



Tarea 1 - conversión de RGB/Conversión de Escala de Grises

Asignatura: Procesamiento de imágenes

Profesor: D. Sc. Gerardo García Gil

Alumno: Carlo Pinedo Suarez

Registro: 19310163 **Ciclo:** 2022-A

Ingeniería en Desarrollo de Software

Centro de Enseñanza Técnica Industrial (CETI)

INTRODUCCIÓN

Los espacios RGB, son modelos de color empleados en la reproducción del color. Un espacio de color indica la manera en que un color está definido, y su objetivo es facilitar la especificación de los colores de una forma normalizada. El espacio RGB (R-red, G-green, B-blue) se emplea en los monitores de una computadora, televisores, proyectores de vídeo y todos aquellos sistemas que utilizan combinaciones de materiales que fosforecen en el rojo, verde y azul. Todos los colores posibles que pueden ser creados por la mezcla de estas tres luces de color son aludidos como el espectro de color de estas tres luces. Los colores rojo, verde y azul fueron escogidos porque cada uno corresponde aproximadamente con uno de los tres tipos de conos sensitivos al color en el ojo humano. El modelo matemático de este espacio es el llamado cubo unitario de color, que es un cubo formado por los vectores unitarios en el espacio tridimensional.

La colorización es una técnica fotográfica que consiste en transferir colores a una imagen que se encuentre en tonos de gris (blanco y negro), sepia o monocromáticos.

La coloración de imágenes en tonos gris es una nueva área de investigación ya que es utilizada para incrementar el atractivo visual de imágenes antiguas en blanco y negro, películas o ilustraciones científicas.

Según algunos científicos el contenido y la percepción de la información de algunas imágenes puede mejorar mediante las variaciones de cromaticidad y luminancia del color; que mediante la variación de tonos de gris. Se sabe que una imagen en gris tiene pérdida de información de su color real puesto que el ancho de banda del color gris se compone de 256 tonos, mientras que el color verdadero tiene un ancho de banda de 256 x 256 x 256, lo cual nos hace imposible encontrar una manera directa de regresar del color gris al color original.

La conversión de una foto digital en color a blanco y negro va más allá de la simple desaturación de los colores, y se puede hacer para imitar cualquiera de una amplia gama de aspectos creados mediante el uso de filtros de color en la fotografía de película en blanco y negro. La conversión que no tiene en cuenta el color de una imagen y el tema de interés puede diluir el mensaje artístico y puede crear una imagen que parezca descolorida o sin gama tonal.

La conversión de una foto digital en color a blanco y negro utiliza los mismos principios que con los filtros de color en la fotografía de película, excepto que los filtros se aplican a cada uno de los tres canales de color RGB en una imagen digital (ver profundidad de bits). Ya sea que lo especifique o no, todas las técnicas de conversión tienen que usar alguna combinación ponderada de cada canal de color para producir un brillo en escala de grises. Algunas técnicas asumen una combinación para ti, aunque las más poderosas te dan control total. Cada uno hace sus propias compensaciones entre potencia y facilidad de uso, por lo que es posible que algunas técnicas se adapten mejor solo a ciertas tareas.

Para la mayoría de las editoriales o sociedades entomológicas publicar todas las imágenes en color, supondría un coste inasumible. En muchas normas de publicación se solicitan gráficos en blanco y negro, siendo habitual que el coste de las láminas en color repercuta en bolsillo del autor. Más excepcionalmente, en algunas sociedades las ilustraciones en color son gratuitas pero limitadas por el consejo de redacción en función a su calidad, necesidad y, por supuesto, capacidad presupuestaria.

En lo referente a los dibujos, es habitual que se recomiende el uso de tinta negra, plumilla o estilográfico y la realización de los gráficos con el sistema de manchas de puntos. El dibujo, por otra parte, presenta ciertas características que son muy apreciadas en comparación con la fotografía: la capacidad de perfilar, mostrar, deformar, destacar y componer la estructura de la imagen con libertad extrema. El proceso de la confección de un dibujo manual ha cambiado notablemente. Antaño se tomaba el espécimen, o la preparación microscópica, y se copiaba. Hoy en día, con la amplia difusión del uso de una cámara digital, se hacen una serie de fotografías y se mandan por internet al ilustrador. Bien es sabido que en la actualidad existen muy completos y complejos programas de edición de fotografías digitales (mapas de bits). Con ellos es posible plasmar, retocar y realizar casi cualquier idea, por muy descabellada que parezca, pero, a medida que se requiere mayor sofisticación, también se eleva ostensiblemente el conocimiento que se debe adquirir para la utilización de los mismos. Los aplicativos de diseño vectorial (imagen a partir de fórmulas matemáticas) también son muy sofisticados, pero el dibujo de puntos, incluso con una tableta digitalizadora y un lápiz óptico, todavía resulta más lento que la realización manual.

La binarización de fotografías es muy utilizada en la reproducción de imágenes, en la visión artificial y en la segmentación y reconocimiento entrópico de caracteres (OCR). El dibujo de láminas en blanco y negro con el sistema de puntos, además de gran habilidad, requiere altas dosis de paciencia y mucho tiempo. Es por ello que, en el intento de acortar el proceso, en este trabajo analizamos el uso de algoritmos de binarización y trámado. Los resultados, obviamente, no tienen la calidez y calidad de la exquisita manufactura de un sofisticado gráfico manual, pero, sin embargo, se acorta mucho el tiempo de proceso.

ANTECEDENTES

El color, según Isaac Newton, es una sensación que se produce en respuesta a una estimulación nerviosa del ojo, causada por una longitud de onda luminosa. El ojo humano interpreta colores diferentes dependiendo de las distancias longitudinales.

El color tiene un elemento subjetivo; nos produce muchas sensaciones, sentimientos, diferentes estados de ánimo, nos transmite mensajes, nos expresa valores, situaciones y sin embargo ... no existe más allá que nuestra percepción visual.

El color ha sido estudiado, por científicos, físicos, filósofos y artistas. Cada uno en su campo y en estrecho contacto con el fenómeno del color, llegaron a diversas conclusiones, muy coincidentes en algunos aspectos o bien que resultaron muy satisfactorias y como punto de partida para posteriores estudios.

El filósofo Aristóteles (384-322 AC) definió que todos los colores se conforman con la mezcla de cuatro colores y además otorgó un papel fundamental a la incidencia de luz y la sombra sobre los mismos. Estos colores, que denominó, como básicos eran los de tierra, el fuego, el agua y el cielo.

Siglos más tarde, Leonardo Da Vinci (1452-1519) definió al color como propio de la materia. Adelantó mas definiendo la siguiente escala de colores básicos: el blanco como principal ya que permite recibir a todos los demás colores, el amarillo para la tierra, el verde para el agua, el azul para el cielo, el rojo para el fuego y el negro para la oscuridad.

Con la mezcla de esta escala podía formar a los demás, aunque también observó que el verde surgía de alguna mezcla. El sistema de color aditivo en el que se basa el modelo de color RGB (Red Green Blue) y que da color a los píxeles de nuestros monitores fue creado por el físico escocés James Clerk Maxwell.

Una escala de grises o valor dentro del contexto de las artes gráficas y la educación artística es el sistema ordenado y gradual que cubre un rango limitado de valores de luminosidad entre el blanco, el gris y el negro. El número de valores que abarcan las escalas de grises es variable dependiendo del uso que se le vaya a dar, aunque suele ser una cifra baja y sencilla de manejar, entre 3 y 10 valores, por cuestiones prácticas.

La escala de grises más simple es la de tres valores: blanco, gris y negro; seguida por la de cinco valores: blanco, gris claro, gris medio, gris oscuro y negro. Añadiendo a la escala de cinco valores un nuevo paso intermedio entre cada valor se obtiene la popular escala de 9 valores propuesta por Denman Ross en 1907.

DESARROLLO

El uso del color en el procesamiento de imágenes está motivado por dos factores principales; El primer color es un poderoso descriptor que a menudo simplifica la identificación y extracción de objetos de una escena. En segundo lugar, el ser humano puede discernir miles de tonos de color y intensidades, en comparación con unas dos docenas de tonos de gris. En el Modelo RGB, cada color aparece en su espectro primario componentes de rojo, verde y azul. Este modelo se basa en Sistema de coordenadas Cartesianas. Imágenes representadas en color RGB El modelo consta de tres imágenes componentes. Uno para cada primaria, cuando se introducen en un monitor RGB, estas tres imágenes se combinan en la pantalla de fósforo para producir una imagen de color compuesta. los números de bits utilizados para representar cada píxel en el espacio RGB es llama la profundidad de píxel. Considere una imagen RGB en la que cada uno de las imágenes roja, verde y azul es una imagen de 8 bits. Bajo estas condiciones se dice que cada píxel de color RGB tiene una profundidad de 24 un poco.

En los sistemas de visión artificial es recurrente la conversión de la imagen a color en su correspondiente escala de grises para su posterior binarización. Este proceso, que forma parte del procesamiento de la imagen en la búsqueda de mejorar sus propiedades y facilitar la extracción de características, se ejecuta casi siempre recorriendo a un único algoritmo desde el espacio de color RGB. Este artículo presenta algunas de las alternativas en la conversión a escala de grises de imágenes a color RGB.

Imagen en escala de grises

También se conoce como imagen de intensidad, escala de grises o nivel de gris. Matriz de clase uint8, uint16, int16, simple o doble cuyo píxel valores especifican valores de intensidad. Para matrices simples o dobles, los valores oscilan entre [0, 1]. Para uint8, los valores oscilan entre [0,255]. Para uint16, los valores van desde [0, 65535]. Para int16, valores rango de [-32768, 32767].

Niveles de gris

Los niveles de gris representan el número de intervalo de cuantificación en gris Procesamiento de imágenes a escala. En la actualidad, los más utilizados El método de almacenamiento es almacenamiento de 8 bits. Hay 256 niveles de gris en un 8 imagen en escala de grises de bits, y la intensidad de cada píxel puede tener de 0 a 255, siendo 0 negro y 255 blanco nosotros.

Otro método de almacenamiento comúnmente utilizado es el almacenamiento de 1 bit. Ahí son dos niveles de gris, siendo 0 negro y 1 blanco imagen binaria, que, se utiliza con frecuencia en imágenes médicas, es siendo referido como imagen binaria. Como las imágenes binarias son fáciles para operar, otras imágenes de formato de almacenamiento a menudo se convierten en imágenes binarias cuando se utilizan para mejorar o mejorar detección.

Conversión a escala de grises

Para convertir un color de un espacio de color basado en un modelo de color RGB típico comprimido por gamma (no lineal) a una representación en escala de grises de su luminancia, la función de compresión gamma debe eliminarse primero a través de la expansión gamma (linealización) para transformar la imagen en un espacio de color RGB lineal, de modo que se pueda aplicar la suma ponderada adecuada a los componentes de color lineal Rlinear, Glinear,Blinear para calcular la luminancia lineal Ylineal, que luego se puede comprimir gamma de nuevo si el resultado de la escala de grises también se va a codificar y almacenar en un espacio de color no lineal típico.

$$C_{\text{linear}} = \begin{cases} \frac{C_{\text{srgb}}}{12.92}, & \text{if } C_{\text{srgb}} \leq 0.04045 \\ \left(\frac{C_{\text{srgb}} + 0.055}{1.055} \right)^{2.4}, & \text{otherwise} \end{cases}$$

donde Csrgb representa cualquiera de los tres primarios sRGB con compresión gamma (Rsrgb , Gsrgb y Bsrgb , cada uno en el rango [0,1]) y Clinear es el valor de intensidad lineal correspondiente (Rlinear , Glineal y Blineal , también en el rango [0,1]).

Para imágenes en espacios de color como Y'UV y sus afines, que se utilizan en TV y video en color estándar sistemas como PAL, SECAM y NTSC, un componente de luminancia no lineal (Y') se calcula directamente a partir de intensidades primarias comprimidas con gamma como una suma ponderada, que, aunque no es una representación perfecta de la luminancia colorimétrica, se puede calcular más rápidamente sin la expansión y compresión gamma utilizado en cálculos fotométricos/colorimétricos. En los modelos Y'UV y Y'IQ utilizados por PAL y NTSC, el El componente rec601 luma (Y') se calcula como:

$$Y' = 0.299R' + 0.587G' + 0.114B'$$

Fragmento de código usado para la conversión a escala de grises de una imagen RGB

```
%Impresion de imagen
A = imread("pikachu.jpg");
[M,N] = size(A);
figure(1);
imshow(A);

%Escala de grises

imagen = imread("pikachu.jpg");
i = 1 : M;
j = 1 : N;

x = (imagen(i,j)*0.299+imagen(i,j)*0.587+imagen(i,j)*0.114);
imagen(i,j,1) = x;
imagen(i,j,2) = x;
imagen(i,j,3) = x;

figure(2);
imshow(imagen);

A = imread("pikachu.jpg");
y=rgb2gray(A);

figure(3);
imshow(y);
```

Binarización

La mayor parte de los algoritmos para reconocer escritura están escritos a partir de imágenes binarias, por lo que se hace conveniente el paso de una imagen en niveles de gris (o color) a una binaria, además esto permite reducir el volumen de los datos a tratar.

La binarización de una imagen digital consiste en convertir la imagen digital en una imagen en blanco y negro, de tal manera que se preserven las propiedades esenciales de la imagen.

Uno de los métodos para poder binarizar una imagen digital es mediante el histograma de dicha imagen. A través del histograma obtenemos una gráfica donde se muestran el número de píxeles por cada nivel de gris que aparecen en la imagen. Para binarizar la imagen, se deberá elegir un valor adecuado dentro de los niveles de grises (umbral), de tal forma que el histograma forme un valle en ese nivel. Todos los niveles de grises menores al umbral calculado se convertirán en negro y todos los mayores en blanco.

Fragmento de código usado para la binarización de una imagen previamente convertida a escala de grises

```
%Binarizacion
A = imread("pikachu.jpg");
for i = 1:M
    for j = 1:N
        if  imagen(i,j) < 128
            imagen(i,j,1) = 0;
            imagen(i,j,2) = 0;
            imagen(i,j,3) = 0;

        elseif imagen(i,j) > 128
            imagen(i,j,1) = 255;
            imagen(i,j,2) = 255;
            imagen(i,j,3) = 255;

        end
    end
end
figure(4);
imshow(imagen);

%Experimento

im=255-experimento;

figure(5);
imshow(im);
```

RESULTADOS

Imagen original



Conversión a escala de grises



Binarización



CONCLUSIONES

La caracterización de los dispositivos de adquisición (cámaras) ha facilitado la determinación de las mejores condiciones de uso en aplicaciones colorimétricas. Se han desarrollado herramientas de análisis que permiten obtener, de forma automática o semiautomática, medidas sobre pequeñas diferencias de color, realce de contornos y segmentación, relacionándolas con la respuesta que en operaciones similares proporciona la visión humana. Se presentan resultados experimentales y numéricos en un conjunto de aplicaciones que abarcan operaciones seleccionadas en diversos campos. En el campo de la inspección industrial, se trata la evaluación de la uniformidad en el color de las muestras textiles. En el tratamiento de imágenes, se propone un nuevo método de realce de imágenes en color.

En la interpretación de imágenes oftálmicas, se realiza la extracción de características en imágenes de dos tipos: imágenes de complicaciones derivadas del uso de lentes de contacto e imágenes del fondo del ojo sútiles para el diagnóstico y seguimiento del glaucoma. Esta tesis contribuye a aumentar las capacidades potenciales de los sistemas de visión artificial para ser utilizados en aplicaciones que requieren una evaluación e interpretación de la información del color, aplicaciones que tradicionalmente se llevan a cabo mediante la visión humana entrenada de técnicos o especialistas y que están todavía lejos de la automatización y la objetivización.

BIBLIOGRAFIA

- [1] Wilhelm Burger , Mark J. Burge, Principles of Digital Image Processing, Springer-Verlag London Limited, 2009
- [2] What is Colour? [En línea] disponible en: http://www.poynton.com/notes/colour_and_gamma/ColorFAQ.html#RTFToC7 (Consultado el 22 de agosto de 2022)
- [3] Guild, John, "The colorimetric properties of the spectrum" [En línea] disponible en: <http://www.jstor.org/pss/91229> (Consultado el 6 de Marzo de 2011) HSL and HSV, [En línea] disponible en: http://en.wikipedia.org/wiki/HSL_and_HSV , (Consultado el 22 de agosto de 2022)
- [4] Johnson, Stephen (2006). Stephen Johnson on Digital Photography (<https://books.google.com/books?id=0UVRXzF91gcC&q=grayscale+black-and-white-continuous-tone&pg=PA17>). O'Reilly. ISBN 0-596-52370-X.

Anexos

Código implementado

```
%Impresion de imagen
A = imread("pikachu.jpg");
[M,N] = size(A);
figure(1);
imshow(A);

%Escala de grises

imagen = imread("pikachu.jpg");
i = 1 : M;
j = 1 : N;

x = (imagen(i,j)*0.299+imagen(i,j)*0.587+imagen(i,j)*0.114);
imagen(i,j,1) = x;
imagen(i,j,2) = x;
imagen(i,j,3) = x;

figure(2);
imshow(imagen);

A = imread("pikachu.jpg");
y=rgb2gray(A);

figure(3);
imshow(y);
```

```
%Binarizacion
A = imread("pikachu.jpg");
for i = 1:M
    for j = 1:N
        if imagen(i,j) < 128
            imagen(i,j,1) = 0;
            imagen(i,j,2) = 0;
            imagen(i,j,3) = 0;

        elseif imagen(i,j) > 128
            imagen(i,j,1) = 255;
            imagen(i,j,2) = 255;
            imagen(i,j,3) = 255;

        end
    end
end
figure(4);
imshow(imagen);

%Experimento - negativo de una imagen

im=255-experimento;

figure(5);
imshow(im);
```

Asignatura: Procesamiento de imágenes

Profesor: D. Sc. Gerardo García Gil

Alumno: Carlo Pinedo Suarez

Registro: 19310163 **Ciclo:** 2022-B

Ingeniería en Desarrollo de Software

Centro de Enseñanza Técnica Industrial (CETI)

INTRODUCCIÓN

El procesamiento digital de imágenes ha adquirido, en años recientes, un papel importante en las tecnologías de la información y el cómputo. Actualmente, es la base de una creciente variedad de aplicaciones que incluyen diagnóstico médica, percepción remota, exploración espacial, visión por computadora, etc. Como resultado directo de la reducción en el precio de las computadoras, el procesamiento digital de imágenes actualmente se puede efectuar (aunque con ciertas limitantes) en una computadora personal.

La proliferación de nuevos equipamientos con capacidad para realizar millones de operaciones por segundo y su extensión a la vida cotidiana y a todo tipo de usuario, ha hecho posible que el análisis y procesamiento de imágenes digitales se constituya en un gran campo de estudio. En la actualidad, esta tecnología se encuentra incorporada incluso en todo tipo de equipamiento doméstico, como cámaras digitales, escaners y teléfonos celulares, entre otros.

En términos históricos, la utilización de imágenes radiográficas para diagnóstico clínico data prácticamente desde el descubrimiento de los rayos X en 1895. Incluso, las imágenes funcionales a partir de la emisión de fotones (rayos γ) por parte de radionucleidos ya cuenta con más de 90 años de antigüedad. Sin embargo, las imágenes eran adquiridas sobre films radiográficos o directamente in vivo, por lo que su correcto procesamiento no ha explotado su real potencialidad sino hasta la incorporación de la tecnología que permitió digitalizarlas.

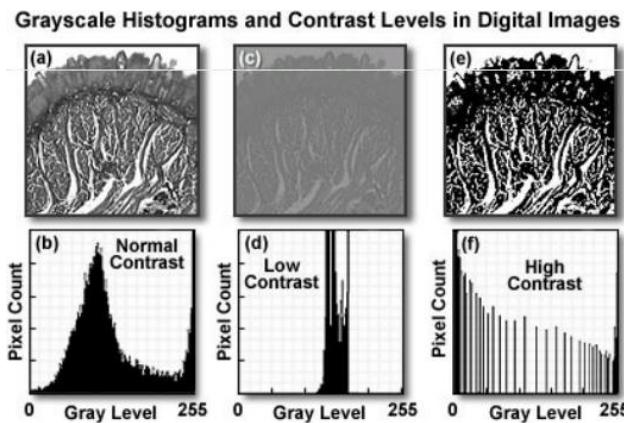
El motivo principal de esta “aparición tardía” del procesamiento de imágenes ha sido entonces, debido a los requerimientos de hardware tanto para el procesamiento de las mismas como para la representación de estas en sistemas gráficos de alta performance. Paralelamente a este desarrollo, la formulación de algoritmos para el procesamiento ha seguido los avances tecnológicos logrando un alto grado de sofisticación y manipulación de imágenes en tiempo casi real.

Una herramienta actual para el procesamiento de imágenes es el uso de histogramas. El histograma es una representación estadística que nos indica la proporción de píxeles de cada tonalidad que hay en una imagen. Si es negro o blanco significa que representa los niveles de grises o luminosidad. Recordemos que dos colores diferentes pueden tener una misma luminosidad o, dicho de otro modo, que si los desaturamos ambos corresponderán a un mismo gris. También podemos ver los histogramas correspondientes a cada canal R (rojo), G (verde), B (azul), y entonces se nos representará con el color del canal correspondiente. Para simplificar la cuestión, a partir de ahora sólo veremos histogramas de representación de la luminosidad.

DESARROLLO

Histograma simple

Supongamos dada una imagen en niveles de grises, siendo el rango de 256 tonos de gris (de 0 a 255). El histograma de la imagen consiste en una gráfica donde se muestra el número de píxeles, n_k , de cada nivel de gris, r_k , que aparecen en la imagen. En el siguiente ejemplo podemos ver tres imágenes con sus correspondientes histogramas.



El análisis estadístico derivado del histograma puede servir para comparar contrastes e intensidades entre imágenes. El histograma podría ser alterado para producir cambios en la imagen. Por ejemplo, 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 Histogramas imagen. La forma usual de binarizar una imagen es eligiendo un valor adecuado o umbral, u , dentro de los niveles de grises, tal que el histograma forme un "valle" en ese nivel. Todos los niveles de grises menores que u se convierten en 0 (negro), y los mayores que u se convierten en 255 (blanco).

Para el caso de esta práctica en el caso del histograma simple donde la imagen original fue transformada a blanco y negro para después proceder a mostrar su dicho histograma, básicamente lo que se hizo fue leer la imagen original para obtener una referencia de esta, después se mando a llamar la función que convierte a escala de grises y luego en otra variable se guardaron los valores de cada pixel formando así el histograma.

El código implementado fue el siguiente:

```
%Impresion de imagen
A = imread("pikachu.jpg");
subplot (3,1,1);
imshow(A);

%Histograma simple
in = funcion_escala_grises("pikachu.jpg");
subplot (3,1,2);
imshow(in);
[fil,col]=size(in);
pixmax=256;
tam=zeros(pixmax);
for x=1:fil
    for y=1:col
        xy=in(x,y);
        for val=1:pixmax
            if xy==val
                tam(val+1)=tam(val+1)+1;
            end
        end
    end
end
subplot (3,1,3);
plot(tam);
```

Histograma acumulado

Un histograma acumulado es un gráfico de líneas que se usa para presentar las frecuencias absolutas de los valores de una distribución en el cual la altura del punto asociado a un valor de las variables es proporcional a la frecuencia de dicho valor.

El histograma acumulativo es una variante del histograma normal, refleja información importante para las operaciones de pixel, por ejemplo, para equilibrar un histograma.

La función del histograma acumulativo se define como una función monótona creciente en donde todos los valores a partir del segundo valor son recalculados con la suma de su valor inmediato anterior. El histograma acumulativo esta definido por la siguiente ecuación:

$$H(i) = \sum_{j=0}^i H(j) \text{ para } 0 \leq i < K$$

El código implementado es el siguiente:

```
%Histograma Acumulativo
%vi=[1:256];
vo=0;
for ru=1:256
    H(ru)=vo+tam(ru);
    vo=H(ru);
end
figure(2);
subplot (3,1,1);
imshow(A);
subplot (3,1,2);
imshow(in);
subplot (3,1,3);
plot(H);
```

Histograma lineal o ecualizado

La ecualización de histograma es una técnica para ajustar los niveles de contraste y ampliar el rango de intensidad en una imagen digital. Por lo tanto, mejora la imagen, lo que facilita la extracción de información y el procesamiento posterior de la imagen.

Reparte de forma más o menos uniforme los valores del histograma.

Dada una imagen $M \times N$, con n píxeles para cada nivel r , la ecualización del histograma consiste en realizar la siguiente transformación sobre los niveles de intensidad de la imagen:

$$S_k = T(rk) = (L - 1) \sum_{j=0}^k pr(rj) = \frac{L - 1}{MN} \sum_{j=0}^k nj$$

lo que resulta en una dispersión del histograma en un rango mayor dentro del intervalo $[0, L-1]$. La principal ventaja de este método es que es completamente “automático”.

