TECNOLÓGICO NACIONAL DE MÉXICO

Instituto Tecnológico Superior de Teziutlán

IMPLEMENTACION DE ESTRATEGIAS DE CONTROL VISUAL PARA MARCAR ENCUARTE TRACERO

TESIS QUE PRESENTA:

Alfredo Carrasco Aráoz

Como requisito parcial para obtener el título de:

MAESTRO EN SISTEMAS COMPUTACIONALES

HOJA DE FIRMAS

La presente tesis titulada: "**IMPLEMENTACION DE ESTRATEGIAS DE CONTROL VISUAL PARA MARCAR ENCUARTE TRACERO"** fue realizada bajo la dirección del comité de asesores indicado, ha sido aprobada por el mismo y aceptada como requisito

parcial para obtener el título de:

MAESTRO EN SISTEMAS COMPUTACIONALES

DIRECTOR:

DR. JÓSE GUILLERMO CEBADA REYES _____

1er. CO-DIRECTOR:

M. CRISTINA JOAQUÍN SALAS _____

2o. CO-DIRECTOR

M. JACOBO ROBLES CALDERÓN _____

AGRADECIMIENTOS

Agradezco el apoyo de mi familia mi esposa Rosa María Mata Pérez y mi hija Mariajose Estefanía Carrasco Mata que siempre me impulsaron para continuar con mis estudios.

Agradezco especialmente a mi director de tesis el Dr. José Guillermo Cebada Reyes por guiarme en el desarrollo de el presente trabajo.

A los docentes que me dieron catedra por su dedicación al compartir sus conocimientos a lo largo de la maestría.

Al maestro Julio Víctor Galindo Rojas jefe del departamento de posgrado e investigación, por el apoyo brindado durante el periodo en que desarrolle mis estudios de posgrado

DEDICATORIAS

Dedico el presente trabajo de tesis a mi esposa Rosa María Mata Pérez quien ha sido una compañera de vida extraordinaria quien siempre ha sabido impulsarme para avanzar juntos en la vida, buscando siempre el bien estar de la familia le agradezco haberse cursado en mi camino y compartir tanto los momentos agradables y sobre todo por estar ahí en los momentos difíciles, hombro a hombro luchando por salir adelante.

A mi hija Mariajose Estefanía carrasco Mata quien es el motor de mi vida el porqué de mi día a día, la que con sus sueños y alegrías llenan de felicidad mi vida.

INDICE GENERAL

Índice de Figuras

Capítulo 1

Generalidades del proyecto

Introducción

En Teziutlán Puebla y sus alrededores la principal industria es la de la confección de prendas de vestir, dentro del ramo existen empresas que se dedican a elaborar pantalón de mezclilla de exportación, para estas compañías la calidad es un rubro de vital importancia, por otro lado, existen tareas sencillas que son realizadas por personas y que son factibles de automatizar utilizando técnicas de visión artificial complementándolas con técnicas de control de motores y control electroneumático.

En el proceso de ensamble del pantalón de mezclilla se realiza la operación de marcado de bolsa trasera (nombre que recibe la tarea en la empresa donde se desarrolló el trabajo) esta operación se realiza de forma manual, y es precisamente esta condición motiva desarrollar un sistema para la automatización de este proceso. La problemática consiste en generar un algoritmo computacional que utilice visión artificial para generar las coordenadas donde se ubicaran las marcas de apoyo para pegar la bolsa trasera del pantalón. Para realizar el análisis de la imagen se hace uso del algoritmo de localización de esquinas de Harris, utilizado para detectar los puntos de interés de la imagen, a fin de recabar la información necesaria para generar localizar las coordenadas

de la posición de las marcas y posteriormente generar el código G necesario para que el sistema electromecánico realice el trazado de las marcas.

El objetivo de este trabajo es el desarrollar un algoritmo de computación, que utilice técnicas de visión artificial y con apoyo de robot cartesiano de dos ejes realice marcas que sirvan de guía para el pegado de la bolsa al encuarte trasero del pantalón de mezclilla de forma automatizada.

Para alcanzar el objetivo general se desarrolla los siguientes objetivos particulares.

- Desarrollar un algoritmo de visión artificial, para procesar la imagen y determinar las coordenadas donde se localizarán las marcas de referencia,
- Construir un prototipo electromecánico, para lograr la automatización,
- Realizar, así como el circuito electrónico que permita la implementación de del sistema de visión, para así poder obtener la base de imágenes en tiempo real.

El trabajo se organiza en cuatro secciones, en la primera se expone la problemática, la justificación del trabajo, la hipótesis, así como los alcances y limitaciones como último tema el estado del arte donde se muestran los trabajos que se han realizado trabajos los cuales destacan la importancia del uso de la visión artificial. El marco teórico y los fundamentos se presentan en el apartado dos en el tercer capítulo se describe cómo se desarrolla el prototipo, como se realiza el proceso de análisis para detectar las esquinas, que se realiza con la

información obtenida del detector, las pruebas, así como los resultados y conclusiones.

1.1 Planteamiento del problema

En el desarrollo de las actividades productivas de las empresas maquiladoras existen procesos semi automatizados, como es la operación de pegado de traba; el cual consiste en que un operario coloca la cintura del pantalón en la posición correcta indicada por unas pequeñas marcas, la maquina corta el tamaño adecuado de traba lo dobla y la cose por los dos extremos; la operación para pegar la etiqueta en la pretina de la prenda se utiliza una maquina robotizada en ella se programa la forma de la etiqueta y el número de puntadas que ha de coser, un operador ubica los pantalones en el lugar donde se cosera la etiqueta y presiona un pedal y la maquina cose la etiqueta.

Sin embargo, para poder realizar el cambio de modelo de prenda o el tamaño de la etiqueta, se requiere que el personal de mantenimiento, deba realizar complicados ajustes, lo cual reduce la precisión de la operación.

En el proceso de ensamble del pantalón de mezclilla, existen tareas no automatizadas como es la de unir la bolsa trasera con el encuarte, está tarea consiste en dos etapas la primera consiste en marcar el encuarte por una persona y la segunda coser la bolsa por otro operario.

En la etapa de marcado un operario realiza unas marcas en forma de ángulo recto, (cabe mencionar que la plantillas cambia de dimensiones de acuerdo a la

talla del pantalón) que indicarán el lugar donde se pegará la bolsa. Como lo muestra la figura1, donde se puede observar a una persona haciendo las marcas para ubicar la bolsa, apoyándose en una plantilla metálica esta se utiliza para que todas las partes queden el en lugar adecuado

Figura 1 Operación de marcado de la bolsa trasera [Fuente Propia 2020].

En entrevista realizada con la licenciada Blanca Sánchez Campos encargada del departamento de producción de la empresa donde se realiza el trabajo, indico que con el transcurrir de la jornada laboral el cansancio merma la productividad del operario y han detectado que el personal dedicado a esta actividad ocupa hasta el 30% del tiempo productivo a otra actividad [1].

Por la naturaleza de las operaciones de la confección las partes a marcar cambian en tamaño, modelo y por ende son derechas e izquierdas, realizar una máquina para realizar el marcado de la prenda es un proceso complicado

tomando en cuenta, las técnicas de control con las que dispone el personal de la empresa CONFETEX S.A.

Para lograr la automatización del marcado del encuarte se implementará el uso de técnicas de visón artificial. Un algoritmo computacional que tendrá la capacidad de reconocer el encuarte trasero y distinguir si es derecho o izquierdo así mismo generara las coordenadas para guiar el sistema que se encargara de realizar las marcas antes señaladas.

1.2 Justificación

En el proceso de confección de prendas existen tareas simples y monótonas las cuales presentan dificultades para su automatización como, por ejemplo: la primera cuando se cambia de talla o modelo de prenda, hay re ajustar la maquina por personal de mantenimiento; una segunda causa es que se deben tomar decisiones en tiempo real, por citar algunas; para dar solución a la automatización se plantea el diseño de una máquina que automatice la tarea de realizar las marcas en el encuarte trasero.

En el ramo de la confección del pantalón de mezclilla el capital humano que interviene en el proceso de marcado del encuarte necesita de capacitación para que pueda realizar alcanzar un nivel de experiencia adecuado para evitar errores en la marcación que se originan por cansancio, descuido o por la in experiencia

del operario siendo esta ultima la principal causa, este problema causa pérdidas de tiempo y perdida de material.

Para realizar el procesamiento de las imágenes captadas por la cámara se recurre a algoritmos de visión artificial con los que se filtrara la imagen para lograr definir el contorno de la prenda, librándolo del ruido impulsivo generado por las condiciones de la iluminación, acto seguido se aplica el algoritmo de detección de esquinas de Harris y con base en las coordenadas de las esquinas y mediante el empleo de la geometría se obtienen las posiciones donde se harán las marcas, que posteriormente se realiza una transformación de pixeles a milímetros para después generar el código G utilizado para enviar la información al hardware que se encargara de grabar la marca en la tela por medio de un láser de baja potencia. El mecanismo para realizar el grabado consta de un robot cartesiana de dos ejes.

1.3 Hipótesis

Para realizar este proceso se implementará un algoritmo de visual serving, al cual se le enviarán imágenes captadas por cámara situada en la parte superior con orientación hacia el área de trabajo. Cuyo objetivo es identificar eficazmente el lugar donde se harán las marcas de referencia para colocar la bolsa en el encuarte trasero del pantalón de mezclilla; a través de un sistema cartesiano de dos ejes, que tomara la referencia de la posición hallada por el algoritmo de visión en tiempo real.

1.4 Objetivo General

Desarrollar un algoritmo de control por visión artificial que por medio de un robot cartesiano de dos ejes realice las marcas para colocar la bolsa en el encuarte trasero del pantalón de mezclilla de forma automatizada.

1.5 Objetivos Específicos

1. Programar un algoritmo de visión artificial para el marcado de referencia en un pantalón.
2. Realizar el Sistema mecánico que se encargara de automatizar el proceso de marcado.
3. Realizar el circuito electrónico para la implementación del sistema de visión artificial.
4. Obtener la base de imágenes en tiempo real

1.6 Alcances

Con el desarrollo del presente trabajo se pretende desarrollar un algoritmo de control para el posicionamiento del sistema de marcado para la colocación de la bolsa trasera, el sistema contara con la capacidad de distinguir si la pieza a marcar corresponde a la parte derecha o izquierda del pantalón, y deberá

compensar la desviación que pudiera presentar a la hora de ser colocada en el área de marcaje.

1.7 Limitaciones

El desarrollo del algoritmo de control se realizará bajo la filosofía open sourse en específico con el lenguaje de programación Python 3.7 y la librería OpenCV 3.0 para el procesamiento de imagen y control. Para la captura de la imagen se utilizará una cámara web genérica de 640 x 480 pixeles (Resolución VGA) a si como solo se procesaran prendas confeccionadas con tela de mezclilla color añil que es la utilizada para confección del pantalón tipo jeans

1.8 Estado del arte

En la actualidad la exigencia de la industria ha crecido gradualmente en cuanto al nivel de calidad de los productos, es debido a esto que las industrias han decidido modernizarse en cuanto a sus niveles de control de calidad. Debido a que la mano de obra es afectada por diferentes factores (cansancio, temperatura, fatiga ocular por nombrar algunos) su confiabilidad no es óptima, es por eso que se presenta la necesidad de utilizar tecnología que pueda realizar el análisis de los productos finales, esta tecnología se denomina visión artificial y se inició en la década de 1960, desde entonces ha avanzado hasta convertirse en nuestros días en la herramienta más utilizada en el control de calidad final de los productos [2].

La visión artificial o también conocida como visión por computadora trata de simular los procesos visuales del hombre y analizarlos por medio del cerebro al igual que los seres humanos, de tal manera que si los hombres pueden transmitir imágenes tomadas por medio de la vista y analizarlas utilizando pulsos enviados al cerebro, una máquina puede, utilizando una cámara web, captar imágenes y enviarlas a un procesador, listo para analizarlas de tal manera que la máquina pueda examinar el color y la forma de ciertos objetos. Aunque en esencia el funcionamiento parece simple, la tarea de identificar desde la cámara, un objeto específico, con el procesador, para que sea seleccionado, resulta demandante en su programación [3].

Para realizar la clasificación de las características de las imágenes captadas por la visión artificial se recurre a algoritmos de machine learning, disciplina en el ámbito de la Inteligencia Artificial (IA) que crea sistemas que aprenden automáticamente. La máquina que realmente aprende es un algoritmo que revisa datos y predice comportamientos futuros. Algunos de estos algoritmos para realizar la clasificación se encuentran:

1. las redes neuronales artificiales que complementan el uso de visión artificial.
2. Deep learning o aprendizaje profundo, hace uso de las redes neuronales artificiales convolucionales que son una variación del Perceptrón Multicapa y pertenecen a los modelos de las RNA, debido a su arquitectura son usadas comúnmente en técnicas de visión artificial para la clasificación y segmentación de imágenes principalmente [4]–[5].

recopilación de las diferentes técnicas de visión artificial complementadas con redes neuronales artificiales.

En el trabajo de Sebastián Amaya et al [2016] se desarrolla un sistema seleccionador de productos utilizando visión artificial y una cámara web integrada al software libre Python, que trabaja en conjunto con las librerías open CV. El principio de funcionamiento del seleccionador está basado en un posicionador de objetos, previamente configurado por el usuario a través del sistema de colores RGB, ese sistema, permite informar al usuario si la pieza ingresada al sistema es de los colores y la forma deseada [6].

La visión artificial ha tenido alto impacto en los procesos industriales, incluyendo a la industria alimenticia. Un ejemplo de esto es el trabajo realizado por P. Constante, A. Gordón y O. Chang [2016] en donde presenta un sistema de visión artificial capaz de detectar características de las fresas usadas en la industria alimenticia [7].

La detección y clasificación de defectos de tela juega un papel importante en la inspección de productos textiles. Muchos defectos de la tela son muy pequeños e indistinguibles, y solo se pueden detectar al monitorear la variación en la intensidad de la saturación de color. Actualmente, en la industria textil, el proceso de detección de defectos se realiza manualmente utilizando mano de obra calificada. Un sistema automatizado de detección e identificación de defectos mejoraría naturalmente la calidad del producto y daría como resultado una

productividad mejorada para satisfacer las demandas de los clientes y también reducir los costos asociados con la calidad deficiente [8]

Saeidi y colaboradores [2011], describieron un sistema de inspección de tela basado en la visión por computadora implementado en una máquina circular para inspeccionar la tela en construcción. La Detección y clasificación de defectos de tejido de punto fue implementada en línea. Las imágenes capturadas se sometieron a un algoritmo de detección de defectos [9].

Capítulo 2

Metodología y Desarrollo

2.1 Fundamentos teóricos

2.1.1 Marco Teórico

La Visión Artificial (VA) usa técnicas provenientes, del procesamiento de señales, de las Ciencias de la Computación, de la Inteligencia Artificial y de la Ingeniería Mecatrónica cuando se utiliza en el contexto de la Robótica. Por otro lado, las aplicaciones de la VA no dejan de aumentar y cada vez son más las áreas de la actividad humana que incorporan algún aspecto de percepción visual, siendo la robótica quien utiliza con más intensidad y amplitud la gran diversidad de técnicas que ofrece la VA, además de representar uno de sus principales retos [10].

La visión artificial trata de simular los procesos visuales del hombre y analizarlos por medio de un cerebro al igual que los seres humanos, de tal manera que, si los hombres pueden transmitir imágenes tomadas por medio de la vista y analizarlas utilizando pulsos enviados al cerebro, una máquina puede, utilizando una cámara web, captar imágenes y enviarlas a un procesador, listo para analizarlas de tal manera que la máquina pueda examinar el color y la forma de ciertos objetos. Aunque en esencia el funcionamiento parece simple, la tarea de identificar desde la cámara, un objeto específico, con el procesador, para que sea elegido por seleccionador, resulta demandante en su programación [11].

Los sistemas de percepción computacional, como también se conoce a la visión artificial, van más allá de medir o detectar, estos sistemas perciben, es decir descifran o reconocen el mensaje sensorial (Figura 2). La información visual es una proyección bidimensional de objetos tridimensionales y, por tanto, la imagen que capta el ojo humano o una cámara digital tiene infinitas interpretaciones posibles. La percepción es un proceso que se distribuye a lo largo del espacio y del tiempo [12].

