

DISEÑO DE UN SISTEMA DE VISIÓN ARTIFICIAL PARA LA CLASIFICACIÓN DE LIMÓN UTILIZANDO RASPBERRY PI

La investigación de este proyecto tiene como objetivo medir ciertas características, utilizando técnicas de visión artificial, características como: el tamaño, el color y el defecto del limón, con la finalidad de mejorar la calidad del producto y reducir los costos que implica el proceso de clasificación del limón en la industria.

Para alcanzar los objetivos se propone un conjunto de técnicas de procesamiento de imágenes que van desde **binarización**, **morfología matemática** hasta transformación del modelo de color RGB a **HSV**.

+ MODELO DE COLOR HSV

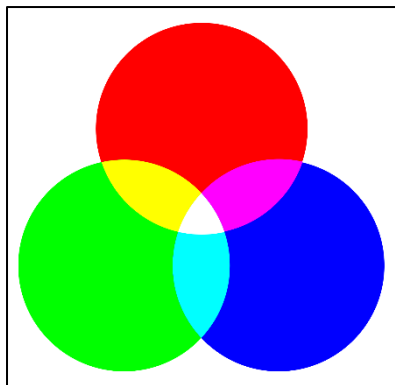
Es un modelo de color en términos de sus **componentes**. En forma simplificada, se trata de una transformación no lineal del espacio de color RGB. En nomenclatura HSV es lo mismo que HSB.

CARACTERÍSTICAS:

MATRIZ (H)

Se representa como un rango de ángulos cuyos valores posibles van de 0 a 360°. Donde cada valor corresponde a un color. Ej: 0° es rojo, 60° es amarillo y 120° es verde.

De forma intuitiva se pueden conocer los valores básicos RGB: Disponemos de 360° divididos en 3 colores RGB, eso da 120° por color, sabiendo esto podemos asociar que: 0 es rojo RGB (1,0,0) / 120° es verde RGB (0,1,0) y 240° es azul RGB (0,0,1). Para colores mixtos se utilizan los grados intermedios.



Ej:

- 0° = rojo RGB (1,0,0)
 - 60° = amarillo RGB (1,1,0) / está entre rojo y verde, por lo tanto 60°.
 - 120° = verde RGB (0,1,0)
 - 180° = celeste RGB (0,1,1) / está entre verde y azul.
 - 240° = azul RGB (0,0,1)
 - 300° = morado RGB (1,0,1) / está entre azul y rojo.
 - 360° = 0° (sistema radial)
-

Se puede observar la secuencia de sumar 60° y añadir o quitar un 1.

SATURACIÓN (S)

Se representa como la distancia al eje de brillo negro-blanco. Los valores posibles van del 0 al 100%. A este parámetro también se le denomina “pureza” por la analogía con la pureza de excitación y pureza colorimétrica en colorimetría. Cuando menor sea la saturación (S) de un color, mayor tonalidad grisácea tendrá y más decolorado estará (más gris).

100% PURO

75% DE SATURACIÓN

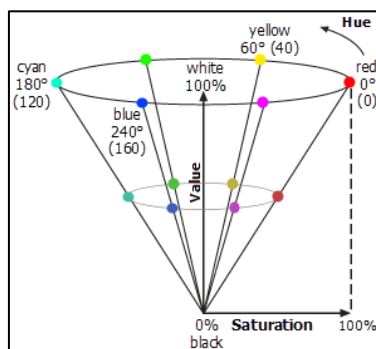
50% SATURACIÓN MEDIA

25% DE SATURACIÓN

0% DE SATURACIÓN

VALOR (V)

Representa la altura en el eje blanco-negro. Los valores van del 0 al 100%. Donde 0% siempre es negro. Dependiendo de la saturación (S), 100% podría ser blanco o un color más o menos saturado. También se le denomina LUMINOSIDAD O NEGRURA, donde 0% significa que no hay brillo (por lo tanto, negro) y 100% significa brillo total, por lo tanto, un espectro entre un color puro (saturación del 100%) y el blanco (saturación de 0%).



El modelo HSV (o HSB, como se prefiera) deriva el espacio RGB y representa los colores combinando tres parámetros: MATRIZ (H), SATURACIÓN (S) y VALOR / LUMINOSIDAD (V), por lo que se parece más a la percepción del color humano. Es más fácil ajustar un color (matriz) por su saturación (pureza) y brillo (valor). En consecuencia, en la mayoría de las aplicaciones gráficas puede definir un color de acuerdo con la ruleta de color HSV.

+ BINARIZACIÓN DE UNA IMAGEN

Consiste en un proceso de reducción de la información de una imagen, en la que sólo persisten dos valores: verdadero y falso, es llevar una imagen a escala de 0 y 1 o, más frecuentemente, por los colores negro (valor de gris 0) y blanco (valor de gris 255).

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

Para este proyecto es necesario aislar la iluminación del ambiente, a una fuente de iluminación artificial: que sea constante para independizar el proceso sea de día o de noche. Para fines didácticos, en este proyecto, se grabarán videos para simular una iluminación constante.

Se utilizará una cámara webcam con puerto USB o un video que haya sido previamente grabado en un archivo con formato AVI o mp4.

Las herramientas de software utilizadas para la visión artificial o visión por computadora son: Python v3.5, junto a las librerías de OPENCV.

En la tarjeta Raspberry Pi, se instalará el sistema operativo Raspbian, que es una versión simplificada y ligera de Linux, con la finalidad de que se ejecuten a mayor velocidad los algoritmos.

Con todo esto, se concluirá, si es factible implementar este tipo de sistemas de visión, para ser ejecutado a una tasa de velocidad suficiente para la clasificación del limón.

OPENCV

Es una biblioteca libre (Open Source) de visión artificial originalmente desarrollada por Intel. OpenCV significa **Open Computer Visión** (en español, Visión Artificial Abierta). En el 2020: se le menciona como la biblioteca más popular de visión artificial. Se ha utilizado en una gran cantidad de aplicaciones: Detección de movimiento, reconocimiento de objetos, reconstrucción 3D a partir de imágenes, etc.

Su popularidad se debe a que es:

- + Libre: permite ser usada libremente para fines comerciales y de investigación
- + Multiplataforma: para múltiples sistemas operativos GNU/Linux, Mac OS X, Windows y Android, y para diversas arquitecturas de hardware como x86, x64(PC), ARM (celulares y Raspberry).
- + Documentada y explicada: la organización tiene la preocupación activa de mantener la documentación de referencia para desarrolladores lo más completa y actualizada posible.

El proyecto pretende proporcionar un entorno de desarrollo fácil de utilizar y altamente eficiente.

ASPECTOS DE LA PROBLEMÁTICA

DESCRIPCIÓN DE LA REALIDAD PROBLEMÁTICA

En términos prácticos, uno de los grandes problemas en la industria de fruta de nuestra región (Piura, PER) es la clasificación del limón para exportación. De una manera eficiente y cumplimiento con los estándares de calidad internacional impuestos por los países importadores (EEUU, Europa, China).

La visión es uno de los sentidos más importantes de los seres humanos, está se emplea para obtener información lumínica del entorno físico y transformarla en una recreación de la realidad externa. En tal sentido, la clasificación se lleva a cabo por parte de personal contratado, y como todo ser humano tiene cansancio físico, esto conlleva a que después de un cierto tiempo, la clasificación del limón no se haga de una manera correcta.

Los parámetros que se utilizan para clasificar correctamente el limón son: **COLOR, TAMAÑO y DEFECTO.**

Si bien algunas empresas, ya tienen resuelto este problema, daremos una solución con una metodología propia.

Entonces:

¿Será posible implementar un sistema de visión artificial para clasificar el limón utilizando una Raspberry Pi?

JUSTIFICACIÓN E IMPORTANCIA DE LA INVESTIGACIÓN

La clasificación manual del limón, después de cierto tiempo, produce la disminución de la capacidad de hacer correctamente este trabajo. El cansancio afecta negativamente nuestros sentidos, por lo tanto, disminuye la eficiencia de la clasificación.

El diseño de este sistema permite reducir costos en mano de obra, indirectamente también reduce los costos en la producción, disminuye los errores que se puedan cometer en la clasificación (por factores humanos). Por ende, se obtendrá una calidad óptima en la selección de este producto.

La deficiencia en la clasificación del limón, a causa del cansancio físico y fatiga de la visión humana, constituye la problemática que ha dado lugar al desarrollo de esta tesis.

OBJETIVOS**OBJETIVO GENERAL**

Diseñar un sistema de visión artificial para la clasificación del limón utilizando Raspberry Pi.

OBJETIVOS ESPECÍFICOS

- Conocer la existencia de un sistema que clasifique el limón.
- Conocer el tiempo de clasificación del limón de los sistemas actuales: manuales y automatizados.
- Determinar la influencia de la clasificación manual del limón
- Determinar el color, el tamaño y el defecto del limón mediante técnicas de visión artificial.

DELIMITACIÓN DE LA INVESTIGACIÓN

- La investigación se diseñará con imágenes tomadas con iluminación artificial. Se utilizarán videos grabados previamente en el formato mp4.
- El tipo de limón que se usará es el sutil, ya que es el que se exporta en nuestra región Piura.
- No se implementarán partes mecánicas para el transporte del limón, en su lugar se simulará y se capturará el paso de este (videos grabados).
- El sistema se implementará utilizando las librerías de visión artificial OPENCV en el lenguaje de programación Python.

VISIÓN ARTIFICIAL

También conocida como visión por computador o visión técnica, es un subcampo de la inteligencia artificial y es el campo de acción más ambicioso del procesamiento digital de imágenes. Básicamente, el objetivo es automatizar tareas de inspección visual, tradicionalmente desempeñadas por el hombre.

Algunos de los objetivos típicos de la visión artificial incluyen:

- La detección, segmentación, localización y reconocimiento de ciertos objetos en imágenes.
- La evaluación de los resultados
- Registros de diferentes imágenes de una misma escena u objeto, hacer concordar un mismo objeto en diversas imágenes.
- Seguimiento de un objeto en una secuencia de imágenes.
- Mapeo de una escena para generar un modelo tridimensional de la escena; tal modelo podría ser usado por un robot o por un simulador de realidad para navegar dentro de la escena.
- Estimación de las posturas tridimensionales de humanos
- Búsqueda de imágenes digitales por su contenido.

LA IMAGEN DIGITAL

Son el principal ingrediente de lo que se conoce como Visión Artificial y se representan mediante algún tipo de codificación, normalmente en una matriz numérica de dos dimensiones. Captura una escena del entorno.

