grados de libertad: tres para posición y tres para orientación. Por tanto, un manipulador debería de contar con mínimo seis GDL, ya que con menos de seis no podrá alcanzar algún punto dada una orientación cualquiera y con más de seis se tratará de un manipulador con cinemática redundante [35].

Articulaciones. Son las partes de un robot que permiten el movimiento relativo entre las partes solidas del brazo o eslabones. Estas pueden ser de revolución (R) o lineales (L).

Casi todos los manipuladores suelen clasificarse de manera cinemática con base en las tres primeras articulaciones del brazo y en la descripción por separado de la muñeca. La mayoría de éstos caen dentro de alguna de las siguientes configuraciones: Cartesiana (PPP), Cilíndrica (RPP), Esférica (RRP), Articulada (RRR), SCARA (RRP), ver Figura 2.3 [35].

Los robots manipuladores se asemejan al brazo humano en que tiene las mismas coyunturas de revolución, lo que los convierte en la configuración más usada para aplicaciones industriales. Cuando el eje de la articulación del hombro y codo son paralelos entre si y perpendiculares respecto al de la base, se le conoce como manipulador tipo codo. Esta clase de robot tiene tres



(a) Cartesiano (b) Cilíndrico (c) Esférico

(d) Articulado (e) SCARA

Figura 2.3 Configuraciones más frecuentes en robots industriales [35].

grados de libertad por lo que puede posicionarse en algún punto de su espacio de trabajo. Sin embargo, para determinar su orientación, regularmente se utiliza otra configuración conocida como de muñeca esférica.

La muñeca de un manipulador se refiere a las articulaciones en la cadena cinemática entre el brazo y el efector final. Muchos robots se ha diseñado con muñecas esféricas porque sus ejes de articulación se interceptan en un punto común, ver Figura 2.4. La importancia de esta configuración radica en que simplifica el análisis cinemático, permitiendo desacoplar el posicionamiento y la orientación del efector final. Típicamente, un manipulador cuenta con tres GDL para la posición, producidos por las tres o más coyunturas del brazo; por su parte el número de GDL para la orientación depende de las articulaciones de la muñeca [39].

Para caracterizar las especificaciones del robot, se utilizan los siguientes conceptos:

- Carga útil. Es el peso que puede cargar sin comprometer el correcto funcionamiento del robot.

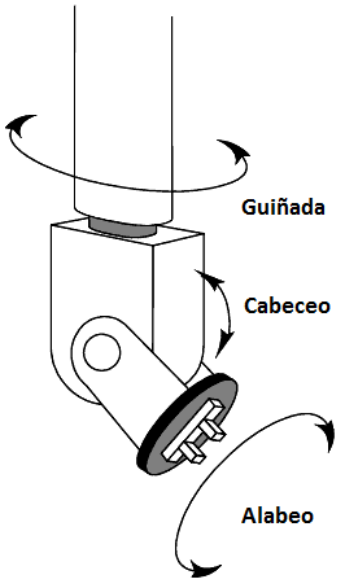


Figura 2.4 Configuración de la muñeca esférica [38].

- Alcance. Es la distancia máxima que puede obtener el efector final dentro de su espacio de trabajo, y este último se define como el volumen total que se abarca cuando el robot ejecuta todos los movimientos posibles.
- Precisión. Es la vecindad de las posiciones medidas, al tratar de llegar a un mismo punto en varios intentos. Este parámetro depende de la resolución del actuador y de los sensores.

En la norma *ISO 9283:1998 Manipulating Robots – performance Criteria and Related methods*, se definen las principales características de funcionamiento de los robots industriales, las formas de especificarlos, las recomendaciones sobre cómo realizar las pruebas para obtener dichas características e incluso la forma de presentar los informes de resultados [40].

2.2 Visión artificial

La visión artificial es el proceso por medio del cual una máquina automáticamente procesa una imagen y la información contenida en esta se utiliza como entrada para otro sistema [41]. Estos procesos se pueden realizar en cualquier dispositivo electrónico, no solo se limita al equipo de cómputo, sino que también puede llevarse a cabo en un DSP, FPGA, en arreglos de procesadores o en sistemas híbridos. La visión artificial está orientada a aplicaciones industriales, pero no

exclusivamente a estas ya que puede emplearse en la solución de problemas prácticos, como el mejorar la seguridad, reducir costos o para cuestiones sociales como cuidar el medio ambiente.

La visión artificial se puede diferenciar de la visión por computadora en la forma de entrada de la información al sistema; en el caso de la visión por computadora se parte de una imagen digital y el papel que desempeña el procesamiento de imágenes es muy importante, en cambio, para la visión artificial los datos de entrada son los mismos objetos a analizar además del entorno donde se encuentran y no solo se enfoca en el procesamiento de imágenes, sino en múltiples áreas como la iluminación, la óptica, la adquisición de señales, la arquitectura de sistemas digitales, los sensores, entre otras. Así pues que la visión artificial está más orientada a problemas prácticos que a cuestiones de investigación [42].

Para diseñar un sistema de visión artificial que sea de utilidad práctica en la industria, se requiere de un enfoque multidisciplinario que abarque conceptos, algoritmos, técnicas y tecnologías, tales como: física, óptica, mecánica, electrónica, sensores, actuadores, interfaces, procesamiento de señales e imágenes, iluminación, lenguajes de programación, inteligencia artificial, ingeniería industrial, entre otros.

2.2.1 Procesamiento de imágenes

El procesamiento de imágenes se puede definir como una serie de operaciones matemáticas aplicadas a una imagen con el fin de obtener otra diferente y que está en función de las operaciones aplicadas. Estás operaciones se pueden agrupar de acuerdo con el tipo de dato que procesan y a veces se ilustran como una pirámide mostrada en la Figura 2.5 [43] [44]. En el nivel más bajo de la pirámide se encuentran las operaciones de pre procesamiento, que son transformaciones de imagen a imagen, con el fin de resaltar la información sobresaliente en ésta; ejemplos de estas son: corrección de la distorsión, filtrado, mejora del contraste, etc. Operaciones de segmentación, como detección de color, de umbral, y etiquetado de componentes ocurren entre los niveles bajo y medio. El propósito de la segmentación es detectar objetos o regiones en la imagen que tengan propiedades comunes entre sí; por lo que la segmentación es una transformación de imágenes a regiones, después de operación viene la clasificación. Las características de cada región son utilizadas para identificar objetos o partes de estos, o para clasificar un objeto en una de varias categorías predefinidas; así pues la clasificación transforma información de las regiones en características y después en etiquetas. Para entonces, la información ya no está basada en imágenes, sino en características y etiquetas

que pueden contener información respecto a la posición de un objeto. En el nivel más alto de la pirámide, está la operación de reconocimiento que deriva en una descripción o interpretación de la escena [45].

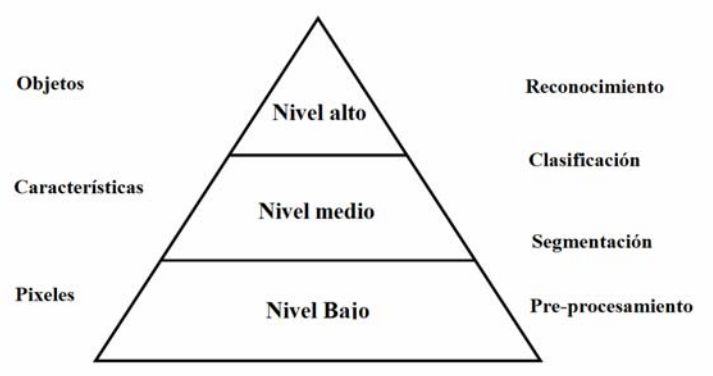


Figura 2.5 Pirámide del procesamiento de imágenes [43].

Por otro lado, en vez de centrarse en las operaciones podemos tomar en cuenta los datos de una imagen los cuales consisten en un arreglo de píxeles, donde cada píxel por si solo tiene muy poca información pero en una imagen hay muchos y conforme ésta es procesada, los píxeles se van agrupando. Es decir, en el nivel más bajo de la pirámide el valor del pixel es alto pero nulo en cuestión de información. Luego, en el nivel intermedio de la pirámide, los datos aún pueden representarse en términos de píxeles, pero los valores de estos pueden tener mejor significado, es decir, tal vez representen una etiqueta asociada con una región. De cada región un conjunto de atributos pueden sobresalir que caractericen a esta; generalmente hay pocos atributos en una región, y cada uno de ellos contiene información significante que puede utilizarse para distinguirla de otra región o de otro objeto en la imagen. Para el nivel intermedio, los valores de los píxeles son menores pero con una mayor cantidad de información. Finalmente en el nivel más alto, los atributos pueden ser usados para clasificar el objeto en varias categorías, o para tener una descripción del objeto.

El procesamiento de imágenes se puede clasificar en función del efecto deseado en la imagen de salida.

- Mejora de la imagen. Implica la capacidad de mejorar subjetivamente la calidad de la imagen o la capacidad para detectar objetos en la imagen de tal forma que sean más fáciles de reconocer. Algunas operaciones dentro de esta clasificación son: reducción del ruido, mejora del contraste, corrección del color y mayor nitidez en los bordes.

- Restauración de imágenes. Utilizando las razones que causaron la degradación de la imagen crea un modelo del proceso de degradación que se usa para derivar un proceso inverso y restaurar la imagen.
- Reconstrucción de imágenes. Consiste en reestructurar la información a una forma más útil; ejemplos de esto son la imagen de súper resolución, construida a base de imágenes de baja resolución y la tomografía.
- Análisis de imágenes. Se refiere al uso de computadoras para extraer información de las imágenes, el resultado se compone de alguna forma de medición.
- Reconocimiento de patrones. Consiste en la identificación de objetos o patrones en las imágenes [45].

2.2.2 Representación de imágenes

Una imagen puede ser representada por una función continua de dos variables $f(s,t)$; para convertir esta imagen, en una digital, es necesario muestrear y cuantificar la función. La *imagen digital* puede representarse entonces como un arreglo en dos dimensiones, $f(x,y)$, que contiene M filas y N columnas, donde (x,y) son coordenadas discretas $x = 0,1,2\ldots,M-1$ y $y = 0,1,2\ldots,N-1$. En general, el valor o amplitud de una imagen en cualquier coordenada (x,y) se denota por $f(x,y)$ donde x y y son enteros. Cuando una imagen es generada, los valores de amplitud son proporcionales a la energía radiada por un objeto; como resultado $f(x,y)$ debe ser finito y diferente de cero. Además $f(x,y)$ se puede describir por dos componentes, la iluminación y la reflectancia; ambos expresados por $i(x,y)$ y $r(x,y)$ respectivamente. Combinando ambas componentes con $f(x,y)$ tenemos:

$$f(x,y) = i(x,y)r(x,y) \tag{2.1}$$

Donde

$$0 < i(x,y) < \infty \tag{2.2}$$

y

$$0 < r(x,y) < 1 \tag{2.3}$$

El valor de $i(x,y)$ es determinado por la fuente de iluminación y $r(x,y)$ es determinado por las propiedades de reflectancia del objeto. Definiendo la *intensidad* de una imagen monocromática

en cualquier coordenada (x_0, y_0) como:

$$\ell = f(x_0, y_0) \tag{2.4}$$

y tomando en cuenta 2.1, 2.2 y 2.3 se puede deducir que ℓ esta en el rango:

$$L_{min} \leq \ell \leq L_{max} \tag{2.5}$$

Donde, L_{min} debe ser positivo y L_{max} un número finito. El intervalo $[L_{min}, L_{max}]$ se conoce como *escala de intensidad*, donde $\ell = L_{min}$ se considera como color negro y $\ell = L_{max}$ como color blanco. Al considerar el almacenamiento en hardware, el número de niveles de intensidad está limitado a números enteros potencias de 2, ecuación (2.6).El número de bits necesarios para almacenar una imagen digital está dado por (2.7).

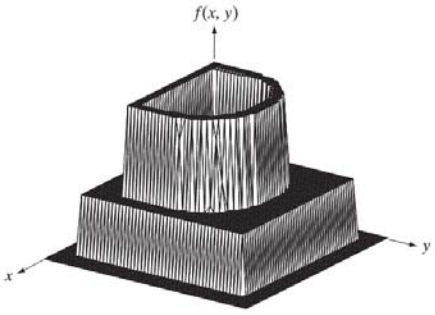
$$L = 2^k \tag{2.6}$$

$$b = M \times N \times k \tag{2.7}$$

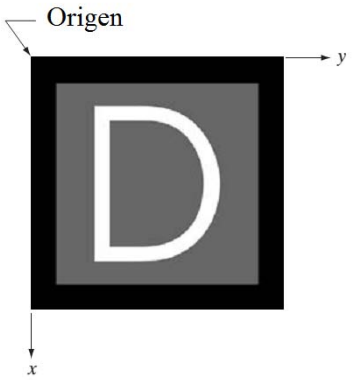
Existen tres formas básicas de representar $f(x,y)$; en la Figura 2.6a se muestra la gráfica de la función, donde (x,y) es la coordenada y z es el valor de f en esa coordenada. La representación en la Figura 2.6b muestra $f(x,y)$ como se vería en un monitor; en esta, la magnitud de cada punto es proporcional al valor de f en dicho punto. Si la magnitud se normalizará a un intervalo $[0,1]$, entonces cada punto en la imagen tendría el valor de 0, 0.5 o 1, lo cual en un monitor se apreciaría como negro, gris ó blanco. La tercera representación, 2.6c, despliega $f(x,y)$ como un arreglo numérico, donde cada elemento representa el valor de f en dicha coordenada; esta representación es muy útil al emplear algoritmos de procesamiento de imágenes, ya que permite trabajar con ciertas partes de la imagen y analizar ésta como valores numéricos. Por consiguiente, un arreglo numérico de $M \times N$ lo podemos representar con la ecuación (2.8).

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,N-1) \\ f(1,0) & f(1,1) & \cdots & f(1,N-1) \\ \vdots & \vdots & \ddots & \vdots \\ f(M-1,0) & f(M-1,1) & \cdots & f(M-1,N-1) \end{bmatrix} \tag{2.8}$$

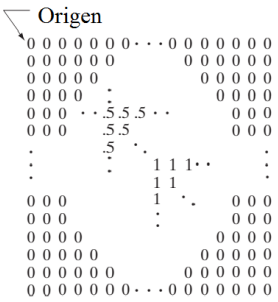
Ambos lados de la ecuación son formas equivalentes de expresar una imagen digital cuantitativamente. Cada elemento de esta matriz se conoce como *pixel*. Algunas veces es más conveniente



(a) Gráfico de una imagen como superficie.



(b) Representación gráfica de una imagen.



(c) Imagen como un arreglo numérico.

Figura 2.6 Diferentes representaciones de una imagen [46].

usar la notación tradicional de matriz para denotar una imagen digital y sus elementos, como se observa en la ecuación (2.9) donde $a_{ij} = f(x = i, y = j) = f(i, j)$, por tanto (2.8) y (2.9) son matrices idénticas.

$$\mathbf{A} = \begin{bmatrix} a_{0,0} & a_{0,1} & \cdots & a_{0,N-1} \\ a_{1,0} & a_{1,1} & \cdots & a_{1,N-1} \\ \vdots & \vdots & \ddots & \vdots \\ a_{M-1,0} & a_{M-1,1} & \cdots & a_{M-1,N-1} \end{bmatrix} \tag{2.9}$$

El origen de una imagen digital se encuentra en la esquina superior izquierda, con el eje positivo x en dirección hacia abajo y el eje positivo y extendido hacia la derecha. Esta es una convención basada en los televisores que barren la imagen empezando en la esquina superior izquierda y desplazándose hacia la derecha y al final de la fila, pasan a la siguiente. [46].

Operaciones con píxeles

Se pueden realizar operaciones aritméticas básicas en los píxeles de las imágenes para obtener efectos que faciliten algoritmos posteriores. Estas operaciones consisten en aplicar cualquier constante a cada pixel de la imagen. También se pueden llevar a cabo operaciones entre imágenes.

$$\begin{aligned} s(x,y) &= f(x,y) + g(x,y) \\ d(x,y) &= f(x,y) - g(x,y) \\ p(x,y) &= f(x,y) \times g(x,y) \\ v(x,y) &= f(x,y) \div g(x,y) \end{aligned} \tag{2.10}$$

Igualmente, las operaciones son entre píxeles correspondientes. s, d, p y v son imágenes de tamaño $M \times N$. También se pueden aplicar operaciones lógicas como AND, NOT, OR y XOR, aunque solo se pueden aplicar a imágenes binarias cuyos píxeles son 0 o 1.

Vecindad de un pixel

Una herramienta muy útil a la hora de trabajar con imágenes en forma de arreglos, es lo que ese conoce como vecindad. Un pixel p con las coordenadas (x,y) tiene 2 vecinos horizontales y verticales, cuyas coordenadas están dadas por:

$$(x+1,y),(x-1,y),(x,y+1),(x,y-1) \tag{2.11}$$

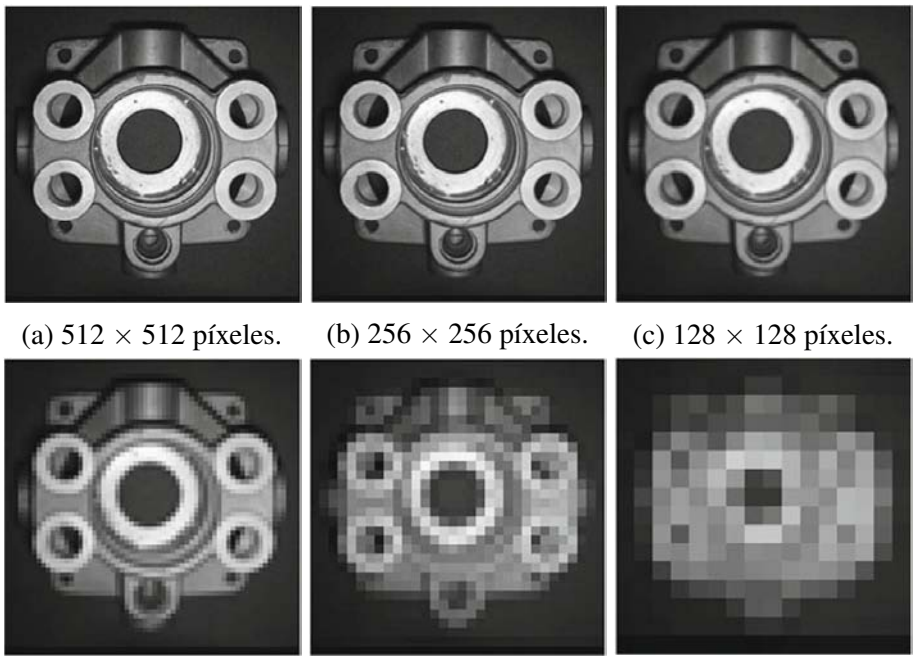
Este conjunto de píxeles, llamado *4-vecindad* de p, se denota por $N_4(p)$. Cada pixel está a una unidad de distancia de (x,y), y algunas posiciones de los vecinos de p caen afuera de la imagen digital si (x,y) está en el borde de la imagen. En estas situaciones se rellenan con ceros los valores de los píxeles que estén fuera de la imagen. Ese mismo pixel p tiene cuatro vecinos diagonales en las coordenadas:

$$(x+1,y+1),(x+1,y-1),(x-1,y+1),(x-1,y-1) \tag{2.12}$$

este conjunto de píxeles se denotan como $N_D(p)$. Estos píxeles junto con los *4-vecindad*, se llaman *8-vecindad* de p y se denotan como $N_8(p)$. De igual manera, si algunos de los puntos $N_D(p)$ y $N_8(p)$ caen fuera de la imagen digital, se rellenan con ceros los valores de los píxeles fuera de la imagen [46].

Tipos de imágenes

Una imagen puede contener una o más bandas que definen la intensidad o color en un determinado pixel, en el caso más simple, cada pixel contiene un solo valor representando el nivel de señal en ese punto de la imagen. La conversión desde este conjunto de valores a una imagen tal como la vemos, se logra por medio de un *mapa de color*; el cual asigna una tonalidad especifica de color a cada valor en la imagen para así dar una representación visual de los valores, ejemplos de estos son la escala de grises y el RGB [47]. El tamaño de una imagen está



(a) 512 × 512 píxeles. (b) 256 × 256 píxeles. (c) 128 × 128 píxeles.

(d) 64 × 64 píxeles. (e) 32 × 32 píxeles. (f) 16 × 16 píxeles.

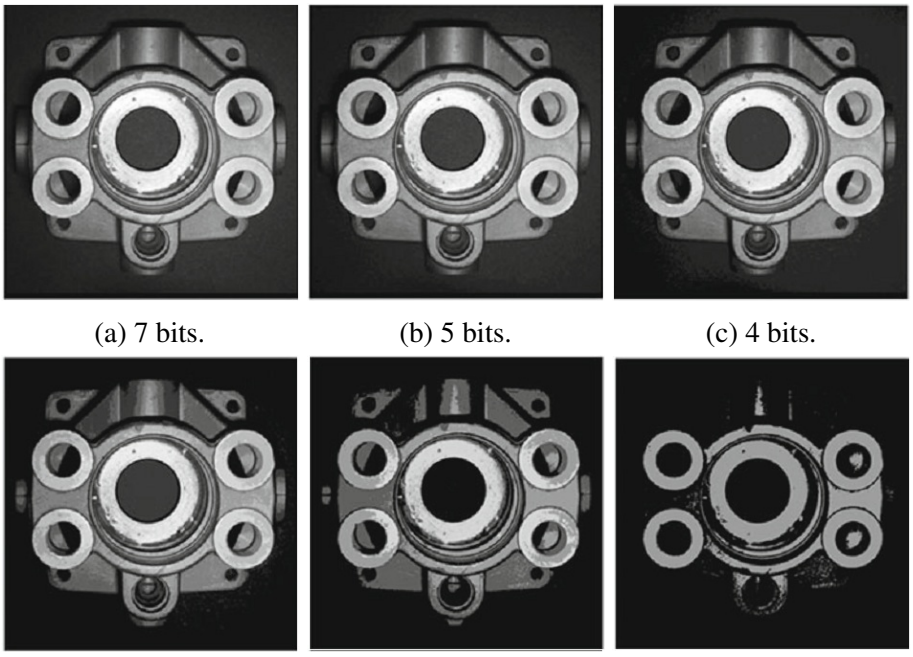
Figura 2.7 Efectos de cambiar la resolución espacial de una imagen. Cada imagen tiene 256 niveles de grises. [42]

definido por su resolución y esta puede especificarse en términos de tres cantidades.

- Resolución espacial. Las dimensiones de la imagen en términos de $M \times N$ definen el número de píxeles usados para cubrir el área visual capturada por la imagen, ver Figura 2.7. Esto se relaciona con el muestreo de la imagen y se refiere a la resolución digital de la imagen.

- Resolución temporal. Para un sistema de captura continuo como el vídeo, esto se refiere al número de imágenes capturadas en un determinado periodo de tiempo. Comúnmente se refiere a *FPS*, donde cada imagen individual se conoce como fotograma.

- Resolución de bit. Esto define el número de valores de color o intensidades que un pixel puede tener y se relaciona con la cuantificación de la imagen, ver Figura 2.8. La resolución de bit hace referencia, al número de bits requeridos para almacenar un pixel en función del nivel L; por ejemplo una imagen binaria solo tiene dos colores, negro y blanco; una imagen en escala de grises normalmente necesita de 8 bits y una imagen a color requiere de 24 bits, 8 por cada banda.



(a) 7 bits. (b) 5 bits. (c) 4 bits.

(d) 3 bits. (e) 2 bits. (f) 1 bit.

Figura 2.8 Efectos de cambiar la resolución de bit de una imagen. Cada imagen tiene un tamaño de 512×512. [42]

A partir de la resolución de bits se pueden deducir diferentes tipos de imágenes:

- Imágenes Binarias. Son arreglos que asignan un solo valor numérico del conjunto $0, 1$ a cada pixel en la imagen. También se les conoce como imágenes lógicas, donde el cero corresponde con el color negro o el fondo de la imagen, y el color blanco con el valor uno o el primer plano de la imagen.

- Imágenes en escala de grises. Los únicos colores en estas imágenes son tonos de gris, desde el negro hasta el blanco; en estas imágenes solo es necesario especificar un solo valor de intensidad para cada pixel, a diferencia de las imágenes RGB que requieren de tres valores de intensidad. Normalmente la intensidad de estas imágenes se almacena

como un entero de 8 bits dando lugar a 256 posibles tonos de gris, aunque el rango de la intensidad puede variar en función de la resolución de bits en la imagen.

- Imágenes RGB. Son ternas ordenadas que asignan tres valores a un pixel y cada valor corresponde a la componente de las bandas de color rojo, verde o azul (RGB). También podemos considerar estas imágenes como tres distintos planos de dos dimensiones cada uno, es decir $M \times N \times C$, donde C corresponde a la banda de color [47].

2.2.3 Segmentación de imágenes

El análisis de imágenes comprende todos los métodos y técnicas que se utiliza para extraer información de una imagen; para ello el primer paso lo constituye la segmentación de imágenes, que se ocupa de descomponer una imagen en sus partes constituyentes, basándose en ciertas características que nos permiten distinguir entre las regiones u objetos de interés; así pues, la segmentación permite simplificar o cambiar la representación de una imagen en otra más significativa o fácil de analizar. La mayoría de las imágenes están constituidas por regiones o zonas que tienen atributos homogéneos como: niveles de gris, textura, momentos, dirección de los bordes, etc; los cuales se aprovechan en la segmentación de imágenes. Estos algoritmos se basan en alguna de las propiedades siguientes:

- *Discontinuidad* en los tonos de gris de los píxeles de un entorno, que permite detectar puntos aislados, líneas y aristas (bordes).
- *Similitud* en los tonos de gris en los píxeles de un entorno, que permita construir regiones por división y fusión, por crecimiento o definición de umbrales [46].

Histograma

El histograma de una imagen digital, con niveles de intensidad en el rango$[0, L-1]$, es una función discreta $h(r_k) = n_k$, donde r_k es k-esimo valor y n_k es el número de píxeles en la imagen con intensidad r_k. Es común normalizar el valor del histograma, dividiendo cada uno de sus componentes por el número total de píxeles en la imagen, denotados como el producto de MN; por tanto un histograma normalizado está dado por $p(r_k) = n_k/MN$, para $k = 0, 1, 2, \ldots, L-1$. Una inspección al histograma puede revelar información de la imagen, como la distribución del contraste y el color, lo cual ayuda a diferenciar el fondo y el primer plano de la imagen.

Otra forma de ver el histograma de una imagen, es como una representación gráfica del número de veces que aparece cada uno de los valores permitidos del pixel en toda la imagen. Para un imagen en escala de grises, el histograma puede construirse con el simple hecho de contar el número de veces que un valor $[0, L-1]$ aparece en la imagen. Véase la Figura 2.9 donde se muestran cuatro niveles de grises característicos: oscuro, claro, bajo contraste y alto contraste; al lado de la imagen se muestra el histograma correspondiente de cada una de las imágenes. El eje horizontal de cada histograma corresponde al nivel de gris, r_k. El eje vertical corresponde al valor de $h(r_k) = n_k$ o a $p(r_k) = n_k/MN$ si los valores están normalizados. Se aprecia que en la imagen oscura los componentes del histograma están concentrados en el lado oscuro de la escala de grises o en el lado izquierdo del histograma. De la misma forma, en la imagen clara los componentes del histograma están concentrados pero en el lado blanco de la escala de grises o en el lado derecho. Una imagen con bajo contraste tiene un histograma muy angosto y centrado en el medio de la escala de grises; en una imagen monocromática, como esta, se implica un aspecto gris oscuro y sin color. Por último se aprecia que los componentes del histograma de la imagen con alto contraste cubren un amplio rango en la escala de grises y además la distribución de los píxeles es casi uniforme. Por tanto, se puede decir que una imagen cuyos píxeles tienden a ocupar el rango completo de la escala de grises y tienen una distribución uniforme, representan una imagen con gran cantidad de detalles a nivel de grises y tiene un alto rango dinámico.

Imágenes binarias

Una imagen binaria consiste en un proceso de reducción de la información de la misma, en la cual como ya se mencionó, sólo persisten dos valores: 0 y 1, es decir negro y blanco. En el proceso y análisis de imágenes, la binarización se emplea para separar las regiones u objetos de interés del resto de la imagen. Las imágenes binarias se usan en operaciones booleanas o lógicas para identificar individualmente objetos de interés o para crear máscaras sobre regiones.

Las imágenes binarias se pueden crear a partir de una imagen en escala de grises usando su histograma y eligiendo un valor de umbral que sirva de referencia para la nueva imagen, a este proceso se le conoce como binarización. Observando el histograma de la Figura 2.10, podemos distinguir el objeto y el fondo de la imagen fácilmente; sin embargo, para que el sistema pueda diferenciarlos, necesita un umbral que le diga dónde termina el objeto y empieza el fondo. Este umbral, se puede seleccionar en el menor de los valles del histograma, siempre y cuando quede entre el objeto y el fondo; como el de la Figura 2.10.

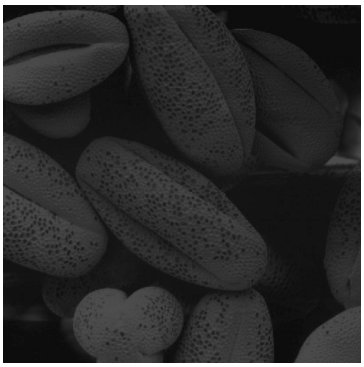
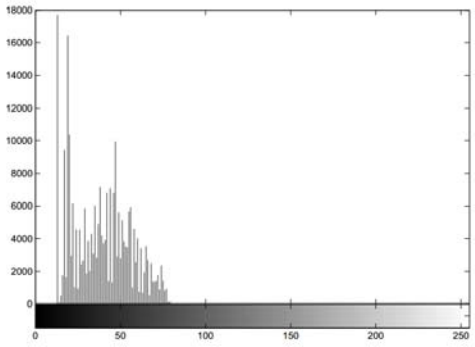



Imagen oscura.

