FACULTAD DE INGENIERÍA Y ARQUITECTURA

ESCUELA PROFESIONAL DE INGENIERÍA ELECTRÓNICA

DISEÑO DE UNA APLICACIÓN MÓVIL EN EL SISTEMA OPERATIVO ANDROID PARA EL RECONOCIMIENTO DE CÓDIGOS DE BARRAS UTILIZANDO PROCESAMIENTO DIGITAL DE IMÁGENES Y MODELO CLIENTE-SERVIDOR PARA ÁREAS DE SOPORTE TÉCNICO

PRESENTADA POR

PABLO MANUEL SAAVEDRA BARRERA

JORGE LUIS SOLORZANO PUENTE

TESIS PARA OPTAR EL TÍTULO DE INGENIERO ELECTRÓNICO

LIMA – PERÚ

2014

USMP
UNIVERSIDAD DE SAN MARTIN DE PORRES
FACULTAD DE INGENIERÍA Y ARQUITECTURA

ESCUELA PROFESIONAL DE INGENIERÍA ELECTRÓNICA

DISEÑO DE UNA APLICACIÓN MÓVIL EN EL SISTEMA OPERATIVO ANDROID PARA EL RECONOCIMIENTO DE CÓDIGOS DE BARRAS UTILIZANDO PROCESAMIENTO DIGITAL DE IMÁGENES Y MODELO CLIENTE - SERVIDOR PARA ÁREAS DE SOPORTE TÉCNICO

TESIS

PARA OPTAR EL TÍTULO PROFESIONAL DE INGENIERO ELECTRÓNICO

PRESENTADO POR

SAAVEDRA BARRERA, PABLO MANUEL
SOLORZANO PUENTE, JORGE LUIS

LIMA, PERÚ

2014

ÍNDICE

RESUMEN

La presente tesis tiene como objetivo ofrecer una solución para el mantenimiento y control de los equipos dentro de un ambiente de trabajo, sea industrial, educacional, almacén, entre otros. En nuestro país, se suele realizar el mantenimiento de un equipo llevando una bitácora del mismo en documentos impresos, que tienden a perderse o a acumular espacio dentro de las oficinas, y en cierto momento deberán ser descartados. La motivación para la presente tesis es evitar la acumulación de reportes impresos de cada una de los trabajos de mantenimiento que se realizan en las instituciones, dando como solución una aplicación móvil que cualquier personal de la compañía podrá instalar en su dispositivo móvil y realizar así los reportes del mantenimiento de dichos equipos de una forma más fácil, permitiendo también la interacción con el supervisor o jefe inmediato del servicio. Asimismo, en la tesis, se combinan algoritmos de procesamiento digital de señales con algoritmos propietarios para el reconocimiento de código de barras mediante el uso de un dispositivo móvil con el sistema operativo Android®, que es uno de los sistemas operativos más populares en el mercado, comparables con iOS® de la empresa Apple® y Microsoft Mobile® de la empresa Microsoft®. Para la simulación, hemos realizado la captura de nueve (09) imágenes de código de barras tomadas en distintos ángulos y en distintas horas del día, con luz artificial o sin ella, en el exterior o bajo techo. Esto para que se pueda demostrar que nuestro sistema trabaja en cualquier ambiente posible de una empresa.

ABSTRACT

This work aims to provide a solution for the maintenance and control of equipment within different work environment as in industries, education, storage, among others. In our country, we are used to perform maintenance on equipment and keep a log of it in printed documents, which is tend to get lost or accumulating office space, which at some point should be discarded. The motivation for this thesis is to avoid the accumulation of forms for each of the maintenance work carried out in institutions, giving as a mobile application solution that any company personnel can install on his mobile device and perform the maintenance reports of such equipment in an easier way and allowing interaction with the supervisor in charge of the service. Further, in the thesis, combines algorithms for digital processing signal with proprietary algorithms to recognize barcodes using a mobile device with the Android® operating system, which is one of the most popular operating systems on the market comparable with iOS® from Apple® and Microsoft Mobile® from Microsoft®. For the simulation, we have made the capture of nine (09) barcode images taken at different angles and at different times of day, with artificial light or without it, outdoors or indoors. With this we can prove that our system works in any possible environment of a company.

INTRODUCCIÓN

El motivo del desarrollo de la presente tesis es facilitar, en una forma práctica, el trabajo que realiza un técnico en el momento de ejecutar el mantenimiento de un equipo, evitan así el uso de los formatos de mantenimiento que suelen ser llenados una vez finalizado dicha tarea.

Como alcance inicial de la presente tesis, se expone el proyecto a los sectores de mediana y pequeña empresa, haciendo posible el desarrolla de la misma en una investigación futura.

Este trabajo se divide en tres (03) partes fundamentales. La parte 1, concierne al diseño mediante algoritmos de procesamiento digital de imágenes y algoritmos propietarios del sistema de reconocimiento de imágenes de código de barras. La parte 2, al diseño de sistema de reconocimiento de imágenes de código de barras en el programa de prueba y error Matlab®. La parte 3, al diseño de una aplicación móvil para el reconocimiento de imágenes de código de barras mediante la plataforma de diseño Eclipse®. Por último la parte 4, a la simulación del modelo cliente - servidor para el almacenamiento de formatos de mantenimiento dentro del dispositivo móvil.

Como justificación económica, el diseño de esta tesis solo conlleva al gasto en relación con la mano de obra. Ya que esta aplicación se puede instalar en un dispositivo móvil según las especificaciones señaladas en la sección de metodología, los costos se reducirían con respecto a la adquisición de equipos lectores de código de barras (para mayor información revisar la sección anexos) en los que se redondean los $ 700 a 1200 (dólares americanos).

En el aspecto medio ambiental, el uso de esta aplicación evitaría la impresión innecesaria de formatos de mantenimiento, los cuales tienen a acumularse en almacenes y oficinas. Este es el aspecto en el cual se observa mayor influencia, ya que también se suprimes costos de manejo de residuos. Y en el aspecto psico-social, el hecho de que el técnico lleve menos objetos al área de trabajo significa reducción del estrés laboral, menos peso al momento de cargar lo cual evitaría problemas ergonómicos, entre otros.

Como objetivo general, se plantea diseñar una aplicación móvil en el sistema operativo Android® como herramienta de ayuda en áreas de soporte técnico haciendo uso de técnicas de procesamiento digital de imágenes para el reconocimiento de códigos de barras y el modelo cliente - servidor.

Dentro de los objetivos específicos, se consideran los siguientes: Diseñar el modelamiento matemático del sistema de reconocimiento de imágenes de códigos de barras. Diseñar de algoritmo propietario para la decodificación y reconocimiento de las líneas de la imagen de código de barras. Diseñar del sistema de reconocimiento de códigos de barras en el programa de prueba y error Matlab®. Diseñar del sistema de reconocimiento de código de barras en el sistema operativo Android® mediante el lenguaje de programación Java Script® en la plataforma de diseño Eclipse®. Por último, simular del modelo cliente - servidor en el sistema operativo Android® mediante el lenguaje de programación Java Script®.

CAPÍTULO I
MARCO TEÓRICO

1.1 Antecedentes

En la mayoría de empresas establecidas en nuestro país, se lleva el control de cada uno de los equipos con los cuales cuenta, esto en base a llevar un mejor manejo de sus activos. Mayormente, el control se realiza mensualmente de acuerdo con lo establecido en cada empresa; sin embargo, se pueden llevar a cabo en forma quincenal, semanal y diariamente, dependiendo del requerimiento de la gerencia.

Por otro lado, el mantenimiento de un equipo no necesariamente se realiza según lo planificado. Este puede realizarse sin previo aviso por un mal funcionamiento de uno de estos o por alguna emergencia en la cual esté involucrado el mismo. Existen cinco (05) tipos de mantenimiento que se realizan dentro de las empresas:

- **Mantenimiento correctivo:** Es el conjunto de tareas destinadas a corregir los defectos que se van presentando en los distintos equipos y que son comunicados al departamento de mantenimiento por los usuarios de los mismos.

- **Mantenimiento preventivo:** Es el mantenimiento que tiene por misión mantener un nivel de servicio determinado en los equipos, programando las intervenciones de sus puntos vulnerables en el momento más oportuno. Suele tener un carácter sistemático, es decir, se interviene aunque el equipo no haya dado ningún síntoma de tener un problema.

- **Mantenimiento predictivo:** Es el que persigue conocer e informar permanentemente del estado y operatividad de las instalaciones mediante el conocimiento de los valores de determinadas variables, representativas de tal estado y operatividad. Para aplicar este mantenimiento, es necesario identificar variables físicas (temperatura, vibración, consumo de energía, etc.) cuya variación sea indicativa de problemas que puedan estar apareciendo en el equipo. Es el tipo de mantenimiento tecnológico, pues requiere de medios técnicos avanzados, y en ocasiones, de fuertes conocimientos matemáticos, físicos y/o técnicos.

- **Mantenimiento cero horas (overhaul):** Es el conjunto de tareas cuyo objetivo es revisar los equipos a intervalos programados bien antes de que aparezca ningún fallo, bien cuando la fiabilidad del equipo ha disminuido, apreciablemente de manera que resulta arriesgado hacer previsiones sobre su capacidad productiva. Dicha revisión consiste en dejar el equipo a cero horas de funcionamiento, es decir, como si el equipo fuera nuevo. En estas revisiones, se sustituyen o se reparan todos los elementos sometidos a desgaste. Se pretende asegurar, con gran probabilidad un tiempo de buen funcionamiento fijado de antemano.

- **Mantenimiento en uso:** Es el mantenimiento básico de un equipo realizado por los usuarios del mismo. Consiste en una serie de tareas elementales (tomas de datos, inspecciones visuales, limpieza, lubricación, reapriete de tornillos) para las que no es necesario una gran formación, sino tal solo un entrenamiento breve. Este tipo de mantenimiento es la base del TPM (Total Productive Maintenance o Mantenimiento Productivo Total).

Para realizar el mantenimiento de un equipo, en nuestro país se viene utilizando lo que son los **formatos de mantenimiento**, que son documentos impresos los cuales llenan los técnicos cada vez que realizan dicho mantenimiento. En estos se deben llenar datos tales como fecha, nombre de la persona responsable del mantenimiento, ubicación, marca, modelo, años de fabricación, breve descripción del mantenimiento a realizar, entre otros. Algunos ejemplos de estos formatos se muestran, a continuación, en la figura (1).

Figura (1) Ejemplo de formato de mantenimiento de equipos.
Fuente: Save Solutions. Calle El Mirador y 85 Avenida Norte #706 Colonia Escalón, San Salvador.

Generalmente, estos formatos, una vez finalizado el mantenimiento del equipo, son entregados al supervisor o jefe inmediato quien lo revisa y los archiva como evidencia lo realizado. Estos son guardados en estantes o almacenes por si ocurriera alguna eventualidad con el equipo, ya que también son sustentos del trabajo en sí. En el periodo de un (01) año laboral se tiende a realizar varios mantenimientos, por lo cual se acumulan dichos documentos y esto conlleva a la reducción de espacios en las oficinas y almacenes. Es por

ello que después de haber finalizado el periodo de un (01) año laboral, se tienden a suprimir dichos formatos en forma gradual.

Se han visto casos en que los clientes presentan quejas y demandas debido a un mal servicio de mantenimiento que se ha realizado años atrás, sin embargo, las empresas ya no cuentan con los documentos sustentatorios de dicho mantenimiento. Esto en conclusión, da como resultado grandes pérdidas de sumas de dinero.

El objetivo de esta tesis es evitar la acumulación de papeles innecesarios dentro de una empresa, optimizando espacios así como recursos. Asimismo, facilitar la realización de los trabajos de mantenimiento en campo, por medio de una aplicación en un dispositivos móviles de bolsillo, evitando así cargar papeles, portafolios, tablillas, entre otros; lo cual solo produce incomodidad para el técnico que llevan a parte de estos sus materiales de trabajo.

La aplicación móvil deberá reconocer las imágenes de código de barras ubicadas en cada uno de los equipos de la empresa. Ya que el alcance de la presente tesis es para medianas y pequeñas empresas, se eligió para el desarrollo de la misma el código de barras European Article Number 8 ó EAN8, ya que la cantidad de equipos dentro del tipo de las empresas antes mencionadas es baja a comparación de una empresa grande. El código de barras se representa como se muestra en la figura (2).

Figura (2) Código de barras del tipo EAN8.

Elaboración: Los autores.

Los valores 123 representan el código de la empresa, los valores 4567 el código del equipo y, el valor 0 el código de control. Otros tipos de códigos de barras se podrán apreciar en la sección anexos. Una vez realizado el proceso de reconocimiento de código de barras, la aplicación tomará el valor de dicho código y lo comparará con los valores ingresados en una base de datos implementada dentro del mismo dispositivo móvil. Al encontrar el valor obtenido, este lo enlazará a la descripción del equipo en el cual se realizará el mantenimiento. Una vez el mantenimiento, se describirá, en una sección de la aplicación, el proceso realizado que será enviado, vía correo electrónico y guardado en la base de datos. En esta sección, se mostrarán los mantenimientos realizados anteriormente al equipo.

1.2 Bases teóricas

El estado del arte mundial, en lo que respecta a detección de códigos de barras está orientado a la detección de bordes mediante la primera derivada de acentuación - Gradiente debido a que resulta más sencillo tomar los planos verticales de los gráficos. Asimismo, está orientado a la segmentación basada a la morfología matemática, transformada Wavelet y redes neuronales. A continuación, veamos los más destacados de estos.

Vicente Castelló Martínez en julio del 2005 presenta el proyecto informático (E80) titulado “Localización de Códigos de Barras en Imágenes Digitales” de la Universitat Jaume I. El proyecto describe un método para la localización y posterior decodificación de códigos de barras presentes en imágenes digitales. El código de barras puede aparecer en cualquier posición y orientación dentro de la imagen, con lo cual debe ser capaz de detectarlo y rectificar su orientación, para así poder decodificarlo de forma correcta.

Una técnica que se presenta, debido a que los códigos de barras poseen unas características muy definidas, como un método adecuado para la

localización de un código de barras dentro de una imagen es la detección de bordes. Además, como los puntos de borde pertenecientes al código presentarán una orientación similar, el objetivo es de alguna forma homogeneizar esta zona donde se encuentra el código, extendiendo el valor de la orientación obtenido de los puntos de borde de los que no lo son. Debido a este proceso de expansión de la orientación, se logrará un conjunto de regiones candidatas, de las cuales deben ser capaces de poder elegir una de ellas como susceptible de contener el código. A partir de esta región, se deberá calcular la orientación que presentan los puntos de borde de la misma, para poder rectificar así la imagen. Una vez conseguida una sub-imagen centrada en el código se intenta decodificar este con éxito, basándose en lo que se conoce del tipo de codificación en concreto.

- **Detección de bordes:** Utiliza el detector de bordes de Canny, debido a las características que posee el código de barras se presenta como una técnica más adecuada para su posible detección, y así se aplica un filtro gaussiano previo, para suavizar la imagen y tratar de conseguir la eliminación del posible ruido.

- **Detección de regiones candidatas:** Se centran en la idea de que a la hora de detectar el código de barras los pixeles que son puntos de borde y forman parte del código van a tener una orientación similar. Con esta característica, lo que se pretende es expandir esta orientación a los vecinos de cada uno de los puntos de borde y así, homogeneizar toda la zona donde se encuentra el código con una misma orientación.

- **Etiquetado y filtrado de regiones:** Explican cómo tras el paso anterior mediante un proceso de agrupamiento, extendiendo la orientación y gracias a una posterior binarización, consiguen obtener una serie de regiones susceptibles de ser la que se busca, utilizar algoritmos de etiquetado de

componentes conexas y realizar luego un filtrado, que debe quedar con la región que contiene el código.

- **Estimación de la orientación:** Indican cómo al poder presentar el código cualquier ángulo dentro de la imagen, se debe calcular la orientación que presentan los puntos de borde que aparecen en esta región susceptible de contener al código y que hemos obtenido en el paso anterior. Una vez conocida esta orientación simplemente se debe rectificar la imagen y conseguir una imagen centrada en el código, para su posterior binarización. De todos los puntos de borde que se encuentran en la región que se ha seleccionado, que es donde se encuentra el código, se queda con el valor de la orientación de cada uno de ellos y se calcula la mediana para conseguir el valor más representativo de la orientación en esa zona. Se calcula la mediana en vez de la media, ya que esta primera resulta ser más robusta, con el resultado se realiza la rotación.

- **Binarización:** Para la binarización, se realiza una ecualización del histograma, esto permite conseguir un mayor contraste entre las barras negras y los espacios. Para la obtención del umbral, a partir del cual realizar la binarización, existen diversos métodos automáticos de búsqueda, como el algoritmo de Otsu que es el empleado en este caso.

- **Decodificación:** Explican cómo basándose en la imagen binarizada del código de barras obtenida en el paso anterior, y empleando el conocimiento que tienen del tipo de código a interpretar (EAN-13), se obtiene la consiguiente decodificación del mismo.

Ramtin Shams y Parastoo Sadeghi ambos del Research School of Information Sciences and Engineering (RSISE), The Australian National University, Camberra ACT 0200 Australia, en el año 2007, publican su

investigación titulada “Bar Code Recognition In Highly Distorted and Low Resolution Images”

Desarrollaron un algoritmo propio, particularmente diseñado para el reconocimiento de códigos de barras donde la imagen puede tener baja resolución, baja calidad o borroso, estar desenfocado, con iluminación no uniforme, ruido o saturación de color. La metodología utilizada, con énfasis en los componentes escáner y decodificación es la siguiente:

- **Pre-procesamiento:** Se convierte la imagen ingresada a un mapa de escala de grises con un algoritmo propio, la intensidad de la escala de grises en la imagen es escalada entre dos valores 0 y 1, donde 0 denota un pixel blanco y 1 denota un pixel negro. Prefieren esta representación de píxeles en blanco y negro, como se asociará barras con picos y espacios con valles en etapas posteriores del algoritmo. Las imágenes grandes (> 2 MP) se muestrean para reducir el costo de procesamiento, que a menudo, toma mucho tiempo el detector de la región de interés (Region of Interest (ROI)). Una vez que los ROI de se detectan, la imagen tamaño original es utilizado por el segmentador.

- **Detector ROI:** Un detector de ROI se utiliza para encontrar áreas dentro de la imagen de entrada, donde hay una posibilidad de que un código de barras puede ser encontrado. La naturaleza direccional de los códigos de barras de identificación se utiliza para este propósito mediante el cálculo de un mapa de gradiente de la imagen de entrada y la selección de áreas, donde la gradiente de fase exhibe algunas características direccionales. La magnitud y fase de la gradiente de la imagen es calculada usando máscara de Sobel 3x3.

- **Segmentación:** El segmentador detecta límites de los códigos de barras y ajusta la orientación, esta se calcula por el detector de ROI. El segmentador

gira la imagen de tal manera que los módulos de código de barras son paralelos al eje vertical. Esto simplifica el proceso de exploración. Con el fin de encontrar los límites, que la búsqueda de zonas de silencio horizontales (grandes áreas de fondo en torno a módulos de código de barras). En esta etapa, la densidad de código es desconocido y que asuma la más pequeña densidad de código posible que puede ser detectado por el algoritmo (1 pixel) con el fin de garantizar los resultados más precisos. Esto dará lugar a la detección de zonas de silencio no válidos dentro de la zona de un código de barras para códigos de barras más grandes. No se puede descartar rápidamente estas abrasiones contando el número de extremos en la fase de exploración.

Debido al ruido y las distorsiones de iluminación, la zona puede presentar variaciones en la secuencia y que se asemejan a los módulos estrechos de un código de barras de alta densidad.

- **Resultados:** Probaron su algoritmo contra una base de datos de imágenes EAN13 de códigos de barras, tomados con un teléfono celular NEC 616. El teléfono tiene una resolución baja 352x288 pixeles. Las imágenes fueron tomadas bajo varias condiciones tonos de luz, orientación, distancia y perspectiva. Su algoritmo fue capaz de decodificar el 47% de las pruebas. Los resultados mostraron un 57% de improbabilidad contra un lector de código de barras comercial, el cual fue capaz de decodificar menos del 30% de las imágenes.

Harsh Kapadia y Alpesh Pastel, ambos profesores asistentes del Dept. of Electrical Engg., Institute of Technology, Nirma University, Ahmedabad, Gujarat, India. El 6 de junio del 2013 publicaron su investigación en el International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, cuyo título es “Application of Hough Transform and Sub-Pixel Edge Detection in 1-D Barcode Scanning”. En este trabajo, se hace énfasis en dos (02) algoritmos: la transformada de Hough y detección de bordes por sub-pixel y

la aplicación de ambos en la lectura de códigos de barras 1D. El sistema está destinado a verificar los códigos de barra on-line. Se enfoca principalmente en dos aspectos:

- Detectar el ángulo si el código de barras está inclinado en la imagen y corregirlo.
- Detectar bordes del código de barras en tiempo real de una imagen borrosa utilizando el detector de bordes por sub-pixel.

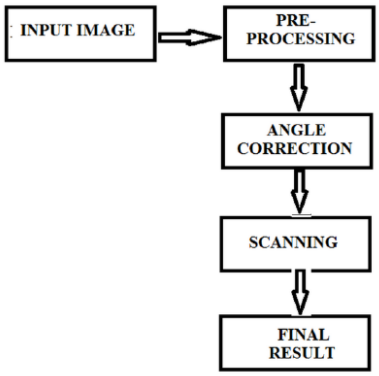


Figura (3) Diagrama de bloques del sistema de reconocimiento de código de barras.

Fuente: Kapadia, H., y Pastel, A. (2013). Application of Hough Transform and Sub-Pixel Edge Detection in 1-D Barcode Scanning. International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering.

En el pre-proceso, la imagen necesita ser convertida a escala de grises. Así la imagen original debe ser mejorada para una lectura precisa. Para conseguir 95 bits, se extrae de la imagen una sola fila de código de barras. Y luego otra parte que el código de barras se retira. En cualquier imagen que contenga código de barras, puede haber una posibilidad de que esté inclinado. El ángulo en el que el código de barras está inclinado se debe encontrar y corregir, esto se hace utilizando la transformada de Hough; La transformada de Hough patentado por Paul Hough en 1962, es básicamente, un método de extracción de características, se emplea para detectar las líneas y encontrar formas arbitrarias de posición en la imagen y se utiliza en el campo de la visión por

computador y procesamiento de imágenes. Como el código de barras contiene líneas rectas que se pueden detectar, entonces los ángulos de las líneas correspondientes, pueden ser encontrados.

En el escaneo se plantea un problema cuando se ve borrosa la imagen de código de barras, una limitación más es que el espacio para el código de barras en la superficie del producto es pequeño. Por lo tanto, se obtendrá una imagen pequeña. Si el código de barras se genera por cualquier software de generación de código de barras, será una imagen ideal, esta imagen no representa problema, Pero la imagen real es en la mayoría de casos borrosa. Así que con el fin de escanear esa imagen con precisión, deben llevarse a cabo algunas operaciones. Una solución puede ser el aumento de la resolución de la imagen utilizando el método de super-resolución, otro detectar bordes de códigos de barras a nivel de sub-píxeles con el método de detección de bordes sub-pixel.

Algunos contratiempos ocurrieron durante el desarrollo, como si el código de barras está rotado en un ángulo de más de 180, es decir, si está en tercero o segundo cuadrante entonces la detección del ángulo mediante transformada de Hough es difícil. Otra importante limitación es que si el tamaño de la imagen se cambia, entonces el software se enfrenta a problemas para escanear el código de barras.

El problema con el reconocimiento en tiempo real de imágenes borrosas está totalmente resuelto. Ahora puede escanear imágenes con precisión tanto ideales y en tiempo real. Algunos problemas se produjeron con la parte de corrección del ángulo como el software no fue capaz de corregir los ángulos como 5, 17, 29, etc. y que estaba corrigiendo con precisión ángulos como 10, 20, 30, 40, 50 etc., pero realizaron modificaciones en el código y las funciones de modo que pueda resolver el problema.

Un problema con interfaz gráfica de usuario es que la cámara encaje imagen captada no puede ser reconocido como cámara portátil no es la cámara para alta resolución. Una cosa más es que la celebración del código de barras que contiene objetos en la mano se enfrenta a los problemas de vibración de modo de imagen no es estable. Estas imágenes no son escaneados. Incluyeron cámara en GUI solo para fines de demostración. También crearon funciones como imagen del proceso antes de la digitalización, las imágenes borrosas, etc., para la mayor parte del software es que no se necesita mucho tiempo para ejecutar y dar resultado. Así que esta lógica utilizada en el código se puede usar para desarrollar un sistema de tiempo real en la industria. Otras modificaciones en el código GUI se pueden realizar de acuerdo con la necesidad del usuario.

Oktem R. de la Atilim Univ., Ankara, Turquía, en el año 2004 publica en "Signal Processing and Communications Applications Conference, 2004. Proceedings of the IEEE 12th" el tema "Barcode localization in Wavelet domain by using binary morphology", en la cual propone un algoritmo basado en la transformada Wavelet para la localización de código de barras en una imagen. El algoritmo está diseñado para permitir la aplicación rápida y eficiente, así como también para sistemas de reconocimiento de código de barras automáticos que se utilizan en las unidades de fabricación y distribución grandes. La región de las áreas de interés se mantienen y otras regiones se descartan mediante la aplicación de la morfología binaria en coeficientes de la transformada Wavelet.

Dong Hang, Jianfu Teng, Zhaoxuan Yang, Yanwei Pang y Meng Wang exponen en "Computer Application and System Modeling (ICCASM)", en el 2010 el tema "2D barcode image binarization based on Wavelet analysis and Otsu's method" donde especifican que es extremadamente difícil binarizar la imagen de código de barras 2D bajo la condición de tanto la iluminación no uniforme y fondo desordenado. Ni el método global ni local de umbralización

puede producir un resultado binario ideal. En este trabajo, se propone un método de binarización de la imagen de código de barras 2D. La contribución del método es que se puede eliminar la iluminación no normalizada especialmente en el fondo desordenado. Esto se logra mediante la estimación de la distribución de la iluminación por el análisis Wavelet y seguido por sacarlo de la imagen de origen de acuerdo con el modelo de formación de la imagen. El método propuesto puede binarizar las imágenes de códigos de barras difíciles con el algoritmo simple de Otsu. Los resultados experimentales reflejan la robustez de este método. Además, puesto que la aplicación de filtrado de transformada Wavelet se adapte a la arquitectura DSP, el método funciona bien en nuestra terminal de reconocimiento de código de barras 2D basado en DSP para la aplicación real.

Shu-Jen Liu del Int. of Inf. Sci. en Acad. Sinica, Taipei, Taiwan, en 1993 publica en “Neural Networks, 1993. IJCNN '93-Nagoya. Proceedings of 1993 International Joint Conference” el proyecto “Camera-based bar code recognition system using neural network” donde propone un sistema de reconocimiento de código de barras usando redes neuronales. Es bien sabido que, en muchas tiendas, se aprobó el lector de código de barras láser en los mostradores de check-out. Sin embargo, existe una limitación importante cuando se utiliza esta herramienta. Es decir, a diferencia del picturing tradicional basado en cámaras, la distancia entre el lector láser (sensor) y el objeto de destino es cercana a cero cuando se aplica el lector.

Esto puede dar lugar a molestias en la automatización de tienda porque operador humano tiene que hacerse cargo de cualquiera de los sensores o los objetos (o ambos). Para el propósito de la automatización en tiendas, el operador humano tiene que ser eliminado del proceso, es decir, un robot con capacidad visual requiere para jugar un papel importante en dicho sistema. En este trabajo, se propone un sistema de reconocimiento de código de barras basado en cámaras usando redes neuronales backpropagation.

El objetivo final de este enfoque consiste en utilizar la cámara en lugar de lector láser de manera que la automatización de tienda se puede lograr. Hay un número de pasos que intervienen en el sistema propuesto. El primer paso del sistema tiene que realizar es la de localizar la posición y orientación del código de barras en la imagen adquirida. En segundo lugar, el sistema propuesto tiene el segmento de código de barras. Por último, se utiliza una red neuronal back-propagation capacitado para llevar a cabo tareas de reconocimiento de código de barras.

CAPÍTULO II
METODOLOGÍA

2.1 Materiales

Para la realización de nuestra tesis se requería como mínimo equipos que se enlistan a continuación:

- **Un (01) dispositivo móvil**, que tenga como mínimo una (01) cámara fotográfica posterior de 5MP (2560 x 1920 pixeles) con auto-focus, y sistema operativo Android® con la versión 4.0 - Ice Cream Sandwich (Otras versiones del sistema operativo se aprecian en la sección anexos).
- **Un (01) programa Matlab®**, versión R2013a (8.1.0.604) de 64-bit (win64).
- **Un (01) programa Eclipse®**, Id for Java Developer versión Luna Service released 1 (4.4.1).
- **Un (01) aplicación Open CV Library**, versión 2.4.9.0 rev. 1 para dispositivos móviles.

2.2 Métodos

2.2.1 Modelamiento matemático de la detección de bordes de la imagen mediante el operador de Canny

2.2.1.1 Operador de Canny

En 1986, Canny propuso un método para la detección de bordes que ofrecía mejores resultados que los métodos vistos anteriormente aunque presentaba una mayor complejidad computacional. El método de Canny se basa en tres criterios principales:

- El criterio de detección, que expresa el hecho de evitar la eliminación de bordes importantes así como no suministrar falsos bordes.
- El criterio de localización, que establece que la distancia entre la posición real y la posición localizada para el borde debe ser minimizada.
- El criterio de respuesta única, que establece la necesidad de que el detector retorne un único punto por cada punto de borde verdadero. Esto implica que el detector no debe encontrar múltiples píxeles de borde donde solo existe uno.

Uno de los métodos relacionados con la detección de bordes es el uso de la primera derivada, que utiliza el valor cero en todas las regiones donde no varía la intensidad y tiene un valor constante en toda la transición de intensidad. Por lo tanto, un cambio de intensidad se manifiesta como un cambio brusco en la primera derivada, característica que es utilizada para detectar un borde, y en la que se basa el algoritmo de Canny. El algoritmo de Canny consta de tres grandes pasos:

a) **Obtención del gradiente:** Para la obtención del gradiente, lo primero que se realiza es la aplicación de un filtro gaussiano a la imagen original con el

objetivo de suavizar dicha imagen y conseguir la eliminación del ruido que pueda existir. Sin embargo, hay que tener cuidado de no aplicar un suavizado excesivo, puesto que se podrían perder ciertos detalles de la imagen y provocar que el resultado final no fuese el esperado. Este suavizado se obtiene promediando los valores de intensidad de los píxeles del entorno de vecindad con una máscara de convolución.

$$\frac{1}{273}\begin{bmatrix}1 & 4 & 7 & 4 & 1\\ 4 & 16 & 26 & 16 & 4\\ 7 & 26 & 41 & 26 & 7\\ 4 & 16 & 26 & 16 & 4\\ 1 & 4 & 7 & 4 & 1\end{bmatrix} \qquad \frac{1}{273}\begin{bmatrix}2 & 4 & 5 & 4 & 1\\ 4 & 9 & 12 & 9 & 4\\ 5 & 12 & 15 & 12 & 5\\ 4 & 9 & 12 & 9 & 4\\ 2 & 4 & 5 & 4 & 1\end{bmatrix}$$

Una vez que se suaviza la imagen, para cada píxel se obtiene la magnitud y módulo (orientación) del gradiente, obteniendo así dos imágenes.

b) Supresión no máxima al resultado del gradiente: Las dos imágenes generadas en el paso anterior sirven de entrada para generar una imagen con los bordes adelgazados. El procedimiento es el siguiente: se consideran cuatro direcciones identificadas por las orientaciones de 0º, 45º, 90º y 135º con respecto al eje horizontal. Para cada píxel se encuentra la dirección que mejor se aproxime a la dirección del ángulo de gradiente.

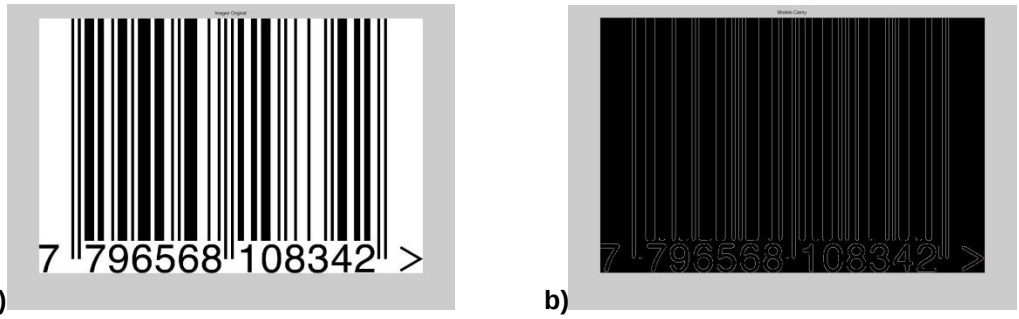
Posteriormente, se observa si el valor de la magnitud de gradiente es más pequeño que al menos uno de sus dos vecinos en la dirección del ángulo obtenida en el paso anterior. De ser así, se asigna el valor 0 a dicho píxel, en caso contrario se asigna el valor que tenga la magnitud del gradiente.

La salida de este segundo paso es una imagen con los bordes adelgazados después de realizarse la supresión no máxima de puntos de borde.

c) **Histéresis de umbral a la supresión no máxima:** La imagen que ha sido obtenida, en el paso anterior, suele contener máximos locales creados por el ruido. Una solución para eliminar dicho ruido es la denominada histéresis del umbral.

El proceso de histéresis consiste en tomar la imagen obtenida en el paso anterior, obtener la orientación de los puntos de borde de la imagen y tomar dos umbrales de forma que el primero sea más pequeño que el segundo. Para cada punto de la imagen se debe localizar el siguiente punto de borde no explorado que sea mayor que el segundo valor de umbral. A partir de dicho punto, se siguen las cadenas de máximos locales conectados en ambas direcciones perpendiculares a la normal del borde siempre que sean mayores que el primer valor de umbral. De esta forma, se marcan todos los puntos explorados y se almacena la lista de todos los puntos en el contorno conectado. Es de este modo cómo se logra eliminar, con este paso, las uniones en forma de Y de los segmentos que confluyen en un determinado punto.

A continuación, se muestra el resultado del uso del operador Canny en una imagen ejemplo de código de barras en la figura (4).



a) **b)**

Figura (4) Detección de bordes de imagen de código de barras mediante el operador de Canny. (4.a) Imagen original. (4.b) Aplicación de operador de Canny. Elaboración: Los autores.

Como conclusión, se puede indicar que el algoritmo de Canny tiene como principal ventaja su gran adaptabilidad para poder ser aplicado a diversos tipos de imágenes, además de no disminuir su rendimiento ante la presencia de ruido en la imagen original. Sin embargo, tiene como desventaja el hecho de que al realizar el suavizado de la imagen se pueden difuminar ciertos bordes, aunque con eso se consiga reducir el ruido.

Este algoritmo es uno de los mejores métodos para la detección de bordes, que aplica métodos de diferencias finitas basado en la primera derivada y cuya popularidad se debe, además de a sus buenos resultados, a su sencillez, que permite una gran velocidad de procesamiento al ser implementado.

Es por este motivo que se seleccionó dicho operador para el desarrollo de este proyecto.

2.2.1.2 Operadores basados en la primera derivada - Gradiente

Las técnicas clásicas de detección de bordes se basan en diferenciar la imagen, esto es, encontrar la derivada respecto de los ejes **x** e **y** o gradiente.

El gradiente de una imagen, en un punto, indica la variación máxima de la función en ese punto. Se define como:

$$G[f(x,y)] = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} G_F \\ G_C \end{bmatrix} = \begin{bmatrix} \frac{\sigma}{\sigma_x} f(x,y) \\ \frac{\sigma}{\sigma_y} f(x,y) \end{bmatrix}$$

El proceso para determinar si un punto es borde o no lo da si el valor del módulo del gradiente supera o no un valor de umbral dado, calculando este módulo como $|G| = \sqrt{(G^2_x + G^2_y)}$. En general el gradiente se suele aproximar mediante la expresión $|G| \approx |G_x| + |G_y|$, que es mucho más simple de implementar en la práctica. Se le conoce como fuerza de borde o contraste de borde que es la diferencia mínima en escala de grises entre el fondo y el borde. Este parámetro puede variar por condiciones de iluminación o por diferencias en las características de la escala de grises.

A continuación, se muestra las derivadas primera y segunda del perfil de nivel de gris de la imagen superior.

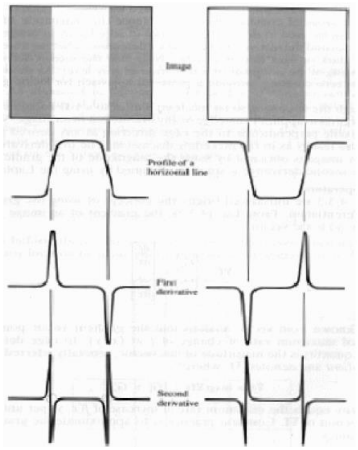


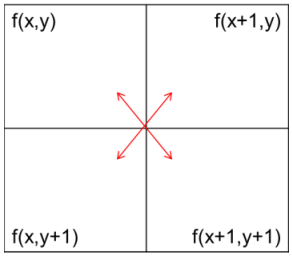
Figura (5) Detección de bordes empleando operadores de derivación. La segunda derivada tiene un cruce por cero en la posición de cada borde.
Fuente: Gonzales, R., y Woods, R. (2007). Digital image processing, third edition. New Jersey, Estados Unidos: Prentice Hall.

Como se puede observar en la figura (5), la primera derivada es positiva para cambio a nivel de gris más claro, negativa en caso contrario y cero en aquellas zonas con nivel de gris uniforme. La segunda derivada presenta valor positivo en la zona oscura de cada borde, valor negativo en la zona clara de cada borde y valor cero en las zonas de valor de gris constante y justo en la posición de los bordes. El valor de la magnitud de la primera derivada nos sirve para detectar la presencia de bordes, mientras que el signo de la segunda derivada nos indica si el pixel pertenece a la zona clara o a la zona oscura. Además, la segunda derivada presenta siempre un cruce por cero en el punto medio de la transición. Esto es muy útil para localizar bordes en una imagen. Aunque hasta el momento se refiere a perfiles unidimensionales, la extensión a dos dimensiones es inmediata. Simplemente, se define el perfil en la dirección perpendicular a la dirección del borde y la interpretación anterior seguiría siendo válida. La primera derivada en cualquier punto de la imagen vendrá dada por la magnitud del gradiente, mientras que la segunda derivada vendrá dada por el operador Laplaciano.