Existen dos tipos de imágenes utilizadas con frecuencia en Visión Artificial:

- a) Imágenes de intensidad: miden la cantidad de luz que incide en un dispositivo fotosensible (sensor). Ej: una fotografía.
- b) Imágenes de alcance (también llamadas imágenes de profundidad o perfiles de superficie): Estiman directamente la escena en una estructura de tres dimensiones (3D), su fundamento radica en el uso de sensores de distancia ópticos y de algún fenómeno físico para adquirir la imagen. Ej: la imagen que obtiene un oftalmólogo sobre el grado de rugosidad de la córnea de una paciente o las imágenes de un radar.

Aunque la filosofía de los diferentes tipos de imágenes es distinta, en cualquier caso, tras su captura tendremos una matriz de valores en dos dimensiones (2D), es decir, una imagen digital.

DISPOSITIVOS DE CAPTURA DE IMÁGENES

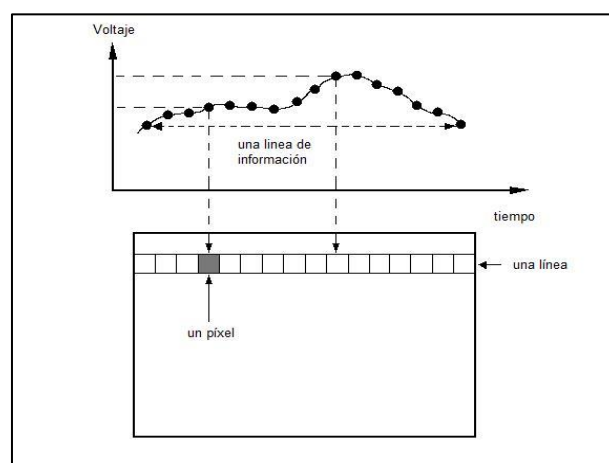
Para la adquisición de imágenes digitales se requieren dos elementos básicos. El primero es un **dispositivo físico sensible (sensor fotosensible)** a una determina banda del espectro electromagnético y que produce una señal eléctrica de salida proporcional al nivel de energía incidente en cualquier instante de tiempo. El segundo, denominado **digitalizador**, es un dispositivo que cumple la función de convertir la señal eléctrica de salida del dispositivo físico en un conjunto discreto de localizaciones del plano de la imagen (**muestreo**) y, después, en la **cuantización** de dicha muestra.

Esto implica:

- Determinar el valor de la imagen continua en cada una de las diferentes localizaciones discretas de la imagen (cada valor localizado de forma discreta se denomina muestra de la imagen).
- Asignar a cada muestra una etiqueta discreta, que es representativa del rango en el que varía la muestra.

Una vez capturada la señal analógica (continua) y cuantificada espacialmente (discreta), se obtiene una imagen digital en el computador, es decir, tendremos una matriz (2D) de números. Éstos son los valores que se manipula con operaciones booleanas mediante software.

En la siguiente fig. se ilustra un ejemplo de cuantización espacial y en amplitud:



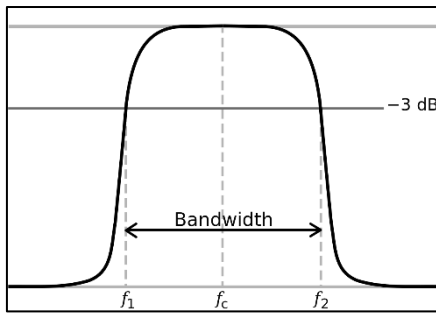
DIGITALIZACIÓN DE UNA SEÑAL ANALÓGICA

MUESTREO Y CUANTIZACIÓN

a) MUESTREO:

Es el proceso de convertir una señal analógica (ej: una función continua en el tiempo o en el espacio) en una secuencia numérica (una función discreta en el tiempo o en el espacio). El teorema de muestreo señala que la reconstrucción (aproximadamente) exacta de una señal continua en el tiempo en banda base (es decir, se refiere a la banda de frecuencias producidas por un transductor, UN MANIPULADOR) a partir de sus N muestras es posible si la señal es **limitada en banda** y la **frecuencia de muestreo** es mayor a dos veces BW de la señal. El teorema de muestreo es comúnmente llamado de Shannon y también conocido como **teorema de Nyquist**.

El proceso de muestreo sobre una señal continua que varía en el tiempo (o en el espacio) es realizado midiendo simplemente los valores de la señal analógica cada T unidades de tiempo (o espacio), llamado intervalo de muestreo. El resultado de este proceso es una secuencia de números, llamadas **muestras**, y en su conjunto son una representación de la imagen original. La **frecuencia de muestreo** f_m es el inverso del intervalo de muestreo $f_m = 1/T$ y se expresa en Hz.



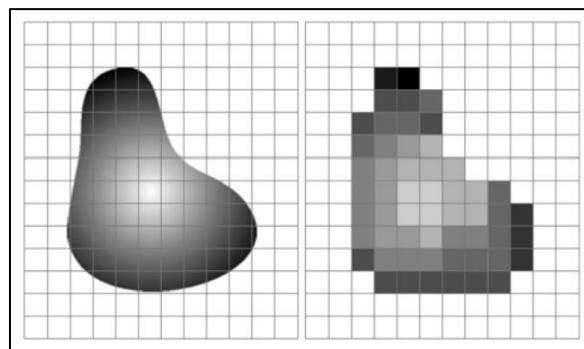
El **ancho de banda (BW)** de una señal analógica es la longitud, medida en HZ, de la extensión de frecuencias en la que se concentra la mayor potencia de la señal

Las condiciones que se deben tomar en cuenta en el proceso de muestreo son:

- Limitar en banda a través de un filtro paso-bajas la señal a muestrear.
- Aplicar el criterio de Nyquist: si conocemos el ancho de banda de la señal analógica, entonces la frecuencia de muestreo (f_m) para lograr una reconstrucción casi perfecta de la señal original deberá ser lo siguiente $f_m \geq 2WB$. Donde f_m es la frecuencia de muestreo que sigue esta condición y se define como **frecuencia de Nyquist**.

DISTORCIÓN: Si las condiciones de muestreo no se satisfacen, se generan replicas de la señal original (alias) que difieren de ésta en su composición, y que además se superponen, generando el efecto de aliasing (solapamiento).

b) CUANTIZACIÓN:



Las imágenes en escala de grises son representadas frecuentemente como matrices de números reales representando las intensidades relativas los píxeles (elementos de la imagen) localizados en las intersecciones de renglones y columnas. Como resultado, las imágenes necesitan dos variables independientes o índices para especificar a cada pixel individualmente; una para los renglones y otra para las columnas.

Por otra parte, las imágenes a color consisten regularmente consisten de una composición de tres imágenes separadas en escala de grises, cada una representa los tres colores primarios; rojo, verde y azul; comúnmente conocido como modelo RGB. Existen otros espacios de color que usan 3 vectores para los colores, por ejemplo, HSV.

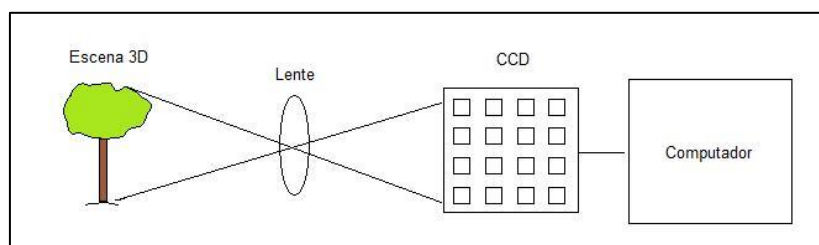
Una imagen puede ser continua respecto al eje de coordenadas X / Y, pero también puede ser continua en amplitud. Para convertir una imagen continua (analógica) a su forma digital, como se vio anteriormente, a la digitalización de los valores de las coordenadas se le llama **muestreo**. Mientras que el proceso de digitalizar la amplitud es llamado **cuantización**. Normalmente, el proceso de adquisición de la imagen se realiza usando un arreglo de sensores. El número de sensores dentro de la matriz establece los **límites del muestreo** en ambas direcciones. En cambio, la **digitalización de la amplitud o cuantización** la realiza cada sensor asignando un valor discreto a ciertos intervalos de amplitudes continuas.

CALIDAD: Claramente se observa que la calidad de la imagen esta determinada en gran medida por el número de muestras (resolución de la matriz) y los niveles discretos de gris usados durante el muestreo y la cuantización respectivamente.

Supongamos que tenemos una señal analógica, que bien podría ser una línea de video analógica. Esta señal (línea) analógica de video se convierte a una imagen digital muestreando la señal analógica a intervalos determinados. El procedimiento consiste en medir voltaje se la señal a intervalos de tiempo fijos. El valor del voltaje en cada instante se convierte a un número que es almacenado y se corresponde con la intensidad de la imagen en ese punto. La intensidad en cada punto depende de las propiedades intrínsecas del objeto que se está viendo y de las condiciones de luz de la escena. Repitiendo este procedimiento para todas las líneas de video que constituyen una imagen, se pueden grabar los resultados obtenidos en el computador, obteniendo una imagen digital que, en definitiva, es una matriz de números.

Además de las cámaras, uno de los sensores más usado para la visión artificial son lo dispositivos de acoplamiento de carga (CCD). Entre los dispositivos CCD, también capaces de producir una señal continua de video, cabe distinguir dos categorías: sensores de exploración de línea y sensores de exploración de área. Estos sensores CCD se basan en elementos semiconductores llamados fotosites. Los fotones procedentes de la escena excitan el elemento semiconductor, de forma que el grado de excitación es proporcional a la cantidad de carga acumulada en el fotosite y, por lo tanto, a la intensidad luminosa en ese punto.

Los fotosites se pueden representar en forma de matriz como se muestra en la siguiente figura.



CAPTURA DE UNA IMAGEN 3D POR UN DISPOSITIVO CCD

La señal de estos sensores se procesa en el propio sensor (cámara) o en otro dispositivo (tarjeta de procesamiento de imágenes digitales) y los valores digitales obtenidos se envían a un computador.

