Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias de la Electrónica

"Diseño e implementación de un sistema mecatrónico para la identificación de dispositivos en tarjetas PCB usando visión por computadora"

Tesis para obtener el título de
Licenciado en Ingeniería Mecatrónica

Presenta:
Juan Carlos Campeche Valencia

Asesores:
Dr. Aldrin Barreto Flores (FCE)
MC. Verónica Edith Bautista López (FCC)
Dr. Salvador E. Ayala Raggi (FCE)

Junio 2016

Agradezco a mis padres Teresa y Juan Carlos por darme su apoyo durante toda mi formación, a mis hermanas Karina y Diana por su amistad y apoyo incondicional.

Así mismo, agradezco a mis asesores de tesis, al Dr. Aldrin Barreto Flores, la M.C. Verónica Edith Bautista López y Dr. Salvador E. Ayala Raggi por ayudarme y guiarme durante todo este proyecto.

Al M.C. Carlos García Lucero y la empresa INTESC electronics & embedded, gracias por su apoyo al compartir su tecnología para poder realizar este proyecto.

Agradezco a los miembros del jurado, a la M.C. Ana María Rodriguez Dominguez, al M.C. Rodrigo Lucio Maya Ramirez y al M.C. José Francisco Portillo Robledo por los comentarios y recomendaciones que ayudaron a mejorar este trabajo.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

El incremento en el uso de la electrónica ha exigido que la identificación de fallas en el proceso de fabricación de circuitos impresos sea rápido y preciso, ya que los procesos manuales de detección de fallas son lentos y costosos pero principalmente son propensos a errores humanos, ya que la evaluación es subjetiva. El proceso de manufactura de un circuito impreso es muy complejo, es por esto que no existe un solo tipo de inspección para encontrar fallas pero la inspección se puede dividir principalmente en dos tipos: la de contacto eléctrico que puede encontrar cortocircuitos o circuitos abiertos y la inspección que no implica un contacto eléctrico que detecta la falta, la mala colocación de dispositivos, el exceso o falta de soldadura y el tamaño de los canales conductores, este tipo de fallas no pueden ser detectadas por la inspección de contacto eléctrico pero afectan el desempeño del circuito impreso [4].

La inspección óptica automática forma parte del tipo de evaluaciones que no es de contacto eléctrico y que hace uso de técnicas de visión por computadora para resaltar y encontrar posibles defectos en circuitos impresos, esta técnica también es muy usada para la colocación automática de componentes [5].

Para poder hacer una inspección óptica automática en un circuito impreso se necesita implementar un sistema de visión por computadora, un sistema de visión se compone básicamente de [6]:

- Sistema de iluminación
- Capturador de imágenes digitales
- Unidad de procesamiento
- Software de procesamiento digital de imágenes

- Elementos mecánicos

Dependiendo de las condiciones físicas y la complejidad del problema a resolver, los elementos de un sistema de visión pueden aumentar o disminuir, lo cual origina dispositivos con pocos elementos o máquinas que tengan un gran número de elementos para tener unas condiciones de trabajo muy específicas.

En la inspección óptica automática de circuitos impresos los elementos de un sistema de visión suelen ser muy específicos, ya que se evalúan componentes muy pequeños y los elementos que forman un circuito impreso tienen propiedades ópticas que obligan a usar un sistema de iluminación muy controlado, además se tienen que usar dispositivos que ayuden a aumentar la precisión de los algoritmos de procesamiento digital de imágenes.

Los sistemas de visión por computadora están avanzando pero no sólo se ha avanzado en sistemas de visión por computadora puros, estos se han complementado con sensores que permiten reducir las deficiencias que tienen los sistemas de visión y aumentan precisión y robustez en los sistemas.

1.1. Presentación de la problemática

1.1.1. Motivación

Los constantes avances tecnológicos a nivel mundial han traído un uso masivo de la electrónica en todas las áreas, lo que ha provocado un aumento en la producción de sistemas electrónicos, esto significa un aumento en la automatización en los procesos de producción para aumentar la cantidad y calidad de los sistemas. Para poder aumentar la velocidad y la calidad de producción de ensambles y componentes se han ido remplazando sistemas clásicos de inspección, por personas que revisan componente a componente por sistemas de visión por computadora, que ofrecen una mayor precisión y velocidad en la inspección y no se cansan. A pesar de que no existen cifras exactas de un porcentaje de errores en la producción de tarjetas PCB, si existe un gran desarrollo tecnológico para intentar minimizar los defectos de producción, ya que es una industria que tiene más de 2800 empresas en el mundo y continúan creciendo. [7]

La visión por computadora aplicada a la inspección automática es un área que ha crecido y se ha vuelto más compleja, ya que los procesos productivos también han aumentado su complejidad, se puede observar que la inspección automática óptica está presente en muchas áreas productivas, de las cuales destaca la soldadura, como puede ser por resistencia eléctrica,

donde se desarrolla un sistema que evalúa la calidad de los puntos de soldadura de una pieza que va soldada a una carrocería [8], también se puede observar que un problema ampliamente atacado es la soldadura blanda en el caso de dispositivos electrónicos donde se evalúa posición [9], otro problema ampliamente atacado es la evaluación geométrica de piezas, donde se busca identificar geometrías de manera muy rápida y precisa [10], la inspección usando visión por computadora también está presente en los problemas de la industria textil, donde se verifica la calidad de la tela [11] o incluso la industria alimenticia donde se analiza por medio de imágenes la calidad de los mariscos y peces [12].

1.1.2. Definición del problema

En este trabajo se implementó un sistema de inspección óptica, usando sistemas embebidos y software Open Source, lo cual evita el pago de licencias y permite un mayor aprendizaje. Este sistema es más económico en comparación con sistemas especializados.

El reto está en aprovechar una de las áreas con mayor crecimiento en los últimos años, los sistemas embebidos, es decir participar en la creación de sistemas que se basen en sistemas embebidos, donde la administración de recursos es crítica. Además se debe crear una interfaz que presente todos los resultados de manera clara para el usuario.

El sistema está constituido por una Raspberry Pi, la cual trabaja con el sistema operativo Raspbian que es la versión optimizada para sistemas embebidos de la distribución de Linux, Debian. La interfaz gráfica se desarrolló en el ambiente de desarrollo QT Creator, ya que permite la programación orientada a objetos en C++. El procesamiento digital de imágenes se realizó principalmente con las bibliotecas de OpenCV que son de código fuente abierto. Las imágenes fueron tomadas con una cámara especial para la Raspberri Pi, la cual es pequeña y tiene una resolución de 5 megapíxeles. Se implementó un sistema de iluminación que resalta los elementos de la tarjeta a inspeccionar y disminuye los reflejos. Se realizó una estructura que mantiene la posición de la cámara, del sistema de iluminación y que evita la entrada de alguna fuente externa de luz, ya que esto afecta el enfoque de la cámara y la calidad de las imágenes. Para que el proceso se realizara de manera automática se usaron los pines GPIO de la Raspberry Pi, los cuales interpretan la señal de un sensor que indica la presencia de la tarjeta PCB y controla un servomotor que posiciona la tarjeta PCB correctamente. Para simular una línea de producción se implementó una banda transportadora, la cual era controlada por la Raspberry Pi y movía las tarjetas hacia el área de inspección. El sistema es capaz de identificar la falta de dispositivos en una tarjeta y además hace la inspección de los pines de los circuitos integrados, mostrando los resultados de cada tarjeta en una interfaz gráfica.

En la Figura 1.1 se muestra el concepto general.

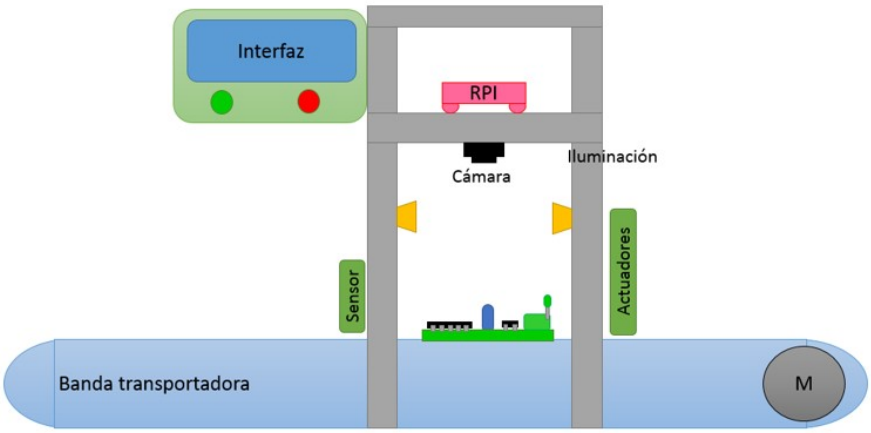


Figura 1.1: Esquema del proyecto.

1.1.3. Trabajos realizados en el área

La inspección automática de circuitos impresos usando visión por computadora es un área que continúa creciendo, es por esto que es necesario conocer los avances que ha tenido en los últimos años, para así conocer cuáles son los métodos más usados y los problemas más atacados. Para conocer estos avances se consultaron artículos de IEEE Xplore y de ScienceDirect, los cuales permiten conocer la actualidad de la inspección de circuitos impresos usando técnicas de visión por computadora.

A continuación se describen de manera breve los artículos consultados y en la Tabla 1.1 se presenta un resumen de estos.

En el primer trabajo relacionado, se hace la inspección de componentes THT (Through-Hole Technology) y SMD (Surface Mount Devices) de una tarjeta PCB [13]. Primero se analiza que métodos de umbralado presentan mejores resultados, donde los métodos de Pun y Otsu generan los mejores resultados para sus imágenes. Posteriormente se aplica la operación morfológica de apertura para determinar cortocircuitos y la operación de cierre para determinar la falta de pines en un circuito integrado.

En el siguiente trabajo, se hace una revisión muy extensa de inspección automática óptica en circuitos impresos enfocándose en la tecnología THT [4]. Se abarca la terminología, el proceso

de manufactura, los diferentes tipos de imágenes que se utilizan, las técnicas de iluminación, los tipos de defectos que se presentan durante el proceso de manufactura y los algoritmos que se tienen para la detección de estos defectos.

En [14], se hace la inspección dispositivos que tienen el empaquetado del tipo ball grid array (BGA). Este tipo de empaquetado permite tener más puertos de entrada y salida pero con la desventaja de que los pines soldados no se pueden ver. Es por esto que se usan sensores láser y de Rayos X para poder obtener las imágenes, en este artículo se evalúa la posición y la cantidad de la soldadura usando técnicas 2D y 3D. Principalmente se usa la técnica de segmentación por umbral, la operación de erosión y algoritmo de relleno por difusión para obtener diferentes características de cada bola de soldadura, como son la redondez, el diámetro y el centro.

En [15], se enfocan en la inspección de dispositivos del tipo SMD, donde se evalúa la posición de circuitos integrados, cabe mencionar que se hace uso de dispositivos y software de la empresa National Instruments. Para hacer la localización se busca una marca que permite la orientación del circuito impreso y posteriormente hace uso de los algoritmos de correlación del módulo NI Vision Development Module para hacer el reconocimiento.

En [16] se muestran dos métodos diferentes, uno para buscar errores en dispositivos SMD y el segundo para analizar las uniones de los dispositivos THT. El primer paso consiste en la transformada de Hough que determina el contorno de la imagen. Posteriormente se cambia a modelo de color YIQ, después se hace un proceso de umbralado y por último operaciones morfológicas para ubicar los componentes. En la parte inferior también hacen uso de la segmentación para determinar la posición y calidad de la soldadura.

En [9] se concentran en la posición de los dispositivos que se colocan con la técnica SMT para saber si están correctamente posicionados en los electrodos del PCB. Se muestra que en ocasiones es mejor trabajar con un solo plano de una imagen, ya que puede resaltar mejor algunas características. Posteriormente se comparan diferentes técnicas de umbralado. Para determinar la rotación de un componente se usa el método de proyección de píxeles sobre el eje vertical y se compara esta proyección con la de un componente colocado correctamente.

En [17] se hace uso de la técnica de correlación Matching pero en este caso usan técnicas de inteligencia artificial, las cuales hacen que la inspección sea más robusta, ya que pueden existir variaciones de iluminación, posición y ruido, las cuales causarían errores en sistemas clásicos pero no en sistemas de inteligencia artificial. Además se combinan con más métodos para poder realizar una correlación múltiple lo que reduce tiempos de cómputo.

En [18] se describe un sistema de visión controlado por un robot manipulador. El cual toma los PCB y los posiciona en una banda transportadora, para que sean transportados al sistema de detección de errores y posteriormente mediante algoritmos de segmentación y distancia entre imágenes se busquen las diferencias con un patrón perfecto. Dependiendo de los resultados otro robot manipulador lo coloca como elemento correcto o incorrecto.

Finalmente se presenta todo un sistema de visión por computadora especializado en la inspección de PCB [19] , el cual está compuesto por mecanismos que se encargan de trasladar y garantizar la posición del PCB y un sistema de iluminación muy controlado que garantiza una imagen con iluminación constante. El algoritmo para hacer la inspección se basa en la diferencia entre imágenes, donde las diferencias se resaltan.

Trabajo	Identificación de componentes	Descripción
[13]	THT y SMD	Umbralado por Pun y Otsu. Operaciones morfológicas de apertura y cierre.
[4]	THT	Técnicas de iluminación y algoritmos de detección.
[14]	BGA	Obtención de imagen por rayos X. Segmentación por umbral, erosión y relleno por difusión.
[15]	SMD	Software y hardware de National instruments. Algoritmos de correlación
[19]	SMD	Se presenta todo un sistema de visión especializado que hace comparación de imágenes.
[9]	SMD	Comparación de planos de una imagen, umbralado por Wang y Otsu. Determinación de inclinación por proyección de píxeles.
[16]	THT y SMD	Transformada de Hough, Modelo de color YIQ, umbralado y operaciones morfológicas
[17]	SMD	Algoritmos de correlación basados en técnicas de inteligencia artificial
[18]	SMD	Un robot manipulador posiciona los PCBs y se calcula la distancia entre imágenes.

Tabla 1.1: Resumen de trabajos revisados

1.2. Objetivos de la tesis

Objetivo general

Desarrollar un sistema mecatrónico capaz de identificar los dispositivos y evaluar los pines de los circuitos integrados de una tarjeta PCB en una linea de producción.

Objetivos específicos

- Desarrollar el sistema de visión por computadora en una Raspberry Pi
- Estudio e instalación de OpenCV en Raspberry PI
- Implementar una banda transportadora que simule una línea de producción
- Implementar algoritmos que identifiquen la falta de dispositivos superficiales en una tarjeta PCB
- Realizar la interfaz gráfica en QT Creator

1.3. Organización de tesis

La tesis está organizada en 5 capítulos. En el capítulo 2, se presentan los fundamentos teóricos de los sistemas que conforman este trabajo. En el capítulo 3, se describe a detalle el desarrollo de los sistemas que forman el sistema de inspección. En el capítulo 4, se presentan los resultados de las pruebas realizadas con el sistema. Finalmente en el capítulo 5, se presentan las conclusiones y el trabajo futuro.

Capítulo 2

Fundamentos Teóricos

En este capítulo se describirán los fundamentos teóricos de los componentes del sistema de visión. Lo anterior se realiza con el objetivo de tener una perspectiva que permita entender de mejor manera el proyecto. Primero se describen los fundamentos de la visión por computadora, como son el procesamiento digital de imágenes y los sistemas de iluminación. Por último, se describe el software y hardware usados en este proyecto.

2.1. Visión por computadora

La visión le permite a muchos seres vivos identificar objetos con el objetivo de usar esa información para tomar decisiones, esto es posible mediante el procesamiento de imágenes que realiza el cerebro. La visión por computadora es el estudio de los procesos que realiza el sistema de visión de los seres vivos para poder entenderlos y desarrollar sistemas que tengan capacidades similares o mejores. La visión por computadora tiene la finalidad de extraer información del mundo físico a partir de imágenes, utilizando una computadora.

La visión por computadora trata de imitar el sentido de la vista, todo este proceso por lo general se define en cuatro etapas [20] como se muestra en la Figura 2.1.

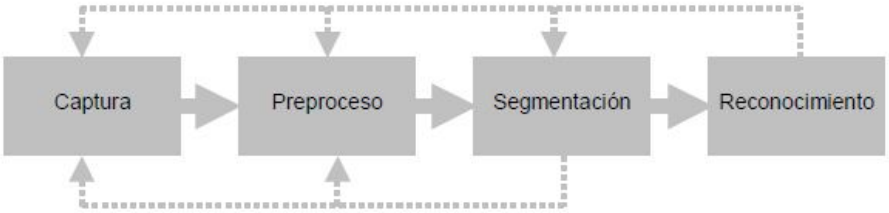


Figura 2.1: Etapas de un sistema de visión por computadora.

La etapa de captura consiste en la adquisición de la imagen digital mediante un sensor. La etapa de preprocesamiento es la encargada de eliminar o reducir los elementos indeseados en una imagen, como pueden ser ruido o reflejos, en esta etapa mediante transformaciones geométricas se corrigen los efectos de distorsión causados en la toma de imágenes. En la etapa de segmentación se aíslan los elementos importantes de una imagen. Por último se interpreta la información de la imagen para así tomar decisiones. Dependiendo el problema también cambia la ejecución de las etapas, ya que muchas veces los resultados no son aceptables y se tiene que regresar a etapas anteriores.

2.1.1. Adquisición y representación de imágenes

El proceso de adquisición de imágenes no sólo implica obtener una imagen mediante una cámara, esta etapa determina cómo se va a atacar el problema y que métodos de procesamiento digital de imágenes se van a usar. En la adquisición de imágenes se elige el tipo de cámara y la iluminación.

Para determinar cual es la cámara apropiada, se deben de tener en cuenta el tamaño del sensor de la cámara, distancia entre la cámara y el objeto y el campo de visión. Para poder determinar estas características es necesario conocer el proceso de captura de imágenes.

Captura de imagen

Para capturar una imagen se necesita un dispositivo que plasme una parte del espectro electromagnético. Para hacer la captura existen principalmente dos alternativas [20]: los dispositivos pasivos que en su mayoría se basan en el principio de la cámara obscura y los dispositivos activos que se basan en el escaneo.

Escaneo

El escaneo a grandes rasgos se compone de un elemento que actúa de emisor y un elemento receptor. El elemento emisor manda un haz de luz (generalmente láser) que choca con la imagen escaneada, posteriormente el reflejo es detectado por el elemento receptor, este proceso se repite hasta obtener la escena deseada [20]. El escaneo no sólo es usado para obtener imágenes, ya que también es usado para obtener escenas 3D.

Principio de cámara obscura

Los dispositivos basados en el principio de cámara obscura [21] tienen la ventaja de ser más simples, rápidos y se parecen más al sistema de visión de los seres humanos. Estos dispositivos son llamados cámaras y pueden ser analógicos y digitales. Se llama cámara obscura porque el dispositivo está formado por una caja cerrada (cámara), la cual permite el paso de luz a través de un orificio ubicado en una de sus paredes, la luz que entra se proyecta sobre la pared opuesta, obteniendo una imagen invertida de la escena que está fuera de la caja cerrada. En 1550 Gerolamo Cardano propuso el uso de un cristal biconvexo para aumentar la luminosidad de la imagen. Con esta aportación se obtuvo el modelo básico de las cámaras fotográficas. Las propiedades de este modelo se explican con el modelo de lente fina.

El modelo de lente fina se compone por una **lente biconvexa** de grosor despreciable, la cual recoge la luz de una escena y la proyecta de una manera nítida a una superficie. Como podemos ver en la Figura 2.2, la lente biconvexa posee un eje de simetría denominado **eje óptico** y un **centro óptico** formado por la intersección del eje óptico y el plano de simetría de la lente. Todo haz de luz que pase por el centro óptico continuará en línea recta y todos los haces que incidan perpendicularmente al plano de simetría se cortan en un punto llamado **foco** situado sobre el eje óptico. Así la **distancia focal** es la distancia del foco al centro óptico de la imagen. Todos los haces de luz de un punto externo **P** son proyectados a un punto conocido como **punto de formación de la imagen** y un conjunto de reflejos de puntos a una misma distancia forman el **plano de formación de imagen**. Así los puntos equidistantes de una figura forman en el plano de formación de imagen la misma figura pero invertida [21].

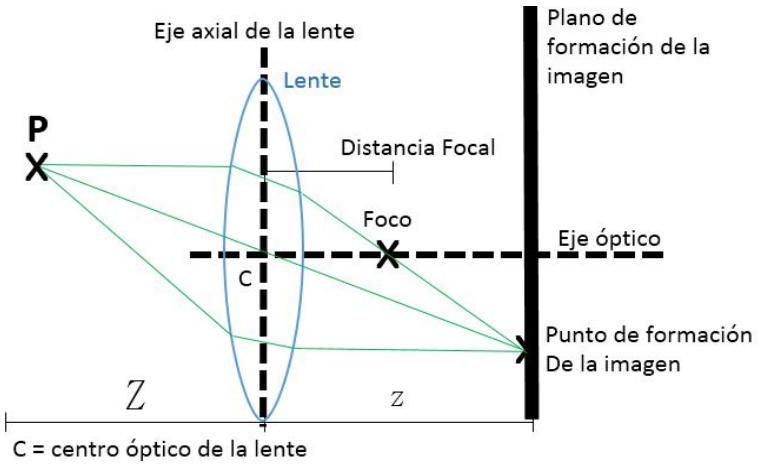


Figura 2.2: Modelo de lente fina.

Si llamamos Z a la distancia del punto P a la lente, z a la distancia del lente al plano de formación de imagen y f a la distancia focal, se cumple la relación (2.1.1). De esta relación se deduce que no es posible hacer una máxima concentración de haces que provienen de diferentes distancias Z. Este proceso de concentración es conocido como enfoque [21].

$$\frac{1}{z} - \frac{1}{Z} = \frac{1}{f} \tag{2.1.1}$$

Por propósitos prácticos la gran mayoría de los dispositivos de captura tienen una distancia focal variable. Al variar la distancia focal se consigue un mayor o menor acercamiento del objeto, este efecto es comúnmente llamado zoom. Como se mencionó anteriormente hay un intervalo de distancias donde se puede obtener una máxima concentración de los haces de luz de un punto (enfoque), este intervalo es comúnmente conocido como **zona nítida**. Si un objeto se encuentra fuera de la zona nítida su proyección de haces no será la óptima provocando que su proyección sea borrosa como se muestra en la Figura 2.3.

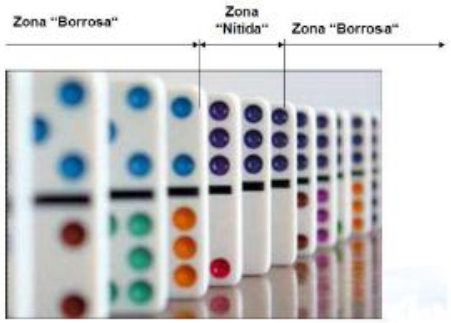


Figura 2.3: Profundidad de campo.

Los elementos de una cámara varían según su aplicación, si se necesita controlar la entrada de luz que penetra a la cámara se debe usar una cámara con diafragma variable pero si la distancia de trabajo es el problema se deben usar tubos de extensión.

2.1.2. Cámara digital de fotografía y video

El modelo de lente fina describe de manera muy simple cómo se proyectan los haces de luz sobre una superficie, este principio es el mismo para las cámaras analógicas y digitales. La principal diferencia está en el elemento que conforma el plano de formación de la imagen [20]. En la cámara analógica el plano está formado por una superficie sensible a la luz llamada película, mientras que en la cámara digital se tiene una matriz de elementos fotosensibles. El funcionamiento de los elementos fotosensibles se basa en el efecto fotoeléctrico, es decir los dispositivos emiten cargas eléctricas proporcionales a la cantidad de fotones que reciben. Estos elementos se basan en tecnologías CCD (Charge Coupled Device) o CMOS (Complementary Metal Oxide Semiconductor). La tecnología CMOS es más barata, ya que se basa en elementos semiconductores clásicos pero la tecnología CCD presenta mayor resolución, es por esto que es la más usada actualmente. Los valores de carga que se generan por la incidencia de luz son infinitos y de poca magnitud, por lo que necesitan ser amplificados y posteriormente discretizar su valor mediante un convertidor analógico digital generando así una imagen digital.

2.1.3. Imagen digital

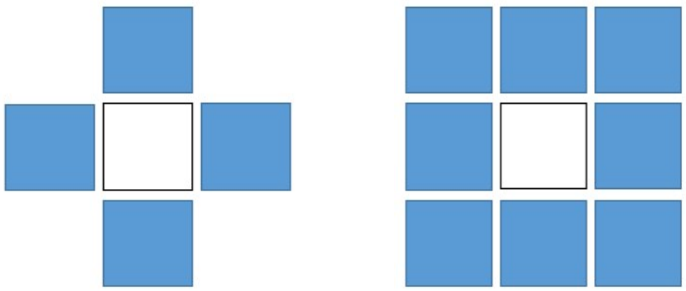
Conocer el proceso de formación de una imagen permite conocer las características que permiten capturar imágenes de calidad. En esta sección se describen conceptos de procesamiento digital de imágenes, desde cómo se representa una imagen digital hasta operaciones sobre la misma.

