

Trabajo Final Inteligencia Artificial – año 2019

Universidad Nacional de Cuyo – Facultad de Ingeniería

Proyecto: Clasificación de piezas metálicas por visión artificial

Nombre y apellido: Carlos Alberto Bustillo López

Legajo: 11586

Carrera: Mecatrónica

Fecha: Junio del 2020

RESUMEN

Este proyecto de visión artificial consiste en la clasificación de piezas metálicas (tornillos, clavos, tuercas y arandelas). El programa consta de una etapa de: Adquisición de imágenes (a través de una caja cerrada con fondo blanco, con iluminación frontal y comunicación por IP entre cámara y computadora); Transformación (para tener una imagen normalizada); Adaptación y preprocesamiento; Filtración (se utilizo filtro Gauss y Sobel) y segmentación; Extracción de rasgos (se utilizo Momentos de Hu) y reducción; Base de datos y rendimiento; Clasificación de objetos con KNN y KMeans. Se definió un agente racional que aprende, no omnisciente y que a partir de los algoritmos KNN y KMeans se entrena, aprenda partir de sus percepciones y le permita ser autónomo gracias a la experiencia adquirida. Se utilizo el lenguaje de programación Python 2.7. Como objetivo se busca obtener un agente que en un entorno con condiciones controladas se desempeñe con un alto rendimiento.

INTRODUCCIÓN

Tipo de Agente

Es un agente que aprende. El aprendizaje permite que el agente opere en medios inicialmente desconocidos y que sea más competente que si sólo utilizase un conocimiento inicial. En este caso aprende debido a los métodos que utiliza KNN y KMeans. La base de datos le proporciona un “modelo” de su entorno, también los algoritmos en base a la “utilidad”.

También el agente es racional eso implica: “En cada posible secuencia de percepciones, un agente racional deberá emprender aquella acción que supuestamente maximice su medida de rendimiento, basándose en las evidencias aportadas por la secuencia de percepciones y en el conocimiento que el agente mantiene almacenado”.

También es un agente no omnisciente debido a que no conoce el resultado de su acción y no actúa de acuerdo a él. La idea es que a partir de los algoritmos KNN y KMeans se entrene, aprenda partir de sus percepciones y le permita ser autónomo gracias a la experiencia adquirida.

Para este proyecto el agente es una computadora donde están alojados los algoritmos KNN y KMeans. Los datos de entrada y salida son adquiridos por cámaras fotográficas, monitor, etc.

REAS

El acrónimo REAS proviene de Rendimiento, Entorno, Actuadores y Sensores, todo ello forma parte de lo que se llama Entorno de trabajo.

Rendimiento	-Confiabilidad de las predicciones con un margen de error pequeño. -Cantidad de predicciones correctas de las imágenes que se están evaluando. -Categorización de imagen correcta.
Entorno	-Es el contexto plasmado en la imagen de entrada. Ampara factores como la iluminación, calidad de la imagen, fondo, etc.
Actuadores	-Es el medio de notificación del usuario donde se muestra la predicción del algoritmo. En este caso es el monitor de la computadora, también podría ser una alarma u otro tipo de alarma. -Visualizar la categorización de una imagen.
Sensores	-Cámara fotográfica. -Extensión Fatkun Batch para obtener las imágenes de internet. -Módulos skimage, cv2, numpy, matplotlib, etc. -Matriz de píxeles de colores

Propiedades del entorno de trabajo

Totalmente observable	Los sensores previamente mencionados le proporcionan acceso al estado completo del medio en cada momento.
Determinista	El estado del medio está totalmente determinado por el estado actual y la acción ejecutada del medio. La imagen de entrada una vez tomada es inalterable en principio.
Episódico	La elección de la acción en cada episodio depende el episodio en sí mismo. Uno puede tomar las fotografías en cualquier orden de los

	tornillo, clavos, arandelas o tuercas y el rendimiento del agente no se vera afectado.
Estático	El entorno no cambia cuando el agente está deliberando. Una vez tomada la fotografía, ese dato de entrada se mantiene inalterable.
Discreto	Las imágenes captadas por las cámaras son discretas, en sentido estricto, pero se tratan típicamente como representaciones continuas de localizaciones e intensidades variables.
Individual	El único ente racional es el agente previamente descrito.

DISEÑO DEL SISTEMA

Reconocimiento de objetos

El reconocimiento de objetos es la tarea de encontrar e identificar automáticamente objetos específicos en una imagen. Actualmente no existe un sistema artificial “reconoce-todo”.

Un sistema de reconocimiento de objetos generalmente esta constituido por las siguientes etapas básicas:

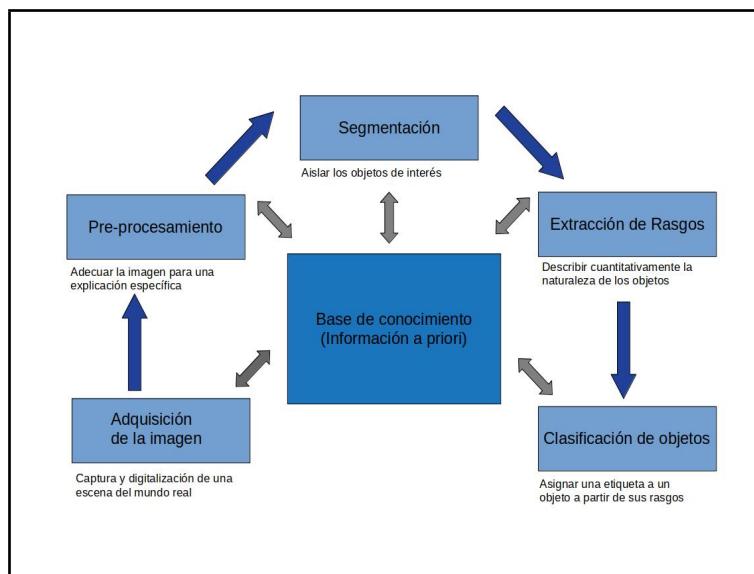


Ilustración 1: Esquema de modelo general.

Desarrollo del sistema reconocimiento de objetos

Problema: Reconocer automáticamente 4 objetos de ferretería distintos (tornillos, tuercas, arandelas y clavos).

Características notables de las imágenes: Ruido, objetos contrastados de el fondo, iluminación heterogénea (zonas más brillantes que otras), diferentes posiciones y localizaciones.

Propuesta de solución: Basado en las etapas básicas de un sistema de reconocimiento de objetos se obtuvo lo siguiente:

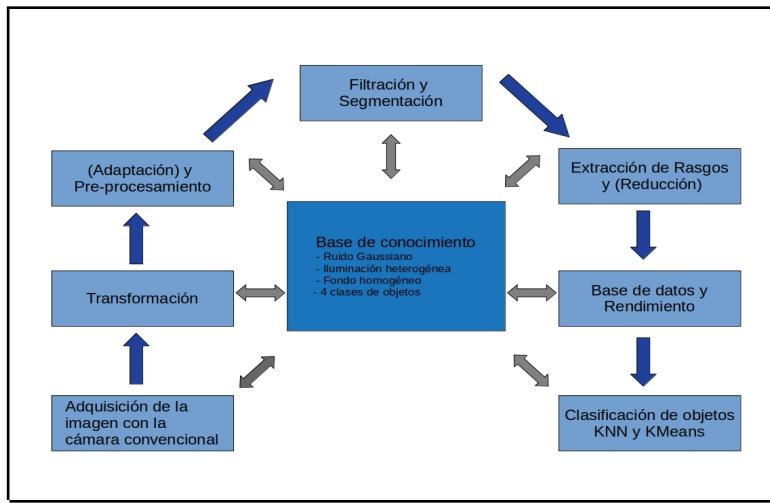


Ilustración 2: Propuesta de modelo para este proyecto.

Breve descripción de objetos de estudio

-Tornillo

Los podemos encontrar de muchas variedades de materiales, tipos y tamaños que existen para distintas aplicaciones. En él se distinguen tres partes básicas: cabeza, cuello y rosca.

Una primera clasificación puede ser: Tornillos tira fondos para madera, autorroscantes y autoperforantes para chapas metálicas y maderas duras, tornillos tira fondos para paredes y muros de edificios, tornillos de roscas cilíndricas y varillas roscadas de un metro de longitud.

Pueden ser de acero dulce, inoxidable, latón, cobre, bronce o aluminio, y pueden estar galvanizados, niquelados, bicromatados, etc.

Otra clasificación sería desde el punto de vista de utilización: Tornillos para usos generales, en miniatura, de alta resistencia, inviolables, de precisión, de titanio, etc.

-Tuerca

Es una pieza mecánica con un orificio central, el cual presenta una rosca, que se utiliza para acoplar a un tornillo, en forma fija o deslizante.

Las características básicas para identificar a una tuerca son: Número de caras, grosor, diámetro del tornillo que encaja en ella y el tipo de rosca.

Tipos de tuercas: Tuerca mariposa o de ala, hexagonal, con rebordes, ciegas, cuadradas, T, etc.

-Arandelas

Es un elemento de montaje con forma de disco delgado con un agujero usualmente en el centro (corona circular), siendo su uso más frecuente el sentar tuercas y cabezas de tornillos. Las arandelas normalmente son de metal o de plástico.

Tipo de arandelas: Plana, de presión, dentada, cónica, de cuero, etc.

-Clavos

Es un objeto delgado y alargado con punta filosa hecho de un metal duro (por lo general acero), utilizado para sujetar dos o más objetos.

Los clavos se clasifican de acuerdo con su uso, el diámetro, acabado y longitud.

Podemos clasificarlos, según el tipo de cabeza, de punta o de cuello: clavo de cabeza ancha, clavo de escarpia, clavo para tornillos, clavo de acero, etc.

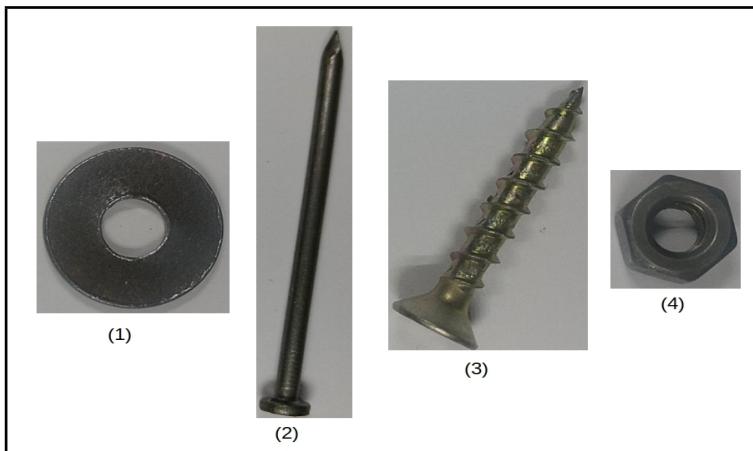


Ilustración 3: Piezas de ferretería utilizados en este proyecto:

(1) Arandela de acero común, con $d_1=6$ mm y $d_2=18$ mm; (2) Clavo de acero común $d=4$ mm y $L=50$ mm; (3) Tornillo para madera autoperforante zincado $d=5$ mm y $L=30$ mm; (4) Tuerca hexagonal $d=5$ mm, hexágono = 11 mm.

Adquisición de la imagen con cámara convencional

La primera etapa, dentro de un proceso de visión computacional es la etapa de adquisición. En este primer paso, se trata de conseguir que la imagen sea lo más adecuada posible para que se pueda continuar con las siguientes etapas. Una correcta adquisición de la imagen supone un paso muy importante para que el proceso de reconocimiento tenga éxito. Dentro de esta etapa existen múltiples factores que atanen directamente al proceso de captura de la imagen, formados fundamentalmente por: el sistema hardware de visión artificial (cámara, óptica, tarjeta de adquisición de imagen, ordenador y software) y el entorno y posicionamiento de los elementos (la iluminación, el fondo, posición correcta de la cámara, ruido eléctrico-óptico externo, etc.).

-Técnicas de iluminación

Un entorno debidamente controlado es imprescindible para obtener unas condiciones de partida optimas que aseguren una perfecta adquisición. Dentro del entorno aparecen como partes fundamentales en las expectativas tanto en calidad como de la imagen buscada: la iluminación, el fondo, la posición de la cámara, etc.

1) La iluminación

La iluminación de la escena tiene que realizarse de una forma correcta, dada la importancia que tiene. Existen fundamentalmente dos formas de iluminación: Iluminación frontal o trasera, y cada una de ellas tienen sus variantes.

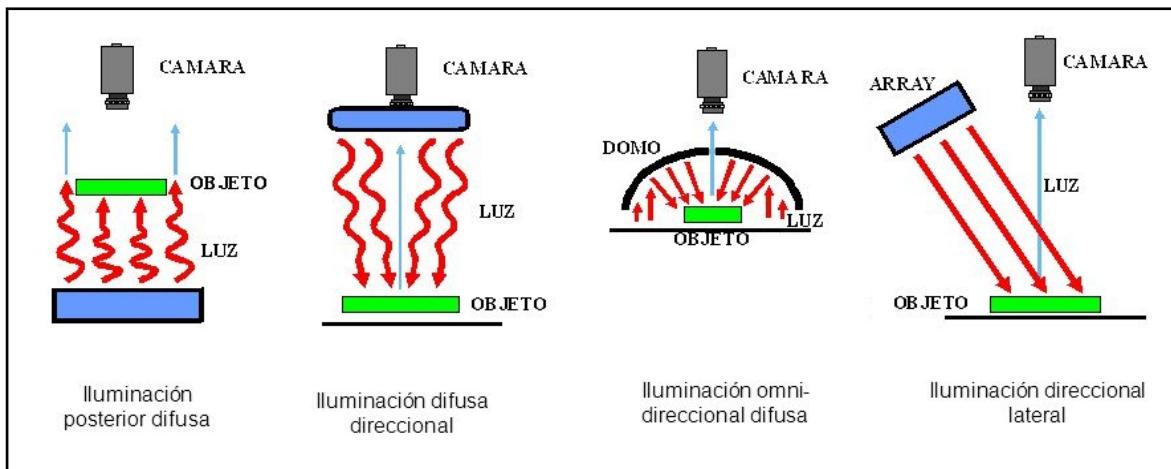


Ilustración 4: Imagen obtenida de: Presentación “Visión por computadora” por Gustavo Cerezo

La iluminación frontal permite distinguir claramente los detalles del mismo, pero esta iluminación puede presentar diferentes inconvenientes como: la creación de sombras y los reflejos. La mejor forma de evitar estos dos efectos, consiste en utilizar luz difusa ya sea con fuentes especiales que generan luz difusa (por ejemplo fibras ópticas que expanden la luz en todas direcciones), por el uso de lámparas circulares de luz que iluminen de forma homogénea o por el uso de luz indirecta. Por otro lado, como el sistema va a trabajar en un entorno industrial, es conveniente evitar las posibles interferencias externas (por ejemplo que las lámparas no estén totalmente fijas o que exista un juego entre la cámara y el soporte), también se usarán paredes que reduzcan lo más posible la entrada de luz ambiental.

2) Fondo

El fondo de la escena cumple un papel esencial cuando se trata de simplificar alguna de las etapas subsiguientes (como puede ser la segmentación). Éste debe ser lo más homogéneo posible y de un color que permita distinguirlo fácilmente de los objetos. Cualquier mancha o defecto que existan en el fondo, puede ocasionar errores en la etapa de reconocimiento.

Utilizando iluminación frontal, el fondo debe de ser lo más opaco posible evitando todo reflejo. El color negro opaco suele ser el más utilizado.