Tipos de detectores

Operador de Roberts

El operador cruzado de Roberts proporciona una aproximación al gradiente usando la técnica de diferencia mostrada, anteriormente, para el caso unidimensional, pero con dos píxeles de diferencia en los ángulos derechos de cada lado como se muestra en la figura (6).



$$D_1 = f(x+1,y) - f(x,y+1)$$
$$D_2 = f(x+1,y+1) - f(x,y)$$
$$D_1^2 + D_2^2 = |D_1| + |D_2|$$

Figura (6) Cruzado de Roberts.

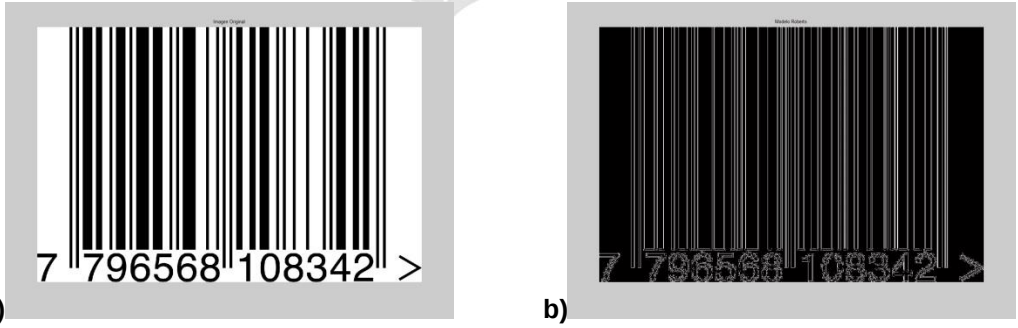
Fuente: Gonzales, R., y Woods, R. (2007). Digital image processing, third edition. New Jersey, Estados Unidos: Prentice Hall.

Presenta como gran desventaja el que muy pocos píxeles de entrada se consideren para hacer la aproximación, lo que provoca que sea muy sensible al ruido y nos permita solamente marcar los puntos de borde, es decir, su localización, pero no la orientación de los mismos. No obstante, esta misma desventaja lo convierte en un operador muy simple que trabaja muy bien con imágenes binarias y con una gran velocidad de cómputo.

Las máscaras que utiliza este operador son:

$$A1 = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \quad A2 = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \quad G_F = z_9 - z_5 \quad G_C = z_8 - z_6$$

A continuación, se muestra el resultado del uso del operador de Roberts en una imagen ejemplo de código de barras en la figura (7).



a) **b)**

Figura (7) Detección de bordes de imagen de código de barras mediante el operador de Roberts. (7.a) Imagen original. (7.b) Aplicación del operador de Roberts.

Elaboración: Los autores.

Ventajas

- Buena respuesta en bordes horizontales y verticales.
- Buena localización.
- Simpleza y rapidez de cálculo.

Desventajas

- Mala respuesta en bordes diagonales.
- Sensible al ruido.
- Empleo de máscaras pequeñas.
- No da información acerca de la orientación del borde.
- Anchura del borde de varios píxeles.

Operador de Sobel

Los operadores de gradiente, en general, tienen el efecto de magnificar el ruido subyacente en la imagen, no obstante, el detector de Sobel se puede ver como la combinación de un filtro de suavizado del ruido con un operador de aproximación imprecisa del gradiente.

Como ya se ha mencionado, utiliza una máscara de 3x3, esta máscara se mueve píxel a píxel, calculando el valor del gradiente para cada uno de ellos (el píxel central de la máscara), aplicando las fórmulas anteriormente mencionadas. Como ya se ha explicado anteriormente, una vez obtenido el valor del gradiente se decide si es un borde o no en función de un umbral prefijado. Este proceso es común a todos los operadores.

Máscaras de Sobel:

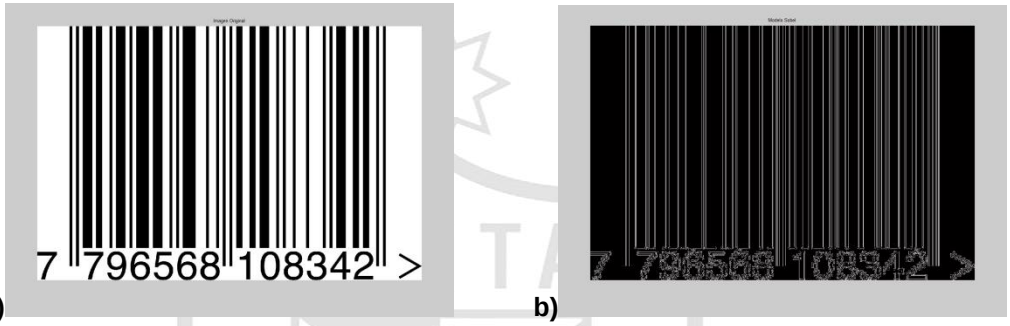
$$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$$G_F = z_7 + 2z_8 + z_9 - z_1 - 2z_2 + z_3$$

$$G_C = z_3 + 2z_6 + z_9 - z_1 - 2z_4 + z_7$$

Como se puede observar en el valor de sus máscaras, Sobel enfatiza el valor de los píxeles cercanos al centro, al proporcionarles un coeficiente de 2. Sobel es el operador usado comúnmente y en la práctica, proporciona una buena detección de bordes diagonales.

A continuación, se muestra el resultado del uso del operador de Sobel en una imagen ejemplo de código de barras en la figura (8).



a) **b)**

Figura (8) Detección de bordes de imagen de código de barras mediante el operador de Sobel. (8.a) Imagen original. (8.b) Aplicación del operador de Sobel. Elaboración: Los autores.

Ventajas

- Buena respuesta en bordes horizontales y verticales.
- Diversidad de tamaños en las máscaras.
- Proporcionan un suavizado además del efecto de derivación.

Desventajas

- Mala respuesta en bordes diagonales.
- Lentitud de cálculo.
- No da información acerca de la orientación del borde.
- Anchura del borde de varios píxeles.

Operador de Prewitt

Es un operador similar al de Sobel, pero se diferencia en los coeficientes, ya que Prewitt no enfatizan a los píxeles cercanos al centro de la máscara, como se puede observar en la misma:

En el caso de Prewitt (K=1), se pondera la información de filas y columnas adyacentes para dar mayor inmunidad al ruido.

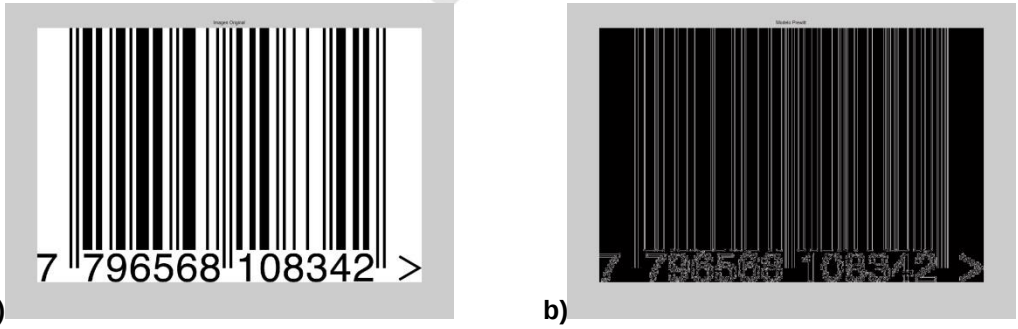
$$\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

$$G_F = z_7 + z_8 + z_9 - z_1 - z_2 + z_3$$

$$G_C = z_3 + z_6 + z_9 - z_1 - z_4 + z_7$$

A diferencia del operador de Sobel, el operador de Prewitt proporciona una mejor detección de los bordes verticales y horizontales en comparación con los bordes diagonales. No obstante, en la práctica no se aprecia una gran diferencia entre ambos.

A continuación, se muestra el resultado del uso del operador de Prewitt en una imagen ejemplo de código de barras en la figura (9).



a) **b)**

Figura (9) Detección de bordes de imagen de código de barras mediante el operador de Prewitt. (9.a) Imagen original. (9.b) Aplicación del operador de Prewitt.

Elaboración: Los autores.

Estos tres operadores revisados hasta ahora: operador de Roberts, operador de Sobel y operador de Prewitt, presentan una característica común que los diferencia de los demás y estriba en que los valores obtenidos por los tres primeros son reales, mientras que los otros son discretos.

Ventajas

- Buena respuesta en bordes horizontales y verticales.
- Poco sensible al ruido.
- Proporciona la magnitud y dirección del borde.

Desventajas

- Mala respuesta en bordes diagonales.
- Lentitud de cálculo.
- Anchura del borde de varios píxeles.

Desventajas genéricas de la detección de bordes basada en la gradiente

Los detectores de bordes hasta ahora comentados tienen en cuenta únicamente la primera derivada, comparándola con un umbral para determinar si existía o no un punto de borde. Esta aproximación presenta problemas de ruido, falsos negativos y falsos positivos, por ejemplo en forma de puntos sueltos, que no se corresponden con ningún verdadero borde.

2.2.1.3 Operadores basados en la segunda derivada - Laplaciana

Existen métodos de detección de bordes que se basan en el cálculo de las derivadas de segundo orden sobre el valor de los píxeles de una imagen. Los operadores que utilizan estos métodos no son dependientes de la dirección de los bordes, al contrario de los operadores de la primera derivada. En ciertas aplicaciones, se recomienda utilizar un operador que sea totalmente independiente de la orientación, ya que puede ofrecer evidentes ventajas computacionales.

Los detectores de bordes que solo consideran la primera derivada dependen de un valor umbral para determinar la existencia de un punto de borde. Para realizar una aproximación más efectiva, únicamente se seleccionan como bordes aquellos puntos en los que se encuentra un máximo en el gradiente de intensidad de la imagen. Esto quiere decir que en los puntos de un borde, habrá un máximo en la primera derivada y un cruce por cero en la segunda derivada. De esta forma, se facilita una localización más precisa de los puntos de borde.

Uno de los operadores más conocidos que utilizan la segunda derivada es el operador de Laplace o Laplaciano.

Operador Laplaciano

El operador Laplaciano es la suma de las segundas derivadas en ambas direcciones y tiene la característica de ser un filtro isotrópico, lo que significa que es invariante a la rotación. Esto quiere decir que si el Laplaciano se aplica a una imagen y esta es girada, se obtendrá el mismo resultado que si la imagen es girada primero y posteriormente, se le aplica el operador Laplaciano.

El Laplaciano de una función **f(x,y)** se define como:

$$\Delta f = \Delta^2 f = \frac{\sigma^2 f}{\sigma x^2} + \frac{\sigma^2 f}{\sigma y^2}$$

El Laplaciano vale cero si **f(x,y)** es constante o cambia de forma lineal su amplitud. El cambio de signo de la función resultante nos indica que en ese lugar existe un cruce por cero, ya sea un cambio de positivo a negativo o viceversa, y, por lo tanto, hace notar la presencia de un borde. Hay que decir también que, a diferencia del gradiente, el Laplaciano no es un vector.

La principal característica de este operador es la capacidad de localizar los bordes de una forma bastante precisa a través de la determinación del cruce por cero. Se fundamenta en que cuando la imagen presenta un cambio de intensidades bastante relevante a lo largo de una determinada dirección, existirá un máximo en la primera derivada a lo largo de dicha dirección y un paso por cero en la segunda derivada. Al considerar imágenes discretas, es muy poco probable que el paso por cero coincida justamente en un píxel de la imagen. Por este motivo, el borde se suele marcar en aquel píxel que contenga un nivel de gris más próximo a cero, teniendo como vecino, al menos, un píxel con nivel de gris de signo contrario.

Otras características importantes del operador Laplaciano son:

- Realiza una buena localización de los bordes siempre que las aristas se encuentren bien separadas y la relación señal-ruido de la imagen sea alta.
- Es totalmente independiente de la orientación del borde por lo que ofrece una buena respuesta en la detección de bordes horizontales, verticales y diagonales.

Como principales inconvenientes, hay que decir que el operador Laplaciano presenta una gran sensibilidad al ruido por tratarse de un operador que utiliza la segunda derivada. Por este motivo, en muchos casos en lugar de aplicarlo directamente, se realiza primero un suavizado de la imagen que se combina, posteriormente, con la aplicación del operador. Además, con este operador pueden darse casos de detección de bordes dobles debido a que sus anchuras no siempre son óptimas.

Por otra parte, también es reseñable que su fiabilidad no es demasiado alta, pudiendo aparecer identificados falsos bordes.

Existen distintas máscaras Laplacianas que representan diferentes aproximaciones del operador Laplaciano. A diferencia de las máscaras utilizadas por los operadores de primera derivada, estas son simétricas rotacionalmente, lo que significa que pueden detectar bordes en todas las direcciones del espacio. Se aplican seleccionando primero la máscara y realizando una operación de convolución sobre la imagen. El signo del resultado de dos píxeles adyacentes proporciona información direccional y nos dice que lado del borde es más o menos oscuro.

Normalmente, estos operadores no son utilizados directamente en el ámbito de la visión artificial, debido a que cualquier operador que involucre a más de una derivada se verá más afectado por el ruido que un operador que utilice una sola derivada. Para evitar el efecto del ruido, será necesario aplicar primero filtros sobre la imagen de entrada.

Operador LoG (Laplaciano de Gaussiano)

El operador Laplaciano del Gaussiano consiste en aplicar el operador Laplaciano a la imagen una vez que ha sido suavizada con un filtro gaussiano. Como ya se ha comentado en el apartado anterior, el operador Laplaciano por sí mismo es bastante sensible al ruido, por lo que es prácticamente imprescindible utilizarlo de forma combinada con un operador de suavizado, en este caso, el filtro gaussiano.

El primer paso que realiza el operador es suavizar la imagen y reducir su ruido convolucionándola con un filtro Gaussiano. Este paso produce un efecto secundario: los bordes se difuminan, por lo que el operador Laplaciano solo considerará como bordes los píxeles que formen un máximo local, gracias a la segunda derivada, pero únicamente considerando aquellos puntos en los que la primera derivada esté por encima de un umbral, para evitar que se detecten bordes insignificantes.

El operador LoG presenta una simetría radial. Tiene una forma muy especial que se asemeja a la de un sombrero mejicano como se aprecia en la figura (10). Por esta razón, este operador es conocido como el “filtro del sombrero mexicano”.

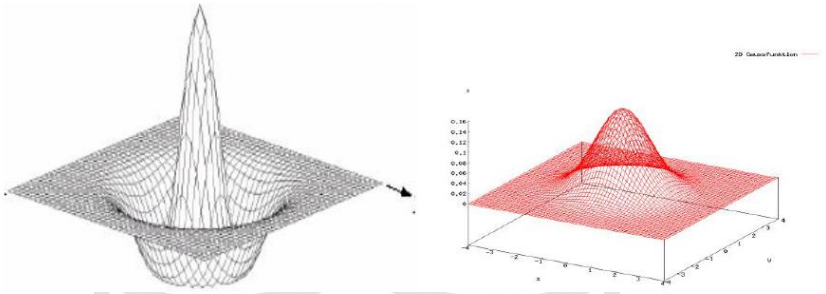


Figura (10) Operador LoG o de sombrero mexicano.
Fuente: Gonzales, R., y Woods, R. (2007). Digital image processing, third edition. New Jersey, Estados Unidos: Prentice Hall.

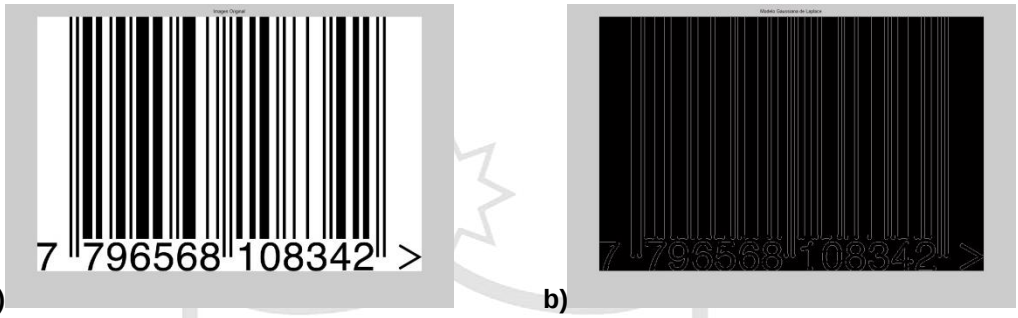
Este método de detección de ejes fue propuesto por primera vez por Marr and Hildreth. Estos autores fueron los que introdujeron el principio de detecciones mediante el método de cruces por cero. El principio en que está basado este método consiste en encontrar las posiciones de una imagen donde la segunda derivada toma el valor cero.

El operador LoG calcula la segunda derivada de la imagen. En las áreas donde la imagen tiene una intensidad constante (su gradiente es cero), la respuesta del operador LoG será cero. Sin embargo, en la vecindad de un cambio de intensidad, la respuesta del LoG será positiva en el lado más oscuro, y negativa en el lado más claro.

El operador LoG también es sensible al ruido. Sin embargo, los efectos de dicho ruido pueden ser reducidos si se ignoran los cruces por cero producidos por pequeños cambios en la intensidad de la imagen. Además, este operador nos

proporciona información sobre la dirección de los ejes, determinada mediante la dirección del cruce por cero.

A continuación, se muestra el resultado del uso del operador LoG en una imagen ejemplo de código de barras en la figura (11).



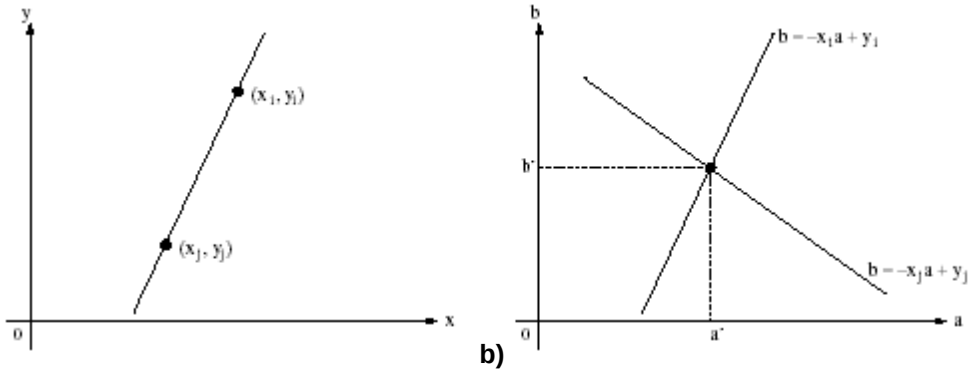
a) **b)**

Figura (11) Detección de bordes de imagen de código de barras mediante el operador LoG (11.a) Imagen original. (11.b) Aplicación del operador LoG. Elaboración: Los autores.

2.2.2 Modelamiento matemático del posicionamiento de la imagen mediante el método básico de identificación por la transformada de Hough

2.2.2.1 Transformada de Hough

Supongamos que para **n** puntos de la imagen se desean encontrar aquellos subconjuntos de puntos que caen en líneas rectas. Una posible solución podría ser, en primer lugar, encontrar todas las líneas determinadas por cada par de puntos y entonces encontrar todos los subconjuntos de puntos que están cerca de cada recta en particular. Este problema así planteado requiere encontrar **$n(n-1)/2 \sim n^2$** rectas y realizar **$n(n(n-1))/2 \sim n^3$** comparaciones de cada punto a línea. Este método no será viable salvo en casos triviales.



a) **b)**

Figura (12) Representación de la recta en los planos XY y ab (12.a) Espacio de parámetros XY. (11.b) Espacio de parámetros ab.

Fuente: Gonzales, R., y Woods, R. (2007). Digital image processing, third edition. New Jersey, Estados Unidos: Prentice Hall.

Una alternativa al método de búsqueda exhaustiva es la transformada de Hough. Consideramos un punto **(x_i,y_i)** y la ecuación de la recta, de la forma pendiente y ordenada al origen:

$$y_i = ax_i + b$$

Por el punto **(x_i,y_i)** pasan infinitas rectas, pero todas satisfacen la ecuación anterior para diferentes valores de **a** y **b**. Sin embargo, escribiendo esta ecuación en la forma:

$$b = ax_i + y_i$$

y considerando el plano **ab** (también denominado espacio de parámetros) da lugar a una única recta para el par **(x_i,y_i)** constante. Si ahora consideramos un segundo punto **(x_j,y_j)**, también va a tener su recta asociada en el espacio de parámetros. Estas dos rectas se cortarán en el espacio de parámetros en un punto **(a',b')**, donde **a'** es la pendiente y **b'** la ordenada al origen de la recta que contiene a los puntos **(x_i,y_i)** y **(x_j,y_j)** en el plano **xy**, como se puede ver en la figura (12). De hecho, todos los puntos de esa recta en el plano **xy** darán lugar

a rectas diferentes en el espacio de parámetros que se cortan en un único punto **(a',b')**.

El atractivo de la transformada de Hough proviene de subdividir el espacio de parámetros en celdas acumuladoras, como se puede ver en la figura (13), donde (a_{min},a_{max}) y (b_{min},b_{max}) son los rangos esperados para la pendiente y la ordenada al origen. La celda de coordenadas **(i,j)** con un valor de acumulador **A(i,j)** corresponde al cuadrado asociado con las coordenadas (a_i,b_j) del espacio de parámetros. Inicialmente se ponen todos los acumuladores a cero. Entonces para cada punto (x_k,y_k) de la imagen, permitimos que el parámetro a pueda tomar cualquier valor de entre los a_i permitidos y calculamos **b** usando la ecuación anterior. Los valores resultantes para el parámetro **b** se redondean hasta los b_j permitidos. Si para un valor a_p resultó un valor b_q se tiene que:

$$A(p,q) = A(p,q) + 1$$

Al final, un valor de **M** en el acumulador **A(i,j)** significa que **M** puntos del plano **xy** caen sobre la recta $y=a_ix+b_j$. La precisión en la co-linealidad de estos puntos depende del número de celdas del espacio de parámetros.

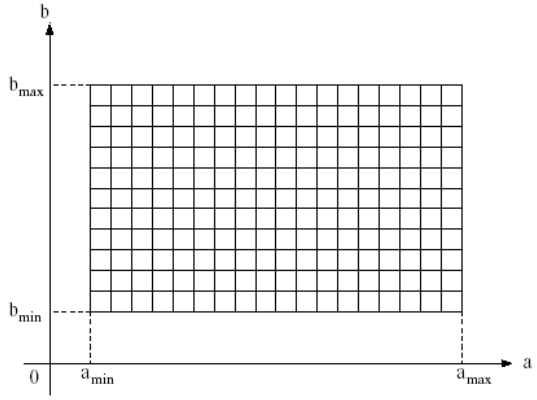


Figura (13) Celdas acumuladoras en el espacio de parámetros ab.

Fuente: Gonzales, R., y Woods, R. (2007). Digital image processing, third edition. New Jersey, Estados Unidos: Prentice Hall.

Si subdividimos el eje a en **K** celdas, para cada punto (x_k, y_k), obtenemos **K** valores de **b** correspondientes a los **K** posibles valores de **A**. Si la imagen tiene **n** puntos, la carga computacional es del orden de n_K. La transformada de Hough es lineal en **n**, y el producto n_K es mucho menor que si hubiéramos empleado una búsqueda exhaustiva, a menos que **K** sea del orden o exceda a **n**.

Un problema que surge al emplear la ecuación de la recta **y=ax+b** para representar una línea es que tanto la pendiente como la ordenada al origen pueden llegar a valer infinito, según la línea se hace vertical. Una forma de solventar este problema consiste en utilizar la representación normal de la recta:

$$\rho = x * \cos\theta + y * \sin\theta$$

En la figura (14), se puede ver el significado de los nuevos parámetros **(ρ, θ)**. El uso de esta representación para construir la tabla de acumuladores es similar al método explicado para las rectas en la forma pendiente y ordenada al origen. A cada punto del plano **xy** corresponde ahora una sinusoide en el plano **ρθ** en lugar de una recta. Al igual que antes, **M** puntos co-lineales a la recta $x*\cos\theta_j + y*\text{sen}\theta_j = \rho i$ darán lugar a **M** sinusoides que se cortan en el punto **(ρi,θj)** en el espacio de parámetros. Incrementando **θ** y calculando **ρ**, obtendremos **M** entradas en el acumulador **A(i,j)** correspondiente al par (ρ_i, θ_j).

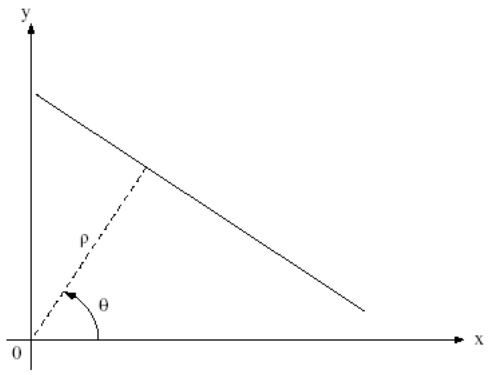


Figura (14) Representación normal de una recta.
Fuente: Gonzales, R., y Woods, R. (2007). Digital image processing, third edition. New Jersey, Estados Unidos: Prentice Hall.

En la figura (15), podemos ver la tabla de acumuladores del espacio de parámetros en este caso. El rango para el ángulo **θ** es **±90º**, medido con respecto al eje de abscisas. Se permiten valores negativos de ρ para rectas por detrás del origen de coordenadas del plano **xy**. Por ejemplo, una recta horizontal corresponde a un valor de **θ=0º** y un valor de **ρ** igual a la ordenada al origen, mientras que una recta vertical corresponde a un valor de **θ=90º** y un valor de **ρ** igual a la abscisa en el origen.

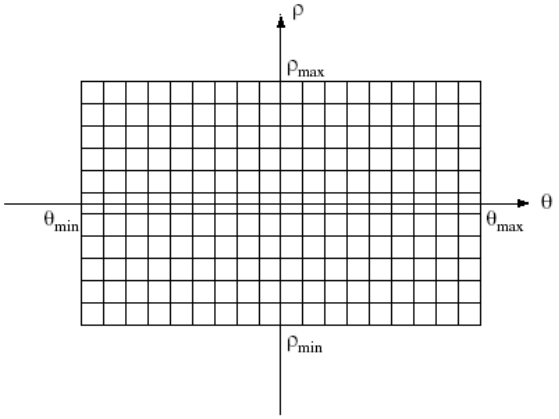


Figura (15) Celdas acumuladoras en el espacio ρθ.
Fuente: Gonzales, R., y Woods, R. (2007). Digital image processing, third edition. New Jersey, Estados Unidos: Prentice Hall.

En la figura (16), se ilustra con un ejemplo cómo funciona la transformada de Hough. La imagen está deformada por cuatro puntos correspondientes a las esquinas de un cuadrado. Estos cuatro puntos dan lugar a cuatro sinusoides en el espacio **ρθ**. Las cuatro sinusoides se cortan en seis puntos, correspondientes a las seis rectas posibles que pasan por los cuatro puntos del plano **xy**, que son a saber, los cuatro lados del cuadrado y las dos diagonales.

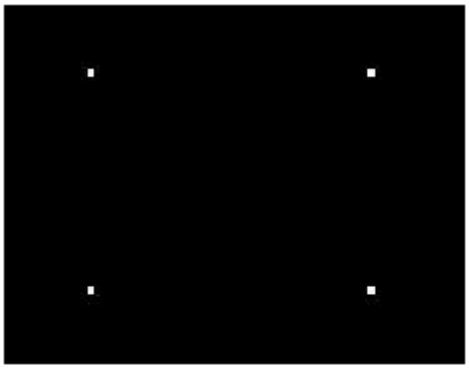
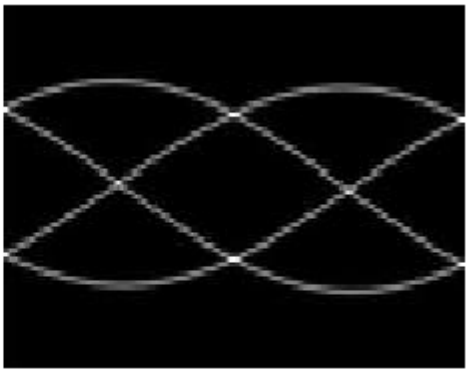
a)  **b)** 

Figura (16) Representación de la transformada de Hough en cuatro (04) puntos de bordes. (16.a) Imagen con cuatro (04) puntos de borde. (16.b) Transformada de Hough mostrando seis (06) puntos de corte correspondientes a las seis (06) rectas que pueden pasar por los cuatro (04) puntos.
Fuente: Gonzales, R., y Woods, R. (2007). Digital image processing, third edition. New Jersey, Estados Unidos: Prentice Hall.

Aunque hemos hecho un análisis para el caso de rectas, la transformada de Hough también es aplicable a cualquier función de la forma:

$$g(v, c) = 0$$

Donde **v** es un vector de coordenadas y **c** es un vector de coeficientes. Por ejemplo, puntos que caen en el círculo:

$$(x - c_1)^2 + (y + c_2)^2 = c_3^2$$

Se pueden detectar empleando también la transformada de Hough. En este caso, tenemos tres parámetros **(c_1, c_2, c_3)**, lo que dará lugar a un espacio de parámetros de tres dimensiones, con celdas con forma de cubo y acumuladores de la forma **A(i,j,k)**. El procedimiento, en este caso, es para cada punto del plano **xy**, para cada c_1 y para cada c_2, calcular el valor de c_3 y actualizar el acumulador correspondiente a **(c_1,c_2,c_3)**. La complejidad de la transformada de Hough es claramente dependiente del tamaño del espacio de parámetros.

$$(x - c_1)^2 + (y + c_2)^2 = c_3^2$$

2.2.2.2 Transformada de Karhunen-Loéve (Hotteling Transform)

Supongamos que cada punto puede ser descrito como un vector:

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

Un conjunto de tales puntos **{x_i}** podrían representar los píxeles que definen un objeto. Otro caso podría ser que **{x_i}** es la intensidad de un posición espacial específica en diferentes frecuencias (es decir, colores). Esto corresponde a los componentes RGB de una imagen en color. El vector promedio de **L** vectores (puntos) **x** se define como:

$$m_x = \varepsilon(x) = \frac{1}{L}\sum_{l=1}^{L} x_l$$

La matriz de covarianza se define como:

$$C_x = \varepsilon\{(x - m_x)(x - m_x)^T\}$$
$$= \frac{1}{L-1}\sum_{l=1}^{L} x_l x_l^T - m_x m_x^T$$

Los elementos de la diagonal principal de $\mathbf{x}$ son la varianza de las variables en $\mathbf{x_i}$.

Los elementos de la diagonal principal de $\mathbf{C_x}$ son una medida de cuan mucho las variables $\mathbf{x_i}$ e $\mathbf{x_j}$ dependen una de la otra.

Si $\mathbf{x_i}$ y $\mathbf{x_j}$ no están correlacionados entonces $\mathbf{C_x(i;j) = 0}$ (la inversa no es necesariamente válida)

Se tiene la libertad para crear nuevas variables **y** en base a combinaciones lineales de las ya existentes de acuerdo con:

$$y = A^T x$$

El valor medio de las nuevas variables vienen dadas por:

$$m_y = \varepsilon(y) = A^T m_x$$

A menudo se define:

$$y = A^T(x - m_x)$$

dando $\mathbf{m_y = 0}$

La covarianza de y será:

$$Cy = AC_x A^T$$

Si **A** es un vector, los puntos en el espacio n-dimensional será proyectado en una línea con coeficiente direccional **A**.

Supongamos que elegimos **A** de acuerdo a:

$$A = \Phi\Lambda^{-1/2}$$

donde las columnas en **Φ** son los vectores propios de **A** y los elementos de la diagonal principal de **Λ** son los valores propios de **A**.

Valores propios

Para una matriz A ϵ R^{NxN} siempre podemos encontrar valores de **N** λ_K **ϵ** R^{1xN} para que:

$$|A - \lambda I| = 0$$

También puede encontrar **N** vectores v_k **ϵ** R^{Nx1} de manera que:

$$Av_k = \lambda_k v_k$$

Donde λ_k es el k-ésimo valor propio y v_k está el vector propio asociado con este valor propio. Los vectores propios constituyen un conjunto orto - normal de vectores base para **A**.

De acuerdo con la definición de valores y vectores propios se cumple lo siguiente para una matriz de covarianza:

$$C = \Phi\Lambda\Phi^T$$

Teniendo en cuenta que utilizamos **A** como una transformación lineal para las variables con covarianza **C**.

$$A^T CA = (\Phi\Lambda^{-1/2})^T C(\Phi\Lambda^{-1/2})$$
$$= (\Lambda^{-1/2})^T \Phi^T C\Phi(\Lambda^{-1/2})$$
$$= (\Lambda^{-1/2})^T \Phi^T \Phi\Lambda\Phi^t\Phi(\Lambda^{-1/2})$$
$$= (\Lambda^{-1/2})^T I\Lambda I(\Lambda^{-1/2})$$
$$= (\Lambda^{-1/2})^t \Lambda(\Lambda^{-1/2})$$

pero como **Λ** es una matriz diagonal es su propia transpuesta y los elementos de la diagonal principal está dada por:

$$a_i = \frac{\lambda}{\sqrt{\lambda}\sqrt{\lambda}} = 1$$

Transformar **x** según $\mathbf{y = A^T (x - m_x)}$ dará $\mathbf{m_y} = 0$ y $\mathbf{C_y} = \mathbf{I}$

En general, no estamos interesados en la ampliación del eje del sistema de coordenadas, así como **Λ** = I y las columnas de **A** son los vectores propios de la matriz de covarianza.

$$C_y = \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_N \end{pmatrix}$$

La transformada encuentra un nuevo sistema de coordenadas a los puntos $\mathbf{\{x_i\}}$ y el primer eje de ese sistema maximiza la varianza de las observaciones.

Si $\mathbf{\{x_i\}}$ son píxeles que describen un objeto, esto es equivalente a localizar el eje principal del objeto.

En la práctica, podemos ver las nuevas variables como **y** no correlacionadas. El valor propio λ_i es la varianza de la i-ésima variable en **y**.

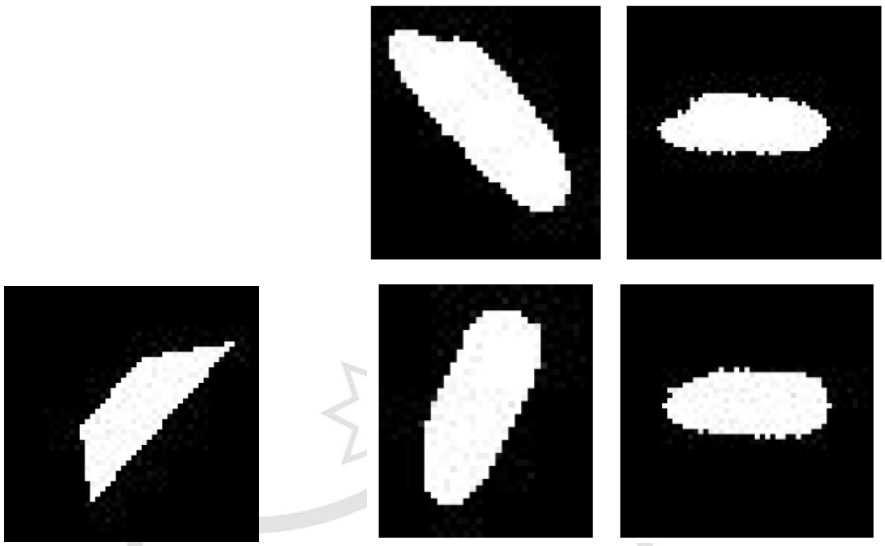
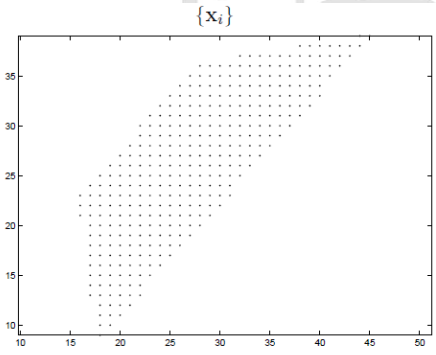
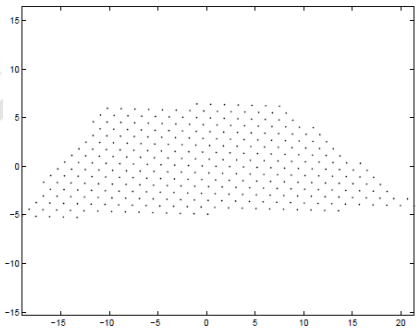


Figura (17) Rotación de la imagen mediante Hotteling Transform.
Fuente: Gonzales, R., y Woods, R. (2007). Digital image processing, third edition. New Jersey, Estados Unidos: Prentice Hall.



$$\mathbf{C} = \begin{pmatrix} 47.1172 & 38.8435 \\ 38.8435 & 51.3251 \end{pmatrix}$$

$$\mathbf{\Phi} = \begin{pmatrix} -0.7260 & 0.6877 \\ 0.6877 & 0.7260 \end{pmatrix}$$



$$\mathbf{y} = \mathbf{\Phi}^T(\mathbf{x} - \mathbf{m})$$

Figura (18) Detección de puntos de interés para la rotación de la imagen.
Fuente: Gonzales, R., y Woods, R. (2007). Digital image processing, third edition. New Jersey, Estados Unidos: Prentice Hall.

En general, es más fácil para representar una curva (figura 1D) de un objeto en 2D y por lo tanto, es deseable que sea capaz de transformar un objeto a una curva.

Una de las formas más sencillas de hacer esto es mediante, primero calcular el centro de masa para el objeto t_p. Desde t_p barremos con un rayo en las direcciones $0 \leq \theta \leq 2\pi$. Para cada dirección se calcula la distancia $r(\theta)$ de t_p hasta el borde del objeto.

Este método es la traducción invariante, ya que todos los cálculos se realizan en relación con el centro de masa del objeto. Al normalizar todo $r(\theta)$ en el intervalo único la señal será invariante en escala.

Para hacer que el método de rotacional sea invariante tenemos para asegurarse que el 0-dirección es siempre la misma. Esto se realiza mediante la rotación del objeto con la transformada Hotelling y utilizando el ángulo para el eje y_1 como una medida de la dirección. El método se hace más robusto al ruido por filtrado pasa baja el contorno antes del cálculo.

2.2.3 Modelamiento matemático de la binarización de la imagen mediante el método basado en la forma del histograma

2.2.3.1 Método basado en la forma del histograma

El histograma de un imagen de niveles de gris se define como una gráfica que a cada valor posible de nivel de gris entre [0,255] le asigna la proporción de pixeles en la imagen con dicho valor. Es decir si la imagen tiene **N** pixeles en total y hay ρ_r con valor **r**, el valor del histograma en **r** será $h(r) = \rho_r/N$.

