UNIVERSIDAD SEÑOR DE SIPÁN

FACULTAD DE INGENIERÍA, ARQUITECTURA Y URBANISMO

ESCUELA ACADÉMICO PROFESIONAL DE INGENIERÍA DE SISTEMAS

TESIS

COMPARACIÓN DE ALGORITMOS DE DETECCIÓN DE BORDES Y VECTORIZACIÓN DE IMÁGENES DE MOLDES TEXTILES

PARA OPTAR EL TITULO PROFESIONAL DE INGENIERO DE SISTEMAS

Autor:
Bach. Vallejos Rodríguez Jair Adbeel

ORCID: https://orcid.org/0000-0003-4722-4928

Asesor:
Mg. Mejia Cabrera Heber Ivan

ORCID: https://orcid.org/0000-0002-0007-0928

Línea de Investigación:
Tecnología, Infraestructura y Medio Ambiente

Pimentel – Perú 2021

COMPARACIÓN DE ALGORITMOS DE DETECCIÓN DE BORDES Y VECTORIZACIÓN DE IMÁGENES DE MOLDES TEXTILES

Aprobación de la Tesis

_____

Bach. Vallejos Rodríguez Jair Adbeel

AUTOR

_____

Mg. Mejia Cabrera Ivan

ASESOR

_____

Dr. Ramos Moscol Mario Fernando

PRESIDENTE DE JURADO

DE TESIS

_____

Mg. Mejia Cabrera Ivan

SECRETARIO DE JURADO

DE TESIS

_____

Mg. Bances Saavedra David Enrique

VOCAL DE JURADO

DE TESIS

DEDICATORIA

A mis padres, por ayudarme a cumplir un objetivo más poniendo todo su esfuerzo, a toda mi familia por su apoyo y cariño, a Erika mi enamorada y amiga quién está siempre animándome a alcanzar algo nuevo.

AGRADECIMIENTOS

Agradezco a mi familia, a mi enamorada, amigos y a todas las personas que estuvieron apoyándome en todo momento durante mi trayecto universitario y han formado parte de mi desarrollo profesional. Agradezco a mis asesores por apoyarme durante la realización de este proyecto con sus conocimientos, su tiempo y paciencia, gracias por todo.

RESUMEN

Las industrias en el Perú están conformadas en su 99.05% por mipymes, de las cuales el 6.7% y 3.3% representan a industrias manufactureras de media y baja tecnología. Estas empresas realizan un proceso llamado patronaje, el cual resulta costoso en términos de recursos, mano de obra y tiempo trabajo, pues se realiza mediante técnicas manuales.

Esta investigación busca facilitar dicho proceso mediante la digitalización, para ello se planteó realizar moldes computarizados siguiendo un proceso que consta en la obtención de imágenes de patrones físicos mediante un protocolo de adquisición de imágenes, la detección de bordes y detección de esquinas para una futura vectorización. Existen diversas técnicas para realizar estas tareas, par ello se realizó una evaluación de las mismas para determinar que técnica se adecúa mejor a la problemática enfrentada. Se evaluó técnicas de detección de bordes como Canny-Deriche, Sobel y un método propuesto basado en la erosión de la imagen, tras la evaluación se dedujo que para este trabajo la técnica adecuada es Sobel, pues muestra calidad visual buena con un valor MSE de 1536 en tiempos de respuesta de 6.1 milisegundos, sobreponiéndose sobre Canny-Deriche y Erosión..

También se evaluó técnicas de detección de esquinas que ayudarán a identificar puntos vectorizables de una imagen, para ello se comparó las técnicas de Harris Corner, con una propuesta de dos técnicas, una evalúa las pendientes del borde de una imagen y otra técnica que recorre pixel a pixel del borde la una imagen. Tras evaluar su desempeño, se logró identificar que la técnica propuesta de recorrido Pixel a Pixel logra precisión de 90%, consumo de 0.9 MB de memoria en 0.57 milisegundos por proceso. Se concluyó que la técnica de recorrido pixel a pixel, es la mejor para reconocer esquinas pues presenta mayor precisión, consumo de memoria hasta 30 veces más bajo y tiempos de respuesta hasta 68 veces más rápido que el método tradicional de Harris.

Palabras Clave: algoritmo, detección de bordes, vectorización, detección de esquinas, imagen vectorial.

ABSTRACT

The 99.05% of Peru's industries are conformed of micro and small businesses, of which 6.7% and 3.3% represent medium and low technology manufacturing industries. These companies carry out a process called patronage, which is costly in terms of resources, labor and time, since it is done by means of manual techniques.

This research seeks to facilitate this process through digitalization, for this purpose it was proposed to make computerized molds following a process that consists of obtaining images of physical patterns through an image acquisition protocol, edge detection and corner detection for future vectorization. There are several techniques to perform these tasks, so an evaluation of them was made to determine which technique best suits the problem faced. Edge detection techniques such as Canny-Deriche, Sobel and a proposed method based on image erosion were evaluated. After the evaluation, was concluded that Sobel is the appropriate technique for this work, since it shows good visual quality with an MSE value of 1536 in response times of 6.1 milliseconds, overlapping on Canny-Deriche and Erosion.

Also evaluated were corner detection techniques that will help identify vectorizable points in an image, for which Harris Corner techniques were compared with a proposal of two techniques, one evaluating the slopes of the edge of an image and another technique that travels pixel by pixel of the edge of an image. After evaluating their performance, it was possible to identify that the proposed technique of pixel by pixel travel achieves 90% accuracy, consuming 0.9 MB of memory in 0.57 milliseconds per process. It was concluded that the technique of pixel by pixel travel, is the best for recognizing corners because it has greater accuracy, memory consumption up to 30 times lower and response times up to 68 times faster than the traditional method of Harris.

Keywords: algorithm, edge detection, vectorization, corner detection, vector image.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

I. INTRODUCCIÓN

1.1. Realidad Problemática

Las industrias en el Perú, abarcada principalmente por medianas, pequeñas y microempresas (Mipymes) representan el 99.05% del entramado empresarial, de las que el 6.7% representan manufacturas de baja tecnología y el 3.3% manufacturas de tecnología media (Ministerio de la Producción, 2017).

Los productos peruanos se encuentran en un buen nivel de preferencia por su lugar de origen a nivel mundial, estos productos mantienen una buena reputación por su calidad. Los envíos a china se incrementaron en un 21.2%, las importaciones de la misma manera aumentaron en un 22.7% y los envíos de productos peruanos al resto del mundo se incrementó en un 9.4%. Los sectores con mayor crecimiento en exportaciones fue el sector agrícola con un incremento del 26.9% y el sector textil con un incremento notable del 12.7% (La República, 2018). En el Perú las exportaciones del sector textil vienen incrementándose a partir del año 2015 que fue el año cuando más decayó, a partir de entonces su incremento es notable, estas estadísticas muestran al sector textil en representación del 7.2% del PBI manufacturero, y su impacto es positivamente notable por la desempeño económico y de los puestos de trabajo que genera (ComexPerú, 2018). Para el mes de enero del año 2019, el valor monetario de las exportaciones en prendas de vestir y otro tipo de confecciones fueron de 89.0493 millones de dólares (BCRP, 2019). Aún por el entorno positivo que representa este sector de exportación, tiene problemas de producción. En el 2015 la empresa Creditex realizó un estudio a su proceso productivo donde se evaluó puntos débiles, uno de esos puntos fue la existencia de un porcentaje elevado de mermas de tela (retazos) que representaron con un 4% del valor final de producción, parte de las causas que generaban esta cantidad de mermas se le atribuyó a la falta de control en la toma de muestras de tela (pág. 22), donde minimizar este 4% representaba un ahorro económico de 100 000 – 150 000 soles (Creditex S.A.A., 2016).

Creditex S.A.A. desarrolló un proyecto llamado 42k, los resultados que obtuvieron fue de un ahorro de 59 000 soles en saldos de tela (retazos)

(Creditex S.A.A., 2016). Si se aplicara un cambio en todas las empresas textiles, el ahorro económico sería beneficioso para las empresas.

Además, las mipymes de confecciones que laboran con tecnología media y baja, enfrentan problemas en el desarrollo de procesos manuales como el patronaje, que consta de la creación y trazado de un molde textil o patrón en la tela. Estos patrones son realizados en papel kraft, cada patrón sirve para el trazado de un molde en cada pliegue de tela, lo que genera un problema en inversión de tiempo y recursos, que se transforma en inversiones de costo por mano de obra y costo de materiales.

El problema de la obtención de muestras de tela se puede reconsiderar con otro método de solución, que además de contribuir con el ahorro económico para la empresa agilizará sus procesos de obtención de muestras. Para ello será necesario crear un método computacional, esto implica tomar muestras de los moldes a utilizar, próximamente realizar un proceso de extracción de bordes y vectorización de estas muestras, para ello existen algoritmos para realizar este proceso (detección de bordes y vectorización). El problema será determinar cuál de estos métodos es el más eficiente y adecuado para solucionar este problema de las muestras de tela. Una investigación realizada en el instituto de tecnología de Rajam en la india, detalla el uso de técnicas de detección de bordes en medicina y procesamiento de imágenes, se usa técnicas como la transformada de wavelet, incluso en preferencia sobre la técnica de canny que es una de las mejores técnicas de detección de bordes, sin embargo los resultados obtenidos en esta investigación demuestran las imágenes procesadas con wavelet no pierden información significativa y tienen mayor calidad visual (Vikas et al., 2016).

Un trabajo comparativo realizado por Ganesan, P. y Savij, G. hicieron una prueba de múltiples métodos de detección de bordes entre los que se mencionan a Canny, Sobel, Prewitt, Roberts y LoG, tras comparar los resultados de estas técnicas mencionadas los autores concluyen que la técnica que mejores resultados muestra es Canny a pesar de procesar imágenes ruidosas. Pero también mencionan que el método de Wavelet es

mejor que Canny en lo que respecta a precisión, pero menciona que wavelet es complicado de comparar con los otros métodos tradicionales (Ganesan & Sajiv, 2018).

Por otro lado se realizó un trabajo de reconocimiento de placas de automóviles a partir de detección de bordes usando el método de sobel, que para este trabajo los resultados fueron alentadores ya que los resultados visuales son buenos, el investigador utilizó Matlab para llevar a cabo sus pruebas donde resalta en muchas ocasiones el uso del software mencionado (Israni & Swapnil, 2016). Cabe resaltar que los diferentes métodos se comportan de diferentes maneras según las características de las imágenes a procesar.

Otro trabajo comparativo de métodos de detección de bordes se realizó utilizando las técnicas de sobel, canny y prewitt, para esta prueba se procesó imágenes de ultrasonido para mostrar el desarrollo del feto en el útero de la madre, los resultados fueron comparados utilizando un cuadro comparativo donde los ítems de comparación son: error cuadrático medio (mse) y proporción máxima de señal a ruido (psnr), y en este cuadro sobel se sobrepone a canny y prewitt con los resultados. (Khairudin & Irmawati, 2017)

Por otra parte con lo que respecta a la vectorización de las imágenes, según la investigación de Barina y Zemcik, ellos propusieron un método de vectorización en 2D donde aseguran proponer un método más rápido que otros métodos conocidos, mencionan el trabajo de Kutil y concluyen que el método propuesto es más rápido debido al recorrido del kernel diagonal de tamaño 2 x 2 que mencionan, diciendo que mientras más corto sea el tamaño del núcleo, menor es el tiempo de carga en la memoria por el número de espacios ocupados. (Barina & Zemcik, 2014)

Yang et al., (2016), propuso un método de vectorización de imágenes, este método se desarrolla por bezigonz (trayectorias compuestas por curvas cerradas de bezier), mencionan que otros trabajos muestran resultados menos eficientes por aproximar la representación de vectores intermedios, de modo que los bezigons resultantes son imperfectos por los errores acumulados, ambigüedades de ajuste, y falta de priors curva, estos problemas

se reflejan mucho más en imágenes de baja calidad. Este método optimiza el bezigon original, de esta manera evita los problemas antes mencionados, superando así otros métodos, incluso a softwares comerciales en términos de calidad de curvas bezier.

1.2. Antecedentes de Estudio

Ganesan & Sajiv, (2018), en su trabajo, A Comprehensive Study of Edge Detection for Image Processing Applications, presentan:

El trabajo de investigación presentado es un trabajo comparativo de los diferentes métodos creados en las últimas décadas para detectar bordes, pero aclarando que la detección de bordes es un problema de aplicación orientada, ya que los resultados del uso de cada técnica responden de manera distinta según el tipo de imagen que se procesa, se llevó a cabo este trabajo y para ello se comparó los métodos Roberts, Sobel, Prewitt, Canny y LoG. Tras poner a prueba dichos métodos, el detector de bordes de Canny dió los mejores resultados incluso en condiciones ruidosas, pero si se habla de precisión el método de wavelets es mucho más exacto y supera los métodos anteriores.

Israni, (2016), en su investigación, Edge detection of license plate using Sobel operator , presentan:

La detección de bordes es un proceso para reconocer límites de objetos y texturas. Los bordes de la imagen se consideran los atributos más importantes porque proporcionan información valiosa para la percepción de la imagen. En cuanto a la detección de bordes, los datos a procesar son muy grandes por lo que la velocidad del procesamiento de la imagen es un problema difícil. El método de detección de bordes de Sobel devuelve bordes a partir de los puntos donde la gradiente de la imagen se considera máxima. Sobel utiliza matrices de gradiente vertical y horizontal de tamaño 3x3, tras aplicar este operador con estas características se demostró que utilizando este método matemático de detección de bordes de sobel a través del simulador del software Matlab es un buen método. Se concluye que el operador sobel tiene ventajas de suavizado al procesar ruido presente en la imagen, pero para localizar bordes complejos no es exacto.

Kaur & Kaur, (2016), en su trabajo, An Edge detection technique with image segmentation using Ant Colony Optimization: A review, fundamentan:

Muchos programas basados en el procesamiento de imágenes requieren de detección y aceptación de las cosas. En la visión por ordenador, segmentar una imagen es dividir una imagen en múltiples segmentos. La segmentación de una imagen tiene como objetivo simplificar o cambiar la representación de una imagen en algo que es más importante y más fácil de analizar. En ese trabajo se desarrolló un método de detección de bordes, la optimización por colonia de hormigas, este método imita el comportamiento forrajeo de las hormigas para localizar respuestas estimadas a problemas de optimización difíciles. Con el uso de esta técnica, aunque los bordes fueron formados por líneas delgadas, continuas y bordes claros, el tiempo de funcionamiento del algoritmo es lento y solo aplica en imágenes en escala de grises. Se concluye que esta técnica de detección de bordes es mejor que otras, aunque necesita mejorar en la lentitud y en cuanto a PSNR y MSE.

Khairudin & Irmawati, (2017), en su investigación, Comparison Methods of Edge Detection for USG Images, presentan:

Esta investigación es un trabajo comparativo de diferentes técnicas de detección de bordes aplicándolas en imágenes de ultrasonido, donde se proporciona información sobre el desarrollo del feto en el útero. El fin de obtener la forma del feto en la imagen de ultrasonido puede ser claramente identificado con el necesario proceso de análisis de una imagen a través de detectar los límites de los objetos, de modo que se pueda diferenciar de un objeto y otro en la imagen de ultrasonido. Para esto se señaló directamente el uso de 3 métodos: Sobel, Canny y Prewitt. Tras aplicar estos métodos, los mejores resultados se obtuvieron con el uso del método de sobel utilizando un valor de umbral de 0,03. Canny y Prewitt obtuvieron en mismo valor en cuanto a MSE (252.31, 251.32) y sobel obtuvo el valor más bajo (251.75, 259.58), aun así, la imagen resultante obtenida no fue capaz de brindar información completa.

Vikas et al., (2016), en su artículo, Edge Detection in Noisy Images Using Wavelet Transform, presentan:

Es su trabajo de investigación presentan detección de bordes basado en la técnica de wavelet, que es muy eficiente para imágenes con mucho ruido, esta técnica de detección de bordes es básica e importante en la medicina y en el procesamiento de imágenes. Usando la técnica de wavelet el ruido se puede reducir de manera efectiva sin pérdidas significativas en la calidad de la imagen. A diferencia con la técnica de Canny, que su primer paso es el suavizado del ruido con el filtro de gauss y posteriormente la detección de bordes. En wavelet combina ambas partes en uno solo, por lo tanto, la técnica basada en wavelet es computacionalmente más eficiente y además que utiliza una técnica de multirresolución. Tras la comparación de la técnica de canny y wavelet, wavelet dio mejores resultados tanto en imágenes ruidosa como en imágenes libres de ruido.

Yang et al., (2016), en su investigación Effective Clipart Image Vectorization through Direct Optimization of Bezigons, presentan:

Bezigons quiere decir trayectorias compuestas por curvas cerradas de bezier, estos han sido usados ampliamente para describir formas de resultados de vectorización de imágenes. Sin embargo, estas técnicas de vectorización infieren los bezigons como aproximar vectores intermedios, en consecuencia, los bezigons resultantes son en ocasiones imperfectas por los errores acumulados, ambigüedades de ajuste y falta de curva priors, en especial a las imágenes de baja resolución. Este trabajo realizó una optimización de estos mismos métodos, se mejoró la estructura de los bezigons. Tras haber puesto a prueba estos métodos se concluye que el método propuesto es mucho mejor tanto en comparaciones cualitativas y cuantitativas, en incluso a los métodos usados por un software comercial de uso común.

Hou et al., (2018), en su trabajo investigativo, Poisson Vector Graphics (PSV), presentan:

En este trabajo de graficos vectoriales de poisson, una extensión de las curvas de difusión populares, para la generación de imágenes sombreadas lisas, esta vez con 2 tipos de primitivas nuevas, Curvas de Poisson y Regiones de Poisson, con los que se puede generar efectos fotorrealistas. Los graficos vectoriales de poisson se diferencian de otros gráficos vectoriales por 3 características: separación explícita de colores y tonos, soporte nativo de clonación de fisuras e inserción de primitivas de modo que los usuarios pueden crear capas. Estos experimentos demuestras que a partir de los gráficos vectoriales de poisson se pueden producir gráficos vectoriales fotorrealistas a partir de cero. Esta herramienta (PVG) es muy útil y se distingue fácilmente de otras herramientas de dibujo mediante la separación de color y tono. Este estudio confirma que PVG es más intuitivo y fácil de usar que las curvas de difusión y sus variantes.

Donati et al., (2018), en su investigación, A complete hand-drawn sketch vectorization framework, presentan:

Este informe tiene como finalidad la creación de un nuevo método de vectorización, el área donde se planteó aplicarse en la industria de la moda. Donati destaca la difícil tarea que es vectorizar bocetos de dibujos a mano, pero es de suma importancia para la industria. El marco propuesto detalla un algoritmo de extracción de líneas basada en una aplicación multi-resolución y un algoritmo de adelgazamiento de traza garabatos para formar líneas limpias de un pixel, por último, la modificación del algoritmo de Schneider.

Este trabajo propone vectorizar bocetos de líneas, asumiendo que el algoritmo que proponen no está orientado al uso en vectorización de imágenes de manchas grandes y como objetivo final la creación de un algoritmo que devuelva una representación vectorizada compuesta por segmentos curvilíneos con el menor número posible de puntos de control. Luego de probar el algoritmo propuesto y los algoritmos mencionados al inicio se logró demostrar que el algoritmo propuesto reduce la cantidad de puntos de control frente al algoritmo de Schneider hasta en un 10% y 30% menos.

Yin et al., (2016), en su investigación, A vectorization method of building edge based on high resolution DSM data, presentan:

Este informe del método novel de vectorización, resuelve completamente el problema de representar imágenes de alta resolución sin presentar pérdidas de calidad visual. Esta solución comprende los siguientes componentes: 1) Mejora significativa de la resolución geométrica de detección de bordes y crestas. Se logra mediante una combinación de filtrado preliminar y análisis local del alto orden de la imagen. 2) Modelos geométricos, que capturan con precisión las líneas curvilíneas y patrones de imagen visual localizados. 3) Captura explícita de singularidades de los modelos, es decir las intersecciones, esquinas y cruces de alta curvatura. Se probó operaciones de edición de vectores como: edición geométrica de forma de curvas, edición de perfiles de color, incrustación de vectores, animación de vectores e imágenes fotográficas.

El método presentado conserva la calidad visual completa de las imágenes de alta resolución del mundo real, sus principales objetos como curvas equipadas con perfiles de color extendidos, captura explícita de singularidades de imagen y detección de bordes y crestas en alta resolución. Estos datos experimentales son siempre destacando la importancia de perfiles de color precisos y flexibles en patrones curvilíneos en la percepción visual humana.

