Centro de Investigación y de Estudios Avanzados
del Instituto Politécnico Nacional

Departamento de Computación

Una herramienta para cuantificar el error de posición de circuitos integrados en la producción de circuitos impresos

Tesis que presenta

Uriel Martínez Hernández

para obtener el Grado de

Maestro en Ciencias en Computación

Director de la Tesis

Dr. Adriano de Luca Pennachia

México, D.F. **Diciembre, 2008**

Resumen

La fabricación de circuitos impresos es un proceso que requiere de sistemas de alta precisión. Esto se debe a la muy alta escala de integración en los circuitos integrados, los cuales cada vez tienen características técnicas más críticas, por ejemplo, la separación entre las terminales de un circuito integrado está en el rango de 0.6 mm a 1.2 mm. También el número de funciones que pueden realizar es mayor, así como también el diseño y fabricación de circuitos impresos es más complejo. Sin embargo, tener circuitos integrados con separación entre terminales reducidas puede causar problemas en el momento de realizar el montaje en el circuito impreso. Existen problemas que se pueden presentar por diferencias existentes entre el circuito impreso diseñado y el circuito impreso fabricado, lo cual provoca que el montaje de circuitos integrados sea erróneo. Estas diferencias existen ya que en la fabricación de circuitos impresos intervienen sistemas mecánicos y ópticos. Así, el problema es identificar las posiciones de los sockets en el circuito impreso fabricado y verificar los resultados contra los datos tomados del diseño original. En esta tesis se presenta una solución para poder conocer el estado de los circuitos impresos antes de pasar por la línea de montaje, específicamente el estado de los sockets de los circuitos integrados. La solución propuesta es tener un sistema semiautomático y fuera de línea, es decir, un sistema que permita elegir la zona que se desea analizar sin que se interfiera con la línea de montaje.

La solución propuesta es el desarrollo de un sistema de software llamado QC-Tool (Quality Control Tool) que utiliza diversas técnicas de Procesamiento Digital de Imágenes, el cual realiza el cálculo de las posiciones de los sockets de los circuitos integrados utilizando un par de puntos de referencia asignados a cada socket, ya que estos son quienes se ven más afectados durante la fabricación de los circuitos impresos. Así, conociendo la posición de los sockets de los circuitos integrados del circuito impreso fabricado se pueden reducir tanto los retrasos como los errores en la línea de montaje, ya que el sistema encargado del montaje puede utilizar esta información para posicionarse directamente en el lugar de montaje correcto, eliminando así el proceso de búsqueda de la posición por parte del sistema. Como se mencionó, este sistema utiliza técnicas de Procesamiento Digital de Imágenes como extracción de objetos, reconocimiento de patrones, modelos de color, entre otras. El sistema ha sido desarrollado en lenguaje C++ y la biblioteca Qt y ha sido probado satisfactoriamente en plataformas como Macintosh, Linux y Windows.

Abstract

The development of Integrated Circuits (ICs) is a process that requires high-precision systems, due to the Very Large Scale Integration (VLSI) technology, an example is the separation between the terminals in the ICs, which is the range of 0.6 mm and 1.2 mm. Also the number of functions that an IC can develop has increased and the design and development of ICs is more complex. A disadvantage of the reduced dimensions of the ICs, generates some problems in the montage process. The problems presented due to the differences between the Printed Circuit Boards (PCBs) designed and the PCBs developed by the producers can produce an wrong montage process of of the ICs on the PCBs. The differences in the PCBs are presented due to the lack of precision of the mechanical or optical systems used in the PCB production. Then, the problem is to identify the positions of the IC's sockets on the PCBs and verify the results versus the data from the original PCB design, by this way the differences in the PCBs can be detected. In this work a solution is proposed to know PCB's status before the assambly line process. The solution is a semi-automatic system, which out of line the user can select the region of the PCB to be analyzed.

The solution proposed has been implemented in software, using many techniques of Digital Image Processing (DIP) and the software developed is called QCTool (Quality Control Tool). The software develops the necessary operations to calculate the positions of the sockets on the PCBs, using a pair of point of reference in each socket. Thus, knowing the socket's positions, the time and errors in the montage process can be reduced. Then, the software uses DIPs techniques, for example, object recognition, pattern recognition and color models. The software has been developed in C/C++ language and Qt library. The system has been tested successfully in Macintosh, Linux and Windows platforms.

Agradecimientos

Dedico esta tesis a mis padres **Armando** y **Josefina**, quienes han sido la parte fundamental en mi formación académica y humana. Gracias a ustedes he aprendido a luchar en la vida y nunca darme por vencido. La vida está llena de obstáculos y gracias a su apoyo incondicional he podido superarlos y salir adelante, no hay palabras para agradecer por todo lo que me han dado, ni para expresar lo mucho que los quiero.

Dedico esta tesis a mis hermanos **Omar** y **Armando**, con quienes he reido, llorado y disfrutado mucho de la vida. Gracias por aguantarme en todo momento, el los tiempos buenos y malos. Gracias por su compañía, gracias por hacerme la vida más alegre y mil gracias por su carño. Gracias, los quiero mucho.

A mis tios **Pedro** y **María Luisa**, y a mis primos **Cristian**, **Eder**, **Pedro** y **Brenda** quienes han estado allí siempre para ayudarnos a mi familia y mí en los tiempos buenos y malos. Nunca voy a olvidar todo lo que han hecho por nosotros y ahora me doy cuenta de que los verdaderos amigos estan en la buenas y en las malas, pero sobre todo en las malas. Mil gracias.

A mis amigos **Eduardo** y **Martín**, con quienes he vivido experiencias buenas y malas, con quienes he crecido y con quienes he tenido una amistad de esas que duran para toda la vida. Gracias amigos.

Al **Dr. Adriano de Luca Pennachia**, por haber sido mi director de tesis y por haber confiado en mi. Gracias por haberme transmitido conocimiento sobre diversas áreas durante todo este tiempo que trabajamos juntos. Gracias por ayudarme a ver la vida de otro modo. Gracias.

Al **Dr. Gerardo de la Fraga**, con quien aprendí muchísimo a través de todos sus cursos impartidos y también a través de sus consejos en la realización de esta tesis. Gracias a usted encontré que el Procesamiento de Imágenes y Graficación por Computadora son muy divertidos y fascinantes y a lo que me gustaría dedicarme en la vida, en realidad le debo mucho. Gracias.

A la **Dra. Xiaoou Li**, por haberse tomado el tiempo en la revisión de esta tesis y mostrarme mis errores, ya que de estos se aprende mucho. Gracias.

A **Sofia Reza**, una persona que siempre sabe escuchar. Gracias Sofi por todo tu apoyo y por ser tan amable.

Esta tesis también se las dedico a todas las personas que me han sabido escuchar y siempre han soportado mi mal carácter. Gracias.

Al **CINVESTAV-IPN** por la oportunidad de realizar mis estudios de maestría y por permitirme conocer a muchas personas muy valiosas.

Al **CONACyT** por el apoyo económico durante toda la maestría.

Agradezco a todo el Departamento de Computación por haberme transmitidos muchísimos conocimientos durante todo este tiempo. A todos los doctores por ser excelentes profesores además de ser buenas personas. Los estudios de maestría en el CINVESTAV-IPN han sido una experiencia fascinante, dandome una formación completa como persona, profesionalmente y sobre todo inspirarme a seguir preparandome y nunca detenerme ante nada, nunca voy a olvidar este maravilloso lugar ni las personas que laboran en él.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

La fabricación de circuitos impresos es un proceso que requiere de sistemas automáticos de alta precisión. En estos circuitos impresos se colocan los componentes electrónicos que de manera conjunta llevan a cabo una ó diversas tareas. Los circuitos impresos generalmente son robustos, baratos y muy confiables. Inicialmente el costo de la producción de un circuito impreso es alto, sin embargo, al producirse estos en grandes volúmenes su costo se abarata [1, 2].

Los tipos de circuitos impresos se pueden dividir en:

- Una sola capa: se utiliza un solo lado de la tarjeta para el montaje de componentes electrónicos
- Doble capa: se utilizan los dos lados de la tarjeta para el montaje de componentes electrónicos
- Multicapa: suelen tener entre 8 y 10 capas, donde cada una de estas tiene diferentes pistas para la conexión de los componentes electrónicos

El sistema automático para la producción de circuitos impresos utiliza un archivo de texto que cuenta con la información necesaria sobre el circuito impreso. Comúnmente a este archivo se le llama Gerber. Este archivo es obtenido a partir del diseño original del circuito impreso, el cual actualmente se puede desarrollar utilizando tanto programas de diseño comerciales como programas de diseño libres, ambos muy potentes. Así, la creación de un circuito impreso es un proceso automático desde el momento en que se diseña el diagrama esquemático de algún diseño electrónico. La figura 1.1 muestra un diagrama general de las etapas de creación de un circuito impreso.

Se ha mencionado que se requiere de sistemas automáticos de alta precisión para la creación de circuito impresos, esto es debido a que actualmente se pueden fabricar Circuitos Integrados ó ICs (del inglés Integrated Circuits) a muy grande escala de

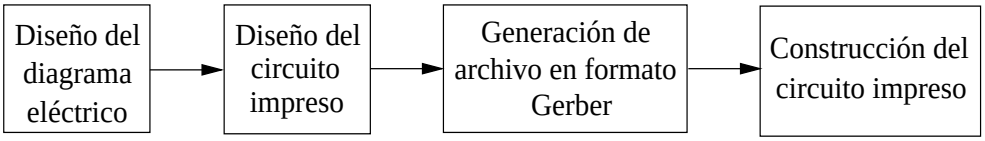


Figura 1.1: Proceso de creación de un circuito impreso

integración ó VLSI (del inglés Very Large Scale Integration). La integración a gran escala esta basada en la inserción de transistores en un IC ó chip, lo cual inició en la década de 1980. Inicialmente los ICs contenían un solo transistor, pero debido al avance gigantesco de la tecnología actualmente los circuitos VLSI tienen la característica de contener millones de transistores en un solo chip, lo cual a su vez permite que un chip pueda integrar un número grande funciones, por ejemplo los microprocesadores y microcontroladores. Otra características de los circuitos VLSI es que cuentan con mayor número de terminales y la separación entre estas es muy pequeña, la cual actualmente se encuentra en el rango de 0.65 mm (0.026 in) a 1.27 mm (0.050 in). Para poder tener esta separación tan pequeña entre las terminales, así como para poder realizar el montaje de los ICs en el circuito impreso, la fabricación de estos ICs se realiza con la tecnología de montaje superficial ó SMT (del inglés Surface Mount Technology). Los componentes electrónicos de montaje superficial son también llamados dispositivos de montaje superficial ó SMD (del inglés Surface Mount Device).

En el montaje de componentes electrónicos en circuitos impresos se tiene dos tipos diferentes de tecnologías; SMT, la cual permite realizar el montaje de componentes de manera superficial, es decir, sin realizar perforaciones en el circuito impreso, mientras que con la tecnología THT (del inglés Through Hole Technology) se requiere perforar el circuito impreso y pasar las terminales del IC a través de las perforaciones en el circuito impreso. Así, al tener que realizar perforaciones con THT, la separación entre sus terminales es mayor. Algunas características de las tecnologías SMT y THT se listan a continuación:

- SMT
  - Peso y tamaño reducidos
  - Costos de fabricación reducidos
  - Montaje en ambas caras del circuito impreso
  - Reducción de interferencia electromagnética
  - El proceso de construcción de los circuitos es complejo
  - El proceso de montaje en los circuitos impresos es complejo
  - El número de componentes montados es grande
- THT

- Peso y tamaño mayores comparado con SMT
- Montaje en una cara del circuito impreso
- Mayor interferencia electromagnética comparado con SMT
- El proceso de montaje es más sencillo comparado con SMT
- El número de componentes montados es reducido comparado con SMT

Los circuitos impresos pueden contener alguna de estas dos tecnologías ó la combinación de ambas. Así, gracias al avance de la tecnología, actualmente se pueden tener ICs muy poderosos y a muy bajo costo cuando se producen en volumen, sin embargo, ha crecido la necesidad de tener sistemas automáticos de muy alta precisión para la construcción de ICs, de circuitos impresos y del montaje de componentes electrónicos.

1.1. Definición del problema

Las diferencias existentes entre el circuito impreso diseñado y el fabricado, son generados durante la fabricación del circuito. Las diferencias se generan ya que los sistemas automáticos encargados de la producción, cuentan con sistemas mecánicos, los cuales tienen cierta tolerancia en la producción de circuitos impresos. Los ICs fabricados con SMT son quienes se ven más afectados dadas sus características, por lo que estos son los objetos de interés en esta tesis.

El montaje de ICs se realiza sobre sockets, el cual lo lleva a cabo un sistema automático de montaje ó AMM (del inglés Automatic Mechanical Mountage system). Se observa en la figura 1.1 que el diseñador de circuitos impresos es quien envía al fabricante de circuitos impresos un archivo en formato Gerber, el cual contiene la información necesaria para poder fabricar el circuito impreso. El fabricante por su parte utiliza este archivo y fabrica el circuito impreso utilizando sistemas automáticos de alta precisión, sin embargo, debido a que típicamente la producción de circuitos impresos se hace en grandes volúmenes, los sistemas pueden descalibrarse, generando así, diferencias en la colocación de pistas, sockets y/o leyendas sobre el circuito impreso.

En la etapa de montaje de componentes electrónicos sobre el circuito impreso, este puede ó no encontrarse dentro de las normas de calidad debido a las diferencias producidas por el fabricante de circuitos impresos. Estas diferencias provocan retardos y/o pérdidas de producción.

Los sockets de ICs cuentan con un par de puntos de referencia cada uno, las cuales facilitan el montaje de ICs. El montaje por parte del sistema AMM se realiza de la siguiente forma; el sistema realiza una búsqueda sobre el circuito impreso en regiones predeterminadas de los puntos de referencia de los sockets, los cuales son utilizados

por el sistemas para realizar el montaje. La búsqueda de los puntos de referencia se realiza de acuerdo al modelo de circuito impreso y este proceso puede consumir un tiempo considerable que se ve reflejado en la producción. Así, las diferencias generadas en el circuito impreso pueden provocar un aumento de tiempo en la búsqueda de los puntos de referencia, además de que si esta búsqueda no es totalmente correcta, los ICs quedarán ligeramente desplazados durante el montaje.

Lo que se necesita es tener un sistema que permita analizar las regiones de interés de los circuitos impresos, en este caso, analizar los sockets de los ICs y localizar el centroide de los puntos de referencia de cada socket. El conocimiento de estos centroides ofrece varios beneficios como:

- Eliminar el tiempo de búsqueda y posicionar el sistema de manera automáticamente en el punto de montaje
- Evitar que los ICs queden desplazados
- Conocer si el estado de los circuitos impresos se encuentra en los estándares de calidad
- Reducción de los tiempos de producción

La figura 1.2 muestra una sección de un circuito impreso escaneado, el cual contiene el tipo de objetos de interés. Se puede observar que la forma de los objetos que se desea analizar es circular, estos objetos son los que dan la referencia para el montaje de los circuitos SMT.

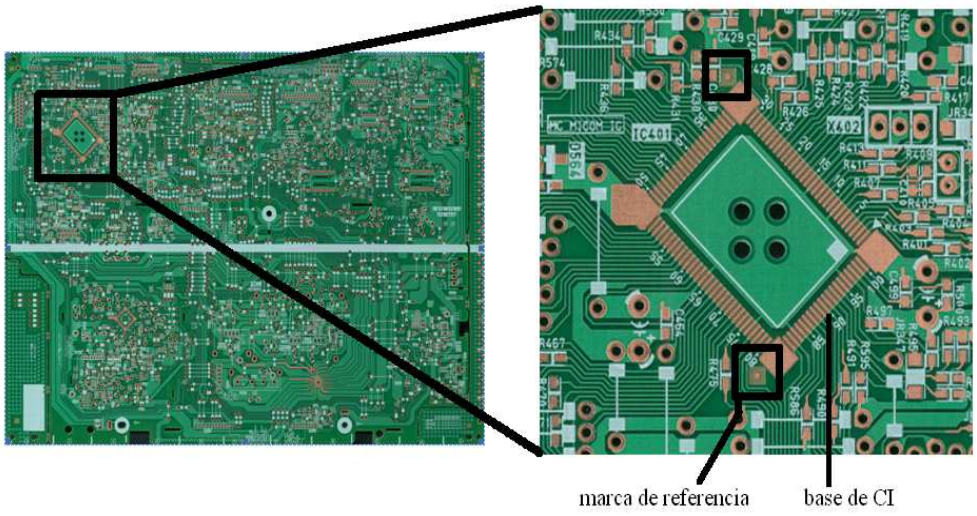


Figura 1.2: Puntos de referencia

1.2. Solución propuesta

La solución propuesta para resolver este problema es el desarrollo de un sistema de software que permita la interacción con el usuario. Las especificaciones del software se muestran en el capitulo 3 (en la pág. 32), las cuales fueron descritas por los ingenieros de SONY Tijuana. El sistema ha sido llamado QCTool (Quality Control Tool), el cual aplica diversas técnicas de Procesamiento Digital de Imágenes para resolver el problema.

Así, el análisis de los circuitos impresos puede realizarse utilizando técnicas de Procesamiento Digital de Imágenes, entre las cuales se encuentran: las operaciones morfológicas, reconocimiento y reconstrucción de objetos, manejo de imágenes a color, entre otras. El marco teórico de las técnicas utilizadas se muestra en el capitulo 2. En esta tesis se ha desarrollado un sistema de software capaz de realizar el análisis de los circuitos impresos, específicamente el análisis de los puntos de referencia de los sockets, utilizando para ello imágenes escaneadas de los circuitos impresos. Este sistema trabaja de manera semiautomática, es decir, se le puede indicar que sección del circuito impreso se desea analizar, teniendo como resultado el valor de los centroides de los puntos de referencia. El sistema ha sido desarrollado con lenguaje C/C++ [3, 4] y utilizando la biblioteca Qt [5, 6, 7].

1.3. Sistema a desarrollar

En el acuerdo realizado con SONY Tijuana se especificó que el sistema debería contar con los siguientes puntos:

- Una Interfaz Gráfica de Usuario (GUI)
- Analizar imágenes de circuitos impresos con resoluciones de 300, 600 y 1200 dpi (puntos por pulgada, dots per inch)
- Analizar los puntos de referencia para cada socket de circuito integrado, los cuales tienen forma circular
- Almacenar los resultados en un archivo de texto
- Funcionamiento en el sistema operativo Windows XP

Así, este sistema tiene como objetivo realizar el cálculo de las posiciones de los circuitos integrados en circuitos impresos, obteniendo también el error cometido por el fabricante entre el circuito impreso fabricado y el diseño original, para lo cual se utilizan los datos del archivo Gerber.

En el desarrollo del sistema se utilizó lenguaje C/C++ para el análisis de las imágenes y la biblioteca Qt para la programación de la interfaz gráfica.

1.4. Estado del arte

Los sistemas computacionales han sido muy utilizado en la industria dado que permiten resolver muchos problemas y automatizar procesos de una manera eficiente y con la mínima intervención del ser humano. En especial, las áreas de gráficos por computadora y procesamiento digital de imágenes han sido utilizados en la producción y control de calidad de diversos processos en la industria [8, 9]. Algunos ejemplos del uso del procesamiento digital de imágenes son:

- industria textil: análisis de texturas y colores
- industria de alimentos: inspección de la calidad de las sustancias utilizadas
- industria electrónica: análisis del montaje de circuito integrados y calidad de circuitos impresos
- medicina: análisis de enfermedades y diagnósticos
- agricultura: análisis del nivel de maduración de frutas y verduras

Estos son solo algunos ejemplos. Esta tesis se enfoca al análisis de circuitos impresos. La industria electrónica requiere de la inspección de los circuitos impresos, montaje de circuitos integrados, así como inspeccionar que no falten componentes electrónicos montados en los circuitos impresos para asegurar la calidad de los productos. Algunos trabajos realizados enfocados también hacia la industria electrónica y específicamente al análisis de circuitos impresos se muestran a continuación.

1.4.1. Análisis y clasificación de imágenes a color para la detección de capacitores y resistencias en tarjetas de montaje superficial

Este trabajo es el más reciente, presentado Jorge Ortíz García en Junio del 2008 [10]. En este trabajo se aborda el problema de la detección de componentes electrónicos, específicamente capacitores y resistencias. El tipo de soldadura utilizada en los circuitos impresos es la soldadura en pasta y en ocasiones la pasta colocada sobre el circuito impreso no es uniforme. Esto puede causar que los componentes electrónicos no queden bien soldados, que queden desplazados ó incluso que no aparezcan en el

circuito impreso.

En este trabajo se presenta una propuesta para detectar algunos de los casos mencionados anteriormente. Para esto, se utilizan diversas técnicas del procesamiento digital de imágenes como operaciones morfológicas, momento invariantes, redes neuronales, histogramas, entre otros.

Con el preprocesamiento, es decir, con las operaciones como erosión, dilatación, abriendo y cerrando, se obtienen imágenes sin ruido provocado por factores externos ó por la digitalización de la imagen. El manejo de imágenes es en escala de grises. Dado que en un circuito impreso se tienen diferentes texturas, con los histogramas de brillo y contraste [11, 9] se puede saber si se esta analizando algún componente electrónico, pista de cobre, la tarjeta ó alguna leyenda. Para detectar si falta o no el elemento buscado, se utilizan los momentos invariantes [12]. Con esta técnica se analizan las características correspondiente a la pasta donde debería de estas montado el componente electrónico y dependiendo de los valores que arrojen los momentos invariantes se puede saber si el componente se encuentra ó no en su lugar. Finalmente, para saber si el componente analizado es una capacitor ó resistencia, se utilizan redes neuronales. La red neuronal utilizada es entrenada con los patrones correspondientes a un capacitor y resistencia, así cuando se analiza un componente, la suma y diferencias de los histogramas de brillo y contraste se pasa como entrada a la red neuronal y al final se indica si el componente es una resistencia ó un capacitor.

El tipo de red neuronal utilizada es de propagación hacia atrás [13]. Todo el sistema fue desarrollado en software utilizando lenguaje C sobre GNU/Linux. Este trabajo fue desarrollado como una herramienta para SONY Tijuana, donde SONY aportó la información requerida para el desarrollo del sistema. Las imágenes utilizadas para el análisis fueron tomadas por un sistema automático, es decir, bajo un ambiente controlado.

1.4.2. Diseño en FPGA de un circuito comparador de imágenes

Este trabajo fue desarrollado por Miguel Ángel Sánchez Martínez en Julio del 2005 [14]. En este trabajo se desarrolla una aquitectura en hardware reconfigurable utilizando un FPGA (Field Programmable Gate Array) para realizar la comparación de circuitos impresos. La idea de realizar un comparador en hardware reconfigurable es porque la mayoría de los sistemas estan hechos en software, lo cual hace difícil cumplir con los requerimiento de tiempo en la industria. El comparador desarrollado en hardware permite obtener resultados en tiempo mucho menores, ya que se pueden realizar las tareas en paralelo.