Los métodos de realce basados en el histograma tienen como fin modificar la masa de puntos asignados a los distintos valores de niveles de gris de forma que la imagen mejore en su visualización. Es evidente que para conseguir esto de una forma coherente deberíamos definir lo que se entiende por el histograma de una imagen con buenas cualidades visuales. Si bien esta definición no existe como tal, si existe la experiencia de que cuando el rango de valores del histograma es el máximo posible y cuando la masa de puntos está repartida entre los posibles valores [0,255] de forma que el histograma se asemeje a la forma de una campana entonces el sistema visual humano consigue una máxima respuesta en su apreciación de la imagen.

A continuación, se enlistan distintos métodos que nos permiten transformar la forma del histograma de una imagen.

Ecualización del Histograma

Supongamos que la transformación es **s = T(r)** siendo **r** un valor de nivel de gris en la imagen original y **s** su valor en la imagen ecualizada. Entonces el primer objetivo será que el rango dinámico de los valores de **s** sea al máximo posible, lo que en condiciones de pantallas normales significa [0,255].

Para que la función **T** además de conseguir el objetivo de alargar el rango dinámico de los valores de niveles de gris no introduzca perturbaciones sobre la imagen debemos imponer algunas condiciones sobre dicha función. Por ahora impondremos tan solo dos condiciones:

- En el intervalo de valores de **r**, **T(r)** debe ser una función monótona creciente.
- $0 \leq T(r) \leq 255$, para $0 \leq r \leq 255$

La primera condición preserva el orden en los pixeles desde el negro al blanco y la segunda condición garantiza que la función **T** conserva sus valores dentro del rango permitido.

Existen infinitas curvas que verifican estas condiciones, de hecho cualquier curva plana que una el origen de coordenadas con el punto [255,255] de manera que no tenga ni máximos ni mínimos y cuyo grafo se mantenga dentro del cuadrado definido por los puntos [0,0], [255,0], [0,255] y [255,255] verifica las condiciones establecidas.
Para seleccionar una curva con un comportamiento adecuado a nuestros deseos debemos y podemos considerar que estas funciones son distribuciones acumuladas de probabilidad, aunque para ello ahora supongamos que el rango de definición de los valores **r** y **s** están en [0,1] en lugar de [0,255]. Esta hipótesis no resta ninguna generalidad al análisis.

Un resultado elemental de la teoría de la probabilidad establece que si tenemos una variable aleatoria **R** que, en nuestro caso, sería la función que asigna valor de nivel de gris a cada pixel, la transformación de dicha variable aleatoria por la función acumulada de probabilidad de dicha variable aleatoria, $\mathbf{P_R(R)}$, define una distribución uniforme de valores. Es decir, que si consideramos $\mathbf{T = P_R}$, el conjunto de valores de s resultante de la transformación **s = T(r)** sigue una distribución uniforme.

Ya que una distribución uniforme supone que todos los puntos tienen la misma masa de probabilidad, parece de interés que usemos esta función **T** como medio de redistribuir la masa de puntos asignada a los distintos niveles de gris. En cada caso particular, debemos de construir la función **T** a partir de la imagen de la siguiente manera:

$$T(r_k) = 255 \sum_{j=0}^{k} \frac{n_j}{N}$$

Para **k =** 0,1,…, 255, donde **N** representa el número total de pixeles en la imagen, y $\mathbf{n_j}$ el número de pixeles con nivel de gris **j**.

Una generalización de la técnica anterior nos permitirá transformar una imagen cuyo histograma sea una distribución uniforme, a otra imagen cuya forma de histograma este definido por nosotros. Como ya mencionamos al principio de esta lección puede ser de interés transformar el histograma de la imagen original de manera que la imagen resultante tenga un histograma definido por una función en forma de campana (gaussiana).

Para llevar a cabo esta transformación habrá que hacer uso de nuevo del resultado antes mencionado de la teoría de la probabilidad. Ahora tendremos que hacer dos transformaciones para conseguir el objetivo,

- Transformar la imagen original a una imagen con un histograma definido por la distribución uniforme: **s = T(r)** siendo **T** la función definida antes.
- Calcular la función inversa del histograma acumulado que queremos que tenga la imagen final: $\mathbf{G^{-1}}$. Si queremos una imagen con un histograma definido por una gaussiana tendremos que calcular la función inversa de la función acumulada gaussiana.
- Transformar de nuevo la imagen haciendo uso de la expresión $\mathbf{z = G^{-1}(s)}$

La primera transformación nos permite pasar de una imagen con unas características dadas por la forma de su histograma a una imagen con un histograma que aproxima una distribución uniforme, y la segunda transformación nos permite pasar de dicha imagen "neutra" a una nueva imagen cuyo histograma tiene la forma dada por la función **G**. En resumen, hemos calculado una nueva imagen a partir de la original usando la transformación $\mathbf{z = G^{-1}[T(r)]}$.

En la práctica la función puede ser calculada de diversas maneras. Si conocemos su expresión analítica entonces tan solo tenemos que calcular su valor, pero si no es así podemos usar una versión digitalizada de la gráfica de función **G** (histograma acumulado) y a partir de dicha versión digitalizada aproximar el valor que estamos calculando. Si observamos cómo son las dos transformaciones antes comentadas podemos ver que la primera transforma valores del eje **x** en valores del eje **y** a través de **T**, y la segunda transforma los valores del eje en valores del eje **x** a través de **G**.

2.2.3.2 Método basado en la clusterización

Método de Kittler e Illingworth: Error mínimo

Al igual que Pun y Kapur se considera al histograma como una estimación de una función de densidad de probabilidad **p(g)**, que es una mezcla de dos distribuciones correspondientes al primer plano y fondo. Se supone, además, que estas distribuciones (primer plano y fondo) tienen una distribución normal con media y desviación estándar. Las funciones de densidad de probabilidad:

$$p_f(t) = \frac{1}{\sigma_f\sqrt{2\pi}} e^{\frac{-(t-u_f)^2}{2\sigma_f^2}}$$

$$p_b(t) = \frac{1}{\sigma_b\sqrt{2\pi}} e^{\frac{-(t-u_b)^2}{2\sigma_b^2}}$$

La función densidad de probabilidad del histograma:

$$p(t) = P_f(t) * p_f(t) + P_b(t) * p_b(t)$$

El error mínimo se alcanza cuando seleccionamos como umbral al valor para el cual

$$P_f(t) * p_f(t) = P_b(t) * p_b(t)$$

Aplicando logaritmo en ambos miembros, resolviendo las ecuaciones y sustituyendo los valores de μ_f, μ_b, σ_f^2, σ_b^2 con valores estimados:

$$\mu_f(t) = \sum_{g=1}^{t} g \frac{p(g)}{P_f(T)}$$

$$\mu_b(t) = \sum_{g=t+1}^{255} g \frac{p(g)}{P_b(T)}$$

$$\sigma^2 f(t) = \sum_{g=1}^{t} (g - \mu_f)^2 * p(g)$$

$$\sigma^2 b = \sum_{g=t+1}^{255} (g - \mu_b)^2 * p(g)$$

Se define una función criterio, la cual se busca un valor mínimo

$$J(t) = 1 + 2\left(P_f(t) log\sigma_f(t) + P_b log\sigma_b(t)\right) - 2\left(P_f(t) log P_f(t) + P_b log P_b(t)\right)$$

$$T^* = Min\{J(t)\}$$

2.2.3.3 Método basado en la entropía

Esta clase de algoritmos explota la entropía de la distribución de los niveles de gris en una imagen. La maximización de entropías es interpretada coma la máxima información trasmitida, Pun (1981) introdujo, en el cálculo del umbral, la función entropía de Shannon, considero la imagen como el resultado señales de una misma fuente. Su criterio se basa en encontrar el umbral óptimo que maximice la entropía entre las dos clases (el objeto y fondo)

Método de Pun

Se consideran los píxeles de una imagen convertida a 256 niveles de gris y se separan en dos niveles principales de gris, el primer plano o foreground y un fondo de base o background. La variable g denotará esos valores de niveles de gris. Para imágenes de 8 bits **g** = 0...255. Para ello procede con la separación de píxeles.

Sea el conjunto de píxeles de la imagen:

$$I = \{g \epsilon I \; 0 \leq g \; \leq 255\}$$

Píxeles correspondiente al objeto (foreground) y el fondo (background):

$$F = \{g \epsilon I \; g > t\}$$
$$B = \{g \epsilon I \; g \leq t\}$$

En el contexto de procesamiento de imágenes, el foreground es el conjunto de pixeles con luminancia menor a un cierto valor **N**, mientras que el background es el conjunto de pixeles con luminancias por encima de este **T**. Calculamos las probabilidades estimadas de cada pixel **g** haciendo el cociente entre $\mathbf{n_g}$ y **N**, siendo $\mathbf{n_g}$ el número de veces que se repite el pixel **g** en la imagen y **N** la cantidad total de pixeles. La función probabilidad para cada nivel de gris:

$$p(g) = \frac{n_g}{N} \qquad g = 0,1, \dots, 255$$

Con:

$$\sum_{g=0}^{255} p(g) = 1 \qquad N = \sum n_g$$

Las probabilidades del objeto y fondo de acuerdo a un umbral **T**, están expresadas como:

$$p_f(g), 0 \leq g \leq t$$
$$p_b(g), t+1 \leq g \leq 255$$

Definimos la función de probabilidad acumulada como se muestra en la siguiente ecuación:

$$P(t) = \sum_{g=0}^{t} p(g)$$

Esta función de probabilidad puede ser considerada como una suma o unión de dos funciones de probabilidad, una para zonas claras (foreground) y otra para zonas oscuras (background).

$$P_f(t) = P_f = \sum_{g=0}^{t} p(g)$$
$$P_b(t) = P_b = \sum_{g=t+1}^{255} p(g)$$

El primer trabajo de Pun define las entropías del fondo y del objeto usando la entropía de Shannon, paramétricamente dependiente del valor umbral **T**.

$$H_f(t) = -\sum_{g=0}^{t} p_f(g) * log p_f(g)$$
$$H_b(t) = -\sum_{g=t+1}^{255} p_b(g) * log p_b(g)$$

La suma de estas dos expresiones puede ser denotada como **H**, indicada en las siguientes ecuaciones:

$$H = H_f(t) + H_b(t)$$

$$H = -\sum_{g=0}^{255} p(g) * logp(g)$$

El umbral óptimo será, entonces, aquel que maximice esta entropía global.

$$T^* = Max\{H(t)\}$$

En su segundo trabajo, Pun interpreta la maximización de las entropías con la maximización de una función **F**.

$$F(t) = \frac{H_f log P_f}{Hlog(\max\{p_0, p_1, \dots, p_t\})} + \frac{1 - H_f log P_b}{Hlog(\max\{p_{t+1}, p_{t+2}, \dots, p_{255}\})}$$

Método de Kapur

Siguiendo las ideas de Pun, Kapur realiza modificaciones cambiando las probabilidades de los elementos:

$$p(g) = p_g / p_t \qquad g = 0,1, \dots, 255$$

Por lo tanto, las entropías del primer plano y fondo quedan definidas como indican las ecuaciones a continuación:

$$H_f(t) = -\sum_{g=0}^{t} \frac{p(g)}{P_t} * log \frac{p(g)}{P_t}$$

$$H_b(t) = -\sum_{g=t+1}^{255} \frac{p(g)}{P_t} * log \frac{p(g)}{P_t}$$

El máximo de la función H será el valor de umbral que optimiza la separación entre primer plano y fondo:

$$H = H_f(t) + H_b(t)$$

$$T^* = Max\{H(t)\}$$

CAPÍTULO III
DESARROLLO DEL PROYECTO

3.1 Análisis matemático del proyecto

3.1.1 Cálculo de la derivación estándar y el umbral de los bordes de la imagen mediante el operador de Canny

3.1.1.1 Cálculo de la desviación estándar (sigma) de la probabilidad de distribución asociada del filtro gaussiano para el suavizado de la imagen de código de barras

El filtrado consiste en aplicar una transformación de forma que se acentúen o disminuyan ciertos aspectos de la imagen. En nuestro caso, se realizara un filtrado gaussiano para eliminar el ruido y suavizar la imagen.

La finalidad es obtener una máscara **N x N** para aplicársela a la imagen. Para ello se toma el eje de coordenadas en el píxel central de una ventana y se sustituyen los valores de **x** e **y**, obteniendo así la máscara. Para la implementación de la máscara, se utiliza la siguiente operación gaussiana:

$$g(x,y) = e^{\frac{-(x^2+y^2)}{2\sigma^2}}$$

Donde **x** e **y** son las coordenadas de la imagen y **σ** la desviación estándar de la probabilidad de distribución asociada. Dicha desviación es el único parámetro del filtro y puede estar relacionado con el tamaño de la máscara, es decir con el área determinada por los pixeles vecinos para aplicar el filtro. Para ello se usa la siguiente relación:

$$C = 2\sqrt{2\sigma^2}$$

Obteniendo el valor de **C**, se halla el tamaño de la máscara teniendo en cuenta que:

$$N \geq 3C$$

Tomemos como ejemplo, la matriz de una imagen representada a continuación, a la cual se le aplicara un filtro gaussiano con una desviación estándar **σ = 0.5**.

$$\begin{bmatrix} 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 \\ 10 & 10 & 10 & 10 & 10 & 10 & 10 & 10 & 10 & 10 \\ 25 & 30 & 30 & 37 & 37 & 37 & 37 & 30 & 30 & 25 \\ 25 & 30 & 30 & 37 & 45 & 45 & 37 & 30 & 30 & 25 \\ 25 & 30 & 30 & 37 & 68 & 68 & 37 & 30 & 30 & 25 \\ 25 & 30 & 30 & 37 & 37 & 37 & 37 & 30 & 30 & 25 \\ 25 & 30 & 30 & 37 & 42 & 42 & 37 & 30 & 30 & 25 \\ 25 & 30 & 30 & 37 & 42 & 42 & 37 & 30 & 30 & 25 \\ 25 & 30 & 30 & 37 & 37 & 37 & 37 & 30 & 30 & 25 \\ 50 & 50 & 50 & 50 & 50 & 50 & 50 & 50 & 50 & 50 \end{bmatrix}$$

Para obtener el tamaño de la máscara, se desarrollan las fórmulas vistas anteriormente:

$$C = 2\sqrt{2\sigma^2} = 2\sqrt{2(0.5)^2} = 1.4142$$

$$N \geq 3C \ \geq 3(1.4142) \geq 4.2426 \ \cong 5$$

Con el valor de **N** se desarrollan las matrices **x** e **y**, las cuales darán como resultado, usando el operador gaussiano, la máscara del filtro.

$$x = \begin{bmatrix} -2 & -1 & 0 & 1 & 2 \\ -2 & -1 & 0 & 1 & 2 \\ -2 & -1 & 0 & 1 & 2 \\ -2 & -1 & 0 & 1 & 2 \\ -2 & -1 & 0 & 1 & 2 \end{bmatrix}; \qquad y = \begin{bmatrix} 2 & 2 & 2 & 2 & 2 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & -1 & -1 & -1 \\ -2 & -2 & -2 & -2 & -2 \end{bmatrix}$$

$$g(x,y) = \begin{bmatrix} 0.0001 & 0.0000 & 0.0003 & 0.0000 & 0.0001 \\ 0.0000 & 0.0183 & 0.1353 & 0.0183 & 0.0000 \\ 0.0003 & 0.1353 & 1.0000 & 0.1353 & 0.0003 \\ 0.0000 & 0.0183 & 0.1353 & 0.0183 & 0.0000 \\ 0.0001 & 0.0000 & 0.0003 & 0.0000 & 0.0001 \end{bmatrix}$$

Por último, se normaliza sumando todos los elementos de la máscara para obtener en qué medida contribuyen los píxeles vecinos al filtrado del píxel central, obteniendo sum = 1.616. El filtro gaussiano resultante será **h**.

$$h(x,y) = \frac{g(x,y)}{sum} = \begin{bmatrix} 0.0000 & 0.0000 & 0.0001 & 0.0000 & 0.0000 \\ 0.0000 & 0.0113 & 0.0837 & 0.0113 & 0.0000 \\ 0.0001 & 0.0837 & 0.6188 & 0.0837 & 0.0001 \\ 0.0000 & 0.0113 & 0.0837 & 0.0113 & 0.0000 \\ 0.0000 & 0.0000 & 0.0001 & 0.0000 & 0.0000 \end{bmatrix}$$

Dicho filtro gaussiano influye el propio píxel central en un 61,88%, los píxeles que se encuentran en las direcciones horizontal y vertical en un 8,37% y los píxeles que se encuentran en las diagonales en un 1,13%. A continuación se aprecia la aplicación del filtro gaussiano en la matriz de la imagen y en la figura (19), el resultado visual en un código de barras respectivamente.

$$\begin{bmatrix} 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 \\ 10 & 10 & 10 & 10 & 10 & 10 & 10 & 10 & 10 & 10 \\ 25 & 30 & 30 & 37 & 37 & 37 & 37 & 30 & 30 & 25 \\ 25 & 30 & 30 & 0.4181 & 3.7665 & 0.5085 & 37 & 30 & 30 & 25 \\ 25 & 30 & 30 & 3.0969 & 42.0784 & 5.6916 & 37 & 30 & 30 & 25 \\ 25 & 30 & 30 & 0.4181 & 3.0969 & 0.4181 & 37 & 30 & 30 & 25 \\ 25 & 30 & 30 & 37 & 42 & 42 & 37 & 30 & 30 & 25 \\ 25 & 30 & 30 & 37 & 42 & 42 & 37 & 30 & 30 & 25 \\ 25 & 30 & 30 & 37 & 37 & 37 & 37 & 30 & 30 & 25 \\ 50 & 50 & 50 & 50 & 50 & 50 & 50 & 50 & 50 & 50 \end{bmatrix}$$

Figura (19) Resultado de la aplicación del filtro gaussiano en la imagen de código de barras.

Elaboración: Los autores.

3.1.1.2 Cálculo de la magnitud y dirección de la gradiente de los contornos de la imagen de código de barras

El algoritmo de Canny encuentra básicamente bordes donde la intensidad en escala de grises de la imagen cambia al máximo. Estas áreas se encuentran mediante la determinación de los gradientes de la imagen. La gradiente en cada píxel de la imagen suavizada se determina mediante la aplicación de lo que se conoce como el operador de Sobel. El primer paso es aproximar el gradiente en las direcciones **x** e **y**, respectivamente, mediante la aplicación de las máscaras que se muestran a continuación:

$$K_{GX} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$$K_{GY} = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Las magnitudes de gradiente (también conocidos como los puntos fuertes de borde), se puede determinar como una medida de la distancia euclidiana mediante la aplicación de la Ley de Pitágoras como se muestra en la siguiente ecuación:

$$\mathrm{G(x,y)} = \left[\mathrm{G_x^2} + \mathrm{G_y^2}\right]^{1/2}$$

Donde $\mathbf{G_x}$ y $\mathbf{G_y}$ son la gradiente en las direcciones **x** e **y** respectivamente. Para reducir la complejidad computacional, se simplifica la ecuación anterior mediante la aplicación de la medida de distancia Manhattan como se muestra a continuación:

$$|G(x,y)| = |G_x| + |G_y|$$

La representación de las magnitudes de la gradiente a menudo muestra los bordes con bastante claridad. Sin embargo, los bordes son típicamente amplios y por tal motivo no indican exactamente donde se ubican estos. Para que sea posible determinarlos, se deberá hallar la dirección de los bordes, la cual vienen dada por la siguiente ecuación:

$$\alpha(x,y) = \tan^{-1}\left(\frac{G_y}{G_x}\right)$$

Los componentes del vector de la gradiente son operadores lineales, pero la magnitud no lo es debido a las operaciones de raíz cuadrada y elevación al cuadrado. Por otro lado, la derivada parcial de la ecuación anterior no es isotrópica, pero la magnitud si lo es. A continuación, se observan los puntos de una imagen en una matriz 3x3. El centro en este caso es $\mathbf{z_5}$ el cual se

considera como **f(x,y)**, z_1 en este caso sería **f(x-1,y-1)**. La aproximación más simple de la derivada de primer orden que satisface las condiciones de estado son $\mathbf{G_x = (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)}$ y $\mathbf{G_y = (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)}$.

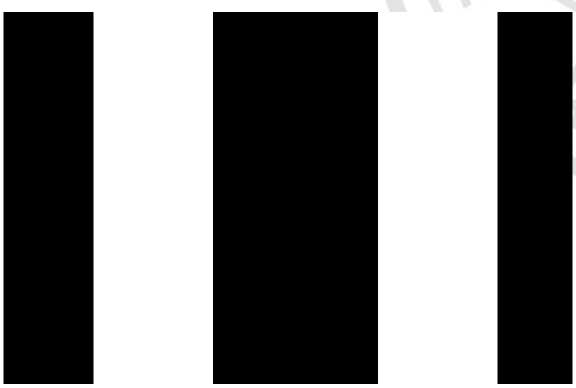
$$\begin{bmatrix} z_1 & z_2 & z_3 \\ z_4 & z_5 & z_6 \\ z_7 & z_8 & z_9 \end{bmatrix}$$

Reemplazando en la ecuación de la magnitud de la gradiente, se tiene la siguiente operación:

$$|G(x,y)| = |(z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)| \\ + |(z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)|$$

En la figura (20), se muestra el borde de un código de barras y la matriz que lo representa, de la cual tomaremos una (01) región la cual será evaluada para hallar la magnitud de la gradiente y la dirección de contorno.

a)



b)

$$\begin{bmatrix} 0 & 0 & 20 & 20 & 0 & 0 & 0 & 20 & 20 & 0 & 0 \\ 0 & 0 & 30 & 30 & 0 & 0 & 0 & 30 & 30 & 0 & 0 \\ 0 & 0 & 30 & 30 & 0 & 0 & 0 & 30 & 30 & 0 & 0 \\ 0 & 0 & 30 & 30 & 0 & 0 & 0 & 30 & 30 & 0 & 0 \\ 0 & 0 & 30 & 30 & 0 & 0 & 0 & 30 & 30 & 0 & 0 \\ 0 & 0 & 30 & 30 & 0 & 0 & 0 & 30 & 30 & 0 & 0 \\ 0 & 0 & 30 & 30 & 0 & 0 & 0 & 30 & 30 & 0 & 0 \\ 0 & 0 & 30 & 30 & 0 & 0 & 0 & 30 & 30 & 0 & 0 \\ 0 & 0 & 30 & 30 & 0 & 0 & 0 & 30 & 30 & 0 & 0 \\ 0 & 0 & 30 & 30 & 0 & 0 & 0 & 30 & 30 & 0 & 0 \\ 0 & 0 & 30 & 30 & 0 & 0 & 0 & 30 & 30 & 0 & 0 \\ 0 & 0 & 20 & 20 & 0 & 0 & 0 & 20 & 20 & 0 & 0 \end{bmatrix}$$

Figura (20) Representación de la imagen de bordes del código de barras en matriz numérica de intensidad de grises. (20.a) Borde de código de barras. (20.b) Matriz del borde.

Elaboración: Los autores.

Se reemplazan los valores en las ecuaciones:

$$|G(x,y)| = |(0+2*30+30)-(0+2*20+20)| \\ +|(20+2*30+30)-(0+2*0+0)| = 140$$

$$\alpha(x,y) = \tan^{-1}\left(\frac{(20+2*30+30)-(0+2*0+0)}{(0+2*30+30)-(0+2*20+20)}\right) = 74.74°$$

Para cada uno de los pixeles se repite el mismo procedimiento, lo cual nos da como resultado la matriz que se muestra a continuación.

$$\begin{bmatrix} 0 & 100 & 20 & 20 & 100 & 0 & 100 & 20 & 20 & 100 & 0 \\ 0 & 120 & 140 & 140 & 120 & 0 & 120 & 140 & 140 & 120 & 0 \\ 0 & 120 & 120 & 120 & 120 & 0 & 120 & 120 & 120 & 120 & 0 \\ 0 & 120 & 120 & 120 & 120 & 0 & 120 & 120 & 120 & 120 & 0 \\ 0 & 120 & 120 & 120 & 120 & 0 & 120 & 120 & 120 & 120 & 0 \\ 0 & 120 & 120 & 120 & 120 & 0 & 120 & 120 & 120 & 120 & 0 \\ 0 & 120 & 120 & 120 & 120 & 0 & 120 & 120 & 120 & 120 & 0 \\ 0 & 120 & 120 & 120 & 120 & 0 & 120 & 120 & 120 & 120 & 0 \\ 0 & 120 & 120 & 120 & 120 & 0 & 120 & 120 & 120 & 120 & 0 \\ 0 & 120 & 120 & 120 & 120 & 0 & 120 & 120 & 120 & 120 & 0 \\ 0 & 120 & 140 & 140 & 120 & 0 & 120 & 140 & 140 & 120 & 0 \\ 0 & 100 & 20 & 20 & 100 & 0 & 100 & 20 & 20 & 100 & 0 \end{bmatrix}$$

Aplicando dicho procedimiento en la imagen de código de barras de la figura (21.a), nos resultara las imágenes de las figuras (21.b) y (21.c).

Figura (20.a) Imagen de código de barras original.

Elaboración: Los autores.

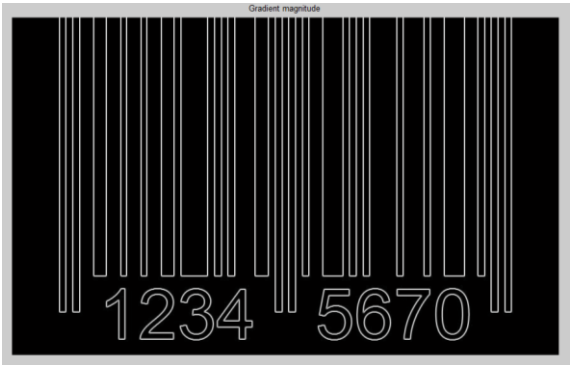


Figura (20.b) Representación de la magnitud de la gradiente de la imagen de código de barras.
Elaboración: Los autores.

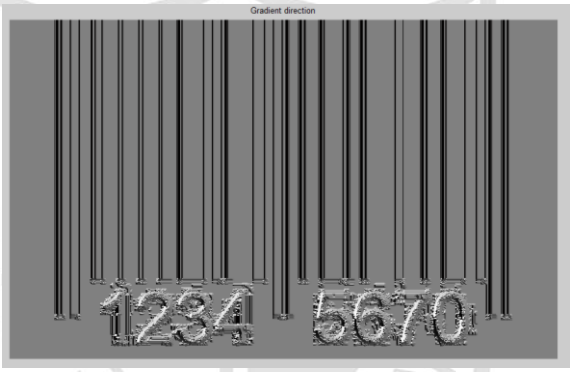


Figura (20.c) Representación de la dirección de la gradiente de la imagen de código de barras.
Elaboración: Los autores.

3.1.1.3 Selección de umbrales máximos y mínimos para la detección de bordes de la imagen de código de barras

Supresión no máxima

El propósito de este paso es convertir los bordes "borrosas" en la imagen de las magnitudes de gradiente a bordes "afilados". Básicamente, esto se hace mediante la preservación de todos los máximos locales en la imagen de gradiente, y eliminación de todo lo demás. El algoritmo es para cada uno de los píxeles en la imagen de gradiente:

1. Redondear la dirección de la gradiente **α** más cercana a 45°, correspondiente a la utilización de un vecindario de 8 conectados.
2. Comparar la intensidad del borde del píxel actual con la intensidad del borde del píxel en la dirección del gradiente positivo y negativo. Es decir, si la dirección del gradiente es norte (**α** = 90°), comparar con los píxeles hacia el norte y el sur.
3. Si la intensidad de borde del píxel actual es más grande; preservar el valor de la magnitud del borde. Si no es así, suprimir (es decir, eliminar) el valor.

Un ejemplo de supresión no máxima se muestra en la figura (22). Casi todos los píxeles tienen direcciones de gradiente apuntando al este. Por lo tanto, se comparan con los píxeles de derecha e izquierda. Los píxeles que resultan ser máxima, en esta comparación, están marcados con bordes blancos. Se suprimen todos los demás píxeles.

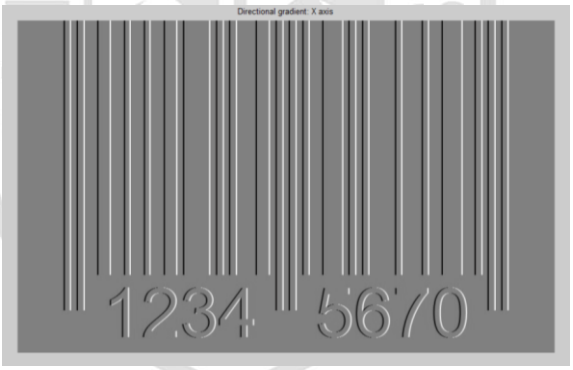


Figura (22) Supresión no máxima de la detección de bordes de la imagen de código de barras
Elaboración: Los autores.

Doble umbral

El borde píxeles restantes después de la etapa la supresión no máxima son (todavía) marcados con su intensidad de píxel a píxel. Muchos de ellos probablemente serán bordes verdaderos en la imagen, pero algunos pueden ser causados por ruido o variaciones de color, por ejemplo, debido a las

superficies ásperas. La forma más sencilla de discernir entre estos sería el uso de un umbral, de modo que solo el borde más fuerte con un cierto valor sería preservado. El algoritmo de detección de bordes de Canny utiliza doble umbral. Píxeles del borde más fuerte que el umbral superior se marcan como fuerte; píxeles del borde más débiles que el umbral bajo se suprimen y píxeles del borde entre los dos umbrales se marcan como débil. El efecto sobre la imagen se muestra en la figura (23). La selección del umbral se seleccionará en la etapa de diseño.

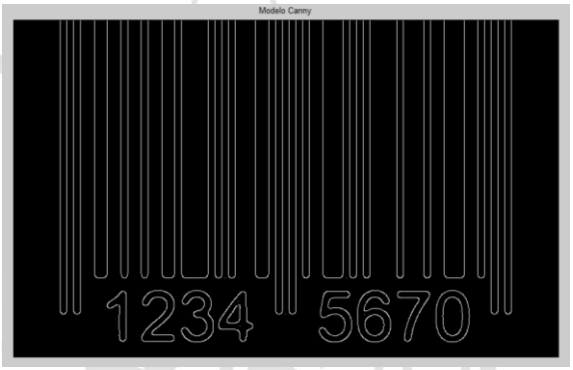


Figura (23) Detección de bordes de la imagen de código de barras con un umbral mínimo de $1e^{-1}$.
Elaboración: Los autores.

Seguimiento de bordes por histéresis

Los bordes fuertes se interpretan como "ciertos bordes", e inmediatamente se pueden incluir en la imagen de borde final. Los bordes débiles se incluyen si y solo si están conectados a los bordes fuertes. La lógica es, por supuesto, que el ruido y otras variaciones pequeñas son poco probables que resulten en un borde fuerte (con ajuste apropiado de los niveles de umbral). Así los bordes fuertes serán casi sólo ser debido a bordes verdaderos en la imagen original. Los bordes débiles o bien puede ser debido a bordes verdaderos o variaciones de ruido y/o de color. El último tipo probablemente se distribuye de forma independiente de bordes en la imagen completa, y por lo tanto sólo una

pequeña cantidad se encuentra adyacente a los bordes fuertes. Los bordes débiles debido a los bordes verdaderos son mucho más propensos a estar conectados directamente a los bordes fuertes.

3.1.2 Cálculo del ángulo de rotación de la imagen de código de barras mediante la detección de líneas por la transformada de Hough

La transformada Hough está diseñada para detectar líneas, en nuestro caso se detectaran dentro de la imagen capturada del código de barras. Usa la representación paramétrica de una línea:

$$\rho = x * \cos(\theta) + y * \sin(\theta)$$

Donde la variable **ρ** es la distancia desde el origen hasta la línea a través de un vector perpendicular de la misma y **θ** es el ángulo entre el eje X y este vector.

El estándar de la transformada Hough (TH) es una matriz de espacio de parámetros cuyas filas y columnas corresponden a los valores de **ρ** y **θ**, respectivamente. Los elementos en el TH representan células de acumuladores. Inicialmente, cada célula se ajusta a cero. Entonces, para cada punto no fondo en la imagen, **ρ** se calcula para cada **θ**. Entonces **ρ** es redondeado a la fila más cercana permitida en TH. Esa célula del acumulador se incrementa; al final de este procedimiento, un valor de Q en TH(r,c) significa que los puntos Q en el plano XY caen en la línea especificada por el **θ**(c) y **ρ**(r). Los valores pico en la TH representa líneas posibles en la imagen de entrada. Para demostrar la transformada de Hough, se toma como ejemplo la ecuación de la recta la cual se representa en el plano XY, como se aprecia en la figura (24) a continuación.

$$x + y = 9$$

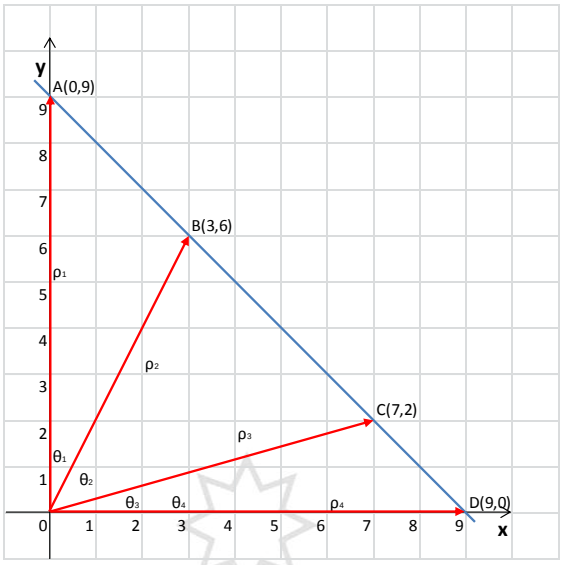


Figura (24) Recta representada en el plano XY.

Elaboración: Los autores.

En dicha recta, se toman cuarto (04) puntos, los cuales son A(0,9), B(3,6), C(7,2) y D(9,0). Para cada uno de estos puntos se tiene un **ρ** y **θ** diferentes, los cuales serán representados en el plano **ρθ** como se muestra en la figura (25) y los valores hallados para cada punto se muestran en la tabla (1).

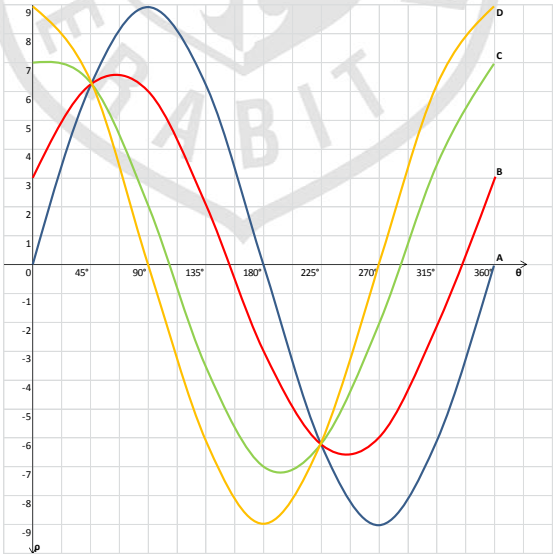


Figura (25) Representación de la transformada de Hough para cada uno de los puntos de la recta.

Elaboración: Los autores.

Tabla (1) Valores hallados para cada uno de los puntos.

		θ								
		0°	45°	90°	135°	180°	225°	270°	315°	360°
ρ	A(0,9)	0	6.36	9	6.36	0	-6.36	-9	-6.36	0
	B(3,6)	3	6.36	6	2.12	-3	-6.36	-6	-2.12	3
	C(7,2)	7	6.36	2	-3.54	-7	-6.36	-2	3.54	7
	D(9,0)	9	6.36	0	-6.36	-9	-6.36	0	6.36	9

Elaboración: Los autores.

Como se puede apreciar, las curvas resultantes de la evaluación de cada uno de los puntos de la recta se cruzan en el punto (6.36, 45°), los cuales son los valores pico de **ρ** y **θ**. Obteniendo estos valores, se utilizará el ángulo hallado para realizar la rotación de la imagen.

En la figura (26), se puede apreciar la captura de un código de barras con un cierto ángulo de inclinación. Para el proceso de decodificación del código, se requiere que dicha imagen se encuentre en forma recta.

Figura (26) Captura de código de barras con ángulo de inclinación θ.
Elaboración: Los autores.

Utilizando el procedimiento anterior, se halla una curva para cada uno de los puntos de la imagen del código de barras lo cual se aprecia en la figura (27).

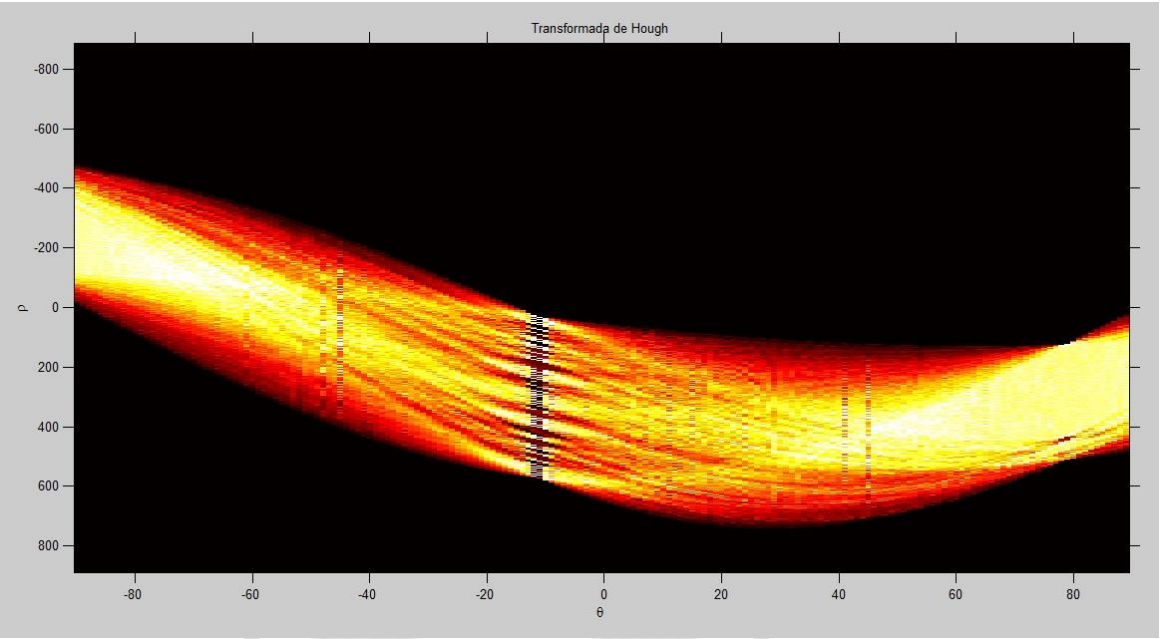


Figura (27) Representación de la transformada de Hough de la imagen de código de barras.

