UNIVERSIDAD TECNOLOGICA NACIONAL
FACULTAD REGIONAL SAN NICOLAS

INGENIERIA EN ELECTRONICA

PROBLEMA DE INGENIERÍA

TECNICAS DIGITALES III

CONTROL DE COSTURAS

Integrantes:

- Gisbert Pablo
- Luzi Fernando

Docentes:

- Profesor: Poblete Felipe
- Auxiliar: Gonzalez Mariano

AÑO 2019

INDICE

OBJETIVOS DEL TRABAJO

Realizar el control de calidad sobre las costuras de paracaídas mediante la utilización de cámaras y visión artificial

MATERIAS INTEGRADAS

• Informática II – Programación C/C++. Arrays. Manejo de memoria. Funciones y Procedimientos.
• Técnicas Digitales III – Procesamiento digital de imágenes. Programación en ambiente de PC

POSIBLES APLICACIONES

Nuestro interés se focalizo en la aplicación de visión artificial como sustitución de la inspección visual de un operario para el control de calidad final, apuntando a lo que representa a las costuras de la tela durante la confección de paracaídas. Así mismo, durante la fabricación de paracaídas no solo las costuras pasan por un control final, sino que también el estado de la tela que se cortara para formar el paracaídas debe ser verificado para constatar que no posea ningún defecto de fábrica que pueda debilitar la misma, por lo que otra posible aplicación podría ser la de controlar posibles fallas en las telas de nylon.
Por otra parte, también podría ser aplicada a otros rubros dentro de la industria textil que requieran otros tipos de patrones para un control en sus costuras.
Además, podría aprovecharse en el caso de una industria textil donde existen gran cantidad de máquinas de coser, poder tener la información de cada una de éstas y se compartan para que en caso de que se registren fallas en las costuras puedan verse replicadas no solo en una maquina aleatoria sino en varias, podrían inferirse por ejemplo que se trata de un lote de hilo que no era el adecuado para dicha costura o que por un cambio en el proveedor de la tela, el tipo de hilo que se utilizaba no es compatible con esta nueva marca de tela, etc.

PROFESORES ENTREVISTADOS

• Mariano González – Procesamiento digital de imágenes, Open CV

BIBLIOGRAFÍA

- *Sitios de Internet*
  - Foros y Blogs sobre OpenCV:
    - https://www.learnopencv.com/
    - http://opencvexamples.blogspot.com/
    - http://opencvlibrary.sourceforge.net
    - https://hetpro-store.com/TUTORIALES/category/programacion/opencv/
    - http://docs.opencv.org/modules/

- *Apuntes de la cátedra*
    - Aplicación de la Visión Artificial a la Identificación de Figuras - Lina Anggeli, Bernardo - Tizi, Fernando - Vera, Marcos
- *Libros*
    - Practical OpenCV – Brahmbhatt, Samarth (2013)

- *Herramientas utilizadas:*
    - **Code::Blocks:** es un IDE (entorno de desarrollo integrado) de código abierto, que soporta múltiples compiladores, que incluye GCC, Clang y Visual C++.
      Está licenciado bajo la Licencia pública general de GNU y está disponible para Windows, Linux y macOS
    - **MinGW (Minimalist GNU for Windows)**: anteriormente conocido como MinGW32, es una implementación de los compiladores GCC para la plataforma Win32, que permite migrar la capacidad de este compilador en entornos Windows. Además, MinGW incluye un conjunto de la API de Win32, permitiendo un desarrollo de aplicaciones nativas para esa plataforma, pudiendo generar ejecutables y bibliotecas usando la API de Windows.
      En la página oficial de CodeBlock (http://www.codeblocks.org/) ya nos permite descargar gratuitamente el instalador para la versión que utiliza el compilador MinGW, (codeblocks-mingw-setup.exe), descargamos e instalamos.
    - **TDM-GCC:** es una herramienta que permite instalar en Windows los compiladores MinGW. Se puede descargar desde la web http://tdm-gcc.tdragon.net/ .
    - **Cmake:** es una herramienta open source que utilizamos para la construcción de OpenCV en windows a partir de los archivos fuente y distintos componentes, codecs, lenguajes de programación, que queramos incluir. Se puede obtener más información y descargar el programa desde la web oficial https://cmake.org/.
    - **OpenCV:** es una librería de visión por computador de código abierto. La web oficial es https://opencv.org/.

La librería está escrita en los lenguajes C y C++ y es compatible con Linux, Windows y Mac OS X.
Para poder utilizarla sin limitaciones en Windows, se debe construir mediante CMake y luego compilar en la PC que se desea utilizar.
El proceso es bastante extenso y hay mucha información en la comunidad, no obstante, dejamos el link de la web que usamos como guía para poder utilizar OpenCV junto con CodeBlocks y MinGW en nuestras Pcs.
https://www.kevinhughes.ca/tutorials/opencv-install-on-windows-with-codeblocks-and-mingw

    - **GIMP (GNU IMAGE MANIPULATION PROGRAM):** es un programa open sourse para la edición de imágenes que utilizamos para hacer los patrones ideales.

DESARROLLO

INTRODUCCIÓN

El problema se enfoca en la fabricación de paracaídas, donde la unión de las telas que lo conforman se realiza mediante costura con hilo. Se utiliza para ello el tipo denominado de doble pespunte (Figura 1). Lograr una buena unión de las telas necesita que se cumpla con ciertos parámetros. Uno de ellos, y el que nos interesa analizar en esta ocasión, es la cantidad de puntadas, cada cierta distancia de la costura. Esta debe estar conformada por entre 7 y 10 puntadas por cada 3cm. Una cantidad menor produciría una unión débil de las telas pudiendo provocar que se separen. Mientras que una cantidad mayor provocaría un debilitamiento de las mismas incrementando la probabilidad de sufrir rasgaduras.

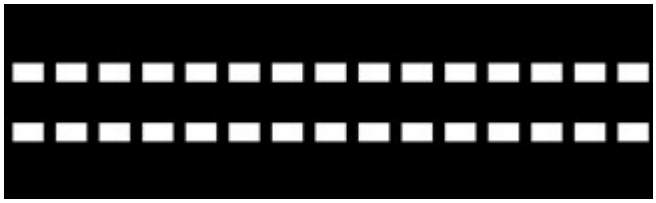


Figura 1

El proceso de control de calidad se realiza de forma visual por un operario al finalizar la costura. Esto reduce el control de calidad a la experiencia en el "ojo" que posea el encargado de esta tarea, por lo que variará entre operarios que realicen dicha verificación. Otro factor sería el tiempo que se tomará cada verificador en observar la costura.

Propuesta de resolución

Entonces, como mejora a este problema, decidimos inclinarnos por automatizar dicho proceso de control de calidad, utilizando visión artificial.

Ubicando una cámara en un punto estratégico, entre la salida de la aguja y una rueda que arrastra la tela ya cosida, manteniendo a esta con una determinada tensión y despejando la salida de la máquina, es posible ver con claridad el patrón de la costura. Si capturamos imágenes a intervalos que permitan la visualización de todas las puntadas, que se determinaran mediante un sensor que capte el movimiento del cabezal y la información del largo promedio de cada puntada, podemos analizar las distintas variantes en la costura y determinar si están dentro de los parámetros estipulados.

De esta manera podemos obtener un control de calidad en tiempo real y, si se encontraran fallas, emitir alarmas que nos informen. Además, podríamos inferir problemas de faltante o corte del hilo que se usa para las costuras, calibración de la puntada promedio, etc. Estos problemas podrían detectarse al instante, evitando la pérdida de tiempo, a diferencia del control actual que arroja los resultados una vez que la etapa de costura del paracaídas concluyó.

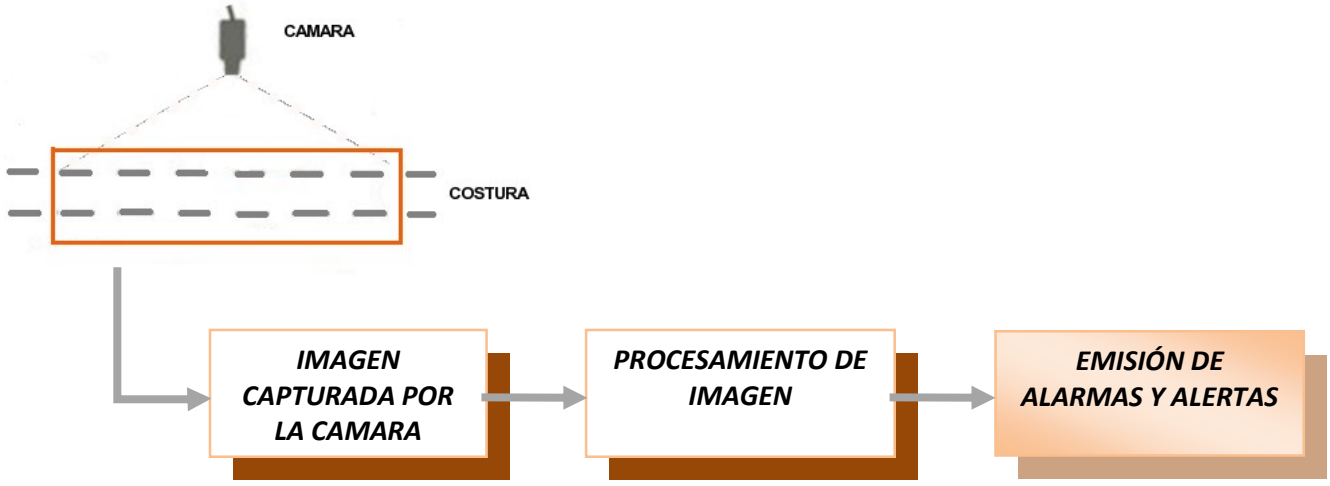
ALCANCE DEL PROBLEMA

Tomaremos como alcance del problema, el diseño de un algoritmo que permita, mediante la imagen de una costura de doble pespunte, donde el hilo sea de diferente color al de la tela empleada, realizar el análisis de las puntadas. Se pretende determinar la longitud de cada puntada pudiendo

diferenciar si está dentro de los límites aceptables identificando la unión de 2 o más de ellas. También se analizará el espaciado entre puntadas de manera que se pueda identificar si la aguja entro y salió muy rápido, o el avance de la tela fue más lento en ese momento, o si hubo un arrastre de la tela mientras la aguja estaba adentro. Esto permitirá, además, inferir sobre la faltante de una o más puntadas.
Todos los parámetros de importancia se configurarán en un archivo de texto permitiendo la utilización del programa para distintas configuraciones de telas, hilos y longitudes de puntadas.
Se proveerá además una sección de calibración para realizar la puesta en marcha. Se permitirá, dimensionar la imagen a partir de un patrón de longitudes conocidas y ajustar los valores de umbral para la detección, por color, de las puntadas.
Vale destacar que el hecho de que el color del hilo sea diferente del color de la tela no es menor ya que de ser estos iguales, deberíamos de emplear técnicas de iluminación para generar efectos que nos ayuden a distinguir las puntadas de la tela, pero ese es otro tema de mayor estudio y no aplica al alcance que fue planteado para nuestro problema. No obstante, podría considerarse como una mejora a futuro.