El código implementado es el siguiente:

```
%Histograma Ecualizado
GIm=funcion_escala_grises("pikachu.jpg");
numofpixels=size(GIm,1)*size(GIm,2);

subplot (3,1,1);
imshow(GIm);

HIm=uint8(zeros(size(GIm,1),size(GIm,2)));
freq=zeros(256,1);
probf=zeros(256,1);
probcl=zeros(256,1);
cum=zeros(256,1);
output=zeros(256,1);

for i=1:size(GIm,1)
    for j=1:size(GIm,2)
        value=GIm(i,j);
        freq(value+1)=freq(value+1)+1;
        probf(value+1)=freq(value+1)/numofpixels;
    end
end

sum=0;
no_bins=255;

for i=1:size(probf)
    sum=sum+freq(i);
    cum(i)=sum;
    probcl(i)=cum(i)/numofpixels;
    output(i)=round(probc(i)*no_bins);
end

for i=1:size(GIm,1)
    for j=1:size(GIm,2)
        HIm(i,j)=output(GIm(i,j)+1);
    end
end

subplot (3,1,2);
imshow(HIm);%Imagen ecualizada
```

```

[fil,col]=size(HIm);
pixmax=256;
tam=zeros(pixmax);
for x=1:fil
    for y=1:col
        xy=HIm(x,y);
        for val=1:pixmax
            if xy==val
                tam(val+1)=tam(val+1)+1;
            end
        end
    end
end
vo=0;
for ru=1:256
    H(ru)=vo+tam(ru);
    vo=H(ru);
end
subplot (3,1,3);

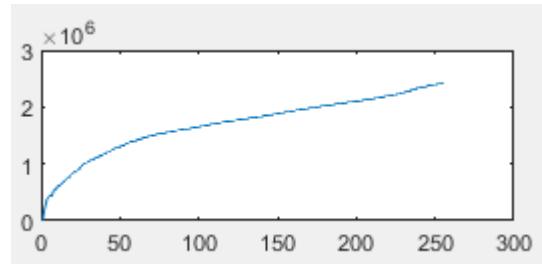
```

RESULTADOS

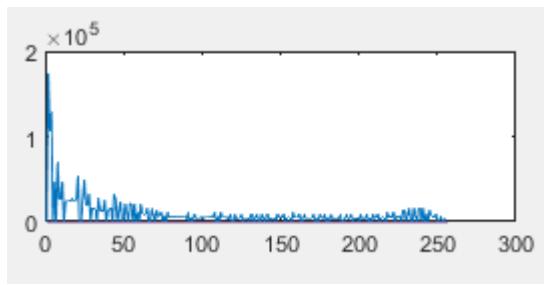
Histograma acumulado

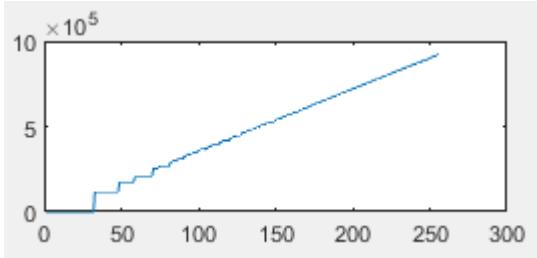


Histograma Simple



Histograma ecualizado





imágenes ya que nos permite observar bastantes características que nos dan información relevante.

CONCLUSIONES

El procesamiento digital de imágenes y la visión por computadora son tópicos que se utilizan cada vez más en diversas áreas, tales como publicidad, cinematografía, meteorología, medicina, etc.

El aprovechar el alto desempeño del MATLAB para procesar matrices y vectores, así como el alto desempeño en graficación permite implementar sistemas de procesamiento digital de imágenes o bien visión por computadora.

Hoy día se estudia cómo mejorar la complejidad en la codificación de los vectores de movimiento y esquemas de cuantización multicapa para una óptima ordenación de los bits.

La optimización de imágenes para diseños de pantalla se deberá tener en cuenta la calidad de la imagen en relación con el menor peso posible.

El contraste tonal hace referencia al brillo de los elementos de una imagen. Si tu foto está compuesta por áreas extremadamente claras y oscuras, se puede considerar de alto contraste. Si presenta una amplia gama de tonos que van desde el blanco al negro, ambos puros, se trata de una imagen de contraste medio.

Finalmente podemos decir que los histogramas son herramientas muy útiles al momento de trabajar con

BIBLIOGRAFIA

1. Es.acervolima.com. 2022. Ecualización de histograma en C | Procesamiento de imágenes – Acervo Lima. [online] Available at: <<https://es.acervolima.com/ecualizacion-de-histograma-en-c-procesamiento-de-imagenes/>> [Accessed 25 August 2022].
2. 2022. [online] Available at: <<https://www.cognex.com/es-mx/products/machine-vision/vision-software/vision-tools/histogram-image-processing>> [Accessed 25 August 2022].
3. Pro.arcgis.com. 2022. Histograma de imágenes—ArcGIS Pro | Documentación. [online] Available at: <<https://pro.arcgis.com/es/pro-app/latest/help/analysis/geoprocessing/charts/image-histogram.htm>> [Accessed 25 August 2022].
4. Choque Huanca, H., 2022. Ecualización del histograma para el procesamiento digital de imágenes estáticas. [online] Repositorio.umsa.bo. Available at: <<https://repositorio.umsa.bo/handle/123456789/17511>> [Accessed 25 August 2022].

Anexos

Código implementado

```
%Impresion de imagen
A = imread("pikachu.jpg");
subplot (3,1,1);
imshow(A);

%Histograma simple
in = funcion_escala_grises("pikachu.jpg");
subplot (3,1,2);
imshow(in);
[fil,col]=size(in);
pixmax=256;
tam=zeros(pixmax);
for x=1:fil
    for y=1:col
        xy=in(x,y);
        for val=1:pixmax
            if xy==val
                tam(val+1)=tam(val+1)+1;
            end
        end
    end
end
subplot (3,1,3);
plot(tam);

%Histograma Acumulativo
%vi=[1:256];
vo=0;
for ru=1:256
    H(ru)=vo+tam(ru);
    vo=H(ru);
end
figure(2);
subplot (3,1,1);
imshow(A);
subplot (3,1,2);
imshow(in);
subplot (3,1,3);
plot(H);

%Histograma Ecualizado
GIm=funcion_escala_grises("pikachu.jpg");
numofpixels=size(GIm,1)*size(GIm,2);

subplot (3,1,1);
imshow(GIm);

HIm=uint8(zeros(size(GIm,1),size(GIm,2)));
freq=zeros(256,1);
probf=zeros(256,1);
probcc=zeros(256,1);
cum=zeros(256,1);
output=zeros(256,1);

%freq cuenta la aparición de cada valor de píxel.
%La probabilidad de cada ocurrencia se calcula por probf.

for i=1:size(GIm,1)
    for j=1:size(GIm,2)
        value=GIm(i,j);
        freq(value+1)=freq(value+1)+1;
        probf(value+1)=freq(value+1)/numofpixels;
    end
end

sum=0;
no_bins=255;

%La probabilidad de distribución acumulada se calcula

for i=1:size(probf)
    sum=sum+freq(i);
    cum(i)=sum;
    probcc(i)=cum(i)/numofpixels;
    output(i)=round(probcc(i)*no_bins);
end

for i=1:size(GIm,1)
    for j=1:size(GIm,2)
        HIm(i,j)=output(GIm(i,j)+1);
    end
end

subplot (3,1,2);
imshow(HIm);%Imagen ecualizada
```

```
[fil,col]=size(HIm);
pixmax=256;
tam=zeros(pixmax);
for x=1:fil
    for y=1:col
        xy=HIm(x,y);
        for val=1:pixmax
            if xy==val
                tam(val+1)=tam(val+1)+1;
            end
        end
    end
end
vo=0;
for ru=1:256
    H(ru)=vo+tam(ru);
    vo=H(ru);
end
subplot (3,1,3);
plot(H);%Muestra el histograma acumulado de la imagen ecualizada
```

Asignatura: Procesamiento de imágenes

Profesor: D. Sc. Gerardo García Gil

Alumno: Carlo Pinedo Suarez

Registro: 19310163 **Ciclo:** 2022-B

Ingeniería en Desarrollo de Software

Centro de Enseñanza Técnica Industrial (CETI)

INTRODUCCIÓN

El procesamiento digital de imágenes ha adquirido, en años recientes, un papel importante en las tecnologías de la información y el cómputo. Actualmente, es la base de una creciente variedad de aplicaciones que incluyen diagnóstico médica, percepción remota, exploración espacial, visión por computadora, etc. Como resultado directo de la reducción en el precio de las computadoras, el procesamiento digital de imágenes actualmente se puede efectuar (aunque con ciertas limitantes) en una computadora personal.

La proliferación de nuevos equipamientos con capacidad para realizar millones de operaciones por segundo y su extensión a la vida cotidiana y a todo tipo de usuario, ha hecho posible que el análisis y procesamiento de imágenes digitales se constituya en un gran campo de estudio. En la actualidad, esta tecnología se encuentra incorporada incluso en todo tipo de equipamiento doméstico, como cámaras digitales, escaners y teléfonos celulares, entre otros.

En términos históricos, la utilización de imágenes radiográficas para diagnóstico clínico data prácticamente desde el descubrimiento de los rayos X en 1895. Incluso, las imágenes funcionales a partir de la emisión de fotones (rayos γ) por parte de radionucleidos ya cuenta con más de 90 años de antigüedad. Sin embargo, las imágenes eran adquiridas sobre films radiográficos o directamente in vivo, por lo que su correcto procesamiento no ha explotado su real potencialidad sino hasta la incorporación de la tecnología que permitió digitalizarlas.

El motivo principal de esta “aparición tardía” del procesamiento de imágenes ha sido entonces, debido a los requerimientos de hardware tanto para el procesamiento de las mismas como para la representación de estas en sistemas gráficos de alta performance. Paralelamente a este desarrollo, la formulación de algoritmos para el procesamiento ha seguido los avances tecnológicos logrando un alto grado de sofisticación y manipulación de imágenes en tiempo casi real.

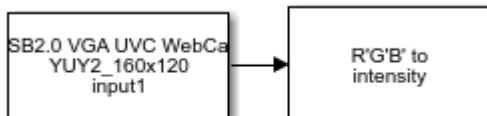
Una herramienta actual para el procesamiento de imágenes es el uso de histogramas. El histograma es una representación estadística que nos indica la proporción de píxeles de cada tonalidad que hay en una imagen. Si es negro o blanco significa que representa los niveles de grises o luminosidad. Recordemos que dos colores diferentes pueden tener una misma luminosidad o, dicho de otro modo, que si los desaturamos ambos corresponderán a un mismo gris. También podemos ver los histogramas correspondientes a cada canal R (rojo), G (verde), B (azul), y entonces se nos representará con el color del canal correspondiente. Para simplificar la cuestión, a partir de ahora sólo veremos histogramas de representación de la luminosidad.

DESARROLLO

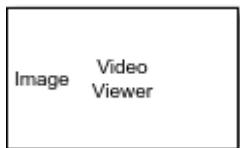
Simulink

Simulink es un entorno de diagramas de bloque que se utiliza para diseñar sistemas con modelos multidominio, simular antes de implementar en hardware y desplegar sin necesidad de escribir código.

Para esta práctica será necesario hacer el uso de la cámara para grabar video ya que se pretende realizar el histograma simple y lineal de este, para esto comenzaremos agregando el bloque que realizara tal tarea y luego convertiremos la imagen rgb a un formato donde podamos medir la intensidad luminosa que es la escala de grises



Para ver el video obtenido se usa lo siguiente:

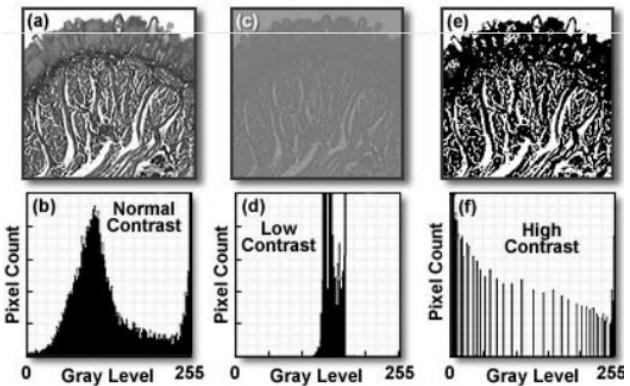


Mas adelante mostraremos el diagrama completo a bloques utilizado, pero por lo pronto hasta aquí será suficiente para la explicación del desarrollo.

Histograma simple

Supongamos dada una imagen en niveles de grises, siendo el rango de 256 tonos de gris (de 0 a 255). El histograma de la imagen consiste en una gráfica donde se muestra el número de píxeles, n_k , de cada nivel de gris, r_k , que aparecen en la imagen. En el siguiente ejemplo podemos ver tres imágenes con sus correspondientes histogramas.

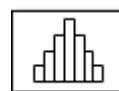
Grayscale Histograms and Contrast Levels in Digital Images



El análisis estadístico derivado del histograma puede servir para comparar contrastes e intensidades entre imágenes. El histograma podría ser alterado para producir cambios en la imagen. Por ejemplo, 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 Histogramas imagen. La forma usual de binarizar una imagen es eligiendo un valor adecuado o umbral, u , dentro de los niveles de grises, tal que el histograma forme un "valle" en ese nivel. Todos los niveles de grises menores que u se convierten en 0 (negro), y los mayores que u se convierten en 255 (blanco).

Para el caso de esta práctica en el caso del histograma simple donde el video original fue transformado a blanco y negro para después proceder a mostrar su dicho histograma, básicamente lo que se hizo fue leer el video original para obtener una referencia de esta, después se mandó a llamar la función que convierte a escala de grises y luego en otra variable se guardaron los valores de cada pixel formando así el histograma.

El bloque implementado en simulink implementado fue el siguiente:



Para mostrar el histograma en este caso yo solo use una función que me permitirá ver el histograma desde un plot, hay que tomar en cuenta que hay que convertir el formato double, mas adelante se vera claramente en el diagrama a bloques, el siguiente es el código usado:

```
function histSimple=Histograma_Simple(imagenGris)

%Histograma simple
histSimple=imagenGris*200;

subplot(2,2,1);
plot(histSimple);
title("Histograma Simple");
end
```

```
function histAcum=Histograma_Acum(imagenGris)

%Histograma simple
histSimple=imagenGris*200;

histAcum(1)=histSimple(1);
for i=2:256
    histAcum(i)=histSimple(i)+histAcum(i-1);
end
subplot(2,2,2);
plot(histAcum);
title("Histograma Acumulado");

end
```

RESULTADOS

Histograma acumulado

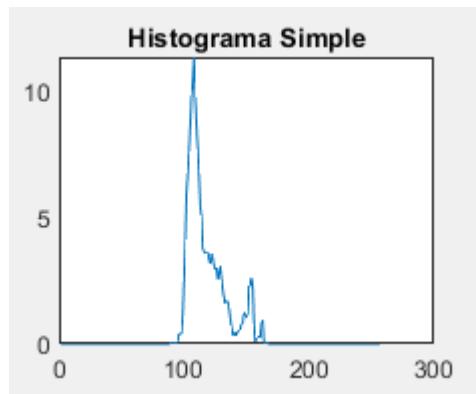
Un histograma acumulado es un gráfico de líneas que se usa para presentar las frecuencias absolutas de los valores de una distribución en el cual la altura del punto asociado a un valor de las variables es proporcional a la frecuencia de dicho valor.

El histograma acumulativo es una variante del histograma normal, refleja información importante para las operaciones de pixel, por ejemplo, para equilibrar un histograma.

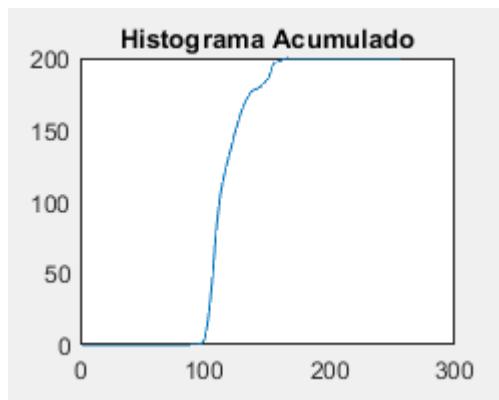
La función del histograma acumulativo se define como una función monótona creciente en donde todos los valores a partir del segundo valor son recalculados con la suma de su valor inmediato anterior. El histograma acumulativo esta definido por la siguiente ecuación:

$H(i) = \sum_{j=0}^i H(j)$ para $0 \leq i < K$ Una vez que se convirtió el video a double después de obtener el histograma simple agregamos en una función el código necesario para realizar dicho histograma:

Histograma Simple



Histograma acumulado



CONCLUSIONES

El procesamiento digital de imágenes y la visión por computadora son tópicos que se utilizan cada vez más en diversas áreas, tales como publicidad, cinematografía, meteorología, medicina, etc.

El aprovechar el alto desempeño del MATLAB para procesar matrices y vectores, así como el alto desempeño en graficación permite implementar sistemas de procesamiento digital de imágenes o bien visión por computadora.

Hoy día se estudia cómo mejorar la complejidad en la codificación de los vectores de movimiento y esquemas de cuantización multicapa para una óptima ordenación de los bits.

La optimización de imágenes para diseños de pantalla se deberá tener en cuenta la calidad de la imagen en relación con el menor peso posible.

El contraste tonal hace referencia al brillo de los elementos de una imagen. Si tu foto está compuesta por áreas extremadamente claras y oscuras, se puede considerar de alto contraste. Si presenta una amplia gama de tonos que van desde el blanco al negro, ambos puros, se trata de una imagen de contraste medio.

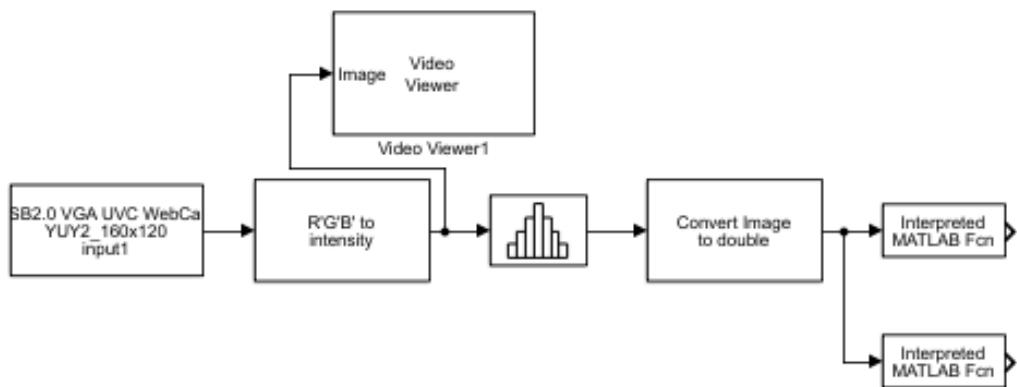
Finalmente podemos decir que los histogramas son herramientas muy útiles al momento de trabajar con imágenes ya que nos permite observar bastantes características que nos dan información relevante.

BIBLIOGRAFIA

1. Es.acervolima.com. 2022. Ecualización de histograma en C | Procesamiento de imágenes – Acervo Lima. [online] Available at: <<https://es.acervolima.com/ecualizacion-de-histograma-en-c-procesamiento-de-imagenes/>> [Accessed 25 August 2022].
2. 2022. [online] Available at: <<https://www.cognex.com/es-mx/products/machine-vision/vision-software/vision-tools/histogram-image-processing>> [Accessed 25 August 2022].
3. Pro.arcgis.com. 2022. Histograma de imágenes—ArcGIS Pro | Documentación. [online] Available at: <<https://pro.arcgis.com/es/pro-app/latest/help/analysis/geoprocessing/charts/image-histogram.htm>> [Accessed 25 August 2022].
4. Choque Huanca, H., 2022. Ecualización del histograma para el procesamiento digital de imágenes estáticas. [online] Repositorio.umsa.bo. Available at: <<https://repositorio.umsa.bo/handle/123456789/17511>> [Accessed 25 August 2022].

Anexos

Diagrama a bloques completo:



Asignatura: Procesamiento de imágenes

Profesor: D. Sc. Gerardo García Gil

Alumno: Carlo Pinedo Suarez

Registro: 19310163 **Ciclo:** 2022-B

Ingeniería en Desarrollo de Software

Centro de Enseñanza Técnica Industrial (CETI)

INTRODUCCIÓN

El procesamiento digital de imágenes ha adquirido, en años recientes, un papel importante en las tecnologías de la información y el cómputo. Actualmente, es la base de una creciente variedad de aplicaciones que incluyen diagnóstico médica, percepción remota, exploración espacial, visión por computadora, etc. Como resultado directo de la reducción en el precio de las computadoras, el procesamiento digital de imágenes actualmente se puede efectuar (aunque con ciertas limitantes) en una computadora personal.

La proliferación de nuevos equipamientos con capacidad para realizar millones de operaciones por segundo y su extensión a la vida cotidiana y a todo tipo de usuario, ha hecho posible que el análisis y procesamiento de imágenes digitales se constituya en un gran campo de estudio. En la actualidad, esta tecnología se encuentra incorporada incluso en todo tipo de equipamiento doméstico, como cámaras digitales, escaners y teléfonos celulares, entre otros.

En términos históricos, la utilización de imágenes radiográficas para diagnóstico clínico data prácticamente desde el descubrimiento de los rayos X en 1895. Incluso, las imágenes funcionales a partir de la emisión de fotones (rayos γ) por parte de radionucleidos ya cuenta con más de 90 años de antigüedad. Sin embargo, las imágenes eran adquiridas sobre films radiográficos o directamente in vivo, por lo que su correcto procesamiento no ha explotado su real potencialidad sino hasta la incorporación de la tecnología que permitió digitalizarlas.

El motivo principal de esta “aparición tardía” del procesamiento de imágenes ha sido entonces, debido a los requerimientos de hardware tanto para el procesamiento de las mismas como para la representación de estas en sistemas gráficos de alta performance. Paralelamente a este desarrollo, la formulación de algoritmos para el procesamiento ha seguido los avances tecnológicos logrando un alto grado de sofisticación y manipulación de imágenes en tiempo casi real.

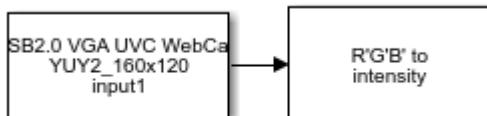
Una herramienta actual para el procesamiento de imágenes es el uso de histogramas. El histograma es una representación estadística que nos indica la proporción de píxeles de cada tonalidad que hay en una imagen. Si es negro o blanco significa que representa los niveles de grises o luminosidad. Recordemos que dos colores diferentes pueden tener una misma luminosidad o, dicho de otro modo, que si los desaturamos ambos corresponderán a un mismo gris. También podemos ver los histogramas correspondientes a cada canal R (rojo), G (verde), B (azul), y entonces se nos representará con el color del canal correspondiente. Para simplificar la cuestión, a partir de ahora sólo veremos histogramas de representación de la luminosidad.

DESARROLLO

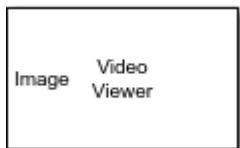
Simulink

Simulink es un entorno de diagramas de bloque que se utiliza para diseñar sistemas con modelos multidominio, simular antes de implementar en hardware y desplegar sin necesidad de escribir código.

Para esta práctica será necesario hacer el uso de la cámara para grabar video ya que se pretende realizar el histograma simple y lineal de este, para esto comenzaremos agregando el bloque que realizara tal tarea y luego convertiremos la imagen rgb a un formato donde podamos medir la intensidad luminosa que es la escala de grises



Para ver el video obtenido se usa lo siguiente:

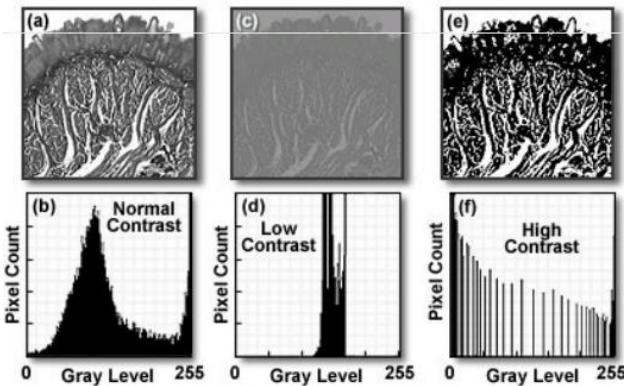


Mas adelante mostraremos el diagrama completo a bloques utilizado, pero por lo pronto hasta aquí será suficiente para la explicación del desarrollo.

Histograma simple

Supongamos dada una imagen en niveles de grises, siendo el rango de 256 tonos de gris (de 0 a 255). El histograma de la imagen consiste en una gráfica donde se muestra el número de píxeles, n_k , de cada nivel de gris, r_k , que aparecen en la imagen. En el siguiente ejemplo podemos ver tres imágenes con sus correspondientes histogramas.

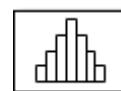
Grayscale Histograms and Contrast Levels in Digital Images



El análisis estadístico derivado del histograma puede servir para comparar contrastes e intensidades entre imágenes. El histograma podría ser alterado para producir cambios en la imagen. Por ejemplo, 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 Histogramas imagen. La forma usual de binarizar una imagen es eligiendo un valor adecuado o umbral, u , dentro de los niveles de grises, tal que el histograma forme un "valle" en ese nivel. Todos los niveles de grises menores que u se convierten en 0 (negro), y los mayores que u se convierten en 255 (blanco).

Para el caso de esta práctica en el caso del histograma simple donde el video original fue transformado a blanco y negro para después proceder a mostrar su dicho histograma, básicamente lo que se hizo fue leer el video original para obtener una referencia de esta, después se mandó a llamar la función que convierte a escala de grises y luego en otra variable se guardaron los valores de cada pixel formando así el histograma.

El bloque implementado en simulink implementado fue el siguiente:



Para mostrar el histograma en este caso yo solo use una función que me permitirá ver el histograma desde un plot, hay que tomar en cuenta que hay que convertir el formato double, mas adelante se vera claramente en el diagrama a bloques, el siguiente es el código usado:

```
function histSimple=Histograma_Simple(imagenGris)

%Histograma simple
histSimple=imagenGris*200;

subplot(2,2,1);
plot(histSimple);
title("Histograma Simple");
end
```

Histograma acumulado

Un histograma acumulado es un gráfico de líneas que se usa para presentar las frecuencias absolutas de los valores de una distribución en el cual la altura del punto asociado a un valor de las variables es proporcional a la frecuencia de dicho valor.

El histograma acumulativo es una variante del histograma normal, refleja información importante para las operaciones de pixel, por ejemplo, para equilibrar un histograma.

La función del histograma acumulativo se define como una función monótona creciente en donde todos los valores a partir del segundo valor son recalculados con la suma de su valor inmediato anterior. El histograma acumulativo esta definido por la siguiente ecuación:

$H(i) = \sum_{j=0}^i H(j)$ para $0 \leq i < K$ Una vez que se convirtió el video a double después de obtener el histograma simple agregamos en una función el código necesario para realizar dicho histograma:

```
function histAcum=Histograma_Acum(imagenGris)

%Histograma simple
histSimple=imagenGris*200;

histAcum(1)=histSimple(1);
for i=2:256
    histAcum(i)=histSimple(i)+histAcum(i-1);
end
subplot(2,2,2);
plot(histAcum);
title("Histograma Acumulado");

end
```

Histograma lineal o ecualizado

La ecualización de histograma es una técnica para ajustar los niveles de contraste y ampliar el rango de intensidad en una imagen digital. Por lo tanto, mejora la imagen, lo que facilita la extracción de información y el procesamiento posterior de la imagen.