De, (2019), en su trabajo de investigación, Vectorization of Architectural Floor Plans" , presentan:

Vectorizar una imagen implica dos problemas principales: Cómo extraer descriptores geométricos adecuados de la imagen rasterizada y como rasterizar una imagen vectorial para para su representación visual. En el presente trabajo se propone un enfoque novedoso de vectorización de imágenes utilizando curvas de difusión como primitivas geométricas. Nuestro enfoque extrae automáticamente curvas de difusión precisas de la imagen. Para esto se segmenta la imagen de entrada en un conjunto de superpixeles mediante un algoritmo de varias capas. Posteriormente se exploran

posiciones límite de estos superpixeles para ubicar los puntos de control para las curvas de difusión y la información de color se muestrea adecuadamente para generar nuestra representación de doble límite. Para representar los gráficos vectoriales se formula una difusión de recorrido aleatorio. Los experimentos realizados en diferentes fotografías demuestran que nuestro enfoque revela con éxito contenidos detallados en la imagen reconstruida, y que el proceso de renderizado se puede realizar en tiempo real en una CPU moderna.

Han et al., (2019), en su trabajo de investigación, An Improved Corner Detection Algorithm Based on Harris , presentan:

Este trabajo de investigación planteó una propuesta de resolver los problemas que presentaba el algoritmo de Harris Corner, ya que podía extraer esquinas demás, para ello se propone un algoritmo mejorado de Harris Corner. El método propuesto consta de sustituir la función gaussiana de suavizado por la función B-spline, luego preseleccionar los puntos de las esquinas para obtener esquinas candidatas, Por último, para mejorar la adaptabilidad del algoritmo se empleó un umbral auto adaptativo cuando se suprime el valor no máximo. Los resultados obtenidos demostraron que el algoritmo propuesto mejora la precisión y eficiencia en la detección de esquinas, además de un buen rendimiento. Se muestra una tabla comparativa de los resultados obtenidos, en la que Harris demuestra un 78% de precisión con 61 esquinas detectadas, de las cuales 48 realmente son esquinas.

1.3. Teorías Relacionadas al Tema

1.3.1. Proceso Productivo de Confección

El proceso de la confección de una prende tiene inicialmente el nombre de diseño, en esta etapa se planea los productos que se elaborarán, se determina de la misma manera los insumos, y se diseñan los molden que se utilizarán. Dependiendo las características del producto, los moldes y los insumos van a variar.

En los últimos años, se han incrementado este tipo de materias técnicas que imparten la oferta educativa del rubro mencionado en institutos y escuelas técnicas, de la misma manera se han incrementado el número de

diseñadores/as peruanos reconocidos a nivel nacional e internacional. Hace unas décadas atrás la falta de mano de obra en empresas dedicadas a este rubro, debido a que el bajo costo laboral y la abundancia de recursos naturales existían. Sin embargo, se ha ido perdiendo esta ventaja, actualmente la generación de marcas propias genera mayor valor agregado a los productos, posicionando nuestras exportaciones en mercados cada vez más competitivos. Posteriormente se lleva a cabo el proceso de corte-precostura, esta etapa consta de la realización de tendido de tela, corte y la respectiva inspección hasta que la tela quede apta para el siguiente proceso. Finalmente se realiza el proceso de costura que se basa en la unión de piezas para conseguir la forma del diseño original (Ministerio de Producción [PRODUCE], 2015, p.157).

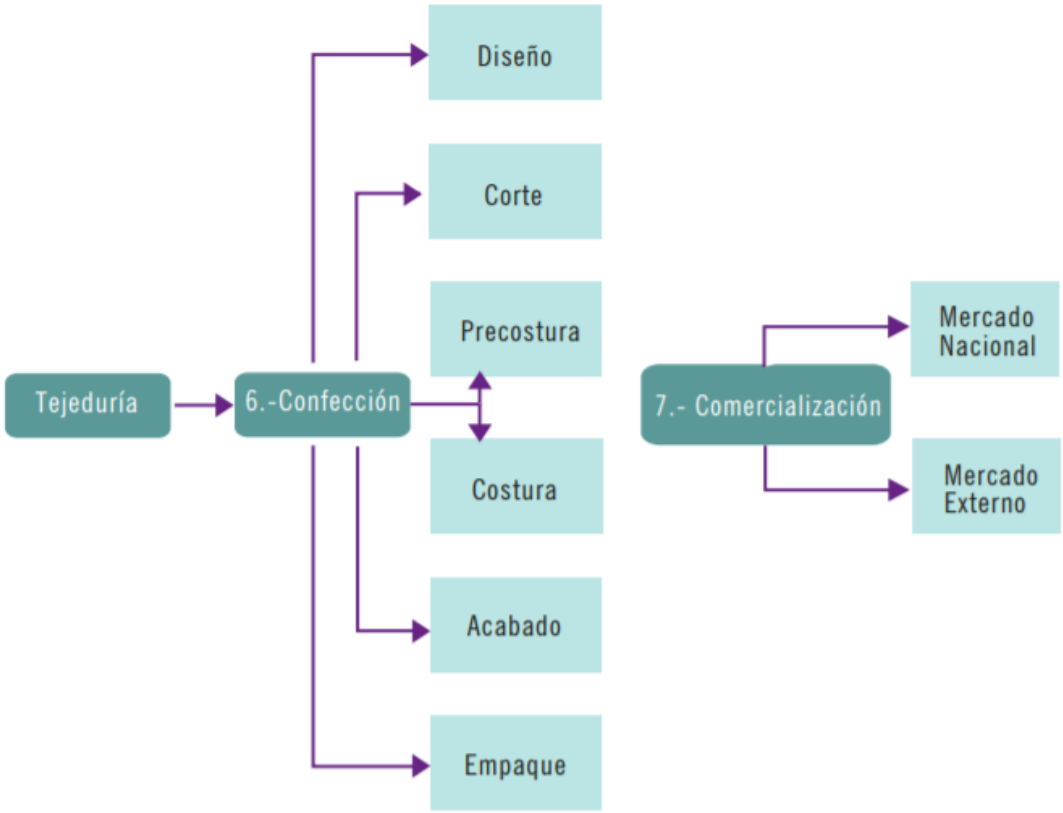


Figura 1. Proceso productivo de confecciones. Fuente: *(Ministerio de la Producción, 2017)*

1.3.1.1. Tendido

Este proceso consta de extender la tela en capas de manera uniforme a lo largo de la mesa de corte, posteriormente a esto se toman medidas aplican los moldes y se procede a cortar. Este proceso debe ser realizado tratando de no dañar la tela o aceptablemente dañar la menor cantidad posible, evitar estirones. Después de extender la tela en capas de unos 10 a 20 cm. aprox. deben evitarse que se formen dobladuras y/o aglomeraciones de tela en cualquier punto de la mesa para evitar cortes defectuosos. Además, la mesa de corte deberá ser adecuada para este tipo de procesos, debe ser lisa y pulida, libre de asperezas y abolladuras, para evitar formación de bordes defectuosos o roturas de la tela (Escobar, 2014, p. 142).

1.3.1.1.1. Tendido cara arriba

Este proceso inicia en un extremo de la mesa, allí se fija el rollo mientras avanza el carro tendiendo la tela, va fijando la tela hasta llegar al otro extremo donde se corta el ancho de la tela. Posteriormente el carro vuelve al punto de inicio sin dejar tela, y al llegar empieza a replicar el proceso antes mencionado. Este tipo de extendido es común para estampados de cuadros o rayas (Escobar, 2014, p. 142).

1.3.1.1.2. Tendido cara a cara (Zig – Zag)

Al igual que en el tipo de tendido anterior el inicio de tela se deposita en un extremo y la máquina se avanza hacia el otro extremo, dejando la tela en la mesa de manera simultánea. Al llegar al segundo extremo, éste se dobla y el carro retrocede extendiendo nuevamente la tela en reversa y así repitiendo el proceso. Se considera este el tipo de extendido más rápido (Escobar, 2014, p. 143).

1.3.1.1.3. Tendido cara a cara girando

El inicio de este tipo de tendido es similar al de extendido en una cara, en un extremo de la mesa, pero al llegar al otro extremo lo que hace el carro es volver al inicio sin cortar, esto genera una vuelta doble, de esta manera vuelve al principio y el proceso se repite otra vez (Escobar, 2014, p. 143).

1.3.1.1.4. Tendido escalonado

Este tipo de tendido es copiado de la extendida cara arriba, donde el ciclo de trabajo es similar o casi igual, menos la longitud de las capas. Esto es debido a que se intenta extender cantidades diferentes para tallas de tela distintos. Este tipo de tendido permite procesar varios pedidos en una sola marcada (Escobar, 2014, p. 144).

1.3.1.2. Trazado

En el ámbito de confección, un patrón es modelo que se realiza en un papel para luego ser replicada en el tejido. (Escobar, 2014, p. 75).

1.3.1.2.1. Patrones Domésticos

Es un patrón casero hecho en papel, marcado para recortar en distintas tallas. Los patrones domésticos usualmente son realizados en papel de seda (Escobar, 2014, p. 75).

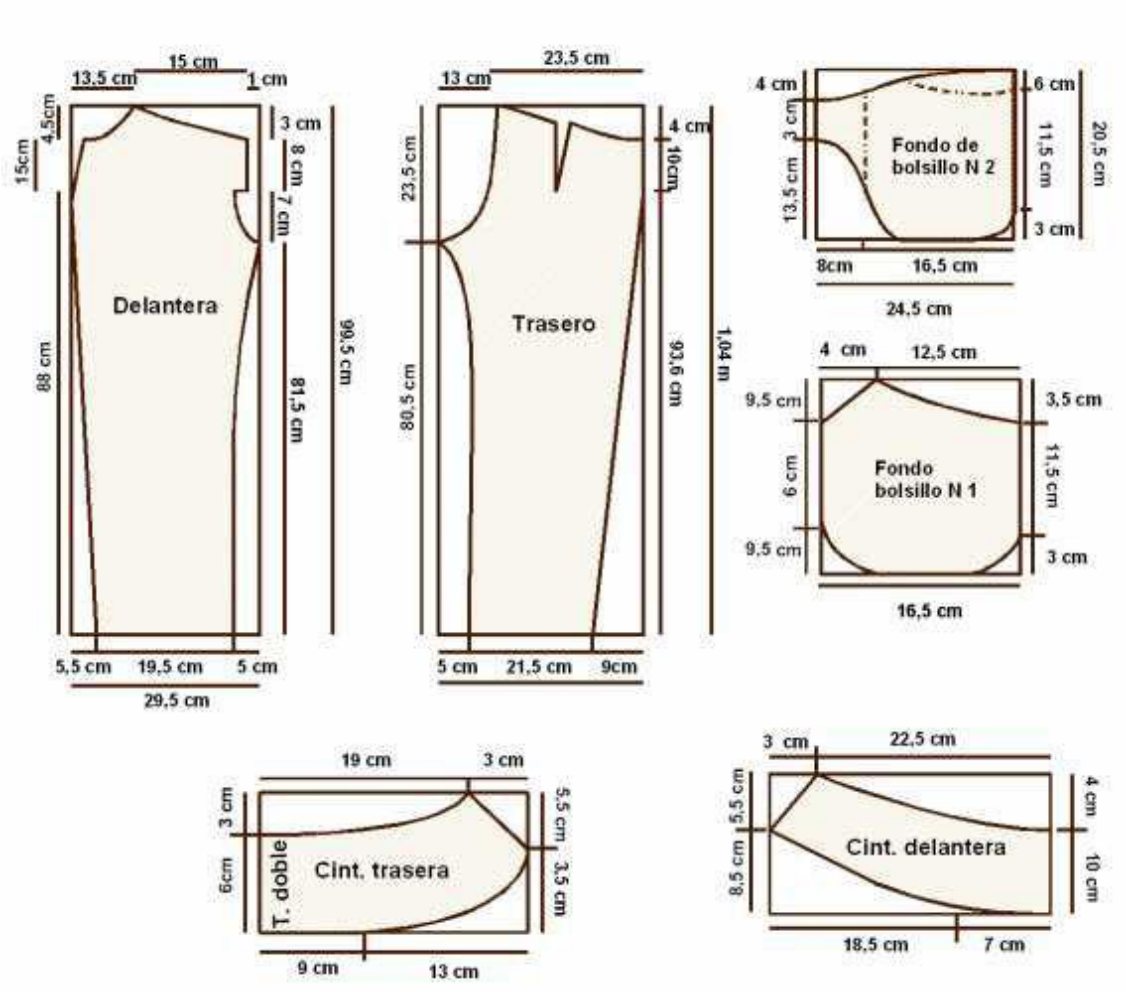


Figura 2. Patrones domésticos. Fuente: *(Escobar, 2014)*

1.3.1.2.2. Patrones Industriales

En el caso de los procesos para fábricas con producciones mayores el diseño se empieza con la representación aproximada de la idea del diseñador. Utilizando papel kraff para realizar el patrón y este pasa a ser revisado y/o remodelado. Tras ser aprobado este patrón se realiza a un tejido de prueba y pasa ser confeccionada una prenda. Tras realizar una valuación de mercado, se planifica la producción a escala, usualmente haciendo uso de programas CAD (Escobar, 2014, p. 75)

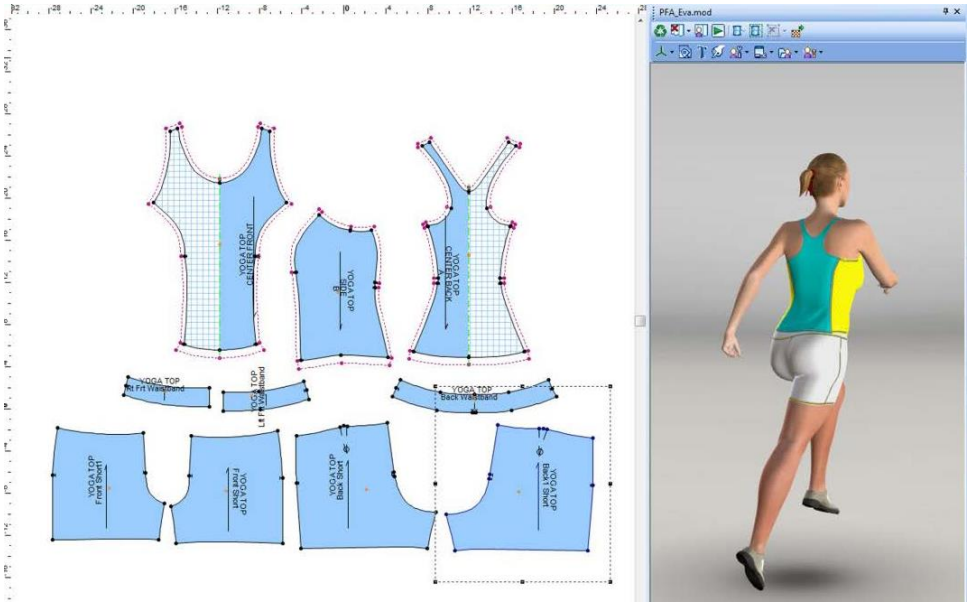


Figura 3. Interfaz gráfica de un sistema de patronaje industrial. Fuente*: (Escobar, 2014)*

1.3.1.3. Corte

Escobar, (2014) describe el procedimiento del corte automático:

1) Se sitúa la cortadora en un lado de la mesa de corte.
2) Se fija el inicio de la tela sobre la cortadora.
3) Se coloca un plástico sobre la tela a cortar.
4) Se realiza el corte haciendo uso de una cortadora manual, moviéndose dentro del área fijada.
5) Mientras avanza el corte, la parte ya cortada va quedando sobre una banda de salida. (p. 171)

1.3.2. Imagen

1.3.2.1. Tipos de Imagen

1.3.2.1.1. Imágenes Digitales Rasters o Mapa de Bits

Una imagen en mapa de bits es un archivo presentado por una rejilla en forma de cuadrilátero conformado por pixeles, o visualmente como puntos que representan un color que se muestran en la pantalla del ordenador, una hoja u otro medio de visualización.

Un mapa de bits está determinado por dimensiones de ancho y altura definidas por el número de pixeles según su orientación y el número de bits, que determina la cantidad de valores (colores) que cada pixel puede representar (UNED, 2008 p. 15)

1.3.2.1.2. Imágenes Vectoriales

Para UNED (2008), las imágenes vectoriales o gráficos vectoriales son conformados con primitivas geométricas los cuales pueden ser: puntos y líneas, que, de la misma manera, son gráficos basados en la construcción de ecuaciones matemáticas.

Actualmente todos los ordenadores realizan una conversión de gráficos vectoriales a gráficos bitmap para poderlos visualizar en pantalla. Se puede agrupar un conjunto de trazos formando objetos, y crear formas que permiten el uso de curvas de Bézier, degradados de color, etc.

En un gráfico vectorial, cada línea o vector posee características determinadas como: contorno, color y grosor. Una imagen vectorial es grabada como lista de puntos que describe cada uno de sus vectores resultantes, detallando sus propiedades.

Las imágenes vectoriales son independientes de la resolución, ya que no dependen de un mapa de píxeles. Por lo tanto, no presentan cambios en la variación de resolución. (UNED, 2008, pp. 25-26)

Ventajas

a) Los gráficos vectoriales ocupan menos espacio de almacenamiento que una imagen de mapa de bits.
b) Las imágenes vectoriales no alteran su calidad visual al ser editados.
c) Los parámetros vectoriales pueden ser guardados y modificados indefinidamente en el futuro.
d) Algunos formatos de imagen vectorial permiten animación de los vectores, esta se realiza de forma sencilla mediante la reubicación de sus puntos en los x, y o z.
e) Es posible controlar los colores de forma independiente de contornos

rellenos, texturas, degradados, transparencias, entre otros detalles.

f) Se puede controlar con la forma, orientación y orden de los elementos de la imagen.

Desventajas

a) En gráficos vectoriales no pueden representar imágenes complejas como fotografías, pero existen algunos formatos vectoriales que admiten composiciones mixtas mezclando trazos vectoriales y partes de imagen en mapas de bits.
b) La demora de presentación de la imagen vectorial depende de la cantidad de datos que procesar.
c) Las imágenes vectoriales siempre serán mostradas en mapas de bits, aún tratándose de imágenes vectoriales, los sistemas de muestra usan un sistema e mapas de bits, por lo que para ser visualizado o impreso se hace una conversión de la imagen vectorial a bitmap. (pp. 26 – 27)

1.3.3. Procesamiento de Imágenes

1.3.3.1. Segmentación

1.3.3.1.1. Detección de Bordes

Algunas técnicas de detección de bordes más conocidas se mencionan a continuación:

1.3.3.1.1.1. Canny

Para (Morse, 2000) el algoritmo de canny utiliza múltiples escalas para detectar mejor los bordes. Particularmente utiliza núcleos gaussianos de desenfoque de múltiples tamaños y selecciona la más estable. Su fórmula se describe como:

$$\frac{\partial}{\partial \bar{n}}(G * g) = 0$$

Dónde: g representa la imagen, n el borde normal y G representa el núcleo de desenfoque gaussiano.

El algoritmo funciona de la siguiente manera:

1) Seguir los pasos 2 al 6 para valores ascendentes de la desviación estándar σ.

2) Convolucionar una imagen g con un filtro gaussiano de escala σ.
3) Estimar las direcciones de bordes locales n usando la ecuación antes mencionada en cada pixel de la imagen.
4) Encontrar los bordes utilizando una ecuación de supresión no máxima (cruces por ceros).
5) Calcular la magnitud del borde (gradiente).
6) Aplicar humbralización en los bordes de la imagen con histéresis para eliminar espurias.
7) Agregar información sobre las aristas a escala múltiple usando síntesis de características. (pp. 60 – 61)

1.3.3.1.1.2. Sobel

Para (Morse, 2000), los núcleos de sobel se basan por las diferencias en el centro del filtro, y estos pixeles reciben mayor peso cuando se promedia:

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

$\partial/\partial x$

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

$\partial/\partial y$

Figura 4. Matriz de representación de núcleos de Sobel. Fuente: *(Morse, 2000)*

Ahora se dividirá entre 8 para encontrar componentes del vector gradiente. Esto quiere decir que los núcleos de sobel también pueden considerarse 3x3 aproximaciones a la primera derivada del núcleo gaussiano, que esto equivale a primero desenfocar la imagen y luego calcular la primera derivada. Esto se debe a que la convolución (y las derivadas) son conmutativas y asociativas:

$$\frac{\partial}{\partial x}(I * G) = I * \frac{\partial}{\partial x} G$$

Esa fórmula es el principio básico para recordar. (p. 55)

1.3.3.2. Vectorización

Algoritmos de Vectorización

1.3.3.2.1. Esqueleto – Guiado

La idea desarrollada en este algoritmo es utilizar el hecho de que el esqueleto de la imagen del personaje refleja el rastro de escritura, lo que ayuda al algoritmo de vectorización a extraer características importantes de trazo de contornos ruidosos. El desarrollo del proceso se describe así:

1) Primero se obtiene el esqueleto y el contorno suavizado mediante pre procesamiento.
2) Los puntos sobresalientes en el contorno se detectan aplicando nuestro algoritmo de eliminación de costos.
3) Los puntos sobresalientes en la región importante se seleccionan como puntos de esquina, mientras que los otros se clasifican como puntos de unión.
4) Finalmente, la curva ajustada es generado por un algoritmo dinámico de curvas.