ESTRATEGIA PROPUESTA

Este control de costura la realizaremos mediante un sistema de procesamiento de imagen, del cual una cámara situada en un punto estratégico del proceso, detecte y cuente la cantidad de puntadas realizadas sobre la tela. La cámara obtendrá tomas de fotos cada un intervalo de tiempo de manera que pueda determinarse si las puntadas de esa muestra se encuentran dentro de los valores aceptable para ser considerada esa costura como buena. En el caso de que detecte alguna irregularidad en las puntadas de la costura se emitirá la alarma y/o alerta correspondiente.
Esto nos permitirá así tener un control de la calidad de la costura casi en tiempo real, lo que llegado al caso ganaríamos en tiempo si quisiésemos descartar la tela antes de terminarla por completo o corregir ciertas fallas a tiempo.



Para el procesamiento de imágenes utilizaremos librerías de OpenCV específicas para estas tareas

TEORÍA DE FUNCIONAMIENTO

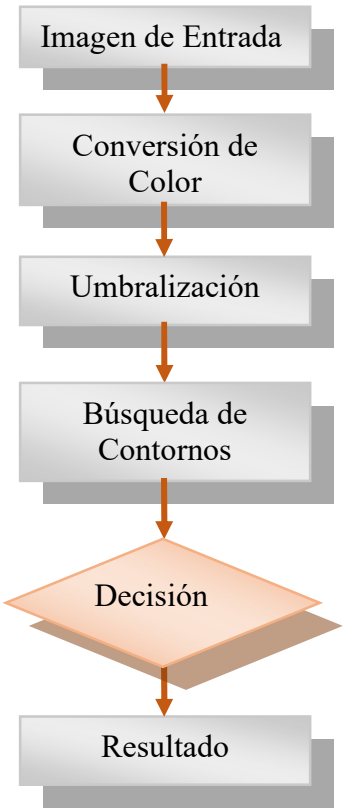
Cámara utilizada.
La cámara de video que utilizamos fue una de marca Genius tipo webcam de 1280x720 pixeles de resolución

Iluminación
Debemos tener una buena fuente de luz uniforme, de manera que las puntadas y más aún la distancia entre las mismas puedan ser detectadas de buena forma, porque de lo contrario cualquier falta de iluminación o de manera parcial y no uniforme podría generar sombras y/o ruido indeseable

Obtención de imágenes.
Una vez que se realizaron los cálculos necesarios y se eligieron la cámara apropiada el siguiente paso fue la realización del montaje.
Una vez terminada la instalación se tomaron muestras con el fin de empezar a estudiar el problema. Se analizaron distintas imágenes.

A partir de las imágenes de muestras tomadas empezamos a diseñar el software para realizar el procesamiento de las imágenes.

DIAGRAMA GENERAL



Descripción de los bloques:

Imagen de Entrada: Esta será la imagen a analizar (previamente hecha su adquisición y digitalización por el medio correspondiente). Las imágenes de entrada se capturarán de manera que lo que se analizarán serán fotos cada un instante de tiempo definido en proporción a una cantidad conocida de puntadas realizadas por el operador de la máquina.

Filtrado: Para realizar el análisis de una imagen es necesario el filtrado con el objeto de eliminar ruidos que puedan provocar un mal funcionamiento del sistema, entonces, la adecuada elección del filtro es fundamental.
Para decidir el tipo de filtro a utilizar es importante saber el origen o tipo de ruido al que pueden estar sometidas las imágenes de entrada (no será lo mismo un filtro de suavizado cuando, en realidad se requiere uno de realce), una vez determinado se prueban las diferentes alternativas hasta que se logre el mejor resultado.

Conversión de Color: Para poder admitir imágenes a color, es decir, de 3 canales (RGB o BGR), la conversión de color es necesaria para que no haya conflicto con el pasaje de parámetros a las funciones de la librería OpenCV que se utilizarán, ya que algunas de ellas solo soportan imágenes en escala de grises (1 canal).

Umbralización: Lo que se logra con ella es poder distinguir el fondo de la imagen de interés en sí, y así poder "separar" la información de manera que a partir de un umbral establecido se puede decir que pertenecerá al fondo y que a la imagen de interés. La imagen resultante será binaria. Para este caso lo que utilizamos fue un filtro por color de OpenCV *inRange*. Está es una función que permite filtrar una ventana de valores para pixeles en una imagen. Esta da como resultado una imagen binaria (blanco y negro), donde el blanco representa a todos los pixeles que fueron válidos para el rango filtrado.

Búsqueda de contornos: Una vez obtenida la imagen binaria nos quedará diferenciada en ella las puntadas en blanco, la cual será nuestro punto de interés para analizar y el resto del fondo en negro. Aprovechando de esta situación lo siguiente que establecimos fue la de detectar esas líneas punteadas, para esto utilizamos la función *findcountors* que nos permite encontrar los N contornos, ya sea internos como externos que puedan existen en una imagen binaria. Esta función nos dará como resultado un vector con los valores de los correspondientes contornos pertenecientes a la imagen. Como en nuestro patrón de costuras tiene una doble línea de puntos, lo que luego hicimos fue separar en un vector los valores de los contornos de lo que sería nuestra línea de puntadas superior y en otro vector los contornos de la línea de costura inferior. Luego ordenamos los vectores de contornos tanto de la línea superior como inferior con el objetivo de poder enumerarlas e identificar casa puntada de izquierda a derecha para marcar sobre la imagen original los contornos de las costuras y luego efectuar la estrategia de decisión sobre si la costura está correctamente realizada o se encuentra deficiente.

Decisión: Se evaluará si las imágenes capturadas y el patrón de costura correcto calculando los espacios entre puntadas y la longitud propia de cada puntada, tanta para la línea superior como para la línea inferior.

Resultado: Se informará al operador de la maquina sobre las detecciones que se realicen sobre la costura.

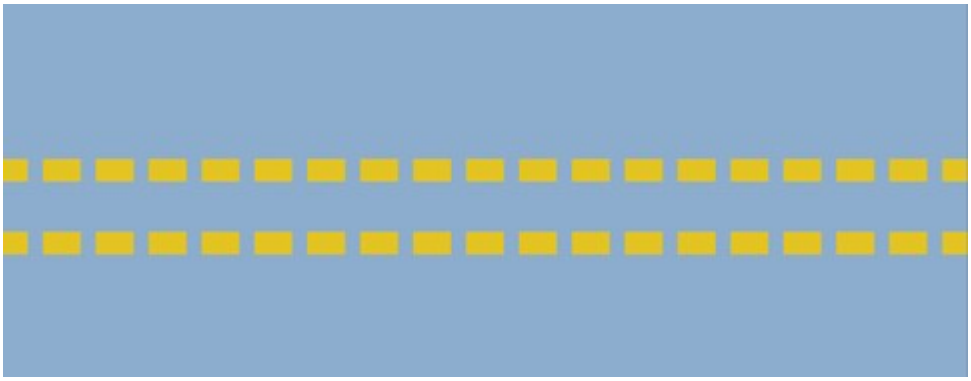
En caso que se presente alguna incongruencia que represente una costura mal lograda, se emitirá una alarma y mensaje que nos advierta que se ha detectado una costura defectuosa. Permitiendo que el usuario pueda intervenir de acuerdo a su procedimiento para tal situación.

DESCRIPCION DEL PROTOTIPO

Durante el transcurso del desarrollo del proyecto fuimos pasando por diferentes alternativas en cuanto al tipo de filtrado para obtener la mejor imagen con la cual tomar la decisión acertada sobre si la costura está dentro del patrón considerado como aceptable.

Para la elección del tipo de filtro que nos fuera más efectivo utilizar para poder identificar las puntadas de las costuras en nuestro proyecto, creamos una imagen de una costura ideal que nos sirviera de patrón y así dividir nuestro problema en dos partes. Una primera parte que conste de una imagen ideal, creada a partir de una costura real, donde a partir de los filtros elegidos adecuadamente obtendremos como resultado la imagen binaria sin ningún tipo de error.
Y de una segunda parte donde a partir de la obtención de esa imagen binarizada, podremos obtener un algoritmo que sea capaz de encontrar y medir la longitud tanto de las puntadas como de su espaciado entre puntada y puntada.

Para la creación del patrón de una costura ideal, utilizando la herramienta gráfica gratuita GIMP, donde optamos por definir unas puntadas de 25mm de largo, 15mm de ancho y 10 mm de espaciado entre una puntada y la siguiente. La imagen patrón puede verse a continuación:



PRIMERA PARTE: FILTRADO DE LA IMAGEN

En una primera instancia propusimos que la imagen pasaría por un filtro blur. Este tipo de filtros desenfoca y difumina la imagen original produciendo una homogeneización de los pixeles. De esta forma pretendíamos eliminar ruido en la tela.
Iniciamos probando algunos filtros en OpenCV, y nos decidimos por el GaussianBlur. Al avanzar con las pruebas terminamos por desecharlo ya que producía alteraciones que podían afectar a la longitud de la puntada.