Elaboración: Los autores.

Dicho procedimiento da como resultado los valores (887,11°), de los cuales se toma el ángulo para la rotación de la imagen del código de barras, dando como resultado la figura (28).

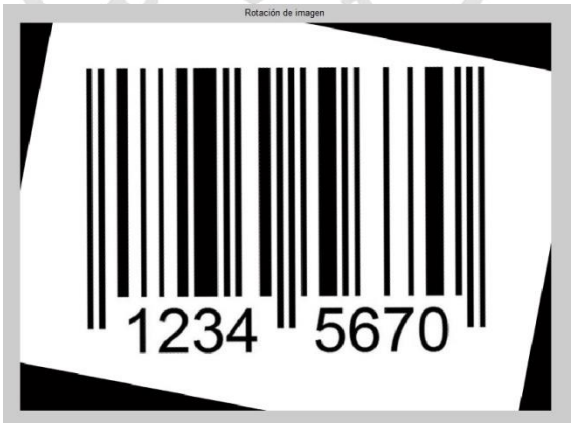


Figura (28) Imagen de código de barras rotada en -11° respecto al eje X.

Elaboración: Los autores.

3.1.3 Cálculo del umbral de binarización de la imagen de código de barras mediante el método basado en la forma del histograma

La binarización de una imagen consiste en el proceso de reducción de la información de la misma, en la que solo persisten dos valores: verdadero y falso. En una imagen digital, estos valores pueden representarse por los valores 0 y 1 o, en nuestro caso, por los colores negro (valor de gris 0) y blanco (valor de gris 255).

La binarización se emplea para separar las regiones u objetos de interés en una imagen del resto. Las imágenes binarias se usan en operaciones booleanas o lógicas para identificar individualmente objetos de interés o para crear máscaras sobre regiones.

En muchos casos, una imagen binaria es el resultado de una segmentación por niveles de gris o de una segmentación por selección de un rango de color determinado. En otros casos, una imagen binaria es simplemente el resultado de una selección interactiva de regiones de interés, las cuales se utilizarán como máscaras de comparación o referencia.

El histograma de una imagen digital con niveles de grises en el rango de **[0,L-1]** es la función discreta:

$$h(r_k) = n_k$$

Donde $\mathbf{r_k}$ es el nivel de gris y $\mathbf{n_k}$ es el número de pixeles en la imagen que tiene el nivel de gris $\mathbf{r_k}$. En la práctica, es común normalizar el histograma dividiendo cada uno de los valores por el número total de pixeles en la imagen, lo cual se denota por **n**. Entonces, la forma normalizada del histograma es dado por:

$$p(r_k) = \frac{n_k}{n}$$

Para **k = 0,1,…,L-1**. En términos generales, **$p(r_k)$** da un estimado de la probabilidad de ocurrencia del nivel de gris **r_k**. La suma de todos los componentes de un histograma normalizado es igual a **1**. Para cualquier **r** que cumpla las condiciones antes mencionadas, se centrara la atención en las transformaciones de la forma:

$$s = T(r) \qquad 0 \leq r \leq 1$$

Eso produce un nivel **s** para cada valor del pixel **r** en la imagen original. Se asume que la función de la transformada **T(r)** cumple las siguientes condiciones:

a) **T(r)** es un valor unitario y monótono que se incrementa en el intervalo **0 ≤ r ≤ 1**.
b) **0 ≤ T(r) ≤ 1** para **0 ≤ r ≤ 1**.

El requerimiento en a) es que **T(r)** sea un valor unitario necesario para garantizar que la transformada inversa exista, y la condición monótona persevere el incremento del orden desde negro hasta el blanco en la imagen de salida. La función de la transformada que no es monótonamente creciente puede resultar en, por lo menos, una sección del rango de intensidad invertida, además produce algunos niveles de grises invertidos en la imagen de salida. Si bien esto puede ser un efecto deseable en algunos casos, eso no es lo que se está buscando. Por último, la condición b) garantiza que los niveles de gris de salida estarán en el mismo rango que los niveles de entrada. En la figura (29), se muestra un ejemplo de la función de transformada que obedece las dos condiciones. La transformada inversa desde **s** hacia **r** se representa de la siguiente manera:

$$r = T^{-1}(s) \qquad 0 \leq s \leq 1$$

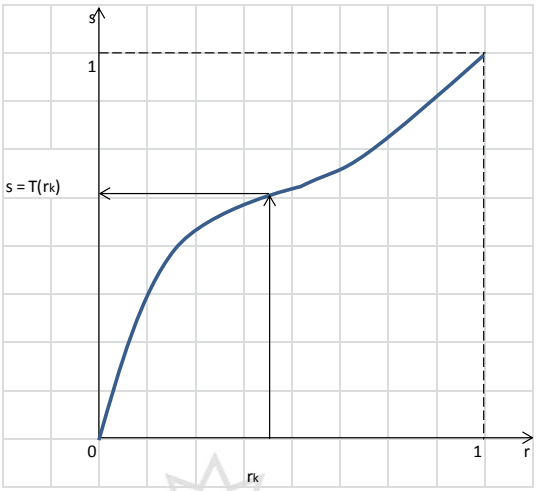


Figura (29) Función de transformada de nivel de gris con valor unitario y crecimiento monótono.

Elaboración: Los autores.

La ecualización del histograma entonces viene dado por:

$$s_k = T(r_k) = (L-1)\sum_{j=0}^{k} p_r(r_j) = \frac{L-1}{n}\sum_{j=0}^{k} n_j$$

Dado como ejemplo la imagen de código de barras en escala de grises capturada con la cámara del dispositivo móvil como se muestra en la figura (30) de tamaño 2560 x 1920 de 8 bits, se tomarán los primeros ocho valores de $\mathbf{r_k}$ mostrados en la tabla (2) para hallar la probabilidad de ocurrencia de cada uno de estos niveles, y así hallar el mejor umbral posible para realizar la binarización.

Figura (30) Imagen de código de barras en escala de grises capturada con la cámara del dispositivo móvil.
Elaboración: Los autores.

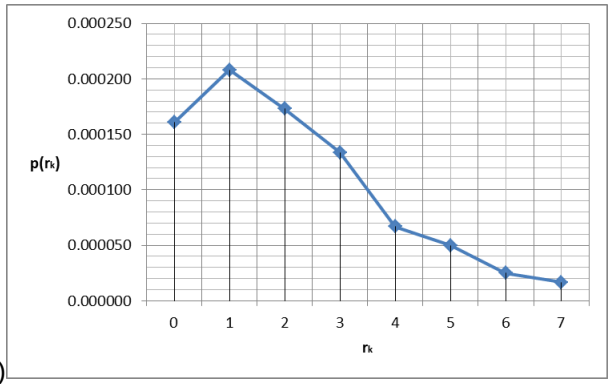
Tabla (2) Distribución de la intensidad y valores del histograma de la imagen de código de barras.

r_k	n_k	$p(r_k)$	$T(r_k)$	$p(s_k)$
0	790	0.000161	0.040824	0.000161
1	1023	0.000208	0.093689	0.000208
2	850	0.000173	0.137614	0.000173
3	656	0.000133	0.171514	0.000133
4	329	0.000067	0.188516	0.000067
5	245	0.00005	0.201176	0.00005
6	122	0.000025	0.207481	-
7	81	0.000016	0.211667	0.000041

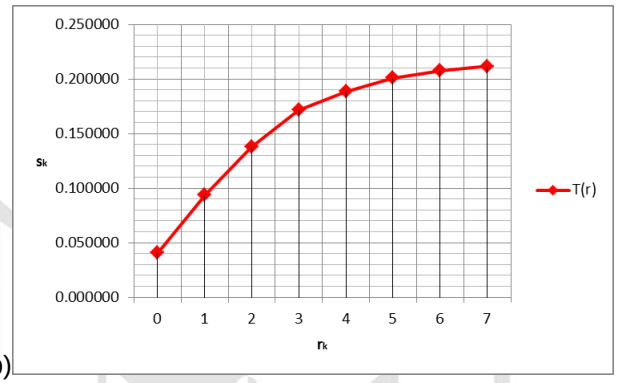
Elaboración: Los autores.

Para los valores hallados en la tabla (2), se grafica la función de la transformada T(r) y la ecualización del histograma tal y como se muestra en la figura (31).

a)



b)



c)

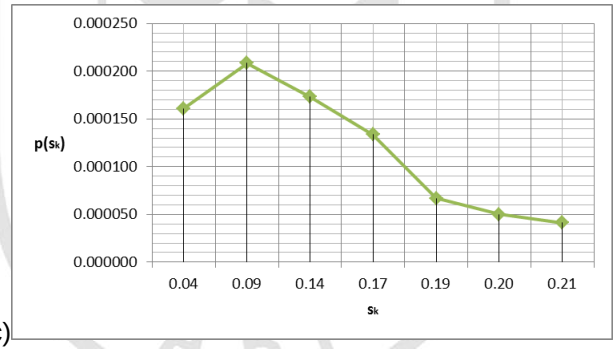


Figura (31) Representación de método basado en la forma del histograma (31.a) Histograma original. (31.b) Función de la transformada. (31.c) Histograma ecualizado.

Elaboración: Los autores.

Una vez obtenidas cada una de las probabilidades de ocurrencia para cada uno de los niveles de $\mathbf{r_k}$ con ayuda del programa de prueba y error Matlab®, se obtuvo que la mayor probabilidad de ocurrencia se daba en el valor $\mathbf{r_k = 118}$. Con dicho valor de umbral se comparan cada uno de los pixeles de la imagen, obteniendo la binarización de la misma como se muestra en la figura (32).

Figura (32) Binarización de la imagen de código de barras.

Elaboración: Los autores.

3.2 Diseño del proyecto

3.2.1 Diseño del sistema de reconocimiento de códigos de barras utilizando algoritmos de procesamiento digital de imágenes

En esta sección, detallaremos los pasos a seguir para diseñar el sistema de reconocimiento de códigos de barras mediante el uso de los algoritmos de procesamiento digital de imágenes, que utilizamos para hallar cada una de las especificaciones de diseño señaladas en la sección de análisis. Para esto, se realizará el cálculo de cada una de estas especificaciones en las imágenes de muestras, las cuales fueron capturadas en tres (03) ambientes y tiempos distintos, tal y como se muestran en las figuras (33), (34) y (35):

a) b) c)

Figura (33) Imágenes de código de barras tomadas en el exterior (Hora 10:00 ~ 12:00 hrs. día soleado). (33.a) Imagen de código de barras con aproximación al eje horizontal. (33.b) Imagen de código de barras con leve inclinación anti horaria. (33.c) Imagen de código de barras con leve inclinación horaria.

Elaboración: Los autores.

a) b) c)

Figura (34) Imágenes de códigos de barras tomadas en un ambiente bajo techo (Hora 15:00 ~ 17:00 hrs. día soleado). (34.a) Imagen de código de barras con aproximación al eje horizontal. (34.b) Imagen de código de barras con leve inclinación anti horaria. (34.c) Imagen de código de barras con leve inclinación horaria.

Elaboración: Los autores.

a) b) c)

Figura (35) Imágenes de código de barras tomadas de noche en presencia de luz artificial color blanco (Hora 21:00 ~ 23:00 hrs.). (35.a) Imagen de código de barras con aproximación al eje horizontal. (35.b) Imagen de código de barras con leve inclinación anti horaria. (35.c) Imagen de código de barras con leve inclinación horaria.

Elaboración: Los autores.

3.2.1.1 Detección de bordes de la imagen de código de barras mediante el operador de Canny

Desviación estándar (σ) del filtro gaussiano para la suavización de la imagen del código de barras

Como se mencionó en la sección de análisis, el objetivo de realizar el filtrado de la imagen es disminuir el ruido y suavizar la imagen. Es por ello que para seleccionar un valor de **σ** que se ajuste al diseño, se deberá tener en cuenta lo siguiente:

- Si el valor de **σ** es bajo, se sumará mayor proporción de ruido en la detección de bordes.
- Si el valor de **σ** es alto, se producirá mayor distorsión en los bordes detectados.

En tal sentido, tomaremos como referencia los valores **σ = 1**, **σ = 3**, **σ = 5** y **σ = 7**.

- Valor de desviación estándar **σ = 1**

$$C = 2\sqrt{2\sigma^2} = 2\sqrt{2(1)^2} = 2.8284$$

$$N \geq 3C \geq 3(2.8284) \geq 8.4853 \cong 9$$

$$x = \begin{bmatrix} -4 & -3 & -2 & -1 & 0 & 1 & 2 & 3 & 4 \\ -4 & -3 & -2 & -1 & 0 & 1 & 2 & 3 & 4 \\ -4 & -3 & -2 & -1 & 0 & 1 & 2 & 3 & 4 \\ -4 & -3 & -2 & -1 & 0 & 1 & 2 & 3 & 4 \\ -4 & -3 & -2 & -1 & 0 & 1 & 2 & 3 & 4 \\ -4 & -3 & -2 & -1 & 0 & 1 & 2 & 3 & 4 \\ -4 & -3 & -2 & -1 & 0 & 1 & 2 & 3 & 4 \\ -4 & -3 & -2 & -1 & 0 & 1 & 2 & 3 & 4 \\ -4 & -3 & -2 & -1 & 0 & 1 & 2 & 3 & 4 \end{bmatrix}$$

$$y = \begin{bmatrix} 4 & 4 & 4 & 4 & 4 & 4 & 4 & 4 & 4 \\ 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 \\ 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -2 & -2 & -2 & -2 & -2 & -2 & -2 & -2 & -2 \\ -3 & -3 & -3 & -3 & -3 & -3 & -3 & -3 & -3 \\ -4 & -4 & -4 & -4 & -4 & -4 & -4 & -4 & -4 \end{bmatrix}$$

$$h(x,y) = \frac{1}{6.2820} \begin{bmatrix} 0.0000 & 0.0000 & 0.0000 & 0.0002 & 0.0003 & 0.0002 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0001 & 0.0015 & 0.0067 & 0.0111 & 0.0067 & 0.0015 & 0.0001 & 0.0000 \\ 0.0000 & 0.0015 & 0.0183 & 0.0821 & 0.1353 & 0.0821 & 0.0183 & 0.0015 & 0.0000 \\ 0.0002 & 0.0067 & 0.0821 & 0.3679 & 0.6065 & 0.3679 & 0.0821 & 0.0067 & 0.0002 \\ 0.0003 & 0.0111 & 0.1353 & 0.6065 & 1 & 0.6065 & 0.1353 & 0.0111 & 0.0003 \\ 0.0002 & 0.0067 & 0.0821 & 0.3679 & 0.6065 & 0.3679 & 0.0821 & 0.0067 & 0.0002 \\ 0.0000 & 0.0015 & 0.0183 & 0.0821 & 0.1353 & 0.0821 & 0.0183 & 0.0015 & 0.0000 \\ 0.0000 & 0.0001 & 0.0015 & 0.0067 & 0.0111 & 0.0067 & 0.0015 & 0.0001 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.0002 & 0.0003 & 0.0002 & 0.0000 & 0.0000 & 0.0000 \end{bmatrix}$$

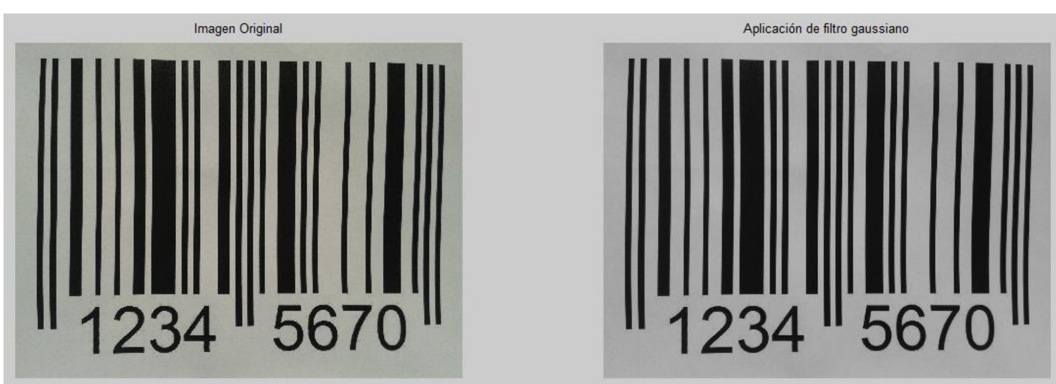
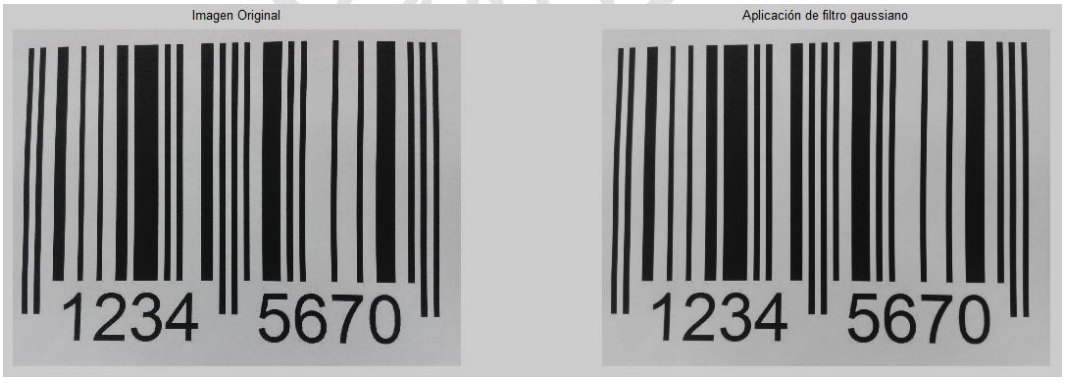


Figura (36) Comparación de las imágenes de código de barras antes y después de aplicar el filtro gaussiano con σ = 1 tomadas en el exterior.
Elaboración: Los autores.



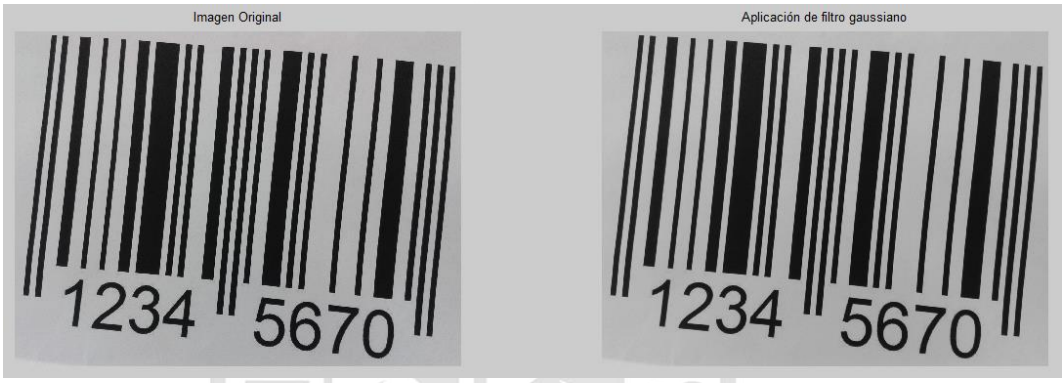


Figura (37) Comparación de las imágenes de código de barras antes y después de aplicar el filtro gaussiano con σ = 1 tomadas bajo techo.
Elaboración: Los autores.

Figura (38) Comparación de las imágenes de código de barras antes y después de aplicar el filtro gaussiano con σ = 1 tomadas de noche en presencia de luz artificial color blanco.
Elaboración: Los autores.

Como se puede apreciar, se difuminó en un mínimo el contorno de imágenes de código de barra; sin embargo, no se obtuvo una diferencia notable.

- Valor de desviación estándar **σ = 3**

$$C = 2\sqrt{2\sigma^2} = 2\sqrt{2(3)^2} = 8.4853$$

$$N \geq 3C \geq 3(8.4853) \geq 25.4558 \cong 25$$

$$x = \begin{bmatrix} -12 & -11 & -10 & \cdots & & -1 & 0 & 1 & & \cdots & 10 & 11 & 12 \\ -12 & -11 & & & & & \vdots & & & & & 11 & 12 \\ -12 & & & & -2 & -1 & 0 & 1 & 2 & & & & 12 \\ & & & -3 & -2 & -1 & 0 & 1 & 2 & 3 & & & \\ \vdots & & \cdots & -3 & -2 & -1 & 0 & 1 & 2 & 3 & \cdots & & \vdots \\ & & & -3 & -2 & -1 & 0 & 1 & 2 & 3 & & & \\ -12 & & & & -2 & -1 & 0 & 1 & 2 & & & & 12 \\ -12 & -11 & & & & & \vdots & & & & & 11 & 12 \\ -12 & -11 & -10 & \cdots & & -1 & 0 & 1 & & \cdots & 10 & 11 & 12 \end{bmatrix}$$

$$y = \begin{bmatrix} 12 & 12 & 12 & & \cdots & & & 12 & 12 & 12 \\ 11 & 11 & & & & & & & 11 & 11 \\ 10 & & & & \vdots & & & & & 10 \\ \vdots & & & 3 & 3 & 3 & & & & \vdots \\ & & 2 & 2 & 2 & 2 & 2 & & & \\ 1 & & 1 & 1 & 1 & 1 & 1 & & & 1 \\ 0 & \cdots & 0 & 0 & 0 & 0 & 0 & \cdots & & 0 \\ -1 & & -1 & -1 & -1 & -1 & -1 & & & -1 \\ & & -2 & -2 & -2 & -2 & -2 & & & \\ \vdots & & & -3 & -3 & -3 & & & & \vdots \\ -10 & & & & \vdots & & & & & -10 \\ -11 & -11 & & & & & & & -11 & -11 \\ -12 & -12 & -12 & & \cdots & & & -12 & -12 & -12 \end{bmatrix}$$

$$h(x,y) = \frac{1}{18.8692} \begin{vmatrix} 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0001 & 0.0002 & 0.0003 & 0.0002 & 0.0001 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.0002 & 0.0006 & 0.0013 & 0.0021 & 0.0025 & 0.0021 & 0.0013 & 0.0006 & 0.0002 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0002 & 0.0011 & 0.0035 & 0.0080 & 0.0131 & 0.0155 & 0.0131 & 0.0080 & 0.0035 & 0.0011 & 0.0002 & 0.0000 & 0.0000 \\ 0.0000 & 0.0002 & 0.0011 & 0.0048 & 0.0155 & 0.0357 & 0.0588 & 0.0695 & 0.0588 & 0.0357 & 0.0155 & 0.0048 & 0.0011 & 0.0002 & 0.0000 \\ 0.0000 & 0.0006 & 0.0035 & 0.0155 & 0.0498 & 0.1146 & 0.1889 & 0.2231 & 0.1889 & 0.1146 & 0.0498 & 0.0155 & 0.0035 & 0.0006 & 0.0000 \\ 0.0001 & 0.0013 & 0.0080 & 0.0357 & 0.1146 & 0.2636 & 0.4346 & 0.5134 & 0.4346 & 0.2636 & 0.1146 & 0.0357 & 0.0080 & 0.0013 & 0.0001 \\ 0.0002 & 0.0021 & 0.0131 & 0.0588 & 0.1889 & 0.4346 & 0.7165 & 0.8465 & 0.7165 & 0.4346 & 0.1889 & 0.0588 & 0.0131 & 0.0021 & 0.0002 \\ 0.0003 & 0.0025 & 0.0155 & 0.0695 & 0.2231 & 0.5134 & 0.8465 & 1.0000 & 0.8465 & 0.5134 & 0.2231 & 0.0695 & 0.0155 & 0.0025 & 0.0003 \\ 0.0002 & 0.0021 & 0.0131 & 0.0588 & 0.1889 & 0.4346 & 0.7165 & 0.8465 & 0.7165 & 0.4346 & 0.1889 & 0.0588 & 0.0131 & 0.0021 & 0.0002 \\ 0.0001 & 0.0013 & 0.0080 & 0.0357 & 0.1146 & 0.2636 & 0.4346 & 0.5134 & 0.4346 & 0.2636 & 0.1146 & 0.0357 & 0.0080 & 0.0013 & 0.0001 \\ 0.0000 & 0.0006 & 0.0035 & 0.0155 & 0.0498 & 0.1146 & 0.1889 & 0.2231 & 0.1889 & 0.1146 & 0.0498 & 0.0155 & 0.0035 & 0.0006 & 0.0000 \\ 0.0000 & 0.0002 & 0.0011 & 0.0048 & 0.0155 & 0.0357 & 0.0588 & 0.0695 & 0.0588 & 0.0357 & 0.0155 & 0.0048 & 0.0011 & 0.0002 & 0.0000 \\ 0.0000 & 0.0000 & 0.0002 & 0.0011 & 0.0035 & 0.0080 & 0.0131 & 0.0155 & 0.0131 & 0.0080 & 0.0035 & 0.0011 & 0.0002 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.0002 & 0.0006 & 0.0013 & 0.0021 & 0.0025 & 0.0021 & 0.0013 & 0.0006 & 0.0002 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0001 & 0.0002 & 0.0003 & 0.0002 & 0.0001 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \end{vmatrix}$$

Figura (39) Comparación de las imágenes de código de barras antes y después de aplicar el filtro gaussiano con σ = 3 tomadas en el exterior.

Elaboración: Los autores.

Figura (40) Comparación de las imágenes de código de barras antes y después de aplicar el filtro gaussiano con σ = 3 tomadas bajo techo.

Elaboración: Los autores.

Figura (41) Comparación de las imágenes de código de barras antes y después de aplicar el filtro gaussiano con σ = 3 tomadas de noche en presencia de luz artificial color blanco.

Elaboración: Los autores.

Para este caso, los bordes de las imágenes de código de barras se difuminaron en una mayor proporción que en el caso anterior. La discontinuidad en forma de pixeles en los bordes se suavizó brindando una diferencia notable.

- Valor de desviación estándar **σ = 5**

$$C = 2\sqrt{2\sigma^2} = 2\sqrt{2(5)^2} = 14.1421$$

$$N \geq 3C \geq 3(14.1421) \geq 42.4264 \cong 43$$

$$x = \begin{bmatrix} -21 & -20 & -19 & \cdots & & -1 & 0 & 1 & & \cdots & 19 & 20 & 21 \\ -21 & -20 & & & & & \vdots & & & & & 20 & 21 \\ -21 & & & & -2 & -1 & 0 & 1 & 2 & & & & 21 \\ & & & -3 & -2 & -1 & 0 & 1 & 2 & 3 & & & \\ \vdots & & \cdots & -3 & -2 & -1 & 0 & 1 & 2 & 3 & \cdots & & \vdots \\ & & & -3 & -2 & -1 & 0 & 1 & 2 & 3 & & & \\ -21 & & & & -2 & -1 & 0 & 1 & 2 & & & & 21 \\ -21 & -20 & & & & & \vdots & & & & & 20 & 21 \\ -21 & -20 & -19 & \cdots & & -1 & 0 & 1 & & \cdots & 19 & 20 & 21 \end{bmatrix}$$

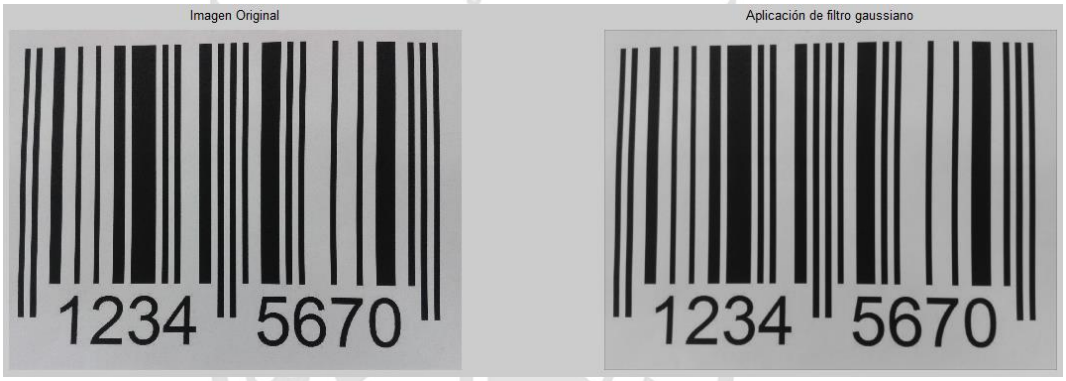
$$y = \begin{bmatrix} 21 & 21 & 21 & & \cdots & & 21 & 21 & 21 \\ 20 & 20 & & & & & & 20 & 20 \\ 19 & & & & \vdots & & & & 19 \\ \vdots & & & 3 & 3 & 3 & & & \vdots \\ & & 2 & 2 & 2 & 2 & 2 & & \\ 1 & & 1 & 1 & 1 & 1 & 1 & & 1 \\ 0 & \cdots & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ -1 & & -1 & -1 & -1 & -1 & -1 & & -1 \\ & & -2 & -2 & -2 & -2 & -2 & & \\ \vdots & & & -3 & -3 & -3 & & & \vdots \\ -19 & & & & \vdots & & & & -19 \\ -20 & -20 & & & & & & -20 & -20 \\ -21 & -21 & -21 & & \cdots & & -21 & -21 & -21 \end{bmatrix}$$

$$h(x,y) = \frac{1}{31.4120}$$

0.0000	0.0000	0.0000	0.0000	0.0000	0.0001	0.0001	0.0002	0.0003	0.0003	0.0003	0.0002	0.0001	0.0001	0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000	0.0001	0.0003	0.0007	0.0011	0.0015	0.0017	0.0015	0.0011	0.0007	0.0003	0.0001	0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	0.0001	0.0002	0.0007	0.0015	0.0030	0.0050	0.0067	0.0074	0.0067	0.0050	0.0030	0.0015	0.0007	0.0002	0.0001	0.0000	0.0000
0.0000	0.0000	0.0002	0.0007	0.0022	0.0055	0.0111	0.0183	0.0247	0.0273	0.0247	0.0183	0.0111	0.0055	0.0022	0.0007	0.0002	0.0000	0.0000
0.0000	0.0001	0.0006	0.0022	0.0067	0.0166	0.0334	0.0550	0.0743	0.0821	0.0743	0.0550	0.0334	0.0166	0.0067	0.0022	0.0006	0.0001	0.0000
0.0001	0.0003	0.0015	0.0055	0.0166	0.0408	0.0821	0.1352	0.1827	0.2019	0.1827	0.1352	0.0821	0.0408	0.0166	0.0055	0.0015	0.0003	0.0001
0.0001	0.0007	0.0030	0.0111	0.0334	0.0821	0.1653	0.2725	0.3679	0.4066	0.3679	0.2725	0.1653	0.0821	0.0334	0.0111	0.0030	0.0007	0.0001
0.0002	0.0011	0.0050	0.0183	0.0550	0.1353	0.2725	0.4493	0.6065	0.6703	0.6065	0.4493	0.2725	0.1353	0.0550	0.0183	0.0050	0.0011	0.0002
0.0003	0.0015	0.0067	0.0247	0.0743	0.1827	0.3679	0.6065	0.8187	0.9048	0.8187	0.6065	0.3679	0.1827	0.0743	0.0247	0.0067	0.0015	0.0003
0.0003	0.0017	0.0074	0.0273	0.0821	0.2019	0.4066	0.6703	0.9048	1.0000	0.9048	0.6703	0.4066	0.2019	0.0821	0.0273	0.0074	0.0017	0.0003
0.0003	0.0015	0.0067	0.0247	0.0743	0.1827	0.3679	0.6065	0.8187	0.9048	0.8187	0.6065	0.3679	0.1827	0.0743	0.0247	0.0067	0.0015	0.0003
0.0002	0.0011	0.0050	0.0183	0.0550	0.1353	0.2725	0.4493	0.6065	0.6703	0.6065	0.4493	0.2725	0.1353	0.0550	0.0183	0.0050	0.0011	0.0002
0.0001	0.0007	0.0030	0.0111	0.0334	0.0821	0.1653	0.2725	0.3679	0.4066	0.3679	0.2725	0.1653	0.0821	0.0334	0.0111	0.0030	0.0007	0.0001
0.0001	0.0003	0.0015	0.0055	0.0166	0.0408	0.0821	0.1352	0.1827	0.2019	0.1827	0.1352	0.0821	0.0408	0.0166	0.0055	0.0015	0.0003	0.0001
0.0000	0.0001	0.0006	0.0022	0.0067	0.0166	0.0334	0.0550	0.0743	0.0821	0.0743	0.0550	0.0334	0.0166	0.0067	0.0022	0.0006	0.0001	0.0000
0.0000	0.0000	0.0002	0.0007	0.0022	0.0055	0.0111	0.0183	0.0247	0.0273	0.0247	0.0183	0.0111	0.0055	0.0022	0.0007	0.0002	0.0000	0.0000
0.0000	0.0000	0.0001	0.0002	0.0007	0.0015	0.0030	0.0050	0.0067	0.0074	0.0067	0.0050	0.0030	0.0015	0.0007	0.0002	0.0001	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000	0.0001	0.0003	0.0007	0.0011	0.0015	0.0017	0.0015	0.0011	0.0007	0.0003	0.0001	0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.0001	0.0001	0.0002	0.0003	0.0003	0.0003	0.0002	0.0001	0.0001	0.0000	0.0000	0.0000	0.0000	0.0000

Figura (42) Comparación de las imágenes de código de barras antes y después de aplicar el filtro gaussiano con σ = 5 tomadas en el exterior.

Elaboración: Los autores.



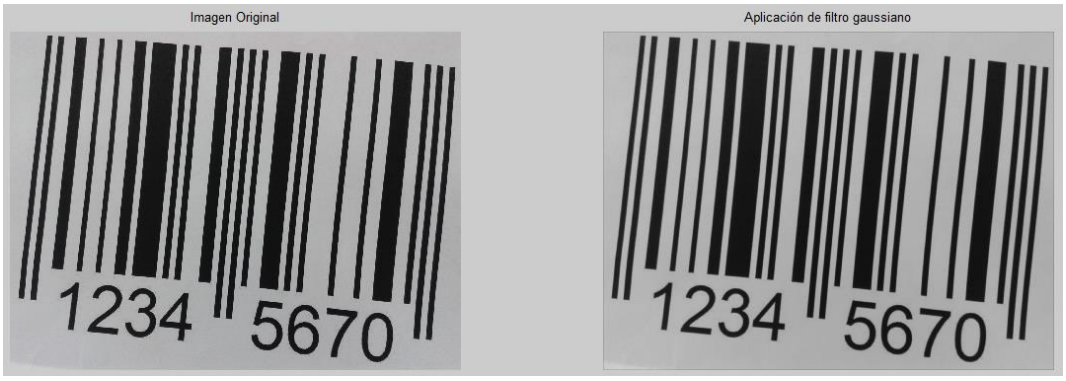


Figura (43) Comparación de las imágenes de código de barras antes y después de aplicar el filtro gaussiano con σ = 5 tomadas bajo techo.

Elaboración: Los autores.

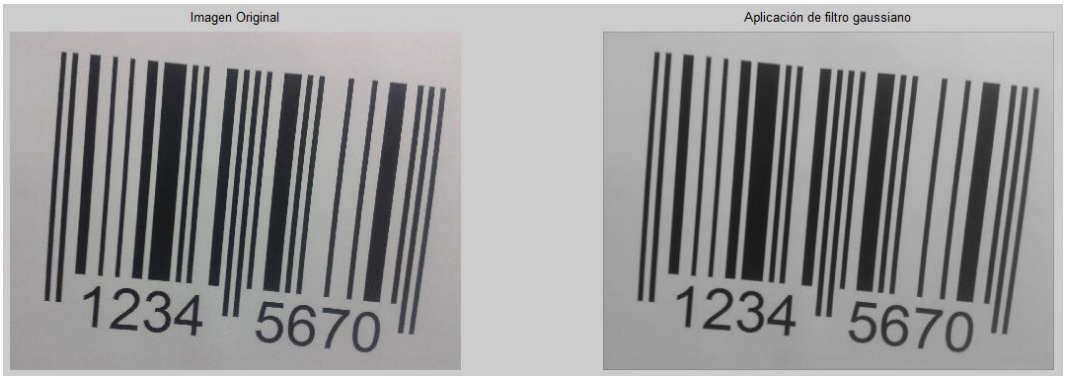


Figura (44) Comparación de las imágenes de código de barras antes y después de aplicar el filtro gaussiano con σ = 5 tomadas de noche en presencia de luz artificial color blanco.
Elaboración: Los autores.

Al igual que el caso anterior, los bordes de las imágenes de código de barras se difuminaron en una mayor proporción, lo cual produce una mayor resolución y esto facilitaría la decodificación de la misma. La discontinuidad en forma de pixeles en los bordes se suavizo brindando una diferencia notable.