Reparte de forma más o menos uniforme los valores del histograma.

Dada una imagen $M \times N$, con n píxeles para cada nivel r , la ecualización del histograma consiste en realizar la siguiente transformación sobre los niveles de intensidad de la imagen:

$$Sk = T(rk) = (L - 1) \sum_{j=0}^k pr(rj) = \frac{L - 1}{MN} \sum_{j=0}^k nj$$

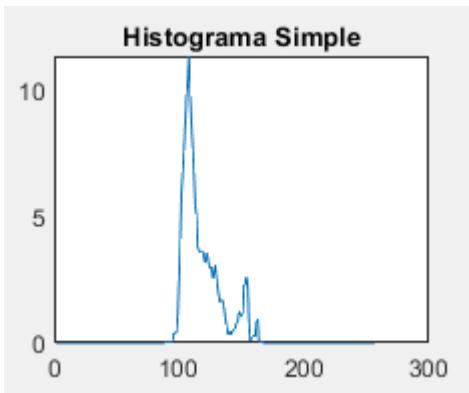
lo que resulta en una dispersión del histograma en un rango mayor dentro del intervalo $[0, L-1]$. La principal ventaja de este método es que es completamente "automático".

El código implementado es el siguiente:

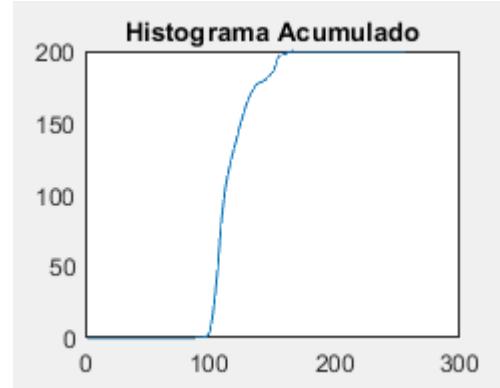
```
function IM = Ecualizacion_Lineal(u)
%Determinar el histograma de la imagen %fuente (u)
[datos,x]=imhist(u);
[R,C]=size(u);
sum=0;
IM=zeros(R,C);
H=zeros(1,256);
%Calcular Histograma Acumulado
for cont=1:256
    H(cont)=datos(cont)+sum;
    sum=H(cont);
end
%Aplicar la Ecuación 2 para la %ecualización lineal
for re=1:R
    for co=1:C
        temp=floor(u(re,co)*255);
        IM(re,co)=H(temp+1)/(R*C);
    end
end
```

RESULTADOS

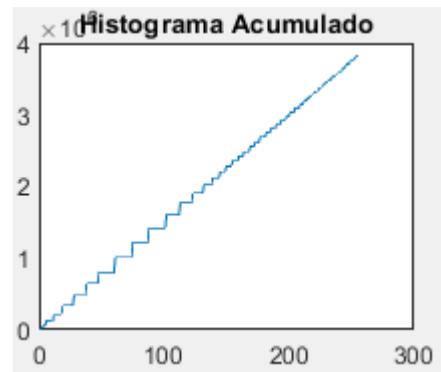
Histograma Simple



Histograma acumulado



Histograma Ecualizado



CONCLUSIONES

El procesamiento digital de imágenes y la visión por computadora son tópicos que se utilizan cada vez más en diversas áreas, tales como publicidad, cinematografía, meteorología, medicina, etc.

El aprovechar el alto desempeño del MATLAB para procesar matrices y vectores, así como el alto desempeño en graficación permite implementar sistemas de procesamiento digital de imágenes o bien visión por computadora.

Hoy día se estudia cómo mejorar la complejidad en la codificación de los vectores de movimiento y esquemas de cuantización multicapa para una óptima ordenación de los bits.

La optimización de imágenes para diseños de pantalla se deberá tener en cuenta la calidad de la imagen en relación con el menor peso posible.

El contraste tonal hace referencia al brillo de los elementos de una imagen. Si tu foto está compuesta por áreas extremadamente claras y oscuras, se puede considerar de alto contraste. Si presenta una amplia gama de tonos que van desde el blanco al negro, ambos puros, se trata de una imagen de contraste medio.

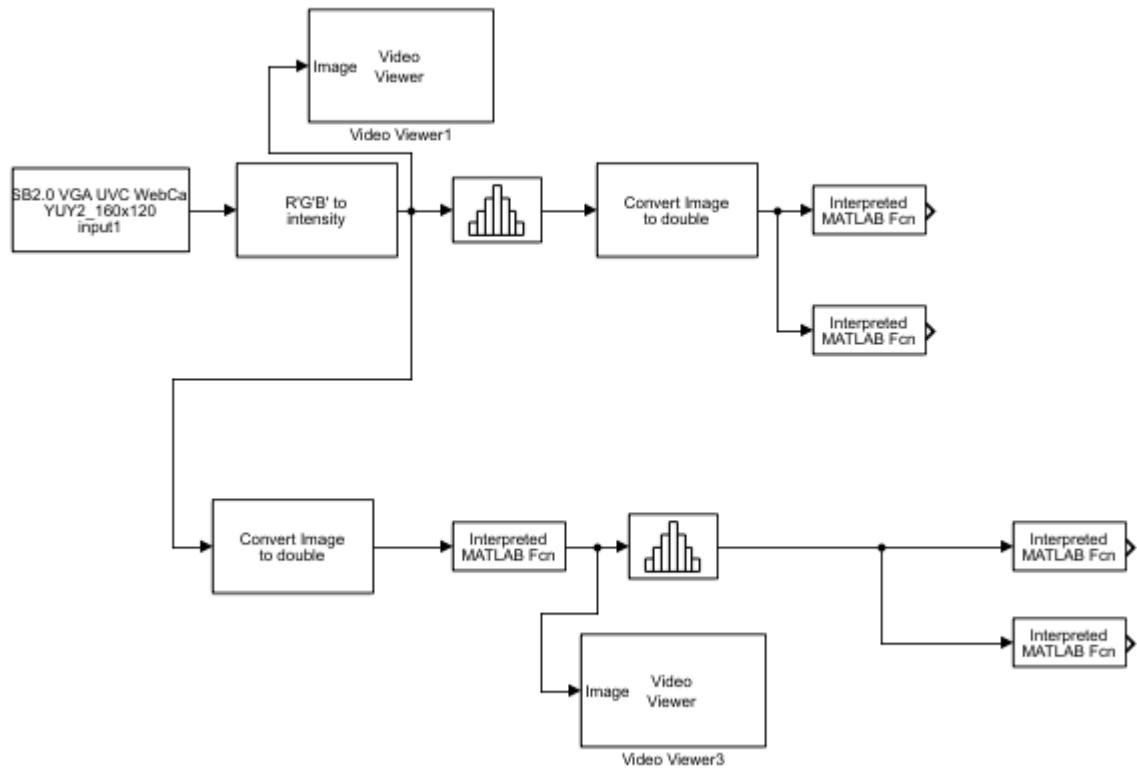
Finalmente podemos decir que los histogramas son herramientas muy útiles al momento de trabajar con imágenes ya que nos permite observar bastantes características que nos dan información relevante.

BIBLIOGRAFIA

1. Es.acervolima.com. 2022. Ecualización de histograma en C | Procesamiento de imágenes – Acervo Lima. [online] Available at: <<https://es.acervolima.com/ecualizacion-de-histograma-en-c-procesamiento-de-imagenes/>> [Accessed 25 August 2022].
2. 2022. [online] Available at: <<https://www.cognex.com/es-mx/products/machine-vision/vision-software/vision-tools/histogram-image-processing>> [Accessed 25 August 2022].
3. Pro.arcgis.com. 2022. Histograma de imágenes—ArcGIS Pro | Documentación. [online] Available at: <<https://pro.arcgis.com/es/pro-app/latest/help/analysis/geoprocessing/charts/image-histogram.htm>> [Accessed 25 August 2022].
4. Choque Huanca, H., 2022. Ecualización del histograma para el procesamiento digital de imágenes estáticas. [online] Repositorio.umsa.bo. Available at: <<https://repositorio.umsa.bo/handle/123456789/17511>> [Accessed 25 August 2022].

Anexos

Diagrama a bloques completo:



Asignatura: Procesamiento de imágenes

Profesor: D. Sc. Gerardo García Gil

Alumno: Carlo Pinedo Suarez

Registro: 19310163 **Ciclo:** 2022-B

Ingeniería en Desarrollo de Software

Centro de Enseñanza Técnica Industrial (CETI)

INTRODUCCIÓN

El análisis y procesamiento de imágenes se realiza a través de computadoras, debido a la complejidad y el número de cálculos necesarios para realizarlo. Es por esto que, si bien la formulación matemática necesaria para su realización data de varios siglos atrás, la posibilidad real de utilizarla de forma cotidiana en la práctica clínica ha sido posible recién en las últimas décadas, gracias al avance en las tecnologías del hardware.

La proliferación de nuevos equipamientos con capacidad para realizar millones de operaciones por segundo y su extensión a la vida cotidiana y a todo tipo de usuario, ha hecho posible que el análisis y procesamiento de imágenes digitales se constituya en un gran campo de estudio. En la actualidad, esta tecnología se encuentra incorporada incluso en todo tipo de equipamiento doméstico, como cámaras digitales, scaners y teléfonos celulares, entre otros.

En términos históricos, la utilización de imágenes radiográficas para diagnóstico clínico data prácticamente desde el descubrimiento de los rayos X en 1895 (Röentgen). Incluso, las imágenes funcionales a partir de la emisión de fotones (rayos γ) por parte de radionucleidos ya cuenta con más de 90 años de antigüedad (Hevesy & Seaborg, 1924). Sin embargo, las imágenes eran adquiridas sobre films radiográficos o directamente *in vivo*, por lo que su correcto procesamiento no ha explotado su real potencialidad sino hasta la incorporación de la tecnología que permitió digitalizarlas.

El motivo principal de esta “aparición tardía” del procesamiento de imágenes ha sido entonces, debido a los requerimientos de hardware tanto para el procesamiento de las mismas como para la representación de estas en sistemas gráficos de alta performance. Paralelamente a este desarrollo, la formulación de algoritmos para el procesamiento ha seguido los avances tecnológicos logrando un alto

grado de sofisticación y manipulación de imágenes en tiempo casi real.

El procesamiento digital de imágenes es el conjunto de técnicas que se aplican a las imágenes digitales con el objetivo de mejorar la calidad o facilitar la búsqueda de información.

El proceso de filtrado es el conjunto de técnicas englobadas dentro del preprocesamiento de imágenes cuyo objetivo fundamental es obtener, a partir de una imagen origen, otra final cuyo resultado sea más adecuado para una aplicación específica mejorando ciertas características de la misma que posibilite efectuar operaciones del procesado sobre ella. Los principales objetivos que se persiguen con la aplicación de filtros son:

Suavizar la imagen. Reducir la cantidad de variaciones de intensidad entre pixeles vecinos. Eliminar ruido. Eliminar aquellos pixeles cuyo nivel de intensidad es muy diferente al de sus vecinos y cuyo origen puede estar tanto en el proceso de adquisición de la imagen como en el de transmisión. Realzar bordes. Destacar los bordes que se localizan en una imagen. Detectar bordes. Detectar los pixeles donde se produce un cambio brusco en la función intensidad. Por tanto, se consideran los filtros como operaciones que se aplican a los pixeles de una imagen digital para optimizarla, enfatizar cierta información o conseguir un efecto especial en ella. El proceso de filtrado puede llevarse a cabo sobre los dominios de frecuencia y/o espacio.

DESARROLLO

Filtrado en el Dominio del Espacio.

Las operaciones de filtrado se llevan a cabo directamente sobre los píxeles de la imagen. En este proceso se relaciona, para todos y cada uno de los puntos de la imagen, un conjunto de píxeles próximos al pixel objeto con la finalidad de obtener una información útil, dependiendo del tipo de filtro aplicado, que permita actuar sobre el pixel concreto en que se está llevando a cabo el proceso de filtrado para de este modo obtener mejoras sobre la imagen y/o datos que podrían ser utilizados en futuras acciones o procesos de trabajo sobre ella.

Existen 2 tipos de filtros para el dominio del espacio: Filtros lineales (filtros basados en máscaras de convolución) y filtros no lineales. Dependiendo de la implementación, en los límites de la imagen se aplica un tratamiento especial (se asume un marco exterior de ceros o se repiten los valores del borde) o no se aplica ninguno. Es por ello, que el tipo de filtrado queda establecido por el contenido de dicha máscara utilizado.

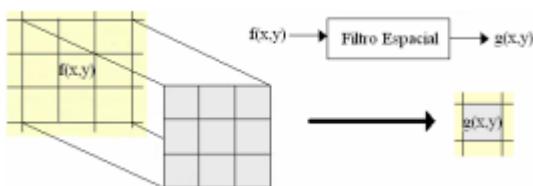


Figura 1. Matriz de coeficientes en el uso de la máscara.

Filtros lineales

Un filtro lineal realiza una suma de productos entre una imagen f y una ventana de filtro w . El tamaño de w ($m \times n$) define el vecindario y sus coeficientes determinan la naturaleza del filtro. En este ejemplo, el píxel modificado es

$$g(x, y) = w(-1, -1)f(x-1, y-1) + \dots + w(0, 0)f(x, y) + \dots + w(1, 1)f(x+1, y+1)$$

Filtros de suavizado

El objetivo de este tipo de filtros es reducir las variaciones de intensidad entre píxeles vecinos.

Entendiendo como ruido la información no deseada que contamina la imagen. El origen puede estar tanto en el proceso de adquisición de la imagen (errores en los sensores), como en el de transmisión (debido a interferencias en el canal de transmisión).

Características:

- Los filtros lineales para reducción de ruido se implementan mediante la convolución espacial.
- La suma de los coeficientes de un filtro suavizante debe ser la unidad para evitar introducir un sesgo de intensidad en el filtrado.
- El filtro paso bajos más simple calcula la media de los vecinos del píxel $f(x, y)$ y se denomina filtro de media aritmética o filtro de caja(filtro box).

Filtro box

El Filtro box es el más antiguo y sencillo de los filtros que se encargan de hacer el suavizado de una imagen. este Filtro debido a los bordes tan agudos qué tiene y su comportamiento frecuencial, resulta ser un Filtro de suavizado, además de su sencillez no es muy recomendable, su nombre proviene de su gráfica donde parece una caja, este afecta a cada pixel de la imagen de igual manera ya que todos sus coeficientes contienen el mismo valor.

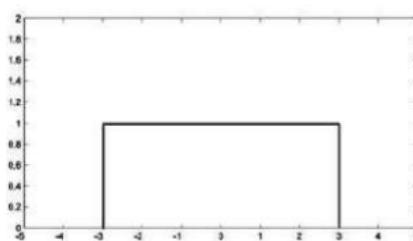


Figura 2. Gráfico bidimensional del filtro box.

Se visita cada píxel de la imagen y se reemplaza por la media de los píxeles vecinos. Se puede operar mediante convolución con una máscara determinada.

La ventana de un filtro box se implementa como:

$$w = \frac{1}{mn} \begin{bmatrix} 1 & 2 & \cdots & m \\ 1 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ \vdots \\ n \end{bmatrix}$$

Y su ecuación es básicamente la siguiente, donde podemos ver que tal cual es una operación para obtener la media.

$$g(x, y) = \frac{\sum_{(s,t) \in w} f(s, t)}{\sum_{(s,t) \in w} w(s, t)}$$

Para esta práctica se implementarán 2 máscaras en específico para este Filtro uno de 9 por 9 al que llamaremos Filtro de borrado y otro de 5 por 5 al que llamaremos Filtro box, el código implementado es el siguiente:

```
%Filtro de borrado
Im=double(imagen);
ImR=Im;
for r=2:M-1
    for c=2:N-1
        ImR(r,c)=1/9*(Im(r-1,c-1)+Im(r-1,c)+Im(r-1,c+1) ...
            +Im(r,c-1)+Im(r,c)+Im(r,c+1) ...
            +Im(r+1,c-1)+Im(r+1,c)+Im(r+1,c+1));
    end
end
ImR=uint8(ImR);
figure(2);
imshow(ImR);
```

```
%Filtro box
Im=double(imagen);
ImR=Im;
for r=3:M-2
    for c=3:N-2
        ImR(r,c)=1/25*(Im(r-2,c-2)+Im(r-1,c-2)+Im(r,c-2) ...
            +Im(r+1,c-2)+Im(r+2,c-2) ...
            +Im(r-2,c-1)+Im(r-1,c-1) ...
            +Im(r,c-1)+Im(r+1,c-1) ...
            +Im(r+2,c-1)+Im(r-2,c) ...
            +Im(r-1,c)+Im(r,c)+Im(r+1,c) ...
            +Im(r+2,c)+Im(r-2,c+1)+Im(r-1,c+1) ...
            +Im(r,c+1)+Im(r+1,c+1)+Im(r+2,c+1) ...
            +Im(r-2,c+2)+Im(r-1,c+2)+Im(r,c+2) ...
            +Im(r+1,c+2)+Im(r+2,c+2));
    end
end
ImR=uint8(ImR);
figure(3);
imshow(ImR);
```

Filtro de Gauss

El filtro gaussiano, es un filtro lineal que se usa para embozzonar imágenes y eliminar ruido. Es similar al filtro de media pero se usa una máscara diferente, modelizando la función gaussiana:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Y su grafica bidimensional es la siguiente:

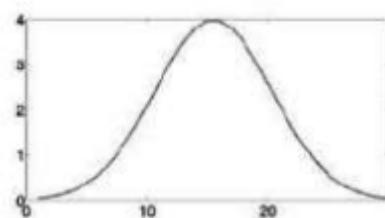


Figura 3. Gráfico bidimensional del filtro de Gauss.

Pero para entender la máscara aplicada en ese Filtro hay que observar su gráfica tridimensional donde como podemos ver se forma una especie de cono donde en la mascarilla o máscara el valor central representa la punta del cono qué es la intensidad mayor, y siguiendo la distribución de la misma gráfica podemos ver que los valores van de decrementando lo mismo pasa en la máscara uno de los valores que rodean el punto central van decrementando.

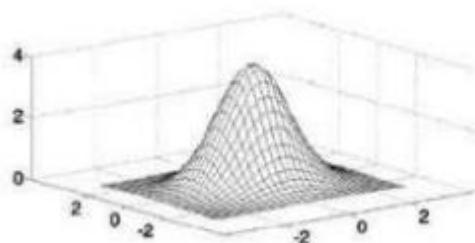


Figura 4. Gráfico tridimensional del filtro de Gauss.

Esta es la máscara usada para esta práctica;

0	1	2	1	0
1	3	5	3	1
2	5	9	5	2
1	3	5	3	1
0	1	2	1	0

Figura 5. Mascara del filtro de Gauss.

El código implementado será el siguiente:

```
%Filtro de Gauss
Im=double(imagen);
ImR=Im;
for r=3:M-2
    for c=3:N-2
        ImR(r,c)=1/57*(Im(r-2,c-2)*0+Im(r-1,c-2)*1+Im(r,c-2)*2 ...
            +Im(r+1,c-2)*1+Im(r+2,c-2)*0+Im(r-2,c-1)*1 ...
            +Im(r-1,c-1)*3+Im(r,c-1)*5+Im(r+1,c-1)*3 ...
            +Im(r+2,c-1)*1+Im(r-2,c)*2+Im(r-1,c)*5 ...
            +Im(r,c)*9+Im(r+1,c)*5+Im(r+2,c)*2 ...
            +Im(r-2,c+1)*1+Im(r-1,c+1)*3+Im(r,c+1)*5 ...
            +Im(r+1,c+1)*3+Im(r+2,c+1)*1+Im(r-2,c+2)*0 ...
            +Im(r-1,c+2)*1+Im(r,c+2)*2+Im(r+1,c+2)*1 ...
            +Im(r+2,c+2)*0);
    end
end
ImR=uint8(ImR);
figure(4);
imshow(ImR);
```

Filtro Laplaciano

Filtro Laplaciano crea una máscara que realza los pixeles con respecto a su vecindad atreves del aumento de su nivel de grises. Se encuentra definido primera mente por un operador Laplaciano:

$$\frac{\partial f(x)}{\partial x} = f(x+1) - f(x)$$

Este tipo de filtro se basa en un operador derivativo, por lo que acentúa las zonas que tienen gran discontinuidad en la imagen

Grafica bidimensional:

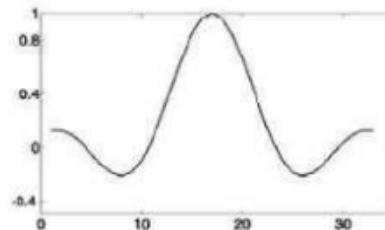


Figura 6. Gráfico bidimensional del filtro de Laplace.

Al igual que para el Filtro gaussiano la máscara se obtiene a partir en su gráfica tridimensional por eso procederemos a mostrarla.

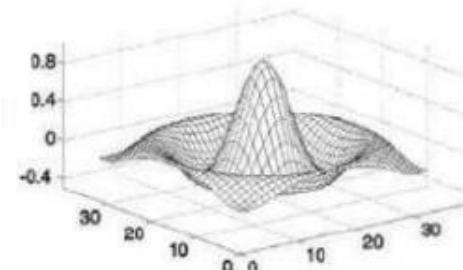


Figura 7. Gráfico tridimensional del filtro de Laplace.

Por último, esta es la mascarilla que se usará para este Filtro en la realización de esta práctica.

0	0	-1	0	0
0	-1	-2	-1	0
-1	-2	16	-2	-1
0	-1	-2	-1	0
0	0	-1	0	0

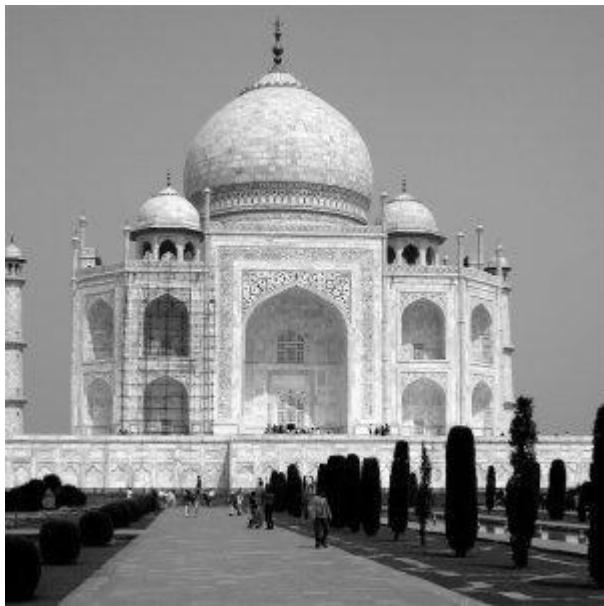
Figura 8. Mascara del filtro de Laplace.

El código implementado será el siguiente:

```
%Filtro Laplaciano
Im=double(imagen);
ImR=Im;
for r=3:M-2
    for c=3:N-2
        ImR(r,c)=(Im(r-2,c-2)*0+Im(r-1,c-2)*0+Im(r,c-2)*-1 ...
            +Im(r+1,c-2)*0+Im(r+2,c-2)*0+Im(r-2,c-1)*0 ...
            +Im(r-1,c-1)*-1+Im(r,c-1)*-2+Im(r+1,c-1)*-1 ...
            +Im(r+2,c-1)*0+Im(r-2,c)*-1+Im(r-1,c)*-2 ...
            +Im(r,c)*16+Im(r+1,c)*-2+Im(r+2,c)*-1 ...
            +Im(r-2,c+1)*0+Im(r-1,c+1)*-1+Im(r,c+1)*-2 ...
            +Im(r+1,c+1)*-1+Im(r+2,c+1)*0+Im(r-2,c+2)*0 ...
            +Im(r-1,c+2)*0+Im(r,c+2)*-1+Im(r+1,c+2)*0 ...
            +Im(r+2,c+2)*0;
    end
end
ImR=uint8(ImR);
figure(5);
imshow(ImR);
```

RESULTADOS

Imagen original



Filtro de borrado (Filtro box 9x9)



Filtro box 5x5



Filtro de Gauss



Filtro de Laplace



CONCLUSIONES

En este trabajo se implementaron filtros distintos para suavizar imágenes como lo son el Filtro box o Filtro de media, Filtro de gauss y el Filtro laplaciano o mexican hat.

Como ya vimos el suavizado de una imagen nos permite reducir el ruido de esta para tener una imagen mejor con la cual trabajar más adelante. Podemos ver que cada uno de ellos nos entrega un resultado distinto sin embargo depende de cuál será nuestra aplicación para decidir cuál usar, cada uno de estos juega con los contrastes de cada pixel de la imagen.

También hay que destacar que al ser filtros espaciales estos dependen de los pixeles que se tengan alrededor por tanto mientras más pixeles se utilicen mayor variación habrá en el resultado que se obtenga, para caso de Gauss y Laplace también dependerá del valor de los coeficientes previos en su ecuación.

BIBLIOGRAFIA

- [1] R. Castillo. "Procesamiento Digital de Imágenes Empleando Filtros Espaciales". [iiis.org](https://www.iiis.org/CDs2013/CD2013SCI/CIS_CI_2013/PapersPdf/CA780XP.pdf). https://www.iiis.org/CDs2013/CD2013SCI/CIS_CI_2013/PapersPdf/CA780XP.pdf (accedido el 18 de septiembre de 2022).
- [2] E. Cuevas, M. Diaz y J. Camarena, Tratamiento de imágenes con MATLAB. México: Alfaomega, 2017.
- [3] O. Juárez. "Filtros-laplaciano, mediana y media". [studocu](https://www.studocu.com/pe/document/universidad-peruana-de-ciencias-aplicadas/matematica-computacional/filtros-laplaciano-mediana-y-media/18914123). <https://www.studocu.com/pe/document/universidad-peruana-de-ciencias-aplicadas/matematica-computacional/filtros-laplaciano-mediana-y-media/18914123> (accedido el 18 de septiembre de 2022).
- [4] W. Gómez. "Análisis de Imágenes Digitales". [cinvestav](https://www.tamps.cinvestav.mx/~wgomez/diapositivas/AID/Clase04.pdf). <https://www.tamps.cinvestav.mx/~wgomez/diapositivas/AID/Clase04.pdf> (accedido el 18 de septiembre de 2022).