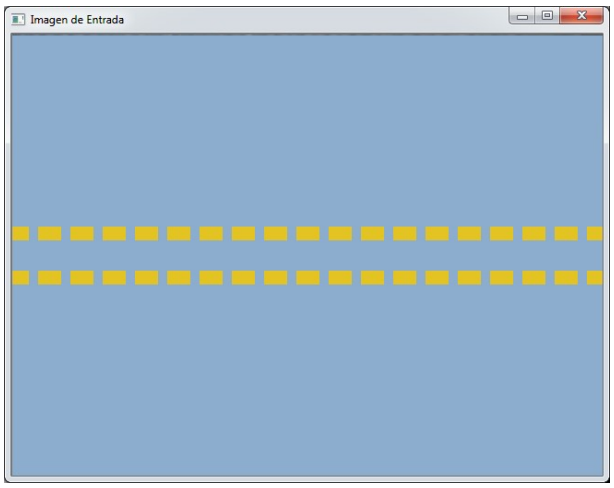
Otro punto importante del procesamiento de imágenes es el de la binarización. Para obtener una imagen binaria Opencv presenta diferentes funciones, entre ellas:

- inRange():Permite crear un filtro ventana para cada uno de los canales de una imagen.

- threshold(): Es un filtro tipo umbral, que permite digitalizar una imagen (crear imagen binaria) a partir de los pixeles que superen cierto límite.
- adaptiveThreshold(): Está función es similar a la anterior, sólo que aplica un filtro adaptativo tipo umbral.
- Canny(): Permite detectar bordes, desarrollo de John F. Canny en 1986.

Dentro de estas posibilidades y alternativas llegamos a encontrar que la función que mejor se adaptaba a nuestra necesidad era la función inRange(), ya que nos permitía hacer una buena discriminación de las puntadas a partir de su color.
En las sucesivas pruebas realizadas se pretendió utilizar la función threshold() para lograr una mejor separación de colores antes de aplicar inRange() pero observamos que las puntadas se deformaban un poco y que podia evitar un análisis correcto.
Sin embargo, tanto la función threshold() como Canny() son utilizadas en la calibración de dimensiones.

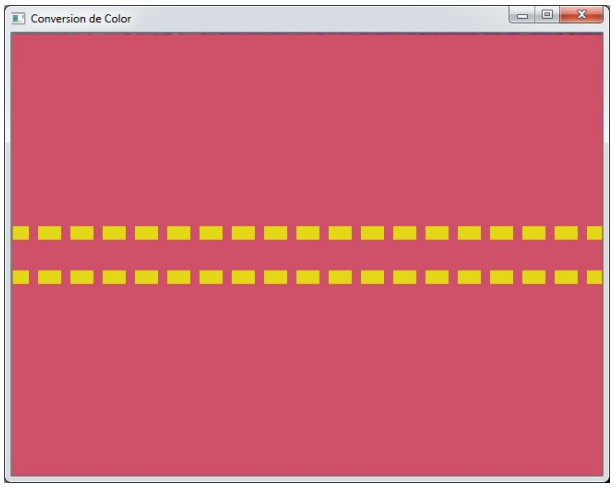
Entonces para empezar a comentar como realizamos esta primera parte, partimos de la premisa que como imagen de entrada capturada será el patrón de costura idealizado, a groso modo para obtener un prototipo, donde el programa de visión pueda verificar los controles de costuras propiamente establecidos al inicio del proyecto.



Una vez tomada la imagen de patrón, la primera función de OpenCV utilizada en nuestro caso fue para realizar la conversión de colores de nuestra imagen original. La función usada fue la siguiente:

cvtColor(frame, hsv, CV_BGR2HSV);

Esta función convierte una imagen de entrada de un espacio de color a otro. En el caso de una transformación desde-hacia el espacio de color RGB (Red-Green-Blue), el orden de los canales debe especificarse explícitamente (RGB o BGR). Hay que tener en cuenta que el formato de color predeterminado en OpenCV a menudo se denomina RGB, pero en realidad es BGR (Blue-Green-Red) (los bytes se invierten). Así, logramos la transformación del espacio de colores de un modelo BGR hacia uno HSV(Hue-Saturation-Value) que nos será de utilidad para aplicar la siguiente función de filtrado.

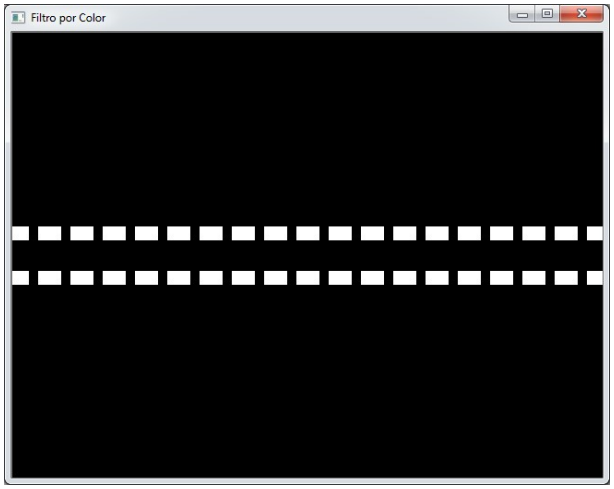


Nos pareció más conveniente la utilización de un filtro por color, donde obtuvimos los resultados deseados. Su función en Opencv es la siguiente:

inRange(hsv, Scalar(iLowH, iLowS, iLowV), Scalar(iHighH, iHighS, iHighV), hsv);

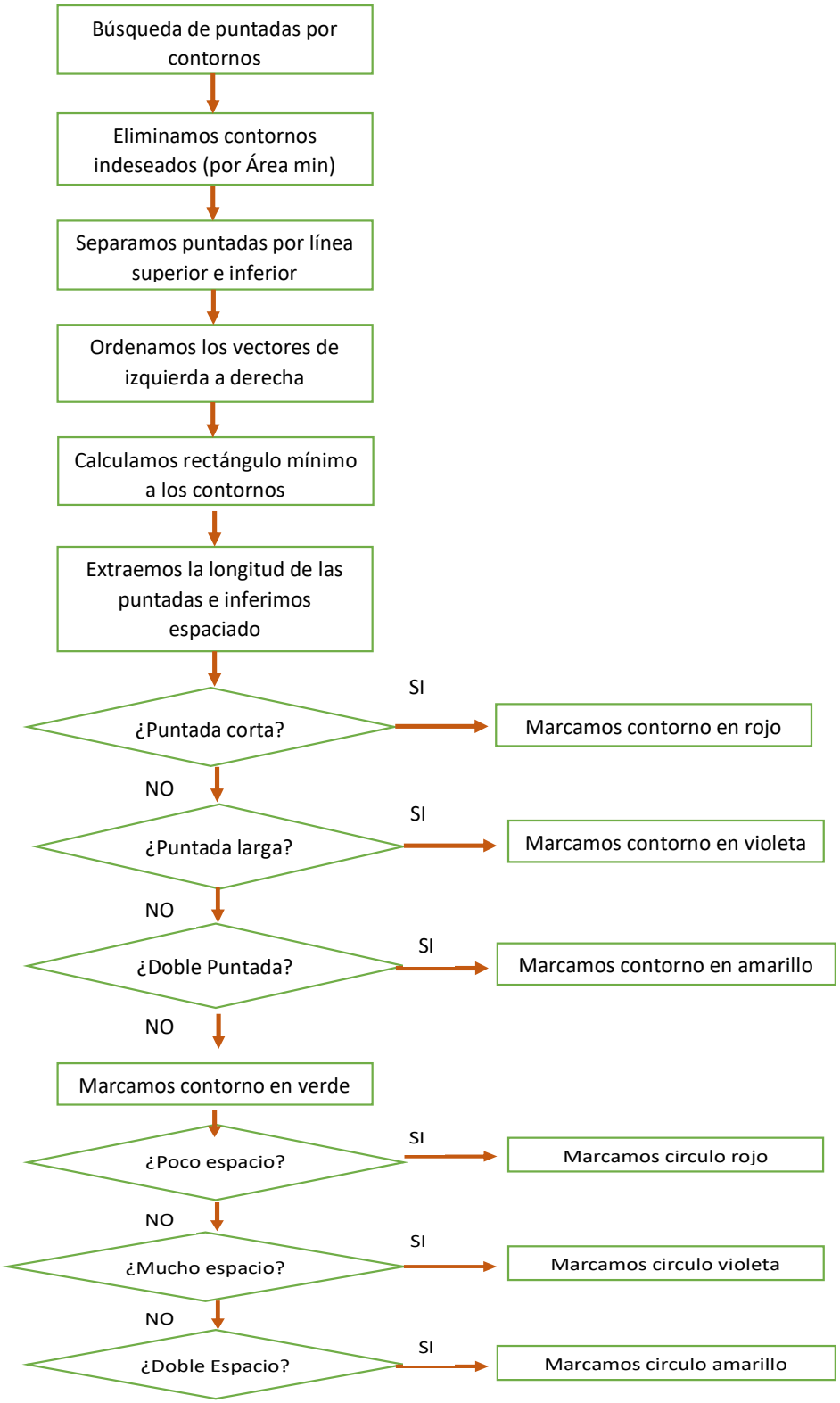
Está función permite filtrar una ventana de valores para pixeles en una imagen. Por ejemplo, está función se utilizaría en el procedimiento para crear un rastreador por color, para obtener como resultado una imagen binaria con el objeto detectado.
Para nuestro propósito el color que tenemos que detectar es el del color del hilo de la puntada que en la imagen diseñada es amarillo, para esto ajustamos la ventana de valores (correspondiente a las componentes que determinan el color amarillo) para la cual se filtraran los pixeles de la imagen. De la aplicación del filtro obtenemos como resultado la imagen binaria, donde los pixeles de la imagen que se encuentran en el rango de valores para los que fueron ajustados, serán transformados en pixeles blancos, y los pixeles cuyos colores se encuentren fuera de la ventana de filtrado serán transformado en pixeles negros.