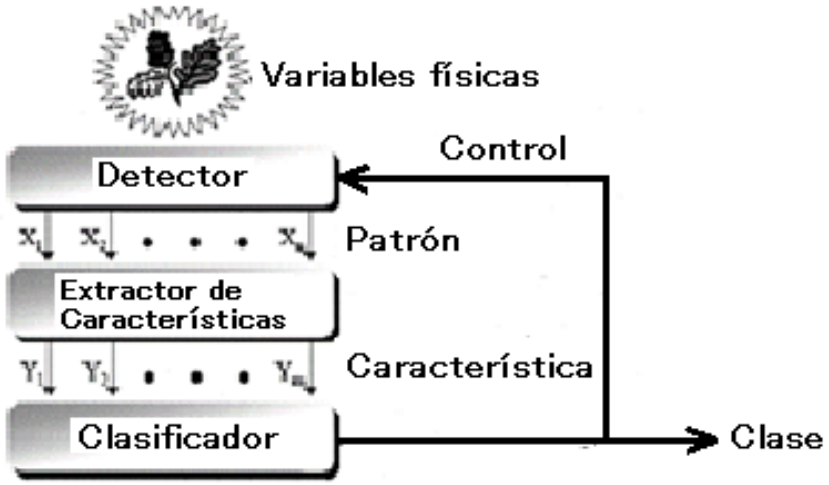


Figura 2 Metodología para el desarrollo detectores [12].

El lenguaje de programación seleccionado para la realización del presente trabajo fue Python el cuales es un lenguaje de programación multiparadigma, esto significa que el usuario puede desarrollar tareas para diversos propósitos, tales como interfaces gráficas, programación orientada a objetos, administración de bases de datos, entre otros. Dicho lenguaje se desarrolla bajo una licencia de código abierto aprobada por OSI, por lo que es de libre uso y distribuible para propósitos académicos e incluso comerciales, dicha licencia es administrada por Python Software Foundation.

El segundo complemento clave es la aplicación de OpenCV la cual es una librería de código abierto utilizada ampliamente para visión artificial la cual fue construida para proporcionar una infraestructura común para aplicaciones de visión por computadora, opera sobre una licencia BSD lo cual hace que sea fácil para las empresas utilizar y modificar el código [13].

Las cámaras de video y de fotografía utilizan el espacio de color RGB, por ser el más extendido para capturar imágenes a color, debido a sus características este formato tiene gran importancia en aplicaciones de visión artificial. El trabajar con el formato con el que se capturan las imágenes reduce los errores de alteraciones de color.

La figura 3 muestra el cubo representativo de del espacio RGB de colores viene definido por la mezcla colores primarios: rojo, verde y azul, los valores de cada uno de los colores. Forman una tupla de tres coordenadas en él se representan el color negro por el vector [r=0, g=0, b=0] y el blanco [r=255, g=255, b=255]. La escala de grises se representa por la diagonal del cubo [14].

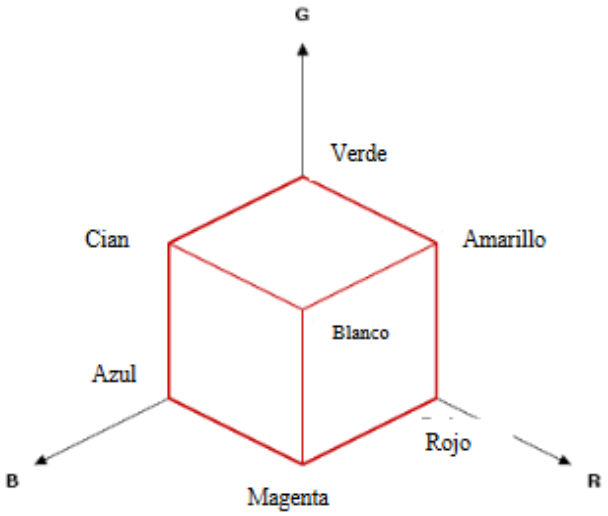


Figura 3 Representación del espacio RGB [14].

El espacio HSV representa uno de los espacios de coordenadas más clásicos e intuitivos existentes en la literatura. Su interpretación geométrica viene determinada por un cono de base quasi-hexagonal, como se muestra en la figura 4. Con esta representación del espacio de color, cada color trabaja con 3 componentes básicas: matiz, saturación y brillo. El matiz, h HSV, hace referencia al valor de cromaticidad o clase de color. La saturación, s HSV, se refiere a las longitudes de onda que se suman a la frecuencia del color, y determina la cantidad de blanco que contiene un color. Contra menos saturado este un color más cantidad de blanco, y contra más saturado este un color menor cantidad de blanco. En definitiva, la saturación representa la pureza e intensidad de un color. Así, la falta de saturación viene dada por la generatriz en la representación del cono HSV. Esa falta de saturación representa la gama de grises desde el blanco hasta el negro. La luminancia, v HSV, se corresponde con la apreciación subjetiva de claridad y oscuridad [14].

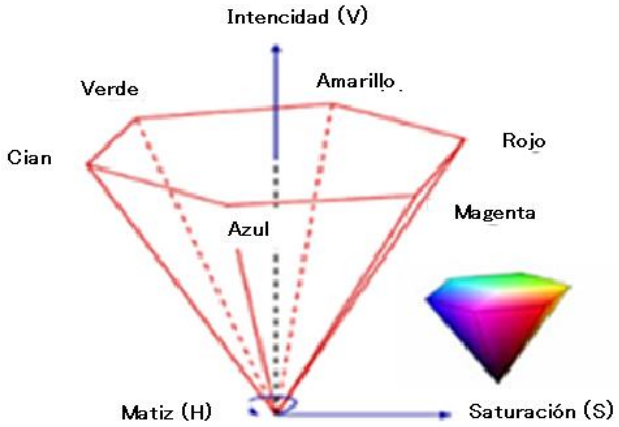


Figura 4 Representación del espacio HSV [14].

Para el análisis de la imagen se prefiere el espacio de color HSV debido a que el canal de la matriz H indica el tipo de color, esta propiedad es útil para segmentar objetos en función de su color, el valor de la saturación S indica que tan vivo es el color y finalmente la cantidad de brillo de la imagen. El empleo de esta forma de representar el color permite reducir el gasto computacional, utilizado en el análisis de la imagen siendo esta la razón de su implementación [24].

El esquema, que se seguirá en el proceso clasificación de imágenes digitales se muestra en la Figura 5 este procedimiento consiste en extraer los candidatos es decir separar de la imagen los contornos de interés, seguido a esto se extraen las características de interés en el caso del proyecto son las esquinas del contorno acto seguido se clasifican los candidatos que cumplen con las especificaciones para tomarse en cuenta, en el cuarto y quinto paso se toman las decisiones y final mente se evalúa el rendimiento[15].

Figura 5 Proceso de clasificación de imágenes [15]

Una vez definido el esquema de recolección de imágenes se utilizará la metodología de desarrollo ***ciclo de vida en V*** también llamada de cuatro etapas diseñada por Alan Divis, los niveles que la forman son:

Nivel 1 en este nivel se está en contacto directamente con el cliente en el inicio y final del proyecto, se definen los requisitos y especificaciones del proyecto a desarrollar.

Nivel 2 se dedica a las características funcionales del sistema propuesto. Puede considerarse el sistema como una caja negra, y caracterizarla únicamente con aquellas funciones que son directa o indirectamente visibles por el usuario final, se traduce en un documento de análisis funcional.

Nivel 3 En esta etapa se definen las características de los componentes de hardware y software que han de conformar el sistema conformando la arquitectura propiamente dicho.

Nivel 4 en esta fase se desarrollan los módulos del programa.

En general las metodologías tienen procesos en común agrupándose como lo muestra el diagrama siguiente [Figura 6], donde se muestra la secuencia siguiente: en la etapa de análisis se definen los requerimientos es decir las necesidades del usuario esto es importante para tener un punto de donde partir para el diseño del algoritmo, una vez que se terminado este trabajo se procede a la programación de los módulos que compondrán el software, seguido a esto se proceden con las pruebas de funcionamiento y estrés cuando se tiene un producto terminado se procede a la documentar el trabajo realizado para entregarlo y ponerlo en funcionamiento, todo desarrollo requiere de mantenimiento es decir ir actualizando el software para que siga brindando el servicio ara el cual fue diseñado.

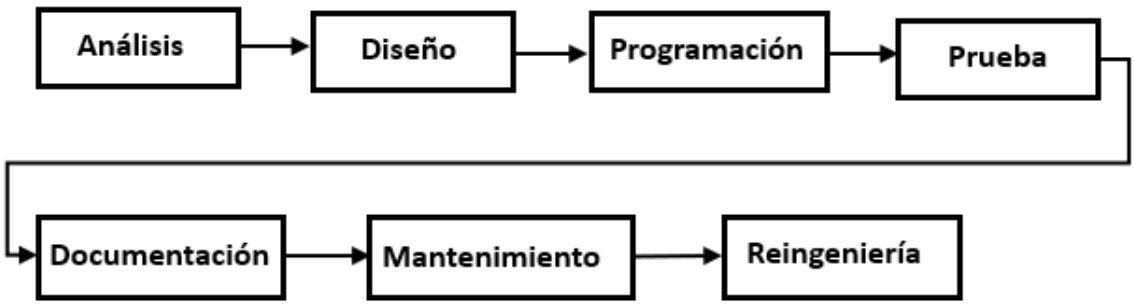


Figura 6 Procesos en común de las diferentes metodologías [16].

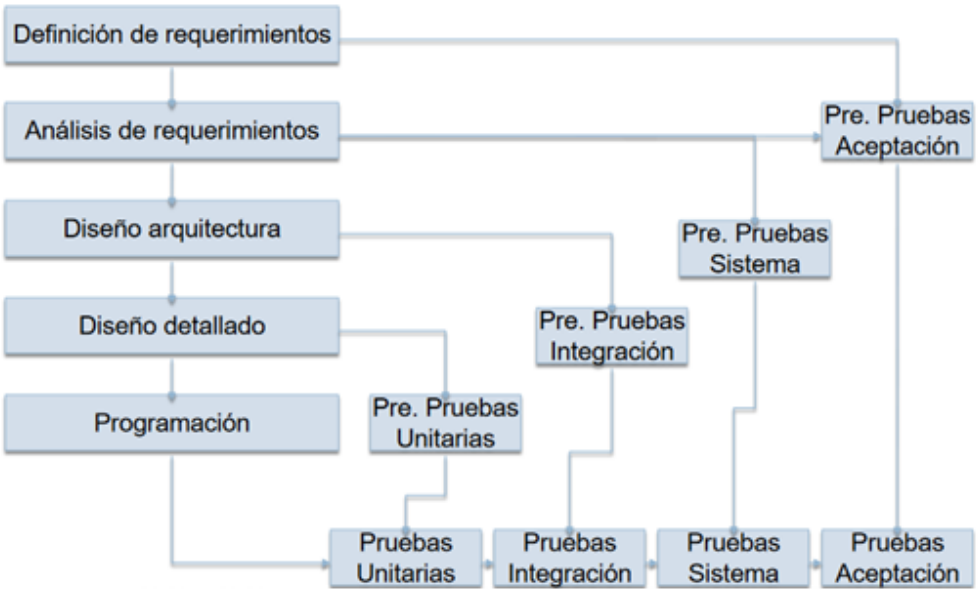


Figura 7 Ciclo de vida en V [16].

La figura 7 muestra gráficamente como se desarrolla la evolución del desarrollo del software, que parte con el conocimiento de los requerimientos tanto funcionales como no funcionales en esta etapa el cliente (departamento de producción de la empresa CONFETEX S.A. de C.V.) define exactamente sus necesidades y sé realiza la etapa de pruebas de aceptación. En la segunda etapa se define la arquitectura con la que contara el sistema es decir lo que el usuario verá en su monitor y al termino se realizan las pruebas de funcionamiento, como siguiente paso consiste en determinar el hardware y se realizan las pruebas de integración con lo que se consigue que el hardware interactúe correctamente

Finalmente, se prosigue a generar los módulos que contendrá el software lo que se traduce en la progresión del sistema en el lenguaje de programación elegido, se termina el siclo con las pruebas unitarias, diseñadas para detectar las posibles fallas que se puedan presentar [16].

En el desarrollo del algoritmo es necesario determinar los requerimientos funcionales y no funcionales para lo cual se utiliza la plantilla propuesta por la IEEE, la tabla 1 se muestran los principales requerimientos funcionales y no funcionales que permitirán la programación, que como indica la ingeniería de software un requisito funcional define una funcionalidad del sistema computacional así como un requisito no funcional es más un concepto de diseño de la interfaz de usuario [16].

Tabla 1 requerimientos funcionales y no funcionales [Fuente propia 2020]

Identificación del requerimiento	**Nombre del requerimiento**	**Descripción del requerimiento**	**Prioridad del requerimiento**
Funcional	Identificación del usuario	Los usuarios deberán identificarse para acceder a cualquier parte del sistema.	Alta
Funcional	Procesamiento de la imagen	La imagen se deberá procesar en tiempo real	Alta
Funcional	Cambio de modelo	Para realizar al cambio de modelo de las prendas a procesar debe de realizarse de manera fácil y rápida	Alta
Funcional	Comunicación	El sistema utilizara el protocolo serial RS232 implementado a través de una conexión USB CDC para la comunicación con la interfaz de comunicación con el Hardware.	Alta
No funcional	Interfaz del sistema	El sistema presentara una interfaz de usuario sencilla para que sea de fácil manejo a los usuarios del sistema	Alta
No funcional	Desempeño	Garantizar el desempeño del sistema informático a los diferentes usuarios. En este sentido la información almacenada o registros realizados podrán ser consultados y actualizados permanente y simultáneamente, sin que se afecte el tiempo de respuesta.	Alta

Con base a los requerimientos del sistema se obtienen los diagramas de casos, de secuencia, parcial de dominio, y de clases que se describen a continuación: El diagrama de casos de uso (Figura 8) muestra de forma gráfica, los requerimientos funcionales del sistema, brinda un punto de partida para el

desarrollo del algoritmo, muestra como interactúan los actores (Usuarios y otros sistemas externos) con el programa.

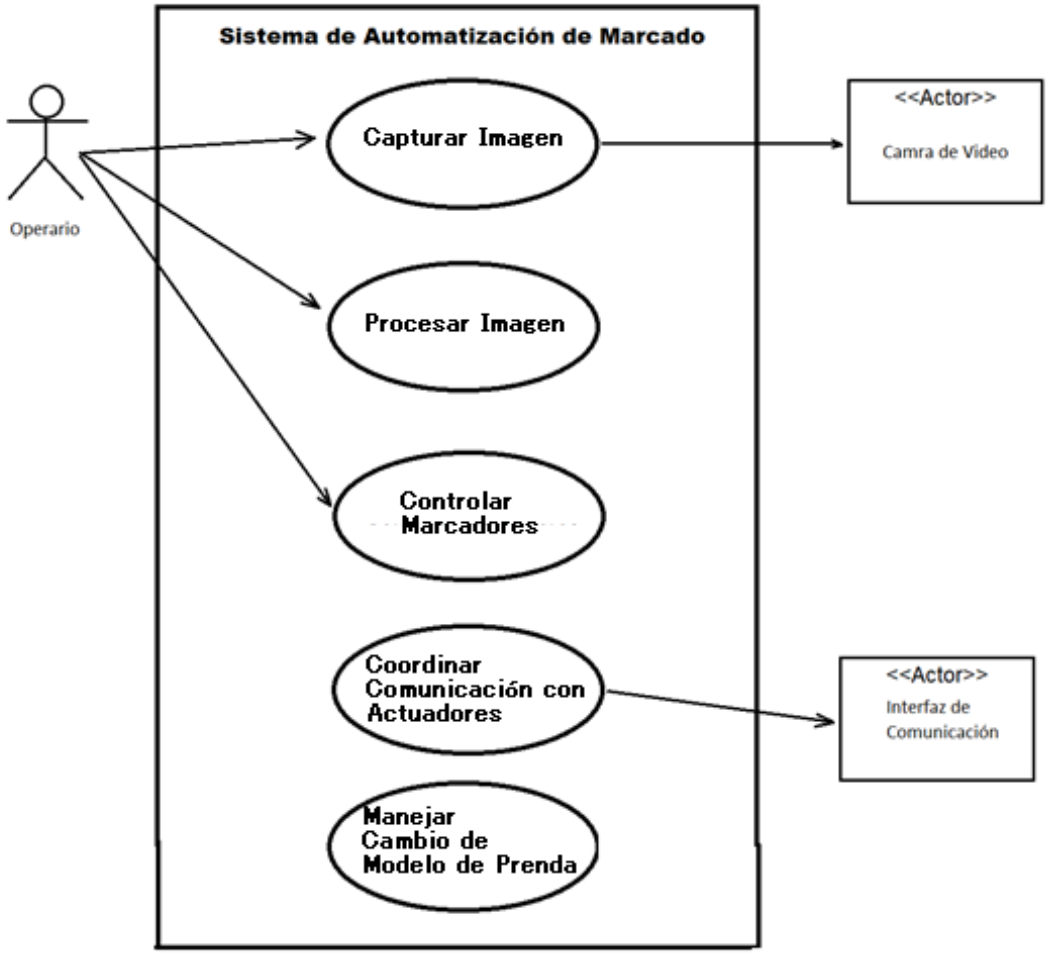


Figura 8 Diagrama de casos de uso [Fuente propia 2020]

En el diagrama muestra las partes que contendrá el programa, la primera consiste en capturar la imagen por una cámara de video, la segunda procesar la imagen aplicando un filtro digital, y localizando las esquinas de interés, la tercera determina las coordenadas para ubicar al sistema de marcado, la cuarta realiza lo conducente para establecer la comunicación con los actuadores que realizaran

las marcas y la última etapa es la encargada de realizar lo necesario para cambiar el modelo de la prenda.