Anexos

Código implementado :

```
imagen=imread("buda.jpg");
imagen=rgb2gray(imagen);
figure(1);
imshow(imagen);
[M,N,dir] = size(imagen);

%Filtro de borrado
Im=double(imagen);
ImR=Im;
for r=2:M-1
    for c=2:N-1
        ImR(r,c)=1/9*(Im(r-1,c-1)+Im(r-1,c)+Im(r-1,c+1) ...
                      +Im(r,c-1)+Im(r,c)+Im(r,c+1) ...
                      +Im(r+1,c-1)+Im(r+1,c)+Im(r+1,c+1));
    end
end
ImR=uint8(ImR);
figure(2);
imshow(ImR);

%Filtro box
Im=double(imagen);
ImR=Im;
for r=3:M-2
    for c=3:N-2
        ImR(r,c)=1/25*(Im(r-2,c-2)+Im(r-1,c-2)+Im(r,c-2)+Im(r+1,c-2) ...
                      +Im(r+2,c-2)+Im(r-2,c-1)+Im(r-1,c-1) ...
                      +Im(r,c-1)+Im(r+1,c-1)+Im(r+2,c-1)+Im(r-2,c) ...
                      +Im(r-1,c)+Im(r,c)+Im(r+1,c)+Im(r+2,c) ...
                      +Im(r-2,c+1)+Im(r-1,c+1)+Im(r,c+1) ...
                      +Im(r+1,c+1)+Im(r+2,c+1)+Im(r-2,c+2) ...
                      +Im(r-1,c+2)+Im(r,c+2)+Im(r+1,c+2)+Im(r+2,c+2));
    end
end
ImR=uint8(ImR);
figure(3);
imshow(ImR);
```

```

%Filtro de Gauss
Im=double(imagen);
ImR=Im;
for r=3:M-2
    for c=3:N-2
        ImR(r,c)=1/57*(Im(r-2,c-2)*0+Im(r-1,c-2)*1+Im(r,c-2)*2 ...
            +Im(r+1,c-2)*1+Im(r+2,c-2)*0+Im(r-2,c-1)*1 ...
            +Im(r-1,c-1)*3+Im(r,c-1)*5+Im(r+1,c-1)*3 ...
            +Im(r+2,c-1)*1+Im(r-2,c)*2+Im(r-1,c)*5 ...
            +Im(r,c)*9+Im(r+1,c)*5+Im(r+2,c)*2 ...
            +Im(r-2,c+1)*1+Im(r-1,c+1)*3+Im(r,c+1)*5 ...
            +Im(r+1,c+1)*3+Im(r+2,c+1)*1+Im(r-2,c+2)*0 ...
            +Im(r-1,c+2)*1+Im(r,c+2)*2+Im(r+1,c+2)*1 ...
            +Im(r+2,c+2)*0);

    end
end
ImR=uint8(ImR);
figure(4);
imshow(ImR);

%Filtro Laplaciano
Im=double(imagen);
ImR=Im;
for r=3:M-2
    for c=3:N-2
        ImR(r,c)=(Im(r-2,c-2)*0+Im(r-1,c-2)*0+Im(r,c-2)*-1 ...
            +Im(r+1,c-2)*0+Im(r+2,c-2)*0+Im(r-2,c-1)*0 ...
            +Im(r-1,c-1)*-1+Im(r,c-1)*-2+Im(r+1,c-1)*-1 ...
            +Im(r+2,c-1)*0+Im(r-2,c)*-1+Im(r-1,c)*-2 ...
            +Im(r,c)*16+Im(r+1,c)*-2+Im(r+2,c)*-1 ...
            +Im(r-2,c+1)*0+Im(r-1,c+1)*-1+Im(r,c+1)*-2 ...
            +Im(r+1,c+1)*-1+Im(r+2,c+1)*0+Im(r-2,c+2)*0 ...
            +Im(r-1,c+2)*0+Im(r,c+2)*-1+Im(r+1,c+2)*0 ...
            +Im(r+2,c+2)*0);

    end
end
ImR=uint8(ImR);
figure(5);
imshow(ImR);

```

Asignatura: Procesamiento de imágenes

Profesor: D. Sc. Gerardo García Gil

Alumno: Carlo Pinedo Suarez

Registro: 19310163 **Ciclo:** 2022-B

Ingeniería en Desarrollo de Software

Centro de Enseñanza Técnica Industrial (CETI)

INTRODUCCIÓN

El análisis y procesamiento de imágenes se realiza a través de computadoras, debido a la complejidad y el número de cálculos necesarios para realizarlo. Es por esto que, si bien la formulación matemática necesaria para su realización data de varios siglos atrás, la posibilidad real de utilizarla de forma cotidiana en la práctica clínica ha sido posible recién en las últimas décadas, gracias al avance en las tecnologías del hardware.

La proliferación de nuevos equipamientos con capacidad para realizar millones de operaciones por segundo y su extensión a la vida cotidiana y a todo tipo de usuario, ha hecho posible que el análisis y procesamiento de imágenes digitales se constituya en un gran campo de estudio. En la actualidad, esta tecnología se encuentra incorporada incluso en todo tipo de equipamiento doméstico, como cámaras digitales, scaners y teléfonos celulares, entre otros.

En términos históricos, la utilización de imágenes radiográficas para diagnóstico clínico data prácticamente desde el descubrimiento de los rayos X en 1895 (Röentgen). Incluso, las imágenes funcionales a partir de la emisión de fotones (rayos γ) por parte de radionucleidos ya cuenta con más de 90 años de antigüedad (Hevesy & Seaborg, 1924). Sin embargo, las imágenes eran adquiridas sobre films radiográficos o directamente in vivo, por lo que su correcto procesamiento no ha explotado su real potencialidad sino hasta la incorporación de la tecnología que permitió digitalizarlas.

El motivo principal de esta “aparición tardía” del procesamiento de imágenes ha sido entonces, debido a los requerimientos de hardware tanto para el procesamiento de las mismas como para la representación de estas en sistemas gráficos de alta performance. Paralelamente a este desarrollo, la formulación de algoritmos para el procesamiento ha seguido los avances tecnológicos logrando un alto

grado de sofisticación y manipulación de imágenes en tiempo casi real.

El procesamiento digital de imágenes es el conjunto de técnicas que se aplican a las imágenes digitales con el objetivo de mejorar la calidad o facilitar la búsqueda de información.

El proceso de filtrado es el conjunto de técnicas englobadas dentro del preprocesamiento de imágenes cuyo objetivo fundamental es obtener, a partir de una imagen origen, otra final cuyo resultado sea más adecuado para una aplicación específica mejorando ciertas características de la misma que posibilite efectuar operaciones del procesado sobre ella. Los principales objetivos que se persiguen con la aplicación de filtros son:

Suavizar la imagen. Reducir la cantidad de variaciones de intensidad entre pixeles vecinos. Eliminar ruido. Eliminar aquellos pixeles cuyo nivel de intensidad es muy diferente al de sus vecinos y cuyo origen puede estar tanto en el proceso de adquisición de la imagen como en el de transmisión. Realzar bordes. Destacar los bordes que se localizan en una imagen. Detectar bordes. Detectar los pixeles donde se produce un cambio brusco en la función intensidad. Por tanto, se consideran los filtros como operaciones que se aplican a los pixeles de una imagen digital para optimizarla, enfatizar cierta información o conseguir un efecto especial en ella. El proceso de filtrado puede llevarse a cabo sobre los dominios de frecuencia y/o espacio.

DESARROLLO

Filtros no lineales

En el procesamiento de señales , un filtro no lineal (o no lineal) es un filtro cuya salida no es una función lineal de su entrada. Es decir, si el filtro emite señales R y S para dos señales de entrada r y s por separado, pero no siempre emite $\alpha R + \beta S$ cuando la entrada es una combinación lineal $\alpha r + \beta s$.

Los filtros no lineales tienen muchas aplicaciones, especialmente en la eliminación de ciertos tipos de ruido que no son aditivos . Por ejemplo, el filtro mediano se usa ampliamente para eliminar el ruido de pico, que afecta solo a un pequeño porcentaje de las muestras, posiblemente en cantidades muy grandes.

Filtro de la mediana

Este tipo de Filtro suele ser utilizado para eliminar ruido en la imagen.

El funcionamiento es el siguiente, Se visita cada píxel 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 seleccionando el que queda en medio.

Tiene la desventaja de ser un Filtro no lineal pero la ventaja de dar muy buenos resultados en el Filtro de sal y pimienta

Un filtro de mediana es más eficaz que la convolución cuando el objetivo es reducir el ruido y mantener los bordes al mismo tiempo.

La idea principal del filtro de mediana es reemplazar cada entrada con el valor mediano de su vecindario.

Para la práctica usaremos 9 píxeles así que veremos un ejemplo similar, aquí tenemos una matriz donde vemos que el centro tiene un valor de 111:

5	6	7
6	111	8
7	8	9

El Filtro de mediana tomará todos estos valores de la vecindad y los ordenará.

5	6	6	7	7	8	8	9	111
---	---	---	---	---	---	---	---	-----

Una vez ordenados podemos ver que el valor central es el 7, este es el valor que reemplazará al pixel.

5	6	7
6	7	8
7	8	9

Características y usos

El filtro de mediana permite eliminar los valores atípicos sin limitarse a realizar un cálculo promedio que tenderá a contaminar los valores vecinos con este valor atípico y difuminar la imagen.

El filtro de mediana respeta el contraste de la imagen (si multiplicamos todos los valores por una constante positiva, el orden de los valores no cambia) y el brillo de la imagen (añadir una constante tampoco modifica el orden).

En áreas donde la intensidad es monótona (solo aumenta o solo disminuye), el filtro deja la imagen sin cambios. Respeta los contornos y elimina los valores extremos.

Para esta práctica se implementará el siguiente código donde básicamente se lee la imagen y se le aplica lo mencionado anteriormente:

```
%Mediana
ImR=Im;
vecinos=ones(1,9);
for i=2:M-1
    for j=2:N-1
        contVec=1;
        for r=-1:1
            for c=-1:1
                vecinos(contVec)=Im(i+r,j+c);
                contVec=contVec+1;
            end
        end
        vecOrd=sort(vecinos);
        ImR(i,j)=vecOrd(5);
    end
end
subplot(2,2,2);
imshow(uint8(ImR));
title("Filtro de Mediana");
```

Ruido Sal y Pimienta

El ruido de sal y pimienta también se conoce como ruido de impulso bipolar. La característica de este ruido es que el valor de gris del píxel de ruido es significativamente diferente del píxel vecino, y el valor de gris de los píxeles restantes permanece sin cambios. Píxeles brillantes u demasiado oscuros. El ruido de sal y pimienta afecta seriamente la calidad visual de la imagen y causa dificultades en la detección de bordes, la textura o la extracción de puntos de la imagen.

Este tipo de ruido suele producirse cuando la señal de la imagen es afectada por intensas y repentinas perturbaciones o impulsos.

Una forma efectiva para la reducción de este tipo de ruido es mediante el uso de filtros de mediana o filtros morfológicos. A diferencia de los filtros basados en la media empleados para la reducción de ruido Gaussiano, la salida de los filtros medianos utiliza el valor de la mediana de la vecindad de cada píxel, por lo que este tipo de ruido puede ser rápida y eficientemente eliminado.

En la siguiente imagen de ejemplo podemos ver claramente lo que se trata de decir sobre este tipo de ruido:



Figura 1. Ejemplo de imagen con ruido sal y pimienta.

El código implementado para aplicar este tipo de ruido es bastante sencillo ya que sólo hay que agregar puntos negros (pimienta) y puntos blancos (sal), de manera que sólo hay que agregar valores ya sean ser 0 o 255 sustituyendo valores aleatorios de intensidades de pixeles.

El código será el siguiente:

```
%Sal y pimienta
ImS=Im;
cantPuntos=round((M*N)*0.01);

for i=0:cantPuntos
    x=round((rand()*(M-1))+2);
    y=round((rand()*(N-1))+2);
    if x>=M
        x=M-1;
    end
    if y>=N
        y=N-1;
    end
    ImS(x,y)=255;
end

for i=0:cantPuntos
    x=round((rand()*(M-1))+2);
    y=round((rand()*(N-1))+2);
    if x>=M
        x=M-1;
    end
    if y>=N
        y=N-1;
    end
    ImS(x,y)=0;
end

subplot(2,2,3);
imshow(uint8(ImS));
title("Ruido Sal y pimienta");
```

RESULTADOS

Para esta práctica es importante mencionar que el Filtro de mediana se aplicará 2 veces uno a la imagen original, después como vimos anteriormente hay un código para agregar ruido de sal y pimienta, que será agregado a la imagen original, y a esta última imagen es la que le volveremos a hacer el filtrado de mediana para ver cómo se corrigen los puntos blancos y negros.

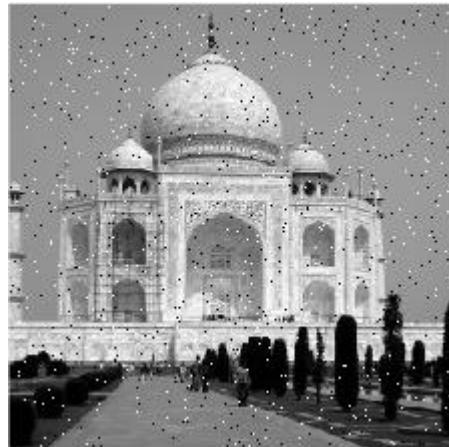
Imagen original



Imagen original con filtro de mediana



Imagen con ruido sal y pimienta



Filtro de mediana con sal y pimienta



CONCLUSIONES

En ese trabajo se usó o un tipo de Filtro para suavizar imágenes, el Filtro de mediana que sirve para mantener todos los píxeles de una vecindad dentro del mismo rango de valores de intensidad como lo vimos en el ejemplo donde el valor más alto eliminado también esto se aplica claro para valores muy pequeños de intensidad. también se aplicó el ruido sal y pimienta, un tipo de ruido que nos puede alterar valores de una imagen durante el procesamiento de esta y como no se mencionaba cambia drásticamente valores de intensidad en un píxel, con valores muy altos (sal) y valores muy bajos (pimienta), afortunadamente el Filtro anteriormente mencionado nos suaviza este inconveniente.

Hay que sacar también que el Filtro utilizado en esta práctica sigue siendo un Filtro espacial ya que el valor del píxel a modificar depende de sus vecinos.

BIBLIOGRAFIA

- [1] J. Evans. "Actividad 2: Ruido Sal y Pimienta". Ramón González. <http://ramon-gzz.blogspot.com/2013/02/actividad-2-ruido-sal-y-pimienta.html> (accedido el 24 de septiembre de 2022).
- [2] W. Gómez. "Análisis de Imágenes Digitales". cinvestav. <https://www.tamps.cinvestav.mx/~wgomez/diapositivas/AID/Clase04.pdf> (accedido el 24 de septiembre de 2022).
- [3] R. Ramírez. "Imagen ruido ruido de sal y pimienta ruido gaussiano (para agregar) - programador clic". programador clic. <https://programmerclick.com/article/9745807835/> (accedido el 24 de septiembre de 2022).
- [4] "Filtrado de mediana de 2D - MATLAB medfilt2- MathWorks América Latina". MathWorks - Creadores de MATLAB y Simulink - MATLAB y Simulink - MATLAB & Simulink. <https://la.mathworks.com/help/images/ref/medfilt2.html> (accedido el 24 de septiembre de 2022).

Asignatura: Procesamiento de imágenes

Profesor: D. Sc. Gerardo García Gil

Alumno: Carlo Pinedo Suarez

Registro: 19310163 **Ciclo:** 2022-B

Ingeniería en Desarrollo de Software

Centro de Enseñanza Técnica Industrial (CETI)

INTRODUCCIÓN

El análisis y procesamiento de imágenes se realiza a través de computadoras, debido a la complejidad y el número de cálculos necesarios para realizarlo. Es por esto que, si bien la formulación matemática necesaria para su realización data de varios siglos atrás, la posibilidad real de utilizarla de forma cotidiana en la práctica clínica ha sido posible recién en las últimas décadas, gracias al avance en las tecnologías del hardware.

La proliferación de nuevos equipamientos con capacidad para realizar millones de operaciones por segundo y su extensión a la vida cotidiana y a todo tipo de usuario, ha hecho posible que el análisis y procesamiento de imágenes digitales se constituya en un gran campo de estudio. En la actualidad, esta tecnología se encuentra incorporada incluso en todo tipo de equipamiento doméstico, como cámaras digitales, scaners y teléfonos celulares, entre otros.

En términos históricos, la utilización de imágenes radiográficas para diagnóstico clínico data prácticamente desde el descubrimiento de los rayos X en 1895 (Röentgen). Incluso, las imágenes funcionales a partir de la emisión de fotones (rayos γ) por parte de radionucleidos ya cuenta con más de 90 años de antigüedad (Hevesy & Seaborg, 1924). Sin embargo, las imágenes eran adquiridas sobre films radiográficos o directamente *in vivo*, por lo que su correcto procesamiento no ha explotado su real potencialidad sino hasta la incorporación de la tecnología que permitió digitalizarlas.

El motivo principal de esta “aparición tardía” del procesamiento de imágenes ha sido entonces, debido a los requerimientos de hardware tanto para el procesamiento de las mismas como para la representación de estas en sistemas gráficos de alta performance. Paralelamente a este desarrollo, la formulación de algoritmos para el procesamiento ha seguido los avances tecnológicos logrando un alto

grado de sofisticación y manipulación de imágenes en tiempo casi real.

El procesamiento digital de imágenes es el conjunto de técnicas que se aplican a las imágenes digitales con el objetivo de mejorar la calidad o facilitar la búsqueda de información.

El proceso de filtrado es el conjunto de técnicas englobadas dentro del preprocesamiento de imágenes cuyo objetivo fundamental es obtener, a partir de una imagen origen, otra final cuyo resultado sea más adecuado para una aplicación específica mejorando ciertas características de la misma que posibilite efectuar operaciones del procesado sobre ella.

El filtro de mediana es una técnica no lineal de mejora de la señal que en la teoría de Procesamiento Digital de Imágenes (PDI) encuentra su aplicación en la mejora de la imagen. Esta técnica se utiliza comúnmente para aplicaciones tales como el suavizado de imágenes y la eliminación de ruidos, presentando la ventaja con respecto a otros operadores de que mantiene la información de borde. Aparte de esta característica, el filtro de mediana se adapta muy bien a cierto tipo de ruido: el ruido impulsivo o de sal y pimienta.

Dada su utilidad y sencillez, este filtro es uno de los más utilizados en PDI para la reducción de ruido coloreado. Sin embargo, el mayor inconveniente es que es un proceso lento en comparación con filtros espaciales clásicos.

DESARROLLO

Simulink

Simulink es un entorno de diagramas de bloque que se utiliza para diseñar sistemas con modelos multidominio, simular antes de implementar en hardware y desplegar sin necesidad de escribir código.

Para esta práctica será necesario hacer el uso de la cámara para grabar video ya que se pretende realizar el histograma simple y lineal de este, para esto comenzaremos agregando el bloque que realizará tal tarea y luego convertiremos la imagen rgb a un formato donde podamos medir la intensidad luminosa que es la escala de grises

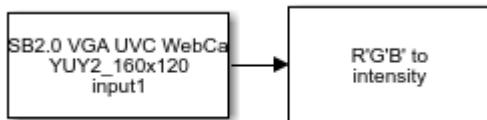


Figura 1. Bloques para obtener video de la cámara.

Para ver el video obtenido se usa lo siguiente:

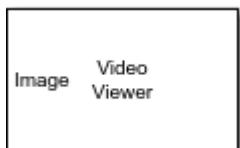


Figura 2. Bloque que mostrara el resultado obtenido.

Mas adelante mostraremos el diagrama completo a bloques utilizado, pero por lo pronto hasta aquí será suficiente para la explicación del desarrollo.

Filtro de la mediana

Este tipo de Filtro suele ser utilizado para eliminar ruido en la imagen.

El funcionamiento es el siguiente, Se visita cada píxel 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 seleccionando el que queda en medio.

Tiene la desventaja de ser un Filtro no lineal pero la ventaja de dar muy buenos resultados en el Filtro de sal y pimienta

Un filtro de mediana es más eficaz que la convolución cuando el objetivo es reducir el ruido y mantener los bordes al mismo tiempo.

La idea principal del filtro de mediana es reemplazar cada entrada con el valor mediano de su vecindario.

Para la práctica usaremos 9 píxeles así que veremos un ejemplo similar, aquí tenemos una matriz donde vemos que el centro tiene un valor de 111:

5	6	7
6	111	8
7	8	9

El Filtro de mediana tomará todos estos valores de la vecindad y los ordenará.

5	6	6	7	7	8	8	9	111
---	---	---	---	---	---	---	---	-----

Una vez ordenados podemos ver que el valor central es el 7, este es el valor que reemplazará al pixel.

5	6	7
6	7	8
7	8	9

Características y usos

El filtro de mediana permite eliminar los valores atípicos sin limitarse a realizar un cálculo promedio que tenderá a contaminar los valores vecinos con este valor atípico y difuminar la imagen.

El filtro de mediana respeta el contraste de la imagen (si multiplicamos todos los valores por una constante positiva, el orden de los valores no cambia) y el brillo de la imagen (añadir una constante tampoco modifica el orden).

En áreas donde la intensidad es monótona (solo aumenta o solo disminuye), el filtro deja la imagen sin cambios. Respeta los contornos y elimina los valores extremos.

Ruido Sal y Pimienta

El ruido de sal y pimienta también se conoce como ruido de impulso bipolar. La característica de este ruido es que el valor de gris del pixel de ruido es significativamente diferente del pixel vecino, y el valor de gris de los pixeles restantes permanece sin cambios. Pixeles brillantes u demasiado oscuros. El ruido de sal y pimienta afecta seriamente la calidad visual de la imagen y causa dificultades en la detección de bordes, la textura o la extracción de puntos de la imagen.

Este tipo de ruido suele producirse cuando la señal de la imagen es afectada por intensas y repentinas perturbaciones o impulsos.

Una forma efectiva para la reducción de este tipo de ruido es mediante el uso de filtros de mediana o filtros morfológicos. A diferencia de los filtros basados en la media empleados para la reducción de ruido Gaussiano, la salida de los filtros medianos utiliza el valor de la mediana de la vecindad de cada pixel, por lo que este tipo de ruido puede ser rápida y eficientemente eliminado.

Configuración simulink

Para poder realizar esta práctica hay que realizar algunas configuraciones empezando por la cámara la siguiente imagen contiene los parámetros requeridos para esta, hay que destacar que estos datos son hechos en el primer bloque mostrado en el diagrama final del anexo, llamado from video device.

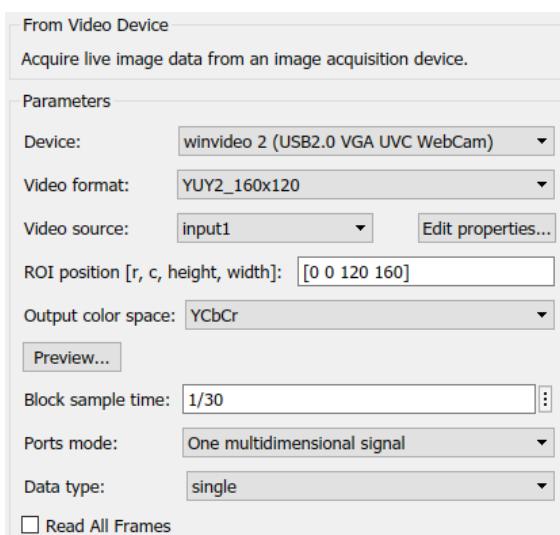


Figura 3. Configuración del bloque de la cámara.

Después hay que configurar el bloque que convertirá la imagen recibida a escala de grises.

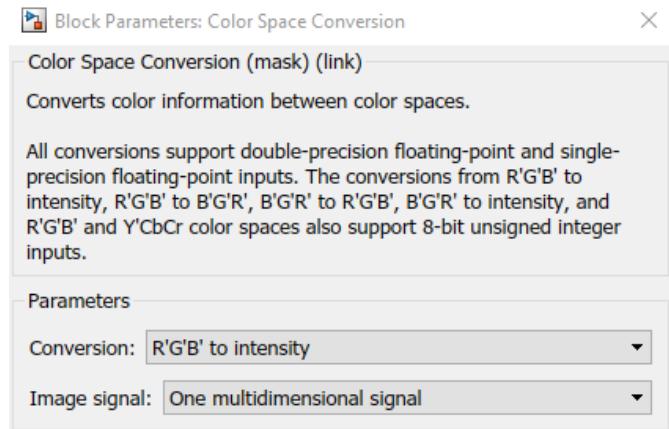


Figura 4. Configuración del bloque para escala de grises.

Después sigue agregar el bloque que nos permite convertir el formato de la imagen a double, recordemos que está en uint8.

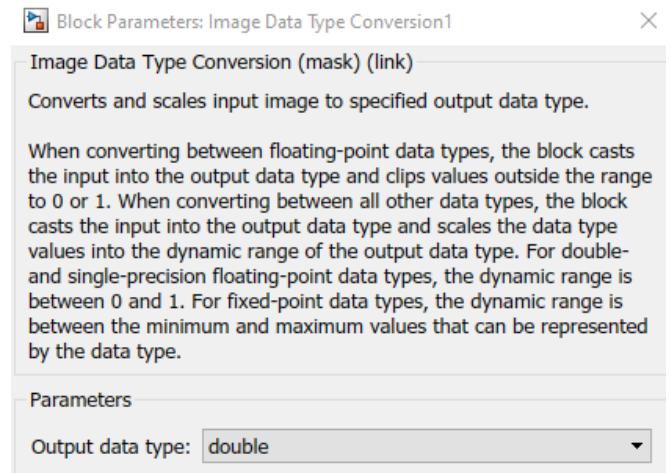


Figura 5. Configuración para cambiar el tipo de la imagen.

Y por último hay que agregar las Funciones que dan nombre a esta práctica empezando por el ruido sal y pimienta el código implementado se verá a continuación.