- Valor de desviación estándar **σ = 7**

$$C = 2\sqrt{2\sigma^2} = 2\sqrt{2(7)^2} = 19.7990$$

$$N \geq 3C \geq 3(19.7990) \geq 59.3970 \cong 59$$

$$x = \begin{bmatrix} -29 & -28 & -27 & \cdots & & -1 & 0 & 1 & & \cdots & 27 & 28 & 29 \\ -29 & -28 & & & & & \vdots & & & & & 28 & 29 \\ -29 & & & & -2 & -1 & 0 & 1 & 2 & & & & 29 \\ & & & -3 & -2 & -1 & 0 & 1 & 2 & 3 & & & \\ \vdots & & \cdots & -3 & -2 & -1 & 0 & 1 & 2 & 3 & \cdots & & \vdots \\ & & & -3 & -2 & -1 & 0 & 1 & 2 & 3 & & & \\ -29 & & & & -2 & -1 & 0 & 1 & 2 & & & & 29 \\ -29 & -28 & & & & & \vdots & & & & & 28 & 29 \\ -29 & -28 & -27 & \cdots & & -1 & 0 & 1 & & \cdots & 27 & 28 & 29 \end{bmatrix}$$

$$y = \begin{bmatrix} 29 & 29 & 29 & & \cdots & & 29 & 29 & 29 \\ 28 & 28 & & & & & & 28 & 28 \\ 27 & & & & \vdots & & & & 27 \\ \vdots & & & 3 & 3 & 3 & & & \vdots \\ & & 2 & 2 & 2 & 2 & 2 & & \\ 1 & & 1 & 1 & 1 & 1 & 1 & & 1 \\ 0 & \cdots & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ -1 & & -1 & -1 & -1 & -1 & -1 & & -1 \\ & & -2 & -2 & -2 & -2 & -2 & & \\ \vdots & & & -3 & -3 & -3 & & & \vdots \\ -27 & & & & \vdots & & & & -27 \\ -28 & -28 & & & & & & -28 & -28 \\ -29 & -29 & -29 & & \cdots & & -29 & -29 & -29 \end{bmatrix}$$

$$h(x,y) = \frac{1}{43.9804} \begin{bmatrix} 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0001 & 0.0001 & 0.0001 & 0.0002 & 0.0002 & 0.0002 & 0.0001 & 0.0001 & 0.0001 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0001 & 0.0001 & 0.0003 & 0.0004 & 0.0006 & 0.0007 & 0.0008 & 0.0007 & 0.0006 & 0.0004 & 0.0003 & 0.0001 & 0.0001 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0001 & 0.0002 & 0.0005 & 0.0010 & 0.0016 & 0.0023 & 0.0028 & 0.0031 & 0.0028 & 0.0023 & 0.0016 & 0.0010 & 0.0005 & 0.0002 & 0.0001 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.0001 & 0.0003 & 0.0008 & 0.0017 & 0.0033 & 0.0054 & 0.0078 & 0.0096 & 0.0103 & 0.0096 & 0.0078 & 0.0054 & 0.0033 & 0.0017 & 0.0008 & 0.0003 & 0.0001 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0001 & 0.0003 & 0.0009 & 0.0023 & 0.0051 & 0.0096 & 0.0159 & 0.0227 & 0.0281 & 0.0302 & 0.0281 & 0.0227 & 0.0159 & 0.0096 & 0.0051 & 0.0023 & 0.0009 & 0.0003 & 0.0001 & 0.0000 & 0.0000 \\ 0.0000 & 0.0001 & 0.0002 & 0.0008 & 0.0023 & 0.0058 & 0.0128 & 0.0244 & 0.0402 & 0.0574 & 0.7120 & 0.0764 & 0.7120 & 0.0574 & 0.0402 & 0.0244 & 0.0128 & 0.0058 & 0.0023 & 0.0008 & 0.0002 & 0.0001 & 0.0000 \\ 0.0000 & 0.0001 & 0.0005 & 0.0017 & 0.0051 & 0.0128 & 0.0281 & 0.0535 & 0.0882 & 0.1260 & 0.1561 & 0.1677 & 0.1561 & 0.1260 & 0.0882 & 0.0535 & 0.0281 & 0.0128 & 0.0051 & 0.0017 & 0.0005 & 0.0001 & 0.0000 \\ 0.0001 & 0.0003 & 0.0010 & 0.0033 & 0.0096 & 0.0244 & 0.0535 & 0.1017 & 0.1677 & 0.2397 & 0.2969 & 0.3189 & 0.2969 & 0.2397 & 0.1677 & 0.1017 & 0.0535 & 0.0244 & 0.0096 & 0.0033 & 0.0010 & 0.0003 & 0.0001 \\ 0.0001 & 0.0004 & 0.0016 & 0.0054 & 0.0159 & 0.0402 & 0.0882 & 0.1677 & 0.2765 & 0.3951 & 0.4895 & 0.5258 & 0.4895 & 0.3951 & 0.2765 & 0.1677 & 0.0882 & 0.0402 & 0.0159 & 0.0054 & 0.0016 & 0.0004 & 0.0001 \\ 0.0001 & 0.0006 & 0.0023 & 0.0078 & 0.0227 & 0.0574 & 0.1260 & 0.2397 & 0.3951 & 0.5647 & 0.6997 & 0.7752 & 0.6997 & 0.5647 & 0.3951 & 0.2397 & 0.1260 & 0.0574 & 0.0227 & 0.0078 & 0.0023 & 0.0006 & 0.0001 \\ 0.0002 & 0.0007 & 0.0028 & 0.0096 & 0.0281 & 0.0712 & 0.1561 & 0.2969 & 0.4895 & 0.6997 & 0.8669 & 0.9311 & 0.8669 & 0.6997 & 0.4895 & 0.2969 & 0.1561 & 0.0712 & 0.0281 & 0.0096 & 0.0028 & 0.0007 & 0.0002 \\ 0.0002 & 0.0008 & 0.0031 & 0.0103 & 0.0302 & 0.0764 & 0.1677 & 0.3189 & 0.5258 & 0.7515 & 0.9311 & 1.0000 & 0.9311 & 0.7515 & 0.5258 & 0.3189 & 0.1677 & 0.0764 & 0.0302 & 0.0103 & 0.0031 & 0.0008 & 0.0002 \\ 0.0002 & 0.0007 & 0.0028 & 0.0096 & 0.0281 & 0.0712 & 0.1561 & 0.2969 & 0.4895 & 0.6997 & 0.8669 & 0.9311 & 0.8669 & 0.6997 & 0.4895 & 0.2969 & 0.1561 & 0.0712 & 0.0281 & 0.0096 & 0.0028 & 0.0007 & 0.0002 \\ 0.0001 & 0.0006 & 0.0023 & 0.0078 & 0.0227 & 0.0574 & 0.1260 & 0.2397 & 0.3951 & 0.5647 & 0.6997 & 0.7752 & 0.6997 & 0.5647 & 0.3951 & 0.2397 & 0.1260 & 0.0574 & 0.0227 & 0.0078 & 0.0023 & 0.0006 & 0.0001 \\ 0.0001 & 0.0004 & 0.0016 & 0.0054 & 0.0159 & 0.0402 & 0.0882 & 0.1677 & 0.2765 & 0.3951 & 0.4895 & 0.5258 & 0.4895 & 0.3951 & 0.2765 & 0.1677 & 0.0882 & 0.0402 & 0.0159 & 0.0054 & 0.0016 & 0.0004 & 0.0001 \\ 0.0001 & 0.0003 & 0.0010 & 0.0033 & 0.0096 & 0.0244 & 0.0535 & 0.1017 & 0.1677 & 0.2397 & 0.2969 & 0.3189 & 0.2969 & 0.2397 & 0.1677 & 0.1017 & 0.0535 & 0.0244 & 0.0096 & 0.0033 & 0.0010 & 0.0003 & 0.0001 \\ 0.0000 & 0.0001 & 0.0005 & 0.0017 & 0.0051 & 0.0128 & 0.0281 & 0.0535 & 0.0882 & 0.1260 & 0.1561 & 0.1677 & 0.1561 & 0.1260 & 0.0882 & 0.0535 & 0.0281 & 0.0128 & 0.0051 & 0.0017 & 0.0005 & 0.0001 & 0.0000 \\ 0.0000 & 0.0001 & 0.0002 & 0.0008 & 0.0023 & 0.0058 & 0.0128 & 0.0244 & 0.0402 & 0.0574 & 0.7120 & 0.0764 & 0.7120 & 0.0574 & 0.0402 & 0.0244 & 0.0128 & 0.0058 & 0.0023 & 0.0008 & 0.0002 & 0.0001 & 0.0000 \\ 0.0000 & 0.0000 & 0.0001 & 0.0003 & 0.0009 & 0.0023 & 0.0051 & 0.0096 & 0.0159 & 0.0227 & 0.0281 & 0.0302 & 0.0281 & 0.0227 & 0.0159 & 0.0096 & 0.0051 & 0.0023 & 0.0009 & 0.0003 & 0.0001 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.0001 & 0.0003 & 0.0008 & 0.0017 & 0.0033 & 0.0054 & 0.0078 & 0.0096 & 0.0103 & 0.0096 & 0.0078 & 0.0054 & 0.0033 & 0.0017 & 0.0008 & 0.0003 & 0.0001 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0001 & 0.0002 & 0.0005 & 0.0010 & 0.0016 & 0.0023 & 0.0028 & 0.0031 & 0.0028 & 0.0023 & 0.0016 & 0.0010 & 0.0005 & 0.0002 & 0.0001 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0001 & 0.0001 & 0.0003 & 0.0004 & 0.0006 & 0.0007 & 0.0008 & 0.0007 & 0.0006 & 0.0004 & 0.0003 & 0.0001 & 0.0001 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0001 & 0.0001 & 0.0001 & 0.0002 & 0.0002 & 0.0002 & 0.0001 & 0.0001 & 0.0001 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \end{bmatrix}$$

Figura (45) Comparación de las imágenes de código de barras antes y después de aplicar el filtro gaussiano con σ = 7 tomadas en el exterior.
Elaboración: Los autores.

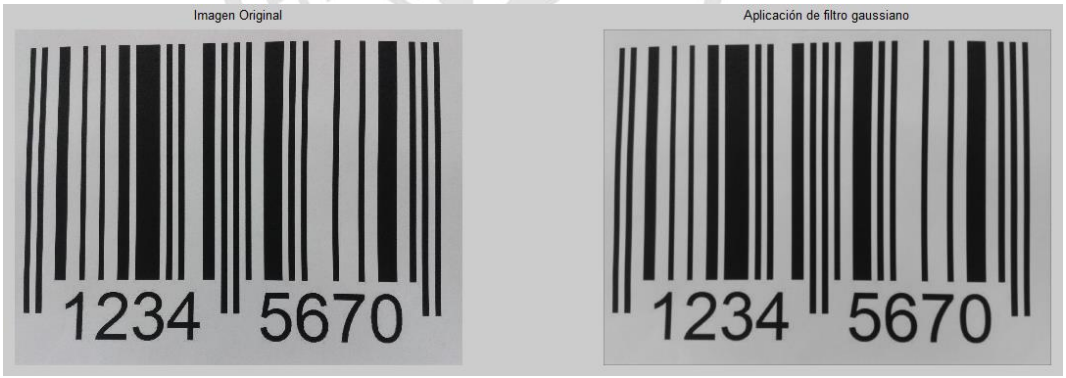


Figura (46) Comparación de las imágenes de código de barras antes y después de aplicar el filtro gaussiano con σ = 7 tomadas bajo techo.
Elaboración: Los autores.

Figura (47) Comparación de las imágenes de código de barras antes y después de aplicar el filtro gaussiano con σ = 7 tomadas de noche en presencia de luz artificial color blanco.
Elaboración: Los autores.

Como se puede apreciar, los bordes se difuminaron en mayor proporción lo cual podría producir errores al momento de decodificar la imagen de código de barras. Al momento de aplicar el operador Canny, este encontrará bordes inexistentes que afectarían el producto final.

Como se pudo apreciar, anteriormente, en las imágenes, se llega a la conclusión de que la desviación estándar del filtro gaussiano sigma (**σ**) que se seleccionará como especificación de diseño para nuestro proyecto es el valor de **σ = 5**, ya que brinda el grado óptimo de suavización de la imagen requerido en el proceso de la detección de bordes por medio del operador Canny.

Magnitud y dirección de la gradiente de los bordes de la imagen de código de barras

Para hallar la magnitud o puntos más fuertes de los bordes y la dirección de los mismos se tomarán como ejemplo una imagen de cada grupo de imágenes capturadas con la cámara del dispositivo móvil; sin embargo, se realizarán los resultados para cada una de estas, puesto que la imagen de código de barras es de 2560 x 1920 pixeles, el cálculo para cada uno de estos es muy extenso por lo cual se elaborará para una sección de la imagen y el resto será realizado mediante el programa de prueba y error Matlab®.

a) 

b) 

Figura (48) Selección de área para realizar el cálculo de la magnitud y dirección de los bordes del código de barras (48.a) Imagen de código de barras con aproximación al eje horizontal tomada en el exterior (48.b) Sección de la imagen de código de barras a evaluar.
Elaboración: Los autores.

En la figura (48) se muestra la sección de la imagen de código de barras a evaluar para obtener la magnitud y la dirección de la gradiente, para ello se obtiene la matriz de pixeles de dicha sección como se muestra a continuación:

214	214	214	215	65	21	11	0	0	0	0	0	0	13	64	215
214	214	215	215	65	24	13	0	0	0	0	0	0	12	63	215
214	215	215	215	64	22	12	0	0	0	0	0	0	15	62	215
215	215	215	215	65	22	11	0	0	0	0	0	0	15	62	215
215	215	215	215	65	21	13	0	0	0	0	0	0	15	62	215
215	215	215	63	65	23	14	0	0	0	0	0	0	12	62	215
215	215	215	65	26	23	13	0	0	0	0	0	11	11	64	215
215	215	215	63	23	16	18	0	0	0	0	0	12	10	64	215
215	215	215	215	23	17	0	0	0	0	0	0	11	10	64	215
215	215	215	64	22	16	0	0	0	0	0	0	12	9	35	215
215	215	215	65	21	12	0	0	0	0	0	0	14	9	32	215
215	215	215	65	23	11	0	0	0	0	0	0	13	31	30	215
215	215	215	63	23	13	0	0	0	0	0	0	15	36	215	215
215	215	215	62	23	13	0	0	0	0	0	0	13	36	215	215
215	215	215	63	22	13	0	0	0	0	0	0	12	34	215	215
215	215	215	64	21	12	0	0	0	0	0	0	14	35	215	215
215	215	215	61	21	15	0	0	0	0	0	0	14	35	215	215
215	215	215	62	22	12	0	0	0	0	0	0	12	33	215	215

$$|G(x,y)| = |(65 + 2 * 23 + 14) - (65 + 2 * 22 + 11)| + |(11 + 2 * 13 + 14) - (65 + 2 * 65 + 65)| = 214$$

$$\alpha(x,y) = \tan^{-1}\left(\frac{(11 + 2 * 13 + 14) - (65 + 2 * 65 + 65)}{(65 + 2 * 23 + 14) - (65 + 2 * 22 + 11)}\right) = 88.63°$$

Aplicando dicho procedimiento en la imagen de código de barras de la figura (47.a), nos dará como resultado las imágenes de la figura (49).

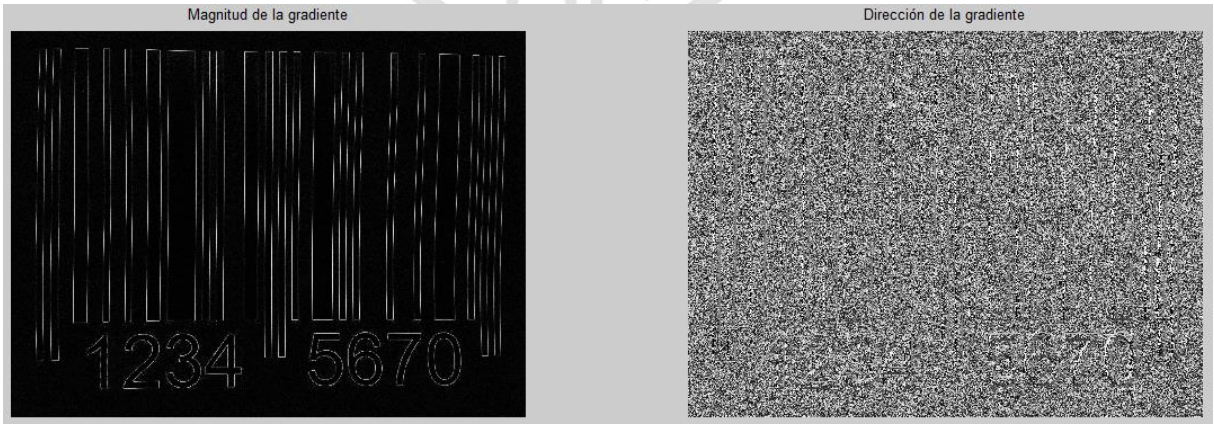


Figura (49) Representación de la magnitud y dirección de la gradiente de los bordes de la imagen de código de barras.

Elaboración: Los autores.

Del mismo modo, se presenta a continuación el mismo procedimiento para las demás imágenes:

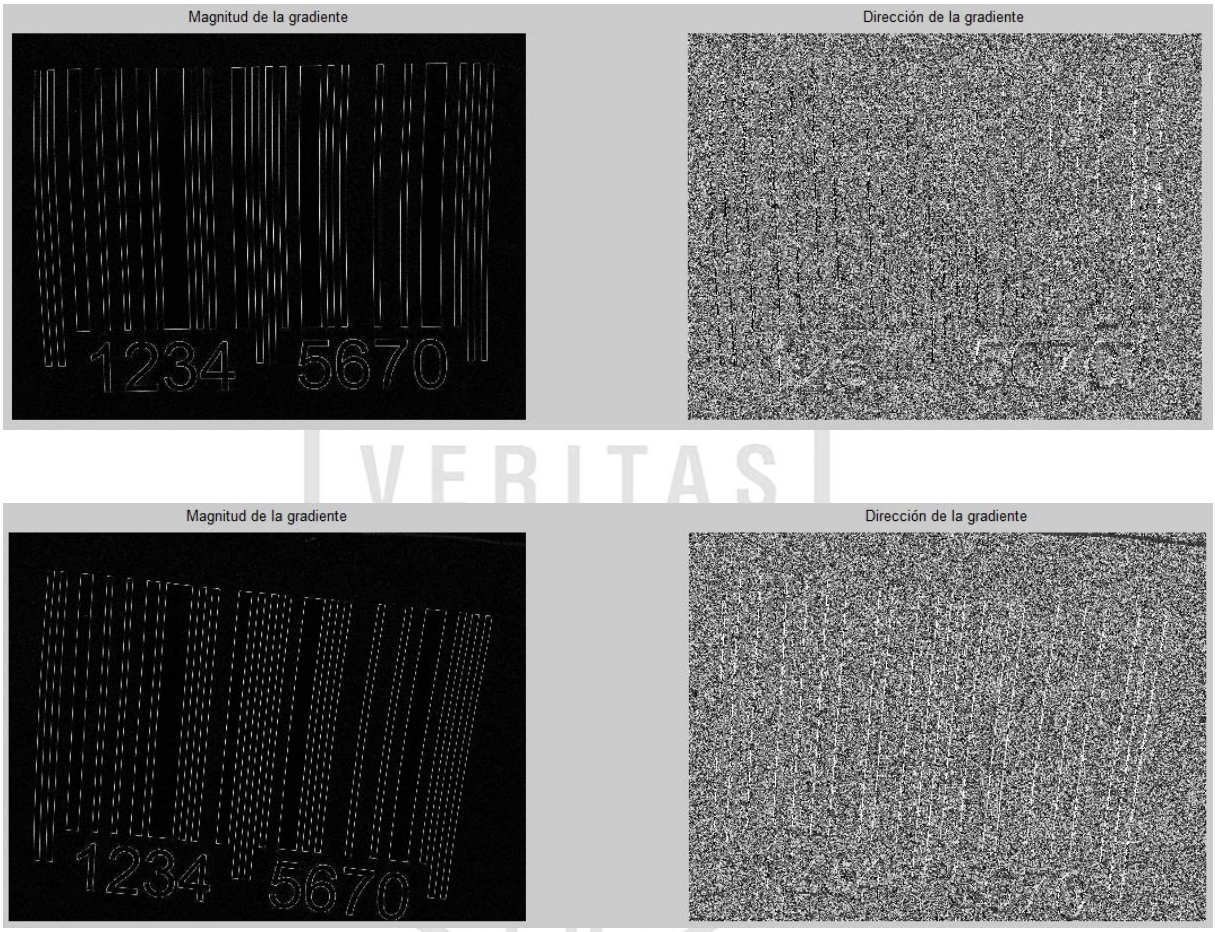


Figura (50) Representación de la magnitud y dirección de la gradiente de los bordes de las imágenes de código de barras tomadas en el exterior.
Elaboración: Los autores.

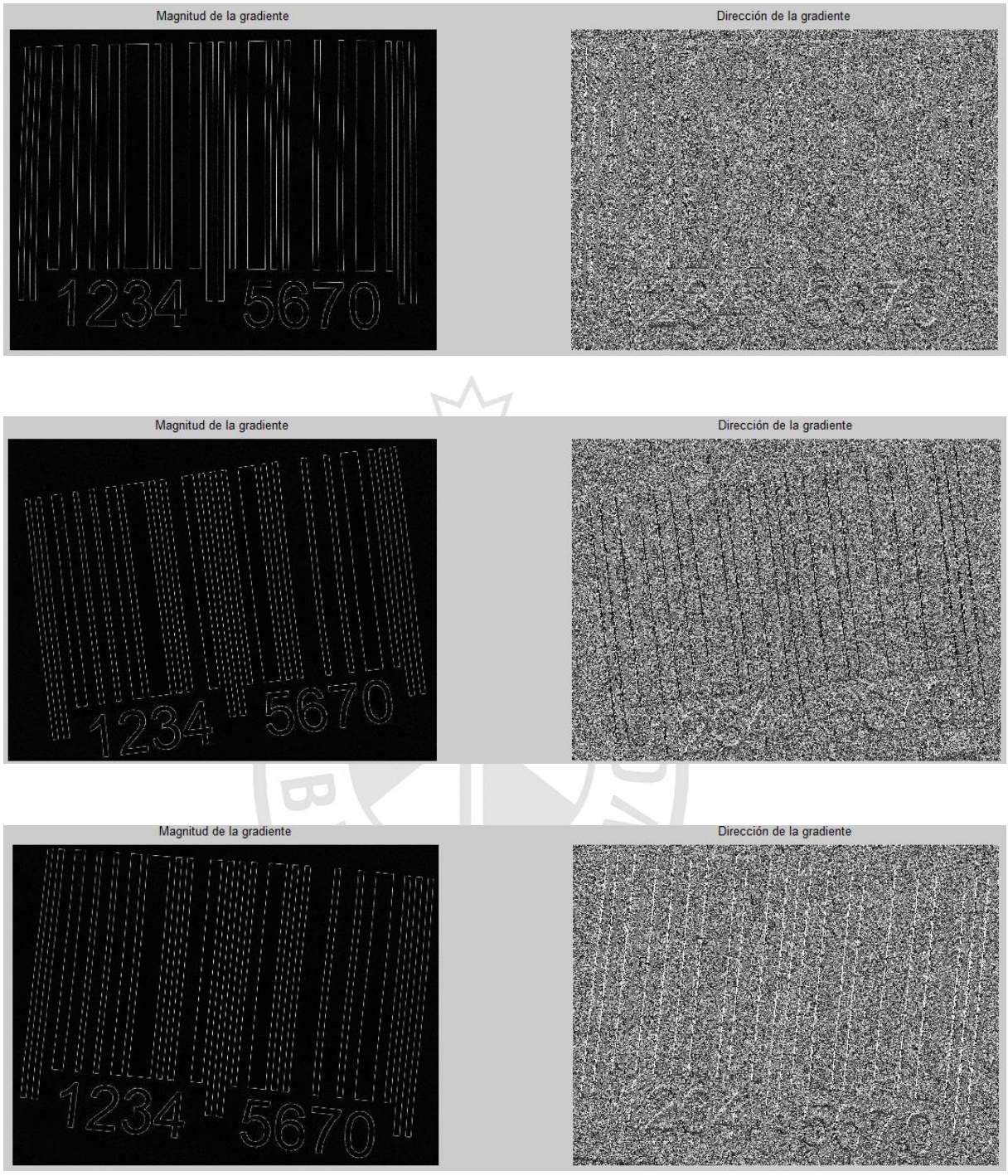


Figura (51) Representación de la magnitud y dirección de la gradiente de los bordes de las imágenes de código de barras tomadas en un ambiente bajo techo. Elaboración: Los autores.

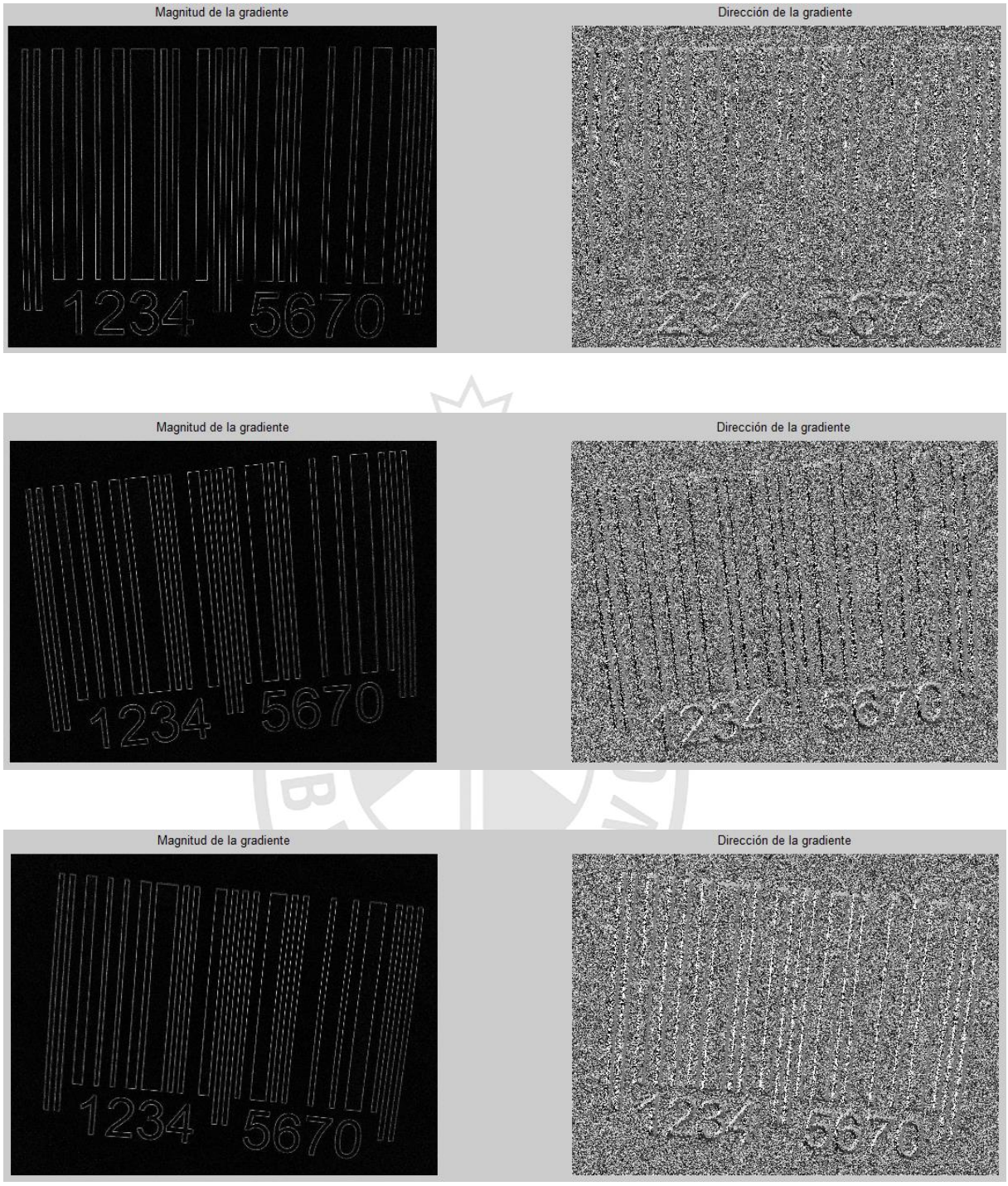


Figura (52) Representación de la magnitud y dirección de la gradiente de los bordes de las imágenes de código de barras tomadas de noche en presencia de luz artificial color blanco.

Elaboración: Los autores.

Selección de umbrales para la detección de bordes de la imagen de código de barras

Tal y como se mencionó en la etapa de análisis, el operador Canny requiere de dos (02) umbrales para la detección de los bordes: uno superior y uno inferior. Para nuestro caso es específico, ya que se solo se quieren detectar los bordes sin importan si son débiles o fuertes, solo utilizaremos el umbral inferior, lo cual indica que valores menos a este serán suprimidos. Cabe resaltar que el umbral menor se encuentra entre los valores de **1 > μ > 0**.

En tal sentido y mediante el uso del programa de prueba y error Matlab®, se testeará el umbral menor con los valores **μ = 0.1**, **μ = 0.5** y **μ = 0.9** con la imagen de código de barras de la figura (53). Una vez hallado el valor que satisfaga nuestro proyecto, se utilizará para las demás imágenes capturadas con el dispositivo móvil.

Figura (53) Imagen original de código de barras
Elaboración: Los autores.

a) b)

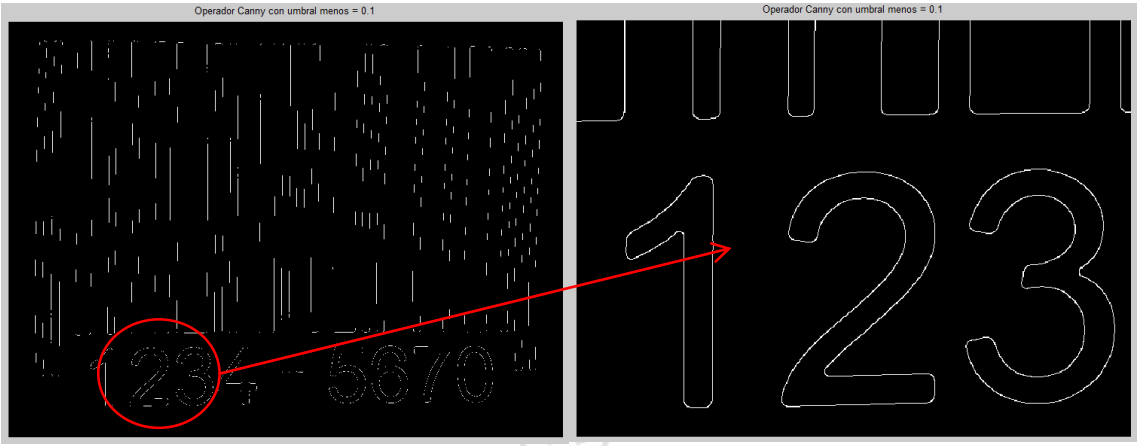


Figura (54) Detección de bordes mediante el operador de Canny con un umbral de μ = 0.1. (64.a) Operador de Canny aplicado a la imagen de código de barras (64.b) Acercamiento de la imagen para apreciar los resultados en los bordes. Elaboración: Los autores.

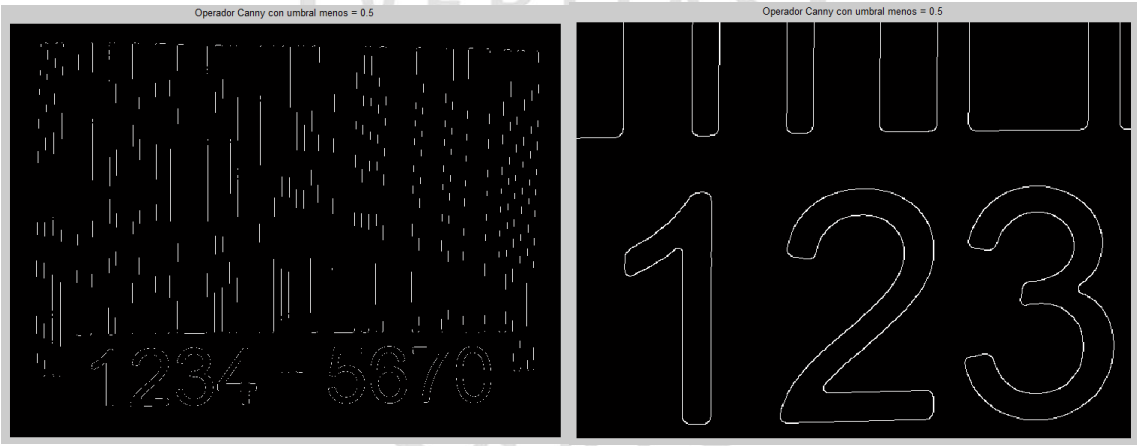


Figura (55) Detección de bordes mediante el operador de Canny con un umbral de μ = 0.5. (65.a) Operador de Canny aplicado a la imagen de código de barras (65.b) Acercamiento de la imagen para apreciar los resultados en los bordes. Elaboración: Los autores.

Figura (56) Detección de bordes mediante el operador de Canny con un umbral de μ = 0.9. (66.a) Operador de Canny aplicado a la imagen de código de barras (66.b) Acercamiento de la imagen para apreciar los resultados en los bordes.
Elaboración: Los autores.

Según lo observado, en la figura (54) se aprecia que se detectan más cantidad de bordes sin que estos sean ruido. Cuando el valor de umbral menor tiende a aumentar, este pierde cantidad de información debido a que los bordes detectados tienen bajos valores de magnitud de gradiente. En tal sentido, el valor óptimo para el proyecto en cuestión seria **μ = 0.1**.

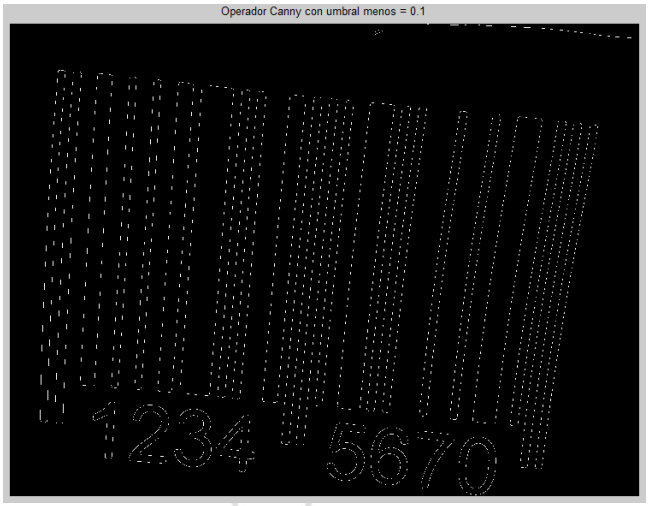
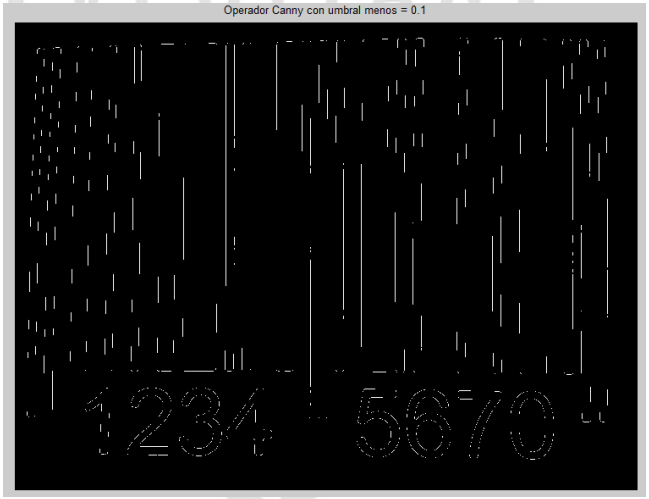
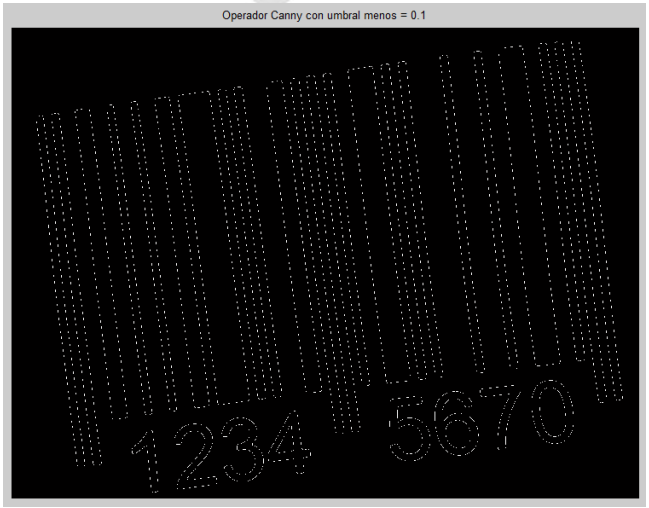


Figura (57) Operador de Canny aplicado a las imágenes de código de barras tomadas en el exterior

Elaboración: Los autores.





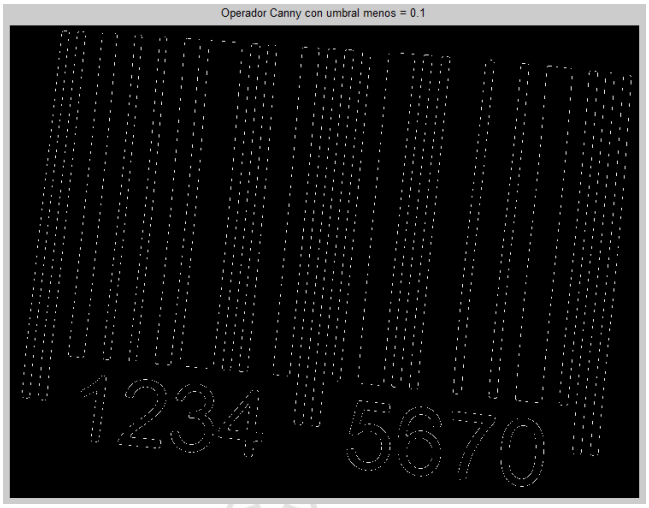


Figura (58) Operador de Canny aplicado a imágenes de código de barras tomadas en un ambiente bajo techo

Elaboración: Los autores.

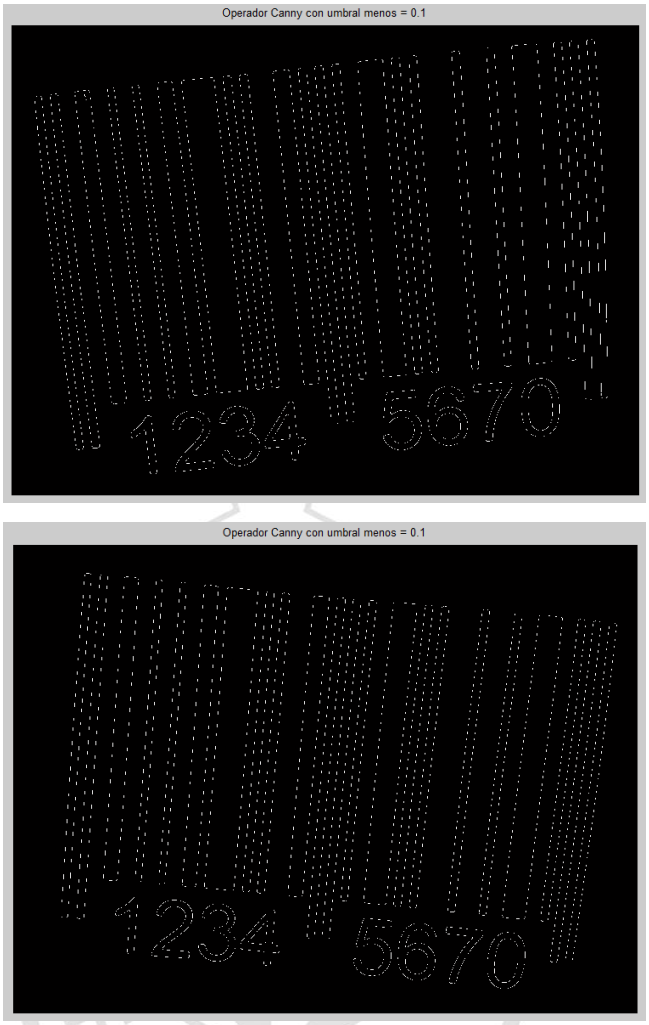


Figura (59) Operador de Canny aplicado a imágenes de código de barras tomadas de noche en presencia de luz artificial color blanco
Elaboración: Los autores.