Diagrama de secuencia (Figura 9) muestra con mayor detalle cómo se establece el flujo de información en el programa desde que se introduce la pieza a marcar, hasta que sale de la máquina.

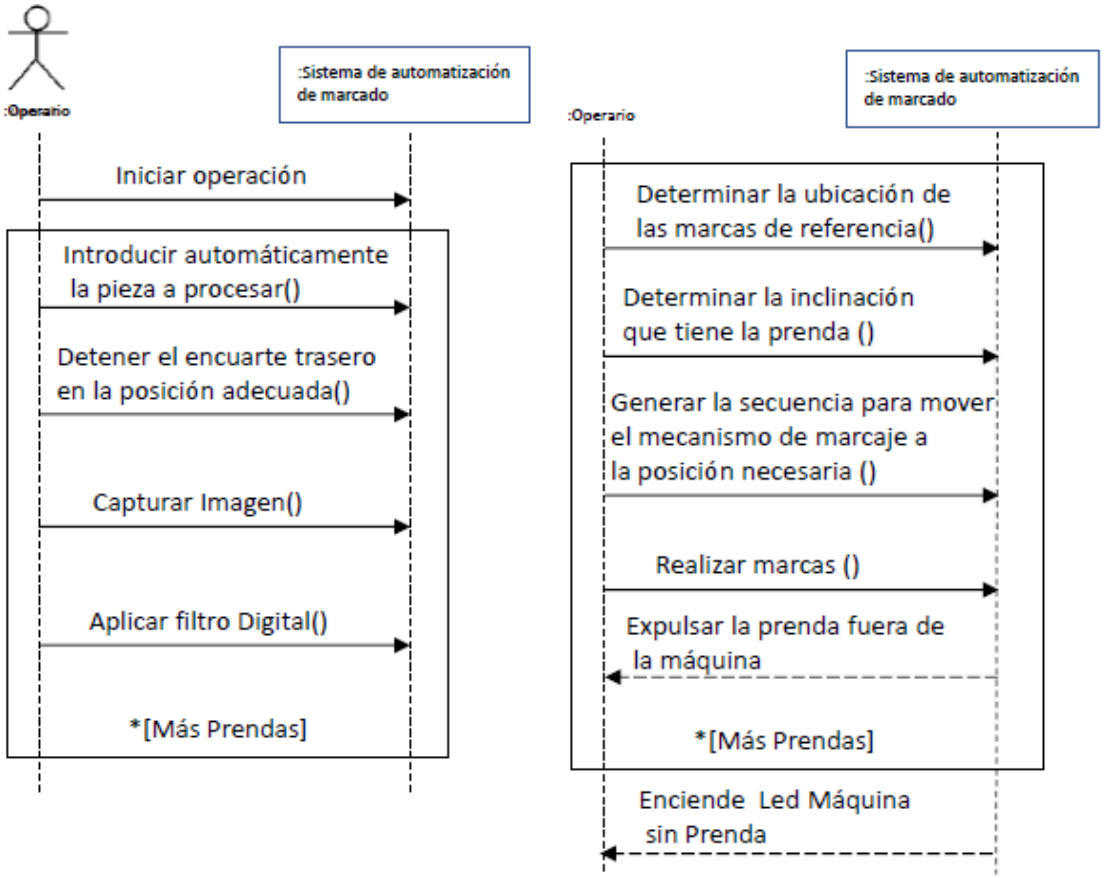


Figura 9 Diagrama de secuencia [Fuente Propia2020]

El proceso comienza cuando el algoritmo manda la señal al conjunto de actuadores dispuestos para tomar la prenda y colocarla en el área donde se marcará, como segunda actividad captura la imagen a la que posteriormente le

aplica un filtro digital en una tercera etapa. En las etapas posteriores se toman las referencias se determina la inclinación de la prenda para compensar la desviación que pudiera existir, se generan las coordenadas a donde se han de mover los actuadores, para realizar las marcas y en la etapa final expulsa la prenda de la máquina.

El diagrama parcial de dominio representa gráficamente de las clases conceptuales del mundo real no son componentes de software, en ellos se contienen los conceptos de la realidad física, estos diagramas expresan el entendimiento del funcionamiento del sistema y contribuye a formar un criterio para el diseño y que permite comprender al sector a que están destinados así como también son punto de partida para el modelado porque permiten suponer el funcionamiento interno del codigo para la programación orientada a objetos.

En el diagrama del sistema (Figura 10) muestra la relación de las clases que permitirán capturar la imagen y generar la información para realizar las marcas, la primera tarea es la de capturar la imagen por medio de la clase imagen donde también se filtra y se aplica el algoritmo de localización de esquinas de Harris, esa información pasa al la siguiente clase llamada coordenadas en ella se localizan las coordenadas donde se encontraran las marcas la información generada pasa a la clase código G donde se forma le código G en cargado de portar la información para que los actuadores realicen las marcas finalmente los datos pasan a la clase comunicación donde se envía las ordenes utilizando el puerto serial a la interfaz de comunicación con los periféricos.

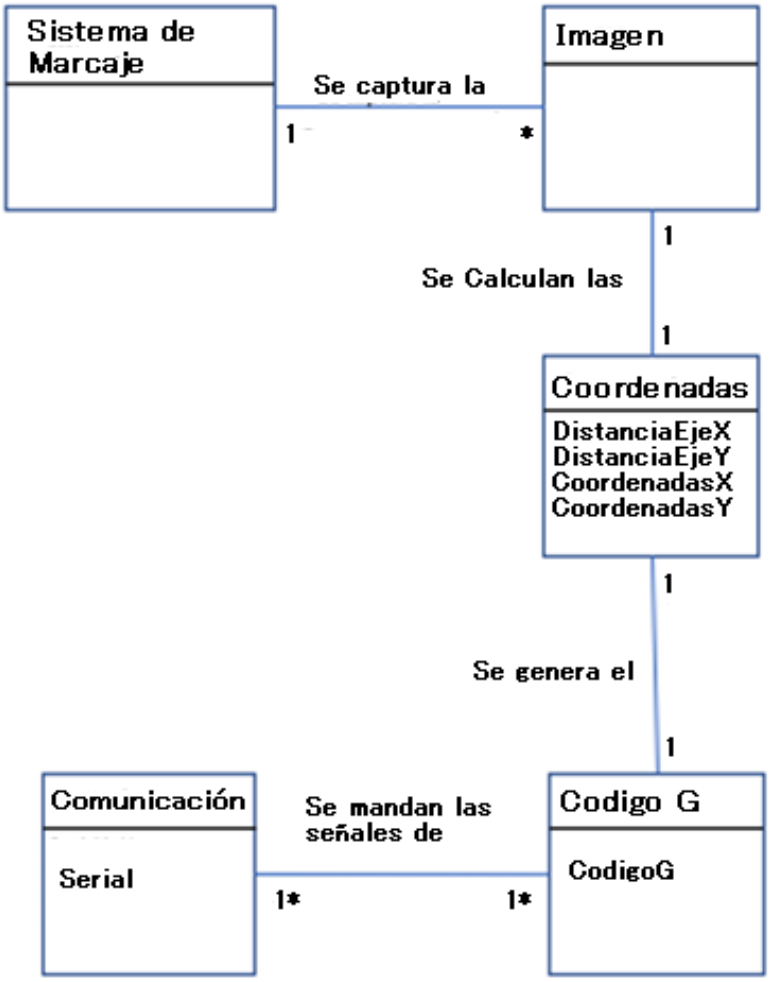


Figura 10 Diagrama parcial de dominio [Fuente propia2020]

El diagrama de clases es muy útil por el motivo de que permite trazar claramente la estructura que tendrá el sistema desde un punto de vista estático, este diagrama no incluye la forma en que se comportan los elementos en ejecución.

La figura 11 muestra el diagrama del sistema que se desarrollará en él se muestra la clase principal que es la encargada de coordinar el flujo del programa de ella se desprenden dos clases secundarias la clase procesamiento de imagen y la

clase control. La clase procesamiento de imagen se encarga de obtener la imagen a procesar una vez que tiene la información se pasa a la subclase filtrado

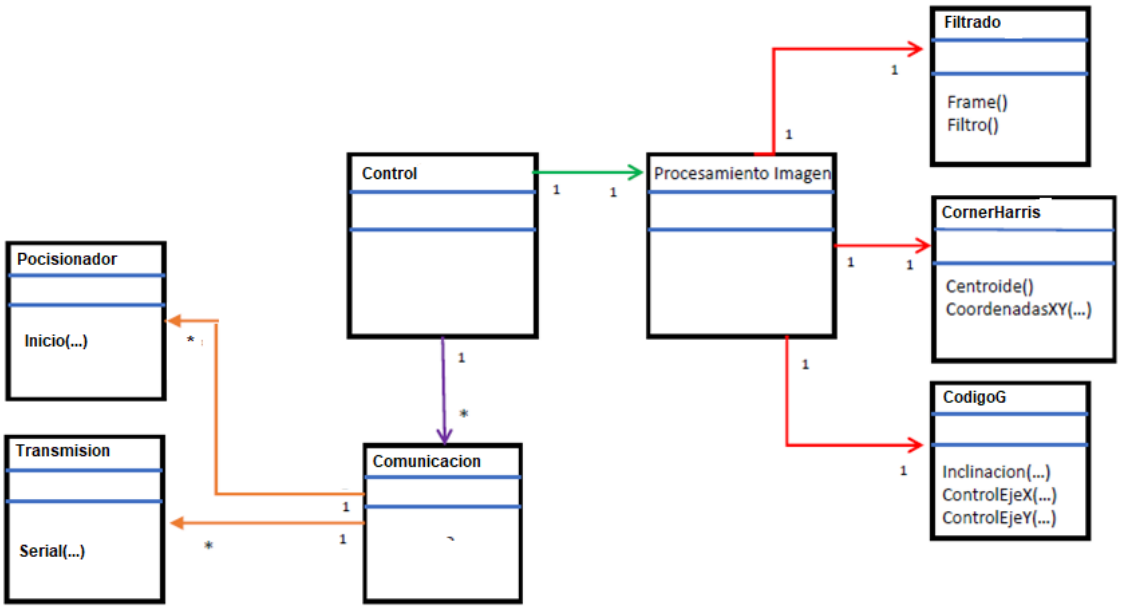


Figura 11 Diagrama de clases [Fuente propia 2020]

donde se aplica el filtro digital gaussiano que devuelve la imagen filtrada la cual se pasa a la subclase cornerHarris en ella se aplica el algoritmo de localización de esquinas de Harris al determinar los centroides de las equinas calcula la posición donde se ubicarán las marcas.

La clase control recibe la localización de las marcas cabe mencionar que las longitudes que recibe están en función de los pixeles de la imagen, por lo que la clase realiza una transformación a milímetros, el resultado de la transformación se manda a la subclase comunicación, que a su vez la manda a la subclase codigoG para general el código G que ha de enviarse de vuelta a la clase control para enviarla la subclase trasmisión que es la encargada de mandar los datos a

la interfaz de comunicación con los actuadores, otra función de la clase comunicación es la de mandar la información a través de la sub clase posicionador para que el sistema de posicionamiento ubique la pieza a procesar en el lugar correcto.

Capítulo 3

Implementación y pruebas

En la actualidad la exigencia de la industria ha crecido gradualmente en cuanto al nivel de calidad de los productos, es debido a esto que las industrias han decidido modernizarse en cuanto a sus niveles de control de calidad. Y sabiendo que debido a que la mano de obra es afectada por diferentes factores (cansancio, temperatura, por citar algunos) su confiabilidad no es óptima, es por eso que se presenta la necesidad de utilizar tecnología que pueda realizar el análisis de los productos finales, esta tecnología se denomina visión artificial y se inició en la década de 1960, desde entonces ha avanzado grandemente hasta convertirse en nuestros días en la herramienta más utilizada en el control de calidad final de los productos [2].

3.1 Desarrollo de prototipo

En el presente apartado se describe el desarrollo de un prototipo a escala y funcional que permitiera realizar las pruebas del algoritmo de control, para realizar las mercas el encuarte, la escala propuesta es de 1:2, esta decisión se tomó a partir del material con el que se disponía.

Las partes principales que forman el prototipo se muestran en la figura 12 La máquina está conformada por una estructura o cuerpo principal formado por una

estructura metálica (120) el cuerpo principal tiene la tarea de soportar las diferentes partes que componen al dispositivo, la base de recepción de la prenda (121) cuenta con una cubierta abatible (122) que tiene como función levantar la prenda para que conserve la altura para que el sistema de posicionamiento puede tomar la t (123) pieza que se toma para colocarla en la posición se marcado, en la parte de la base fijo se encuentran ubicados el sistema de posicionamiento y el robot cartesiano (124) en cargado de marcar la prenda.

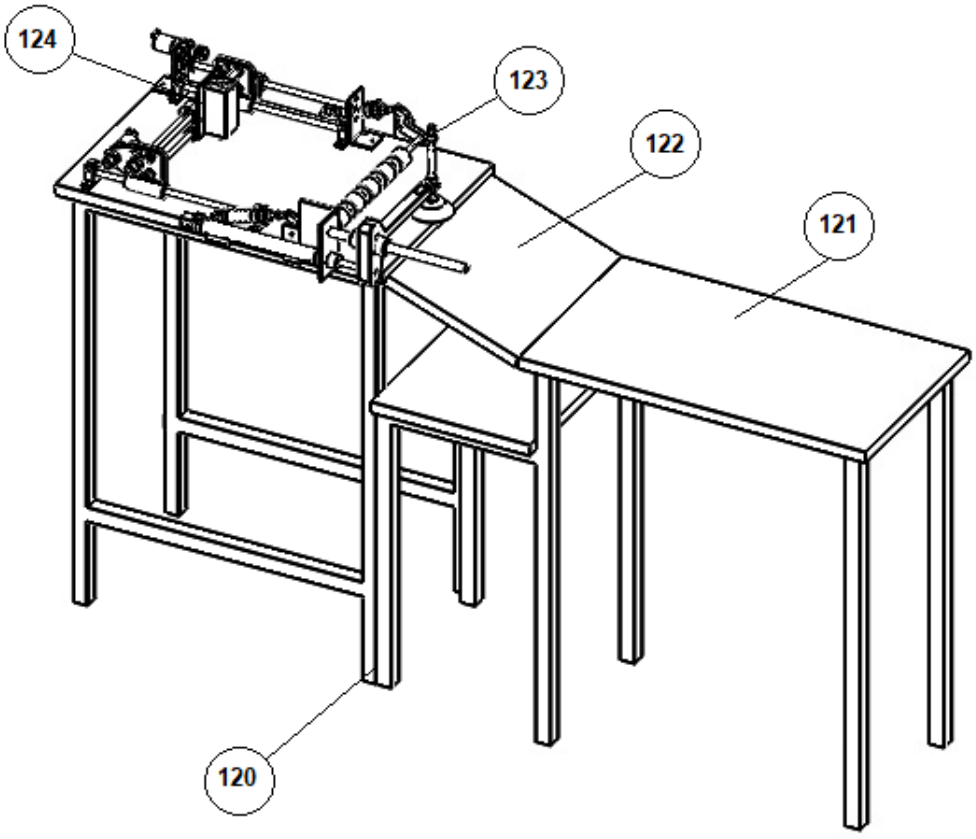


Figura 12 Propuesta de equipo de para el marcado del encuarte trasero [Fuente propia 2020]

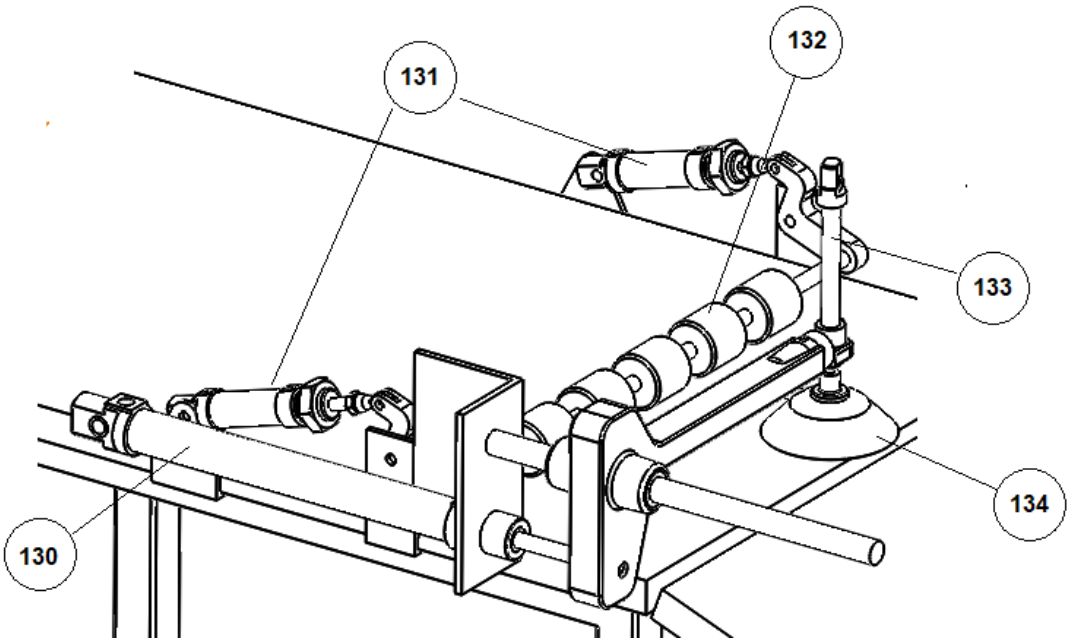


Figura 13 Sistema de posicionamiento del encuarte trasero [Fuente propia 2020]

El sistema para colocar el encuarte trasero (Figura 13) consiste en un cilindro neumático (130) que tiene por función colocar a la ventosa(134), el cilindro neumático (133) se utiliza para bajar la ventosa para que al activarse la válvula generadora de vacío tome la pieza, una vez que el medio de sujeción es regresado a su posición original los actuadores neumáticos (131)bajan los rodillos (132) sujetando la tela, esperan a que el sujetador suelte la pieza para iniciar a desplazar el encuarte a la posición establecida sujetándola hasta que se terminan de realizar las marcas, acto seguido procede a desalojar la prenda, para repetirse nuevamente el ciclo.