El código implementado es el siguiente:

```

function ImB = sal_pimienta(ImA)
ImB=ImA;

[M,N,Dim]=size(ImB);
cantPuntos=round((M*N)*0.01);
for i=0:cantPuntos
x=round((rand()*(M-1))+2);
y=round((rand()*(N-1))+2);
if x>=M
x=M-1;
end
if y>=N
y=N-1;
end
ImB(x,y)=255;
end
for i=0:cantPuntos
x=round((rand()*(M-1))+2);
y=round((rand()*(N-1))+2);
if x>=M
x=M-1;
end
if y>=N
y=N-1;
end
ImB(x,y)=0;
end

```

el código para implementar el filtro de mediana es el siguiente:

```

function ImR = filtro_mediana(ImB)
ImR=ImB;

[M,N,Dim]=size(ImR);
vecinos=ones(1,9);
for i=2:M-1
for j=2:N-1
contVec=1;

for r=-1:1
for c=-1:1

vecinos(contVec)=ImB(i+r,j+c);
contVec=contVec+1;
end
end

vecOrd=sort(vecinos);
ImR(i,j)=vecOrd(5);
end
end

```

Finalmente, ya sólo queda agregar los bloques con los que podremos ver el vídeo obtenido la salida, los llamados video viewer.

RESULTADOS

Como podemos ver en los resultados el ruido sal y pimienta fue implementado de forma correcta ya que de forma aleatoria se pueden ver puntos negros y blancos distribuidos en la imagen.

Imagen con ruido sal y pimienta

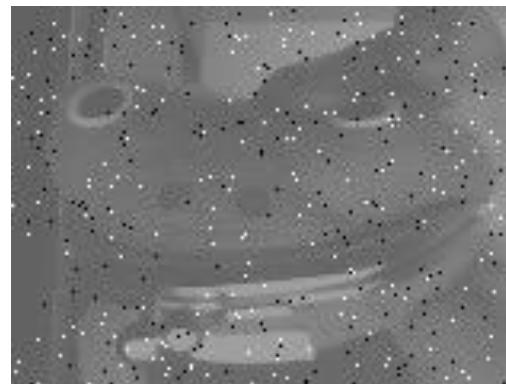


Figura 6. Imagen con ruido sal y pimienta.

Una vez que se implementó el ruido sal y pimienta se aplicó el Filtro de mediana para poder atenuar esos abruptos cambios de intensidad y cómo podemos ver en la siguiente imagen el filtro funcionó de forma correcta.

Filtro de mediana con sal y pimienta



Figura 7. Imagen procesada con Filtro de mediana.

CONCLUSIONES

En ese trabajo se usó un tipo de Filtro para suavizar imágenes, el Filtro de mediana que sirve para mantener todos los píxeles de una vecindad dentro del mismo rango de valores de intensidad como lo vimos en el ejemplo donde el valor más alto eliminado también esto se aplica claro para valores muy pequeños de intensidad. también se aplicó el ruido sal y pimienta, un tipo de ruido que nos puede alterar valores de una imagen durante el procesamiento de esta y como no se mencionaba cambia drásticamente valores de intensidad en un píxel, con valores muy altos (sal) y valores muy bajos (pimienta), afortunadamente el Filtro anteriormente mencionado nos suaviza este inconveniente.

Hay que sacar también que el Filtro utilizado en esta práctica sigue siendo un Filtro espacial ya que el valor del píxel a modificar depende de sus vecinos.

BIBLIOGRAFIA

- [1] J. Evans. "Actividad 2: Ruido Sal y Pimienta". Ramón González. <http://ramon-gzz.blogspot.com/2013/02/actividad-2-ruido>

Anexos

Diagrama a bloques completo:

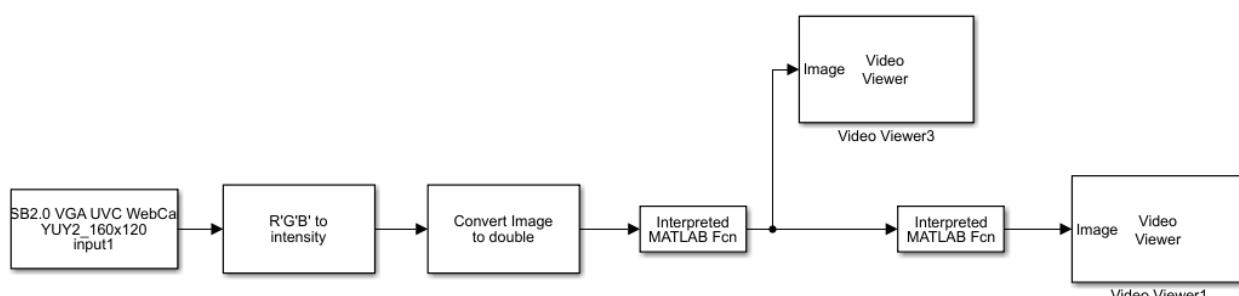


Figura 8. Diagrama bloques en simulink para aplicar ruido sal y pimienta y Filtro de mediana.

sal-y-pimienta.html (accedido el 24 de septiembre de 2022).

- [2] W. Gómez. "Análisis de Imágenes Digitales". cinvestav. <https://www.tamps.cinvestav.mx/~wgomez/diapositivas/AID/Clase04.pdf> (accedido el 24 de septiembre de 2022).
- [3] R. Ramírez. "Imagen ruido ruido de sal y pimienta ruido gaussiano (para agregar) - programador clic". programador clic. <https://programmerclick.com/article/9745807835/> (accedido el 24 de septiembre de 2022).
- [4] "Filtrado de mediana de 2D - MATLAB medfilt2-MathWorks América Latina". MathWorks - Creadores de MATLAB y Simulink - MATLAB y Simulink - MATLAB & Simulink. <https://la.mathworks.com/help/images/ref/medfilt2.html> (accedido el 24 de septiembre de 2022).

Asignatura: Procesamiento de imágenes**Profesor:** D. Sc. Gerardo García Gil**Alumno:** Carlo Pinedo Suarez**Registro:** 19310163 **Ciclo:** 2022-B**Ingeniería en Desarrollo de Software****Centro de Enseñanza Técnica Industrial (CETI)**

INTRODUCCIÓN

El análisis y procesamiento de imágenes se realiza a través de computadoras, debido a la complejidad y el número de cálculos necesarios para realizarlo. Es por esto que, si bien la formulación matemática necesaria para su realización data de varios siglos atrás, la posibilidad real de utilizarla de forma cotidiana en la práctica clínica ha sido posible recién en las últimas décadas, gracias al avance en las tecnologías del hardware.

La detección de movimiento es el proceso de detectar un cambio en la posición de un objeto con respecto a su entorno, o los cambios en el entorno en relación con un objeto. La detección de movimiento se puede lograr por métodos tanto mecánicos y electrónicos. Cuando la detección de movimiento se lleva a cabo por organismos naturales, se llama la percepción del movimiento.

Este proyecto presenta un enfoque para una segmentación de movimiento basado en regiones en tiempo real, utilizando un umbral adaptativo en una escena, con el foco en un sistema de videovigilancia. Para indicar las regiones de máscara de movimiento en una escena, en lugar de determinar el valor de umbral manualmente, utilizamos un método de umbral adaptativo para elegir automáticamente el valor de umbral.

El proceso de filtrado es el conjunto de técnicas englobadas dentro del preprocesamiento de imágenes cuyo objetivo fundamental es obtener, a partir de una imagen origen, otra final cuyo resultado sea más adecuado para una aplicación específica mejorando ciertas características de la misma que posibilite efectuar operaciones del procesado sobre ella.

La detección de cambios tiene como objetivo identificar diferencias en el estado de un objeto, región o fenómeno mediante su observación en diferentes momentos temporales.

Podemos tener la errónea intuición de que la detección de imágenes sea una tarea sencilla, pero veremos que realmente no lo es y de hecho es un gran problema para resolver. Nosotros los humanos podemos ver una foto y reconocer inmediatamente cualquier objeto que contenga de un vistazo rápido, si hay objetos pequeños o grandes, si la foto es oscura o hasta algo borrosa. Imaginemos un niño escondido detrás de un árbol donde apenas sobresale un poco su cabeza o un pie.

Al analizar el problema, Este sugiere crear un sistema automatizado en tiempo real en que se pueda obtener una secuencia de imágenes para luego procesarla con una aplicación mediante la programación orientada objeto; y por qué Orientada objeto, dado a su proximidad a las entidades del mundo real que surgen del modelado, mejora enormemente la captura y validación de requisitos del problema mismo.

La umbralización adaptativa o variable nos permite establecer un valor de umbral que se calcula o varía en base a las características locales del entorno que se evalúa, esto nos permitirá segmentar imágenes que contengan fondos con distintos niveles de grises o iluminación no uniforme.

DESARROLLO

Simulink

Simulink es un entorno de diagramas de bloque que se utiliza para diseñar sistemas con modelos multidominio, simular antes de implementar en hardware y desplegar sin necesidad de escribir código.

Para esta práctica será necesario hacer el uso de la cámara para grabar video ya que se pretende realizar el histograma simple y lineal de este, para esto comenzaremos agregando el bloque que realizara tal tarea y luego convertiremos la imagen rgb a un formato donde podamos medir la intensidad luminosa que es la escala de grises

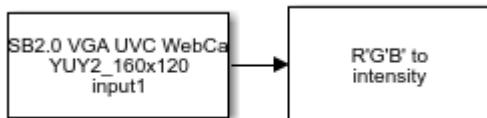


Figura 1. Bloques para obtener video de la cámara.

Para ver el video obtenido se usa lo siguiente:

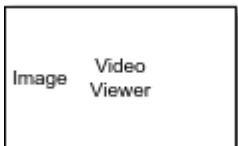


Figura 2. Bloque que mostrara el resultado obtenido.

Mas adelante mostraremos el diagrama completo a bloques utilizado, pero por lo pronto hasta aquí será suficiente para la explicación del desarrollo.

Detección de cambio de posición en imágenes

Para segmentar la imagen, Supongamos que tenemos 2 imágenes, las imágenes son una subsecuencia lineal con cierto retraso t entre ellos. Si queremos comparar cada pixel de las 2 imágenes y nos damos cuenta de que todos son iguales, podemos concluir las 2 imágenes son idénticas. Pero si no lo hacen, podríamos decir que hay algo sucedió durante el tiempo de retardo t . Alguien podría colocar un objeto delante de la cámara, por lo que sería automático el reconocimiento de aquel objeto. Y sí, esta es la idea que vamos a utilizar para la detección de movimiento.

El método que he descrito, es también llamado "diferencial de imágenes". Es el resultado de restar 2 imágenes.

$$f_{dif}(x, y) = f_1(x, y) - f_2(x, y)$$

Figura 3. Ecuación para el diferencial de imágenes.

En nuestra practica vamos a utilizar una secuencia de imágenes, por lo que iremos tomando 2 imágenes consecutivas y las procesaremos, las llamaremos la primera imagen anterior y la segunda, del frame actual. Primero restaremos las imagen actual menos la anterior. Después de esto se obtiene una imagen binaria.

$$\delta I_{fin} = I_t - I_{t-1}$$

Figura 4. Ecuación que describe la resta entre los frames.

Configuración simulink

Para poder realizar esta práctica hay que realizar algunas configuraciones empezando por la cámara la siguiente imagen contiene los parámetros requeridos para esta, hay que destacar que estos datos son hechos en el primer bloque mostrado en el diagrama final del anexo, llamado from video device.

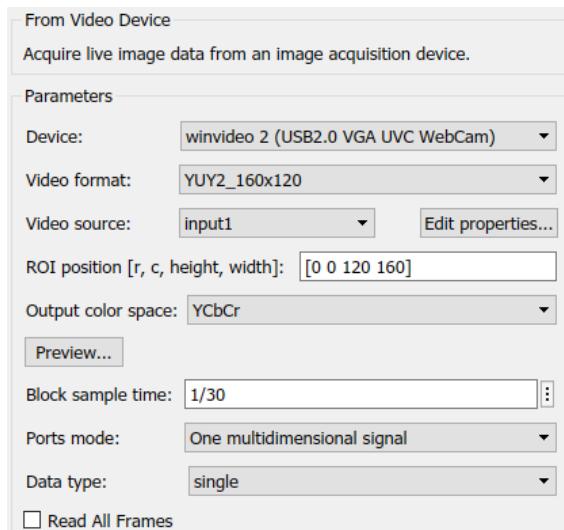


Figura 5. Configuración del bloque de la cámara.

Después hay que configurar el bloque que convertirá la imagen recibida a escala de grises.

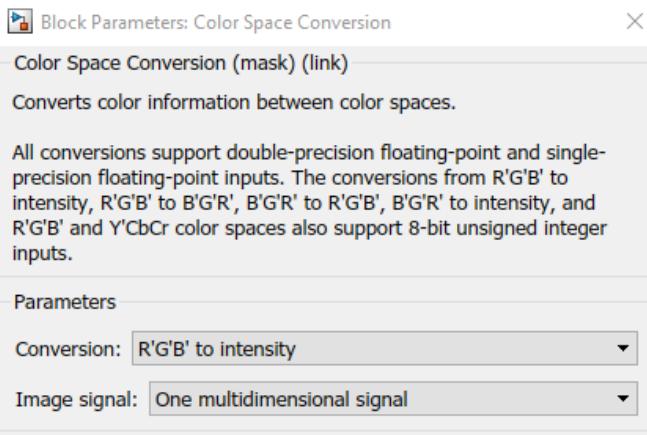


Figura 6. Configuración del bloque para escala de grises.

Después sigue agregar el bloque que nos permite convertir el formato de la imagen a double, recordemos que está en uint8.

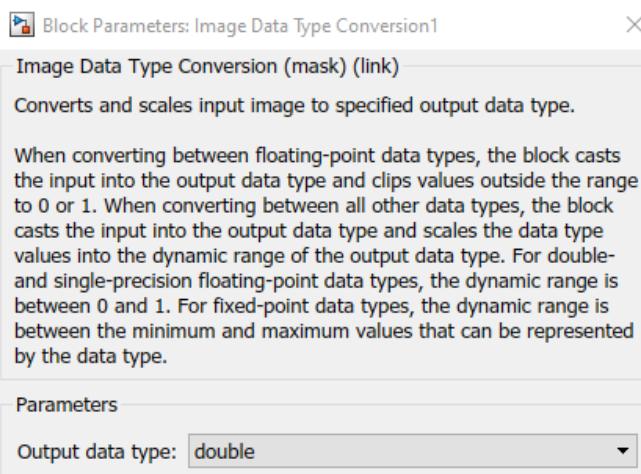


Figura 7. Configuración para cambiar el tipo de la imagen.

En la realización de esta práctica tenemos 2 opciones una vez que se le ha cambiado el formato al vídeo o imagen podemos enviar el video a una función en matlab escrita por nosotros donde simplemente se reste una imagen por otra como lo vemos en la siguiente imagen:

```
function resultado = restaImagenes(x,y)
%Aqui hacemos la resta entre imagenes
resultado=y-x;
```

O bien podemos usar una función predeterminada de simulink donde el resultado es básicamente el mismo sólo que utilizando el siguiente bloque donde de igual manera se hace una resta.

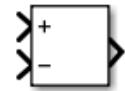


Figura 8. Bloque de resta.

Finalmente, ya sólo queda agregar los bloques con los que podremos ver el vídeo obtenido la salida, los llamados video viewer.

RESULTADOS

Imagen a escala de Grises primer frame



Figura 9. Imagen a escala de grises primer frame.

Imagen a escala de Grises segundo frame



Figura 10. Imagen a escala de grises segundo frame.

La idea al realizar la resta es que cuando haya un movimiento se obtenga un valor de intensidad. como vemos en el resultado de la siguiente imagen. en la teoría claro si no hay un movimiento o algún objeto dentro de la imagen que se mueva no habrá una diferencia entre los pixeles entonces el resultado será cero y por tanto la imagen o en ese punto de la imagen debería verse el color negro, caso contrario cuando se detecta movimiento el color debería inversa en una intensidad de gris.

Imagen resultante de la diferencia



Figura 11. Imagen con movimiento detectado.

CONCLUSIONES

Como podemos ver en los resultados el desarrollo de esta práctica se comprobó con éxito. La idea de manejar niveles de intensidad en escala de grises para comprobar el movimiento en un vídeo es bastante interesante y puede usarse en diversos proyectos que requieran la detección algún simple cambio entre frames. Claro que las imágenes obtenidas con un Filtro adecuado pueden mejorarse.

Básicamente lo que se realizó en esta práctica fue realizar una simple operación de resta entre pixeles donde el pixel actual fue comparado con el píxel del frame anterior eso con el fin de obtener una diferencia de intensidad, al haber un cambio entre una imagen y otra podemos notar un color dentro de la escala de Grises, en cambio si no hubo un movimiento la resta debería dar un valor de cero y esa parte en la imagen como se pudo observar permanece en color negro.

BIBLIOGRAFIA

- [1] Cubillos M., F. (2013). Apuntes introducción a Simulink. LabControl. Recuperado 29 de agosto de 2022, de https://www.labcontrol.cl/sites/labcontrol/ies/apuntes_simulink_2013_0.pdf.
- [2] W. Gómez. "Análisis de Imágenes Digitales". cinvestav. <https://www.tamps.cinvestav.mx/~wgomez/diapositivas/AID/Clase04.pdf> (accedido el 24 de septiembre de 2022).
- [3] Cuevas, O. (2013). Motion Segmentation and Tracking | Home Page. Departamento de Electronica. http://profesores.elo.utfsm.cl/~agv/elo329/1s13/projects/reports/Ramirez_Fernandez/index.html
- [4] "Filtrado de mediana de 2D - MATLAB medfilt2- MathWorks América Latina". MathWorks - Creadores de MATLAB y Simulink - MATLAB y Simulink - MATLAB & Simulink. <https://la.mathworks.com/help/images/ref/medfilt2.html> (accedido el 24 de septiembre de 2022).

Anexos

Diagrama a bloques completo:

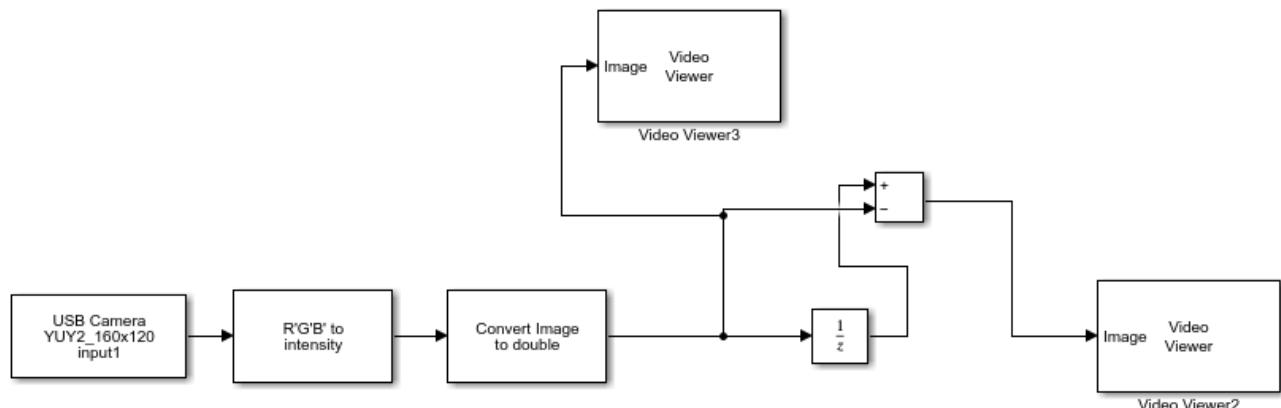


Figura 12. Diagrama bloques en simulink para aplicar el retraso de cuadros.

Asignatura: Procesamiento de imágenes

Profesor: D. Sc. Gerardo García Gil

Alumno: Carlo Pinedo Suarez

Registro: 19310163 **Ciclo:** 2022-B

Ingeniería en Desarrollo de Software

Centro de Enseñanza Técnica Industrial (CETI)

INTRODUCCIÓN

El ser humano percibe al mundo que lo rodea esencialmente a través de su sentido de la vista. Más del 99% de la información que procesa el cerebro humano es captada a través de sus ojos. Es por ello, que la idea de aumentar y mejorar la percepción del mundo a partir de la visión por computador juega un papel importante en el desarrollo científico y tecnológico del quehacer humano.

La visión por computador es una rama de la inteligencia artificial, que tiene como propósito la descripción explícita y significativa de las características de los objetos físicos, captados en una imagen digital. Este objetivo se logra mediante la combinación de dos elementos: el primero es un tratamiento de bajo nivel de una imagen, como es el reconocimiento de bordes y esquinas, remover ruidos, mejorar el contraste, entre otros., y el segundo es un procesamiento de alto nivel, como es el caso de la segmentación de una imagen, reconocimiento de texturas, reconocimiento de objetos, unión de imágenes y detección de movimiento. Un sistema de visión no sólo se enfoca al problema de la extracción de información, sino también el cómo debe ser extraída, representada y empleada.

Uno de los procesos fundamentales en visión por computador, es la detección y extracción de las características de los objetos presentes en la escena. Los bordes y las esquinas son considerados como una de las características básicas. Los bordes y las esquinas sirven como elementos de referencia, llamados puntos de control o puntos prominentes, para procesos de alto nivel. Dichos puntos de control son útiles en los siguientes procesos: el primero es la unión de imágenes; algoritmos de correspondencia entre dos o más imágenes, en segundo tenemos el reconocimiento de objetos; por ejemplo, la detección de las esquinas de un poliedro, en tercero la calibración de cámaras;

encontrar los parámetros de las deformaciones proyectivas y factores de escala provocados por la arquitectura de la cámara, y por cuarto la reconstrucción tridimensional; a través de un conjunto de imágenes, información de contexto y parámetros de calibración de la cámara, se puede lograr una reconstrucción tridimensional de la escena observada.

Debido a que pequeños errores en las mediciones en el plano de la imagen provocan, en general, errores significativos en su interpretación tridimensional, la ubicación precisa de la esquina impacta directamente en la calidad de dicha interpretación, ya que se afectan a los procesos de más alto nivel; por ejemplo a los algoritmos de calibración y reconstrucción tridimensional. En este sentido, la exactitud en la ubicación de las esquinas es un factor fundamental en el desarrollo de un sistema de visión por computador. El detector de esquinas de Harris es un operador para detectar esquinas que es utilizado comúnmente en algoritmos de visión por computadora para extraer estas esquinas e inferir características de una imagen. Fue introducido por Chris Harris y Mike Stephens en 1988 como una improvisación al detector de esquinas de Moravec. Comparado con su antecesor, el detector de esquinas de Harris toma los diferenciales del marcador de esquinas en una cuenta con referencia a la dirección directa, en vez de usar rutas de desplazamiento cada ángulo de 45°, y ha sido probado con más precisión para distinguir entre bordes y esquinas. Esta versión ha sido improvisada y adaptada a muchos algoritmos para preprocesamiento de imágenes para aplicaciones subsecuentes.

DESARROLLO

Esquinas en una imagen

Un punto de esquina puede ser la intersección de dos líneas, o un punto en dos cosas adyacentes con diferentes direcciones principales, es decir, un punto que es particularmente prominente en un determinado aspecto. Hasta ahora no existe una definición matemática clara de los puntos de las esquinas. Los puntos de esquina pueden reducir efectivamente la cantidad de datos de información al tiempo que retienen las características importantes de los gráficos de la imagen, haciendo que el contenido de la información sea alto, mejorando efectivamente la velocidad de cálculo y propiciando la coincidencia confiable de imágenes.

Los puntos de esquina desempeñan un papel muy importante en los campos de la visión por computadora, como la reconstrucción de escenas en 3D, la estimación de movimiento, el seguimiento de objetivos, el reconocimiento de objetivos, el registro de imágenes y la coincidencia.

En la práctica, la mayoría de los métodos denominados detección de esquinas suelen detectar puntos de interés, en lugar de esquinas únicas.

Los algoritmos de detección de esquinas se pueden resumir en tres categorías: detección de esquinas basada en imágenes en escala de grises, detección de esquinas basada en imágenes binarias y detección de esquinas basada en curvas de contorno.

Algoritmo de Harris

El algoritmo desarrollado por Harris y Stephens se basa en la idea de que una esquina es un punto de la imagen donde el valor del gradiente muestra un valor alto en distintas direcciones simultáneamente. Este tiene como característica que es robusto al distinguir entre esquinas y bordes, los cuales muestran un valor alto del gradiente, pero en una sola dirección; además, presenta un alto grado de robustez con respecto a la orientación, por lo que no importa la alineación de la esquina.

Matriz de estructuras

El cálculo del algoritmo de Harris se basa en el desarrollo de la primera derivada parcial en un pixel $I(x,y)$ en dirección horizontal y vertical, tal que:

$$I_x(x,y) = \frac{\partial I(x,y)}{\partial x} \quad y \quad I_y(x,y) = \frac{\partial I(x,y)}{\partial y}$$

Figura 1. Ecuación de la primera derivada parcial de un píxel

Para cada pixel de la imagen (x,y) son calculadas tres diferentes cantidades, que serán denominadas $HE_{11}(x,y)$, $HE_{22}(x,y)$ Y $HE_{12}(x,y)$, donde:

$$HE_{11}(x,y) = I_x^2(x,y)$$

$$HE_{22}(x,y) = I_y^2(x,y)$$

$$HE_{12}(x,y) = I_x(x,y) \cdot I_y(x,y)$$

Figura 2. Coeficientes de la matriz de estructuras.

Estos valores pueden ser interpretados como aproximaciones de los elementos de la matriz de estructuras definida como HE , tal que:

$$HE = \begin{bmatrix} HE_{11} & HE_{12} \\ HE_{21} & HE_{22} \end{bmatrix}$$

Figura 3. Matriz de estructuras.

Donde $HE_{12}=HE_{21}$.