3.2.1.2 Ángulo de rotación de la imagen de código de barras mediante la detección de líneas por la transformada de Hough

En la sección de análisis, se demostró que la transformada Hough se encarga de la rotación de la imagen de código de barras, para que esta se ubique en una forma en la cual se pueda decodificar sin presentar errores. Es por ello que para este caso tomaremos como ejemplo la imagen de

la figura (60) que trazaremos una (01) recta y la evaluaremos mediante este método.

a) b)

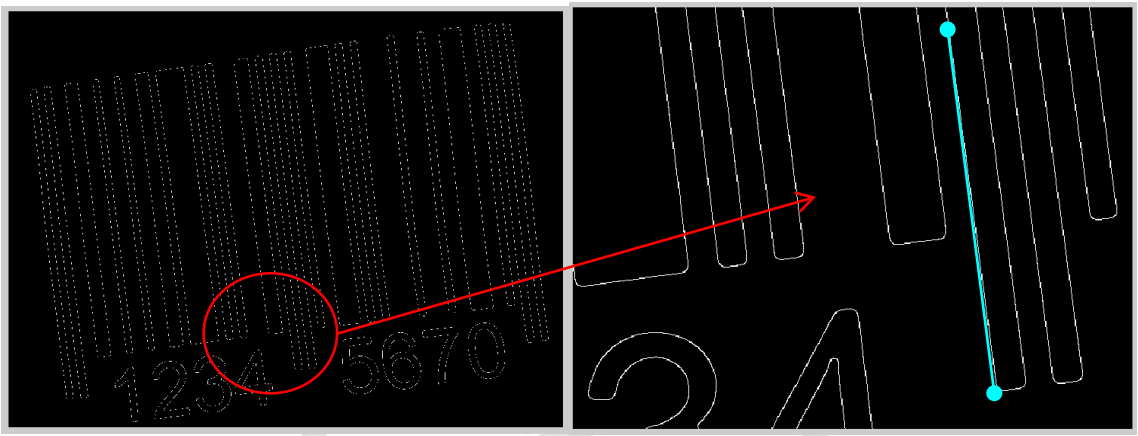


Figura (60) Selección de recta para realizar la transformada de Hough. (60.a) Operador de Canny aplicado a una imagen de código de barras. (60.b) Recta seleccionada de las líneas de control del código de barras.

Elaboración: Los autores.

En dicha recta, se toman cuarto (04) puntos, que son A(1284,563), B(1302,416), C(1312,1564) y D(1321,261). Para cada uno de estos puntos se tiene un **ρ** y **θ** diferentes, que serán representados en el plano **ρθ** como se muestra en la figura (61) y los valores hallados para cada punto se muestran en la tabla (3).

Tabla (3) Valores hallados para cada uno de los puntos.

ρ				θ (rad)								
	Punto	Eje X	Eje Y	0	0.785398163	1.57079633	2.35619449	3.14159265	3.92699082	4.71238898	5.49778714	6.283185307
	A	1284	563	1284	1306	563	-510	-1284	-1306	-563	510	1284
	B	1302	416	1302	1215	416	-626	-1302	-1215	-416	626	1302
	C	1312	335	1312	1165	335	-691	-1312	-1165	-335	691	1312
	D	1321	261	1321	1119	261	-750	-1321	-1119	-261	750	1321

Elaboración: Los autores.

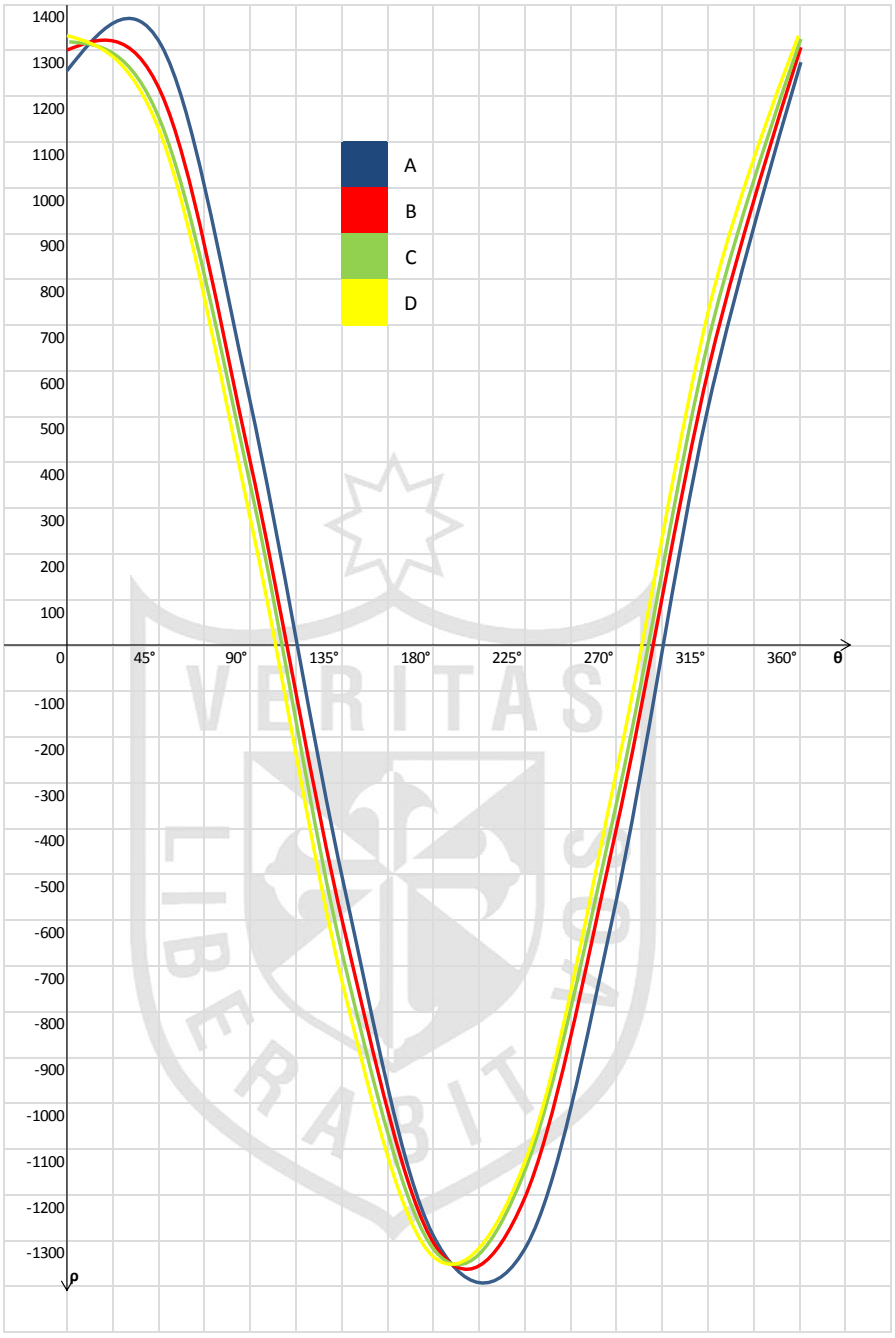


Figura (61) Representación de la transformada de Hough para cada uno de los puntos de la recta.

Elaboración: Los autores.

Del cálculo antes realizado se hallan los valores pico **ρ** y **θ** que son (1343, 7°), y el valor de **θ** se usara para el giro de la imagen original del código de barras, como se muestra en la figura (62).

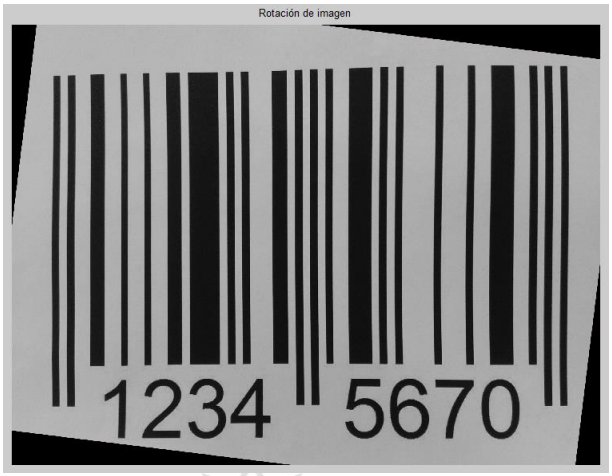


Figura (62) Imagen original de código de barras rotada mediante la transformada de Hough.

Elaboración: Los autores.

Del mismo modo, se aplica el mismo método para rotar las demás imágenes de código de barras.

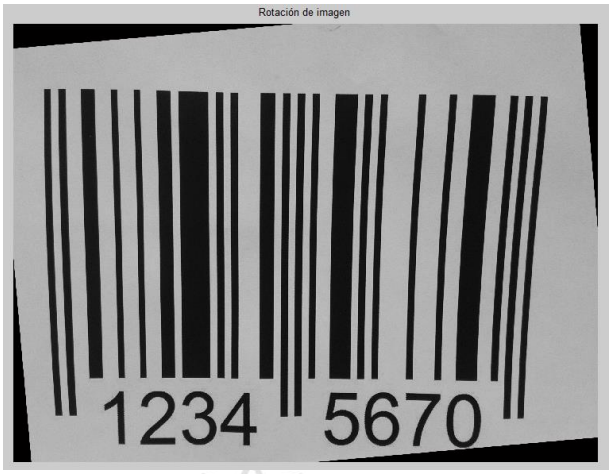


Figura (63) Imágenes de código de barras rotadas mediante la transformada de Hough tomadas en el exterior.

Elaboración: Los autores.

Figura (64) Imágenes de código de barras rotadas mediante la transformada de Hough tomadas en un ambiente bajo techo.

Elaboración: Los autores.

Figura (65) Imágenes de código de barras rotadas mediante la transformada de Hough tomadas de noche en presencia de luz artificial color blanco.
Elaboración: Los autores.

3.2.1.3 Umbral de binarización de la imagen de código de barras mediante el método basado en la forma del histograma

Para poder encontrar el umbral de binarización de una imagen, debemos antes hallar la mayor probabilidad de ocurrencia de la intensidad $\mathbf{r_k}$ en la imagen. Para este ejemplo tomaremos la imagen de código de barras de la figura (66).

Figura (66) Imagen original de código de barras tomada en el exterior.
Elaboración: Los autores.

Para esta imagen se tomaron todos los valores de r_k y de n_k para poder evaluar la probabilidad en que la intensidad de este sea la mayor, la cual corresponderá al umbral de binarización. Para esto se diseñó la tabla (4) con cada uno de estos valores y se muestra también, en forma gráfica, en la figura (67).

Tabla (4) Tabla de probabilidad de ocurrencia del nivel de gris r_k dentro de la imagen de código de barras.

r_k	n_k	$p(r_k)$	r_k	n_k	$p(r_k)$	r_k	n_k	$p(r_k)$	r_k	n_k	$p(r_k)$
0	21	0%	**32**	11556	0%	**64**	1958	0%	**96**	803	0%
1	31	0%	**33**	10500	0%	**65**	774	0%	**97**	576	0%
2	86	0%	**34**	9187	0%	**66**	1295	0%	**98**	1099	0%
3	453	0%	**35**	8683	0%	**67**	672	0%	**99**	918	0%
4	366	0%	**36**	6191	0%	**68**	888	0%	**100**	1631	0%
5	262	0%	**37**	8872	0%	**69**	704	0%	**101**	1249	0%
6	823	0%	**38**	3538	0%	**70**	1046	0%	**102**	624	0%
7	1529	0%	**39**	5251	0%	**71**	1051	0%	**103**	995	0%
8	1772	0%	**40**	3397	0%	**72**	846	0%	**104**	830	0%
9	2387	0%	**41**	2781	0%	**73**	1445	0%	**105**	998	0%
10	6240	0%	**42**	2206	0%	**74**	679	0%	**106**	932	0%
11	1803	0%	**43**	4517	0%	**75**	669	0%	**107**	1765	0%
12	5695	0%	**44**	4642	0%	**76**	763	0%	**108**	720	0%
13	16921	0%	**45**	1369	0%	**77**	1009	0%	**109**	792	0%
14	26292	1%	**46**	2276	0%	**78**	667	0%	**110**	1716	0%
15	31239	1%	**47**	3909	0%	**79**	555	0%	**111**	1118	0%
16	77602	2%	**48**	1455	0%	**80**	622	0%	**112**	917	0%
17	93546	2%	**49**	2256	0%	**81**	894	0%	**113**	987	0%
18	159961	3%	**50**	1610	0%	**82**	717	0%	**114**	623	0%
19	161857	3%	**51**	1783	0%	**83**	1261	0%	**115**	1260	0%
20	246263	5%	**52**	1279	0%	**84**	816	0%	**116**	935	0%
21	59567	1%	**53**	2135	0%	**85**	408	0%	**117**	1683	0%
22	59270	1%	**54**	2791	0%	**86**	653	0%	**118**	879	0%
23	131996	3%	**55**	1065	0%	**87**	624	0%	**119**	1306	0%
24	116475	2%	**56**	2021	0%	**88**	739	0%	**120**	823	0%
25	85673	2%	**57**	1089	0%	**89**	629	0%	**121**	1123	0%
26	55655	1%	**58**	1000	0%	**90**	1193	0%	**122**	841	0%
27	57136	1%	**59**	1289	0%	**91**	540	0%	**123**	819	0%
28	15978	0%	**60**	1521	0%	**92**	595	0%	**124**	1606	0%
29	22670	0%	**61**	1215	0%	**93**	1207	0%	**125**	749	0%
30	24487	0%	**62**	924	0%	**94**	757	0%	**126**	785	0%
31	11247	0%	**63**	998	0%	**95**	755	0%	**127**	1752	0%

r_k	n_k	$p(r_k)$	r_k	n_k	$p(r_k)$	r_k	n_k	$p(r_k)$	r_k	n_k	$p(r_k)$
128	1129	0%	160	74700	2%	192	1	0%	224	0	0%
129	1217	0%	161	99389	2%	193	2	0%	225	0	0%
130	1815	0%	162	96025	2%	194	0	0%	226	0	0%
131	1255	0%	163	144915	3%	195	0	0%	227	0	0%
132	769	0%	164	215532	4%	196	0	0%	228	0	0%
133	1019	0%	165	173299	4%	197	0	0%	229	0	0%
134	2176	0%	166	213174	4%	198	0	0%	230	0	0%
135	1251	0%	167	322861	7%	199	0	0%	231	0	0%
136	1672	0%	168	313359	6%	200	0	0%	232	0	0%
137	1060	0%	169	289703	6%	201	0	0%	233	0	0%
138	1509	0%	170	284560	6%	202	0	0%	234	0	0%
139	1364	0%	171	199619	4%	203	0	0%	235	0	0%
140	1419	0%	172	186945	4%	204	0	0%	236	0	0%
141	2692	0%	173	151929	3%	205	0	0%	237	0	0%
142	1369	0%	174	143421	3%	206	0	0%	238	0	0%
143	1246	0%	175	66753	1%	207	0	0%	239	0	0%
144	3227	0%	176	50223	1%	208	0	0%	240	0	0%
145	2088	0%	177	33056	1%	209	0	0%	241	0	0%
146	2304	0%	178	13927	0%	210	0	0%	242	0	0%
147	3423	0%	179	8202	0%	211	0	0%	243	0	0%
148	1970	0%	180	7382	0%	212	0	0%	244	0	0%
149	2002	0%	181	3843	0%	213	0	0%	245	0	0%
150	3097	0%	182	1710	0%	214	0	0%	246	0	0%
151	5598	0%	183	1071	0%	215	0	0%	247	0	0%
152	4599	0%	184	528	0%	216	0	0%	248	0	0%
153	6103	0%	185	186	0%	217	0	0%	249	0	0%
154	6124	0%	186	246	0%	218	0	0%	250	0	0%
155	10663	0%	187	99	0%	219	0	0%	251	0	0%
156	13338	0%	188	56	0%	220	0	0%	252	0	0%
157	25866	1%	189	17	0%	221	0	0%	253	0	0%
158	25332	1%	190	4	0%	222	0	0%	254	0	0%
159	29858	1%	191	5	0%	223	0	0%	255	0	0%

Elaboración: Los autores.

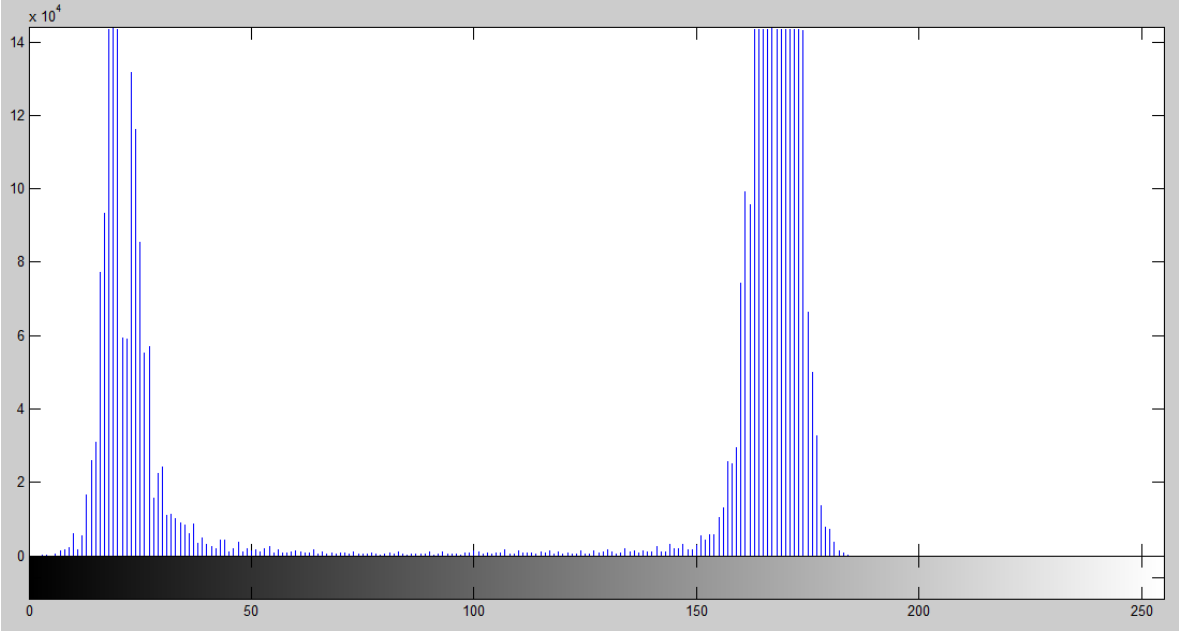


Figura (67) Demostración gráfica del histograma de la imagen de código de barras.

Elaboración: Los autores.

Como se puede apreciar, existen 322861 pixeles que se encuentran en el nivel de gris $\mathbf{r_k}$ = 167. Ya que este valor es la máxima cantidad de pixeles de esta intensidad, se deberá tomar un valor menor para que satisfaga las especificaciones de nuestro proyecto. Por lo tanto, y según las pruebas realizadas, se disminuirá en 50 el nivel de $\mathbf{r_k}$ para que se visualice en forma correcta la imagen de código de barras para poder ser decodificada. Una vez realizada la binarización, el resultado será el que se muestra en la figura (68) siguiendo la siguiente condición que:

- $\mathbf{r_k} > 117 \rightarrow$ blanco
- $\mathbf{r_k} \leq 117 \rightarrow$ negro

Figura (68) Imagen de código de barras binarizada mediante el método basado en la forma del histograma.
Elaboración: Los autores.

Para poder hallar el umbral de binarización para nuestro proyecto, deberemos encontrar el valor del umbral de cada una de las imágenes de prueba y obtener el promedio de las misma.

μ = 174

μ = 180

μ = 181

μ = 181

μ = 181

μ = 164

Figura (69) Imágenes de código de barras binarizadas por el método basado en la forma del histograma.
Elaboración: Los autores.

Con estos valores obtenemos el promedio de los pixeles mayoritarios en la imagen de código de barras que es igual a 174.22. A este valor le restamos el valor de 50, lo cual nos da como resultado la especificación de diseño del umbral de binarización para nuestro proyecto que será 124.

3.2.2 Diseño de algoritmo propietario basado en métodos comparativos para el reconocimiento de líneas de la imagen del código de barras

En esta sección, detallaremos las consideraciones que se tuvieron para la creación del algoritmo propietario para el reconocimiento de líneas

dentro de una imagen de código de barras. Para demostrarlo, utilizaremos la imagen de código de barras de la figura (70), la cual ya fue rotada y binarizada para el siguiente proceso.

Figura (70) Imagen de código de barras rotada y binarizada.
Elaboración: Los autores.

Lo primero será detectar la línea central de la imagen de código de barras con la que se va a trabajar. Para esto ya sabemos que la imagen es del tamaño **N x M** = 2560 x 1920 donde el segundo valor representa el tamaño vertical de la imagen, en este caso, será considerada como el eje Y, ya que el valor del tamaño vertical de la imagen es par, tomaremos el valor siguiente, lo cual se representa como se muestra a continuación.

$$F_C = \frac{M}{2} + 1$$

Una vez obtenida dicha fila, se procese a iniciar el conteo de la misma hasta que la detección de la primera línea negra (0), como se muestra en la figura (71), los valores anteriores se consideran como el contorno de la imagen de código de barras, los cuales serán suprimidos.

Figura (71) Fila central de la imagen de código de barras.
Elaboración: Los autores.

Luego se inicia el conteo de los pixeles de la primera línea negra (0) para calcular el tamaño y el promedio de la misma. Una vez que el conteo detecte el primer pixel de color blanco (255), se calcula los valores antes mencionados, donde se deberá establecer el tamaño de cada una de las líneas negras (0) y blancas (255) que se cuenten a continuación.

Cuando se detecte un valor de cualquier de los dos (02) valores posibles de los pixeles mayor al establecido en la primera línea, este se considerará como un valor diferente a una (01) línea.

Establecido esto, el sistema contará cada uno de los valores de la fila seleccionada los cuales serán acumulados en una matriz [0 255], donde se tendrá la lectura del reconocimiento del código de barras. Dicho resultado se muestra en la figura (72).

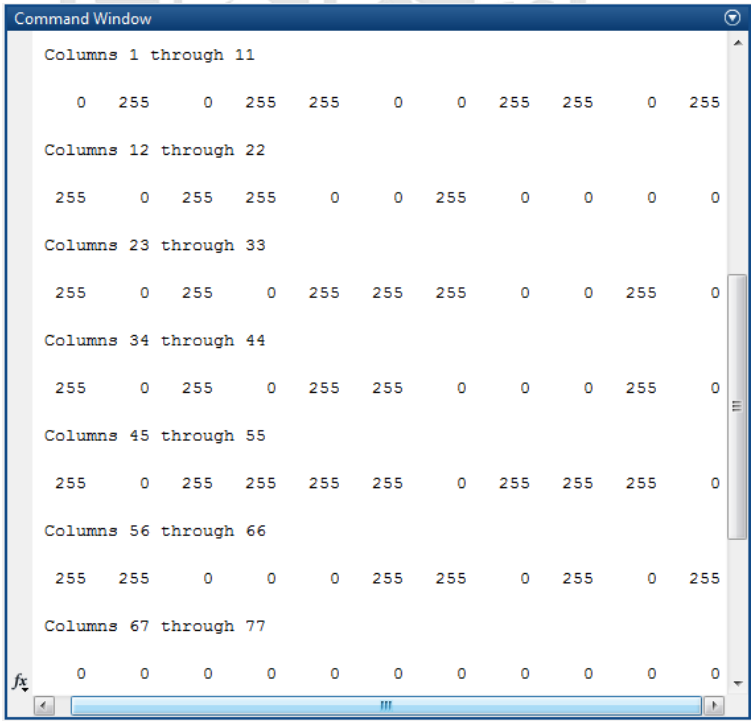


Figura (72) Reconocimiento de líneas de la imagen de código de barras.
Elaboración: Los autores.

3.2.3 Diseño del sistema de reconocimiento de códigos de barras asistido por computadora (Computer Aided Design Matlab®)

Se explicará paso a paso, la implementación de algoritmos utilizados en el diseño de nuestro proyecto en el programa de prueba y error Matlab®. Los comandos del sistema podrán ser visualizados en la sección anexos. En este ejemplo, utilizaremos la imagen de código de barras de la figura (73).

Figura (73) Imagen original de código de barras.
Elaboración: Los autores.

Lectura de imagen

El comando “imread” llama a la dirección de la carpeta que hayamos declarado la ubicación de las imágenes a utilizarse. El comando “figure” abrirá una nueva ventana donde se mostrara la imagen a leer. El comando “imshow” mostrará la imagen “dia_1.jpg” en dicha ventana. Por último el comando “title” mostrará el título de la imagen en la parte superior, en este caso “Imagen original de código de barras”. Dicha representación se muestra en la figura (74).

Figura (74) Imagen original de código de barras en la ventana del programa Matlab®.

Elaboración: Los autores.

Cambio a escala de grises

Para realizar la detección de bordes, se requiere que la imagen de código de barras se represente en forma binaria, por lo cual se deberá cambiar a escala de grises. Las constantes con las cuales se multiplica cada uno de los valores RGB cambian toda tonalidad de colores a grises desde los niveles [0 255]. El resultado se muestra en la figura (75).

Figura (75) Imagen de código de barras en escala de grises.

Elaboración: Los autores.

Detección de bordes mediante el operador Canny

Como se especificó, anteriormente, la desviación estándar (**σ**) del filtro gaussiano esta presentada por “sigma” y el valor del umbral menor del operador de Canny por “thresh”.El resultado se muestra en la figura (76).

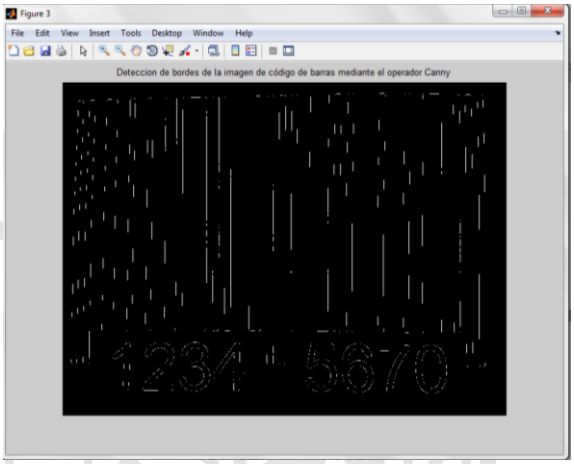


Figura (76) Detección de bordes de la imagen de código de barras mediante el operador de Canny.
Elaboración: Los autores.

Rotación de la imagen mediante la transformada de Hough

El comando “hough” acumula los valores **ρ** y **θ** en una matriz acumuladora. El comando “houghpeaks” coge de dicha matriz los valores pico de estos, donde se toma el valor pico de **θ** para realizar la rotación de la imagen mediante el comando “imrotate”. El resultado se muestra en la figura (77). En este caso, la imagen no muestra signos de rotación ya que se encuentra direccionada al eje horizontal.

Figura (77) Rotación de la imagen de código de barras mediante la transformada de Hough.

Elaboración: Los autores.

Binarización de la imagen mediante el método basado en la forma del histograma

El valor del umbral de binarización hallado, anteriormente, es comparado con cada uno de los valores de intensidad de los pixeles de la imagen para que solo puedan tener valores de 0 (negro) y 255 (blanco). El resultado se muestra en la figura (78).

Figura (78) Binarización de la imagen de código de barras mediante el método basado en la forma del histograma.

Elaboración: Los autores.

Reconocimiento de líneas mediante algoritmo propietario

Como se explicó anteriormente, el algoritmo propietario para el reconocimiento de líneas del código de barras se basa en hallar el tamaño de la primera línea negra (0) que se halla al lado izquierdo de la imagen y desde ese punto se inicia el conteo de cada una de las líneas. El resultado de todo el algoritmo se muestra en la figura (79).

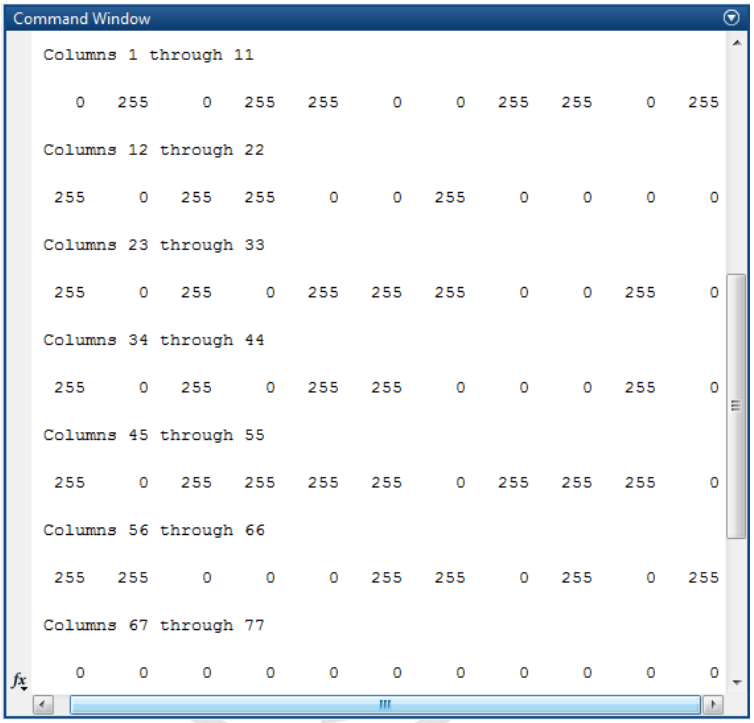


Figura (79) Matriz de los valores hallados en el reconocimiento de líneas de la imagen de código de barras.
Elaboración: Los autores.

3.2.4 Diseño de la aplicación móvil para el reconocimiento y decodificación de códigos de barras asistido por computadora (Computer Aided Design Eclipse®)

Una vez realizado el sistema de reconocimiento de códigos de barras asistido por el programa de prueba y error Matlab®, nos basamos de este para

la creación de la aplicación móvil en el sistema operativo Android®. Para esto se utilizara la interface de programación Eclipse® el cual es utiliza el lenguaje de programación Java®. Asimismo se tendrá que instalar las librerías de procesamiento digital de imágenes OpenCV, tanto en el computador como en el dispositivo móvil. El diseño del mismo se podrá apreciar en la parte de pruebas y resultados.

3.2.5 Simulación del modelo cliente - servidor para el almacenamiento de formatos de mantenimiento dentro del dispositivo móvil

A continuación, revisaremos el procedimiento de creación y puesta en marcha de la base de datos y la simulación del cliente - servidor.

1. El primer paso para poner en marcha la base de datos es la creación del archivo en donde se va guardar la tabla con la información.

```
private static final String DB_NAME = "equipos.sqlite";
private static final int DB_SCHEME_VERSION = 1;
public DbHelper(Context context) {
super(context, DB_NAME, null, DB_SCHEME_VERSION);
```

2. Luego, se crean las variables para los campos de información.

```
public static final String CN_ID = "_id";
public static final String CN_CODIGO = "codigo";
public static final String CN_DESCRIPCION = "descripcion";
public static final String CN_MARCA = "marca";
public static final String CN_MODELO = "modelo";
public static final String CN_NUMSERIE = "numserie";
public static final String CN_FECHA = "fecha";
public static final String CN_TRABAJO = "trabajo";
```

3. Se crea la tabla de la base de datos.

```
public static final String CREATE_TABLE="create table " +TABLE_NAME+ " ("
+ CN_ID + " integer primary key autoincrement,"
+ CN_CODIGO + " text not null,"
```

```
+ CN_DESCRIPCION + " text,"
+ CN_MARCA + " text,"
+ CN_MODELO + " text,"
+ CN_NUMSERIE + " text,"
+ CN_FECHA + " text,"
+ CN_TRABAJO + " text);";

@Override
public void onCreate(SQLiteDatabase db) {
db.execSQL(DataBaseManager.CREATE_TABLE);
```

Tabla (5) Base de datos ingresado en el dispositivo móvil.

ID	CODIGO	DESCRIPCION	MARCA	MODELO	NUMSERIE	FECHA	TRABAJO
1	12345670	Electrobomba	Pedrollo	Vmx 15/50	DX13-4567	12/11/2013	Instalación
2	43218769	Motor estacionario	Honda	GX270H QH	KDDL-134	07/12/2013	Instalación
3	56781236	Compresora	Karson	Stratos	FT-325734	12/12/2013	Instalación
4	76543210	Generador a gasolina	Bauker	Gas-134	KL-2314	20/12/2013	Instalación
5	12345670	Electrobomba	Pedrollo	Vmx 15/50	DX13-4567	12/03/2014	Mantenimiento preventivo
6	43218769	Motor estacionario	Honda	GX270H QH	KDDL-134	18/03/2014	Mantenimiento preventivo
7	56781236	Compresora	Karson	Stratos	FT-325734	20/03/2014	Mantenimiento preventivo
8	76543210	Generador a gasolina	Bauker	Gas-134	KL-2314	25/03/2014	Mantenimiento preventivo
9	12345670	Electrobomba	Pedrollo	Vmx 15/50	DX13-4567	12/07/2014	Mantenimiento preventivo
10	43218769	Motor estacionario	Honda	GX270H QH	KDDL-134	18/07/2014	Mantenimiento preventivo
11	56781236	Compresora	Karson	Stratos	FT-325734	20/07/2014	Mantenimiento preventivo
12	76543210	Generador a gasolina	Bauker	Gas-134	KL-2314	25/07/2014	Mantenimiento preventivo
13	12345670	Electrobomba	Pedrollo	Vmx 15/50	DX13-4567	12/11/2014	Mantenimiento preventivo
14	43218769	Motor estacionario	Honda	GX270H QH	KDDL-134	18/11/2014	Mantenimiento preventivo
15	56781236	Compresora	Karson	Stratos	FT-325734	20/11/2014	Mantenimiento preventivo
16	76543210	Generador a gasolina	Bauker	Gas-134	KL-2314	25/11/2014	Mantenimiento preventivo

Elaboración: Los autores.

4. Los datos se ingresan a la base de datos mediante campos de texto.

```
public void insertar(String codigo, String descripcion, String marca,
String modelo, String numserie, String fecha, String trabajo) {
db.insert(TABLE_NAME, null, generarContentValues(codigo,descripcion,
marca,modelo,numserie,fecha,trabajo) );
```

5. Los datos se leen haciendo uso de un cursor y utilizando como dato de comparación el campo de código de la tabla.

Tabla (6) Comparación del valor obtenido en la base de datos.

ID	CODIGO	DESCRIPCION	MARCA	MODELO	NUMSERIE	FECHA	TRABAJO
1	12345670	Electrobomba	Pedrollo	Vmx 15/50	DX13-4567	12/11/2013	Instalación
2	43218769	Motor estacionario	Honda	GX270H QH	KDDL-134	07/12/2013	Instalación
3	56781236	Compresora	Karson	Stratos	FT-325734	12/12/2013	Instalación
4	76543210	Generador a gasolina	Bauker	Gas-134	KL-2314	20/12/2013	Instalación
5	12345670	Electrobomba	Pedrollo	Vmx 15/50	DX13-4567	12/03/2014	Mantenimiento preventivo
6	43218769	Motor estacionario	Honda	GX270H QH	KDDL-134	18/03/2014	Mantenimiento preventivo
7	56781236	Compresora	Karson	Stratos	FT-325734	20/03/2014	Mantenimiento preventivo
8	76543210	Generador a gasolina	Bauker	Gas-134	KL-2314	25/03/2014	Mantenimiento preventivo
9	12345670	Electrobomba	Pedrollo	Vmx 15/50	DX13-4567	12/07/2014	Mantenimiento preventivo
10	43218769	Motor estacionario	Honda	GX270H QH	KDDL-134	18/07/2014	Mantenimiento preventivo
11	56781236	Compresora	Karson	Stratos	FT-325734	20/07/2014	Mantenimiento preventivo
12	76543210	Generador a gasolina	Bauker	Gas-134	KL-2314	25/07/2014	Mantenimiento preventivo
13	12345670	Electrobomba	Pedrollo	Vmx 15/50	DX13-4567	12/11/2014	Mantenimiento preventivo
14	43218769	Motor estacionario	Honda	GX270H QH	KDDL-134	18/11/2014	Mantenimiento preventivo
15	56781236	Compresora	Karson	Stratos	FT-325734	20/11/2014	Mantenimiento preventivo
16	76543210	Generador a gasolina	Bauker	Gas-134	KL-2314	25/11/2014	Mantenimiento preventivo

Elaboración: Los autores.

Una vez encontrada la igualdad mediante el siguiente código:

```
public ContentValues generarContentValues(String codigo, String
descripcion, String marca, String modelo, String numserie, String fecha,
String trabajo) {
ContentValues valores = new ContentValues();
valores.put(CN_CODIGO, codigo);
valores.put(CN_DESCRIPCION, descripcion);
valores.put(CN_MARCA, marca);
valores.put(CN_MODELO, modelo);
valores.put(CN_NUMSERIE, numserie);
valores.put(CN_FECHA, fecha);
valores.put(CN_TRABAJO, trabajo);
return valores;
        }

public Cursor buscarContacto(String codigo) {   //modificado
String[] columnas = new String[]{CN_CODIGO, CN_DESCRIPCION, CN_MARCA,
CN_MODELO, CN_NUMSERIE};
return db.query(TABLE_NAME,columnas,CN_CODIGO + "=?",new
String[]{codigo},null,null,null);
        }
```

Se obtiene respuesta los datos de la fila pertenecientes a la igualdad del dato comparado, como se muestra en la figura (80).

56781236	Compresora	Karson	Stratos	FT-325734	20/11/2014	Mantenimiento preventivo

Figura (80) Sección de la base de datos que coincide con el valor hallado de código de barras.
Elaboración: Los autores.

Para nuestra tesis, la base de datos y la simulación del modelo cliente - servidor ha utilizado el SQLite®, que es el gestor de base de datos más utilizado en los sistemas Android®.

Tabla (7) Identificación de los parámetros de la simulación del modelo cliente - servidor con SQLite®.