Para el marcado se propone el uso de un robot cartesiano de dos grados de libertad (Figura 14), que se compone de los motores a pasos para el eje X (140) y el eje Y (141), que transfieren su movimiento a los husillos del eje X (143) y al del eje Y (145) los ejes corren sobre las barras que los guían (142, 144), para

realizar las marcas que servirán de referencia para pegar la bolsa trasera se utilizará un láser de baja potencia (146).

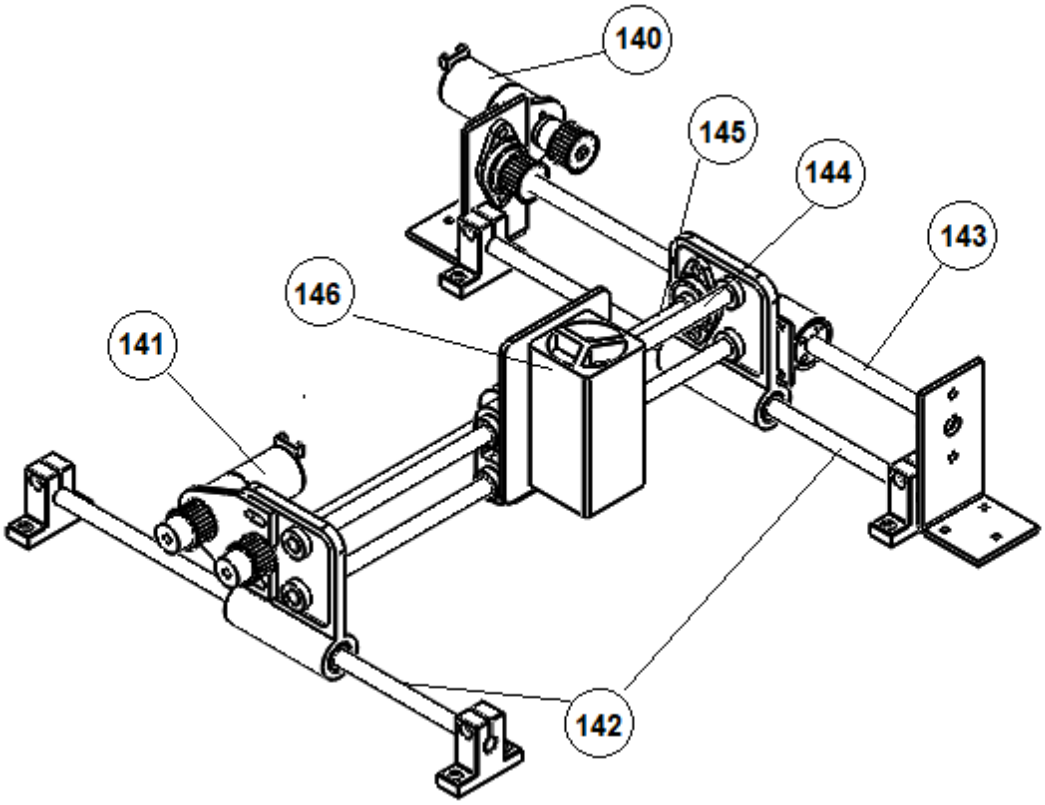
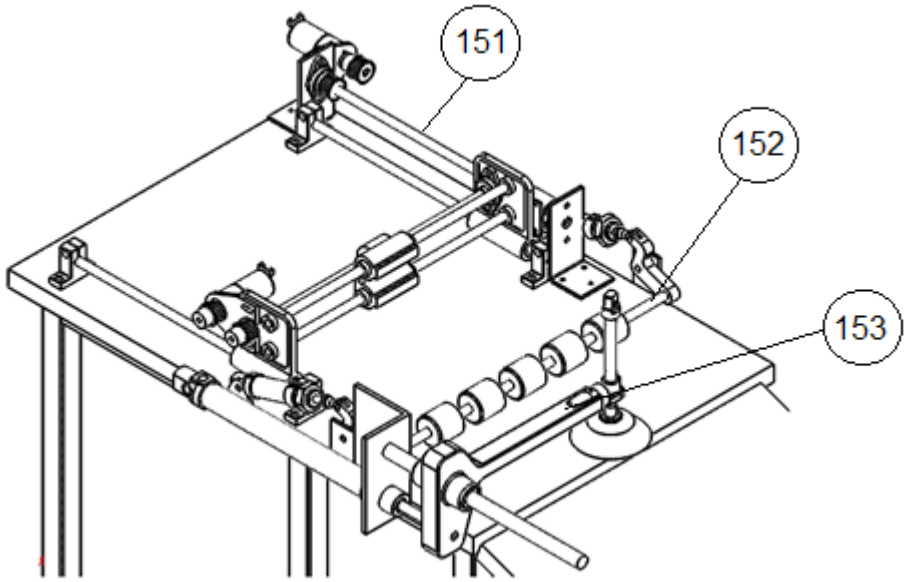
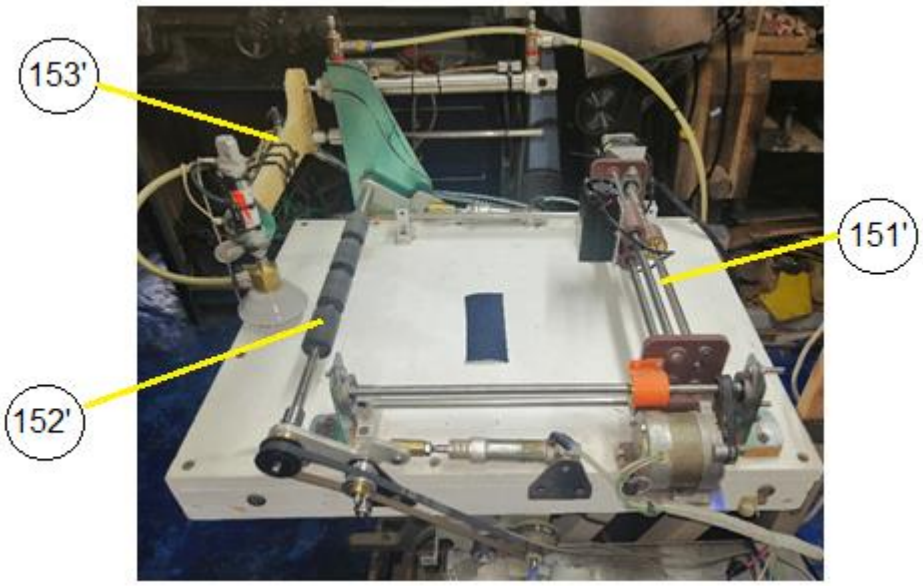


Figura 14 Propuesta de robot cartesiano [Fuente propia 2020]

El montaje del sistema propuesto se observa en la figura 15 donde puede apreciarse en ensamble de los sistemas de posicionado y el robot que realizara el marcado de la prenda.



a)



b)

Figura 15 a) Montaje final del sistema para el marcado en diseño CAD b) Montaje de prototipo de manera física [Fuente propia 2020]

El el montaje físico del sistema donde se puede apreciar el montaje del robot cartesiano (151 – 151’), la ventosa para tomar la parte textil (153 -153’), así como los rodillos utilizados para colocar la prenda en la posición para ser marcada (152 – 152’).

Se puede observar en la figura 16 el montaje de del controlador lógico programable PLC (por sus siglas en ingles), así como los elementos auxiliares. esta parte del del sistema rige el funcionamiento del sistema de posicionamiento de la prenda.

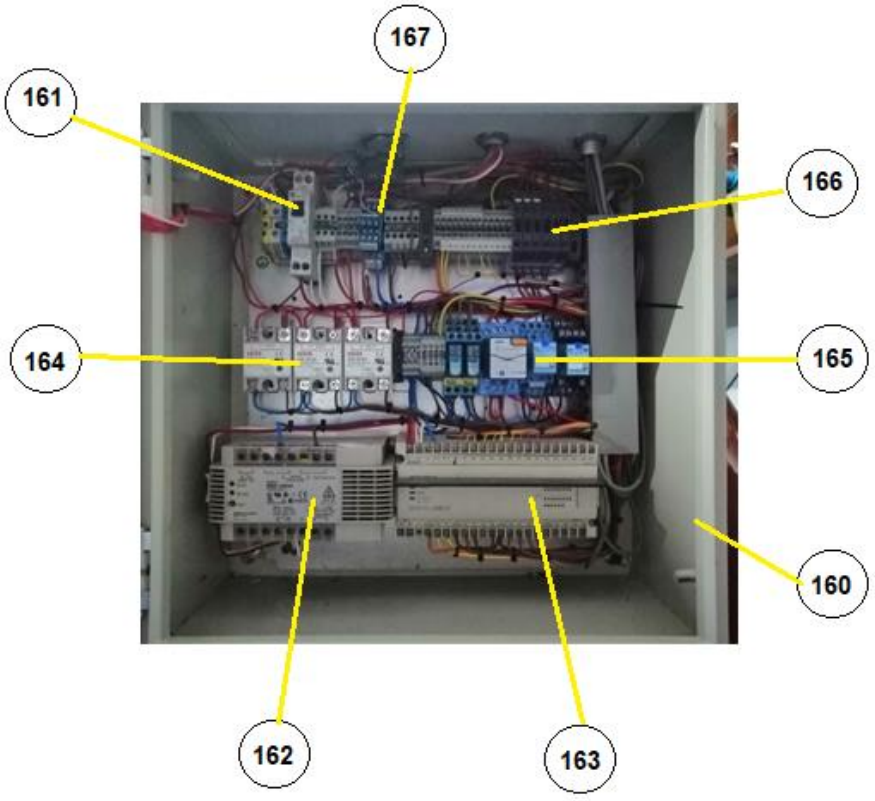


Figura 16 Sistema de control del sistema de posicionamiento [Fuente propia 2020]

El control eléctrico se encuentra alojado en un gabinete eléctrico 500 mm de ancho por 500 mm de alto y 250 mm de fondo. Para la protección eléctrica se utiliza un interruptor termo magnético de 6 Amperes (161), que alimenta a una fuente conmutada (162) con voltaje de alimentación de120 Vac y 24 Vcd y 5 Amperes de salida, esta fuente se utiliza para alimentar las electroválvulas de los cilindros neumáticos que bajan los rodillos que mueven la pieza a procesar, así como los sensores de final de carrera de los cilindros neumáticos. El interruptor también suministra la energía eléctrica para el PLC (163) que se alimenta con una tención que se encuentra en el rango de 90 a 220 Vac.

El controlador lógico programable es quien coordina el funcionamiento del sistema de posicionamiento sus salidas se encuentran conectadas a dos tipos de relevadores loa primeros son de estado sólido (164) estos controlan tres válvulas neumáticas que operan con una alimentación eléctrica de 120 Vac, el segundo grupo de relevadores son electromagnéticos (165), utilizados para alimentar dos electroválvulas con bobinas de 24 Vcd y mandar la señal para la activación del motor que impulsa a los rodillos que colocan la prenda en su lugar. Para dar protección al sistema por fallas eléctricas de las electroválvulas se utilizan clemas portafusibles (166), las entradas de señales tanto de los sensores como las señales provenientes de la computadora encargada de procesar la imagen se reciben en un grupo de clemas destinadas para este propósito (167).

El sistema neumático (Figura 17) está constituido por tres válvulas que operan con corriente alterna y dos válvulas que operan a 24 Vcd. Las electroválvulas de corriente alterna controlan: la primera (170) es la encargada de controlar la

activación del cilindro que mueve a la ventosa la presión de operación de este cilindro se controla con el regulador de presión (174), la segunda electroválvula (171) controla al cilindro encargado de mover el brazo que transporta a el elemento de sujeción y la tercer electroválvula (172) es la que suministra el aire a la válvula generadora de vacío, las dos válvulas de corriente continua (173) activan los cilindros que mueven los rodillos.

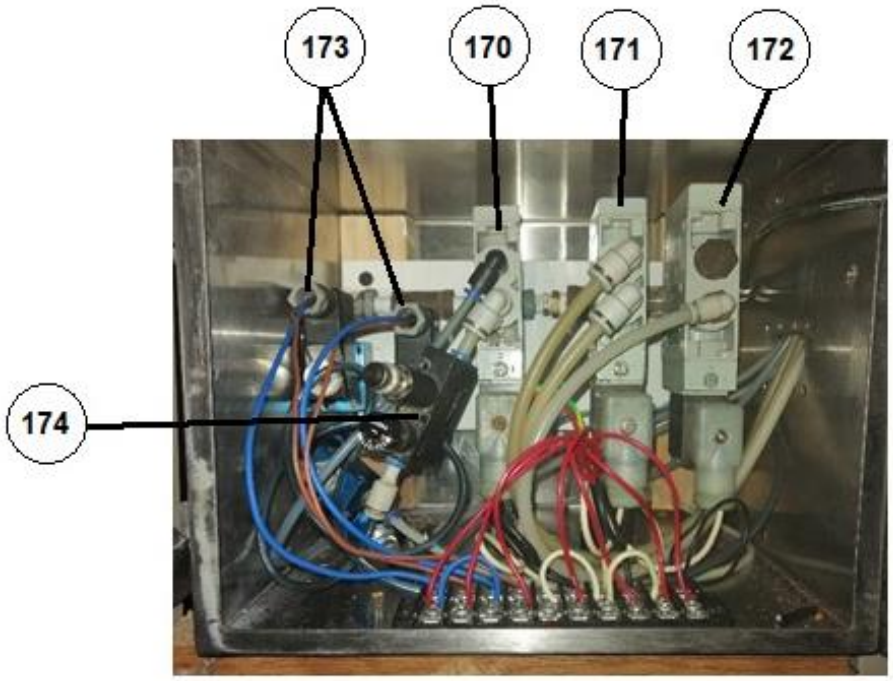


Figura 17 Válvulas neumáticas [Fuente propia 2020]

3.2 Descripción del proceso de detención de esquinas

El programa computacional deberá tener la capacidad para detectar el contorno y las esquinas del encuarte trasero para poder distinguir entre la pieza izquierda y la derecha para con esto localizar las coordenadas reales donde se realizarán las marcas, para obtener la imagen se utiliza una cámara de 0.3 mega pixeles.

El detector de esquinas que se utiliza es el descrito por Chris Harris & Mike Stephens [1988] proponen el uso de un detector combinando, utilizando un detector de esquinas y un detector de contornos, el detector de basa en la matriz de correlación que utiliza el gradiente de la vecindad de un punto p, es decir busca los cambios en la densidad del color gris ya que una esquina es una región con grandes cambios en la densidad del color, [22]

Para determinar el filtro digital a utilizar se realizaron pruebas con una computadora con un procesador de 4 núcleos con 2.1 GHz de velocidad de procesamiento y 8 GB de RAM usando el IDE Pycharm Community, utilizando como interprete Python 3.7, para el análisis de la imagen se utiliza la librería OpenCv versión 3.0.0.

Se desea encontrar un filtro con una ejecución aceptable para que el flujo de la información sea eficaz por lo que para medir el tiempo de ejecución se adhirieron etiquetas con el módulo de tiempo de Python, se obtuvieron 10 tiempos de ejecución y se determinó el promedio de estos.