Una imagen digital es una función bidimensional I(x, y) que cada par de coordenadas (x, y) tiene un elemento llamado píxel (Del inglés "pixel"que abrevia a "picture element") [22]. El modelo matricial es el más común para representar una imagen digital, donde M representa a las filas y N a las columnas de la matriz, como muestra la ecuación (2.1.2).

$$I(x,y) = \begin{bmatrix} I(0,0) & I(1,0) & \cdots & I(N,0) \\ I(0,1) & I(1,1) & \cdots & I(N,1) \\ \vdots & \vdots & \ddots & \vdots \\ I(0,M) & I(1,M) & \cdots & I(N,M) \end{bmatrix} \tag{2.1.2}$$

Cada píxel ofrece información sobre una región elemental en la imagen digital. Dependiendo el tipo de imagen será la información que contenga cada píxel. En imágenes en escala de grises la información que proporcionan es el brillo y en imágenes a color la información corresponde a la intensidad de cada una de las componentes del modelo de color usado. Los píxeles tienen una longitud determinada llamada **Profundidad de color** [22]. En **imágenes binarias** cada píxel tiene un tamaño de 1 bit pudiendo tener sólo dos valores (uno y cero). Si un píxel tiene un tamaño de 8 bits cada píxel puede tener 256 variaciones de intensidad (0 a 255). Las llamadas imágenes de color completo usan 24 bits para cada píxel, es decir 16 777 216 variaciones de color. Es común el uso de imágenes de 32 bits, donde 24 bits son para representar el color completo y los 8 restantes representan la transparencia.

Un objeto es un conjunto de píxeles conectados por algún tipo de vecindad y que además están rodeados por un conjunto de píxeles que tienen el color de fondo de la imagen. La vecindad está definida como la relación de posición que tiene un píxel con respecto a los más cercanos a él [22]. Existen dos tipos de vecindad de píxel: La vecindad de 4-vecinos que sólo considera a los píxeles que se encuentran arriba, abajo, derecha e izquierda y la vecindad 8-vecinos que considera a los correspondientes a la vecindad 4-vecinos y los 4 que están en diagonal al píxel en cuestión como se muestra en la Figura 2.4.



(a) 4-vecinos (b) 8-vecinos

Figura 2.4: Tipos de vecindad

2.1.4. Modelos de Color

Los colores se pueden describir de muchas maneras diferentes. Un modelo de color define los colores mediante componentes de color específicos [20]. El modelo más utilizado es el modelo de color RGB (Red-Green-Blue), ya que es el modelo de color que se usa para desplegar los colores en la computadora. Este modelo crea los colores haciendo una mezcla aditiva de los tres colores primarios, es decir cuanto más rojo, verde y azul se agregue, más se aproxima al color blanco [21]. Una imagen $I(x, y)$ del tipo RGB está compuesta de tres sub imágenes $I_R(x, y)$,$I_G(x, y)$, $I_B(x, y)$ llamadas **canales**, los cuales son del mismo tamaño que $I(x, y)$ como se muestra en la Figura 2.5.

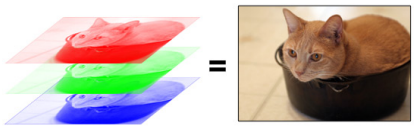


Figura 2.5: Canales de una imagen RGB.

El modelo de color HSV o HSB representa la información del color a través de 3 componentes: la tonalidad (Hue), la saturación (Saturation) y el valor (Value) o brillo (Brightness) [20]. La tonalidad de color se especifica de 0 a 360 grados mientras que los componentes saturación y brillo se definen de 0 a 100 % como se puede ver en la Figura 2.6.

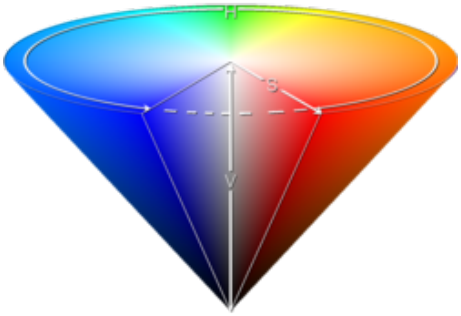
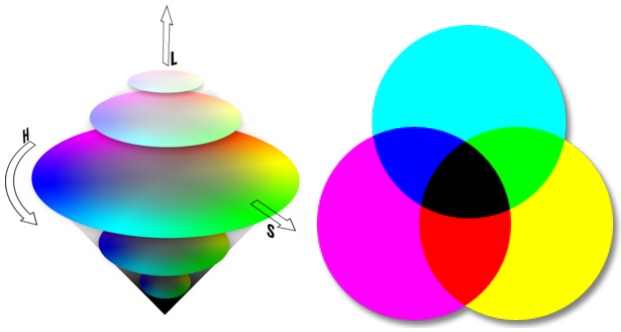


Figura 2.6: Modelo de color HSV

El modelo HSL (Hue-Saturation-Lightness) es muy similar al HSV pero tiene la diferencia de que la parte obscura es simétrica a la parte clara. El modelo CMYK hace la composición del color con cuatro componentes [20]: el Cyan, Magenta, Yellow y blacK, donde se puede obtener

cualquier color variando de 0 a 100 % cada componente. Este modelo a diferencia del RGB es un modelo sustractivo, es decir a más color agregado más se aproxima al color negro [20].



(a) HSL (b) CMYK

Figura 2.7: Otros modelos de color

Cada modelo de color presenta características que son aprovechadas por diferentes aplicaciones. El modelo de color RGB es el usado por escáneres y monitores por lo que es el modelo natural en computación. Los modelos HSV y HSL son los modelos más intuitivos en la selección del color, ya que los colores se presentan como un arco iris y no como una suma de componentes. El modelo CMYK es el usado para crear imágenes que se van a imprimir debido a que es el modelo usado por las impresoras.

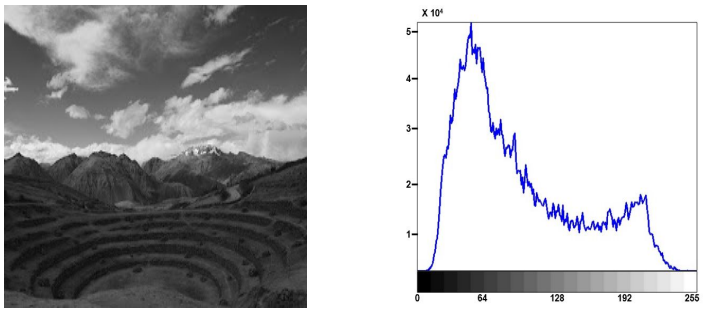
2.1.5. Procesamiento digital de imágenes

El procesamiento digital de imágenes puede definirse como la operación de imágenes digitales mediante computadora [20]. El tipo de procesamiento puede ser de bajo, medio y alto nivel [22]. El procesamiento de bajo nivel también es conocido como preprocesamiento, en este nivel de hacen procesos de suavizado, umbralado, eliminación de ruido y realce de bordes. En el procesamiento de medio nivel se definen procesos tales como la definición de límites y extracción de características. En el alto nivel se establecen relaciones entre los objetos de la imagen.

2.1.6. Histograma

Los histogramas permiten conocer de manera cuantitativa las características de una imagen para así poder evaluar los posibles métodos de procesamiento que se pueden aplicar sobre la imagen.

Un histograma es una distribución que describe la frecuencia con la que se presenta los valores de intensidad de cada píxel en una imagen [22]. El histograma de una imagen a escala de grises (**Histograma de luminosidad**) es el mejor ejemplo para describir las características que nos muestra este. Una imagen a escala de grises contiene valores de intensidad (luminancia) que van del 0 (Negro) al 255 (Blanco), por lo que el eje horizontal de su histograma será el intervalo [0, 255]. El eje horizontal para cualquier imagen es [0, k-1], donde k es la profundidad de color (en bits). El eje vertical de un histograma se compone por la frecuencia de cada valor de intensidad de la imagen como muestra la Figura 2.8



(a) Imagen a escala de grises

(b) Histograma de (a)

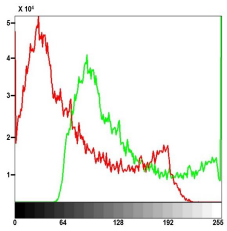
Figura 2.8: Ejemplo de Histograma

Como podemos observar el histograma no proporciona información sobre el origen de cada píxel. Sin embargo nos muestra características de una imagen, como son la iluminación, el contraste y la dinámica [22]. Haciendo un análisis de estas tres características podemos identificar problemas ocurridos durante la toma de imagen que pueden causar problemas en las etapas de procesamiento de la misma.

Iluminación de una imagen

Una imagen tiene una iluminación errónea cuando el extremo izquierdo o derecho del histograma no están ocupados por ningún píxel [22]. Si el extremo izquierdo se encuentra vació, significa que la imagen se tomó con una iluminación excesiva pero si el extremo derecho se encuentra vacío entonces la iluminación fue insuficiente provocando una imagen obscura como se muestra en la Figura 2.9

(a) Exceso de iluminación



(b) Histograma: Verde (a) , Rojo (c)

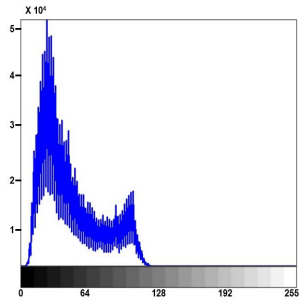
(c) Falta de iluminación

Figura 2.9: Errores de imagen por mala iluminación

Contraste de una imagen

El contraste es el intervalo que se forma entre el máximo valor de intensidad y el mínimo de una imagen [22]. Si este intervalo es pequeño se dice que la imagen tiene un bajo contraste, es decir la diferencia de valores de intensidad entre los objetos y el fondo de una imagen es muy poca. Por el contrario si el contraste es alto la diferencia entre los objetos y el fondo es alta. En procesamiento digital de imágenes es altamente recomendado usar imágenes con buen contraste, ya que los bordes de un objeto se definen mejor, lo que facilita el procesamiento y los resultados son más precisos. En la Figura 2.10 se muestra una imagen con poco contraste como se puede ver la identificación de objetos es más difícil que la Figura 2.8.

(a) Bajo contraste



(b) Histograma

Figura 2.10: Imagen con bajo contraste

Dinámica de una imagen

La dinámica se entiende como el número de valores de intensidad que se presentan en una imagen [22]. Una buena dinámica significa que todos los valores de la profundidad de color son usados. Una buena dinámica evita que se pierda calidad de la imagen en las etapas de procesamiento.

2.1.7. Operaciones homogéneas de una imagen

Se llaman operaciones homogéneas cuando a una imagen $I(x, y)$ se le aplica una función $f[I(x, y)]$ que dependa únicamente del valor del píxel y no de su posición ni su vecindad para obtener una nueva imagen $I'(x, y)$ [20]. Este tipo de operaciones se aplican a cualquier tipo de imagen ya sean de un canal (binarias y escala de grises) o de varios canales (diferentes modelos de color).

Cambios de contraste o iluminación
Modificar el brillo o el contraste de una imagen es una operación recurrente cuando las condiciones de toma de imagen no fueron las óptimas y se quiere mejorar toda la imagen. La ecuación 2.1.3 muestra la operación homogénea [20] para modificar el contraste y la iluminación de manera proporcional en toda la imagen.

$$f(x, y) = c * I(x, y) + b \tag{2.1.3}$$

Donde **c** modifica el valor del contraste y **b** el valor del brillo. Por ejemplo si se obtuvo una imagen muy obscura se debe sumar un valor de b positivo. Por el contrario si es necesario oscurecer la imagen se debe sumar un valor de b negativo. Si se necesita aumentar el contraste el valor de c debe ser mayor que 1 y menor que 1 si se quiere reducir.

Inversión de una imagen
La inversión de una imagen consiste en obtener los complementos de los píxeles de la imagen en el intervalo [0, profundidad de color (p_{max})] [22]. La ecuación 2.1.4 nos muestra la operación que se aplica sobre cada píxel de la imagen $I(x, y)$.

$$f_{inv}(p) = p_{max} - p \tag{2.1.4}$$

Esta operación como su nombre lo dice genera la imagen inversa. Por ejemplo, un píxel con valor 0 (negro), tendrá un nuevo valor de 255 (blanco) como se muestra en la Figura 2.11.

(a) Imagen (b) Imagen invertida

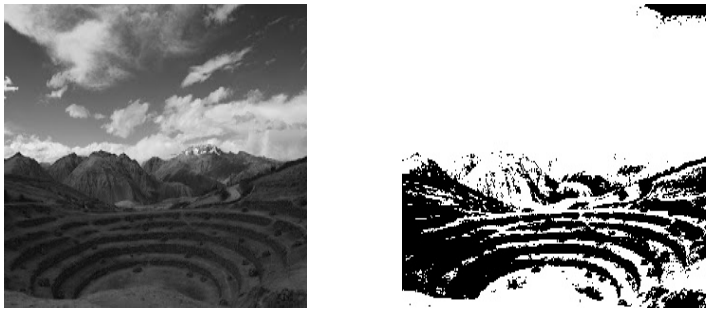
Figura 2.11: Ejemplo de inversión de imágenes

2.1.8. Segmentación por umbral

Esta operación es el método de segmentación más simple de todos y se usa para diferenciar píxeles de interés del resto. Esta diferenciación se hace a partir de un valor llamado **umbral** o **thresh**, el cual es definido por el usuario [23]. La segmentación por umbral se define formalmente como muestra la ecuación (2.1.5):

$$f_{th}(p) = \begin{cases} p_0 & \text{si } p > Umbral \\ p_1 & \text{si } p \leq Umbral \end{cases} \tag{2.1.5}$$

Donde p_0 y p_1 pueden tener el valor que quiera el usuario. Por ejemplo si $p_0 = 0$ y $p_1 = 255$ y el umbral es 50 en una imagen a escala de grises se obtiene el resultado mostrado en la Figura 2.12.



(a) Imagen a escala de grises (b) Imagen segmentada con umbral = 50

Figura 2.12: Ejemplo de segmentación por umbral

La segmentación por umbral es una de las operaciones más usadas en procesamiento digital de imágenes, ya que permite eliminar los píxeles que no son de interés. La segmentación por umbral tiene tantas variantes como el usuario quiera, ya que se puede definir más de un umbral para hacer una separación de 3 regiones.

2.1.9. Operaciones entre imágenes

En procesamiento digital de imágenes en ocasiones es necesario obtener una nueva imagen a partir de dos imágenes. La nueva imagen $I'(x, y)$ se forma a partir de operaciones de píxel entre una imagen $I_A(x, y)$ y otra imagen $I_B(x, y)$ como muestra la Figura 2.13.

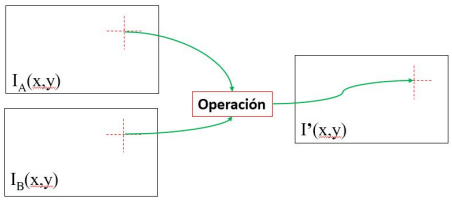


Figura 2.13: Operaciones entre imágenes

Como se muestra en la Figura 2.13, la posición de cada píxel es tomada en cuenta al realizar la operación [21]. Las operaciones pueden ser lógicas (**AND** y **OR**) y aritméticas (**suma** y **resta**). Una aplicación para la suma es la superposición de imágenes para lograr un efecto de combinación. La resta de imágenes es usada para buscar las diferencias entre dos imágenes iguales y su uso más común es identificar movimiento en secuencias de imágenes. Las operaciones lógicas se realizan entre imágenes binarias y un uso común es la generación de máscaras para aplicarlas en imágenes usando la operación lógica AND como se muestra en la Figura 2.14.

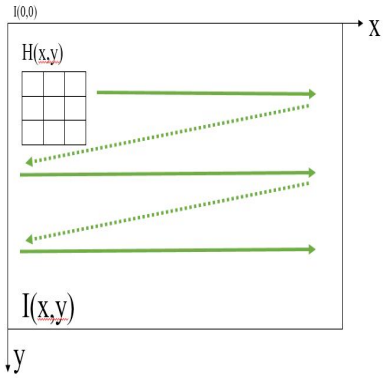
(a) Imagen (b) Imagen Binaria (c) Resultado de operación AND

Figura 2.14: Ejemplo de aplicación de máscaras

2.1.10. Filtros espaciales

Los filtros espaciales son operaciones de píxel realizadas sobre imágenes con el objetivo de mejorar la imagen. A diferencia de las operaciones morfológicas los filtros espaciales consideran la posición y los vecinos de cada píxel [22]. Un filtro espacial está formado por una matriz de coeficientes $H(x, y)$ conocida como **mascarilla** o **kernel**, la cual tiene un tamaño nxn. El proceso de filtrado espacial se describe en cuatro pasos que se pueden apreciar en la Figura 2.15:

1. La mascarilla $H(x, y)$ se coloca en el origen de la imagen $I(x, y)$ formando una región $R(x, y)$
2. Se multiplican elemento a elemento la mascarilla y la región formada, y se realiza una suma de todos los elementos.
3. El resultado obtenido es el píxel de la nueva imagen $I'(x, y)$ tomando la posición de $H(x, y)$ en $I(x, y)$
4. La mascarilla se recorre al siguiente píxel formando una nueva región y el proceso se repite hasta recorrer toda la imagen.



(a) Proceso de filtrado

H(-1,-1)	H(-1,0)	H(-1,1)
H(0,-1)	H(0,0)	H(0,1)
H(1,-1)	H(1,0)	H(1,1)

(b) Mascarilla o kernel

Figura 2.15: Filtrado espacial

En general el proceso matemático de filtrado espacial de una imagen $I(x, y)$ de tamaño M $\times$ N con una mascarilla de tamaño m $\times$ n es descrito por la ecuación (2.1.6)

$$I'(x,y) = \sum_{i=-a}^{a} \sum_{j=-b}^{b} (I(x+i, y+j)) * H(i,j) \tag{2.1.6}$$

Donde $a = (m-1)/2$ y $b = (n-1)/2$ y para generar una imagen completamente filtrada la ecuación (2.1.6) se debe aplicar de $x = 0, 1..., M-1$ y $y = 0, 1..., N-1$.

Convolución

El proceso de pasar la mascarilla $H(x, y)$ por toda la imagen es conocido formalmente como **CORRELACIÓN** que a su vez es un proceso muy parecido a la **CONVOLUCIÓN** con la diferencia de que la mascarilla es rotada 180° (sobre su eje coordenado) [20].

La operación matemática detrás de todos los filtros lineales es la convolución y su resultado depende sólo de la matriz $H(x, y)$, ya que la imagen $I(x, y)$ es constante. Por esta razón los filtros lineales heredan todas las propiedades de la convolución lineal como son: **conmutatividad**, **linealidad** y **asociatividad**.

El nivel de acción de un filtro depende al igual que la operación de correlación únicamente de la matriz $H(x, y)$, es decir de sus componentes y su tamaño. Los filtros cuyos elementos mantengan una relación lineal son llamados **filtros lineales** y de no mantener esta relación son

llamados **filtros no lineales** como son los filtros de mediana, filtros de mínimos y máximos entre otros [22]. El tamaño del filtro influye en el número de vecinos que toma en cuenta para realizar el filtrado. La Figura 2.16 nos muestra cómo afecta el tamaño de un filtro aunque sus elementos sean los mismos.

(a) kernel = 3 × 3 (b) kernel = 7 × 7

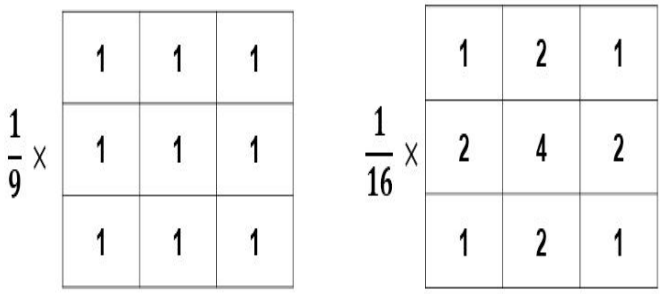
Figura 2.16: Filtros de suavizado con diferentes tamaños

Los filtros también se clasifican por su función, si los filtros realizan un suavizado de los elementos de la imagen son llamados **filtros de suavizado** como el mostrado en la Figura 2.16. Si su función es resaltar los detalles de una imagen los filtros son llamados **filtros de realce**. Es importante mencionar que existe un número infinito de filtros ya que las combinaciones y el tamaño de filtros es infinito.

Filtros de suavizado

Los filtros de suavizado son usados en el preprocesamiento de una imagen con el objetivo de suavizar una imagen o reducir el ruido de la misma. Suavizar se refiere a la eliminación o reducción de detalles pequeños (como líneas o curvas) de una imagen con el objetivo de resaltar a los objetos grandes [21].

El **filtro de promedio** es un filtro con una matriz $H(x, y)$ de tamaño n×n con el valor de todos sus elementos igual a uno. Este filtro obtiene el valor de cada nuevo píxel sacando el promedio del píxel en cuestión y sus vecinos. Debido a que todos los coeficientes del filtro son uno, este filtro tiene la característica de eliminar el ruido de una imagen y suavizarla pero tiene la desventaja de disminuir los bordes de la imagen. En la Figura 2.17 se muestra un filtro con tamaño 3×3. Donde la constante afuera de la matriz depende del número de elementos y en este trabajo sólo se presentan los más significativos.



(a) Kernel filtro de promedio

(b) kernel filtro Gaussinano

Figura 2.17: Filtros de suavizado

El **filtro Gaussiano** se basa en la función de Gauss bidimensional que se muestra en la ecuación (2.1.7).

$$G_\sigma(x, y) = e^{\frac{x^2+y^2}{2\sigma^2}} \tag{2.1.7}$$

Donde la desviación estándar (σ) representa el radio de cobertura de la función de Gauss. En la Figura 2.17 se muestra un filtro de Gauss con tamaño 3×3. El filtro de Gauss no diluye los bordes, debido a que los elementos de su kernel no tienen el mismo peso en la operación, ya que como se observa en la Figura 2.17 el píxel central tiene más peso que sus vecinos.

Filtros de realce

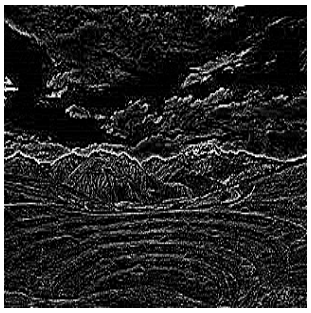
Los filtros de realce también son conocidos como **filtros de diferencia** esto se debe a que entre los elementos de la matriz $H(x, y)$ existen elementos negativos y positivos, lo que ocasiona que existan sumas y restas en el proceso de filtrado [21]. La operación de resta genera que la diferencia de intensidad entre píxeles vecinos sea resaltada, lo que ocasiona que estos filtros sean usados para la detección de bordes en una imagen.

El **filtro de Laplace** o de **sombrero mexicano** es un filtro que se basa en la ecuación (2.1.8) donde σ es la desviación estándar. En este tipo de filtro el único valor positivo es el central. En la Figura 2.18 se muestra una mascarilla del filtro de Laplace con un tamaño 5×5 y el resultado que se obtiene al aplicarlo.

$$M_\sigma(x, y) = \frac{1}{\sqrt{2\pi}\sigma^3}(1 - \frac{x^2 + y^2}{\sigma^2})e^{\frac{x^2+y^2}{2\sigma^2}} \tag{2.1.8}$$

0	0	-1	0	0
0	-1	-2	-1	0
-1	-2	16	-2	-1
0	-1	-2	-1	0
0	0	-1	0	0

(a) Kernel filtro de Laplace



(b) Imagen Filtrada por Laplace

Figura 2.18: Filtro de Laplace

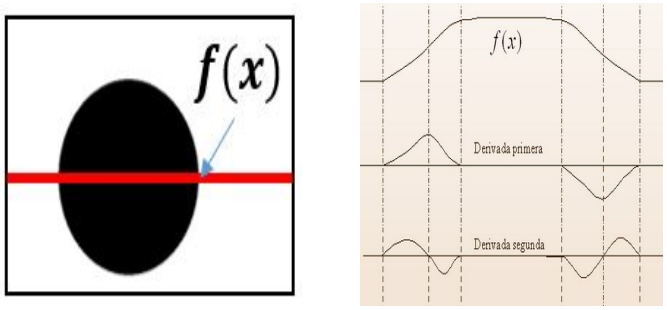
Los filtros que ocupan derivadas para detectar los cambios de intensidad entre píxeles son llamados de manera general **filtros basados en el gradiente** y existen gran variedad de ellos y de gran complejidad [22]. Para entender el funcionamiento básico de los filtros basados en el gradiente se considera únicamente un renglón con cambios de intensidad ($f(x)$) de una imagen a escala de grises. La derivada en una imagen digital se define entre píxeles adyacentes $(x, x+1, x+2)$. La derivada de primer orden se describe en la ecuación (2.1.9).

$$\frac{\partial f}{\partial x} = f(x+1) - f(x) \tag{2.1.9}$$

La primera derivada es cero en zonas de intensidad constante y distinta a cero en zonas con variaciones de intensidad. Mientras más grande sea la variación de intensidad la derivada será de mayor magnitud. La derivada de segundo orden para un renglón ($f(x)$) se define como muestra la ecuación (2.1.10).

$$\frac{\partial^2 f}{\partial x^2} = f(x+1) + f(x-1) - 2f(x) \tag{2.1.10}$$

La segunda derivada es igual a cero en zonas con intensidad constante y a lo largo de rampas con pendientes constantes. Es diferente de cero en escalones, al inicio y al final de las rampas. La Figura 2.19 muestra la respuesta de la primera y la segunda derivada en una imagen.