Para medir la importancia de cada punto destacado, una función la define como la suma de curvatura y términos posicionales:

$$EK = EC_K - EP_K$$

Donde ECk es la curvatura y EPk denota la información posicional.

$$EC_k = \frac{|x'y'' - y'x''|}{(x'^2 + y'^2)^{1.5}}$$

Donde (x,y) son los puntos de la curva C de Pk.

$$EP_k = 1 - \frac{dis_k}{DisT_i}$$

Donde disK es la distancia de Pk a Si, y Si es siempre menor a DisTi porque Pk está en Ri. Si el grado del punto clave del esqueleto de Di, se encontrará

la energía máxima sobresaliente en los puntos de Ri. Luego estos puntos destacados se seleccionarán como puntos de esquina y otros puntos destacados se clasificarán como puntos de unión (Pan et al., 2014).

1.3.3.2.2. SPV

Graficos vectoriales de Poisson (PVG), es una extensión de las populares curvas de difusión (DC), para generar imágenes con sombras suaves. Armado con dos nuevos tipos de primitivas, llamadas curvas de Poisson y regiones de Poisson, PVG puede producir fácilmente efectos fotorrealistas tales como reflejos especulares, sombras centrales, translucidez y halos.

PVG se distingue de otros gráficos vectoriales basados en difusión por 3 características únicas: 1) separación explícita de colores y tonos, que sigue el principio básico de dibujo y facilita la edición; 2) soporte nativo de clonación sin interrupciones en el sentido de que las PC y las RP pueden encajar automáticamente en el fondo objetivo; y 3) primitivas de intersección permitidas (excepto la intersección DC-DC) para que los usuarios puedan crear capas. Mediante extensos experimentos y un estudio preliminar del usuario, demostramos que PVG es una herramienta de autoría simple pero poderosa que puede producir gráficos vectoriales fotorrealistas desde cero.
Pseudocodigo muestra el desarrollo del algoritmo de vectorización:
Un parche $P_s \subset D_s$ y un parche $P_t \subset D_t$ con límite idéntico.
Asegurarse: la mezcla $PVGu_t$ donde P_s se clona sin problemas a P_t.

Inicio algoritmo

If la región fuente P_s es opaca Then

Borrar la región de destino P_t, ∀p ∈ P_t, f (p) = 0

Else

For each DC γ ⊂ Pt Do

Convierta γ a una PC con Laplacian f | γ = (ΔUt) | γ;

Fin For

Fin If

Copie las PC y PR_s en P_s a P_t;

For Each DC γ? ⊂ Ps Do

Convertir γ' a una PC con Laplacian

$f|' = (\Delta u_s)|'_\gamma + (\Delta u_t)|'_\gamma$;

Copie la PC a P_t;

Fin For

Fin algoritmo

(Hou et al., 2018, pp. 1-11)

1.3.3.2.3. The method of square

El proceso que realiza el algoritmo consiste en determinar el punto de inicio. La imagen tomará un paso P con valor p=2, L vendrá ser el lado del cuadrado analizado alrededor del punto actual hasta que encuentre un grupo de pixeles con valor 1 y se defina el punto de equilibrio. Posterior a este se sigue el elemento lineal utilizando el método del cuadrado con centro de coordenadas (x0, y0) donde estas representan las coordenadas del punto actual.

1.3.3.2.4. Random walkers

El algoritmo se desarrolla en 4 etapas principales: 1) Generar Superpixeles de múltiples capas; 2) Fusionar múltiples capas; 3) Extraer curvas de difusión; 4) Rasterizar la imagen vectorial para su visualización.

Para extraer las curvas de difusión, primero se generará un mapa de superpixeles que consta de regiones. El elemento básico en una curva de difusión es una spline cúbica de bezier especificada por un conjunto de puntos de control de color y un conjunto de puntos de color de desenfoque, donde se define la suavidad de la transición de color. Este modelo genera curvas de doble limite, asignando un conjunto de puntos a una curva de difusión, ya que hay una curva adyacente para contener otro conjunto de colores. En el proceso de rasterización, el modelo de límite único introduce valores falsos debido a la dirección normal inexacta y al existir limites superpuestos se visualizará con una línea borrosa, el modelo de vectorización random walkers al utilizar 2 limites en cada curva de difusión evita los efectos de sangrado de color en los extremos de la curva (Dai et al., 2013, pp. 1 - 4).

1.3.4. Programación

1.3.4.1. Lenguajes de Programación

1.3.4.1.1. Java

Al igual que todo lenguaje de programación, Java tiene su propia estructura y sintaxis, java mantiene el paradigma de programación orientada a objetos.

Estructuralmente, Java se inicia con paquetes, tomando un derivación de clases, métodos, variables, constantes, etc. (IBM Developer, 2019).

Algunas librerías y complementos para trabajar con procesamiento de imágenes en java se mencionan a continuación:

1.3.4.1.1.1. ImageJ

ImageJ es un programa basado en la funcionalidad de procesamiento de imágenes digitales en Java. Permite la visualización, edición, análisis, procesamiento, guardado e impresión de imágenes de 8, 16 y 32 bits. Es posible la lectura de formatos de imagen como TIFF, GIF, JPEG, BMP, DICOM, FITS y "en bruto". (ImageJ, 2019).

1.3.4.1.1.2. Fiji

Fiji es una distribución del software de código abierto populares ImageJ se centró en el análisis de imágenes biológica. Fiji utiliza prácticas de ingeniería de software modernos para combinar bibliotecas de software de gran alcance con una amplia gama de lenguajes de scripting para permitir el prototipado rápido de algoritmos de tratamiento de imagen. Fiji facilita la transformación de nuevos algoritmos en ImageJ plugins que se pueden compartir con los usuarios finales a través de un sistema de actualización integrado. Proponemos Fiji como una plataforma para la colaboración productiva entre la informática y de las comunidades de investigación de biología (Schindelin et al., 2012).

1.3.4.2. IDE's de Programación

1.3.4.2.1. Eclipse

El IDE Eclipse es famoso por nuestro entorno de desarrollo integrado Java (IDE), pero tenemos una serie de entornos de desarrollo bastante

fresco, incluyendo nuestra C / C ++ IDE, IDE para JavaScript / Letra de imprenta, PHP IDE, y mucho más.

Se pueden combinar fácilmente múltiples idiomas de apoyo y otras características en cualquiera de nuestros paquetes predeterminados, y el eclipse del mercado permite la personalización virtualmente ilimitada y la extensión (Eclipse Foundation, 2019).

1.4. Formulación del Problema

¿Cuál es el mejor algoritmo de detección de bordes y el mejor método de vectorización de imágenes de moldes textiles?

1.5. Justificación e Importancia de Estudio

Este trabajo de investigación se justifica porque se enfrenta un problema claro en la industria de confecciones, y se realiza esta investigación mediante medios tecnológicos, aplicando conocimientos propios y tomando conocimientos externos que prueban una eficacia en su desarrollo, de esta manera se espera culminar con resultados alentadores y favorables a la investigación.

Este trabajo es importante porque se desea minimizar un problema evidente en el sector textil, y además de ello esta investigación puede servir de base para futuras investigaciones, propuestas de mejoras o continuación del mismo.

1.6. Hipótesis

1. El método de Canny-Deriche es la mejor para detectar bordes.
2. La técnica de Harris Corner la mejor para detectar esquinas vectorizables.

1.7. Objetivos

1.7.1. Objetivo General

Comparar algoritmos de detección de bordes y técnicas de vectorización de imágenes para optimizar la producción de la industria de confección.

1.7.2. Objetivos Específicos

A. Construir una base de datos de imágenes digitales de moldes textiles.
B. Seleccionar algoritmos de detección de bordes y vectorización.

C. Implementar los algoritmos de detección de bordes y vectorización.
D. Evaluar los resultados de la detección de bordes e imágenes vectorizadas.

II. MATERIAL Y MÉTODO

2.1. Tipo y Diseño de Investigación

2.1.1. Tipo de Investigación

Este trabajo investigativo es una investigación tecnológica aplicada de tipo cuantitativa porque analiza múltiples elementos cuantificables y medibles. Toda la información fue obtenida en base a muestras de una población seleccionada, y los resultados son extrapolables a la población determinando un nivel de error y confianza.

2.1.2. Diseño de Investigación

El diseño utilizado es Cuasi-Experimental con post prueba y aplicándose a un solo grupo. Se analizó imágenes de patrones textiles manipulando variables dependientes en función de las variables independientes para observar sus efectos.

2.2. Población y Muestra

2.2.1. Población

Se tuvo una población de 12 algoritmos y métodos de detección de bordes y 7 algoritmos de vectorización de imágenes.

2.2.2. Muestra

Se seleccionó una muestra de 3 algoritmos de detección de bordes y 3 técnicas de vectorización de imágenes.

2.3. Variables, Operacionalización

2.3.1. Variable Independiente

Método de detección de bordes, Algoritmo de vectorización.

2.3.2. Variable Dependiente

Detección de bordes, vectorización.

2.3.3. Operacionalización de Variables

Tabla 1.

Operacionalización de variables.

VARIABLES		INDICADORES	FÓRMULAS	Métodos de Recolección de Datos
V. D	Técnicas de detección de Bordes	MSE	$MSE = \frac{1}{MN}\sum_{i=0}^{M-1}\sum_{j=0}^{N-1}\|\|I(i,j) - K(I,J)\|\|^2$	Observación
		PSNR	$PSNR = 10.log_{10}\left(\frac{MAX_1^2}{MSE}\right) = 20.log_{10}\left(\frac{MAX_1}{\sqrt{MSE}}\right), MAX_I = 2^B - 1$	
	Algoritmos de vectorización de Imágenes	Exactitud	$E = \frac{VP + VN}{VP + FP + VN + FN}$	
		Precisión	$P = \frac{VP}{VP + FP}$	
		Exhaustividad	$R = \frac{VP + VN}{VP + FN}$	
V. I.	Detección de bordes / Vectorización de imágenes	Consumo de memoria RAM	$RAM = \frac{Total\ de\ memoria}{\left(\frac{\sum Uso\ de\ memoria\ por\ proceso}{\sum N°\ de\ procesos\ ejecutados}\right)}$	Observación
		Tiempo de respuesta	$TR = TI - TF,$ $TI = hora\ de\ inicio\ de\ ejecución$ $TF = hora\ de\ fin\ de\ ejecución$	

Nota: (Elaboración Propia)

2.4. Técnicas e Instrumentos de Recolección de Datos, validez y Confiabilidad

2.4.1. Técnicas

A. Observación

Esta técnica fue la utilizada en el desarrollo del proyecto, pues se basa en la observación directa de los resultados mostrados por los algoritmos al realizar una tarea.

2.4.2. Instrumento de recolección de datos, validez y confiabilidad

A. Ficha de observación

Este instrumento de recolección de datos nos permitió llevar un control de los detalles obtenidos en el desarrollo de cada tarea. Estos instrumentos se muestran en los anexos 1 y 2.

2.5. Procesamiento y Análisis de datos

2.5.1. MSE

Error cuadrático medio, esta fórmula mide la incertidumbre de nuestro pronóstico, y nos permite calcular la diferencia entre valores reales y una estimación. También es conocida en estadística como varianza.

$$MSE = \frac{1}{MN}\sum_{i=0}^{M-1}\sum_{j=0}^{N-1}||I(i,j) - K(I,J)||^2$$

Donde:

Tabla 2.

Descripción de las variables de la fórmula MSE.

Variable	Descripción
M	Tamaño de la imagen en la dimensión I.
N	Tamaño de la imagen en la dimensión J.
i	Posición del pixel en la dimensión I.
j	Posición del pixel en la dimensión J.
I	Imagen de entrada.
K	Imagen de salida.

Nota: (Elaboración Propia)

2.5.2. PSNR

Proporción máxima de señal a ruido, se utiliza esta fórmula para calcular la relación entre una señal y el ruido presente que afecta su clara representación. Es una fórmula dependiente del MSE.

$$PSNR = 10.log_{10}\left(\frac{MAX_1^2}{MSE}\right)$$

Donde:

Tabla 3.

Descripción de las variables de la fórmula PSNR.

Variable	Descripción
MSE	Error cuadrático medio.
MAX	Máximo valor de un pixel en la imagen.

Nota: (Elaboración Propia)

2.5.3. Exactitud

La exactitud demuestra el acercamiento de una predicción a su valor verdadero. Su fórmula está definida según la matriz de confusión.

$$Exactitud = \left(\frac{VP + VN}{VP + VN + FP + FN}\right)$$

Donde:

Tabla 4.

Descripción de las variables de la fórmula Exactitud.

Variable	Descripción
VP	Verdaderos Positivos.
VN	Verdaderos Negativos.
FP	Falsos Positivos.
FN	Falsos Negativos.

Nota: (Elaboración Propia)

2.5.4. Precisión

La precisión demuestra el porcentaje de los aciertos detectados.

$$Precisión = \left(\frac{VP}{VP + FP}\right)$$

Donde:

Tabla 5.

Descripción de las variables de la fórmula Precisión.

Variable	Descripción
VP	Verdaderos Positivos.
FP	Falsos Positivos.

Nota: (Elaboración Propia)

2.5.5. Exhaustividad

La exhaustividad demuestra la proporción de los casos positivos que se detectaron correctamente.

$$Exhaustividad = \left(\frac{VP}{VP + FN}\right)$$

Donde:

Tabla 6.

Descripción de las variables de la fórmula Exactitud.

Variable	Descripción
VP	Verdaderos Positivos.
FN	Falsos Negativos.

Nota: (Elaboración Propia)

2.5.6. Consumo de memoria RAM

Representa el promedio de consumo de memoria que se realizó en la ejecución de un proceso.

$$RAM = \frac{Total\ de\ memoria}{\left(\frac{\sum Uso\ de\ memoria\ por\ proceso}{\sum N°\ de\ procesos\ ejecutados}\right)}$$

2.5.7. Tiempo de respuesta

Calcula el tiempo de ejecución de un proceso.

$$TR = Hora\ de\ inicio - Hora\ de\ finalización$$

2.6. Criterios éticos

2.6.1. El conocimiento informado

El principio ético es saber que el tesista está informado teniendo conocimiento de sus derechos y responsabilidades en el transcurso de su investigación.

2.6.2. Manejo de riesgos

Esta investigación busca garantizar la calidad y responsabilidad para los informantes, posteriormente está relacionado al manejo de los datos que se han proporcionado.

2.7. Criterios de rigor científico

2.7.1. Consistencia

La propuesta planteada contiene bases consistentes, poniéndose a prueba con otros métodos establecidos y aprobados por la comunidad científica.

2.7.2. Validez

Los datos utilizados y resultados obtenidos serán evaluados por personas expertas en el tema para corroborar la veracidad de la información expuesta.

2.7.3. Fiabilidad

La investigación se encamina en el cumplimiento de las expectativas de los investigadores, respetando y realizando los objetivos planteados.

ser transferida a investigadores que se enfoquen en contextos similares.

2.7.4. Neutralidad

La investigación garantiza la validez de los datos utilizados y la confiabilidad de los resultados obtenidos, no pudiendo ser alterados por intereses o perspectivas de los investigadores.

III. RESULTADOS

3.1. Resultados de Tablas y Figuras

En esta investigación se evaluaron algoritmos de dos funciones, 3 algoritmos y técnicas de detección de esquinas como Canny-Deriche, Sobel y la técnica propuesta de Erosión, además otros 3 algoritmos de detección de esquinas como Harris Corner, la técnica propuesta de las pendientes y la técnica de recorrido del borde Pixel a Pixel. Se aclara que se realizaron pruebas en 42 imágenes, se tabularon los resultados y se muestra el valor promedio de todos los resultados obtenidos.

Los algoritmos de detección de bordes se evaluaron por el valor de error cuadrático medio, proporción máxima señal – ruido, consumo de memoria ram y tiempo de respuesta por proceso.

El error cuadrático medio (MSE) estadísticamente denota la diferencia entre un estimador y lo que se estima, en el procesamiento de imágenes denota que tanto se ha degradado la imagen original, por lo que un valor más alto demuestra mayor pérdida de información. En la figura 5, el método Canny-Deriche muestra el valor más bajo con un valor MSE de 373, frente al método de Sobel y la técnica de Erosión con valores MSE de 1536 para ambos, esto significa que el método Canny-Deriche pierde menos información tras haber aplicado su filtro de detección de bordes. La diferencia de Canny-Deriche frente a los otros dos métodos se explica porque Canny devuelve una imagen de salida en 32bits, por lo que la diferencia entre la imagen original y la imagen de salida en 32 bits, pues tomando más niveles de gris tendría menor diferencia entre los valores de sus pixeles, mientras que os otros 2 métodos muestran imágenes de salida en pixeles blanco y negro.

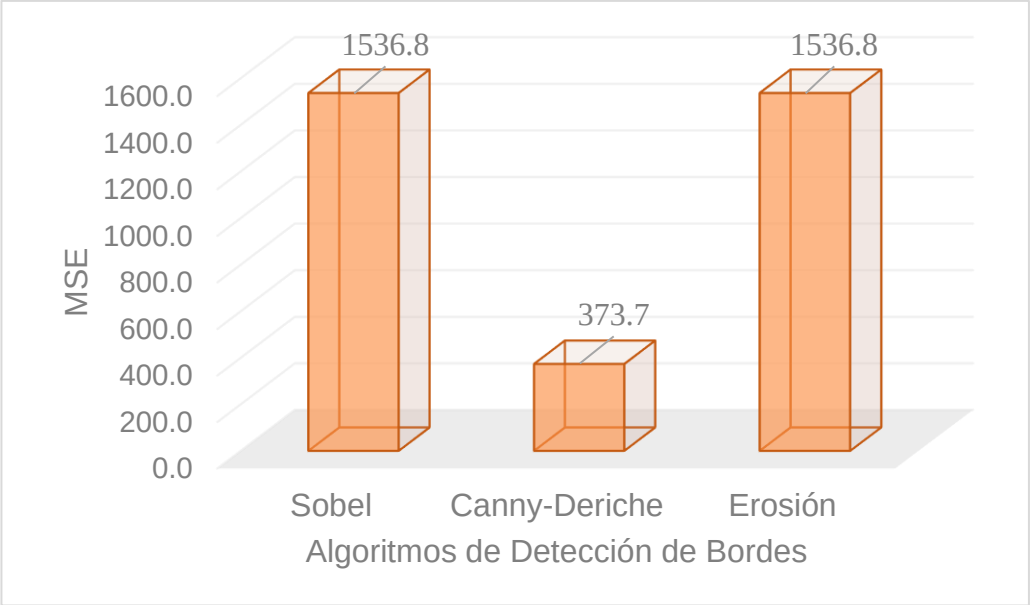


Figura 5. Gráfico comparativo de error cuadrático medio (MSE) de las técnicas de detección de bordes. Fuente: *(Elaboración Propia)*

El PSNR es dependiente de MSE y calcula la relación entre la mayor energía posible de una señal y el ruido que afecta su representación fidedigna, por lo que un valor más alto en PSNR representa una buena salida de la imagen en relación a la señal – ruido. En la figura 6, tras aplicar los algoritmos de detección de bordes y haber calculado el PSNR promedio de todas las muestras evaluadas, se determinó que la técnica de Canny-Deriche tiene el valor más alto con 22.9 de PSNR, frente a las técnicas de Sobel y Erosión con 16.8 de PSNR. El algoritmo Canny-Deriche se sobrepone frente a los otros métodos porque este algoritmo parte con un pre-procesamiento de la imagen aplicando un filtro gaussiano, lo que hace minimizar la presencia de ruido, además, muestra una imagen de salida en 32 bits, donde cada pixel toma entre 256 tonalidades por cada escala de color, frente a los otros métodos de muestran una transformación a blanco y negro que provoca un cambio brusco entre el valor de pixel original y el pixel de salida.