Para este proyecto, se empleó iluminación direccional frontal, que consiste en colocar la cámara apuntando al objeto e iluminándolo en la misma dirección de la cámara, con un bajo ángulo de incidencia. De este modo la cámara recibe la mayor parte de la luz reflejada por el objeto, además se seleccionó iluminación Led (dos focos de diodos emisores de luz) con fondo negro blanco. Descartamos el uso de fondo negro porque no se obtuvieron buenos resultados con él.

-Cámara

Para obtener una imagen correcta sin deformación, es necesario efectuar un perfecto calibrado.

Para la calibración de la posición de la cámara del celular se utilizó la app “Nivel de Burbuja”, que utiliza el acelerómetro y giroscopio (en función del modelo de smartphone) para ayudar a alinear / nivelado , suspender horizontal, vertical o en cualquier ángulo para ajustar cada objeto.

Para este proyecto consideramos $x=0.4^\circ$ y $y=0.2^\circ$.



Ilustración 5: Calibración de la posición de la cámara del celular con la ayuda de la app Nivel de burbuja.

También se podría calibrar mediante una plantilla de puntos, determinando los centros de gravedad de cada círculo y luego las dimensiones de cada uno para constatar que la posición es correcta.

Para tomar las fotografías se utilizó la cámara del celular Motorola Z2 Force. La cámara trasera cuenta con doble lente de 12 Megapíxeles con sensores IMX 386 y f/2.0, color + monocromo.

-Sistema de captación de imágenes

Para evitar las variaciones de las condiciones ambientales y de las distancias de medición, se decidió utilizar una atmósfera controlada mediante una caja cerrada tapizada de negro en su interior, cuenta con dos focos led de 3.5 W cada uno conectados en paralelo conectada a un interruptor para realizar la iluminación frontal. También cuenta con un soporte para el celular paralelo a la base, donde irá el objeto de ferretería a analizar.



Ilustración 6: Sistema de captación de imágenes. Consiste en una caja negra forrada con cartulina negra excepto en la base que tiene cartulina blanca, la cual se cierra al momento de tomar las fotos.

Utilizamos la app “IP Webcam” que convierte el celular en una cámara en red con múltiples opciones de visualización. Funciona en cualquier plataforma con VLC player o navegador web. Emite a través de una red WiFi sin conexión a internet. Con ésto, hacemos que nuestro celular se convierta en servidor, y mediante la dirección IPv4 la ingresamos en el buscador de la computadora, manejamos todo desde ahí y guardamos directamente la foto en nuestra computadora.

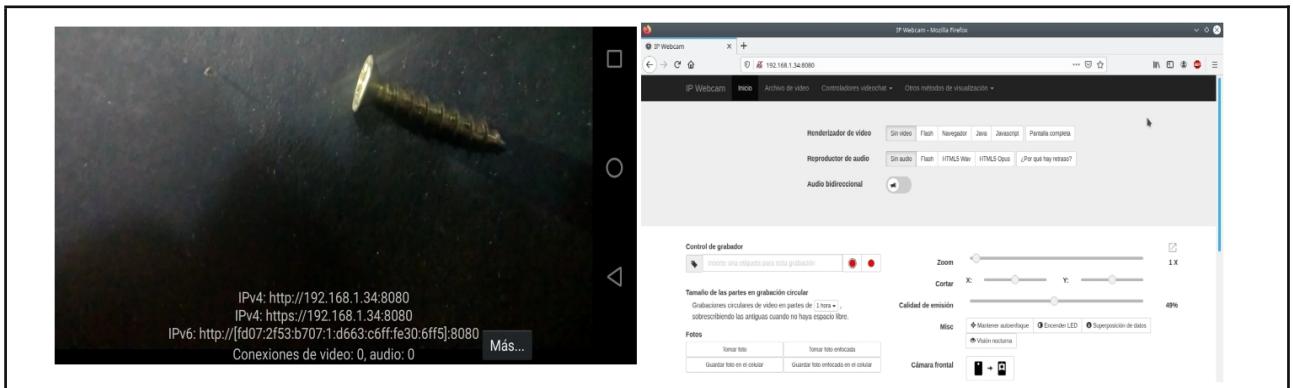


Ilustración 7: Captación de las imágenes con el uso de la app IP Webcam.

Transformación

En esta etapa se cuenta la cantidad de objetos en la imagen, se recorta la imagen si es necesario y se hace una redimensión para tener un formato de imagen normalizado . Todo ésto para tener un estándar de imagen de entrada para la etapa de pre procesamiento.

-Quitar línea de la pared

Al momento de tomar la imagen, debido a la posición de la cámara dentro de la caja toma una parte de la pared perpendicular a la base donde se encuentra el objeto, por eso se hace un recorte y eliminar la franja.

Para eso eliminamos 700 px de izquierda a derecha de la imagen, pasamos de tener una imagen de 1920x1080 a 1220x1080.

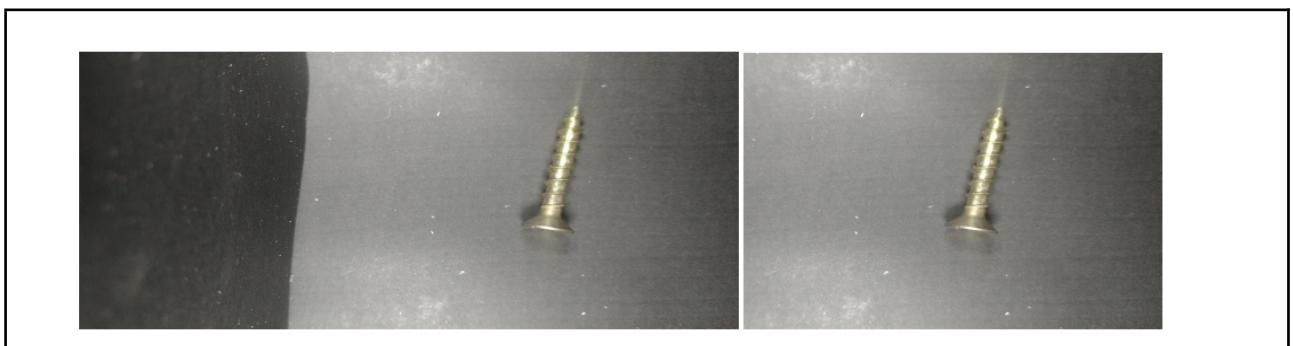


Ilustración 8: Remoción de franja de pared de la caja dentro de la imagen.

-Redimensionamiento de las imágenes

Las imágenes de entrada están en formato png o jpg, las cuales presentan distintos tamaños. Es muy importante trabajar con las mismas dimensiones, porque los descriptores visuales se verán modificados por esta variación como veremos más adelante.

Luego de quitar la franja de pared de la imagen, tenemos una imagen de entrada de 1220x1080 que tiene una estructura de (1080,1220,3) y a la salida tenemos una imagen cuya estructura es de (400,500,3) que es el formato que vamos a utilizar para este proyecto. Ésto último representa una imagen de 500x400 y las 3 capas correspondientes al RGB.

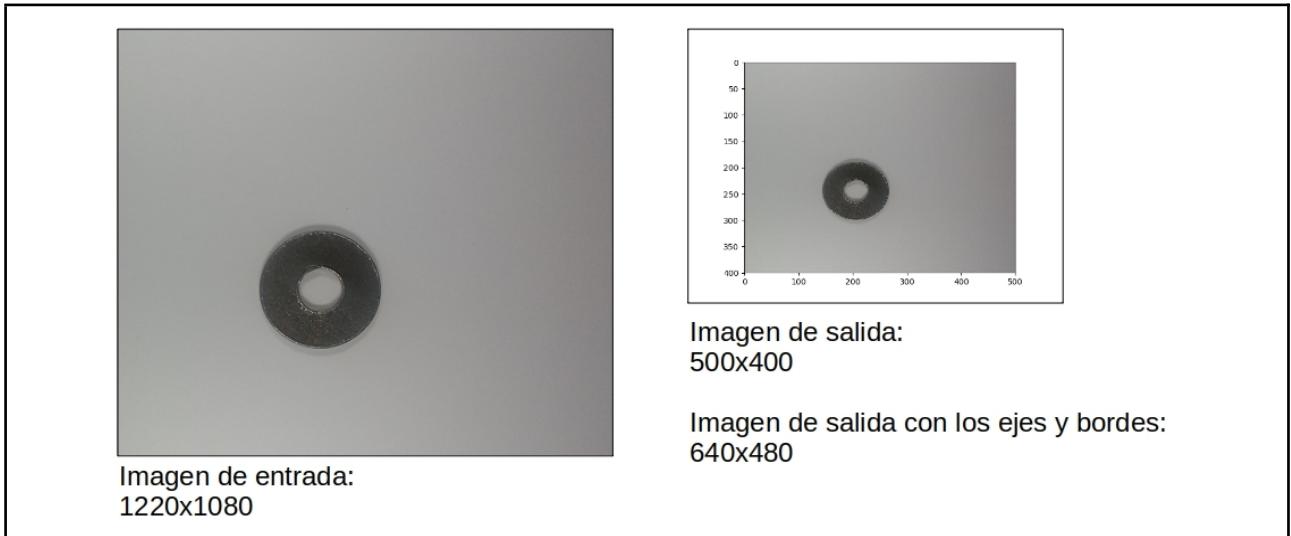


Ilustración 9: Redimensión de la imagen.

-Quitar fondo y convertirlo a uno totalmente blanco

Uno de los problemas que se presenta en la caja donde toman las fotos es que se presenta una sombra que se aprecie muy bien cuando se aplique una segmentación de thresholding, una etapa que veremos más adelante. Esa sombra envuelve a la pieza de ferretería y hace que se pierde entre la sombra. La solución propuesta es cargar la imagen y crear en paralelo una máscara color gris (específicamente tiene la tupla RGB: 100,100, 100 este dato lo obtuve haciendo uso de la página web: www.pinetools.com), con prueba y error se determinó los valores del rect que permite extraer la pieza de ferretería del fondo. Una vez realizado eso, se toma la máscara cortada menos la imagen original y eso dio lugar a una imagen con fondo totalmente blanco. Prácticamente lo que hizo el programa es convertir los píxeles en blanco menores a aquellos píxeles con valores menores a la tupla RGB 100,100,100. Así como convierte los píxeles del fondo también lo hace con la pieza de ferretería, por eso se pierde algo de información de la pieza por esta etapa.

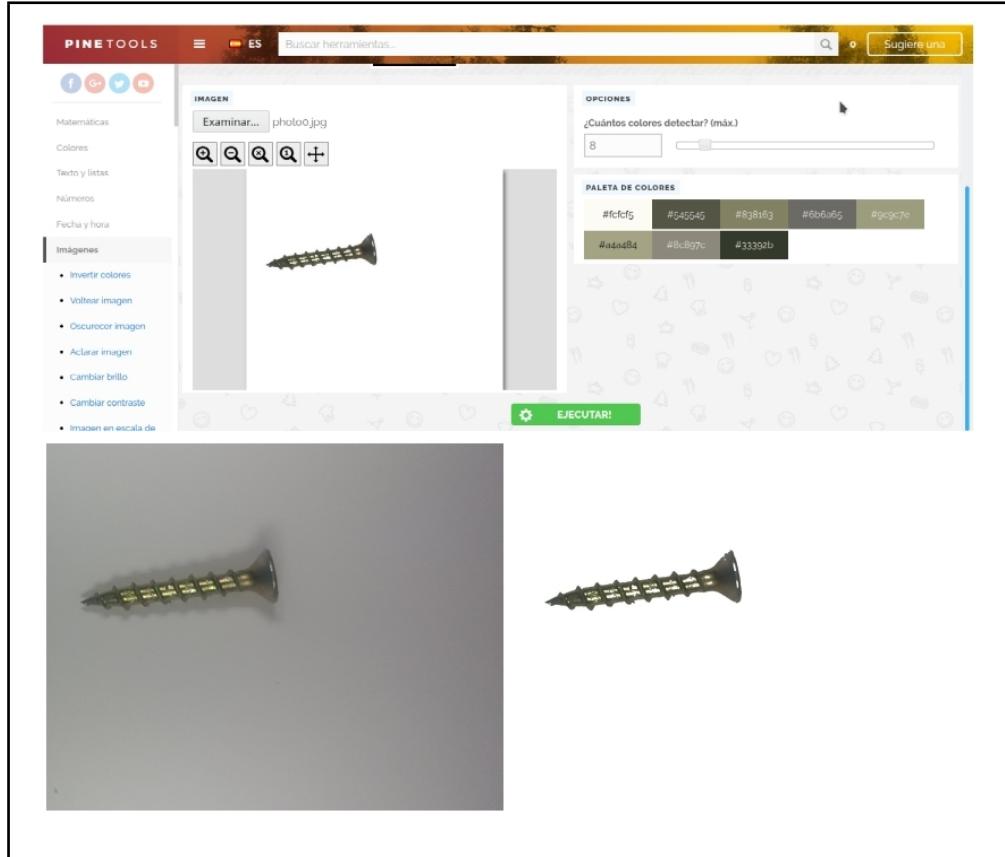


Ilustración 10: (1)En la imagen superior se aprecia como la detección de colores con la página pinetools.(2) En la imagen inferior izquierda, se muestra la imagen sin la pared de la caja con fondo normal.(3) En la imagen inferior derecha, imagen con fondo totalmente blanco.

Adaptación

Esta etapa es opcional. En un principio fue planteada la idea de contar los objetos dentro de la imagen, luego fraccionar esa imagen para cada objeto encontrado y luego recortarla alrededor del objeto encontrado para su posterior análisis.

Una gran ventaja que permite realizar el crop de la imagen, es reducir el espacio de búsqueda al momento de evaluar los algoritmos KNN y Kmeans más adelante. Ésto reduce los tiempos de ejecución significativamente. Pero la cuestión incide que al armar la base de datos con imágenes cortadas para mi algoritmo no tuvo un gran rendimiento respecto a no cortarlas, por eso planteo su implementación aunque posteriormente al momento de armar la base de datos no la utilicé.

-Contar cantidad de elementos para un posterior recorte de los elementos dentro de la imagen
Aplicamos dos métodos distintos para realizar el conteo:

1) Algoritmo de Canny para detectar contornos

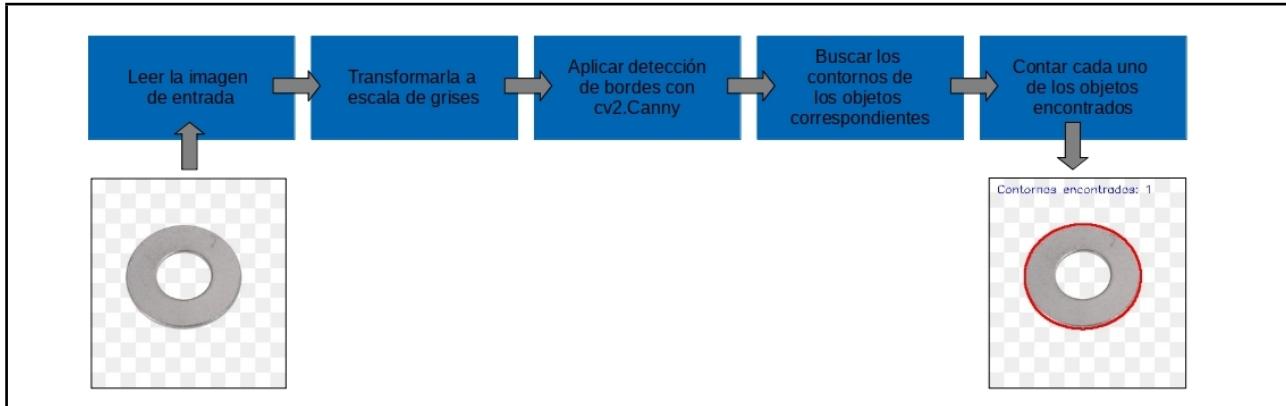


Ilustración 11: Proceso de detección de bordes.