El rango de colores para el filtrado se ajusta por el modelo HSV (del inglés Hue, Saturation, Value – Matiz, Saturación, Valor), también llamado HSB (Hue, Saturation, Brightness – Matiz, Saturación, Brillo), el cual define un modelo de color en términos de sus componentes. El mismo es una transformación no lineal del modelo RGB.



Como podemos observar, luego de aplicar el filtro por color obtenemos la imagen binaria con respecto a la imagen original quedando así discriminado las puntadas en blanco sin importar del fondo u entono que a estas rodean.

SEGUNDA PARTE: BUSQUEDA DE LAS PUNTADAS



Ahora nos queda desarrollar el algoritmo para localizar las puntadas y poder decidir si son correctas o no.

Primero a partir de la imagen binaria aplicamos una función que nos permite encontrar los N contornos externos e internos que contiene dicha imagen,

findContours(hsv, contours, noArray(), CV_RETR_EXTERNAL, CHAIN_APPROX_SIMPLE, Point(0, 0));

La aplicación de esta función nos entregara como resultado los contornos de las puntadas encontradas dentro de la imagen, almacenándolos en un vector. A su vez este vector estará compuesto por N vectores, correspondiente a los puntos (coordenadas) de cada contorno encontrado.

Algunos parámetros de configuración:

CV_RETR_EXTERNAL: Regresa como salida los contornos externos extremos.
CV_CHAIN_APPROX_SIMPLE: Configura el método para la detección de los contornos. Considera solo los segmentos horizontales, verticales y diagonales dejando sólo sus puntos que forman a dichos segmentos. Un ejemplo, sería un rectángulo, el cual estaría formado por 4 puntos.

Ya con el vector completo por los valores de los contornos de las supuestas puntadas, procedemos a eliminar los contornos encontrados cuya área sea menor a un área mínima establecida por nosotros, a sabiendas que por su condición de área mínima no representa a un contorno de puntada real.
Para corroborar que todos los contornos encontrados corresponden a puntadas de la costura, realizamos la comparación del área de cada uno de los contornos respecto a un área mínima, para la cual un contorno mayor a esta pueda ser considerada como una puntada.
Este condicionamiento nos será de utilidad para cuando apliquemos como imagen de entrada una costura real donde la influencia de la iluminación, pueda variar las condiciones idealizadas para este prototipo y puedan aparecer algunos pixeles de contornos pequeños.

Una vez verificado que todos los valores del vector de contornos pertenecen a puntadas reales tomamos la decisión de dividir dicho vector en dos vectores diferentes, identificando un vector perteneciente para las puntadas de la línea superior y otro vector perteneciente a las puntadas de la línea inferior. Para separar las puntadas que pertenecen a la línea superior de las puntadas que pertenecen a la línea inferior, procedimos a calcular los momentos al vector que contiene todos los contornos encontrados.

Para explicar un poco que se tratan estos, podemos decir que los momentos son una medida particular que indica la dispersión de una nube de puntos, y matemáticamente se definen como:

$$M_{ij} = \sum_{x} \sum_{y} x^i y^j I(x, y)$$

donde x, y son las coordenadas de un píxel y la función I(x,y) indica su intensidad. Estamos trabajando sobre una máscara binaria, por tanto la función I(x,y) sólo puede valer 0 ó 1.

Ahora, ¿cómo puede ayudarnos esta fórmula extraña a saber el centro? Hay tres momentos que nos interesan: M00, M01 y M10. Fijémonos que, si estamos calculando M00, la fórmula anterior se transforma en:

$$M_{00} = \sum_x \sum_y x^0 y^0 I(x,y) = \sum_x \sum_y I(x,y)$$

(recordar que $a^0 = 1$, y aquí definimos $0^0 = 1$).

Como I(x,y) sólo puede dar 0 ó 1, la fórmula de M00 es equivalente al número de píxeles cuyo valor es 1. Por tanto, M00 es el área en píxeles de la región blanca.

Para M10, la fórmula original se transforma en esta:

$$M_{10} = \sum_x \sum_y x^1 y^0 I(x,y) = \sum_x \sum_y x I(x,y)$$

Esto es igual a la suma de las coordenadas x de los píxeles cuya intensidad sea 1. Por tanto, dividiendo M10 por M00 obtendríamos el centroide para la componente x:

$$\frac{M_{10}}{M_{00}} = \frac{\sum_x \sum_y x I(x,y)}{\sum_x \sum_y I(x,y)} = \frac{x_1 + x_2 + ... + x_n}{n} = \overline{x}$$

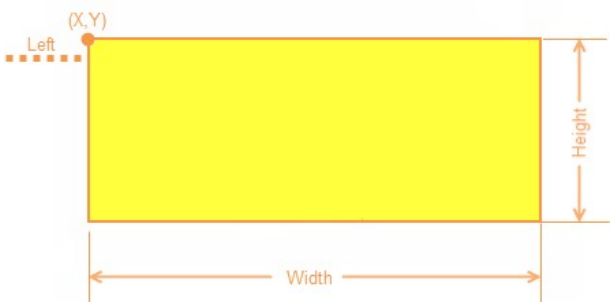
Y lo mismo con M01:

$$\frac{M_{01}}{M_{00}} = \frac{\sum_x \sum_y y I(x,y)}{\sum_x \sum_y I(x,y)} = \frac{y_1 + y_2 + ... + y_n}{n} = \overline{y}$$

Por tanto: para obtener los centroides calcularemos los momentos M00, M10 y M01 de cada contorno y haremos las divisiones anteriores.

Estos nos darán la información necesaria para poder separar las puntadas en estos dos vectores superior e inferior. A su vez aprovechamos ambos vectores, los ordenamos y enumeramos de manera que sean vistos de izquierda a derecha.

Lo siguiente que calculamos es el largo de la puntada y el espaciado entre puntada y puntada, para después dependiendo de su valor tomar la decisión de marcarla como buena o mala según lo establecido. Pero vayamos de a poco, la manera que encontramos de calcular la distancia de la puntada fue utilizando la función *boundingRect.*
Esta función aplicada a cada uno de los contornos del vector, nos calcula el rectángulo mínimo que necesita para encerrar a dicho contorno y devuelve el valor de las coordenadas (x, y) del vértice superior izquierdo, el ancho y su alto propio del rectángulo generado.



De esta manera recorriendo los contornos del vector superior e inferior y aplicando esta función obtenemos de manera directa el valor de la longitud de la puntada con la propiedad del ancho y podemos inferir, teniendo las coordenadas del vértice superior izquierdo de una puntada más el largo de la misma y las coordenadas del vértice superior izquierdo (comienzo de la puntada) de la siguiente puntada, la distancia de espaciado que existe entre una puntada y otra.

Así, realizando los cálculos de los espacios entre puntadas y la longitud de cada una de las mismas, ahora solo nos queda determinar la manera en que tomaremos las decisiones para saber si son aceptables los criterios de calidad o no.

La selección la realizamos primero comparando los espacios entre puntadas, donde identificamos si se presentan alguna de las siguientes posibilidades:

- El espaciado es pequeño por lo tanto las puntadas se encuentran muy juntas.
- El espaciado es grande, por lo tanto, las puntadas están muy separadas.
- El espaciado es demasiado grande (> al doble de la longitud de una puntada), en este caso puede decirse que salto una puntada.

El mismo procedimiento hacemos para el caso de las puntadas, encontrándonos con las siguientes condiciones:

- Longitud de puntada pequeña
- Longitud de puntada larga
- Longitud de puntada muy larga (> al doble de una puntada), pude ser que se zafó una puntada de abajo.

Tanto el espaciado como las puntadas son comparadas con respecto a los valores definidos como tolerables en las que deben encontrarse estas para pasar el control de calidad de la costura.

Otro condicionamiento que encontramos son los casos en que las puntadas se encuentran a los límites de bordes tanto derechos como izquierdo de la imagen. Para esta condición lo que se evalúa para saber si esa puntada debe de ser analizada o no, es calcular el primer punto perteneciente al contorno de la puntada que coincide con el inicio de la imagen (x=0), en este caso podríamos inferir que la puntada no se encuentra completa para la imagen que fue tomada, por lo tanto, será descartada. Por el contrario, si el primer punto del contorno de la puntada se encuentra separada una distancia prudente del inicio del margen de la imagen capturada en ese caso si será tenida en cuenta y analizada con el resto de las puntadas. Este mismo procedimiento será efectuado para las últimas puntadas de la imagen que se encuentran en el borde derecho. Una salvedad seria que estemos detectando que la puntada no está totalmente incluida en la imagen, pero aun así vemos que es muy grande, en este caso, si es tenida en cuenta para informar que hay un problema y que podría representar cierta gravedad.