(a) Imagen analizada (b) Primera y segunda derivada

Figura 2.19: Resalte de bordes ocupando derivadas

Los filtros de realce basados en derivadas son muy sensibles al ruido, ya que el ruido también genera cambios de intensidad que son considerados en el cálculo de la derivada [22].

La ecuación (2.1.11) muestra la primera derivada en el sentido de los renglones y la ecuación (2.1.12) la primera derivada en el sentido de las columnas.

$$H_x^D = \begin{bmatrix} -0{,}5 & 0 & 0{,}5 \end{bmatrix} \tag{2.1.11}$$

$$H_y^D = \begin{bmatrix} -0{,}5 \\ 0 \\ 0{,}5 \end{bmatrix} \tag{2.1.12}$$

A partir de las primeras derivadas se definen **los operadores de Pewitt y Sobel**, el **operador de Roberts**, **el gradiente** y usando las segundas derivadas se define el operador **Laplaciano**.

2.1.11. Operaciones morfológicas

Los filtros espaciales son capaces de modificar los valores de intensidad de los canales de una imagen, además pueden modificar esquinas, bordes y eliminar elementos pequeños. Sin embargo estas modificaciones que hacen sobre las estructuras de una imagen no son controladas. Las operaciones morfológicas influyen sobre las estructuras de una imagen de manera controlada [20]. Estas operaciones usan **estructuras de referencia (H)** como las que se muestra en la Figura 2.20.

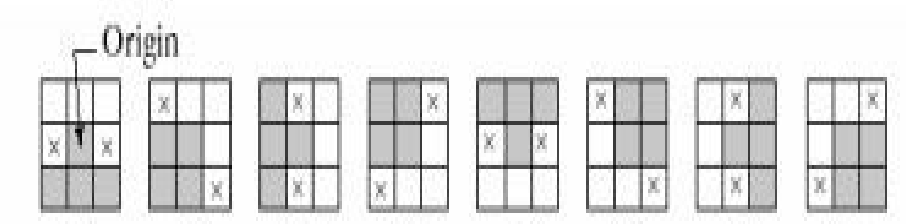


Figura 2.20: Estructuras de referencia de tamaño 3×3

La composición y el tamaño de las estructuras de referencia o kernel determinan cómo es la modificación sobre la imagen. La estructuras de referencia se utilizan igual que un filtro espacial, es decir se hace un barrido por toda la imagen (convolución). La diferencia es que las operaciones que se realizan entre la estructura de referencia y la zona cubierta por la misma son **operaciones lógicas** y no algebraicas. Las operaciones morfológicas básicas son la erosión y la dilatación. A partir de estas operaciones se definen las operaciones de apertura y cierre [23].

Erosión
A grandes rasgos esta operación consiste en disminuir los bordes de un elemento. Esta operación se hace conforme la ecuación (2.1.13).

$$I \ominus H = \{(x', y') = (x + i, y + i) \mid (x', y') \in P_I, \forall (i, j) \in P_H\} \tag{2.1.13}$$

Donde cada píxel en la imagen original será considerado como uno si todos los píxeles debajo del kernel son uno, de otra manera el nuevo valor será cero (erosión). De modo que el tamaño y el tipo erosión depende de la configuración de la estructura de referencia y su tamaño. En la Figura 2.21 se muestra la operación de erosión con kernel de 5×5 compuesto totalmente de unos.

(a) Imagen original (b) Imagen erosionada

Figura 2.21: Operación morfológica de erosión [1]

Dilatación
Esta operación es totalmente contraria a la erosión y su principal objetivo es aumentar capas a los bordes de una estructura. La ecuación (2.1.14) describe este proceso.

$$I \oplus H = \{(x', y') = (x + i, y + i) \mid (x', y') \in P_I, (i, j) \in P_H\} \tag{2.1.14}$$

Donde un píxel debajo del kernel es uno y coincide con al menos un uno de la estructura de referencia, entonces el nuevo valor será uno. En la Figura 2.22 se muestra esta operación realizada con un kernel de 5×5 compuesto totalmente de unos.

(a) Imagen original (b) Imagen dilatada

Figura 2.22: Operación morfológica de dilatación [1]

Apertura
La apertura se define como una erosión seguida por una dilatación [23] como muestra la ecuación (2.1.15).

$$I \circ H = (I \ominus H) \oplus H \tag{2.1.15}$$

Esta operación es comúnmente usada para remover elementos pequeños del fondo (ruido), además cierra los espacios entre estructuras cercanas como muestra la Figura 2.23.

(a) Apertura (b) Apertura para remover ruido

Figura 2.23: Operación morfológica de apertura [1]

Cierre
Se define como una dilatación seguida por una erosión [23] como muestra la ecuación (2.1.16).

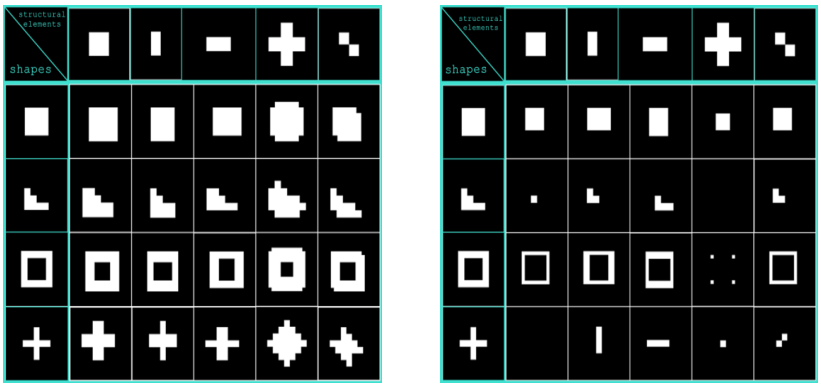
$$I \bullet H = (I \oplus H) \ominus H \tag{2.1.16}$$

Principalmente se usa para cerrar pequeños espacios dentro de una estructura o remover objetos que son más pequeños que la estructura de referencia como se muestra en la Figura 2.24.

(a) Cierre (b) Cierre para remover elementos

Figura 2.24: Operación morfológica de cierre [1]

Tipos de estructuras de referencia
La estructura de referencia determina cómo se va a modificar un objeto [20]. Existen infinitas estructuras de referencia, ya que el usuario define una diferente para cada problema. En la Figura 2.25 se muestra cómo se afecta un mismo elemento por diferentes elementos estructurales.



(a) Dilatación (b) Erosión

Figura 2.25: Diferentes elementos estructurales [1]

Como podemos observar existen elementos que aumentan o reducen las dimensiones de un objeto y pueden modificar las esquinas. También pueden ser usados para eliminar elementos muy específicos.

2.1.12. Comparación y reconocimiento de imágenes

La identificación de patrones es una de las áreas más importantes del procesamiento digital. El método más básico para reconocer un patrón (**template**) dentro de una imagen se basa en la resta de imágenes o **distancia entre imágenes** [22]. Si la resta es igual a cero significa que las imágenes son iguales. Al igual que un filtro espacial se recorre el patrón $R(x, y)$ por toda la imagen $I(x, y)$ pero se hace una resta en lugar de una multiplicación como muestra la ecuación (2.1.17), en lugar de generar una imagen filtrada se genera una imagen $D(x, y)$ que contiene todos los resultados de todas las restas o distancias. Los valores más cercanos a cero significan que se encontraron coincidencias dentro de la imagen.

$$D(x, y) = \sum_{i=-a}^{a} \sum_{j=-b}^{b} I(x + i, y + j) - H(i, j) \tag{2.1.17}$$

Este método presenta muchas deficiencias, ya que pequeños cambios de iluminación modifican completamente el resultado. Para corregir las deficiencias se propone usar como base **la distancia euclidiana** [22] como se muestra en la ecuación (2.1.18).

$$D^2(x,y) = \sum_{i=-a}^{a}\sum_{j=-b}^{b}(I(x+i,y+j) - H(i,j))^2 =$$

$$\underbrace{\sum_{i,j}(I(x+i,y+j))^2}_{A(x,y)} + \underbrace{\sum_{i,j}H(i,j)^2}_{B} - \underbrace{2\sum_{i,j}I(x+i,y+j) * H(i,j)}_{C(x,y)} \tag{2.1.18}$$

Donde $A(x,y)$ es cada sección de la imagen $I(x,y)$ que se va a comparar con el patrón y únicamente depende del desplazamiento. B es constante, ya que sólo es el patrón elevado al cuadrado. $C(x,y)$ es la operación conocida como **correlación** descrita por la ecuación (2.1.6) anteriormente. Analizando la ecuación (2.1.18) se observa que la correlación $C(c,y)$ es el factor que determina el parentesco de imágenes, ya que B sólo considera al patrón y $A(x,y)$ a la imagen analizada. A partir de esto se define el **Coeficiente de Correlación** definido en la ecuación (2.1.19) y es la base de la técnica **Template Matching** [20].

$$CC(x,y) = \frac{C(x,y)}{\sqrt{A(x,y) * B}} \tag{2.1.19}$$

El coeficiente de correlación toma las características de la distancia euclidiana entre imágenes y las relaciona para mejorar la técnica de comparación de imágenes, ya que los factores $A(x,y)$ y B afectan directamente en el cálculo de comparación. Lamentablemente la correlación cruzada sigue siendo inexacto cuando se presentan cambios de iluminación en la imagen [22].

Para mejorar los resultados se agrega en los cálculos el valor promedio del patrón ($\bar{H}$) que sólo se calcula una vez y el factor $\bar{I}(x+i,y+j)$ que es el valor promedio de cada región de coincidencia con el patrón. Al agregar estos factores se forma el **Coeficiente de Correlación Normalizado (NCC)** [20] mostrado en la ecuación (2.1.20).

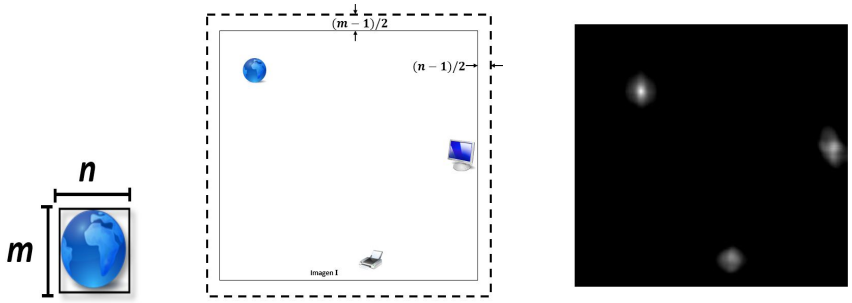
$$NCC(x,y) = \frac{\sum_{i,j}[I(x+i,y+j) - \bar{I}(x+i,y+j)] * [H(i,j) - \bar{H}]}{[\sum_{i,j}(I(x+i,y+j) - \bar{I}(x+i,y+j))^2]^{\frac{1}{2}} * [\sum_{i,j}(H(i,j) - \bar{H})^2]^{\frac{1}{2}}} \tag{2.1.20}$$

Donde:

$$\bar{I}(x+i,y+j) = \tfrac{1}{m*n}\textstyle\sum_{i,j} I(x+i,y+j) \quad \bar{H} = \tfrac{1}{m*n}\textstyle\sum_{i,j} H(i,j) \tag{2.1.21}$$

Matemáticamente el método NCC obtiene valores en el intervalo $[-1,+1]$, donde el valor $+1$ indica una coincidencia perfecta entre patrón y la sección de la imagen, el valor disminuye

a menor coincidencia hasta -1. El intervalo $[-1, +1]$ se adapta mejor a una imagen a escala de grises, es decir el valor -1 corresponde a 0(Negro) y +1 corresponde a 255(Blanco) como se muestra en la Figura 2.26.



(a) Template H (b) Imagen $I(x, y)$ (c) Resultado NCC

Figura 2.26: Template Matching

Se puede apreciar que para aplicar el método NCC es necesario agregar columnas y renglones $I(x, y)$ para garantizar correspondencia entre el patrón y la imagen [20]. Para solucionar este problema existen diferentes alternativas pero la más usada consiste en reflejar la imagen en cada extremo como muestra la Figura 2.27.

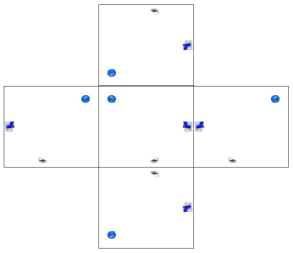


Figura 2.27: Agregar columnas y renglones

2.2. Sistema de Iluminación

Como se describió en la sección 2.1.1 Adquisición y representación de imágenes, una imagen no podría existir sin una fuente de luz. Sin embargo la presencia de luz no es suficiente cuando se habla de fotografía profesional o más tratándose de aplicaciones de visión por computadora, ya que se tienen que obtener imágenes donde los objetos se muestren claros y definidos. Los sistemas de iluminación permiten controlar la forma en que los objetos reflejan la luz a la cámara [24]. Si se tiene una iluminación correcta se obtendrá una imagen que permita un análisis fácil y preciso. Por el contrario si la iluminación no es la adecuada el análisis será difícil y en ocasiones imposible. Por lo tanto la elección de la iluminación es igual de importante que la elección de la cámara. Para hacer una buena selección de un sistema de iluminación es necesario conocer la respuesta de un material expuesto a una radiación electromagnética, en especial la luz visible.

Radiación electromagnética

La radiación electromagnética es un tipo de onda que está formada por tres componentes: un campo magnético (**B**), un campo eléctrico (**E**) y una dirección de propagación [25]. Los campos eléctrico y magnético son perpendiculares entre si y perpendiculares a la dirección de propagación como muestra la Figura 2.28.

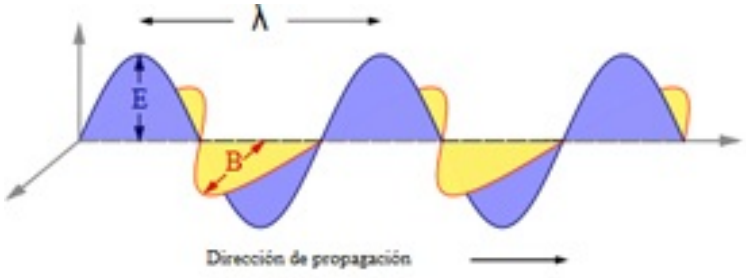


Figura 2.28: Componentes de una onda electromagnética.

La luz, las ondas de radio, los rayos X, rayos gamma, etc. son diversos tipos de ondas electromagnéticas que difieren únicamente en su longitud de onda (λ) y su frecuencia (f), ambas relacionadas con la velocidad $c = \lambda * f$. En la Figura 2.29 se muestra el espectro electromagnético y los nombres asociados a cada intervalo correspondiente a la longitud de onda.

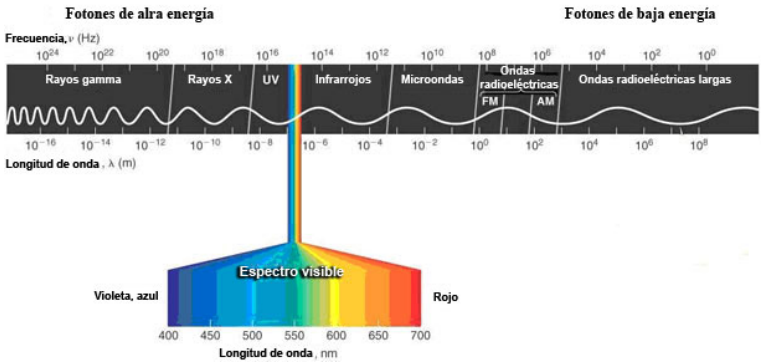


Figura 2.29: Espectro electromagnético.

El ojo humano puede percibir únicamente un intervalo conocido como **luz visible** que corresponde a longitudes de onda entre 400 y 780 nm aproximadamente. El color percibido se determina por la longitud de onda [25]: por ejemplo un color violeta corresponde a una longitud de onda de 400 nm mientras que un rojo corresponde a 650 nm.

Propiedades ópticas de los materiales

Cuando la luz pasa de un medio a otro pueden ocurrir diferentes fenómenos. Una parte se transmite a través del medio, otra es absorbida y otra parte es reflejada [25] como se muestra en la Figura 2.30. La relación (2.2.1) indica que la intensidad I_0 del haz que incide en el medio debe de ser igual a la suma de intensidades transmitidas (I_T), absorbidas (I_A) y reflejadas (I_R).

$$I_0 = I_T + I_A + I_R \tag{2.2.1}$$

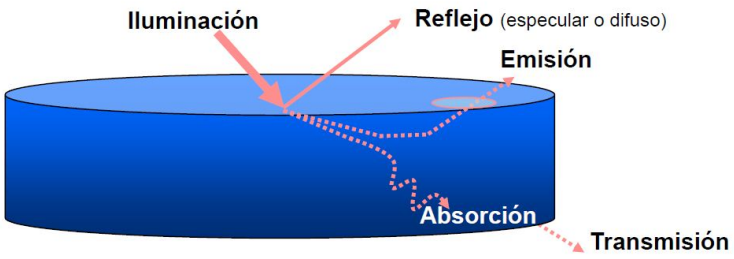
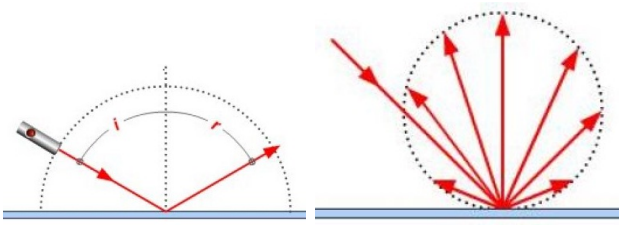


Figura 2.30: Interacción de la luz.

Los materiales que transmiten la luz con muy poca absorción y reflexión se denominan **transparentes**. Los llamados **translúcidos** dejan pasar parcialmente la luz y ésta se dispersa en su interior, ocasionando que los objetos no se distingan claramente si se ve a través de un translucido. Los materiales que no permiten la transmisión de la luz son llamados **opacos**. Estas propiedades descritas de manera breve son muy importantes para seleccionar un sistema de iluminación, ya que dependiendo el material a fotografiar dependerá el tipo de luz.

Reflexión

Cuando la luz se refleja en un objeto ocurren dos tipos de reflexión: la reflexión especular y la reflexión difusa [25]. La reflexión especular ocurre cuando cada rayo incidente se refleja en una única dirección y en un mismo ángulo. Por otro lado en la reflexión difusa el rayo se dispersa en ángulos diferentes como se muestra en la Figura 2.31. En general las superficies lisas o metálicas tienen reflexiones especulares y superficies rugosas provocan reflexiones difusas. En la realidad los objetos exhiben los dos tipos de reflexión.



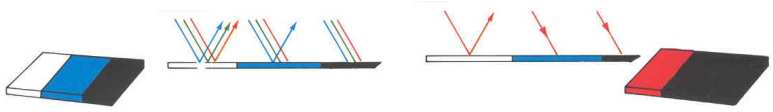
(a) Especular (b) Difuso

Figura 2.31: Tipos de reflexión

Un objeto que provoque reflexiones especulares es un objeto brillante pero lamentablemente en ocasiones esto es malo porque los sensores de la cámara se pueden saturar, ya que la intensidad de reflexión es comparable a la de iluminación. La iluminación de este tipo de objetos debe estar muy controlada debido a que pequeños cambios en los ángulos de iluminación provocan que la reflexión especular desaparezca o se generen grandes reflejos que saturen la cámara. Para este tipo de objetos se recomienda una iluminación difusa. Las reflexiones difusas no son muy brillantes pero a diferencia de las especulares, estas son muy estables. En este caso la reflexión es de 10 a 1000 veces menor que la fuente de luz [25].

Absorción

Cuando la luz llega a un objeto, este puede absorber toda o una parte de la luz. Esta absorción depende de diferentes factores físico-químicos [25] pero en este caso nos concentraremos en el color del objeto. Si una luz blanca (todo el espectro de luz visible) incide sobre un objeto la absorción y reflexión dependerán del color de este. Un objeto es negro cuando absorbe el 100 % de la luz y un objeto es blanco cuando refleja toda la luz [24]. Sin embargo no existen cuerpos totalmente blancos ya que todos los cuerpos absorben, al menos, 10 % de la luz. Un objeto es de un color determinado porque refleja la gama de longitudes de onda que constituyen su color y se absorben las gamas que no constituyan su color, esto es conocido como **Absorción selectiva**.

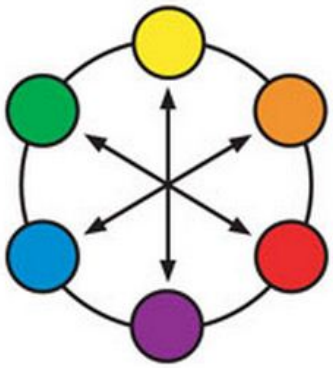


(a) Luz blanca (b) Luz roja

Figura 2.32: Absorción selectiva

La Figura 2.32 nos muestra que el color que percibimos por nuestros ojos y en una cámara es consecuencia de la absorción selectiva que depende de la fuente de iluminación y de la naturaleza del objeto. Ya que si la iluminación es blanca el objeto podrá reflejar todos sus colores. Por el contrario si la fuente de iluminación es roja sólo se obtendrá un reflejo de las partes que tengan un color correspondiente al rojo.

Por las razones anteriormente descritas la iluminación de un sistema de visión puede ser usada para aumentar o disminuir el contraste en una imagen. Para elevar o disminuir el contraste de un objeto se hace uso de una **rueda de color** como la mostrada en la Figura 2.33. Esta rueda muestra los espectros de luz que son contrarios. Si se necesita que un objeto se muestre obscuro se usa una iluminación del color contrario al objeto pero si se necesita que el objeto se muestre claro se usa una fuente de luz del mismo color [25]. El color de la fuente de luz puede ser indispensable como se muestra en el ejemplo de la Figura 2.33.



(a) Rueda de color

(b) Iluminación azul

(c) Iluminación roja

Figura 2.33: Creación de contraste a través de iluminación

Técnicas de iluminación

Hasta ahora sólo se han manejado conceptos generales sobre iluminación. En esta sección se muestran diferentes técnicas de iluminación, las cuales aterrizaran conceptos anteriormente descritos con el fin de obtener la mejor imagen posible, cada técnica presenta puntos a favor y en contra, y su elección dependerá del problema a resolver.

Iluminación frontal

En esta técnica la cámara se posiciona en la misma dirección de la fuente de iluminación [4] como se muestra en la Figura 2.34. Con esta técnica se reducen las sombras pero se generan grandes reflejos, es decir esta técnica es la indicada para materiales que no generen reflejos especulares, como son papel y telas.

(a) Posición de los elementos

(b) Otro tipo iluminación

(c) Con iluminación frontal

Figura 2.34: Iluminación frontal

Iluminación Lateral

Como su nombre lo especifica la fuente de iluminación se coloca en dirección lateral al objeto [4] como se muestra en la Figura 2.35. Esta técnica resalta los defectos generando una sombra en la dirección de la fuente de iluminación pero se genera una sombra por cada relieve.

(a) Posición de los elementos

(b) Sin defectos

(c) Con defectos

Figura 2.35: Iluminación lateral

Iluminación por campo obscuro (Darkfield)

Se llama campo obscuro cuando la fuente de iluminación se encuentra fuera del campo W de visión de la cámara [24], lo que provoca que únicamente se refleje hacia la cámara la luz difusa como muestra la Figura 2.36. Esta técnica destaca detalles en superficies con poco contraste pero no se recomienda con superficies que absorben grandes cantidades de luz. Esta técnica es la que se usa para metales porque se evitan las reflexiones especulares.

(a) Posición de los elementos (b) Otra iluminación (c) Con campo oscuro

Figura 2.36: Iluminación por campo oscuro o Darkfiel

Iluminación trasera (Backlight)

La iluminación está ubicada detrás del objeto [24] quedando ubicada como muestra la Figura 2.38. Esta técnica es la usada para resaltar la silueta de un objeto y también es la idónea para materiales translucidos. El inconveniente de esta técnica es que se pierde la información sobre superficie del objeto.

(a) Posición de los elementos (b) Sin backlight (c) Con Backlight

Figura 2.37: Iluminación trasera o backlight

Iluminación de día nublado (CDI- Cloudy Day Illumination)

Suministra iluminación difusa en la misma dirección que el eje de la cámara como se muestra en la Figura 2.38. Este tipo de iluminación se usa cuando se tiene una superficie especular y se

necesita una iluminación uniforme. Lamentablemente este tipo de iluminación es muy caro por la precisión de su geometría [24].

(a) Posición de los elementos (b) Sin CDI (c) Con CDI

Figura 2.38: Iluminación trasera o backlight

Como se pudo observar en esta sección el sistema de iluminación es un componente clave de un sistema de visión por computadora porque mejora el contraste entre el fondo y el objeto a analizar, además se pueden resaltar características de un objeto para facilitar el procesamiento de imagen.