Este trabajo presenta un sistema para analizar imágenes de circuitos impresos

con pasta depositada, basado en un algoritmo de procesamiento de imágenes por textura implementado en hardware, cuyo propósito es ofrecer un tiempo de cálculo aceptable. El algoritmo extrae una serie de valores de la región analizada y realiza la comparación con lo valores de imágenes de referencia previamente almacenadas y que no cuentan con defectos. Cualquier diferencia entre las series de valores indica una variación entre las imágenes analizadas y por lo tanto, esto implica una falla en el depósito de las soldaduras.

Las imágenes utilizadas son divididas en cuadros disjuntos de 40×40 pixeles. A cada uno de estos segmentos se les aplica una serie de operaciones que permiten construir un histograma de brillo y contraste, los cuales permiten distiguir el tipo de textura analizado. Así, la comparación se basa en histogramas de brillo y contraste de la imagen de prueba y una imagen de referencia. Este sistema puede realizar comparaciones de imágenes de 640×480 pixeles en 24 mseg, lo cual es un buen tiempo para poder realizar el análisis industrial. Aquí, también se hace uso de diversas ténicas de procesamiento de imágenes con hardware.

Para poder validar el sistema, también se desarrolló un software en Java para simular y compara los resultados. Este sistema permite abrir dos imágenes al mismo tiempo; la imagen de prueba y la imagen de referencia. En un solo paso se puede observar detalladamente las diferencias entre las tarjetas y el programa indica en que parte se encuentran las diferencias.

1.4.3. Localización de SMD mediante técnicas de visión artificial implementadas en hardware

Este trabajo fue realizado por Luis Francisco Barbosa en el 2006 [15]. Este sistema fue desarrollado en hardware reconfigurable utilizando un FPGA. El objeto de este sistema es la localización de componentes del tipo SMD (Surface Mount Device). El sistema permite localizar los componentes electrónicos y verificar la posición y orientación del montaje de los componentes electrónicos, asegurando que el circuito impreso se encuentra dentro de los estándares establecidos.

La localización se basa en un umbral de distancia entre patrones. La arquitectura emplea múltiples procesadores y un sistema para la distribución de datos, lo que permite acelerar de forma aceptable la respuesta del sistema. La clasificación de los patrones se realiza utilizando el clasificador del vecino más cercano.

1.5. Descripción de la tesis

En esta sección se muestra una descripción general de la tesis, mostrando las contribuciones y su organización.

1.5.1. Contribuciones

La solución propuesta al problema presentado es la aplicación de diversas técnicas de Procesamiento Digital de Imágenes, para lo cual se ha desarrollado un sistema de software llamado QCTool. Las contribuciones que se tienen con este sistema de software se muestran en la siguiente lista.

- Un sistema de software amigable que utiliza técnicas de Procesamiento Digital de Imágenes para localizar las posiciones de los puntos de referencia de los circuitos integrados en circuitos impresos realizados
- El sistema permite reducir los tiempos de producción al reducir los tiempos de búsqueda que realizan los sistemas mecánicos encargados del montaje de componentes electrónicos
- Reducción de las pérdidas de producción, evitando el montaje de componentes en lugares incorrectos
- Esta tesis también permite mantener y reforzar el vínculo con la industria, lo cual es una parte importante en la formación de los estudiantes
- Se contribuye en la genereación de tecnología para el país, mostrando la capacidad de resolver problemas

1.5.2. Organización de la tesis

Esta tesis está formada por cinco capitulos. Los temas de la tesis están organizados como se muestra en la siguiente lista.

Capitulo 1, se da una introducción y se muestran las tecnologías utilizadas en el diseño y fabricación de circuitos impresos, así como los problemas que se presentan y el problema que se trata de resolver utilizando el procesamiento digital de imágenes. Se da una breve introducción al procesamiento de imágenes y a los sistemas de inspección por computadora. Finalmente se muestra el estado del arte relacionado con el tema de tesis.

Capitulo 2, se muestra detalladamente el marco teórico. Se habla sobre el procesamiento digital de imágenes así como de las diversas técnicas utilizadas en esta tesis. Se muestran algunos ejemplos básicos de la aplicación de las técnicas utilizadas.

Capitulo 3, en este capitulo se realiza el diseño del sistema, tanto de la implementación de los diferentes algoritmos como de la interfaz gráfica. Se desarrolla el modelado del sistema utilizando el modelo de cascada, así como también se muestra el diagrama de objetos y el diagrama de señales y ranuras (signals - slots) que usa Qt.

Se muestran también los bloques de código más esenciales en el desarrollo del sistema.

Capitulo 4, se realiza una explicación a nivel de usuario para comprender el funcionamiento del sistema, desde que se inicia hasta que se finalizan las operaciones. También se muestran los resultados obtenidos utilizando el sistema.

Capitulo 5, finalmente, aquí se describen las conclusiones generales de la tesis y posible trabajo a futuro que se podría realizar para mejorar el sistema.

Capítulo 2

Marco teórico

En este capítulo se presenta una introducción al PDI y sus aplicaciones, así como el marco teórico utilizado en el desarrollo del sistema de software propuesto, llamado QCTool (Quality Control Tool). El procesamiento de imágenes trata sobre el análisis de escenas así como la reconstrucción de modelos en 2D o 3D utilizando varias imágenes. Tiene como subáreas varios procesos como: el mejoramiento de imágenes, detección y reconocimiento de patrones, análisis de escenas y visión por computadora [16]. El sistema desarrollado utiliza diversas técnicas del PDI, entre las cuales se tiene al manejo de modelos de color, operaciones morfológicas, extracción de objetos, reconocimiento y reconstrucción de objetos.

2.1. Procesamiento digital de imágenes

Desde el inicio de la ciencia, la observación visual ha jugado un papel muy importante. En los primero experimentos se tenían descripciones verbales y dibujos hechos a mano. El siguiente gran paso fue la invención de la fotografía, lo que permitió tener resultados que eran documentados con mucha más información así como también permitían realizar descripciones más rápido. Hoy en día, el rápido progreso en la tecnología ha permitido que las computadoras personales cuenten con el poder suficiente para procesar los datos de imágenes. En consecuencia, el procesamiento de imágenes se ha expandido rápidamente desde pequeñas áreas especializadas hasta ser una herramienta científica, como puede ser la aplicación virtualmente en las ciencias naturales y disciplinas técnicas [8].

Una imagen puede ser representada como una función bidimensional $f(x, y)$, donde x, y son coordenadas espaciales y la amplitud de f, para cualquier par de coordenadas (x, y), es llamada intensidad o nivel de gris de la imagen en ese punto, la figura 2.1 muestra el arreglo bidimensional correspondiente a una imagen digital. Cuando x, y los valores de amplitud de f son finitos, cantidades discretas, se dice que se tiene una imagen digital [9]. El procesamiento digital de imágenes se realiza mediante el uso

de la computadora. Así, una imagen digital esta compuesta por un número finito de elementos, cada uno de los cuales tiene una posición y valor particular. Cada elemento en una imagen digital es conocido como pixel.

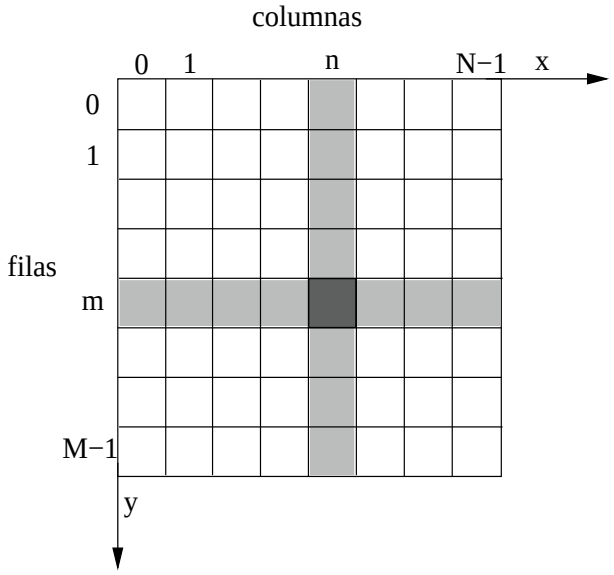


Figura 2.1: Representación del arreglo de valores de una imagen digital

Así, una imagen digital se puede representar como arreglos de valores, donde un valor es una colección de números que describen los atributos de un pixel en una imagen. Estos valores son números enteros entre 0 y 255. Como se ha mencionado, estos números representan la intensidad de un pixel de una imagen en escala de grises o la intensidad de un componente en una imagen de color. Las dimensiones del arreglo correspondiente a una imagen son el ancho y alto, y el número de bits asociados con cada pixel del arreglo es la profundidad [4].

2.1.1. Aplicaciones del PDI

El procesamiento digital de imágenes se ha convertido en área de investigación importante debido al rápido desarrollo de las tecnologías. Sus aplicaciones se extienden a la visión industrial, las imágenes médicas, las imágenes satelitales, el video y el cine digitales, la video vigilancia, el control de tráfico, seguimiento de objetos en movimiento, visión estéreo, reconstrucción tridimensional, restauración e interpretación de imágenes satelitales, reconocimiento de objetos, compresión de imágenes, entre muchas otras.

La aplicación del procesamiento digital de imágenes en la visión industrial se caracteriza por el uso de imágenes electrónicas. El uso en la industria comúnmente es

en el área de producción, lo cual sustituye las funciones visuales humanas. Provee sistemas que incorporan técnicas que realizan tareas basadas en una funcionalidad equivalente a la visión. Así, su objetivo final es controlar procesos, la calidad, las máquinas y robots [17].

Actualmente, con los avances tecnológicos en los sistemas de cómputo (alta velocidad, incremento en la memoria, disminución de los costos, etc.), las diversas técnicas del procesamiento de imágenes, reconocimiento de patrones y la inteligencia artificial han permitido desarrollar sistemas para la industria tanto de software como de hardware en el control de calidad de la producción [18].

La siguiente lista muestra algunas de las áreas de aplicación del procesamiento digital de imágenes [8, 18, 15]:

- **Medicina**: análisis de enfermedades, sangre, DNA, ayuda en diagnósticos, etc.
- **Milicia**: detección y seguimiento de objetos, análisis de terreno, etc.
- **Robótica**: detección de obstáculos, sistemas de visión, etc.
- **Agricultura**: análisis de plantaciones mediante el uso de imágenes digitales
- **Identificación y seguridad**: identificación de huellas dactilares, reconocimiento de caras, etc.
- **Redes de comunicaciones**: compresión de imágenes, teleconferencia, etc.
- **Astronomía**: búsqueda, reconocimiento y seguimiento de objetos en el espacio
- **Industria**: clasificación, búsqueda, control de calidad, procesos, máquinas y robots

2.1.2. Componentes de un sistema de procesamiento de imágenes

Un sistema de propósito general para la adquisición y procesamiento de imágenes consiste de cuatro componentes esenciales [8]:

- Un sistema de adquisición de imágenes. En el caso más simple, este puede ser una cámara CCD o una videograbadora
- Un dispositivo para la conversión de las señales eléctricas de la adquisición de la imagen en una imagen digital que pueda ser almacenada
- Una computadora personal con el poder de procesamiento de imágenes

- Un software de procesamiento de imágenes que proporcione las herramientas necesarias para la manipulación y análisis de imágenes

Estos son los componentes utilizados en cualquier sistema de adquisición en la industria. Es claro que la calidad de la imagen dependerá de la calidad de los componentes utilizados, lo cual también influye en la precisión de los resultados.

2.2. Inspección visual

Inicialmente, la inspección visual para el control de calidad era realizada por un ser humano. Aunque el ser humano se adapta a situaciones nuevas de manera muy rápida, es más lento, se cansa y se aburre, lo que puede provocar errores de apreciación en la inspección [14, 18, 19]; mientras que una máquina nunca se cansa ni se aburre. En la mayoría de los casos el ser humano que realiza la inspección, dará diferente resultados en cada inspección que realice [20].

El procesamiento digital de imágenes permite realizar la inspección de manera automática, utilizando la computadora como interfaz entre el usuario y algún proceso o como en este caso, verificar los puntos de interés de los circuitos impresos seleccionados por un usuario.

Un sistema de inspección visual generalmente cuenta con los cuatro componentes básicos de un sistema de procesamiento de imágenes mencionado anteriormente. Durante la inspección se realizan las tareas básicas del procesamiento digital de imágenes [15, 19], estas tareas son:

- Adquisición de la imagen
- Preprocesamiento
- Segmentación
- Extracción de características
- Reconocimiento de objetos

En la siguiente sección se describen los algoritmos utilizando en el sistema desarrollado para la detección y reconocimiento de los objetos de interés.

2.3. Modelos de color

El uso de color en el procesamiento de imágenes es muy utilizado debido a que el color es un poderoso descriptor que simplifica la identificación y extracción de objetos de las escenas. El procesamiento de imágenes a color se divide en dos grandes

áreas: procesamiento con color real y procesamiento con pseudocolor. En la primera categoría, las imágenes en cuestión, típicamente son tomadas utilizando un sensor de color real como una cámara de TV de color o un escáner de color. En la segunda categoría, el problema es la asignación de un color a un rango de intensidades [9].

A pesar del proceso desarrollado por el cerebro humano en la percepción e interpretación del color, este es un fenómeno fisiopsicológico que aún no se comprende totalmente, la naturaleza física del color puede ser expresada en forma básica. El color que los humanos y otros animales perciben en un objeto, esta determinado por la luz reflejada en el objeto.

El propósito de un modelo de color (también llamado espacio de color o sistema de color) es facilitar la especificación de los colores en algún estándar. En esencia, un modelo de color es una especificación de un sistema coordenado y un subespacio dentro de este sistema, donde cada color es representado por un punto [9]. La mayoría de los modelos de color usados hoy en día, están orientados ya sea hacia el hardware (monitores de color e impresoras) o hacia aplicaciones de manipulación de imágenes. En términos del procesamiento digital de imágenes, los modelos de color orientados a hardware más comúnmente usados son el modelo RGB (Red, Green, Blue), el cual es usado en monitores de color y en video cámaras; el modelo CMY (Cyan, Magenta, Yellow) y CMYK (Cyan, Magenta, Yellow, Black), los cuales son utilizados en impresoras de color; y el modelo HSI (Hue, Saturation, Intensity), el cual corresponde con la forma en como los humanos describen e interpretan el color. Existen muchos modelos de color usados hoy en día debido a que la ciencia del color es un campo en el que existen muchas áreas de aplicación.

2.3.1. Modelo de color RGB

Existen muchas maneras en la que cuantitativamente se puede especificar un color. La forma más directa es utilizar los valores de los colores rojo, verde y azul en el rango [0, 1]. A esta convención se le llama modelo RGB [21]. En el modelo RGB, cada color aparece en su componente espectral primario de rojo, verde y azul. La formación de los colores con el modelo RGB se lleva a cabo con la combinación de los tres porcentajes de sus componentes, como se muestra en la figura 2.2.

Este modelo esta basado en un sistema coordenado cartesiano. El modelo RBG se puede representar con el cubo de colores mostrado en la figura 2.3, en donde los valores de RGB se encuentran en las tres esquinas, el color negro se encuentra en el origen del cubo y el color blanco en la esquina más alejada.

Las imágenes representadas en el modelo de color RGB consisten de tres componentes o tres imágenes, una para cada color primario. Por ejemplo, en un monitor

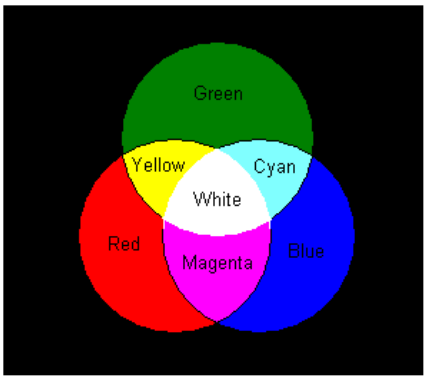


Figura 2.2: Combinación de los componentes RGB en la formación de colores

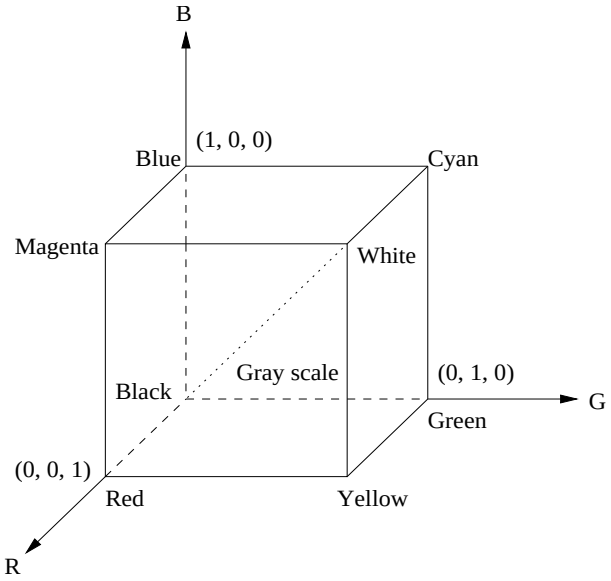


Figura 2.3: Cubo de colores del modelo RGB

RGB, estas tres imágenes se combinan en una pantalla de fósforo para producir una imagen de color compuesta.

2.3.2. Modelos de color CMY

En el modelo RGB, se manejan los colores primario rojo, verde y azul (Red, Green, Blue), mientras que en el modelo CMY se manejan los colores secundarios cian, magenta y amarillo (Cyan, Magenta, Yellow). La mayoría de los dispositivos que depositan pigmentos de color en papel como impresoras de color o copiadoras, reciben datos de entrada en el modelo CMY o alternativamente se puede realizar una conversión del modelo RGB al modelo CMY. Esta conversión se realiza con la operación siguiente:

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{2.1}$$

donde, se asume que los valores de los colores están normalizados en el rango [0, 1]. La operación de conversión muestra que los valores de RGB pueden ser obtenidos de un conjunto de valores CMY, restando los valores individuales de CMY de 1.

2.3.3. Modelo de color HSI

Crear colores en los modelos de color RGB y CMY, así como pasar de un modelo a otro, es un proceso directo. Estos modelos de color son ideales para implementaciones en hardware. Desafortunadamente, los modelos RGB y CMY no funcionan muy bien en la descripción e interpretación de los colores comparados con la forma en que los humanos lo hacen. Por ejemplo, una persona no describe el color de un automóvil dando porcentajes de cada uno de los colores primario que componen el color del automóvil. De hecho, cuando una persona observa una imagen a color ni siquiera piensa que esta los colores están compuestos por la combinación de tres colores.

Cuando los humanos observan un objeto de color, hacen una descripción del color en base al color puro o base, saturación y brillo (hue, saturation, brightness). El modelo de color que funciona de manera similar a como los humanos observan el color es llamado modelo HSI (Hue, Saturation, Intensity). El modelo HSI es una formalización del sistema de color desarrollado por Munsell y comúnmente usado por artistas [22, 23]. Los componentes del modelo HSI se describen a continuación:

- Hue: es un atributo que describe un color puro, por ejemplo amarillo, naranja o rojo
- Saturation: proporciona la medida en que un color puro esta diluido en el color blanco
- Intensity: proporciona el nivel de brillo

El modelo HSI desacopla el componente intensity de los componentes hue y saturation, ya que estos son quienes contienen la información del color en una imagen. Como resultado se tiene que el modelo HSI es una herramienta ideal para el desarrollo de algoritmos de procesamiento de imágenes, basados en descripciones del color, eliminando algunos de los problemas que se presentan cuando se utilizan imágenes en escala de grises. Para comprender mejor el funcionamiento del modelo HSI, se

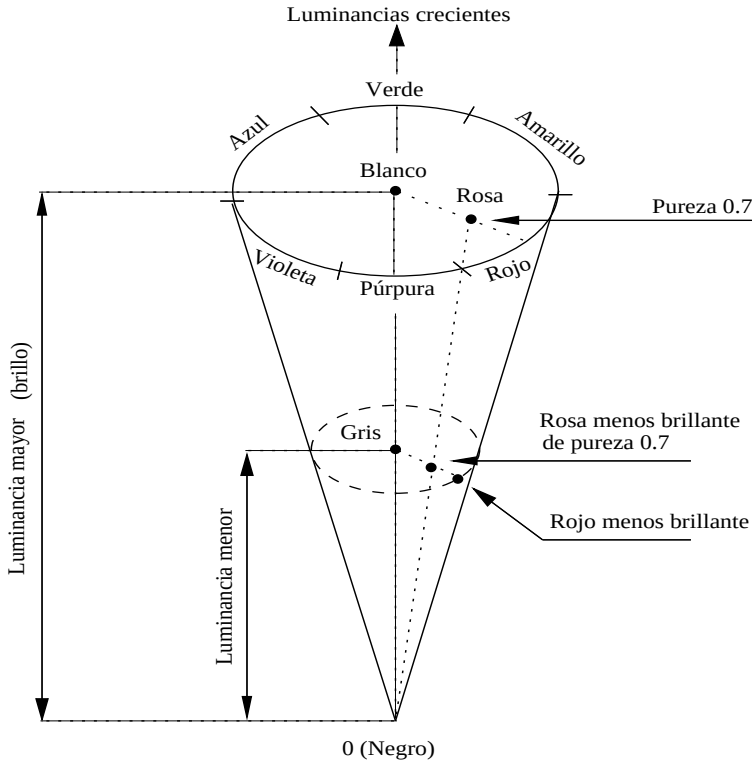


Figura 2.4: Cono del colores del modelo HSI

muestra en la figura 2.4 el cono de colores.

En el cono de colores se tiene la altura, la base y el radio de la base del cono. La base se refiere al componente hue, es decir, al color puro del pixel de una imagen analizada, el cual puede ser amarillo, verde, azul, violeta, purpura o rojo. El valor del radio se refiere al componente saturation, que indica el nivel de pureza del color, es decir, que tan diluido se encuentra el color puro en el color blanco. La altura se refiere a la intensidad, lo cual indica el nivel de brillo.

2.3.4. Transformación de RGB a HSI

Dada una imagen en formato RGB, el componente Hue es calculado usando la siguiente ecuación:

$$r = \frac{R}{R+G+B} \qquad g = \frac{G}{R+G+B} \qquad b = \frac{B}{R+G+B} \tag{2.2}$$

Los componentes Hue, Saturation e Intensity están dados por:

$$h = \begin{cases} \cos^{-1}\left\{\frac{0.5\cdot[(r-g)+(r-b)]}{[(r-g)^2+(r-b)(g-b)]^{\frac{1}{2}}}\right\} & \text{if } b \leq g, \quad h \in [0,\pi] \\ 2\pi - \cos^{-1}\left\{\frac{0.5\cdot[(r-g)+(r-b)]}{[(r-g)^2+(r-b)(g-b)]^{\frac{1}{2}}}\right\} & \text{if } b > g, \quad h \in [\pi,2\pi] \end{cases} \tag{2.3}$$

$$s = 1 - 3 \cdot \text{mín}(r,g,b) \qquad s \in [0,1] \tag{2.4}$$

$$i = \frac{(R+G+B)}{3 \cdot 255} \qquad i \in [0,1] \tag{2.5}$$

En esta tesis se utiliza el modelo HSI para el manejo de las imágenes a color, ya que como se ha dicho, el color es un poderoso descriptor que proporciona mucha información. Las imágenes originalmente se obtienen en formato RGB y se realiza la transformación a formato HSI utilizando las ecuaciones (2.2), (2.3), (2.4) y (2.5). El paso siguiente es realizar algún proceso que permita mejorar la imagen, ya que por diversos factores se pueden tener imágenes degradadas.