De la misma forma analizamos los espacios en los bordes. Si estos son pequeños no son tenidos en cuenta ya que podrían no estar completos. Pero en el caso que el espaciado en el borde sea muy grande (mayor al espaciado normal entre puntada y puntada) lo debemos considerar ya que dependiendo de la longitud de ese espaciado podría representar que se salteo una puntada, que no hay más hilo o este se cortó y de no considerarlo en el análisis nos estaríamos perdiendo de la detección de esta falla grave

Las diferentes condiciones mencionadas serán marcadas en la imagen original según corresponda al error que se detecte. A continuación, detallamos las referencias para cada caso:

Para la separación entre puntadas:

Poco espacio → Círculo Rojo
Mucho espacio → Círculo Violeta
Falta una puntada → Círculo Amarillo

Longitud de la puntada:

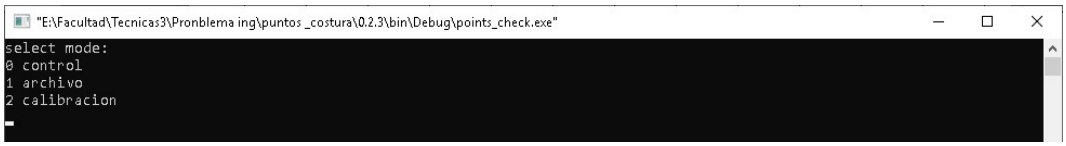
Puntada Normal → Contorno Verde
Puntada Corta → Contorno Rojo
Puntada Larga → Contorno Violeta
Más de 2 untadas juntas → Contorno Amarillo

En la misma imagen, además de marcar los posibles errores encontrados, se mostrará información sobre la cantidad de puntos en la imagen, un promedio de la longitud de las puntadas (PromP) y los espacios (PromE), la cantidad de errores tanto en puntadas (EP) como en espacio entre puntadas (EE), y también una estimación la longitud en la que se encuentran la cantidad de puntadas mostradas.

Para ver la implementación del programa mostramos algunos resultados con diferentes tipos de patrones modificados para poder ver diferentes resultados

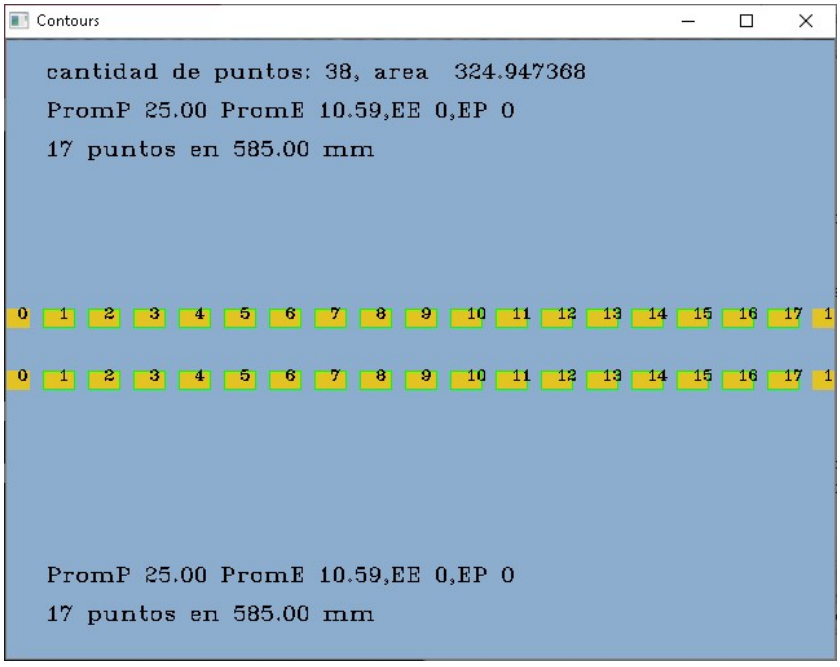
Primeramente, para acceder al análisis de las imágenes y funciones de calibración se provee de un menú en consola al ejecutar la aplicación.
La primera opción a ingresar será para elegir el archivo de configuración de la tela o puntada que se pretenda analizar. La opción 0 se reserva para los patrones ideales. Con cualquier otro número indicamos la configuración deseada. En la carpeta del programa, esta se encuentra con el nombre x_conf.txt, donde x es el número de la configuración. Para este ejemplo que veremos a continuación elegimos el archivo de configuración para los patrones ideales.

Luego nos encontramos con el menú principal



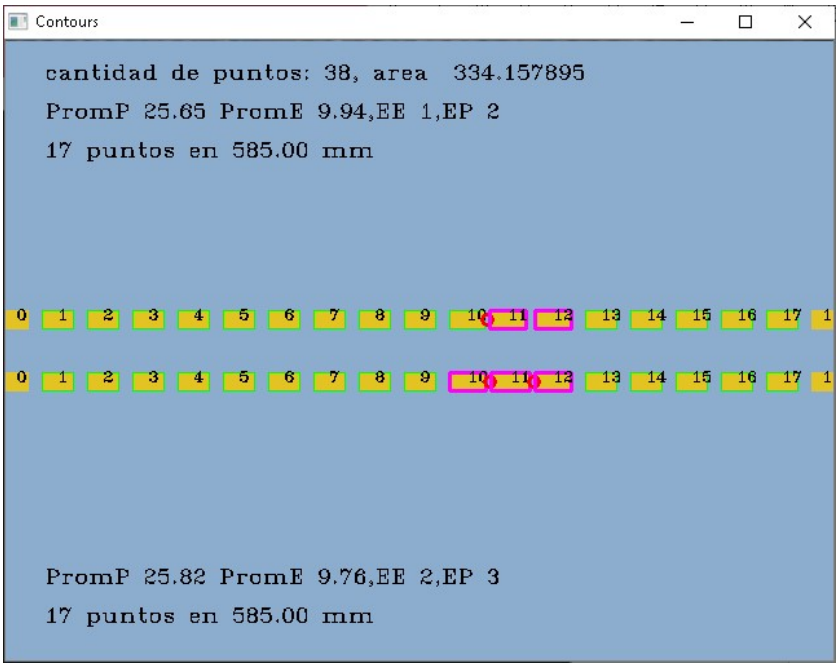
0 Permite ingresar la ruta de un directorio donde previamente hallamos colocado todas las imágenes que nos interesan analizar cuyos nombres deben ser números a partir de 1(1.png).
1 Esta opción permite analizar una única imagen ingresando la ruta y nombre del archivo.
2 Aquí se encuentran las funciones de calibración. Una vez seleccionado pedirá ingresar otra opción según sea para calibrar dimensiones (opción 0) o el filtro de color (opción 1).

En este primer caso procesamos la imagen de una costura ideal donde todas las puntadas y espacios entre ellas son buenas, en la misma imagen podemos ver la información extraída durante el procesamiento de la imagen

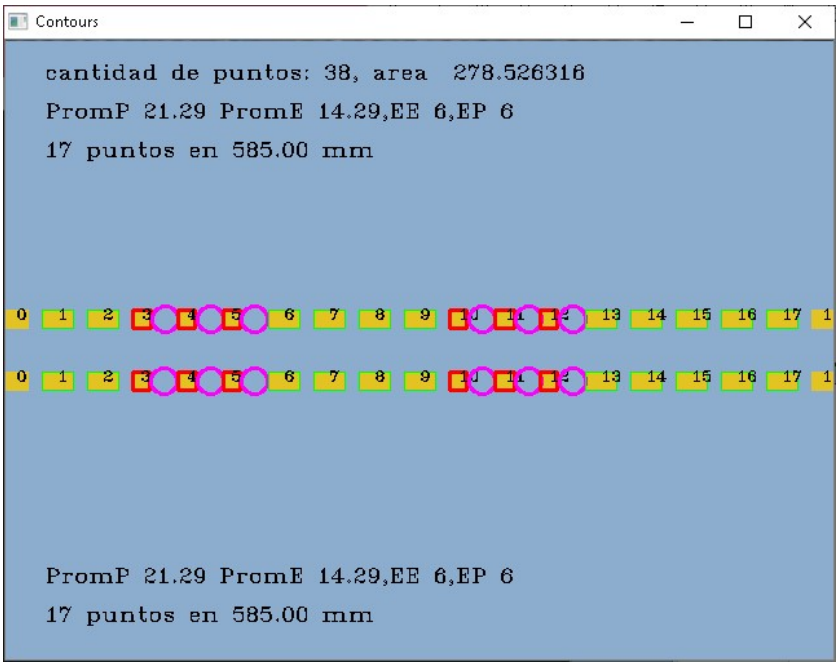


Modificando la imagen anterior de forma de representar posibles errores que podrían aparecer en las costuras, la segunda imagen a analizar fue una costura donde encontramos que el espacio entre un par de puntadas es demasiado próximo, por lo cual estas puntadas se encuentran muy juntas.

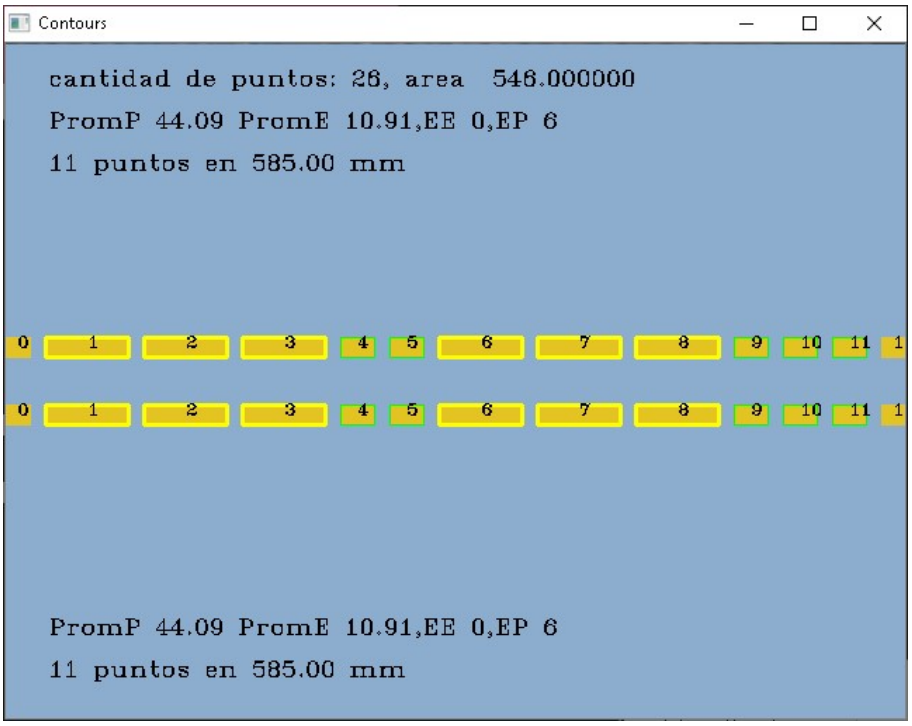
En el análisis nos arrojó que en la línea de costura superior tenemos 2 errores de puntadas (EP) marcadas de color rosa lo cual nos referencia que las puntadas son largas y además 1 error de espaciado (EE) identificado por un punto rojo indicando espaciado muy pequeño.
Las mismas referencias para la línea de costura inferior donde 3 errores de puntadas (EP) y 2 errores de espaciado (EE).
Además, podemos ver la cantidad de puntos marcados en verde que representan los puntos correctos.