2.3. OpenCV

OpenCV es una biblioteca Open Source para desarrollar aplicaciones de visión por computadora con un enfoque en aplicaciones de tiempo real y bajo uso de recursos. Fueron desarrolladas por Intel y liberadas bajo licencia BSD, lo que permite modificarlas, distribuirlas y usarlas en aplicaciones académicas y comerciales [23].

OpenCV tiene más de 500 algoritmos optimizados para realizar álgebra lineal, procesamiento de imágenes, reconocimiento de patrones, captura de video, visualización de imágenes y cuenta con una biblioteca especializada para realizar aprendizaje automático (Machine Learning) [26] como se muestra en la Figura 2.39.

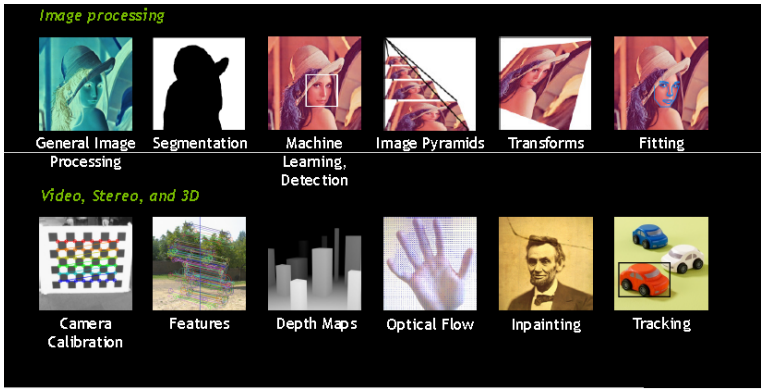


Figura 2.39: Principales algoritmos de OpenCV [1]

Actualmente OpenCV tiene una comunidad activa de 47 mil personas y es usado por empresas, como Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda y Toyota.

Los módulos principales de OpenCV son [26]:

Core
: Es el módulo básico de OpenCV en donde se definen las estructuras de datos básicas como: el contenedor de una imagen, vectores y propiedades de una imagen. También se encuentran funciones básicas usadas por todos los demás módulos.

Highhui
: Este módulo contiene las funciones de interfaz gráfica que permite crear y modificar ventanas. También es la interfaz que contiene los codecs para realizar la captura de imágenes y video.

imgproc
: En este módulo están algunos algoritmos básicos de procesamiento de imágenes, como son filtrado lineal y no lineal de imágenes, transformaciones geométricas, conversiones de espacio de color, entre otras.

video
: Contiene algoritmos básicos para el análisis de video, que incluye estimación de movimiento, eliminación de fondo y algoritmos de seguimiento de objetos.

objdetect
: Incluye algoritmos básicos de reconocimiento de objetos e instancias de las clases predefinidas (por ejemplo, ojos, caras, coches, entre otros).

El código fuente de todos estos módulos puede ser visto y modificado a conveniencia del usuario gracias a la licencia BSD.

Estas bibliotecas están disponibles para los sistemas operativos Linux, Android, MacOS X y Windows. Están desarrolladas para ser usadas en lenguajes de programación como: C, C++, Java y Python.

Para poder usar OpenCV se necesita recompilar completamente la distribución oficial, con el fin de que sea compatible con las instrucciones del procesador que se utilizará. El no hacer este proceso ocasiona que el procesador no reconozca las instrucciones de las bibliotecas de OpenCV. Para realizar el proceso de recompilado en la Raspberry Pi es necesario crear un archivo make con la herramienta CMAKE que contenga las especificaciones de la compilación. Una vez creado este archivo el proceso de recompilado dura de 13 a 15 horas, al finalizar las bibliotecas pueden ser usadas.

2.4. Qt Creator

Qt Creator es un entorno de desarrollo integrado (IDE) que cuenta con un editor, un compilador y depurador. Qt Creator esta basado en la biblioteca QT, que es ampliamente utilizada para desarrollar aplicaciones que puedan funcionar en diferentes plataformas realizando pocos o ningún cambio en el código base. Qt está disponible bajo las licencias GPL v3 y LGPL v2 que permiten su libre uso y modificación de acuerdo con estas licencias, además cuentan con una licencia comercial para poder vender el producto sin necesidad de otorgar el código fuente. Adobe Photoshop Album, Google Earth, KDE, Opera, Skype, VLC Media Player son algunos ejemplos de programas creados con la biblioteca Qt.

Utiliza el lenguaje de programación orientado a objetos C++, además permite realizar programación visual o dirigida por eventos. También se utiliza para la creación de interfaces gráficas, programación web, multihilo, bases de datos, entre otros. En la Figura 2.40 se muestran las principales características que presenta Qt Creator.

Figura 2.40: Características de Qt Creator [2]

Para la creación de los proyectos Qt Creator usa las herramientas qmake y CMake. Cuenta con un editor de código para asistir en la programación de código fuente en C++ y un diseñador de interfaz de usuario basado en código QML, en el cual se puede crear la interfaz del proyecto de manera visual a partir de formas que se van agregando a la ventana.

No sólo permite crear aplicaciones para computadoras de escritorio también se pueden crear aplicaciones móviles. Tiene la opción de trabajar junto a controladores de versiones, como Bazaar, Git, Mercurial, entre otros.

Qt Creator permite elegir entre diferentes compiladores de C++ como: MinGW, GCC, Clang y QCC. En este proyecto se eligió el compilador GCC, ya que la Raspberry Pi y la mayoría de los sistemas embebidos usan distribuciones de Linux y este compilador es el más usado en sistemas Linux y Unix. GCC (**GNU Compiler Collection**) es un compilador portable, lo que significa que soporta diversas arquitecturas de procesadores, entre ellas ARM (procesador usado por Raspberry Pi).

La programación en este ambiente de desarrollo se basa en la creación de un proyecto que relaciona los archivos de cabecera, código fuente y de interfaz gráfica. En los archivos de cabecera se declaran los métodos, atributos y bibliotecas usadas en cada clase. En los archivos de código fuente se programan los métodos de cada clase. Los archivos de interfaz gráfica contienen las especificaciones de las formas del programa.

2.5. Raspberry Pi

La Raspberry Pi es una computadora de bajo costo con dimensiones casi de una tarjeta de crédito (85.6 × 53.9 × 17 mm). El proyecto empezó en el año 2006 por la fundación Raspberry Pi, en 2012 se puso en venta y hasta el primer trimestre de 2015 se vendió un total de 5 millones de unidades de sus diferentes modelos.

El Modelo B se basa en un Broadcom BCM2835 system-on-chip(SoC). Esto quiere decir que la unidad de procesamiento central (ARM1176JZF-S a 700 MHz), la unidad de procesamiento gráfico (Video Core IV) y el hardware de comunicaciones están contenidos en un solo chip. El SoC BCM2835 se encuentra abajo de la memoria RAM de 512 MB en medio de la placa como se muestra en la Figura 2.41.

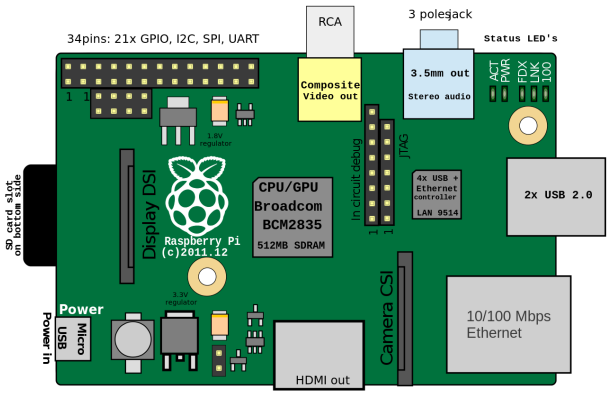


Figura 2.41: Raspberry Pi Modelo B

El BCM2835 utiliza una arquitectura ARM (versión ARMv6), la cual es una arquitectura RISC (Reduced Instruction Set Computer) que lo hace un sistema de bajo consumo energético. Esto evita que la Raspberry Pi necesite disipadores térmicos y permite a la Raspberry Pi funcionar con una fuente de alimentación de sólo 5V 1A.

Los sistemas operativos deben de ser modificados para ser compatibles con la arquitectura ARM de la Raspberry Pi. Las distribuciones de GNU/Linux que fueron optimizadas son principalmente Raspbian, Ubuntu y Fedora, también existe una versión compatible de Windows 10. El sistema operativo y los datos se almacenan en una memoria SD que se conecta en la parte inferior de la tarjeta.

La Raspberry Pi cuenta con tres diferentes estándares de salida de video: compuesto, HDMI y DSI (Display Serial Interface), el cual es utilizado por tablets y smartphones. También cuenta con puertos USB para conectar diferentes periféricos, un puerto CSI (Camera Serial in-

terface), un puerto de Ethernet (10/100 Mbps) y cuenta con 26 pines GPIO (General Purpose Input/Output) que se muestran en la Figura 2.42.

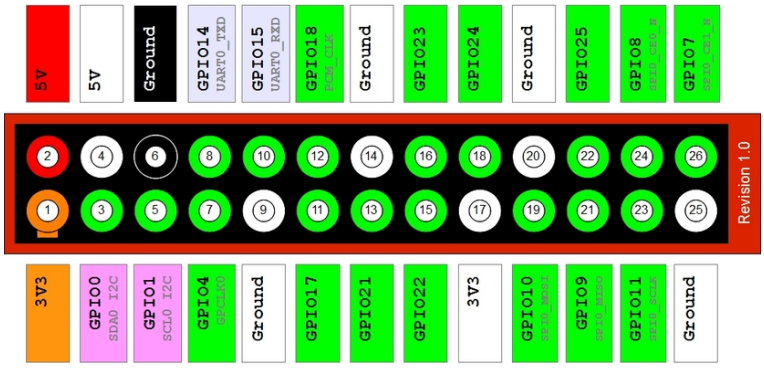


Figura 2.42: Distribución de pines GPIO

Capítulo 3

Desarrollo teórico y práctico

El objetivo de este capítulo es detallar cómo se generó el prototipo del sistema de visión que abarca los componentes mecánicos, el diseño electrónico y la etapa de software.

3.1. Implementación física de la banda transportadora

Las bandas transportadoras desempeñan un rol muy importante en la industria, esto se debe a varias razones entre las que destacan [27]:

- Permiten transportar grandes cantidades de producto
- Grandes distancias de transporte
- Fácil adaptación al terreno
- Su diseño permite que diferentes procesos se realicen sobre la banda

Por estas razones las bandas transportadoras son indispensables en industrias con grandes producciones. El proceso de manufactura de circuitos electrónicos hace uso de bandas transportadoras. En esta parte se detalla la implementación de una banda transportadora que sirve para simular una línea de producción de tarjetas electrónicas.

Definir las dimensiones de la banda transportadora es una parte esencial, ya que el sistema de visión se debe adaptar a las condiciones de la línea de producción. Para dimensionar la banda transportadora primero se debe definir el tipo de tarjetas PCB a analizar.

Para construir la banda transportadora se decidió tomar como base tarjetas de desarrollo como la que se muestra en la Figura 3.1. Debido a que este tipo de tarjetas son muy comunes, lo que permite encontrarlas fácilmente para hacer diferentes pruebas.

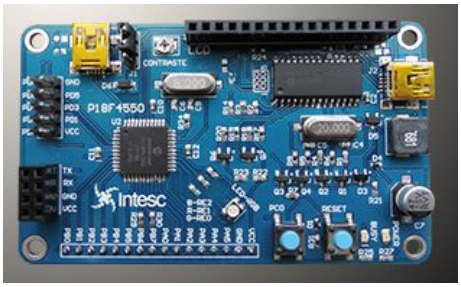


Figura 3.1: Tarjetas PCB usadas como base de diseño

La mayoría de las tarjetas de desarrollo no miden más de 10 cm por lo que se propuso construir una banda transportadora de 15 cm de ancho. Además las tarjetas de desarrollo no pesan más de 200 gramos, por lo que es posible construir una banda transportadora básica [27]. Por esta razón se decidió hacer una banda que tuviera sólo los componentes que se muestran en la Figura 3.2.

Figura 3.2: Componentes principales de la banda transportadora

Para realizar el prototipo de la banda transportadora se usaron perfiles rectangulares de aluminio como elemento de soporte. Se usaron 2 tornillos de 20 cm de largo con 2 cm de diámetro para hacer las flechas. En la flecha de retorno se colocaron rodamientos de bolas para que la banda se pudiera deslizar. En la Figura 3.3 se muestra el prototipo de la banda transportadora.

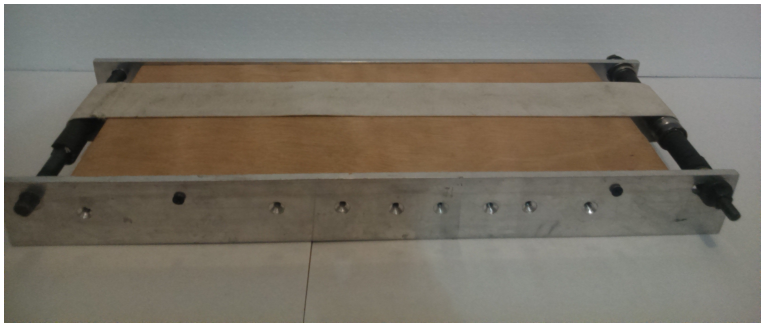
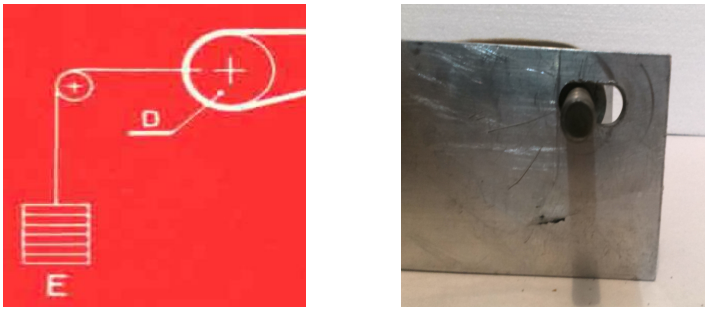


Figura 3.3: Prototipo de banda transportadora

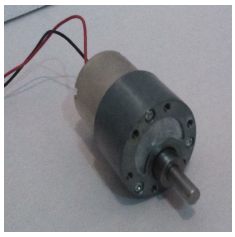
Para evitar que la banda esté floja y ocasione un deslizamiento incorrecto se implementó un mecanismo de tensión sobre el centro de la flecha de retorno [27], este mecanismo se muestra en la Figura 3.4. Si se requiere tensar la banda se recorre la flecha y se aprieta el tornillo.



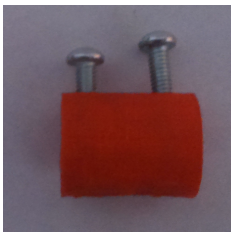
(a) Mecanismo de tensión (b) Tensión de banda

Figura 3.4: Mecanismo de tensión de banda

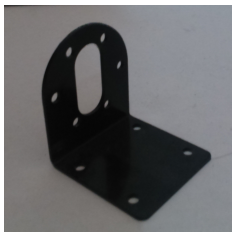
Para transmitirle movimiento a la banda transportadora se acopló un motor eléctrico a la flecha motriz. Se usó un motor de corriente continua con caja de engranes (los detalles eléctricos y de control se tratan en la sección 3.4 (**Automatización del Sistema**). Para acoplar el motor a la flecha motriz se diseñó un acoplamiento rígido con dos tornillos de fijación. Por último se adquirió un soporte para evitar el movimiento del motor. Los elementos anteriormente descritos se muestran en la Figura 3.5.



(a) Motor DC con caja de engranes



(b) Acoplamiento rígido



(c) Soporte de motor

Figura 3.5: Transmisión de banda transportadora

El prototipo final de la banda transportadora se muestra en la Figura 3.6, donde se puede apreciar que se colocaron dos perfiles a lo ancho de la banda, con el propósito de colocar de manera horizontal cada tarjeta PCB.

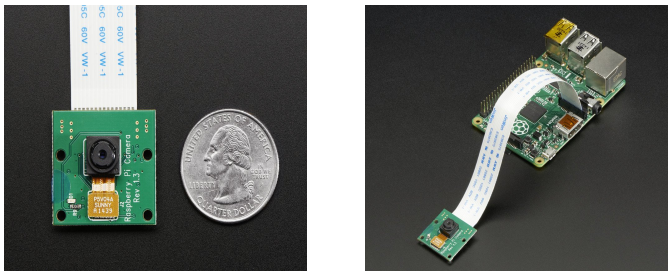
Figura 3.6: Prototipo final de banda transportadora

3.2. Cámara y sistema de iluminación

En esta sección se analizan los pasos y consideraciones para obtener imágenes que faciliten el reconocimiento de dispositivos. Como se describió en el marco teórico la toma de una buena imagen no sólo depende de la cámara, también depende del sistema de iluminación, ya que este considera la posición de la cámara, el control de la luz ambiental, el tipo y la posición de la fuente de luz.

3.2.1. La cámara Raspberry Pi

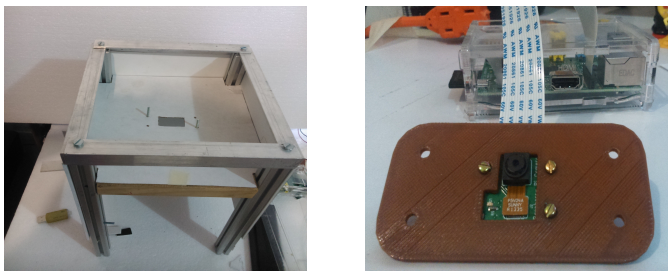
La cámara Raspberry Pi es una cámara pequeña (25mm × 20mm × 9mm), cuyo sensor tiene una resolución nativa de 5 megapíxeles, además puede obtener imágenes de 2592 × 1944 píxeles y es capaz de tomar video a 1080p a 30 cuadros por segundo. Esta cámara se conecta al procesador BCM2835 mediante la interfaz CSI que se encuentra en la parte superior de la Raspberry Pi como se muestra en la Figura 3.7.



(a) Cámara (b) Conexión de cámara

Figura 3.7: Cámara Raspberry Pi [3]

Para obtener imágenes nítidas se debe fijar la distancia de trabajo de la cámara. Por esta razón se construyó una estructura que mantenga la posición de la cámara, esta estructura se muestra en la Figura 3.8. Además mediante impresión 3D se realizó un soporte para la cámara, ya que ésta tiene componentes en su superficie, lo que ocasiona que las imágenes salgan inclinadas.

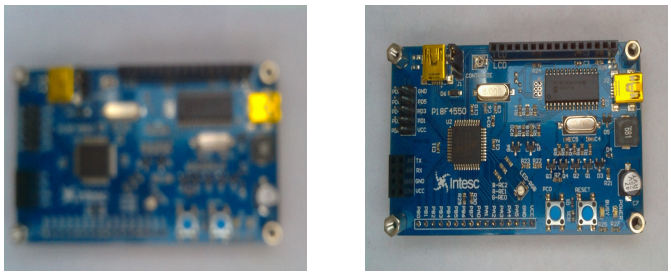


(a) Estructura (b) Soporte de cámara

Figura 3.8: Estructura de toma de imágenes

Una vez fijada la distancia de trabajo se tiene que enfocar la cámara, es decir se deben concentrar los haces de luz reflejados por el objeto hacia el plano de formación de imagen

como se describió en el marco teórico. Para realizar esto se debe modificar la distancia focal de la cámara, girando la lente para acercarla o alejarla. En la Figura 3.9 se muestran imágenes obtenidas variando la distancia focal de la cámara.



(a) Sin enfoque (b) Imagen nítida

Figura 3.9: Enfoque manual de la cámara

Como se puede observar la toma de imagen ya se encuentra dentro de la zona nítida pero si se observa detenidamente, ambas imágenes presentan grandes reflejos en la parte superior. Esto se debe a que el sistema de iluminación no es el correcto.

3.2.2. Iluminación

Tener una imagen nítida es indispensable para las aplicaciones de visión por computadora, sin embargo no es suficiente para aplicaciones donde los detalles son muy importantes. En la Figura 3.9 se obtuvieron imágenes nítidas pero con grandes reflejos. Estos reflejos ocasionan que el procesamiento de la imagen sea más difícil o imposible, ya que los reflejos ocultan las propiedades de un objeto. Por esta razón es necesario analizar las diferentes técnicas de iluminación para obtener la mejor imagen posible. Además se debe controlar la entrada de luz externa para tener una iluminación controlada y uniforme, para evitar la entrada de luz se cerró la estructura que sostiene la cámara.

Las propiedades ópticas de los materiales nos ayudan a determinar el tipo de fuente de luz que se debe usar. Las tarjetas PCB están compuestas por diferentes dispositivos y cada uno tiene sus propiedades ópticas. En general se puede decir que una tarjeta PCB es en su mayoría una superficie metálica que tiene las siguientes propiedades ópticas [25]:

- Es un material opaco
- Absorben todas las frecuencias de luz visible

- Remiten la mayor parte de luz absorbida en la misma longitud de onda (luz reflejada)

Tomando como base las propiedades ópticas de los metales se concluye que no se debe usar fuentes de luz especulares, ya que se formarían grandes reflejos como el que se muestra en la Figura 3.10.

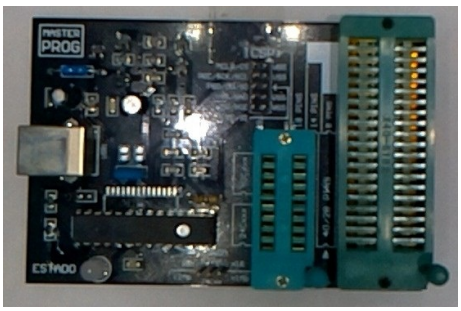


Figura 3.10: Reflejo generado por una luz especular

Las fuentes de luz difusas son la mejor alternativa para iluminar un metal. Para obtener este tipo de luz se usan difusores como los que se muestran en la Figura 3.11. Lamentablemente este tipo de componentes tienen costos muy elevados.

(a) Difusores

(b) Anillo de luz con difusor

Figura 3.11: Obtención de luz difusa

La alternativa que se encontró para iluminación difusa fueron los LED de iluminación trasera que se muestran en la Figura 3.12.

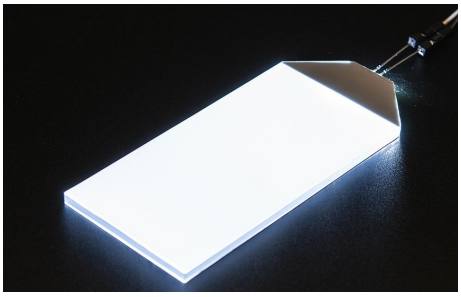


Figura 3.12: LED de iluminación trasera [3]

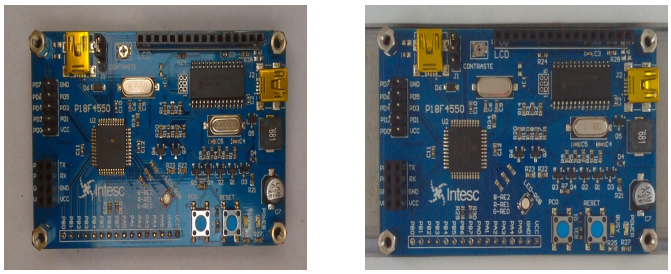
Con la fuente de luz difusa ahora sólo resta determinar que técnica de iluminación es la que genera mejores fotografías.

Evaluación de las técnicas de iluminación

En la sección Técnicas de iluminación del marco teórico se describió las ventajas de cada sistema de iluminación. Haciendo un análisis de las necesidades del sistema de visión las técnicas recomendadas son:

- Iluminación frontal
- Iluminación del día nublado
- Iluminación por campo obscuro

La iluminación frontal presenta la ventaja de no generar sombras en el objeto. Al evaluar esta técnica se obtuvo la imagen de la Figura 3.13(a). Como se puede apreciar se obtiene una imagen sin sombras y con colores claros pero a pesar de usar una luz difusa se siguen notando grandes reflejos que hacen que el color azul se muestre blanco. A pesar de que la técnica de día nublado es una buena alternativa, ésta presenta geometrías que hacen que implementarla sea muy costoso, por lo que se decidió no implementarla. Implementando la iluminación por campo obscuro se obtiene la imagen de la Figura 3.13(b). Se puede observar que ya no se generan reflejos sobre la tarjeta, sin embargo la imagen obtenida es más obscura.



(a) Iluminación Frontal

(b) Iluminación por campo oscuro

Figura 3.13: Evaluación de Técnicas de Iluminación

La técnica de iluminación por campo obscuro es la que ayuda a generar mejores imágenes por esta razón el sistema implementará esta técnica en el sistema de visión.

3.3. Desarrollo Software de Reconocimiento

En esta sección se describen los pasos que se siguieron para obtener los algoritmos de identificación. Primero se describe la etapa de preprocesamiento de la imagen, que consiste en tratar la imagen para resaltar los elementos buscados. Después se presenta el desarrollo del algoritmo de identificación de dispositivos. Por último se muestran los resultados en la interfaz gráfica. En la Figura 3.14 se presenta un esquema general del proceso de identificación de dispositivos.

Figura 3.14: Pasos para la identificación de dispositivos.