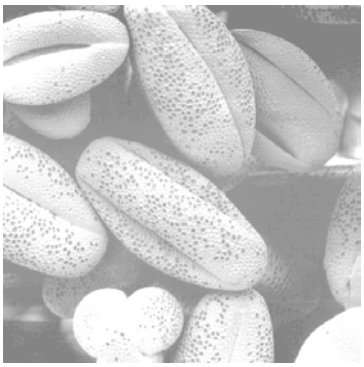
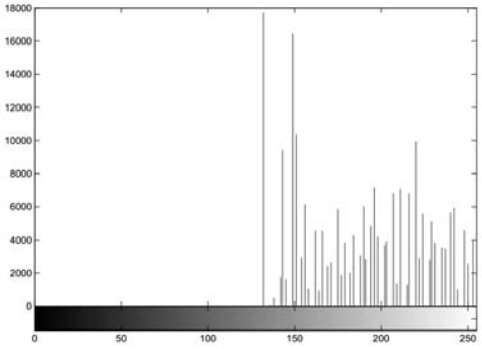



Imagen brillante.

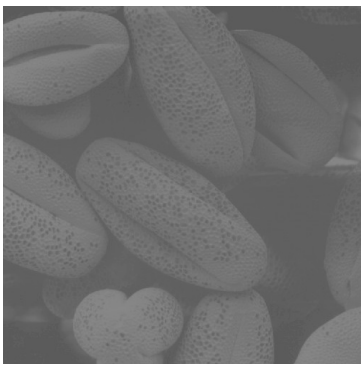
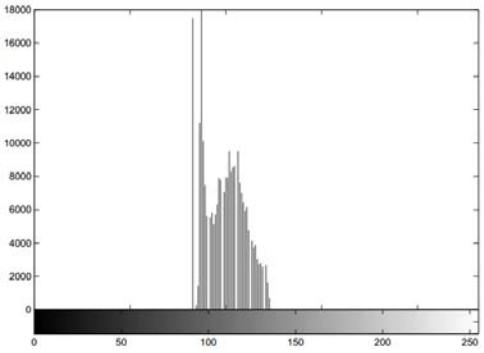



Imagen de bajo contraste.

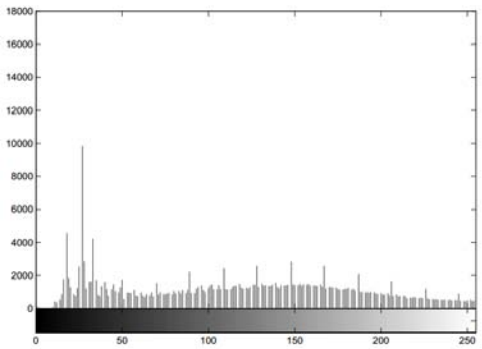



Imagen de alto contraste.

Figura 2.9 Cuatro tipos básicos de imágenes con sus respectivos histogramas [46].

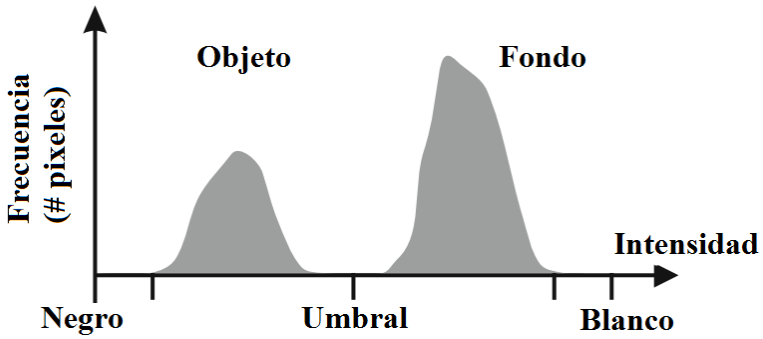


Figura 2.10 El umbral del histograma sirve como referencia para convertir una imagen en escala de grises a imagen binaria.

Una vez conocido el valor del umbral, se compara con cada uno de los píxeles de la imagen para obtener la nueva imagen binaria. Es decir:

$$\begin{aligned} f(x,y) > u &\Rightarrow f(x,y) = 0 \\ f(x,y) \leq u &\Rightarrow f(x,y) = 1 \end{aligned} \tag{2.13}$$

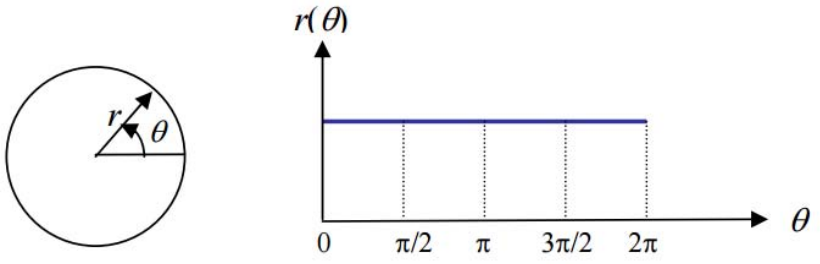
Donde u es la constante de umbral. Después de comparar toda la imagen, al final se obtendrá la imagen binaria, donde los valores $f(x,y) = 1$ representarán el objeto y los valores $f(x,y) = 0$ el fondo de la imagen.

Descriptores de contornos y regiones

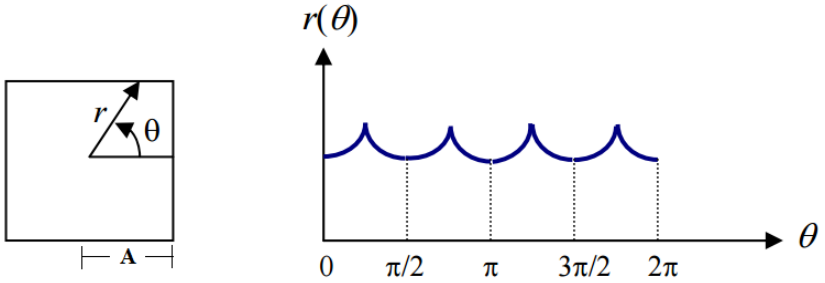
Básicamente para representar una región se puede hacer de dos maneras, por medio de las características externas, es decir el contorno; o por medio de sus características internas, los píxeles que comprenden la región. Seleccionar el esquema de representación es la primera tarea, ya que además se debe describir la región con base en el esquema de representación; por ejemplo, una región puede representarse por su contorno y este a su vez debe ser descrito por parámetros tales como la longitud, orientación de la línea que une sus extremos o el número de concavidades. Algunas veces es necesario usar ambos esquemas, en cuyo caso los parámetros que describen la región deben ser en lo posible invariantes al tamaño, traslación y rotación.

Firma. Esta es una representación de un contorno mediante una función unidimensional con el fin de que sea más simple describir que la forma bidimensional original. Una de las formas más simples de definirla, es a través de la distancia desde un punto interior, como puede

ser el centroide, a cada uno de los puntos del contorno como una función del ángulo, como se muestra en la figura 2.11. La firma es invariante frente a traslaciones pero no lo es ante rotaciones o cambios de escala. Sin embargo, se puede ser invariante frente a rotaciones cuando se encuentra un punto característico del contorno que sea único a partir del cual se comience a generar ésta, por ejemplo, el punto más cercano al centroide. Por otra parte, como los cambios en el tamaño del contorno tienen variaciones en la amplitud de la firma, se puede normalizar de forma que sus valores estén dentro del intervalo $[0,1]$, al dividirla por su amplitud máxima y convertirla en adimensional [46] [48].



(a) $r(\theta)$ es constante



(b) $r(\theta) = Asec\theta$ para $0 \leqq \theta \leqq \pi/4$ y $r(\theta) = Acsc\theta$ para $\pi/4 < \theta \leqq \pi/2$

Figura 2.11 Figuras con sus correspondientes firmas en función del angulo $r(\theta)$ [46].

Función frontera del objeto (BOF). Esta función pertenece a la categoría de firmas pero a diferencia de las otras esta es invariante al tamaño, traslación y rotación. La BOF se genera al graficar la distancia que hay desde el centroide hasta algún punto del contorno; esta firma no está en función de ningún ángulo por ello es invariante a la rotación. Entonces la función frontera se puede obtener aplicando la fórmula de la distancia euclidiana entre dos puntos, los cuales son el centroide y un punto cualquiera perteneciente al contorno, ver ecuación 2.14; como se puede ver la función frontera no es necesaria aplicarla a todo el contorno, pero entre

más puntos se tomen, mayor será la definición de la gráfica.

$$BOF(k) = \sqrt{(P_x - C_x)^2 + (P_y - C_y)^2} \tag{2.14}$$

En donde: (C_x, C_y) son las coordenadas del centroide, (P_x, P_y) son las coordenadas de un elemento del contorno del objeto y k es el k-ésimo elemento del contorno al que se aplicará la BOF, con $k = 1, 2, ...k$.

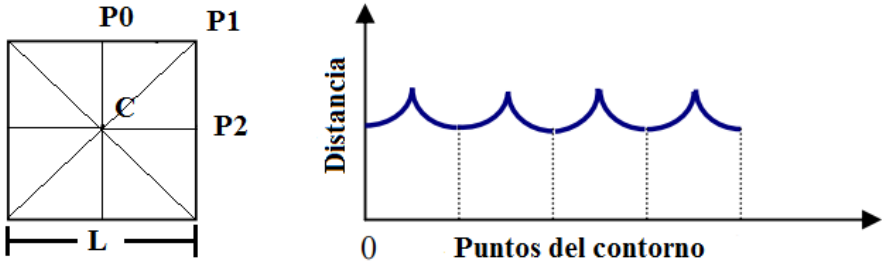


Figura 2.12 Función BOF para un cuadrado.

Área. El área de una imagen se entiende como el número de píxeles que se encuentren en una determinada región de la imagen. En una imagen de M filas y N columnas, el área total de ésta, será el producto de $M * N$. Así pues si solo se desea obtener el área de una determinada región dentro de la imagen o de un objeto en específico, solo hay que contar el número de píxeles que abarca esta región u objeto.

Centroide. El centro geométrico o centroide de una región, es determinado por el conjunto de píxeles pertenecientes a ésta y definidos como $(x_i, y_i), i = 1, 2 \ldots, A$; y se encuentra en el punto (C_x, C_y) definido como:

$$C_x = \frac{\sum_{x,y} x_i}{A}, C_y = \frac{\sum_{x,y} y_i}{A} \tag{2.15}$$

Compacidad. La compacidad de una región es un parámetro que no depende del tamaño de la región y viene dado por:

$$c = \frac{p^2}{A} \tag{2.16}$$

Donde A es el área y p el perímetro de la región. Su valor máximo corresponde a las figuras en forma de círculos y vale (4π) por ello es una medida de circularidad y así los valores diferentes y alejados del parámetro indican objetos alargados, además es invariante ante traslaciones, rotaciones y cambios de escala [48].

2.3 FPGA.

Un FPGA es un dispositivo lógico programable, que internamente está conformado por tres elementos: el bloque lógico configurable o configurable logic block (CLB), las interconexiones y los bloques de entrada/salida (E/S), ver Figura 2.13. Los Bloques de E/S se sitúan en el perímetro de la estructura y proporcionan un acceso de entrada, salida o ambos, para las terminales del dispositivo. La matriz de interconexiones programables permite unir los CLB entre sí o hacía los bloques de E/S.

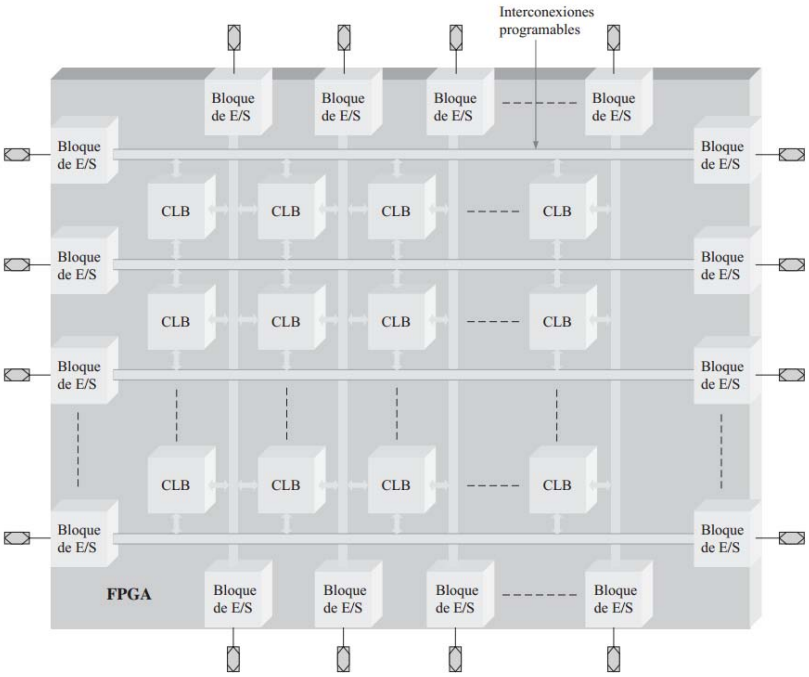


Figura 2.13 Estructura básica y genérica de un FPGA. CLB es un bloque lógico configurable [49].

Los CLB están compuestos por una serie de interconexiones locales y por varios módulos lógicos más pequeños, ver Figura 2.14. Estos módulos, basados en tablas de búsqueda o "Look-up Table (LUT)", pueden configurarse para implementar lógica combinacional, secuencial o

una mezcla de ambas. Una LUT es un tipo de memoria programable que se utiliza para generar funciones combinacionales de sus variables de entrada, ver Figura 2.15.

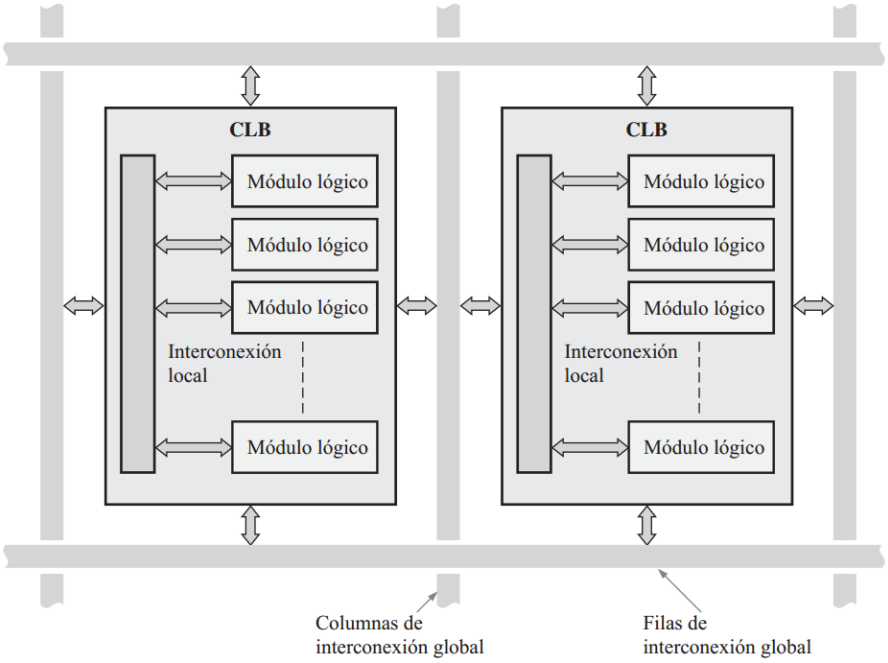


Figura 2.14 Bloques lógicos configurables dentro de la estructura global del interconexiones programables [49].

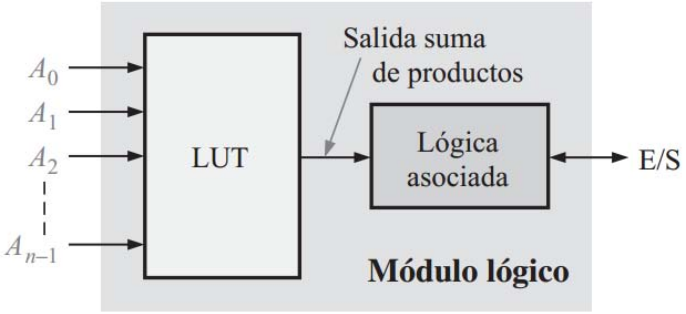


Figura 2.15 Diagrama de bloques de un módulo lógico de un FPGA [49].

Algunos FPGAs, dependiendo de la tecnología de fabricación, pueden ser volátiles y perder todos los datos programados en los CLBs al desconectar el voltaje de alimentación. Por tanto, se incluye una memoria de configuración no volátil dentro del integrado para almacenar los datos de programa y reconfigurar el dispositivo cada vez que se reinicie o se aplique la alimentación.

Los principales fabricantes de FPGAs son Altera y Xilinx, cada uno ofrece modelos con ligeras diferencias en las arquitecturas a las descritas hasta ahora. También, cada uno nombra de

diferente forma a los elementos que componen a los FPGAs, en el caso de Xilinx, los módulos lógicos se conocen como celdas lógicas o logic cell (LC) y al conjunto de dos celdas lógicas idénticas se le denomina rodaja (slice).

Actualmente, los fabricantes de FPGAs incluyen módulos de hardware dedicado para proporcionar funciones específicas y que no pueden reprogramarse. Entre las ventajas de éstos módulos están, el menor consumo de espacio dentro del dispositivo y el menor tiempo de programación por parte del usuario. La desventaja de estos módulos es que las especificaciones se establecen durante la fabricación y posteriormente no pueden cambiarse. Ejemplos de éstos son los microprocesadores, interfaces de E/S, procesadores digitales de señales o digital signal processing (DSP), multiplicadores, etc [49].

Un recurso muy utilizado en este trabajo son las memorias de acceso aleatorio (RAM), estas se clasifican en dos tipos, las memorias distribuidas y las embebidas o Bloques RAM (BRAM). Las primeras deben su nombre a que están dentro de los CLBs, los cuales están repartidos en todo el FPGA. Cada LUT de un CLB puede ser implementado como una RAM de 16 *1, pero también, se pueden conectar en cascada para implementar memorias de mayor capacidad y en configuraciones de 1 o 2 puertos; además las operaciones de escritura son síncronas, mientras que las de lectura son asíncronas. La RAM distribuida, se puede utilizar en aplicaciones que requieren de poco espacio de memoria como FIFOs o registros [50].

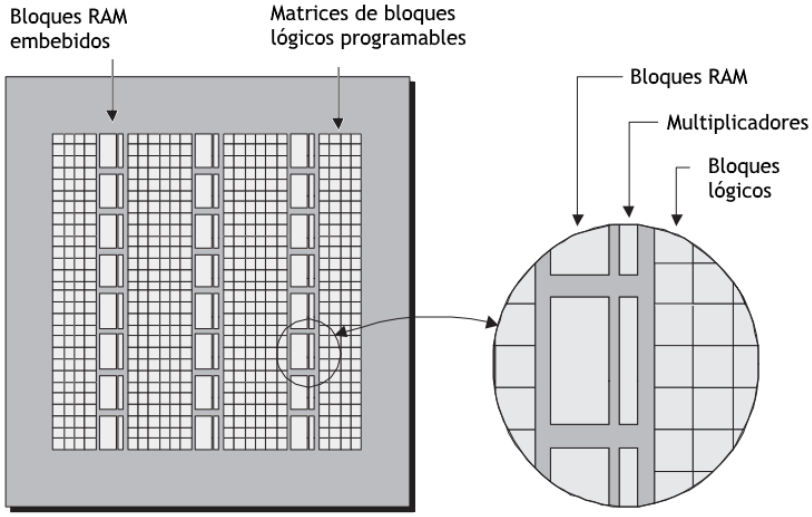


Figura 2.16 Bloques de memoria RAM embebidos en el FPGA y posicionados en columnas [51].

Los BRAM son módulos de hardware dedicado que se utilizan cuando se requieren espacios de almacenamiento en el orden de Kbits. Éstos se utilizan para implementar RAMs, ROMs,

Figura 2.17 A la izquierda, BRAM simple de dos puertos; a la derecha ROM simple de un puerto.

LUTs y registros de corrimiento. Dependiendo de la arquitectura del FPGA, estos bloques pueden ser posicionados en la periferia, en columnas o repartidos en el dispositivo, ver Figura 2.16. Uno solo de éstos, puede almacenar hasta Mbits de información y en un FPGA puede haber hasta cientos de ellos, los cuales se pueden usar de manera independiente o en conjunto para implementar memorias de mayor capacidad y con diferentes formatos. A diferencia de las memorias distribuidas, las operaciones de escritura y lectura en las BRMA son síncronas [52].

En el desarrollo de este trabajo se utilizaran memorias BRAM, por la cantidad de almacenamiento requerido para guardar un fotograma proveniente de la cámara. Éstas se implementan por medio de los IP core de Xilinx y se utilizan en su mayoría memorias RAM simples de dos puertos y memorias ROM de un puerto, ver Figura 2.17

Para diseñar un circuito lógico en un FPGA se pueden utilizar los diagramas esquemáticos que representan de forma gráfica los componentes y sus conexiones; esta metodología no es practica si el diseño es complejo y se compone de miles de elementos. Por tanto, se hace necesaria la utilización de los lenguajes de descripción de hardware o hardware description languages (HDL). Una de las principales características de éstos, es su capacidad para describir algún diseño en distintos niveles de abstracción: funcional, estructural y a nivel de compuertas. El nivel de abstracción funcional describe el comportamiento del circuito, en lugar de tener que especificar los circuitos lógicos; por su parte el nivel estructural describe el hardware en términos de bloques y sus interconexiones. Por último, a nivel de compuertas el circuito se expresa en términos de ecuaciones lógicas [53].

Capítulo 3

Desarrollo del sistema

El éxito en las operaciones de ensamble utilizando robots industriales se basa en la exactitud del propio robot y en el conocimiento preciso del entorno, es decir, saber la geometría de las piezas y su localización en el espacio de trabajo. Los robots manipuladores en el mundo real operan con un alto grado de incertidumbre y requieren sistemas de detección para compensar errores durante los procesos; una amplia variedad de fuentes provocan estas incertidumbres como, errores de posicionamiento del robot, juego en los engranes, deflexión del brazo, falta de mantenimiento y otras perturbaciones que serían difíciles de controlar, por ello se busca un enfoque más simple como el uso de robots guiados por sistemas de visión para la ubicación de piezas en las tareas de ensamble [54].

En este trabajo se realiza un ensamble de dos piezas mecánicas, tipo *peg in hole* asistido por visión para la ubicación y reconocimiento de objetos; el sistema de control y procesamiento está implementado en hardware mediante FPGA. Para lograr el ensamble este sistema debe localizar los objetos de interés y diferenciar la pieza al igual que su contraparte, estas piezas mecánicas tienen formas de polígonos regulares básicos: cuadrados, círculos, triángulos, etc. Para reconocer los objetos, primero hay que ubicarlos y esto requiere calcular las posiciones de las piezas respecto al espacio de trabajo e identificar el lugar de ensamble para cada pieza mediante los parámetros del contorno, área y centroide de los objetos. Entonces, el proceso para realizar el ensamble se puede resumir en las siguientes tareas: adquirir la imagen, procesarla, encontrar el contorno, calcular el centroide, y área de las piezas, encontrar la ubicación de la pieza y de su contraparte, reconocer las piezas, llevar al manipulador a la coordenada del centroide de la pieza y sujetarla, hacer coincidir el centroide de la pieza con el de su contraparte y soltar la pieza para realizar el ensamble.

El FPGA es la parte central del sistema al fungir como unidad de control y procesamiento, todas las tareas se llevan a cabo en él; a lo largo de este capítulo se describe el comportamiento de los circuitos que permiten realizar el ensamble, y para empezar se explicará de forma general el comportamiento del sistema, ver Figura 3.1. Al iniciar su operación, la interfaz *SCCB* configura la cámara para empezar a enviar imágenes del área de trabajo donde se encuentran los objetos; mismas que se adquieren y se convierten a imágenes en escala de grises para guardarse en un bloque de memoria. Desde éste se lee la información al módulo "Segmentación de imágenes" donde se aplican los algoritmos del histograma, valor de umbral, binarización, etiquetado, contorno, área, centroide y cálculo de la función frontera; éste último parámetro se guarda en otro bloque de memoria desde donde es leído por el módulo "Reconocimiento de imágenes", aquí se identifican las figuras por medio de su firma, se comparan con base a su área para distinguir la pieza de la contraparte y estos últimos datos son enviados al módulo "Control de posición", donde se generan las ordenes a enviar al manipulador para que este llegue a la ubicación de la pieza y su contraparte. Por último, el módulo "Interfaz UART1" envía las órdenes al manipulador para realizar el ensamble. De forma paralela se despliega la imagen capturada por medio de la "Interfaz VGA" a un monitor, opcionalmente se envía esta misma imagen capturada, así como el área, centroide y función frontera de los objetos encontrados en la imagen. Después, el sistema queda en espera hasta que se indique el comienzo de la rutina nuevamente.

3.1 Adquisición de imágenes

3.1.1 Cámara

La cámara es el dispositivo usado por el sistema para recibir información del entorno y ser capaz de localizar los objetos de interés, el sensor a utilizar es el modelo *OV7670* de OmniVision, ver Figura 3.2. Éste es un dispositivo *CMOS* que integra en el mismo empaquetado varios módulos y circuitos para realizar funciones de pre procesamiento de imágenes. El OV7670 puede entregar imágenes con resolución *CIF* (Common Intermediate Format) de 352×288 pixeles, otras menores que CIF y *VGA* (Video Graphics Arrays); este último es el formato elegido, con un tamaño de 640×480 lo que da un total de $307,200$ pixeles activos.

A grandes rasgos se puede apreciar la estructura interna del sensor OV7670, ver Figura 3.3, en esta podemos ver que el sensor cuenta con un procesador análogo con el cual se controla el

balance de blancos y la ganancia automática de la imagen. También se aprecia el convetido A/D el cual es de 10 bits y trabaja a un máximo de 12 MHz. Asimismo, el sensor cuenta con un procesador digital de señales (DSP), el cual además de convertir el mapa de color a RGB ó YUV, permite configurar un pre-procesamiento a las imágenes como controlar la saturación, el matiz, el brillo, el contraste, etc.

Para usar la cámara es necesario que el sistema anfitrión, en este caso el FPGA, controle las señales que llegan al módulo de la misma. La alimentación para el módulo puede ser entre 3.3 V y 5.5 V en corriente directa. Como se ve en la Tabla 3.1 los pines PWDN y RESET# están siempre desactivados; note que el pin RESET# es activo bajo. Los pines D7 a D0 son por donde se transmiten las imágenes al FPGA, usando lo pines auxiliares VSYC, HREF y PCLK. El pin XCLK es la entrada de reloj para que el módulo funcione adecuadamente por lo que esta señal es sumamente importante y debe tener una frecuencia entre 10 y 48 MHz, con un ciclo de trabajo entre 45 y 55 % [55]. En este proyecto se usa una frecuencia de 24 MHz con un ciclo de trabajo del 50 %. Por último, los pines SIOC y SIOD son utilizados para configurar las funciones internas del módulo de la cámara, por medio del protocolo de comunicación *SCCB*; el cual se explica en la sección 3.1.2. La configuración de la cámara se realiza por medio de

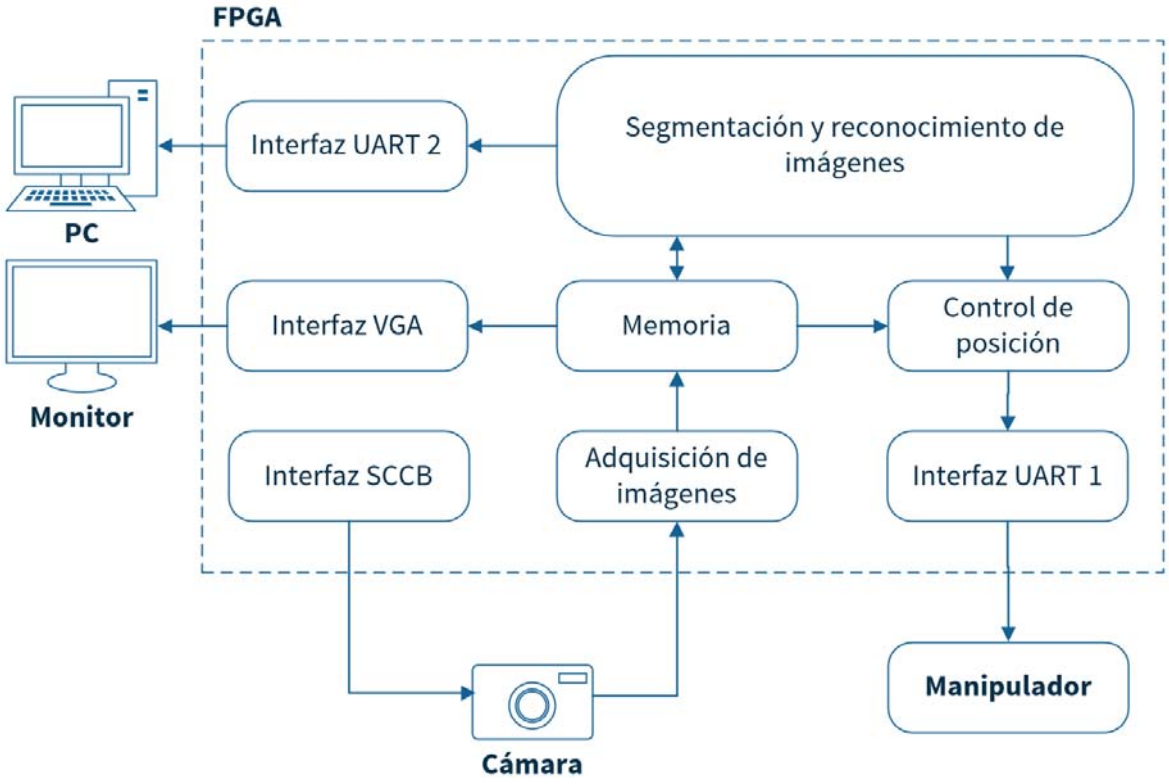


Figura 3.1 Diagrama de bloques del sistema desarrollado.