2.4. Operaciones morfológicas

La palabra morfología denota una rama de la biología que trata con la forma y estructura de plantas y animales. La morfología en el procesamiento de imágenes es una herramienta para la extracción de componentes de una imagen [9]. Las operaciones más comunes son la erosión y la dilatación. Estas operaciones son comúnmente aplicadas sobre imágenes binarias, por lo cual, la dilatación tiende a aumentar el tamaño los objetos mientras que la erosión tiende a adelgazar o incluso a eliminar objetos pequeños [24].

La erosión disminuye el tamaño de los objetos, eliminando pixeles alrededor del objeto. La dilatación aumenta el tamaño de los objetos, agregando pixeles alrededor del objeto. Existen dos técnicas generales para la dilatación y erosión, la técnica que utiliza umbrales predefinidos y la técnica de máscaras [25, 3].

2.4.1. Dilatación

Para los conjuntos A y B en Z^2, donde Z es el conjunto de los números enteros, la dilatación de A por B, denotada como $A \oplus B$, esta definida como:

$$A \oplus B = \{z \mid (B)_z \cap A \neq 0\} \tag{2.6}$$

A es el conjunto de valores originales y B es conocido como elemento estructural o máscara utilizado en la dilatación. El elemento B se desplaza por el conjunto A, ensanchando el conjunto de valores, es decir, la matriz B se desplaza sobre la matriz A, obteniendo un ensanchamiento del objeto en la matriz A. La figura 2.5 muestra un ejemplo de su funcionamiento [9].

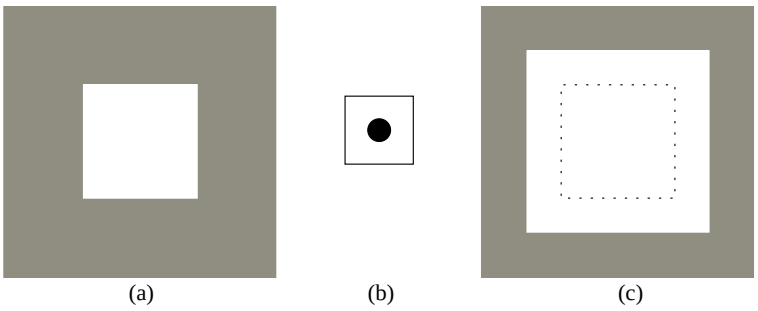


Figura 2.5: Dilatación; (a) imagen original; (b) elemento estructural; (c) imagen dilatada

2.4.2. Erosión

Para los conjuntos A y B en Z^2, donde Z es el conjunto de los números enteros, la erosión de A por B denotada como $A \ominus B$ esta definida como:

$$A \ominus B = \{z \mid (B)_z \subseteq A\} \tag{2.7}$$

A es un conjunto de valores originales y B es conocido como elemento estructural o máscara que definirá el funcionamiento de la erosión. El elemento B se desplaza sobre todo el conjunto A adelgazando el conjunto de valores, es decir, la matriz B se desplaza sobre la matriz A. En la figura 2.6 muestra un ejemplo de su funcionamiento [9].

2.4.3. Abriendo y cerrando

Las operaciones morfológicas abriendo y cerrando ayudan en la separación y unión de objetos. Estos son operadores muy útiles y que son combinación de la erosión y dilatación. La operación abrir se utiliza en la separación de objetos que se encuentran muy cerca, en la separación de objetos que se tocan y que no debería hacerlo y en

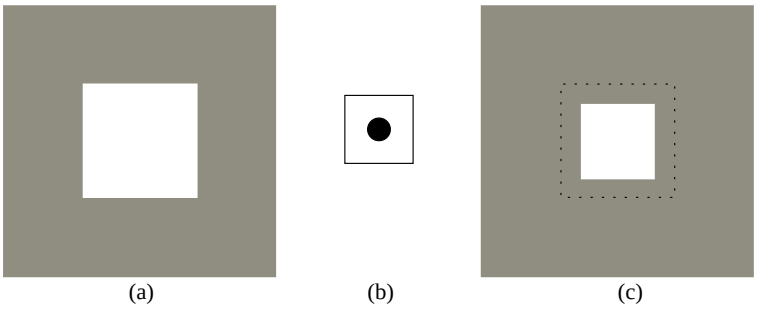


Figura 2.6: Erosión; (a) imagen original; (b) elementos estructural; (c) imagen erosionada

el ensanchamiento de huecos dentro de objetos. La operación cerrar une objetos que están separados y que no deberían estarlo y rellena huecos dentro de objetos [4]. La operación abrir involucra una o más erosiones seguida de una dilatación. Mientras se aplique un mayor número de erosiones se tendrá una mayor separación entre los objetos. La operación cerrar involucra una o más dilataciones seguidas de una erosión. Mientras mayor sea el número de dilataciones se tendrá objetos fuertemente unidos. Las figuras 2.7 y 2.8 muestran el uso de las operaciones abrir y cerrar respectivamente.

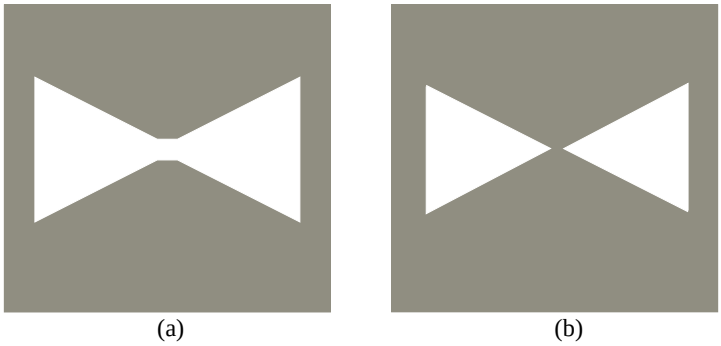


Figura 2.7: Operación abriendo

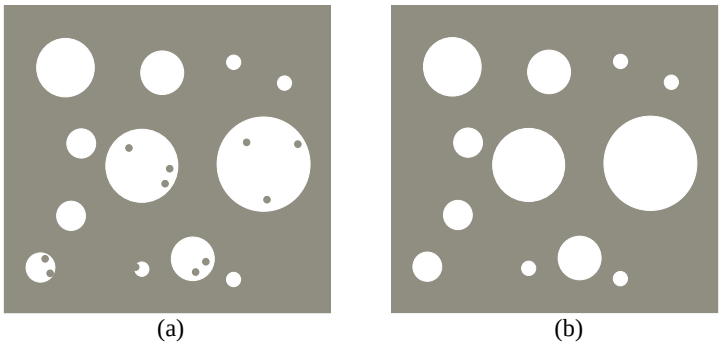


Figura 2.8: Operación cerrando

2.4.4. Aplicación de operaciones morfológicas en círculos

Como se explicó en el capítulo 1, los objetos que se desean analizar en los circuitos impresos son de forma circular. Aplicando las operaciones morfológicas se puede eliminar ruido, rellenar huecos y/o separar objetos. En la figura 2.9a muestra algunas imágenes binarias que presentan afectación por ruido y algunas presentan también huecos en los círculos. La figura 2.9b muestra nuevamente la imágenes después de la aplicación de las operaciones morfológicas y se puede observas que la mejoria ha sido bastante, el ruido ha sido eliminado y los huecos rellenados.

(a)

(b)

Figura 2.9: Uso de operaciones morfológica: (a) círculos afectados con ruido; (b) aplicación de operaciones morfológicas en imágenes binarizadas de los círculos de las referencias para insertar un CI

2.5. Extracción de objetos en imágenes binarias

Una forma de realizar la clasificación de los objetos de una imagen, es conociendo las características de cada uno de los objetos, a estas características se le conoce también como vector de características. La extracción de objetos nos permite separar los objetos de una imagen y obtener el vector de características individualmente. Un algoritmo para analizar objetos individuales resulta mucho más sencillo que uno para analizar la imagen completa, lo cual también se ve reflejado en el costo computacional.

El algoritmo para la extracción de objetos utilizado en esta tesis parte de la idea del algoritmo de Pavlidis [26], donde una forma o figura compuesta por pixeles de un mismo tono, se considera compuesta por tramos. El algoritmo usado fue programado en la biblioteca SCimagen [27]. Un tramo es una secuencia de pixeles de un mismo tono definido por su punto extremo más a la izquierda, un ejemplo de esto se muestra en la figura 2.10.

Figura 2.10: Un tramo definido por su punto izquierdo

También, cada tramo puede estar conectado con otros tramos arriba y abajo. La figura 2.11 muestra los casos de conectividad entre tramos.

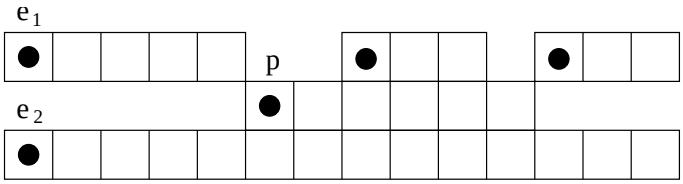


Figura 2.11: Casos posibles de conectividad para un tramo dado p

La descripción del algoritmo de extracción de objetos tomado del articulo [27] se muestra en los siguientes puntos:

- El algoritmo cuenta con una función llamada ligas, que realiza las operaciones sobre el tramo llamado p, como el de la figura 2.11.
- Cuenta el número de tramos arriba y debajo de p
- Si en cualquiera hay más de un tramo, se ponen en la pila. La función regresa los dos puntos extremos de los tramos extremos, en este caso regresa los puntos e_1 y e_2 de la figura 2.11

La extracción de un objeto se realiza rellenándolo con otro tono. La inicialización del algoritmo toma cualquier pixel que forma parte del tramo y posteriormente se obtiene el pixel más a la izquierda del tramo usando una función llamada izquierda [26].

El algoritmo 1 [27] muestra la forma de realizar la extracción de objetos.

La función rellenar, cambia el tono del objeto en extraído. Esto se repite con cada uno de los objetos en la imagen. Así, la entrada del algoritmo de extracción es una imagen con N objetos y la salida son N imágenes con un objeto cada imagen. La imágenes utilizadas por este algoritmo son en blanco y negro. La figura 2.12 muestra un ejemplo de la extracción de objetos.

Una vez que los objetos han sido extraídos, se puede realizar el análisis individual para extraer su vector de características. Algunos ejemplos de características comúnmente utilizadas son el área, el diámetro y los momentos invariantes. El reconocimiento de objetos basado en el vector de características se trata en la siguiente

Algorithm 1: algoritmo de extracción de objetos (de [27])

```
dirección = ARRIBA;
while pop( p ) do
    while 1 do
        ligar( p );
        if e1 == 0 AND e2 == 0 then
            if el tramo no está rellenado then
                rellenar( p );
            break;
        if e1 == 0 then
            dirección = ABAJO;
        if e2 == 0 then
            dirección = ARRIBA;
        if e1 > 0 AND e2 > 0 then
            if dirección == ARRIBA then
                push( e2 );
            else
                push( e1 );
        rellenar( p );
        p = e_direccion;
```

sección.

2.6. Reconocimiento de objetos

En esta sección se introducen algunos conceptos del reconocimiento de patrones así como técnica de momentos invariantes utilizada para esta tesis.

Un patrón es un arreglo de descriptores obtenidos de los objetos. La palabra característica es comúnmente usada en la literatura de reconocimiento de patrones para denotar a un descriptor. Una clase de patrón en una familia de patrones que comparten propiedades comunes. Las clases de patrones son denotadas como w_1, w_2, ..., w_W, donde W es el número de clases. El reconocimiento de patrones realizado por computadora involucra diversas técnicas para realizar la asignación de los patrones a sus respectivas clases automáticamente y con la mínima intervención posible por parte del ser humano [9].

Los patrones pueden ser representados como vectores denotado por la letra x de la misma manera como se muestra en (2.8).

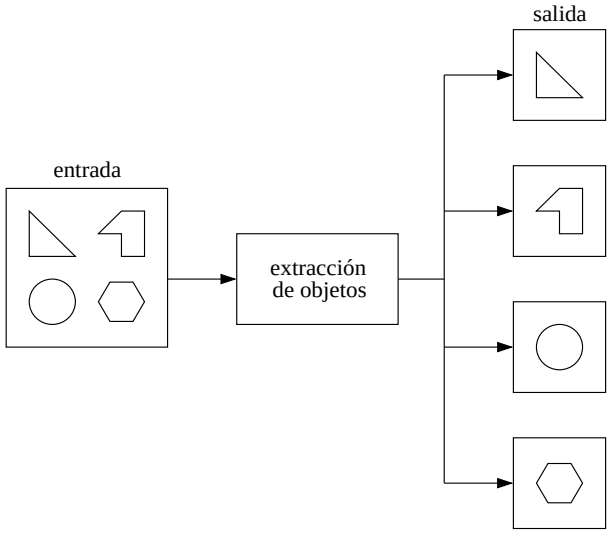


Figura 2.12: Extracción de objetos

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \tag{2.8}$$

donde cada componente x_i representa el i-ésimo descriptor y n es el número total de descriptores asociados con el patrón. Los vectores de patrones son representados como columnas (matrices de $n \times 1$). Así, un patrón puede ser expresado en la forma mostrada en (2.8) o en la forma equivalente $\mathbf{x} = (x_1, x_2, \ldots, x_n)^T$, donde T indica transposición.

2.6.1. Momentos invariantes de Hu

El reconocimiento visual de patrones y caractéres independientes de la posición, tamaño y orientación, es uno de los objetivos en la muchas de las investigaciones. Para alcanzar este objetivo, se deben utilizar métodos insensibles a las transformaciones geométricas. Una técnica muy utilizada en la extracción de características es la de momentos invariantes. La teoría de los momentos invariantes para el reconocimiento de patrones fue introducida por Ming-Kuei Hu [12], por medio del teorema fundamental de momentos invariantes.

Considerese un objeto geométrico S en el espacion X. Se supone la existencia de un grupo de transformaciones admisibles G que actúa en el espacio X. Un invariante escalar de un objeto S es una cantidad que no cambia su valor cuando el objeto S sufre cualquiera de las transformaciones admisibles, tales como rotación, traslación

o escalamiento. Supongase que el objeto S tiene invariantes escalares $I_1, I_2, \ldots, I_n$. Considerese que se obtiene el objeto S' al transformar apropiadamente el objeto S usando transformaciones admisibles, los valores de estos invariantes escalares deben ser indénticos [28].

El uso de los momentos invariantes en imágenes de dos dimensiones es muy utilizado en el reconocimiento de patrones, ya que son muy útiles dado que se puede realizar el reconocimiento de un objeto aún sin importar si ha sufrido transformaciones geométricas como rotación, traslación y/o escalamiento. Los momentos invariantes han sido aplicados ampliamente en el reconocimiento de patrones, por ejemplo, en la localización automática de objetos 2D y 3D [29], en la autenticación de imágenes digitales [30], entre otros.

Los momentos geométricos $m_{p,q}$ de una imagen en escala de grises $f(x, y)$ se define como:

$$m_{p,q} = \sum_{x=1}^{N} \sum_{y=1}^{M} x^p y^p f(x, y) dxdy \tag{2.9}$$

dondo, p y q son los índices de una imagen digital de tamaño $N \times M$. De la ecuación (2.9) se obtienen los siguientes momentos centrales:

$$\mu_{p,q} = \sum \sum (x - \bar{x})^p (y - \bar{y})^p f(x, y) dxdy \tag{2.10}$$

donde:

$$\bar{x} = \frac{m_{1,0}}{m_{0,0}} \qquad \bar{y} = \frac{m_{0,1}}{m_{0,0}} \tag{2.11}$$

$(\bar{x}, \bar{y})$ es el centroide del objeto. Los momentos centralizados se definen como:

$$n_{p,q} = \frac{\mu_{p,q}}{{\mu_{0,0}}^{\gamma}} \qquad \gamma = \frac{p + q + 2}{2} \tag{2.12}$$

En [12], Hu propone un conjunto de momentos invariantes a las transformaciones geométricas como la rotación, traslación y escalamiento. Este conjutno está formado por siete funciones denotadas por ϕ.

$$
\begin{aligned}
\phi_1 &= n_{2,0} + n_{0,2} \\
\phi_2 &= (n_{2,0} + n_{0,2})^2 + 4n_{1,1}^2 \\
\phi_3 &= (n_{3,0} + n_{1,2})^2 + (n_{0,3} - 3n_{2,1})^2 \\
\phi_4 &= (n_{3,0} + n_{1,2})^2 + (n_{0,3} + n_{2,1})^2 \\
\phi_5 &= (n_{3,0} - 3n_{1,2})(n_{3,0} + n_{1,2}) \cdot [(n_{3,0} + n_{1,2})^2 - 3(n_{2,1} + n_{0,3})^2] + (n_{0,3} - 3n_{2,1}) \cdot \\
&\quad (n_{0,3} - n_{2,1}) \cdot [(n_{0,3} + n_{2,1})^2 - 3(n_{2,0} + n_{0,2})^2] \\
\phi_6 &= (n_{2,0} + n_{0,2}) \cdot [(n_{3,0} + n_{1,2})^2 - (n_{0,3} + n_{2,1})^2] + 4n_{1,1}(n_{3,0} + n_{1,2})(n_{0,3} + n_{2,1}) \\
\phi_7 &= (3n_{2,1} - n_{0,3})(n_{3,0} + n_{1,2}) \cdot [(n_{0,3} + n_{2,1})^2 - 3(n_{2,1} + n_{0,3})^2] + (n_{3,0} - 3n_{2,1}) \cdot \\
&\quad (n_{2,1} + n_{0,3}) \cdot [(n_{0,3} + n_{2,1})^2 - 3(n_{3,0} + n_{1,2})^2]
\end{aligned}
$$

Una vez que se han obtenido los momentos se puede establecer la clase para este tipo de patrones. En esta tesis se utilizó el primer momento ϕ_1 como patrón. Como prueba se tomó varias veces el mismo objeto realizando algunas transformaciones geométricas, observando que la variación de los momentos era casi despreciable. El siguiente paso es realizar la clasificación de los objetos tomando en cuenta los patrones. Uno de los métodos más utilizados es el clasificador de mínima distancia.

2.6.2. Clasificador de mínima distancia

Las técnicas de reconocimiento basadas en coincidencias representan a cada clase por un vector de patrones prototipo. Un patrón desconocido es asignado a la clase que se encuentre más cerca en términos de un métrica predefinida. La forma más simple es el clasificador de mínima distancia, lo cual implica el cálculo de la distancia Euclideana entre el patrón desconocido y cada uno de los vectores prototipos. Este clasificados toma la distancia más corta como decisión.

Supongase que definimos el prototipo para cada clase de patrones de la siguiente forma:

$$m_j = \frac{1}{N_j} \sum_{x \in w_j} x_j \qquad j = 1, 2, \ldots, W \tag{2.13}$$

donde N_j es el número de patrones de vectores para la clase w_j y la suma se realiza sobre los vectores. W es el número de clases de patrones. Una forma de determinar la pertenencia a una clase de un vector de patrones x es asignandolo a la clase que se encuentre más cerca. Utilizando la distancia Euclideana para determinar la cercanía de un patrón con una clase reduce el problema de realizar el cálculo de mediciones de distancias:

$$d_j(x) = \|x - m_j\| \qquad j = 1, 2, \ldots, W \tag{2.14}$$

donde $\|a\| = (a^T a)^{\frac{1}{2}}$ es la norma Euclideana. Entonces, se realizar la asignación del vector de patrones x a la clase w_i si $d_i(x)$ es la distancia mínima. La distancia mínima implica la mejor coincidencia con una clase.

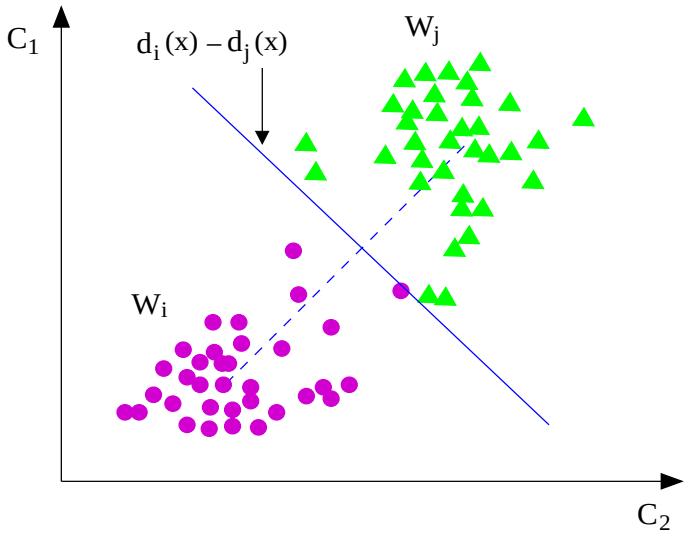


Figura 2.13: Ejemplo del clasificación de mínima distancia

En la figura 2.13 se ilustra la forma de funcionamiento del clasificador de mínima distancia y que utiliza la distancia Euclideana. En esta tesis, como ya se ha mencionado, se utiliza el primer momento invariante ϕ_1 como patrón. Dado que solo se requiere conocer un tipo de objeto, en este caso se ha establecido un rango para determinar si el objeto pertenece o no a una clase.

2.7. Reconstrucción de círculos

La imagen captada por un dispositivo electrónico u óptico puede presentar degradaciones producidas por ruido en el sensor, borrosidad debido a un mal enfoque de la cámara, movimiento de la cámara o la resolución de la cámara.

El objetivo de la reconstrucción es mejorar la calidad de la imagen que se ha visto afectada por algunas de las razones mencionadas. Existen varias técnicas utilizadas en la reconstrucción de objetos para mejorar la calidad de la imagen. En este caso, el objetivo es reconstruir círculos. En artículo de Umbach [31] se presentan cinco algoritmos para la reconstrucción de círculos, siendo el algoritmo llamado Modified Least-Square (MLS), el algoritmo más estable y de solución cerrada. Este algoritmo trabaja con las posiciones de los pixeles del perímetro del círculo. El algoritmo MLS

además de la reconstrucción permite calcular el centroide tomando encuenta también los pixeles del perímetro.