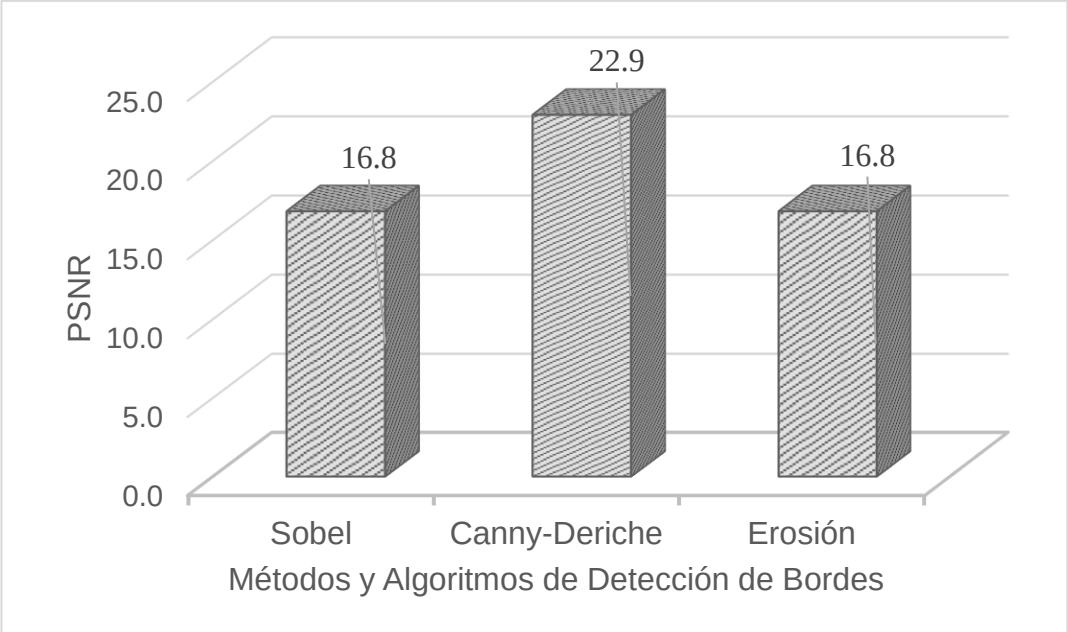


Figura 6. Comparación de proporción máxima señal - ruido (PSNR) de las técnicas de detección de bordes. Fuente: *(Elaboración Propia)*

La figura 7 muestra la comparativa de consumo de recursos en cuanto a memoria ram de los algoritmos seleccionados, donde el algoritmo de Canny-Deriche presenta el valor más alto de consumo de recursos con un promedio de 19.989 MB de memoria consumida por proceso, frente a el método de Erosión con un consumo promedio de 2.8 MB de memoria ram por proceso y con el valor más bajo el algoritmo de Sobel con un consumo promedio de 1.4 MB de memoria ram por proceso.

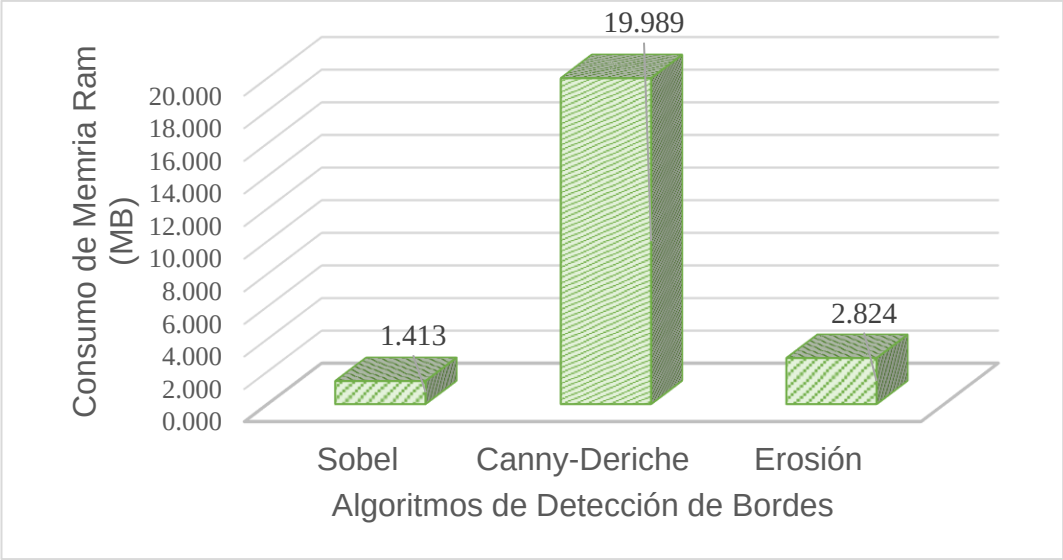


Figura 7. Gráfico comparativo de consumo de memoria ram en las técnicas de detección de bordes. Fuente: *(Elaboración Propia)*

Los tiempos de respuesta demuestran la rapidez de un algoritmo al realizar la detección de los bordes en toda la imagen. Los tiempos de respuesta obtenidos tras la aplicación de los algoritmos seleccionados fueron 6.1 ms en el algoritmo de Sobel, 7 ms tras aplicar la técnica de Erosión y 104.3 ms tras aplicar el algoritmo Canny-Deriche. Dando los mejores resultados el algoritmo de Sobel.

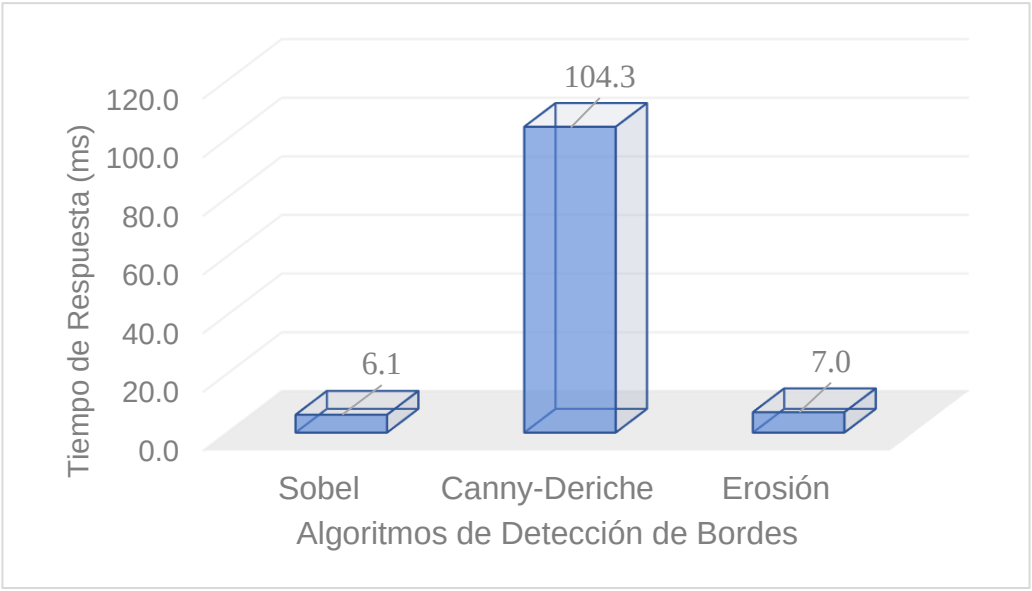


Figura 8. Comparación de los tiempos de respuesta de las técnicas de detección de bordes. Fuente: *(Elaboración Propia)*

El proceso de vectorización se midió a través de algoritmos de detección de esquinas vectorizables, los cuales son tomados como puntos de referencia y posteriormente unir estos puntos con vectores, dando como resultado la imagen vectorizada. Estos algoritmos se midieron haciendo uso de indicadores como exactitud, precisión, exhaustividad, consumo de memoria y tiempo de respuesta.

La exactitud muestra la proporción de detecciones que se realizaron correctamente del total de detecciones realizadas. La figura 9 muestra a la técnica de las pendientes con el promedio de 10% de exactitud, seguido del algoritmo Harris Corner con un promedio de 11% de exactitud y la técnica de detección de esquinas pixel a pixel con el valor más alto de 79% de exactitud.

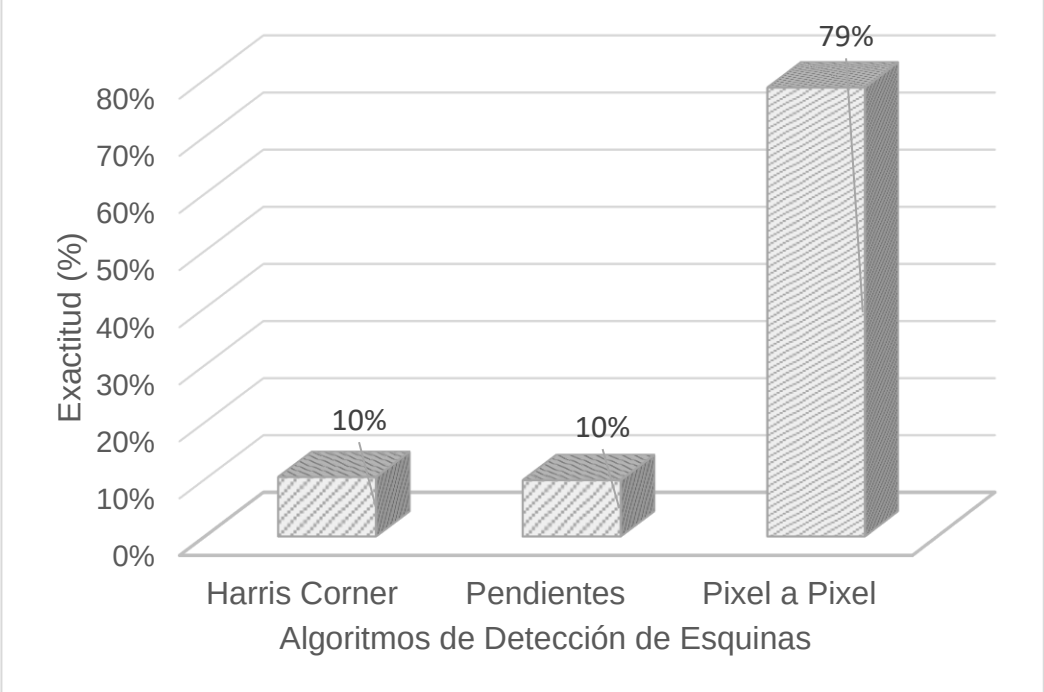


Figura 9. Comparación de exactitud de las técnicas seleccionadas de detección de esquinas vectorizables. Fuente: *(Elaboración Propia)*

La precisión denota la proporción de las detecciones positivas de esquinas que fueron correctas. El cuadro estadístico de la figura 10 muestra al algoritmo Harris Corner con el porcentaje de precisión más bajo de 11%, seguido de la técnica de las Pendientes con el 20% de precisión y con el puntaje más alto la técnica Pixel a Pixel con el 90% de precisión.

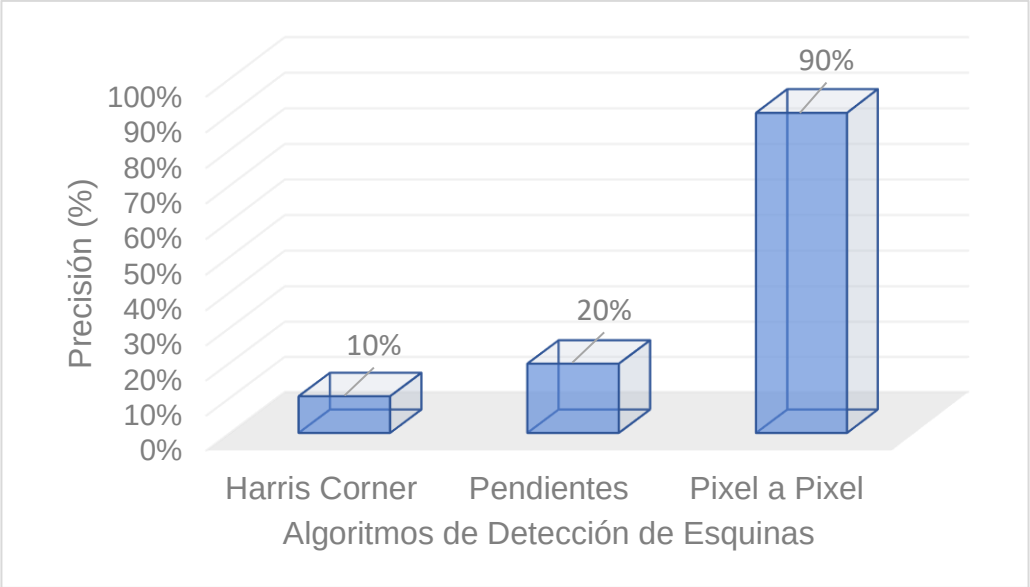


Figura 10. Comparación de precisión de las técnicas seleccionadas de detección de esquinas vectorizables. Fuente: *(Elaboración Propia)*

La fórmula de exhaustividad calcula el porcentaje de esquinas reales que han sido detectadas. En las pruebas realizadas el algoritmo Harris Corner detectó la mayoría de esquinas existentes, por lo que obtendría un porcentaje de exhaustividad del 94%, seguido de la técnica Pixel a Pixel con 83% de exhaustividad y por último la técnica de pendientes con el 36% de exhaustividad.

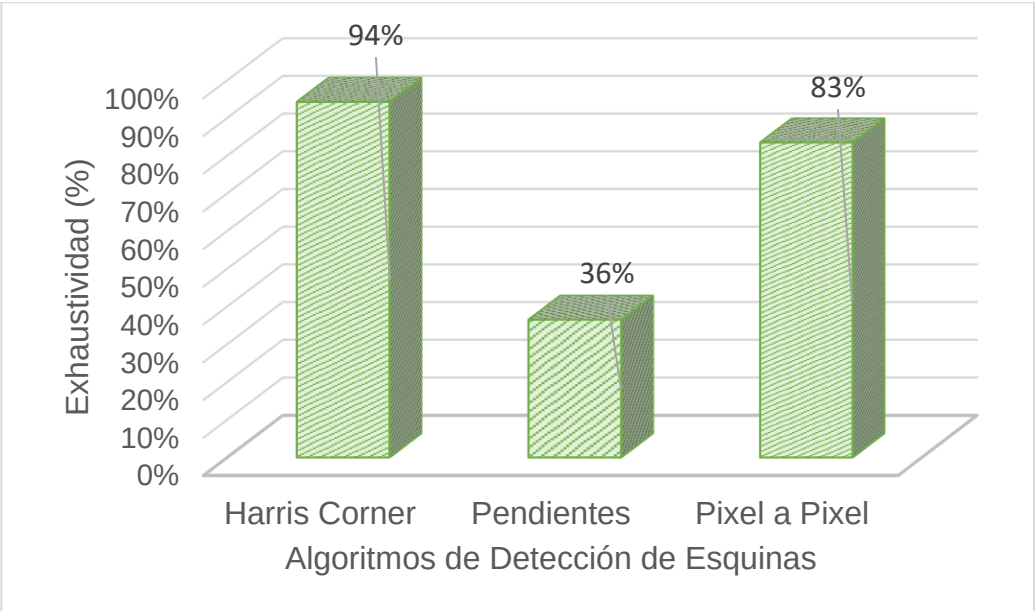


Figura 11. Comparación de exhaustividad de las técnicas seleccionadas de detección de esquinas vectorizables. Fuente: *(Elaboración Propia)*

La figura 12 muestra la comparativa de consumo de recursos en cuanto a memoria ram de los algoritmos evaluados, el algoritmo Harris Corner registró el consumo de memoria más alto con un promedio de 28.3 MB de memoria ram consumidos por proceso, seguido de la técnica de las pendientes con un registro promedio de 1.6 MB de memoria ram por proceso y por último la técnica Pixel a Pixel con el promedio más bajo con 0.9 MB de memoria ram consumidos por proceso.

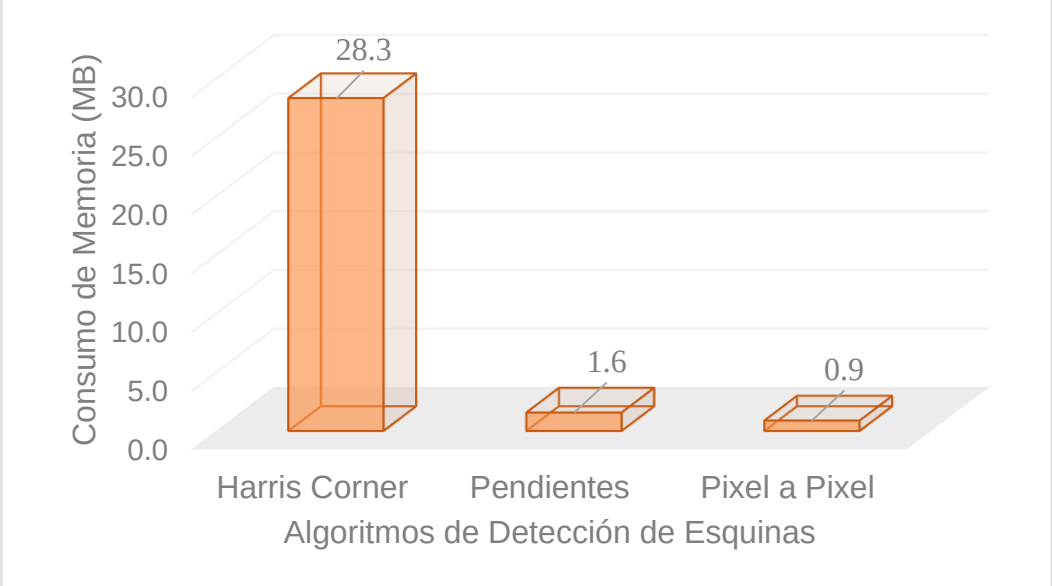


Figura 12. Comparación de consumo de memoria ram de las técnicas seleccionadas de detección de esquinas vectorizables. Fuente: *(Elaboración Propia)*

Los tiempos de respuesta demuestran la rapidez de un algoritmo al realizar una tarea, es esta comparativa el tiempo que demora en extraer esquinas vectorizables en una imagen de bordes. En la figura 13 muestra la comparativa de resultados obtenidos en las pruebas realizadas, el algoritmo Harris Corner con el valor más alto, registrando una demora de 68.52 ms de tiempo de respuesta, seguido de la técnica de las pendientes registrando una demora promedio de 1.11 ms de tiempo de respuesta y terminando con la técnica Pixel

a Pixel con los mejores resultados, registrando un tiempo de respuesta de 0.57 ms.

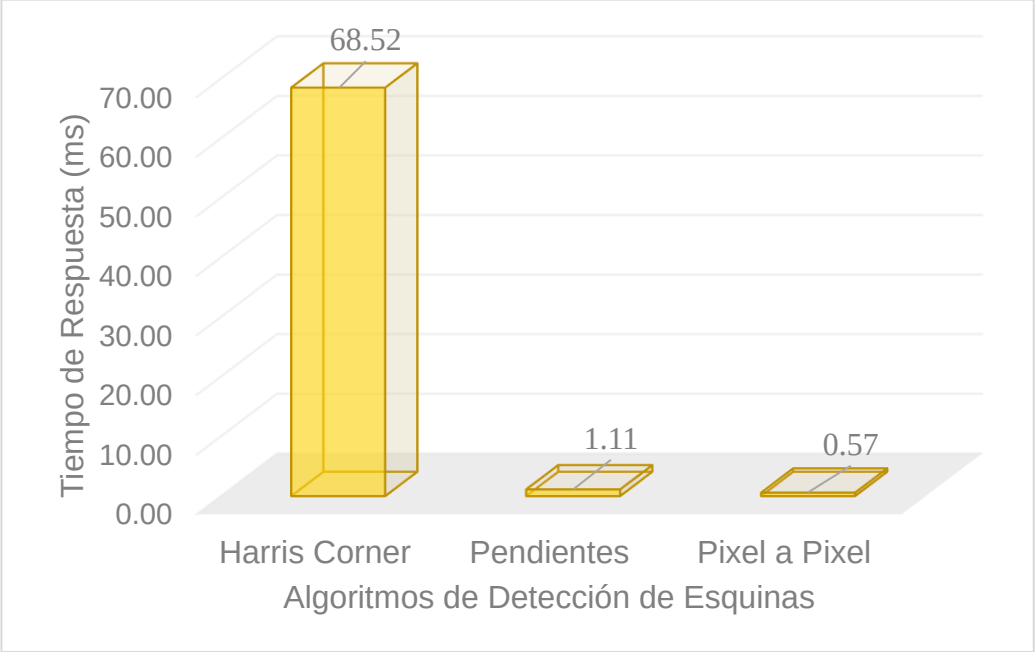


Figura 13. Comparación de tiempo de respuesta de las técnicas seleccionadas de detección de esquinas vectorizables. Fuente: *(Elaboración Propia)*

3.2. Discusión de Resultados

Tras haber aplicado los algoritmos de detección de bordes seleccionados en el proceso, cabe mencionar que Canny-Deriche es muy eficiente, sería el algoritmo con mejores resultados estadísticos de los tres algoritmos en cuanto a MSE y PSNR, pero a su vez el algoritmo que más recursos consume, resultados similares fueron obtenidos por Khairudin en su trabajo de investigación, donde señala que Canny tiene un alto consumo de recursos y un mayor tiempo de respuesta, aunque demuestra obtener valores menores en MSE y más elevados en cuanto a PSNR (Khairudin & Irmawati, 2017).