(a) Parte frontal (b) Parte posterior

Figura 3.2 Cámara OV7670

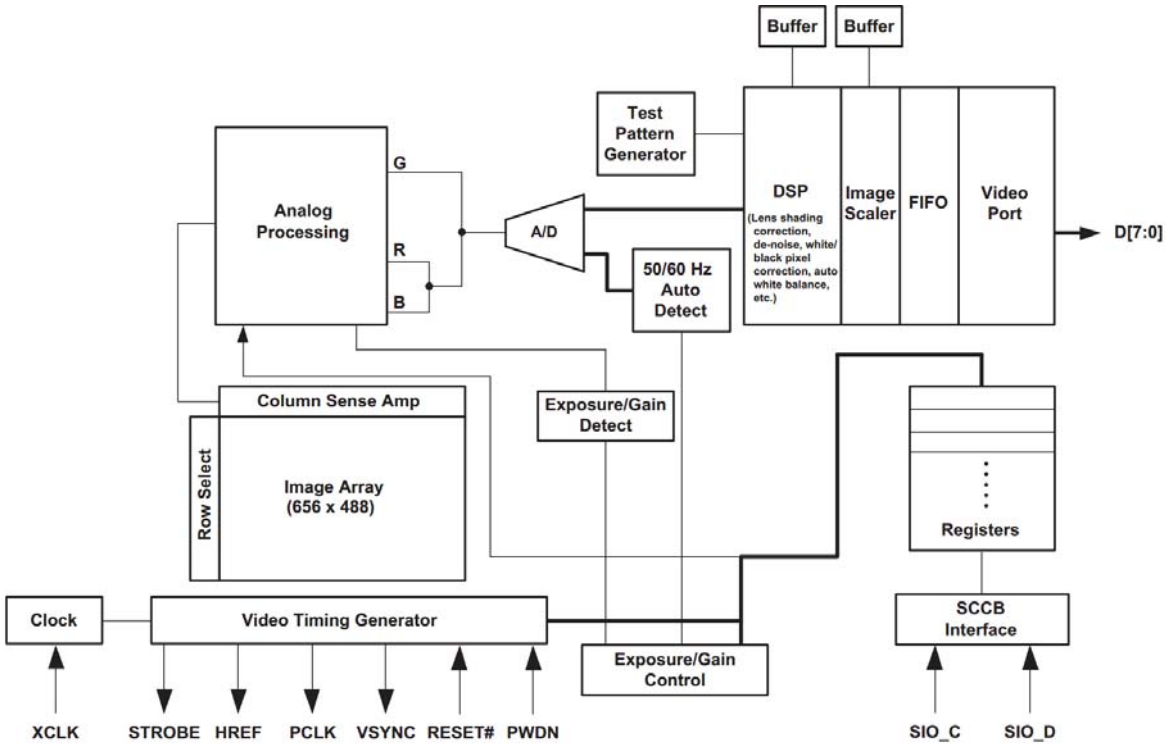


Figura 3.3 Diagrama de bloques del sensor OV7670.

registros, los cuales hay que establecer cada vez que se encienda esta y antes de comenzar a utilizarla; de lo contrario se usarán los valores predeterminados de fábrica.

Nombre	Tipo de pin	Función/Descripción
3V3	Polarización	Fuente de alimentación
GND	Polarización	Tierra
SIOC	Entrada	Señal de reloj de la interfaz SCCB
SIOD	Entrada/Salida	Datos de I/O de la interfaz SCCB
VSYNC	Salida	Señal "Vertical Sync"
HREF	Salida	Señal "HREF"
PCLK	Salida	Señal de reloj "Pixel"
XCLK	Entrada	Señal de reloj para el sensor
[D7... D0]	Salida	Señal de componente de video
RESET#	Entrada	Reajusta los registros al valor predeterminado
PWDN	Entrada	1: Cámara en modo de espera. 0: Modo normal.

Tabla 3.1 Descripción de pines del sensor OV7670.

3.1.2 Interfaz SCCB

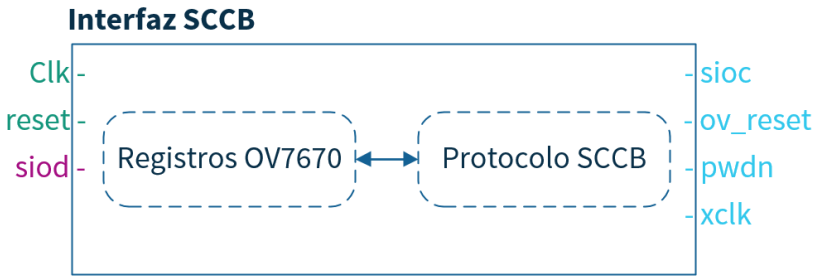


Figura 3.4 Diagrama de bloques para el módulo Interfaz SCCB implementado en el FPGA.

Para establecer los parámetros de funcionamiento de la cámara se utiliza el protocolo de comunicación serial SCCB creado por OmniVision Technology especialmente para usarse con sus dispositivos de video. Este protocolo utiliza al menos dos líneas de comunicación: SIOC la señal del reloj que es proporcionada por el dispositivo maestro y donde cada "1" lógico durante la etapa de transmisión indica un bit transmitido; la otra señal usada por el protocolo

es SIOD, esta señal es bidireccional, por ella se transmiten los datos entre el maestro y el esclavo, y solo puede cambiar en los "0" lógico de la señal SIOC. Para que el maestro transmita información al esclavo se hace por medio de ciclos de escritura que pueden ser de 3 o 2 fases. Se comenta únicamente y de forma general el funcionamiento de los ciclos de escritura de tres fases puesto que no hay recepción de datos desde la cámara, para más información sobre el protocolo consultar la hoja de datos [56].

Cada ciclo de transmisión consta de tres fases cada una compuesta por 9 bits, de los cuales 8 bits son de datos y el noveno es *don't care*. El bit más significativo es el primero en ser enviado. La fase 1 representa la dirección ID del esclavo, la fase 2 es la subdirección o registro dentro del esclavo y la fase 3 contiene el byte a escribir. La fase 1 selecciona el dispositivo esclavo con el cual se desea establecer la comunicación, cada dispositivo tiene una dirección ID única, y se compone de 7 bits más el bit menos significativo que es el selector de lectura o escritura, 1 o 0 respectivamente, este bit especifica la dirección de transmisión del presente ciclo; para la cámara ov7670 su dirección ID es 42_{16} para escribir a y 43_{16} para leer de esta. La fase 2 identifica el registro a escribir en el dispositivo; existen 202 registros en el ov7670 cada uno de 8 bits, estos van desde la dirección 00_{16} hasta la $C9_{16}$. En estos registros están distribuidas todas las instrucciones que describen el comportamiento de la cámara [55].

La implementación en el FPGA del protocolo SCCB, consta de dos bloques; el primero (Registros OV7670) almacena la dirección ID, los registros y sus valores a escribir en la cámara, ver Figura 3.4. El segundo bloque (Protocolo SCCB) se encarga de transmitir la información de las fases en tiempo y forma hacia la cámara, cada que termina un ciclo de escritura se lee del primer bloque el valor correspondiente para la fase 2 y 3; así continua el proceso hasta que se terminan de escribir todos los registros deseados, ver Figura 3.5. Al terminar de escribir en éstos es necesario activar la señal RESET# de la cámara por al menos 1 ms antes de ver aplicados los cambios.

3.1.3 Recepción de datos

La cámara soporta diferentes formatos de salida, entre ellos están: Raw RGB, RGB 565/555/444, YUV (4:2:2) y YCbCr (4:2:2). El formato seleccionado es el RGB 444, ya que facilita el manejo de datos y la impresión de éstos en pantalla, puesto que el puerto VGA del FPGA también usa este formato; además que el procesamiento realizado no necesita de mucha profundidad de color. El formato RGB444, implica que cada pixel tiene $L = 2^4 = 16$ niveles de intensidad para

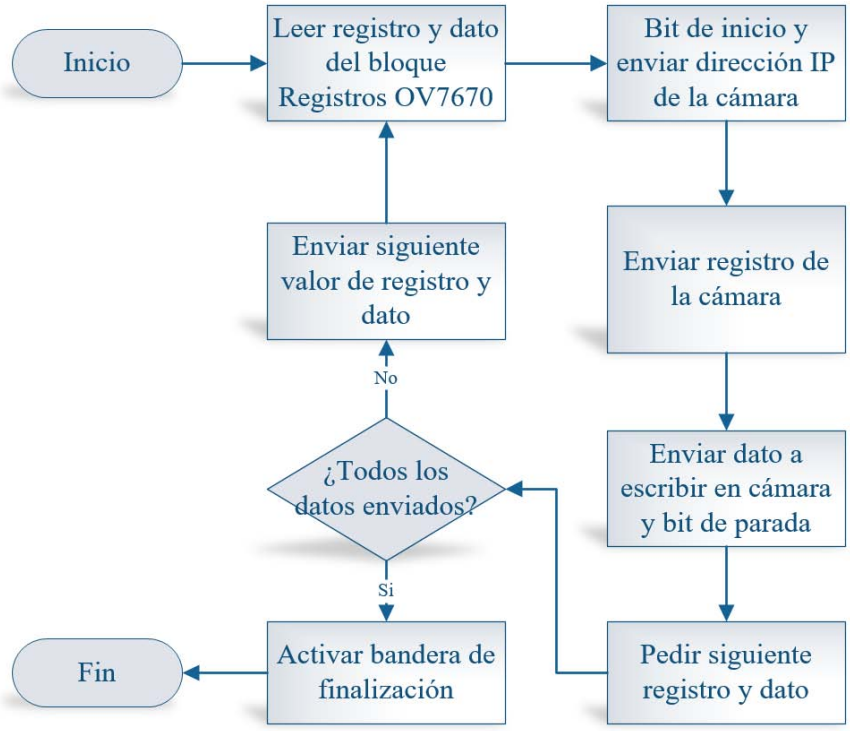


Figura 3.5 Diagrama de flujo del protocolo de comunicación SCCB entre la cámara y el FPGA.

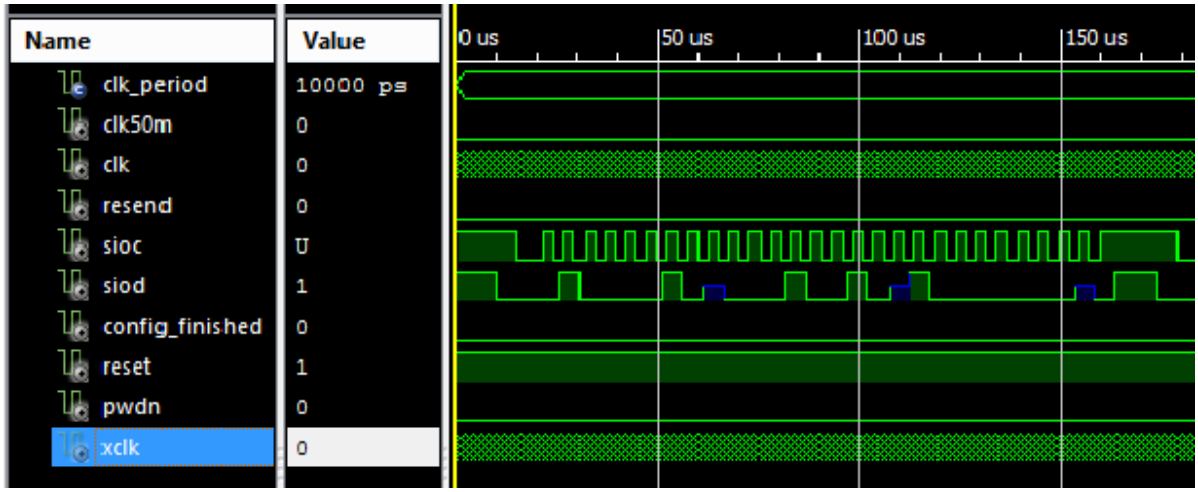


Figura 3.6 Simulación del protocolo SCCB en ISIM.

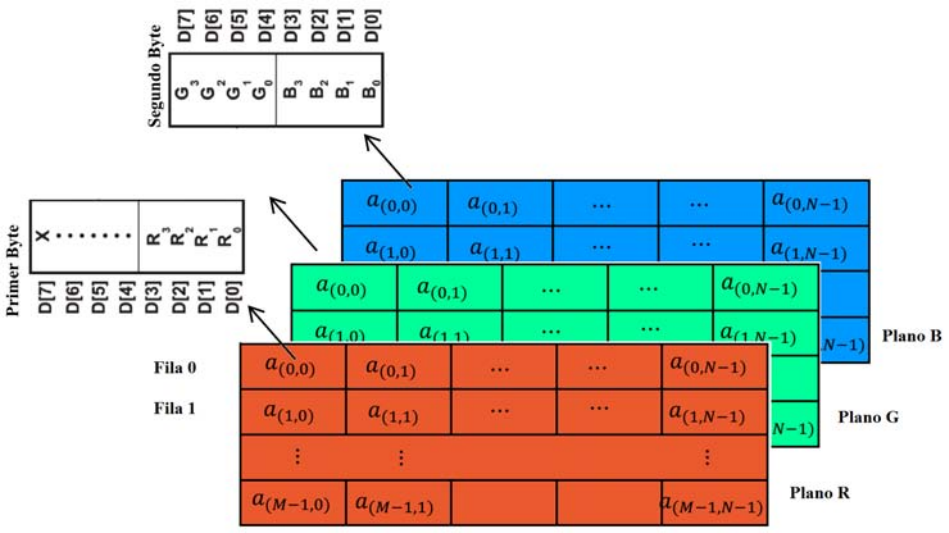


Figura 3.7 Formato de salida RGB444 para las imágenes de la cámara.

cada plano de color; este formato de salida necesita 12 bits o 2 bytes para representar un solo píxel y el valor de cada uno es definido por las señales R, G y B, ver Figura 3.7.

La imagen de salida es un arreglo numérico ordenado ya que los píxeles salen de la cámara iniciando por el elemento $(a_{0,0})$ de la fila 0 y al llegar al último elemento de la fila $(a_{0,N-1})$ se prosigue con la siguiente fila y así sucesivamente hasta terminar con todos los píxeles; este proceso se repite con cada imagen, ver Figura 3.7. El protocolo de salida que utiliza la cámara para enviar las imágenes al anfitrión es similar al VGA. Una señal a color de video VGA está compuesta por 5 señales diferentes: dos de sincronización (HSYNC y VSYNC) y tres analógicas de color (R, G, B). En el caso de la cámara son: dos señales de sincronización (HREF y VSYNC) y un bus de datos de 8 bits, D[7:0], por donde se enviaran las señales digitales de R, G y B. Estas cinco señales son activas altas. La salida de datos del bus D, esta sincronizada con la señal de reloj de la cámara, PCLK; cada pulso de reloj el bus de datos cambia, pero solo cada dos pulsos de reloj el bus de datos manda un nuevo pixel; así que para enviar toda una fila de un cuadro de la imagen se requieren de 1280 pulsos de reloj, ver Figura 3.8. Al terminar el envío de una fila la señal HREF se desactiva y pasa un tiempo antes de que vuelva a activar para enviar una nueva fila. De forma análoga se comporta la señal VSYNC, que es activa después de terminar de enviar un cuadro y antes de enviar el siguiente, ver Figura 3.9 [55] [56].

La señal de reloj PCLK es muy importante ya que determina la sincronización del envío de datos de la cámara al FPGA, también las señales HREF y VSYNC se ven afectadas por PCLK.

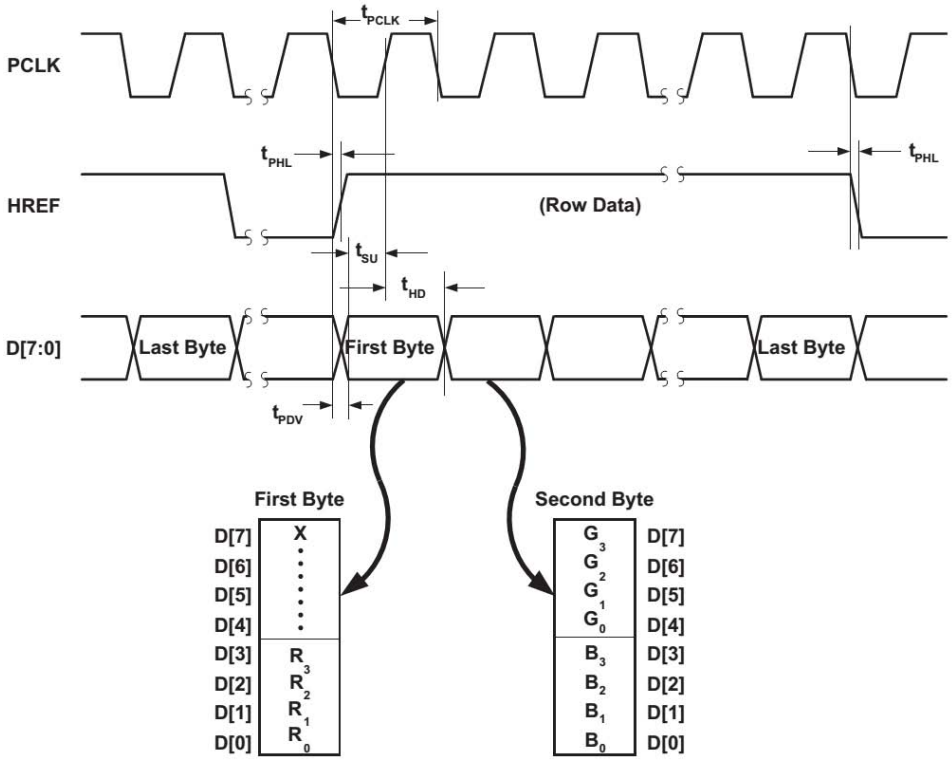


Figura 3.8 Diagrama de tiempo para el formato RGB444.

La frecuencia de PCLK se puede calcular por medio de:

$$PCLK = F_{int} = F(inputclock) * DBLV[7:6]/2 * (Bit[5:0]+1) \tag{3.1}$$

donde, $F(inputclock)$ es la frecuencia de la señal XCLK; $DBLV[7:6]$ es el valor del PLL configurado en el registro DBLV en la dirección $6B_{16}$ de la cámara y $Bit[5:0]$ es una constante que va desde 0 hasta 31 y se tiene que configurar en el registro CLKRC en la dirección 11_{16} de la cámara [57]. La señal PCLK también determina el número de cuadros por segundo que envía la cámara al FPGA, de acuerdo con la Figura 3.9, el tiempo que está activa una imagen es de $510 \times t_{LINE}$; que en términos del periodo de PCLK es $799,680 \times t_{pclk}$. Puesto que la frecuencia de XCLK es de 24 MHz, entonces la frecuencia de PCLK es de 12 MHz que equivale a tener una frecuencia de 15 FPS a la salida de la cámara.

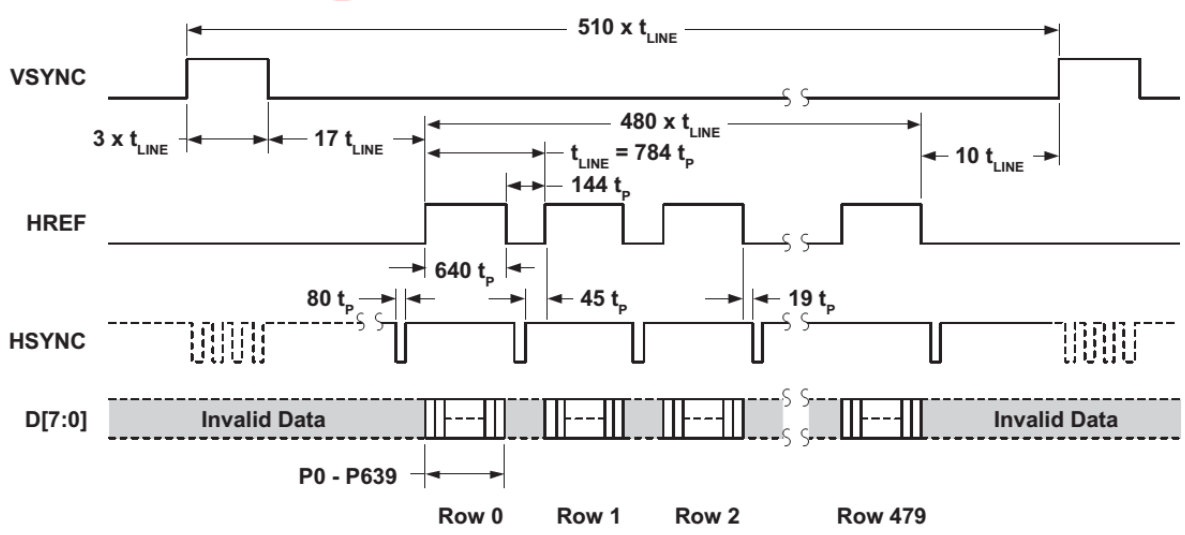


Figura 3.9 Diagrama de tiempo de las señales de salida de la cámara para un cuadro.

$$t_p = 2 \times t_{pclk}$$

Figura 3.10 Ejemplo de la conversión de una imagen RGB a escala de grises utilizando el método de luminosidad. A la izquierda, imagen original y a la derecha imagen convertida.

3.1.4 Imágenes RGB a escala de grises

Las imágenes proporcionadas por la cámara son en formato *RGB444*, sin embargo, para reducir al mínimo los datos necesarios para representar una imagen, tener un mejor desempeño computacional y obtener de forma rápida y fácil las propiedades geométricas y morfológicas de los objetos, se emplean imágenes binarias en todos los algoritmos utilizados en este trabajo [58]. Por tal motivo se deben transformar las imágenes en RGB a binarias y para ello antes se necesita convertirlas a escala de grises. Existen al menos dos métodos para convertir una imagen de color a escala de grises, el método del promedio y el de luminosidad. En el primero como su nombre lo dice, se calcula el promedio del pixel de todas las componentes del color; por tanto, el color rojo, verde y azul resultan con el mismo nivel de gris al ser convertidos. El método de luminosidad se calcula de acuerdo con la ecuación 3.2 en donde cada componente de color recibe un determinado peso, ya que el ojo humano es más sensible a las componentes del canal verde y rojo que al azul [59]; por esta razón se utiliza este método. En la figura 3.10 se aprecia el resultado al aplicar el método de luminosidad a una imagen RGB; y en el anexo A se incluye el código de Matlab para ambos métodos. Las imágenes que salen de la cámara son en formato RGB444 por lo que cada componente de color solo tiene 4 bits de resolución y después de aplicar el método de luminosidad la imagen en escala de grises también tiene 4 bits de resolución pero solo una componente.

$$(0.299 * R + 0.587 * G + 0.114 * B) \tag{3.2}$$

System Generator

System Generator es una herramienta de alto nivel con la cual se puede programar en el entorno gráfico de Simulink de Matlab, utilizando bloques proporcionados por Xilinx especialmente para este propósito. Posteriormente System Generator se encarga de transformar estos bloques en código HDL. Los bloques de Simulink sirven de apoyo tanto para generar estímulos en el diseño como para visualizar, analizar y almacenar los datos resultantes; pero estos bloques no se traducen a HDL, solo los bloques de System Generator son sintetizables. La ventaja de ésta herramienta es que se pueden llevar a cabo simulaciones y co-simulaciones del código de forma más rápida sin necesidad de realizar códigos de prueba (testbench). Puesto que en el FPGA resulta más laborioso usar números flotantes, se decidió probar el ambiente de system generator para generar el código del bloque RGBtoGrises, en este bloque solo se aplica la ecuación 3.2, ver Figura 3.12 y Figura 3.11. Debido a que representar un número decimal con

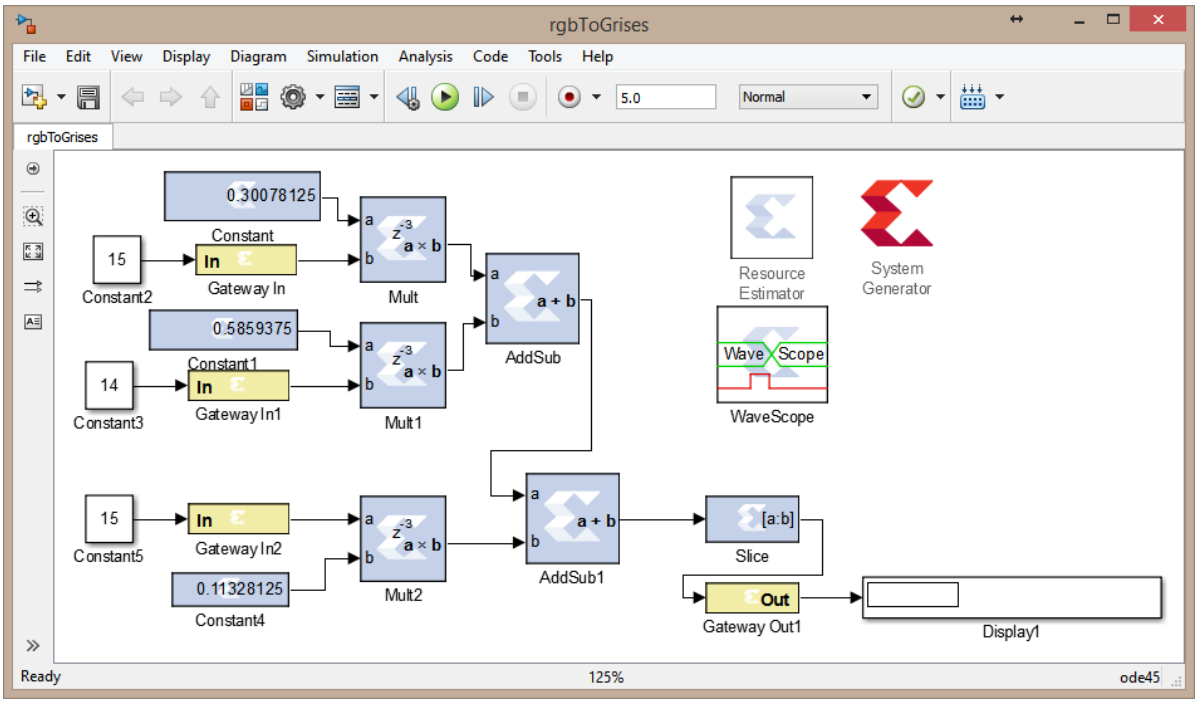


Figura 3.11 Método de luminosidad programado en System Generator.

precisión requiere de muchos bits, solo se utilizaron ocho bits para representar cada uno de los coeficientes de la ecuación 3.2, por lo que ésta quedo como $(0.300 * R + 0.585 * G + 0.113 * B)$.

Implementación en FPGA

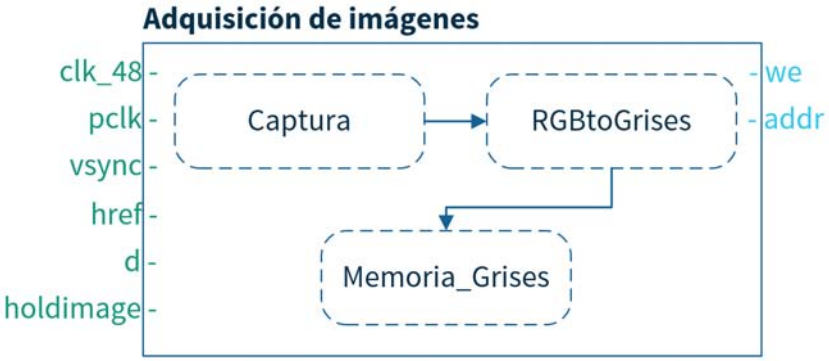


Figura 3.12 Diagrama de bloques para el módulo de Adquisición de imágenes implementado en el FPGA.

El bloque "Captura" está sincronizado con la señal PCLK, por lo que en cada estado activo de la señal se lee el bus de datos D, y cada dos ciclos se junta el byte anterior con el actual para formar

el pixel actual y guardarlo en un registro temporal "dout" de 12 bits, donde la información está contenida de la siguiente manera: dout(11:8) plano R, dout(7:4) componente G y dout(3:0) plano B , ver Figura 3.8 y Figura 3.12. Este registro dout pasa al bloque "RGBtoGrises" el cual aplica el método de luminosidad para convertir el píxel actual a un píxel con solo una componente en la escala de grises y con $L = 2^4 = 16$; este nuevo valor se guarda en la "Memoria_Grises". Para almacenar un fotograma completo con resolución VGA, se requieren tener $640 \times 480 = 307,200$ localidades de memoria cada una con 4 bits de longitud, para implementar esta memoria se utilizan las BRAM del FPGA. Para direccionar los píxeles en grises a la memoria se necesitan al menos tres señales, una que representa el dato a escribir, otra con la dirección en memoria donde se almacena el dato y la última es la habilitación para escribir en memoria (activa alta); estas señales son: "gray_out", "addr" y "we" respectivamente, ver Figura 3.13. Los píxeles de la imagen se van guardando en la memoria en el orden en que van entrando al FPGA, de forma que los píxeles de la primera fila de la imagen se almacenan en las localidades 0 a 639, la segunda fila en las localidades 640 a 1279 y así sucesivamente hasta guardar el último píxel (479,639) en la localidad de memoria 307,199.

3.2 Segmentación y reconocimiento de imágenes

3.2.1 Histograma y valor de umbral

Una vez que se tienen las imágenes en escala de grises hay que convertirlas a binarias para poder aplicar los algoritmos que detecten la morfología de la pieza; para ello se va a utilizar antes que nada el histograma de la imagen. Para encontrarlo hay que realizar un barrido visitando cada uno de los pixeles y contando el número de veces que aparece cada uno de los valores permitidos en la imagen. Por ejemplo, en la Figura 3.14a se presenta un imagen en escala de grises con 8 bits de profundidad de color, es decir 256 valores permitidos para cada píxel, y en la Figura 3.14b se observa el histograma de esta imagen.