El centroide del círculo se calcula de la siguiente forma:

$$a_M = \frac{DC - BE}{AC - B^2} \tag{2.15}$$

$$b_M = \frac{AE - BD}{AC - B^2} \tag{2.16}$$

donde:

$$A = n\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2 \tag{2.17}$$

$$B = n\sum_{i=1}^{n} x_i y_i - \left(\sum_{i=1}^{n} x_i\right)\left(\sum_{i=1}^{n} y_i\right) \tag{2.18}$$

$$C = n\sum_{i=1}^{n} y_i^2 - \left(\sum_{i=1}^{n} y_i\right)^2 \tag{2.19}$$

$$D = 0.5\left\{n\sum_{i=1}^{n} x_i y_i^2 - \left(\sum_{i=1}^{n} x_i\right)\left(\sum_{i=1}^{n} y_i^2\right) + n\sum_{i=1}^{n} x_i^3 - \left(\sum_{i=1}^{n} x_i\right)\left(\sum_{i=1}^{n} x_i^2\right)\right\} \tag{2.20}$$

$$E = 0.5\left\{n\sum_{i=1}^{n} y_i x_i^2 - \left(\sum_{i=1}^{n} y_i\right)\left(\sum_{i=1}^{n} x_i^2\right) + n\sum_{i=1}^{n} y_i^3 - \left(\sum_{i=1}^{n} y_i\right)\left(\sum_{i=1}^{n} y_i^2\right)\right\} \tag{2.21}$$

El centroide del círculo reconstruido está representado por a_M y b_M. Existen también varias forma de obtener el perímetro de un círculo. En esta tesis, el perímetro se obtiene realizando la diferencia entre un objeto S y el objeto S erosionado, lo cual nos da un objeto S' que es el perímetro del círculo. La figura 2.14 muestra un ejemplo de la reconstrucción de círculos.

Así, con el algoritmo MLS, el cálculo del centroide se puede realizar con mayor precisión. Como se ha mecionado, este algoritmo es estable y de solución cerrada, de este modo, es posible reconstruír un círculo con solo tres puntos, obviamente mientras más puntos se tengas se tendrá una mejor reconstrucción.

La figura 2.14(a) muestra el perímetro de los circulos, el cual se ha obtenido mediante la técnica de erosión. Se puede observar que los perímetros no corresponden a circulos perfectos, sino que tienen deformaciones en el perímetro. Dadas las condiciones en que son tomadas las imagenes de los circuitos impresos, se ha observado que las deformaciones se presentan en mayor grado en el lado derecho de los círculos. Esto

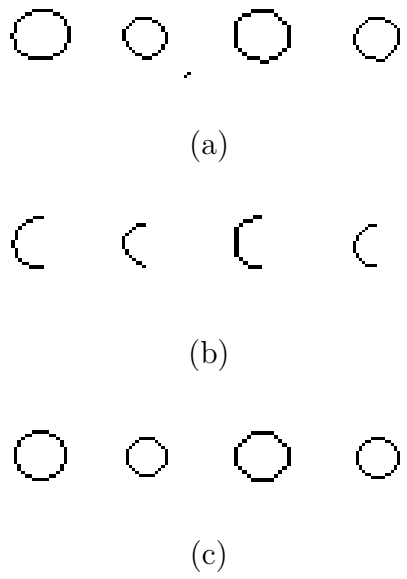


Figura 2.14: Reconstrucción de círculos: (a) círculos a reconstruir; (b) puntos utilizados en la reconstrucción; (c) círculos reconstruidos

nos lleva a tener que descartar la parte derecha de los círculos, utilizando solamente la parte izquierda para la reconstrucción, esto se puede observar en la figura 2.14(b). Finalmente, la figura 2.14(c) muestra los círculos reconstruidos y a partir de los cuales se puede realizar el cálculo de los centroides.

Capítulo 3

Diseño del sistema QCTool

En este capítulo se presenta el diseño y desarrollo del sistema, estableciendo los requerimientos, la arquitectura, la funcionalidad e implementación. El desarrollo de un sistema de software es un proceso complejo que se puede desarrollar desde cero; sin embargo, cada vez más los sistemas se desarrollan ampliando y modificando los sistemas existentes.

Aunque no existe una metodología ideal en el desarrollo de software, se tienen algunas actividades fundamentales, estas son:

- **especificación del software**: se debe definir la funcionalidad del software y las restricciones en sus operaciones
- **diseño e implementación del software**: se debe producir software que cumpla con su especificación
- **validación del software**: se debe validar el software para asegurar que hace lo que el cliente desea
- **evolución del software**: el software debe evolucionar para cumplir con los cambios en las necesidades del usuario

Se ha mencionado que no existe una metodología ideal en el desarrollo de software, esto se observa en las organizaciones donde tienen muchos enfoques distintos para el desarrollo de software. Los procesos del desarrollo pueden incluir técnicas anticuadas o pueden no tomar ventaja de las buenas prácticas de la ingeniaría de software [32]. En el desarrollo del sistema se ha utilizado lenguaje C/C++ para la implementación de los algoritmo y la biblioteca Qt 4.0 [5] para el diseño de las interfases.

El modelo utilizado para el desarrollo del software es el modelo de cascada. Las etapas de este modelo se muestran en la figura 3.1.

Así, se desarrolló el sistema siguiendo el modelo de cascada, esto fue para realizar un trabajo ordenado, ya que de otro modo sería como trabajar a ciegas.

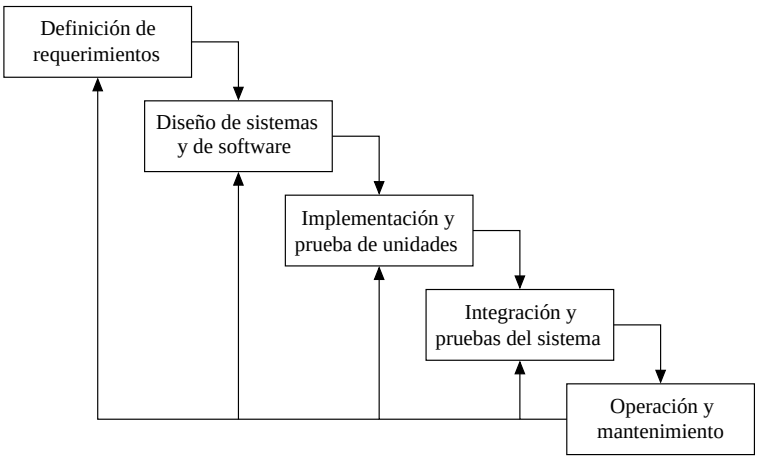


Figura 3.1: Modelo de cascada

3.1. Requerimientos del sistema

En esta sección se establecen detalladamente las características del sistema, su funcionalidad y las interfases de usuario que permitirán interactuar con el usuario. Esta sección corresponde al primer bloque del modelo de cascada, el cual es tal vez el más importante, ya que se necesita entender el problema y lo que realmente necesita el usuario.

3.1.1. Definición de los requerimientos del usuario

En esta sección se muestran los requerimientos del usuario a grandes rasgos.

El software debe ser capaz de localizar las marcas de referencia así como su centroide de cada uno de los sockets en el circuito impreso. El usuario es quién debe especificará la zona en la cual se realizará el proceso de búsqueda tanto de las marcas de referencia como de su centroide, es decir, el software debe ser del tipo semiautomático. Dado que se cuenta con los datos de los archivos Gerber de los circuitos impresos analizados, se debe tener la opción de introducir los valores de los archivos Gerber y obtener la diferencia entre el valor del centroide calculado por el software y el valor del archivo Gerber. Toda la información proporcionada por el software debe de ser almacenada para un análisis posterior por parte del personal calificado para esto.

Las imágenes que se utilizarán en el sistema serán digitalizadas utilizando un escaner. Las imágenes serán a color y debe ser posible trabajar con formatos BMP, JPEG y PNG. El usuario solo debe tener la capacidad para analizar los marcas de referencia, sin alterar el estado de las imágenes.

El software debe contar con una interfaz gráfica amigable, es decir, entendible y fáil de usar, además debe ser capaz de informar el estado de las operaciones realizadas por el usuario. El análisis debe realizarse sobre una marca de referencia a la vez. Para tener un mejor control del análisis, también se debe tener la opción de introducir el modelo del circuito integrado analizado para un par de marcas de referencia. El manual del software debe contener ayuda sobre la instalación y funcionamiento.

3.1.2. Especificación de los requerimientos del sistema

En esta sección se muestran los requerimientos con más detalle. Tomando en cuenta estos requerimientos se inicia el diseño y desarrollo del sistema.

- El objetivo del sistema es el análisis de imágenes a color en diversos formatos como BMP, PNG y JPEG
- Las imágenes digitales de los circuito impresos a analizar serán obtenidas mediante el uso de un escaner.
- El sistema manejará resoluciones de 300 dpi, 600 dpi y 1200 dpi con precisiones de 84, 42 y 21 micras respectivamente. Se eligen estos niveles de precisión dado que son los más comúnes en los escaners comerciales
- La resolución de la imagen se definirá al momento de escanear la imagen
- En el análisis de cada circuito impreso se debe indicar la resolución de la imagen de acuerdo al nivel de precisión deseado
- El análsis se realizará solamente en las regiones indicadas por el usuario
- El sistema debe informar al usuario en caso de no haber localizado el centroide, en caso de haber elegido una región donde no se encuentra ninguna marca de referencia, en caso de haber elegido más de una marca de refencia y/o en caso de haber realizado alguna operación no permitida
- La información del Gerber correspondiente al circuito impreso analizado, debe poderse introducir al sistema para realizar una comparación entre los resultados del sistema y los del Gerber
- El sistema debe tener la opción de guardar los resultados
- El sistema recibirá al inicio del análisis datos como el tamaño del circuito impreso y su resolución
- Se debe tener la opción de cambiar la configuración inicial del circuito impreso
- Se debe contar con un icono de ayuda, la cual debe incluir tanto la forma de instalación como el funcionamiento del sistema

- La interfaz gráfica de usuario debe ser amigable, es decir, entendible para cualquier persona
- El sistema se instalará sobre el sistema operativo Windows XP
- Se debe tener un archivo de instalación y desinstalación del sistema

De acuerdo a la descripción, se hará uso del Procesamiento Digital de Imágenes. La implementación de los algoritmo se realizará con lenguaje C/C++ y el desarrollo de las interfases se realizará con la biblioteca Qt.

3.2. Arquitectura del sistema

La arquitectura general del sistema se muestra en la figura 3.2, donde se puede ver que consta de cuatro etapas fundamentales:

- adquisición de la imagen: en este bloque se realiza en escaneo de la imagen con un nivel de resolución de 300 dpi, 600 dpi o 1200 dpi y la imagen digitalizada se almacena en formato BMP, JPEG o PNG
- preprocesamiento: en este bloque se aplican diversas operaciones para mejorar la calidad de la imagen y obtener mejores resultados en los bloques posteriores. Las operaciones que se aplican en este bloque y que fueron explicadas en el capitulo 2 son la transformación de modelos de color, específicamente el modelo HSI, binarización de la imagen, operaciones morfológicas y segmentación. Todas estas operaciones permiten mejorar la precisión de los resultados
- análisis de la imagen: en este bloque se reliza el análisis de la imagen de acuerdo a ciertos parámetros establecidos, en este caso, se realiza la búsqueda y cálculo del centroide de las marcas de referencia de los sockets del circuito impreso. Básicamente las operaciones que se realizan en este bloque son la extracción, reconocimiento y reconstrucción de objetos.
- resultados: en este bloque se obtienen los resultados del análisis así como el cálculo de la diferencia entre los valores obtenidos por el sistema y los valores de referencia

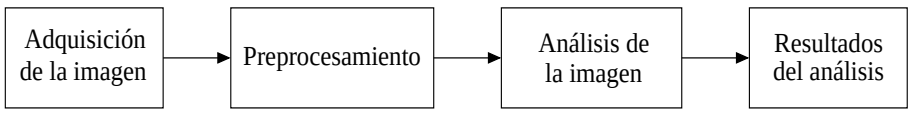


Figura 3.2: Arquitectura general

En la siguiente sección se muestran de manera detallada las operaciones que se llevan a cabo en cada bloque de la arquitectura.

3.3. Diseño del sistema

En esta sección se muestra el diseño del sistema, así como algunos diagramas que permiten entender mejor el funcionamiento del sistema.

La adquisición de los circuitos impresos se realiza mediante el uso de un escaner. El tipo de escaner utilizado para la digitalización de los circuitos impresos es de *cama plana* y como se ha mencionado, las resoluciones que se pueden manejar por el sistema son 300 dpi, 600 dpi y 1200 dpi. Este proceso es realizado por parte del usuario, sin embargo, es necesario establecer los requisitos como el tipo de escaner y los niveles de resolución aceptados.

Es claro que la precisión de los resultados depende de la calidad de la imagen digitalizada, sin embargo, a mayor calidad mayor capacidad y tiempo de cómputo, por lo que el equipo de cómputo debe contar con suficientes recursos, por ejemplo: procesador Intel Pentium IV ó superior, disco duro de 80 GB, memoria de acceso aleatorio (RAM) de 1 GB, entre otros.

La figura 3.3 muestra un ejemplo de un circuito impreso digitalizado con una resolución de 300 dpi.

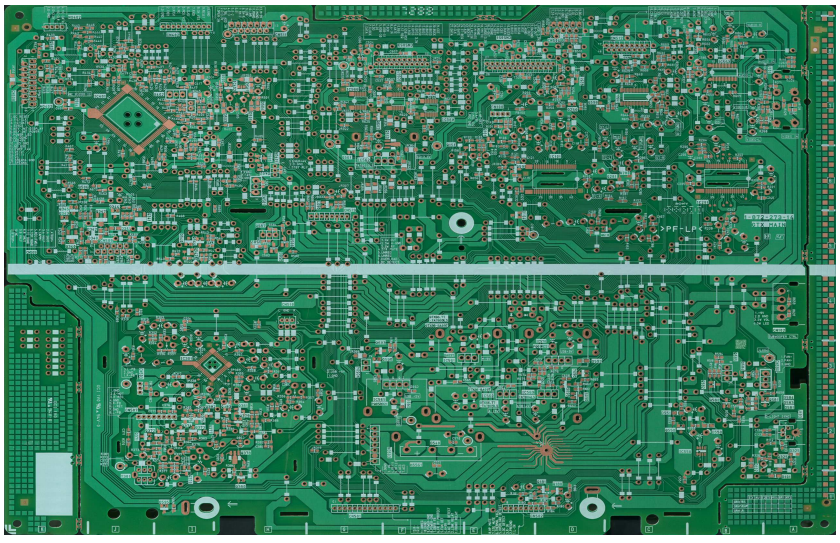


Figura 3.3: Circuito impreso digitalizado a 300 dpi

Mientras la imagen de la figura 3.3 de 300 dpi y con un espacio de disco de 5 MB, una imagen de 1200 dpi puede llegar a ocupar 300 MB en espacio en disco, por la cual su análisis requiere mayor tiempo y recursos de cómputo.

3.3.1. Preprocesamiento

El preprocesamiento es una tarea muy importante, ya que permite mejorar la calidad de la imagen que por razones externas ha sufrido dregadaciones en diferentes niveles. Son varias las operaciones que pueden aplicarse en el preprocesamiento, entre las cuales se encuentran la erosión, dilatación, abriendo, cerrando y los diferentes tipos de filtros, entre otros. En esta sección se muestra la forma y las condiciones bajo las cuales son aplicadas dichas operaciones. En el capitulo 2 se explica de forma detallada el funcionamiento de estas operaciones.

Transformación RGB a HSI

El primer paso es realizar la lectura de la imagen a color en formato RBG y realizar la transformación al formato HSI. La transformación se realiza de acuerdo a la ecuaciones mostradas del capitulo 2. Este proceso se muestra en la figura 3.4.

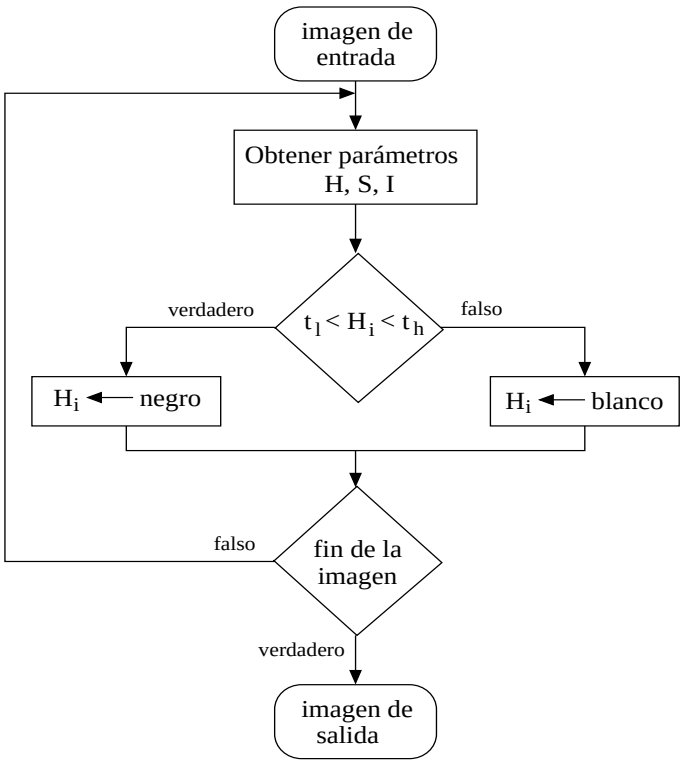


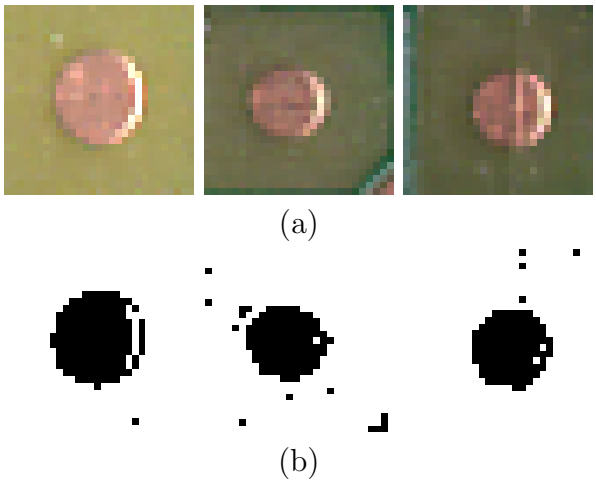
Figura 3.4: Diagrama de transformación de RGB a HSI

Cada pixel de la imagen es transformado al formato HSI y el diagrama muestra que el componente H (Hue) es utilizado para analizar si su valor se encuentra entre los umbrales inferior y superior. En el caso verdadero, al pixel correspondiente se le asigna el valor 0 o color negro, de lo contrario, es asignado el valor 1 o color blanco.

Los objetos que se desean analizar en los circuitos impresos son de color cobre como se muestra en la figura 3.5 y su valor equivalente en el formato HSI esta en el rango de 5.50 y 6.50. Este valor se puede medir en radianes o grados utilizando para ello el cono de colores del formato HSI.

Figura 3.5: Color de los círculos de interés

Los umbrales fueron obtenidos experimentalmente analizado los objetos en diferetes posiciones del circuito impreso. Se observó que a pesar de tener diferentes intensidades de luz en el circuito impreso, el valor del parámetro Hue es muy similar en todos los puntos analizados, obteniendo como resultado los umbrales inferior y superior.



(a)

(b)

Figura 3.6: Ejemplos de la transformación HSI a binario

El objetivo en esta etapa es obtener imagenes binarias que servirán en las etapas posteriores de reconocimiento y reconstrcción. En la figura 3.6 se muestra algunos ejemplos de la transformación HSI. En la figura 3.6a se muestran las imágenes a color y en la figura 3.6b las imágenes de salida en formato binario.

En este caso, dado que se estan trabajando con imáganes a color en formato HSI, solamente necesitamos manejar un umbral inferior y superior, lo cual no es posible

si se trabaja desde un inicio con imágenes en formato RGB o imágenes en escala de grises, ya que debido a que la intensidad de luz en el circuito impreso no es uniforme, se tendrían que cambiar los umbrales para cada punto que se analice, lo cual no es factible para un sistema automático. En las figuras 3.6(a) se observa que el brillo no es uniforma en todas las imágenes, sin embargo, el parámetro Hue nos permite trabajar independientemente del brillo de la imagen, así el umbral obtenido experimentalmente funciona correctamente para todos los puntos del circuito impreso.

Erosión y dilatación

La segmentación o separación de los cŕculos de la imagen con este método es mucho más eficiente que utilizar imágenes en escala de grises, sin embargo, aun así la segmentación presenta ruido u objetos que no son de interés para el análisis, esto se puede ver en las imágenes de la figura 3.6b. En estas imágenes, los círculos presentan tanto huecos como pixeles de sobra fuera de los círculos, los cuales pueden afectar a las etapas posteriores. Una forma muy común de reducir y/o eliminar el ruido es con las operaciones morfológicas erosión y dilatación. La explicación detallada sobre el funcionamiento de estas operaciones se puede ver en el capitulo 2.

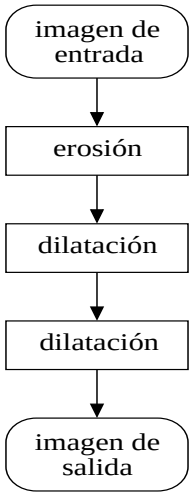


Figura 3.7: Diagrama de aplicación de operaciones morfológicas

La figura 3.7 muestra el orden en que las operaciones morfológicas son aplicadas. La primera erosión permite reducir los pixeles fuera de los círculos pero también se ve afectado el círculo, por lo cual es aplicada la dilatación, para neutralizar el efecto de la erosión. Una segunda dilatación es aplicada para rellenar los huecos presentes dentro de los círculos, de hecho las dos dilataciones intervienen en el rellenado de huecos. Las imágenes de la figura 3.8 corresponden a las imágenes de la figura 3.6(b) después de la aplicación de las operaciones morfológicas en el orden en que se muestra

en el diagrama de la figura 3.7.

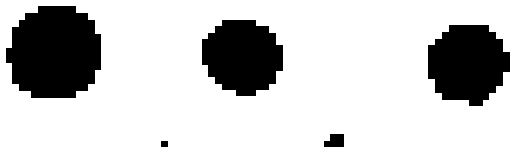


Figura 3.8: Resultado de la aplicación de operaciones morfológicas

Así, se puede observar que el preprocesamiento incrementa la calidad de las imágenes cuando estas presentan algún tipo de ruido. Es posible que el nivel de ruido en las imágenes sea de mayor intensidad que en los ejemplos mostrados, de tal modo que no sea posible obtener imágenes libres de ruido, sin embargo, en la etapa siguiente que corresponde al reconocimiento de objetos se puede resolver este problema extrayendo solo los objetos de interes utilizando el método de los momentos invariantes. Las imágenes de salida de esta etapa cuentan con las condiciones necesarias para entrar al proceso de reconocimiento.

3.3.2. Reconocimiento de círculos

En la etapa anterior se preparó la imagen para realizar la extracción de objetos. Existen diversos métodos para el reconocimiento de objetos, en esta tesis se utiliza el método de los momentos invariates de Hu [12]. Como se mencionó en el capitulo 2, podemos extraer el vector de características de todos los objetos dentro de una imagen y en base a estos, seleccionar el objeto adecuado. El diagrama de la figura 3.9 muestra la secuencia de pasos en el reconocimiento de círculos.