	CLIENTE	SERVIDOR
DISPOSITIVO	Celular	Celular
SISTEMA OPERATIVO	Android®	Android®
PROGRAMA	Aplicación desarrollada	Aplicación desarrollada y SQLite®

Como se puede observar en la tabla (7), el cliente y el servidor son el mismo dispositivo móvil, la comunicación y transmisión de datos es interna ya que el archivo donde se guarda la tabla y los datos se crea dentro de la memoria del dispositivo.

Si bien es cierto el uso del SQLite® no llega a ser un modelo cliente - servidor, para realizar las pruebas de nuestro Rapid Prototyping, ha sido de gran utilidad por que simula dicho modelo real, en el siguiente paso a seguir en futuras actualizaciones de nuestra tesis se procederá a implementar el modelo cliente - servidor con los programas MySQL® y PHP.

CAPÍTULO IV
PRUEBAS Y RESULTADOS

Una vez finalizada la programación en el sistema operativo Android® e instalada en el dispositivo móvil, se procede a realizar las pruebas correspondientes. La figura (81) muestra los pasos a seguir de la aplicación para el reconocimiento de la imagen de código de barras. Para explicar esta parte se tomaron las siguientes fotografías, que se explicarán luego.

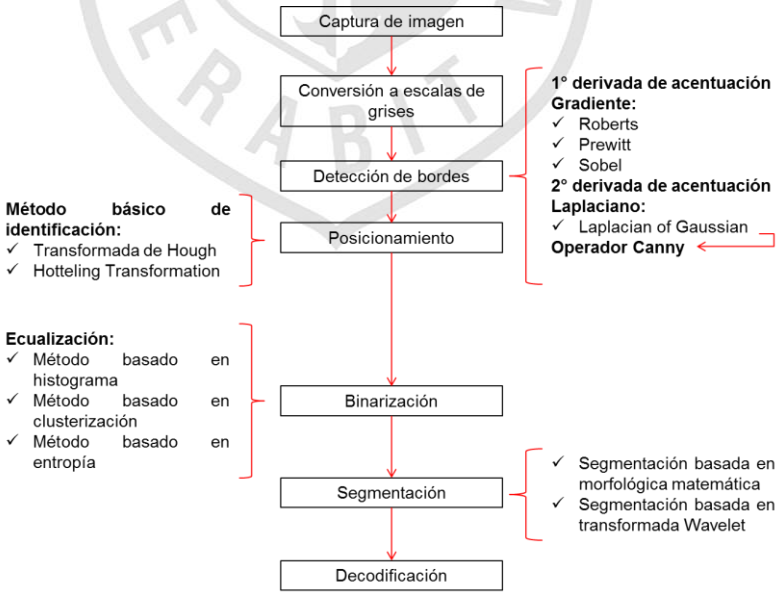


Figura (81) Diagrama de bloques del sistema de reconocimiento de imágenes de código de barras.

Elaboración: Los autores.

Se inicia la aplicación presionando la pantalla del dispositivo móvil en el icono de código de barras como se muestra en figura (82).

Figura (82) Pantalla de inicio del dispositivo móvil.
Elaboración: Los autores.

Una vez iniciada la aplicación, esta mostrará el menú que se aprecia en la figura (83).

Figura (83) Menú principal de la aplicación de reconocimiento de imágenes de código de barras.
Elaboración: Los autores.

El siguiente paso a seguir es capturar la imagen de código de barras, por lo cual se logra presionando el botón de “Escanear Código de Barras”. Una vez presionado dicho botón, se iniciaría la cámara del dispositivo móvil tal como se muestra en la figura (84).

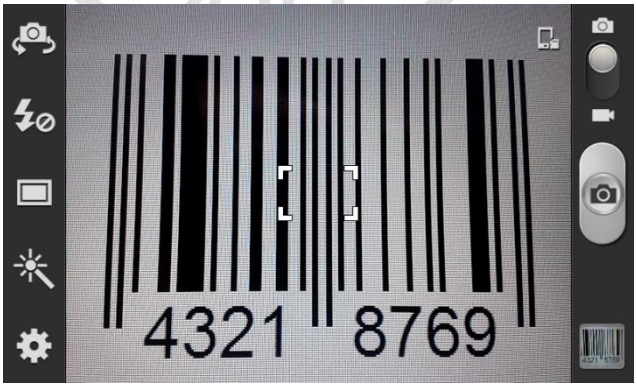


Figura (84) Uso de la cámara del dispositivo móvil para la captura de la imagen del código de barras.
Elaboración: Los autores.

Luego, se presiona el botón para capturar la imagen. Una vez presionada, el sistema operativo del dispositivo móvil brindará dos (02) opciones: “Cancelar“ y “Guardar” como se muestra en la figura (85). Pulsando el botón “Cancelar” la imagen capturada no se guardará en la memoria del dispositivo móvil y se reiniciará el proceso de captura de imagen. Por otro lado, si se pulsa el botón “Guardar”, la imagen capturada será guardada en la memoria del dispositivo móvil e iniciará el proceso de reconocimiento de la imagen. Una vez guardada la imagen, se regresa a la pantalla de inicio de la aplicación como se muestra en la figura (86), en la que figura el código de barras.

Figura (85) Menú de captura de imágenes de la cámara del dispositivo móvil.
Elaboración: Los autores.

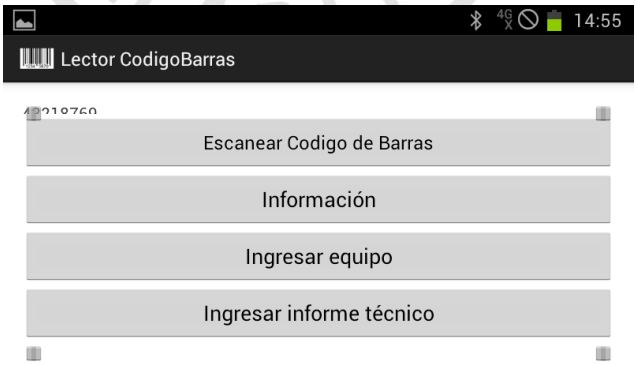


Figura (86) Menú principal de la aplicación de reconocimiento de imágenes de código de barras en el cual figura el valor hallado.
Elaboración: Los autores.

Una vez obtenido el código, ingresamos a la pantalla “Información”, en la cual se mostrarán los datos del equipo enlazado al código de barras ingresado. Dicho pantalla se mostrará en la figura (87). El código obtenido se mostrará en la parte inferior, luego se presionará el botón “Actualizar” para que la información del equipo se muestre en los demás campos.

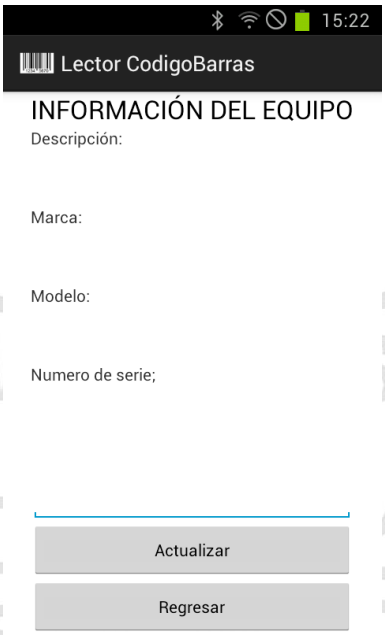


Figura (87) Pantalla “Información” mostrando el equipo enlazado al código “43218769”. Elaboración: Los autores.

En la pantalla “Ingresar equipo”, como se muestra en la figura (88), se puede ingresar un equipo nuevo enlazándolo a un nuevo código. Para estos, se llenan los datos y el código al cual se va a enlazar y se presiona el botón “Ingresar”.

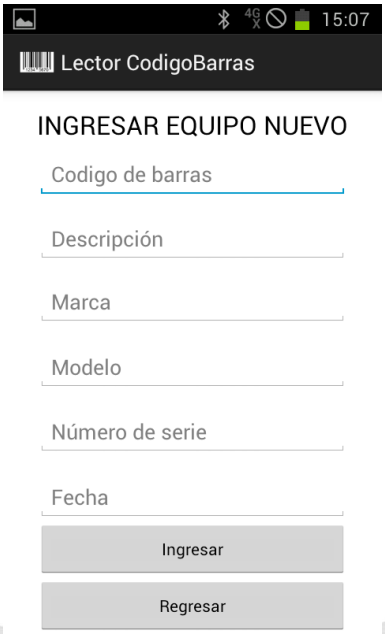


Figura (88) Pantalla "Ingresar equipo" para el ingreso de un nuevo equipo a la base de datos.

Elaboración: Los autores.

En la pantalla "Ingresar informe técnico", se podrá ingresar la fecha y una breve descripción del mantenimiento realizado, el cual podrá ser enviado vía correo electrónico al supervisor o jefe inmediato. Dicho proceso de muestra en las figuras. Se deberá presionar el botón "Ingresar informe" para que el formato de mantenimiento sea guardado en la base de datos. El valor predeterminado para el ingreso de caracteres en el recuadro de "Trabajo realizado:" es de 200. Dicho valor puede ser modificado, pero ya que el informe es una breve descripción de la tarea realizada, se considera esa cantidad de caracteres para este diseño.

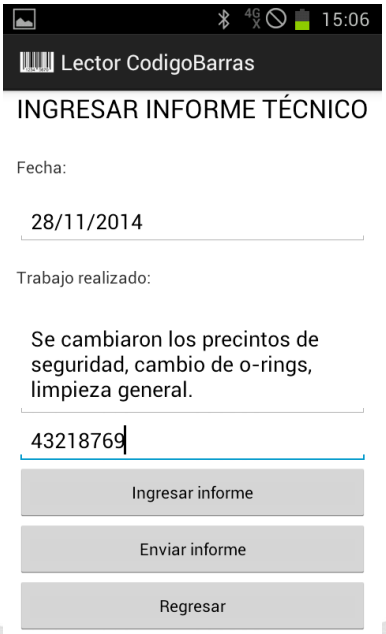


Figura (89) Formato de mantenimiento para ser llenado por el técnico.

Elaboración: Los autores.

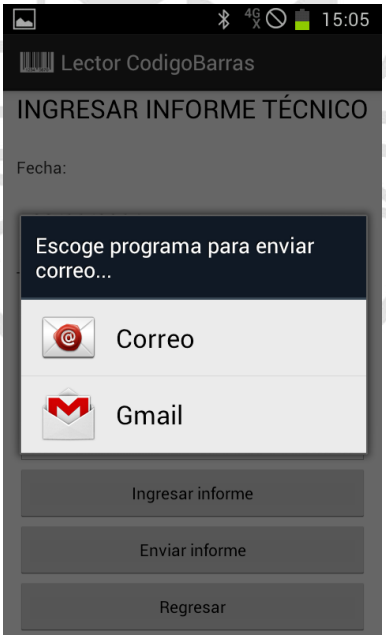


Figura (90) Selección de plataforma de correo electrónico para el envío del formato de mantenimiento.

Elaboración: Los autores.

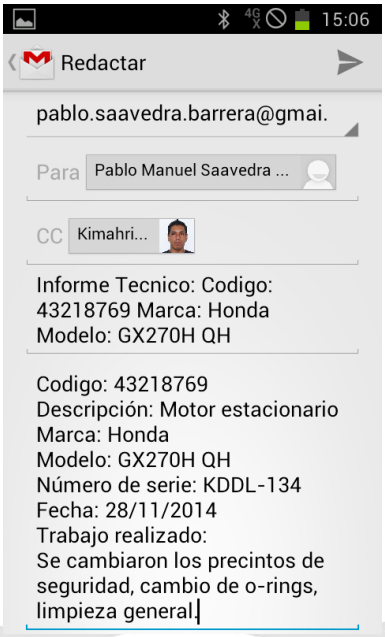


Figura (91) Correo electrónico con ingreso automático de datos del formato de mantenimiento.

Elaboración: Los autores.

En la carpeta DCIM del dispositivo móvil, se guardan las imágenes de los procesos realizados. Dichas imágenes no se mostrarán en el producto final ya que solo se muestran como material para el entendimiento del proceso de reconocimiento.

Figura (92) Detección de bordes de la imagen de código de barras mediante el operador de Canny.

Elaboración: Los autores.

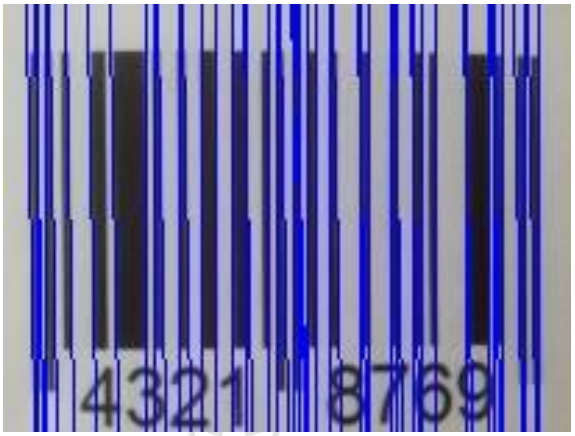


Figura (93) Detección de las líneas en la imagen de código de barras mediante la transformada de Hough.

Elaboración: Los autores.

Figura (94) Rotación de la imagen de código de barras.

Elaboración: Los autores.

Figura (95) Binarización de la imagen de código de barras mediante el método basado en la forma del histograma.
Elaboración: Los autores.

Según las pruebas realizadas, obtenemos un 95% de probabilidad de tener éxito al momento de utilizar la aplicación. Existe un 5% de probabilidad de ocurrencia de errores normales de la decodificación que se dan por diferentes razones tales como la iluminación, sombras yuxtapuestas, movimientos al momento de enfocar, entre otros. Por otro lado, también se presentan errores en la programación que en una próxima optimización del proyecto se revisarán.

CAPÍTULO V
DISCUSIONES Y APLICACIONES

Esta aplicación se puede usar tanto en ámbitos industriales como en educativos, administrativos, tecnológicos, entre otros; por lo cual no se limita para solo un ambiente laboral.

La idea de abarcar este tema de tesis fue inicialmente aplicarlo para el mantenimiento de equipos; sin embargo, esta puede ser utilizada para el control de inventarios en un almacén, la identificación de personal que ingresa a un ambiente designado, entre otras aplicaciones que requieran de un método de etiquetado.

Dependiendo del tamaño de la empresa a utilizar la aplicación, esta puede ser modificada para que sea capaz de reconocer códigos de barras de los tipos 8EAN, 10EAN, 12EAN, entre otros, los cuales solo varían dependiendo de la cantidad de activos que posee la empresa.

El producto final podrá ser comercializado una vez que el reconocimiento de las imágenes de código de barras no presente errores. Para esto se deberá investigar sobre la mejora de la resolución de las imágenes resultantes del

reconocimiento de código de barras las cuales se guardan en la memoria del dispositivo móvil en la carpeta DCIM.

CONCLUSIONES

1. El desarrollo de la presente tesis cumple con la tendencia en el avance del estado del arte mundial, el cual sigue los parámetros de detección, rotación, binarización, decodificación y segmentación, más sin embargo no es la máxima tecnología ya diseñada. Muchos de los diseños ya creados provenientes de los países orientales superan con respecto al estado del arte, pero se debe considerar que se realizó este diseño con los conocimientos adquiridos durante nuestros periodos dentro de la universidad.

2. Las especificaciones de diseño han sido comprobadas, brindado un resultado óptimo al momento de utilizar la aplicación.

3. Las consideraciones que se tuvieron para la selección del dispositivo móvil en el cual se instalara la aplicación fueron debidas a la resolución que brinda la cámara del mismo. Para resoluciones mayores o menores, la aplicación podrá ser ajustada cambiando las especificaciones de diseño.

RECOMENDACIONES

1. El diseño podría ser adaptable a un sistema de redes neuronales, para que cada uno de los valores sea almacenado para su posterior evaluación y optimización del sistema.

2. La aplicación puede ser modificada para el reconocimiento no solo de un código de barras del tipo 8EAN, sino también para otros de mayor tamaño, dependiendo de la empresa que la utilizara.

FUENTES DE INFORMACIÓN

Bibliográficas:

1. Aas, K., y Eikvil, L. (1997). Decoding bar codes from human-readable characters. *Pattern Recognitions Letters,* 18: 1519 - 1527.
2. Azernikov, S. (2008) *Sweeping solids on manifolds*. New York, Estados Unidos: ACM.
3. Badenas, J., Llopis, J., y Coltell, O. (1997). *Curso práctico de programación en C y C++*. España: Universitat Jaume I.
4. Castelló, V. (2005). *Localización de códigos de barras en imágenes digitales*. España: Universitat Jaume I.
5. Canny, J. (1987 noviembre). A computational approach to edge detection. Pattern analysis and machine intelligence, *IEEE Transactions*, 8(6): 679 - 698.
6. Gonzales, R., y Woods, R. (2007). *Digital image processing, third edition.* New Jersey, Estados Unidos: Prentice Hall.
7. Hang, D., Teng, J., Yang, Z., Pang, Y., y Wang, M. (2010). 2D barcode image binarization based on Wavelet analysis and Otsu's method. Computer Application and System Modeling (ICCASM). Taiyuan, China: North University of China.

8. Kapadia, H., y Pastel, A. (2013). Application of Hough Transform and Sub-Pixel Edge Detection in 1-D Barcode Scanning. *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering*.
9. Liu, S. (1993) Camera-based bar code recognition system using neural network. *Neural Networks, 1993. IJCNN '93-Nagoya. Proceedings of 1993 International Joint Conference*. Taipei, Taiwan: Int. of Inf. Sci.
10. Mai, F., Hung, Y., Zhong, H., y Sze, W. (2008 agosto). A hierarchical approach for fast and robust ellipse extraction. *Pattern Recognition*, 41(8): 2512 - 2524.
11. Moeslund, T. (2012). *Introduction to video and image processing*. Estados Unidos: Springer.
12. Oktem, R. (2004). Barcode localization in Wavelet domain by using binary morphology. *Signal Processing and Communications Applications Conference, 2004. Proceedings of the IEEE 12th*.
13. Shams, R., y Sadeghi, P. (2007). *Barcode recognition in highly distorted and low resolution images*. Camberra, Australia: The Australian National University.
14. Sucar, L., y Gómez, G. (2012). *Visión Computacional*. Puebla, México.

Electrónicas:

1. Classical feature detection. <http://www.cse.iitk.ac.in/users/amit/courses/768/vision/robyn/imageproc/node2.html>.
2. Derivative-based operations. <http://www.ph.tn.tudelft.nl/Courses/FIP/noframes/fip-Derivati.html>.
3. Gradiente. <http://www.vinuesa.com/dip/gradiente.pdf>.
4. http://members.fortunecity.es/davidweb2/visart/bordes.htm
5. http://poseidon.tel.uva.es/~carlos/ltif10001/segmenclasica.pdf

6. Aplicación para manejo y visualización de imágenes. http://www.tele.ucl.ac.be/PEOPLE/DOUXCHAMPS/ieee1394/coriander.
7. History of barcodes. http://www.basics.ie/History.htm. Ultimo acceso: Junio 2005.
8. Información general sobre códigos de barras. http://www.serebella.com/encyclopedia/article-Barcode.html.
9. Información general sobre impresoras de códigos de barras. http://www.dimension-x.com/cf-onsprn2.htm.
10. Russ Adams. “Información general sobre códigos de barras”. http://www.barcode-1.com.

ANEXOS

1. Sistema operativo Android®.
2. Código de barras.
3. Lector de código de barras.
4. Sistema de reconocimiento de líneas del código de barras asistido por computadora (Computer Aided Design Matlab®)

ANEXO 1

SISTEMA OPERATIVO ANDROID

CONCEPTO

Es un sistema operativo móvil basado en Linux, que junto con aplicaciones middleware está enfocado para ser utilizado en dispositivos móviles con pantalla táctil como smartphones, tablets, Google TV y otros dispositivos.

Cuenta con una plataforma de descarga de aplicaciones y juegos llamada Google Play, la cual tiene más de 100 mil opciones para descargar, la mayoría gratuita.

El sistema operativo inicialmente fue desarrollado por Android Inc., a la cual Google respaldó económicamente y más tarde compró en el 2005.

Android fue presentado en el 2007 en un consorcio de compañías de hardware, software y telecomunicaciones.

EVOLUCIÓN

Este sistema operativo “open source” para dispositivos móviles ha evolucionado mucho desde su lanzamiento.

La primera versión, Android 1.0 que Google lanzó con el smartphone HTC G1 ha sido superada en cuanto a funcionalidades y estabilidad por las últimas versiones.

El sistema operativo está compuesta por 12 millones de líneas de código, incluyendo 3 millones de líneas XML, 2.8 millones de líneas de lenguaje C, 2.1 millones de líneas Java y 1.75 millones de líneas de C++.

COMPONENTES PRINCIPALES

- **Aplicaciones:** las aplicaciones incluyen un cliente de correo electrónico, programa de SMS, calendario, mapas, navegadores, contactos y otros. Todas las aplicaciones están escritas en lenguaje de programación Java.
- **Marco de trabajo de aplicaciones:** La arquitectura está diseñada para simplificar la reutilización de componentes.
- **Bibliotecas:** Android incluye un conjunto de biblioteca de C/C++ usadas por varios componentes del sistema.
- **Runtime de Android:** Android incluye un set de bibliotecas base que proporcionan la mayor parte de las funciones en las bibliotecas base del lenguaje Java.
- **Núcleo Linux:** Android depende de Linux para servicios base del sistema como seguridad, gestión de memoria, gestión de procesos, pila de red y modelo de controladores. El núcleo también actúa como una capa de abstracción entre el hardware y el resto de la pila de software.

Android®, al contrario que otros sistemas operativos para dispositivos móviles como iOS o Windows Phone, se desarrolla de forma abierta y se puede acceder tanto al código fuente, como a la lista de incidencias donde se pueden ver problemas aún no resueltos y reportar problemas nuevos.

CARACTERISTICAS

- **Framework de aplicaciones:** permite el reemplazo y la reutilización de los componentes.
- **Navegador integrado:** basado en el motor open source Webkit.

- **SQlite:** base de datos para almacenamiento estructurado que se integra directamente con las aplicaciones.
- **Multimedia:** soporte para medios con formatos comunes de audio, video e imágenes planas (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF).
- **Máquina virtual Dalvik:** base de llamadas de instancias muy similar a Java.
- **Telefonía GSM:** dependiente del terminal.
- **Bluetooth, EDGE, 3g y Wifi:** dependiente del terminal.
- **Cámara, GPS, brújula y acelerómetro:** dependiente del terminal.
- **Pantalla táctil**

HISTORIAL DE ACTUALIZACIONES

Android ha visto numerosas actualizaciones desde su liberación inicial, Estas actualizaciones al sistema operativo base típicamente arreglan bugs y agregan nuevas funciones. Las versiones de Android reciben el nombre de postres en inglés.

- **A: Apple Pie (v1.0)** Fue la primera versión comercial del software, fue lanzado el 23 de setiembre de 2008. El primer dispositivo Android fue el HTC Dream. Entre sus características más notables resaltan: Android Market, navegador web, soporte cámara, carpetas para agrupar aplicaciones, acceso a correo electrónico por web, sincronización de Gmail, sincronización Google Contacts, sincronización Google Calendar, Google Maps, Google Search, Chat Google Talk, soporte para Wifi y Bluetooth, marcación por voz.

- **B: Banana Bread (v1.1)** El 9 de febrero del 2009 fue lanzada para el HTC Dream solamente. La actualización resolvió fallas y agregó una serie de características: detalles y reseñas disponibles cuando un usuario busca negocios en los mapas, pantalla en llamada más larga por defecto cuando

están en uso el manos libre, posibilidad de guardar archivos adjuntos en los mensajes y añadido soporte para marquesina en diseños de sistemas.

- **C: Cupcake (v1.5)** El 30 de abril de 2009 la actualización fue lanzada basada en el núcleo Linux 2.6.27 e incluye varias características nuevas y correcciones de interfaz de usuario. Es un tema que se utilizaría para todos las versiones de aquí en adelante: soporte para Widgets, grabación y reproducción de formatos MPEG-4 y 3GP, características copiar y pegar agregadas al navegador Web, pantallas de transiciones animadas, agregada opción auto-rotación entre otras.

- **D: Donut (v1.6)** El 15 de setiembre del 2009 fue lanzado esta actualización e incluyó numerosas características nuevas tales como: mejora de la búsqueda por voz y texto, habilidad para ver capturas de las aplicaciones en Android Market, galería, cámara y videocámara con mejor integración, múltiple selección de imágenes para ser eliminadas, mejoras de velocidad en aplicaciones de cámara y de búsqueda entre otras.

- **E: Éclair (v2.0 / v2.1)** Lanzado el 26 de octubre del 2009, se caracteriza por: velocidad de hardware optimizada, interfaz de usuario renovada, nuevas listas de contactos, mejoras en Google Maps, zoom digital, fondos de pantalla animados, soporte integrado de flash, teclado virtual mejorado entre otras.

- **F: Froyo (v2.2)** fue lanzado el 20 de mayo del 2010, tiene las siguientes características: optimización general del sistema Android, la memoria y el rendimiento, mejora en la velocidad de las aplicaciones, lanzador de aplicaciones mejorado con accesos directos a las aplicaciones de teléfono, actualización del Market con actualizaciones automáticas, permite desactivar el tráfico de datos, compartir contactos por Bluetooth, 2.2.1 / 2.2.2 / 2.2.3 actualizadas durante el 2010 con arreglos de errores.

- **G: Gingerbread (v2.3)** Fue lanzado el 6 de diciembre del 2010 y contenía: soporte para dispositivos móviles, actualización del diseño de la interfaz del usuario, soporte para pantallas extras grandes y resoluciones mayores, soporte mejorado para desarrollo de código nativo, recolección de elementos ocurrentes para un mayor rendimiento, un administrador de descargas para descargar archivos grandes, soporte nativo para múltiples cámaras. Hubo mejoras al sistema y arreglos en fallas del 2.2.3 al 2.3.7.

- **H: Honeycomb (v3.0 / v3.1 / v3.2)** Fue lanzado el 22 de febrero del 2011, sus características son: mejor soporte para tablets, escritorio 3D con Widgets rediseñados, mejoras en la navegación Web, sistema multitarea mejorada, soporte para videochat mediante Google Talk, añade soporte para una gran variedad de periféricos y accesorios con conexión USB como teclados, ratones hubs, dispositivos de juego y cámaras digitales.

- **I: Ice Cream Sandwich (v4.0)** Lanzada el 19 de octubre del 2011. Es una versión que unifica el uso de cualquier dispositivo tantos teléfonos, tablets, televisores, netbooks, etc. Incluye: interfaz limpia y moderna, multitarea mejorada, gestor de tráfico de datos de internet, corrector de texto rediseñado y mejorado, captura de pantalla en 2 botones, reconocimiento de voz de usuario.

- **J: Jelly Bean (v4.1 / v4.2)** Lanzado el 9 de julio del 2012, basado en el núcleo de Linux 3.0.31. Fue una actualización incremental con el enfoque primario de mejorar la funcionalidad y rendimiento de la interfaz del usuario. Sus características son: mejora la fluidez y la estabilidad, ajuste automático de Widgets cuando se aluden al escritorio, dictado por voz mejorado, notificaciones mejoradas, con acceso más rápido a más información, Google Chrome se convierte en el navegador por defecto de Android, Gestual Mode para personas discapacitadas visualmente, cambios en la interfaz como la nueva barra de búsqueda. Google anunció la nueva versión con un comunicado de prensa bajo el eslogan “A new flavor of Jelly Bean”. El

primer dispositivo en correr Android 4.2 fue el Nexus 4 de LG y el Nexus 10 de Samsung, los cuales fueron lanzados el 13 de noviembre del 2012.

- **K: KitKat (v4.4)** Fue lanzado el 5 de noviembre del 2013 con la finalidad de unificar su plataforma operativa entre dispositivos móviles. Sus características son: simplificación de procesos para reducir el uso de memoria, integración del Google Now, contactos frecuentes tendrán prioridad en la agenda telefónica, identificación inteligente de llamadas, mensajería instantánea mediante el HangOuts.

VENTAJAS

- El código es abierto, gracias a esto cualquier persona puede realizar una aplicación para Android®.
- Hoy en día hay más de 100,000 aplicaciones disponibles para teléfonos Android®, gran parte de ellas gratuitas.
- Android® es multitarea, es capaz de hacer funcionar a la vez varias aplicaciones.
- Android® se puede personalizar totalmente la pantalla

DESVENTAJAS

- Android® es multitarea, no siempre cierra todas las aplicaciones así que hace falta tener una aplicación que cierre las que se encuentren abiertas.
- Duración de la batería, se gasta rápidamente.
- Android® es poco intuitivo, problema provocado por la interfaz.
- Android® está desfragmentado, cada modelo de teléfono móvil se ha de adaptar a Android por lo que no es la misma versión.

ANEXO 2

CÓDIGO DE BARRAS

El código de barras consiste en un sistema de codificación creado a través de series de líneas y espacios paralelos de distinto grosor. Generalmente se utiliza como sistema de control ya que facilita la actividad comercial del fabricante y del distribuidor, por lo que no ofrece información al consumidor, si no datos de operaciones aplicados a identificar productos, llevar control de inventarios, carga y descarga de mercancías, disminuir tiempos de atención en ventas.

Una de las principales ventajas es que los datos almacenados en un código de barras poder ser leído de manera precisa y rápida.

En la actualidad se están realizando varias especificaciones para implementar una nueva tecnología llamada RFID Radio Frequency IDentification por sus siglas en inglés, el funcionamiento de RFID implica que a cada producto se le integre un tag (es una etiqueta que contiene una antena transmisora). Estos productos deben ser leídos por un decodificador RFID que capta las señales de radiofrecuencia.

El mundo del código de barras, es un área de investigación y desarrollo constante, la innovación a lo largo de estos años, ha dado paso a la aparición de diversas simbologías, pues los campos de aplicación y sus características han determinado o sectorizado su uso en particular.

La simbología es considerada el lenguaje de la tecnología de código de barras. Una simbología es la forma en que se codifica la información en la estructura de las barras y espacios del símbolo de código de barras.

TIPOS DE SIMBOLOGÍAS

La diversificación de estos tipos de códigos de barras, se debe a que las simbologías fueron diseñadas para resolver problemas específicos. De acuerdo al tipo de necesidad existen o no requisitos que se deben cumplir para poder comerciar o identificar según las normas del mercado, se deben optar por el sistema de codificación más adecuado.

La selección de la simbología dependerá del tipo de aplicación donde va a emplearse el código de barras. El tipo de carácter, numérico o alfanumérico, la longitud de los caracteres, el espacio que debe ocupar el código o la seguridad, son algunos de los factores que determinarán la simbología a emplear.

Las principales características que definen a una simbología de código de barras son las siguientes:

- Numéricas o alfanuméricas
- De longitud fija o de longitud variable
- Discretas o continuas
- Auto verificación.

DENSIDAD

Es la anchura del elemento (barra o espacio) más angosto dentro del símbolo de código de barras. Está dado en mils (milésimas de pulgada). Un código de barras no se mide por su longitud física sino por su densidad.

WNR: (Wide to Narrow Ratio)

Es la razón del grosor del elemento más angosto contra el más ancho. Usualmente es 1:3 o 1:2.

QUIET ZONE: (ZONA DE SEGURIDAD O ÁREA MUERTA O ÁREA DE VACIO)

Es el área blanca al principio y al final de un símbolo de código de barras. Esta área es necesaria para una lectura conveniente del símbolo.

Las simbologías se dividen a su vez en:

- Primera dimensión (Códigos Lineales)
- Segunda dimensión (Códigos Bidimensionales)

EUROPEAN ARTICLE NUMBERING (E.A.N.) AHORA GS1

El EAN es la versión propia del UPC europea, se creó en 1976.El sistema de codificación EAN es usado tanto en supermercados como en comercios. Es un estándar internacional, creado en Europa y de aceptación mundial. Identifica a los productos comerciales por intermedio del código de barras, indicando país-empresa-producto con una clave única internacional. Hoy en día es casi un requisito indispensable tanto para el mercado interno como internacional.

El EAN-13 es la versión más difundida del sistema EAN y consta de un código de 13 cifras (uno más que el UPC) en la que sus tres primeros dígitos identifican al país, los seis siguientes a la empresa que manufactura o comercia, los tres números posteriores al artículo y finalmente un dígito verificador, que le da seguridad al sistema. Este dígito extra se genera en base a un algoritmo que multiplica y suma posiciones pares e impares basados en la cadena que lo antecede, esto confines de dar certeza a la cadena de caracteres verificándolo.

Para artículos de tamaño reducido se emplea el código EAN-8, que es la versión reducida del mismo.

A continuación veamos otros tipos de códigos de barras según la simbología de los mismos:

CÓDIGO DE BARRAS DE PRIMERA DIMENSIÓN o CÓDIGOS LINEALES

Universal Product Code (U.P.C.)

UPC es la simbología más utilizada en el comercio minorista en EEUU, pudiendo codificar solo números.

El estándar UPC (denominado UPC-A) es un número de 12 dígitos. El primero es llamado “número del sistema”. La mayoría de los productos tienen un “1″ o un “7″ en esta posición. Esto indica que el producto tiene un tamaño y peso determinado, y no un peso variable. Los dígitos del segundo al sexto representan el número del fabricante. Esta clave de 5 dígitos (adicionalmente al “número del sistema”) es única para cada fabricante, y la asigna un organismo rector evitando códigos duplicados. Los caracteres del séptimo al onceavo son un código que el fabricante asigna a cada uno de sus productos, denominado “número del producto”. El doceavo carácter es el “dígito verificador”, resultando de un algoritmo que involucra a los 11 números previos.

Esto se creó en 1973 y desde allí se convirtió en el estándar de identificación de productos, se usan desde entonces en la venta al detalle y la industria alimenticia.

Para productos cuyo tamaño es mínimo se emplea el UPC-E

La industria editorial ha agregado suplementos de dos a cinco dígitos al final del símbolo UPC-A, utilizados por lo general para la fecha de publicación o el precio:

European Article Numbering (E.A.N.) ahora GS1

El EAN es la versión propia del UPC europea, se creó en 1976.El sistema de codificación EAN es usado tanto en supermercados como en comercios. Es un estándar internacional, creado en Europa y de aceptación mundial. Identifica a los productos comerciales por intermedio del código de barras, indicando país-empresa-producto con una clave única internacional. Hoy en día es casi un requisito indispensable tanto para el mercado interno como internacional.

El EAN-13 es la versión más difundida del sistema EAN y consta de un código de 13 cifras (uno más que el UPC) en la que sus tres primeros dígitos identifican al país, los seis siguientes a la empresa que manufactura o comercia, los tres números posteriores al artículo y finalmente un dígito verificador, que le da seguridad al sistema. Este dígito extra se genera en base a un algoritmo que multiplica y suma posiciones pares e impares basados en la cadena que lo antecede, esto confines de dar certeza a la cadena de caracteres verificándolo.

Para artículos de tamaño reducido se emplea el código EAN-8, que es la versión reducida del mismo.

CÓDIGO 39

Se desarrolló en el año 1974, porque algunas industrias necesitaban codificar el alfabeto así como también números en un código de barras, Tiene un uso difundido en el medio para identificar inventarios y para propósitos de seguimiento en las industrias, es decir esta simbología es actualmente la más usada para aplicaciones industriales y comerciales para uso interno ya que permite la codificación de caracteres numéricos, letras mayúsculas y algunos símbolos como -, ., $, /, +, % y “espacio”. Se utilizan sólo dos grosores tanto para barras como para espacios.

Particularmente no recomendamos el uso de esta simbología, a pesar de encontrarse fuentes TrueType en el mercado que ayudan a su masificación en nuestro medio. El código 39 produce una barra relativamente larga y puede no ser adecuada si la longitud es un factor de consideración.

CÓDIGO 128

Este código de barras fue creado en 1981 y se utiliza cuando es necesaria una amplia selección de caracteres más de lo que puede proporcionar el Código39. El Código 128 utiliza 4 diferentes grosores para las barras y los espacios y tiene una densidad muy alta, ocupando en promedio sólo el 60% del espacio requerido para codificar información similar en Código 39. Puede codificar los 128 caracteres ASCII.

Cuando la dimensión de la etiqueta es importante, el código 128 es una buena alternativa porque es muy compacta lo que resulta en un símbolo denso. Esta simbología se usa a menudo en la industria de envíos donde el tamaño de la etiqueta es importante.

ENTRELAZADO 2 de 5 (Interleaved 2 of 5)

Otra simbología muy popular en la industria de envíos, el entrelazado 2 de 5 es ampliamente usada por la industria del almacenaje también. Es una simbología compacta la hemos visto en cajas de cartón corrugado que se utilizan para ser enviadas en los almacenes.

Se basa en la técnica de intercalar caracteres permitiendo un código numérico que utiliza dos grosores. El primer carácter se representa en barras, y el segundo por los espacios que se intercalan en las barras del primero. Es un código muy denso, aunque siempre debe haber una cantidad par de dígitos. La posibilidad de una lectura parcial es alta especialmente si se utiliza un lector láser. Por lo tanto, generalmente se toman ciertas medidas de seguridad, como codificar un carácter de verificación al final del símbolo

CODABAR o NW7

El Codabar aparece en 1971 y encuentra su mayor aplicación en los bancos de sangre, donde como medio de identificación y verificación automática son indispensables, es una simbología de longitud variable que codifica solo números. Utiliza dos tipos de grosores para barras y espacios y su densidad es similar a la del Código 39.

POSNET

Es sólo para el Servicio Postal de Estados Unidos, esta simbología codifica los códigos postales para un procesamiento más rápido de entrega del correo. Su aparacicion, el año 1980

CÓDIGOS DE BARRAS DE SEGUNDA DIMENSIÓN o BIDIMENSIONALES

La principal ventaja de utilizar códigos de 2 dimensiones es que pueden contener una gran cantidad de información que puede ser leída de manera rápida y confiable, sin necesidad de acceder a una base de datos en donde se almacene dicha información (el caso de los códigos de 1 dimensión) e inclusive esta información físicamente puede ser contenida en un espacio aun menor a un código de 1 dimensión.

Su estructura los hace poder capaz de soportar maltrato y deterioro. Los códigos de 2D se pueden construir con muchos grados de redundancia, duplicando así la información en su totalidad o sólo los datos vitales. La redundancia aumenta las dimensiones del símbolo pero la seguridad del contenido se incrementa notablemente.

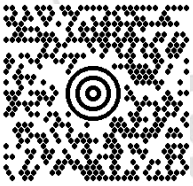
Se han realizado pruebas con algunas simbologías 2D, dañando físicamente hasta un 40% de la estructura del código y han sido leídos perfectamente. (Este es un atributo que dependiendo de la estructura de los datos puede variar en un rango de 20% a 40%).