Calculo de los valores y vectores propios

La matriz E debido a su simetría, puede ser diagonalizada de tal forma que:

$$E = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$

Figura 4. Matriz E diagonalizada.

donde λ_1 y λ_2 , son los valores propios de la matriz E, y son calculados de acuerdo con:

$$\lambda_{1,2} = \frac{\text{tr}(E)}{2} \pm \sqrt{\left(\frac{\text{tr}(E)}{2}\right)^2 - \det(E)}$$

Figura 5. Valores propios de la matriz E.

donde $\text{tr}(E)$ implica la traza de la matriz E y $\det(E)$ el determinante de la matriz E. Desarrollando las operaciones de la traza y determinante de la ecuación, se obtiene:

$$\lambda_{1,2} = \frac{1}{2}(A+B \pm \sqrt{A^2 - 2AB + B^2 + 4C^2})$$

Figura 6. Valores propios de la matriz E con traza y determinante desarrollados.

Como nota hay que agregar que cuando $\lambda_1 = \lambda_2 = 0$ entonces no se ha detectado ningún cambio abrupto de intensidad, es decir, no se ha detectado ningún borde. Para el caso donde $\lambda_1 > 0$ y $\lambda_2 = 0$, se considera que ha habido un cambio de escalón de intensidad independientemente de la orientación del borde.

El algoritmo de Harris utiliza esta propiedad como de manera significativa e implementa la función como medida de valor de la esquina:

$$V(x,y) = \det(E) - \alpha(\text{tr}(E))^2$$

Figura 7. Ecuación para determinar valor de la esquina.

Que nos lleva a la siguiente ecuación:

$$V(x,y) = (A \cdot B - C^2) - \alpha(A + B)^2$$

Figura 8. Ecuación para determinar valor de la esquina desarrollada.

Donde alfa controla la sensibilidad del algoritmo, si este valor es grande, es menos sensible a los bordes esquina, caso contrario a cuando es pequeño.

Determinación de los puntos esquina

Un punto de la imagen (X,Y) es considerado como punto esquina potencial si se cumple la condición:

$$V(x,y) > t_h$$

Figura 9. Condición para determinar un punto esquina.

donde t_h es el umbral y normalmente su valor típico se encuentra dentro del intervalo de 900 a 10000, dependiendo del contenido de la imagen. Por lo que, a partir de la aplicación de la ecuación, se tendrá una matriz binaria conteniendo números 1 (verdadero) donde la condición fue cumplida, y 0 (falso) donde no fue válida.

Para impedir la localización de regiones altamente pobladas de puntos esquinas que se calculan debido a una alta sensibilidad del valor α , se seleccionan sólo aquellos pixeles cuyo valor $V(x,y)$ es el más grande dentro de una determinada vecindad.

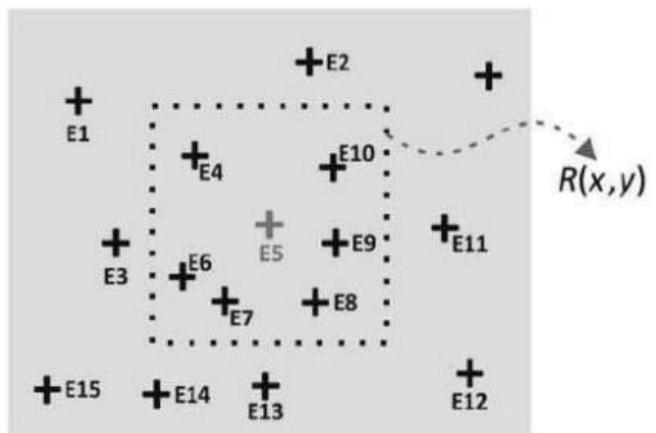


Figura 10. Ejemplo de puntos esquina encontrados en una imagen.

Implementación del código usado para esta práctica

El código implementado es el siguiente:

Para empezar, extraemos la imagen y la convertimos a escala de Grises, obtenemos el tamaño de la imagen, después inicializamos 2 matrices con ceros que nos servirán más adelante e inicializamos la matriz que nos servirá para el pre filtrado.

```

I = imread("ejemplo.png");
Ir=rgb2gray(I);
[m,n, dim]=size(Ir);
U=zeros(size(Ir));
S=zeros(size(Ir));
h=ones(3,3)/9;

Rp=A+B;
Rp1=Rp.*Rp;
Q=((A.*B)-(C.*C))-(alfa*Rp1);

th=1000;
U=Q>th;

```

Ahora usando el siguiente comando imfilter es cómo se aplicará la máscara h previamente inicializada, ahora inicializamos las matrices con las que calcularemos los gradientes Hx y Hy, para después hacer el cálculo.

```

Id=double(Ir);
If=imfilter(Id,h);
Hx=[ -0.5 0 0.5];
Hy=[ -0.5;0;0.5];
Ix=imfilter(If,Hx);
Iy=imfilter(If,Hy);

```

Luego obtenemos los coeficientes de la matriz de estructuras e inicializamos los valores de nuestro Filtro de Gauss.

```

HE11=Ix.*Ix;
HE22=Iy.*Iy;
HE12=Ix.*Iy;

Hg=[0 1 2 1 0; 1 3 5 3 1; 2 5 9 5 2; 1 3 5 3 1; 0 1 2];
Hg=Hg*(1/57);

```

Se hace el filtrado y después le damos un valor a alfa de los anteriormente mencionados en este caso 0.04, recordemos que este valor es pequeño, se tendrá una sensibilidad mayor.

```

A=imfilter(HE11,Hg);
B=imfilter(HE22,Hg);
C=imfilter(HE12,Hg);

alfa=0.04;

```

Ahora se obtiene el valor de la magnitud de la esquina y obtenemos la matriz U.

```

pixel=10;

for r=1:m
    for c=1:n
        if(U(r,c))
            I1=[r-pixel 1];
            I2=[r+pixel m];
            I3=[c-pixel 1];
            I4=[c+pixel n];
            datxi=max(I1);
            datxs=min(I2);
            datyi=max(I3);
            datys=min(I4);
            Bloc=Q(datxi:1:datxs, datyi:1:datys);
            MaxB=max(max(Bloc));
            if(Q(r,c)==MaxB)
                S(r,c)=1;
            end
        end
    end

```

Finalmente marcamos sobre la imagen las esquinas calculadas por el algoritmo de Harris, las esquinas encontradas se verán marcadas con una cruz.

```

figure(3);
imshow(Ir);

hold on;

for r=3:m-1
    for c=3:n-2
        if(S(r,c))
            plot(c,r,'+', 'MarkerSize',8);
        end
    end
end

```

RESULTADOS

Para esta práctica se buscó una imagen que tuviera bastantes esquinas como es la siguiente que contiene bastantes triángulos, esto con el fin de observar mejor los resultados del algoritmo.

Imagen original



Figura 11. Imagen original con muchas figuras y esquinas.

Imagen a escala de grises



Figura 12. Imagen convertida a escala de grises.

Imagen después de usar el algoritmo de Harris

Como podemos ver según la sensibilidad y los parámetros que hemos establecido se encontraron bastantes puntos esquina en toda la imagen, en teoría, al modificar la sensibilidad y el umbral podríamos encontrar más o menos puntos.

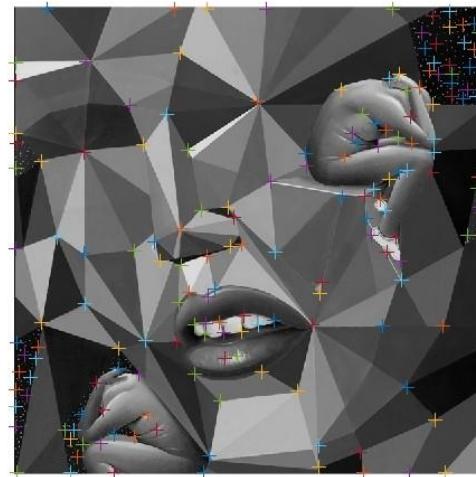


Figura 13. Puntos esquina detectados en la imagen.

CONCLUSIONES

Para esta práctica se implementó el algoritmo de Harris como detector de esquinas un algoritmo bastante útil y que según la teoría nos permite hacer más adelante operaciones muy útiles para el reconocimiento de objetos, por ejemplo, fue bastante sencillo y no hubo algún fallo relevante que mencionar más allá del hecho de que para realizar operaciones elemento a elemento menores de un arreglo o matriz es necesario usar el operador “.”.

también hay que mencionar que para detectar más esquinas o menos es necesario modificar el umbral o la sensibilidad ya que según sabemos son estos los que permiten detectar los puntos esquina de mejor manera, también podría mencionarse que al hacer un mejor filtrado que mejore la calidad de la imagen este algoritmo podría ser más exacto, pero para fines prácticos el algoritmo de gauss aplicado es bastante útil.

BIBLIOGRAFIA

- [1] E. Cuevas, M. Diaz y J. Camarena, Tratamiento de imágenes con MATLAB. México: Alfaomega, 2017.
- [2] Castillo, W. (2022b). Algoritmo de detección de esquinas de Harris - programador clic. programador clic.
<https://programmerclick.com/article/51231377446/>
- [3] O Méndez, M. R. (2022). ALGORTIMO DE HARRIS PARA DETECCIÓN DE ESQUINAS. Academia.edu - Share research.
https://www.academia.edu/44924614/ALGORITIMO_DE_HARRIS_PARA_DETECCIÓN_DE_ESQUINAS

Asignatura: Procesamiento de imágenes

Profesor: D. Sc. Gerardo García Gil

Alumno: Carlo Pinedo Suarez

Registro: 19310163 **Ciclo:** 2022-B

Ingeniería en Desarrollo de Software

Centro de Enseñanza Técnica Industrial (CETI)

INTRODUCCIÓN

La IA o Inteligencia Artificial está cambiando el mundo tal y como lo conocemos, aportando nuevas soluciones cada vez más increíbles. El reconocimiento de imágenes es una de las áreas que más rápidamente está creciendo en los últimos años.

Una aplicación de la inteligencia artificial es la detección de objetos en imágenes, pero aquí no surge la pregunta ¿Qué es el reconocimiento de imágenes?, El reconocimiento de imágenes es un proceso ejecutado por un software de inteligencia artificial capaz de reconocer imágenes utilizando algoritmos matemáticos complejos. La IA es capaz de identificar, analizar y comparar las matrices de bits que componen una imagen digital, con el objetivo de realizar alguna acción a partir de la información obtenida.

Gracias al reconocimiento de imágenes se pueden automatizar ciertas tareas que para el humano supondrían una gran inversión de tiempo y recursos. Es capaz de identificar personas, lugares, objetos y cualquier otro elemento que se encuentre en una imagen. Su número de aplicaciones ha ido creciendo con el tiempo, y se espera que lo siga haciendo en el futuro gracias a la constante evolución de la IA. El reconocimiento de imágenes se apoya también en el Machine Learning y el Deep Learning, dos vertientes de la IA que permiten a las máquinas ir aprendiendo a medida que van analizando las imágenes. Cuanto más imágenes procesan, más afinan sus resultados y se obtienen mejores resultados. El reconocimiento de imágenes utiliza un conjunto de tecnologías y técnicas de aprendizaje automático para crear redes neuronales artificiales. Se forman varias capas de neuronas que pueden comunicarse entre ellas, formando así una estructura que puede analizar la información de cada píxel de una imagen.

Actualmente, las técnicas de aprendizaje computacional y de procesamiento de imagen tienden al análisis del movimiento de los humanos en ambientes no controlados, ambientes en donde no se pueden ejercer acciones de mando en aspectos como la dirección del movimiento de las personas, y en características del medio como el nivel de luz presente al momento de la captura de la imagen que arrojan datos probabilísticos de operación. El usar cámaras web en el conteo de objetos ayuda a poder hacer monitoreo del sistema desde cualquier otra computadora con internet, esto mediante un programa de escritorio remoto que permite operar la computadora donde se encuentra la aplicación. El crear un sistema que analice y obtenga el número de objetos en imágenes en 2-D, y procese imágenes de escenas en 3-D es importante con el fin de determinar el número de objetos de la escena, aunque se encuentren traslapados.

En prácticas anteriores, cuando fue tratado el filtro de la mediana, pudo observarse que éste tiene la capacidad de modificar las estructuras bidimensionales presentes en la imagen. Dentro de estos cambios puede citarse la modificación de las esquinas de estructuras, redondeándolas; o bien, la eliminación de puntos o estructuras delgadas como líneas o pequeños artefactos. Por lo anterior, puede decirse que el filtro de la mediana reacciona selectivamente a la forma de las estructuras locales de la imagen. Sin embargo, la operación de este filtro no puede controlarse, es decir, no puede ser usado para reaccionar con estructuras descritas de acuerdo con su forma o estructura. La idea principal de esta práctica es utilizar algunos filtros morfológicos, los cuales son explicados en la práctica 16, para detectar objetos mediante sus bordes utilizando la herramienta simulink de Matlab.

DESARROLLO

Parte uno para binarizar imagen

Para empezar el desarrollo de esta práctica en la ventana de comandos de Matlab utilizaremos algunas de las Funciones del toolbox image adquisition y ejecutamos el siguiente código:

```
// crea un objeto de video con el controlador de Windows  
>> vid=videoinput('winvideo');  
//permite visualizar una pantalla del frame actual  
>> preview(vid);  
// permite iniciar la captura de las 10 imágenes por cámara.  
>> start(vid);  
//carga las imágenes a la variable data  
>> data = getdata(vid);  
//se selecciona la imagen número 5  
>> imagen=data(:,:,:,:5);  
//se visualiza la imagen  
>> imshow(imagen);  
// de esta manera se llama a las herramientas de imagen  
>> imtool(imagen);  
//Se carga las herramientas de Simulink  
>>simulink
```

Figura 1. Código para obtener límites superiores e inferiores.

Al momento de utilizar la función preview(vid) nos deberá aparecer una ventana de vídeo que permitirá visualizar la imagen actual de nuestra cámara. después de ejecutar los siguientes 3 comandos llegamos a imshow(imagen), donde veremos la imagen 5 mostrar en la figura 2, aquí es importante mencionar que en mi cámara no obtiene un formato RGB la imagen siguiente contiene un formato yuy2, por lo que se tiene que hacer la conversión.

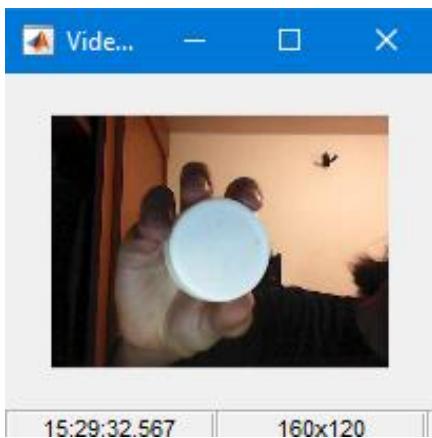


Figura 2. Imagen obtenida por la cámara de vídeo.

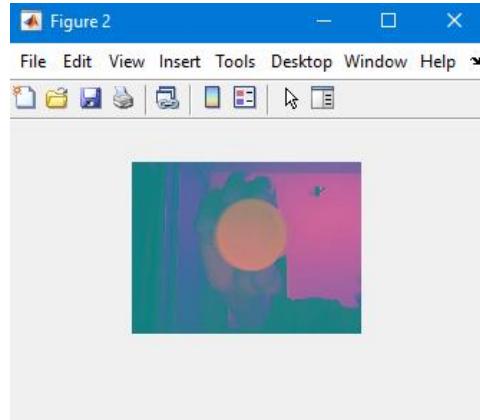


Figura 3. Imagen YUY2 convertida a RGB .

Ahora procedemos a escribir la función imtool(imagen) que nos abrirá la siguiente ventana como lo vemos en la imagen de la figura 4.

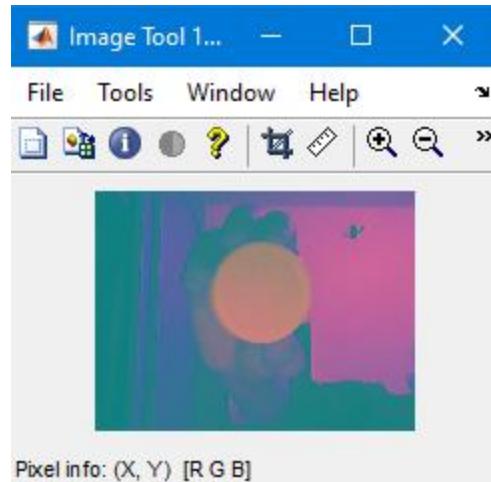


Figura 4. Imagen después de abrir imtool.

después daremos clic en la opción qué dice insertar valores de píxeles que se encuentra en la Barra de herramientas como vemos en la siguiente imagen:



Figura 5. Barra de herramientas de imtool.

Como podemos ver se nos abre una ventana que nos muestra los valores del centro de la cruz de la figura 6.

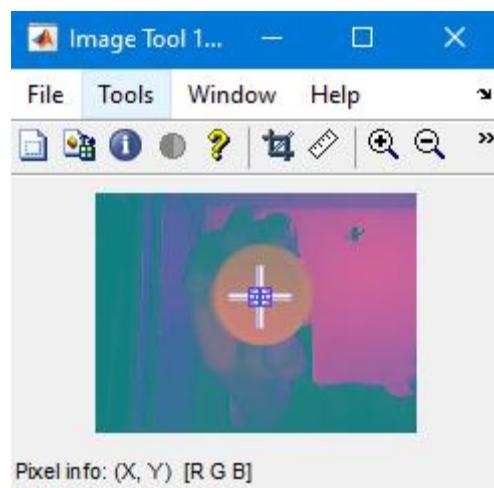


Figura 6. Marcador que nos dará el valor del píxel elegido.

Imagen.... este método es bastante sencillo y consiste básicamente en realizar un recorrido con el puntero alrededor de toda la figura para observar cuáles son los intervalos máximos y mínimos de intensidades RGB. En la siguiente tabla vemos estos valores.

Color	Valores mínimos	Valores máximos
R	69	91
G	148	149
B	101	103

ya que matlab cambia el rango de cero a 255 para intensidades al realizar una operación de resultado tipo double, necesitamos realizar la conversión de estos valores a intervalos de cero y uno. Los resultados los vemos en la siguiente tabla:

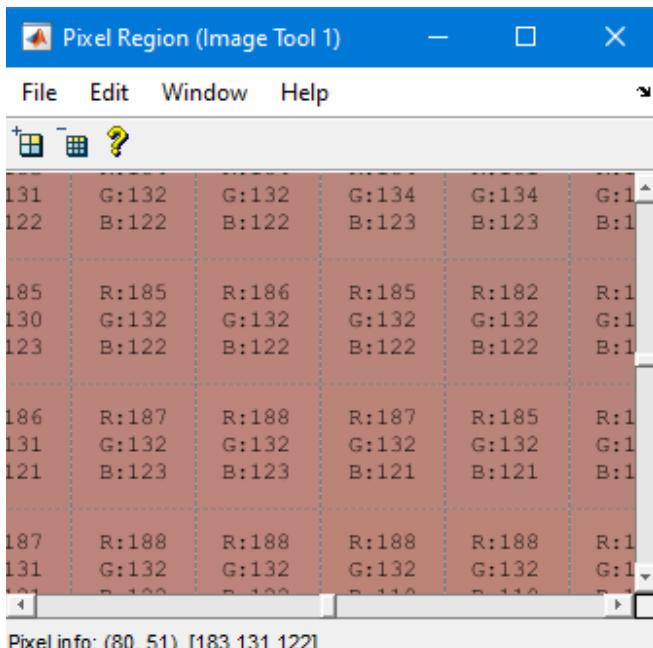


Figura 7. Matriz de valores RGB de la imagen de la figura 6.

Color	Valores mínimos	Valores máximos
R	0.270588	0.356862
G	0.580392	0.584313
B	0.396078	0.403921

Hasta aquí ya le dimos características de color a la figura a filtrar, ahora necesitamos implementar la librería de bloques para el procesamiento de imágenes y el vídeo de simulink.

Para este punto vamos a explicar un poco sobre lo que es simulink y los bloques que vamos a estar utilizando.

Simulink es un entorno de diagramas de bloques que se utiliza para diseñar sistemas con modelos multidominio, simular antes de implementar en hardware y desplegar sin necesidad de escribir código.

El primer bloque que utilizaremos será el llamado `from video device` que obtiene la imagen de la cámara de vídeo.

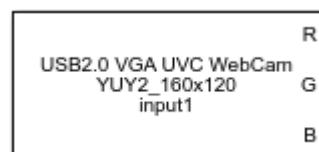


Figura 8. Bloque para obtener el vídeo.

En ella podemos ver distintas intensidades de cada uno de los colores según el modelo RGB. como hemos visto cada uno de sus 3 colores varía de cero a 255 niveles de intensidad, claro con su combinación se puede obtener cualquier otro tipo de color.

Ahora el siguiente paso es identificar el objeto del cual deseamos hacer el filtrado. Para esto debemos realizar un muestreo sobre la figura como lo vemos en la

La configuración para este bloque es la siguiente:

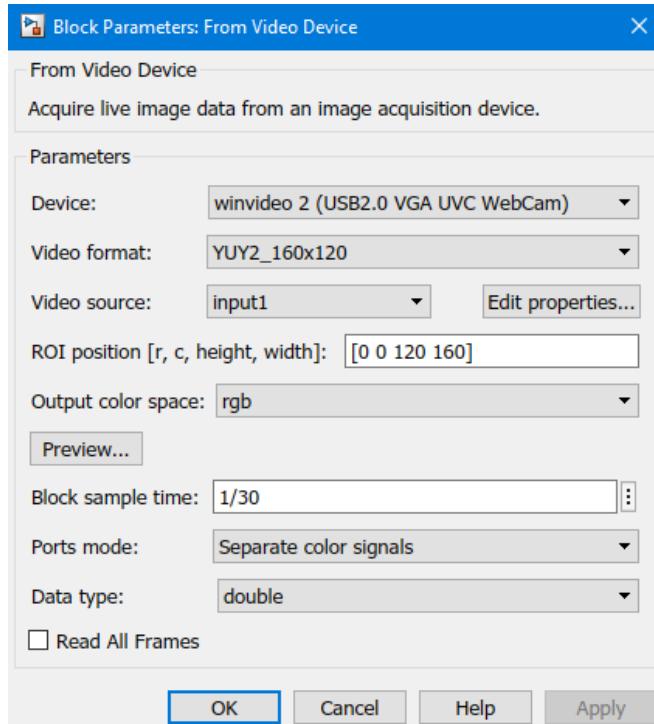


Figura 9. Configuración del bloque para obtener el vídeo.

Ahora procederemos a colocar el bloque llamado compare to constant, la idea con esto es que pongamos 6 bloques que nos servirán para limitar los valores máximos y mínimos previamente obtenidos para en la imagen binaria poder observar en blanco sólo el objeto que queremos detectar, el fondo obviamente será de color oscuro.

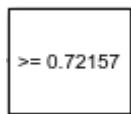


Figura 10. Bloque usado para la comparación de los Datos de entrada.

La configuración para este bloque depende de los valores ya mencionados en la tabla deberemos cambiar el operador y el valor constante.

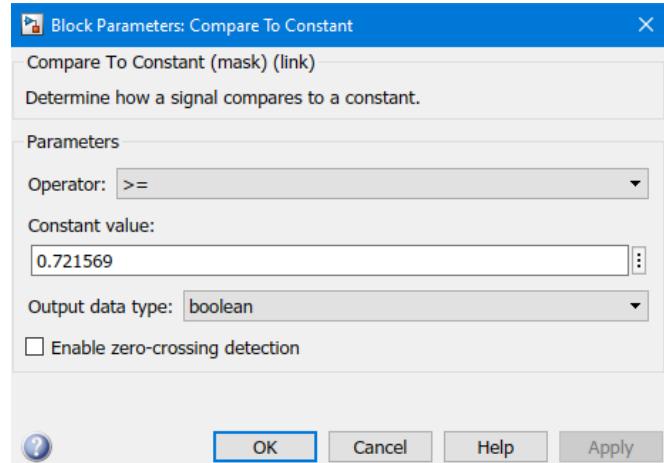


Figura 11. Configuración del bloque de comparación de los Datos de entrada.

Lo que sigue es colocar una compuerta And que como ya mencionamos lo que queremos es limitar los valores que se obtienen entonces únicamente los pixeles que se encuentren dentro del rango establecido nos darán un valor de uno, los no deseados nos darán cero.

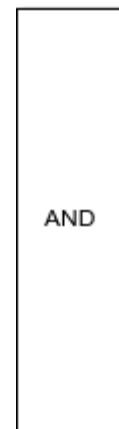


Figura 12. Bloque para la compuerta lógica AND.

Para este bloque el único que cambiar es el número de entradas que serán 6 pero no.

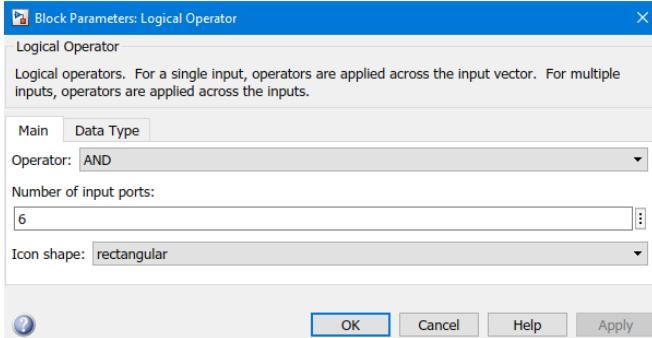


Figura 13. Configuración de la compuerta lógica AND.

Ahora agregamos el bloque para erosionar la imagen de vídeo y otro bloque para dilatarla.

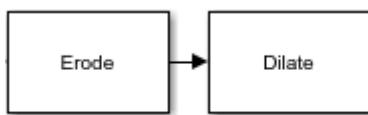


Figura 14. Bloques para la dilatación y erosión de la imagen.