El diagrama muestra que se debe realizar la extracción de los objetos dentro de la imagen, en este caso se utiliza la biblioteca SCimagen [27], a continuación se obtiene el vector de características de cada objeto, utilizando para esto el método de los momentos invariantes de Hu [12]. El primer momento invariante es utilizado para verificar si se trata de un círculo. Para realizar la comparación, previamente han sido analizados diferentes círculos y se obtuvieron los valores de referencia para la comparación experimentalmente. El cuadro 3.1 muestra el valor de los momentos invariantes para diferentes círculos tomados de diferentes circuitos impresos.

En el cuadro 3.1 se muestran solo los primeros cuatro momentos tomados de diferentes círculos. Se puede observar que el momentos que se mantiene más estable en el primero, de aquí que se halla elegido este parámetro como referencia de comparación. El rango de valores para realizar la comparación con el primer momentos se seleccionó experimentalmente, en el cuadro 3.1 se muestran 16 diferentes mediciones, sin embargo, se analizaron aproximadamente 30 círculos en diferentes posiciones de

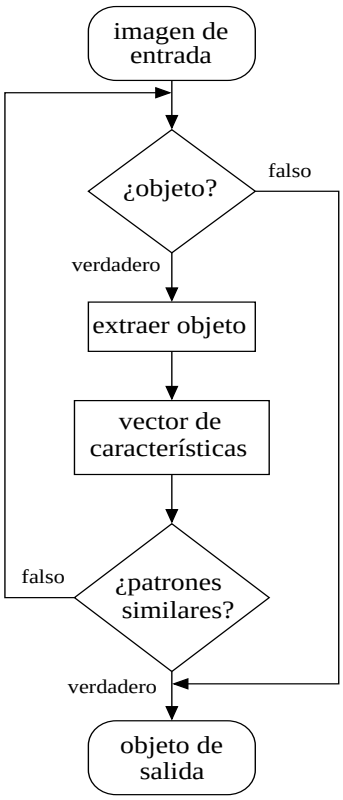


Figura 3.9: Diagrama del reconocimiento de círculos

los circuitos impresos. El primer círculo es ideal, es decir, este círculo fue creado manualmente para conocer sus momentos cuando no presentan ningún tipo de ruido, los círculos siguientes presentan algún tipo de degradación.

Analizando todos los círculos de los circuitos impreso, se encontro experimentalmente el rango de valores para el primer momento. Estos rangos se establecieron como se muestra en el cuadro 3.2.

Es decir, para que se considere a un objeto como un círculo, debe cumplir la siguiente condición:

$$t_l \leq M_1 \leq t_h$$

Si los patrones del objeto actual que se está comparando coinciden con los patrones de referencia, entonces este objeto se utiliza en la siguiente etapa de reconstrucción.

Siguiendo la lista de requerimientos, se debe tomar en cuanta también el caso en donde se reconocen más de un círculo y el caso en el que no se detecta ninguno, el manejo de estos casos se muestra en el diagrama de la figura 3.10. Estos casos no

Cuadro 3.1: Momentos de diferentes círculos del circuito impreso

-	Círculo	Momentos			
		1	2	3	4
1		0.158950	0.000000	0.000000	0.000000
2		0.160453	0.000325	0.000006	0.000040
3		0.163140	0.000253	0.000007	0.000404
4		0.160744	0.000008	0.000003	0.000149
5		0.168065	0.000027	0.000009	0.000909
6		0.160973	0.000011	0.000005	0.000198
7		0.167875	0.000143	0.000004	0.000926
8		0.168017	0.000069	0.000012	0.000923
9		0.172024	0.000134	0.000047	0.001294
10		0.162886	0.000024	0.000008	0.000362
11		0.161694	0.000023	0.000001	0.000227
12		0.174500	0.000322	0.000032	0.001661
13		0.159487	0.000000	0.000039	0.000055
14		0.166478	0.000049	0.000057	0.000690
15		0.168595	0.000192	0.000007	0.000978
16		0.172005	0.000065	0.000164	0.001105

Cuadro 3.2: Rango de valores para el primer momento

-	umbral inferior	umbral superior
momento 1	0.150000	0.180000

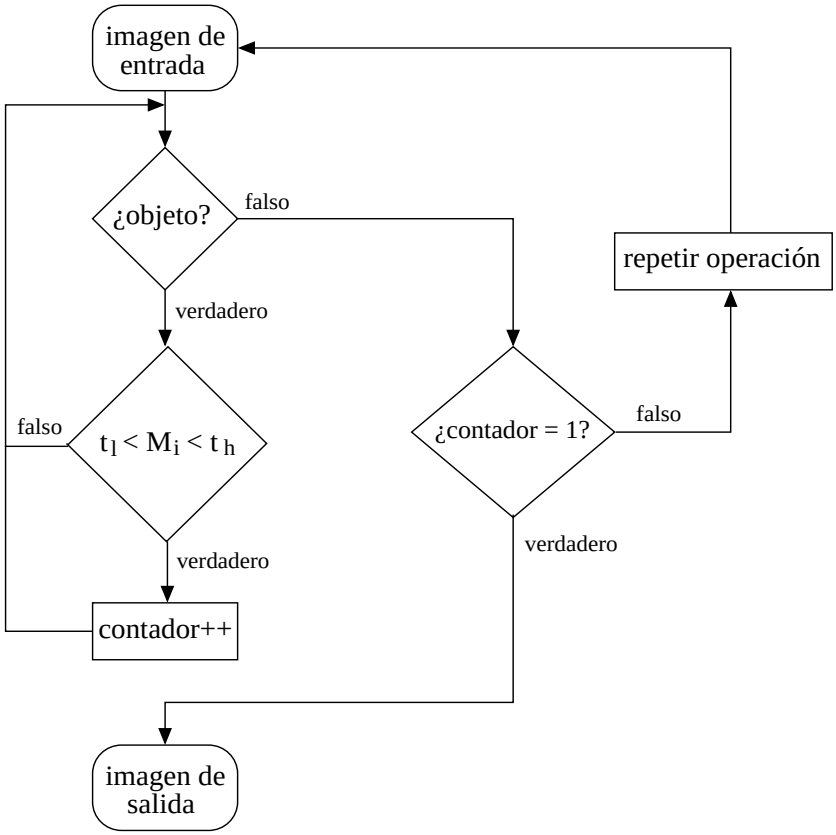


Figura 3.10: Diagrama para los casos en que se encuentran más de uno o ningún círculo

son permitidos, por lo que si alguno de estos se presenta, debe repetirse el proceso de búsqueda o elegir una nueva imagen, en el caso contrario, el objeto detectado se utiliza en la siguiente etapa de reconstrucción.

3.3.3. Reconstrucción de círculos

Los círculos extraídos de las imágenes presentan deformaciones y a pesar de que se han aplicado operaciones morfológicas para reducir y/o eliminar el ruido, su forma no es la de un círculo ideal. En esta sección se muestra el resultado de la aplicación del algoritmo de reconstrucción llamado Modified Least-Square (MLS) [31], el cual se basa en las posiciones de los pixeles del perímetro de los círculos.

Este proceso se explica detalladamente en el capitulo 2 (en la pág. 28). El diagrama de la figura 3.11 muestra la secuencia de pasos para realizar la reocnstrucción.

El primer paso es obtener el perímetro del círculo que se va a reconstruir. La forma de realizar esto es con la diferencia entre el círculo original y el mismo círculo erosionado como se muestra en la figura 3.11. El siguiente pso es aplicar el algoritmo MLS. Para tener mejores resultados en la reconstrucción, solo se utiliza la parte del

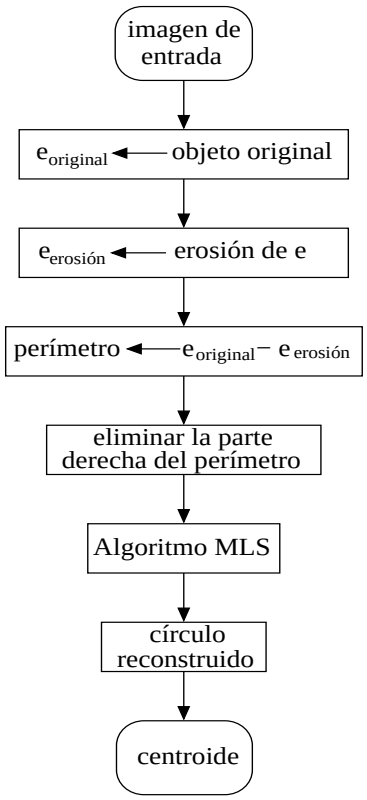


Figura 3.11: Diagrama de reconstrucción de círculos

perímetro que se ha sido menos afectada por el ruido, en este caso, esto corresponde a la parte izquierda del perímetro. Esto se debe a que la luz no es uniforme en el circuito impreso durante la digitalización con el escaner. En la figura 2.14 se muestran algunos ejemplos de este procedimiento de reconstrucción.

Una vez que el círculo ha sido reconstruido, es momento de calcular el centroide. Este valor es calculado como se muestra en el capitulo 2, aplicando las siguientes ecuaciones.

$$\begin{aligned} a_M &= \frac{DC - BE}{AC - B^2} \\ b_M &= \frac{AE - BD}{AC - B^2} \end{aligned}$$

El centroide esta representado por (a_M, b_M). Sin embargo, lo que realmente se necesita es el centroide del círculo respecto a un punto de referencia general que se encuentra en el circuito impreso. Por esto, previamente debe ser calculado el centroide del punto de referencia general. Al realizar esto, el centroide correcto se ha localizado.

3.3.4. Diagramas del sistema

En esta sección se muestran algunos diagramas para mostrar el funcionamiento del sistema y las operaciones permitidas por el sistema.

Diagrama de casos de uso y de secuencia del sistema

En la figura 3.12 se muestran el diagrama de casos de uso de las operaciones que puede realizar el usuario en el sistema. El usuario puede realizar varias operaciones como: abrir una imagen, seleccionar diferentes regiones de análisis, localizar de manera automática los centroides, realizar la comparación entre los resultados calculados y los valores de referencia, así como almacenar los resultados para un análisis posterior por parte de usuario.

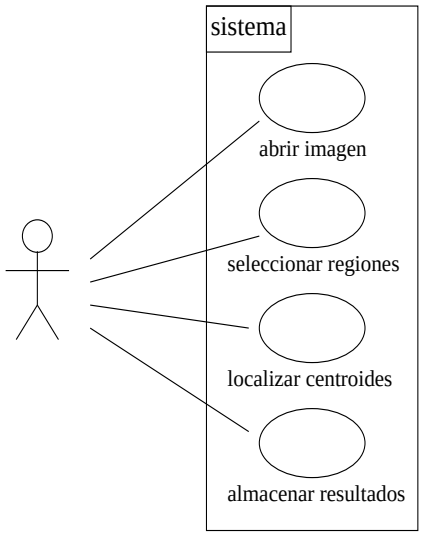


Figura 3.12: Operaciones que puede realizar el usuario

Esto muestra que el sistema debe tener las opciones mostradas en el diagrama, así como también la interfaz gráfica debe contar con las opciones de abrir imágenes, seleccionar regiones, entre otras.

El proceso completo que debe realizar el usuario para el análisis de las marcas de referencia sobre un circuito impreso, se muestra en el diagrama de secuencia de la figura 3.13.

En este diagrama se muestra el orden en que se efectúan las operaciones tanto por el usuario como por el sistema. La primera operación es crear un nuevo proyecto, la siguiente operación es seleccionar la imagen que será analizada. La operación de establecer parámetros es ejecutada automáticamente por el sistema, sin embargo, también es necesario que el usuario ingrese información sobre el circuito impreso como

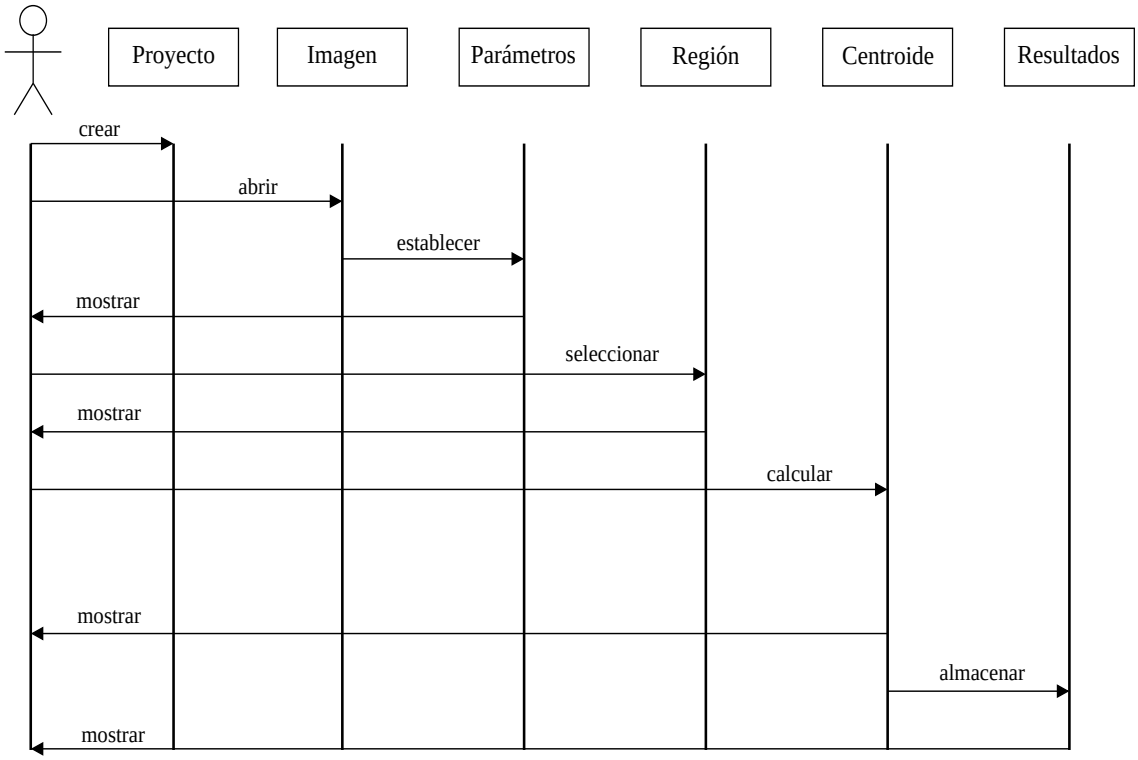


Figura 3.13: Diagrama de secuencia

son el ancho y alto de la tarjeta. El usuario debe seleccionar la región en la cual se desea realizar el análisis y automáticamente el sistema muestra la región seleccionada y lista para calcular el centroide del objeto localizado. Los resultados pueden ser almacenados, esta operación también es ejecutada por el usuario.

Como se puede observar, con este diagrama se tiene un mejor panorama del funcionamiento del sistema, de aquí también se puede ver que algunos de los objetos que se necesitarán para la implementación del sistema son: crear proyecto, abrir imagen, calculo de centroides, entre otros. A partir de esto se puede inicar con el diagrama de clases e implementación del sistema.

3.4. Implementación del sistema

Esta etapa requiere del entendimiento de la etapa del diseño del sistema, ya que aquí es donde se la traducción de las ideas en el código para el desarrollo del sistema. El lenguaje C/C++ [4, 3, 33, 34] ha sido utilizado para la implementación de los algoritmos, mientras que para el desarrollo de la interfaz gráfica ha sido utilizada la biblioteca Qt de Trolltech [35, 5, 6, 7].

3.4.1. Interfaz principal

La interfaz gráfica debe contar con todas las opciones necesarias para llevar a cabo el análisis, además de que debe ser amigable para el usuario. Las operaciones permitidas para el usuario descritas en la definicón de requerimientos son: Nuevo proyecto, Abrir imagen, Cerrar proyecto, Propiedades, Localizar Centroide y Ayuda.

En la figura 3.14 se muestra la interfaz gráfica generada con la implementación del siguiente código. El código mostrado es una parte de la definición de la clase para la interfaz gráfica.

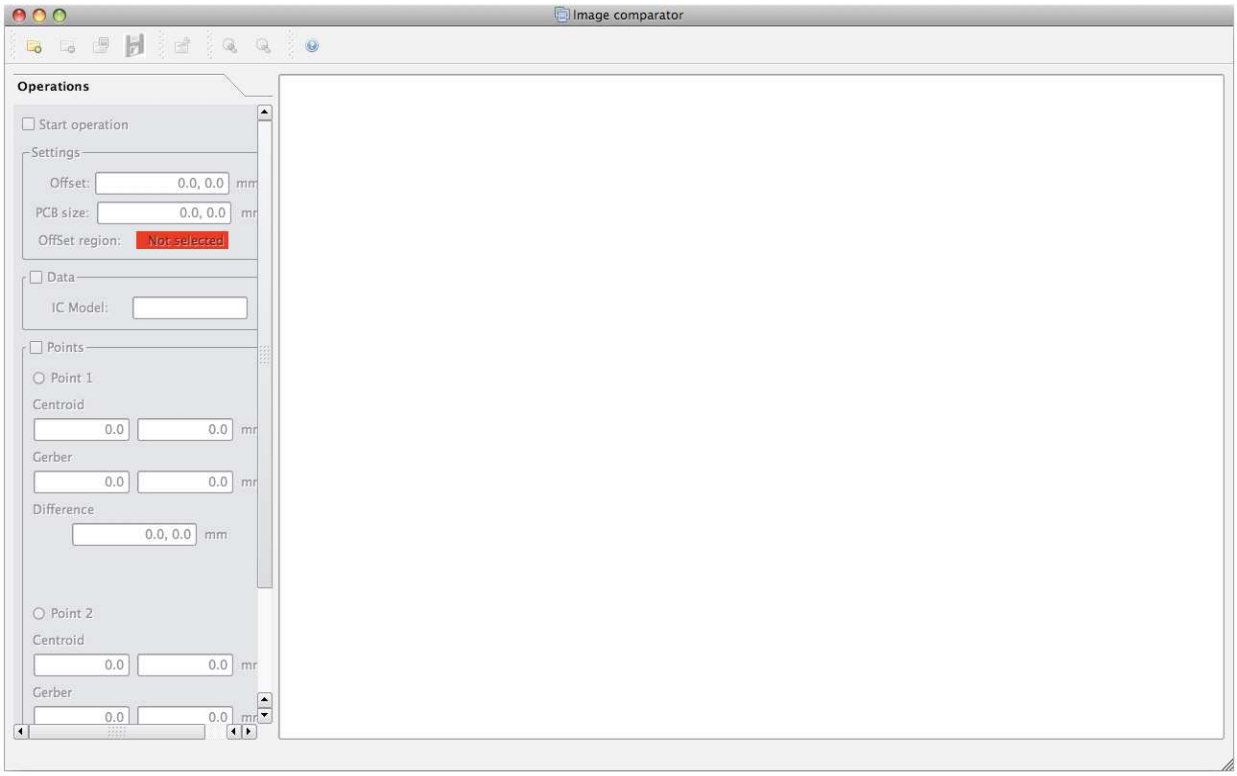


Figura 3.14: Interfaz gráfica principal

```
class DiagramView: public Q3CanvasView
{
    Q_OBJECT

    public:
        DiagramView(Q3Canvas *canvas, QWidget *parent = 0, const char *name = 0,
        Qt::WindowFlags f = 0);
```

```
        void getPosition(int *a, int *b){*a = px, *b = py;};
        void getFinalPosition(int *a, int *b){*a = pxfin, *b = pyfin;};
        void initDraw(bool);
};

class MainWindow: public QMainWindow
{
    Q_OBJECT

    public:
        MainWindow(Q3Canvas *canvas, QWidget *parent = 0, const char *name = 0,
        Qt::WindowFlags f = 0);
        void openTheImage(QString);

    private:
        DiagramView *diagram;
        Q3Canvas *canvas;
        Q3Canvas *canvas2;
        QMatrix matrix;
};
```

A continuación se muestra la implementación de las funciones que debe realizar el sistema, así como la interacción con el usuario.

3.4.2. Crear un proyecto

La creación de un nuevo proyecto requiere de la siguiente información proporcionada por el usuario:

- Nombre del proyecto
- Ruta donde se almacenará el proyecto

El sistema automáticamente crea una carpeta con el mismo nombre del proyecto introducido, así como también crea un archivo de texto con el mismo nombre del proyecto y será utilizado por el sistema para almacenar los resultados del análisis correspondiente al proyecto actual. La figura 3.15 muestra la ventana para insertar la información referente al proyecto.

El código para esta operación se muestra a continuación.

```
void Project::setProjectLocation()
{
    QString dir =  QFileDialog::getExistingDirectory(this,
    tr("Project Location"), "", QFileDialog::ShowDirsOnly);
```

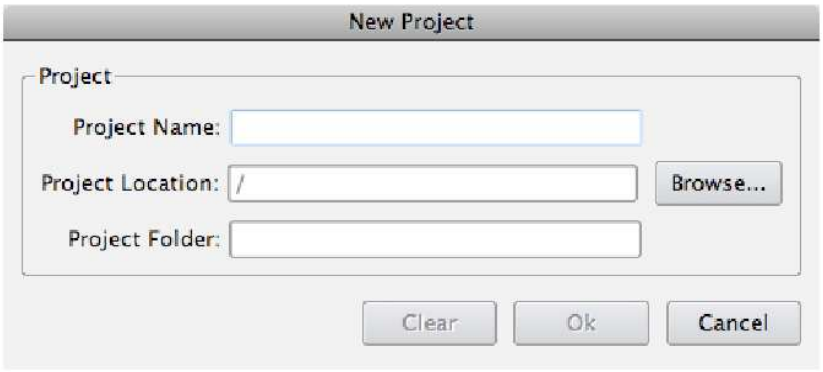


Figura 3.15: Ventana de creación de un proyecto

```
    txtProjectLocation->setText(dir);
}

void Project::setProjectFolder()
{
    QString tempLoc = txtProjectLocation->text();
    int n = tempLoc.length();
    if( tempLoc[ n - 1 ] == '/' )
        txtProjectFolder->setText(txtProjectLocation->text() +
        txtProjectName->text());
    else
        txtProjectFolder->setText(txtProjectLocation->text() + "/" +
        txtProjectName->text());
}

void Project::createProjectFolder()
{
    FILE *arch;
    QDir projectDir;

    projectDir.QDir::mkdir(txtProjectFolder->text());
    arch = fopen(txtProjectFolder->text() + "/" + txtProjectName->text()
    + ".txt", "w");
    fclose(arch);
}
```

3.4.3. Abrir una imagen

La apertura de la imagen es una operación que está permitida solo si existe algún proyecto activo, de otro modo, esta operación se deshabilita. El sistema permite abrir imagenes con formato BMP, PNG y JPEG. La figura 3.16 muestra el la interfaz después de haber abierto una imagen, en este caso, en formato JPEG con una resolución

de 600 dpi.