3.3.1. Preprocesamiento de imagen

El preprocesamiento de la imagen permite eliminar los elementos que dificultan la identificación, así como resaltar los elementos importantes. Debido a que se está trabajando con una Raspberry Pi, la cual tiene recursos limitados, se deben optimizar todos los algoritmos para reducir los tiempos de procesamiento.

Consideración de recursos

Los algoritmos de procesamiento digital de imágenes hacen operaciones sobre los píxeles de la imagen, ya sea de manera individual o en conjunto. Esto quiere decir que el tamaño de la imagen determina el número de operaciones que se realizan en un solo algoritmo. Para ejemplificar lo anterior se toman las operaciones de píxel que son las que ejecutan sólo una operación por píxel. Esto significa que en una imagen **A** con resolución de 10×10 necesita un total de 100 operaciones para realizar las operaciones de píxel. En una imagen **B** de tamaño 50×50 el número de operaciones necesarias es de 2500. El tamaño de la imagen determina la cantidad de operaciones que se realizan, ésta cantidad se incrementa si el algoritmo es más

complejo. Por ejemplo al aplicar un filtro espacial con un kernel de tamaño 4×4. El número de operaciones que se realizan al aplicar el filtro a la imagen **A** es de 800 y de 20000 para la imagen **B**.

Las imágenes capturadas por la cámara tienen una resolución de 2592×1944 por lo que para ejecutar el algoritmo más básico se necesitan 5 038 848 operaciones. Por esta razón es necesario diseñar un algoritmo que no ejecute operaciones innecesarias. Para lograr esto se plantea extraer unicamente la imagen del PCB de toda la imagen, con el objetivo de no aplicar algoritmos de procesamiento donde no sea necesario.

El primer paso para realizar la separación consiste en elevar el contraste entre la tarjeta y el fondo. Para lograr esto los componentes de la banda transportadora se construyeron de color blanco. La imagen obtenida se muestra en la Figura 3.15.

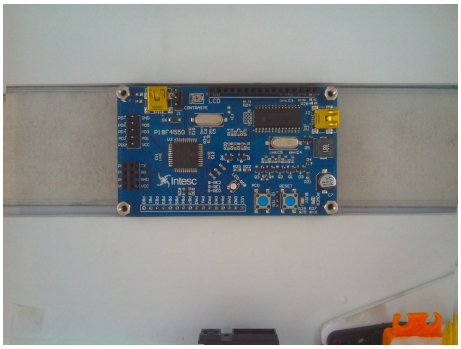
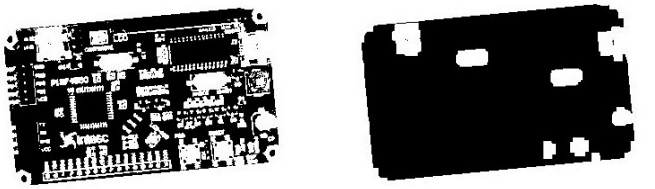


Figura 3.15: Imagen obtenida

La banda transportadora no sólo aumenta el contraste con su color blanco, también posiciona la tarjeta PCB de manera horizontal, lo que reduce las etapas de preprocesamiento. Una vez que se tiene la tarjeta en posición horizontal un servomotor la posiciona en medio de la imagen. El siguiente paso es aplicar un algoritmo que calcula la posición del vértice superior izquierdo de la tarjeta PCB. El diagrama de flujo de la Figura 3.16 presenta los pasos que se ejecutan para extraer la imagen del PCB de la imagen completa.

Figura 3.16: Diagrama de flujo de recorte de imagen

Después de hacer el umbralado a la imagen a escala de grises los bordes de la tarjeta no quedan bien definidos como se muestra en la Figura 3.17, lo que ocasiona que el algoritmo localice incorrectamente los bordes. Para solucionar este problema se aplicó la operación morfológica de apertura con un elemento rectangular de tamaño 4×4 [22] con el objetivo de llenar espacios generados en los bordes para tener bordes constantes como se muestra en la Figura 3.17.



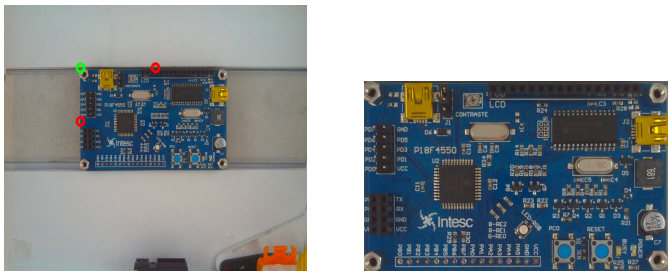
(a) Imagen umbralada (b) Operación de apertura

Figura 3.17: Definición de bordes

Una vez que los bordes se encuentran bien definidos, se determina el ángulo de inclinación del PCB mediante operaciones de barrido vertical. Determinado el ángulo de inclinación se realiza una **Transformación Geométrica de Rotación** [22] para obtener una imagen totalmente horizontal como se muestra en la Figura 3.18.

Aplicando las funciones de localización de bordes (superior e izquierdo) obtenemos el vértice

superior izquierdo. El alto (**740**) y ancho (**1250**) en píxeles de la tarjeta PCB es siempre constante. Con todo esto se aplica la función **Rect** de OpenCV que permite extraer una zona rectangular de una imagen. Los resultados del algoritmo localización y la aplicación de la función de extracción de zonas de OpenCV se muestra en la Figura 3.18. Los círculos rojos representan el resultado de los barridos y en color verde el vértice superior izquierdo.



(a) Algoritmo de localización de vértice

(b) Imagen extraída

Figura 3.18: Extracción de zona de interés

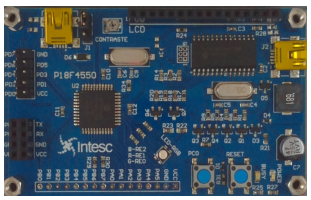
Esta primera etapa de preprocesamiento reduce la imagen de 5038848 a 925000 píxeles, es decir se reduce aproximadamente un 80 % el número de operaciones. Cabe mencionar que las características de la cámara no permiten acercarla más a la tarjeta para obtener una imagen más cercana. Es importante resaltar que si la aplicación lo necesita se tendría que adquirir una cámara con lentes de acercamiento.

Reducción de ruido

Esta etapa de preprocesamiento tiene el objetivo reducir el ruido de la imagen. Los filtros espaciales de suavizado reducen el ruido pero muchos de ellos reducen los bordes de la imagen. Para no diluir los bordes al suavizar la imagen se usa un filtro de Gauss [22] con núcleo de tamaño 3×3 como se muestra en la Figura 3.19.

$$\frac{1}{16} \times \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

(a) Núcleo de filtro de Gauss



(b) Imagen filtrada

Figura 3.19: Suavizado por filtrado espacial

A pesar que a simple vista no se nota la diferencia entre la imagen filtrada y la imagen original siempre es importante aplicar un filtro de reducción de ruido para mantener una calidad promedio entre imágenes.

Sustracción del fondo de la imagen

Como se mencionó anteriormente la etapa de preprocesamiento tiene la función de modificar la imagen para resaltar los elementos importantes en una escena. La sustracción del fondo de una imagen es un método muy útil para resaltar los cambios de una escena, como objetos en movimiento. La aplicación de este método depende de las condiciones de cada imagen pero principalmente existen dos técnicas [23]:

1. Haciendo una resta entre imágenes y un umbralado cuando el fondo es constante como muestra la Figura 3.20

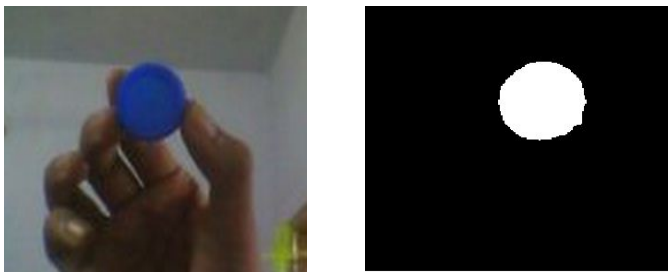
(a) Modelo del Fondo

(b) Imagen analizada

(c) Foregroundmask

Figura 3.20: Extracción de fondo por resta

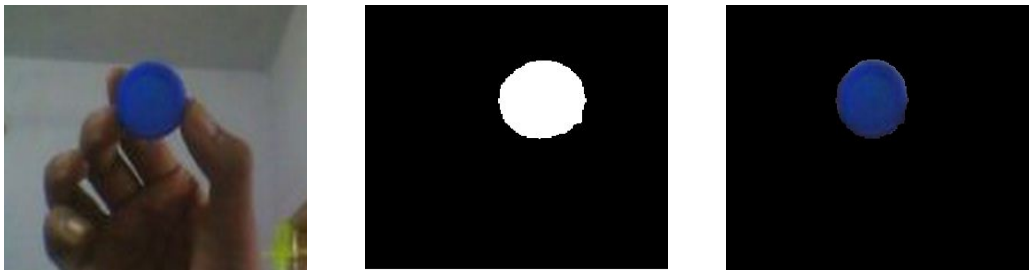
2. Segmentación basada en color (Figura 3.21).



(a) Imagen analizada (b) Foregroundmask

Figura 3.21: Extracción de fondo por filtrado de color

Cualquiera de estas técnicas generara una máscara (**Foregroundmask**) que lleva a primer plano los elementos que no formen parte del fondo. El uso de ésta depende totalmente del procesamiento posterior, en muchas ocasiones se aplican algoritmos sobre la imagen binaria y en otras la máscara se aplica sobre la imagen original para mantener los detalles de los objetos. La operación AND entre imágenes [22] se utiliza para aplicar una máscara a una imagen como se muestra en la Figura 3.22.



(a) Imagen analizada (b) Foregroundmask (c) Operación AND

Figura 3.22: Aplicación de una máscara

Segmentación basada en color

Como se vio anteriormente la segmentación basada en color nos permite generar una máscara a partir de un "filtrado"de color. Para generar esta máscara es necesario poder clasificar los píxeles por colores para poder eliminarlos o mantenerlos. Esta clasificación no es simple en el modelo de color RGB, ya que en este modelo genera los colores a partir de una mezcla aditiva de los canales rojo, verde y azul [21]. Esto significa que la segmentación por color en el modelo

RGB es una tarea con muchos inconvenientes y sobre todo poco práctica, ya que las tonalidades de un solo color dependen de tres variables al mismo tiempo.

Los modelos de color HSV y HSL son los modelos idóneos para realizar la segmentación basada en color, ya que como se explicó en el marco teórico representan la información del color a través de 3 componentes: la tonalidad, la saturación y valor (HSV) o iluminación (HSL). Estos modelos presentan en su componente tonalidad un arcoíris perfectamente definido de todos los colores en un rango de 0 a 360 grados como se muestra en la Figura 3.23.

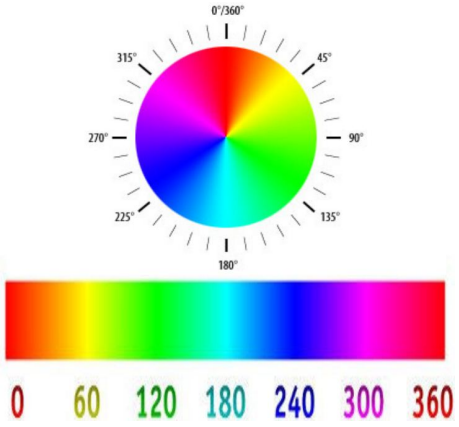


Figura 3.23: Componente tonalidad

Este modelo de color permite hacer la segmentación de un color a partir de la selección de un rango de tonalidad, después haciendo combinaciones de los componentes saturación y valor se obtienen todos los tipos del color seleccionado. Por ejemplo el rango [60,180] representa el color verde y modificando los valores de la saturación y la iluminación se obtienen verdes obscuros o claros. Por estas razones es mejor realizar la conversión modelo RGB al modelo HSV.

Algoritmo de conversión de modelo RGB a HSV

El algoritmo para realizar la conversión del modelo RGB al HSV se basa en 4 pasos [23]:

1. Los valores R,G,B son divididos entre 255 para obtener un rango [0,1], se obtiene el máximo y el mínimo de las componentes y se obtiene la diferencia entre ellos.

$$\begin{aligned} R' &= R/255 \\ G' &= G/255 \\ B' &= B/255 \\ Cmax &= max(R', G', B') \\ Cmin &= min(R', G', B') \\ \Delta &= Cmax - Cmin \end{aligned} \tag{3.3.1}$$

2. El canal de tonalidad se obtiene con la siguiente relación. Donde mod es el resto después de dividir entre 6.

$$H = \begin{cases} 60° \times (\frac{G'-B'}{\Delta} mod 6) & Cmax = R' \\ 60° \times (\frac{B'-R'}{\Delta} + 2) & Cmax = G' \\ 60° \times (\frac{R'-G'}{\Delta} + 6) & Cmax = B' \end{cases} \tag{3.3.2}$$

3. El canal de saturación se obtiene con la siguiente relación.

$$S = \begin{cases} 0 & Cmax = 0 \\ \frac{\Delta}{Cmax} & Cmax \neq 0 \end{cases} \tag{3.3.3}$$

4. El componente de valor o intensidad corresponde al valor máximo.

$$V = Cmax \tag{3.3.4}$$

OpenCV permite la conversión de modelo RGB al HSV, con la diferencia de que la tonalidad se representa en el rango [0,180], la saturación y la intensidad con el rango [0,255].

Algoritmo de sustracción de color de fondo de la tarjeta PCB

Para generar una máscara que elimine el color de fondo de la imagen se ejecuta el algoritmo que se muestra en la Figura 3.24.

Figura 3.24: Diagrama de flujo de sustracción de fondo

Para realizar la selección de los rangos de valores de los componentes HSV se desarrolló un programa en Qt Creator, el cual realiza en tiempo real la segmentación basada en color. El programa tiene 3 barras de seguimiento para los límites inferiores $(H_{min}, S_{min}, V_{min})$ y 3 para los límites superiores $(H_{max}, S_{max}, V_{max})$ como se muestra en la Figura 3.25.

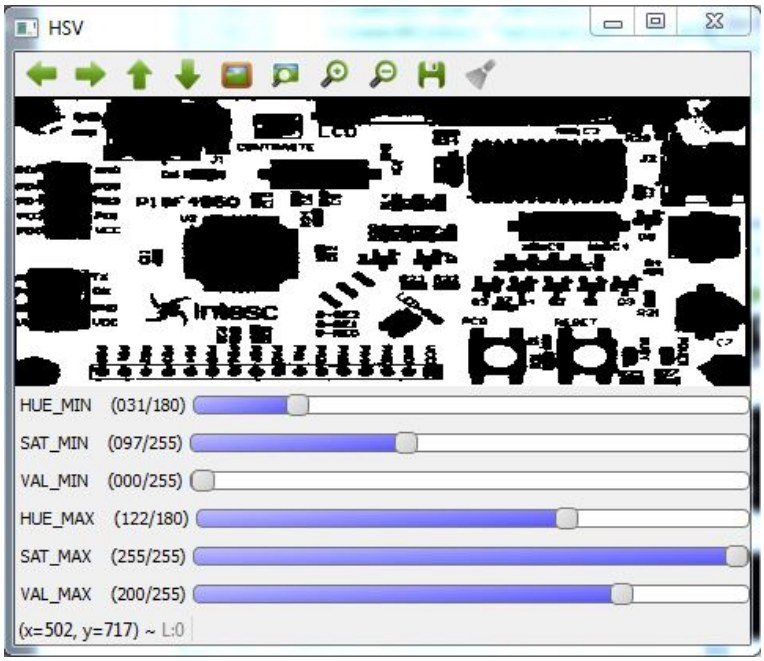
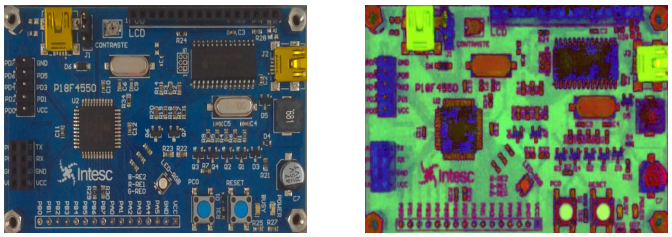


Figura 3.25: Programa para realizar segmentación basada en color

Con este programa se determinó el rango de valores que permiten generar una máscara para ser aplicada sobre la imagen original y así eliminar el color azul de la imagen RGB.

$$\begin{array}{ll} H_{min} = 31 & H_{max} = 122 \\ S_{min} = 97 & S_{max} = 255 \\ V_{min} = 0 & V_{max} = 200 \end{array} \tag{3.3.5}$$

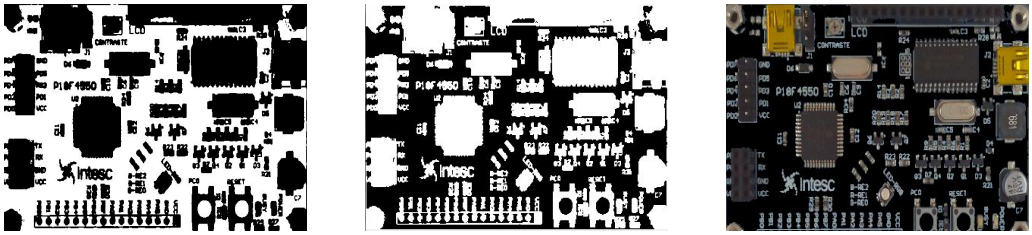
El primer paso para extraer el fondo de la imagen es cambiar de modelo de color RGB al HSV como se muestra en la Figura 3.26.



(a) Imagen RGB (b) Imagen HSV

Figura 3.26: Cambio de modelo de color

Posteriormente a la imagen HSV se hace un umbralado superior e inferior en cada uno de sus canales (H,S,V) con los valores de la ecuación (3.3.5). Con este umbralado obtenemos una máscara que elimina el color azul de fondo de la placa como se observa en la Figura 3.27 (a). Al aplicar la operación AND con esta máscara se generaría una imagen que esconda los dispositivos y no el fondo por lo que tiene que usar la inversión de la misma (b). Posteriormente mediante la operación AND le aplicamos la máscara invertida a la imagen RGB como se muestra a continuación.



(a) Máscara por umbralado (b) Máscara invertida (c) Aplicación de máscara

Figura 3.27: Aplicación de máscara

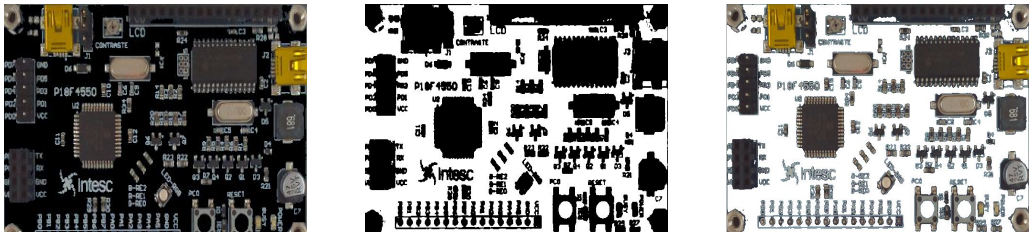
Como podemos observar se logra eliminar el fondo de la imagen pero es fácil observar que existe muy **poco contraste entre el fondo y los componentes**, ya que ambos son de color negro. Para solucionar ésto se plantea que el fondo se muestre blanco. Lamentablemente el proceso de aplicar una mascara blanca no es tan simple como el de colocar una máscara negra con una sola operación lógica AND. Para poder aplicar una máscara blanca es necesario realizar los pasos que se muestran a continuación:

1. Aplicar la máscara invertida a la imagen RGB
2. La máscara original se convierte a imagen RGB
3. Se suman las dos imágenes anteriores

En el primer paso se obtiene la imagen descrita en la Figura 3.27 (c) con el objetivo de que las zonas a eliminar tengan un valor de 0 en todos los canales (negro). En el segundo paso se genera una imagen RGB que tenga en sus tres canales:

- 255 en todas las zonas a eliminar
- 0 en las zonas con dispositivos

Con el objetivo de que en la suma algebraica de imágenes las zonas con dispositivos mantengan exactamente su valor, es decir se les sumará un valor de 0 a cada uno de sus canales. Por el contrario a las zonas negras se les sumará 255. El proceso de generación de una máscara blanca se muestra en la Figura 3.28.



(a) Máscara por AND (b) Máscara en RGB (c) Suma de (a) y (b)

Figura 3.28: Generación de máscara blanca

3.3.2. Identificación de dispositivos

En esta sección se presentan los pasos para determinar si los dispositivos se encuentran o no en la tarjeta PCB. La etapa de preprocesamiento anteriormente descrita ayuda a que los algoritmos de identificación tengan mejores resultados.

Template Matching

El proceso para identificar la presencia de dispositivos se basa en la técnica de comparación de imágenes **Template Matching**. Como se explicó en el marco teórico esta técnica se basa en la **Convolución entre imágenes**, sólo que la mascarilla (**Template**) se encuentra rotada 180° sobre si misma por lo que esta variación es conocida como **correlación**. Para realizar la identificación de dispositivos se usará el **Coeficiente de Correlación Normalizado (NCC)**, también conocido como coeficiente de correlación lineal de Pearson [6]. El NCC mejora los resultados del coeficiente de correlación, ya que se consideran las diferencias de iluminación locales de cada región de coincidencia con el patrón, como describen las ecuaciones (2.1.20) y (2.1.21).

El método NCC genera una imagen con valores que van del rango -1 (Ninguna coincidencia) a +1 (Coincidencia perfecta). En OpenCV el rango de valores entregado por este método esta normalizado al rango [0,1].

El resultado generado por el Método NCC muestra zonas blancas donde existe un porcentaje de coincidencia entre el patrón y la región de coincidencia y zonas obscuras donde la coincidencia es casi nula. Por esta razón la siguiente etapa de procesamiento consiste en determinar que porcentaje de coincidencia determina el parentesco entre el patrón y la zona.

Algoritmo de identificación

Después de toda la etapa de preprocesamiento se obtiene una imagen con alto contraste entre los dispositivos y el fondo. Los algoritmos de identificación de dispositivos dependen totalmente de una buena etapa de preprocesamiento, ya que sin ésta su precisión es baja y en ocasiones es nula.

El algoritmo de reconocimiento de dispositivos se desarrolla en una función llamada **match**. Esta función es llamada por una parte del programa principal. La función match realiza la secuencia que se muestra en la Figura 3.29.

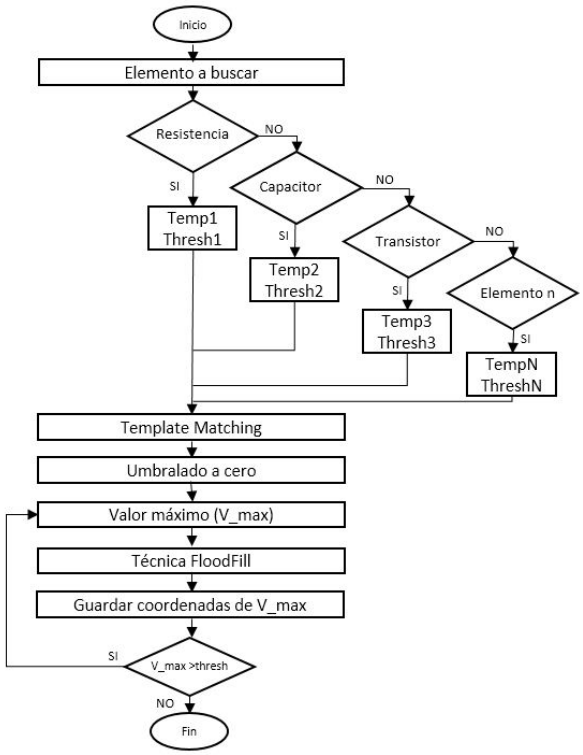
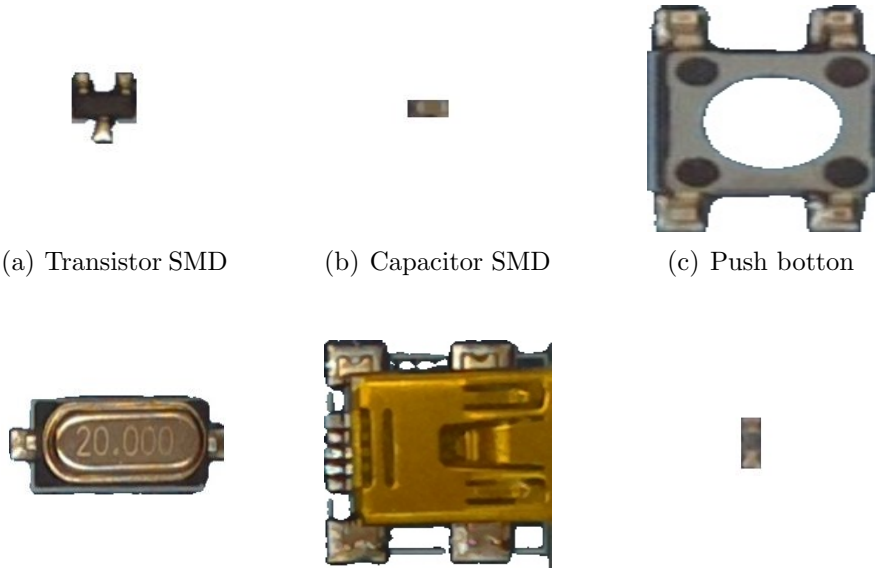


Figura 3.29: Algoritmo para localizar dispositivos

En el primer paso la función match recibe del programa principal un número que le indica el patrón correspondiente a un dispositivo, además se asignan valores específicos para tener un mejor desempeño del algoritmo. En la Figura 3.30 se muestran los diferentes patrones que puede identificar la función match.