El algoritmo de Canny es un operador que utiliza un algoritmo de múltiples etapas para detectar una amplia gama de bordes en imágenes, cuyo propósito es descubrir el algoritmo óptimo de detección de bordes. Para que un detector de bordes pueda ser considerado óptimo debe cumplir los siguientes puntos: buena detección, buena localización y respuesta mínima.

Cv2.Canny nos exige una imagen de entrada, primer umbral (minVal) y segundo umbral(maxVal). Para que los bordes se dibujen o no, se necesita de dos umbrales, minVal y maxVal. De este modo, todos los valores superiores se dibujaran. Los bordes que estén dentro del maxVal y minVal se dibujaran solo si se encuentran conectados a los que superan maxVal. Por el contrario todos los bordes debajo de minVal no se dibuja.

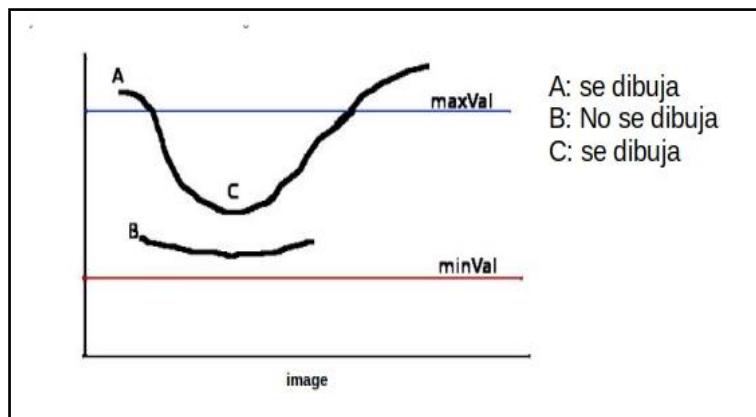


Ilustración 12: Imagen obtenida de la librería OpenCv sobre funcionamiento del algoritmo de Canny.

Una vez que tengamos claro el uso de estos dos umbrales podemos hacer pruebas para analizar como se van dibujando los distintos bordes dada una imagen. Se presentaron una serie de resultados a distintos valores de umbrales: Para el caso de una arandela con umbrales ($\text{minVal}=100$, $\text{maxVal}=200$) la detectaba bien, pero ese valor para múltiples arandelas o tornillos contaba mucho más de lo que debería, debido a la complejidad de varios objetos en una imagen o la cantidad de filetes en el tornillo. Para éstos casos, utilice un rango de umbrales más amplio ($\text{minVal}=10$, $\text{maxVal}=800$); si ese mismo rango de umbral tan grande se lo aplica a una sola arandela nunca logra encontrarla, y remarca el contorno de la imagen.

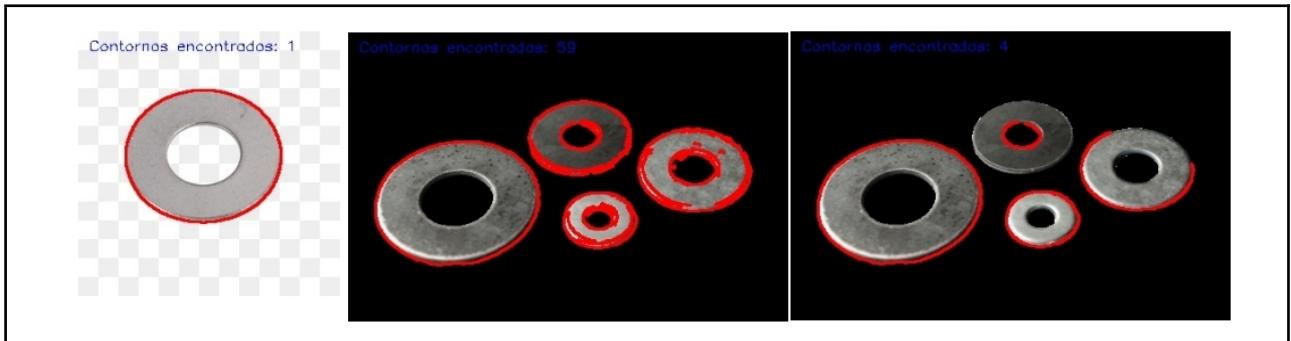


Ilustración 13: Resultados de detección de bordes con diferentes tipos de umbrales.

La cantidad de contornos encontrados equivale a la cantidad de objetos dentro de la imagen. Es clave saber el número de objetos, porque mi algoritmo de KNN y Kmeans, como veremos más adelante, solo evaluá un objeto por imagen.

Conclusión: Si se prueba este algoritmo con imágenes de computadora (sin brillos, ni sombras, perfectamente planas, fondo homogéneo, etc) el contorno es continuo, y entra en vigencia que la cantidad de contorno equivale a la cantidad de objetos. Debido a que los bordes no son continuos, esa correspondencia no se cumple, pero seguimos utilizando este programa, no como contador de objetos, sino que ahora con el mismo podemos detectar objetos, luego recortarlos y extraerlos de la imagen.

2) Uso de umbralización simple (Thresholding) para detectar contornos

No siempre con Canny se obtuvieron buenos resultados para las distintas piezas de ferretería, por eso también utilice este método.

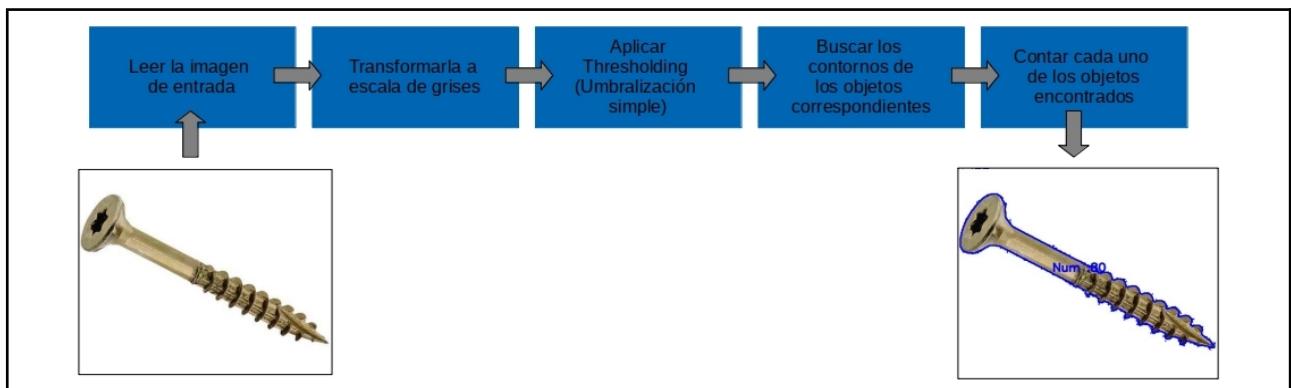


Ilustración 14: Proceso de detección de objetos con umbralización simple.

Ya que el fondo de la imagen que utilizamos es blanco, usamos cv2.THRESH_BINARY_INV, para obtener el fondo negro, de este modo obtendremos un modo binario, que nos servirá para encontrar los contornos de la imagen. Determinaremos los momentos de la imagen con cv2.moments para luego encontrar los centroides de los mismos.

$$c_x = \frac{M_{10}}{M_{00}} \quad c_y = \frac{M_{01}}{M_{00}} \quad \text{donde } M \text{ denota los momentos.}$$

Esta forma de encontrar el contorno, es muy buena para el caso de los tornillos para remarcar todos los filetes y demás características del tornillo, pero al utilizar la cantidad de contornos como

parámetro para contar, no es muy preciso porque existen muchas discontinuidades entre filetes y cuenta de más. Para el caso de las arandelas, tiene problemas con el formato .png, cuenta los objetos y el fondo.

La conclusión de utilizar un contador de objetos a partir del número de contornos detectado en la imagen, es bastante complejo porque: Se necesitaría una imagen de entrada .jpg, con fondo blanco y uniforme, la cual fue tomada de forma perpendicular a la pieza de ferretería, y los parámetros de los programas de Adaptacion.py deberían ser definidos *a priori*, ya que varia para cada pieza analizada.

-Recortar imagen a partir del Algoritmo de Canny

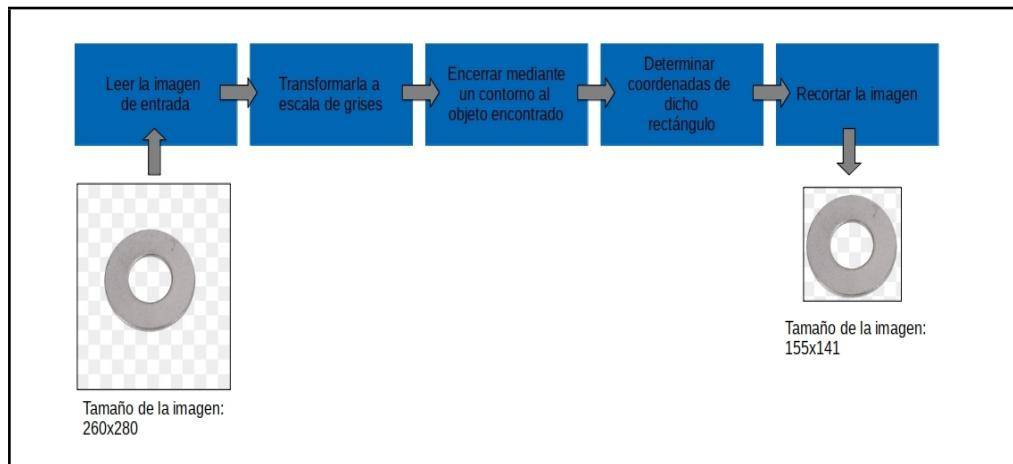


Ilustración 15: Se hace uso del algoritmo Canny para el recorte de la imagen.

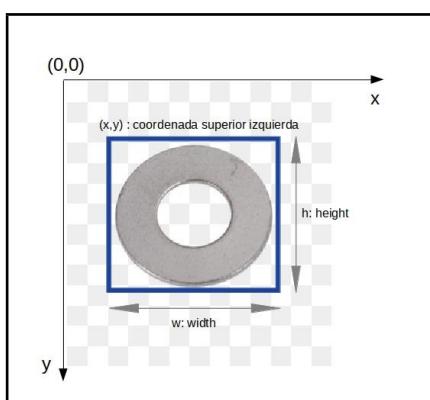


Ilustración 16: Sistema de referencia para el corte.

Para esta etapa, también utilizamos el algoritmo de Canny. Se determina el contorno, para establecer los bordes en la imagen. Se determina las coordenadas del rectángulo que encierra al objeto en la imagen, esas coordenadas son el punto (x,y) de la esquina superior izquierda, la altura y el ancho del objeto. Y con éstos datos nos permite hacer el crop de la imagen.

Conclusión: la imagen de entrada debe tener fondo uniforme y homogéneo. Si se realiza el corte de la imagen nuevamente hay que normalizarla a 500x400.

Pre-procesamiento

En el pre procesamiento se adecua la imagen para una aplicación específica, en este caso para extraer información de clavos,tornillos,arandelas o tuercas de la imagen normalizada para poder analizarla.

-Descomposición de la imagen en una tupla de valores R, G y B(Representación)

Una forma de obtener información relevante para el estudio de la imagen es descomponerla en sus 3 capas RGB donde queda al final una matriz con los correspondientes colores rojo, verde y azul. Sería una tupla de tres enteros en la imagen a color o simplemente un entero para el caso de la imagen a escala de grises. Estamos trabajando con imágenes de 8 bits, las cuales representan sus valores en un rango de 0 a 255 en números enteros, este rango también es conocido como Rango Dinámico.

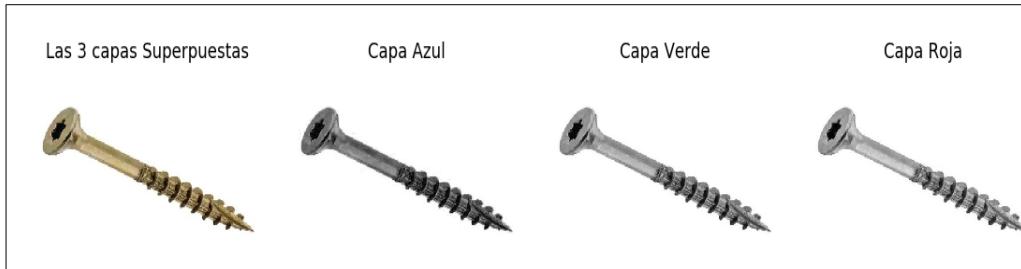


Ilustración 17: Descomposición de capas de una imagen. Se puede apreciar que las capas verde y rojo tienen mayor intensidad mientras que la capa azul es más opaca u oscura.

-Conversión de imagen original a escala de grises

Esta conversión se realiza calculando un equivalente “E” formado a partir de los tres planos de la imagen a color. En su forma más sencilla se establece este equivalente como el promedio, es decir:

$$E(x, y) = \frac{R(x, y) + G(x, y) + B(x, y)}{3}$$



Ilustración 18: Conversión de imagen RGB a escalas de grises utilizando pesos WG, WR y WB arbitrarios ingresados manualmente. Para este caso cada peso es $\frac{1}{3}$.

Como varia la iluminación que se tiene en las imágenes presenta una subjetiva iluminación, propia del modelo RGB hace que utilizando el promedio de las imágenes con un valor grande en la componente rojo y/o verde tengan una apariencia oscura. El efecto contrario sucede donde el contenido del plano azul es grande, mostrando en su versión a escala de grises una apariencia muy clara. Con el objetivo de solventar ésto se considera como una mejor aproximación calcular una combinación lineal de todos los planos, como ponerlo en función de sus pesos, definida como:

$$E(x, y) = WR * R(x, y) + WG * G(x, y) + WB * B(x, y)$$

Donde WR, WG y WB son los coeficientes que definen la transformación. Los cuales varían de acuerdo el método considerado. Por ejemplo:

WR=0.299 WG=0.589 WB=0.114 Método utilizado para señales a color en TV

WR=0.2125 WG=0.7154 WB=0.072 Método ITU-BT.709 para codificación digital a color

WR= $\frac{1}{3}$ WG= $\frac{1}{3}$ WB= $\frac{1}{3}$ Se obtiene una imagen más oscura

Una consideración de importancia es la distorsión Gamma producida en las señales de TV que afecta de manera no lineal que se resuelve con los siguientes pesos:

WR=0.309 WG=0.609 WB=0.082

Al hacer una escala de grises mediante la combinación lineal se obtiene imágenes menos oscuras que utilizando el promedio.