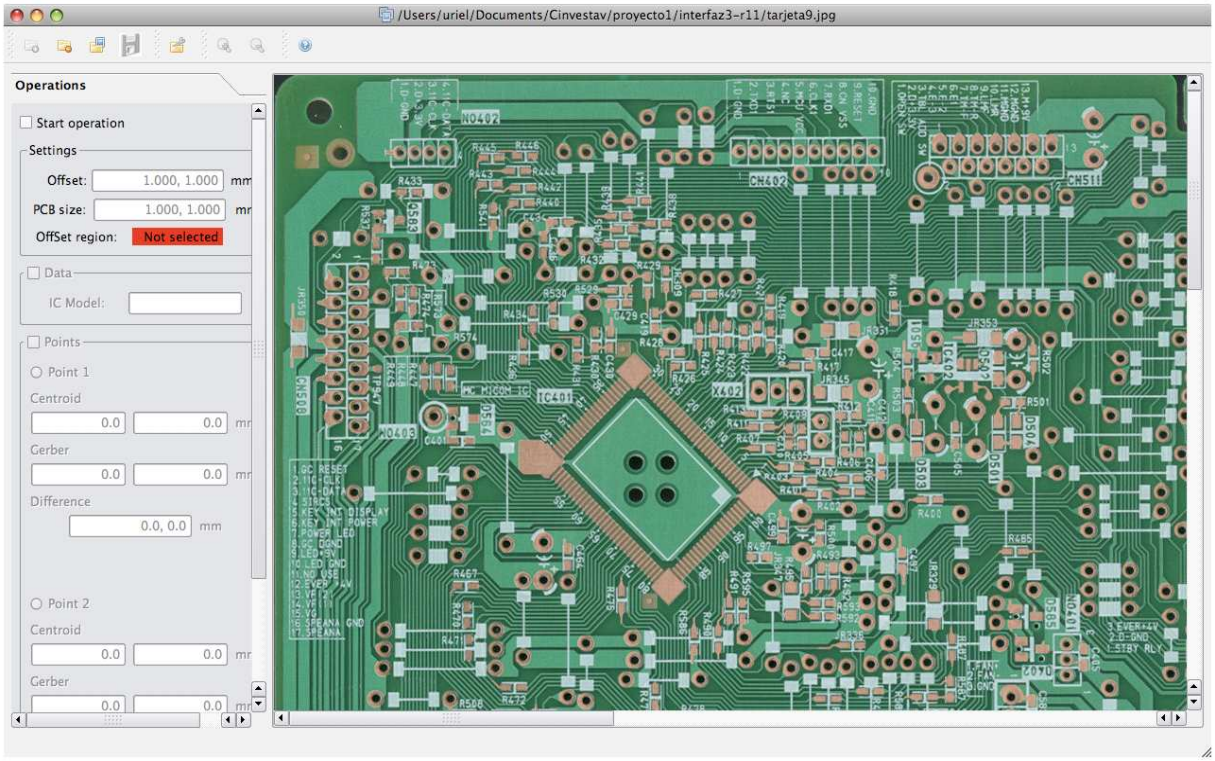


Figura 3.16: Imagen JPEG abierta en la interfaz

Como ya se ha mencionado, el sistema puede manejar imágenes de 300 dpi, 600 dpi y 1200 dpi. Mientras mayor sea la resolución, mayor será el tiempo de cómputo así como también el tiempo en cargar la imagen en la interfaz aumenta. El código para abrir la imagen se muestra a continuación.

```
void MainWindow::fileOpen()
{
    QFileDialog *fd = new QFileDialog(diagram, "Open file", "", "TRUE");

    fd->setDirectory(fd->directory());
    fd->setMode(QFileDialog::ExistingFile);
    fd->setFilter(tr("Images (*.pnm *.pgm *.pbm *.bmp *jpg)"));
    fd->setViewMode(QFileDialog::List);

    if( fd->exec() == QDialog::Accepted )
    {
        fileName = "";
```

```
        fileName = fd->selectedFile();

        image = new QImage(fileName);
        wmax = image->width();
        hmax = image->height();

        printf("\nOpen\n");
        diagram->resizeContents(wmax, hmax);
        canvas->resize(wmax, hmax);
        canvas->setBackgroundPixmap(QPixmap::fromImage(*image));
        diagram->setCanvas(canvas);
        matrix.reset();

        setWindowTitle(fileName);

        if( fileName != "vacio" )
        {
         widget1InitSettings->setPixelXY(wmax,hmax);
         emit openImageReady();
        }
    }
}
```

3.4.4. Establecer parámetros del circuito impreso

El establecimiento de los parámetros del circuito impreso, incrementa la precisión de los resultados. Los parámetros que deben ser introducidos: El tamaño de la tarjeta y la posición del punto de referencia general u offset de la tarjeta. Estos valores son conocidos por el usuario, por lo cual no hay inconveniente en su uso para el análisis. La figura 3.17 muestra la ventana para introducir los parámtros.

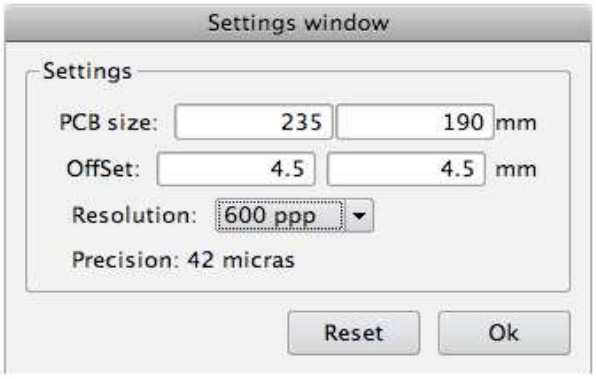


Figura 3.17: Parámetros del circuito impreso actual

El código de la definición de la clase de inicialización de los parámtros se muestra

a continuación.

```
class InitSettings: public QWidget
{
    Q_OBJECT

    public:
        InitSettings(QWidget *parent = 0, int x = 0, int y = 0);
        void setPixelXY(int, int);
        int getPrecision();
        void getOffSet(float *, float*);
        void getImageSize(float *, float*);
        void showErrorMessage();

    public slots:
        void reset();
        void setImageResolution();
        void setInitSettings();

    private:
        QGroupBox *dataGroupBox;
        QLineEdit *txtSizeX, *txtSizeY, *txtOffSetX, *txtOffSetY,
        *txtWMax, *txtHMax;
        QComboBox *comboResolution;
        QPushButton *btnReset, *btnOk;

        int intPx, intPy, valRes;
        float floatPx, floatPy, factor;
        float wmax, hmax, offsetx, offsety;
};
```

3.4.5. Seleccionar región de búsqueda

La selección de las regiones para realizar la búsqueda de los círculos y posteriormente calcular los centroides, es una operación interactiva, es decir, el usuario puede elegir libremente con el ratón de la computadora la región que le interése. Esta selección se ha representado con cuadros de color azul, para ubicar fácilmente su posción. La figura 3.18 muestra una región marcada por un cuadro azul, la cual ha sido seleccionada por el usuario.

```
void DiagramView::mouseMoveEvent(QMouseEvent *e)
{
    if( startDraw == true )
    {
```

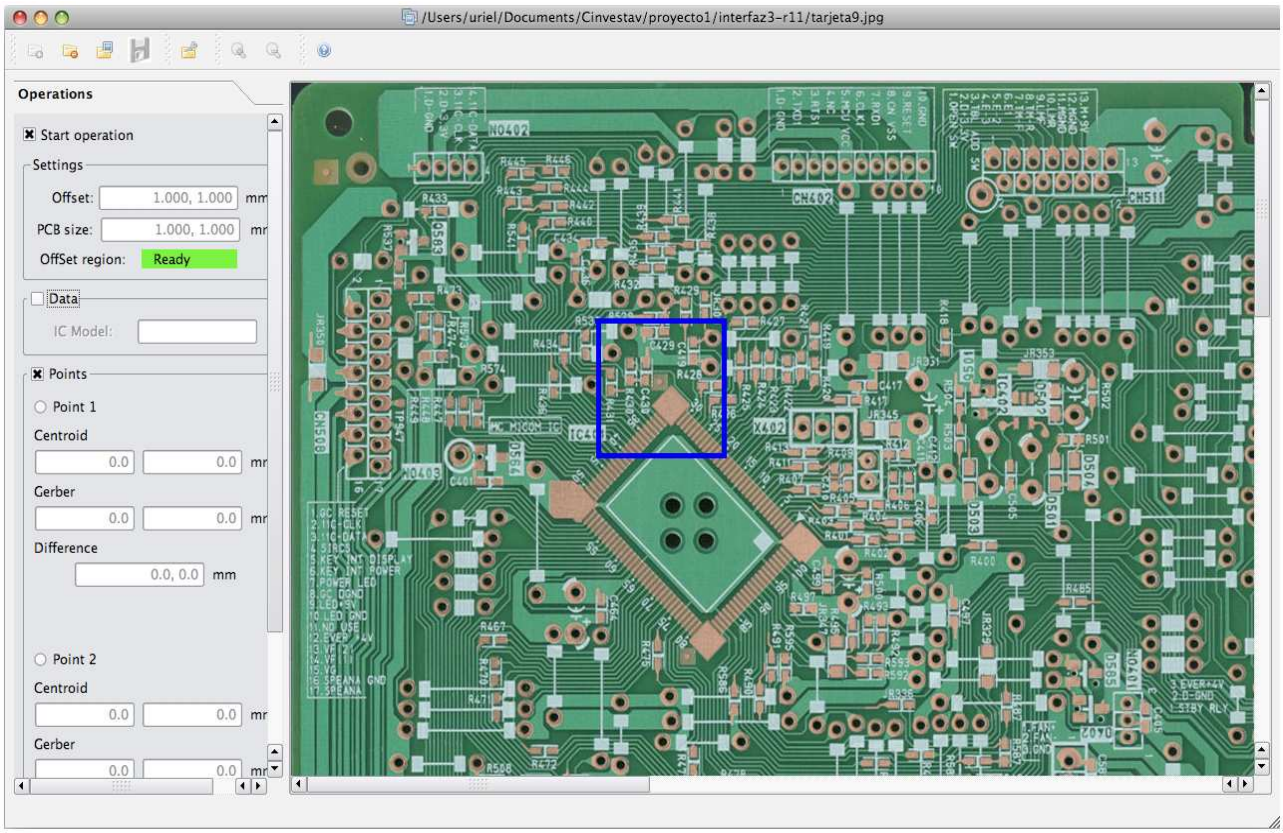


Figura 3.18: Selección de una región de búsqueda

```
        QPoint c = e->pos();
        this->viewportToContents(c.x(), c.y(), pxfin, pyfin);

        if( pxfin >= 0 && pyfin >= 0 )
        {
         int pxtemp = pxfin - px;
         int pytemp = pyfin - py;
         if( pxtemp < 0 )
             pxtemp = 0;
         if( pytemp < 0 )
             pytemp = 0;
         box->setSize(pxtemp, pytemp);
         box->show();
         canvas()->update();
        }
    }
}


void DiagramView::mousePressEvent(QMouseEvent *e)
```

```
{
    if( startDraw == true )
    {
        QPoint c = e->pos();
        this->viewportToContents(c.x(), c.y(), px, py);
        if( px >= 0 && py >= 0 )
        {
            emit changePosition();
            box->move(px, py);
        }
    }
}
```

3.4.6. Calcular el centroide

Este proceso requiere de otros procesos previos, es decir, procesos del preprocesamiento, lo cuales son de gran importancia para poder obtener buenos resultados. En seguida se muestra la implementación de todos estos procesos.

Tranformación de RGB a HSI

La implementación para realizar la transformación de RGB a HSI es como se muestra en el siguiente código.

```
void Puntos::rgbToHsi( int pxini, int pyini, int pxfin, int pyfin )
{
    for( int i = 0; i < pixelY; i++ )
    {
        for( int j = 0; j < pixelX; j++ )
        {
            value = imgSource.pixel(j + pxini, i + pyini);
            color.setRgb(value);
            color.getRgb(&intR, &intG, &intB);

            r = (double)intR / (double)(intR + intG + intB);
            g = (double)intG / (double)(intR + intG + intB);
            b = (double)intB / (double)(intR + intG + intB);

            if( r <= b )
                mh = acos((0.5*((r-g)+(r-b)))/(sqrt(((r-g)*(r-g))+((r-b)
                *(g-b)))));
            else
                mh = 2.0*3.1416 - (acos((0.5*((r-g)+(r-b)))/(sqrt(((r-g)
                *(r-g))+((r-b)*(g-b))))));
```

```
            if( r < g && r < b )
                min = r;
            if( g < r && g < b )
                min = g;
            if( b < r && b < g )
                min = b;

            ms = 1.0 - (3.0*min);
            mi = (double)( intR + intG + intB ) / 3.0;
        }
    }
}
```

Esta implementación ha sido realizada siguiendo la teoría de modelos de color mostrada en el capitulo 2.

Aplicación de operaciones morfológicas

Se ha explicado que las operaciones morfológicas permiten reducir y/o eliminar el ruido presente en una imagen, una expliación más detallada sobre esto, se muestra en el capitulo 2. El código utilizado para reducir el nivel de ruido es el siguiente. El código realizar una dilatación, es decir, rellena los huecos presentes sobre la imagen, se puede ver como una matriz de 3×3 que se desplaza sobre la imagen original. Para rellenar un pixel, se necesita que alrededor de éste, existan más de dos pixeles con un valor igual a 6, dado que este valor es el que se obtiene al realizar la transformación al model HSI.

```
for( int i = 1; i < pixelY-1; i++ )
{
    cont = 0;
    for( int j = 1; j < pixelX-1; j++ )
    {
        if( tempImagen[ i ][ j ] < 0 || tempImagen[ i ][ j ] > 6 )
        {
                cont = 0;
                if( tempImagen[ i - 1 ][ j - 1 ] == 6 )
                    cont++;
                if( tempImagen[ i - 1 ][ j ] == 6 )
                    cont++;
                if( tempImagen[ i - 1 ][ j + 1 ] == 6 )
                    cont++;
                if( tempImagen[ i ][ j - 1 ] == 6 )
                    cont++;
                if( tempImagen[ i ][ j + 1 ] == 6 )
                    cont++;
```

```
                if( tempImagen[ i + 1 ][ j - 1 ] == 6 )
                    cont++;
                if( tempImagen[ i + 1 ][ j ] == 6 )
                    cont++;
                if( tempImagen[ i + 1 ][ j + 1 ] == 6 )
                    cont++;

                if( cont > 2 )
                    tempImagen[ i ][ j ] = 6;
                else
                    tempImagen[ i ][ j ] = 0;
                cont = 0;
        }
    }
}
```

Las imágenes deben encontrarse en formato binario al entrar a este proceso. La máscara o elemento estructural utilizada en este proceso es una matriz de 3×3 pixeles. Una característica del elemento estructural es que debe ser impar para poder contar con un pixel central sobre quien se efectúan las operaciones.

Reconocimiento de círculos

Los momentos invariantes de HU [12] es el método utilizado para el reconocimiento de círculos. La biblioteca SCimagen [27] contiene funciones que permiten obtener y manejar los momentos, estas funciones han sido reutilizadas para el diseño del sistema. Algunas de sus características han sido modificadas para adecuarlas a las necesidades de este sistema. Las funciones principales que han sido reutilizadas de la biblioteca SCimagen son: *find_object*, *find_point_object*, *invariantes* y *moment_norm*. El código de estas funciones no se muestra aquí, puede consultarse [27] para más información.

Una vez que se obtienen los momentos, se necesita realizar la clasificación para saber si el objeto analizado es del tipo buscado, el código de este proceso se muestra a continuación.

```
int clasificadorMD( double *phis, float area )
{
    double minMoment, maxMoment;
    float minArea, maxArea;

    minMoment = 0.150000;
    maxMoment = 0.190000;
    minArea = 40.0;
    maxArea = 10000.0;
```

```
    if( area >= minArea && area <= maxArea )
    {
        if(  phis[ 0 ] >= minMoment && phis[ 0 ] <= maxMoment )
        {
            tipoDeObjeto = "circulo detectado";
            return 1;
        }
        else
        {
            tipoDeObjeto = "No reconocido";
            return 0;
        }
    }
}
```

Reconstrucción y cálculo de centroides

Esta es la etapa donde se realiza la reconstrucción y el cálculo de los centroides de los círculos, al igual que en las etapas anteriores, aquí también se trabaja con imágenes en formato binario. El algoritmo de reconstrucción de círculos MLS [31] trabaja con los pixeles del perímetro del círculo. El código para obtener el perímetro es el siguiente.

```
/*calculo de puntos erosionados*/
for( int i = 1; i < rows-1; i++ )
{
    for( int j = 1; j < cols-1; j++ )
    {
        if( data[ i ][ j ] == 0 )
        {
            cont = 0;
            if( data[ i - 1 ][ j - 1 ] == 255 )
                cont++;
            if( data[ i - 1 ][ j ] == 255 )
                cont++;
            if( data[ i - 1 ][ j + 1 ] == 255 )
                cont++;
            if( data[ i ][ j - 1 ] == 255 )
                cont++;
            if( data[ i ][ j + 1 ] == 255 )
                cont++;
            if( data[ i + 1 ][ j - 1 ] == 255 )
                cont++;
            if( data[ i + 1 ][ j ] == 255 )
                cont++;
            if( data[ i + 1 ][ j + 1 ] == 255 )
                cont++;
```

```
            if( cont < 2 )
                tempFittingCircle[ i ][ j ] = 0;
            else
                tpuntos++;
        }
    }
}

int tempPuntoX[ tpuntos ], tempPuntoY[ tpuntos ];
int px = 0, py = 0;

/*calculo del borde a partir de la imagen original y la erosionada*/
for( int i = 0; i < rows; i++ )
{
    for( int j = 0; j < cols; j++ )
    {
        if( data[ i ][ j ] == tempFittingCircle[ i ][ j ] )
            tempFittingCircle[ i ][ j ] = 255;
        else
        {
         tempFittingCircle[ i ][ j ] = 0;
         tempPuntoX[ px ] = j;
         tempPuntoY[ py ] = i;
         px++;
         py++;
        }
    }
}
```

A partir de los pixeles del perímetro se puede iniciar la reconstrucción. La implementación requiere del cálculo del centro del círculo y del cáluclo del radio, esto se muestra en el siguiente código.

```
for( int i = 0; i < n; i++ )
    sumaX = sumaX + tempRecPuntoX[ i ];
for( int i = 0; i < n; i++ )
    sumaX2 = sumaX2 + ( tempRecPuntoX[ i ] * tempRecPuntoX[ i ] );
for( int i = 0; i < n; i++ )
    sumaX3 = sumaX3 + ( tempRecPuntoX[ i ] * tempRecPuntoX[ i ] *
    tempRecPuntoX[ i ] );
for( int i = 0; i < n; i++ )
    sumaY = sumaY + tempRecPuntoY[ i ];
for( int i = 0; i < n; i++ )
    sumaY2 = sumaY2 + ( tempRecPuntoY[ i ] * tempRecPuntoY[ i ] );
for( int i = 0; i < n; i++ )
```

```
    sumaY3 = sumaY3 + ( tempRecPuntoY[ i ] * tempRecPuntoY[ i ] *
    tempRecPuntoY[ i ] );
for( int i = 0; i < n; i++ )
    sumaXY = sumaXY + ( tempRecPuntoX[ i ] * tempRecPuntoY[ i ] );
for( int i = 0; i < n; i++ )
    sumaXY2 = sumaXY2 + ( tempRecPuntoX[ i ] * ( tempRecPuntoY[ i ]
     * tempRecPuntoY[ i ] ) );
for( int i = 0; i < n; i++ )
    sumaYX2 = sumaYX2 + ( tempRecPuntoY[ i ] * ( tempRecPuntoX[ i ]
     * tempRecPuntoX[ i ] ) );

parA = n * sumaX2 - ( sumaX * sumaX );
parB = n * sumaXY - ( sumaX * sumaY );
parC = n * sumaY2 - ( sumaY * sumaY );
parD = 0.5 * ( n * sumaXY2 - ( sumaX * sumaY2 ) + n * sumaX3 -
( sumaX * sumaX2 ) );
parE = 0.5 * ( n * sumaYX2 - ( sumaY * sumaX2 ) + n * sumaY3 -
( sumaY * sumaY2 ) );

ar = ( parD * parC - parB * parE ) / ( parA * parC - parB * parB );
br = ( parA * parE - parB * parD ) / ( parA * parC - parB * parB );
```

El centro del círculo esta representado por (ar, br). En la reconstrucción solo es utilizado el lado izquiero del perímetro, ya que el lado derecho se ve más afectado por la intensidad de luz durante la digitalización, además de que el tiempo de reconstrucción disminuye al tener menos pixeles.

3.4.7. Diagrama de objetos del sistema

La figura 3.19 muestra el diagrama de clases del sistema. En este diagrama se observan los objetos utilizados, sus atributos y métodos, así como la forma en como se encuentran relacionados.

Los objetos principales de los cuales heredan el resto de los objetos son: *QWidget*, *QMainWindow* y *Q3CanvasView*. El objetos principal donde se realizan la mayor parte de la operaciones como preprocesamiento y análisis de imágenes es *Operations*. Los objetos encargados de la interaz gráfica, así como de la interacción con el usuario con: *MainWindow*, *DiagramView*, *WindowView*, *Window*, *Project* y *Settings*.