Una vez conocemos el histograma debemos encontrar un valor de umbral para convertir la imagen en binaria. El umbral varía con cada imagen, no siempre puede ser el mismo valor puesto que cada histograma también varía, por eso para cada imagen es necesario calcularlo. Gracias a que la imagen se compone de dos regiones: objetos y fondo, el histograma siempre tendrá dos crestas como en la Figura 2.10, normalmente siempre habrá un intervalo $[a,b]$ entre estas crestas, entonces tomamos el valor del umbral como la posición donde se encuentra el

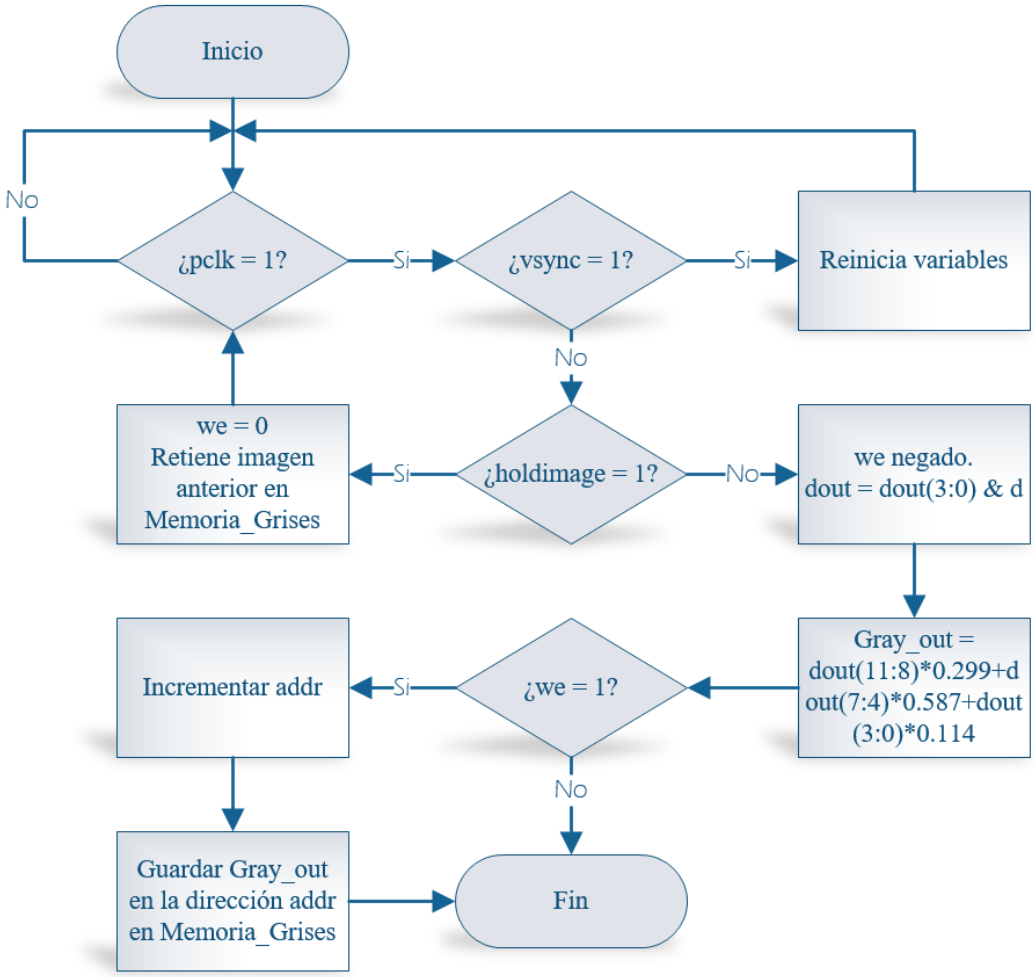
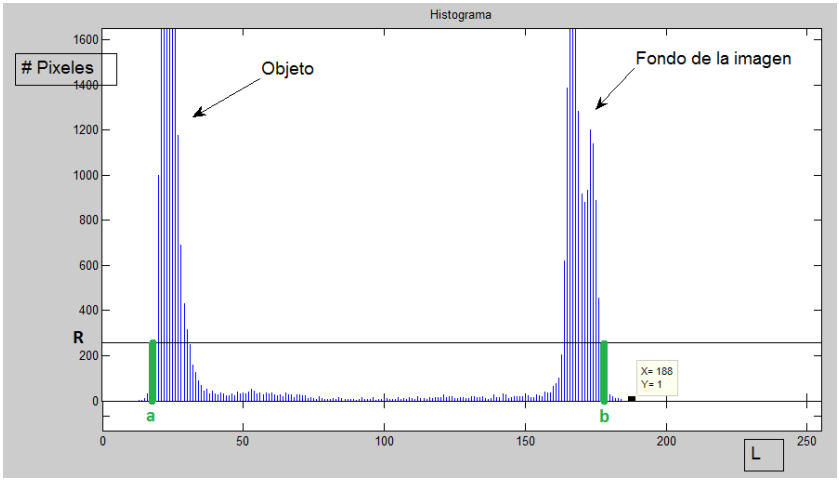


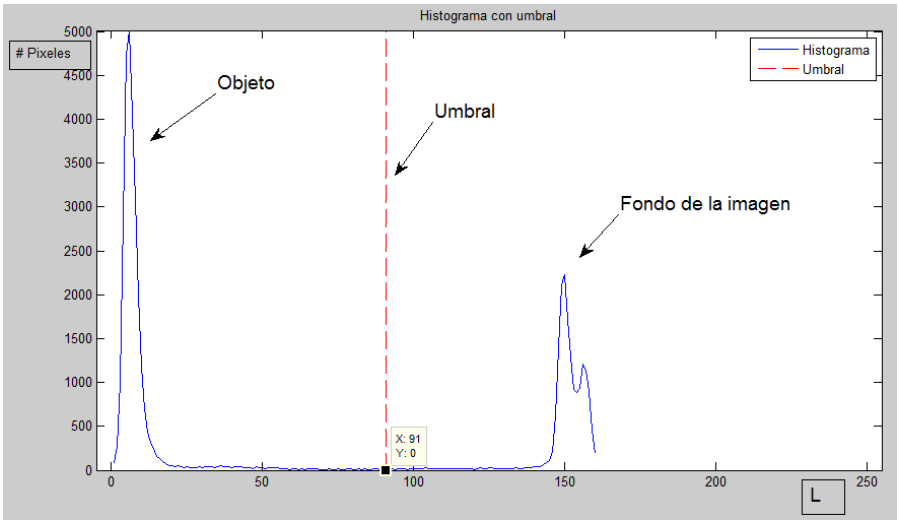
Figura 3.13 Diagrama de flujo para el módulo de adquisición de imágenes en el FPGA.

mínimo del intervalo $[a,b]$; de esta forma el valor de umbral siempre estará entre las crestas del histograma que representan el fondo y el primer plano del objeto, ver Figura 3.14b. Cabe mencionar que el rango de este segundo histograma es el intervalo $[a,b]$ y no el rango del histograma original, ver Figura 3.14c. En la Figura 3.15 se muestra el diagrama de flujo para obtener el valor de umbral a partir del histograma de una imagen con alto contraste, este algoritmo se utilizó para obtener el umbral en la Figura 3.14c y también este mismo algoritmo se implementa en el FPGA.

(a) Objeto (rectángulo) en contraste con el área de trabajo



(b) Obtener el histograma de la imagen.



(c) Calcular el valor de umbral que permite separar el objeto de la imagen.

Figura 3.14 Método usado para encontrar un objeto dentro de una imagen.

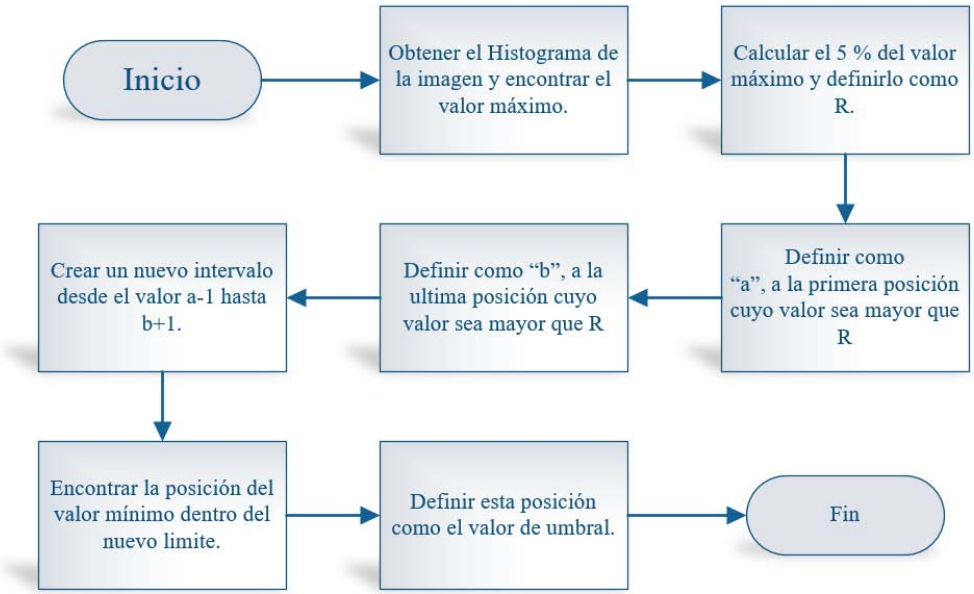
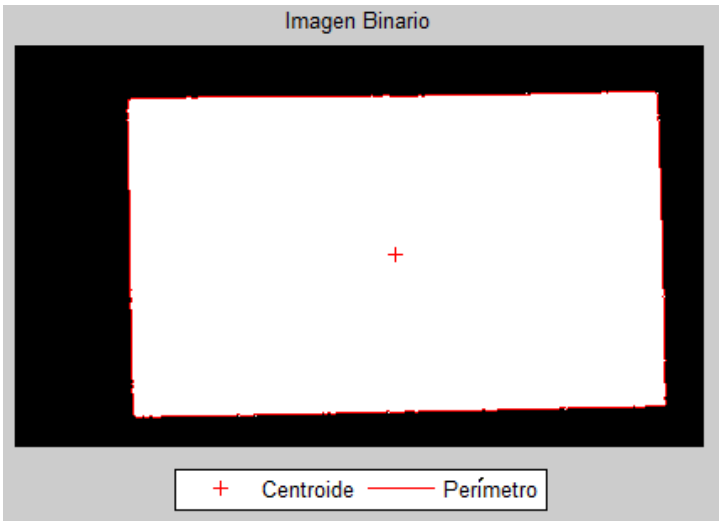


Figura 3.15 Diagrama de flujo para obtener el valor umbral en un histograma.

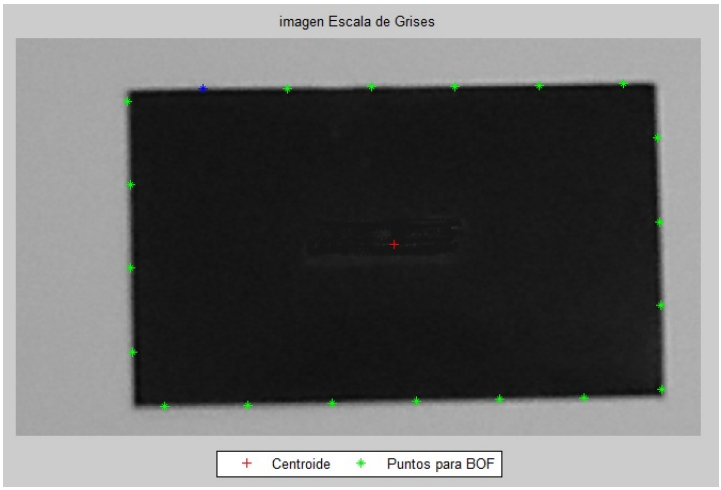
3.2.2 Imagen binaria

La imagen binaria solo puede tomar uno de dos valores en sus pixeles, 1 o 0, donde el 1 puede representar el fondo de la imagen y el 0 el objeto o al contrario el 1 el objeto y el 0 el fondo; esta última configuración es la que elige como predeterminada de ahora en adelante para los algoritmos utilizados en Matlab y en el FPGA. Para generar una imagen binaria necesitamos una imagen en escala de grises y el valor del umbral u, por ejemplo, al utilizar la imagen de la Figura 3.14a y el valor umbral $u = 91$ obtenemos la Figura 3.16a. El algoritmo para obtener la imagen binaria se basa en realizar un barrido de izquierda a derecha y de abajo a arriba por cada uno de los píxeles y dependiendo de la ecuación 3.3 obtendremos el nuevo valor para cada pixel.

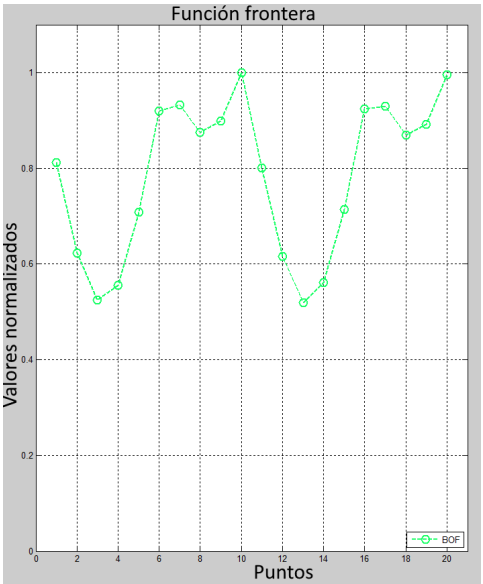
$$\begin{aligned} f(x,y) > u &\Rightarrow f(x,y) = 1 \\ f(x,y) \leq u &\Rightarrow f(x,y) = 0 \end{aligned} \tag{3.3}$$



(a) Posición del centroide y contorno del objeto.



(b) Elementos seleccionados del contorno para aplicar la BOF.



(c) Función frontera del objeto.

Figura 3.16 Función frontera aplicada a la imagen 3.14a

3.2.3 Etiquetado, contorno, área y centroide

Entre las regiones presentes en una imagen binaria se encuentran aquellas que corresponden con los objetos de interés; y para poder identificar los píxeles que pertenecen a cada una de las regiones y asociar estas a los objetos para extraer sus características, es necesario obtener las componentes conexas presentes en la imagen. Si en la imagen aparecen varias regiones, no se podrán obtener el contorno, área y centroide de cada una hasta que estas no estén identificadas de forma individual. Al proceso de asignar una misma etiqueta a cada uno de los píxeles que pertenece a una componente conexa se le conoce como etiquetado; esta etiqueta es un valor numérico que permita identificar todos los píxeles que forman parte de una misma región. Así pues la imagen de etiquetas permite la extracción de características de cada una de las regiones conexas de la imagen; para definir las componentes conexas se usará la 8-vecindad [58]. Por otro lado, los contornos de una componente conexa contienen información muy valiosa, pues proporcionan la silueta de los objetos, además que al juntar esta información con la del área se puede conocer la compacidad y tener una idea de la forma del objeto.

El algoritmo más sencillo para conocer el contorno de una región es el de la tortuga de Papert o square tracing, sin embargo, este presenta fallas al aplicarse a imágenes binarias en las cuales las regiones tienen agujeros [58]. El algoritmo a utilizar para realizar el etiquetado de regiones y obtener el contorno está basado en el propuesto por Fu Chang [60], el cual tiene la característica de no solo encontrar los contornos externos e internos de una región, sino que simultáneamente realiza el etiquetado de la imagen binaria, todo esto en un único barrido de la imagen. Encontrar el área y centroide de las regiones en la imagen necesitaría de otro algoritmo que barriera la imagen una vez más; sin embargo, para reducir las tareas a realizar por el sistema y el tiempo de ejecución del mismo, se tomó el algoritmo de Chang [60] y se modificó para poder calcular también el área y centroide de cada una de las regiones en la imagen, y ahora si en un solo barrido. A continuación se describe el funcionamiento de este; y en la Figura 3.17 se muestra el algoritmo y en el anexo A se encuentra el código para su simulación en Matlab.

Una condición que se debe cumplir para poder aplicar el algoritmo de Chang es que no sé accede a posiciones fuera de los límites de la imagen, es decir, que no haya regiones en contacto con estos límites o que todos los píxeles en los bordes de la imagen sean 0. Este algoritmo utiliza la imagen binaria y asume que las regiones tienen valor 1 y que el fondo es 0. El algoritmo puede explicarse en función de las diferentes decisiones que se adoptan según las propiedades de los diferentes puntos de una región. Al iniciarse el barrido de una fila desactivamos la bandera, *etiq_activa = 0*, que sirve para indicar si una región está siendo

B: *Imagen binaria de tamaño (filas,columnas,1 bit de profundidad)*
E. *Imagen etiquetada de tamaño (filas,columnas,4 bit de profundidad)*
cen: *Guarda los centroides de cada una de las regiones*

Función *SeguidorEtiquetadoSimultaneo(B)***es**
  Inicializar $netiq = 0, E = 0, C = \varnothing$
  Para cada *x=2* **to** *filas-1* **haz**
    etiq_activa=0
    Para cada *y=1* **to** *columnas-1* **haz**
      Si *B(x,y)==1* **entonces**
        Si $etiq\_activa \neq 0$ **entonces**
          Si *E(x,y)==0* **entonces**
            E(x,y)=etiq_activa %Etiquetar la región etiq_activa
            area(etiq_activa)+1 %Cálculo del area
            Cx(etiq_activa)=Cx(etiq_activa)+y
            Cy(etiq_activa)=Cx(etiq_activa)+x
          Fin
        en caso contrario
          etiq_activa=E(x,y)
          Si *etiq_activa==0* **entonces**
            netiq=netiq+1 %Nueva región
            etiq_activa=netiq
            P_e=(x,y) %Punto inicial del contorno externo
            (area,Cx,Cy,C_e,E)=**SeguidorContorno(B,E,etiq_activa,**P_e**,0, area,Cx,Cy)**
            Añadir c_e a C(etiq_activa)
            E(x,y)=etiq_activ
          Fin
        Fin
        %Cálculo del centroide de la región etiq_activa
        cenx(etiq_activa)=$Cx(etiq\_activa)area(etiq\_activa)$
        ceny(etiq_activa)=$Cy(etiq\_activa)area(etiq\_activa)$
      en caso contrario
        Si $etiq\_activa \neq 0$ **entonces** %Se abandono la región
          Si *E(x,y)==0* **entonces** %Punto en agujero interno
            P_i=(x-1,y) %Punto inicial del contono interno
            (area,Cx,CyC_i,E)=**SeguidorContorno(B,E,etiq_activa,**P_i**,3, area,Cx,Cy)**
            Añadir c_i a C(etiq_activa)
          Fin
          etiq_activa=0
        Fin
      Fin
    Fin
  Fin
Fin

Figura 3.17 Algoritmo para el cálculo del área, centroide, contorno y etiquetado simultáneo.

inspeccionada; entonces, durante el barrido de una fila de la imagen binaria $B(x,y)$ se llevan a cabo los siguientes pasos:

1. Encontrar por primera vez un punto externo (P_e) de una región, en este momento se crea una nueva etiqueta *netiq* con la que será etiquetada la región y se asigna a la variable *etiq_activa*. El punto P_e es un punto del contorno exterior de la región netiq. Mediante una llamada a la función SEGUIDORCONTORNO(), ver Figura 3.18, se creará una lista con los puntos de este contorno y se les etiquetará con el valor *etiq_activa* en la imagen de etiquetas *E(x,y)*, también se incrementará la variable *área* y las coordenadas x, y se sumaran a las variables *cy* y *cx* para el cálculo del centroide. Esta función, con independencia de si el contorno es externo o interno, marca en E con un valor de 15 todos los vecinos del contorno que no pertenecen a la región. El marcado previene que un contorno se recorra más de una vez.

2. Si mientras la bandera *etiq_activa* es activa y diferente de cero, se encuentra un punto en $B(x,y) = 1$, entonces $E(x,y) = etiq\_activa$, $area(netiq) = area(netiq) + 1, cx(netiq) = cx(netiq) + y$ y cy(netiq)=cy(netiq)+x.

3. Si mientras la bandera *etiq_activa* es activa y diferente de cero, se encuentra un punto $B(x,y) = 0$ entonces:

   - Si su etiqueta es 15 implica que es un punto vecino de los contornos ya recorridos de la región, no se hace nada, la bandera se desactiva, *etiq_activa=0*.

   - Si su etiqueta es 0, significa que es el primer punto encontrado de uno de los posibles agujeros de la región y el punto inmediatamente superior $P_i = (x, y-1)$ por fuerza es un punto de un contorno interior. Una llamada a la función SEGUIDORCONTORNO() creará una lista con los puntos de este contorno interno y los etiquetará con el valor *etiq_activa* en la imagen de etiquetas $E(x,y)$.

4. Si estando *etiq_activa* desactivada se encontrase un punto B(x,y) del contorno exterior o interior de la región etiquetado con un valor netiq, obviamente esto implica que el punto anterior B(x-1,y) está a 0. Activaremos la bandera, *etiq_activa=E(x,y)*, que indica que una región está siendo visitada.

La función SEGUIDORCONTORNO() realiza varios cometidos:

- Obtiene los contornos externo e internos de una región.

- Marca con el valor de etiqueta correspondiente los puntos de contorno.

etiq: *Etiqueta de la región*
$P_{fin} = (x,y)$: *Coordenadas del punto de entrada. Se almacenan al final del contorno*
codigo: *Código inicial de búsqueda respecto a* P_{act}*(0 exterior, 3 interior)*
c: *Vector con las coordenadas del contorno*
area *Guarda la cuenta del área de cada una de las regiones*
Cx *Guarda las coordenadas en x del centroide de cada una de las regiones*
Cy *Guarda las coordenadas en y del centroide de cada una de las regiones*

Función *SeguidorContorno(B,E,etiq,*P_{fin}*,codigo,area,Cx,Cy)***es**
    Inicializar c=0
    contador=1
    (P_{sig},codigo)=**SiguientePunto(B,E,**P_{fin}**,codigo)**
    c(contador)=P_{sig} %Primer punto del contorno
    Si $P_{sig} \neq P_{fin}$ **entonces** %El punto no es un punto aislado
        seguir=1 **Mientras** *seguir=1* **:** %Busqueda en los 7 vecinos
            Si *E(*$P_{sig}x, P_{sig}y$*==0)* **entonces**
                E($P_{sig}x, P_{sig}y$)=etiq %Etiqueta puntos del contorno
                area(etiq)=area(etiq)+1 %Cálculo del área
                Cy(etiq)=Cy(etiq)+$P_{sig}x$
                Cx(etiq)=Cx(etiq)+$P_{sig}y$
            Fin
            codigo=MOD(codigo+6,8) $P_{ant} = P_{sig}$
            (P_{sig},codigo)=**SiguientePunto(B,E,**P_{ant}**,codigo)**
            Si $P_{sig} == c(1)$ *AND* $P_{ant} == P_{fin}$ **entonces**
                seguir=0 %Condición de parada
            en caso contrario
                contador=contador+1
                c(contador)=P_{sig}
            Fin
        Fin
    Fin
    Regresar c
Fin

Figura 3.18 Función Seguidor Contorno().

- Marca con 15, los puntos 8-vecinos de los contornos que pertenecen al fondo.

5	6	7
4	**P**	0
3	2	1

Figura 3.19 Código de vecindad de Moore del punto P.

La función SEGUIDORCONTORNO() debe garantizar que no termina antes de lo debido al visitar de nuevo el punto de inicio, la condición de parada consiste en guardar el punto inicial P_e o P_i, (que se guardará en la última posición del contorno, y se conoce como punto P_{fin}) y su sucesor P_{ini} (que se guardará en la primera posición del contorno). La condición de parada ocurre cuando el punto actual P_{act} y su sucesor P_{sig} coinciden respectivamente con P_{fin} y P_{ini}. Además, esta función debe considerar la presencia de una región monopíxel, que se detecta al comprobar que P_{ini} coincide con P_{fin}.

La función SEGUIDORCONTORNO() debe decidir para cada punto P_{act} del contorno donde buscar su sucesor P_{sig}. Salvo en los casos particulares de los puntos de entrada a la función, cualquiera que sea la configuración del punto P_{act}, el punto sucesor P_{sig} se hallará buscando en el sentido horario uno de sus 8-vecinos. Para decidir por cual de esos 8-vecinos se inicia la búsqueda, se utiliza la denominada vecindad de Moore o códigos de Freeman, ver Figura 3.19. El código correspondiente a los puntos iniciales de los contornos es diferente según se trate del punto inicial de un contorno exterior o interior. Si el punto de entrada P_e corresponde a un contorno exterior, este punto tiene sus 8-vecinos de la fila situada encima, igual a 0; por tanto, basta con comenzar la búsqueda en el píxel situado al Este, es decir, código = 0. Si el punto de entrada P_i corresponde a un contorno interior, este punto está situado al Norte del primer punto del agujero de la región, y el píxel que precede a este último es un punto de la región, 8-vecino de P_i situado al Suroeste, es decir código = 3. Para el resto de los casos, todo lo que se necesita es saber el código del punto P_{act} respecto a su antecesor P_{ant}. Basta calcular el nuevo código de posición mediante la operación $codigo = MOD(codigo + 6, 8)$, donde MOD() es la operación que regresa el residuo de una división.

La función SIGUIENTEPUNTO() realiza la búsqueda del siguiente punto del contorno en la 8-vecindad de P_{act}, partiendo del vecino situado en la posición determinada por *codigo* y siguiendo las manecillas del reloj. Si tras completar el ciclo de 7 visitas no se encontrará ningún

punto, significaría que el punto P_{act}, es decir P_e, es un punto aislado. A partir de esta posición inicial se visitan los vecinos de P_{act} tal que:

- Si el punto P_{sig} no pertenece a la región, se marca con un valor de etiqueta no válido, 15, y se pasa al siguiente vecino según el movimiento de las agujas del reloj.
- Si el punto P_{sig} pertenece a la región se devuelven sus coordenadas y el nuevo valor *codigo*.

Los puntos del contorno se almacenan en una memoria diferente para cada etiqueta al igual que el número de elementos del contorno y el área que conforman cada región. La información de cada contorno se utiliza posteriormente para el cálculo de la función frontera, mientras que la información del área y del centroide se utiliza para el cálculo de la compacidad y para realizar el ensamble de piezas respectivamente.

```
Función SiguientePunto(B,E,P_act,codigo)es
    exito=0
    contador=1
    Mientras exito==0 AND contador<8 :        %Busqueda en los 7 vecinos
        P_sig = (x,y) = P_act+Vecino(codigo)                %Nueva coordenada
        Si B(x,y)==0 entonces            %El punto no pertenece a la región
            E(x,y)=15                                    %Se marca el pixel
            codigo=MOD(codigo+1,8)  %Siguiente posición según agujas del
            reloj
        en caso contrario
            exito=1
        Fin
        contador=contador+1
    Fin
    Si exito==0 entonces
        P_sig = P_act          %No se encontro ningún vecino:  punto aislado
    Fin
    Regresar P_sig,codigo
Fin
```

Figura 3.20 Función SiguientePunto().

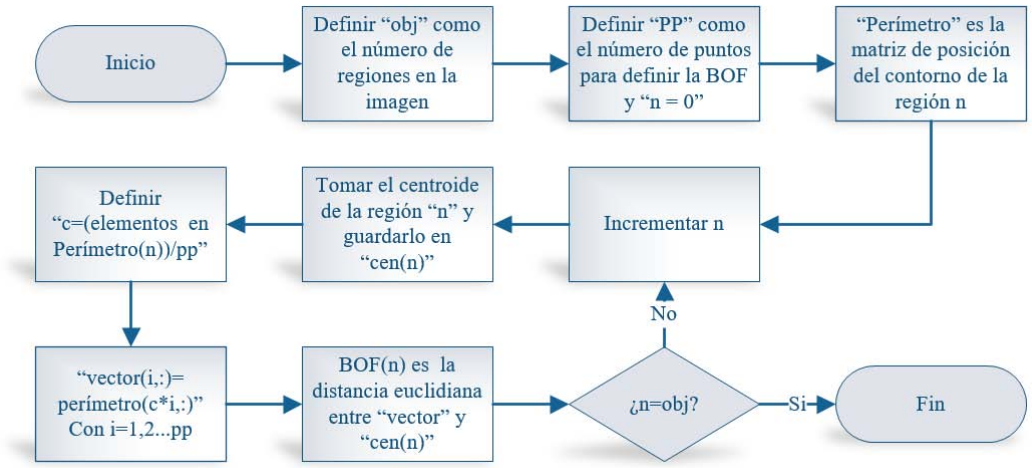


Figura 3.21 Diagrama para obtener la BOF de PP puntos

3.2.4 Función frontera (BOF)

La Función Frontera del objeto permite al sistema saber con mayor certeza la descripción de las regiones previamente etiquetadas por el algoritmo Seguidor etiquetado simultáneo. La BOF obtiene la firma de un objeto o forma, la cual como se mencionó, no cambia con la traslación, rotación o tamaño de este; de hecho lo único que puede cambiar en la BOF es la resolución con la que se representa, esto es, el número de puntos que se toman del contorno. Por ejemplo, la Figura 3.14a tiene $k = 20$, esto significa que toma 20 muestras del contorno distribuidas uniformemente, ver Figura 3.16b, donde en verde se muestran las posiciones de los puntos tomados del contorno y en la Figura 3.16c se muestra la función frontera. Si el valor de k disminuye, la función pierde definición y es más difícil reconocer el objeto; en cambio sí k aumenta la función podrá distinguir mejor los cambios en el contorno del objeto, en especial si es muy irregular, la desventaja es que requiere de más espacio en memoria para almacenarse y tiempo de ejecución para calcularse. En este trabajo se toman 64 puntos para representar la función frontera. En la Figura 3.21 se encuentra el diagrama de flujo para calcular la función frontera y en el anexo A se encuentra el código de Matlab para simulación.