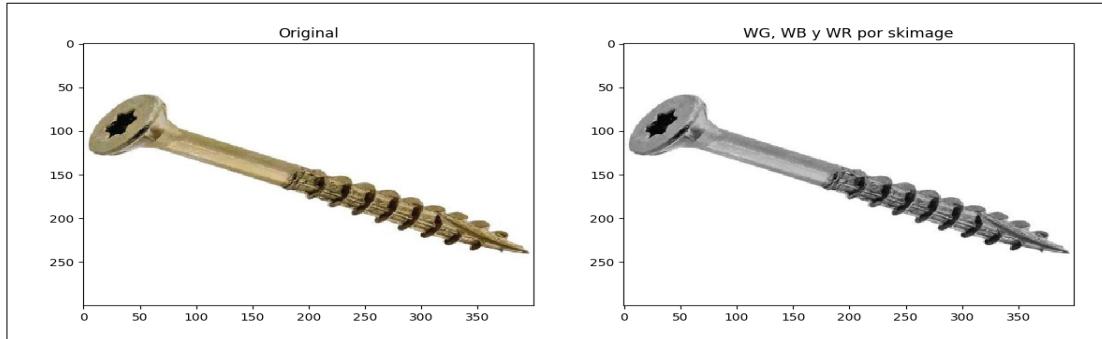


Ilustración 19: Para este proyecto los pesos WR, WG y WB los proporciona el módulo de skimage, lo que hace esa función es calcular la luminancia de una imagen RGB.

Filtración

El objetivo de la filtración es suavizar la imagen, eliminar ruido, realzar la imagen y detectar bordes.

Los filtros digitales constituyen uno de los principales modos de operar en el procesamiento de imágenes digitales. Pueden usarse para distintos fines, pero en todos los casos, el resultado sobre cada píxel depende de los píxeles de su entorno.

Una imagen se puede filtrar en el dominio del espacio, trabajando directamente sobre los píxeles de la imagen, o en el dominio de la frecuencia, donde las operaciones se llevan a cabo en la transformada de Fourier de la imagen .

-Filtros en el dominio del espacio

Las operaciones espaciales de filtrado se definen en un entorno de vecindad del punto a transformar (x,y).

Los filtros en el dominio del espacio pueden clasificarse en:

1. Filtros lineales (filtros basados en máscaras de convolución).
2. Filtros no lineales.

-Filtro Gaussiano (Filtro lineal)

Este filtro se usa para embozzonar imágenes y eliminar ruido.

Modelizando la función gaussiana:

$$G(x, y) = \frac{1}{2 * \pi * \sigma^2} \exp\left(\frac{-(r^2 + c^2)}{2 * \sigma^2}\right)$$

Donde “r” son el número de filas desde el centro y “c” el número de columnas desde el centro.

Al aplicarlo sobre un tornillo de nuestra base de datos, se puede modificar alguno de los parámetros (por ejemplo σ) para ver como varia el filtrado de la imagen.

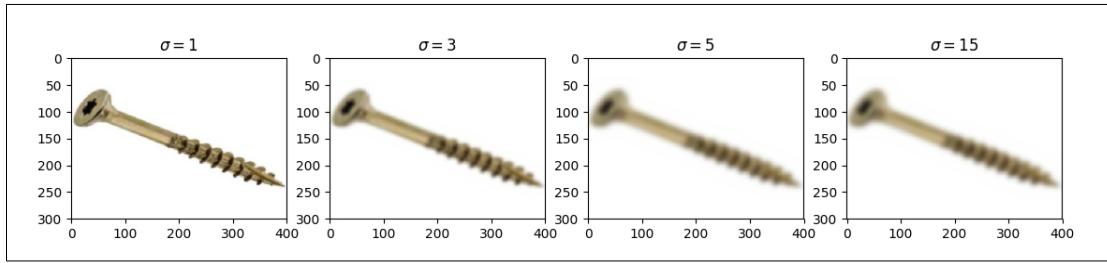


Ilustración 20: Puede observarse que si σ aumenta demasiado, la imagen comienza a hacerse borrosa.

-Filtro Sobel y Roberts

El filtro Sobel detecta los bordes horizontales y verticales separadamente sobre una imagen en escala de grises. Las imágenes en color se convierten en RGB en niveles de grises. El resultado es una imagen transparente con líneas negras y algunos restos de color.

El filtro Roberts obtiene buena respuesta ante bordes diagonales, fácil y rápido de computar. Ofrece buenas prestaciones en cuanto a localización. El gran inconveniente de este operador es su extremada sensibilidad al ruido y tiene una respuesta débil a los verdaderos bordes, a menos que sean muy pronunciados, por tanto tiene pobres cualidades de detección. Para este propósito funciona mejor el operador Sobel.

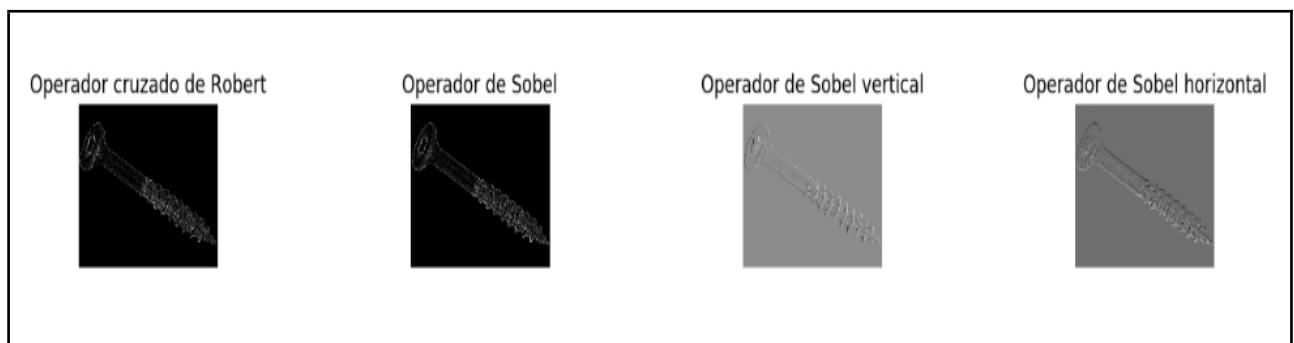


Ilustración 21: Filtro Sobel y Robert.

-Filtro Gaussiano + Sobel

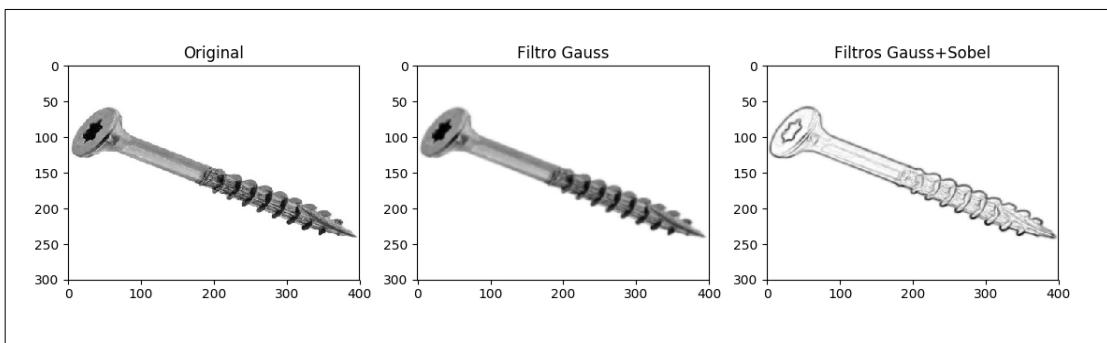


Ilustración 22: Se puede aprecia que se tiene la imagen en negativo de la imagen obtenida con el filtrado con Gauss y Sobel.

-Filtro Perona Malik o Difusión Anisotrópica

Se usa para preservar los detalles de la forma original de los objetos y reducir el ruido eficientemente. Reduce el ruido en las regiones homogéneas y no en los bordes. Cada una de las imágenes resultantes de esta familia se dan como una convolución entre la imagen y el filtro gaussiano 2D isotrópico, donde el ancho del filtro aumenta con el parámetro.

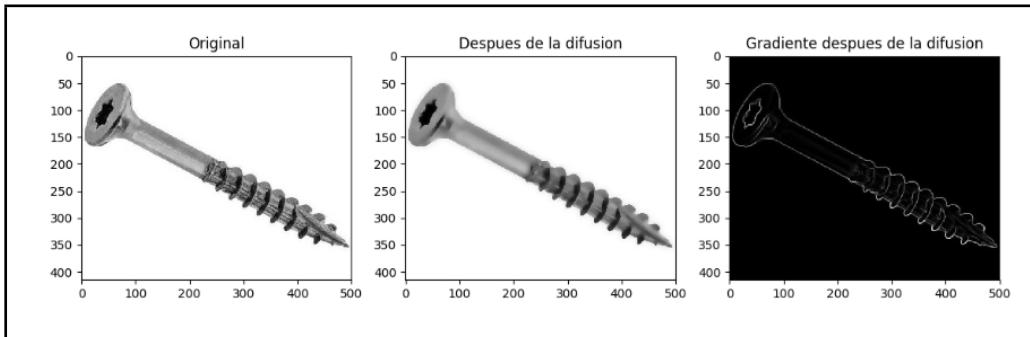


Ilustración 23: En (1) se tiene la imagen original, en (2) es la imagen después de la difusión y en (3) es el gradiente después de la difusión.

-Filtro Laplace, Median, Frangi y Prewitt

El filtro Laplace detecta bordes en la imagen usando el método laplaciano, que produce bordes finos de un píxel de ancho.

El Laplaciano no se suele usar directamente en la práctica por ser muy sensible al ruido, por lo que suele ser sumado y restado (según la imagen utilizada) con la imagen original para realzar los contornos. Por eso se suele usar primero un filtro gaussiano para eliminar el ruido, lo que da lugar al filtro Laplaciano del gaussiano (LoG).

El filtro Median se suele usar para eliminar el ruido de la imagen. Consiste en visitar cada pixel de la imagen y se reemplaza por la mediana de los píxeles vecinos. La mediana se calcula ordenando los valores de los píxeles vecinos en orden y se selecciona el que queda en medio.

El filtro de Frangi se utiliza para detectar crestas continuas en una imagen (ej. arrugas, contornos, filetes del tornillo, etc) y calcular la fracción de la imagen que contiene estos elementos. Consiste en calcular los vectores hessianos para definir la similitud de una región de la imagen con el elemento buscado.

El filtro Prewitt hace uso del operador Prewitt particularmente utilizado para la detección de bordes. Se basa en hacer girar la imagen con un filtro pequeño, separable y de valores enteros en direcciones horizontales y verticales y, por lo tanto, es relativamente económico en términos de cálculos como operadores Sobel. Por otro lado, la aproximación de gradiente que produce es relativamente cruda, en particular para variaciones de alta frecuencia en la imagen.

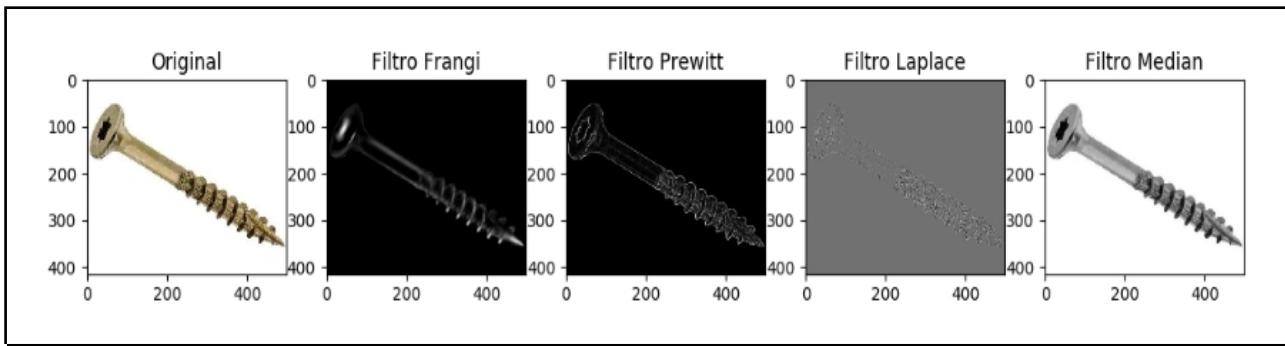


Ilustración 24: Otros filtros. Se utilizó `plt.rcParams['image.cmap'] = 'gray'` por eso las imágenes obtenidas como resultado de cada filtrado es en escala de grises, sino hubiera sido así se puede apreciar una mayor variación para cada caso.

Segmentación

El objetivo es aislar los objetos de interés dejando una imagen más limpia de valores binarios, a la que ya se le podrá extraer un determinado tipo de características.

La segmentación puede ser supervisado o no supervisado. Por ejemplo se puede utilizar el script de LabelImage que al ejecutarlo despliega una pantalla para ir segmentando el objeto de interés en la imagen por imagen, en este caso sería supervisado. Para este proyecto optamos por una segmentación no supervisada, ya que se trabaja con base de datos y luego la clasificación es sin intervención humana.

-Thresholding

Es una técnica para dividir una imagen en dos (o más) clases de píxeles, que generalmente se denominan "primer plano" y "fondo", dependiendo del threshold.

Sí $a(x, y) \Rightarrow th$ entonces $b(x, y) = 0$
 caso contrario $b(x, y) = 255$

Suponiendo "a" la imagen original y "b" la imagen resultante o imagen segmentada.

1) Thresholding Supervisado

Manualmente se modifican los valores de "th" para ver las variaciones en la imagen segmentada.

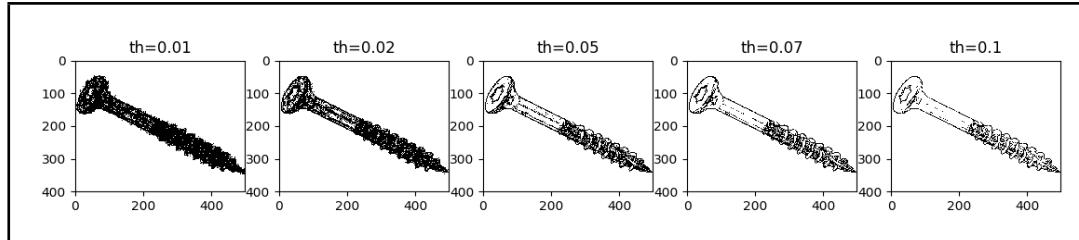


Ilustración 25: Diferentes valores de th .

2) Thresholding No Supervisado

Se utilizó el módulo de skimage para observar los distintos métodos no supervisados.

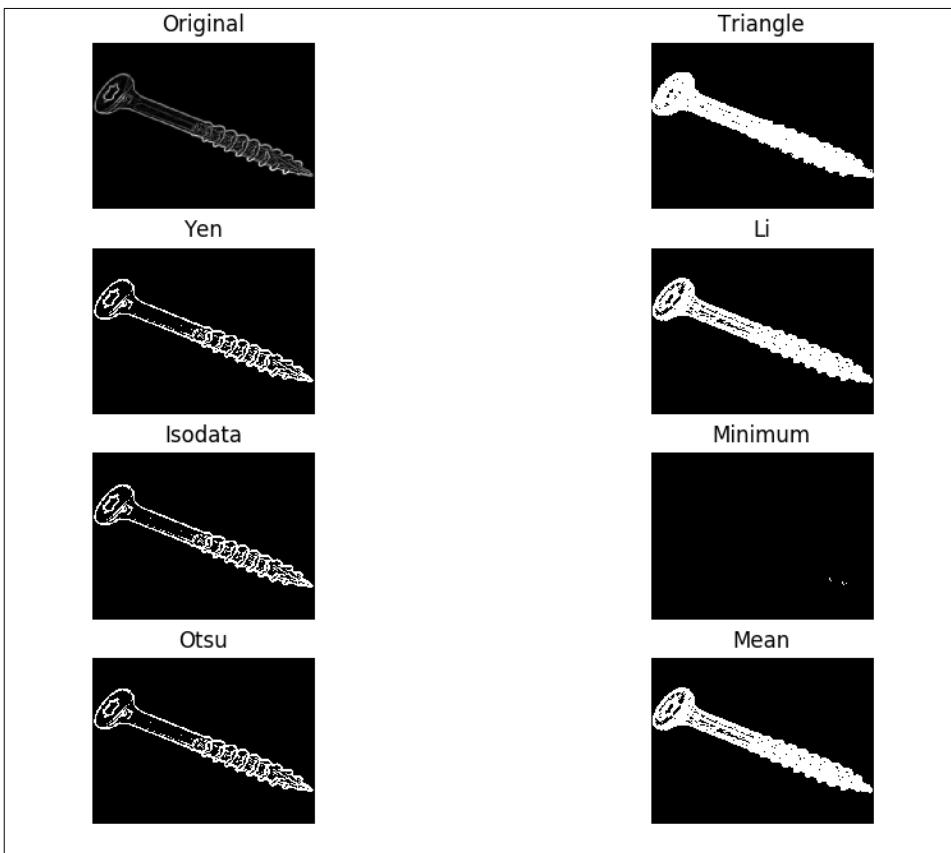


Ilustración 26: Se muestra los distintos métodos de thresholding que permite el módulo skimage.