El método propuesto de detección de bordes a partir de la erosión, resalta la diferencia de costo computacional frente al algoritmo Canny-Deriche, siendo el método propuesto más rápido y menos costoso en consumo de memoria, además se podría optimizar hasta en 20% de costo computacional.

Analizando exactamente el caso propuesto, la diferencia visual entre los 3 métodos seleccionados es poco notorio y no afecta la diferencia estadística en el caso propuesto, por lo que se resalta además que en tiempos de respuesta el algoritmo de Sobel es mucho más rápido, frente a Canny-Deriche por 98.2 ms de diferencia y frente a la técnica de erosión por 0.9 ms de diferencia.

Con respecto a los algoritmos de vectorización, cabe mencionar que el proceso de vectorización tiene mayor rigor sobre la tarea de detección de esquinas que vienen a ser los puntos de referencia en la vectorización. Para realizar esta prueba se seleccionó 3 técnicas, donde el método propuesto Pixel a Pixel mantiene valores promedios de exactitud, precisión y exhaustividad de 79%, 90% y 83% respectivamente, mostrando mejores resultados frente al método de Harris y la técnica de las pendientes.

Según los resultados de Shi, mostrados en los trabajos previos resalta un análisis comparativo del algoritmo de Harris y un algoritmo mejorado de Harris, donde especifica que el método tradicional de Harris extrae un número elevado de esquinas detectadas, de las que un gran número de ellas son detecciones erróneas (Han et al., 2019). Los resultados obtenidos por Shi guardan relación además con el trabajo investigativo de Guiming, quien aporta que el método tradicional de Harris extrae demasiadas esquinas falsas, haciendo una comparación con un método propuesto basado también en Harris, logrando minimizar la selección de esquinas detectadas. (Guiming & Jidong, 2018).

Los resultados obtenidos muestran eficiencia del método propuesto, además de sobreponerse al método de Harris con una diferencia de 27.4 MB de memoria consumida por proceso y 67.95 ms de diferencia por proceso en tiempo de respuesta, lo que significa que para nuestra problemática y metodología sea una buena propuesta, pero considerar que los métodos utilizados pueden ser más o menos eficientes en condiciones y metodologías distintas.

3.3. Aporte Práctico

En esta investigación, se formuló un proceso para la construcción de la base de datos de imágenes de moldes textiles o patrones textiles, este proceso estuvo determinado de la siguiente manera:

Figura 14. Proceso de adquisición de imágenes y creación de la base de datos. Fuente: *(Elaboración Propia)*

La estructura de adquisición de imágenes se elaboró a partir de soportes de madera (2cm de ancho y 2cm de profundidad), se le dio forma cúbica, con una altura de 65cm y un base de 70cm2 (70x70). Los 4 soportes horizontales posicionados en la parte inferior de la estructura quedaron ubicados a alturas de 35cm y 40cm respectivamente, con la finalidad de no interferir con el área de visión de la cámara. La cámara quedó posicionada en la parte superior central del cubo.

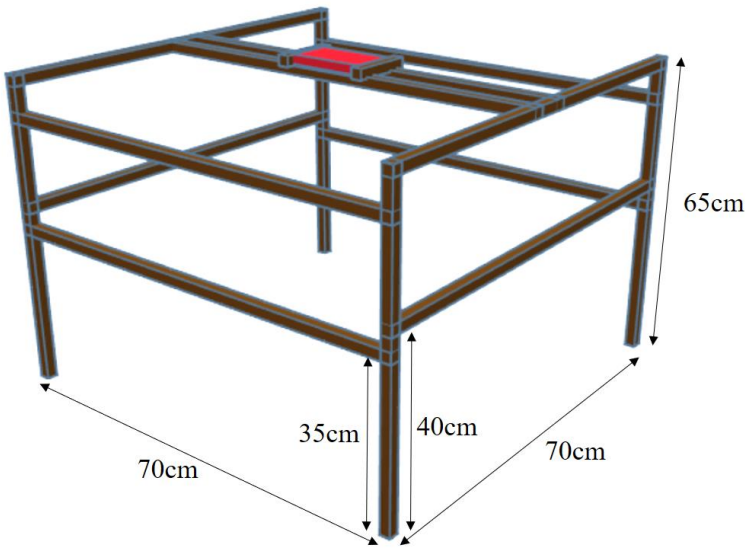


Figura 15. Representación de la estructura para adquirir imágenes. Fuente: *(Elaboración Propia)*

Las características del dispositivo de adquisición de imágenes digitales se pueden ver en la tabla 7:

Tabla 7.

Características del dispositivo de adquisición de imágenes digitales.

Característica	**Detalle**
Resolución	15.93 MP
Tamaño del sensor	4.74mm x 3.56mm
Apertura	f/2.0
Factor de Multiplicación	7.29
Distancia Focal	3.57mm
Tamaño de Pixel	1.029µm
Cantidad de Pixeles	3456x4608 (15925248)

Nota: (Elaboración Propia)

La configuración del dispositivo de adquisición de imágenes digitales se puede ver en la tabla 8:

Tabla 8.

Configuración del dispositivo de adquisición de imágenes digitales.

Característica	**Detalle**
Resolución	15.93 MP
Relación de aspecto	(4:3)
Apertura	f/2.0
Valor ISO	100
Distancia focal	3.57mm
Tiempo de exposición	1/199s
Dimensión de Fotografía	3456p x 4608p
Flash	Desactivado
Balance de Blancos	0 (AWB)
Exposición (EV)	-1

Nota: (Elaboración Propia)

La adquisición de imágenes se realizó en un ambiente controlado, y el protocolo que se siguió fue: una altura de la cámara al objeto de 65cm,

iluminación controlada a través de luz led blanca a 15cm de altura del filtro superior, dicho filtro fue a base de tela blanca, además se utilizó una cubierta de la misma tela alrededor de la estructura. El objeto a capturar quedó superpuesto sobre una superficie color negro, mientras el objeto fue de color claro para una mejor discriminación de colores por parte de los algoritmos.

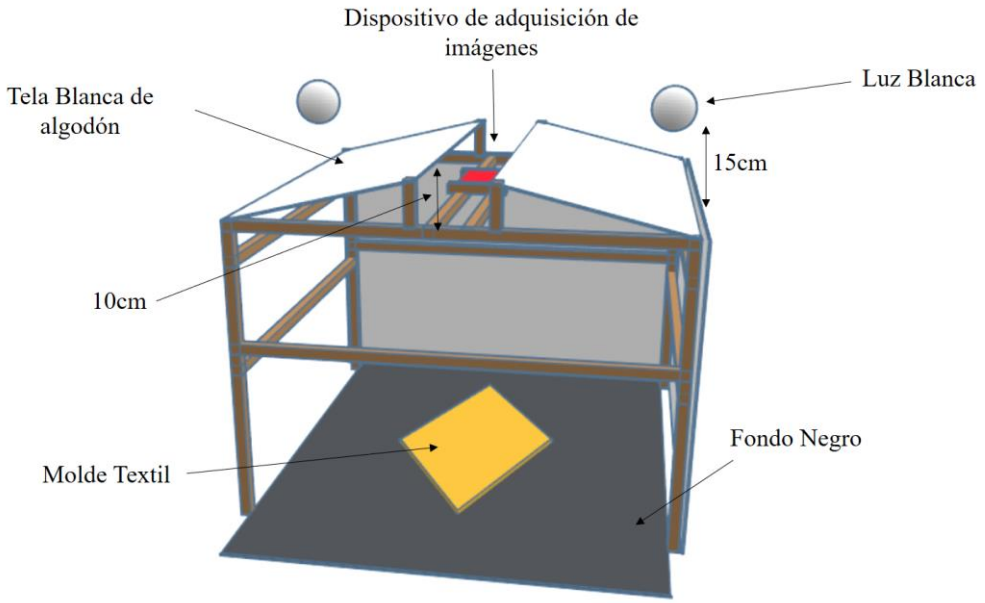
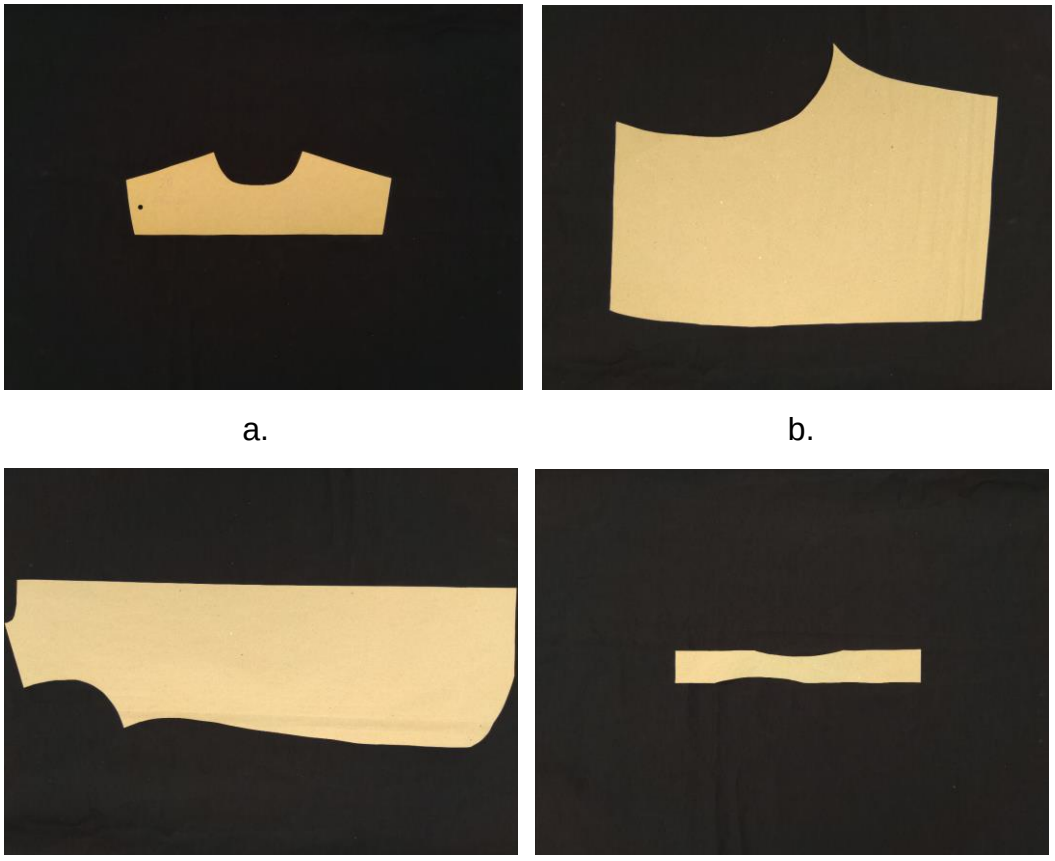


Figura 16. Representación de la estructura de adquisición de imágenes aplicando el protocolo planteado. Fuente: *(Elaboración Propia)*

Después de haber desarrollado los procedimientos antes mencionados se obtuvo una base de datos de 42 imágenes de moldes diferentes, pertenecientes a moldes de 10 prendas de vestir.



a. b.

c. d.

Figura 17. Imágenes capturadas de moldes textiles aplicando el protocolo de adquisición de imágenes. a. Camisa de varón. b. Short de mujer. c. Vestido de niña. d. Precinto de blusa de mujer. Fuente: *(Elaboración Propia)*

Las imágenes quedaron organizadas en carpetas nombradas según el número de prenda a la que pertenecían.

Figura 18. Organización de las imágenes capturadas mediante el protocolo de adquisición. Fuente: *(Elaboración Propia)*

Se realizó una búsqueda de investigaciones anteriores basados en el estudio de algoritmos de detección de bordes y vectorización, esta búsqueda se realizó en repositorios científicos como IEEE Xplore, Science Direct y ResearchGate, aplicando cadenas de búsqueda, ver tabla 9.

Tabla 9.

Cadenas de búsqueda aplicadas en los repositorios científicos.

CADENAS DE BÚSQUEDA
“Image and edge detection and algorithm”
“Images and methods and edge detection”
“Vectorization and algoritm and method and technique and image”
“Vectorization and image and algoritm”

Nota: (Elaboración propia)

Se obtuvieron investigaciones relacionadas a la aplicación de los distintos algoritmos y métodos, según los cuales se realizó un cuadro comparativo de los resultados del desempeño de cada algoritmo o método evaluados según el investigador.

Tabla 10.

Cuadro representativo de los resultados de investigaciones.

	Técnicas				
	Wavelet	Canny	Sobel	Prewitt	Otra Propuesta
(Ganesan & Sajiv, 2018)	x	x			
(Israni & Swapnil, 2016)			x		
(S. Kaur & Kaur, 2017)					x
(Khairudin & Irmawati, 2017)		x		x	
(Vikas et al., 2016)	x				
(Burnham et al., 1998)		x			
(B. Kaur, 2013)	x				x
(Thanikkal et al., 2019)		x			
(Patel & Patel, 2017)					x

Nota: (Elaboración Propia)

Para la selección de los algoritmos de detección de bordes se realizó un cuadro comparativo de los algoritmos y métodos de detección de bordes aplicados en estudios anteriores, según el cuadro comparativo los investigadores tuvieron mejores resultados aplicando las técnicas Canny y Wavelet. En un estudio, Ganesan determina que Canny es una técnica que incluye un proceso de eliminación de ruido mediante un filtro gausiano y se recomienda usar en imágenes con considerable presencia de ruido (Ganesan & Sajiv, 2018) . Para Vikas, en su estudio determina que la técnica de Wavelet aplica filtros de multiresolución, excelentes para imágenes de curvas complejas y con fuerte presencia de ruido (Vikas et al., 2016). Esta investigación se planteó desarrollarse en imágenes en formato de escala de grises, por lo tanto, se consideró aplicar técnicas no complejas y se escogió usar un método mejorada de Canny, el método Canny-Deriche, Sobel y proponer una nueva técnica para realizar una comparación de su desempeño para este caso.

Para la selección de técnicas de vectorización se determinó proponer 3 técnicas, que fueron comparadas entre sí.

Al empezar la implementación de las técnicas seleccionadas se pasó todas las imágenes a un proceso de redimensión, se transformó a 512x683 manteniendo la proporción de la dimensión original, a fin de no perder información relevante de la imagen. Posteriormente se convirtió del formato RGB a escala de grises en una transformación a 8 bits. La implementación de las técnicas seleccionadas estará basada en imageJ, un programa de procesamiento de imágenes desarrollado a base de lenguaje de programación java.

Para aplicar el método de Canny-Deriche se utilizó un plugin implementado en imageJ por (Meys J. & Boudier, T.), este plugin se importó desde imageJ y se aplicó.

Para implementar la técnica de sobel, se transforma la imagen en formato B/N aplicando el filtro de otsu, posteriormente se utilizó el plugin ya implementado en el entorno imageJ.

La técnica que se propuso para detectar bordes se realizó aplicando una conversión de imagen a B/N mediante el filtro de Otsu, posteriormente se duplicó la imagen teniendo imagen A y B respectivamente. B pasó otro filtro de erosión, este filtro consta de eliminar pixeles de los bordes de una imagen y en nuestro proceso se realizó una erosión de 2 pixeles. El siguiente fragmento de código muestra el funcionamiento de este filtro, en la línea uno y dos extrae las dimensiones de la imagen a procesar, la tercera línea cambia el formato bi-dimensional de la imagen en un vector de tipo byte, la cuarta línea es el mapa de distancias euclidianas que muestra los valores de pixeles más cercanos de fondo – objeto o viceversa, la quinta línea hace referencia a los valores de las esquinas de los bits de la imagen como un cálculo de diagonales, a partir de la línea seis realiza un recorrido de la imagen en función de las diagonales de los pixeles y aplica una condicional al pixel de EDM que guarda las distancias euclidianas, de modo que si ese valor en la posición actual es menor al radio de erosión ingresado se cambia

el valor del pixel actual (objeto) por el valor del pixel de fondo. En imageJ la función se aplica con: Process / Binary / Erode.

```
int width = ip.getWidth();
    int height = ip.getHeight();
    byte[] bPixels = (byte[])ip.getPixels();
    float[] fPixels = (float[])floatEdm.getPixels();
    Rectangle roiRect = ip.getRoi();
    for (int y=roiRect.y; y<roiRect.y+roiRect.height; y++)
        for (int x=roiRect.x, p=x+y*width; x<roiRect.x+roiRect.width; x++, p++)
            if (fPixels[p] <= radius)
                bPixels[p] = (byte)backgroundValue;
    return;
```

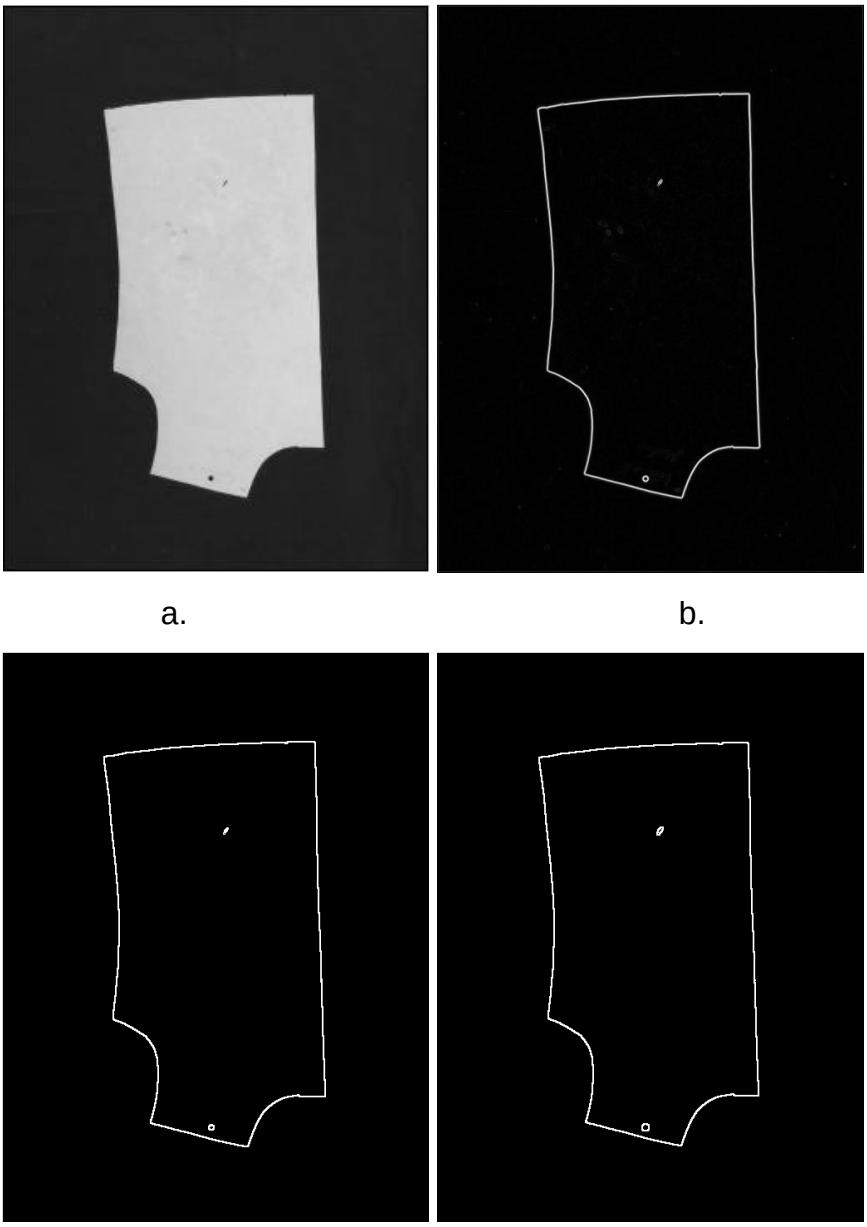
Figura 19. Fragmento del código fuente del algoritmo de erosión de la imagen. Fuente: *(ImageJ, 2020)*

Como último paso se aplicó una diferencia de imágenes A – B dando como resultado una nueva imagen que solo contiene los pixeles diferentes entre A y B. En la línea uno y dos, los parámetros ImagePlus reciben las imágenes a procesar, en las líneas tres y cuatro transforma las imágenes a un formato numérico para procesar, las líneas cinco en adelante recorre la imagen y realiza la resta de los valores de las imágenes y al terminar el recorrido la imagen B mostrará el resultado. En imageJ esta función se aplica con: Process / Image Calculator/ Difference.

```
ImagePlus img1 = WindowManager.getImage(wList[0]);
ImagePlus img2 = WindowManager.getImage(wList[1]);
ImageProcessor ip1 = img1.getProcessor();
ImageProcessor ip2 = img2.getProcessor();
for(int i=0; i<ip1.getHeight; i++){
for(int j=0; j<ip1.getWidht; j++){
        if(ip1.getPixel(i,j)-ip2.getpixel(i,j) < 0) {
                ip.putPixel(i, j, 0);
        }else{
                ip.putPixel(i, j, ip1.getPixel(i,j)-ip2.getpixel(i,j);
}}}
```

Figura 20. Fragmento del código fuente del algoritmo de diferencia de imágenes. Fuente: *(Elaboración Propia)*

Tras haber aplicado estos algoritmos se obtuvieron los siguientes resultados.



a. b.

c. d.