Para [Blanes 1989] la calidad de una señal transformada (f) con respecto a la señal de origen (g) se puede determinar con una evaluación de error cuadrático medio MSE (por sus siglas en ingles) y la relación de señal de pico a ruido PSNR (por sus siglas en ingles).

Al tener ancho x y altura de la imagen siendo MAX es el valor máximo que puede tomar un píxel, MSE y PSNR se pueden obtener con las expresiones 1 y 2 respectivamente [17].

$$MSE = \frac{1}{(xy)} \sum_{i=1}^{x} \sum_{j=1}^{y} \left(f(i,j) - g(i,j) \right)^2 \quad (1)$$

Donde:

$MSE \;\; Error\ Cuadrático\ Medio$
$X \;\; Ancho\ de\ la\ imagen$
$Y \;\; Alto\ de\ la\ imagen$
$f \;\; Transformada$
$g \;\; Señal\ de\ Origen$

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

Donde:

$PSNR \;\; Relación\ de\ Señal\ de\ Pico\ a\ Ruido$
$Max \;\; Valor\ Maximo\ de\ Pixel$
$MSE \;\; Error\ Cuadrático\ Medio$

Valores pequeños de MSE indican que la correlación ente la señal original y la transformada es mayor debido a que el error es reducido. Para PSNR los valores típicos fluctúan entre 30 y 50 dB correspondiendo valores altos a imágenes de alta calidad [18]

OpenCV ofrece los filtros: filtro bilateral, filtro con núcleo gaussiano (gaussianBlur), filtros de caja (desenfoque) y su equivalente con la caja estándar (boxFilter), filtro de convolución con el núcleo específico (filtro2D) y el filtro de mediana (medianaBlur) [13].

Para evaluar el desempeño de los diferentes filtros se desarrolló un programa en Python y la librería OpenCV que consiste en tomar una imagen (la misma para todos los filtros a probar) aplicar el filtro bajo prueba el resultado se guarda en una variable, posteriormente se calculan MSE y PSNR mediante las expresiones 1 y 2 respectivamente, para conocer el tiempo de ejecución se utilizó el módulo time en Python que permite obtener el tiempo del reloj en de ejecución de la computadora ese instante, por lo que se ejecuta la instrucción time() asignando el valor a una variable inmediatamente antes de ejecutar el filtro y así mismo se vuelve a ejecutar el módulo time inmediatamente después de la ejecución del filtro y solo basta realizar la resta de los tiempos para determinar el tiempo que tardo en ejecutar el segmento de código. Así entonces la tabla 2 muestra los resultados obtenidos y con base a ellos se elige el filtro Gaussiano para el filtrado de la imagen.

Tabla 2. Resultados de la aplicación del filtro [fuente propia 2020]

	Filtros	**PSNR (dB)**	**MSE**	**Tiempo de Ejecución promedio (ms)**
1	Filtro Biilateral	50.61	1.69	56.8
2	Filtro blur	33.96	78.37	21.1
3	BoxFilter	33.96	78.37	20.2
4	GaussianBlur	36.18	46.99	35.8
5	Filter2D	33.96	78.37	89.5
6	MedianBlur	34.29	72.47	168.8

El filtro Gaussiano seleccionado es un filtro con características de filtro pasa-bajos los coeficientes se determinan por los valores de una función de distribución Gaussiana determinada por la siguiente expresión:

$$f_{(x)} = ae^{\frac{-(x-\mu)^2}{2\sigma^2}} \quad (3)$$

Donde:

a $Conastante > 0$
μ $Mediana$
σ^2 $Varianza$

Este filtro es mayormente utilizado para el suavizado de las imágenes con presencia de ruido impulsivo generado por la mala iluminación o por la pobre respuesta de los pixeles en los pixeles de la cámara. El filtrado se realiza recorriendo una máscara o kernel que consiste en una vecindad de 3 x 3 que es la más utilizada (pudiendo existir las grandes siempre en número impar de columnas y renglones), el kernel se caracteriza por asignar un mayor peso al pixel central y a los que se acerquen a este y un menor peso a los que se alejen expresión 4 [19].

$$W = \frac{1}{16} * \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 2 \end{bmatrix} \quad (4)$$

Finalmente, el motivo de aplicar un filtro es la de definir de mejor forma el contorno de la imagen a analizar, reduciendo con esto el ruido indeseado producto de la variación de la iluminación que se pudiera presentar es precisó mencionar que las imágenes que se tomaron para el tratamiento fueron tomadas en un entorno no controlado, se tomaron en las instalaciones de la empresa donde se está desarrollando el proyecto.

Para el proceso de detección de las esquinas se emplea el método de detección corner Harris que es una de las técnicas más usadas para la localización de rasgos puntuales gracias a su robustez ante los cambios de iluminación así mismo el cambio de la escala y el ruido que pudiera tener la imagen [20].

Antes de continuar se debe distinguir entre vértice y esquina, los vértices se forman a lo largo del contorno, sin embargo, las esquinas se forman por la intersección de tres o más superficies. Antes de pasar a la descripción del algoritmo de detención de esquinas es menester establecer algunos conceptos de donde parte el método de Harris.

En la mayoría de los detectores de esquinas se basan en el gradiente de la imagen para comprender el proceso de la detección considérense dos imágenes de gradiente G_x Y G_y tomada de una imagen cualquiera en un punto *p* tome se una vecindad *V* el rededor de *p* y la siguiente matriz *C* que caracteriza la estructura de los niveles de gris de los pixeles dentro de *V* como lo muestra la expresión 5 [21]:

$$C = \begin{bmatrix} \sum_V G_x^2 & \sum_V G_x G_y \\ \sum_V G_x G_y & \sum_V G_y^2 \end{bmatrix} \tag{5}$$

La clave se encuentra en los valores de *C,* así como en su interpretación geométrica como la matriz es simétrica diagonalizada por una rotación de ejes coordenados lo que da como resultado la expresión 6:

$$C = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \quad (6)$$

Donde:

$\lambda_1 \; y \; \lambda_2 \; don \; los \; valores \; propios \; de \; C \; los \; dos \; positivos$

Los valores pueden tener tres casos:

1. V es perfectamente uniforme (contiene los niveles de grises muy parecidos ya que el gradiente es cero dentro de V.
2. V tiene un borde tipo paso que cambia de negro a blanco en este caso se tiene que λ_1>0 y λ_2=0, el valor propio asociado a λ_1 es paralelo a al gradiente.
3. Contiene la esquina de un cuadrado negro con fondo blanco; el valor de los valores propios se encuentra en el intervalo $\lambda_1 \geq \lambda_2 \geq 0$.

Sosa [2013] propone un procedimiento simple para la detención de esquinas donde el procedimiento recibe como entrada una imagen en escala de gris $f(x, y)$, dos umbrales λ_2 y *u* y el tamaño de la ventana (2*N* +1) pixeles para una ventana cuadrada entonces:

1. Calcular los gradientes en x e y sobre la imagen $f(x, y)$
2. Para cada punto *p* de $f(x, y)$
   a) Calcular c para una vecindad *V* de (2N+1) x (2N+1)
   b) Computar λ_2 el valor propio más pequeño de *C*
   c) Si λ_2 > *u*, guardar las coordenadas de *p* en la lista L.
3. Ordenar L en orden descendiente conforme a λ_2

4. Recorrer la lista ordenada de arriba abajo, y para cada punto p borrar todos los puntos de la lista en la vecindad de *p*

La salida de este procedimiento es una lista de esquinas para las cuales $\lambda_2 > u$ y cuyas vecindades no se traslapan. En la mayoría de de los algoritmos *N* y *u* se obtienen por prueba y error la experiencia ha demostrado que los valores comprendidos entre $2 \leq N \leq 10$ da buenos resultados [21].

La matriz hessiana es base del detector de esquinas de Harris que se define como:

Sea f una función real de n valores definida como sigue: $f: \mathbb{R}" \to \mathbb{R}, x \longmapsto f(x)$ donde se supone que todas las segundas derivadas de f existen entonces la matriz hessiana se define como [18]:

$$H(p) = \begin{bmatrix} f_{xx}(p) & f_{xy}(p) \\ f_{xy}(p) & f_{yy}(p) \end{bmatrix} \tag{7}$$

Harris et al [1988] propone un detector combinado de esquinas y bordes basado en la función de autocorrelación local para la interpretación del mundo en tres dimensiones demostrando que funciona sin restricciones, lo que buscaban era la comprensión de las escenas naturales, que contienen carreteras, edificios, árboles, arbustos, por citar algunos [22].

El método que proponen se basa en una matriz de correlación que caracteriza la distribución del gradiente alrededor de un pixel *p* con coordenadas (*x, y*):

$$A(x, y: t) = G(x, y: t) = \begin{bmatrix} L_x^2(x, y: t) & L_x(x, y: t)L_y(x, y: t) \\ L_x(x, y: t)L_y(x, y: t) & L_x^2(x, y: t) \end{bmatrix} \quad (8)$$

Donde:
t $Parametro\ de\ filtrado\ de\ la\ gausiana$
$L_u(p; t)$ $es\ la\ derivada\ gaussiana\ en\ la\ dirección\ de\ u\ en\ la\ imagen\ f\ en\ p$

Si se evalúa t = 0, $A(x, y: t = 0)$ la matriz se transforma en la matriz hessiana sin la aplicación de la gaussiana.

A partir de la ecuación (8) el detector de Harris de esquinas adopta la siguiente forma:

$$KHS(p) = \det(A) - kTr(A)^2 \quad (9)$$

Donde

KSH $Respuesta\ del\ método\ de\ Harris$
A $Matriz\ de\ coorelación$
k $Parametro\ de\ sensibilidad$

El parámetro k se calcula de forma empírica de la forma:

$$K(p) > sup\{K(p_v)|\ p_v \in V, p_v \neq p\} \wedge K(p) > u \quad (10)$$

Donde V es la vecindad de tamaño m x m alrededor de p y u es un umbral previamente definido empíricamente como se comentó en párrafos anteriores [21].

El diagrama de flujo (Figura 18) muestra el proceso que llevo a cabo para filtrar la imagen y detectar las esquinas por el método de Harris, como primer paso se captura la imagen procedente de la cámara, posteriormente se convierte la imagen a escala de grises, se aplica el filtro gaussiano, la imagen filtrada se transforma en una imagen binaria invertida, como último paso se aplica el algoritmo de detección de esquinas de Harris y se marcan las esquinas encontradas.

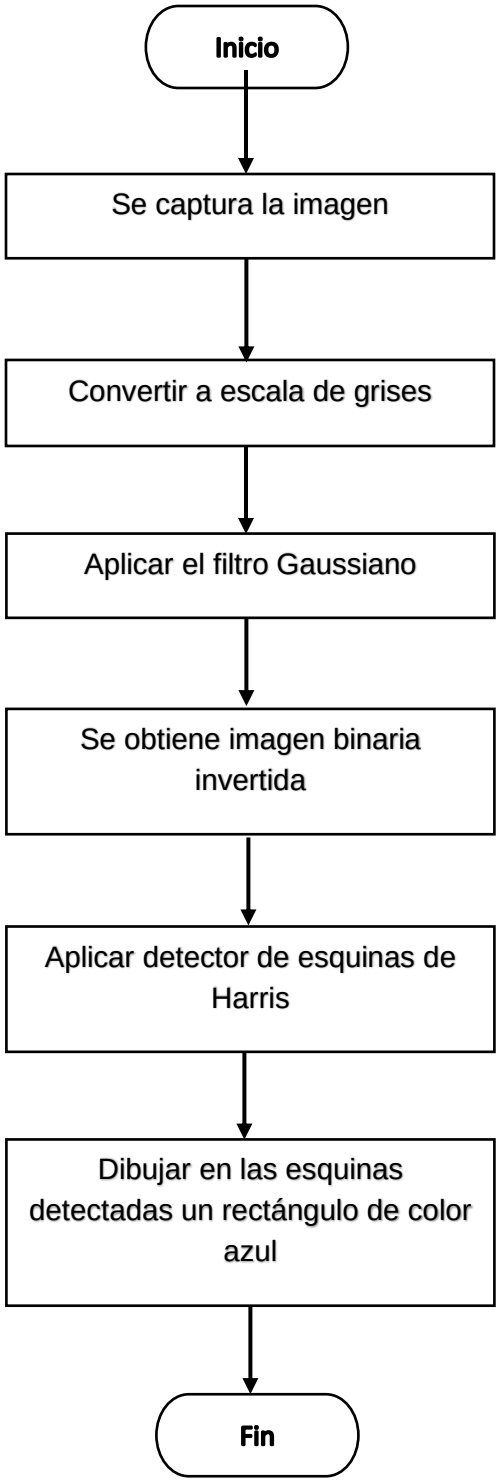


Figura 18 Diagrama de flujo depara detectar esquinas en encuarte [Fuente propia 2020]

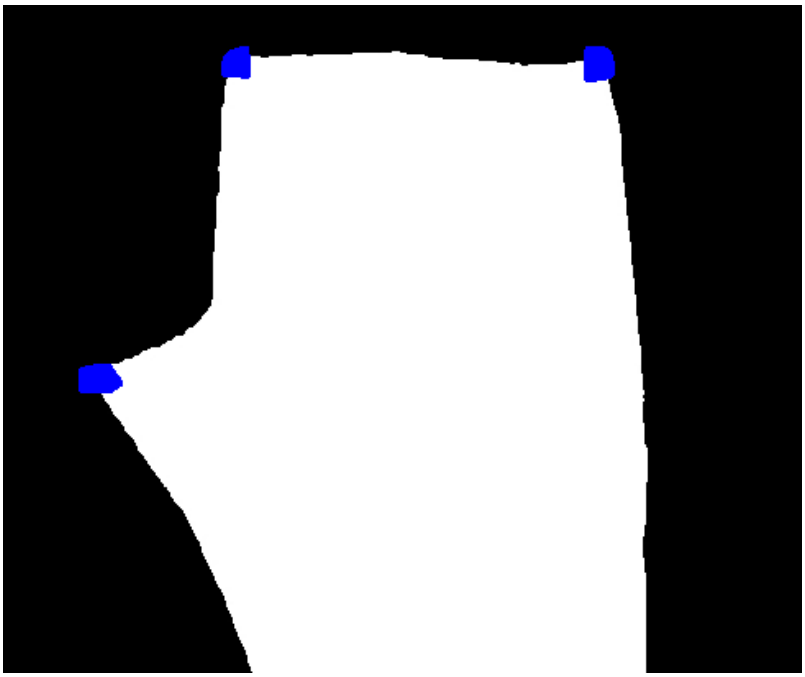


Figura 19 Detección de esquinas con el algoritmo de Harris [Fuente Propia 2020]

La figura 19 muestra la salida de la imagen del algoritmo de Harris en ella se puede observar las esquinas encontradas marcadas en color azul. Sobre la imagen binaria invertida.

La tabla 3 muestra la salida numérica de los valores de las coordenadas de las esquinas encontradas.

Tabla 3 centroides arrojados por el algoritmo de detención de esquinas de Harris

	Coordenada X	Coordenada Y
Esquina superior Izquierda	21	21
Esquina superior derecha	336	34
Esquina inferior derecha	465	289

Los valores de la tabla las coordenadas de los centroides la región del gradiente de la vecindad del punto *p* que marca la ubicación de las esquinas, para una mejor comprensión las esquinas son marcadas con rojo la esquina superior izquierda, con verde la esquina superior derecha y de color azul la esquina inferior derecha, en conjunto con las líneas que servirán de apoyo para hacer las marcas figura 20 se muestra.