IMÁGENES BLANCO/NEGRO Y COLOR

En definitiva, e independientemente del tipo de sensor utilizado, la imagen que ha de ser tratada por el computador se presenta digitalizada espacialmente en forma de matriz con una resolución de $M \times N$ elementos.

a) BLANCO / NEGRO

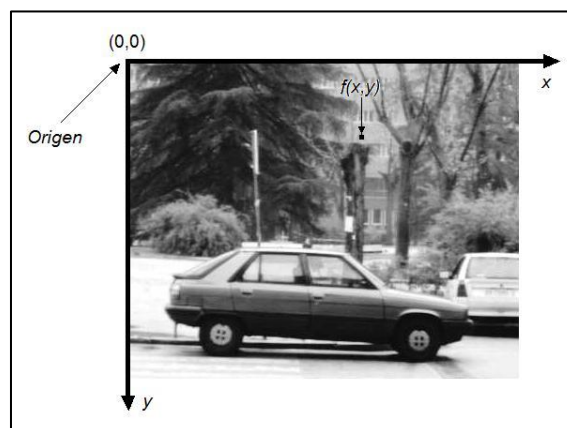
Si la imagen es en Blanco y Negro (B/N), se almacena un valor por cada píxel. Suele utilizar un rango de valores para su representación, que generalmente es 0 a $2^n - 1$. Uno de los valores más utilizados de n es 8; esto significa que el rango de valores para este caso varía de 0 a 255. En donde, el 0 representa el negro absoluto y el 255, el blanco absoluto. Esto indica que podemos tener una precisión en los grises posibles de 256. El hecho de utilizar 256 niveles (escalas) es porque con 8 bits (1 Byte) del computador se pueden codificar 256 valores distintos que van desde la combinación 0000 0000, que representa el nivel 0, hasta la combinación 1111 1111, que representa el nivel 255.

b) IMÁGENES EN COLOR

En el caso de las imágenes en color, los elementos de la matriz vienen dados por tres valores, que representan cada uno de los componentes básicos del color en cuestión. Estos componentes son el Rojo (R), Verde (G) y Azul (B), el conocido modelo RGB. En este caso el conjunto de valores (0,0,0) es el negro absoluto; el (255,255,255), el blanco absoluto; el (255,0,0), el rojo puro; el (0,255,0), el verde puro; el (0,0,255), el azul puro. Como es lógico, la combinación de distintos valores proporciona otros colores.

REPRESENTACIÓN DE IMÁGENES DIGITALES

Como ya se ha mencionado antes, el término imagen se refiere a una función de intensidad bidimensional, la cual puede ser representada como $f(x,y)$, donde x e y , o bien, i y j son las coordenadas espaciales y el valor de f en cualquier punto (x,y) o (i,j) es proporcional al nivel de gris de la imagen en ese punto (PÍXEL).



CONVENCIÓN DE EJES UTILIZADA PARA LA REPRESENTACIÓN DE IMÁGENES DIGITALES

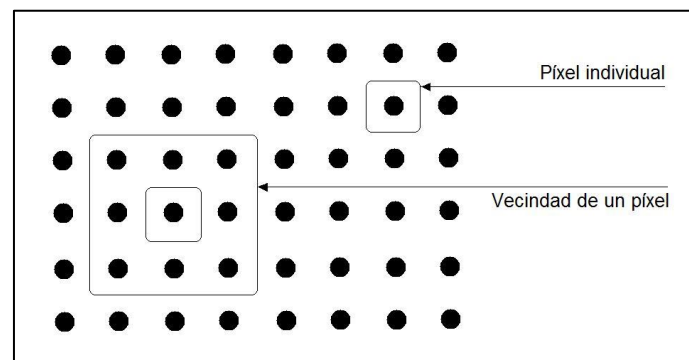
PROCESAMIENTO Y ANÁLISIS DE IMÁGENES DIGITALES

- El **procesamiento de imágenes** puede ser visto como una transformación de una imagen a otra, es decir, a partir de una imagen, se obtiene otra imagen modificada.
- El **análisis** es una transformación de una imagen en algo distinto a una imagen; en consecuencia, el análisis es un determinado tipo de información representando a una descripción o una decisión. En la mayoría de los casos, las técnicas de análisis de imágenes digitales son aplicadas a imágenes que han sido procesadas previamente. Desde el punto de vista de un observador humano, el análisis de imágenes es una tarea fácil y rápida. La capacidad de percepción humana permite procesar rápidamente una imagen para detectar en ella características de interés, por ejemplo, bordes o regiones. Es decir, análisis del contenido.

PROCESAMIENTO BÁSICO DE IMÁGENES

El procesamiento de datos en el sistema de visión puede enfocarse desde 2 perspectivas:

- a) Alteración píxel a píxel de los datos en una escala global (individuales).
- b) Operaciones basadas en múltiples puntos (vecindad).



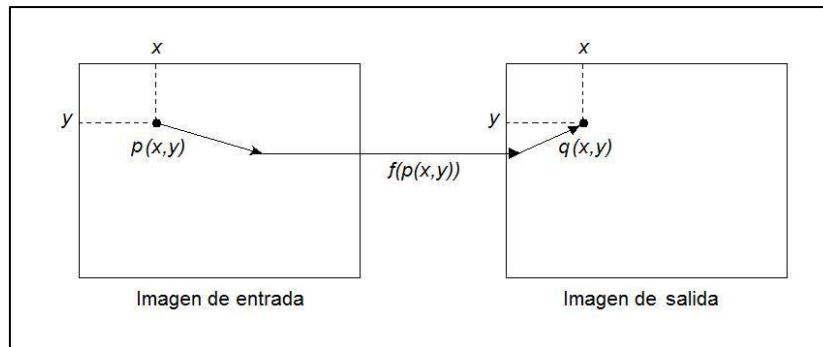
La generación de un nuevo píxel en una imagen será bien una función del valor de cada píxel en su localización individual, o bien de los valores de los píxeles en la vecindad de un píxel dado.

Existen otras operaciones, que no se clasifican ni como individuales ni como vecindad, ya que transforman las imágenes por otros procedimientos; son transformaciones que operan globalmente sobre los valores de intensidad de la imagen. Cuyo efecto es un realzado de la imagen original, operaciones aritméticas y lógicas, estas últimas basadas en la teoría del álgebra de Boole, y operaciones que realizan transformaciones geométricas, sin modificar los valores de intensidad.

OPERACIONES INDIVIDUALES

Las operaciones individuales implican la generación de una nueva imagen modificando el valor del píxel de una simple localización basándose en una regla global aplicada a cada localización de la imagen original. El proceso consiste en obtener el valor del píxel de una localización dada en la imagen, modificándolo mediante una operación lineal o no lineal y colocando el valor

del nuevo píxel en la correspondiente localización de la imagen nueva. El proceso para todas y cada una de las localizaciones de los píxeles en la imagen original.



OPERACIÓN INDIVIDUAL

Como se aprecia en la figura anterior, el operador individual es una transformación uno a uno. El operador f se aplica a cada píxel en la imagen o sección de la imagen de entrada y la salida depende únicamente de la correspondiente del correspondiente píxel de entrada; la salida es independiente de los píxeles adyacentes. La función transforma el valor del nivel gris de cada píxel en la imagen y el nuevo valor se obtiene a través de la ecuación:

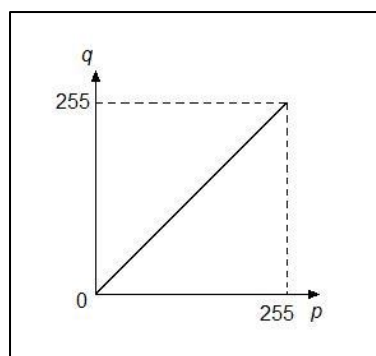
$$q(x,y) = f(p(x,y))$$

Observar que sigue manteniendo la misma posición, pero los valores han cambiado. Por lo tanto, la imagen resultante es de la misma dimensión que la original.

OPERADOR IDENTIDAD

Este operador crea una imagen de salida que es sencillamente idéntica a la imagen de entrada. La función de transformación es:

$$q = p$$



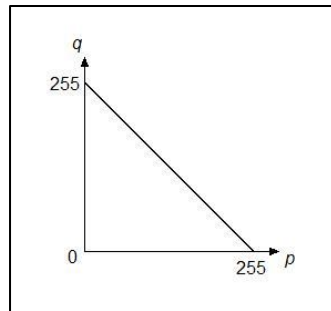
REPRESENTACIÓN DEL OPERADOR IDENTIDAD

OPERADOR INVERSO O NEGATIVO

Este operador crea una imagen de salida que es la inversa de la imagen de entrada. Este operador es útil en diversas aplicaciones tales como imágenes

médicas. Para una imagen con valores de gris en el rango de 0 a 255 la función de transformación es:

$$q = 255 - p$$

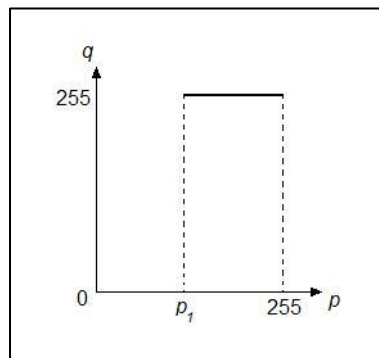


REPRESENTACIÓN DEL OPERADOR INVERSO

OPERADOR UMBRAL

Esta clase de transformación crea una imagen de salida binaria a partir de una imagen de grises, donde el nivel de transición está dado por el parámetro de entrada p_1 . La función de transformación es la siguiente:

$$q = \begin{cases} 0 & \text{para } p \leq p_1 \\ 255 & \text{para } p > p_1 \end{cases}$$



REPRESENTACIÓN DEL OPERADOR UMBRAL

TRANSFORMACIÓN DE VECINDAD

En las operaciones, el valor del nuevo píxel en la imagen de salida depende de una combinación de los valores de los píxeles en una vecindad de la imagen original. Es decir, se refiere a convertir una imagen en otra diferente, por medio de operaciones del píxel teniendo en cuenta la información que ofrecen los píxeles vecinos.

Dentro de la categoría de operaciones de vecindad se incluye las operaciones de filtrado. Las operaciones de filtrado tienen la particularidad de eliminar un determinado rango de frecuencias de una imagen.

NOCIONES Y PROPIEDADES DE VECINDAD

$(x-1, y-1)$	$(x, y-1)$	$(x+1, y-1)$
$(x-1, y)$	(x, y)	$(x+1, y)$
$(x-1, y+1)$	$(x, y+1)$	$(x+1, y+1)$

Se dice que todo píxel p , de coordenadas (x, y) , tiene cuatro píxeles que establecen con él una relación de vecindad horizontal o vertical, que son:

HORIZONTAL: $(x-1, y)$ y $(x+1, y)$

VERTICAL: $(x, y-1)$ y $(x, y+1)$

Estos 4 píxeles definen lo que se conoce como entorno de vecindad-4 y nos referimos a ellos como $E_4(p)$.