El método global de segmentación por valor umbral es muy sensible a las variaciones en la luminosidad de la imagen. Una solución alternativa es homogeneizar la luminosidad durante el preprocesamiento de la imagen, por ejemplo realizando una corrección de sombras o por medio de una imagen de referencia que nos permita homogeneizar la luminosidad. Eso es lo que hicimos en la etapa de transformación porque se nos presentaba este problema, por eso se decidió enmascarar la imagen y dejar un fondo totalmente blanco.

Además del problema de la luminosidad, pueden aparecer otros problemas que se pueden reducir tratando la imagen antes de segmentar. A menudo se utilizan técnicas de reducción de la borrosidad o de incremento de la nitidez de los bordes. Los métodos del valor umbral siempre utilizan información unidimensional de la imagen (normalmente un valor de intensidad o un valor de gris). No se tienen en cuenta otras informaciones, como por ejemplo los diferentes colores.

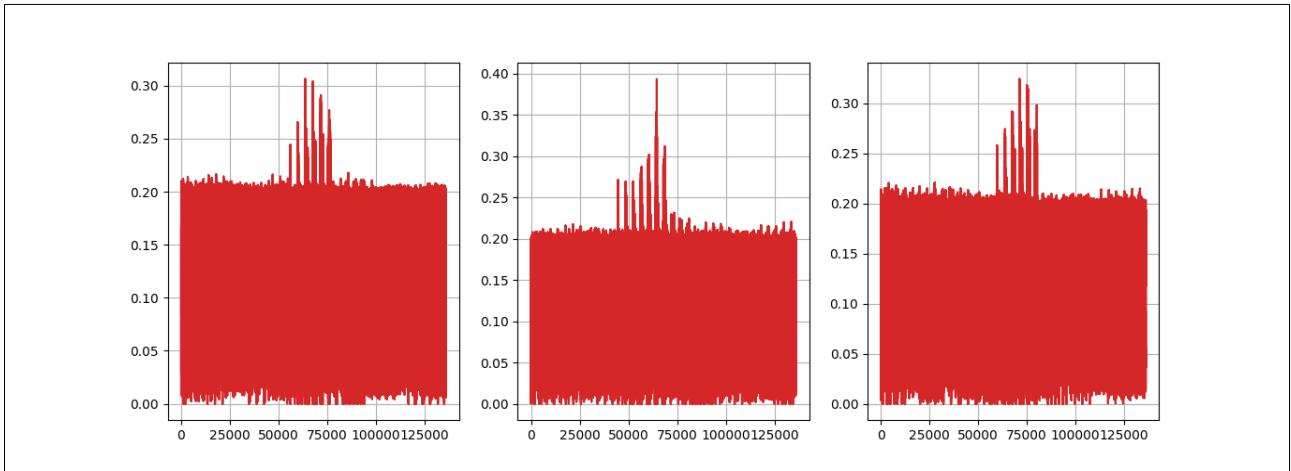
Extracción de Rasgos

El objetivo es describir cuantitativamente la naturaleza de los objetos.

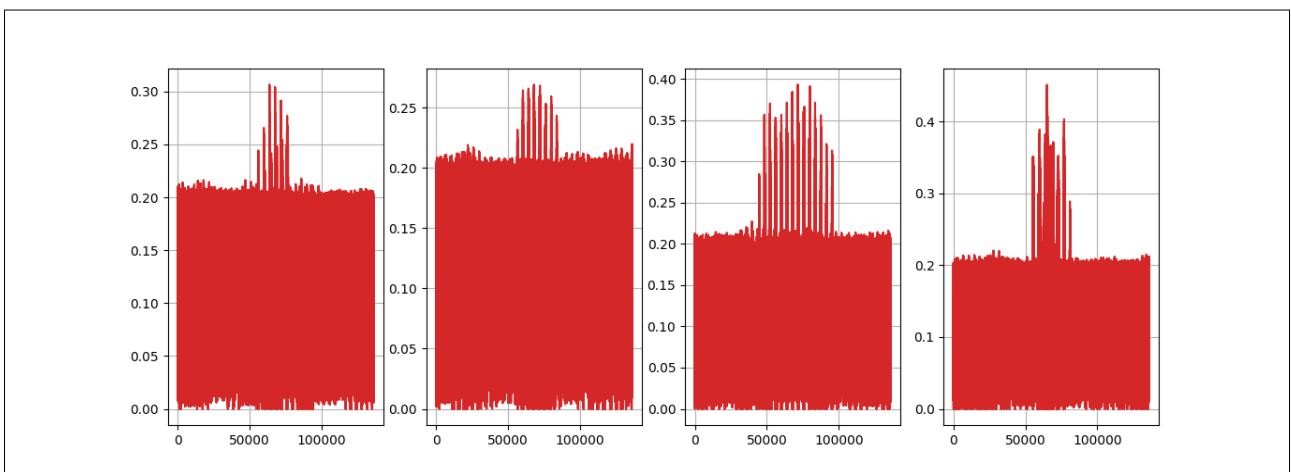
-Histograma de gradientes orientados (HOG)

Es un tipo de descriptor de características, mediante un proceso se obtiene ciertos valores de una misma imagen que puede describir a la misma. El tornillo, clavo, tuerca o arandela quedaría definida por esos valores. La idea esencial es que la apariencia y forma de un objeto local dentro de una imagen pueda ser descrita por la intensidad de la distribución de la dirección de los gradientes.

Se realizó un análisis para 3 tornillos distintos.



Se realizó un análisis para tornillo, tuerca, arandela y clavo, en ese mismo orden se presenta en el gráfico siguiente.



Conclusión: Se utiliza mucho con el propósito de detección de bordes. En mi implementación tiene un alto costo computacional y temporal respecto a los otros extractores de características. Se puede apreciar en el excel “Definir Hog, Haralick y Hu” como varían sus valores.

-Momentos de Hu

La forma característica de un objeto se puede cuantificar mediante sus momentos, los cuales describen la distribución de los píxeles sobre el plano. Los momentos deben ser invariantes a las transformaciones geométricas que puede sufrir el objetos y discriminantes para objetos con formas distintas.

Los momentos de Hu son siete descriptores invariantes que cuantifican la forma de un objeto.

Los primeros 6 momentos de Hu son invariantes a la transformación, escala, rotación y escala; mientras que el 7mo cambia para el reflejo de la imagen. El primer momento de Hu, es análogo, al momento de inercia alrededor del centroide, donde las intensidades de los píxeles son análogas a densidad física.

Muchas veces por fines prácticos se toman los primeros 2 momentos.

$$\text{Rasgo 1: } \theta_1 = \eta_{20} + \eta_{02}$$

$$Rasgo\ 2:\theta_2=(\eta_{20}-\eta_{02})^2+4*\eta_{11}^2$$

donde los momento centrales normalizados de segundo orden se calculan como:

$$\eta_{20}=\frac{m_{20}-\frac{m_{10}^2}{m_{00}}}{m_{00}^2}, \quad \eta_{02}=\frac{m_{02}-\frac{m_{01}^2}{m_{00}}}{m_{00}^2} \quad y \quad \eta_{11}=\frac{m_{11}-\frac{m_{10}*m_{11}}{m_{00}}}{m_{00}^2}$$

y los momentos geométricos (éstos son la base de los momentos de Hu) de orden p+q se calculan como:

$$m_{pq}=\sum_x \sum_y x^p * y^q * I(x,y)$$

Para este proyecto primero se considero utilizar los primeros 4 momentos de Hu porque según el análisis del excel “Definir HOG, Haralick y Hu” son los que más se parecían. Para este proyecto usamos una librería que los calcula directamente.

$$Rasgo\ 3:\theta_3=(\eta_{30}-3*\eta_{12})^2+(3*\eta_{21}-\eta_{03})^2$$

$$Rasgo\ 4:\theta_4=(\eta_{30}+\eta_{12})^2+(\eta_{21}+\eta_{03})^2$$

Existe un trabajo de J. Flusser que plantea una teoría general sobre conjuntos completos e independientes de invariantes rotacionales derivados de momentos. Demostró que el conjunto invariantes de Hu no es ni completo ni independiente: por un lado el 3er momento es redundante, pues es dependiente de otros, y por otro falta un invariante, propuesto por Flusser y denominado “octavo invariante de Hu”. Así que también se evaluó solo los momentos de Hu 1,2 y 4 en el excel “Definir HOG, Haralick y Hu”.

Conclusión: Todas las configuraciones consideradas acá, se evaluarán con el programa Rendimiento.py, en función del mejor porcentaje de predicciones definirá qué configuración se usará.

-Textura de Haralick

La textura de la imagen es una cuantificación de la variación espacial de los valores de tono gris, por eso en este proyecto para la extracción recibe la imagen en escala de grises.

Este método se basa en las distribuciones de probabilidad conjunta de pares de píxeles. Una componente esencial es la definición de ocho celdas de resolución vecinas más cercanas que definen diferentes matrices para diferentes ángulos ($0^\circ, 45^\circ, 90^\circ, 135^\circ$) y distancias entre los píxeles vecinos horizontales.

Haralick describió 14 estadísticos que pueden calcularse a partir de la matriz de coincidencia con la intención de describir la textura de la imagen:

1	Segundo momento angular	8	Suma de entropía
2	Contraste	9	Entropía
3	Correlación	10	Diferencia de varianza
4	Varianza	11	Diferencia de entropía
5	Inverse Difference Moment	12	Info. Medida de correlación 1
6	Suma promedio	13	Info. Medida de correlación 2
7	Suma de varianza	14	Máximo coeficiente de correlación

Para este proyecto utilizamos una librería que calcular las primeras 13 características de Haralick, porque la 14 es ambigua. En el excel “Definir HOG, Haralick y Hu” ([Ver Tabla](#)) se puede apreciar como varían los parámetros para cada pieza de ferretería. Se obtuvieron bastantes similitudes con todos los parámetros y solo con 1,2 y 4. Posteriormente se evaluó su rendimiento con respecto a las predicciones con el programa Rendimiento.py.

-Histograma de color

En las imágenes digitales, un histograma de color representa el número de píxeles que tienen colores en cada una de las listas fijas de rangos de colores, que se extienden sobre el espacio de color de la imagen, es decir, el conjunto de todos los posibles colores.

A continuación se presenta una imagen filtrada en escala de grises, donde se encuentra los picos es aproximadamente donde se encuentran los contornos.

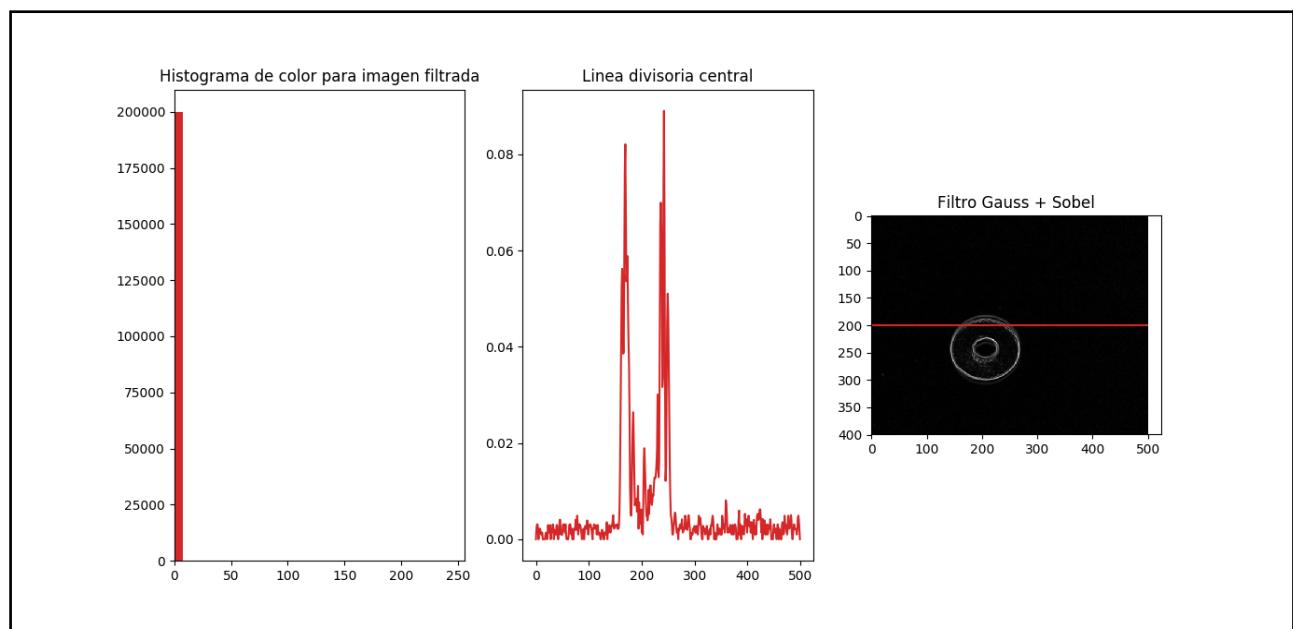


Ilustración 27: Se muestra un histograma de color con abcisas bins y ordenadas número de píxeles.

Conclusión: Se muestra una extracción de características por color, pero debido que usamos piezas de acero con colores similares, no se consideró para la implementación en el código final.

Reducción de Dimensionalidad

Es el proceso de reducción del número de variables aleatorias que se trate, y se puede dividir en selección de la función y extracción de la función.

Un inconveniente que se presentó al momento de realizar el proyecto fue que debido a las variaciones en las dimensiones de la imagen (magnitud de la dimensión de los vectores de características) afectando los gráficos anteriores.

Se puede utilizar estadísticos como la media aritmética y desviación estándar para los distintos vectores describan su magnitud y dispersión.