Figura 21. Resultados de la detección de bordes aplicando las técnicas seleccionadas. a. Imagen original 8 bits. b. Canny-Deriche. c. Threshold + sobel. d. Método Propuesto. Fuente: *(Elaboración Propia)*

La vectorización de las imágenes de bordes obtenidos se realizó en un proceso que consta de 3 pasos: Detección de esquinas, extracción de coordenadas y unir dichas coordenadas mediante líneas polinomiales.

Se implementaron 3 técnicas de detección de esquinas: método de Harris, detección de esquinas mediante pendientes y mediante el recorrido del borde.

El método de Harris conocido como Harris Corner en la web, implementado como una técnica popular de detección de esquinas, documentado en la página web de GitHub. (Burger W. & Burge M., 2017) Harris plantea el siguiente código, las primeras dos líneas extrae el dimensiones de la imagen partiendo del punto 1 al punto N-1 de cada dimensión, con el fin de no provocar un error al procesar cada pixel con valor 0 o N, en X y Y. La línea tres extrae el valor del pixel de harrismap en la posición (x,y), la cuarta línea hace una comparación de la medida mínima calculada de en un proceso anterior, de ser verdadera esa condición pasa a la línea cinco, esta condición llama a otra función la cual hace un recorrido de los 8 pixeles vecinos de (x,y), de modo que al encontrar valores de los pixeles vecinos mayores al punto actual determinando que es una esquina el cual es guardada como esquina en la línea seis. Posteriormente a este código se recorre la imagen nuevamente pintando los pixeles que contienen una esquina.

```
for (int y=1; y<this.height-1; y++) {
    for (int x=1; x<this.width-1; x++) {
        double h = harrismap[x][y];
        if (h<this.minMeasure) continue;
        if (!isSpatialMaxima(harrismap, (int)x, (int)y)) continue;
        getCorner(canal).add( new Corner(x,y,h) );
    }
}
```

Figura 22. Fragmento del código fuente del algoritmo Harris Corner. Fuente*: (Burger W. & Burger M., 2017)*

El método de Harris Corner pinta en forma de cruz los pixeles de las esquinas, asigna las posiciones 2x2 en vertical y horizontal en función de un pixel x,y de un color verde en la escala RGB, que en escala de grises representa un valor de pixel 85. Para extraer estas coordenadas se hace un recorrido de la imagen mostrando las posiciones de los pixeles que sean de valor 85. El siguiente código hace un recorrido de la imagen de salida del método de Harris asignando el valor 150, a las marcas de Harris e imprimiendo las coordenadas de ubicación de las esquinas en la imagen.

```
for (int j = 0; j < corners.getHeight(); j++) {
        for (int i = 0; i < corners.getWidth(); i++) {
            if (corners.getPixel(i, j) == 85) {
                corners.putPixel(i, j, 150);
                corners.putPixel(i, j + 1, 150);
                corners.putPixel(i, j + 2, 150);
                corners.putPixel(i, j + 3, 150);
                corners.putPixel(i, j + 4, 150);
                corners.putPixel(i - 2, j + 2, 150);
                corners.putPixel(i - 1, j + 2, 150);
                corners.putPixel(i + 1, j + 2, 150);
                corners.putPixel(i + 2, j + 2, 150);
      system.out.println(i+“  ”+(j+2));     }}}}
```

Figura 23. Fragmento del código fuente de la técnica de extracción de coordenadas de una imagen procesada por el algoritmo Harris Corner. Fuente: *(Elaboración Propia)*

La detección de esquinas mediante pendientes recorre la imagen de manera diagonal hasta encontrar un pixel que forme parte del borde seleccionado, e inicia un recorrido partiendo del punto A, avanza 20 pixeles de borde y marca un punto B, vuelve a recorrer 20 pixeles de borde hasta marcar un punto C. Teniendo los 3 puntos hace un cálculo representado por la siguiente fórmula matemática.

$$m = \frac{Y_2 - Y_1}{X_2 - X_1}$$

Dónde: m es la pendiente, X y Y representa las coordenas de un punto, se considera que siempre Y_1 o X_1 representen al punto central, en este caso el punto B. Tras tener las pendientes de la recta AB y CB se calcula la diferencia, de modo que a mayor diferencia de las pendientes existe un cruce de rectas perpendiculares que forma un ángulo más cerrado.

El siguiente código, esta estructura repetitiva while() realiza un recorrido en diagonal de pixeles hasta encontrar un pixel blanco.

```
while (camino == false) {
        if (ip.getPixel(x, y) == 255) {
            camino = true;
            inicio = x;
            fin = y;
        } else {
            x++;
            y++;
        }
    }
```

Figura 24. Fragmento del código fuente del recorrido en diagonal de una imagen. Fuente: *(Elaboración Propia)*

Esta estructura repetitiva hace el recorrido del borde donde se moviliza en orientación arriba, abajo, izquierda y derecha, u, d, l y r respectivamente. La secuencia de este método es el recorrido de izquierda – derecha según su orientación, y al llegar a un pixel de borde instancia una función de validación.

```
while (camino == true) {
        if (ip.getPixel(x, y) == 255 && label == 'r') {
            angulo(); label = 'u'; y = y - 1;
        }
        if (ip.getPixel(x, y) == 0 && label == 'r') {
            label = 'd';  = y + 1;
        }
        if (ip.getPixel(x, y) == 255 && label == 'u') {
            angulo(); label = 'l';  x = x - 1;
        }
        if (ip.getPixel(x, y) == 0 && label == 'u') {
            label = 'r';  x = x + 1;
        }
        if (ip.getPixel(x, y) == 255 && label == 'd') {
            angulo(); label = 'r';  x = x + 1;
        }
        if (ip.getPixel(x, y) == 0 && label == 'd') {
            label = 'l';  x = x - 1;
        }
        if (ip.getPixel(x, y) == 255 && label == 'l') {
            angulo(); label = 'd';  y = y + 1;
        }
        if (ip.getPixel(x, y) == 0 && label == 'l') {
            label = 'u';  y = y - 1;
       }
       If(x == inicio && y == fin){
          camino = false;
       }
```

Figura 25. Fragmento del código fuente del recorrido del borde de una imagen. Fuente: *(Elaboración Propia)*

La función hace un recorrido del borde pixel a pixel contando las iteraciones, al llegar a 1 marca el primer punto, al llegar a 10 marca el segundo punto y al llegar a 20 marca el tercer punto e instancia la función calcular que determina si existe o no un ángulo.

```
private void angulo() {
        iterac++;
        if (iterac == 1) {
            pix = x;
            piy = y;
        }
        if (iterac == 10) {
            pmx = x;
            pmy = y;
        }
        if (iterac == 20) {
            pfx = x;
            pfy = y;
            calcular();
        }
        if (angle == true && iterac == 20) {
            System.out.println(pmx + "  " + pmy);
            angle = false;
            iterac = 0;
        }
             iterac = 0;
}
```

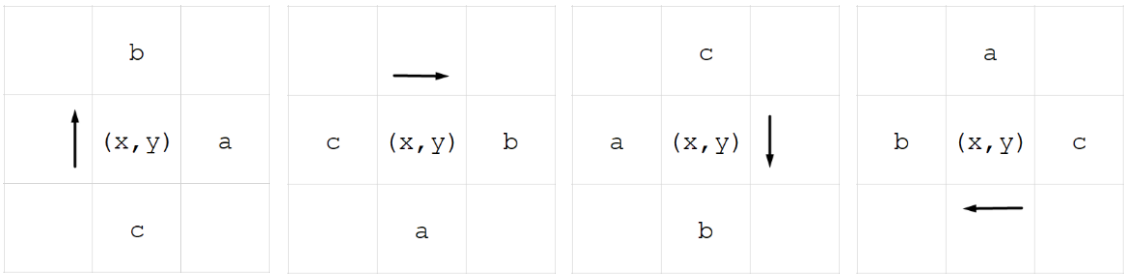
Figura 26. Fragmento del código fuente de la selección de puntos de referencia. Fuente: *(Elaboración Propia)*

La función aplica la fórmula matemática descrita anteriormente.

```
private void calcular() {
      difx = (double) (piy - pmy) / (pix - pmx);
      dify = (double) (pfy - pmy) / (pfx - pmx);
      if (difx - dify < 0) {
      if (-1 * (difx - dify) > 0.4 && (-1 * (difx - dify)) < 10) {
              angle = true;
          }
       } else {
          if (difx - dify > 0.4 && difx - dify < 10) {
              angle = true;
          }
       }}
```

Figura 27. Fragmento del código fuente del cálculo de las pendientes. Fuente: *(Elaboración Propia)*

La detección de bordes mediante el recorrido de borde pixel a pixel se inició realizando una búsqueda del borde tal como se representa en la figura 24. Posteriormente el proceso de seguimiento de borde se realizó bajo la referencia de la figura 28, donde a y b son los pixeles a evaluar, c es la posición anterior de (x,y) que va definiendo la orientación del movimiento. Existen cuatro combinaciones posibles para a y b: a=255 y b=0, a=255 y b=255, a=0 y b=255 o a=0 y b=0. Los movimientos de (x,y) dependen de los valores de a y b en función de su orientación; Si a=250 y b=0 avanza hacia adelante, si a=255 y b=255 avanza hacia la izquierda, antes teniendo en cuenta una excepción para evaluar una posible esquina invertida que se denota en la figura 33, si a =0 y b=0 avanza hacia la derecha y si a=0 y b=255 regresa hacia su posición anterior.



a. b. c. d.

Figura 28. Orientación de (x,y) según su posición anterior. a. hacia arriba. b. hacia derecha. c. hacia abajo. d. hacia izquierda. Fuente: *(Elaboración Propia)*

Al realizar en recorrido de la figura 28, se analiza la presencia de una esquina mediante en valor de los pixeles vecinos, si a=0 y b=0 o a=0 y b=255 es posible que exista una esquina, se evalúa como se representa en la figura 29, 31 y 33.

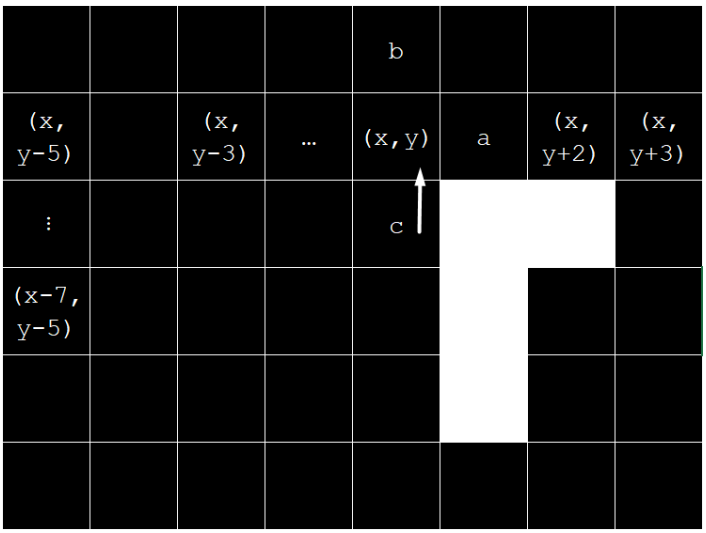


Figura 29. Representación del análisis de vecinos del pixel (x,y) con valores en a=0 y b=0; Fuente: *(Elaboración propia)*

El código fuente queda representado de la siguiente manera. Donde dist es la distancia con la que evaluará la esquina.

```
for (int i = 0; i < dist; i++) {
    if (img.getPixel(x - (dist / 2) + i, y) == 255) {
        corner = false;
    }
    if(img.getPixel(x - 5, y + i) == 255){
    corner = false;
    }
}
```

Figura 30. Fragmento del código fuente de la evaluación de una esquina aplicando el método propuesto. Fuente: *(Elaboración Propia)*

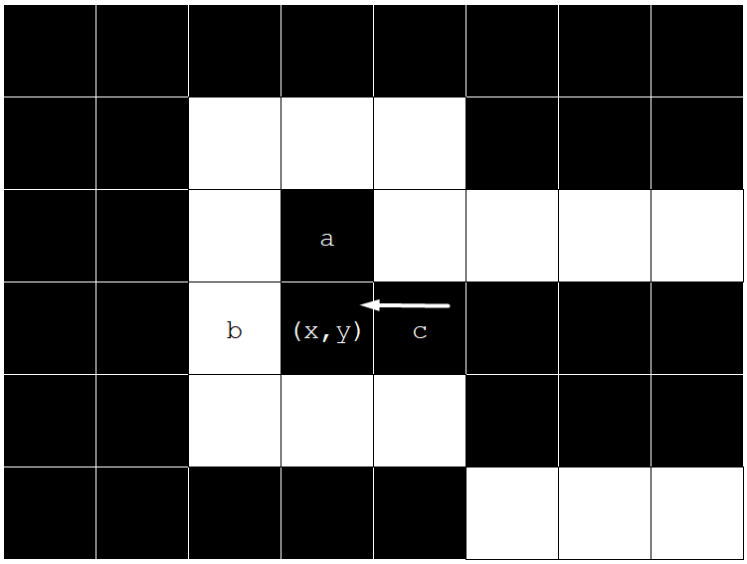


Figura 31. Representación del pixel (x,y) con valores en a=0 y b=255, donde (x,y) regresa a la posición c. Fuente: *(Elaboración Propia)*

El código fuente queda representado de la siguiente manera. Donde r es la dirección donde se moverá la posición actual.

```
if (ip.getPixel(x,y-1) == 0 && ip.getPixel(x-1,y) == 255) {
    System.out.println(x+"  "+y);
    moverse('r');
    break;
}
```

Figura 32. Fragmento del código fuente del funcionamiento del proceso mostrado en la figura 31. Fuente: *(Elaboración Propia)*

Si a=255 y b=255 puede ocurrir una excepción, que un tercer pixel d=255, donde definitivamente representa una esquina invertida.

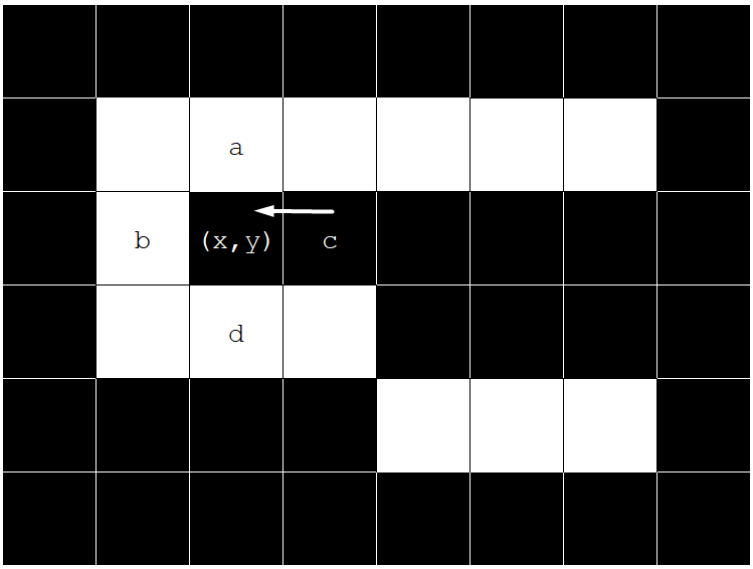


Figura 33. Representación de una esquina invertida, donde (x,y) regresa a "c". Fuente: *(Elaboración propia)*

La representación del código fuente se muestra a continuación.

```
if (ip.getPixel(x,y-1) == 255 && ip.getPixel(x-1,y) == 255) {
    if (ip.getPixel(x,y+1) == 255) {
        moverse('r');
        break;
     } else {
        moverse('d');
        break;
    }
 }
```

Figura 34. Fragmento del código fuente del funcionamiento del proceso mostrado en la figura 33. Fuente: *(Elaboración Propia)*

Tras aplicar estas técnicas de detección de esquinas se obtuvo los siguientes resultados.

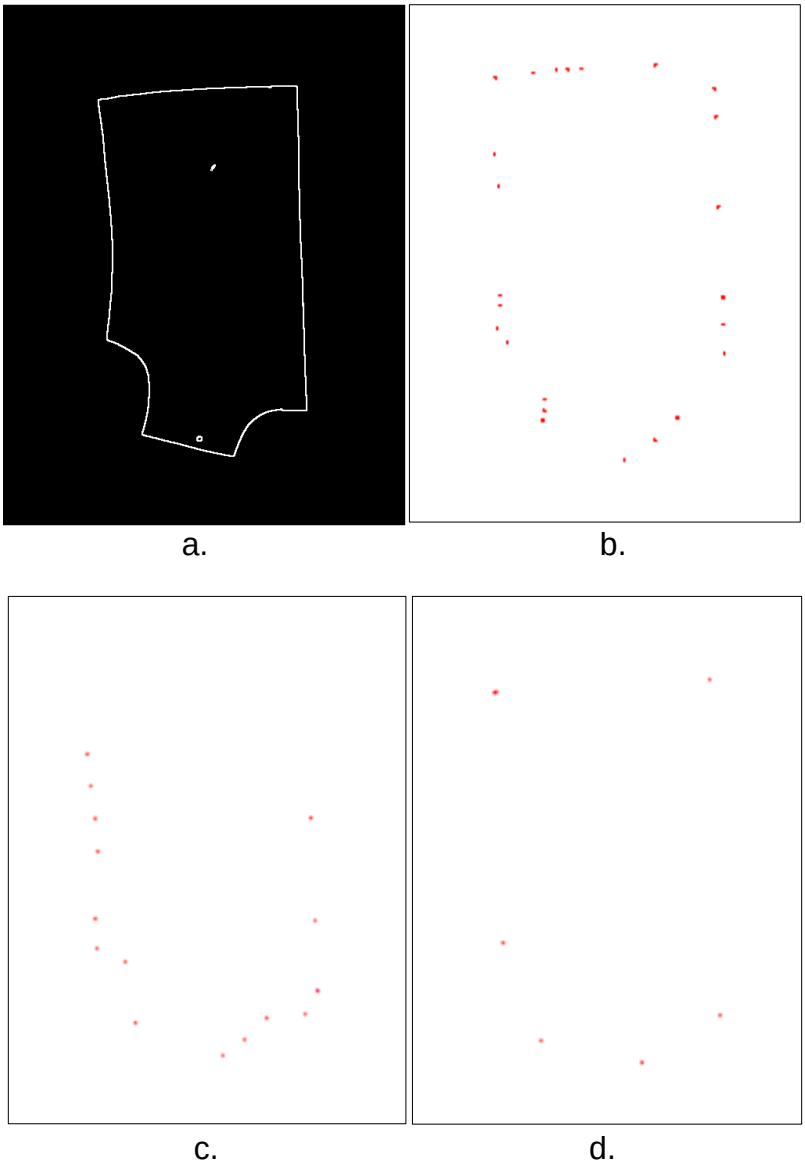


Figura 35. Resultados de la aplicación de las técnicas seleccionadas de detección de esquinas. a. Bordes de la imagen aplicando la técnica de Sobel. b. Esquinas detectadas por Harris Corner. c. Esquinas detectadas por Pendientes de rectas. d. Esquinas detectadas por recorrido pixel a pixel. Fuente: *(Elaboración Propia)*

Para completar el proceso de vectorización de las imágenes se realizó la extracción de coordenadas de las rectas existentes en la imagen. Cada recta quedó compuesta por punto extremos que son las esquinas detectadas y puntos internos o llamados puntos de control. La extracción de puntos de control se realizó mediante una medida simétrica de cada 50 pixeles a partir del punto de origen o esquina. Estas coordenadas fueron almacenadas en 2

listas independientes de su dimensión (X o Y). Cada lista inicia y termina en una esquina detectada, tal como se muestra en la figura 36.

```
contador++;
        if (contador > 50) {
            ejeX = ejeX + " " + x;
            ejeY = ejeY + " " + y;
            contador = 0;
        }
```

Figura 36. Fragmento de código fuente que almacena las coordenadas de los puntos de control de la recta. Fuente: *(Elaboración Propia)*

La representación gráfica de las rectas fue realizada utilizando el método interpolationSpline de la librería commoms-math de java, esta implementa el método de interpolación de puntos mediante splines cúbicos. Las imágenes resultantes se muestran en la figura 37.