Los cuatro vecinos diagonales de p tienen coordenadas:

$(x-1, y-1), (x+1, y-1), (x-1, y+1), (x+1, y+1)$

Y nos referimos a ellos como $E_D(p)$. Estos píxeles junto con los $E_4(p)$ se llaman los vecinos-8 de p y se denotan como $E_8(p)$.

Existen excepciones dadas cuando el píxel (x, y) es un punto del borde de la imagen, en cuyo caso algunos de los vecinos definidos anteriormente simplemente no existen.

CONECTIVIDAD

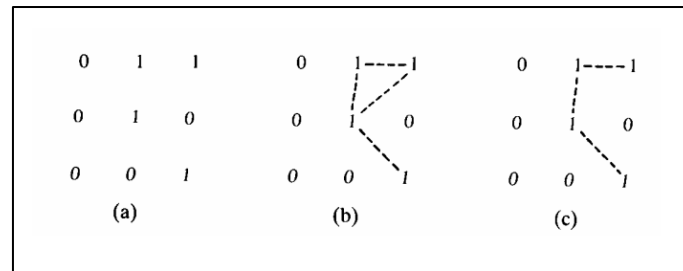
Sea V el conjunto de valores de intensidad de los píxeles que se permiten estén adyacentes, por ejemplo, si sólo se desea que exista conectividad entre los píxeles con intensidades 80, 81 y 83, entonces $V = \{80, 81, 83\}$.

Consideremos tres tipos básicos de conectividad:

- **Conectividad-4:** Dos píxeles p y q , con valores de V , están 4-conectados si q están en el conjunto $E_4(p)$.
- **Conectividad-8:** Dos píxeles p y q , con valores de V , están 8-conectados si q están en el conjunto $E_8(p)$.
- **Conectividad-m (mixta):** Dos píxeles p y q con valores de V están m-conectados o mezclados si:
 - q está en el conjunto $E_4(p)$, o bien
 - q está en $E_D(p)$ y $E_4(p) \cap E_4(q)$ está vacío.

La conectividad entre píxeles es un concepto utilizado para establecer los límites en objetos y regiones de los componentes en una imagen. La conectividad mezclada es una modificación de la conectividad-8, y se introduce para eliminar las conexiones multi-trayectoria.

Un píxel p es contiguo a otro píxel q si están conectados. Se puede definir la adyacencia-4, 8 o m, dependiendo del tipo de conectividad específica. Dos subconjuntos imagen S_1 y S_2 son contiguos si algún píxel de S_1 es contiguo a algún píxel de S_2 .



En la fig.: (a) arreglo de píxeles; (b) Píxel central con conectividad-8; (c) conectividad-m (mezclada / mixta) del mismo píxel.

Un camino desde el píxel p con coordenadas (x,y) hasta un píxel q con coordenadas (s,t) es una secuencia de varios píxeles con coordenadas:

$$(x_0, y_0), (x_1, y_1), \dots, (x_n, y_n)$$

Donde $(x_0, y_0) = (x, y)$ y $(s, t), (x_i, y_i)$ son adyacentes a $(x_{(i-1)}, y_{(i-1)})$, con $1 \leq i \leq n$, y n es la longitud del camino. Se pueden definir caminos-4, 8 o m dependiendo del tipo de adyacencia usada.

Si p y q son píxeles de una imagen S , entonces p está conectado a q en S si existe un camino desde p hasta q formado de píxeles pertenecientes a S . Es decir, dado un píxel p cualquiera perteneciente a S , al conjunto de píxeles de S que están conectados a p se llaman **componente conectado de S** . Se deduce que dos píxeles cualesquiera de un componente conectado están a su vez conectados entre sí y que los componentes conectados distintos son disjuntos.




Un camino simple es un camino sin píxeles repetidos y un camino cerrado es un camino simple en el cual el primer píxel es un vecino del último.

OPERACIONES DE FILTRADO

Las operaciones de filtrado basan su operatividad en la convolución de la imagen utilizando el denominado **núcleo de convolución**. Cabe distinguir dos tipos de filtros: paso alto y paso bajo, que en teoría de señales supone que los primeros dejan pasar las altas frecuencias de la señal y los segundos, las bajas. En el caso de las imágenes nos referimos a frecuencias espaciadas. De una forma, las altas frecuencias se asocian a cambios bruscos de intensidad en pequeños intervalos espaciales, es decir, bordes. Mientras que las bajas frecuencias se refieren a los cambios lentos (suaves) en la intensidad.

FILTROS PASO BAJO

Su objetivo es suavizar la imagen, son útiles cuando se supone que la imagen tiene gran cantidad de ruido y se requiere eliminar (reducir). También pueden utilizarse para resaltar la información correspondiente a una determinada escala (tamaño de la matriz de filtrado); por ejemplo, en el caso de que se quiere eliminar la variabilidad asociada a los tipos de cubierta presentes en la imagen uniformizando de esta manera su respuesta.

OPERACIÓN	NÚCLEO	IMAGEN RESULTANTE
Identidad	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Desenfoque de cuadro (Filtro de media)	$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Desenfoque gaussiano 3x3	$\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

EN LA TABLA SE APRECIAN NÚCLEOS REPRESENTATIVOS DE FILTROS PASO BAJO

- **Un núcleo:** También llamado kernel, matriz de convolución o máscara es una matriz pequeña que se usa para desenfoque, enfoque, realce, detección de bordes y más. Esto se logra realizando una convolución entre un núcleo y una imagen.
- **Filtro gaussiano:** Simulan una distribución gaussiana bivariable. El valor máximo aparece en el píxel central y disminuye hacia extremos. El resultado es un conjunto de valores entre 0 y 1. Para transformar la matriz obtenida a una matriz de números enteros se divide toda la matriz por el menor de los valores obtenidos. La expresión matemática para calcularla es:

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

$$G(x, y) = \frac{g(x, y)}{\min_{x, y}(g(x, y))}$$




Donde: s es el parámetro de desviación típica. Cuanto menor sea s, más rápido disminuyen los valores hacia los extremos.

FILTRO PASO ALTO

Su objetivo es resaltar las zonas de mayor variabilidad eliminando lo que sería la componente media, precisamente la que detectan los filtros de paso bajo. Por otra parte, si partimos de la premisa que la respuesta de cada píxel está **contaminada** por la de los píxeles vecinos ya que, considerando la superficie terrestre como lambertiana, la radiación reflejada por un píxel se reparte hacia los píxeles vecinos. Los filtros de pasa alto consiguen en parte eliminar esta contaminación.

En física, una superficie de Lambert o **lambertiana** es una superficie ideal que refleja la energía incidente desde una dirección igual en todas las direcciones, por lo cual, al variar el punto de vista, su luminancia no cambia.

En matemáticas, y en particular en el análisis de funciones, una **convolución** es un operador matemático que transforma dos funciones f y g en una tercera función que expresa cómo la forma de una es modificada por la otra. Se define como la integral del producto de las dos funciones después de que una se invierte y se desplaza. Y la integral se evalúa para todos los valores de desplazamiento, produciendo la función de convolución. **f*g**.

OPERACIÓN	NÚCLEO	IMAGEN RESULTANTE
Identidad	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Enfocar	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Detección de bordes	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	

EN LA TABLA SE APRECIAN NÚCLEOS REPRESENTATIVOS DE FILTROS PASO ALTO

- **Sustracción de la media:** Si a la imagen original se le resta el resultado de pasarle un filtro de paso bajo se consigue resaltar la variabilidad local (los cambios bruscos de intensidad).
- **Filtros basados en las derivadas:** La segunda derivada (la derivada de la derivada) nos da información acerca de la forma (Ej: si ladera es recta, cóncava o convexa) de la superficie. Nos va informar de como son estos cambios, más o menos bruscos, que se producen entre píxeles contiguos.

HISTOGRAMA

El histograma de una imagen es la representación analítica de la distribución relativa de cada valor posible de píxel de la imagen, y en caso de imágenes grises de 8 bits será un vector de 256 componentes, siendo la componente i el número de píxeles de nivel i en la imagen, dividido entre el número total de píxeles.

El histograma es **formalmente** la función estadística de la densidad de probabilidad en forma discreta de los distintos niveles de grises dentro de la imagen.

Si se integra, el resultado es la función de distribución $\omega[i]$.

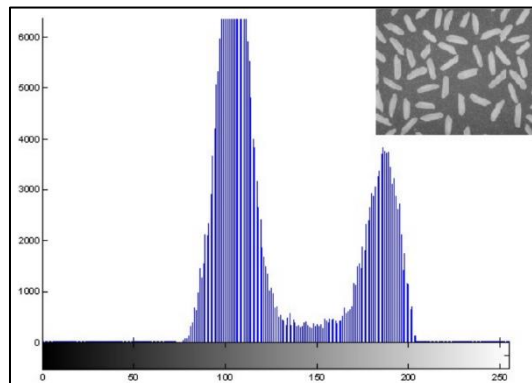
$$\text{histograma}[i] = \frac{N_{\text{píxeles de nivel } i}}{N_{\text{total}}}, \quad 0 \leq i \leq 255$$

$$\omega[i] = \sum_{j=0}^i \text{histograma}[j], \quad 0 \leq i \leq 255$$

Es decir, supongamos una imagen S en niveles de grises i , siendo el rango de 256 tonos de gris ($0 \leq i \leq 255$). El **histograma** de la imagen consiste en una gráfica donde se muestra el número de píxeles de cada valor de gris ($N_{\text{píxeles de nivel } i}$) que aparecen en la imagen, dividido entre el número total de píxeles (N_{total}).

El histograma es utilizado para binarizar una imagen digital, es decir, convertirla en una imagen en blanco y negro, de tal manera que se preserven las propiedades esenciales de la imagen. La forma usual de binarizar una imagen es eligiendo un valor de **umbral** (u), dentro de los niveles de grises, tal que el histograma forme un “valle” en ese nivel. Dicho de otra forma, todos los niveles de grises menores que u se convierten en 0 (negro), y todos los mayores que u se convierten en 255 (blanco).

Antes de binarizar una imagen para segmentar los objetos de su fondo, se debe fijar el umbral (u) adecuado. Si la imagen presenta unas zonas de objeto y otras de fondo bien diferenciadas, el histograma tendrá una forma como se muestra en la ilustración.



En la fig.: Histograma de imagen con objetos fondo diferenciados, y el umbral adecuado se sitúa en el valle de la curva.