Implementación en FPGA

La imagen binaria se obtiene a partir de la información almacenada en Memoria_Grises, a la cual se calcula el histograma, el valor de umbral y el resultado se guarda en Memoria_Binaria1 y Memoria_Binaria2, estas memorias contienen 307,200 localidades de 1 bit de longitud cada

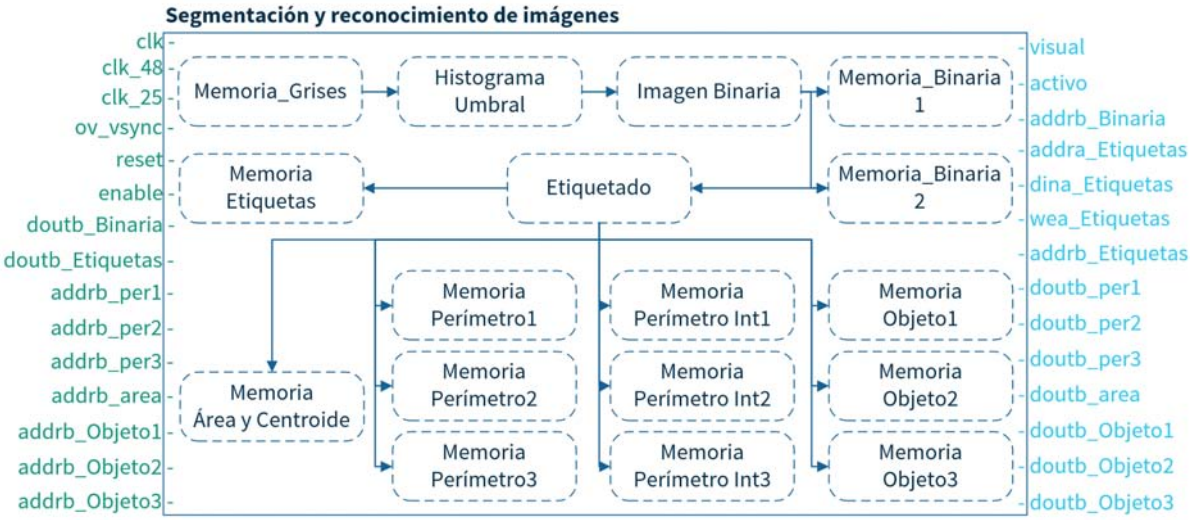


Figura 3.22 Diagrama de bloques del módulo segmentación y reconocimiento de imágenes.

una. En el bloque Etiquetado se implementa el algoritmo de la Figura 3.17, el de la BOF y el reconocimiento de figuras por medio del conteo de los puntos máximos y mínimos. Este bloque requiere de una imagen binaria, por eso toma los datos de la Memoria_Binaria2, a su vez también produce la matriz de etiquetas y esta se guarda en la Memoria Etiquetas que es una BRAM de 307,200 localidades de 4 bits de longitud.

El algoritmo de Etiquetas igualmente produce como salida el contorno externo e interno (en caso de perforaciones en la región) de cada región en forma de un arreglo de números que se componen de la coordenada X e Y; puesto que el valor máximo que pueden tomar X o Y es de 479 o 639, entonces se necesitaran 10 bits para representar cada una de estas variables. Estas variables se concatenan, XY, para formar un solo registro de 20 bits y pueda ser almacenado en la "Memoria Perímetro#" o "Memoria Perímetro Int#" según sea contorno externo o interno; la profundidad de las memorias debe ser especificado para tener un límite en cuanto al número de coordenadas del contorno que se pueden almacenar, para el caso del límite externa las memorias tienen 2048 localidades y para el interno tienen 512 localidades. Por tanto, si el contorno de las regiones supera el máximo de localidades que se pueden almacenar en memoria, se presentarán datos incorrectos al calcular la BOF.

El algoritmo de Etiquetado también proporciona como salidas el área y las coordenadas del centroide de cada una de las regiones; estos valores se almacenan en la Memoria Área y centroide. El área de cada región se almacena en las primeras localidades, una región por localidad, hasta haber guardado todas las regiones; después se almacena el centroide de cada

región, primero la coordenada X y luego la Y, le siguen el número de elementos en el contorno, una localidad por región. Por último se almacena un número en cada localidad que indica la forma de cada región; 1 si es un círculo, 2 si es un triángulo, 3 si es un trapecio, 4 si es un cuadrado y 6 si la figura tiene una forma diferente a las anteriores. Ya por último el algoritmo de la BOF toma los puntos del contorno y el valor del área de las respectivas memorias, obtiene la función frontera y el resultado se guarda en las Memoria Objeto1, Memoria Objeto2 o Memoria Objeto3, según corresponda.

El algoritmo de la BOF requiere de operaciones matemáticas como la división y la raíz cuadrada las cuales no están definidas en el lenguaje de VHDL, por lo que para su implementación se tuvo que elaborar las funciones que realizan estas operaciones; para el caso de la división se utilizó el "restoring division algorithm" y para la raíz cuadrada el "non-restoring square root algorithm" [61] [62]

3.3 Ensamble de piezas

Todo el sistema de visión tiene como objetivo realizar el ensamble de dos piezas muy simples y para esto se han definido tres pasos que debe cumplir el sistema para lograrlo. El primero es identificar las piezas a ensamblar, el segundo es juntar la pieza con la contraparte y el último es asegurar que ambas piezas tengan la misma orientación. La primera tarea, permite al sistema diferenciar de entre diversas formas de objetos aquellos que sean semejantes para realizar el ensamble entre estos, por ejemplo, distinguir la forma de un círculo de la de un cuadrado. Es aquí donde la BOF aporta la información necesaria para hacer la distinción. En particular se usan los puntos máximos y mínimos locales para poder diferenciar una forma de otra, es decir, en el caso de un cuadrado este cuenta con 4 máximos y mínimos a diferencia de un triángulo donde solo hay 3. Por tanto, solo hay que buscar entre los objetos aquellos que tengan el mismo valor en ese parámetro. La segunda tarea se refiere a ubicar la posición tanto de la pieza como de la contraparte y definir cual se ensambla sobre la otra. Esto se lleva a cabo por medio del cálculo del centroide de la pieza que nos permite saber la posición. Normalmente es la pieza de menor tamaño la que se une con la de mayor tamaño, así pues al comparar el valor del área se sabe cuál de los dos objetos es la pieza y cual su contraparte; donde la pieza se ensambla en la contraparte, ver Figura 3.23. Por último, las piezas pueden estar en el área de trabajo no solo en posiciones aleatorias, sino también en direcciones aleatorias, lo cual implica por ejemplo que la figura de un trapecio solo tiene una forma de hacer el ensamble y esta es que cada lado de la

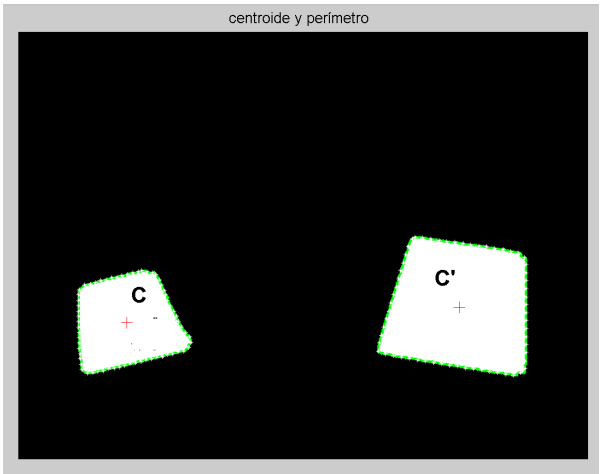


Figura 3.23 Una parte del ensamble consiste en hacer coincidir el centroide de ambas piezas, $C = C'$.

pieza coincida con el lado correcto de la contraparte. Para ello también se utiliza la BOF, con la cual podemos seleccionar un punto característico en la función de la pieza y hacerlo coincidir con el mismo punto pero en la función de la contraparte, logrando hacer que coincidan las direcciones de la pieza y la contraparte en el ensamble, ver Figura 3.24.

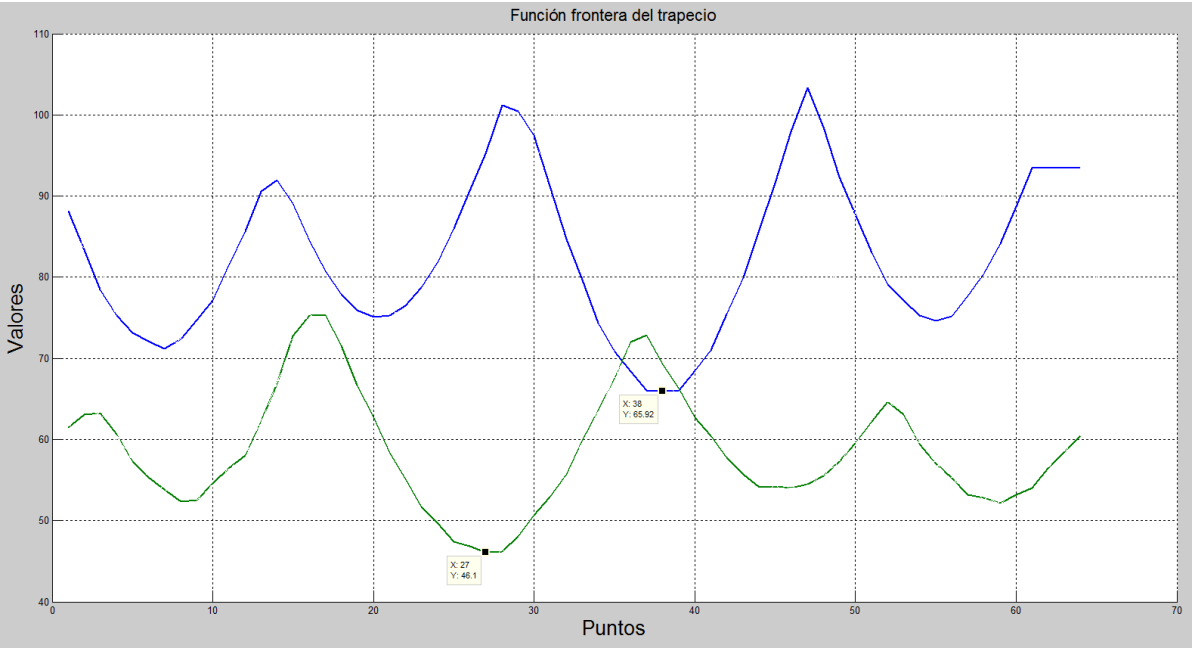


Figura 3.24 Por medio de la BOF podemos distinguir una figura de otra. Además al alinear puntos en común en ambas funciones aseguramos la misma orientación de las piezas en el ensamble.

3.3.1 Labot Pro 5

El sistema de visión descrito anteriormente busca satisfacer la necesidad de realizar ensambles de piezas en ambientes no estructurados; y como medio de actuación para el sistema se utiliza un brazo robótico que se integra en una celda de manufactura experimental. Este brazo robótico o manipulador antropomórfico semi-industrial, modelo Labot Pro 5 fue fabricado por la empresa mexicana *Robótica CRYA* y diseñado especialmente para propósitos de enseñanza. El manipulador Labot Pro 5 cuenta con 5 grados de libertad, base, hombro, codo, muñeca, mano y efector final (pinza); es manejado con un software propio del fabricante que utiliza control punto a punto; y se comunica con la PC por medio de un puerto serial [63]. El control punto a punto quiere decir que para trasladarse a cualquier posición, el robot no toma en cuenta la trayectoria, puesto que lo único que importa es alcanzar el punto en cuestión; por tanto, si se requiere que el robot cubra cierta trayectoria, es necesario hacer que vaya a tantos puntos sobre la trayectoria como sea posible.

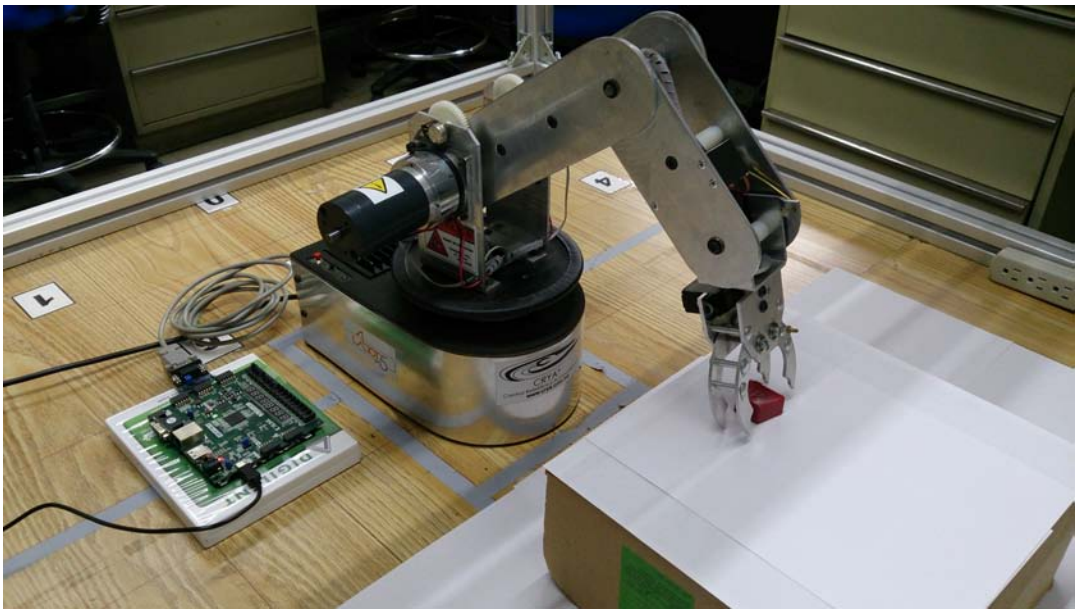


Figura 3.25 Manipulador robótico antropomórfico Labot Pro 5.

De acuerdo con la hoja de datos del fabricante, para suministrar comandos al manipulador es necesario enviarle una cadena de caracteres formada por la letra mayúscula del motor que se quiere mover, según la relación de la Tabla 3.2, y un número entero entre 500 y 2500, equivalente a una posición dentro del rango de movimiento del motor. Por ejemplo, si se quiere mover el motor de la articulación del hombro a la mitad de su rango de posiciones, se le enviará

Articulación	Motor
Base	A
Hombro	B
Codo	C
Muñeca	D
Mano	E
Pinza	F

Tabla 3.2 Asignación de letras para transmisión de comandos al manipulador.

la cadena de caracteres $B1500$ [63]. Resulta importante mencionar que, aunque el cambio mínimo dentro del rango de posiciones de los motores para el hombro y el codo es una unidad, en la práctica se observó que los motores solamente cambian de posición cada 20 unidades.

3.3.2 Espacio de trabajo

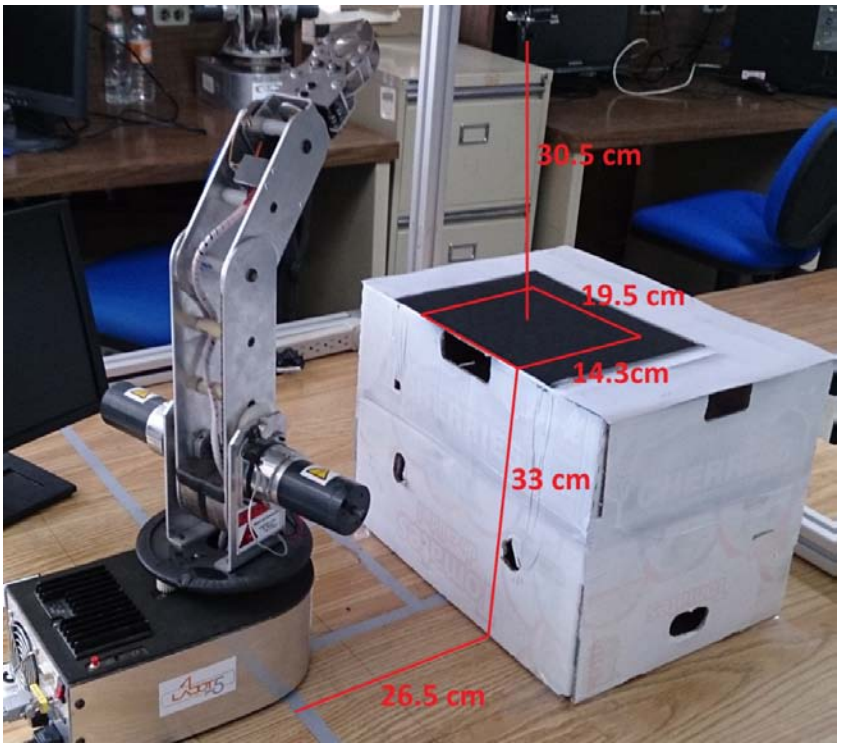


Figura 3.26 Espacio de trabajo del sistema, limitado al área de visión de la cámara.

Dos aspectos clave a considerar cuando se opera un manipulador robótico son la dirección de rotación de los ejes y su espacio de trabajo. Conocer a fondo ambos es necesario para evitar

accidentes, tanto para el propio robot como para quien y lo que lo rodea. Al volumen total que abarca el órgano o efector final cuando el robot ejecuta todos los movimientos posibles, se le conoce como espacio de trabajo. Es prudente tener bien entendida la dirección de rotación y los valores reales de los rangos de los ejes del manipulador, ya que los que provee el fabricante resultaron erróneos, ver Tabla 3.6.

El espacio de trabajo del sistema está limitado en función de dos aspectos:

- El primero es el rango de movimientos del robot en los motores B y C u hombro y codo ya que estos no alcanza a posicionar la pinza cerca de la base del manipulador; esto no se debe a algún mecanismo de seguridad para no dañarse a sí mismo, sino que al enviar una orden mayor a 2250 en la articulación del hombro o codo, el brazo entra en una zona de inestabilidad donde las posiciones y el comportamiento son erráticos e incluso ha llegado a colisionar con la mesa de trabajo.

- El segundo tiene que ver con el área de visión de la cámara. Con el fin de evitar la intromisión del brazo en las imágenes, la cámara está situada de tal forma que el robot no aparezca en escena mientras se realiza la captura de imágenes. El fin de no captar el brazo en las imágenes es para reducir el procesamiento realizado en el FPGA, ya que en caso contrario, tendrían que aplicarse filtros para ignorar el brazo.

Por estas razones, se buscó la posición en la que el robot, sin caer en la zona de inestabilidad, pudiera satisfacer el área de visión de la cámara. Por tanto, fue necesario improvisar una plataforma sobre la mesa de trabajo, para cumplir con ambas limitaciones; resultando en un área de trabajo rectangular de 19.5x14.3 cm, localizada a 26.5 cm de la base del robot, a 33 cm sobre la mesa de trabajo y a 30.5 cm abajo de la cámara, ver Figura 3.26. Esta área de trabajo fue dividida en 81 sectores que representan la relación entre el centroide del objeto y los comandos del manipulador para llegar a esa posición, es decir si tengo la posición (x_1, y_1) que representa el centroide del objeto 1, entonces los comandos A1690, B2150, C2150 y D1600 del manipulador lo colocan al alcance del centroide del objeto1. Para encontrar los 81 sectores se dividió la imagen en 9 partes iguales tanto horizontal como verticalmente, después se buscaron los comandos del manipulador para llegar a las cuatro esquinas del área de trabajo. Los sectores están designados por un número y letra, que representan su posición sobre el eje X,Y; la primera posición A1 está en la esquina del origen de la imagen. Los comandos de las esquinas A1 y A9 solo difieren en los valores de A, lo cual es normal ya que el motor A corresponde a la base del manipulador; lo mismo sucede con los comandos de las posiciones I1 e I9. Entonces el rango de valores para la fila A se tomó como la diferencia entre A1 y A9, y para encontrar los valores

de los sectores restantes de A se dividió el rango de A en partes iguales. De forma análoga se encuentran los valores de los comandos para el resto de los sectores, ver Tabla 3.3.

Por otra parte en la Figura 3.27 y la Tabla 3.4, se observan las piezas a utilizar para el desarrollo de las pruebas; estas fueron tomadas de un juguete, el cual requiere que los niños reconozcan las formas básicas de los objetos y las inserten en su lugar correspondiente. De forma análoga, nuestro sistema tiene que reconocer la forma de las piezas y colocarlas en su contraparte que tiene la misma forma pero de mayor tamaño. Estas piezas son indicadas para las pruebas por su tamaño y peso que son fáciles de sujetar por la pinza del manipulador, además que por su morfología presenta una signatura fácil de reconocer al aplicar la BOF. Las piezas se pintaron de color blanco por dos razones: para tener un mejor contraste respecto al fondo negro que asegure un histograma de alto contraste, ver Figura 2.10; asimismo esta configuración de colores se escogió de manera que las sombras producidas por los mismos objetos no deformaran la morfología del objeto en las imágenes binarias. Otra forma de explicar lo anterior es que si la configuración fuera objeto negro sobre fondo blanco, al momento de convertir la imagen a binaria, las sombras pasarían a ser parte del objeto puesto que serían detectadas como valores oscuros en el histograma de la imagen y esto provocaría resultados incorrectos a la hora de calcular el contorno y el centroide. Otro arreglo realizado al espacio de trabajo es el cubrir con una cartulina el área situada por arriba de la cámara, con el fin de atenuar la incidencia de luz sobre el espacio de trabajo y reducir el reflejo del fondo negro hacia la cámara, lo que a su vez disminuye el ruido en la adquisición de las imágenes.

Figura 3.27 Piezas de plástico y sus contrapartes a utilizar en el ensamble.

Centroide en píxeles	**X**	0	71	142	213	284	355	426	497	568
		-70	-141	-212	-283	-354	-425	-496	-567	-639
Y	**Sector**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0 -52	**A**	**A1690** **B2150** **C2150** **D1600**	A1735 B2150 C2150 D1600	A1781 B2150 C2150 D1600	A1826 B2150 C2150 D1600	A1872 B2150 C2150 D1600	A1918 B2150 C2150 D1600	A1963 B2150 C2150 D1600	A2009 B2150 C2150 D1600	**A2055** **B2150** **C2150** **D1600**
53 -105	**B**	A1680 B2095 C2156 D1450	A1728 B2095 C2156 D1450	A1775 B2095 C2156 D1450	A1822 B2095 C2156 D1450	A1870 B2095 C2156 D1450	A1917 B2095 C2156 D1450	A1964 B2095 C2156 D1450	A2012 B2095 C2156 D1450	A2059 B2095 C2156 D1450
106 -158	**C**	A1671 B2041 C2162 D1400	A1720 B2041 C2162 D1400	A1768 B2041 C2162 D1400	A1816 B2041 C2162 D1400	A1865 B2041 C2162 D1400	A1913 B2041 C2162 D1400	A1961 B2041 C2162 D1400	A2010 B2041 C2162 D1400	A2058 B2041 C2162 D1400
159 -212	**D**	A1662 B1987 C2169 D1350	A1712 B1987 C2169 D1350	A1763 B1987 C2169 D1350	A1814 B1987 C2169 D1350	A1865 B1987 C2169 D1350	A1915 B1987 C2169 D1350	A1966 B1987 C2169 D1350	A2017 B1987 C2169 D1350	A2068 B1987 C2169 D1350
213 -265	**E**	A1652 B1933 C2175 D1350	A1705 B1933 C2175 D1350	A1758 B1933 C2175 D1350	A1810 B1933 C2175 D1350	A1863 B1933 C2175 D1350	A1915 B1933 C2175 D1350	A1968 B1933 C2175 D1350	A2020 B1933 C2175 D1350	A2073 B1933 C2175 D1350
266 -318	**F**	A1643 B1878 C2181 D1250	A1697 B1878 C2181 D1250	A1751 B1878 C2181 D1250	A1805 B1878 C2181 D1250	A1860 B1878 C2181 D1250	A1914 B1878 C2181 D1250	A1968 B1878 C2181 D1250	A2023 B1878 C2181 D1250	A2077 B1878 C2181 D1250
319 -372	**G**	A1634 B1824 C2188 D1100	A1690 B1824 C2188 D1100	A1745 B1824 C2188 D1100	A1801 B1824 C2188 D1100	A1857 B1824 C2188 D1100	A1912 B1824 C2188 D1100	A1968 B1824 C2188 D1100	A2024 B1824 C2188 D1100	A2080 B1824 C2188 D1100
373 -425	**H**	A1624 B1769 C2194 D1350	A1681 B1769 C2194 D1350	A1745 B1769 C2194 D1350	A1801 B1769 C2194 D1350	A1857 B1769 C2194 D1350	A1912 B1769 C2194 D1350	A1968 B1769 C2194 D1350	A2024 B1769 C2194 D1350	A2080 B1769 C2194 D1350
426 -479	**I**	**A1615** **B1715** **C2200** **D1300**	A1674 B1715 C2200 D1300	A1733 B1715 C2200 D1300	A1793 B1715 C2200 D1300	A1852 B1715 C2200 D1300	A1911 B1715 C2200 D1300	A1971 B1715 C2200 D1300	A2030 B1715 C2200 D1300	**A2090** **B1715** **C2200** **D1300**

Tabla 3.3 Comandos para el manipulador en función de la posición del centroide del objeto.

Figura	Medidas (cm)
Cuadrado	L = 2.6
Circulo	D = 3.1
Trapecio	B = 3.3 b = 2.3 c,d = 2.1
Circulo2	D = 4

Tabla 3.4 Medidas de las piezas a utilizar para el ensamble

3.4 Interfaces de salida

3.4.1 VGA

El arreglo gráfico de video (VGA) es un protocolo para desplegar gráficos en un monitor CRT o LCD, el protocolo involucra realizar un barrido por la pantalla a la vez que se utilizan dos señales: hsync y vsync, para sincronizar la ubicación en la pantalla donde se debe "pintar" un valor. Un controlador VGA simplemente enciende o apaga las señales horizontal y vertical de manera síncrona de tal forma que el monitor sepa cuando mostrar el siguiente dato. Al principio de una fila, la señal HSYNC se activa por un periodo de tiempo y después se vuelve a desactivar pero en ese momento no se puede mostrar nada en la pantalla, a este periodo de le conoce como, "front porch", pasado este periodo ya se podrá imprimir en pantalla. La señal HSYNC permite imprimir 640 píxeles antes de entrar en un estado llamado: "back porch"; el cual indica que se ha terminado la porción visible de la pantalla, para terminar el back porch, la señal HSYNC se activa y vuelve a empezar el ciclo. Lo mismo sucede con la señal VSYNC, solo que esta representa el número de filas a imprimir antes de que se actualice toda la pantalla [64]. El protocolo VGA se utiliza para desplegar en pantalla la imagen en escala de grises o la imagen binaria adquirida por la cámara, previo al inicio del procesamiento de imágenes.

3.4.2 UART-Labot Pro 5

El manipulador Labot Pro está diseñado para conectarse al puerto serial de una computadora y utilizarse con el programa de Labview; sin embargo, puede utilizarse cualquier otro programa que pueda tener acceso a los puertos seriales del equipo para enviar órdenes al robot, o también desde hardware por medio del protocolo UART. Para establecer correctamente la comunicación con el brazo antropomórfico, la comunicación serial debe estar establecida a una velocidad de 9600 baudios, 8 bits y sin paridad. Para lograr la comunicación con el sistema anfitrión

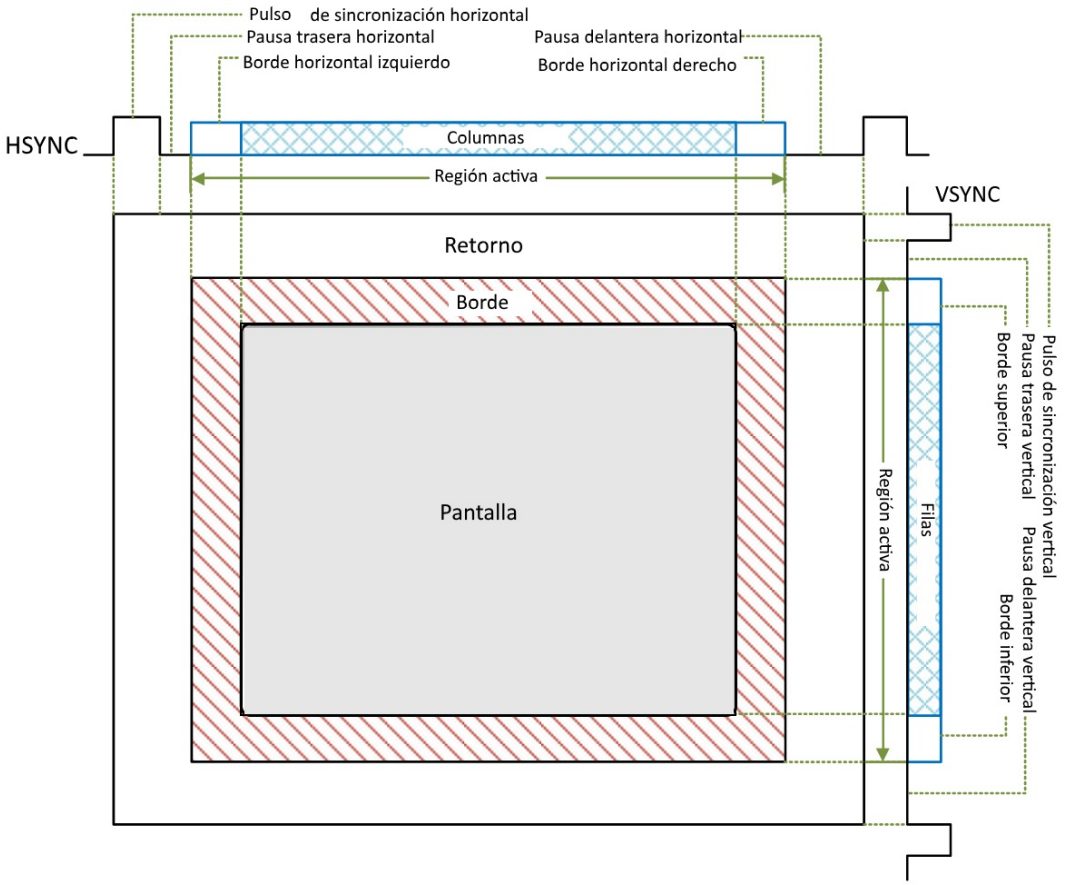


Figura 3.28 Comportamiento de las señales HSYNC y VSYNC [64].

del FPGA, se utilizó un convertidor FTDI232, el cual es un circuito integrado especialmente diseñado para convertir las señales lógicas de RS232 a TTL y viceversa, ver Figura **??**. Este integrado se utiliza principalmente para reducir los niveles de tensión de RS232(±15 V a ±3 V) a 3.3 V. Los datos a enviar desde el FPGA hacia el manipulador consisten en comandos que indican las posiciones que deben tomar cada uno de los actuadores; el formato debe indicar el motor y la posición, por ejemplo: A1200, B1300, C1500, etc; estos comandos pueden tomar alguno dentro del rango mostrado en la Tabla 3.6.