En esta otra imagen podemos ver una costura donde se presentan puntadas marcadas en rojo donde hace referencia que se trata de puntadas muy cortas y también espacios entre puntadas grande donde se marcan con círculos rosas sobre la imagen.



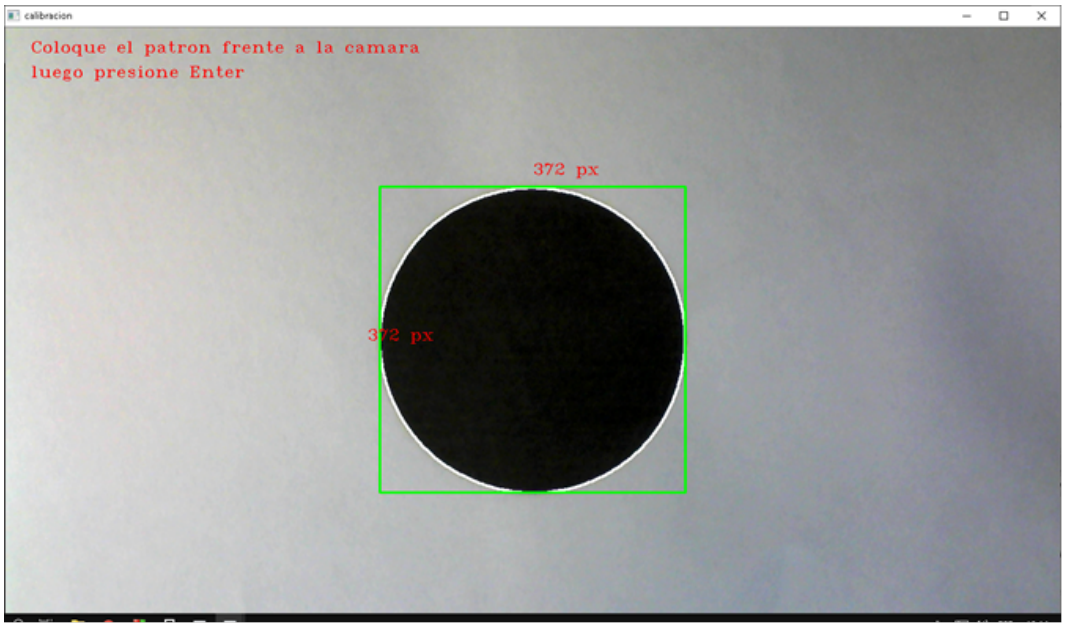
Por último, analizamos un tipo de costura donde se propuso modelizar que la puntada no engancho, es decir, no existe espacio entre puntadas o en otra forma de verlo sería como si se tratase de una puntada doble por su largo. De esa manera en el análisis de la misma podemos ver como se ve marcada un error de este tipo mediante su demarcación en un color amarillo.



RESULTADOS DE LAS PRUEBAS

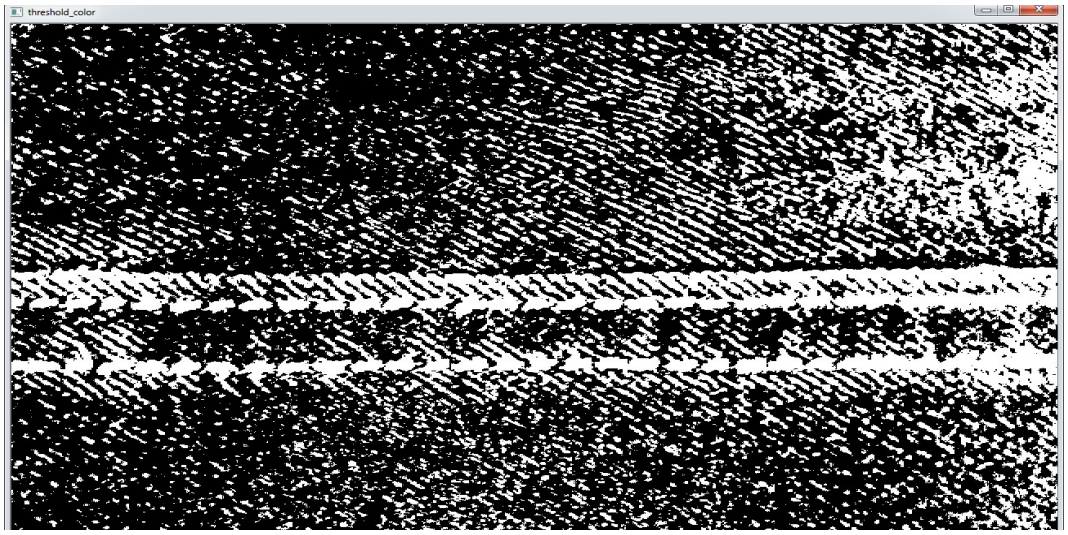
Para realizar las pruebas reales y ver los resultados, tomamos varias fotos de una costura similar a la que habíamos creado como una imagen patrón ideal para el prototipo. Para tomar una secuencia de fotos de la tela con la costura de nuestro interés, colocamos la cámara en una posición fija de manera idéntica a como se encontraría colocada en la máquina de coser durante el control de la costura y con una iluminación adecuada.
Luego, una vez colocada la cámara en posición y haber tomado unos cuantos fotogramas de muestra de dicha costura, realizamos la calibración de la cámara tanto de los pixeles por mm, como también la configuración de los parámetros ideales para el filtro por color tomando como referencia para calibración la tela a procesar.

Para la calibración de los pixeles por milímetros que la cámara reconocerá, el programa nos pedirá que coloquemos un objeto patrón de dimensiones conocidas previamente, entonces una vez colocado el patrón conocido delante de la cámara, el algoritmo de calibración calculara las dimensiones del objeto en pixeles. Para nuestro caso utilizamos un circulo de 35mm de diámetro. Podemos observar a continuación en la imagen nos arroja que el circulo tiene un diámetro de 372 pixeles, entonces una vez que le demos “Enter” para validar los pixeles del objeto reconocido frente a la cámara, nos pide al usuario que ingrese el diámetro real del circulo en mm, en nuestro caso 35mm. Así obtenemos el valor en mm por cada pixel de la imagen. En este ejemplo serian 372px /35mm=10,65px por mm. Por último, nos indica si deseamos guardar estos valores presionando “Enter” o si queremos cancelarlo con Escape.

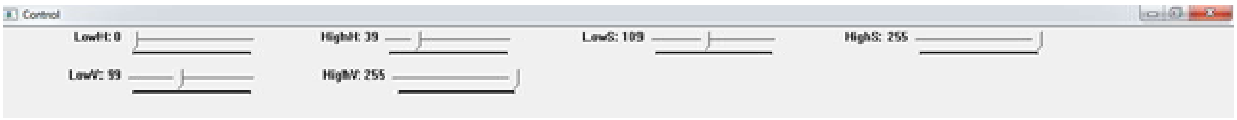


Después sigue la calibración de las componentes que corresponden al filtro por color, y para realizar la misma utilizamos una de las imágenes de muestra capturada por la cámara. Así podremos calibrar de manera óptima el filtro teniendo como referencia el color de hilo usada en costura y que será la que queremos discriminar del resto de la imagen. Entonces cargamos la imagen con la que trabajaremos la cual vemos a continuación:

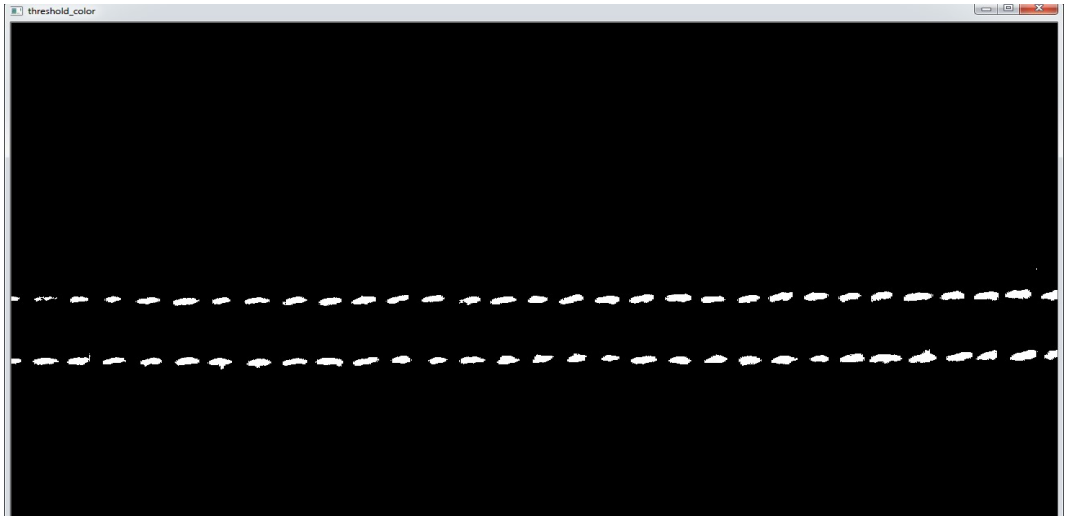
A la imagen original se le aplica el filtrado por color con las componentes HSV (Matiz, Saturación y Valor) por defecto ya cargadas y a la salida del filtro da como resultado la imagen que puede observarse abajo.



Lo que el usuario debe realizar es un ajuste de configuración a su preferencia de las componentes H, S, V mediante sus respectivas barras deslizables que le permitirán ir variando los valores de dichas componentes. Estas trackbars aparecerán agrupadas en una ventana denominada control.