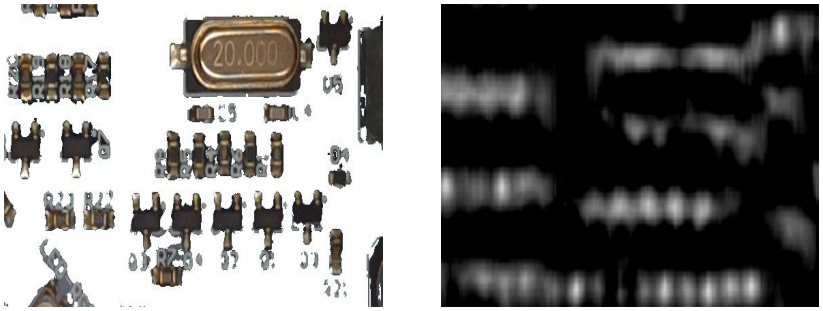
(a) Transistor SMD (b) Capacitor SMD (c) Push botton

(d) Oscilador SMD (e) Conector USB (f) Resistencia SMD

Figura 3.30: Patrones en función match

Cabe mencionar que a cada patrón se le aplicó todo el preprocesamiento para aumentar la efectividad del algoritmo. El programa esta diseñado para **aumentar el número de patrones** que se pueden identificar. Para realizar esto únicamente se tiene que agregar un nuevo caso a la función switch.

El siguiente paso del algoritmo consiste en aplicar el método de Coeficiente de Correlación Normalizado con el patrón elegido por el programa principal. En la Figura 3.31 se muestran los resultados del método NCC aplicado a una sección de la imagen preprocesada.



(a) Imagen preprocesada (b) Resultado de NCC

Figura 3.31: Método de NCC con transistor

Como puede observarse el resultado de NCC presenta en color blanco los elementos que se parecen más a la imagen patrón. Como se mencionó los colores más blancos implican una mayor coincidencia. Para hacer la discriminación entre elementos que son verdaderamente candidatos se hace un umbralado.

Usar una segmentación por umbral simple generaría perdida de información importante, ya que la imagen se separaría sólo en dos valores. Por esta razón se hace uso de una variación conocida como **Segmentación por umbral a Cero** [23]. La diferencia con el umbralado simple es que el umbral no separa la imagen en dos conjuntos como se muestra la ecuación 3.3.6.

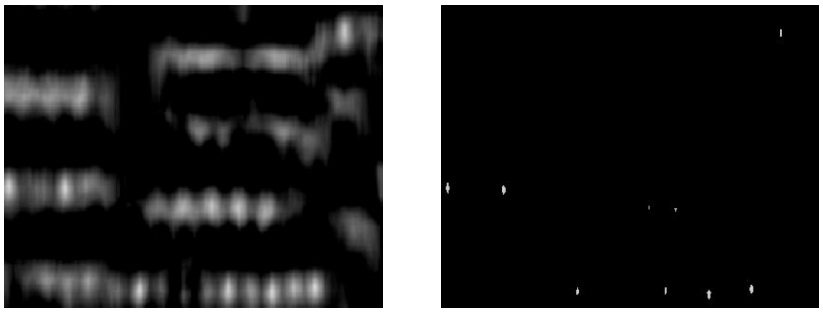
$$f_{th}(p) = \begin{cases} I(x,y) & \text{si } p > Umbral \\ 0 & \text{si } p \leq Umbral \end{cases} \tag{3.3.6}$$

Como indica la ecuación 3.3.6 el umbralado a cero mantiene todos los valores de la imagen que se encuentran por encima del valor del umbral y convierte en cero todos los valores que estén por debajo como se muestra en la Figura 3.32.

(a) Representación del umbralado a cero (b) Umbralado Simple (c) Umbralado a cero

Figura 3.32: Umbralado a cero

Al aplicar el umbralado a cero a la imagen resultante del NCC se obtienen las secciones más parecidas a la imagen patrón. En la Figura 3.33 se muestra la aplicación de umbralado a cero al resultado del NCC.



(a) Resultado de NCC (b) Umbralado a cero

Figura 3.33: Método de umbralado a cero

La siguiente etapa de procesamiento realiza la clasificación de todos los candidatos. Primero busca al píxel que tenga el máximo valor (más blanco), compara su valor con un valor mínimo para garantizar parentesco y por último guarda su posición en un vector. Una vez que guarda la posición se necesita eliminar ese punto para no volver a localizarlo y poder buscar al siguiente candidato, ya que de no hacer este paso siempre localizaría el mismo candidato. Para hacer la eliminación de candidatos encontrados se usa el algoritmo de relleno por difusión [23] (**Floodfill**). Este algoritmo en procesamiento de imágenes consiste en los siguientes pasos:

1. Dar un punto de inicio o punto semilla.
2. Dar un valor de límite superior e inferior.
3. Dar un nuevo valor para rellenar.
4. Comienza la difusión a partir del punto de inicio.
5. Cambia el valor de cada píxel que se encuentre dentro de los límites superiores e inferiores.

En la Figura 3.34 se muestra cómo se desarrolla el algoritmo FloodFill con cinco puntos semilla diferentes, cinco nuevos valores, un valor inferior (10) y un valor superior (240).

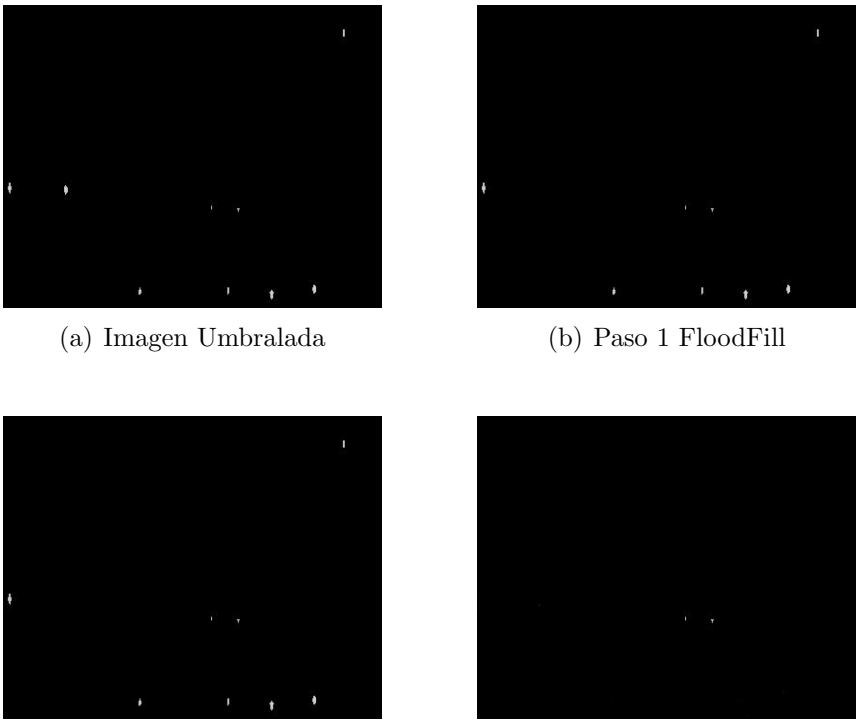
(a) Imagen original (b) Después de FloodFill

Figura 3.34: Ejemplo de uso de FloodFill

En el algoritmo de identificación se aplica el algoritmo de FloodFill con los siguientes parámetros:

- Punto semilla = Posición (x,y) de píxel con máximo valor
- Nuevo valor = 0 (Negro)
- Límite inferior = Diferencia mínima permitida - 0.2
- Límite superior = diferencia máxima aceptada + 0.2

En la Figura 3.35 se muestra cómo se aplica el algoritmo Floodfill a la imagen umbralada a cero para encontrar a todos los dispositivos. Cabe mencionar que no tiene que dejar toda la imagen en negro, ya que cada valor máximo debe cumplir un parentesco mínimo.



(a) Imagen Umbralada (b) Paso 1 FloodFill

(c) Paso 2 FloodFill (d) Último paso

Figura 3.35: Funcionamiento de algoritmo Floodfill

Una vez que todos los puntos que superan el parentesco mínimo son localizados y guardados. Se retorna al programa principal un vector con todas las posiciones y en el **elemento [0] del vector** se coloca el número de dispositivos encontrados. Este valor determina el número de operaciones que se deben realizar en la siguiente etapa de procesamiento. Es importante mencionar que se debe poner una variable de seguridad que garantice que no se sobre cargue la memoria y evite que el programa deje de funcionar. Este proceso se realiza con todos y cada uno de los dispositivos que indique el programa principal.

3.3.3. Comparación de posiciones

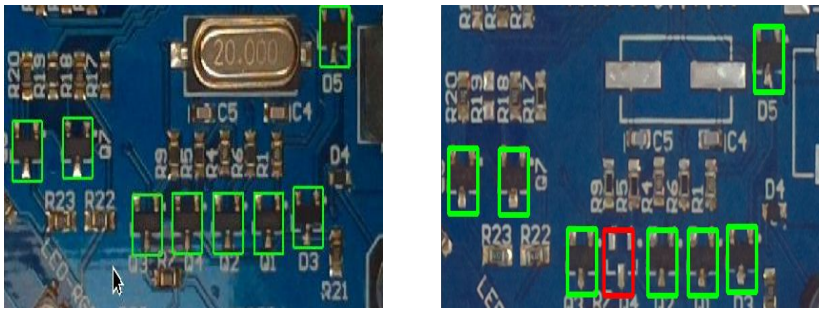
En esta etapa se comparan las posiciones de los dispositivos localizados por el algoritmo de identificación con las posiciones teóricas de donde debería de estar cada dispositivo. Esta comparación se realiza con un algoritmo de búsqueda y reconocimiento que se describe a continuación:

1. Se recibe el vector que contiene el número de dispositivos encontrados y su posición.

2. Se crea el vector **resultados** que contendrá el número de dispositivos que están colocados correctamente.
3. Se compara la posición teórica perfecta uno ([1]) con todos los dispositivos encontrados.
   a) Si existe conciencia ⇒ resultados[1] = 1
   b) Si no hay conciencias ⇒ resultados[1] = 0
4. El proceso continua hasta la ultima posición teórica [n]
5. Se retorna el vector resultados

El vector resultados contiene que dispositivo se encontró o no. Si el vector está lleno de unos significa que la tarjeta PCB tenía todos los dispositivos. Pero cada cero y su posición en el vector significa que el dispositivo está ausente en una parte especifica de la tarjeta.

El vector **resultados** se usa para dibujar el resultado del proceso de identificación sobre la imagen. En **color verde** se presentan los dispositivos encontrados (valores 1 en el vector resultados) y en **color rojo** se muestran los dispositivos ausentes (valores 0 en vector resultados). Debido a que se hace la identificación de diferentes dispositivos existen diferentes vectores resultados, es decir uno para cada dispositivo buscado. En la Figura 3.36 se muestra cómo se presentan los resultados de la identificación dependiendo de cada vector obtenido.



(a) Resultados = [1, 1, 1, 1, 1, 1, 1, 1]

(b) Resultados = [1, 1, 1, 1, 0, 1, 1, 1]

Figura 3.36: Resultado de identificación de transistores

3.3.4. Análisis de dispositivos SMD

Una vez confirmada la presencia de un dispositivo es necesario analizar el estado del mismo. Para realizar este análisis se desarrolló un algoritmo que determina la presencia de sus pines

y además la cantidad de soldadura que contiene cada uno. Se toma como base la posición encontrada por el algoritmo template matching y el algoritmo se ejecuta únicamente en el área abarcada por el dispositivo.

El algoritmo que realiza el análisis de los dispositivos se muestra en la Figura 3.37.

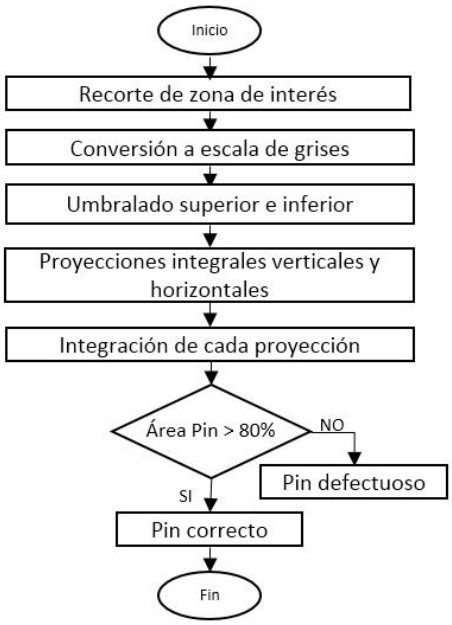
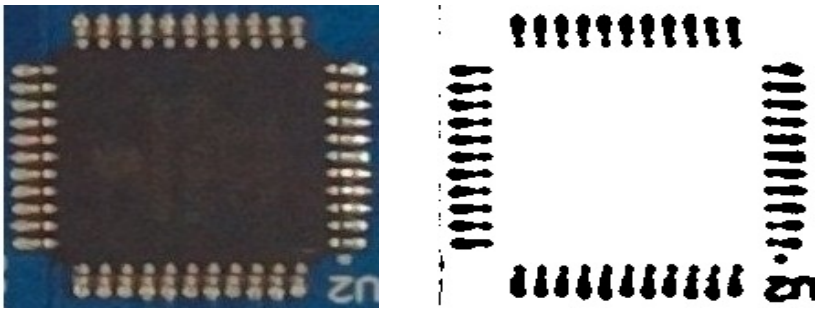


Figura 3.37: Diagrama de flujo de análisis de dispositivos SMD

En el primer paso se determina la zona a analizar, después se hace una conversión de una imagen RGB a escala de grises. En el siguiente paso se realiza un umbralado inferior y superior con el objetivo de resaltar los pines de un dispositivo como muestra la Figura 3.38.



(a) Dispositivo SMD

(b) Umbralado superior e inferior

Figura 3.38: Análisis de dispositivo SMD

Dependiendo el dispositivo a analizar se realizará un número determinado de proyecciones integrales. Los dispositivos del tipo **Dual in-line package(DIP)** se analizan haciendo dos proyecciones integrales verticales u horizontales. En el caso de dispositivos **Quad Flat Package(QFP)** es necesario realizar una proyección por cada linea de pines, es decir cuatro. En la Figura 3.39 se muestra el funcionamiento del algoritmo desarrollado para hacer cuatro proyecciones integrales.

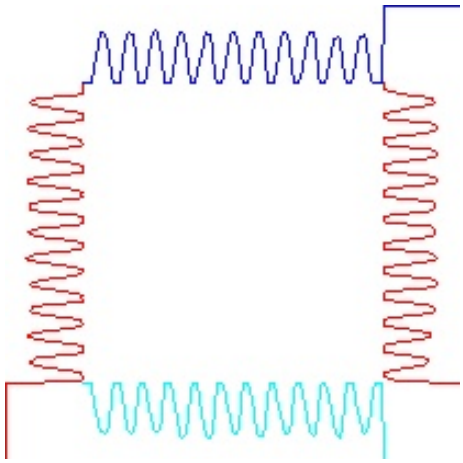


Figura 3.39: Proyección integral horizontal y vertical de dispositivo QFP

Sumando todos los píxeles que se encuentran bajo cada curva, se calcula el área correspondiente a cada pin. Este valor se compara con el valor 30, que es el área promedio que ocupa un pin en buen estado. Si el área calculada supera el 80 %(24 píxeles) respecto al valor de un pin correcto se considera como aceptable. De no ser así se marca como deficiente pero si no supera el 20 % se marca como inexistente.

3.4. Automatización del sistema

Para que la revisión de las tarjetas PCB se realice de manera automática es necesario que todos los componentes del sistema sigan una secuencia. En esta sección se describe el proceso de automatización del sistema, así como los componentes que lo componen y cómo se comunican entre ellos. Este proyecto se divide en tres bloques fundamentales:

1. La banda transportadora
2. El sistema de visión
3. La etapa de control

La automatización del sistema consiste en coordinar las tareas de estos tres bloques fundamentales. La lógica de funcionamiento que se plantea se muestra en el diagrama de flujo que se muestra en la Figura 3.40.

Figura 3.40: Diagrama de flujo de funcionamiento del sistema

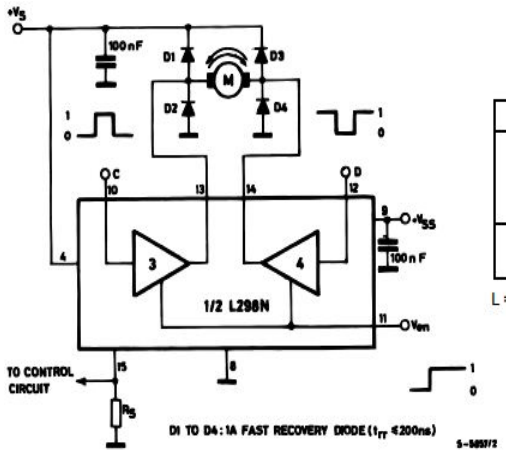
3.4.1. Control de la banda transportadora

En la sección de **implementación de banda transportadora** se presentaron los elementos que componen la banda transportadora y las razones de su construcción. En esta sección se presenta la puesta en marcha de la misma. Un motor de DC con caja de engranes es el elemento que transmite movimiento a la banda transportadora. Por esta razón la automatización de la banda transportadora se enfoca únicamente en el control del motor eléctrico. Las características del motor se muestran a continuación:

- Voltaje nominal: 12 V
- Par nominal: 1.3-3.7 Kg.cm
- Relación de reducción: 44, 56, 90
- Velocidad (sin carga): 45-85 RPM

Etapa de potencia

Las pruebas realizadas con este motor moviendo la banda transportadora nos indican que tiene un consumo de 300 mA y de 320 mA cuando mueve las tarjetas PCB. Para realizar la etapa de potencia del motor se decidió utilizar un puente H **L298N**. Este controlador tiene la capacidad de controlar hasta 2 motores de DC, además soporta 4 amperios en CD y posee un disipador térmico que evita que alcance elevadas temperaturas. La configuración para conectar este controlador con nuestro motor se obtuvo de la hoja de datos del controlador y se muestra en la Figura 3.41



Inputs		Function
V_{en} = H	C = H ; D = L	Forward
	C = L ; D = H	Reverse
	C = D	Fast Motor Stop
V_{en} = L	C = X ; D = X	Free Running Motor Stop

L = Low H = High X = Don't care

Figura 3.41: Configuración del L298N

Detección de posición

Para detectar la presencia de tarjetas que se mueven sobre la banda transportadora se hace uso de un sensor infrarrojo **Sharp GP2Y0A21YKOF** como el que se muestra en la Figura 3.42.

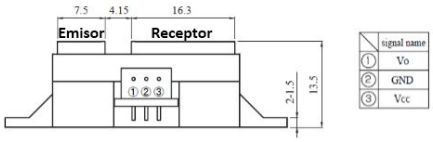


Figura 3.42: Sharp GP2Y0A21YKOF

Este tipo de sensores emplean el método de triangulación mediante un pequeño sensor de detección de posición (PSD) lineal para determinar la distancia de los objetos. Un emisor manda pulsos de luz infrarroja que son reflejados en un objeto que se encuentra dentro del campo de visión. El sensor PSD detecta el ángulo de reflexión formado, si el ángulo es pequeño significa que el objeto está lejos pero si el ángulo es grande entonces el objeto está cerca.

Este sensor puede medir la distancia de un objeto que se encuentra hasta 80 cm. Dependiendo la distancia y el tipo de material, el sensor genera un voltaje analógico por su pin $V_{\circ}$. La curva distancia-voltaje de este sensor se muestra en la Figura 3.43.

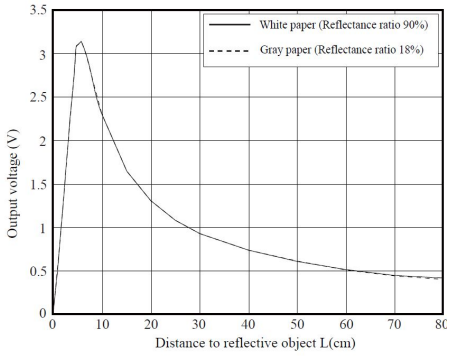


Figura 3.43: Curva Distancia-Voltaje

En este trabajo no se necesita conocer la distancia del objeto, simplemente se debe determinar su presencia. Para esto se implementó un amplificador operacional en configuración **comparador** con el circuito integrado **LM339N** como se muestra en la Figura 3.44.

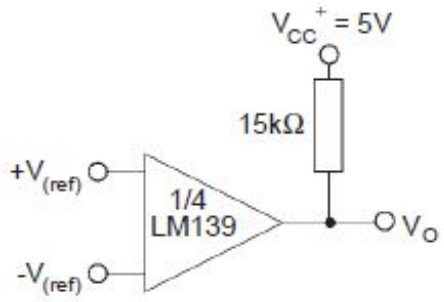


Figura 3.44: Amplificador operacional como comparador de voltaje

Para determinar el voltaje de referencia se realizó una pequeña caracterización del sensor, donde se midiera el voltaje generado por el sensor sin la presencia de la tarjeta PCB y otro con la presencia de ésta. Los valores obtenidos se muestran en la Tabla 3.1.

Presencia	Voltaje
Sin PCB	0.3
Con PCB	2.5

Tabla 3.1: Voltajes generados por el sensor infrarrojo

Tomando como base estos valores se propuso que el voltaje de referencia fuera de 1.8 V. Este voltaje se generó con un divisor de voltaje generado por una resistencia de $20K\Omega$ y otra de $33K\Omega$. Con esta configuración el comparador de voltaje entregaría 5V si el sensor detecta la presencia de alguna tarjeta y 0V si no se detecta nada.

Posicionamiento de la tarjeta PCB

Cuando el sensor detecta la presencia de una tarjeta PCB se necesita que ésta se posicione para poder hacer la toma de una fotografía. El posicionamiento de la tarjeta PCB es una etapa clave para obtener imágenes claras, sin efectos de movimiento y en una posición que facilite la etapa de procesamiento de imágenes. El posicionamiento de la tarjeta PCB es realizado por un **Servomotor de PWM**.

Un servomotor es un actuador que dispone en su interior un motor de CD, una caja de engranes, un sensor de posición y un circuito integrado que tiene una etapa de control [28]. Son llamados servomotores porque pueden mantener la posición de su rotor en un ángulo determinado. En este trabajo se usó un servomotor que mantiene la posición de su rotor en ángulos de 0° a 180° como se muestra en la Figura 3.45.

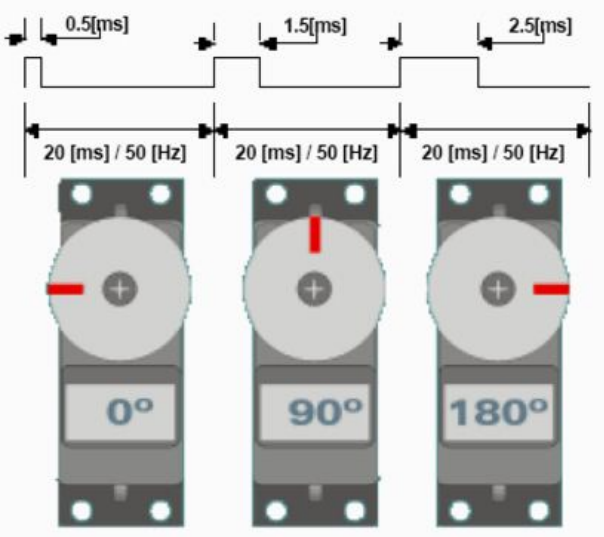


Figura 3.45: Funcionamiento de un servomotor de PWM

Para poder controlar la posición del rotor del servomotor es necesario aplicar un pulso a una de las tres terminales del mismo. Este pulso depende de la marca de cada servomotor. La mayoría de los servomotores necesitan recibir un pulso con una frecuencia de 50 Hz. El ciclo de trabajo (**Duty Cycle**) de este pulso determina la posición que toma el servomotor. En la Figura 3.45 se muestran los ciclos de trabajo que se deben aplicar a un servomotor para obtener diferentes posiciones de su rotor.

3.4.2. Desarrollo de programa e interfaz gráfica en QT Creator

En esta sección se describe el desarrollo del programa principal, el cual es el encargado de controlar las tareas que ejecuta la Raspberry Pi. Este programa y la interfaz gráfica están desarrollados en Qt Creator 4, que es una version antigua pero ligera, lo que la hace ideal para sistemas embebidos. Las tareas que ejecutan son las siguientes:

- Controlar todo el sistema desde GUI
- Interpretar las señales del sensor infrarrojo
- Controlar el motor de la banda transportadora
- Controlar el servomotor
- Indicar la captura de imagen

- Ejecutar los algoritmos de visión por computadora
- Presentar los resultados en interfaz gráfica

En la Figura 3.46 se muestra cómo es la interacción de todo el sistema de visión con el programa principal.