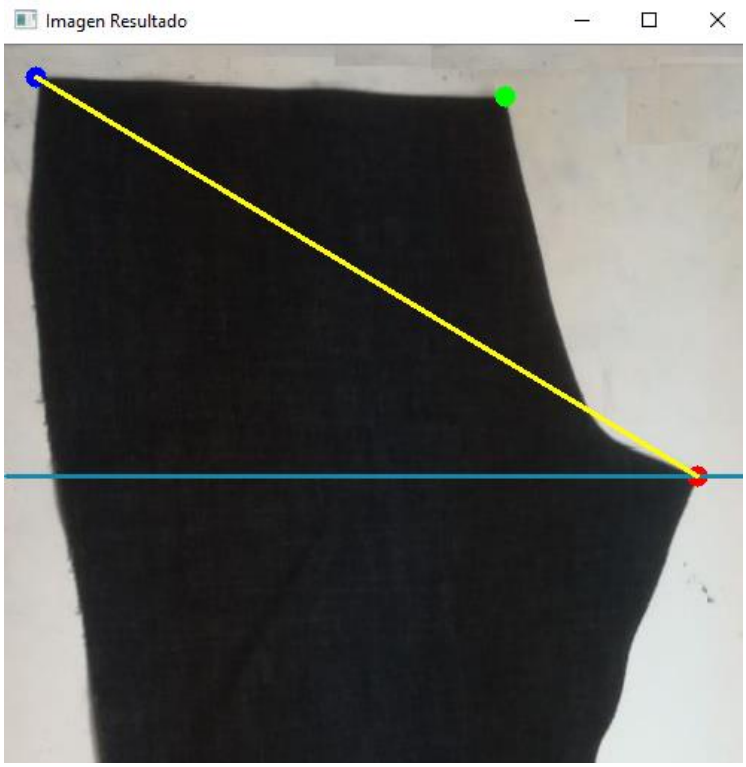


Figura 20 Salida del algoritmo con líneas de apoyo [Fuente propia 2020]

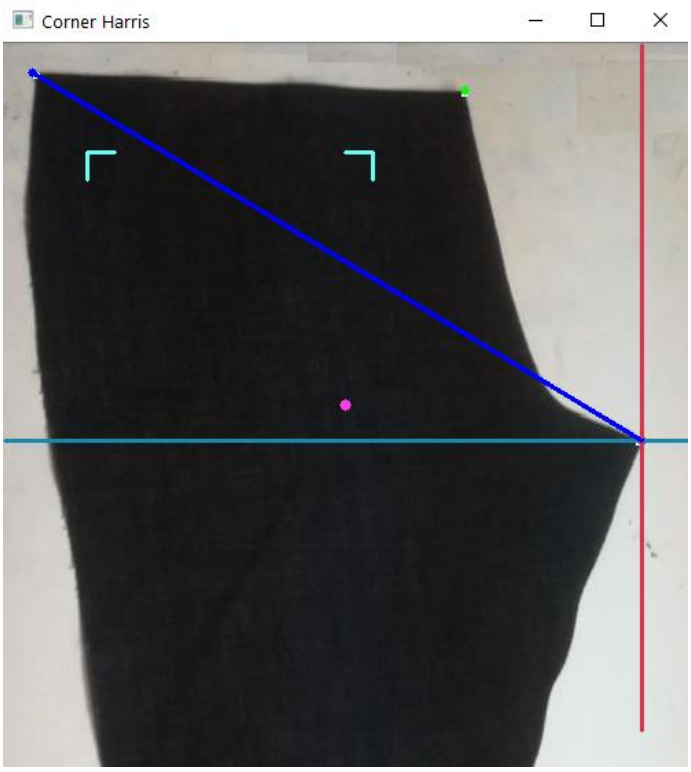


Figura 21 Encuarte con las marcas en la posición de referencia [Fuente propia 2020]

En la figura 21 se muestra las líneas que se tomaran como referencia en el pantalón de mezclilla, estas se obtienen al realizar las proporciones geométricas adecuadas con base a la diagonal debido a que la distancia de la esquina superior izquierda a la esquina del tiro es constante en cada prenda y proporcional entre tallas lo que permite establecer relaciones para ubicar las marcas de referencia.

Para determinar donde se ubicarán las marcas en la prenda que se va a marcar se determina el ángulo de la diagonal principal y se compara con la diagonal de la imagen de referencia (Figura 21).

3.3 Interpretación de las esquinas

Las prendas a marcar no son de una misma medida es necesario determinar un sistema para poder ajustar los valores de la prenda para eso se observó que las piezas cambian de tamaño bajo una proporción como lo muestra la tabla 4 en ella muestra las medidas de cintura y cadera para el pantalón de mezclilla para mujer los datos se extrajeron en la empresa CONFETEX S.A. de C.D, lugar donde se está desarrollando él trabajo

Tabla 4 medidas de cintura y cadera del pantalón de mezclilla para mujer [Fuente propia 2020]

Talla	Cintura cm	Cadera cm
28 (5)	65	85
30 (7)	70	90
32 (9)	75	95
34 (11)	80	100
36 (13)	85	105
38 (15)	90	110

La tabla muestra que el incremento en el tamaño de la cintura conserva una relación lineal con las diferentes tallas, lo que permite poder relacionar la posición con respecto a la distancia que existe en las componentes horizontal y vertical de la línea diagonal entre los puntos 1 y 3 con la posición del punto 1 que corresponde a la cintura. En la figura 22 se muestran los puntos de interés de las citadas marcas donde se colocará la bolsa trasera.

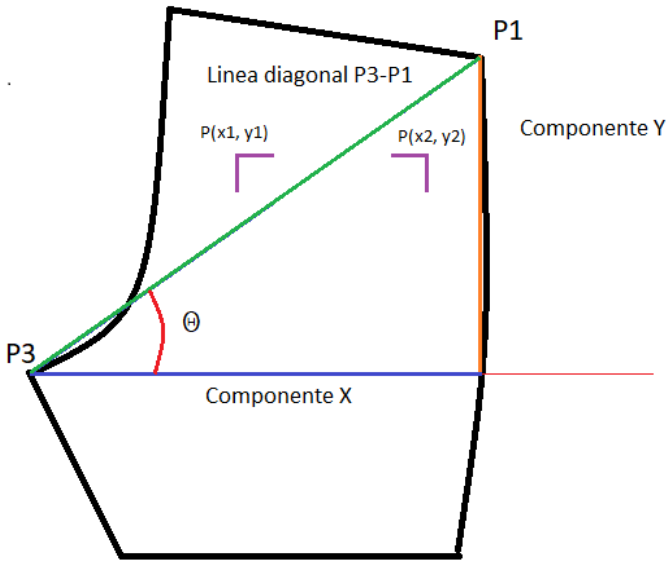


Figura 22 Localización de los puntos de interés para realizar las marcas [Fuente propia 2020]

Para realizar las marcas de manera automática se utilizará un láser (Figura 23) de baja potencia en las fibras superficiales, de la marca RATTM MOTOR de las siguientes características:

- Longitud de onda: 405nm
- Potencia: 500 mW
- Corriente eléctrica: <2A
- Voltaje de entrada: 12 Vdc
- Temperatura de funcionamiento: -10 ~ + 40
- Tamaño: 33*33*75mm

Figura 23 Láser para realizar las marcas montado en el robot cartesiano [Fuente propia 2020]

Para determinar los puntos se utilizó la siguiente relación matemática:

$$px1 = \Big(x3 + \big((x1 - x3) * 0.484848\big)\Big) \tag{11}$$

$$px2 = \Big(x3 + \big((x1 - x3) * 0.863232\big)\Big) \tag{12}$$

$$py1 = py2 = (y1 + ((y3 - y1) * 0.218181 \tag{13}$$

Donde:

$x1$ $Coordenada\ en\ x\ del\ centroide\ del\ punto\ 1$
$x3$ $Coordenada\ en\ x\ del\ centroide\ del\ punto\ 3$
$y1$ $Coordenada\ en\ y\ del\ cntroide\ del\ punto\ 1$
$y3$ $Coordenada\ en\ y\ del\ centroide\ del\ punto\ 3$
$px1$ $Coordenada\ del\ punto\ donde\ se\ colocara\ la\ bolsa\ en\ el\ eje\ X$
$px2$ $Coordenada\ del\ punto\ donde\ se\ colocara\ la\ bolsa\ en\ el\ eje\ X$
$py1$ $Coordenada\ del\ punto\ donde\ se\ colocara\ la\ bolsa\ en\ el\ eje\ Y$
$py2$ $Coordenada\ del\ punto\ donde\ se\ colocara\ la\ bolsa\ en\ el\ eje\ Y$

Cabe mencionar que las coordenadas se encuentran con relación a los pixeles de la imagen que corresponden a los valores absolutos del cuarto cuadrante del plano cartesiano.

La Figura 24 muestra el lugar y forma que tendrán las marcas considerando que la posición es la correcta, para compensar la desviación que pudiera tener la pieza se tomarán las coordenadas polares de los puntos de las esquinas de las marcas mostradas en la imagen, las que se tomaran como referencia.

Figura 24 Muestra como resultado la forma y posición de las marcas que posteriormente se harán con el láser [Fuente propia 2020].

Al colocar una nueva pieza se determina el ángulo de la diagonal que pasa por los puntos 1 y 3 en la posición correcta con el ángulo formado por la diagonal citada en la nueva posición, se extrae la diferencia entre el ángulo original y el formado por la nueva imagen; como el punto 3 se toma como referencia principal permite tomar el siguiente criterio para verificar hacia donde se encuentra la

desviación, si el resultado la diferencias de ángulos es positivo la pieza se encuentra rotada a la derecha, por el contrario si el resultado es negativo la prenda se movió hacia la izquierda.

Tomando como ejemplo la desviación hacia la izquierda la ubicación de las coordenadas de los puntos de interés se puede conseguir si se toma el punto 3 como centro en torno al cual se desplazan las marcas.

Para determinar la ubicación se determina el ángulo que se desplazó la diagonal mediante la expresión:

$$\theta_d = \theta_1 - \theta_2 \tag{14}$$

Donde

θ_d $Diferencia\ de\ ángulos$
θ_1 $Ángulo\ de\ la\ linea\ diagonal\ de\ referencia$
θ_2 $Angulo\ de\ la\ linea\ diagonal\ de\ la\ pieza\ a\ procesar$

Por lo tanto, para determinar las coordenadas nuevas componentes de los puntos se utiliza la conversión de las coordenadas polares a rectangulares mediante las ecuaciones 7 y 8.

$$X = |r| COS\ \theta_p \tag{15}$$

$$Y = |r| SEN\ \theta_p \tag{16}$$

Donde:

X $Componente\ en\ x\ del\ punto\ para\ la\ marca$
Y $Componente\ en\ y\ del\ punto\ para\ la\ marca$
$|r|$ $Valor\ absoluto\ del\ valor\ de\ la\ distancia\ \ del\ P3\ a\ la\ marca\ de\ referencia$
θ_p $Ángulo\ del\ P3\ a\ la\ marca\ \ se\ define\ como: \theta_d + \theta_1$

La figura 25 muestra el desplazamiento de las nuevas marcas para la corrección de la desviación.

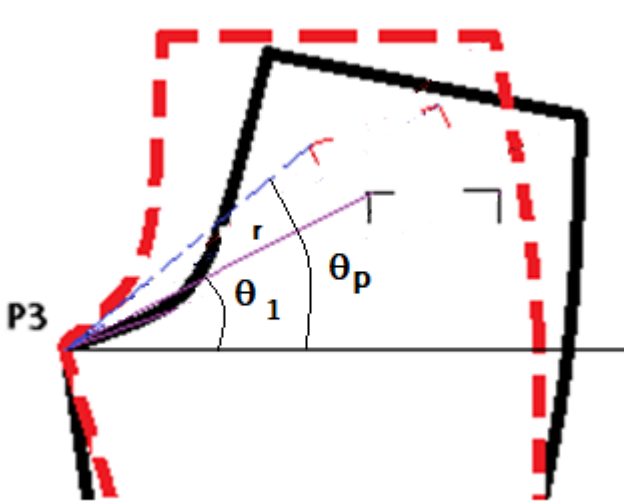


Figura 25 Muestra la posición de las marcas corregidas y los ángulos correspondientes [Fuente Propia 2020]

El proceso de marcado de la prenda requiere que se realice la conversión de las referencias en pixeles (que hasta ahora se han manejado), a milímetros por motivo de que el sistema de locomoción recibe las instrucciones en código Gerber también llamado código G

Para llevar a la posición de trabajo a el láser encargado de realizar el marcado se utiliza la librería GBRL para Arduino, de código abierto la cual fue diseñado

para ser usado en el control de máquinas de control numérico, el firmware utiliza el código g para recibir las coordenadas que posteriormente son transformadas en pulsos eléctricos que serán enviados a los drivers de los motores a pasos de los ejes del robot cartesiano.

El código es un estándar de comandos (estándar ISO 6983) que son utilizados para la programación de maquinados, se pueden crear los programas de forma manual, pero en la actualidad existen softwares que generan automáticamente el listado de instrucciones para un maquinado, grabado o impresión 3D, con el advenimiento de la impresión 3D se ha vuelto popular.

La tabla 3 muestra algunos de los códigos empleados en el anexo 1 se puede consultar el listado completo.

Tabla 3 ejemplos de código G

Código G	**Descripción**
G00:	Posicionamiento rápido (sin maquinar)
G01:	Interpolación lineal (maquinando)
G02:	Interpolación circular (horaria)
G03:	Interpolación circular (antihoraria)

Cada comando debe de cumplir con un formato en específico formado de la siguiente forma:

N## G## F## X## Y## Z## S## T## M##, donde

N##: Número de línea del programa.

G##: Define el movimiento y la función.

X##: Declara la posición horizontal.

Y##: Declara la posición vertical.

Z##: Declara la profundidad.

F##: Velocidad de alimentación.

S##: Velocidad del husillo.

T##: Selección de herramientas.

M##: Funciones diversas, tales como encender y apagar algo, como el refrigerante, movimiento de indexación, bloqueo de ejes entre otros.

Así por ejemplo para enviar una instrucción que trace una línea en diagonal se tiene que utilizar una interpolación lineal denominada por el código G1 seguida de los parámetros de velocidad y las coordenadas a donde se dirigirá el laser para realizar el grabado, para la el robot cartesiano solo se necesitan tres parámetros el primero es la velocidad del movimiento (F##), las coordenada en X (X##), la coordenada en Y (Y##), el comando tiene la siguiente forma: G1 F500 X300 Y200 [25].

El programa utiliza la librería Pyserial para Python este complemento permite la comunicación serial que se utiliza para mandar los códigos G al microcontrolador que sirve de enlace con los drivers de los motores a pasos que le dan movimiento al robot cartesiano.

3.4 Pruebas

Para realizar las pruebas se contó con el apoyo de la técnica en diseño de modas y patronaje Laura Carrasco Aráoz quien realizo los patrones a la escala de 1:2 de los encuartes traseros de las tallas 36, 34 y 32 del pantalón de mezclilla para caballero (Figura 26).

(a) (b)

Figura 26 a) Encuartes de las tallas 36, 34, 32 del pantalón de mezclilla para caballero b) Encuarte izquierdo y derecho [Fuente propia 2020].

Los datos que servirán de guía para determinar la orientación de la pieza y poder localizar los vértices donde se realizaran las marcas, se obtienen tomando una fotografía al encuarte colocado en la en la posición correcta (271) (figura 27) posteriormente se calcula la ubicación de las marcas con la relación matemática antes descrita, una vez determinados los datos se guardan en una base de datos del tipo csv (valores separados por comas por sus siglas en ingles). La figura 27,

donde se puede ver el encuarte colocado en la posición para ser marcado (271), el láser que realizara la marca (272) y el robot cartesiano que desplazara el láser (273).

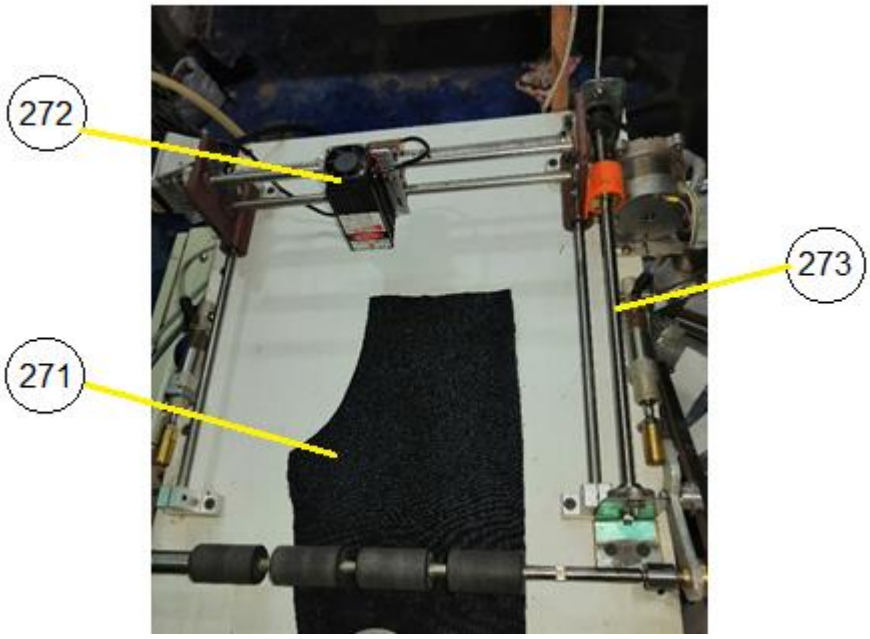


Figura 27 Posición en el prototipo del encuarte trasero para tomar los datos base [Fuente propia 2020]

La salida del algoritmo para determinar los valores de los puntos que se tomaran como base para posteriormente realizar el marcado de los encuartes indicados por dos puntos amarillos, ellos indican la ubicación de los vértices de las marcas el programa calcula el valor de la longitud de la línea diagonal amarilla, el ángulo entre las dos líneas, las coordenadas en forma polar de los puntos (figura 28) y los guarda en un archivo con extensión .csv, la intención de guardar los datos de forma permanente es brindar la facilidad de cambiar de modelo o marca de manera sencilla.