La modificación de cada componente se verá reflejada inmediatamente en la imagen que muestra la salida del filtro, donde podremos ver cómo afectan los cambios de las respectivas componentes y si el filtro está respondiendo de la manera que pretendemos. Observamos en la siguiente imagen, la respuesta del filtro una vez encontrado los valores óptimos para su configuración.



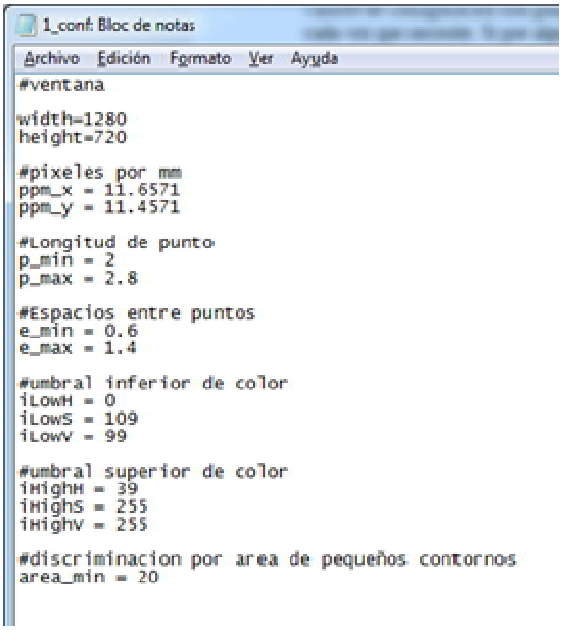
Teniendo los valores apropiados para el filtro óptimo, le damos “Enter” y estos parámetros se guardarán. Caso contrario para salir de la calibración aplicamos “Escape” y saldremos de la misma.

Cabe destacar que este paso de calibración tanto de los pixeles por milímetro como de la configuración para el filtro solo se realiza la primera vez para la puesta en marcha del mismo. Estos valores de configuración son guardados en un archivo que el programa utilizara como referencia cada vez que necesite. Si por alguna razón la posición de la cámara varia o hubo un cambio en el

color de hilo utilizado en la costura, ahí si debería de volverse a calibrar por verse modificado las condiciones iniciales establecidas en la puesta en marcha anterior.

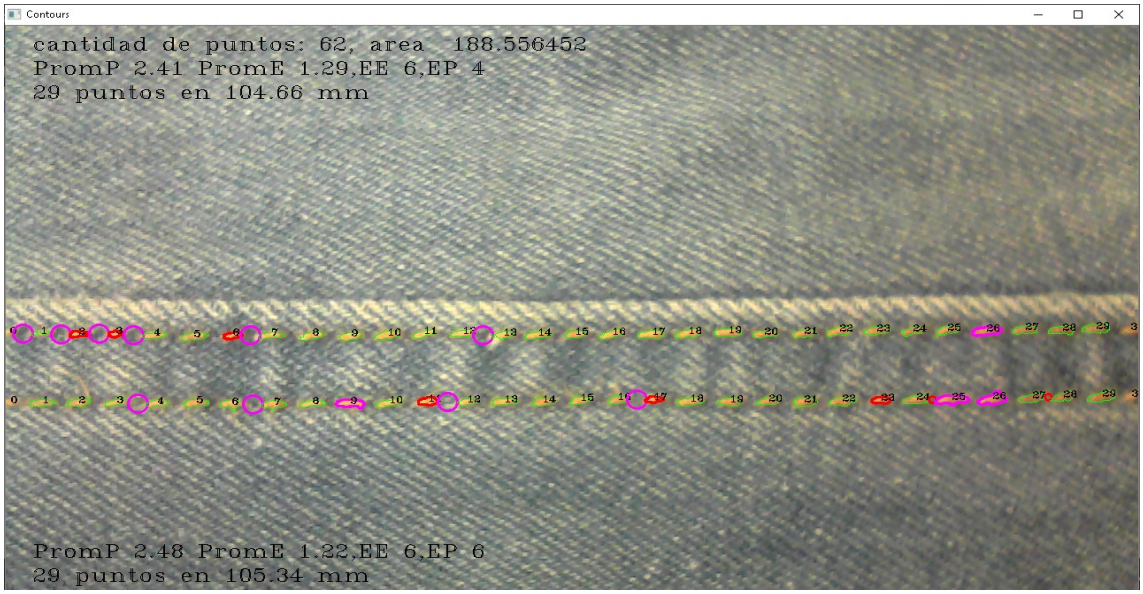
Dentro del archivo de configuración además podemos definir:

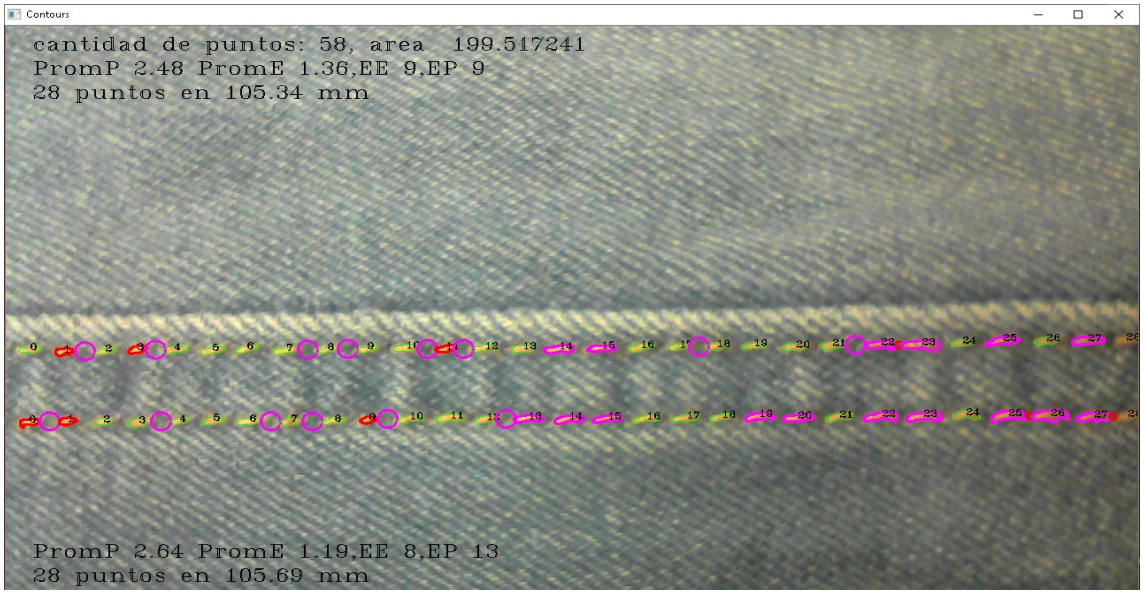
- **Resolución imagen capturada**
- **Pixeles por mm**, estos son guardados en el archivo cuando hacemos la calibración de dimensiones.
- **Longitud de punto**, establecemos el rango de tolerancia para una puntada normal.
- **Espacios entre puntos**, rango de tolerancia en que el espacio es aceptable.
- **Umbrales inferiores/superiores de color**, son los cargados durante la calibración del color del hilo.
- **Discriminación de área pequeña**, es un área mínima por la cual detecta contornos de esa dimensión, son discriminadas ruido



Terminado ya la parte que representaba la calibración, pasamos a la parte más importante referida al procesamiento de las imágenes.
Para esto como mencionamos antes, tomamos varias imágenes de la tela con las costuras que analizaremos. Al tener varias imágenes lo que pretendimos hacer fue simular la detección de la cámara situada en la máquina de coser, donde la cámara capturara una imagen correspondiente a las puntadas. Esta captura de imágenes se determinará mediante un sensor magnético ubicado en la máquina de coser que contara las veces que baja la aguja (por ende, en condiciones normales la cantidad de puntadas que se aplican sobre la tela) y sabiendo cual es el largo promedio de la puntada podremos estimar cuantas nuevas puntadas se originaron y así sacar un nuevo fotograma cada vez que se dé la señal. De esta manera evitamos que la cámara este procesando en todo momento hasta en momentos que no serían necesario, como en caso que el operador de la máquina de coser no esté realizando ninguna costura.
Igualmente, esta parte del problema se nos escapa de nuestro alcance que habíamos establecido, por tal motivo es que para simular esa situación decidimos hacer de forma manual las tomas de imágenes, y así originar una secuencia de fotogramas que representaría esa situación relatada.

A continuación, podemos ver unas imágenes de los resultados obtenidos luego de aplicar la secuencia de fotogramas con la tela de puntadas utilizadas para simulación del caso real en la detección del control de las costuras, donde se observan buenos resultados obtenidos si se comparan los resultados obtenidos para las imágenes de una puntada idealizada con los del análisis de una costura real.
Puede observarse que la imagen que la tela sobre la que se encuentra la costura no es homogénea, sino que tiene variación en su entramado pasando por colores claros a oscuro. Esto podría favorecer a confundirse con la detección de las puntadas, por eso el filtro por color en este caso nos facilita muchísimo esto ya que, filtrando por el color del hilo, que en este caso es de color naranja se puede distinguir las puntadas del resto del fondo que contiene la imagen.





El resto de las secuencias de imágenes que se analizaron se pueden encontrar en el anexo.