Las respectivas configuraciones son las siguientes:

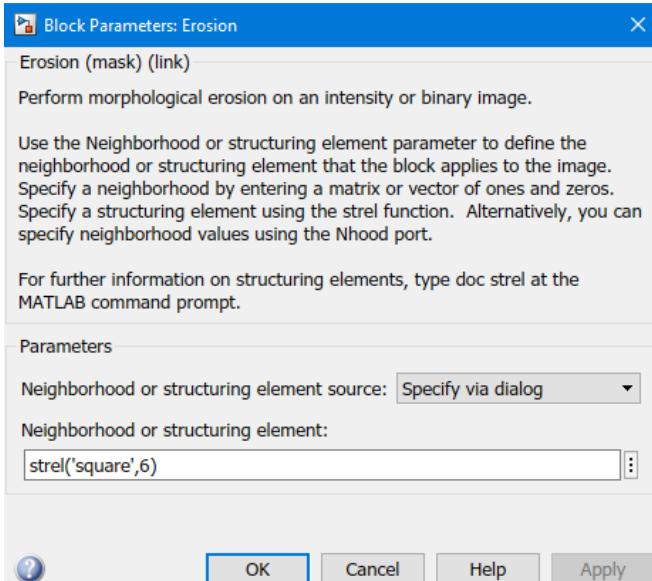


Figura 15. Configuración del bloque para erosión de la imagen.

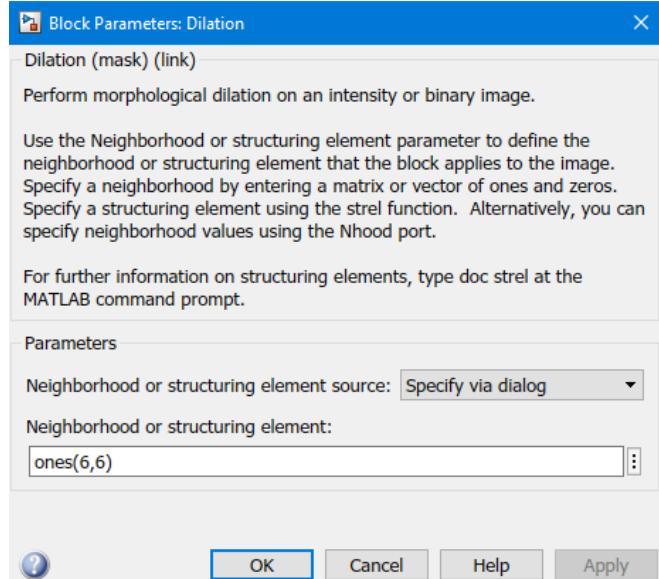


Figura 16. Configuración del bloque para la dilatación de la imagen.

Finalmente agregamos el bloque con el que podremos ver el vídeo obtenido.

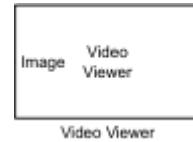


Figura 17. Bloque con el que obtendremos el vídeo procesado.

Una vez corremos el código podremos ver una imagen como la siguiente donde los puntos blancos representan el objeto que queremos detectar y claro como se observa es una imagen binaria.

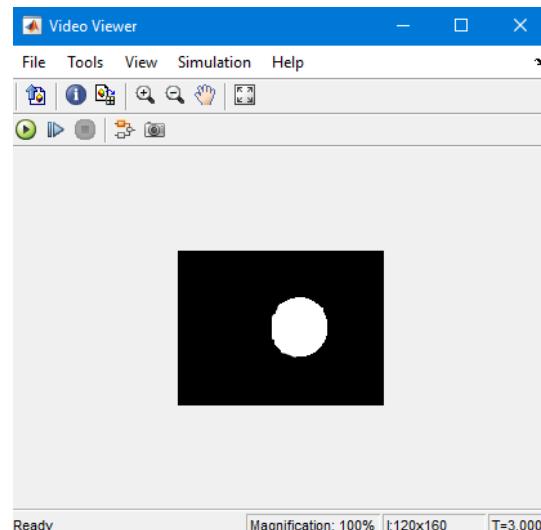


Figura 18. Vídeo procesado para la binarización de la imagen.

El diagrama completo lo encontraremos en los anexos figura 27.

Parte dos para la detección del objeto

Hasta ahora lo que tenemos es la binarización de la imagen, Separando el fondo del objeto que queremos detectar como vemos en la figura.... Lo que sigue es agregar los siguientes bloques.

Para esto empezamos con el bloque de blob análisis, que se ve así:

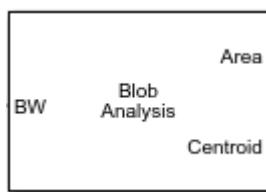


Figura 19. Bloque Bob Analysis.

Las configuraciones de este bloque son las siguientes:

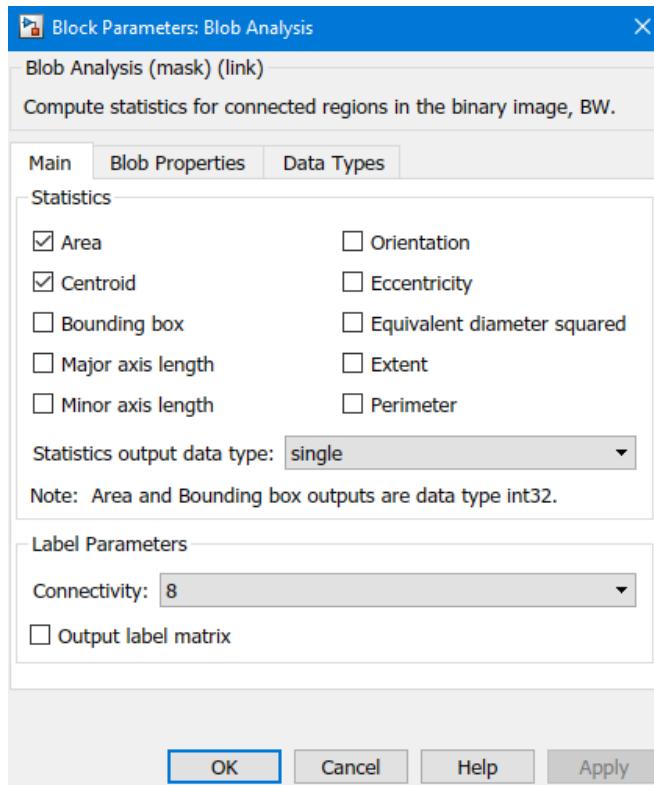


Figura 20. Configuración del bloque Bob Analysis.

Pasamos a agregar ahora el bloque llamado Maximum, el cual nos proporcionará valores máximos de la señal

de entrada. si el modelo se está ejecutando este bloque actualiza sus resultados en todo momento.

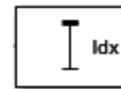


Figura 21. Bloque con el que obtendremos los valores máximos.

Seguimos con el bloque llamado variable selector, este bloque selecciona y ordena las filas o columnas de entrada de acuerdo a un específico vector de índices. El parámetro Selector Mode determina si el bloque usará la misma forma de índices para cada entrada o usará diferentes índices. cuando los índices son datos tipo booleano, el bloque realiza una indexación lógica.

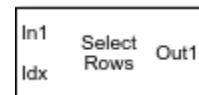


Figura 21. Bloque que ordena las filas o columnas de entrada.

Las configuraciones de este bloque son las siguientes:

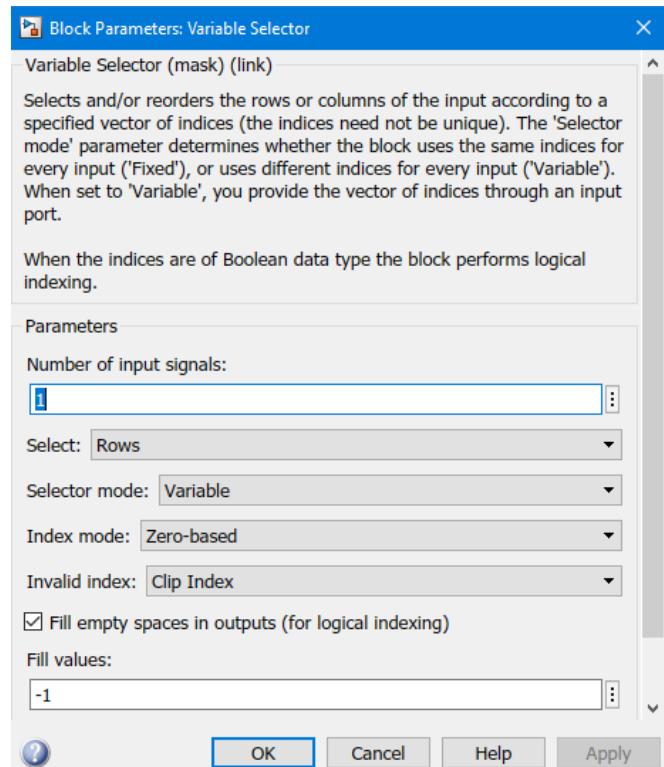


Figura 22. Configuración del bloque que ordena las filas o columnas de entrada.

Finalmente seguimos con el último bloque llamado Draw makers, el cual marca las posiciones sobre imágenes dibujando círculos, signos, estrellas o rectángulos. se utiliza el puerto Pts para especificar una matriz 2 por n, de columnas y filas pares, donde n es el número total de marcas y cada columna fila pares se definen en el centro de la marca.

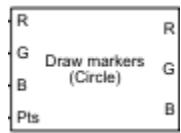


Figura 23. Bloque con el que dibujaremos el círculo que detecta el objeto.

Las configuraciones son las siguientes:

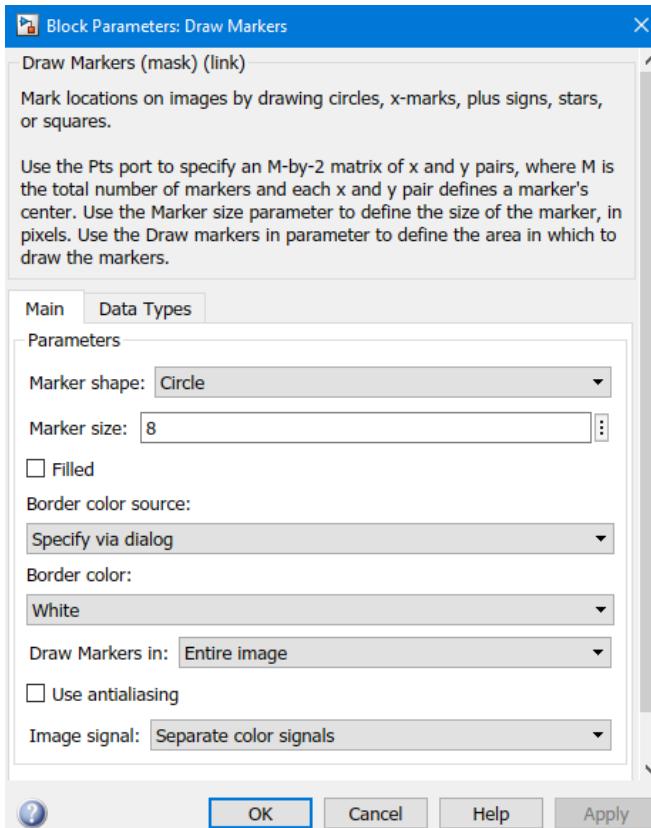


Figura 24. Configuración del bloque con el que dibujaremos el círculo que detecta el objeto.

El diagrama completo lo podemos ver en los anexos en la figura 28.

Resultado de la detección de objetos

Como podemos ver en la siguiente imagen el programa funciona correctamente, se dibuja un círculo negro sobre la tapa que es el objeto el cual deseamos detectar.

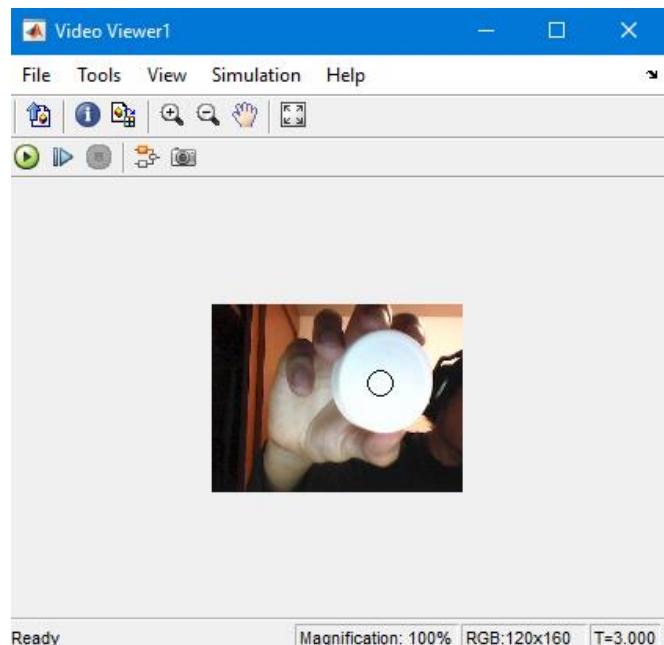


Figura 25. Resultados de la detección del objeto.

Claro aunque se mueva la tapa se realizará un seguimiento a esta, por tanto el círculo dibujado de color negro la seguirá.

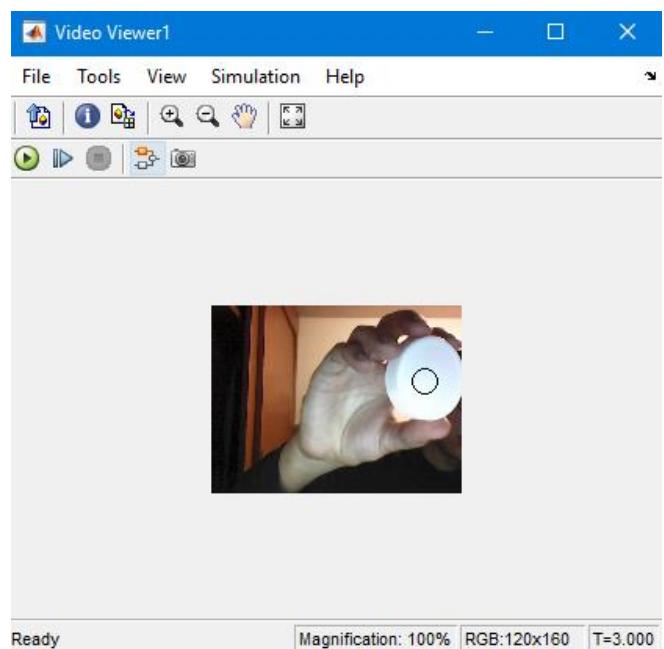


Figura 26. Objeto movido para identificar el seguimiento de la detección.

CONCLUSIONES

Se realizó un programa a bloques en simulink el cual fue capaz de detectar objetos haciendo uso de filtros morfológicos para la detección de bordes como lo son la erosión y la dilatación, la idea de estos filtros como ya vimos es detectar bordes en imágenes binarias para esta práctica, evidentemente este tipo de filtros tiene otras aplicaciones como lo pueden ser en la mejora y tratamiento de una imagen cualquiera.

hay que destacar que para poder tener con mejor claridad la detección es necesario modificar los parámetros de los pixeles detectados en el borde ya que esto modifica la sensibilidad que será capaz de detectar el programa. también algo interesante es que para convertir la imagen de forma binaria hicimos Uso de compuertas lógicas como la and, donde ingresamos los valores de vídeo y sólo se detectaron o pintaron de blanco los valores que deseábamos mediante comparaciones, estas comparaciones fueron elegidas mediante límites que nosotros mismos detectamos al Hacer una ampliación de la imagen y hacer Uso de las herramientas de Matlab.

BIBLIOGRAFIA

- [1] E. Cuevas, M. Diaz y J. Camarena, Tratamiento de imágenes con MATLAB. México: Alfaomega, 2017.
- [2] Cubillos M., F. (2013). Apuntes introducción a Simulink. LabControl. Recuperado 30 de octubre de 2022, de https://www.labcontrol.cl/sites/labcontrol/fies/apuntes_simulink_2013_0.pdf.
- [3] R. Gonzalez y R. Woods, Digital Image Processing, 3a ed. Miami: Pearson, 2017.
- [4] A. Ramírez. "Reporte de Búsqueda, detección y conteo de objetos". cimat. http://www.cimat.mx/~alram/VC/ramirez_segmObjDetecc.pdf (accedido el 30 de octubre de 2022).

Anexos

Diagrama bloques de la primera parte de la práctica:

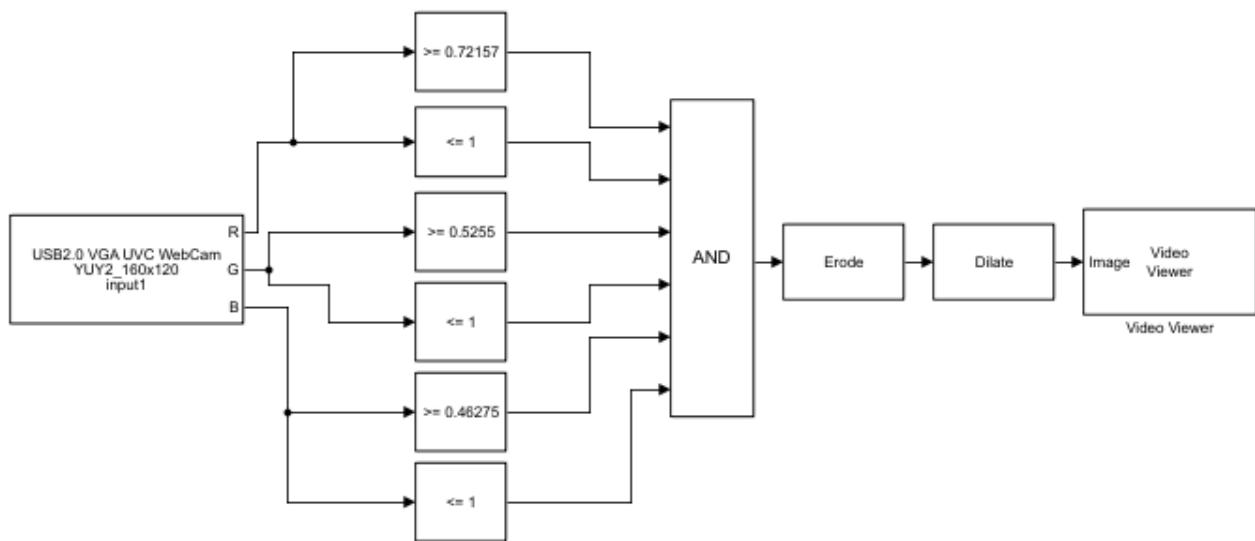


Figura 27. Diagrama bloques usado para la binarización de la imagen.

Diagrama bloques de la segunda parte de la práctica:

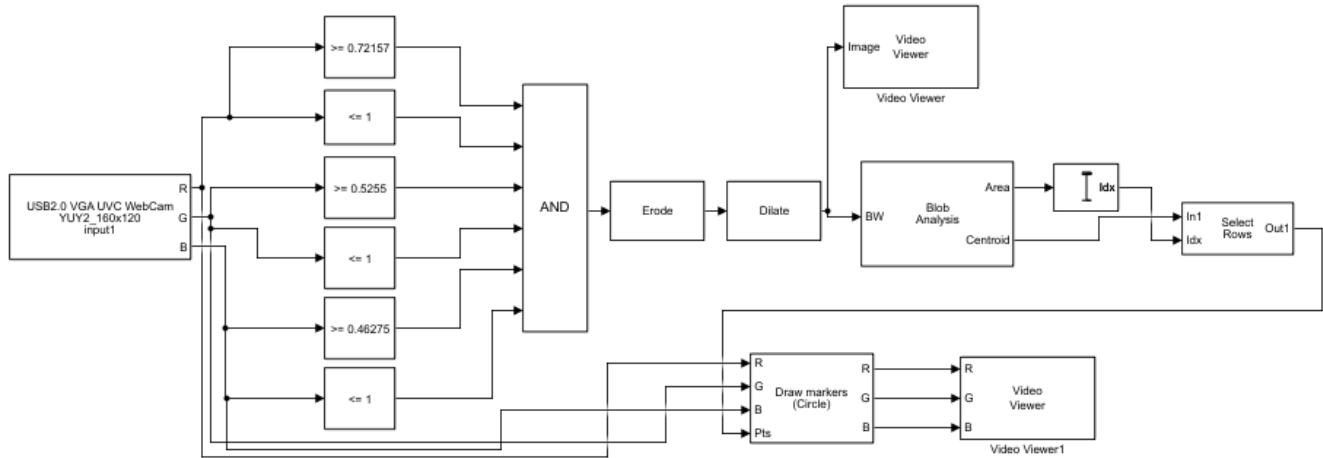


Figura 28. Diagrama o bloques completo para la detección de objetos.

Asignatura: Procesamiento de imágenes

Profesor: D. Sc. Gerardo García Gil

Alumno: Carlo Pinedo Suarez

Registro: 19310163 **Ciclo:** 2022-B

Ingeniería en Desarrollo de Software

Centro de Enseñanza Técnica Industrial (CETI)

INTRODUCCIÓN

La visión es uno de los sentidos de percepción más importantes en los seres humanos, aunque una incapacidad visual no impide el desarrollo de otras actividades mentales. Uno de los principales objetivos de la visión por computadora es poder diferenciar los objetos presentes en una imagen (en este caso, en una imagen digital), de tal manera, que esto logre que la identificación de estos sea una tarea más fácil de realizar. El procesamiento de imágenes se centra en 2 áreas principales, las cuales son: a) Mejora de la calidad de imágenes para la interpretación de los humanos. b) Procesamiento de los datos de una escena para que una computadora tenga una percepción y ejecute operaciones según sea lo extraído de la imagen.

Actualmente, las técnicas de aprendizaje computacional y de procesamiento de imagen tienden al análisis del movimiento de los humanos en ambientes no controlados, ambientes en donde no se pueden ejercer acciones de mando en aspectos como la dirección del movimiento de las personas, y en características del medio como el nivel de luz presente al momento de la captura de la imagen que arrojan datos probabilísticos de operación. El usar cámaras web en el conteo de objetos ayuda a poder hacer monitoreo del sistema desde cualquier otra computadora con internet, esto mediante un programa de escritorio remoto que permite operar la computadora donde se encuentra la aplicación. El crear un sistema que analice y obtenga el número de objetos en imágenes en 2-D, y procese imágenes de escenas en 3-D es importante con el fin de determinar el número de objetos de la escena, aunque se encuentren traslapados.

En prácticas anteriores, cuando fue tratado el filtro de la mediana, pudo observarse que éste tiene la capacidad de modificar las estructuras bidimensionales presentes en la imagen. Dentro de estos cambios puede citarse la modificación de las esquinas de estructuras, redondeándolas; o bien, la eliminación de puntos o estructuras delgadas como líneas o pequeños artefactos. Por lo anterior, puede decirse que el filtro de la mediana reacciona selectivamente a la forma de las estructuras locales de la imagen. Sin embargo, la operación de este filtro no puede controlarse, es decir, no puede ser usado para reaccionar con estructuras descritas de acuerdo con su forma o estructura.

Los filtros morfológicos fueron concebidos para ser usados sobre imágenes binarias, es decir, sobre aquéllas cuyos pixeles sólo tienen dos posibles valores: 1 y 0, blanco negro. Las imágenes binarias se encuentran en un gran número de aplicaciones, especialmente en procesamiento de documentos. En lo sucesivo, definiremos a los pixeles de la estructura como los correspondientes a 1, y los del fondo a 0.

Los filtros morfológicos se basan en operaciones de conjuntos, que ejecutan alteraciones simples a las imágenes en forma. Las dos operaciones morfológicas más comunes son la dilatación y erosión, y de sus combinaciones surgen las operaciones de apertura, cierre y gradiente morfológico. Generalmente, los filtrados morfológicos se aplican conjunto a los filtrados de suavizado, con lo que se llenan espacios de la imagen y se elimina el ruido de esta, manteniendo la seguridad y privacidad de la información.

DESARROLLO

Filtros morfológicos para la detección de bordes

Los filtros morfológicos se especifican mediante la definición de dos aspectos: la operación que desempeñan (erosión o dilatación) y su correspondiente estructura de referencia. El tamaño y la forma de la estructura de referencia son dependientes de la aplicación. En la práctica, son frecuentemente utilizadas las estructuras de referencia mostradas en la figura 1.

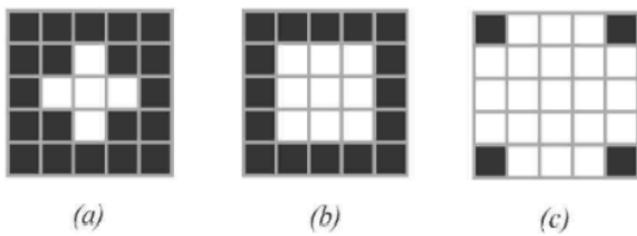


Figura 1. Estructuras de referencia típicas de diferentes tamaños.
(a) Estructura de 4 vecinos, (b) estructura de 8 vecinos y (c) estructura de disco pequeño.

Nosotros para realizar esta práctica usaremos el inciso a de la figura uno.

Para continuar hay que decir que una aplicación típica de las operaciones morfológicas es la extracción de los bordes de los objetos contenidos en una imagen binaria.

Básicamente para la detección de bordes lo que se hace es una operación de erosión o de dilatación. De manera que la ecuación que describe cómo detectar un borde mediante la dilatación es la siguiente:

$$\beta(IM) = (IM \oplus B) - IM$$

Figura 2. Ecuación que describe cómo detectar un borde mediante la dilatación.

Esto significa que la imagen de borde corresponde a la imagen dilatada menos la original.

También hay que señalar que el borde calculado con esta operación corresponde al borde externo de la imagen original. lo que quiere decir que los puntos del borde no son parte de los puntos del objeto original.

Otra manera de mantener algunas de las proporciones del objeto original es lograr determinar el borde

tomando en cuenta la imagen original menos la imagen erosionada como lo describe la siguiente ecuación:

$$\beta(IM) = IM - (IM \ominus B)$$

Figura 3. Ecuación que describe cómo detectar un borde mediante la erosión.

Relleno de imágenes

El relleno de agujeros o el relleno de regiones en el procesamiento de imágenes es una operación morfológica. Además, como su nombre podría sugerir, llena agujeros dentro de formas huecas. Definimos estas formas huecas o agujeros como píxeles de fondo encerrados por un borde de píxeles de primer plano.

Todos los procesos morfológicos se basan en dos operaciones fundamentales, que son la erosión y la dilatación. En nuestro caso aquí, el relleno de agujeros se basa en el proceso de dilatación, que básicamente hace crecer los objetos representados por píxeles de primer plano.