OPERACIONES MORFOLÓGICAS

Las operaciones morfológicas son métodos que tiene su origen en la teoría de conjuntos. En el procesamiento de imágenes, se suele aplicar sobre imágenes binarias, donde se ha hecho una segmentación previa, separando el fondo (marcado como 0) de los objetos de interés (marcados como 1). Una imagen binaria es un conjunto de valores organizados en una cuadrícula, en la cual cada píxel sólo puede tener dos valores, 0 ó 1. Como es lógico suponer, al tener una imagen en esas condiciones es mucho más fácil encontrar y distinguir características estructurales.

Las operaciones morfológicas procesan estas imágenes binarias basándose en la forma de sus objetos de interés. En general, Toman una imagen binaria como entrada y dan como resultado otra imagen binaria. El valor de cada píxel en la imagen de salida se obtiene con operaciones no lineales sobre el píxel de entrada y sus vecinos. En general, las operaciones morfológicas se usan para:

- Supresión de ruidos
- Simplificación de formas
- Destacar la estructura de objetos (detección de envolvente, amplificación, reducción).
- Descripción de objetos (área, perímetro).

Las operaciones morfológicas suelen aplicarse sobre imágenes binarias, con el objetivo de encontrar y distinguir características estructurales.

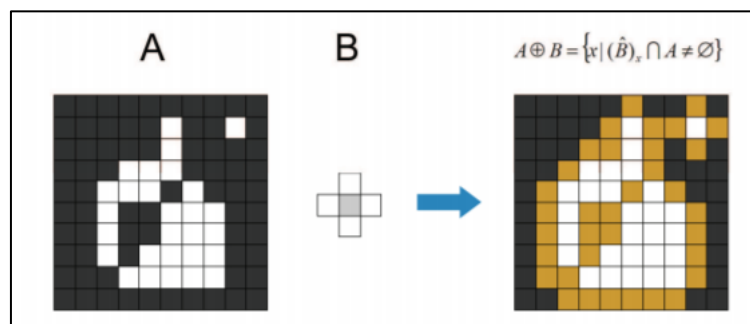
Las dos operaciones morfológicas más conocidas son la erosión y la dilatación, cuyo objetivo principal es simplificar las imágenes para su posterior análisis, conservando todas sus características.

DILATACIÓN

Se define como “aquellos píxeles x tales que la intersección de la estructura B situada sobre x y la imagen A es distinto que el conjunto vacío”.

Considerando que para la intersección de A y B sólo se toman en cuenta los píxeles que correspondan a los objetos de primer plano (píxeles ‘1’). El elemento estructural B indica de qué forma se llevará a cabo la dilatación.

La operación de **dilatar una imagen se puede describir como** un crecimiento de los píxeles situados alrededor de los bordes de los objetos. En general, este método marca como ‘1’ todos los píxeles que formen parte del fondo de la imagen, pero que al mismo tiempo estén en contacto directo con el objeto. Esto permite aumentar en uno el nivel de píxeles en el perímetro de cada objeto, que sufre un crecimiento de tamaño, y al mismo tiempo permite rellenar posibles huecos dentro del mismo.



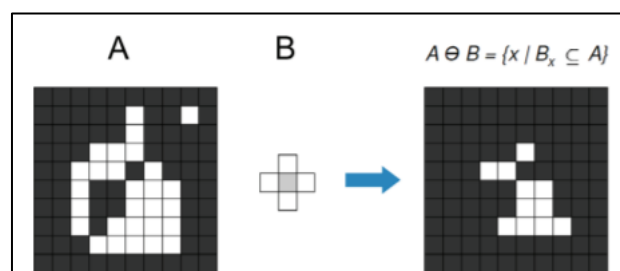
RESULTADO DE LA DILATACIÓN SOBRE UNA IMAGEN BINARIA

El estado de cualquier píxel de salida es obtenido aplicando una regla determinada al píxel de entrada y a sus vecinos. La **regla para obtener un obtener una imagen dilatada** es la siguiente: <<Si cualquier píxel vecino de entrada es ‘1’, entonces el píxel de salida también es ‘1’. En cualquier otro caso el píxel de salida será ‘0’>>.

EROSIÓN

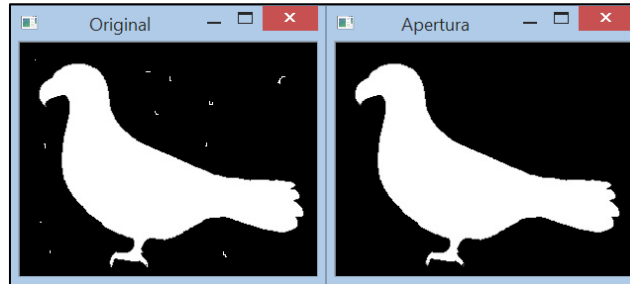
Se define como “aquellos píxeles x tales que la estructura B situada sobre x pertenezca en su totalidad a la imagen A ”. Considerando que para la condición “ B pertenece a A ” sólo se toman en cuenta los píxeles de los objetos en primer plano (aquellos marcados como ‘1’). La erosión es la operación morfológica dual a la dilatación y usualmente se concibe como una reducción de la imagen original.

La erosión, realiza la operación contraria a la dilatación. Este método marca como ‘0’ todos los píxeles que pertenezcan al borde de un objeto.



RESULTADO DE LA EROSIÓN SOBRE UNA IMAGEN BINARIA

El estado de cualquier píxel de salida es obtenido aplicando una regla determinada al píxel de entrada y a sus vecinos. La regla para obtener una imagen erosionada es la siguiente: <<Si todos los píxeles vecinos del píxel de entrada están a '1', entonces el píxel de salida será '1'. En cualquier otro caso, el píxel de salida será '0'>>.



RESULTADO DE LA DILATACIÓN Y EROSIÓN SOBRE UNA IMAGEN BINARIA

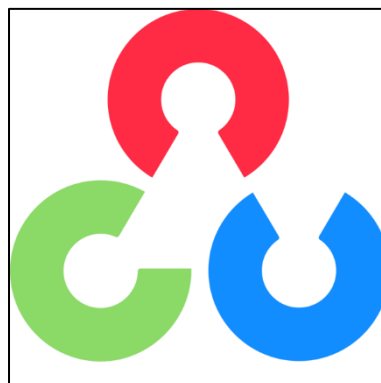
Aplicando en conjunto estas dos operaciones, erosión y dilatación, se obtienen interesantes resultados en cuanto a la eliminación de ruido o de objetos demasiado pequeños para resultar de interés.

CONCEPTO DE OPENCV

Es una biblioteca multiplataforma que permite desarrollar aplicaciones de visión por ordenas en tiempo real. Se centra en el procesamiento de imágenes, captura de vídeo y análisis, incluyendo características de detección de rostros y objetos.

OpenCV es una biblioteca libre de visión por ordenador originalmente desarrollada por Intel; desde que apareció en su primera versión alfa en el mes de enero de 1999, se ha utilizado en infinidad de aplicaciones. Desde sistemas de seguridad con detección de movimiento, hasta aplicativos de control de procesos donde es necesario el reconocimiento de objetos.

Contiene más de 500 funciones que abarcan una gran gama de área en el proceso de visión, como reconocimiento de objetos (faces), calibración de cámaras, visión estéreo y visión robótica.



LOGO DE OPENCV

CARACTERÍSTICAS PRINCIPALES

- Optimizado para procesamiento de imágenes en tiempo real y aplicaciones de visión artificial.
- La interfaz primaria de OpenCV está en C++
- También hay interfaces C, Python y JAVA completas
- Las aplicaciones OpenCV se ejecutan en Windows, Android, Linux, Mac e iOS.
- Optimizado para procesadores Intel.

MÓDULOS OPENCV

OpenCV tiene una estructura modular. A continuación, se enumeran los módulos principales de OpenCV.

- **NÚCLEO:** Este es el módulo básico de OpenCV. Incluye estructuras de datos básicas (por ejemplo, estructura de datos Mat) y funciones básicas de procesamiento de imágenes. Este módulo también es ampliamente utilizado por otros módulos como highgui, etc.
- **HIGHGUI:** Este módulo ofrece funciones sencillas de interfaz de usuario, varios códecs de imagen y vídeo, capacidades de captura de imagen y vídeo, manipulación de ventanas de imágenes, manejo de barras de seguimiento y eventos de ratón, etc. Si desea capacidades de interfaz de usuario avanzadas, debe utilizar marcos de interfaz de usuarios como Qt, WinForms, etc.
- **Imgproc:** Este módulo incluye algoritmos básicos de procesamiento de imágenes. Incluyendo filtrado de imágenes, transformaciones de imagen, conversiones de espacio de color y etc.
- **Vídeo:** Este módulo de análisis de video incluye algoritmos de seguimiento de objetos, algoritmos de sustracción de fondo y etc.
- **Objdetect:** Esto incluye algoritmos de detección y reconocimiento de objetos para estándar.

MODELOS DE COLOR

Los modelos de color se pueden dividir en dos clases de modelos:

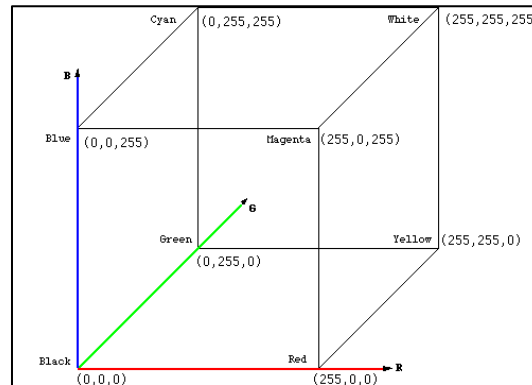
- a) **MODELOS SENSORIALES:** Orientados a los equipos (es decir, las cámaras, monitores, etc.)
- b) **MODELOS PERCEPTUALES:** Se asemejan más a la percepción humana del color y que, en general, están orientados al procesamiento de imágenes y visión artificial.

MODELOS SENSORIALES

Dentro de los modelos sensoriales de color existen 3 modelos más comúnmente utilizados: RGB, CMY e YIQ.

a) MODELO RGB

Es modelo básico de color que utiliza las componentes primarias rojo, verde y azul, normalizadas. De esta forma los colores se representan en coordenadas dentro de un cubo unitario.