3.4.8. Diagrama de señales - ranuras (signals - slots) del sistema

La comunicación entre objetos utilizando la biblioteca Qt esta basado en el mecanismo de *señales* y *ranuras* (*signals* y *slots*). Esta es una de las características que permiten que la programación en Qt sea muy cómoda e intuitiva a diferencia de otros

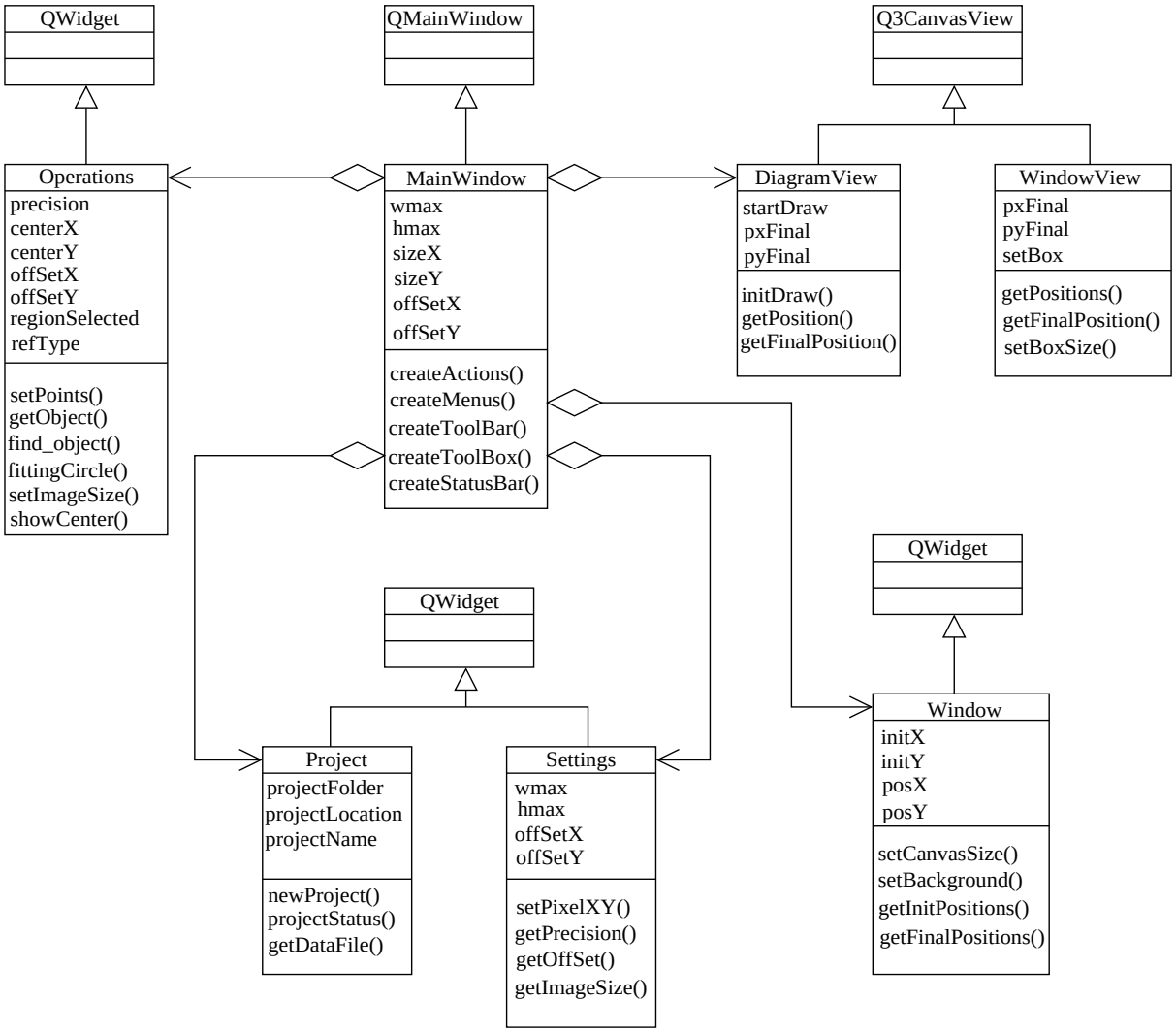


Figura 3.19: Diagrama de objetos del sistema

frameworks.

En la figura 3.20 se muestra un diagrama donde se puede observar como se comunican los diferentes objetos del sistema utilizando el mecanismo de *señales - ranuras*. El funcionamiento de este mecanismo es simple, un objeto transmisor que desea comunicarse con otro objeto debe mandar uns señal al objeto receptor, quien la recibe utilizando una ranura. Esto puede ser utilizado para activar procesos, pasar información, entre otros.

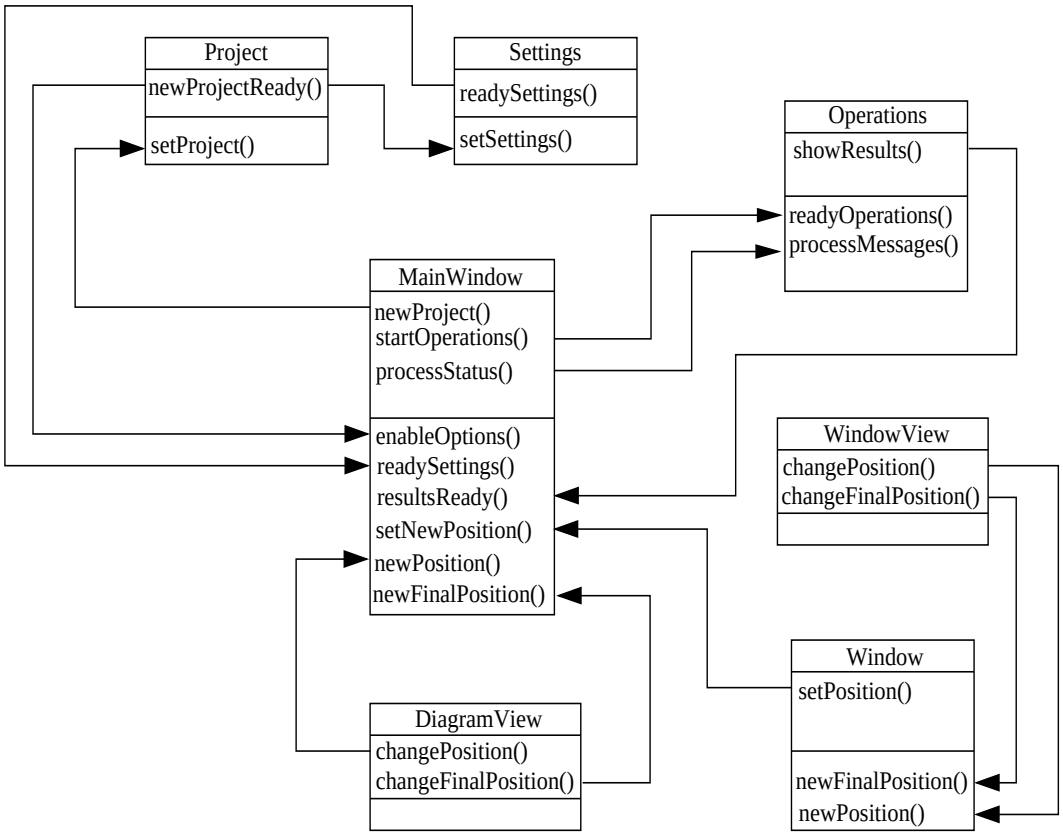


Figura 3.20: Diagrama de señales - ranuras (signals - slots)

Capítulo 4

Descripción del sistema QCTool

En esta sección se hace una descripción del sistema, mostrando las opciones con las que cuenta la interfaz gráfica. También se muestra la forma de realizar las operaciones básicas como: crear un proyecto, abrir una imagen, realizar el análisis y almacenar los resultados. Se muestran también los resultados finales, es decir, los resultados de analizar diferentes circuitos impresos.

4.1. Interfaz gráfica de usuario

La interfaz gráfica principal del sistema se muestra en la figura A.1. La interfaz se divide en tres partes: *menú de opciones*, *área de análisis de imágenes* y *área de trabajo*.

La figura 4.2 muestra el menú con las opciones para el análisis de imágenes.

- [1]. Nuevo proyecto
- [2]. Cerra proyecto
- [3]. Abrir imagen
- [4]. Guardar
- [5]. Propiedades
- [6]. Acercamientos
- [7]. Ayuda

Las opciones *Cerrar proyecto, Abrir imagen, Guardar, Propiedades y Acercamientos* inicialmente se encuentran deshabilitadas y pasan a estado habilitado cuando se crea un proyecto nuevo. Las propiedades de la imagen se establecen después de abrir una imagen nueva y estas propiedades pueden ser cambiadas en cualquier momento

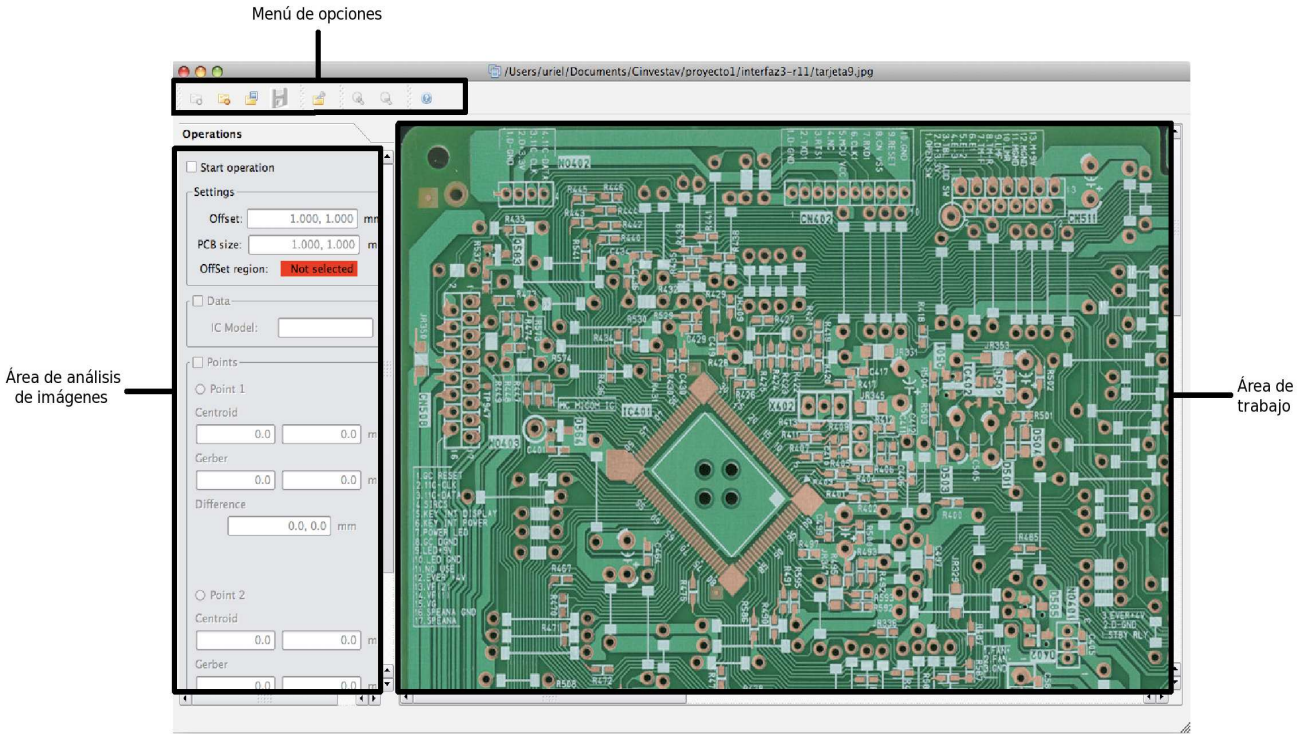


Figura 4.1: Interfaz gráfica principal del sistema

del análisis. La opción de ayuda del programa muestra el funcionamiento del programa así como también algunos ejemplos del análisis de circuitos impresos.

Figura 4.2: Interfaz gráfica principal del sistema

Creación de un proyecto

El análisis de una imagen requiere de la creación de un proyecto. Para acceder a esta acción, se puede dar un click en el botón correspondiente en la barra de menú o se puede acceder con la combinación de teclas **Ctrl + N**. La figura 4.3 muestra la ventana donde se debe proporcionar la información sobre el proyecto, de otro modo el análisis no se podrá llevar a cabo.

En este ejemplo, se ham insertado valores de prueba para poder continuar con el análisis del circuito impreso. Al dar click en el botón Aceptar, se crear automática-

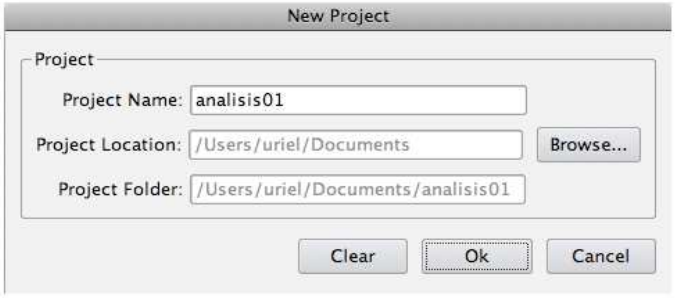


Figura 4.3: Ventana de creación del proyecto

mente un folder en la ruta especificada con el mismo nombre del proyecto y dentro se crea un archivo de texto también con el mismo nombre del proyecto.

Abrir una imagen

El siguiente paso es abrir la imagen con la cual se realizará el análisis. Se pueden utilizar imágenes en formato BMP, PNG y JPEG. El botón correspondiente a esta acción se encuentra en la barra de menú o se puede abrir una imagen con la combinación de teclas **Ctr + O**. La figura 4.4 muestra la ventana para seleccionar la imagen que se abrirá. En esta ventana solo se encuentran activas las imágenes con los formatos permitidos.

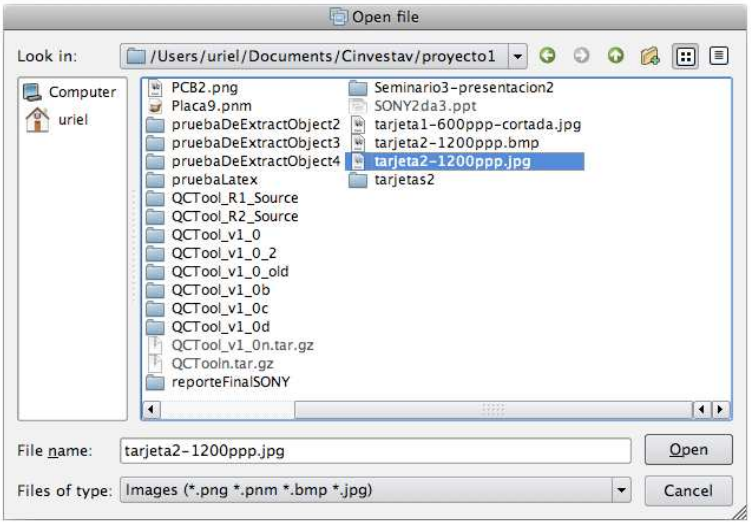


Figura 4.4: Cuadro de dialogo para la sección de la imagen

Después de seleccionar la imagen de trabajo, se abre automáticamente la ventana para establecer las propiedades del circuito impreso como: tamaño y posición del punto de referencia general. La figura 4.5 muestra esta ventana. En este caso, el tamaño del circuito impreso es de 324.15 mm $\times$ 244.25 mm y la posición de la referencia

general es (4.5, 233.5).

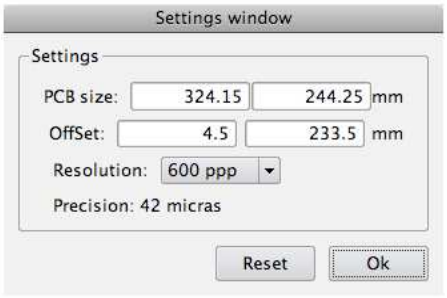


Figura 4.5: Cuadro de dialogo de propiedades de la imagen

Análisis de la imagen

Una vez que se ha abierto la imagen y sus propiedades han sido establecidas, se inicia el proceso de búsqueda y cálculo de los centroides en el circuito impreso. La figura 4.6 muestra una región seleccionada con un cuadro dentro del circuito impreso. Cuando se selecciona una región, automáticamente se abre una ventana que contiene un acercamiento de la región seleccionada, donde se realizará la búsqueda del círculo y el cálculo de su centroide, esto se puede observar en la figura 4.7.

La ventana que se muestra en la figura 4.7 permite realizar una mejor selección de la región que se analizará. Durante el análisis de la imagen, se pueden tener tres casos diferentes: (1) no se encontro ningun círculo de interés, (2) se localizaron más de dos círculos de interés y (3) se localizó el círculo de interés.

El primer caso se presenta cuando la región seleccionada no contiene ningun círculo de interés y en este caso se le avisa al usuario que vuelva a realizar la selección. El segundo caso se presenta cuando dentro de la región se encuentran más de dos círculos, por ejemplo, cuando dos CIs se encuentran muy cerca uno de otro y por consiguiente también sus puntos de referencia se encuentran muy cerca, en este caso también se le pide al usuario que realice nuevamente la selección. El tercer caso es el caso ideal y simplemente se muestra el resultado de análisis, es decir, el centroide del círculo localizado. El tamaño predeterminado del cuadro para seleccionar la región es de 50 pixeles, el tamaño mínimo es de 20 pixeles y el máximo es de 200 pixeles.

La figura 4.8 muestra los centroides de los puntos de referencia del circuito IC401. Estos resultados se mantienen temporalmente hasta que se realice el análisis de otra región del circuito impreso. Para no perder los resultados obtenidos, se tiene la opción de guardar toda la información mientras se realiza el análisis, esto se explica en la siguiente sección.

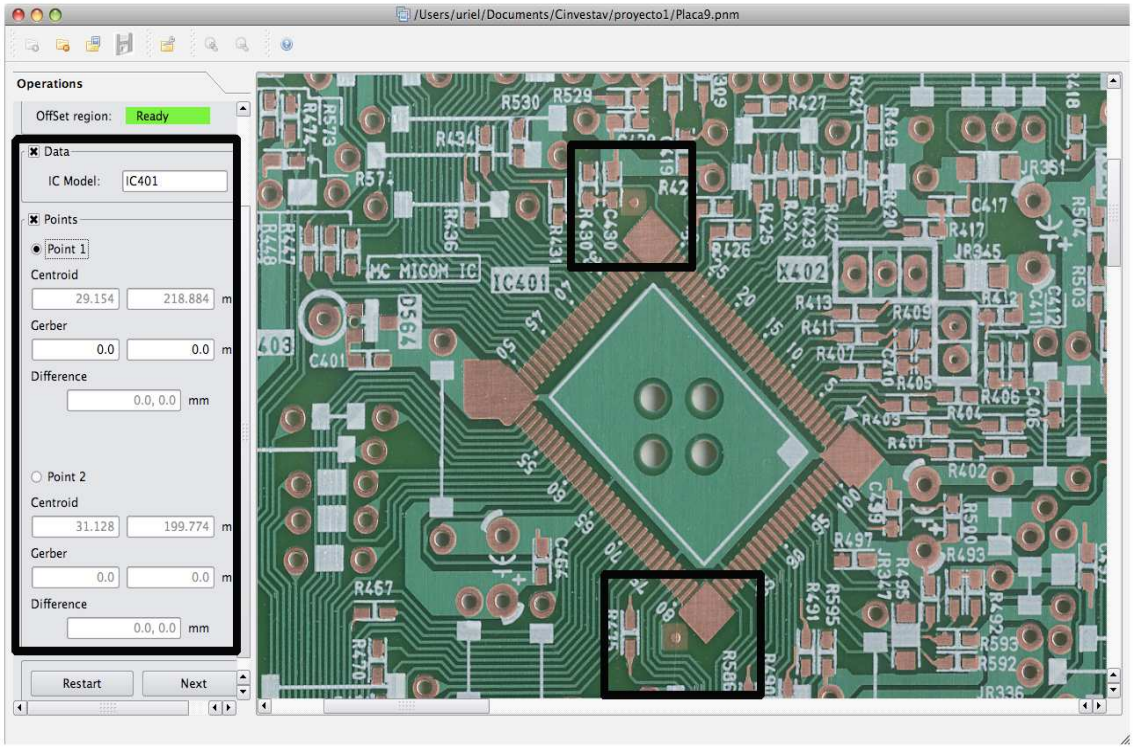


Figura 4.6: Selección de regiones y cálculo de centroides

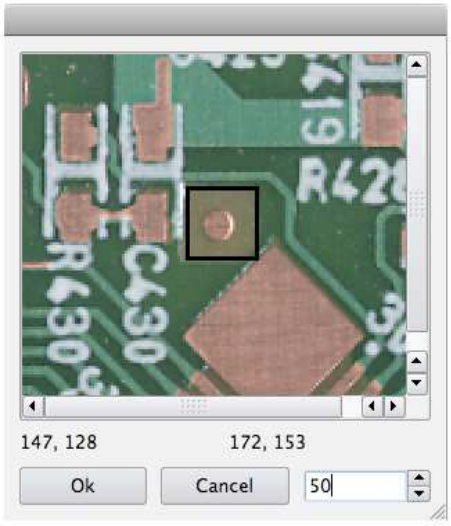


Figura 4.7: Acercamiento de la región seleccionada

En la figura 4.6 se observan unas regiones resaltadas; las regiones de la parte derecha corresponden a los puntos del circuito IC401 analizado, en la parte izquierda se muestran los valores de los centroides para los dos puntos del circuito IC401. Abajo de cada centroide se tienen los campos de texto donde se permite al usuario introducir

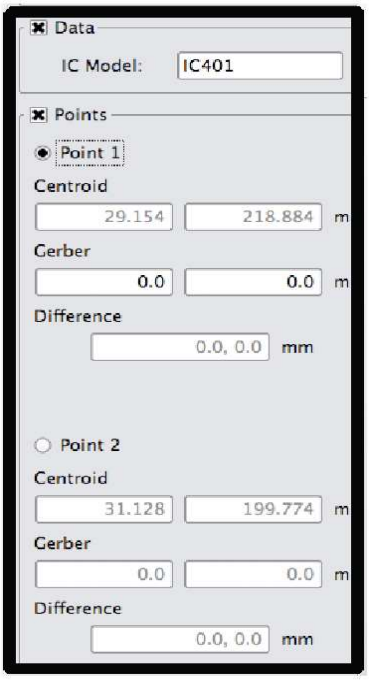


Figura 4.8: Resultados del análisis de un CI

los valores de referencia para obtener la diferencia de las mediciones, esta operación es opcional. Este análisis se puede repetir para todos los puntos de referencia de los circuitos integrados en el circuito impreso.

Almacenamiento de resultados

Finalmente, todos los resultados del análisis completo del circuito impreso pueden ser almacenados, lo cual sirve para un análisis posterior por parte del usuario y para conocer cual es el estado del circuito impreso, es decir, si la calidad del circuito impreso se encuentra dentro de los estándares garantizados por el fabricante. Los archivos son almacenados en el archivo de texto creado duranta la creación del proyecto. La figura 4.9 muestra el archivo de texto con los resultados del análisis. En este archivo, información que se muestra es: modelo del circuito integrado analizado, el valor de los centroides, el valor de referencia y la diferencia entre estos valores.

4.2. Resultados

Utilizando la herramienta desarrollada se han analizado diferentes circuitos impresos. Esta herramienta es mucho más fácil de utilizar que otros sistemas electrónicos que realizan la misma función pero que su operación es compleja. El cuadro 4.1 muestra algunos ejemplos de regiones de circuitos impresos analizados, así como los

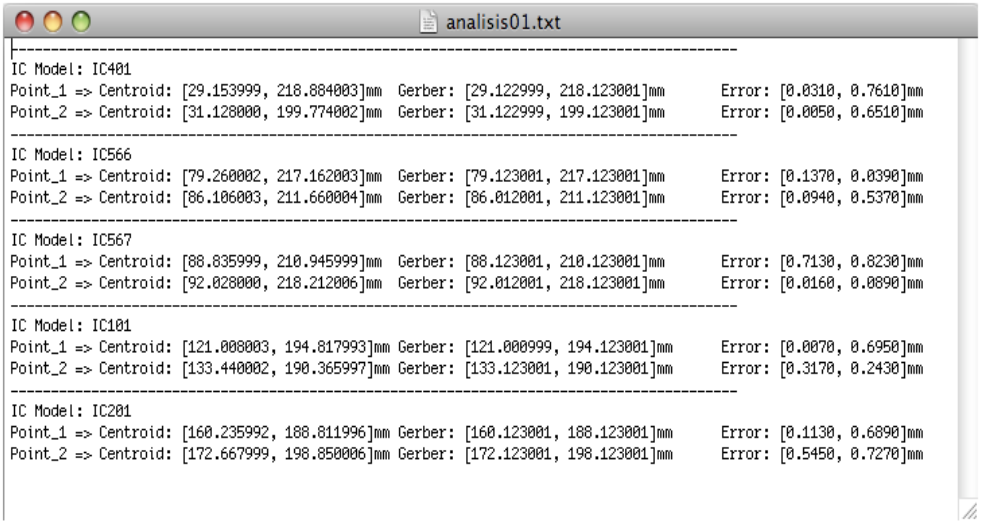


Figura 4.9: Archivo de texto con los resultados del análisis

resultados obtenidos.