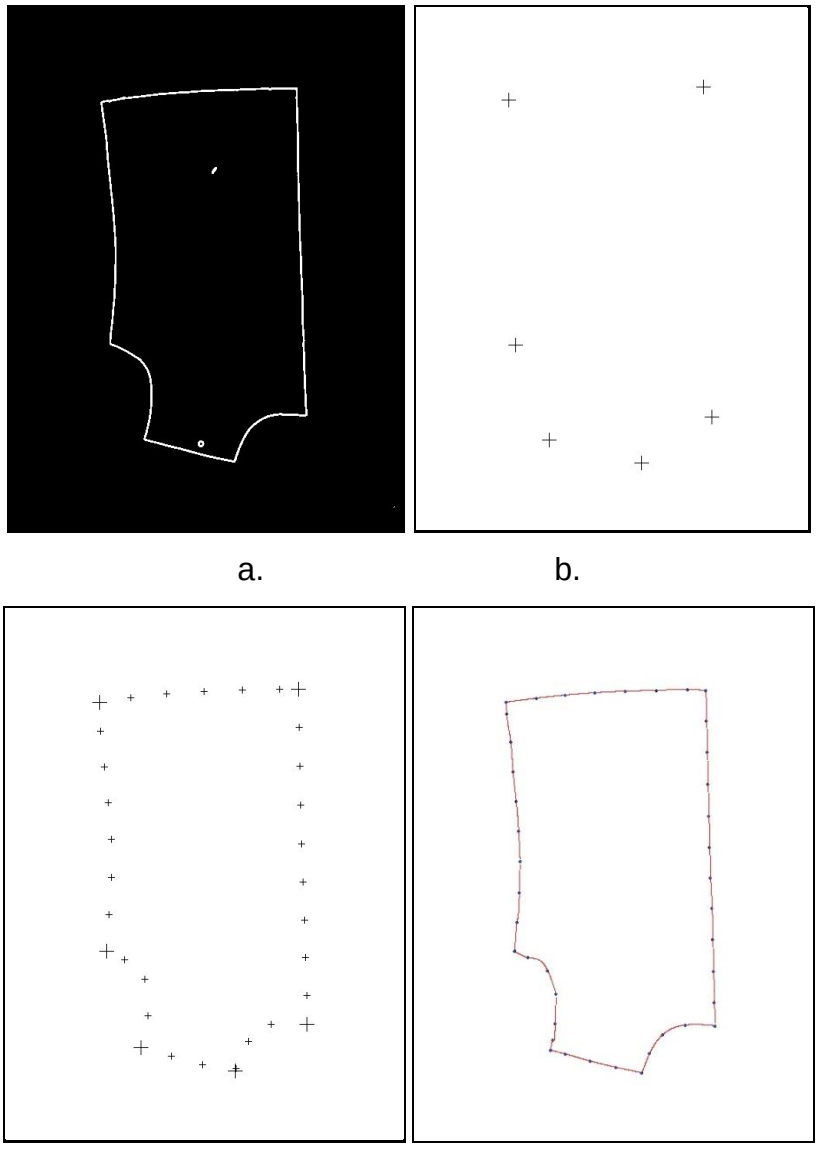


Figura 37. Resultados del proceso de vectorización de la imagen. a. Imagen original. b. Imagen con las esquinas detectadas mediante técnica propuesta. c. Imagen de las rectas formadas por esquinas y puntos de control. d. Imagen vectorial formada a partir de interpolación de puntos mediante splines cúbicos. Fuente: (*Elaboración Propia)*

IV. CONCLUSIONES Y RECOMENDACIONES

4.1. Conclusiones

a) El algoritmo de detección de bordes Sobel se considera la mejor técnica de detección de bordes para resolver el caso propuesto, puesto que demuestra información visual similar a las de os otros métodos comparados, pero, además, el consumo de recursos computacionales es el más bajo.

b) La técnica propuesta de detección de bordes por erosión es una técnica más simple en comparación a Canny-Deriche, además de mostrar consumo de recursos hasta 6 veces más bajo y tiempo de respuesta hasta 13.9 veces más rápido.

c) El algoritmo Harris Corner no fue el mejor algoritmo para detectar esquinas usando la metodología propuesta, pues genera excesivamente falsos positivos.

d) El algoritmo de detección de esquinas propuesto Pixel a Pixel, demostró un porcentaje de precisión de 90%, además en comparativa al método tradicional de Harris, Pixel a Pixel demostró consumo de memoria ram hasta 30.4 veces más bajo y tiempo de respuesta de hasta 68.9 veces más rápido.

4.2. Recomendaciones

a) Se recomienda recrear una base de datos a base de imágenes estandarizadas, de esta forma los algoritmos se ejecutan en las mismas condiciones siendo sus resultados más confiables.

b) Los resultados obtenidos en esta investigación no representan la misma credibilidad para ser aplicados en distintas metodologías, estos podrían ser más o menos confiables según las condiciones aplicadas.

REFERENCIAS

Barina, D., & Zemcik, P. (2014). DIAGONAL VECTORISATION OF 2-D WAVELET LIFTING David Barina Faculty of Information Technology Brno University of Technology Czech Republic Pavel Zemcik Faculty of Information Technology Brno University of Technology Czech Republic. *International Conference on Image Processing(ICIP)*, 2978–2982.

BCRP. (2019). *Textiles - Prendas de Vestir y otras Confecciones*. https://estadisticas.bcrp.gob.pe/estadisticas/series/mensuales/resultados/PN01578BM/html/2018-7/2019-1/

Burger Wilhem, & Burge Mark J. (2017). *Source Code | Digital Image Processing*. https://imagingbook.com/source/

Burnham, J., Hardy, J., Meadors, K., & Picone, J. (1998). A Comparison of the Roberts, Sobel, Robinson, Canny, and Hough Image Detection Algorithms. *Proceedings of IEEE Southeastcon*. https://doi.org/http://www.isip.piconepress.com/publications/unpublished/conferences/1998/ieee_secon/edge_detection/

Choi, W.-J., Choi, T.-S., Hasanabadi, H., Zabihi, M., Mirsharif, Q., Ye, X., Lin, X., Dehmeshki, J., Slabaugh, G., Beddoe, G. R., Liu, Y., Zhang, D. D., Lu, G., Ma, W., Peng, B., Zhang, L., Zhang, D. D., Müller, H., Michoux, N., … Kandwal, R. (2013). Dip Textbook. *IEEE Transactions on Biomedical Engineering*, *5*(1), 507–523. https://doi.org/10.1109/TBME.2009.2017027

ComexPerú. (2018). ComexPerú - Sociedad de Comercio Exterior del Perú. In *2018* (p. 1).

Creditex S.A.A. (2016). *Reducción de Mermas*.

Dai, W., Luo, T., & Shen, J. (2013). Automatic image vectorization using superpixels and random walkers. *Proceedings of the 2013 6th International Congress on Image and Signal Processing, CISP 2013*, *2*(Cisp), 922–926. https://doi.org/10.1109/CISP.2013.6745296

De, P. (2019). Vectorization of Architectural Floor Plans. *2019 12th International Conference on Contemporary Computing, IC3 2019*, 1–5.

https://doi.org/10.1109/IC3.2019.8844930

Donati, L., Cesano, S., & Prati, A. (2019). A complete hand-drawn sketch vectorization framework. *Multimedia Tools and Applications*, *78*(14), 19083–19113. https://doi.org/10.1007/s11042-019-7311-3

Eclipse Foundation. (2019). *Eclipse desktop & web IDEs | The Eclipse Foundation*. https://www.eclipse.org/ide/

Escobar, J. C. (2014). *Confección de prendas de vestir*.

Ganesan, P., & Sajiv, G. (2018). A comprehensive study of edge detection for image processing applications. *Proceedings of 2017 International Conference on Innovations in Information, Embedded and Communication Systems, ICIIECS 2017*, *2018-Janua*, 1–6. https://doi.org/10.1109/ICIIECS.2017.8275968

Guiming, S., & Jidong, S. (2018). Multi-Scale Harris Corner Detection Algorithm Based on Canny Edge-Detection. *2018 IEEE International Conference on Computer and Communication Engineering Technology, CCET 2018*, 305–309. https://doi.org/10.1109/CCET.2018.8542206

Han, S., Yu, W., Yang, H., & Wan, S. (2019). An Improved Corner Detection Algorithm Based on Harris. *Proceedings 2018 Chinese Automation Congress, CAC 2018*, 1575–1580. https://doi.org/10.1109/CAC.2018.8623814

Hou, F., Sun, Q., Fang, Z., Liu, Y. J., Hu, S. M., Hao, A., Qin, H., & He, Y. (2018). Poisson Vector Graphics (PVG). *IEEE Transactions on Visualization and Computer Graphics*, *PP*(8), 1. https://doi.org/10.1109/TVCG.2018.2867478

IBM Developer. (2019). *Introduction to Java programming, Part 1: Java language basics – IBM Developer*. https://developer.ibm.com/tutorials/j-introtojava1/

ImageJ. (2019). *Introduction*. https://imagej.nih.gov/ij/docs/intro.html

ImageJ. (2020). *Plugins*. https://imagej.nih.gov/ij/plugins/

Israni, S., & Swapnil, J. (2016). *Edge Detection of License Plate*. 3561–3563.

Kaur, B. (2013). *An Edge Detection Approach Based on Wavelets*. *2*(9), 2836–2840.

Kaur, S., & Kaur, P. (2017). An Edge detection technique with image segmentation

using Ant Colony Optimization: A review. *Proceedings of 2016 Online International Conference on Green Engineering and Technologies, IC-GET 2016*. https://doi.org/10.1109/GET.2016.7916741

Khairudin, M., & Irmawati, D. (2017). Comparison methods of edge detection for USG images. *Proceedings - 2016 3rd International Conference on Information Technology, Computer, and Electrical Engineering, ICITACEE 2016*, 85–88. https://doi.org/10.1109/ICITACEE.2016.7892416

La República. (2018). *Las exportaciones peruanas*. 2018. https://larepublica.pe/politica/1276793-exportaciones-peruanas/

Meys, J., & Boudier, T. (n.d.). *Wiki de documentación de ImageJ*. Detección de Bordes Mediante Filtrado Canny-Deriche. Retrieved November 15, 2020, from https://imagejdocu.tudor.lu/plugin/filter/edge_detection/start

Ministerio de la Producción. (2017). Industria Textil y Confecciones: Estudio de investigación sectorial. *Textil Y Confecciones*, 157.

Morse, B. S. (2000). Image Processing Review , Neighbors , Connected Components , and Distance. *Image Processing*, 1998–2000.

Pan, W., Lian, Z., Tang, Y., & Xiao, J. (2014). Skeleton-guided vectorization of Chinese calligraphy images. *2014 IEEE International Workshop on Multimedia Signal Processing, MMSP 2014*. https://doi.org/10.1109/MMSP.2014.6958805

Patel, A., & Patel, A. (2017). Performance enhancement in image edge detection technique. *2016 International Conference on Signal and Information Processing, IConSIP 2016*. https://doi.org/10.1109/ICONSIP.2016.7857478

Schindelin, J., Arganda-Carreras, I., Frise, E., Kaynig, V., Longair, M., Pietzsch, T., Preibisch, S., Rueden, C., Saalfeld, S., Schmid, B., Tinevez, J.-Y., White, D. J., Hartenstein, V., Eliceiri, K., Tomancak, P., & Cardona, A. (2012). Fiji: an open-source platform for biological-image analysis. *Nature Methods*, *9*(7), 676–682. https://doi.org/10.1038/nmeth.2019

Thanikkal, J. G., Dubey, A. K., & Thomas, M. T. (2019). Advanced Plant Leaf Classification Through Image Enhancement and Canny Edge Detection. *2018*

7th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), 1–5. https://doi.org/10.1109/icrito.2018.8748587

UNED. (2008). *Gráficos vectoriales y de mapa de bits*. 1–177.

Vikas, P., Lakshmi, M. S., Rajkumar, M. S., & Prasad, P. M. K. (2016). Edge detection in noisy images using wavelet transform. *RAECE 2015 - Conference Proceedings, National Conference on Recent Advances in Electronics and Computer Engineering*, 36–39. https://doi.org/10.1109/RAECE.2015.7510222

Yang, M., Chao, H., Zhang, C., Guo, J., Yuan, L., & Sun, J. (2016). Effective clipart image vectorization through direct optimization of bezigons. *IEEE Transactions on Visualization and Computer Graphics*, *22*(2), 1063–1075. https://doi.org/10.1109/TVCG.2015.2440273

Yin, Y., Zhao, H., Yang, F., Shang, D., & Liu, J. (2016). A vectorization method of building edge based on high resolution DSM data. *International Geoscience and Remote Sensing Symposium (IGARSS)*, *2016-Novem*, 1572–1574. https://doi.org/10.1109/IGARSS.2016.7729401

ANEXOS

Anexo 1. Resolución de Ampliación de Vigencia de Proyecto de Tesis

FACULTAD DE INGENIERÍA, ARQUITECTURA Y URBANISMO
RESOLUCIÓN N°2325-2020/FIAU-USS

Pimentel, 17 de noviembre de 2020

VISTO:
El oficio N°0237-2020/FIAU-IS-USS de fecha 12 de noviembre de 2020, de la Dirección de Escuela profesional de INGENIERÍA DE SISTEMAS con el que remite el Acta de reunión N°2610-2020 del Comité de investigación de la referida Escuela profesional, para el desarrollo de la Tesis presentada por estudiantes del Programa de estudios de INGENIERÍA DE SISTEMAS, y;

CONSIDERANDO:

Que, de conformidad con la Ley Universitaria N° 30220 en su artículo 48º que a letra dice: "La investigación constituye una función esencial y obligatoria de la universidad, que la fomenta y realiza, respondiendo a través de la producción de conocimiento y desarrollo de tecnologías a las necesidades de la sociedad, con especial énfasis en la realidad nacional. Los docentes, estudiantes y graduados participan en la actividad investigadora en su propia institución o en redes de investigación nacional o internacional, creadas por las instituciones universitarias públicas o privadas.";

Que, de conformidad con el Reglamento de grados y títulos en su artículo 21° señala: "Los temas de trabajo de investigación, trabajo académico y *tesis* son *aprobados por el Comité de Investigación* y derivados a la facultad o Escuela de Posgrado, según corresponda, para la emisión de la resolución respectiva. El *periodo de vigencia de los mismos será de dos años*, a partir de su aprobación. En caso un tema perdiera vigencia, el Comité de Investigación evaluará la ampliación de la misma.

Que, de conformidad con el Reglamento de grados y títulos en su artículo 24º señala: La tesis es un estudio que debe denotar rigurosidad metodológica, originalidad, relevancia social, utilidad teórica y/o práctica en el ámbito de la escuela profesional. Para el grado de doctor se requiere una tesis de máxima rigurosidad académica y de carácter original. Es individual para la obtención de un grado; *es individual o en pares para obtener un título profesional*. Asimismo, en su artículo 25° señala: "El tema debe responder a alguna de las líneas de investigación institucionales de la USS S.A.C.".

Que, mediante documentos de vistos, el Comité de investigación de la referida Escuela profesional acordó aprobar la ampliación de la vigencia de las tesis que se detallan en el Acta de reunión N°2610-2020, de la línea de investigación de INFRAESTRUCTURA, TECNOLOGÍA Y AMBIENTE, a cargo de estudiantes del Programa de estudios INGENIERÍA DE SISTEMAS.

Estando a lo expuesto, y en uso de las atribuciones conferidas y de conformidad con las normas y reglamentos vigentes;

SE RESUELVE:

ARTÍCULO ÚNICO: AMPLIAR VIGENCIA, de la Tesis a cargo de los estudiantes del Programa de estudios de **INGENIERÍA DE SISTEMAS** que se detallan en el anexo de la presente Resolución, perteneciente a la línea de investigación de **INFRAESTRUCTURA, TECNOLOGÍA Y AMBIENTE**, hasta el **31 de julio de 2021.**

REGÍSTRESE, COMUNÍQUESE Y ARCHÍVESE

Dr. Mario Fernando Ramos Moscol
Decano - Facultad de Ingeniería, Arquitectura y Urbanismo
UNIVERSIDAD SEÑOR DE SIPÁN SAC.

MBA. María Noelia Sialer Rivera
Secretaria Académica / Facultad de Ingeniería, Arquitectura y Urbanismo
UNIVERSIDAD SEÑOR DE SIPÁN SAC.

Cc: Interesado, Archivo

Anexo 2. Ficha de observación para evaluar algoritmos de detección de bordes.

		ALGORITMO SOBEL				ALGORITMO CANNY-DERICHE				TÉCNICA DE EROSIÓN			
		MSE	PSNR	Consumo Ram	T. Respuesta	MSE	PSNR	Consumo Ram	T. Respuesta	MSE	PSNR	Consumo Ram	T. Respuesta
DATABASE DE IMÁGENES	Imagen 01												
	Imagen 02												
	Imagen 03												
	Imagen 04												
	Imagen 05												
	Imagen 06												
	Imagen 07												
	Imagen 08												
	Imagen 09												
	Imagen 10												
	Imagen 11												
	Imagen 12												
	Imagen 13												
	Imagen 14												
	Imagen 15												
	Imagen 16												
	Imagen 17												
	Imagen 18												
	Imagen 19												
	Imagen 20												
	Imagen 21												
	Imagen 22												
	Imagen 23												
	Imagen 24												
	Imagen 25												
	Imagen 26												
	Imagen 27												
	Imagen 28												
	Imagen 29												
	Imagen 30												
	Imagen 31												
	Imagen 32												
	Imagen 33												
	Imagen 34												
	Imagen 35												
	Imagen 36												
	Imagen 37												
	Imagen 38												
	Imagen 39												
	Imagen 40												
	Imagen 41												
	Imagen 42												
	Imagen 43												
	Imagen 44												

Anexo 3. Formato de ficha de observación para evaluar los algoritmos de detección de esquinas vectorizables.

		ALGORITMO									
		Esquinas Detectadas	Esquinas Reales	VP	FP	FN	Exactitud	Presición	Recall	Consumo Mem. Ram	Tiempo de Respuesta
DATABASE DE IMÁGENES	Imagen 01										
	Imagen 02										
	Imagen 03										
	Imagen 04										
	Imagen 05										
	Imagen 06										
	Imagen 07										
	Imagen 08										
	Imagen 09										
	Imagen 10										
	Imagen 11										
	Imagen 12										
	Imagen 13										
	Imagen 14										
	Imagen 15										
	Imagen 16										
	Imagen 17										
	Imagen 18										
	Imagen 19										
	Imagen 20										
	Imagen 21										
	Imagen 22										
	Imagen 23										
	Imagen 24										
	Imagen 25										
	Imagen 26										
	Imagen 27										
	Imagen 28										
	Imagen 29										
	Imagen 30										
	Imagen 31										
	Imagen 32										
	Imagen 33										
	Imagen 34										
	Imagen 35										
	Imagen 36										
	Imagen 37										
	Imagen 38										
	Imagen 39										
	Imagen 40										
	Imagen 41										
	Imagen 42										
	Imagen 43										
	Imagen 44										

Anexo 4. Valores obtenidos en las pruebas realizadas.