CUBO UNITARIO DE COLOR PARA MODELO RGB (1 BYTE)

Cada color se representa como un vector del origen y la diagonal principal corresponde a la escala de grises. En este modelo se basan las cámaras y receptores de TV. Sin embargo, se tienen problemas al aplicarlo a procesamiento de imágenes (ecualización) y visión (no lineal).

b) MODELO CMY

Se basa en los colores secundarios (cian, magenta, amarillo). Este se puede obtener del modelo de RGB de la siguiente forma:

$$\begin{matrix} C & 1 & R \\ M & 1 - G \\ Y & 1 & B \end{matrix}$$

FÓRMULA PARA OBTENER CMY

Se usa este modelo al combinar colores (depósito de segmentos) en papel, como en impresoras y copiadoras de color.

c) MODELO YIQ

En el modelo YIQ se separa la información de intensidad o luminancia (Y) de la información de color (I, Q). Se obtiene mediante la siguiente transformación a partir de las componentes del RGB:

$$\begin{matrix} Y & 0.299 & 0.587 & 0.114 & R \\ I & 0.596 & -0.275 & -0.231 & G \\ Q & 0.212 & -0.523 & 0.311 & B \end{matrix}$$

FÓRMULA PARA OBTENER YIQ

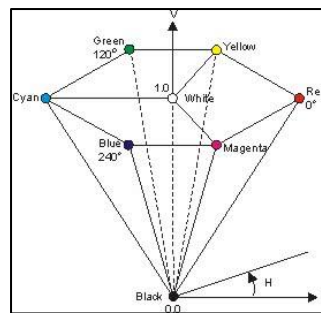
Este es el sistema que se utiliza para la transmisión de TV a color. Tiene dos ventajas: (i) la separación de la información de luminancia para compatibilidad con receptores de blanco y negro y, (ii) el uso de mayor ancho de banda (bits) que es más importante para la percepción humana.

MODELOS PERCEPTUALES

Los sistemas anteriores están orientados a los equipos, mientras que los siguientes modelos, llamados modelos perceptuales, tiene cierta similitud con la percepción humana, por lo que están más enfocados a visión artificial. Estos sistemas, generalmente, utilizan una representación en base a los parámetros perceptuales: croma (H), saturación (S) e intensidad (I).

a) MODELOS HSV

El modelo HSV (Hue, Saturación, Value) se obtiene deformando el cubo RGB de forma que se convierte en una pirámide hexagonal invertida. En el vértice se tiene el negro, en las esquinas del hexágono los 3 primarios y secundarios, y en su centro el blanco. El modelo HSV se ilustra en forma geométrica en la siguiente figura.



MODELO DE COLOR HSV

De esta forma el eje vertical representa la brillantez o valor (V), el horizontal la saturación (S) y el ángulo de la proyección horizontal el croma (H). La conversión de RGB a HSV se logra mediante las siguientes ecuaciones:

$$\begin{aligned}
 V &= M; [0,1] \\
 \text{Si: } M = m, S &= 0; \text{ sino, } S = \frac{(M - m)}{M}; [0,1] \\
 \text{Si: } m = B, H &= \frac{120(G - m)}{R + G - 2m}; [0,360] \\
 \text{Si: } m = R, H &= \frac{120(B - m)}{B + G - 2m}; [0,360] \\
 \text{Si: } m = G, H &= \frac{120(R - m)}{R + B - 2m}; [0,360]
 \end{aligned}$$

FÓRMULAS DE CONVERSIÓN DE RGB A HSV

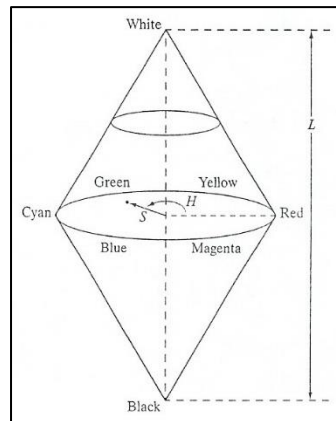
Donde:

$m = \text{Min}(R;G;B)$ y $M = \text{Max}(R;G;B)$. La brillantez (V) y saturación (S) están normalizadas (entre cero y uno) y el croma (H) está entre 0 y 360 grados.

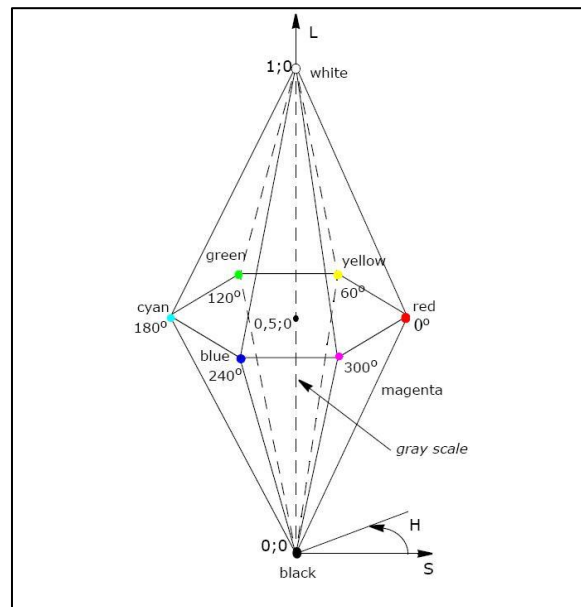
b) MODELO HLS

El modelo HLS (Hue, Level, Saturación) se basa en coordenadas polares en 3 dimensiones, obteniéndose un espacio en forma de 2 conos unidos en su base. El vértice inferior corresponde a negro, el superior a blanco, el eje vertical representa la brillantez (L), el horizontal la saturación (S) y el

ángulo de la proyección horizontal el croma (H). El espacio geométrico del modelo HLS se muestra en la figura:



MODELO DE COLOR HLS



UN ESPACIO EN FORMA DE 2 PIRÁMIDES UNIDAS POR SU BASE

COMPARACIÓN ENTRE MODELOS

Desde el punto de vista de visión artificial, los aspectos importantes a evaluar en los diferentes modelos son:

- **LINEALIDAD:** que exista una relación lineal entre los atributos del color y la percepción del color.
- **UNIFORMIDAD:** que el espacio de color sea uniforme en cuanto a su correspondencia con la percepción del color.
- **SINGULARIDADES:** que no existan singularidades en el espacio, es decir, puntos donde haya cambios bruscos en cuanto a la relación con la percepción del color.
- **ANALOGÍA A LA PERCEPCIÓN HUMANA:** que el modelo se asemeje a la forma en que los humanos percibimos el color.

LA RASPBERRY PI

¿QUÉ ES UNA RASPBERRY PI?

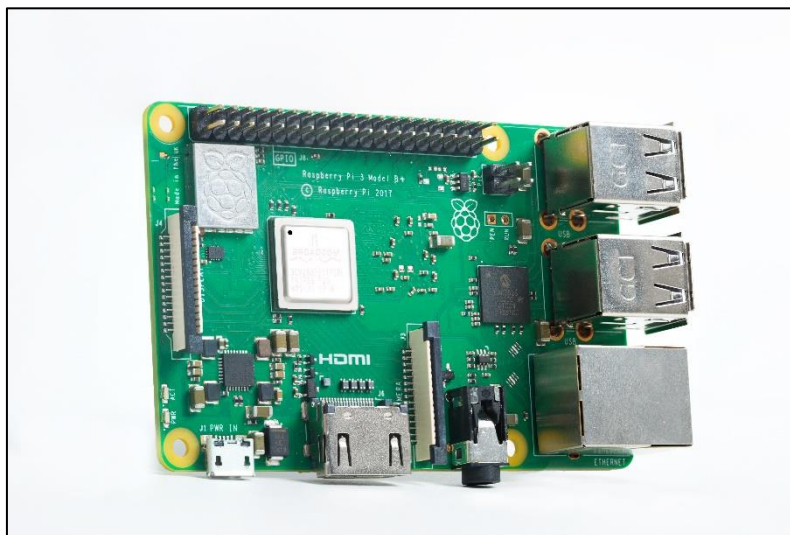
La Raspberry Pi es una computadora del tamaño de una tarjeta de crédito, de bajo costo, que se conecta a un monitor y utiliza un teclado y un mouse estándar. A nivel lógico, es capaz de hacer todo lo que esperaría que hiciera una computadora de escritorio. Además, la Raspberry Pi tiene la capacidad de interactuar con el mundo exterior y se ha utilizado en una amplia gama de proyectos de procesamiento digital.

RASPBERRY PI 3 MODEL B +

La minicomputadora escogida para este proyecto es una **Raspberry Pi 3 Model B+**, que es la versión final de la gama Raspberry Pi 3.

Especificación técnica:

- Broadcom BCM2837B0, Cortex-A53 (ARMv8) SoC de 64 bits a 1,4 GHz
- SDRAM LPDDR2 de 1 GB
- LAN inalámbrica IEEE 802.11.b / g / n / ac de 2,4 GHz y 5 GHz, Bluetooth 4.2, BLE
- Gigabit Ethernet sobre USB 2.0 (rendimiento máximo 300 Mbps)
- Cabecera GPIO extendida de 40 pines
- HDMI de tamaño completo
- 4 puertos USB 2.0
- Puerto micro SD para cargar su sistema operativo y almacenar datos
- Entrada de alimentación de 5 V / 2,5A DC
- Soporte Power-over-Ethernet (PoE)



RASPBERRY PI 3 MODEL B+

En general:

- Procesador de 4 núcleos a 1,4 GHz y arquitectura 64 bits
- LAN inalámbrica de doble banda
- Bluetooth 4.2 / BLE

- Ethernet más rápido (Gigabit Ethernet hasta 300 Mbps)
- Compatibilidad con alimentación PoE

La CPU de las Raspberry Pi 3+ también del fabricante Broadcom se llama BCM2837B0 y es una CPU con juego de instrucciones de 64bits de la familia ARMv8.

El procesador tiene 4 núcleos y de tipo Cortex-A53 a una velocidad de 1.4GHz, el chip incluye una GPU con dual core a 400MHz. El ARM Cortex-A53 incluye 32KB de memoria caché a nivel 1 y 512KB de caché de nivel 2. Además, de estas cubierto por una película metálica, heat spreader, para ayudar a disipar el calor.