3.4.3 UART-PC

Con el fin de corroborar los resultado obtenidos por el FPGA, se implementó un protocolo de comunicación UART, el cual específicamente envía los resultados del: área, centroide, BOF y la imagen binaria; a la computadora, ver Figura 3.30. Las comunicación se realiza a una

Descripción	Simbolo	Tiempo	Duración
Pixel Clock	t_{clk}	39.7 ns	25.175 MHz
H Sync	t_{hs}	3.813 us	96 Píxeles
H Back Porch	t_{hbp}	1.907 us	48 Píxeles
H Front Porch	t_{hfp}	0.636 us	16 Píxeles
H Addr Video Time	t_{haddr}	25.422 us	640 Píxeles
H L/R Border	t_{hbd}	0 us	0 Píxeles
V Sync	t_{vs}	0.064 ms	2 Filas
V Back Porch	t_{vbp}	1.048 ms	33 Filas
V Front Porch	t_{vfp}	0.318 ms	10 Filas
V Addr Video Time	t_{vaddr}	15.253 ms	480 Filas
V T/B Border	t_{vbd}	0 ms	0 Filas

Tabla 3.5 Tabla de tiempos para el protocolo VGA.

Articulación	**Rango de movimiento del fabricante**		**Rango de movimiento real**	
	Grados	Comandos	Grados	Comandos
A	360°	0500-2500	110°	0750-2550
B	180°	0500-2500	100°	0900-2250
C	180°	0500-2500	125°	0800-2250
D	180°	0500-2500	50°	0900-2150
E	90°	0500-2500	65°	0600-2000
F	-	0500-2500	-	0600-1500

Tabla 3.6 Rangos de movimiento de las articulaciones del manipulador Labot Pro 5.

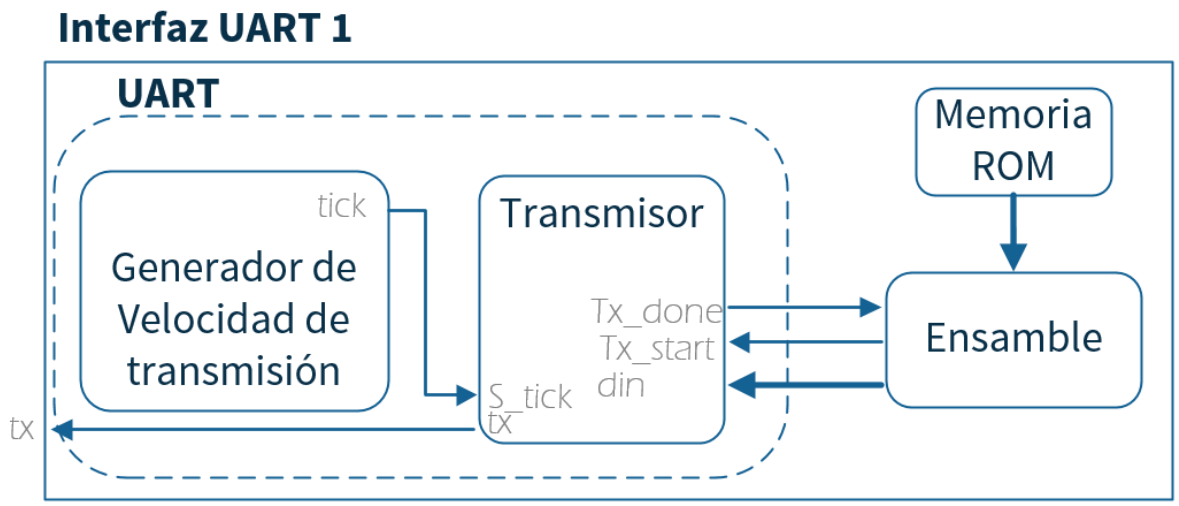


Figura 3.29 Diagrama de bloques del módulo interaz UART1

velocidad de 38400 bps, 8 bits de datos, 1 bit de parada, sin paridad; y se reciben los datos con el programa de Matlab donde se almacenan y grafican para ser interpretados y comparados con los resultados de los mismos algoritmos que en el FPGA pero implementados en Matlab. El protocolo de comunicación funciona de la siguiente manera:

- El bloque Controlador en el FPGA espera hasta que un dato sea recibido por el pin RX.
- Si el dato recibido es "A" se procede con el envío del contenido de Memoria_Binaria2, es decir los 307200 píxeles de la imagen binaria.
- Si el dato recibido es "B" se procede con el envío del contenido de Memoria Área y centroide, es decir los valores del área, centroide, perímetro y forma del objeto de cada una de las regiones.
- Si el dato recibido es "C" se procede con el envío del contenido de Memoria Objeto1, que corresponde con los valores de la función frontera de la región 1.
- Si el dato recibido es "D" se procede con el envío del contenido de Memoria Objeto2, que almacena los datos de la función frontera de la región 2.
- Si el dato recibido es "E" se procede con el envío del contenido de Memoria Objeto3, es decir los 64 puntos de la función frontera de la región 3.
- Si el dato recibido es diferente a las letras antes mencionadas, se descarta y el sistema queda en espera de recibir otro dato.

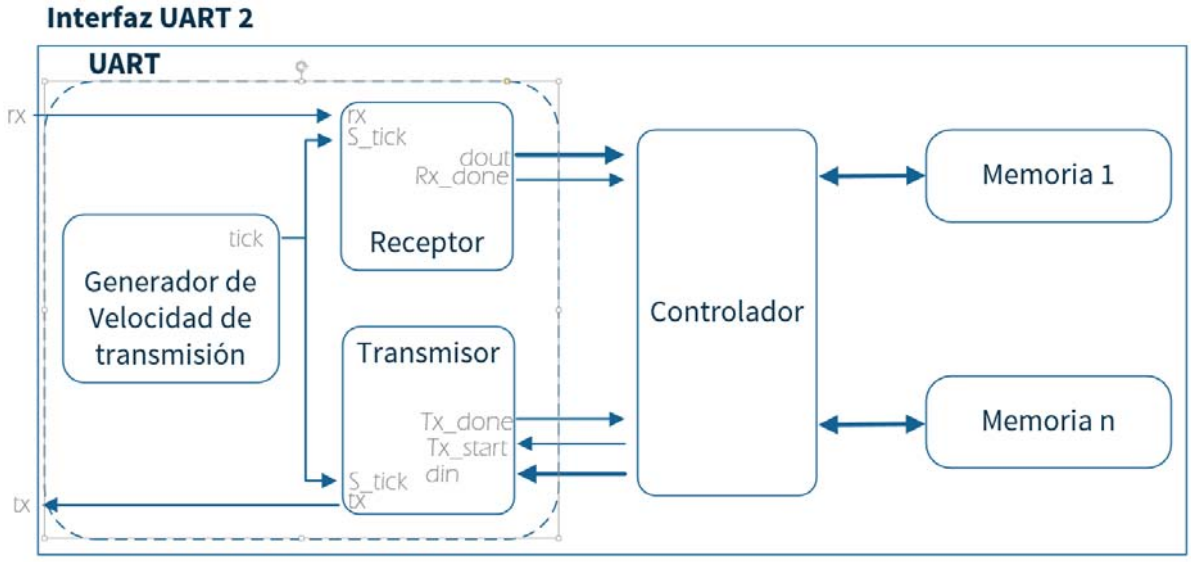


Figura 3.30 Diagrama de bloques del módulo interfaz UART2.

Capítulo 4

Pruebas y resultados experimentales

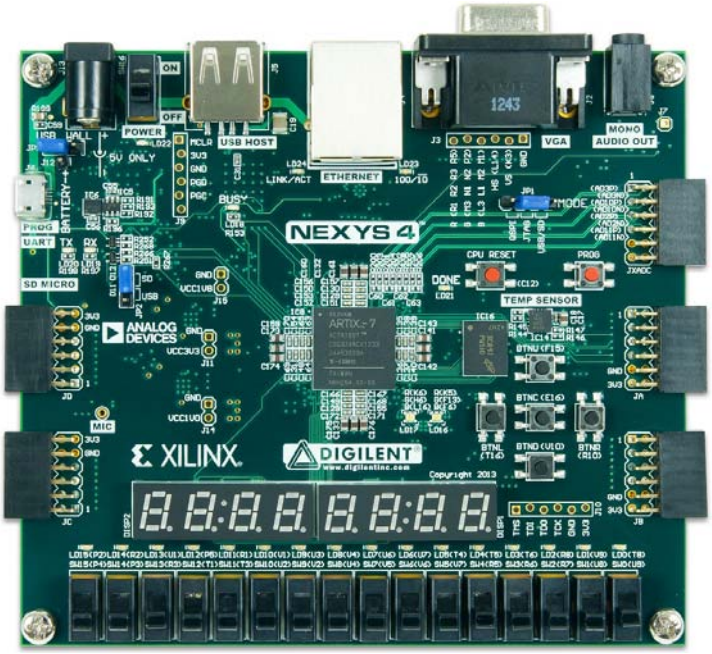


Figura 4.1 Tarjeta de desarrollo Nexys 4.

La tarjeta de desarrollo Nexys4 se utiliza en este trabajo para implementar todo el sistema de adquisición y procesamiento de imágenes, así como el control del manipulador Labot Pro 5. La tarjeta Nexys4 es una plataforma de desarrollo para circuitos digitales basada en el FPGA Artix-7 (XC7A100T-1CSG324C) del fabricante Xilinx. El ambiente de programación utilizado fue el ISE 14.6 de Xilinx y se usaron herramientas como Isim y ChipScope para la simulación funcional de los módulos y para el monitoreo de las señales en tiempo real [65]. El diseño del

sistema se implemento con la metodología de comportamiento bajo el lenguaje VHDL y se utilizó el programa Adept de Digilent para la programación del FPGA [66].

En la Tabla 4.1 se muestran las terminales de E/S que se utilizaron en el FPGA, de éstas, 18 fueron usadas por la cámara OV7670, 1 para la comunicación UART con el manipulador, 14 de la interfaz VGA para mostrar las imágenes adquiridas en el monitor, 2 para la interfaz UART con la computadora, 6 indicadores de estado (leds), 7 interruptores y una para la señal de reloj; en total 49 pines de E/S todos con un voltaje de 3.3 V. En el caso de los interruptores se utilizó un módulo eliminador del rebote de la señal, basado en un contador que deshabilita la entrada por un tiempo de 40 ms.

El procedimiento para que funcione el sistema es: programar el FPGA dos veces, en el monitor se desplegaran las imágenes en escala de grises adquiridas por la cámara. Activar el interruptor SW15 para mostrar las imágenes binarias y habilitar el interruptor SW14 para detener la adquisición de imágenes, verificar que en el monitor se observen los objetos en escena y ni un solo píxel de ruido; en caso de presentarse ruido se deberá desactivar y activar el interruptor SW14 hasta que se vean los objetos sin ruido alguno. Un punto blanco en el monitor que no pertenezca a algún objeto es ruido, ocasionado por la reflexión de la luz, por alguna basura en la escena o por algún defecto presentado en la cámara. Posteriormente, presionar el botón BTNC para llevar acabo el procesamiento de imágenes, el indicador LD17 G1 estará activo mientras el proceso esté en marcha, apenas un parpadeo, al final del procesamiento se enciende el indicador LD0. Después, presionar el botón BTNU para que el manipulador realice el ensamble de las piezas, el indicador LD16 B2 estará activo mientras el ensamble no se haya concluido y al final se enciende el indicador LD1. Para finalizar activar SW0 para llevar al manipulador a su posición de inicio y dar por terminado el funcionamiento del sistema.

4.1 Precisión del manipulador

Con el fin de comprobar la precisión del manipulador proporcionada por el fabricante se realizó esta prueba, la cual consiste en encender el manipulador y partiendo de su posición de inicio llevarlo a un punto cualquiera dentro del espacio de trabajo del sistema, este punto será P1, marcar la posición de P1 y después volver a la posición de inicio; se repitió esta tarea cinco veces, ver Figura 4.2. Después, se apagó el manipulador, se dejó pasar un tiempo y se volvió a repetir la prueba entre el punto de inicio y P1. Posteriormente se realizó la prueba con otro punto P2 y un tercer punto P3. Para saber si se llega al punto seleccionado, se colocó en el

Pin FPGA	Señal	Pin Dispositivo	Descripción
E3	clk	-	Señal de Reloj del FPGA
U9	Paro_Emergencia	SW0	Pone al brazo en posición de inicio.
R3	background	SW13	0, si los objetos son blancos.
P3	swhold	SW14	Detiene la adquisición de imágenes.
P4	sw_color	SW15	0, imagen binaria en monitor; 1, imagen en escala de grises.
T8	fin_Etiq	LD0	Señala el termino del procesamiento de imágenes.
V9	fin_Ensamble	LD1	Señala el termino del ensamble de las piezas.
F13	Etiq_On	LD17 G1	Señala que el algoritmo de etiquetado esta en proceso.
L16	Ensamble_On	LD16 B2	Señala que el ensamble de piezas esta en proceso.
K6	Emergencia_On	LD17 R1	Señala que el manipulador va a la posición de inicio.
H6	led_tx_brazo	LD16 G2	Señala el envío de comandos al manipulador.
C12	btnCpuReset	CPU_RESET	Reinicio de todo el sistema.
E16	IniciaEtiq	BTNC	Iniciar el procesamiento de imágenes.
F15	IniciaEnsamble	BTNU	Inicia el proceso de ensamblado
E17	tx_brazo	JA4	Salida de Interfaz UART2
C4	rx_pin	J6	Entrada de Interfaz UART1
D4	tx_pin	J6	Salida de Interfaz UART1
A3	vgaRed(0)	RED0	Señal de salida Interfaz VGA
B4	vgaRed(1)	RED1	Señal de salida Interfaz VGA
C5	vgaRed(2)	RED2	Señal de salida Interfaz VGA
A4	vgaRed(3)	RED3	Señal de salida Interfaz VGA
B7	vgaBlue(0)	BLU0	Señal de salida Interfaz VGA
C7	vgaBlue(1)	BLU1	Señal de salida Interfaz VGA
D7	vgaBlue(2)	BLU2	Señal de salida Interfaz VGA
D8	vgaBlue(3)	BLU3	Señal de salida Interfaz VGA
C6	vgaGreen(0)	GRN0	Señal de salida Interfaz VGA
A5	vgaGreen(1)	GRN1	Señal de salida Interfaz VGA
B6	vgaGreen(2)	GRN2	Señal de salida Interfaz VGA
A6	vgaGreen(3)	GRN3	Señal de salida Interfaz VGA
B11	Hsync	HS	Señal de salida Interfaz VGA
B12	Vsync	VS	Señal de salida Interfaz VGA
K2	ov_pwdn	JC1-PWDN	Modo de apagado de la cámara
E7	ov_d(0)	JC2-D0	Componente de salida de la cámara
J3	ov_d(2)	JC3-D2	Componente de salida de la cámara
J4	ov_d(4)	JC4-D4	Componente de salida de la cámara
K1	ov_reset	JC7-RESET	Reinicio de los registros de la cámara
E6	ov_d(1)	JC8-D1	Componente de salida de la cámara
J2	ov_d(3)	JC9-D3	Componente de salida de la cámara
G6	ov_d(5)	JC10-D5	Componente de salida de la cámara
H4	ov_d(6)	JD1-D6	Componente de salida de la cámara
H1	ov_xclk	JD2-XCLK	Entrada de reloj para la cámara
G1	ov_href	JD3-HREF	Salida HREF de la cámara
G3	ov_siod	JD4-SIOD	E/S de la interfaz SCCB
H2	ov_d(7)	JD7-D7	Componente de salida de la cámara
G4	ov_pclk	JD8-PCLK	Salida de reloj de la cámara
G2	ov_vsync	JD9-VSYNC	Salida VSYNC de la cámara
F3	ov_sioc	SIOC	Señal de reloj de la interfaz SCCB

Tabla 4.1 Pines de E/S utilizados en el FPGA

Recursos lógicos	Usados	Disponibles	Porcentaje
Bloques lógicos	4735	126,800	3 %
LUTs	10,411	63,400	16 %
IOBs	47	210	22%
36Kb BRAM	94	135	69%
18Kb BRAM	19	270	7%

Tabla 4.2 Recursos lógicos del FPGA utilizados por el sistema.

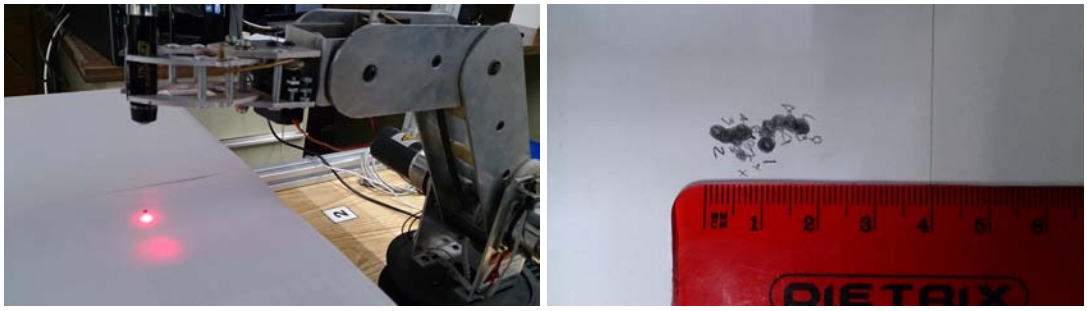


Figura 4.2 Método para encontrar la precisión del manipulador, con base en las marcas dejadas por un apuntador.

efector final un apuntador láser que señala sobre una hoja de papel blanca para realizar la marca. Para obtener los resultados se dibuja un círculo que envuelva a todos los puntos de cada prueba; el círculo de color azul, representa las muestras que se tomaron antes de apagar y encender el manipulador, el círculo de color verde es el conjunto de muestras que se tomaron después de volver a encender el manipulador, ver Figura 4.3. De los resultados se puede deducir que al apagar y encender el manipulador provoca que las posiciones de los puntos P se desplacen al menos 0.9 cm; y que mientras el manipulador no se vuelva a encender el rango de precisión esta en al menos 0.6 cm. En general podemos decir que la precisión del manipulador en cualquier punto del espacio de trabajo es en promedio de 1.4 cm.

4.2 Adquisición de imágenes

En la Tabla 4.3, se presentan los valores utilizados en la cámara para la configuración de esta, formato RGB444 con un reloj interno de 12 MHz a 15 FPS; los registros no mencionados se dejan con el valor predefinido de fábrica. Durante las pruebas se observó que de manera esporádica se presentan puntos blancos en el monitor los cuales desaparecen con el tiempo, éstos puntos son un problema de la cámara ya que tiene un comportamiento aleatorio en toda la

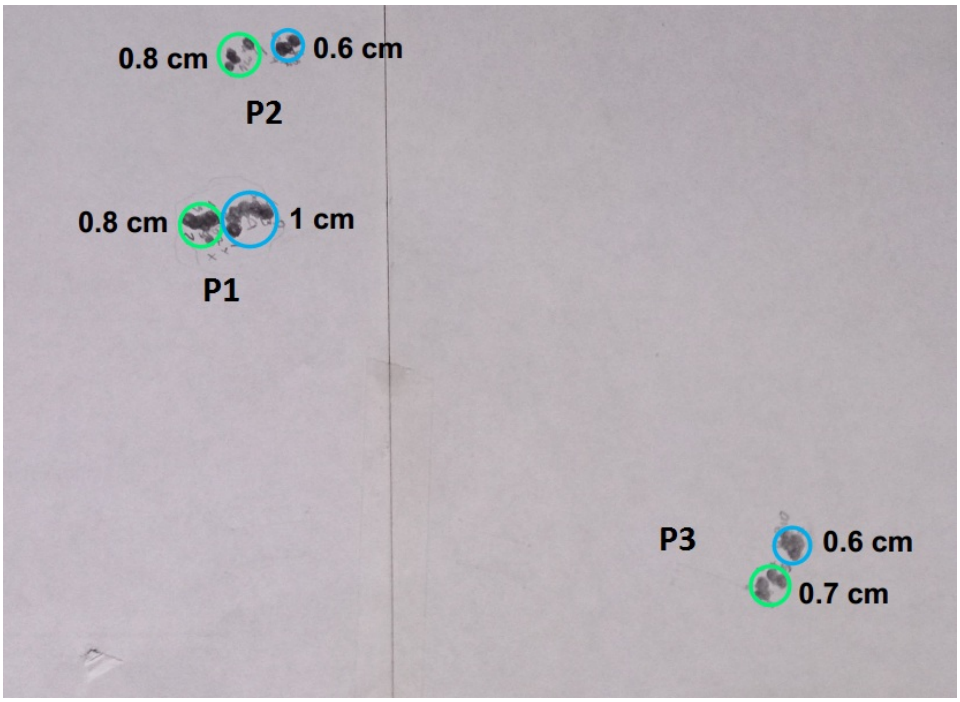


Figura 4.3 Resultados de la precisión del manipulador. Cada color representa el conjunto de muestras que se realizaron antes de apagar el manipulador.

imagen. En la Figura 4.5 se presenta un ejemplo de la adquisición de imágenes con la cámara y de la interfaz VGA con el monitor.

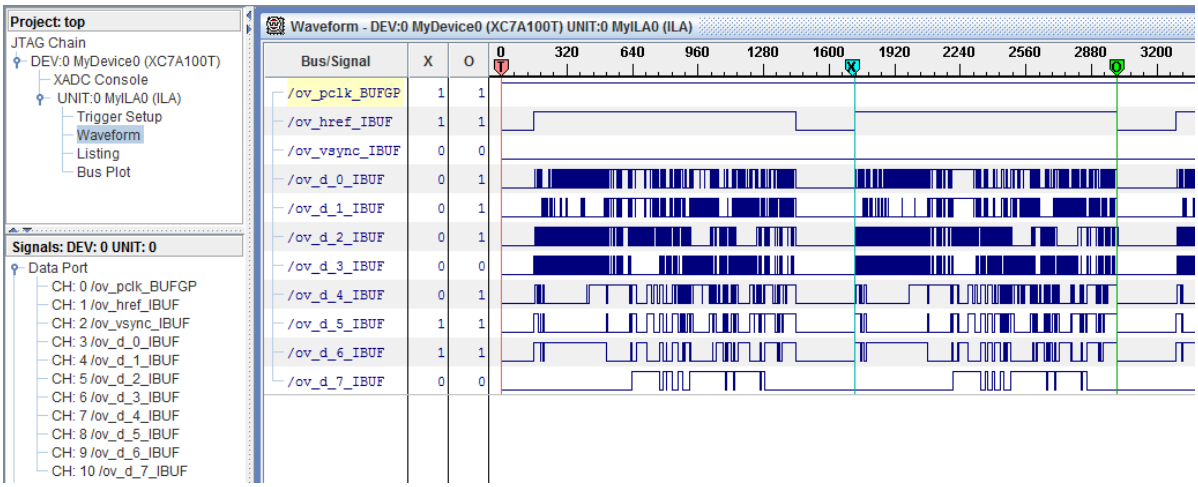


Figura 4.4 Ejemplo del envío de una imagen al FPGA. Se puede apreciar la señal HREF que marca el fin de una fila en la imagen

Dirección (HEX)	**Valor (HEX)**	**Descripción**
12	80	Restablece todos registros al valor preestablecido.
12	04	Selecciona el formato RGB como salida
11	00	Pre-scalar para el reloj interno
0C	00	Configuración del tamaño de imagen
3E	00	Configuración del tamaño de imagen
8C	02	Selecciona el formato de salida RGB444
40	90	Desactivar el formato de salida RGB555
3A	04	Desactivar la imagen negativa
14	38	Control de ganancia automática
4F	99	Coeficiente 1 de la matriz de color
50	99	Coeficiente 2 de la matriz de color
51	00	Coeficiente 3 de la matriz de color
52	28	Coeficiente 4 de la matriz de color
53	71	Coeficiente 5 de la matriz de color
54	99	Coeficiente 6 de la matriz de color
58	9E	Signo de la matriz de color
17	16	8 MSB de la señal HSTAR
18	04	8 MSB de la señal HSTOP
32	80	3 LSB de las señales HSTAR y HSTOP
19	02	8 MSB de la señal VSTRT
1A	7B	8 MSB de la señal VSTOP
03	06	2 LSB de las señales VSTRT y VSTOP
69	00	Ganancia para los canales R, G, B
74	10	Control de ganancia digital
B1	0C	Automatic Black Level Calibration (ABLC)
76	C1	Corrección de pixeles negros y blancos
66	01	Activar ajuste de lente
13	E7	Activa AGC, AWB y AEC
6b	0A	Configuración del PLL

Tabla 4.3 Valores para configurar los registros de la cámara

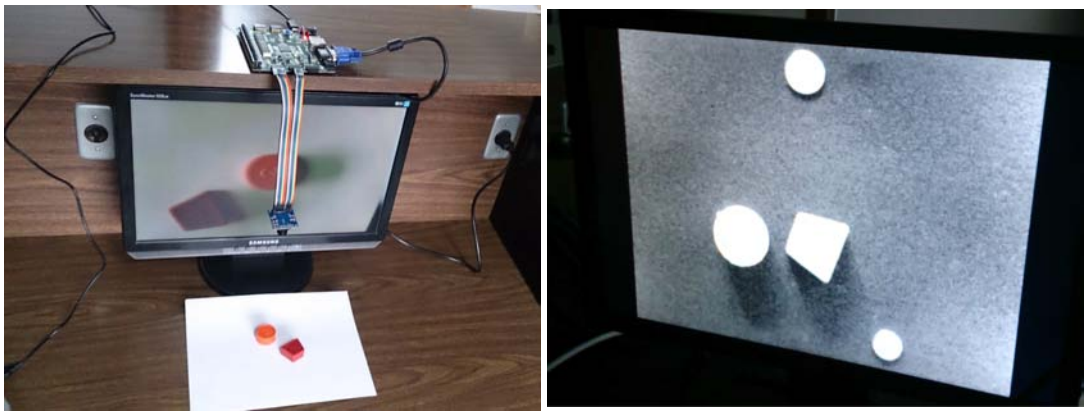


Figura 4.5 Ejemplo del funcionamiento de la adquisición de imágenes y del protocolo de comunicación VGA. A la izquierda se muestra una imagen sin procesamiento alguno; y a la derecha se despliega otra imagen en escala de grises.

4.3 Relación entre píxeles y milímetros

Con esta prueba se busca estimar el factor de conversión entre píxeles y milímetros necesario para poder conocer las posiciones reales de los objetos en el espacio de trabajo. La prueba consiste en colocar un objeto en un plano paralelo al plano de la cámara, adquirir una imagen, comparar las medidas obtenidas en la imagen con las del objeto real y estimar un factor de conversión. Se sabe que el lente de la cámara produce una curvatura en los bordes de la imagen y para compensar esta curvatura en el cálculo de la relación se toman cinco muestras del mismo objeto en diferentes posiciones de la imagen una en el centro y una en cada una de las esquinas, de tal forma que se encuentre un valor aproximado para toda la imagen. Las medidas a comparar en cada una de las posiciones del objeto son el ancho (A) y largo (L) de este, la relación está dada por: A_r/A_i y L_r/L_i, donde los subíndices r e i representan la medida real o de la imagen, respectivamente. La muestra cuando el objeto está colocado en el centro de la imagen no presenta una gran deformación respecto a las otras cuatro muestras, pero para encontrar el factor de conversión se toma el valor promedio obtenido de las 5 muestras. El objeto utilizado para estas pruebas fue el cuadrado mencionado en la Tabla 3.27. En la Figura 4.6 se presenta la suma de las cinco muestras del objeto, donde se observan las posiciones del cuadrado y también se pueden apreciar unas ligera deformaciones de los objetos que están en las esquinas de la imagen por el efecto de la lente. En la Tabla 4.4, se presentan los resultados de las relaciones para cada una de las posiciones del cuadrado y al final se incluye la relación promedio que es $1mm = 3.638píxeles$.

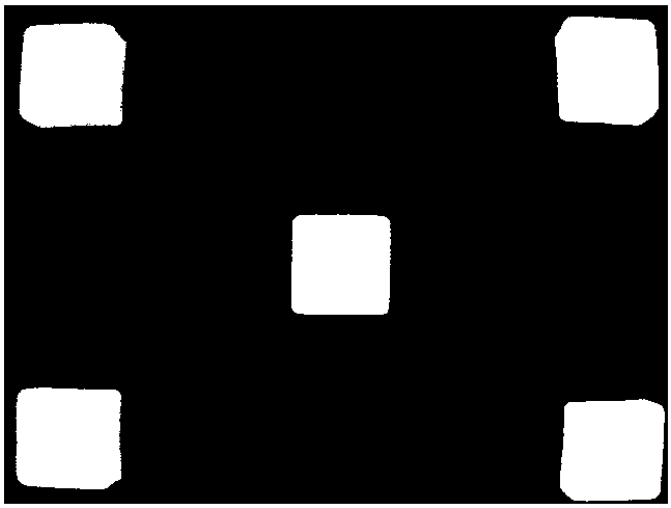


Figura 4.6 Imagen utilizada para obtener la relación entre píxeles y milímetros.

Posición	**Distancia X (Pixeles)**	**Distancia Y (Pixeles)**	**Relación**
Centro	93	94	1 mm = 3.5961
Esq. Inf. Derecha	96	94	1 mm = 3.653
Esq. Inf. Izquierda	99	92	1 mm = 3.673
Esq. Sup. Derecha	93	97	1 mm = 3.653
Esq. Sup Izquierda	93	95	1 mm = 3.628
Promedio	94.8	94.4	1 mm = 3.637

Tabla 4.4 Relación entre píxeles y milímetros utilizada en el sistema

4.4 Ensamble de piezas

El ensamble de piezas se realiza después del procesamiento de imágenes y como resultado del cálculo del contorno, área, centroide y la BOF de cada región en la imagen, sin estos parámetros no se podría realizar el ensamble. Este se realizó al principio solo con dos piezas pero se puede extender el funcionamiento para realizarse con más piezas en función del almacenamiento del FPGA.

Para realizar el acople de las piezas el módulo ensamble, ver Figura 3.1, inicia por colocar al manipulador en una posición definida como casa, que sirve como punto de partida para moverse a las demás posiciones. Lo anterior se debe a que el manipulador presenta problemas de precisión al tratar de moverse a posiciones cercanas entre sí, por eso la posición de casa se encuentra separada del área de visión de la cámara. En función del número de máximos y mínimo en la BOF, se asigna una etiqueta que diferencia entre la forma de las regiones. Se lee el valor de esta etiqueta y se compara con las etiquetas de las otras regiones para conocer si hay otra figura con la misma forma en la región de interés. Si se encuentran figuras con la misma forma se continua con el ensamble, se leen los valores de las áreas de ambas figuras y se comparan para determinar la pieza con el área menor y la pieza con el área mayor; ya que en este caso la pieza con el área menor se introducirá en la pieza con área mayor, de esta forma se identifica la pieza y su contraparte.