Tabla 11.

Valores obtenidos en los criterios de evaluación al aplicar el algoritmo de detección de bordes sobel

	ALGORITMO SOBEL			
	MSE	PSNR	Consumo Mem. Ram (kb)	Tiempo de Respuesta (ms)
Imagen 01	815.953125	19.01415151	443	11
Imagen 02	638.5195313	20.07906175	1681	4
Imagen 03	879.9257813	18.68634318	1614	4
Imagen 04	532.3007813	20.86923257	1570	4
Imagen 05	482.8125	21.29301855	1538	4
Imagen 06	815.953125	19.01415151	1538	4
Imagen 07	1478.613281	16.43225758	1538	3
Imagen 08	1835.894531	15.49232633	1538	8
Imagen 09	1931.25	15.27241864	1015	6
Imagen 10	1742.953125	15.71794654	1681	12
Imagen 11	203.9882813	25.03475142	571	7
Imagen 12	1015.113281	18.06565851	1570	6
Imagen 13	946.3125	18.37045784	1570	6
Imagen 14	1652.425781	15.94958399	1538	6
Imagen 15	879.9257813	18.68634318	1570	6
Imagen 16	2444.238281	14.24936819	1538	6
Imagen 17	1931.25	15.27241864	1570	7
Imagen 18	2129.203125	14.84863266	1538	5
Imagen 19	638.5195313	20.07906175	1570	7
Imagen 20	2781	13.68879372	1036	5
Imagen 21	1652.425781	15.94958399	1109	5
Imagen 22	1835.894531	15.49232633	1141	7
Imagen 23	1314.457031	16.94333967	1109	5
Imagen 24	1236	17.2106189	1015	8
Imagen 25	1742.953125	15.71794654	1681	7

Imagen 26	1835.894531	15.49232633	1614	6
Imagen 27	2336.8125	14.44456494	1570	6
Imagen 28	2029.019531	15.05794133	1570	7
Imagen 29	1159.957031	17.48638459	1538	6
Imagen 30	946.3125	18.37045784	1538	5
Imagen 31	2129.203125	14.84863266	1538	6
Imagen 32	1395.328125	16.68404013	1538	6
Imagen 33	2898.082031	13.50969687	1538	7
Imagen 34	482.8125	21.29301855	1538	7
Imagen 35	2129.203125	14.84863266	1538	6
Imagen 36	1395.328125	16.68404013	1538	6
Imagen 37	2129.203125	14.84863266	1538	6
Imagen 38	2029.019531	15.05794133	1538	7
Imagen 39	1564.3125	16.18756845	1538	6
Imagen 40	1015.113281	18.06565851	1538	6
Imagen 41	1742.953125	15.71794654	1538	5
Imagen 42	2666.332031	13.87166131	1538	5
Imagen 43	2781	13.68879372	1538	7
Imagen 44	1395.328125	16.68404013	1538	5

Nota: (Elaboración Propia)

Tabla 12

Valores obtenidos en los criterios de evaluación al aplicar el algoritmo de detección de bordes Canny-Deriche

	ALGORITMO CANNY-DERICHE			
	MSE	PSNR	Consumo Mem. Ram (kb)	Tiempo de Respuesta (ms)
Imagen 01	309	23.23121881	21264	153
Imagen 02	435.7382813	21.73854645	21181	145
Imagen 03	309	23.23121881	20948	219
Imagen 04	203.9882813	25.03475142	20715	97
Imagen 05	435.7382813	21.73854645	20282	103

Imagen 06	309	23.23121881	20520	96
Imagen 07	309	23.23121881	20282	104
Imagen 08	435.7382813	21.73854645	20520	95
Imagen 09	391.078125	22.20816837	20556	94
Imagen 10	391.078125	22.20816837	20697	103
Imagen 11	309	23.23121881	20449	93
Imagen 12	348.8320313	22.70464004	20465	94
Imagen 13	391.078125	22.20816837	20373	96
Imagen 14	271.5820313	23.79179329	20398	97
Imagen 15	391.078125	22.20816837	20373	94
Imagen 16	236.578125	24.39105775	20397	95
Imagen 17	348.8320313	22.70464004	20388	96
Imagen 18	271.5820313	23.79179329	20319	96
Imagen 19	203.9882813	25.03475142	20319	97
Imagen 20	309	23.23121881	20319	98
Imagen 21	435.7382813	21.73854645	20331	94
Imagen 22	391.078125	22.20816837	20363	95
Imagen 23	309	23.23121881	20345	94
Imagen 24	482.8125	21.29301855	20397	94
Imagen 25	435.7382813	21.73854645	20326	93
Imagen 26	391.078125	22.20816837	20356	101
Imagen 27	97.76953125	28.22876828	20395	95
Imagen 28	638.5195313	20.07906175	20440	96
Imagen 29	309	23.23121881	20388	92
Imagen 30	271.5820313	23.79179329	20374	93
Imagen 31	203.9882813	25.03475142	20346	111
Imagen 32	435.7382813	21.73854645	20349	94
Imagen 33	1314.457031	16.94333967	20345	97
Imagen 34	348.8320313	22.70464004	20439	96
Imagen 35	236.578125	24.39105775	20474	93
Imagen 36	271.5820313	23.79179329	20362	101
Imagen 37	271.5820313	23.79179329	20357	92
Imagen 38	146.0507813	26.48576476	20344	97

Imagen 39	203.9882813	25.03475142	20344	95
Imagen 40	348.8320313	22.70464004	20457	95
Imagen 41	236.578125	24.39105775	20346	95
Imagen 42	348.8320313	22.70464004	20370	94
Imagen 43	309	23.23121881	20344	94
Imagen 44	1395.328125	16.68404013	21281	221

Nota: (Elaboración Propia)

Tabla 13

Valores obtenidos en los criterios de evaluación al aplicar la técnica de detección de bordes de Erosión

	TÉCNICA DE EROSIÓN			
	MSE	PSNR	Consumo Mem. Ram (kb)	Tiempo de Respuesta (ms)
Imagen 01	815.953125	19.01415151	2990	8
Imagen 02	638.5195313	20.07906175	3066	8
Imagen 03	879.9257813	18.68634318	3591	8
Imagen 04	532.3007813	20.86923257	2989	7
Imagen 05	482.8125	21.29301855	2989	8
Imagen 06	815.953125	19.01415151	2989	8
Imagen 07	1478.613281	16.43225758	2564	8
Imagen 08	1835.894531	15.49232633	3632	8
Imagen 09	1931.25	15.27241864	2966	8
Imagen 10	1742.953125	15.71794654	2966	8
Imagen 11	203.9882813	25.03475142	2966	8
Imagen 12	1015.113281	18.06565851	2966	7
Imagen 13	946.3125	18.37045784	2966	6
Imagen 14	1652.425781	15.94958399	2990	6
Imagen 15	879.9257813	18.68634318	2587	6
Imagen 16	2444.238281	14.24936819	2966	5
Imagen 17	1931.25	15.27241864	2966	6

Imagen 18	2129.203125	14.84863266	2966	6
Imagen 19	638.5195313	20.07906175	2966	6
Imagen 20	2781	13.68879372	2966	5
Imagen 21	1652.425781	15.94958399	2966	6
Imagen 22	1835.894531	15.49232633	2966	6
Imagen 23	1314.457031	16.94333967	2564	6
Imagen 24	1236	17.2106189	2966	7
Imagen 25	1742.953125	15.71794654	2966	6
Imagen 26	1835.894531	15.49232633	2966	6
Imagen 27	2336.8125	14.44456494	2258	8
Imagen 28	2029.019531	15.05794133	2564	6
Imagen 29	1159.957031	17.48638459	2990	7
Imagen 30	946.3125	18.37045784	1927	6
Imagen 31	2129.203125	14.84863266	3022	8
Imagen 32	1395.328125	16.68404013	2990	8
Imagen 33	2898.082031	13.50969687	3050	8
Imagen 34	482.8125	21.29301855	3018	8
Imagen 35	2129.203125	14.84863266	2587	8
Imagen 36	1395.328125	16.68404013	3018	7
Imagen 37	2129.203125	14.84863266	3018	7
Imagen 38	2029.019531	15.05794133	2477	7
Imagen 39	1564.3125	16.18756845	2994	7
Imagen 40	1015.113281	18.06565851	2564	7
Imagen 41	1742.953125	15.71794654	2994	7
Imagen 42	2666.332031	13.87166131	2994	6
Imagen 43	2781	13.68879372	2994	6
Imagen 44	1395.328125	16.68404013	2360	8

Nota: (Elaboración Propia)

Tabla 14

Valores obtenidos en los criterios de evaluación al aplicar el algoritmo de detección de esquinas Harris Corner

	ALGORITMO HARRIS CORNER									
	Esquinas Detectadas	Esquinas Reales	VP	FP	FN	Exactitud	Precisión	Recall	Consumo Mem. Ram (kb)	Tiempo de Respuesta (ms)
Imagen 01	28	4	4	24	0	0.14	0.14	1.00	26101	169
Imagen 02	45	4	4	41	0	0.09	0.09	1.00	26423	75
Imagen 03	23	4	4	19	0	0.17	0.17	1.00	25360	73
Imagen 04	45	6	6	39	0	0.13	0.13	1.00	25084	69
Imagen 05	44	5	5	39	0	0.11	0.11	1.00	25429	190
Imagen 06	77	6	6	71	0	0.08	0.08	1.00	25520	63
Imagen 07	51	6	6	45	0	0.12	0.12	1.00	24129	61
Imagen 08	62	4	4	58	0	0.06	0.06	1.00	24111	62
Imagen 09	31	4	4	27	0	0.13	0.13	1.00	23991	65
Imagen 10	33	6	6	27	0	0.18	0.18	1.00	24061	65
Imagen 11	59	6	6	53	0	0.10	0.10	1.00	24128	62
Imagen 12	108	4	4	104	0	0.04	0.04	1.00	24123	63
Imagen 13	92	5	5	87	0	0.05	0.05	1.00	24013	63
Imagen 14	13	4	4	9	0	0.31	0.31	1.00	24023	61
Imagen 15	57	5	5	52	0	0.09	0.09	1.00	24135	61
Imagen 16	74	5	5	69	0	0.07	0.07	1.00	24077	63
Imagen 17	25	4	4	21	0	0.16	0.16	1.00	24112	63

Imagen 18	55	4	4	51	0	0.07	0.07	1.00	24110	62
Imagen 19	24	4	4	20	0	0.17	0.17	1.00	24081	63
Imagen 20	13	3	3	10	0	0.23	0.23	1.00	24093	62
Imagen 21	62	6	6	56	0	0.10	0.10	1.00	24106	64
Imagen 22	56	5	5	51	0	0.09	0.09	1.00	24058	62
Imagen 23	45	3	3	42	0	0.07	0.07	1.00	24016	62
Imagen 24	48	4	4	44	0	0.08	0.08	1.00	24077	64
Imagen 25	41	6	6	35	0	0.15	0.15	1.00	24093	62
Imagen 26	39	6	6	33	0	0.15	0.15	1.00	24069	63
Imagen 27	64	5	5	59	0	0.08	0.08	1.00	24048	60
Imagen 28	96	5	5	91	0	0.05	0.05	1.00	24057	61
Imagen 29	43	6	6	37	0	0.14	0.14	1.00	24065	62
Imagen 30	39	6	6	33	0	0.15	0.15	1.00	24069	66
Imagen 31	26	3	3	23	0	0.12	0.12	1.00	24000	62
Imagen 32	84	4	4	80	0	0.05	0.05	1.00	24010	62
Imagen 33	93	4	4	89	0	0.04	0.04	1.00	24057	60
Imagen 34	44	4	4	40	0	0.09	0.09	1.00	24069	63
Imagen 35	84	10	10	74	0	0.12	0.12	1.00	24070	62
Imagen 36	31	4	4	27	0	0.13	0.13	1.00	24065	62
Imagen 37	21	4	4	17	0	0.19	0.19	1.00	24062	62
Imagen 38	90	8	8	82	0	0.09	0.09	1.00	24057	61
Imagen 39	78	9	8	70	1	0.10	0.10	0.89	24039	64

Imagen 40	92	3	3	89	0	0.03	0.03	1.00	24061	62
Imagen 41	76	9	9	67	0	0.12	0.12	1.00	24040	65
Imagen 42	56	4	4	52	0	0.07	0.07	1.00	230921	63
Imagen 43	87	7	7	80	0	0.08	0.08	1.00	24029	64
Imagen 44	98	4	4	94	0	0.04	0.04	1.00	23986	62

Nota: (Elaboración Propia)

Tabla 15

Valores obtenidos en los criterios de evaluación al aplicar la técnica de detección de esquinas Pendientes

	TÉCNICA DE PENDIENTES									
	Esquinas Detectadas	Esquinas Reales	VP	FP	FN	Exactitud	Precisión	Recall	Consumo Mem. Ram (kb)	Tiempo de Respuesta (ms)
Imagen 01	6	4	1	5	3	0.08	0.17	0.25	2326	2
Imagen 02	7	4	2	5	2	0.14	0.29	0.50	1095	2
Imagen 03	5	4	2	3	2	0.20	0.40	0.50	2353	1
Imagen 04	8	6	4	4	2	0.25	0.50	0.67	2006	1
Imagen 05	5	5	2	3	3	0.20	0.40	0.40	1686	1
Imagen 06	15	6	4	11	2	0.13	0.27	0.67	1540	1
Imagen 07	4	6	1	3	5	0.13	0.25	0.17	1094	2
Imagen 08	11	4	1	10	3	0.05	0.09	0.25	2324	1
Imagen 09	5	4	1	4	3	0.10	0.20	0.25	1661	2

Imagen 10	6	6	1	5	5	0.08	0.17	0.17	1932	1
Imagen 11	9	6	2	7	4	0.11	0.22	0.33	662	2
Imagen 12	14	4	4	10	0	0.14	0.29	1.00	1932	1
Imagen 13	17	5	1	16	4	0.03	0.06	0.20	1888	2
Imagen 14	0	4	0	0	4	0.00	0.00	0.00	1888	2
Imagen 15	5	5	1	4	4	0.10	0.20	0.20	2663	5
Imagen 16	9	5	2	7	3	0.11	0.22	0.40	998	1
Imagen 17	0	4	0	0	4	0.00	0.00	0.00	1930	1
Imagen 18	8	4	1	7	3	0.06	0.13	0.25	1931	1
Imagen 19	5	4	2	3	2	0.20	0.40	0.50	1931	1
Imagen 20	3	3	1	2	2	0.17	0.33	0.33	1886	0
Imagen 21	11	6	1	10	5	0.05	0.09	0.17	1886	1
Imagen 22	13	5	2	11	3	0.08	0.15	0.40	1886	0
Imagen 23	8	3	3	5	0	0.19	0.38	1.00	1931	1
Imagen 24	6	4	1	5	3	0.08	0.17	0.25	1931	1
Imagen 25	6	6	3	3	3	0.25	0.50	0.50	1886	1
Imagen 26	8	6	2	6	4	0.13	0.25	0.33	1886	1
Imagen 27	17	5	0	17	5	0.00	0.00	0.00	1931	1
Imagen 28	29	5	2	27	3	0.03	0.07	0.40	1885	1
Imagen 29	9	6	1	8	5	0.06	0.11	0.17	585	1
Imagen 30	8	6	3	5	3	0.19	0.38	0.50	1235	1
Imagen 31	5	3	1	4	2	0.10	0.20	0.33	866	0

Imagen 32	16	4	2	14	2	0.06	0.13	0.50	910	0
Imagen 33	12	4	4	8	0	0.17	0.33	1.00	1989	2
Imagen 34	0	4	0	0	4	0.00	0.00	0.00	1205	1
Imagen 35	19	10	3	16	7	0.08	0.16	0.30	1839	1
Imagen 36	6	4	1	5	3	0.08	0.17	0.25	1205	1
Imagen 37	4	4	0	4	4	0.00	0.00	0.00	528	0
Imagen 38	16	8	5	11	3	0.16	0.31	0.63	1163	0
Imagen 39	19	9	3	16	6	0.08	0.16	0.33	1185	1
Imagen 40	16	3	2	14	1	0.06	0.13	0.67	1779	1
Imagen 41	12	9	0	12	9	0.00	0.00	0.00	1459	1
Imagen 42	12	4	2	10	2	0.08	0.17	0.50	245	1
Imagen 43	14	7	2	12	5	0.07	0.14	0.29	1459	1
Imagen 44	16	4	2	14	2	0.06	0.13	0.50	1733	0

Nota: (Elaboración Propia)

Tabla 16

Valores obtenidos en los criterios de evaluación al aplicar la técnica de detección de esquinas Pixel a Pixel

	TÉCNICA PIXEL A PIXEL									
	Esquinas Detectadas	Esquinas Reales	VP	FP	FN	Exactitud	Precisión	Recall	Consumo Mem. Ram (kb)	Tiempo de Respuesta (ms)
Imagen 01	3	4	3	0	1	0.75	1.00	0.75	743	2
Imagen 02	5	4	4	1	0	0.80	0.80	1.00	602	0
Imagen 03	3	4	3	0	1	0.75	1.00	0.75	619	1
Imagen 04	7	6	5	2	1	0.63	0.71	0.83	1557	1
Imagen 05	5	5	5	0	0	1.00	1.00	1.00	1185	1
Imagen 06	7	6	6	1	0	0.86	0.86	1.00	1204	1
Imagen 07	3	6	3	0	3	0.50	1.00	0.50	1191	1
Imagen 08	4	4	4	0	0	1.00	1.00	1.00	1145	1
Imagen 09	4	4	4	0	0	1.00	1.00	1.00	1037	0
Imagen 10	5	6	5	0	1	0.83	1.00	0.83	180	0
Imagen 11	5	6	5	0	1	0.83	1.00	0.83	180	0
Imagen 12	4	4	4	0	0	1.00	1.00	1.00	1119	0
Imagen 13	6	5	5	1	0	0.83	0.83	1.00	178	1
Imagen 14	6	4	0	6	4	0.00	0.00	0.00	178	1
Imagen 15	6	5	5	1	0	0.83	0.83	1.00	1042	1
Imagen 16	2	5	2	0	3	0.40	1.00	0.40	1029	1
Imagen 17	5	4	4	1	0	0.80	0.80	1.00	805	1

Imagen 18	3	4	3	0	1	0.75	1.00	0.75	199	1
Imagen 19	6	4	4	2	0	0.67	0.67	1.00	835	0
Imagen 20	4	3	3	1	0	0.75	0.75	1.00	866	1
Imagen 21	5	6	5	0	1	0.83	1.00	0.83	200	1
Imagen 22	2	5	2	0	3	0.40	1.00	0.40	842	1
Imagen 23	3	3	3	0	0	1.00	1.00	1.00	866	1
Imagen 24	6	4	4	2	0	0.67	0.67	1.00	854	1
Imagen 25	4	6	4	0	2	0.67	1.00	0.67	155	0
Imagen 26	7	6	6	1	0	0.86	0.86	1.00	194	1
Imagen 27	5	5	5	0	0	1.00	1.00	1.00	1077	0
Imagen 28	5	5	5	0	0	1.00	1.00	1.00	1055	0
Imagen 29	4	6	4	0	2	0.67	1.00	0.67	1287	0
Imagen 30	5	6	5	0	1	0.83	1.00	0.83	1225	1
Imagen 31	3	3	3	0	0	1.00	1.00	1.00	1245	0
Imagen 32	4	4	4	0	0	1.00	1.00	1.00	1494	0
Imagen 33	0	4	0	0	4	0.00	0.00	0.00	341	0
Imagen 34	3	4	3	0	1	0.75	1.00	0.75	1832	0
Imagen 35	10	10	10	0	0	1.00	1.00	1.00	2286	1
Imagen 36	4	4	4	0	0	1.00	1.00	1.00	530	0
Imagen 37	3	4	3	0	1	0.75	1.00	0.75	2797	1
Imagen 38	7	8	7	0	1	0.88	1.00	0.88	0	0
Imagen 39	8	9	8	0	1	0.89	1.00	0.89	0	0

Imagen 40	3	3	3	0	0	1.00	1.00	1.00	1421	1
Imagen 41	6	9	6	0	3	0.67	1.00	0.67	0	0
Imagen 42	4	4	4	0	0	1.00	1.00	1.00	753	1
Imagen 43	7	7	7	0	0	1.00	1.00	1.00	1674	0
Imagen 44	3	4	3	0	1	0.75	1.00	0.75	1674	0

Nota: (Elaboración Propia)