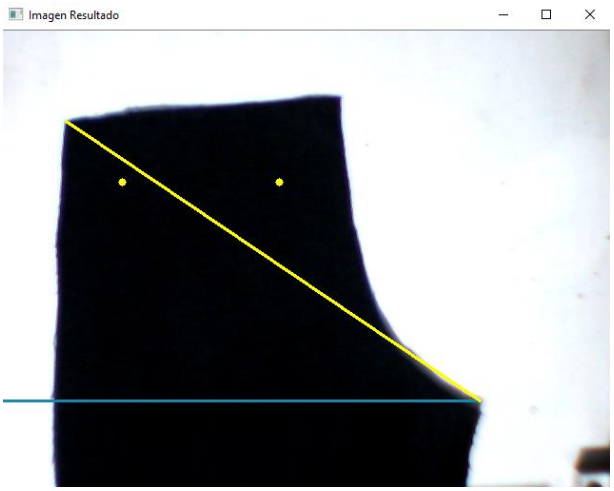


Figura 28 Salida del algoritmo para localizar los valores para el procesamiento de las prendas [Fuente propia 2020]

Figura 29 Encuartes identificados por el algoritmo [Fuente propia2020]

Para identificar si la parte es derecha o izquierda se evalúa el valor de la coordenada en X de la esquina inferior si el valor es menor al valor de la componente en x de la esquina más próxima a ella entonces el encuarte es izquierdo en caso contrario el encuarte es derecho. Al ejecutar el algoritmo distinguió entre el encuete derecho del izquierdo (Figura 29).

Para poder mandar la información en código G es preciso establecer una relación de cuantos pixeles equivalen a un milímetro para ello se captura una imagen con un recorte de tela de dimensiones conocidas, se procesa la imagen, se detectan las esquinas y se cuentan los pixeles que hay entre sus esquinas en el eje de las ordenadas y se dividen entre la longitud conocida de la tela y de obtiene la equivalencia buscada.

Fabla 4 Lectura pixeles para una distancia de 104 mm de longitud

	Cantidad de Pixeles en la dirección de X
1	265
2	266
3	265
4	266
5	270
6	265
7	265
8	265
9	259
10	266

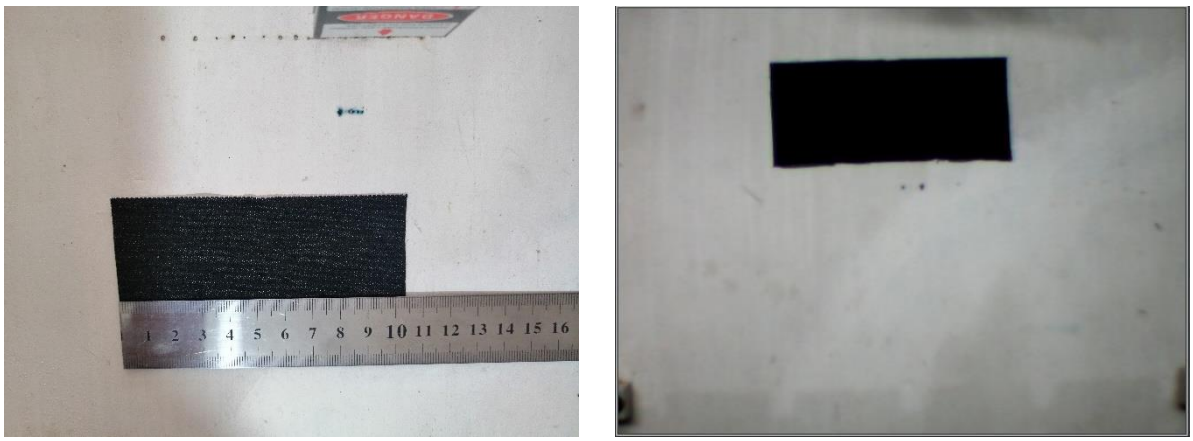
La tabla 4 muestra los valore que se obtuvieron al tomar la imagen del patrón que se utilizó para determinar la relación pixeles-milímetros; para recolectar los datos se tomaron fotografías con el patrón ubicado en diferentes lugares del área de trabajo, la figura 30 muestra el patrón que se utilizó, así como una muestra de la imagen capturada utilizada para determinar relación utilizada calcular la relación. con los valores capturados se realizó un promedio de las diez medidas tomadas dando como resultados 265.2 pixeles, este valor se divido por la longitud del

patrón que tiene un valor de 104 mm lo que da como resultado de 2.55 por lo tanto la relación queda establecida de la siguiente manera:

$$D_{mm} = \frac{D_{px}}{2.55} \tag{17}$$

donde:

D_{mm} $Dixtancia\ en\ milimetros$
D_{px} $Distancia\ en\ pixeles$



(a) (b)

Figura 30 a) Trozo de tela utilizado como patrón para obtener los datos b) Fotografía tomada por el sistema para calcular la relación pixeles-milímetros [Fuente propia 2020].

3.5 Resultados y conclusiones

3.5.1 Discusión de los resultados

Durante la puesta en marcha del sistema de fue necesario desarrollar una tarjeta electrónica para enlazar las señales del del micro controlador con las requeridas por el PLC, debido a que el micro controlador opera con voltajes de 5 V cd y el controlador lógico programable necesita para reconocer una señal de entrada de 24 Vcd, el circuito electrónico se muestra en la figura 31 para acoplar los diferentes tipos de señales se utilizó el opto acoplador G4 ODC5 de la marca OPTO 22 que tiene una señal de entrada de 5 Vcd y un voltaje máximo de salida de 60 Vcd y una capacidad de corriente de 3 Amperes así como una resistencia de aislamiento de 4000 V (transitorios) [23]

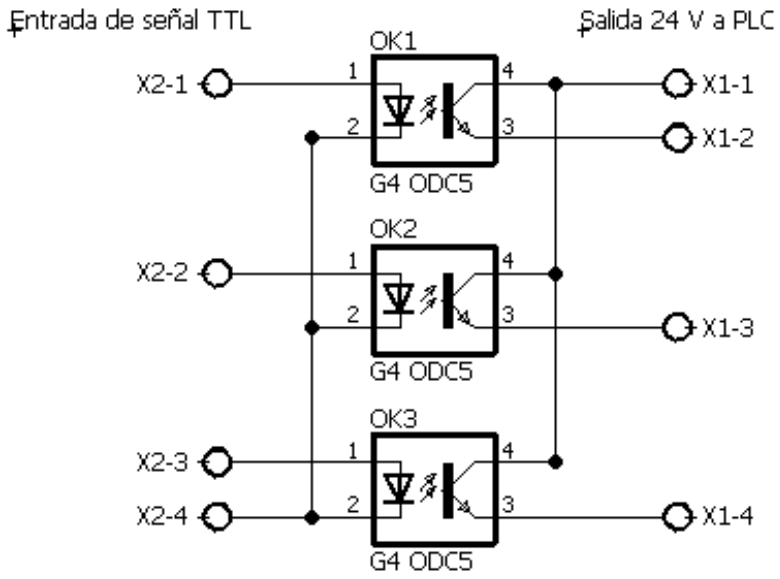


Figura 31 Circuito para acople de señales

Las salidas tienen por función la primera comunicar la señal de inicio para el sistema de posicionamiento, la segunda corresponde a la señal de parada de los

rodillos que posicionan a la pieza en su lugar y la tercera corresponde a la señal de fin de ciclo.

En la ejecución del algoritmo de detección de esquinas de Harris por primera se implementaron valores en torno a 2 en los parámetros del módulo de OpenCv.

La instrucción **"cv2.cornerHarris(img,blockSize,ksize,k)"** sirve para aplicar el algoritmo de detección de esquinas de Harris. los parámetros necesarios para su correcta ejecución son: img representa la imagen a procesar, blockSize indica el tamaño del gradiente también llamado vecindario considerado para la detección de esquinas, ksize representa el parámetro de apertura y finalmente k que es el parámetro libre o empírico de la ecuación de Harris [13].

Los valores iniciales de los parámetros se propusieron de forma aleatoria: `cv2.cornerHarris(blur,2,3,0.14)` con estas condiciones se arrojaron resultados erróneos así que de forma heurística se ajustaron hasta que se detectaron correctamente las esquinas de interés. los valores se muestran en la línea de código siguiente:

"`esquina = cv2.cornerHarris(blur1, 3, 9, 0.021))`"

Una ves detectadas las esquinas se determina si el encuarte es derecho o izquierdo de acuerdo a la posición de la coordenada en x del punto 3 y el centroide de la imagen, así como la inclinación del encuarte con relación a los puntos 1 y 2 con el centroide. Esto es posible ya que las esquinas detectadas se guardan en un arreglo donde la posición 0 la ocupa el centroide y las posiciones

siguientes se ocupan en orden de aparición de las esquinas, por tanto si el punto 2 fuese el formado por la esquina superior derecha (representada en el algoritmo por la variable X1) como la primera esquina encontrada el encuarte tendrá un giro hacia la izquierda ya que la esquina superior izquierda se encontraría por debajo del punto dos y por tanto la coordenada en x del punto tiene un valor mayor al de la coordenada en x del centroide (representado por la variable X0 en el algoritmo) del encuarte; la sección de código mostrada toma la decisión antes descrita (el código completo se muestra en el anexo 2).

```
# detectamos el sentido de la pieza
if x3 < x0:    # encuarte derecho
      print('el encuarte es derecho')
      teta1 = anguloP1
      teta2 = anguloP2
      teta0 = anguloReferencia

      if x0 < x1: # Si el punto se encuentra después del centroide
            # ('determinamos la longitud de la diagonal principal')
            dgl = math.sqrt(math.pow(x3, 2) + math.pow(x1, 2))
            alfa = math.atan((y2 - y3) / (x1 - x3))

            '''Determinamos los ángulos nuevos'''
            alfa0 = alfa - anguloReferencia
            alfa1 = anguloP1 + alfa0
            alfa2 = anguloP2 + alfa0

            '''Calculamos las distancias'''
            r1 = 0.730681396 * dgl
            r2 = 0.9413953 * dgl
```

Con la dirección de la pieza detectada se calculan las coordenadas de donde se efectuarán las marcas, y mediante la relación pixeles descrita en el capítulo anterior se convierten los pixeles a milímetros.

Para generar la secuencia de códigos G se programó la clase códigoG esta se encarga de tomar los valores de las coordenadas de los vértices de contenidos

en la variable marcas y mediante su método convierte transforma los valores en pixeles a milímetros como se puede observar en el fragmento de código siguiente:

```
class CodigoG:

def convierte(self, marcas): # método para convertir los
valores a milímetros

return marcas * 0.39216
```

de igual gorma el método codigoGerber genera los la información de la posición que se envía a la interfaz que realiza el control de los actuadores la fracción del código siguiente muestra el arreglo que devuelve el método para ser enviado por el puerto serial hacia el microcontrolador encargado de hacer el enlace entre la computadora y los drivers de los motores encargados de la motricidad del robot carteciano, así como de la activación del láser.

```
codigosG.append('G00 F200 ' + 'X' + pInicialX10 + ' Y'
                + pInicialY10 + ' \n')
codigosG.append('G00 F200 Z1 \n')
codigosG.append('G01 F100 ' + 'X' + pInicialX11 + ' Y'
                + pInicialY11 + ' \n')
codigosG.append('G01 F100 ' + 'X' + pInicialX12 + ' Y'
                + pInicialY12 + ' \n')
codigosG.append('G00 F200 Z-1 \n')
codigosG.append('G00 F200 ' + 'X' + pInicialX20 + ' Y'
                + pInicialY20 + ' \n')
codigosG.append('G00 F200 Z1 \n')
codigosG.append('G01 F100 ' + 'X' + pInicialX21 + ' Y'
                + pInicialY21 + ' \n')
codigosG.append('G01 F100 ' + 'X' + pInicialX22 + ' Y'
                + pInicialY22 + ' \n')
codigosG.append('G00 F200 Z-1 \n')
```

La figura 31 muestra el gravado de las marcas por el láser, en ella puede observase que la marca es apenas perceptible, indicativo de que dispositivo propuesto para realizar el marcaje no es el adecuado por lo que se replantea utilizar un láser de mayor potencia, como mejora del proyecto.

Figura 31 laser realizando la marca en el encuarte racero

La figura 32 muestra la marca realizada por el láser en la imagen se puede observar que la marca es apenas visible esto es a consecuencia de que la potencia del dispositivo propuesto para hacer las marcas es insuficiente, por lo que se propone remplazarlo por uno con mayor potencia y que cuente con tarjeta controladora lo que permitirá ajustar la potencia para marcar la prenda de forma adecuada.

Figura 32 marcas realizadas por el laser

3.5.2 Conclusiones

En el desarrollo del trabajo se pudo apreciar que el problema planeado de localizar las esquinas se puede solucionar aplicando la técnica de segmentación de imagen y el algoritmo de Harris para la detección de estas. El aplicar el filtrado a la imagen que se analizará trae como beneficio que al ejecutar el algoritmo para la detección de esquinas las esquinas localizadas sean las de interés y no se generan falsos positivos. Con la localización de las esquinas fue posible extraer las coordenadas de su ubicación permitiendo trazar marcas para colocar las piezas del ensamble de la prenda. Finalmente, a través de un tratamiento por geometría analítica se hace la conversión de pixeles a la distancia que han de moverse los actuadores.

También se pudo observar que para realizar la marcación se requiere de una mayor potencia en el láser ya que las marcas no se percibían de forma correcta, aunque el algoritmo trabaja de manera adecuada el marcado no se realizó de

forma visible debido a que la potencia el láser fue insuficiente para formar una marca visible.

Para mejorar el funcionamiento del dispositivo se propone como trabajo futuro se propones dos puntos el primero utilizar una red una red neuronal convolucional para el clasificador, la cual tendrá la tarea de distinguir si la prenda a procesar corresponde al tipo deseado es decir si corresponde al tipo de prenda que se está procesando y al introducir ora por ejemplo una parte delantera la maquina pueda excluirla del proceso como segundo punto agregar la capacidad de obtener automáticamente las coordenadas de nuevos modelos con solo mostrar la prenda terminada y final mente colocar un láser infrarrojo de mayor potencia.

Referencias

[1] L. B. S. Campos, Interviewee, Entrevista para conocer las necesidades de la empresa CONFETEX S.A. de C.V. [Entrevista]. 15 febrero 2019.

[2] C. Daniel Ortega, F. Ernesto Moyano, C. Gustavo Raúl Sbrugnera, and T. José Gabriel Tejerina, "Técnicas de Implementación de Visión Estereoscópica en Robótica," pp. 1850–2946, 2013.

[3] A. Voulodimos, N. Doulamis, A. Doulamis, and E. Protopapadakis, "Deep Learning for Computer Vision: A Brief Review," *Comput. Intell. Neurosci.*, vol. 2018, pp. 1–13, 2018.

[4] S. Gutstein, O. Fuentes, and E. Freudenthal, "Knowledge Transfer in Deep Convolutional Neural Nets," *Int. J. Artif. Intell. Tools*, vol. 17, no. 03, p. 555, 2008.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Adv. Neural Inf. Process. Syst.*, pp. 1–9, 2012.

[6] S. Zapata, D. Velásquez, and I. Pardo, "Desarrollo e Implementación de un Sistema de Visión Artificial Basado en Lenguajes de Uso Libre para un Sistema Seleccionador de Productos de un Centro," *Lámpsakos*, no. Cim, pp. 43–50, 2016.

[7] P. Constante, O. Chang, E. Pruna, and I. Escobar, "Artificial Vision Techniques for Strawberry ' s Industrial Classification," *Ieee Lat. Am. Trans.*, vol. 14, no. 6, pp. 2576–2581, 2016.

[8] R. S.Sabeenian, M. E. Paramasivam, and P. M. Dinesh, "Computer Vision based Defect Detection and Identification in Handloom Silk Fabrics," *Int. J. Comput. Appl.*, vol. 42, no. 17, pp. 41–48, 2012.

[9] I. Erényi and J. Pongrácz, "Quality control in textile industry via machine vision," *Microprocess. Microprogramming*, vol. 32, no. 1–5, pp. 807–813, 1991.

[10] M. Zarbin, C. Montemagno, J. Leary, and R. Ritch, "Artificial vision.," *Panminerva Med.*, vol. 53, no. 3, pp. 167–177, 2011.

[11] A. Voulodimos, N. Doulamis, A. Doulamis, and E. Protopapadakis, "Deep Learning

for Computer Vision: A Brief Review," Comput. Intell. Neurosci., vol. 2018, pp. 1–13, 2018.

[12] López Peña Antonio, "Coursera," *Curso: detección de objetos*, 2014. [Online]. Available: https://www.coursera.org/learn/deteccion-objetos/l.