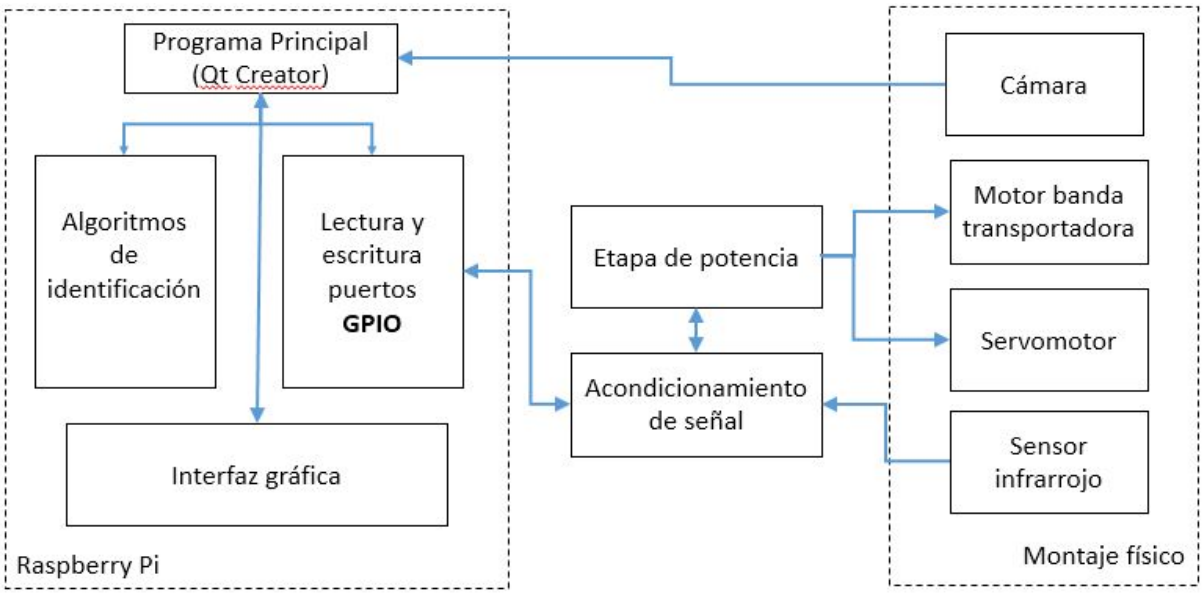


Figura 3.46: Tareas de programa principal

Como puede observarse el programa principal se divide en tres bloques principales: la interfaz gráfica, la lectura / escritura de puertos GPIO y la ejecución de algoritmos de visión por computadora. La ejecución de estos bloques simultáneamente posiblemente no seria un problema en una computadora personal o en programas especializados de cómputo, como Matlab y Labview. Pero en un sistema embebido los recursos son muy limitados y la buena administración de estos es indispensable para tener un buen desempeño.

Hilos de procesamiento

Al realizar diferentes pruebas de funcionamiento se presentó el problema de que la Raspberry Pi tenia problemas de desempeño al manipular los puertos GPIO y la interfaz gráfica. Ya que la interfaz gráfica presentaba problemas de congelamiento al hacer la lectura y escritura de puertos lo que ocasionaba no poder pulsar botones, además la información no se mostraba correctamente y en algunas ocasiones el programa se detenía inesperadamente.

Para solucionar este problema se evaluaron diferentes alternativas pero la que solucionó el problema fue el uso de hilos de procesamiento (threads). Al implementar hilos de procesamiento el usuario le especifica al procesador las secciones de código que debe ejecutar en paralelo. Con el objetivo de que el CPU organice los procesos y le de tiempos de ejecución específicos a cada segmento de código como se muestra en la Figura 3.47.

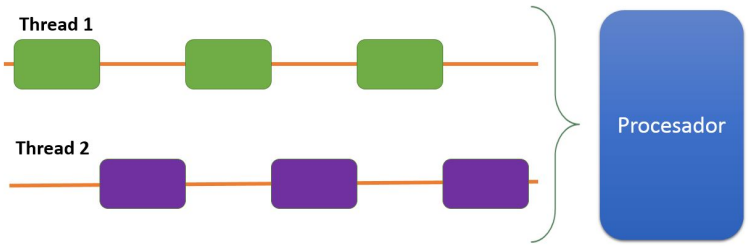


Figura 3.47: Ejecución con hilos

Con esto se organizan los recursos de una mejor manera permitiendo que las lecturas de puertos y la interfaz gráfica funcionen de manera correcta. El programa principal se dividió en dos hilos. El **hilo principal** ejecuta la GUI y los algoritmos de visión. Un **hilo secundario** el cual es el encargado de leer y escribir por los puertos GPIO.

Interfaz gráfica

La interfaz gráfica del programa se desarrollo en Qt Creator 4 y para poder hacer la comunicación entre los objetos de interfaz gráfica y los objetos del programa de visión se implementaron **Signals and Slots**, ya que sin esta implementación el intercambio de información es imposible. Debido a que los métodos de la interfaz gráfica no son públicos, por lo que se necesitan señales para acceder a estos. La interfaz gráfica del programa se muestra en la Figura 3.48.

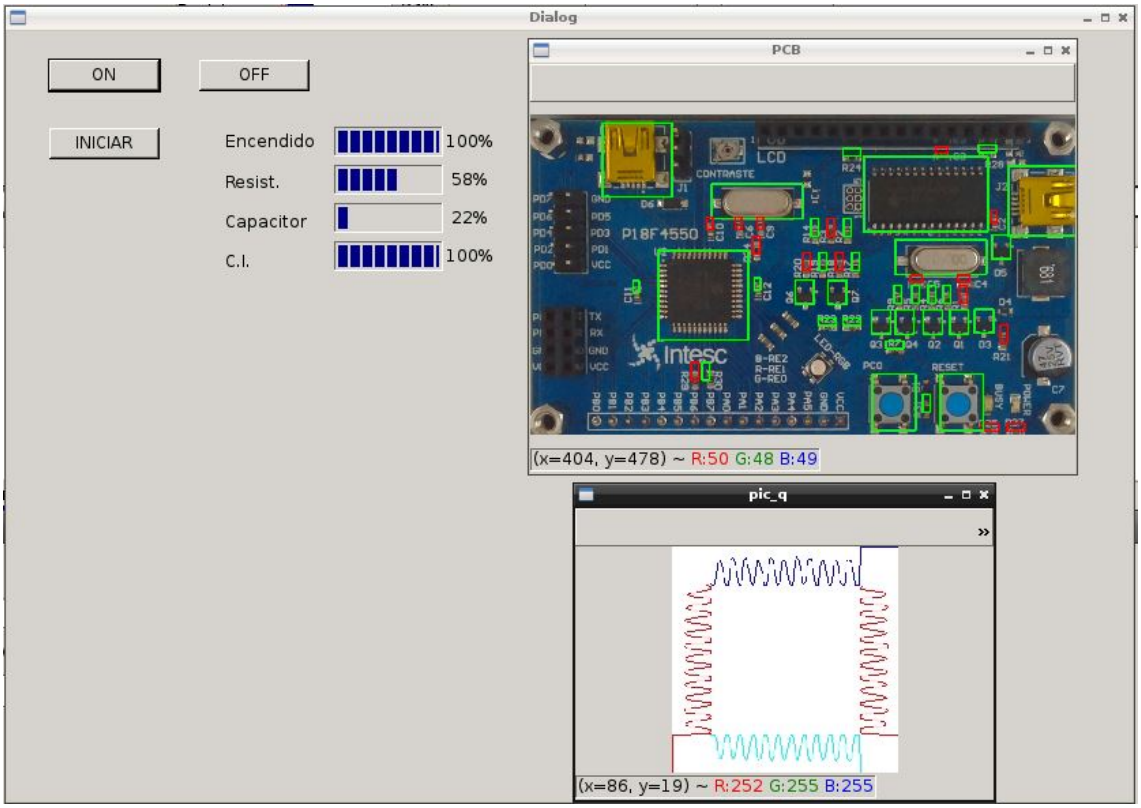


Figura 3.48: Interfaz gráfica del sistema de identificación

Los botones de la interfaz tienen las siguientes funciones:

- **ON** - Prende el sistema
- **OFF** - Apaga el sistema
- **INICIAR** - Inicia el sistema de identificación

Las barras de progreso nos muestran el porcentaje de identificación de cada elemento:

- **Resistencias** - Porcentaje de resistencias encontradas
- **Capacitores** - Porcentaje de capacitores encontrados
- **Transistores** - Porcentaje de transistores encontrados
- **Osciladores** - Porcentaje de osciladores encontrados

Las imágenes nos muestran el resultado de los análisis:

- **PCB** - Resultados gráficos de la identificación de dispositivos
- **pic_q** - Resultados gráficos del análisis del circuito integrado QFP
- **pic_d** - Resultados gráficos del análisis del circuito integrado DFP

En la imagen PCB se remarcan en color **verde** los dispositivos que fueron correctamente localizados y el color **rojo** indica que un dispositivo no se localizó o no estaba en la posición correcta.

Programa principal

El programa principal se desarrolló en C++ que es un lenguaje orientado a objetos, ya que el uso de objetos permite que la programación se realice de una manera ordenada, además las modificaciones del programa se implementan sin afectar todo el código.

Las bibliotecas de OpenCV no se pueden usar directamente en un sistema embebido, ya que son liberadas para procesadores específicos por lo que es necesario hacer un recompilado de estas. Este proceso se realizó con CMAKE y tardó aproximadamente 15 horas.

En la Figura 3.49 se muestra el diagrama de ejecución de los métodos principales del sistema de identificación de dispositivos.

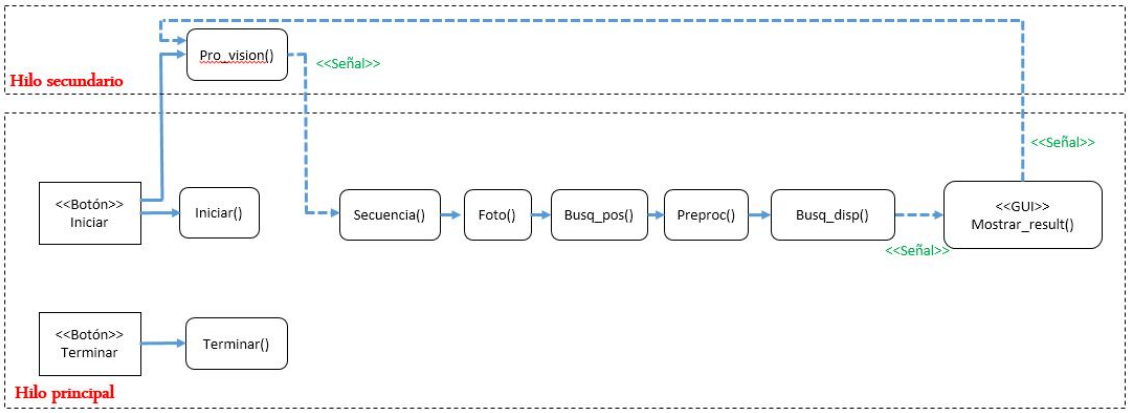


Figura 3.49: Diagrama de ejecución del programa principal

La descripción de los métodos principales se muestra a continuación:

- **Iniciar(Botón)**.- Invoca al método iniciar() e inicia el hilo de procesamiento secundario(Pro_vision()).
- **Terminar(Botón)**.- Apaga todo el sistema.
- **Iniciar()**.- Prende la banda transportadora y coloca al servomotor en posición.
- **Pro_vision()**.- Inicia una secuencia de lectura de puertos. Manda una señal de inicio a secuencia() al tener una lectura positiva del sensor.
- **Secuencia()**.- Controla la ejecución del algoritmo de visión.
- **Foto()**.- Captura de imagen.
- **Busq_pos()**.- Busca el extremo superior izquierdo y hace un recorte de la zona de interés.
- **Preproc()**.- Realiza la etapa de preprocesamiento.
- **Busq_disp()**.- Inicia los métodos de identificación de dispositivos.
  - Identificación de transistores(match_trans(), verif_trans()).
  - Identificación de resistencias(match_resist(), verif_resist()).
  - Identificación de capacitores(match_cap(), verif_cap()).
  - Identificación de osciladores(match_osc(), verif_osc()).
  - Identificación y análisis de circuitos integrados(match_ci(), verif_ci(), anali_smd2(),anali_smd4()).
  - Identificación de conectores USB(match_usb(), verif_usb()).
  - Identificación de botones(match_bot(), verif_bot()).
  - Al terminar manda una señal con los resultados a mostrar_result().
- **Mostrar_result()**.- Dibuja los resultados de la identificación de dispositivos sobre la imagen y presenta las estadísticas sobre barras de progreso en la interfaz gráfica. Vuelve a iniciar el método Pro_vision().

El programa principal está compuesto de 15 métodos, los cuales se encargan de que el sistema detecte las tarjetas PCB, las posicione, haga la identificación de dispositivos, muestre los resultados y se controle desde una interfaz gráfica.

Capítulo 4

Pruebas y resultados

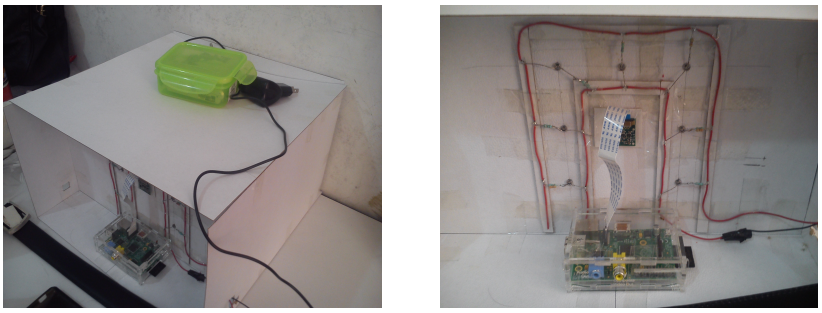
En este capítulo se muestran los resultados experimentales obtenidos cuando todos los elementos del sistema descritos en el capítulo anterior se ponen a trabajar juntos. Además se describen otro tipo de problemas que se presentan cuando se hace un sistema de visión.

4.1. Prototipos del sistema de identificación

El desarrollo de un sistema de visión depende del problema que se quiere resolver, es por esto que no existe una regla de diseño de estos. Además la teoría está enfocada a los algoritmos, temas como la iluminación y la captura de imágenes cuentan con principios físicos que sirven de guía. Pero la mejor imagen se obtiene realizando pruebas de los diferentes sistemas. Para obtener el sistema de visión final se tuvieron que elaborar diferentes prototipos, los cuales ayudaron a conocer mejor el problema.

En el primer prototipo se intentó hacer la identificación de dispositivos únicamente con la cámara y la Raspberry Pi. Este sistema no producía buenas imágenes, ya que simples cambios en la iluminación ambiental generaba imágenes muy diferentes.

En el segundo prototipo se intentó solucionar el problema con una iluminación constante y evitando la iluminación ambiental. Esto se intentó con el sistema que se muestra en la Figura 4.1.



(a) Prototipo dos (b) Sistema de Iluminación

Figura 4.1: Segundo Prototipo

Como puede observarse este sistema tiene una iluminación del tipo anillo lo que nos ayudaría a tener imágenes muy claras. Lamentablemente se generaban imágenes con grandes reflejos, además el posicionamiento de la tarjeta se tenía que realizar de manera manual.

En el tercer prototipo se agregó la banda transportadora que ayudaría a que la inspección se hiciera de manera automática, se probaron diferentes sistemas de iluminación y se determinó que la iluminación por campo oscuro era la óptima. Además se agregó un microcontrolador que automatizaba el sistema como se muestra en la Figura 4.2.

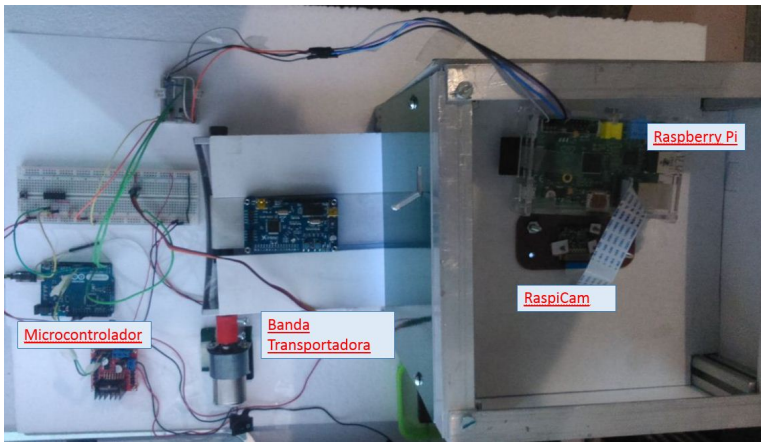
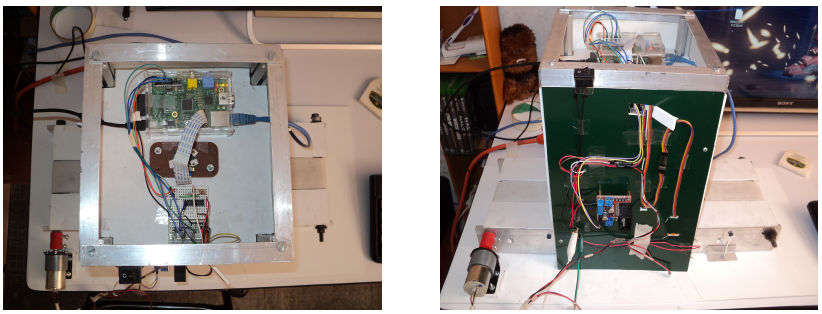


Figura 4.2: Tercer prototipo

El microcontrolador se comunicaba vía comunicación serial RS-232 con la Raspberry Pi y con ésto se automatizó el sistema. Este prototipo realizaba correctamente todas las tareas pero el microcontrolador era un sistema **redundante**, ya que su función podía ser realizada por la Raspberry Pi. Además agregaba complejidad en hardware al necesitar acoplar el voltaje del

microcontrolador (5v) y el de la Raspberry Pi(3.3v), además se necesitaba otro compilador y programa únicamente para el funcionamiento del microcontrolador.

Por esta razón se decidió eliminar el microcontrolador y que la Raspberry Pi ejecutara también las funciones del microcontrolador. Con esta decisión se llegó al prototipo final que se muestra en la Figura 4.3



(a) Vista Superior (b) Vista lateral

Figura 4.3: Prototipo Final

La eliminación del microcontrolador no fue simple, ya que los recursos limitados de la Raspberry Pi hicieron que la programación fuera más compleja obligando a usar hilos de procesamiento y métodos que se describieron en la sección **3.4.2 Desarrollo de programa e interfaz gráfica**.

4.2. Pruebas de funcionamiento

Para evaluar el desempeño del sistema de inspección la empresa que fabrica las tarjetas de desarrollo proporcionó una serie de tarjetas, las cuales presentan diferentes escenarios de falla que ocurren durante la producción de estas.

En el primer escenario se evaluó una tarjeta, la cual contaba con todos sus dispositivos. Esta evaluación se realizó con todos los parámetros que se obtuvieron en el desarrollo práctico. En la Figura 4.4 se muestra el resultado luego de poner a funcionar el sistema y colocar la tarjeta sobre la banda dejando que la identificación se realizara de manera automática.

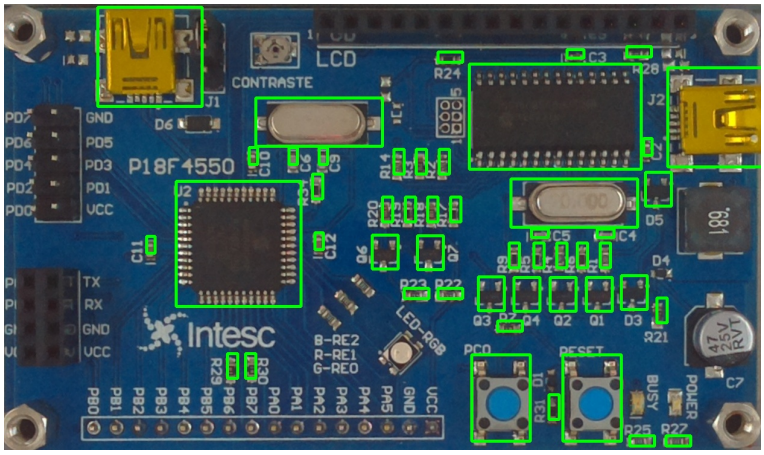
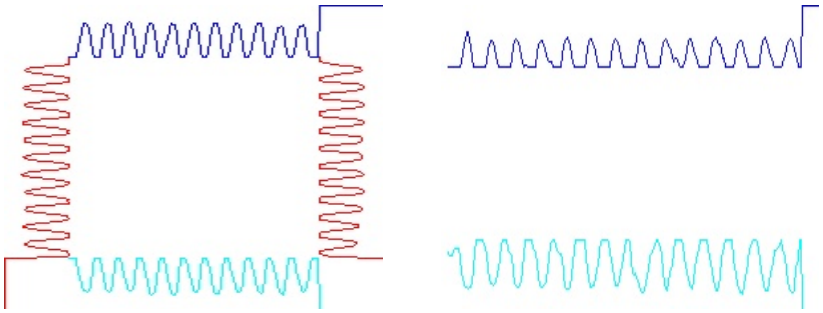


Figura 4.4: Prueba número uno

Como se puede observar, se identificaron todos los dispositivos de la tarjeta. Debido a que se detectó la presencia de circuitos integrados, el sistema prosigue a hacer un análisis de estos. En la Figura 4.5 se muestra el resultado de este análisis.



(a) C.I tipo QFP (b) C.I tipo DFP

Figura 4.5: Análisis de circuitos de prueba uno

El análisis, basado en proyecciones integrales tanto horizontales como verticales, indica que los pines de ambos circuitos se encuentran en perfectas condiciones. Debido a que la separación entre cada pin es la adecuada y además la magnitud de soldadura es suficiente, se determina que los circuitos integrados se ensamblaron correctamente. Cabe mencionar que la proyección de los pines superiores del circuito DFP no presentan la misma magnitud que los inferiores, esto se debe a que la altura del circuito y la perspectiva de la cámara ocasionan que exista una sombra y se observen más pequeños los pines, sin embargo, permite apreciar de buena manera las características de los pines superiores, como son la separación y la magnitud.

La siguiente prueba consiste en evaluar el caso contrario, es decir, una tarjeta que no tenga ningún dispositivo. Con esto se evalúa la respuesta del algoritmo ante la ausencia de cada tipo de dispositivo. En la Figura 4.6 se presenta el resultado obtenido después de evaluar la tarjeta con el sistema de inspección.

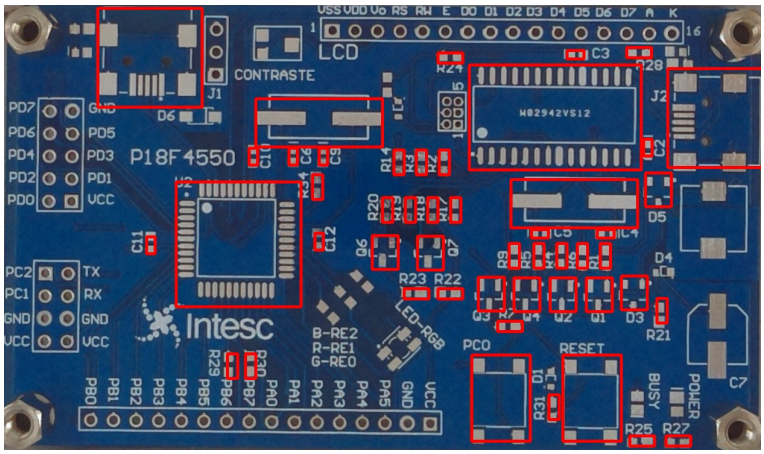


Figura 4.6: Prueba número dos

El sistema en este escenario no identificó ningún dispositivo, lo que nos muestra que el sistema tiene un buen desempeño. Sin embargo, en ocasiones se presentan falsos positivos con las resistencias y capacitores, debido a que la diferencia que se presenta entre el patrón y la imagen es muy pequeña cuando estos dispositivos están presentes o no. Al no identificar circuitos integrados el algoritmo no realiza ningún análisis sobre estos.

En la tercer prueba se hace la evaluación de una tarjeta a la que le faltan diferentes dispositivos. Esta evaluación ayuda a simular una situación real, donde a simple vista parece que se encuentran todos los dispositivos, ya que están presentes la mayoría, lo que puede ocasionar confusiones debido a que a simple vista parecen estar todos los componentes. Al poner la tarjeta dentro del sistema nos muestra los resultados de la Figura 5.