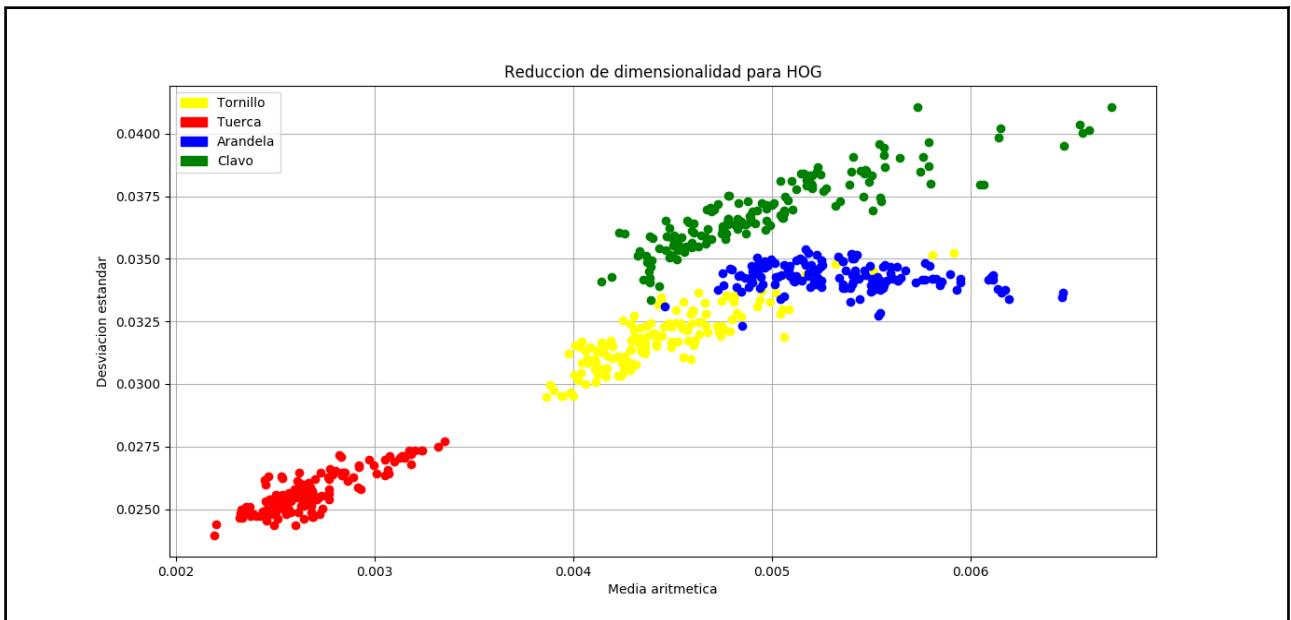


Ilustración 28: Reducción de dimensionalidad para el Histograma de Gradientes Orientados, para este gráfico se contó con una base de datos de 600 fotos. Cada pieza de ferretería esta formada por 150 fotos cada uno.

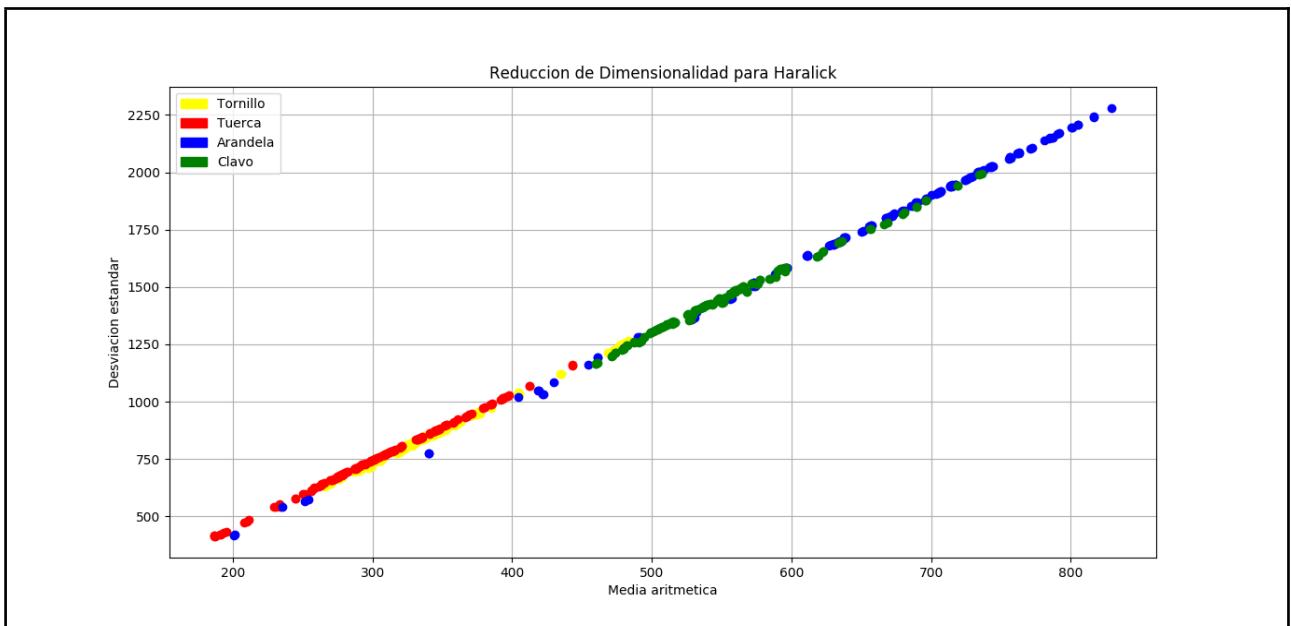


Ilustración 29: Reducción de dimensionalidad para Haralick, para este gráfico se contó con una base de datos de 600 fotos. Cada pieza de ferretería esta formada por 150 fotos cada uno.

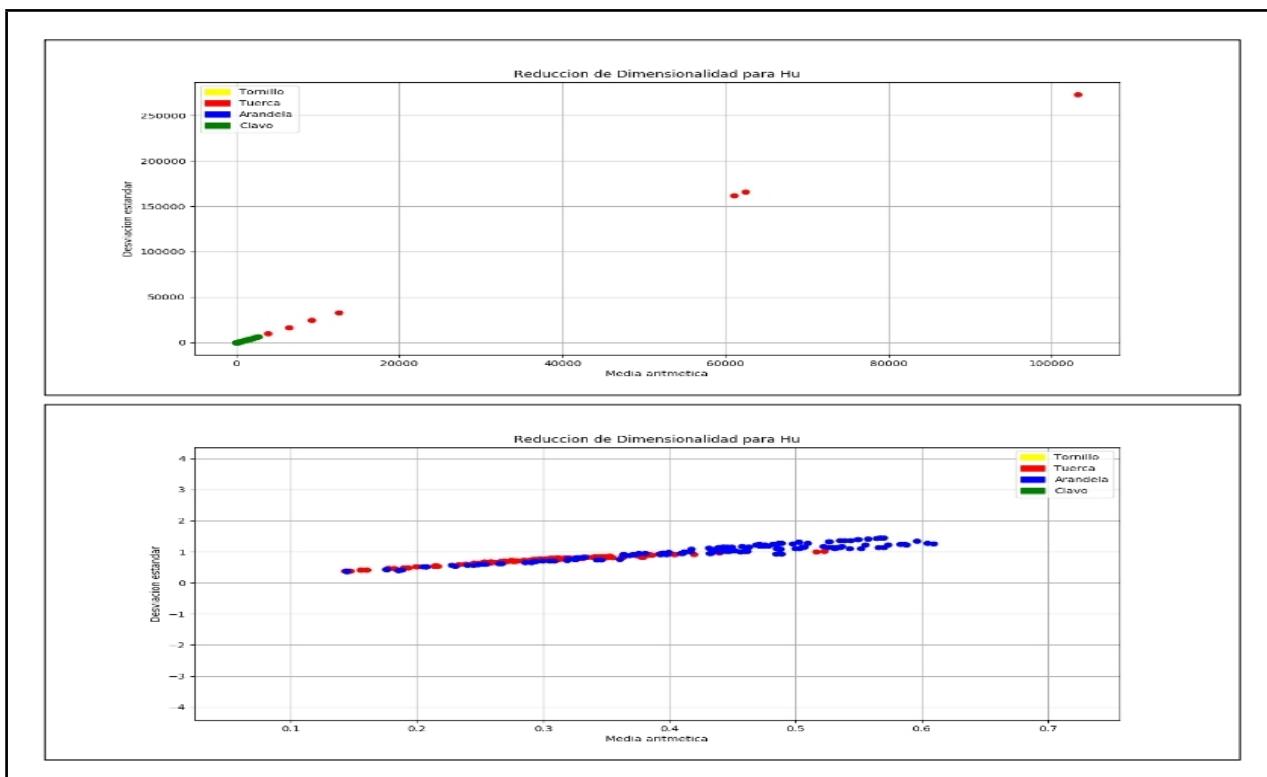


Ilustración 30: Reducción de dimensionalidad para los momentos de Hu, para este gráfico se contó con una base de datos de 600 fotos. Cada pieza de ferretería esta formada por 150 fotos cada uno. Se le hizo un zoom a la primer imagen superior para poner en evidencia los momentos de Hu de la tuerca y arandela, que están superpuestos uno encima del otro.

Conclusión: Mediante la reducción de dimensionalidad se logró una mejor forma de visualización de HOG, pero no tanto Haralick y los momentos de Hu para este caso.

RENDIMIENTO

Base de Datos

Se debe obtener la etiqueta o “label” para cada pieza de ferretería. Para clasificarlo de una mejor manera se crearon unas carpetas llamadas YTornillos, YTuercas, YArandelas e YClavos correspondiente a los elementos que se van a categorizar(imágenes cortadas con fondo totalmente blanco), tanto para el YTest y YTrain.

-Overfitting y underfitting

Son principales causas al obtener malos resultados en los análisis. Cuando se entrena los modelos se intenta “hacerlos encajar” (en inglés fit), los datos de entrada entre ellos y con la salida. Se puede traducir overfitting como sobreajuste y underfitting como subajuste y ambos hacen referencia al fallo de nuestro modelo al generalizar y/o encajar el conocimiento que pretendemos que adquieran.

Deberemos encontrar un punto medio en el aprendizaje de nuestro modelo en el que no estemos incurriendo en underfitting y tampoco en overfitting. Para reconocer este problema deberemos subvencionar nuestro conjunto de datos de entrada para entrenamiento en dos: uno para entrenamiento y otro para la Test que el modelo no conocerá de antemano. Esta división se suele hacer del 80% para entrenar y 20% para test. El conjunto de Test deberá tener muestras diversas en lo posible y una cantidad de muestras suficiente para poder comprobar los resultados una vez entrenado el modelo.

Si el modelo entrenado con el conjunto de entrenamiento tiene un 90% de aciertos y con el conjunto de test tiene un porcentaje muy bajo, esto señala claramente un problema de overfitting.

Si en el conjunto de Test sólo se acierta un tipo de clase (por ejemplo tornillos) o el único resultado que se obtiene es siempre el mismo valor será que se produjo un problema de underfitting.

Se tomaron 760 imágenes en total. Se destinó un 80% de imágenes para el entrenamiento y el 20% para el test. Cada pieza de ferretería en YTest cuenta con 39 imágenes y para YTrain 151 imágenes.

KNN

K-Nearest Neighbors (KNN), también llamado el vecino más cercano, puede ser utilizado para clasificación y problemas de regresión predictiva, en la industria es más utilizado para problemas de clasificación, donde la mayoría de los conjuntos de datos del mundo real no siguen supuestos teóricos matemáticos.

Implementación: Facilidad para interpretar las salidas (predicciones), bajo tiempo de cálculo y el poder de predicción.

-Pasos del algoritmo KNN

- 1) Medir la distancia euclídea entre el patrón que se desea reconocer y todas las muestras del conjuntos de entrenamiento.
- 2) Identificar los K patrones de entrenamiento más cercanos al patrón que se desea reconocer (i.e., aquellos que obtuvieron menor distancia euclídea) y obtener la etiqueta de clase de cada vecino.
- 3) El patrón que se desea reconocer se asigna a la clase que obtuvo el mayor número de concurrencias o votos.

-¿Cómo se decide el número de vecinos de KNN?

El número de vecinos (K) en KNN, es un hiperparámetro que debe elegir en el momento de la construcción del modelo. Se puede pensar en K como una variable de control para el modelo de predicción.

En el caso de una pequeña cantidad de vecinos, el ruido tendrá una mayor influencia en el resultado, y una gran cantidad de vecinos lo hará computacionalmente costoso. También una pequeña cantidad de vecinos son más flexibles, tendrán un sesgo bajo pero una varianza alta y un gran número de vecinos tendrá un límite de decisión más uniforme, lo que significa una varianza más baja pero un sesgo más alto. Osea KNN funciona mejor con un menor número de características que un gran número de características.

Se recomienda un número impar si el número de clases es par. También se puede verificar generando el modelo en diferentes valores de K y verificar su rendimiento (es lo que hacemos en el programa Rendimiento.py).

-Maldición de la Dimensionalidad

El aumento de la dimensión también conduce al problema del sobreajuste. Para evitar el sobreajuste, los datos necesarios deberán crecer exponencialmente a medida que aumente el número de dimensiones y se ha corroborado que en grandes dimensiones, la distancia euclídea ya no es útil. Este problema de dimensión superior se conoce como la Maldición de la Dimensionalidad.

Ésto nos ocurría cuando utilizábamos todos los 7 momentos de Hu o los 14 momentos de Haralick, en nuestro algoritmo con la base de datos relativamente pequeña que teníamos.

-Ventajas y desventajas

La fase de entrenamiento de la clasificación de vecino K más cercano es mucho más rápida en comparación con otros algoritmos de clasificación. No hay necesidad de entrenar un modelo para la generalización, es por eso que KNN se conoce como el “Algoritmo de aprendizaje simple y basado en instancias”. El valor de salida para el objeto se calcula por el promedio del valor de K vecinos más cercanos.

La fase de prueba de la clasificación de vecino K más cercano es más lenta y costosa en términos de tiempo y memoria. Requiere una gran memoria para almacenar todo el conjunto de datos de entrenamiento para la predicción. KNN requiere escalado de datos porque usa la distancia euclídea entre dos puntos de datos para encontrar vecinos más cercanos. La distancia euclídea es sensible a las magnitudes. Las características con altas magnitudes pesarán más que las características con bajas magnitudes. KNN tampoco es adecuado para datos de grandes dimensiones.

-¿Cómo mejorar KNN?

Para obtener mejores resultados, se recomienda encarecidamente normalizar los datos en la misma escala, es lo que se hace en la etapa de Transformación. Generalmente, el rango de normalización considerado entre 0 y 1. Este algoritmo no es adecuado para los datos de grandes dimensiones. En tales casos, la dimensión debe reducirse para mejorar el rendimiento, como se aprecia en los gráficos de la etapa de Reducción.