Después se lee el centroide de la pieza con menor área y se localiza a que sector pertenece de acuerdo con la Tabla 3.3. El sector está definido por una letra y un número los cuales a su vez definen los comandos a enviar al manipulador. Al ser constantes estos comandos se almacenan en una memoria ROM, cada localidad de memoria almacena un carácter y para llevar al manipulador a cualquier punto se necesitan 4 cadenas de 5 caracteres cada una. El número y letra que determinan el sector en la Tabla 3.3, también determinan la localidad de memoria en donde se encuentran las 4 cadenas para llegar a ese sector. Al llegar la pinza del manipulador a dicho sector se recoge la pieza y se regresa a la posición de casa. Posteriormente, se lee el centroide de la pieza con mayor área y se localiza el sector al que pertenece para llevar al manipulador a este sector, donde se abre la pinza dejando caer la pieza sobre su contraparte. Para terminar, el manipulador regresa a su posición de casa y espera a que se inicie el proceso nuevamente. En la Tabla 4.5, se muestran los resultados de algunas pruebas realizadas con las piezas y sus contrapartes, donde los cálculos obtenidos por el FPGA se enviaron a Matlab para su registro, en estos resultados se puede comparar el centroide obtenido por el FPGA en píxeles y de ahí estimar el centroide en centímetros; la relación entre milímetros y píxeles

utilizada es de $1mm = 3.638Píxeles$. A propósito, el centroide que sé está calculando para los

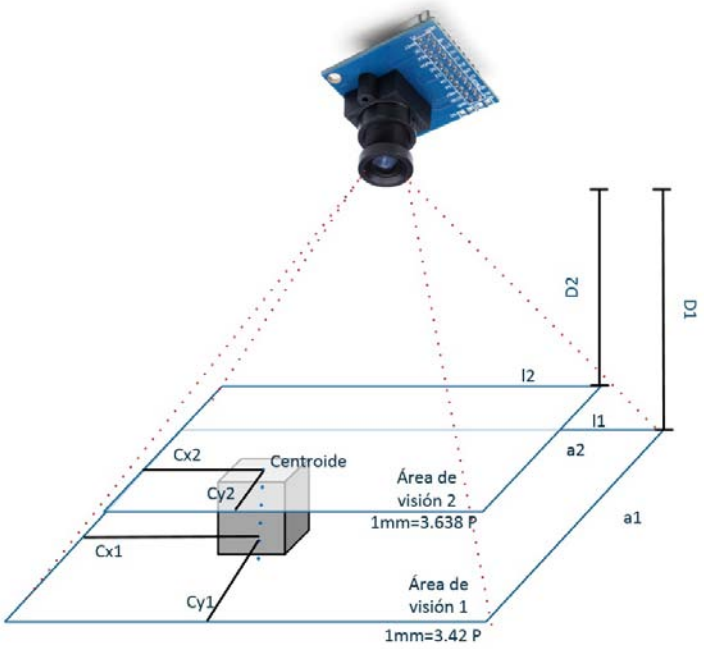


Figura 4.7 Al variar la distancia entre la cámara y el objeto cambia la relación entre milímetros y píxeles.

objetos es a la altura de sus caras, sin embargo las mediciones del centroide real se tomaron a la altura de la base de los objetos y a esta altura la relación entre píxeles y milímetros cambia ligeramente, es decir en la base de los objetos la relación es de $1mm = 3.42Píxeles$, ver Figura 4.7. Esta relación se encontró de manera análoga a la descrita anteriormente, salvo que esta vez se utilizaron unas figuras de foamy en forma de círculo con apenas 2 mm de espesor. Al utilizar la relación de 3.42 para calcular el centroide de los objetos los resultados obtenidos por el FPGA difieren de los reales a lo sumo en 0.6 cm.

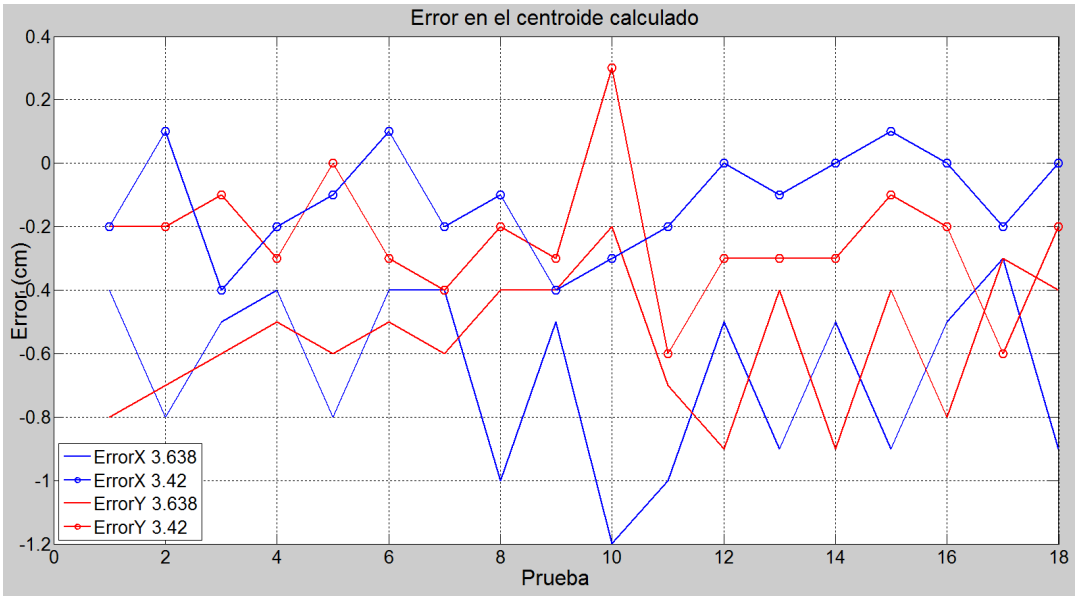


Figura 4.8 Error en el cálculo del centroide entre el FPGA y el valor real.

Prueba	Piezas	Centroide de la pieza		Rel. 1mm = 3.42pix		Rel. 1mm = 3.63pix		Ensamble Realizado
		Real (cm)	FPGA (Pixeles)	FPGA (cm)	Error(cm)	FPGA (cm)	Error(cm)	
1	Trapecio	(3.7,9.8)	(121,328)	(3.5,9.6)	(-0.2,-0.2)	(3.3,9.0)	(-0.4,-0.8)	No
	Contraparte	(14.4,9.3)	(495,311)	(14.5,9.1)	(+0.1,-0.2)	(13.6,8.6)	(-0.8,-0.7)	
2	Trapecio	(3.6,9.5)	(111,323)	(3.2,9.4)	(-0.4,-0.1)	(3.1,8.9)	(-0.5,-0.6)	No
	Contraparte	(3.3,3.4)	(107,107)	(3.1,3.1)	(-0.2,-0.3)	(2.9,2.9)	(-0.4,-0.5)	
3	Cuadrado	(5.2,4.3)	(512,158)	(5.1,4.3)	(-0.1,0.0)	(14.1,4.3)	(-0.8,-0.6)	Si
	Contraparte	(14.9,4.9)	(174,139)	(15.0,4.6)	(+0.1,-0.3)	(4.8,3.8)	(-0.4,-0.5)	
4	Cuadrado	(4.1,3.3)	(135,100)	(3.9,2.9)	(-0.2,-0.4)	(3.7,2.7)	(-0.4,-0.6)	Si
	Contraparte	(14.5,3.1)	(491,100)	(14.4,2.9)	(-0.1,-0.2)	(13.5,2.7)	(-1.0,-0.4)	
5	Trapecio	(3.6,3.3)	(111,104)	(3.2,3.0)	(-0.4,-0.3)	(3.1,2.9)	(-0.5,-0.4)	No
	Contraparte	(15.4,2.2)	(518,87)	(15.1,2.5)	(-0.3,+0.3)	(14.2,2.4)	(-1.2,-0.2)	
6	Circulo	(12.6,2.5)	(423,66)	(12.4,1.9)	(-0.2,-0.6)	(11.6,1.8)	(-1.0,-0.7)	No
	Contraparte	(8.0,10.8)	(272,360)	(8.0,10.5)	(0.0,-0.3)	(7.5,9.9)	(-0.5,-0.9)	
7	Circulo	(12.5,2.2)	(423,65)	(12.4,1.9)	(-0.1,-0.3)	(11.6,1.8)	(-0.9,-0.4)	No
	Contraparte	(8.0,10.8)	(273,360)	(8.0,10.5)	(0.0,-0.3)	(7.5,9.9)	(-0.5,-0.9)	
8	Circulo	(16.7,5.0)	(574,166)	(16.8,4.9)	(+0.1,-0.1)	(15.8,4.6)	(-0.9,-0.4)	Si
	Contraparte	(8.0,10.8)	(274,362)	(8.0,10.6)	(0.0,-0.2)	(7.5,10.0)	(-0.5,-0.8)	
9	Estrella	(2.5,9.2)	(80,294)	(2.3,8.6)	(-0.2,-0.6)	(2.2,8.1)	(-0.3,-0.3)	Si
	Contraparte	(15.4,3.4)	(527,109)	(15.4,3.2)	(0.0,-0.2)	(14.5,3.0)	(-0.9,-0.4)	

Tabla 4.5 Resultados de algunas pruebas de ensamble peg in hole, usando ambas relaciones entre milímetros y píxeles.

4.5 Validación del cálculo de la BOF

Con el fin de conocer la fiabilidad de los resultados de la BOF obtenidos con el FPGA, se envían y comparan los resultados con los obtenidos por medio de Matlab. Ambos métodos utilizan el algoritmo de la Figura 3.21 para encontrar la BOF de una región pero debido a que son formas diferentes de programar la implementación varia un poco, en el Anexo A se lista el código para obtener la BOF en Matlab.

En los resultados de las pruebas se muestran en primera instancia la imagen adquirida por la cámara donde solo se observan dos regiones, algunas con ruido dentro de la región lo cual por el momento no afecta la prueba. En segunda instancia se muestra la función frontera para las dos regiones, la línea continua representa el valor calculado en Matlab y la línea punteada muestra el resultado obtenido por el FPGA, ver Figura 4.9 y Figura 4.13. En cada una de las figuras se aprecia que el resultado del FPGA conserva la signatura de cada región de la imagen y en algunos casos los resultados son prácticamente iguales. Se aprecia un corrimiento en los valores obtenidos por el FPGA, ver Figura 4.13; sin embargo, hay que tomar en cuenta que el resultado del FPGA está dado en números enteros a diferencia de Matlab que el resultado es en números reales. Esta diferencia ocasiona que al determinar el valor de "c" en el algoritmo de la BOF, este puede diferir por una unidad; esto se debe a que el resultado de la división en el FPGA redondea siempre hacia abajo y en Matlab no. Puesto que "c" indica los puntos a tomar del contorno, entonces, si este dato es diferente las funciones tomaran valores diferentes pero siempre conservando la signatura de la región u objeto.

El sistema fue capaz de reconocer las dos piezas a ensamblar de entre tres piezas, en más de una prueba. En especial en la Figura 4.12 se muestra un ejemplo donde se colocaron una figura en forma de estrella de 5 picos, así como una pieza con forma de trapecio y su respectiva contraparte. Por medio de la detección de los máximos y mínimos de la BOF el sistema puedo discernir de entre las piezas cuales eran aquellas que se complementaban y realizo el ensamble. Aun así, hubo algunas pruebas que no fueron satisfactorias porque la BOF presentó ciertas irregularidades por deformaciones en las imágenes capturadas debido a la deformación causada por la lente, ver Figura 4.11.

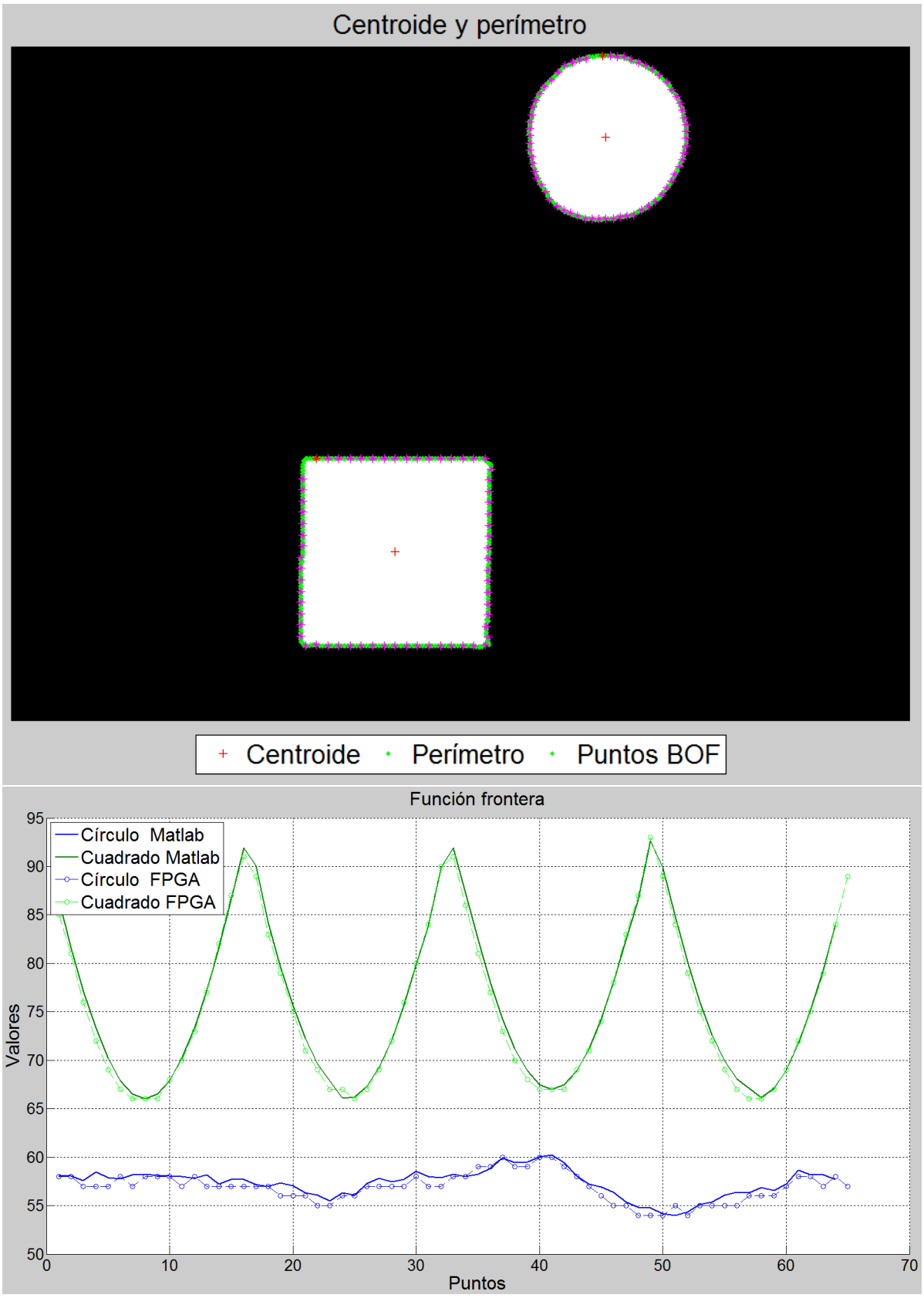


Figura 4.9 Prueba 7, pieza con forma de círculo y contraparte con forma de cuadrado.

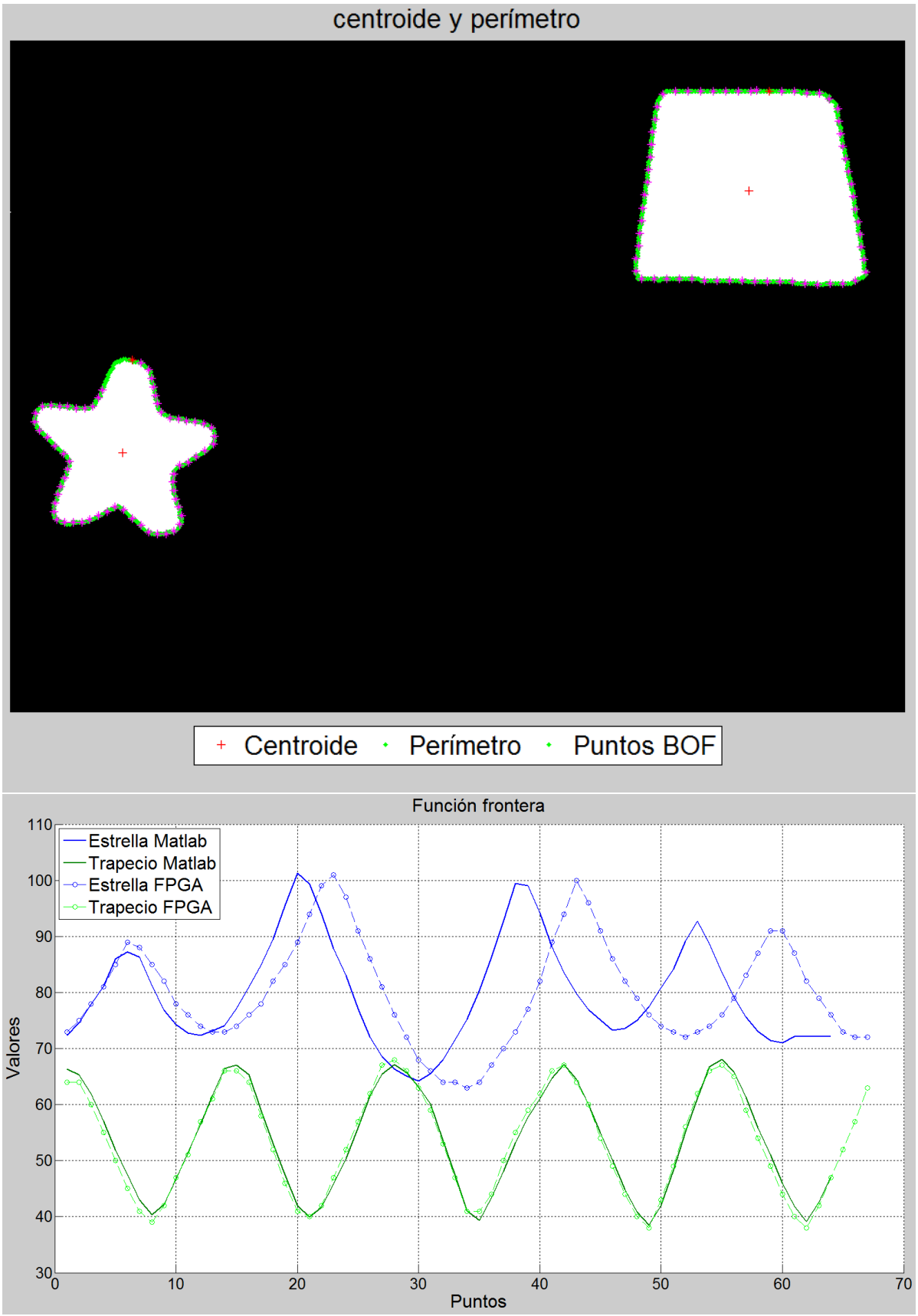


Figura 4.10 Prueba 9, pieza con forma de estrella y contraparte con forma de trapecio.

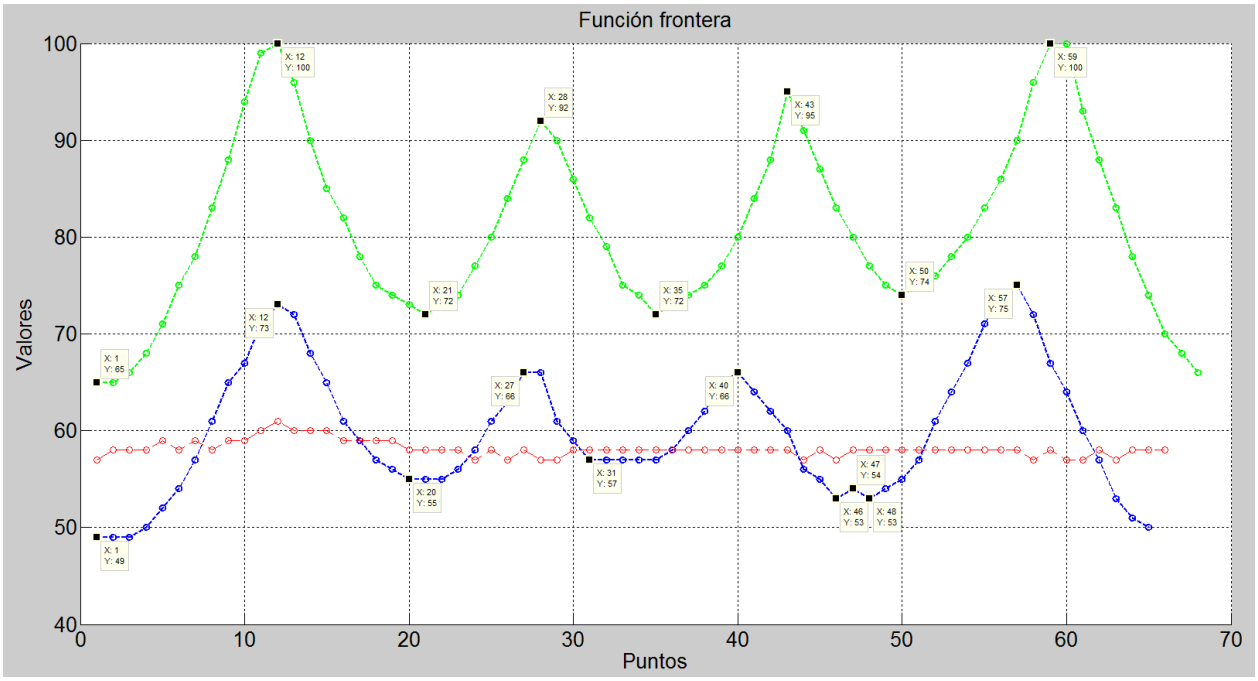


Figura 4.11 En azul y verde se muestra la BOF de dos figuras con forma de trapecio, en el caso de la azul no se reconoció el trapecio debido a que tiene 5 máximos.

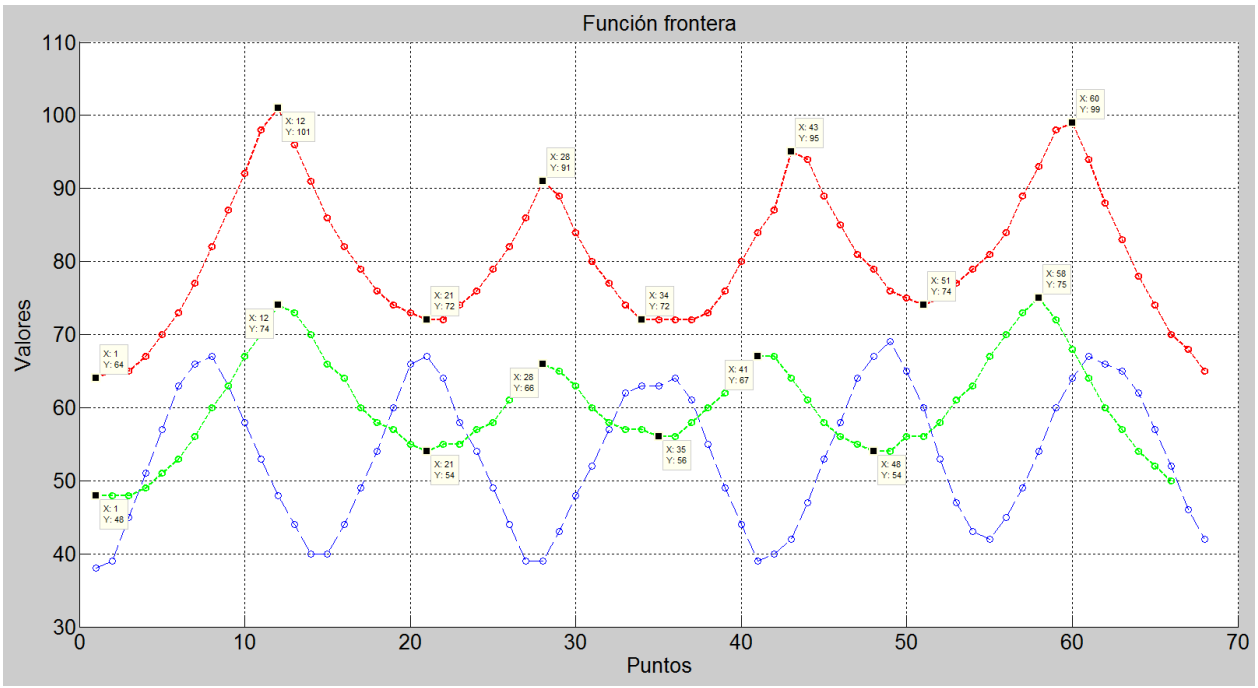


Figura 4.12 En azul BOF de una estrella de 5 picos, en verde BOF de un trapecio y en rojo BOF de un trapecio de mayor tamaño. El número de máximos determina los vértices de las figuras.

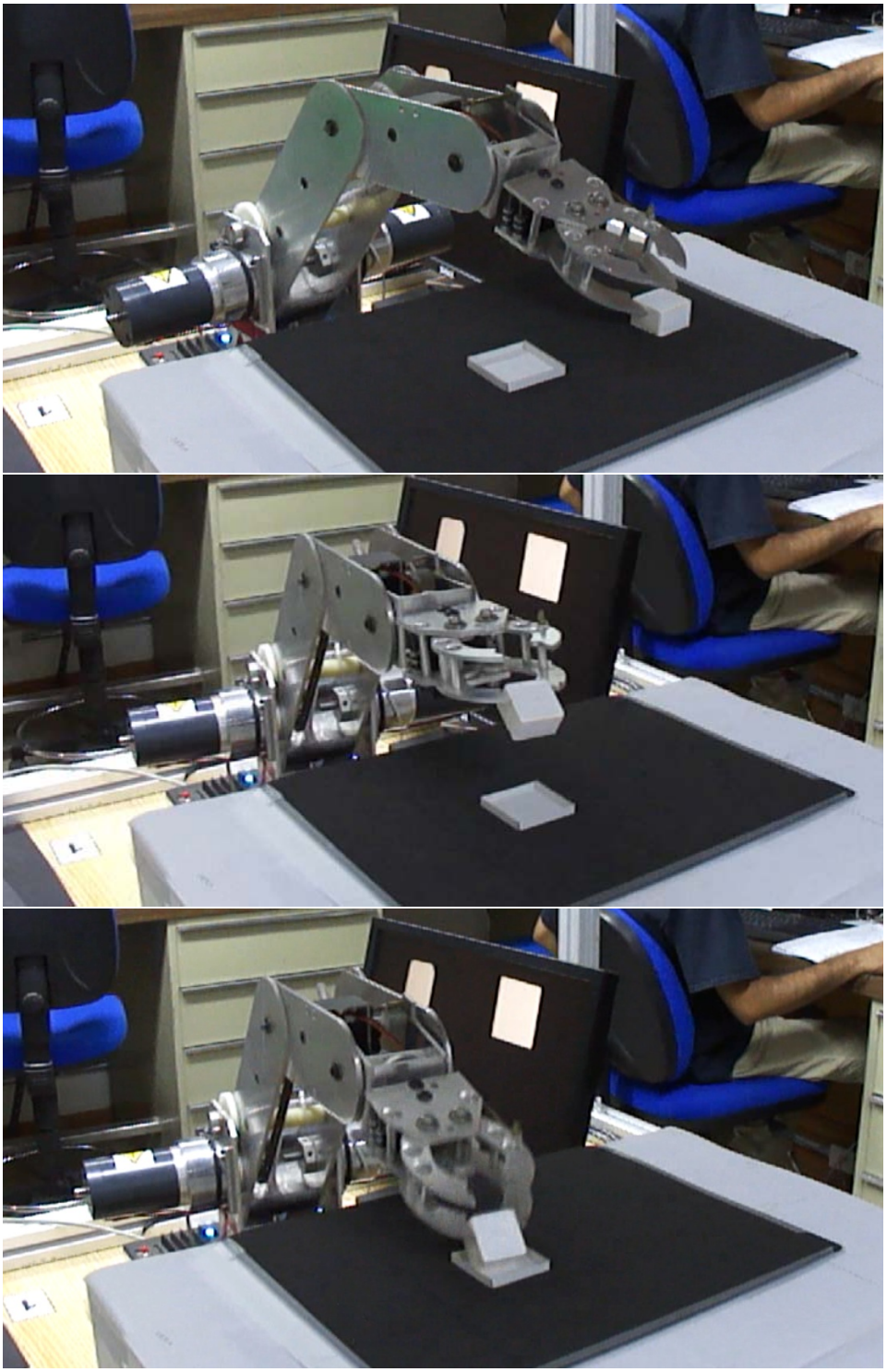


Figura 4.13 Ejemplo de ensamble de la prueba 4

Capítulo 5

Conclusiones

Este trabajo consistió en desarrollar un sistema de visión artificial que adquiere y procesa imágenes con el fin de realizar ensambles de tipo inserción en una celda de manufactura experimental. El desarrollo del sistema en FPGA involucró implementar protocolos de comunicación SCCB, UART y VGA para poder adquirir y transmitir información sobre las piezas del ensamble. También, se tuvieron que implementar algoritmos para detectar el histograma de una imagen en escala de grises y obtener los descriptores de regiones en una imagen como el área, contorno, centroide y la función frontera. De igual manera se mejoró el algoritmo propuesto por Chang, el cual no solo etiqueta sino que ahora también calcula el área y centroide de todas las regiones en una imagen con un solo barrido. Se planteó un algoritmo sencillo para encontrar el valor de umbral en una imagen en escala de grises a partir de su histograma.