PDF 417

Conocido como un código de dos dimensiones, es una simbología de alta densidad no lineal. Pero la diferencia entre éste y los otros tipos de código de barras, es que el PDF417 es en realidad un Portable Data File (Archivo apilado) es decir, no se requiere consultar a un archivo, este contiene toda la información, ya que tiene una capacidad de hasta 1800 caracteres numéricos, alfanuméricos y especiales. Un código así, permite almacenar nombre, foto y alguna otra información pertinente.

El código consiste en un patrón de marcas (17,4), los subjuegos están definidos en términos de valores particulares de una función discriminadora, cada subjuego incluye 929 codewords (925 para datos, 1 para los descriptores de longitud y por lo menos 2 para la corrección de error) disponibles y tiene un método de dos pasos para decodificar los datos escaneados. Cuenta con mecanismos de detección y corrección de errores: 9 niveles de seguridad lo que permite la lectura y decodificación exitosa aun cuando el daño del código llegue hasta un 40%.

MAXICODE



Es una simbología de alta densidad creada por UPS (United Parcel Service). En la actualidad esta simbología es de dominio público y está especificada bajo las normas ANSI (MH10.8.3M-1996). Es utilizado para procesamiento de información a alta velocidad.

La estructura del Maxicode consiste de un arreglo de 866 hexágonos utilizados para el almacenamiento de datos en forma binaria. Estos datos son almacenados en forma seudo-aleatoria. Posee un blanco o “bull” utilizado para localizar a la etiqueta en cualquier orientación. Es posible codificar hasta 100 caracteres en un espacio de una pulgada cuadrada. Este símbolo puede ser decodificado sin importar su orientación con respecto al lector. La simbología utiliza el algoritmo de Reed-solomon para corrección de error. Esto permite la recuperación de la información contenida en la etiqueta cuando hasta un 25 % de daño en la etiqueta.

DATAMATRIX

La Datamatrix de Siemens, es un código bidimensional diseñado para almacenar mucha información en un espacio muy pequeño. Un símbolo de la matriz de datos puede almacenar entre un y 500 caracteres. El símbolo es también escalable entre un 1 cuadrado de milipulgada a un cuadrado de 14 pulgadas, significa que un símbolo de la Datamatrix tiene una densidad teórica máxima de 500 millones de caracteres por pulgada, La densidad práctica, por supuesto, será limitada por la resolución de la tecnología de la impresión y de la lectura usada. El código tiene varias otras características interesantes. Puesto que la información es codificada por la posición algo relativa absoluta del punto de la posición del punto, no es tan susceptible a los defectos de la impresión como es la clave de barras tradicional. El esquema de codificación tiene un de alto nivel de la redundancia con el "scattered" a través del símbolo. Según la compañía, esto permite que el símbolo sea leído correctamente incluso si la parte de ella falta. Cada símbolo Datacode tiene dos lados adyacentes impresos como barras sólidas, mientras que los lados adyacentes restantes se imprimen como serie de puntos cuadrados equidistantes. Estos patrones se utilizan para indicar la orientación y la densidad de la impresión del símbolo.
Existen 2 subconjuntos principales de símbolos de Datamatrix. Desarrollados en 1989 por International DataMatrix Inc. La versión de dominio público es la ECC 200, y la otra también por International DataMatrix en 1995 que tiene una capacidad alfanumérica de 2335 caracteres.

Usando una codificación para la corrección de error, fueron utilizadas para la mayor parte de las instalaciones iniciales de los sistemas de Datamatrix, estas versiones se refieren como ECC-000 a ECC-140. El segundo subconjunto es ECC-200 referido y utiliza técnicas de la corrección de error de Reed-Solomon. ECC-000 a 140 símbolos todos tienen un número impar de módulos a

lo largo de cada lado cuadrado. Los símbolos ECC-200 tienen un número par de módulos en cada lado. La capacidad de datos máxima de un símbolo ECC-200 es 3116 dígitos numéricos, o 2335 caracteres numéricos alfa, en módulos de un símbolo 144 ajustan.

Los usos más populares para Datamatrix son la marca de pequeños artículos tales como circuitos integrados y tarjetas de circuitos impresos. Estos usos le permiten usar la capacidad de codificar aproximadamente cincuenta caracteres de datos en un símbolo 2 o 3m m cuadrados y el hecho de que el código se puede leer con solamente un cociente del contraste del 20 por ciento. El código es leído por la cámara de vídeo del CCD o scanner CCD. Los símbolos se pueden leer en distancias que se extienden de contacto a 36 pulgadas lejos.

CÓDIGO DATASTRIP

El código de Datastrip fue llamado Softstrip y desarrollado originalmente por Softstrip Systems. Es el más viejo de los símbolos de dos dimensiones. Este código es propiedad en la actualidad por Datastrip Inc. Es un sistema patentado de codificación y de la exploración que permite, datos, gráficos e incluso convirtió el sonido a digital imprimiéndose en papel normal bajo un formato altamente condensado y leído sin error en una computadora.

Los componentes principales de son patrones gráficos impresos (el Datastrip) y lectores electro ópticos. Un código de Datastrip consiste en un patrón de matriz, abarcando áreas blancos y negros muy pequeños, rectangulares (o DiBits). Los marcadores abajo del lado y a través de la tapa de la tira (línea, tablero de damas y estante del comienzo) contienen la información de la alineación para los lectores de código de Datastrip y aseguran integridad

de datos. La información principal contiene los detalles sobre los datos almacenados en la tira: nombre de fichero, número de octetos, densidad de la tira de los datos, etc. El método de codificación de Datastrip, incluye pedacitos de paridad en cada línea codificada, ofrece capacidades excelentes de corrección de la confiabilidad y error. Las tiras de los datos tienen típicamente hasta 5/8 pulgada de ancho y hasta 9 pulgadas de largo. La densidad de datos puede variar a partir 150 a 1.000 octetos por pulgada cuadrada, dependiendo de la tecnología de la impresión usada para producir las tiras.

El código de Datastrip se puede producir con éxito por la mayoría de los tipos de matriz de punto, de laser (impresoras laser centralizadas muy de alta velocidad incluyendo), de chorro de tinta o de impresoras térmicas. El código de Datastrip se puede reproducir en la mayoría de los tipos de papel (papel prensa incluyendo) y de plástico, usando los procesos de impresión convencionales, extendiéndose de las fotocopiadoras de la oficina (para las tiras de una densidad más baja) a las prensas de tela de alta velocidad. Las tiras de la baja densidad (hasta 1.100 octetos por tira de 9 pulgadas) se pueden producir en la mayoría de las impresoras por punto. Las tiras que contienen hasta 3.500 octetos se pueden producir usando tecnologías de la impresión por láser. Las tiras muy de alta densidad (hasta 4.800 octetos) requieren métodos de producción más sofisticados usando técnicas fotográficas. Los códigos de Datastrip se leen con sus scanners propietarios. y el lector debe estar en contacto con el código. Este código fue promovido originalmente para promover la publicación de software en libros y compartirlos en una forma legible por una máquina. El código ahora es usado por los sistemas de impresión en tarjetas de identificación.

DOT CODE A (Código de Puntos A)

El Dot Codes A (también conocido como el código de puntos de Philips) es uno de un número limitado de símbolos de código del punto. Estos símbolos fueron diseñados para la identificación única de objetos en un área relativamente pequeña, o para la marca directa por tecnologías de marcaje de precisión. El símbolo consiste en un arsenal cuadrado de puntos que se extienden a partir de 6 x 6, a 12 x 12, hasta la última que va mas de 42 billones, Billón, Un billón de artículos individuales pueden distinguirse o codificarse. Los usos incluyen la identificación de la cristalería de laboratorio y el marcaje de prendas en lavandería.

3-DI

3-DI fue desarrollado por Lynn Ltd y es un código propietario. 3-DI utiliza pequeños símbolos circulares. Es el más adecuado para las marcas de identificación en las superficies brillantes, metálicos curvos, como los instrumentos quirúrgicos.

ARRAYTAG

ArrayTag fue inventado por el Dr. Warren D. Little de la University of Victoria y es un código propietario. El símbolo se compone de símbolos elementales hexagonales con un borde patentado complementando que se imprimen ya sea solos o en grupos secuenciados. ArrayTags puede codificar cientos de personajes y puede ser leído a distancias de hasta 50 metros y está optimizado para la lectura a distancia o en situaciones de iluminación variable.

La principal aplicación del código es para hacer un seguimiento troncos y madera.

AZTEC CODE

El Código Aztec fue inventado por Andy Longacre de Welch Allyn Inc. en 1995 y es de dominio público. Fue diseñado para la facilidad de impresión y facilidad de decodificación. Los símbolos son cuadrados en general en una cuadrícula con una plaza central, buscador de ojo de buey. El símbolo de código Aztec más pequeño es 15×15 módulos cuadrados, y el más grande es de 151×151. El símbolo más pequeño de código de Aztec codifica 13 caracteres alfabéticos o 12 numéricos, mientras que el mayor símbolo de código de Aztec codifica 3832 números o 3067 caracteres alfabéticos o 1914 bytes de datos. No se requiere una zona tranquila fuera de los límites del símbolo. Hay 32 tamaños en todos, seleccionados por el usuario con cantidades de codificación Reed-Solomon de error de 5% a 95% en la región de datos. El nivel recomendado es de 23% de la capacidad símbolo más palabras en clave 3.

Todos los valores de los bits 8-pueden ser codificados. Los valores 0 a 127 se interpreta como el conjunto de caracteres ASCII, mientras que los valores de 128 - 255 se interpretan como la norma ISO 8859-1, alfabeto latino N ° 1. Dos datos no se pueden codificar caracteres, FNC1 para la compatibilidad con algunas aplicaciones existentes y las secuencias de escape de ECI para la codificación estandarizada de información de los mensajes interpretación.

CODABLOCK

Codablock es una simbología de códigos de identificación de la ICS Identcode-Systeme. Fue inventado por Heinrich Oehlmann y fue originalmente un stacker de símbolos del Código 39.

Cada símbolo Codablock contiene de 1 a 22 filas. El número de caracteres por fila es una función de la dimensión X del símbolo. En otras palabras, cada fila puede contener una cantidad variable de caracteres. Cada símbolo tiene un comienzo y un grupo de barras de parada que se extiende la altura del símbolo. Cada fila tiene un indicador de carácter fila dos, y la última fila del símbolo tiene un dígito opcional de comprobación. El programa de impresión de símbolos debe calcular no sólo el número de líneas necesarias como en las otras simbologías de stacker, sino también calcular el número de caracteres por fila y la densidad de impresión necesario para adaptarse mejor a los datos en el símbolo Codablock.

El código es un proceso continuo, de longitud variable que puede codificar un Código 39, el conjunto de caracteres (10 dígitos, letras 26, el espacio, y los símbolos 6) y no es más denso que un símbolo de código 39 con una densidad de impresión dada. Por ejemplo, la densidad de datos máxima es de 56 caracteres alfanuméricos por pulgada cuadrada con un símbolo utilizando una dimensión X de aproximadamente 7,5 milésimas de pulgada y una relación de barra de 2 a 1.

La versión actual es Codablock F que es básicamente un stacker de códigos de 128 símbolos. Un símbolo Codablock F consta de entre 2 y 44 filas, cada uno de hasta un máximo de 62 caracteres de ancho de símbolo. Cada fila puede ser leído por un lector de código estándar de 128 y contiene información

relativa sobrecarga adicional a la fila de numeración y el tamaño del símbolo para permitir la salida decodificada de cada fila para ser re-ensamblados para reproducir el mensaje completo en la secuencia correcta.

La ventaja de este código es que puede ser leído por escáneres de haz láser con muy pocas modificaciones. Codablock fue adoptada por los bancos de sangre en alemania para la identificación de sangre.

CODE 1 (Code One)

Código1 fue inventado por Ted Williams en 1992 y fue la primera de dominio público. Utiliza un patrón de busqueda de barras horizontales y verticales que cruzan el centro del símbolo. El símbolo puede codificar datos ASCII, los datos de corrección de errores, y datos binarios codificados. Hay 8 tamaños que van desde el código con el código 1A , 1H. Código 1A puede contener 13 caracteres alfanuméricos o dígitos 22, mientras que el código 1H puede contener 2218 caracteres alfanuméricos o dígitos 3550. La versión más grande símbolo mide 134x 148x de ancho por alto. El propio código se puede hacer en muchas formas, tales como una L, U o en forma de T.

Código1 se utiliza actualmente en la industria del cuidado de salud para etiquetas de los medicamentos y la industria de reciclaje para codificar contenido de los contenedores y su clasificación.

CODE 16K

Código 16K fue desarrollado por Ted Williams en 1989 para proporcionar una sencilla forma de imprimir y decodificar múltiples filas. Williams también desarrollo el Código 128, y la estructura de 16K se basa en el código 128. No es coincidencia, 128 al cuadrado pasó a ser igual a 16.000 o 16K, para abreviar. El Código 16K resolvio un problema inherente con el código 49. Código 49 requiere una gran cantidad de memoria para las tablas de codificación y decodificación de algoritmos. 16K es una simbología apilada.
Cada símbolo de código de 16K contiene de 2 a 16 filas, con 5 caracteres ASCII por fila. Además, hasta 107 de 16 hileras símbolos pueden concatenar hasta 8,025 caracteres ASCII, o 16.050 dígitos numéricos. En el modo extendido, los tres primeros caracteres en cada símbolo fila 16 define el carácter modo, el orden del símbolo fila 16 en el bloque, y el número total de símbolos en el bloque.

El código es un proceso continuo, de longitud variable que puede codificar el ASCII completo de 128 caracteres. El valor mínimo de la dimensión X es de 7,5 milésimas de pulgada para un símbolo para ser leídos por un lector desconocido. La altura de la barra mínima es de 8 veces y la dimensión X es La densidad de datos máxima de 208 caracteres alfanuméricos por pulgada cuadrada o 417 dígitos numéricos por pulgada cuadrada cuando el símbolo se imprime en 7,5 milésimas de pulgada. En la industria del cuidado de la salud por ejemplo, un símbolo de código de 16K impreso con un 7,5 millones dimensión X , un número de 10 dígitos , la fecha de vencimiento de 5 dígitos, y un código de lote de 10 caracteres alfanuméricos, podría encajar en un símbolo de medición sólo 0.35 pulgadas de .61 pulgadas.

El Código 16K puede ser leído escaners láser o CCD. Las filas se pueden escanear en cualquier orden. Después de que la última fila se ha escaneado, el

lector de código de barras coloca la información en la secuencia correcta. Las etiquetas pueden ser impresas por las tecnologías de impresión estándar.

Code 49

Código 49 fue desarrollado por David Allais en 1987 en la Corporación de Intermec para cubrir una necesidad de llevar grandes cantidades de información en un símbolo muy pequeño. Código 49 logra esto mediante el uso de una serie de símbolos de código de barras apiladas una encima de otra. Cada símbolo puede tener entre dos y ocho filas. Cada fila se compone de una zona tranquila de liderazgo, un modelo de partida, cuatro palabras de datos que codifican ocho caracteres, con el último carácter de un carácter de verificación fila, un patrón de parada, y una zona final tranquilo. Cada fila codifica los datos en exactamente 18 bares y espacios de 17, y cada fila está separada por una barra de separación de un módulo de alta (separador de fila).

El código tiene una longitud variable que puede codificar el ASCII completo de 128 caracteres. Su estructura es en realidad un cruce entre la UPC y el código 39. Intermec ha puesto el código en el dominio público.

El valor mínimo de la dimensión X es de 7,5 milésimas de pulgada para un símbolo para ser leídos por un lector . Si se asume una dimensión X de 7.5 milésimas de pulgada, y un mínimo de 8 altura del símbolo fila de .5475 pulgadas, la densidad máxima teórica es de 170 caracteres alfanuméricos por pulgada cuadrada. Para un símbolo de la industria de la salud, un número NDC de 10 dígitos, fecha de vencimiento de cinco dígitos y un código de lote de 10 caracteres alfanuméricos, el símbolo sería .3 pulgadas por .53 pulgadas. A los 15 dígitos de un codigo de un circuito impreso el número de serie,el código 49 sería sólo 0.1 pulgadas por .3 pulgadas.

El escaneado del código 49CODIGOS DE BARRAS DE SEGUNDA DIMENSIÓN o BIDIMENSIONALES/p puede hacerse escaners láser escáneres o CCD.

Intermec tiene escáners CCD que decodifican código 49, junto con las simbologías de códigos de barras estándar. Las etiquetas pueden ser impresas por las tecnologías de impresión estándar.

COLORCODE

Desarrollado por investigadores de la Universidad de Yonsei en Corea, ColorCode ™ posee la propiedad de ser bidimensional y está diseñado para almacenar direcciones URL y ser leído por una cámara de teléfono celular. Permite un reconocimiento de códigos indexados, que son a su vez vinculados a los datos. La matriz de bloques y los datos analógicos correspondientes al número de colores son digitalizadas y luego procesados por un servidor dedicado usando direcciones registradas en los códigos.

CP CODE

CP Código es un código propietario desarrollado por CP Tron, Inc. se compone de símbolos matriz cuadrada con un Buscador de periféricos en forma de L y de las marcas adyacentes de tiempo. Visualmente es similar a un código Datamatrix

DATAGLYPHS

DataGlyphs es un código propietario desarrollado por Xerox PARC. El código se compone de un patrón de fondo gris de pequeño “\” y “/” datos binarios de codificación, incluyendo los patrones de sincronización y corrección de errores. Cada marca puede ser tan corto como 1/100 de una pulgada (0,25 mm). Las densidades de 1000 de 8 bits por pulgada cuadrada se puede usar este código. DataGlyph es tolerante a las marcas de tinta, copias malas, e incluso grapas a través del símbolo debido a la corrección de errores internos.
Los DataGlyphs están diseñados para combinar el diseño del producto impreso. DataGlyphs pueden ser logos o tonos detrás de texto o gráficos. Las aplicaciones incluyen cuestionarios, correo directo formularios de respuesta y las encuestas y tarjetas de visita y se leen utilizando un escáner de imágenes.

HCCB

El color de alta capacidad en formato de código de barras de código de barras se aprovecha de los dispositivos avanzados de imagen de la computadora junto con el poder de procesamiento para permitir mayor densidad de almacenamiento de datos sobre medios de comunicación impresos análogos. El formato logra esto mediante el uso de una forma de código de barras ,símbolos diferentes en combinación con colores múltiples por símbolo. Se trata de un código propietario.

HUECODE

HueCode es un código propietario desarrollado por Robot Design Associates. El código se compone de bloques de células que contienen más de un bit por celda. Esto se hace mediante el uso de tonos de gris o color. El símbolo se puede imprimir en plástico o papel. Densidades de información varían de acuerdo con las técnicas específicas utilizadas, pero bajo rango entre 640 bytes / sq. pulgadas utilizando impresoras láser a más de 40.000 Bytes / sq. in uso de las impresoras de sublimación de tinta. HueCode se lee con un escáner de superficie plana establecido en 400×400 dpi Software y de propiedad. El código está destinado a almacenar la información de texto sobre las espaldas de tarjetas de visita o tarjetas médicas.

INTACTA.CODE

INTACTA.CODE ™ es un código propietario desarrollado por Tecnologías intacta, Inc. y pueden tomar los datos binarios, tales como los archivos ejecutables, vídeo, texto, audio (o una combinación de archivos) y luego aplicar la compresión INTACTA.CODE ™ , que lo codifica y los motores de corrección de errores crean una envoltura para los datos que le permite ser distribuido de forma segura mientras se mantiene la integridad del formato y el contenido.

MINICODE

MiniCode fue desarrollado por Omniplanar, Inc. (ahora propiedad de Honeywell) y es propietario. Se compone de los símbolos de matriz cuadrada con un método patentado de codificación de datos, tanto de baja resolución (de seguimiento / aplicaciones de clasificación), y datos de alta resolución (envío de solicitudes de manifiesto).

QR Code

Código QR (Quick Response Code) es un código matriz desarrollada por Nippondenso ID Systems y es de dominio público. los codios QR Code son de forma cuadrada y puede ser fácilmente identificado por su patrón buscador de anidados se alternan casillas claras y oscuras en tres esquinas del símbolo. Tamaño del símbolo máximo es de 177 módulos cuadrados, capaces de codificar 7366 caracteres numéricos, o 4464 caracteres alfa numéricos. Una característica importante de la simbología es su capacidad para codificar los caracteres kanji y kana japoneses directamente. Código QR está diseñado para la lectura rápida el uso de cámaras CCD y la tecnología de procesamiento de imágenes debido a la disposición del patrón del buscador

SNOWFLAKE CODE

Snowflake Code es un código propietario desarrollado por Electronic Automation Ltd en 1981. El código es una matriz cuadrada de puntos discretos, es posible codificar más de 100 dígitos numéricos en un espacio de sólo 5 mm x 5 mm. el factor de corrección de error permite hasta 40% del código dañado y pudiendo ser legible.

El código se utiliza en la industria farmacéutica y tiene una ventaja de que puede ser aplicado a productos y materiales en una amplia variedad de formas, incluyendo las etiquetas impresas, de chorro de tinta de impresión, grabado por

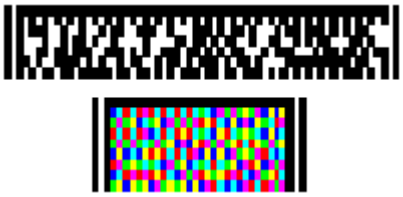
láser, sangría o perforación de orificios. Automatización Electrónica Ltd. fue adquirido en marzo de 1999 por Videojet Systems International Inc., una división de Marconi Data Systems Inc., que ahora ya no existen.

SuperCode

SuperCode fue inventado por Ynjiun Wang en 1994 y es de dominio público. La simbología utiliza una estructura de paquete, una variante de una simbología de varias filas. Hay reglas precisas para la colocación horizontal de los caracteres de símbolo en un paquete, pero una mayor libertad en la colocación de paquetes de forma vertical y horizontal que ofrece una matriz de columnas y filas en una simbología de varias filas. La estructura del paquete de SuperCode asegura que cada carácter de símbolo que codifica una palabra de código de corrección de datos o error es adyacente a un carácter de símbolo que codifica la dirección del paquete. Así, la secuencia de palabras de código se conoce con independencia de cómo los paquetes están dispuestos. Esto no sólo permite formas de símbolos no rectangulares, pero los paquetes no tienen estar entre sí físicamente.

El número máximo de caracteres de datos por símbolo en el nivel más bajo de la corrección de errores es 4.083 caracteres de datos alfanuméricos, números 5.102, o 2.546 bytes. SuperCode tienen corrección de errores, en base a algoritmos de corrección de errores Reed-Solomon , que pueden ser utilizados no sólo para detectar errores sino para corregir palabras del código erróneamente descodificados o faltantes. Un usuario puede seleccionar uno de los 32 niveles de corrección de errores.

ULTRACODE



Ultracode fue desarrollado por Zebra Technologies y es de dominio público. El símbolo se compone de una tira de longitud variable de columnas de píxeles. El código incluye los modos numéricos y alfanuméricos, con un avanzado lenguaje / código de la página disposiciones de manejo, y niveles seleccionables de corrección de errores Reed-Solomon. Tanto en blanco y negro como en la versión de mayor densidad de color son compatibles. La simbología utiliza pares de columnas verticales de cualquiera de las siete células blanco y negro (claro / oscuro) u 8 multicolores (normalmente blanco, rojo, verde y azul o cian, magenta, amarillo y negro) para codificar cada dato como un punto en los planos de carácter de un grupo plano lenguaje 43.

Las simbologías ULTRACODE difieren de las bidimensiones, por su corrección de errores , de aspecto similar a los códigos de barras lineales no están posicionados como de alta capacidad. ULTRACODE es especialmente adecuado para la impresión directa con una precisión lineal baja.

ANEXO 3

LECTOR DE CÓDIGOS DE BARRAS

El lector de códigos de barras es un dispositivo electrónico que por medio de un láser lee un código de barras y emite el número que muestra el código de barras, no la imagen. Básicamente, consiste en el escáner propiamente dicho (que mediante un láser lee el código), un decodificador y un cable o antena Wi-fi que actúa como interfaz entre el decodificador y el terminal o la computadora.
La función del escáner es leer el símbolo del código de barras y proporcionar una salida eléctrica a la computadora, correspondiente a las barras y espacios del código de barras. Sin embargo, es el decodificador el que reconoce la simbología del código de barras, analiza el contenido del código de barras leído y transmite dichos datos a la computadora en un formato de datos tradicional. Tiene varios medios de conexión: los más modernos por orden de aparición USB, Bluetooth, Wi-fi, los más viejos puerto serie, incluso directamente al puerto PS2 del teclado por medio de un adaptador, cuando se pasa un código de barras por el escáner es como si se hubiese escrito en el teclado el número del código de barras.

Un escáner puede tener el decodificador incorporado en el mango o puede tratarse de un escáner sin decodificador que requiere una caja separada, llamada interfaz o emulador. Los escáneres sin decodificador también se utilizan cuando se establecen conexiones con escáneres portátiles tipo “batch” (por lotes) y el proceso de decodificación se realiza mediante el terminal propiamente dicho.

El lector de códigos de barras fue, en 1974, la primera aplicación comercial del láser. El primer registro fue el precio de un empaque de chicles.

LECTURA DE CÓDIGOS DE BARRAS

Los códigos de barras se leen pasando un pequeño punto de luz sobre el símbolo del código de barras impreso. Solo se ve una fina línea roja emitida desde el escáner láser. Pero lo que pasa es que las barras oscuras absorben la fuente de luz del escáner y la misma se refleja en los espacios luminosos. Un dispositivo del escáner toma la luz reflejada y la convierte en una señal eléctrica.

El láser del escáner (fuente de luz) comienza a leer el código de barras en un espacio blanco (la zona fija) antes de la primera barra y continúa pasando hasta la última línea, para finalizar en el espacio blanco que sigue a ésta. Debido a que el código no se puede leer si se pasa el escáner fuera de la zona del símbolo, las alturas de las barras se eligen de manera tal de permitir que la zona de lectura se mantenga dentro del área del código de barras. Mientras más larga sea la información a codificar, más largo será el código de barras necesario. A medida que la longitud se incrementa, también lo hace la altura de las barras y los espacios a leer.

INTERFACES

Todas las aplicaciones pueden aceptar la salida que produce un lector de código de barras, siempre y cuando se posea el equipo necesario. Los lectores de códigos de barras se encuentran con distintas interfaces de conexión al PC. Existen modelos de lectores que tienen solamente una interfaz integrada, pero hay algunos de ellos que aceptan varias interfaces. Basta con un simple cambio de cables y una reconfiguración para utilizar una interfaz u otra.

Interfaz PS2 de teclado

Cuando se requiere que el decodificador sea de teclado se utiliza lo que se conoce como keyboard wedge, el cual se conecta a la entrada PS2 o terminal. Este tipo de lectores se conectan directamente al puerto PS2 del teclado y ofrecen una salida idéntica a la de éste. Suelen ofrecer un patrón que permite conectar al mismo tiempo un teclado y el lector. Cuando lees un código de barras el lector envía al ordenador los datos como si hubiesen sido escritos con el teclado (el número que corresponde al código de barras leído), lo que hace que su utilización sea muy sencilla con cualquier programa que espere una entrada de teclado. Sin embargo, este tipo de interfaz tiene algunos inconvenientes. Por ejemplo, la escritura del código será siempre completa, es decir, no puedes dividir el código en varias partes. El lector no es capaz de devolver cuatro cifras, y luego el resto. Obviamente, siempre hay que asegurarse que el cursor del sistema está sobre la casilla/documento que queremos rellenar, el lector no se preocupa de eso y devolverá su salida allí donde estemos situados.

Interfaz USB

Son lectores de última generación. Envían la información más rápidamente que los anteriores y su conexión es más simple. No necesitan alimentación añadida, pues la que obtienen por esta interfaz es suficiente.

RS-232

Los escáneres que se conectan a la interfaz RS-232 (o interfaz serie) necesitan utilizar un software especial que recupera la información enviada por el escáner de códigos de barras y la coloca allí donde se le indique. Esta interfaz es algo más sofisticada que la de teclado, y nos ofrece un mejor control sobre el destino de la lectura del código.

TIPOS

Existen cuatro tipos principales de lectores:

Lápiz Óptico

Uno de los más populares ha sido desde siempre el lector denominado lápiz óptico, que durante años ha sido el elegido, sobre todo de comercios minoristas, debido a su reducido costo y su pequeño y manipulable tamaño.

No obstante, muchos usuarios prefieren utilizar otro tipo de dispositivos, debido a que los lectores lápiz requieren una amplia habilidad del encargado de manejar el dispositivo, no suelen ser muy resistentes a los golpes y caídas, y ofrecen una lectura lenta e incluso a veces no logran captar la información del código, ya que sólo funcionan bien con códigos que han sido impresos en una excelente calidad.

Scanner CCD (Charge Coupled Device)

Otro modelo muy utilizado suelen ser aquellos escáner del tipo CCD, los cuales utilizan un sensor foto detector del tipo CCD, es decir un dispositivo de carga acoplada, que contiene un conjunto de LEDs, que se encargan de emitir fuentes de luz y forma para obtener la información del código.

Si bien este tipo de dispositivos requieren que el código se halle en contacto físico con el lector para hacer posible la lectura, lo cierto es que ofrecen una lectura rápida y eficaz, ya que al contrario del escáner de lápiz óptico no producen degradación de la imagen cuando ésta es escaneada.

Dentro de los lectores del tipo CCD, también se encuentra un modelo que si bien no requiere contacto directo con el código, ya que permiten una lectura por proximidad, lo cierto es que no funcionan de manera correcta ante superficies irregulares.

En cuanto al tipo de lectores láser, la mayoría de los modelos disponibles en la actualidad suelen permitir una lectura eficaz y veloz por proximidad al código de barras.

Dispositivos Láser

Debido a la potente tecnología de la luz láser brindan mejores resultados que el escáner del tipo CCD y lápices ópticos, permitiendo una lectura correcta en cualquier tipo de superficie, independientemente de si el código se halla impreso en una superficie curva o irregular.

- **Dispositivo láser tipo pistola:** Es uno de los dispositivos láser más utilizado, que funcionan por intermedio de un mecanismo que activa el escáner en el momento en que se encuentra enfrentado al código que se desea leer. Gracias a este procedimiento, es posible evitar la lectura accidental de códigos.

Los lectores láser de pistola están compuestos por un espejo que oscila dentro del dispositivo y que hace posible el recorrido a través de toda la superficie del código de barras, sin necesidad de que el usuario deba mover el lector.

Su ventaja radica en que permite la lectura de códigos en cualquier tipo de superficie, incluso cuando éstos se hallan en mal estado, y por otra parte son uno de los dispositivos más resistentes a condiciones hostiles.

En la mayoría de los casos, permiten una lectura a una distancia máxima de 20 cm, aunque también existen lectores especiales que ofrecen la posibilidad de alcanzar lecturas a una distancia de hasta 5 metros.

- **Escáner Láser fijos:** Similares al tipo de lectores láser de pistola, los denominados escáner láser fijo, funcionan de la misma manera que el anterior, pero su diferencia radica en que deben fijarse a una superficie y no necesitan ser manipulados por el usuario.

Láser fijo Omnidireccionales

Como su nombre lo indica se caracterizan por permitir la lectura de códigos en cualquier dirección, ya que se componen de un conjunto de espejos que producen un patrón unidimensional. Son los que comúnmente se utilizan en los supermercados, ya que son lo más recomendados para obtener una lectura precisa e inmediata, sin errores.

Tanto los lectores láser, como los CCD y los omnidireccionales se configuran leyendo comandos de programación impresos en menús de códigos de barras. Hay algunos que se configuran con interruptores dip, o enviándoles los comandos de programación vía línea serie también sirven como lectores manuales.

TERMINALES PORTÁTILES

Los terminales portátiles se utilizan para colección de datos en lugares donde es difícil llevar una computadora, como en un almacén o para trabajo en campo. Generalmente se diseñan para uso industrial. Las terminales portátiles cuentan con display pequeño, teclado, puerto serie, puerto para conexión de un lector externo de código de barras y son programables. Algunas de ellas tienen el lector de código de barras integrado, y éste puede ser laser, CCD o lápiz. La memoria RAM con que cuentan puede variar de unos 64K hasta 4 MB en terminales más sofisticadas. Las terminales más sofisticadas tienen radios, permitiéndose así una interacción en línea con el host.

La forma en que se programan depende de la marca y del modelo: Pueden tener un lenguaje nativo o programarse mediante un generador de aplicaciones que genera un código interpretable por la terminal. Algunas tienen sistema operativo MS-DOS y consiguientemente pueden programarse en lenguajes de

alto nivel. Los lectores soportados por la mayoría de éstas terminales son HHLC (CCD o laser) y lápiz óptico (wand emulation).

FORMA DE USO DE LAS TERMINALES

Una operación típica de una de estas terminales es la siguiente:

- Aparecen preguntas en pantalla
- Se leen los datos pedidos con el escáner o se digitan manualmente
- Se validan los mismos si es necesario
- Se repite el procedimiento las veces que sea necesario
- Cuando se tiene la información completa, se descargan los datos vía serial a una computadora en donde finalmente son procesados.

Obviamente pueden existir otras variantes, pero el manejo básico de estas terminales es el mismo.

ANEXO 4

SISTEMA DE RECONOCIMIENTO DE LÍNEAS DEL CÓDIGO DE BARRAS ASISTIDO POR COMPUTADORA (COMPUTER AIDED DESIGN MATLAB®)

```
I=imread('dia_1.jpg');
figure(1);
imshow(I);
title('Imagen original de código de barras');
R1=I(:,:,1);
G1=I(:,:,2);
B1=I(:,:,3);

Y=0.299*R1 + 0.587*G1 + 0.114*B1;
Z=Y;
figure(2);
imshow(Y);
title('Imagen de código de barras en escala de grises');
thresh=0.1;
sigma=5;

C=edge(Y,'canny',thresh,sigma);
BW=C;
figure(3);
imshow(C);
title('Deteccion de bordes de la imagen de código de barras
mediante el operador Canny');
[H,theta,rho] = hough(BW);
peak = houghpeaks(H);
barAngle = theta(peak(2));
J=imrotate(Z,barAngle,'bilinear','crop');
```

```
figure(4);
imshow(J);
title('Rotación de imagen de código de barras mediante la
transformada de Hough');
imD=double(J);
[f,c]=size(imD);
    for i=1:f
        for j=1:c
            if imD(i,j)<=124
                nuevaI(i,j)=0;
            else
                nuevaI(i,j)=255;
            end
        end
    end
imB = uint8(nuevaI);
figure(5);
imshow(imB);
title('Binarización de la imagen de código de barras
mediante el método basado en la forma del histograma');

%%%%%%%%%%VARIABLES MATEMATICA CODIGO DE BARRAS%%%%%%%%%%
K=imB;    %COPIA DE LA MATRIZ PARA ANALIZAR
G=imB;    %COPIA DE LA MATRIZ PARA ANALIZAR
[f1,c1]=size(K);
f1=f1;
c5=c1;
c2=c1;
f2=f1;

%%%%%%%%%%VERIFICAR SI NUMERO ES PAR%%%%%%%%%%
```

```
if(mod(f1,2)~=0) % Ver modulo de numero base 2
        f1=(f1-1)/2;
else
        f1=f1/2;
end

%%%%%%%%%%COPIA DE FILA PARA ANALISIS%%%%%%%%%%
i=f1;
i2=f1;

%%%%%%%%%%TAMAÑO DE LA PRIMERA BARRA NEGRA%%%%%%%%%%
V=zeros(1,c2);
b=1;
    for j=1:c1
        if imB(i,j)<=0
           V(b)=V(b)+1;
         else
            b=b+1;
        end
    end

%%%%%%%%%%VARIABLES Y VECTORES%%%%%%%%%%
[f3,c3]=size(V);

f4=f3;
c4=c3;
i=f3;
VINT=zeros(1,50);    %VECTOR CON ANCHOS DE BARRAS
VCOM=zeros(1,50);    %VECTOR CON COMIENZOS DE BARRAS
d=1;
```

```
%%%%%%%%%%CREAR MATRIZ PARA EXTRAER EL INTERVALO%%%%%%%%%%
VCOMACU=0;
for j=1:c3
    if V(i,j)>0
        VINT(d)=V(i,j);
        VCOM(d)=j+VCOMACU;
        VCOMACU=V(i,j)+VCOMACU;
        d=d+1;
    else
    end
end

%%%%%%%%%%SE SACA UN PROMEDIO ENTRE LAS DOS PRIMERAS
BARRAS%%%%%%%%%%
INT=VINT(1);

%%%%%%%%%%IDENTIFICACIÓN DE BARRAS%%%%%%%%%%
INT2=INT;   %VARIABLE PARA HAYAR LA MITAD DE LA BARRA
if(mod(INT2,2)~=0) % Ver modulo de numero base 2
        INT2=(INT2-1)/2;
else
        INT2=INT2/2;
end
VARCOM1=VCOM(1);%+INT2;    %IDENTIFICACION DE BARRAS MITAD
PRIMERA BARRA

%%%%%%%IDENTIFICACIÓN DE NÚMERO DE BARRAS NEGRAS%%%%%%%%%%
[VCOMF,VCOMC]=size(VCOM);
a=1;
CONTBN=0;
for j=1:VCOMC
```

```
    if VCOM(a)==0
        a=a+1;
    else
        CONTBN=CONTBN+1;
        a=a+1;
    end
end

%%%%%%%%%%PROCESO DE DETECCIÓN%%%%%%%%%%
b=1;
VRESUL=zeros(1,100);
e=1;
ACUMFIN=0;
ACUMFIN1=0;
ACUMFIN2=0;
  for i=1:CONTBN
      VARCOM1=VCOM(b);
      b=b+1;
      VARCOM2=VCOM(b);
      for i3=1:7
      if VARCOM1+(INT2*2)<=VARCOM2
        for j=VARCOM1:VARCOM1+INT
            if G(i2,j)==0
                ACUMFIN1=0;
                ACUMFIN=ACUMFIN+ACUMFIN1;
            else
                ACUMFIN1=255;
                ACUMFIN=ACUMFIN+ACUMFIN1;
            end
        end
        ACUMFIN=ACUMFIN/INT;
```

```
        VRESUL(e)=ACUMFIN;
        e=e+1;
        VARCOM1=VARCOM1+INT;
        ACUMFIN=0;
        else
       end
       end
  end
VRESUL=double(VRESUL);
[f,c]=size(VRESUL);
    for i=1:f
        for j=1:c
            if VRESUL(i,j)<=100
                nuevaJ(i,j) = 0;
            else
                nuevaJ(i,j) = 255;
            end
        end
    end
nuevaJ
```