[13] "Image Filtering — OpenCV 3.0.0-dev documentation," 2010. [Online]. Available:http://docs.opencv.org/3.0-beta/modules/imgproc/doc/filtering.html#. [Accessed: 31-Aug-2017].

[14] R. C. Gonzalez and R. E. Woods, *Digital Image Processing (3rd Edition)*. 2007.

[15] D. Sebasti Comas and G. J. Meschino, "Segmentación de Imágenes mediante Reconocimiento de Patrones," *Esc. y Work. Argenino en Ciencias las Imagenes*, 2014.

[16] R. Casallas and A. Yie "Ingenieria de Software ciclos de cida y metodologias Univ Los Andes Fac Ing. 2010

[17] J. S. Blanes and J. L. Gorricho, "Técnicas de Evaluación de la Calidad de la Imagen . Tendencias y métricas basadas en Bordes.," UPC. 2010.M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989

[18] F. Gallegos, V. Ponomaryov, O. Pogrebnyak y L. N. d. Rivera, «Filtros Robustos DiferentesRM-KNN con Diferentes Funciones de Influencia para la Supresión de Ruido Impulsivo en Imágenes Digitales,» Computación y Sistemas, vol. 6, nº 3, pp. 183-195, 2003

[19] L. M. Quintana, F. Hubert y E. Marañon, «"OPTIMIZACIÓN DE LOS FITROS MEDIANA-GAUSSIANO PARA UNA MEJOR CONVERGENCIA DEL SNSKE EN LA SEGMENTACIÓN DE IMÁGENES MÉDICAS",» de Informatica 2013 XV Convencion y Feria, Santiago de Cuba, 2013

[20] J. H. Sossa, Visión Artificial Rasgos Descriptores para el Reconocimiento de Objetos, Madrid España: Ra-Ma, 2013

[21] J. S. Cuenca, RECONOCIMIENTO DE OBJETOS POR DESCRIPTORES DE FORMA, BARCELONA: Universitat de Barcelona, 2008.

[22] C. Harris, M. Stephens, A Combined Corner and Edge Detector, Proc of 4th. Aley Vision Conference, Manchester, Vol. 15:147-151, 1988

[23] OPTO 22, Data Sheet G4 Diggital DC Output Modules, 2012.

[24] J. Howse, OpenCV Computer Vision whit Python, Birmingham B3 2PB, UK., 2013

[25] Smid, P. CNC Control Setup for Milling and Turning. Estados Unidos de América: Industrial Press 2005.

ANEXOS

Anexo 1 Código G ISO 6983

Códigos Generales

G00: Posicionamiento rápido (sin maquinar)

G01: Interpolación lineal (maquinando)

G02: Interpolación circular (horaria)

G03: Interpolación circular (antihoraria)

G04: Compás de espera

G10: Ajuste del valor de offset del programa

G20: Comienzo de uso de unidades imperiales (pulgadas)

G21: Comienzo de uso de unidades métricas

G28: Volver al home de la máquina

G32: Maquinar una rosca en una pasada

G36: Compensación automática de herramienta en X

G37: Compensación automática de herramienta en Z

G40: Cancelar compensación de radio de curvatura de herramienta

G41: Compensación de radio de curvatura de herramienta a la izquierda

G42: Compensación de radio de curvatura de herramienta a la derecha

G70: Ciclo de acabado

G71: Ciclo de maquinado en torneado

G72: Ciclo de maquinado en refrentado

G73: Repetición de patrón

G74: Taladrado intermitente, con salida para retirar virutas

G76: Maquinar una rosca en múltiples pasadas

G96: Comienzo de desbaste a velocidad tangencial constante

G97: Fin de desbaste a velocidad tangencial constante

G98: Velocidad de alimentación (unidades/min)

G99: Velocidad de alimentación (unidades/revolución)

Códigos Misceláneos

M00: Parada opcional

M01: Parada opcional

M02: Reset del programa

M03: Hacer girar el husillo en sentido horario

M04: Hacer girar el husillo en sentido antihorario

M05: Frenar el husillo

M06: Cambiar de herramienta

M07: Abrir el paso del refrigerante B

M08: Abrir el paso del refrigerante A

M09: Cerrar el paso de los refrigerantes

M10: Abrir mordazas

M11: Cerrar mordazas

M13: Hacer girar el husillo en sentido horario y abrir el paso de refrigerante

M14: Hacer girar el husillo en sentido antihorario y abrir el paso de refrigerante

M30: Finalizar programa y poner el puntero de ejecución en su inicio

M31: Incrementar el contador de partes

M37: Frenar el husillo y abrir la guarda

M38: Abrir la guarda

M39: Cerrar la guarda

M40: Extender el alimentador de piezas

M41: Retraer el alimentador de piezas

M43: Avisar a la cinta transportadora que avance

M44: Avisar a la cinta transportadora que retroceda

M45: Avisar a la cinta transportadora que frene

M48: Inhabilitar Spindle y Feed override (maquinar exclusivamente con las velocidades programadas)

M49: Cancelar M48

M62: Activar salida auxiliar 1

M63: Activar salida auxiliar 2

M64: Desactivar salida auxiliar 1

M65: Desactivar salida auxiliar 2

M66: Esperar hasta que la entrada 1 esté en ON

M67: Esperar hasta que la entrada 2 esté en ON

M70: Activar espejo en X

M76: Esperar hasta que la entrada 1 esté en OFF

M77: Esperar hasta que la entrada 2 esté en OFF

M80: Desactivar el espejo en X

M98: Llamada a subprograma

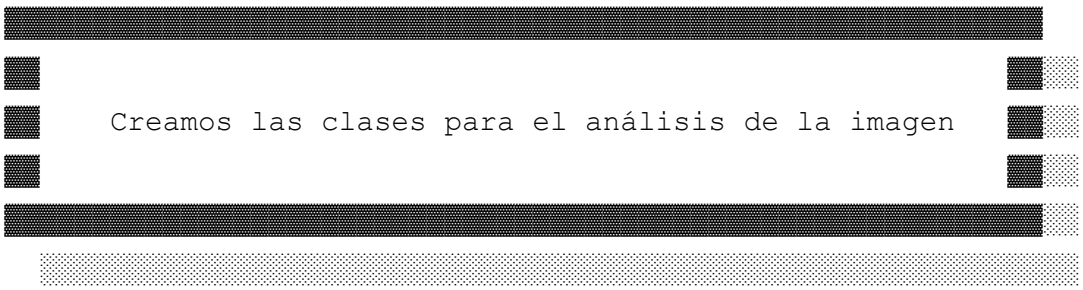
M99: Retorno de subprograma

Anexo 2 Código de Programación

```
# !/usr/bin/env python
# -*- coding: utf-8 -*-

'''Importamos las librerias a utilizar'''
import cv2
import numpy as np
import math
import serial
import time
import os

'''
```




```
'''
class Imagen:

    def filtro(self, imagen):
        ''' Recibimos la imagen a procesar'''
        img = imagen
        # combertimos a escala de grises
        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

        # creamos la imagen binaria
        ret, binaria = cv2.threshold(gray, 127, 255,
         cv2.THRESH_BINARY_INV)

        return cv2.GaussianBlur(binaria, (5, 5), 0)
```

```
    def cornerHarris(self, blur):
        # localizamos las esquinas con el algoritmo de harris
        blur1 = np.float32(blur)
        esquina = cv2.cornerHarris(blur1, 3, 9, 0.021)
        ret, unbralEsquina = cv2.threshold(esquina, 0.01 *
         esquina.max(), 255, 0)
        dst = cv2.dilate(unbralEsquina, None)

        # definimos el umbral para un valor óptimo, puede variar
          según la imagen
        # img[dst > 0.01*dst.max()] = [255, 255, 255]
        dst = np.uint8(dst)
        # encuentra centroides
       ret, labels, stats, centroids =
         cv2.connectedComponentsWithStats(dst)

        criteria = (cv2.TERM_CRITERIA_EPS +
           cv2.TERM_CRITERIA_MAX_ITER, 10, 0.1)
        corners = cv2.cornerSubPix(blur1, np.float32(centroids),
         (5, 5), (-1, -1), criteria)
        res = np.hstack((centroids, corners))
        res = np.int0(res)
        #img[res[:, 1], res[:, 0]] = [0, 0, 255]
        #img[res[:, 3], res[:, 2]] = [255, 0, 0]

        corners = corners.astype(int)

        return corners


class Marcas:

    def coordenadas(self, corners, anguloReferencia, anguloP1,
anguloP2, img1):
```

```
# definimos las variables
puntos = np.zeros((3, 2))

x0 = int(corners[0][0])
y0 = int(corners[0][1])

x1 = int(corners[1][0])
y1 = int(corners[1][1])

x2 = int(corners[2][0])
y2 = int(corners[2][1])

x3 = int(corners[3][0])
y3 = int(corners[3][1])

# detectamos el sentido de la pieza

if x3 < x0:
    # encuarte derecho
    print('el encuarte es derecho')
    teta1 = anguloP1
    teta2 = anguloP2
    teta0 = anguloReferencia

    if x0 < x1: # Si el punto se encuentra despues del
                  centroide
# ('determinamos la longitud de la diagonal principal')
      dgl = math.sqrt(math.pow(x3, 2) + math.pow(x1, 2))
        alfa = math.atan((y2 - y3) / (x1 - x3))

        '''Determinamos los angulos nuevos'''
        alfa0 = alfa - anguloReferencia

        alfa1 = anguloP1 + alfa0
        alfa2 = anguloP2 + alfa0
```

```
            puntos[2][0] = alfa1
            puntos[2][1] = alfa2

            '''Calculamos las distancias'''
            r1 = 0.730681396 * dgl
            r2 = 0.9413953 * dgl

            '''Calculamos las coodenadas cartecianas'''
            px1 = r1 * math.cos(alfa1)
            py1 = r1 * math.sin(alfa1)
            px2 = r2 * math.cos(alfa2)
            py2 = r2 * math.sin(alfa2)

            coorx1 = x3 + px1
            coory1 = y3 + py1
            coorx2 = x3 + px2
            coory2 = y3 + py2

            puntos[0][0] = coorx1
            puntos[0][1] = coory1-5
            puntos[1][0] = coorx2
            puntos[1][1] = coory2-23

            #puntos = puntos.astype(int)

            return puntos

        elif x0 < x2:
            #Siel punto se encuetra despues del centroide

    # ('determinamos la longitud de la diagonal principal')
          dgl = math.sqrt(math.pow(x3, 2) + math.pow(x2, 2))
            alfa = math.atan((y2 - y3) / (x2 - x3))
```

```
            '''Determinamos los angulos nuevos'''
            alfa0 = alfa - anguloReferencia
            alfa1 = anguloP1 + alfa0
            alfa2 = anguloP2 + alfa0
            puntos[2][0] = alfa1
            puntos[2][1] = alfa2

            '''Calculamos las distancias'''
            r1 = 0.581396 * dgl
            r2 = 0.813953 * dgl

            '''Calculamos las coodenadas cartecianas'''
            px1 = r1 * math.cos(alfa1)
            py1 = r1 * math.sin(alfa1)
            px2 = r2 * math.cos(alfa2)
            py2 = r2 * math.sin(alfa2)

            coorx1 = x3 + px1
            coory1 = y3 + py1
            coorx2 = x3 + px2
            coory2 = y3 + py2

            puntos[0][0] = coorx1
            puntos[0][1] = coory1
            puntos[1][0] = coorx2
            puntos[1][1] = coory2

            #puntos = puntos.astype(int)

            return puntos

    #Encuarte Izquierdo
    else:
        print('el encuarte es izquierdo')
        if x0 > x1:
```

```
                cv2.line(img, (x1, y1), (x3, y3), (255, 0, 0),
2)
                angulo = math.atan((y3 - y1) / (x3 - x1))
                '''locate points to made marks'''
                puntoX0 = int(x3 - ((x3 - x1) * 0.484848))
                puntoX1 = puntoX0 + 20
                puntoX3 = int(x3 - ((x3 - x1) * 0.863232))
                puntoX2 = puntoX3 - 20
                puntoy0 = int(y1 + ((y3 - y1) * 0.2181818))
                puntoy1 = puntoy0 + 20


            elif x0 > x2:
                cv2.line(img, (x2, y2), (x3, y3), (0, 255, 255),
2)
                angulo = math.atan((y3 - y2) / (x3 - x2))
                '''Dibujamos las marcas'''
                puntoX0 = int(x3 + ((x2 - x3) * 0.484848))
                puntoX1 = puntoX0 + 20
                puntoX3 = int(x3 + ((x2 - x3) * 0.863232))
                puntoX2 = puntoX3 - 20
                puntoy0 = int(y2 + ((y3 - y2) * 0.2181818))
                puntoy1 = puntoy0 + 20
                print('los puntos ', puntoX0, ' ', puntoX1, ' ',
puntoX2, ' ', puntoX3)
                print('los puntos y son ', puntoy0, ' ',
puntoy1)


class CodigoG:

    def convierte(self, marcas):

        return marcas * 0.39216

    def codigoGerber(self, puntosG, anguloAlfa1, anguloAlfa2):
```

```
        codigosG = []

        '''Determinamos los puntos para el primer punto'''
        pInicialX10 = str(puntosG[0][0] + (math.sin(anguloAlfa1)
* 10))
        pInicialY10 = str(puntosG[0][1] + (math.cos(anguloAlfa1)
* 10))
        pInicialX11 = str(puntosG[0][0])
        pInicialY11 = str(puntosG[0][1])
        pInicialX12 = str(puntosG[0][0] + (math.cos(anguloAlfa1)
* 10))
        pInicialY12 = str(puntosG[0][0] - (math.sin(anguloAlfa1)
* 10))

        '''Determinamos los puntos para el segundo punto'''
        pInicialX20 = str(puntosG[1][0] - (math.cos(anguloAlfa2)
* 10))
        pInicialY20 = str(puntosG[1][1] + (math.sin(anguloAlfa2)
* 10))
        pInicialX21 = str(puntosG[1][0])
        pInicialY21 = str(puntosG[1][1])
        pInicialX22 = str(puntosG[1][0] + (math.sin(anguloAlfa2)
* 10))
        pInicialY22 = str(puntosG[1][1] + (math.cos(anguloAlfa2)
* 10))

        ''' generamos el arreglo que contendra el codigo G para
el trazo de las marcas'''
        codigosG.append('G00 F200 ' + 'X' + pInicialX10 + ' Y'
                        + pInicialY10 + ' \n')
        codigosG.append('G00 F200 Z1 \n')
        codigosG.append('G01 F100 ' + 'X' + pInicialX11 + ' Y'
                        + pInicialY11 + ' \n')
        codigosG.append('G01 F100 ' + 'X' + pInicialX12 + ' Y'
                        + pInicialY12 + ' \n')
        codigosG.append('G00 F200 Z-1 \n')
        codigosG.append('G00 F200 ' + 'X' + pInicialX20 + ' Y'
                        + pInicialY20 + ' \n')
```

```
        codigosG.append('G00 F200 Z1 \n')
        codigosG.append('G01 F100 ' + 'X' + pInicialX21 + ' Y'
                        + pInicialY21 + ' \n')
        codigosG.append('G01 F100 ' + 'X' + pInicialX22 + ' Y'
                        + pInicialY22 + ' \n')
        codigosG.append('G00 F200 Z-1 \n')
        codigosG.append('G00 F200 X00 Y00 \n')
        return codigosG

'''
```


```
'''
#configuramos el puerto serial
com = serial.Serial('COM5', 9600, timeout=1)
time.sleep(1.8)

#iniciamos las variables
angulos = np.zeros((4,1))
# importamos la imagen a procesar
filename = 'Derecho3.jpg'
img = cv2.imread(filename)

#instanciamos la clase filtro
analisisImagen = Imagen()
coordenadas = Marcas()
codigoG = CodigoG()

imagenFiltrada = analisisImagen.filtro(img)
esquinas = analisisImagen.cornerHarris(imagenFiltrada)
archivo = open('angulos.txt', 'r')
```

```
i = 0
for linea in archivo:
    angulos[i][0] = float(linea)
    i = i + 1

archivo.close()
i = 0
marcass = coordenadas.coordenadas(esquinas, angulos[0][0],
angulos[1][0], angulos[2][0], img)
marcas = marcass.astype(int)
convercion = codigoG.convierte(marcas)
alfa_1 = marcass[2][0]
alfa_2 = marcass[2][1]
gerberCode = codigoG.codigoGerber(convercion, alfa_1, alfa_2)

print('las esquinas  se encuentran en 2: ', esquinas)

print('los vertices de las marcas se encuentran en: ', marcas)
print('La convercion es ', convercion)
print('los angulos alfa son ', alfa_1, 'y  ', alfa_2)
print('los codigos gerber son ', gerberCode)

cv2.imshow('Imagen Resultado ', imagenFiltrada)

'''Pulsamos una tecla para salir'''
if cv2.waitKey(0) & 0xff == 27:
    cv2.destroyAllWindows()
```