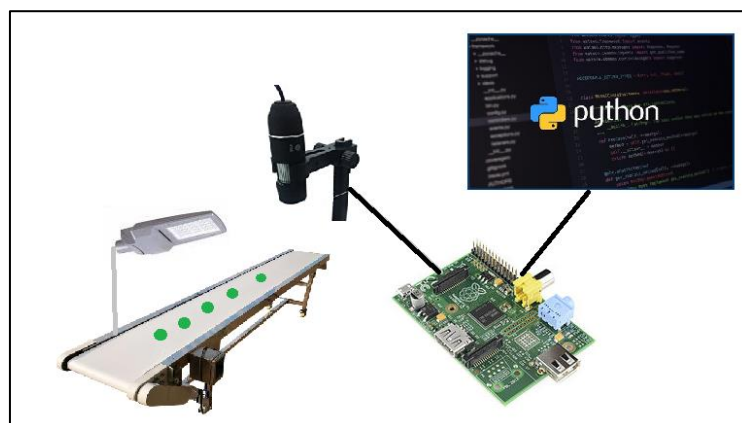
MÉTODOS Y PROCEDIMIENTOS

Para cumplir con los objetivos de la investigación, como lo es, la clasificación del limón, en base al color, tamaño y defecto, se seguirá la siguiente metodología:

- + Recolección de información de minicomputadora Raspberry Pi
- + Instalación y configuración de Python y OpenCV, tanto en computadora personal, laptop, como en Raspberry Pi.
- + Recolección de limón sutil para capturar imágenes.
- + Diseño de algoritmos de clasificación:
 - Captura de imágenes de limón
 - Análisis de histogramas
 - Transformación del espacio de color para determinar color y madurez
 - Medir ancho y alto de cada limón para determinar su tamaño
- + Pruebas y resultados de los algoritmos de clasificación en computadora personal
- + Pruebas y resultados de los algoritmos de clasificación en Raspberry Pi.

RESULTADOS Y DISCUSIÓN

El proceso de empaque del limón, se inicia desde la recepción de materia, hasta el embalado. En la figura se muestra el diagrama del sistema de clasificación del limón.



SISTEMA DE CLASIFICACIÓN Y DE VISIÓN ARTIFICIAL

La parte mecánica, como la faja transportadora, Los actuadores y el resto del proceso no son parte de este proyecto de tesis.

La parte de Visión artificial, objeto de esta tesis, se puede apreciar en la figura anterior, consiste de una cámara web (SENSOR), con la iluminación adecuada (CONDICIONES INICIALES), la tarjeta Raspberry Pi (CONTROLADOR) y la pantalla de visualización (HMI).

Para desarrollar esta parte del proceso, se ha utilizado el software de programación Python y las librerías de visión artificial, OpenCV.

En cuanto a la aplicación, se puede observar en la siguiente figura, en un diagrama de flujo del programa:

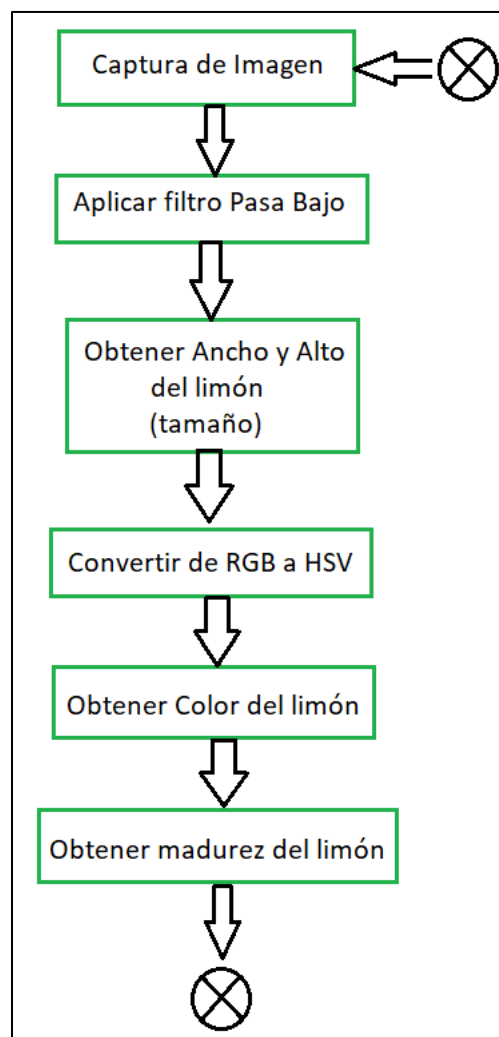


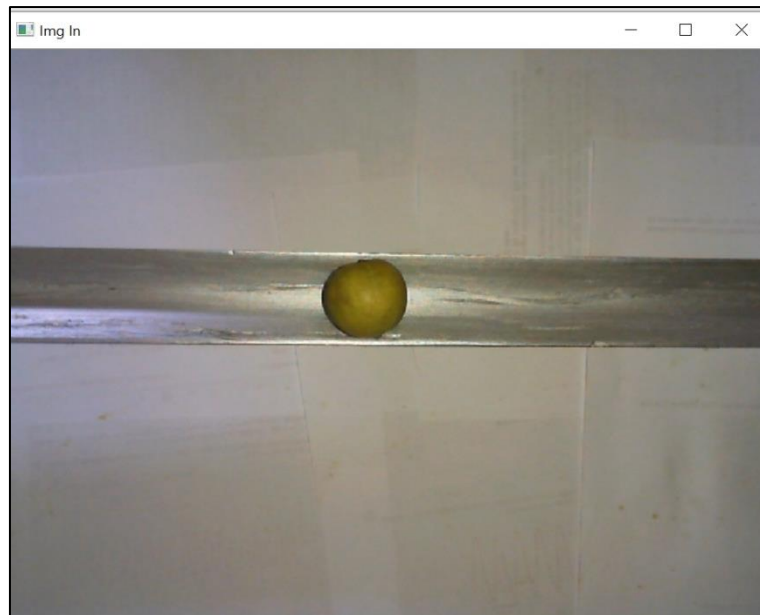
DIAGRAMA DE FLUJO DEL PROGRAMA

ACONDICIONAMIENTO DE LA ESCENA

Con la finalidad de que los algoritmos sean más simples y evitar mucho procesamiento, se acondiciona la escena con una buena iluminación. Se trata de que la luz sea difuminada de tal manera de que no se reflecte, produciendo brillo que afecte las etapas posteriores de procesamiento.

CAPTURA DE IMÁGENES

Las imágenes se capturan con una cámara web, de medianas prestaciones (SD), es decir con una resolución de 640 x 480 (480p = 0.3Megapíxeles). El formato con que se capturan los frames es en RGB a una velocidad de 30 cuadros por segundo. No se utilizan lentes de aumento, ya que la distancia de la cámara a la escena es de 30cm. Esta distancia, hay que tenerla en cuenta para hallar el tamaño de los limones.



CAPTURA DE IMAGEN CON ACONDICIONAMIENTO DE ILUMINACIÓN LED

APLICAR FILTRO PASABAJOS

Con la finalidad de mitigar el ruido, como consecuencia de la adquisición del video, se implementa un filtro de mediana de 7 x 7.

Es una operación o función que permite aplicar un filtro de desenfoque a una imagen. Este tipo de filtros es muy útil para quitar ruido de alta frecuencia (píxeles cambiando muy rápido) para una imagen.

- `imgF = cv2.blur(img, (9,9))`
Aplica un filtro de caja normalizado de 9x9. De este modo, este filtro toma el promedio de todos los píxeles bajo el área del kernel y reemplaza al elemento central por este promedio
- `imgF = cv2.GaussianBlur(img, (15, 15), 15)`
En este enfoque, en lugar de un filtro de caja que consta de coeficientes iguales, se utiliza un núcleo gaussiano. Este tipo de filtrado es muy eficaz para eliminar el ruido gaussiano de la imagen. El ruido gaussiano se encuentra asociado con la EMI y RFI, es imposible evitar este ruido, pero podemos mitigarlo.



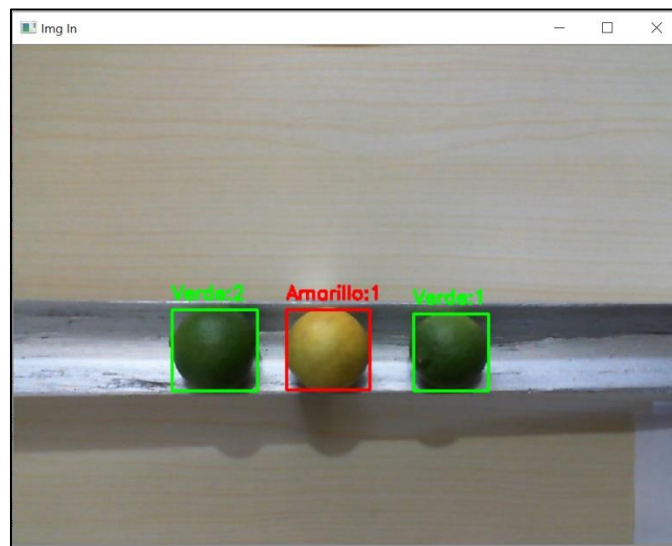
APLICACIÓN DEL FILTRO MEDIANA `[cv2.blur(img, (9,9))]`. IMAGEN ORIGINAL (IZQUIERDA) E IMAGEN FILTRADA (DERECHA)

CONVERTIR DE RGB A HSV

La adquisición de cada fotograma se recibe en el formato o en el modelo de color RGB. Pero necesitamos un modelo que describa mejor los colores y en una sola componente, y este es el modelo de color HSV (Tono, Saturación y Valor). Con este método, podemos describir mejor el color con una sola componente que es el Tono, ajustando su pureza y brillo.

OBTENER COLOR DE LIMÓN

Ya con las imágenes transformadas, es decir, pasadas al modelo de color HSV, podemos definir los rangos de valores para el color verde y amarillo.



DETECCIÓN DE COLOR VERDE Y AMARILLO DEL LIMÓN

Tanto el color verde como el amarillo tiene un rango, que se irán buscando de acuerdo a las imágenes de los limones.

En el modelo de color HSV, el rango de valores para el **limón verde** es:

MÍNIMO HSV = (30, 100, 0) to MÁXIMO HSV = (60, 250, 200)

El rango de valores para el **limón amarillo** es:

MÍNIMO HSV = (10, 150, 0) to MÁXIMO HSV = (30, 255, 255)

OBTENER MADUREZ O DEFECTO DEL LIMÓN

Para detectar la madurez o defecto del limón, utilizamos también el modelo color HSV, ya que el defecto es de color marrón rojizo.



DETECCIÓN DE MADUREZ (DEFECTO) DEL LIMÓN

- cv2.inRange():

Está una función que permite filtrar una ventana de valores para píxeles en una imagen. Por ejemplo, esta función se utilizaría en el procedimiento de crear un rastreador (sniffer) por color, para obtener como resultado una imagen binaria con el objeto detectado. Para el resultado de la imagen en blanco y negro, el blanco representa a todos los píxeles que fueron válidos para el rango filtrado.