CONCLUSIONES

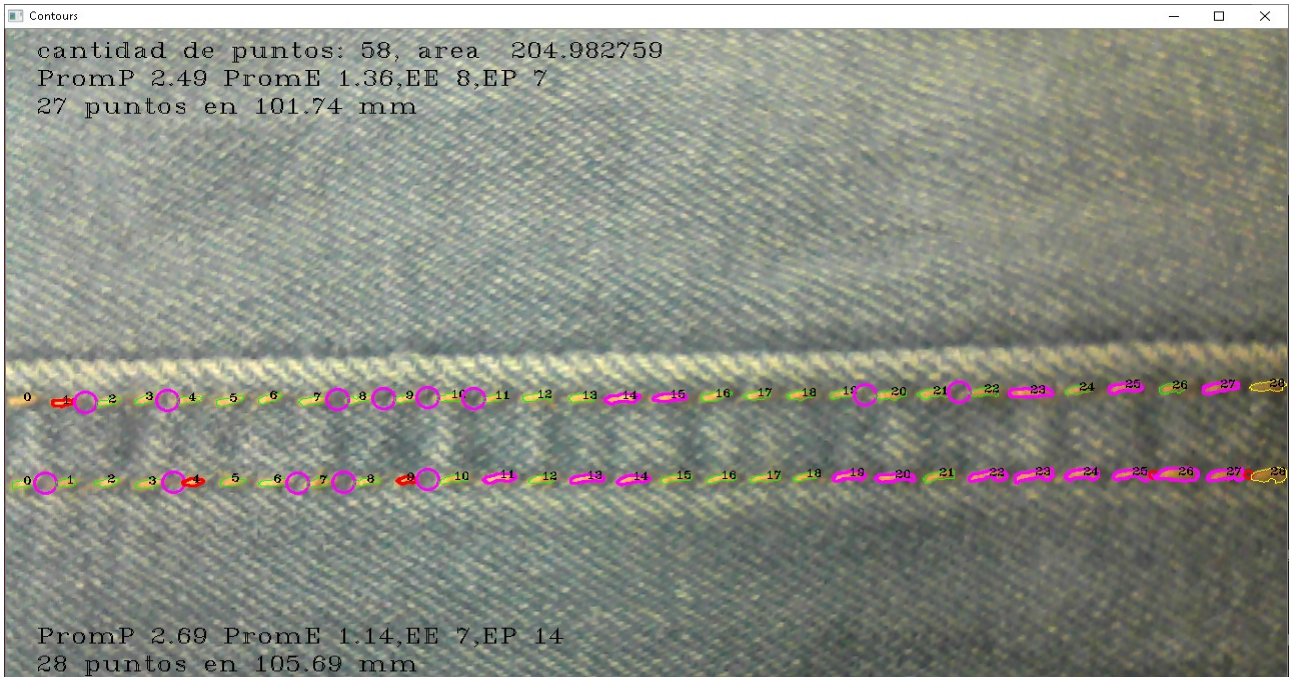
En conclusión, el software OpenCV es una herramienta poderosa y flexible para el análisis y tratamiento de imágenes, que al ser libre permite a cualquier programador ya sea de C/C++ o Python con cualquier entorno de programación, realizar programas para alguna aplicación determinada, ya sean en tiempo real, como de post y pre procesamiento, volviéndose una opción interesante para implementar en muchísimo tipo de aplicaciones

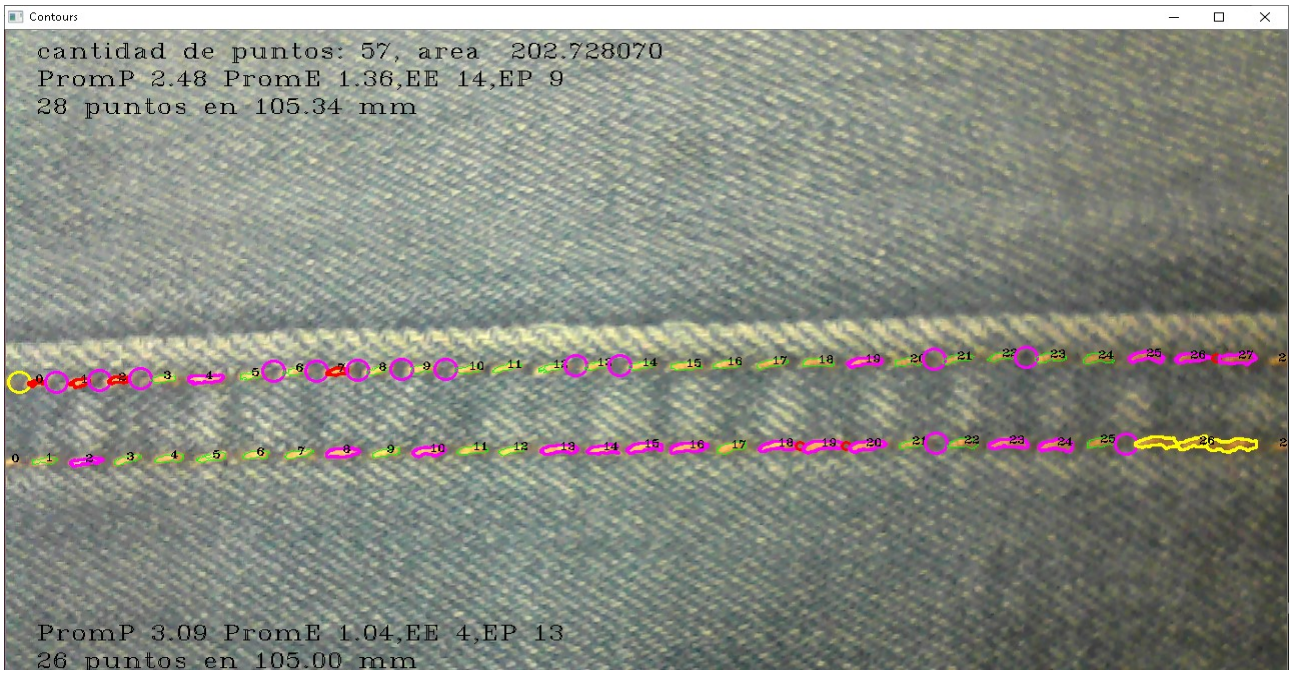
En nuestro caso descubrimos que en la rama de visión artificial existen muchísimas maneras y alternativas para desarrollar un proyecto u problema, por las numerosas funciones de gran potencia que existen para el procesamiento de imagen, por lo que significa una ardua dedicación y estudio para establecer que solución será la óptima para adoptar.
No obstante, hoy en día este campo de inteligencia artificial está creciendo rápidamente y por esto hay muchísimos foros colaborativos y blogs donde puede encontrarse de gran cantidad información de interés para nuestra búsqueda.

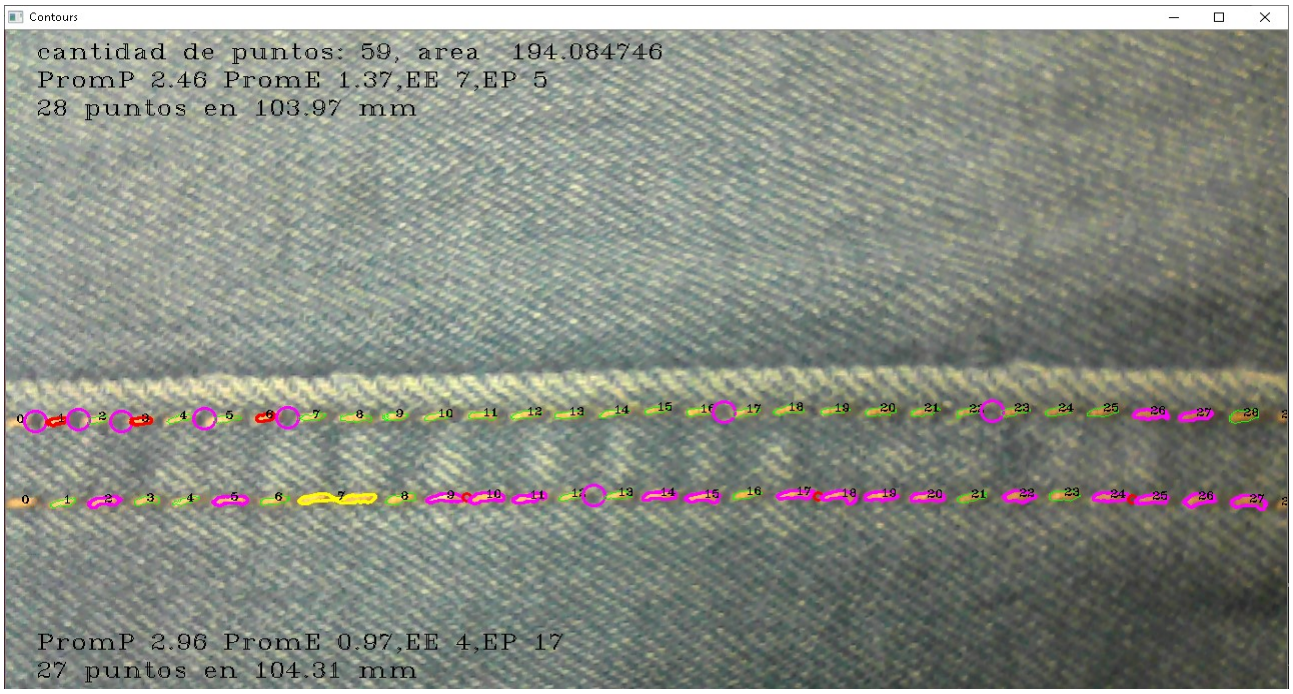
ANEXOS:

FOTOS DE PANTALLA

Adjuntamos más imágenes de la secuencia de fotogramas que fueron procesados a partir de la tela de costura.







ANEXO VIDEO CONSTRUCCION DE PARACAIDAS

En el siguiente link se puede ver el video documental producido por el canal Discovery Max donde se evidencia el problema de ingeniería propuesto:
¿Cómo se hace?: Paracaídas - Discovery Max
https://www.youtube.com/watch?v=D5SpJ2biKag (Proceso de revisión visual de la costura a partir del minuto 1:50)

ANEXO ARCHIVOS DEL PROGRAMA

En el siguiente enlace pueden encontrarse los archivos correspondientes al proyecto desarrollado, incluyen archivos código fuentes, ejecutable, imágenes patrones e imágenes reales.

https://drive.google.com/open?id=1dfe2_7QPXQyuEKWYBIBNeoKU8UdYnCtb