En la columna **a** se muestra la región de análisis, la columna **b** muestra las imágenes cuando ha pasado por el preprocesamiento, la columna **c** muestra los perímetros de los círculos, la columna **d** muesta las secciones de los perímetros utilizadas en la reconstrucción, la columna **e** muestra los círculos reconstruidos, los cuales son utilizados en el cálculo de los centroides, la columna **f** muestra los centroides calculados por el sistema.

4.2.1. Aplicación de QCTool en circuitos impresos proporcionados por SONY Tijuana

La validación del sistema desarrollado permite verificar que los resultados calculados por el sistema se encuentran dentro de los parámetros establecidos. Para la validación, SONY Tijuana nos proporcionó dos circuitos impresos, indicando los puntos de referencia y sus posiciones. Estos circuitos impresos se muestran en las figuras 4.10 y 4.11. Los circuitos impresos se digitalizaron a una resolución de 1200 dpi. El resultado del análisis de cada circuito impreso se muestra en las figuras 4.12 y 4.13, que corresponden al circuito impreso de la figura 4.10 y 4.11 respectivamente.

Los resultados obtenidos fueron aceptados por los ingenieros de SONY Tijuana. Cabe mencionar que la precisión de los resultados depende de la resolución de las imágenes. Los tiempos de respuesta también dependen de las características del equipo de cómputo utilizado. En este caso, al ser QCTool un sistema de software fuera de línea, el tiempo no es un factor que afecte su desempeño.

Cuadro 4.1: Análisis de diferentes regiones de un circuito impreso

-	Región	Resultados				
	a	b	c	d	e	f
1	-					(14, 13)
2						(14, 12)
3						(11, 14)
4						(15, 14)
5						(10, 10)
6						(11, 12)
7						(11, 10)
8						(9, 11)
9						(12, 12)
10						(17, 16)
11						(10, 10)
12						(10, 10)
13						(10, 11)
14						(11, 13)
15						(12, 11)
16						(13, 15)

El sistema ha sido desarrollado para Windows XP, se han realizado pruebas de instalación en veriones de Windows 2000, Windows Me y Windows XP y no se han presentado problemas. Finalmente, se tiene un sistema de software que cuenta con los requerimientos y restricciones definidos por el usuario. El sistema ya ha sido utilizado por los ingeniero de SONY Tijuana y se han hecho las correcciones en los cambios y fallas reportadas, después de varias semanas de pruebas, el sistema se ha mantenido estable y funcionando correctamente de acuerdo a lo especificado.

Figura 4.10: Circuito impreso A

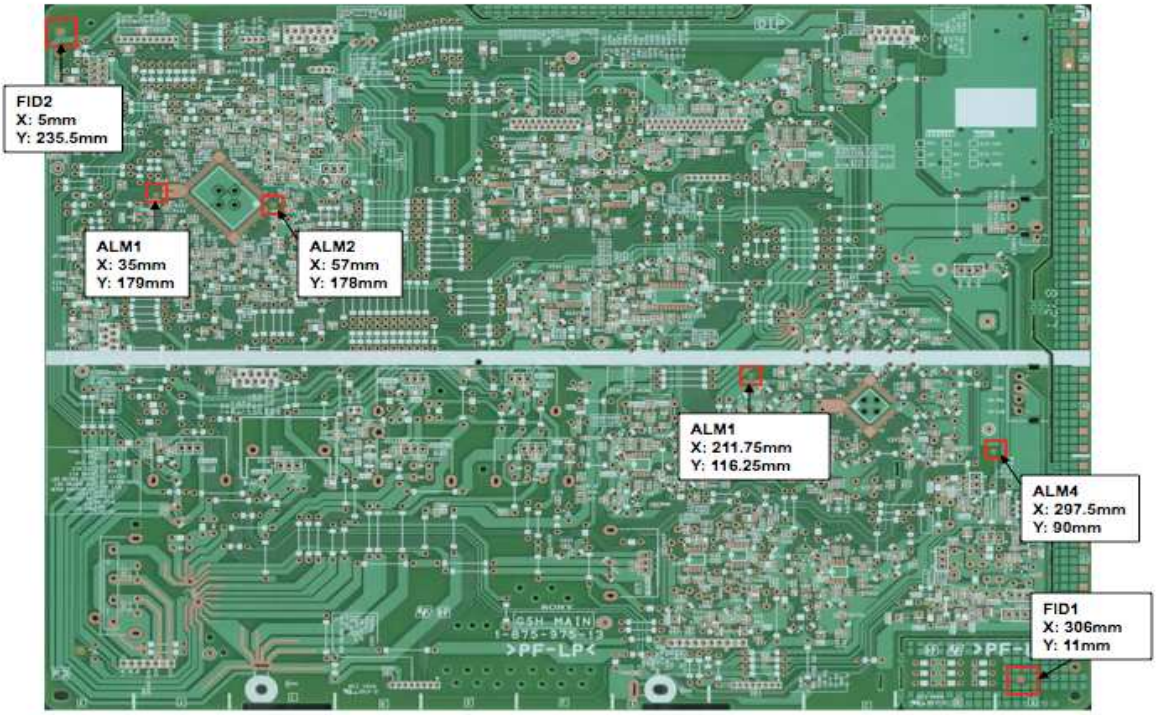


Figura 4.11: Circuito impreso B

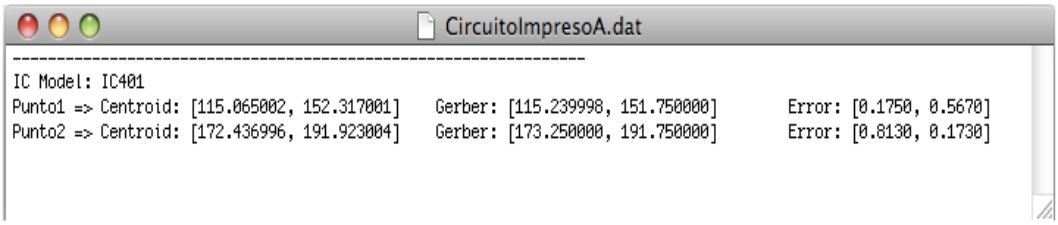


Figura 4.12: Resultados del análisis del circuito impreso A

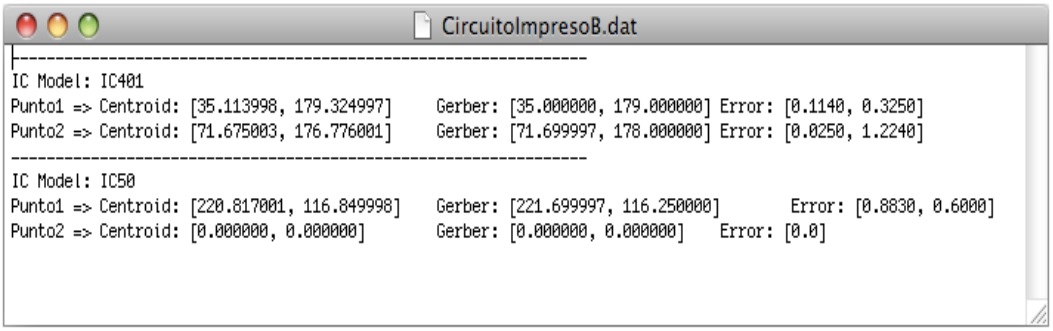


Figura 4.13: Resultados del análisis del circuito impreso B

Capítulo 5

Conclusiones

La tecnología SMT ha permitido tener cada vez más, componentes electrónicos y circuitos impresos de menor tamaño, aprovechando al máximo el espacio. En realidad, este avance tecnológico ha creado otros problemas como la precisión en el montaje de componentes eletrónicos, precisión en la fabricación, contaminación ó interferencia electromagnética, entre otros. En esta tesis se ha abordado el problema de la precisión en la fabricación de circuitos impresos. La solución que se ha desarrollado en este trabajo, para asegurar la precisión de las posiciones de las bases de los CI en el circuito impreso, ha sido utilizar diversas técnicas de Procesamiento Digital de Imágenes (PDI), desarrollando un software de aplicación específica, de acuerdo a las características del problema y a las necesidades del usuario. El software QCTool permite al usuario elegir diferentes regiones de trabajo dentro del circuito impreso que se esta analizado y localizar el centroide de los puntos de referencia críticos, es decir, que se ven más afectados durante la fabricación del circuito impreso. A través de esta tesis se ha mostrado que con el PDI se puede resolver el problema, en la medida de que se puede conocer el estado del circuito impreso y saber si se encuentra dentro de los estándares establecidos por el fabricante. Es posible realizar este proceso de análisis aún cuando las imágenes utilizadas presentan cierto grado de ruido ó baja calidad, debido a factores como ruido durante la digitalización, baja resolución, errores en el circuito impreso, entro otros.

El proyecto surgió en la empresa SONY Tijuana al darse cuenta de que el problema estaba presente en sus tarjeta electrónicas. El software QCTool ha sido entregado a SONY Tijuana y ha pasado por varias pruebas por parte de los ingenieros encargados de analizar el estado de las tarjetas y las correcciones solicitadas han sido realizadas. El sistema se encuentra en funcionamiento en SONY Tijuana, lo cual no ha producido ningún problema en cuanto a su instalación, ya que este es un sistema fuera de linea, lo cual no produce ningún tipo de retraso en la producción.

- QCTool funciona correctamente de acuerdo a las especificaciones

- QCTool ha sido entregado e instalado en los equipos de cómputo por los ingenieros de SONY Tijuana, sin presentarse ningun problema
- QCTool permite reducir los tiempos de producción, al reducir los tiempo de montaje de componentes electrónicos
- La precisión de los resultados que ofrece QCTool ha sido evaluada por los ingenieros de SONY Tijuana y se encuentra dentro de las especificaciones
- QCTool presenta una interfaz muy amigable, lo cual hace su uso muy intuitivo y fácil en comparación con los sistemas de hardware
- QCTool ha sido diseñado y probado obteniendo resultados satisfactorios en diferentes plataformas como Macintosh, Linux y Windows
- Este trabajo ha permitido realizar la publicación de un articulo en la revista JART (Journal of Applied Research and Technology) de la UNAM
- Se ha realizado también el registro de derechos de autor en el INDAUTOR (Instituto Nacional del Derecho de Autor). El registro se realizó el 2 de octubre del 2008 y el número de registro es 03-2008-091813132600-01. La hoja de registro de derechos de autor se muestra en el apéndice A
- El sistema QCTool actualmente se encuentra en funcionamiento en SONY Tijuana, en el apéndice **??** se muestra la hoja donde se confirma por parte de SONY Tijuana el uso y la utilidad del sistema QCTool

5.1. Trabajo futuro

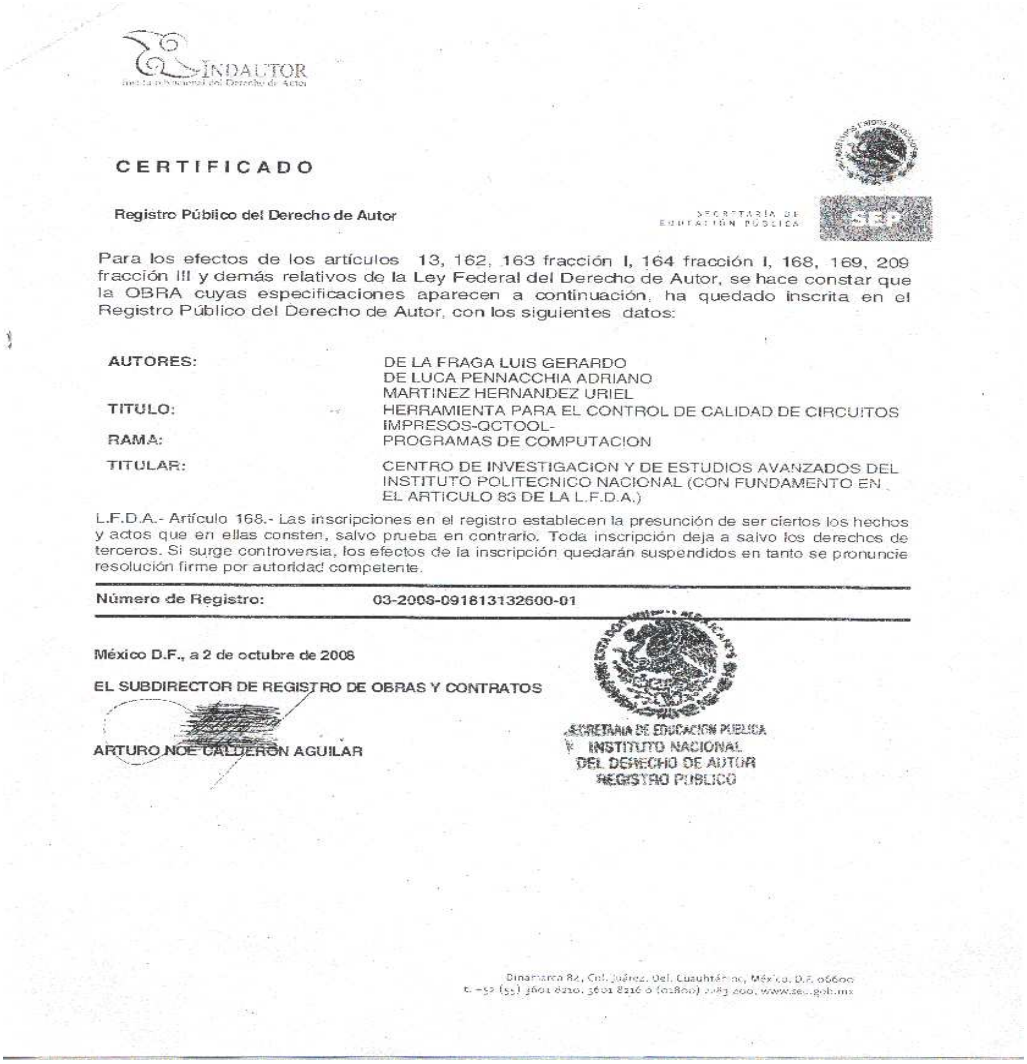
El sistema desarrollado puede reconocer y analizar objetos circulares, ya que esta es la forma de los marcas de referencia para el montaje de CI. Las características de esta aplcación que quedan como trabajo a futuro son:

- Reconocimiento de otros objetos como triángulos y cuadrados, para realizar esto, se deben analizar los tipos de objetos que se desean encontrar de la misma manera en como se realizó el análisis de los círculos, obtener los valores de los momentos y definir el rango para el reconocimiento de los objetos
- Realizar la lectura de los valores de referencia directamente del archivo Gerber, esto puede realizarse agregando un módulo de lectura y que reconozca e interprete los valores dentro del archivo
- Implementar un sistema totalmente automático, con lo cual se podría realizar un análisis sin intervención del ser humano

- Desarrollar un sistema que se puede incorporar directamente en la linea de producción
- Realizar una aplicación que resuelva más de un problema presente en la fabricación de circuito impresos y/o montaje de componentes electrónicos

Apéndice A

Registro del sistema QCTool



INDAUTOR

CERTIFICADO

Registro Público del Derecho de Autor

SECRETARÍA DE EDUCACIÓN PÚBLICA

Para los efectos de los artículos 13, 162, 163 fracción I, 164 fracción I, 168, 169, 209 fracción III y demás relativos de la Ley Federal del Derecho de Autor, se hace constar que la OBRA cuyas especificaciones aparecen a continuación, ha quedado inscrita en el Registro Público del Derecho de Autor, con los siguientes datos:

AUTORES:	DE LA FRAGA LUIS GERARDO DE LUCA PENNACCHIA ADRIANO MARTINEZ HERNANDEZ URIEL
TITULO:	HERRAMIENTA PARA EL CONTROL DE CALIDAD DE CIRCUITOS IMPRESOS-QCTOOL-
RAMA:	PROGRAMAS DE COMPUTACION
TITULAR:	CENTRO DE INVESTIGACION Y DE ESTUDIOS AVANZADOS DEL INSTITUTO POLITECNICO NACIONAL (CON FUNDAMENTO EN EL ARTICULO 83 DE LA L.F.D.A.)

L.F.D.A.- Artículo 168.- Las inscripciones en el registro establecen la presunción de ser ciertos los hechos y actos que en ellas consten, salvo prueba en contrario. Toda inscripción deja a salvo los derechos de terceros. Si surge controversia, los efectos de la inscripción quedarán suspendidos en tanto se pronuncie resolución firme por autoridad competente.

Número de Registro: 03-2008-091813132600-01

México D.F., a 2 de octubre de 2008

EL SUBDIRECTOR DE REGISTRO DE OBRAS Y CONTRATOS

ARTURO NOE CALDERON AGUILAR

SECRETARIA DE EDUCACION PUBLICA
INSTITUTO NACIONAL
DEL DERECHO DE AUTOR
REGISTRO PUBLICO

Dinamarca 82, Col. Juárez, Del. Cuauhtémoc, México, D.F. 06600
t. +52 (55) 3601 8210, 3601 8216 o (01800) 2083 200. www.sep.gob.mx

Figura A.1: Hoja de registro de derechos de autor

Bibliografía

[1] CIMOR. *Proceso de fabricación de circuitos impresos.* http://www.cimor.com.mx/proceso.htm.

[2] Wikipedia. *Circuito impreso.* http://es.wikipedia.org/wiki/Circuito_impreso, November 2008.

[3] Harley R. Myler and Arthur R. Weeks. *Computer Imaging Recipes in C.* Prentice Hall, 1993.

[4] Dwayne Phillips. *Image Processing in C.* R & D, 1994.

[5] *http://trolltech.com.*

[6] Eirik Eng. *Qt GUI Toolkit.* http://www.linuxjournal.com/article/201, November 1996.

[7] Boudewijn Rempt and Cameron Laird. *Visual Development with Qt 3.0.* http://www.linuxjournal.com/article/4749, June 2002.

[8] Bernd Jähne. *Digital Image Processing.* Springer, 2002.

[9] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing.* Prentice Hall, 2nd edition, 2001.

[10] Jorge Ortíz García and Adriano de Luca Pennacchia. *Análisis y clasificación de imágenes a color para la detección de capacitores y resistencias en tarjetas de montaje superficial.* Tesis de Maestría, CINVESTAV, México, Junio 2008.

[11] Eugenia Montiel, Alberto S. Aguado, and Mark S. Nixon. Texture classification via conditional histograms. *Pattern Recognition Letters*, pages 26(11): 1740–1751, August 2005.

[12] Ming-Kuei Hu. *Visual Pattern Recognition by Moment Invariants.* IRE Transactions on Information Theory, February 1962.

[13] Stuart Russell and Peter Norvig. *Inteligencia Artificial, un enfoque moderno.* Prentice Hall, 2nd edition, 2004.

[14] Miguel Ángel Sánchez Martínez and Adriano de Luca Pennacchia. *Diseño en FPGA de un circuito comparador de imágenes*. Tesis de Maestría, CINVESTAV, México, Julio 2005.

[15] Luis Francisco Barbosa Santillán and Adriano de Luca Pennacchia. *Localización de SMD mediante técnicas de visión artificial implementadas en hardware*. Tesis de Maestría, CINVESTAV, México, 2006.

[16] James D. Foley, Andries van Dam, Steven K. Feiner, and John F. Hughes. *Computer Graphics, principles and practice*. Addison Wesley, 2nd edition, 1996.

[17] Nicu Sebe, Ira Cohen, Ashutosh Garg, and Thomas S. Huang. *Machine Learning in Computer Vision*. Springer, 2005.

[18] Madhav Moganti, Fikret Ercal, Cihan H. Dagli, and Shou Tsunekawa. Automatic pcb inspection algorithms: A survey. *Computer Vision and Image Understanding: CVIU*, pages 63(2):287–313, 1996.

[19] Elias N. Malamas, Euripides G. M. Petrakis, Michalis Zervakis, Laurent Petit, and Jean-Didier Legat. *A survey on industrial vision systems, applications and tools*. February 2003.

[20] Duncan Wilson, Alistair Greig, John Gilby, and Robert Smith. *Uncertainty in automated inspection, the problems in the real world*.

[21] Kenneth R. Castleman. *Digital Image Processing*. Prentice Hall, 1996.

[22] A. H. Munsell. *A Color Notation*. Munsell Color Company, 1939.

[23] A. H. Munsell. *A Grammar of Color*. Van Nostrand-Reinhold, 1969.

[24] John L. Semmlow. *Biosignal and Biomedical Image Processing*. Marcel Dekker, 2004.

[25] John C. Russ. *The Image Processing Handbook*. CRC Press, 1999.

[26] T. Pavlidis. *Algorithms for Graphics and Image Processing*. Computer Science Press, 1982.

[27] Julio Cornejo Herrera, Adriana Lara López, Ricardo Landa Becerra, and Luis Gerardo de la Fraga. *Una librería para procesamiento de imagen: scimagen*. VIII Conferencia de Ingeniería Eléctrica, CINVESTAV, México, Septiembre 2002.

[28] Hermilio Sánchez. *Optimización de una medida de semejanza para objetos tridimensionales a partir de invariantes y transformaciones*. IIMAS, México, Marzo 2000.

[29] Raúl Pinto Elias and Humberto Sossa. Localización automática de objetos 2d y 3d en imágenes. *Computación y Sistemas*, pages 26–34, Febrero 2002.

[30] Mariana Vonzoy Villuendas, Aarón Ruiz Zúñiga, Mariko Nakano Miyatake, and Héctor Pérez Meana. *Marca de agua semifrágil para autenticación de imágenes digitales*. SEPI-ESIME, México.

[31] Dale Umbach and Kerry N. Jones. *A few methods for fitting circles to data*. IEEE Transactions on Instrumentation and Measurement, December 2003.

[32] Ian Sommerville. *Ingeniería de Software*. Addison Wesley, 6th edition.

[33] Paul M. Embree and Bruce Kimble. *C Language Algorithms for Digital Signal Processing*. Prentice Hall, 1st edition, 1991.

[34] Kurt Wall. *Programación en linux con ejemplos*. Prentice Hall, 1st edition, 2000.

[35] Jasmin Blanchette and Mark Summerfield. *C++ GUI Programming with Qt 3*. Prentice Hall, 2004.