OBTENER TAMAÑO DEL LIMÓN

Para obtener **el tamaño del limón**, vamos a hacer uso del ancho y la altura devuelta por la función **cv2.boundingRect**, que se basa en los momentos de Hu y que depende del área del objeto, en este caso del limón.

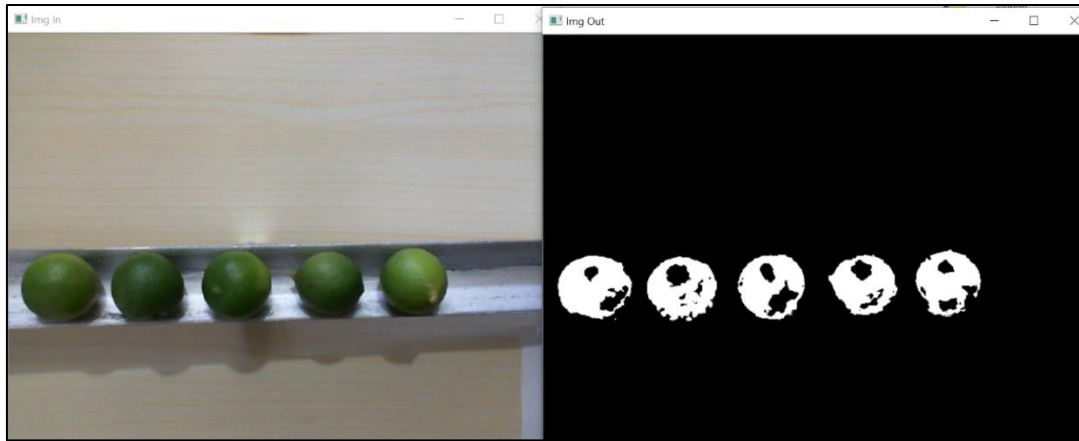
- cv2.boundingRect():

Es un rectángulo recto, no considera la rotación del objeto. Así que será el área del rectángulo delimitador.

Ej: Sea (x, y) la coordenada superior izquierda del rectángulo y (w, h) su anchura y altura.

```
x,y,w,h = cv2.boundingRect(f)
diam = (w + h) / 2
diametro = (diam * 35) / referD
```

Al momento de hallar o de detectar los colores **se segmenta el objeto** o sea se extrae el limón del fondo, que es lo que nos interesa. Las imágenes que se han obtenido al aplicar la función `cv2.inRange()` sobre la imagen HSV tiene el siguiente aspecto:



SEGMENTACIÓN DEL LIMÓN

Como puede observarse en la figura anterior, tenemos como resultado (imagen derecha) una imagen binaria, que con color blanco se observan los pixeles de los objetos (limones) y de color negro se observan los pixeles que pertenecen al fondo.

El siguiente paso es **encontrar los contornos de los objetos**. Para hacerlo llamamos la función

- **`cv2.findContours()`:**

Nos devolverá una lista con todos los contornos que encuentre en la imagen. Cada contorno individual es una matriz de la biblioteca Numpy de coordenadas (x, y) de los puntos fronterizos del objeto. Ahora, a partir de los contornos hallados, el siguiente paso es encontrar la ubicación y el ancho, así como la altura de cada objeto.

Entonces, manteniendo fija la distancia de 30cm, para hallar el diámetro de cada objeto, limón, obtenemos el promedio del ancho y la altura del limón. Este dato lo tenemos en función del número de pixeles, usando un valor de referencia para un limón de 35mm de diámetro. Ahora con una regla de 3 simple, podemos obtener el nuevo diámetro de cada limón que queremos averiguar.

Por lo tanto, tenemos las siguientes operaciones:

$$\text{Diámetro} = (\text{Diam} \times 35) / \text{Referencia}$$

Diámetro de referencia del limón en mm = 35 mm

Referencia = 78 pixeles (valor conocido para un limón de 35mm)

Diámetro: es la nueva medida del limón en mm

Diam, es el nuevo valor obtenido en pixeles del limón entrante.

Con esta fórmula, podemos clasificar el limón, dándole un cierto rango como, chico, mediano, grande etc.



VISUALIZACIÓN EN MM DEL DIAMETRO DEL LIMÓN

CONCLUSIONES

- Se ha desarrollado un sistema prototipo que puede ser una alternativa económica basada en visión artificial en tiempo real que utiliza únicamente una mini computadora, como la Raspberry Pi y una cámara web en un ambiente con iluminación controlada como cámara de inspección. Este dispositivo fue desarrollado con la intención de proponer, parte de todo un sistema de clasificación, utilizando visión artificial.
- Se diseñó un sistema de visión artificial para la clasificación del limón utilizando, primero una computadora personal y luego una Raspberry Pi.
- Se logró procesar el color, el tamaño y la madurez o defecto del limón utilizando visión artificial.
- En una computadora personal con prestaciones medias (i7, 8GB de RAM) se obtiene un alto grado de desempeño. La adquisición de la imagen y el procesamiento se hace en aproximadamente 50mseg, es decir casi a 20 cuadros por segundo. Tiempo suficiente para procesar los limones que pasen en una faja transportadora, donde estas pueden ser procesadas, utilizando este tipo de cámara. Es necesario hacer un análisis más riguroso si se quiere implementar para una aplicación industrial, teniendo en cuenta además que para una industria se requiere hardware industrial. Con la minicomputadora Raspberry Pi, la adquisición y el procesamiento se hace en 400mseg, es decir a casi 2 cuadros por segundo, tiempo insuficiente para aproximarnos a una implementación industrial.

- En la región Piura, específicamente Cieneguillo, provincia de Sullana, existe un sistema de Clasificación de limón utilizando visión artificial, pero por razones de políticas de privacidad de la empresa, no se dan a conocer detalles.
- El tipo de clasificación manual, es realizado por personal humano, el proceso de clasificación depende de la habilidad del personal contratado, la calidad de la clasificación y la producción depende de los trabajadores, lo que implica que el proceso se ve afectado por diferentes factores que afectan al ser humano, como cansancio y fatiga visual, la iluminación del ambiente, esto afecta a la calidad del producto, pudiendo provocar pérdidas económicas y la desconfianza en el cliente final.
- La gran ventaja de trabajar con Python, además de ser software libre, es que es multiplataforma, es decir, el mismo código sin alterarlo puede ejecutarse en diferentes sistemas operativos, como Linux, Windows, etc.
- El proyecto pretende proporcionar un entorno de desarrollo fácil de utilizar y altamente eficiente. Esto se ha logrado, realizando su programación en Python, pero se puede llevar el código al lenguaje de programación C++, para optimizar y aumentar la velocidad de procesamiento, ya que este lenguaje es compilado. OpenCV puede además utilizar el sistema de funciones primitivas de rendimiento integradas de Intel, un conjunto de rutinas de bajo nivel específicas para procesadores Intel.

RECOMENDACIONES

- Se recomienda clasificar el tamaño del limón utilizando componentes mecánicos, como rodillos separados ciertas distancias, con la finalidad de poder regular el calibre que se desea. De esta manera, se ahorraría tiempo de procesamiento para hallar el tamaño del limón.
- Se recomienda no utilizar la tarjeta Raspberry Pi, debido a la baja velocidad de procesamiento. La implementación debe ser una computadora con hardware de mucho mejores prestaciones.
- Se recomienda también, utilizar cámaras de más alta velocidad de captura, por ejemplo, de 120 frames por segundo.
- Se puede aumentar también la velocidad de procesamiento, utilizando el lenguaje de programación C++.
- Se recomienda implementar en un siguiente proyecto de tesis, continuar la implementación de la parte mecánica con sus respectivos actuadores.

ANEXO: CÓDIGO DEL PROGRAMA

```

import cv2
import numpy as np

referD = 70

min_am = np.array([10,150,0])
max_am = np.array([30,255,255])

min_ve = np.array([31,150,0])
max_ve = np.array([60,250,250])

min_df = np.array([0,150,100])
max_df = np.array([20,210,250])

cap = cv2.VideoCapture('vidLim00001.mp4')

while(cap.isOpened()):
    ret, img = cap.read()
    if(ret == True):
        imgF = cv2.blur(img, (9,9))
        #imgF = cv2.GaussianBlur(img, (15, 15), 15)
        imgHSV = cv2.cvtColor(imgF, cv2.COLOR_BGR2HSV)

        imgB = cv2.inRange(imgHSV, min_am, max_am)

        contornos,_ = cv2.findContours(imgB.copy(), cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
        for contour in contornos:
            area = cv2.contourArea(contour)
            if(area > 1500):
                #cont = cont + 1
                x,y,w,h = cv2.boundingRect(contour)
                diam = (w + h) / 2
                diametro = (diam * 35) / referD
                #print('Limon:' + str(cont), 'pix=' + str(diam), 'mm='
+ str(diametro))
                cv2.rectangle(img, (x,y), (x+w,y+h), (0,255,255), 2)
                cv2.putText(img, 'Amarillo:'+str(round(diametro)),
(x,y-10), cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0,255,255), 2)

        imgB = cv2.inRange(imgHSV, min_ve, max_ve)
        contornos,_ = cv2.findContours(imgB.copy(), cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
        for contour in contornos:
            area = cv2.contourArea(contour)
            if(area > 1500):
                #print(area)
                x,y,w,h = cv2.boundingRect(contour)

```

```

        diam = (w + h) / 2
        diametro = (diam * 35) / referD
        #print('Limon:' + str(cont), 'pix=' + str(diam), 'mm='
+ str(round(diametro)))
        cv2.rectangle(img, (x,y), (x+w,y+h), (0,255,0), 2)
        cv2.putText(img, 'Verde:'+str(round(diametro)), (x,y-
10), cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0,255,0), 2)

    imgB = cv2.inRange(imgHSV, min_df, max_df)

    contornos,_ = cv2.findContours(imgB.copy(), cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
    for contour in contornos:
        area = cv2.contourArea(contour)
        if(area > 500):
            x,y,w,h = cv2.boundingRect(contour)
            cv2.rectangle(img, (x,y), (x+w,y+h), (0,0,200), 2)
            cv2.putText(img, 'Defecto:', (x,y+h+20),
cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0,0,200), 2)

        cv2.imshow('Video',img)
        if(cv2.waitKey(30) == ord('q')):
            break
    else:

        break

cap.release()
cv2.destroyAllWindows()

```