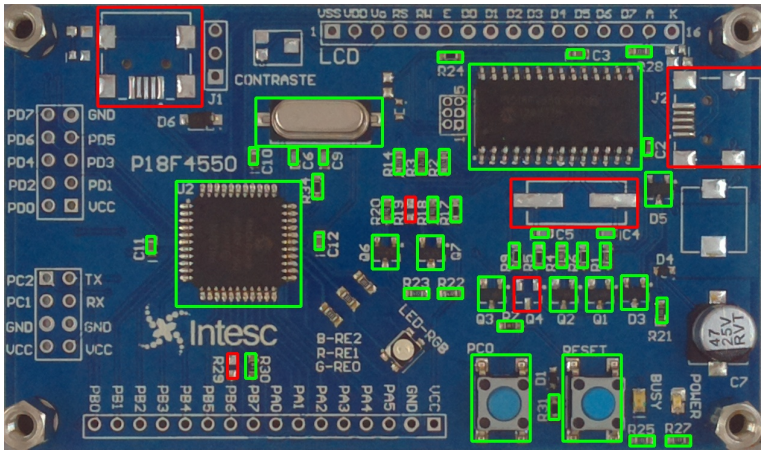
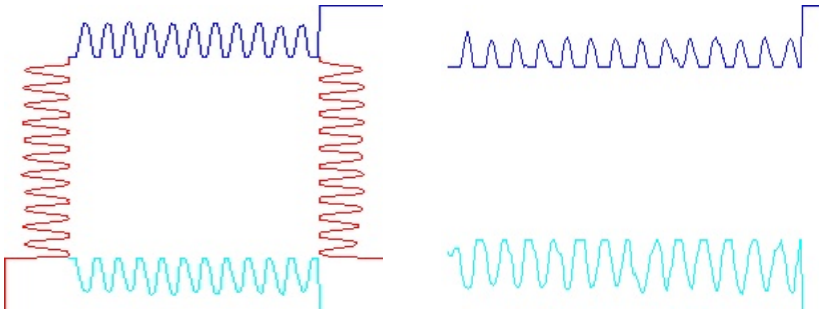


Figura 4.7: Prueba número tres

Como se puede observar el sistema muestra en color rojo la ubicación de los dispositivos que faltan en la tarjeta analizada. En esta tarjeta los dispositivos ausentes son:

- 1 Oscilador
- 1 Transistor
- 2 Resistencias
- 2 Conectores USB

Con esto se obtiene nuevamente un correcto funcionamiento del sistema y la tarjeta tendrá que ser sometida a un proceso de retrabajo para colocar los dispositivos faltantes. Los circuitos integrados se encontraron dentro condiciones aceptables como se muestra en la Figura 4.8.



(a) C.I tipo QFP (b) C.I tipo DFP

Figura 4.8: Análisis de circuitos prueba tres

En la cuarta prueba se evaluó una tarjeta diferente. Al someter esta tarjeta en el sistema se obtuvieron los resultados que se muestran en la Figura 4.9.

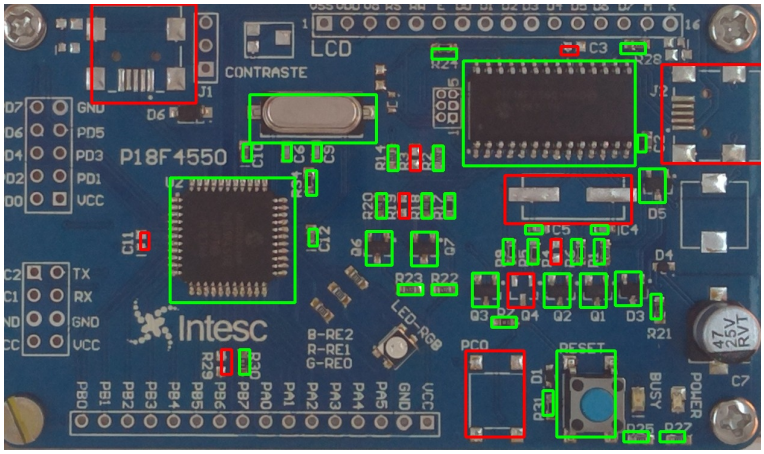
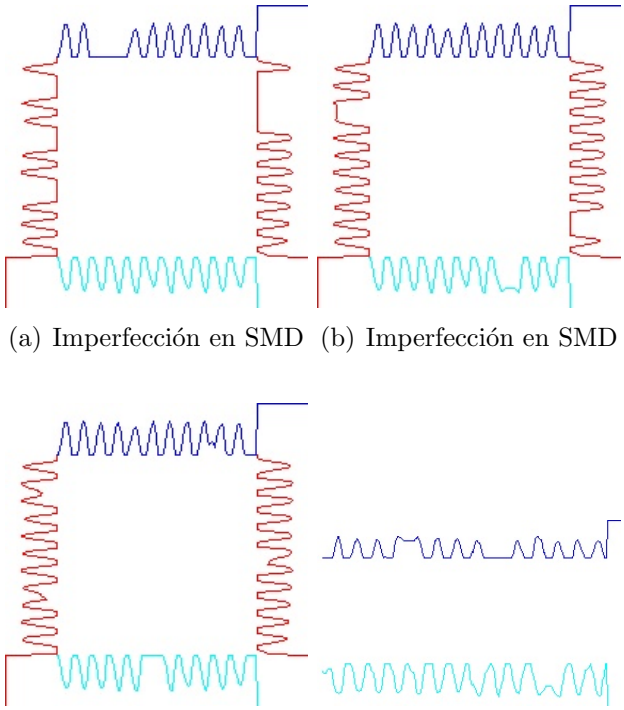


Figura 4.9: Prueba número cuatro

La etapa de identificación de componentes se realizó de manera exitosa y los circuitos integrados no presentaron ninguna imperfección. La etapa de identificación mostró que faltaban los siguientes componentes:

- 1 Oscilador
- 1 Transistor
- 4 Resistencias
- 2 Capacitores
- 2 Conectores USB
- 1 Push Button

Posteriormente se le realizaron modificaciones a los pines de los circuitos integrados con el objetivo de comprobar el desempeño del algoritmo de evaluación de estos. Estas modificaciones incluían poner sobre los pines un exceso de soldadura o un material de otro color. En la Figura 4.10 se presentan los resultados de cuatro diferentes pruebas realizadas.



(a) Imperfección en SMD (b) Imperfección en SMD

(c) Imperfección en SMD (d) Imperfección en SMD

Figura 4.10: Imperfecciones en C.I. SMD

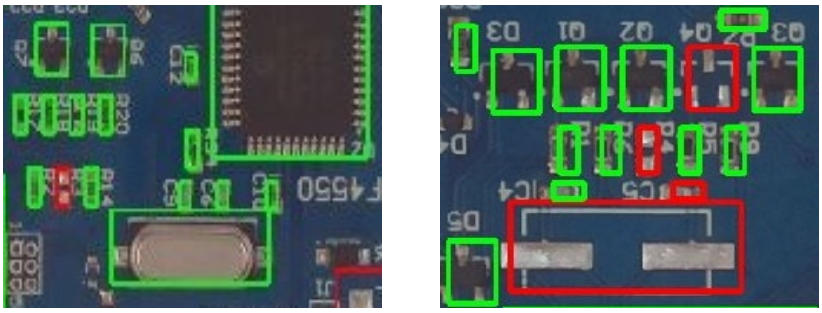
Para calcular la efectividad de identificación del sistema de inspección, se evaluaron 15 nuevas tarjetas, en las cuales se modificaba el tipo de ausencia, es decir, a cada tarjeta se le quitaba un componente diferente en cada prueba. Se obtenía un porcentaje de efectividad del 100 %, si el sistema identificaba correctamente la presencia de cada dispositivo y además no hacia falsas identificaciones (falsos positivos). El porcentaje de efectividad bajaba, si el sistema no fue capaz de identificar un dispositivo (falso negativo) o generó una identificación errónea. Los resultados que el sistema mostró en la interfaz gráfica se registraron y se compararon como se describió anteriormente. En la tabla 4.1 se muestran los resultados de todo este conjunto de pruebas.

Como se puede observar en la tabla 4.1, el algoritmo identifica perfectamente la presencia de osciladores, circuitos integrados, botones y conectores USB, es decir, tiene una precisión del 100 % en dispositivos de un tamaño grande. Ésto se debe a que son dispositivos que tienen formas y colores muy específicos. También se identifican sin error dispositivos más pequeños como transistores, a pesar de que son muy parecidos a otros dispositivos. El algoritmo empieza a tener deficiencias con dispositivos muy pequeños, debido a que la resolución de la cámara no permite tener una imagen a detalle de estos. Con esta resolución el tamaño de estos dispositivos es de 14 x 28 píxeles, lo que ocasiona que muchas de sus características no se definan correctamente o se pierdan. En la Figura 4.11 se presentan los errores que se obtuvieron con el algoritmo

	Porcentaje efectividad						
# Prueba	Transistores	Osciladores	C.I.	Resistencias	Capacitores	Conectores USB	Botones
1	100 %	100 %	100 %	100 %	100 %	100 %	100 %
2	100 %	100 %	100 %	100 %	100 %	100 %	100 %
3	100 %	100 %	100 %	92 %	100 %	100 %	100 %
4	100 %	100 %	100 %	96 %	89 %	100 %	100 %
5	100 %	100 %	100 %	88 %	89 %	100 %	100 %
6	100 %	100 %	100 %	100 %	89 %	100 %	100 %
7	100 %	100 %	100 %	96 %	100 %	100 %	100 %
8	100 %	100 %	100 %	100 %	89 %	100 %	100 %
9	100 %	100 %	100 %	96 %	100 %	100 %	100 %
10	100 %	100 %	100 %	88 %	89 %	100 %	100 %
11	100 %	100 %	100 %	88 %	100 %	100 %	100 %
12	100 %	100 %	100 %	92 %	89 %	100 %	100 %
13	100 %	100 %	100 %	100 %	89 %	100 %	100 %
14	100 %	100 %	100 %	96 %	89 %	100 %	100 %
15	100 %	100 %	100 %	92 %	100 %	100 %	100 %

Tabla 4.1: Resumen de evaluaciones

de inspección al identificar resistencias y capacitores. El sistema identificó la presencia de la resistencia con la etiqueta R19, sin embargo en esta posición no estaba presente una resistencia como se muestra en la Figura 4.11(a). En ocasiones, el algoritmo no hacía la identificación de un dispositivo a pesar de que estaba presente como se puede observar en la Figura 4.11(b) con el capacitor con la etiqueta C5.



(a) Falso positivo (b) Falso Negativo

Figura 4.11: Deficiencias en la identificación

Los errores mostrados en la Figura 4.11, corresponden a la prueba número cuatro de la tabla 4.1. Se obtuvo un porcentaje de efectividad en la identificación de resistencias del 96 % porque se identificó de manera errónea una resistencia. Al tener un error en la identificación de un capacitor el porcentaje de efectividad de capacitores baja hasta 89 %, ya que el número de capacitores es menor.

De manera similar en la prueba número once, el sistema identificó de manera errónea tres

resistencias, lo que corresponde a un porcentaje de efectividad del 88 %, aunque en esta prueba el sistema identifico de manera exitosa los capacitores. En la prueba número catorce no se identificaron correctamente un capacitor y una resistencia ocasionando unos porcentajes de efectividad del 89 % y 96 % respectivamente.

Un factor que aumenta la posibilidad de encontrar falsos positivos y negativos, es el contraste entre una resistencia y el fondo, ya que los sockets donde se sueldan los extremos de la resistencia son casi del mismo color y tamaño que los extremos de una resistencia, lo que ocasiona que exista un gran parentesco entre el socket y la resistencia. Este problema se corregiría de manera fácil con una cámara de mayor resolución, ya que se tendrían más detalles del dispositivo que permitan hacer la distinción entre el fondo y la resistencia. El único inconveniente sería que los costos y el tiempo de procesamiento aumentarían. De manera general, se obtiene un buen porcentaje de identificación por el sistema y el algoritmo.

El sistema de visión realiza todas sus funciones en aproximadamente 18 segundos, los tiempos de procesamiento de cada etapa se presentan en la tabla 4.2.

# Etapa	Tiempo de procesamiento
Captura de imagen	4 s
Pre procesamiento	2 s
Identificación	10 s
Análisis SMD	2 s
Total	18 s

Tabla 4.2: Tiempos de procesamiento del sistema

Como se puede observar en la tabla 4.2, la etapa que consume mayor tiempo es la de identificación, esto se debe a que se ejecutan algoritmos que realizan una gran cantidad de operaciones como el método de coeficiente de correlación normalizado. La siguiente etapa que consume más tiempo es la captura de imagen, ya que la imagen capturada es de una resolución alta, lo que ocasiona que a la Raspberry Pi le tome tiempo poder guardarla. Las etapas de pre procesamiento y análisis SMD no consumen mucho tiempo porque en estas se realizan operaciones básicas.

Capítulo 5

Conclusiones

En el presente trabajo se desarrolló una sinergia entre visión por computadora y sistemas embebidos, para crear un sistema de inspección industrial. Se involucraron dispositivos que permitieron crear un sistema de inspección automático de bajo costo que ofrece resultados similares a los reportados en la literatura y cuyas características principales son:

- Un promedio de efectividad en la identificación de dispositivos del 97 %.
- Análisis del ensamble de dispositivos SMD en la tarjeta PCB.
- Los resultados de la inspección se presentan de manera clara en una interfaz gráfica.
- Se hace uso de software Open Source, que permite crear sistemas económicos, ya que no se tienen que pagar costos por licencias.
- Se implementa un sistema de visión con únicamente una Raspberry Pi modelo B, la cual cuenta con Linux embebido.
- Se obtienen buenos resultados usando sólo una cámara de 5 Megapíxeles.
- Es posible aumentar el tipo de tarjetas a inspeccionar únicamente agregándolas a la lógica de programación, ya que el algoritmo principal permite la inclusión de nuevas tarjetas.

El proceso de desarrollo para realizar la identificación de dispositivos en una tarjeta PCB fue un reto muy grande principalmente por tres razones. La primera es que en un espacio muy reducido se encuentra una gran cantidad de información, como son texturas, formas y colores. Ésto ocasiona que no exista un método o algoritmo óptimo, por lo que se tuvieron que ocupar combinaciones o ambos. La segunda razón consiste en que las pequeñas dimensiones de los componentes obligan a que se trabaje con resoluciones altas, lo que ocasiona que los tiempos

de procesamiento aumenten o se obligue a aumentar el poder de procesamiento de todos los sistemas. La última razón es que se analizaban elementos metálicos y soldadura, por lo que encontrar un sistema de iluminación adecuado fue un proceso muy largo y difícil debido a que los reflejos que se generaban eran muy grandes.

Trabajar con software Open Source me permitió aumentar mis conocimientos más allá de la programación, ya que muchos programas de licencia de propietario te permiten desarrollar aplicaciones de manera muy fácil, tienen módulos más especializados, además son muy robustos a detalles de programación, ésto es algo muy útil, sin embargo, cuando se trata de aprender es algo contraproducente, ya que se esconden los detalles que permiten comprender mejor una disciplina al esconder detalles importantes.

A pesar de resolver satisfactoriamente el problema, el trabajo cuenta con puntos de mejora, los cuales son:

- La resolución de la cámara no permite hacer la distinción de las matrículas de las resistencias SMD, ya que son números muy pequeños. Por ende se tendría que recurrir a una cámara con mayor resolución y además telecéntrica.
- La gran complejidad del algoritmo llevó al límite a la Raspberry Pi modelo B, ya que el procesamiento se realiza en más de 10 segundos.

Como trabajo futuro se proponen explotar los siguientes aspectos:

- La inspección de más tarjetas de desarrollo de la misma empresa que nos proporcionó la tarjeta presentada en este trabajo.
- Migrar el sistema a una Raspberri Pi 2 que cuenta con 4 núcleos y un 1Gb de memoria RAM. Con el objetivo de que la inspección sea 4 veces más rápida.
- La implementación como servidor de todo el sistema, con el objetivo de que se pueda controlar desde cualquier computadora que conozca su dirección IP y así evitar el uso de un monitor.
- La optimización del sistema operativo para que sea de uso exclusivo del sistema de visión y se convierta en un dispositivo "plug and play"para su venta.

Publicaciones

En esta sección se presenta la evidencia de participación en III Congreso Internacional de Robótica y Computación (CIRC 2016), realizado en Los Cabos, Baja California Sur, del 2 al 4 Mayo del 2016. Se presentó la ponencia llamada "Sistema mecatrónico para la identificación de dispositivos en tarjetas PCB usando visión por computadora", cuyo articulo fue incluido en las memorias del evento.

SEP

SECRETARÍA DE EDUCACIÓN PÚBLICA

EL INSTITUTO TECNOLÓGICO DE LA PAZ

OTORGA EL PRESENTE

RECONOCIMIENTO

A

JUAN CARLOS CAMPECHE VALENCIA, M.C. VERÓNICA EDITH BAUTISTA LÓPEZ, M.C. CARLOS GARCÍA LUCERO, DR. SALVADOR AYALA RAGGI Y DR. ALDRIN BARRETO FLORES

POR SU ARTÍCULO: SISTEMA MECATRÓNICO PARA LA IDENTIFICACIÓN DE DISPOSITIVOS EN TARJETAS PCB USANDO VISIÓN POR COMPUTADORA
PRESENTADO EN EL TERCER CONGRESO INTERNACIONAL DE ROBÓTICA Y COMPUTACIÓN
CELEBRADO DEL 2 AL 4 DE MAYO DEL AÑO EN CURSO.

LA PAZ, B.C.S., 4 DE MAYO DE 2016.

SECRETARÍA DE EDUCACIÓN PÚBLICA
TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE LA PAZ
DIRECCIÓN

ING. JESÚS DAVID ESTRADA RUIZ
DIRECTOR

84

Investigación y Desarrollo en Robótica y Computación

Sistema mecatrónico para la identificación de dispositivos en tarjetas PCB usando visión por computadora

Juan Carlos Campeche Valencia[1], Aldrin Barreto Flores[1], Verónica Edith Bautista López[2], Carlos García Lucero[1], Salvador E. Ayala Raggi[1]

Facultad de Ciencias de la Electrónica[1]
Facultad de Ciencias de la Computación[2]
Benemérita Universidad Autónoma de Puebla
Av. San Claudio y 18 Sur, Ciudad Universitaria, Puebla, Puebla, México
Campeche.200928008mail.com

Abstract— **The automated optical inspection (AOI) in the electronic manufacturing industry is a critical process to reduce manufacturing cost and assembly issues. This paper describes a mechatronic system that is able to identify if an electronic device or component is on a printed circuit board (PCB) using machine vision techniques, also a conveyor belt is used to automate the inspection process. The system mainly include a illumination system, an image processing module, algorithms of machine vision based on OpenCV and implemented in a Raspberry Pi, an user interface made in QT Creator and a microcontroller that is employed to automate the system. This system is a low cost alternative in the non-contact and non-destructive inspection systems.**

Keywords— **Automated optical inspection; machine vision; Raspberry PI.**

I. Introducción

En el proceso de manufactura de un circuito impreso existen principalmente 2 tipos de inspecciones para encontrar fallas: las de contacto eléctrico y sin contacto eléctrico. La inspección que no implica contacto eléctrico evalúa la ausencia, la mala colocación de dispositivos, exceso o falta de soldadura y el tamaño de canales conductores [1]. Esta inspección se puede realizar por personas pero el proceso es muy lento y es muy susceptible a errores humanos debido al cansancio y a las pequeñas dimensiones de los dispositivos, es por esto que se han implementado sistemas de visión por computadora para realizar una inspección que garantice la presencia de los dispositivos. La inspección de tarjetas PCB es un campo en crecimiento, ya que la industria cuenta con alrededor de 2800 empresas y continúan creciendo [2], lo que ocasiona que existan investigaciones como en [3] donde se verifica la posición de resistencias superficiales o [4] donde se usa aprendizaje basado en programación genética para encontrar los posibles defectos. Lamentablemente estos sistemas son muy exclusivos debido a su alto costo por lo que solo pueden ser adquiridos por empresas con grandes niveles de producción, por esta razón se propone el desarrollo de un sistema basado en software Open Source, como son las bibliotecas de visión por computadora de OpenCV y el entorno de desarrollo del lenguaje C++ como QT Creator, además de que el sistema estará soportado en una Raspberry Pi lo que lo convierte en un sistema pequeño y económico.

II. Metodología

Para realizar el sistema de inspección automática óptica se desarrolló un sistema de visión por computadora compuesto por una cámara digital (RaspiCam), una Raspberry Pi, un sistema de iluminación, un servomotor, un sensor infrarrojo, un microcontrolador Arduino Leonardo y una banda transportadora. La tarjeta PCB a ser evaluada es una tarjeta de desarrollo para microcontroladores, la cual tiene una gran diversidad de dispositivos que pueden ser identificados. Estas tarjetas PCB se colocan sobre la banda transportadora, posteriormente un sensor infrarrojo detecta la presencia de una tarjeta y un servomotor posiciona la tarjeta para iniciar la toma de imágenes. Se capturan las imágenes con un fondo blanco para elevar el contraste, un entorno cerrado para evitar la entrada de luz ambiental y tener una iluminación controlada. La fuente de luz son LEDs blancos difusos y se evita la iluminación frontal porque las tarjetas PCB son superficies altamente reflejantes. La primera parte del algoritmo de inspección consiste en hacer un preprocesamiento de la imagen para eliminar zonas que no son de interés y resaltar componentes importantes. En la Figura 1 se muestra el diagrama de flujo del funcionamiento general del sistema de inspección.

ISBN: 978-607-97128-2-2

Bibliografía

[1] OpenCV, *www.opencv.org*.

[2] Q. Creator, *www.qt.io/ide*.

[3] Adafruit, *www.adafruit.com*.

[4] M. Moganti, F. Ercal, C. H. Dagli, and S. Tsunekawa, "Automatic pcb inspection algorithms: a survey," *Computer vision and image understanding*, vol. 63, no. 2, pp. 287–313, 1996.

[5] F. Xie, A. Uitdenbogerd, and A. Song, "Detecting pcb component placement defects by genetic programming," pp. 1138–1145, 2013.

[6] D. Koniar, L. Hargas, A. Simonova, M. Hrianka, and Z. Loncova, "Virtual instrumentation for visual inspection in mechatronic applications," *Procedia Engineering*, vol. 96, pp. 227–234, 2014.

[7] F. Guo and S.-a. Guan, "Research of the machine vision based pcb defect inspection system," pp. 472–475, 2011.

[8] S.-M. Kim, Y.-C. Lee, and S.-C. Lee, "Vision based automatic inspection system for nuts welded on the support hinge," *SICE-ICASE, 2006. International Joint Conference*, pp. 1508–1512, 2006.

[9] H. Wu, G. Feng, H. Li, and X. Zeng, "Automated visual inspection of surface mounted chip components," *Mechatronics and Automation (ICMA), 2010 International Conference on*, pp. 1789–1794, 2010.

[10] X.-N. Tang and Y.-N. Wang, "Visual inspection of workpiece quality," pp. 434–438, 2011.

[11] D. Schneider, T. Holtermann, F. Neumann, A. Hehl, T. Aach, and T. Gries, "A vision based system for high precision online fabric defect detection," pp. 1494–1499, 2012.

[12] X. Tao, H. Li, J. Lu, and X. Wang, "Computer vision technology for seafood quality evaluation," pp. 1888–1891, 2012.

[13] D. Demir, S. Birecik, F. Kurugöllü, M. Sezgin, İ. Ö. Bucak, B. Sankur, and E. Anarim, "Quality inspection in pcbs and smds using computer vision techniques," vol. 2, pp. 857–861, 1994.

[14] M. Yu, G.-Y. Jiang, S.-L. He, B.-K. Yu, and R.-D. Fu, "New approach to vision-based bga package inspection," vol. 2, pp. 1107–1110, 2002.

[15] H. Zhao, J. Cheng, and J. Jin, "Ni vision based automatic optical inspection (aoi) for surface mount devices," pp. 352–6, 2009.

[16] N. S. S. Mar, P. Yarlagadda, and C. Fookes, "Design and development of automatic visual inspection system for pcb manufacturing," *Robotics and Computer-Integrated Manufacturing*, vol. 27, no. 5, pp. 949–962, 2011.

[17] N. Dong, C.-H. Wu, W.-H. Ip, Z.-Q. Chen, and K.-L. Yung, "Chaotic species based particle swarm optimization algorithms and its application in pcb components detection," *Expert Systems with Applications*, vol. 39, no. 16, pp. 12501–12511, 2012.

[18] M. Can, M. JianXu, and M. JianPing, "Research and develop on pcb defect intelligent visual inspection robot," pp. 1–4, 2012.

[19] H. M. Judi, K. Kapoyah, and N. Sha'ari, "Visualising automatic product inspection of pcb units," vol. 1, pp. 204–207, 2009.

[20] R. C. Gonzalez, *Digital image processing*. Pearson Education India, 2009.

[21] R. Szeliski, *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[22] E. Cuevas, D. Zaldívar, and M. Pérez, *Procesamiento digital de imágenes con MATLAB y Simulink*. 2010.

[23] G. Bradski and A. Kaehler, *Learning OpenCV: Computer vision with the OpenCV library*. .º'Reilly Media, Inc.", 2008.

[24] D. Martin, "A practical guide to machine vision lighting," *Midwest Sales and Support Manager, Advanced illumination*, 2007.

[25] W. D. Callister, *Introducción a la ciencia e ingeniería de los materiales 2*, vol. 1. Reverté, 2002.

[26] R. Laganière, *OpenCV 2 Computer Vision Application Programming Cookbook: Over 50 recipes to master this library of programming functions for real-time computer vision*. Packt Publishing Ltd, 2011.

[27] I. P. S.A.I.C, "Manual de cálculo de cintas transportadoras," *Division articulos varios*.

[28] B. C. Kuo, *Sistemas de control automático*. Pearson Educación, 1996.