KMeans

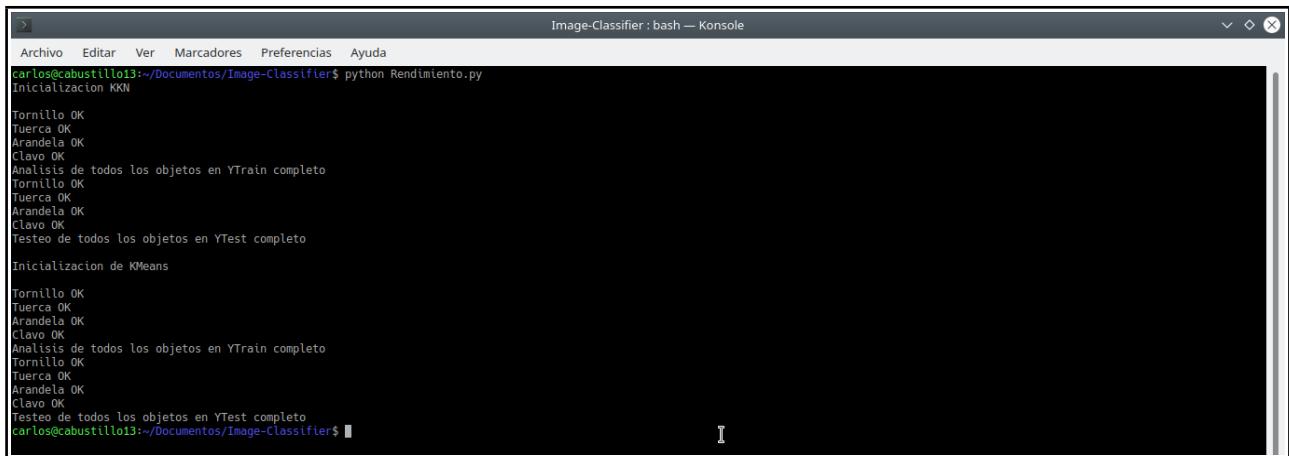
Es un algoritmo de agrupación que divide las observaciones en K agrupaciones. La agrupación es un tipo de aprendizaje automático no supervisado que tiene como objetivo encontrar subgrupos homogéneos de manera que los objetos en el mismo grupo (agrupaciones) sean más similares entre sí que los demás.

-Pasos del algoritmo KMeans

- 1) Inicialización: Se elige la localización de los centroides K grupos aleatoriamente.
- 2) Asignación: Se asigna a cada dato el centroide más cercano.
- 3) Actualización: Se actualiza la posición del centroide a la media aritmética de las posiciones de los datos asignados al grupo.
- 4) Se repiten los paso 2) y 3) hasta que ya no haya cambios.

Se recomienda normalizar los datos, ya que ayuda al clustering porque los grupos se forman a partir de la distancia euclídea cuadrada. Esto se realiza en la etapa de la Transformación. También se recomienda evitar caer en la maldición de la dimensionalidad.

Evaluación del rendimiento



```
Image-Classifier : bash — Konsole
Archivo Editar Ver Marcadores Preferencias Ayuda
carlos@cabustillo13:~/Documentos/Image-Classifier$ python Rendimiento.py
Inicialización KKN
Tornillo OK
Tuerca OK
Arandela OK
Clavo OK
Analisis de todos los objetos en YTrain completo
Tornillo OK
Tuerca OK
Arandela OK
Clavo OK
Testeo de todos los objetos en YTest completo
Inicialización de KMeans
Tornillo OK
Tuerca OK
Arandela OK
Clavo OK
Analisis de todos los objetos en YTrain completo
Tornillo OK
Tuerca OK
Arandela OK
Clavo OK
Testeo de todos los objetos en YTest completo
carlos@cabustillo13:~/Documentos/Image-Classifier$
```

Ilustración 31: Captura de pantalla del programa Rendimiento.py.

En la tabla de excel “Tabla de comparación de los métodos de extracción” ([Ver Tabla](#)) se aprecia el proceso completo para la selección del tipo de extractor de característica, hiperparámetro K, tipo de segmentación, recorte de imagen, color de fondo y tipo de filtro.

Luego de todo el proceso de análisis y prueba, se seleccionó que las siguientes características:

-La imagen debe ser recortada 700 píxeles respecto al borde izquierdo, para eliminar la franja de la pared. Posteriormente redimensionada a 500x400 píxeles y luego realizar un enmascaramiento para que tenga un fondo totalmente blanco, ésto se realiza en la etapa de transformación.

-La imagen no debe ser recortada en la etapa de adaptación, osea se debe tomar directamente la imagen de la fase de transformación para la base de datos.

-Se utilizaran el momento de Hu: 1,2 y 4.

-Se hará un filtrado con Gauss+Sobel.

-No se utilizará threshondilng, por eso no se utiliza la etapa de segmentación.

-El hiperparámetro K se evaluará para K=1 y K=9. Se eligió esos valores porque se aprecia que para esos valores se tiene un alto rendimiento y como se tiene un número par de piezas de ferretería se eligió K=9 en vez de K=8.

-Para KNN

Con todas las consideraciones anteriores, se logró un rendimiento: 92% para K=1 y 94% para K=9.

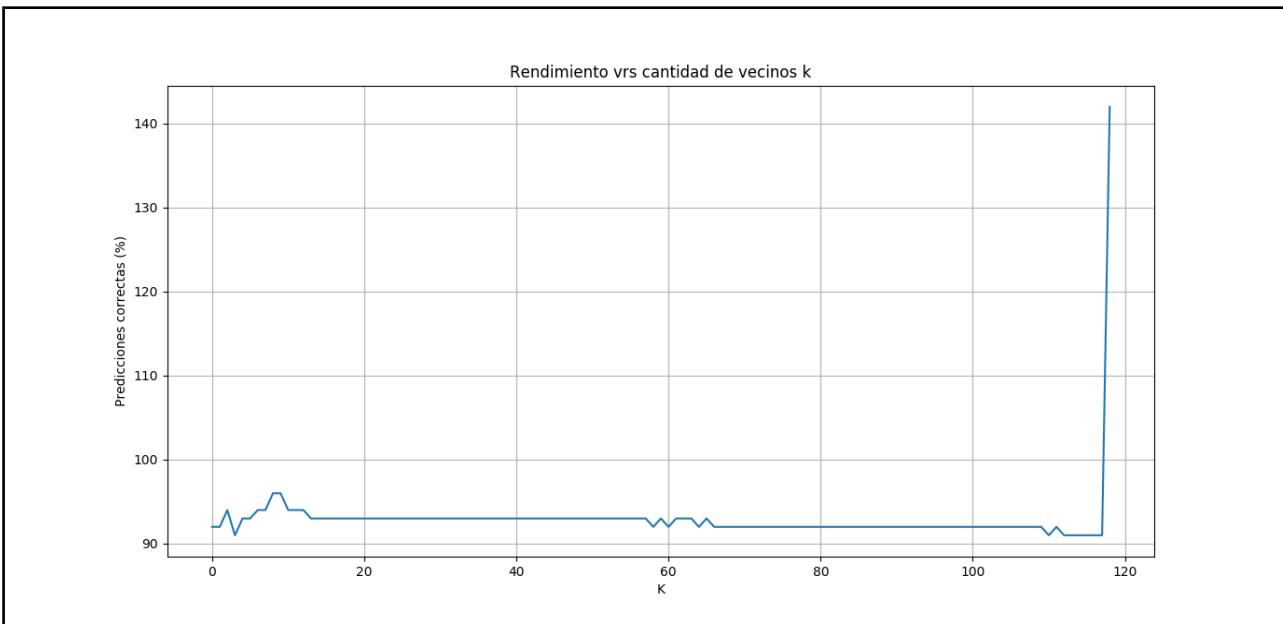


Ilustración 32: Se muestra el rendimiento del algoritmo KNN que obtuvo evaluando las 156 imágenes de la carpeta YTest.

-Para KMeans

Con todas las consideraciones anteriores se logró un rendimiento de 91%.

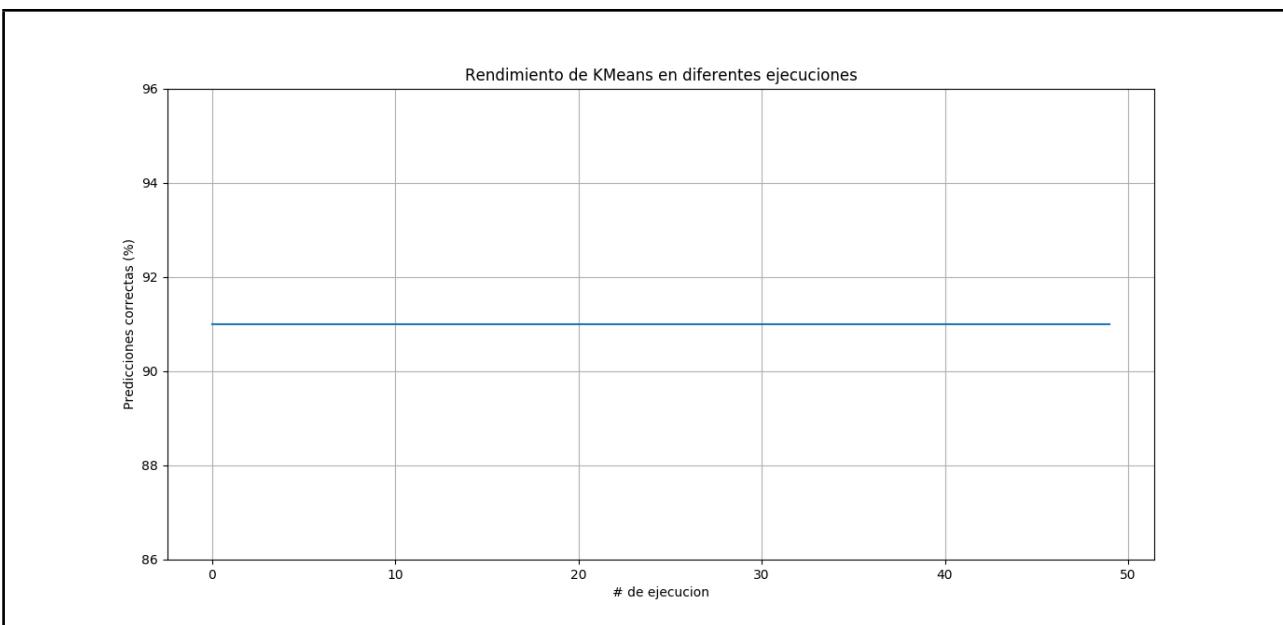


Ilustración 33: Se muestra el rendimiento del algoritmo KMeans que obtuvo evaluando las 156 imágenes de la carpeta YTest.

-Conclusión

Las predicciones que fallan suelen ser porque el algoritmo no distingue una arandela de una tuerca y viceversa. Ésto suele pasar porque:

-Se toma una imagen aproximadamente perpendicular al plano de una cara de las tuercas, y debido que el enmascaramiento hace que se pierde información de los bordes de la tuerca, el algoritmo lo toma como una arandela. Lo recomendable es que al tomar la imagen una tuerca tome parte del espesor de la tuerca para asegurar que la reconozca correctamente.

-Hay ciertos puntos en la caja, donde se produce mucho brillo y sombras en la arandela, y al momento del enmascaramiento se pierde información debido a eso, y por eso se pierde información de la imagen. Y en muchos casos falla haciendo la predicción que es una tuerca en lugar de una arandela.

CÓDIGO

1) Primero se hizo el programa en forma separada para evaluar cada uno de las funciones a implementar en el proyecto. Ésto se puede observar en la carpeta “Programas por separado”.

2) Luego se pone en evidencia cada etapa del proyecto por separado con su respectivo programa en Python. Haga clic en cada hipervínculo para mostrar la ubicación del programa y poder verlo.

-Transformación: Se muestra quitar línea de fondo, redimensión de imagen y quitar fondo → convertir el fondo a uno totalmente blanco.

[Ver código.](#)

-Adaptación: Se muestra el algoritmo de Canny para detectar bordes, uso de thresholding para detectar bordes y recortar imagen a partir del Algoritmo de Canny.

[Ver código.](#)

-Preprocesamiento: Se muestra la descomposición de una imagen en la tupla de valores R, G y B, conversión de RGB a escala de grises utilizando librerías y conversión de RGB a escalas de grises utilizando pesos WG, WR y WB arbitrarios ingresados manualmente.

[Ver código.](#)

-Filtración: Se muestra la implementación del filtro Sobel y Roberts, filtro Gaussiano+Sobel, Perona Malik, filtro Laplace, Median, Frangi y Prewitt.

[Ver código.](#)

-Segmentación: Se muestra thresholding supervisado y thresholding no supervisado.

[Ver código.](#)

-Extracción: Se muestra la representación de Histograma de color para una imagen filtrada y comparación de características Hu, HOG y Haralick para las distintas piezas de ferretería.

[Ver código.](#)

-Reducción: Se muestra la reducción de dimensionalidad para Hu, Haralick y HOG.

[Ver código.](#)

-Rendimiento: Se determina el rendimiento para KNN y KMeans para distintos métodos de extracción de características. Contiene gran parte de los programas previamente mencionados.

[Ver código.](#)

-Clasificación: Este es el programa principal del proyecto, con este se evaluá la clasificación para cada imagen. Contiene gran parte de los programas previamente mencionados.

[Ver código.](#)

EJEMPLO DE APLICACIÓN Y RESULTADOS. CLASIFICACIÓN DE OBJETOS KNN Y KMEANS.



Ilustración 34: Se procede a hacer análisis de este tornillo para este ejemplo de aplicación.

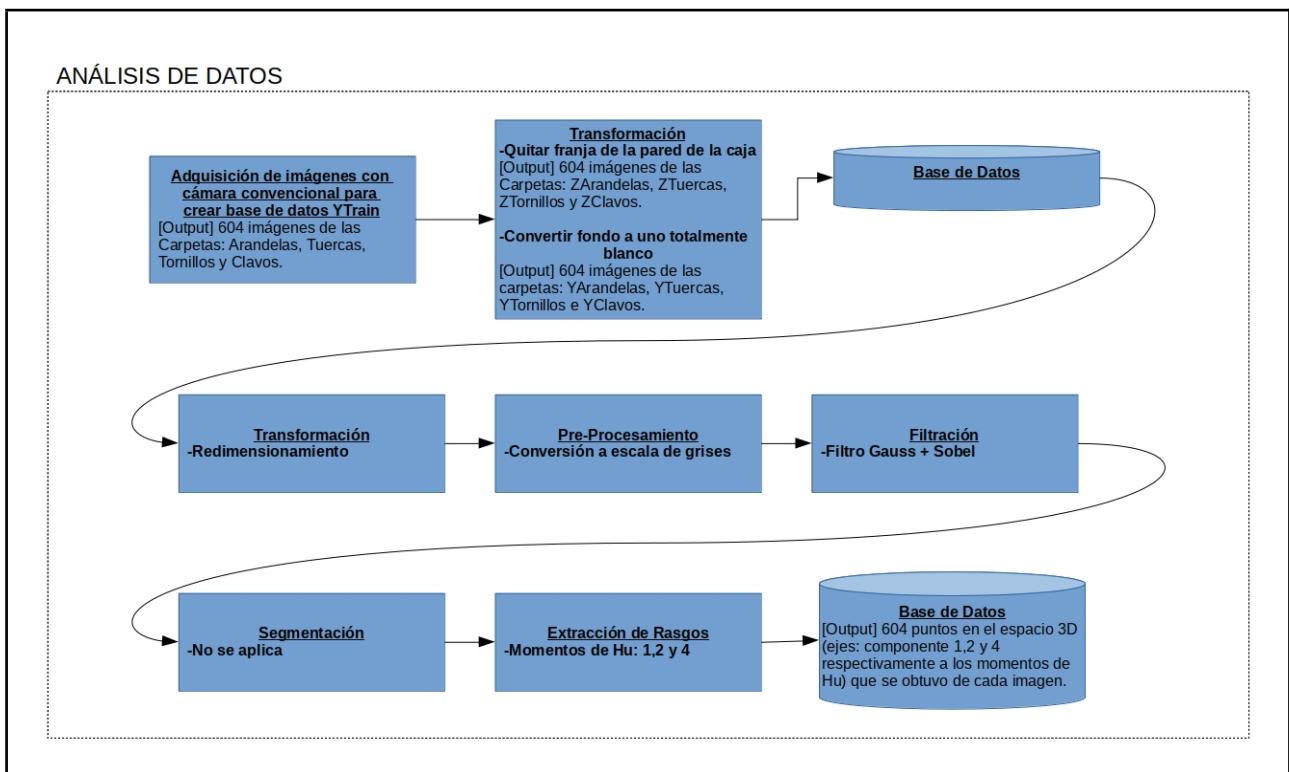


Ilustración 35: Diagrama de flujo de la parte de "Análisis de datos" del programa Clasificación.py.