En cuanto a la implementación, se utilizó una tarjeta de desarrollo Nexys 4 de Digilent, lo que permitió realizar la adquisición y procesamiento de imágenes, al igual que la manipulación del robot en una sola plataforma, teniendo los siguientes beneficios: reconocer la posición de las piezas a ensamblar y su posición en un tiempo aproximado de 8.9 ms con base en los resultados de las simulaciones, reducir el espacio ocupado normalmente por una computadora, disminuir drásticamente el consumo de energía y tener la opción a desarrollar un sistema en un circuito integrado programable (SoPC). Respecto al manipulador, se obtuvo un comportamiento suficiente para probar el sistema de visión y utilizarse en experimentos, pero aún lejos para presentarse como prototipo debido a su baja repetibilidad y precisión.

De los resultados obtenidos se puede decir que el sistema en general funciono adecuadamente en más del 50 % de los eventos. Este bajo resultado se atribuye al juego mecánico del manipulador

y a que se utilizan potenciómetros como sensores de posición los cuales suelen ser no muy precisos. Otro factor que afecta el bajo indice de eventos, es la perspectiva con la cual los objetos se muestran en la imagen, porque lejos del centro de la cámara, se aprecian tanto la cara superior como las laterales, lo que se interpreta por el sistema como una pieza diferente. Por otro lado, el error en el cálculo del centroide tiene un valor máximo de 1.2 cm, lo cual pareciera mucho si no se tiene en cuenta que la altura de las piezas altera la relación entre píxeles y milímetros.

Es necesario tener en cuenta que la iluminación juega un papel fundamental en los sistemas de visión, en este caso se evitó para simplificar el sistema, pero a cambio fue necesario utilizar condiciones especiales, como colocar un fondo oscuro y pintar las piezas de blanco. Aun así, las imágenes obtenidas contenían ruido debido a los reflejos de luz por fuentes externas lo que a su vez dificulta el procesamiento; esto provocó que fuese necesario adquirir las imágenes manualmente y mostrarlas en el monitor para asegurar que no hubiera ruido en estas, porque el más mínimo ruido impide el funcionamiento de los siguientes algoritmos.

Se mostró el gran potencial de la BOF en el reconocimiento de objetos donde proporciona mucha información sobre estos y es fácil de implementar. La principal ventaja de la BOF es que es invariante al tamaño, traslación y rotación de las regiones lo cual resulta muy útil en los sistemas de ensamble por inserción.

Por último, se alcanzó el objetivo planteado al principio del trabajo y se ha confirmado que es viable desarrollar sistemas de visión en FPGA, sobre todo en tareas industriales y con las ventajas que estos representan: pequeños, de bajo consumo, aptos para aplicaciones móviles y con un alto desempeño para el procesamiento de información. También, se ha comprobado que la aplicación de la visión artificial en los procesos de ensamble, implica nuevos problemas que resolver a cambio de grandes beneficios en los procesos de manufactura, en especial en ambientes poco estructurados.

5.1 Trabajo a futuro

El sistema de ensamble guiado por visión presentado en esta tesis cuenta con mucho trabajo por desarrollar. Para comenzar, es necesario encontrar un método más robusto para reconocer los objetos en la imagen mediante la BOF, aparte de los puntos máximos y mínimos, de tal manera que el método sea menos susceptible al ruido en la imagen. Igualmente se puede buscar

una relación entre el tamaño de la región y el número de puntos mínimos para el cálculo de la BOF de forma que se pueda reducir el espacio en memoria o el procesamiento de información y que aun se pueda distinguir la firma del objeto.

La función frontera además de ser muy útil en el reconocimiento de objetos, también puede ser empleada en las tareas de ensamble para ayudar a orientar la pieza sobre la contraparte. En este trabajo se realizó el ensamble de las piezas por medio de sus centroides, haciendo que coincidiera el de la pieza con su contraparte; no obstante, no se tomó en cuenta la orientación de la pieza sobre la contraparte y puesto que ambas pueden tener orientaciones diferentes faltaría alinear esta para concluir el ensamble. Lo anterior se puede lograr al encontrar un punto característico en la BOF de ambas figuras y hacerlos coincidir, de esta forma se podría asegurar que las piezas tuvieran la misma orientación, y que el ensamble fuera correcto.

Es necesario utilizar un sistema de iluminación que ayude a reducir el ruido en las imágenes ya que esto a su vez disminuye la complejidad de los algoritmos a implementar. Al integrar los sistemas de iluminación con los de visión se pueden realzar detalles particulares del objeto con el fin de realizar tareas de inspección, clasificación, entre otras.

En cuanto al manipulador es necesario buscar un reemplazo que permita tener un mayor espacio de trabajo, una mejor repetibilidad y una mayor precisión; o en su defecto, implementar un lazo de control cerrado que mejore el posicionamiento del manipulador. Lo anterior con el fin de aplicarse a objetos de diversos tamaños y ensambles con formas más variadas y complejas.

Por su parte el diseño del hardware en el FPGA, puede ser más eficiente si se usan técnicas como *pipeline* u otras estrategias de programación. Asimismo, hace falta desplegar la información obtenida de la imagen (centroide, área, BOF, etc) en el monitor u otras interfaces de salida, para que el usuario pueda seguir el estado del sistema. De igual manera se puede colocar una interfaz de entrada para que el usuario pueda seleccionar diferentes algoritmos a aplicar o cambiar los parámetros de estos.

Referencias

[1] P. Kellett, "Roadmap to the future." http://www.robotics.org/content-detail.cfm?content_id=1647, Agosto 2009. Consultado, septiembre 2015.

[2] N. Herakovic, *Robot vision*, ch. Robot vision in industrial assembly and quality control processes. InTech, Mar 2010.

[3] S. Jorg, J. Langwald, and M. Nickl, "Fpga based real-time visual servoing," in *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, vol. 1, pp. 749–752, Aug 2004.

[4] J. S. Hu, C. N. Y. C, J. J. Yang, J. J. Wang, L. R. G, M. C. Chien, Y. J. Chang, C. Y. Kai, and S. H. Su, "Fpga-based embedded visual servoing platform for quick response visual servoing," in *Control Conference (ASCC), 2011 8th Asian*, pp. 263–268, 2011.

[5] S. Y. Nof, W. E. Wilhelm, and H. J. Warnecke, *Industial Assembly*. Springer, 1997.

[6] M. P. Groover, *Fundamentos de manufactura moderna*. Mc Graw Hill, 2007.

[7] Y.-L. Kim, B.-S. Kim, and J.-B. Song, "Hole detection algorithm for square peg-in-hole using force-based shape recognition," in *Automation Science and Engineering (CASE), 2012 IEEE International Conference on*, 2012.

[8] W. Newman, Y. Zhao, and Y.-H. Pao, "Interpretation of force and moment signals for compliant peg-in-hole assembly," in *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, vol. 1, pp. 571–576, 2001.

[9] H. Park, J.-H. Bae, J.-H. Park, M.-H. Baeg, and J. Park, "Intuitive peg-in-hole assembly strategy with a compliant manipulator," in *Robotics (ISR), 2013 44th International Symposium on*, pp. 1–5, 2013.

[10] J. Takahashi, T. Fukukawa, and T. Fukuda, "Passive alignment principle for robotic assembly between a ring and a shaft with extremely narrow clearance," *Mechatronics, IEEE/ASME Transactions on*, vol. PP, no. 99, 2015.

[11] H.-C. Song, Y.-L. Kim, and J.-B. Song, "Automated guidance of peg-in-hole assembly tasks for complex-shaped parts," in *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, 2014.

[12] L. Lin, Y. Yang, Y. Song, B. Nemec, A. Ude, J. Rytz, A. Buch, N. Kruger, and T. Savarimuthu, "Peg-in-hole assembly under uncertain pose estimation," in *Intelligent Control and Automation (WCICA), 2014 11th World Congress on*, pp. 2842–2847, 2014.

[13] M. Peña-Cabrera, I. Lopez-Juarez, R. Rios-Cabrera, and J. Corona-Castuera, "Machine vision approach for robotic assembly," *Assembly Automation*, vol. 25, no. 3, pp. 204–216, 2005.

[14] M. Ejiri, "Machine vision technology: past, present and future," in *Intelligent Robots and Systems '90. 'Towards a New Frontier of Applications', Proceedings. IROS '90. IEEE International Workshop on*, vol. 1, pp. 31–40, IEEE, 3-6 Jul 1990.

[15] J. Jia, G. W. Krutz, and H. W. Gibson, "Corn plant locating by image processing," in *Proc. SPIE 1379, Optics in Agriculture*, vol. 1379, Nov 1991.

[16] J. Jia, "A machine vision application for industrial assembly inspection," in *Machine Vision, 2009. ICMV '09. Second International Conference on*, pp. 172–176, Dec 2009.

[17] A. Pretto, S. Tonello, and E. Menegatti, "Flexible 3d localization of planar objects for industrial bin-picking with monocamera vision system," in *Automation Science and Engineering (CASE), 2013 IEEE International Conference on*, pp. 168–175, Agu 2013.

[18] F. Spenrath, M. Palzkill, A. Pott, and A. Verl, "Object recognition: Bin-picking for industrial use," in *Robotics (ISR), 2013 44th International Symposium on*, pp. 1–3, Oct 2013.

[19] C. Hema, M. Paulraj, R. Nagarajan, and S. Yaacob, "Object localization using stereo sensors for adept scara robot," in *Robotics, Automation and Mechatronics, 2006 IEEE Conference on*, pp. 1–5, Jun 2006.

[20] H.-Y. Kuo, H.-R. Su, S.-H. Lai, and C.-C. Wu, "3d object detection and pose estimation from depth image for robotic bin picking," in *Automation Science and Engineering (CASE), 2014 IEEE International Conference on*, pp. 1264–1269, Aug 2014.

[21] K. Boehnke, "Object localization in range data for robotic bin picking," in *Automation Science and Engineering, 2007. CASE 2007. IEEE International Conference on*, pp. 572–577, Sept 2007.

[22] K. Rahardja and A. Kosaka, "Vision-based bin-picking: recognition and localization of multiple complex objects using simple visual cues," in *Intelligent Robots and Systems '96, IROS 96, Proceedings of the 1996 IEEE/RSJ International Conference on*, vol. 3, pp. 1448–1457, Nov 1996.

[23] A. Pochyly, T. Kubela, M. Kozak, and P. Cihak, "Robotic vision for bin-picking applications of various objects," in *Robotics (ISR), 2010 41st International Symposium on and 2010 6th German Conference on Robotics (ROBOTIK)*, pp. 1–5, Jun 2010.

[24] R. Epson, "Epson robots." http://robots.epson.com/, 2015. Consultado, marzo 2015.

[25] A. Fernandes, L. F. E. Moreira, and J. Mata, "Machine vision applications and development aspects," in *Control and Automation (ICCA), 2011 9th IEEE International Conference on*, pp. 1274–1278, IEEE, 19-21 Dec 2011.

[26] P. David, "Implementing video analysis in xilinx fpgas." http://japan.xilinx.com/publications/archives/magazines/emb05.pdf, April 2009. Consultado, febrero 2015.

[27] S. Gallagher, "Systems dsp algorithms into fpgas," November 2010.

[28] Z. Zhou, "Fpga implementation of computer vision algorithm," Master's thesis, Electrical Engineering, University of California Riverside, 2014.

[29] C. Kyrkou, C. Ttofis, and T. Theocharides, "Fpga-accelerated object detection using edge information," in *Field Programmable Logic and Applications (FPL), 2011 International Conference on*, pp. 167–170, IEEE, 5-7 Sept 2011.

[30] M. de la Fuente, J. Echanobe, I. del Campo, L. Susperregui, and I. Maurtua, "Development of an embedded system for visual servoing in an industrial scenario," in *Industrial Embedded Systems (SIES), 2010 International Symposium on*, pp. 192–196, 2010.

[31] M. A. Sanchez-Martínez, "Diseño en fpga de un circuito comparador de imágenes," Master's thesis, Departamento de Ingeniería Eléctrica, CINVESTAV, Julio 2005.

[32] S. J. Huang and S. S. Wu, "Vision-based robotic motion control for non-autonomous environment," in *Control Conference (ECC), 2007 European*, pp. 1455–1462, Jul 2007.

[33] M. P. Groover, *Fundamentals of modern manufacturing materials, processes and systems*. John Wiley & Sons, 2010.

[34] S. Kalpakjian and S. R. Schmid, *Manufacturing engineering and technology*. Prentice Hall International, 2001.

[35] S. B. Niku, *Introduction to robotics: analysis, control, applications*. John Wiley & Sons, 2 ed., 2011.

[36] R. KUKA, "Kuka robotics corp.." http://www.kuka-robotics.com, 2015. Consultado, septiembre 2015.

[37] R. ABB, "ABB Group." http://www.abb.com/, 2015. Consultado, septiembre 2015.

[38] I. Adept Technology, "Adept Technology, Inc.." /urlhttp://www1.adept.com/, 2015. Consultado, septiembre 2015.

[39] A. Padilla Salazar, "Integración de un manipulador robótico de 6 grados de libertad controlado por voz como elemento central a una celda de manufactura experimental con modo sensorial de visión," Master's thesis, Posgrado en Ingeniería - UNAM, 2013.

[40] I. International Organization for Standardization, "Online browsing platform (obp)." https://www.iso.org/obp/ui#iso:std:iso:9283:ed-2:v1:en, 1998. Consultado, Octubre 2015.

[41] W. E. Snyder and H. Qi, *Machine vision*. Cambridge University Press, 2004.

[42] B. G. Batchelor (Ed.), *Machine vision handbook*. Londres: Springer, 2012.

[43] A. Downton and D. Crookes, "Parallel architectures for image processing," *Electronics Communication Engineering Journal*, vol. 10, pp. 139–151, Jun 1998.

[44] C. Weems, "Architectural requirements of image understanding with respect to parallel processing," *Proceedings of the IEEE*, vol. 79, pp. 537–547, Apr 1991.

[45] D. G. Bailey, *Design for embedded image processing on FPGAs*. Singapur: John Wiley & Sons, 2011.

[46] R. Gonzalez and R. Woods, *Digital Image Processing*. Prentice Hall, 3 ed., 2008.

[47] C. Solomon and T. Breckon, *Fundamentals of Digital Image Processing*. Wiley Blackwell, 2011.

[48] J. M. Pérez, "Universidad de málaga." http://www.lcc.uma.es/~munozp/, Julio 2012. Consultado, septiembre 2015.

[49] T. L. Floyd, *Fundamentos de ssistema digitales*. Pearson, 2006.

[50] Xilinx Inc, "Using look-up tables as distributed ram in spartan-3 generation fpgas," Application Note XAPP464, Xilinx Inc, Mar 2005.

[51] C. Maxfield, *The Design Warrior's Guide to FPGAs*. Newnes, 2004.

[52] Xilinx Inc, "Using block ram in spartan-3 generation fpgas," Application Note XAPP463, Xilinx Inc, Mar 2005.

[53] D. G. Maxinez and J. A. Jara, *VHDL El arte de programar sistema digitales*. CECSA, 2002.

[54] I. O. R. Peña, M.; Lopez, "Invariant object recognition robot vision system for assembly," in *Electronics, Robotics and Automotive Mechanics Conference*, vol. 1, pp. 30,36, Sept 2006.

[55] OmniVision Technologies, Inc., "Ov7670/ov7171 cmos vga (640x480) camerachip sensor with omnipixel technology.," Datasheet 1.4, OmniVision Technologies, Inc., August 21 2006.

[56] OmniVision Technologies, Inc., "Omnivision serial camera control bus (sccb) functional specification," Application Note 2.2, OmniVision Technologies, Inc., June 2007.

[57] OmniVision Technologies, Inc., "Ov7670/ov7171 cmos vga (640x480) camerachip implementation guide," Application Note 1.0, OmniVision Technologies, Inc., September 2005.

[58] E. de la Fuente López and F. M. Trespaderne, *Visión artificial industrial: Procesamiento de imágenes para inspección automática y Robótica*. Valladolid: Universidad de Valladolid, 2013.

[59] A. G. Piñera, *Óptica visual*. Diego Marin Librero Editor, S.L., 2004.

[60] F. Chang, C. J. Chen, and C. J. Lu, "A linear-time component-labeling algorithm using contour tracing technique," *Computer Vision and Image Understanding*, vol. 93, no. 2, pp. 206 – 220, 2004.

[61] B. Parhami, *Computer Arithmetic Algorithms and Hardware Designs*. New York: Oxford university press, Inc, 2000.

[62] Y. Li and W. Chu, "A new non-restoring square root algorithm and its vlsi implementations," in *Computer Design: VLSI in Computers and Processors, 1996. ICCD '96. Proceedings., 1996 IEEE International Conference on*, pp. 538–544, Oct 1996.

[63] R. CRYA, *Manual del sistema Labot Pro 5*, 2008.

[64] Digilent Inc., "Vga display controller." https://learn.digilentinc.com/Documents/269. Consultado, septiembre 2015.

[65] Xilinx Inc., "Hardware developer zone." http://www.xilinx.com/support/download/index.html/content/xilinx/en/downloadNav/design-tools.html, Junio 2013. Consultado, Septiembre 2015.

[66] Digilent Inc., "Digilent adept." http://www.digilentinc.com/Products/Detail.cfm?NavPath=2,66,828&Prod=ADEPT2. Consultado, Septiembre 2015.

Anexo A

Códigos de Matlab

Conversión de RGB a escala de grises

Método del promedio

```
function [imageGray] = toGrayAverage(imagen)
imagen=uint16(imagen);
[row,col,plane]=size(imagen);
image2=zeros(row,col);
image2=uint8(image2);
for i = 1:row
    for j = 1:col
            image2(i,j)=(imagen(i,j,1)+imagen(i,j,2)+imagen(i,j,3))/3;
    end
end
imageGray=uint8(image2);
```

Método de luminosidad

```
function [imageGray] = toGrayLuminosity(image1)
[row,col,plane]=size(image1);
image2=zeros(row,col);
image2=uint8(image2);
```

```
for i = 1:row
    for j = 1:col
        image2(i,j)= ((image1(i,j,1)*0.299)+(image1(i,j,2)*0.587)+
        (image1(i,j,3)*0.114));
    end
end
imageGray=image2;
```

Cálculo del histograma

```
function [valores] = Mihistograma(imagenGray)
vecHistograma = zeros(1,256);
[row,col] = size(imagenGray);
for i = 1:row
    for j = 1:col
       pixel = imagenGray(i,j);
       vecHistograma(pixel+1) = vecHistograma(pixel+1)+1;
    end
end
valores = vecHistograma';
end
```

Cálculo del umbral

```
function  [ValorMin,umbral] = Umbral_Histograma(histograma)

R = fix(max(histograma)*0.05);
a = find(histograma>R,1,'first');
b = find(histograma>R,1,'last');

if a==1
   a=2;
end

if b==256
   b=255;
end

histograma2=histograma(a-1:b+1);
```

```
[ValorMin,umbral]=min(histograma2);
umbral=a-2+umbral;
end
```

Binarización

```
function [ imagenBin ] = ImagenBinaria( imagenGray,umbral )
[row,col] = size(imagenGray);
imagenBin=zeros(row,col);
for i=1:row
    for j=1:col
        if imagenGray(i,j) < umbral
            imagenBin(i,j) = 1;
        else
            imagenBin(i,j) = 0;
        end
    end
end
end
```

Seguidor Etiquetado

```
function [area,cen,E,ext,int]=SeguidorEtiquetado(B)

[rows_b,cols_b]=size(B);
netiq=0;
area=zeros(1,7);
Cx=zeros(20,1);
Cy=zeros(20,1);
cen=zeros(20,3);
C_i=0;
int=0;
E=zeros(rows_b,cols_b);

for x=2:rows_b-1
 etiq_activa=0;
  for y=2:cols_b-1
   if B(x,y)==1
    if etiq_activa~=0  %etiquetando la region etiq_activa
```

```
18      if E(x,y)==0
19       E(x,y)=etiq_activa;
20       area(etiq_activa)=area(etiq_activa)+1;  %calculo del area
21       Cx(etiq_activa)=Cx(etiq_activa)+y;
22       Cy(etiq_activa)=Cy(etiq_activa)+x;
23      end
24     else
25      etiq_activa=E(x,y);
26      if etiq_activa==0 %Nueva region
27       netiq=netiq+1;
28       etiq_activa=netiq;
29       P_e=[x y];    %Punto inicial del contorno exterior
30       [area,Cx,Cy,C_e,E]=SeguidorContorno(B,E,etiq_activa,P_e,0,area,Cx,Cy);
31       [r,~]=size(C_e);
32       ext(1:r,2*etiq_activa-1:2*etiq_activa)=C_e; %numero impar 2*n-1
33      end
34     end
35     cen(etiq_activa,:,:)=[etiq_activa,round(Cx(etiq_activa)/area(etiq\\_activa)),
   round(Cy(etiq_activa)/area(etiq_activa))];
36    else
37     if etiq_activa~=0  %Acabamos de abandonar la region
38      if E(x,y)==0%primer punto encontrado en agujero interno
39       P_i=[x-1,y]; %Punto inicial del contorno interior
40       [area,Cx,Cy,C_i,E]=SeguidorContorno(B,E,etiq_activa,P_i,3,area,Cx,Cy);
41       [r,~]=size(C_i);
42       int(1:r,2*etiq_activa-1:2*etiq_activa)=C_i; %numero impar 2*n-1
43      end
44      etiq_activa=0;
45     end
46    end
47   end
48  end
49 for x=1:rows_b    %Quitar los valores -1 de la Matriz E
50  for y=1:cols_b
51   if E(x,y)==-1
52    E(x,y)=0;
53   end
54  end
55 end
56 end
```

Seguidor Contorno

```
function [area,Cx,Cy,c,E]=SeguidorContorno(B,E,etiq,Pfin,codigo,area,Cx,Cy)

contador=1;
[Psig,codigo,E]=SiguientePunto(B,E,Pfin,codigo);
c(contador,:)=Psig;    %Primer punto del contorno
if not(isequal(Psig,Pfin))%Psig~=Pfin
 seguir=1;
 while seguir==1
  if E(Psig(1),Psig(2))==0
   E(Psig(1),Psig(2))=etiq;
   area(etiq)=area(etiq)+1;
   Cx(etiq)=Cx(etiq)+Psig(2);  %se prepara para calcular el centroide
   Cy(etiq)=Cy(etiq)+Psig(1);
  end
  codigo=mod(codigo+6,8);
  Pant=Psig;
  [Psig,codigo,E]=SiguientePunto(B,E,Pant,codigo);
  if isequal(Psig,c(1,:)) && isequal(Pant,Pfin)
   seguir=0;
  else
   contador=contador+1;
   c(contador,:)=Psig;
  end
 end
end
end
```

Siguiente Punto

```
function [Psig,codigo,E]=SiguientePunto(B,E,Pact,codigo)
%Pact=(x,y), coordenadas del punto actual en el contorno
%Psig=(x,y), coordenadas del punto sucesor Psig si existe,
%o Psig=Pact si no existe

exito=0;
contador=1;
```

```
 while exito==0 && contador<8  %Busqueda en los 7 vecinos
  switch codigo           %Nuevas coordenadas de busqueda
   case 0
    Psig=[Pact(1),Pact(2)+1];
   case 1
    Psig=[Pact(1)+1,Pact(2)+1];
   case 2
    Psig=[Pact(1)+1,Pact(2)];
   case 3
    Psig=[Pact(1)+1,Pact(2)-1];
   case 4
    Psig=[Pact(1),Pact(2)-1];
   case 5
    Psig=[Pact(1)-1,Pact(2)-1];
   case 6
    Psig=[Pact(1)-1,Pact(2)];
   case 7
    Psig=[Pact(1)-1,Pact(2)+1];
   end
   if B(Psig(1),Psig(2))==0  %El punto no pertenece a la region
    E(Psig(1),Psig(2))=-1;%Marcamos el pixel
    codigo=mod(codigo+1,8);%Siguiente posicion segun agujas del reloj
   else
    exito=1;
   end
   contador=contador+1;
  end
  if exito==0
   Psig=Pact; %No se ha encontrado ningun vecino: punto aislado
  end
end
```

Cálculo de la BOF

```
clear all
close all
clc
prompt = 'Nombre de la Prueba?';
str=input(prompt,'s');
name1=strcat(str,'imagen','.txt');
```

```
name2=strcat(str,'area','.txt');
name3=strcat(str,'bof1','.txt');
name4=strcat(str,'bof2','.txt');
name5=strcat(str,'bof3','.txt');
mat1=dlmread(name1);
area_cen=dlmread(name2);
bof1=dlmread(name3);
bof2=dlmread(name4);
bof3=dlmread(name5);
pix=3.42;
centroidePix=[area_cen(5) area_cen(6); area_cen(8) area_cen(9);
area_cen(11) area_cen(12)];
datosmm=num2str([area_cen(1:3)', centroidePix/pix], '%6.0f')

[area,cen,E,ext,int]=SeguidorEtiquetado(mat1);
[~,objetos]=size(ext);
PP=64;
puntos=zeros(PP,2);
[fil,objetos]=size(ext);

%Graficar los centroides
figure(1)
imshow(mat1), title('centroide y perimetro');
hold on
plot(cen(1:objetos/2,2),cen(1:objetos/2,3),'r+')
% plot(centroidePix(:,1),centroidePix(:,2),'bx')

%extraer de la matriz de perimetros externos, un perimetro por objeto
for w=1:objetos/2
    [pos]=find(ext(:,2*w),1,'last');
    per=ext(1:pos,2*w-1:2*w);
    plot(per(:,2),per(:,1),'g.')
end

%Calculo de la BOF
for n=1:objetos/2
    cx_temp=cen(n,2);
    cy_temp=cen(n,3);
    [pos]=find(ext(:,2*n),1,'last');
    perimetro_temp=ext(1:pos,2*n-1:2*n);
    [Prow,Pcol] = size(perimetro_temp);
    c = round(Prow/PP);
```

```
    for i = 1:PP
        if (i*c) <= Prow
            puntos(i,:) = perimetro_temp(i*c,:);
        else
            puntos(i,:) = perimetro_temp(Prow,:);
        end
    end
    m1(1:PP)=cx_temp;
    m2(1:PP)=cy_temp;
    mat=[m2' m1'];
    cuadrado=(puntos-mat).^2;
    raiz=sqrt(sum(cuadrado,2));
    BOF(1:PP,n)=raiz;

    %Grafica de la BOF
    plot(puntos(:,2),puntos(:,1),'m*');
    legend('Centroide','Perimetro','Puntos
    BOF','Location','Southoutside','Orientation','horizontal');
    plot(puntos(1,2),puntos(1,1),'r*');
end

label=cen(1:objetos/2,1);
la=num2str(label);
figure(2)
hold on
plot(BOF,'LineWidth',2)
plot(bof1,'b--o')
plot(bof2,'g--o')
plot(bof3,'r--o')
grid
legend(la,'Location','SouthEast');
title('Funcion frontera')
xlabel('Puntos')
ylabel('Valores')
```

Recepción de datos del FPGA

```
clear all
close all
clc
```

```
4  prompt = 'Nombre de la Prueba?';
5  str=input(prompt,'s');
6
7  delete(instrfind({'Port'},{'COM4'})); %Borrar conexiones previas
8  s = serial('COM4','BaudRate',38400);  %Crear una conexion serie1
9  n=307200;
10 valimage=zeros(1,n);
11 fopen(s);                              %Abrir el puerto
12 fprintf(s, 'A');                       %Mandar A
13 for i=1:n
14 valimage(1,i) = fread(s,1,'uchar');
15 end
16 fclose(s);                             %cierra puerto
17
18 fopen(s);                            %Abrir el puerto
19 fprintf(s, 'B');                     %Mandar B
20 n=64;
21 valarea=zeros(1,n);
22 for i=1:n                            %Leer el puerto serie
23 valarea(1,i) = fread(s,1,'uchar');
24 end
25 fclose(s);                            %cierra puerto
26
27 fopen(s);                            %Abrir el puerto
28 fprintf(s, 'C');                     %Mandar C
29 n=128;
30 valbof1=zeros(1,n);
31 for i=1:n
32 valbof1(1,i) = fread(s,1,'uchar');
33 end
34 fclose(s);                            %cierra puerto
35
36 fopen(s);                            %Abrir el puerto
37 fprintf(s, 'D');                     %Mandar D
38 n=128;
39 valbof2=zeros(1,n);                  %Leer el puerto serie
40 for i=1:n
41 valbof2(1,i) = fread(s,1,'uchar');
42 end
43 fclose(s);                            %cierra puerto
44
45 fopen(s);                            %Abrir el puerto
```

```
46 fprintf(s, 'E');                    %Mandar E
47 n=128;
48 valbof3=zeros(1,n);
49 for i=1:n
50 valbof3(1,i) = fread(s,1,'uchar');
51 end
52 fclose(s);                          %cierra puerto
53 delete(s);
54 clear s;
55
56 mat=reshape(valimage,[640,480]);
57 mat=rot90(rot90(rot90(mat)));
58 mat=flip(mat,2);
59 imshow(mat)
60
61 %concatena los bytes entrantes en palabras de 16 bits
62 a=uint8(valarea);
63 a1=a(1:2:32);
64 a2=a(2:2:32);
65 for i=1:16
66 area_cen(i)=bin2dec([dec2bin(a1(i),8) dec2bin(a2(i),8)]);
67 end
68 %convierte de pixeles a milimetros y reacomoda los datos
69 pix=3.42;
70 areas=[area_cen(1);area_cen(2);area_cen(3)];
71 centroidePix=[area_cen(5) area_cen(6); area_cen(8) area_cen(9);
72 area_cen(11) area_cen(12)];
73 centroidemm=centroidePix/pix;
74 datos=[areas, centroidemm]
75 bof1=valbof1(1,1:find(valbof1,1,'last'));
76 bof2=valbof2(1,1:find(valbof2,1,'last'));
77 bof3=valbof3(1,1:find(valbof3,1,'last'));
78 name1=strcat(str,'imagen','.txt');
79 name2=strcat(str,'area','.txt');
80 name3=strcat(str,'bof1','.txt');
81 name4=strcat(str,'bof2','.txt');
82 name5=strcat(str,'bof3','.txt');
83 dlmwrite(name1,mat);
84 dlmwrite(name2,area_cen);
85 dlmwrite(name3,bof1);
86 dlmwrite(name4,bof2);
87 dlmwrite(name5,bof3);
```