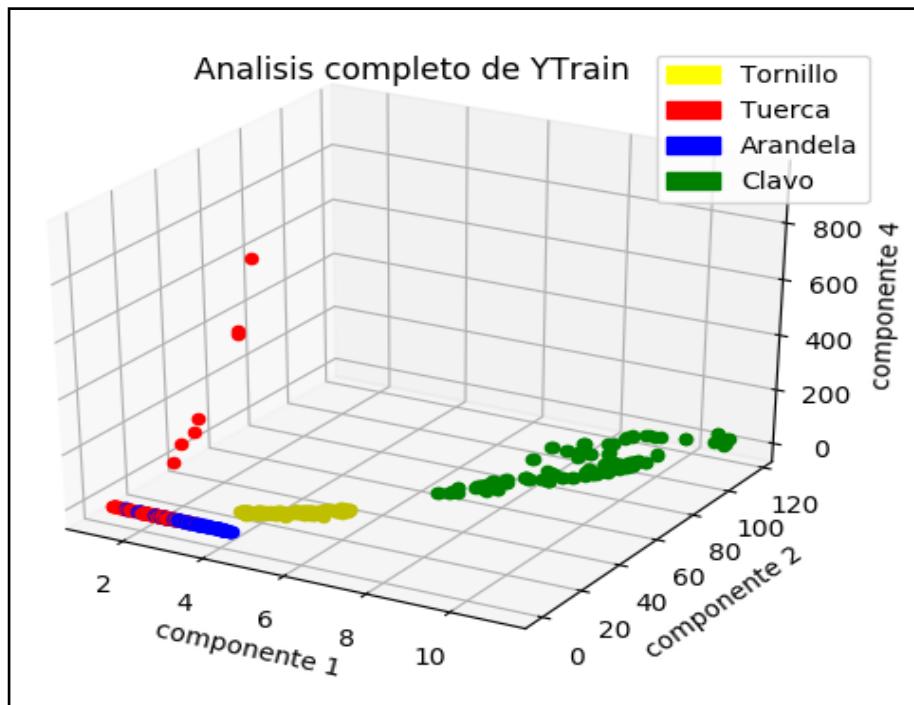


Ilustración 36: Base de datos formada por 604 imágenes en la carpeta YTrain.

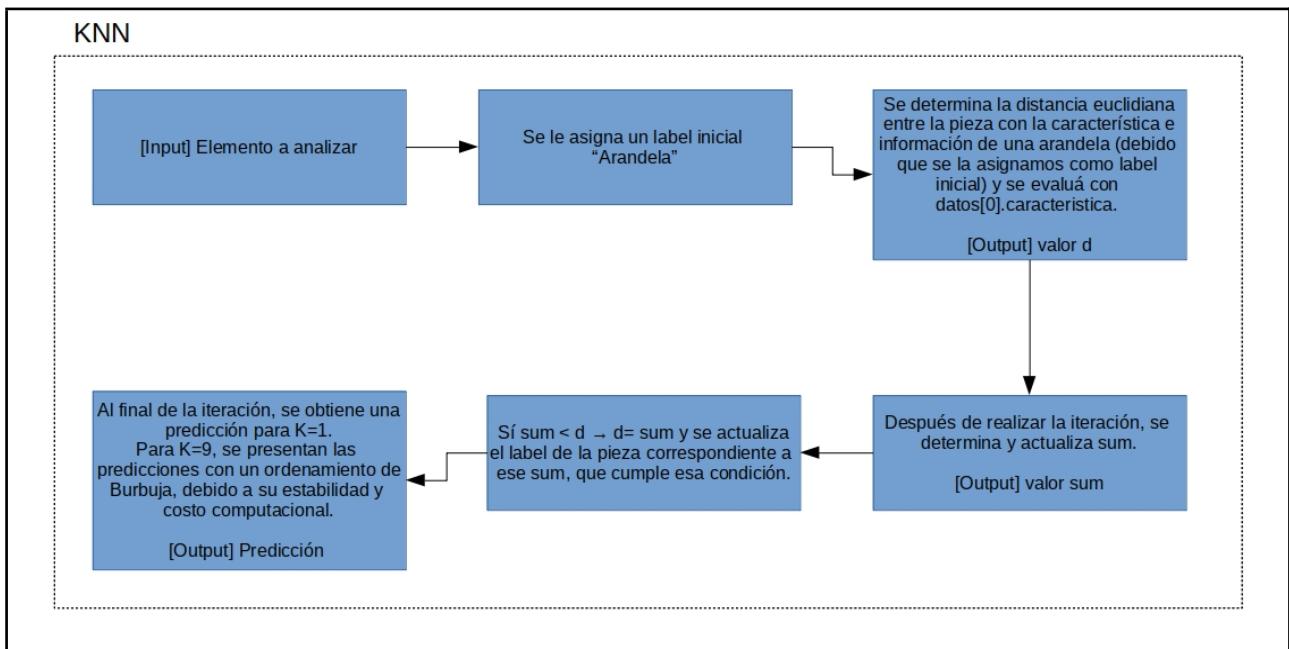


Ilustración 37: Diagrama de flujo de la parte de "KNN" del programa Clasificación.py.

Image-Classifier : bash — Konsole

```

Archivo Editar Ver Marcadores Preferencias Ayuda
carlos@cabustillo13:~/Documentos/Image-Classifier$ python Clasificacion.py
Tornillos OK
Tuercas OK
Arandelas OK
Clavos OK
Analisis completo de la base de datos de YTrain
Cantidad de imagenes analizadas:
604
Introduce numero de la foto: 0
Inicializacion KNN
Prediccion para KNN con K=1:
Clavo

Predicciones para KNN con K=9:
Clavo
Clavo
Clavo
Clavo
Clavo
Clavo
Clavo
Clavo
Clavo

Inicializacion KMeans
Ubicacion de los means finales
Tornillo Tuerca Arandela Clavo
(155, 179, 123, 147)
Mean mas cercano
Tornillo Tuerca Arandela Clavo
(38.55264600580347, 58.07650809973184, 57.74712824369017, 22.85743406669089)
Prediccion para KMeans:
Clavo
carlos@cabustillo13:~/Documentos/Image-Classifier$ 
```

Ilustración 38: Captura de pantalla del programa en ejecución Clasificacion.py.

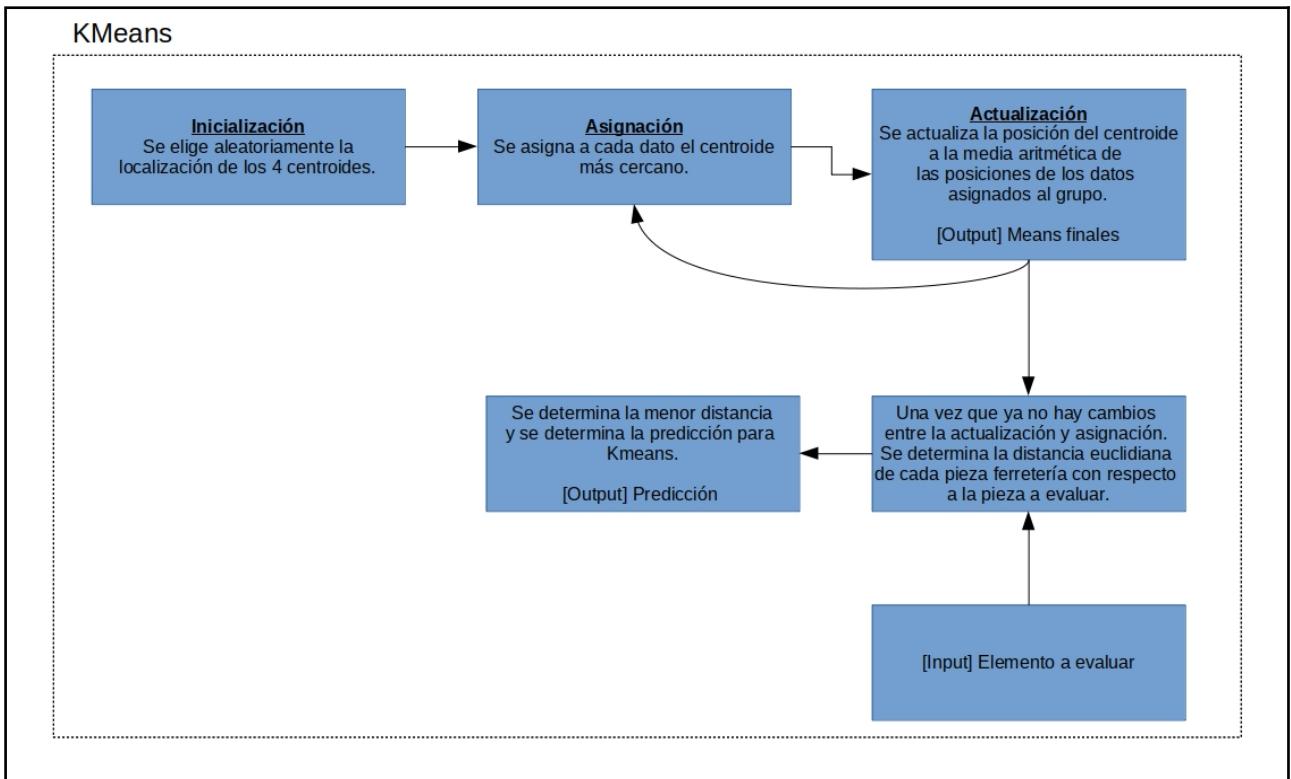


Ilustración 39: Diagrama de flujo de la parte de "KMeans" del programa Clasificación.py.

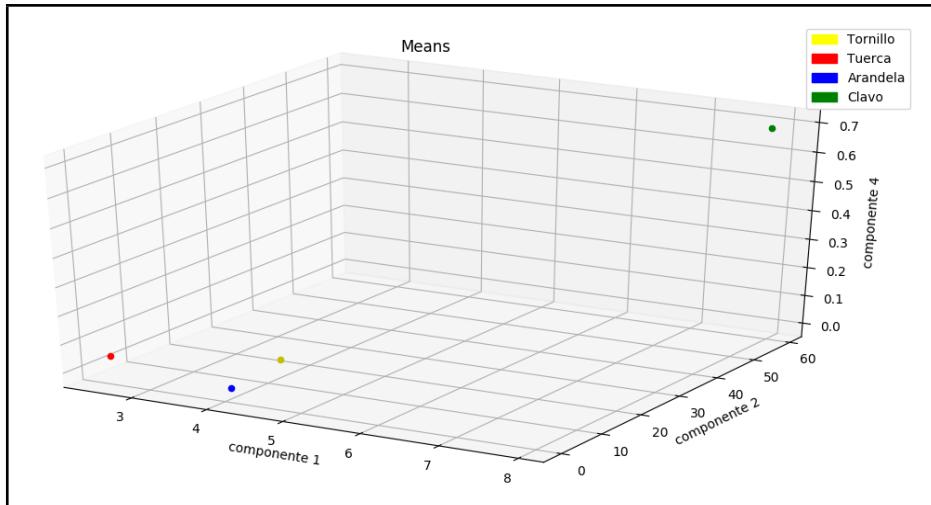


Ilustración 40: Means finales obtenido a partir de las 604 imágenes de YTrain.

CONCLUSIONES

- Debido al sistema operativo Kubuntu utilizado existieron muchos conflictos con las librerías utilizadas, por eso las imágenes se tuvieron que procesar en partes. Por un lado la eliminación de la franja de pared de la caja y conversión del fondo a uno totalmente blanco, y por otro lado el correspondiente análisis.
- Las etapas utilizadas fueron: Transformación para el redimensionamiento de la imagen de 1220x1080 a 500x400, para tener un tamaño normalizado de las dimensiones; Preprocesamiento para convertir la imagen a una escala de grises; Filtración se utiliza el filtro gaussiano para suavizar bordes y el filtro sobel para detectar bordes; Segmentación no se utilizó; Extracción de características analizamos los momentos de Hu: 1, 2 y 4.
- Con el uso de una caja cerrada con iluminación frontal y fondo blanco (a diferencia del fondo negro, con éste se ponía en evidencia las sombras que se producían debido a la disposición de los focos en la caja), se obtuvo un alto rendimiento superior al 90% tanto para KNN y KMeans. Por otro lado debido a la disposición dentro de la cámara en la caja se tenía que realizar transformaciones a la imagen de entrada.
- Se tomaron 720 imágenes para el análisis de este proyecto. El 80% se destinó al entrenamiento y el otro 20% al test. Dentro de la caja existe una región en la cual no se producen sombras ni brillos, debido a la iluminación frontal, pero el problema que si la base de datos se tomará solo en esta región los momentos de Hu entre tuercas y arandelas son muy similares disminuyendo el rendimiento por falsos positivos. En base a eso, se logra determinar una región que ampara todas estas consideraciones que es donde tomamos cada imagen para la base de datos.
- Con el algoritmo KNN se obtuvieron mejores resultados, tanto para K=1 y K=9, pero presenta un alto tiempo de ejecución. Recordar que KNN compara el vector de características de la imagen a clasificar con cada vector de características de cada una de las imágenes de la base de datos YTrain. Mientras que KMeans es más rápido en ejecución debido que calcula la distancia a los 3 means principales.

- Para futuras implementaciones, podría pensarse en un sistema de reconocimiento automático de éstos objetos, los cuales van por una cinta transportadora, para su clasificación en caja o embolsados. Si en la línea de producción aparece un elemento que, captado por una cámara y una vez digitalizada la imagen, se parece (dentro de un margen de confianza en el parecido) a un tornillo, tuerca, arandela o clavo, se dirigirá a la caja correspondiente. Si no se tirará en una caja de elementos desconocidos. De acuerdo la velocidad de la cinta transportadora se puede elegir entre el algoritmo KNN y KMeans, siendo éste último más rápido en tiempo de ejecución.

También se podría plantear una cinta transportadora transparente por la que circula tornillos, tuercas, arandelas y clavos, donde se necesite desarrollar un sistema que cuente cuántas unidades de cada tipo hay en cada momento en un intervalo de la cinta. Se supone que la iluminación se realiza contraluz con lo que se obtienen las siluetas. Acá se puede utilizar el criterio el número de agujeros presentes en la pieza y la desviación típica de las distancias del perímetro al centro de la pieza.

BIBLIOGRAFÍA

[Libros]

- Inteligencia Artificial. Un Enfoque Moderno por Stuart Russell - Peter Norvig.
- Técnicas y algoritmos básicos de visión Artificial por Grupo de Investigación EDMANS
- Presentación “Reconocimiento de objetos en fotografías” por Cinvestav Tamaulipas.
- Procesado digital de señales - 2: Fundamentos para comunicaciones y control por Eduard Bertran Albertí.

[Web]

- <https://docs.opencv.org>
- <https://docs.gimp.org>
- <https://www.aprendemachinelearning.com>
- <https://pythoneyes.wordpress.com>
- <https://www.datacamp.com>

ÍNDICE

Título	pág. 1
Resumen	pág. 2
Introducción	pág. 3
Diseño del sistema	pág. 4
Código	pág. 30
Ejemplo de aplicación	pág. 31
Resultados	pág. 31
Conclusiones	pág. 34
Bibliografía	pág. 35