Nos ocupamos de varias operaciones en el procesamiento de imágenes y el relleno de agujeros es uno de los no lineales, lo que significa que su resultado depende solo de las posiciones de los píxeles de primer plano y de fondo. Por tanto, en los procesos morfológicos en general, dividimos los píxeles en conjuntos.

En primer lugar, debemos establecer un punto dentro de la forma que queremos llenar con píxeles de primer plano. A partir de ahí, dilatamos tantas veces este punto, que rellena la forma. Sin embargo, para controlar la dilatación, necesitamos dilatar solo los píxeles dentro del borde, por lo que establecemos una condición que lo haga por nosotros.

Debido a que la dilatación afecta a cada píxel de primer plano de la imagen, debemos aplicar este proceso de dilatación en una imagen separada. Por lo tanto, colocamos estos píxeles de relleno donde está la forma en la imagen de entrada y combinamos las imágenes al final.

Este proceso está definido por la siguiente ecuación:

$$X_k = (X_{k-1} \oplus B) \cap A^C$$

Figura 4. Ecuación que describe la dilatación de forma recursiva.

Se utiliza como punto de partida un punto al interior del objeto que se desea llenar como muestra la imagen X_0 y se realiza la operación dilatación con el elemento B . Esta dilatación expande la imagen en todas las direcciones. Sin embargo al realizar la intersección con la imagen A^C , se logra mantener los puntos dilatados siempre dentro de los bordes de la imagen original. Este proceso se repite n veces hasta que la imagen X_k sea idéntica a la imagen X_{k-1} .

Es decir, hasta que no se generen nuevos cambios y el proceso se detenga.

Esta es la forma más sencilla en que podría demostrar este proceso. Sin embargo, tiene una desventaja, que es un tiempo de computación prolongado. Cuanto más grande sea la forma que queramos llenar, más dilataciones necesitará realizar y más tiempo llevará hacerlo.

Desarrollo del código en Matlab

```
%Obtenemos la imagen
imo=imread('plaquetas.jpg');
im=rgb2gray(imo);
imD=double(im);
%Mostrar imagen original
figure(1),imshow(im);
[f, c]=size(imD);
for i=1:f
    for j=1:c
        if imD(i,j)>=135
            nuevaI(i,j) = 0;
        else
            nuevaI(i,j) = 255;
        end
    end
end
imB = uint8( nuevaI);
%Mostramos la imagen binarizada
figure(2),imshow(imB);
B=[0 1 0;1 1 1;0 1 0];
```

```
%Erosionamos recursivamente
E=imerode(imB,B);
E=imerode(E,B);
E=imerode(E,B);
E=imerode(E,B);
%E=imdilate(E,B);
%E=imdilate(E,B);

E=imfill(E,'holes');
Ib=imB;
%Se obtienen las dimensiones de la imagen binaria
%donde 0 es el fondo y 1 los objetos
[m ,n]=size(Ib);
%La imagen binaria se convierte a double, para
%que pueda contener valores mayores a 1.
Ibd=double(Ib);
%PASO 1. Se asignan etiquetas iniciales
%Se inicializan las variables para las etiquetas e
%y para las colisiones k
e=2;
k=1;
%Se recorre la imagen de izquierda a derecha y de
%arriba a abajo
for r=2:m-1
    for c=2:n-1

        %Si los píxeles vecinos son ceros se asigna una
        %etiqueta y se incrementa el número de etiquetas.
        if(Ibd(r,c)==1)
            if((Ibd(r,c-1)==0)&&(Ibd(r-1,c)==0))
                Ibd(r,c)=e;
                e=e+1;
            end
        %Si uno de los píxeles vecinos tiene una etiqueta
        %asignada esta etiqueta se le asigna al píxel
        %actual.
            if(((Ibd(r,c-1)>1)&&(Ibd(r-1,c)<2))|| ...
               ((Ibd(r,c-1)<2)&&(Ibd(r1,c)>1)))
                if(Ibd(r,c-1)>1)
                    Ibd(r,c)=Ibd(r,c-1);
                end
                if(Ibd(r-1,c)>1)
                    Ibd(r,c)=Ibd(r-1,c);
                end
            end
        %Si varios de los pixeles vecinos tienen una etiqueta
        %asignada se le asigna una de ellas a este pixel.
            if((Ibd(r,c-1)>1)&&(Ibd(r-1,c)>1))
                Ibd(r,c)=Ibd(r-1,c);
```

```

%Las etiquetas no usadas son registradas como
%colisiones
    if((Ibd(r,c-1))~= (Ibd(r-1,c)))
        ec1(k)=Ibd(r-1,c);
        ec2(k)=Ibd(r,c-1);
        k=k+1;
    end
end
end
end

%PASO 2. Se resuelven colisiones
for ind=1:k-1
    %Se detecta la etiqueta mas pequeña de las que
    %participan en la colisión. El grupo píxeles
    %pertenecientes a la etiqueta menor absorben a los
    %de la etiqueta mayor.
    if(ec1(ind)<=ec2(ind))
        for r=1:m
            for c=1:n
                if (Ibd(r,c)==ec2(ind))
                    Ibd(r,c)=ec1(ind);
                end
            end
        end
    end
    if(ec1(ind)>ec2(ind))
        for r=1:m
            for c=1:n
                if (Ibd(r,c)==ec1(ind))
                    Ibd(r,c)=ec2(ind);
                end
            end
        end
    end
end

%La función unique entrega los valores de la matriz
%(Ibd), únicos, es decir que no se repiten, por lo
%que serán entregados solamente aquellos valores que
%permanecieron al resolver el problema de las
%colisiones.
w = unique(Ibd);
t=length(w);

%PASO 3. Re-etiquetado de la imagen
%Se re-etiquetan los píxeles con los valores mínimos
Ime=bwlabel(E,8);
for r=1:m
    for c=1:n
        for ix=2:t
            if (Ime(r,c)==w(ix))
                Ime(r,c)=ix-1;
            end
        end
    end
end

total_objetos=max(max(Ime));
for i=1:total_objetos
    array(i)= numel(find(Ime==i));
end
%Se preparan los datos para despliegue
E=mat2gray(Ime);
Ime=bwlabel(E,8);

%Se encuentra los objetos contenidos en Ib
L=bwlabel(E,8);
%Se obtiene el numero de objetos detectados
idx=max(max(L));
%Se recorre para cada objetos
for o=1:idx
    %Se selecciona un objeto
    O=L==o;
    %Se calcula el centroide
    H=regionprops(double(O),'centroid');
    %Se guardan sus posiciones en la Matriz Pt
    Pt(o,1)=H.Centroid(1);
    Pt(o,2)=H.Centroid(2);
end

%Se despliega la imagen
% Im=E;
% Im=double(Im);
figure(3),imshow(E),title(['Numero de Elementos:' ...
    ,num2str(total_objetos)])
disp('El numero total de objetos en la imagen es:')
disp(total_objetos)
%Se retiene el objeto grafico
hold on
%Se grafican uno a uno los centroides
for d=1:idx
    text(Pt(d,1),Pt(d,2),strcat('\color{magenta}' ...
        ,num2str(array(d))), 'FontSize',10);
end

aux=0;
for i=1:idx
    for j=1:idx-1
        if(array(j)>array(j+1))
            aux=array(j);
            array(j)=array(j+1);
            array(j+1)=aux;
        end
    end
    disp(array(j));
end
for d=1:idx
    arreglo1(d)=array(d);
    str = '-';
    [num2str(d) str num2str(arreglo1(d))];
end
for d=1:idx
    text(210,d*8,strcat('\color{red}A de:',num2str(d) ...
        , ' es:',num2str(array(d))), 'FontSize',10);
end

```

RESULTADOS

Para esta práctica se buscó una imagen que tuviera distintos objetos separados y que fueran distinguibles entre sí, la imagen usada muestra plaquetas vistas del microscopio.

Imagen original en blanco y negro

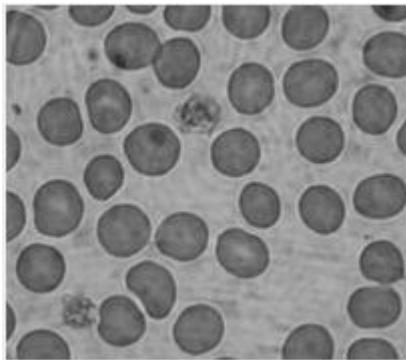


Figura 5. Imagen de plaquetas en blanco y negro.

Imagen binarizada

Hay que tomar en cuenta que para poder ver de una forma más clara la imagen es necesario modificar los valores al momento de ser la binarización, yo lo hice con distintas imágenes para comprobarlo y con distintos valores en el condicional if al momento de dar 0 y 255.

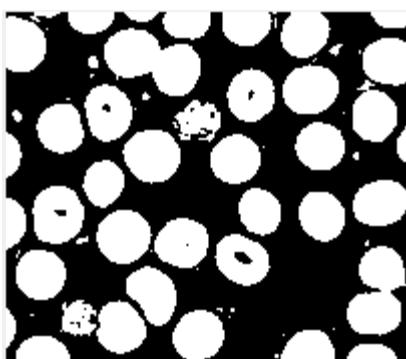


Figura 6. Imagen de plaquetas binarizada.

Imagen erosionada

Como ya se mencionó en el desarrollo de la práctica es necesario erosionar la imagen para poder detectar un borde y como también se mencionó la imagen se erosionó recursivamente este es el resultado de la operación realizada varias veces.

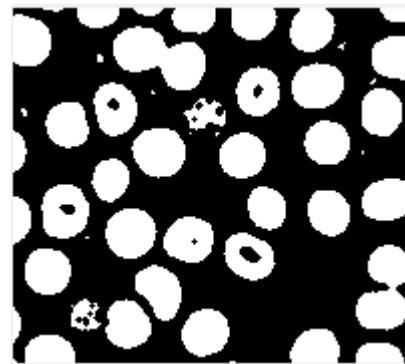


Figura 7. Imagen de plaquetas erosionada.

Plaquetas con el relleno aplicado y con los objetos contabilizados

Después de que se aplicó la erosión a la imagen como se mencionó en el desarrollo se realizó el relleno mediante el algoritmo de fill hole, la imagen que se muestra es esta pero con los objetos ya contabilizados, se detectaron 51 objetos en total como bien lo dice la imagen.

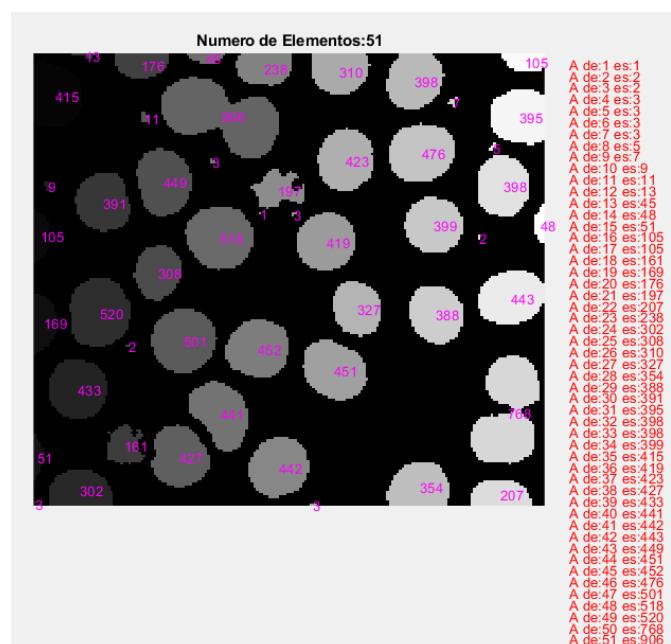


Figura 8. Imagen con plaquetas contabilizadas

CONCLUSIONES

En esta práctica se realizó un algoritmo capaz de detectar objetos mediante los bordes aplicando filtros morfológicos erosión y dilatación, también se aplicó un algoritmo de relleno para poder facilitar la detección.

Como se puede ver en los resultados el código final funciona correctamente sólo que hay que tomar en cuenta algunos parámetros como lo son la sensibilidad al momento de binarizar la imagen original y que tantas veces se va a erosionar o dilatar la imagen.

también hay que mencionar que de la manera en la que está construida el algoritmo al momento de hacer el conteo de objetos se detectan intensidades con valor de 255, lo que quiere decir que sí tu binarizas tu imagen y el objeto deseado queda en color negro, tienes que hacer un pequeño ajuste para que éste quede de color blanco.

Evidentemente las aplicaciones que este programa pueda tener son bastantes como lo vemos aquí para el conteo de plaquetas e incluso industrialmente para contabilizar la producción.

BIBLIOGRAFIA

- [1] E. Cuevas, M. Diaz y J. Camarena, Tratamiento de imágenes con MATLAB. México: Alfaomega, 2017.
- [2] A. Juárez. "How To Make Hole Filling In Image Processing Work In C# - Epoch Abuse". Epoch Abuse. <https://epochabuse.com/hole-filling-in-image-processing/> (accedido el 28 de octubre de 2022).
- [3] R. Gonzalez y R. Woods, Digital Image Processing, 3a ed. Miami: Pearson, 2017.
- [4] A. Ramírez. "Reporte de Búsqueda, detección y conteo de objetos". cimat. http://www.cimat.mx/~alram/VC/ramirez_seg_mObjDetecc.pdf (accedido el 28 de octubre de 2022).

Asignatura: Procesamiento de imágenes

Profesor: D. Sc. Gerardo García Gil

Alumno: Carlo Pinedo Suarez

Registro: 19310163 **Ciclo:** 2022-B

Ingeniería en Desarrollo de Software

Centro de Enseñanza Técnica Industrial (CETI)

INTRODUCCIÓN

Hoy en día, no paran de aparecer nuevas tecnologías que nos facilitan un poco más nuestro trabajo en el sector de la arquitectura o la ingeniería. Unas más acertadas, y otras menos, pero ahí están como opción para utilizarlas. Hacer una auditoría o un inventario de material que podamos tener en el almacén o en la propia obra «in situ» puede llegar a ser una tarea muy tediosa, sobre todo, si hablamos de elementos pequeños; material de PVC, tuberías, ladrillos, armado, etc. Para una pequeña empresa del sector de la construcción, puede que no sea demasiado tiempo el conteo, pero cuando hablamos ya de grandes empresas constructoras, obras enormes o almacenes de dimensiones importantes, el control del inventario puede llegar a ser una auténtica pesadilla. Aunque ya podemos ver algunas soluciones.

La idea de esta práctica es crear una aplicación que sea capaz de facilitar ese trabajo mediante el uso de inteligencia artificial, aplicando lo que llamamos filtros morfológicos tanto de erosión como de dilatación.

Los filtros morfológicos fueron concebidos para ser usados sobre imágenes binarias, es decir, sobre aquéllas cuyos píxeles sólo tienen dos posibles valores: 1 y 0, blanco negro. Las imágenes binarias se encuentran en un gran número de aplicaciones, especialmente en procesamiento de documentos. En lo sucesivo, definiremos a los píxeles de la estructura como los correspondientes a 1, y los del fondo a 0.

Los filtros morfológicos se basan en operaciones de conjuntos, que ejecutan alteraciones simples a las imágenes en forma. Las dos operaciones morfológicas más comunes son la dilatación y erosión, y de sus combinaciones surgen las operaciones de apertura, cierre y gradiente morfológico. Generalmente, los filtrados morfológicos se aplican conjunto a los filtrados

de suavizado, con lo que se rellenan espacios de la imagen y se elimina el ruido de esta, manteniendo la seguridad y privacidad de la información.

Las operaciones primarias morfológicas son la erosión y la dilatación. A partir de ellas podemos componer las operaciones de apertura y clausura. Son estas dos operaciones las que tienen mucha relación con la representación de formas, la descomposición y la extracción de primitivas.

DESARROLLO

Para desarrollar esta práctica es necesario conocer los siguientes conceptos, también explicaré el algoritmo fill hole que será implementado.

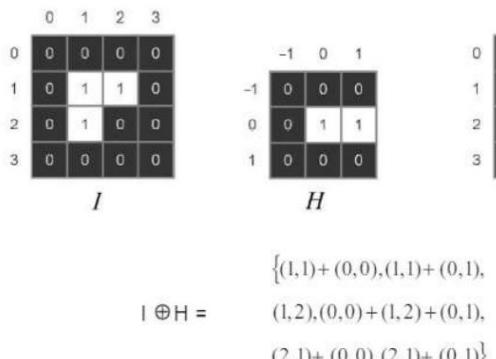
Dilatación

La dilatación es una operación morfológica correspondiente a la idea intuitiva del crecimiento, o de añadir una capa de píxeles a la estructura de la imagen. La manera específica en la que es controlado el crecimiento es por medio de una figura considerada como estructura de referencia. Esta operación es definida en la notación de conjuntos de puntos como:

$$I \oplus H = \{(x', y') = (x + i, y + j) | (x', y') \in P_I, (i, j) \in P_H\}$$

Según la expresión de la diapositiva anterior, el conjunto de puntos que constituyen la dilatación de una imagen | y la estructura de referencia H queda definido por todas las posibles combinaciones de los pares de coordenadas de los conjuntos de puntos PI y PH. Se podría también interpretar a la operación de dilatación como el resultado de añadir a los píxeles 1 de la imagen (PI) la forma correspondiente a la estructura de

referencia. Lo anterior puede ser ilustrado en la imagen siguiente.

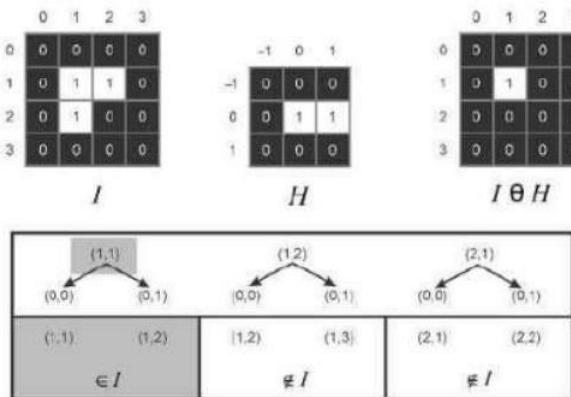


Erosión

La operación quasi inversa de la dilatación es la erosión, la cual expresada en la notación de conjunto de puntos es definida como:

$$I \ominus H = \{(x', y') | (x' + i, y' + j) \in P_I(i, j), \forall (i, j) \in P_H\}$$

Esta expresión dice que para cada punto (x', y') de la imagen, el resultado se compone por los puntos (x', y') para los cuales todos los posibles valores $(x' + i, y' + j)$ se encuentran en I . La figura de la siguiente diapositiva muestra un ejemplo de la erosión desempeñada entre una imagen I y la estructura de referencia H .



Este proceso puede ser interpretado de la siguiente manera: Un pixel (Y, Y) resultado de la erosión es uno (1) si la estructura de referencia centrada en éste pixel coincide en forma con el contenido de la imagen. La figura de la diapositiva 9 muestra cómo la estructura de referencia centrada en el pixel $(1,1)$ de la imagen

coincide en forma con el contenido de la misma, por lo que el resultado de la erosión para este pixel es 1.

Relleno de imágenes

El relleno de agujeros o el relleno de regiones en el procesamiento de imágenes es una operación morfológica. Además, como su nombre podría sugerir, llena agujeros dentro de formas huecas. Definimos estas formas huecas o agujeros como píxeles de fondo encerrados por un borde de píxeles de primer plano.

Todos los procesos morfológicos se basan en dos operaciones fundamentales, que son la erosión y la dilatación. En nuestro caso aquí, el relleno de agujeros se basa en el proceso de dilatación, que básicamente hace crecer los objetos representados por píxeles de primer plano.

Nos ocupamos de varias operaciones en el procesamiento de imágenes y el relleno de agujeros es uno de los no lineales, lo que significa que su resultado depende solo de las posiciones de los píxeles de primer plano y de fondo. Por tanto, en los procesos morfológicos en general, dividimos los píxeles en conjuntos.

En primer lugar, debemos establecer un punto dentro de la forma que queremos llenar con píxeles de primer plano. A partir de ahí, dilatamos tantas veces este punto, que rellena la forma. Sin embargo, para controlar la dilatación, necesitamos dilatar solo los píxeles dentro del borde, por lo que establecemos una condición que lo haga por nosotros.

Debido a que la dilatación afecta a cada píxel de primer plano de la imagen, debemos aplicar este proceso de dilatación en una imagen separada. Por lo tanto, colocamos estos píxeles de relleno donde está la forma en la imagen de entrada y combinamos las imágenes al final.

Este proceso está definido por la siguiente ecuación:

$$X_k = (X_{k-1} \oplus B) \cap A^C$$

Figura 4. Ecuación que describe la dilatación de forma recursiva.

Se utiliza como punto de partida un punto al interior del objeto que se desea llenar como muestra la imagen X0 y se realiza la operación dilatación con el elemento B. Esta dilatación expande la imagen en todas las direcciones. Sin embargo al realizar la intersección con la imagen AC, se logra mantener los puntos dilatados siempre dentro de los bordes de la imagen original. Este proceso se repite n veces hasta que la imagen Xk sea idéntica a la imagen Xk-1.

Es decir, hasta que no se generen nuevos cambios y el proceso se detenga.

Esta es la forma más sencilla en que podría demostrar este proceso. Sin embargo, tiene una desventaja, que es un tiempo de computación prolongado. Cuanto más grande sea la forma que queramos llenar, más dilataciones necesitará realizar y más tiempo llevará hacerlo.

Desarrollo del código en Matlab

Como se mencionó en la introducción de la práctica la idea es crear un programa capaz de contabilizar material para una obra el código está orientado en este aspecto usaremos la imagen de unos ladrillos para exemplificar el funcionamiento.

```
%Obtenemos la imagen
imo = imread('ladrillos.jpg');
im = rgb2gray(imo);
imD = double(im);
%Mostrar imagen original
figure(1), imshow(im);
[f, c] = size(imD);
for i=1:f
    for j=1:c
        if imD(i,j) >= 90
            nuevaI(i,j) = 255;
        else
            nuevaI(i,j) = 0;
        end
    end
end
imB = uint8(nuevaI);
%Mostramos la imagen binarizada
figure(2), imshow(imB);
B=[0 1 0;1 1 1;0 1 0];
```

```
%Erosionamos recursivamente
E=imerode(imB,B);
E=imerode(E,B);
E=imerode(E,B);
E=imerode(E,B);
E=imdilate(E,B);
E=imdilate(E,B);
E=imdilate(E,B);

%Muestro la imagen erosionada
figure(4), imshow(E);

E=imfill(E,'holes');
Ib=imB;
%Se obtienen las dimensiones de la imagen binaria
%donde 0 es el fondo y 1 los objetos
[m ,n]=size(Ib);
%La imagen binaria se convierte a double, para
%que pueda contener valores mayores a 1.
Ibd=double(Ib);
%PASO 1. Se asignan etiquetas iniciales
%Se inicializan las variables para las etiquetas e
%y para las colisiones k
e=2;
k=1;

%Se recorre la imagen de izquierda a derecha y de
%arriba a abajo
for r=2:m-1
    for c=2:n-1
        %Si los píxeles vecinos son ceros se asigna una
        %etiqueta y se incrementa el número de etiquetas.
        if(Ibd(r,c)==1)
            if((Ibd(r,c-1)==0)&&(Ibd(r-1,c)==0))
                Ibd(r,c)=e;
                e=e+1;
            end
        end
        %Si uno de los píxeles vecinos tiene una etiqueta
        %asignada esta etiqueta se le asigna al píxel
        %actual.
        if(((Ibd(r,c-1)>1)&&(Ibd(r-1,c)<2))|| ...
           ((Ibd(r,c-1)<2)&&(Ibd(r1,c)>1)))
            if(Ibd(r,c-1)>1)
                Ibd(r,c)=Ibd(r,c-1);
            end
            if(Ibd(r-1,c)>1)
                Ibd(r,c)=Ibd(r-1,c);
            end
        end
    end
end
```

```

%Si varios de los píxeles vecinos tienen una etiqueta
%asignada se le asigna una de ellas a este pixel.
if((Ibd(r,c-1)>1)&&(Ibd(r-1,c)>1))
    Ibd(r,c)=Ibd(r-1,c);
%Las etiquetas no usadas son registradas como
%colisiones
if((Ibd(r,c-1))==(Ibd(r-1,c)))
    ec1(k)=Ibd(r-1,c);
    ec2(k)=Ibd(r,c-1);
    k=k+1;
    end
end
end
end

%PASO 2. Se resuelven colisiones
for ind=1:k-1
%Se detecta la etiqueta mas pequeña de las que
%participan en la colisión. El grupo píxeles
%pertenecientes a la etiqueta menor absorben a los
%de la etiqueta mayor.
    if(ec1(ind)<=ec2(ind))
        for r=1:m
            for c=1:n
                if (Ibd(r,c)==ec2(ind))
                    Ibd(r,c)=ec1(ind);
                end
            end
        end
    end
    if(ec1(ind)>ec2(ind))
        for r=1:m
            for c=1:n
                if (Ibd(r,c)==ec1(ind))
                    Ibd(r,c)=ec2(ind);
                end
            end
        end
    end
end

%La función unique entrega los valores de la matriz
%(Ibd), únicos, es decir que no se repiten, por lo
%que serán entregados solamente aquellos valores que
%permanecieron al resolver el problema de las
%colisiones.
w = unique(Ibd);
t=length(w);

%PASO 3. Re-etiquetado de la imagen
%Se re-etiquetan los píxeles con los valores mínimos.
Ime=bwlabel(E,8);
for r=1:m
    for c=1:n
        for ix=2:t
            if (Ime(r,c)==w(ix))
                Ime(r,c)=ix-1;
            end
        end
    end
end

total_objetos=max(max(Ime));
for i=1:total_objetos
    array(i)= numel(find(Ime==i));
end
%Se preparan los datos para despliegue
E=mat2gray(Ime);
Ime=bwlabel(E,8);

%Se encuentra los objetos contenidos en Ib
L=bwlabel(E,8);
%Se obtiene el numero de objetos detectados
idx=max(max(L));
%Se recorre para cada objetos
for o=1:idx
%Se selecciona un objeto
    O=L==o;
%Se calcula el centroide
    H=regionprops(double(O),'centroid');
%Se guardan sus posiciones en la Matriz Pt
    Pt(o,1)=H.Centroid(1);
    Pt(o,2)=H.Centroid(2);
end
%Se despliega la imagen
% Im=E;
% Im=double(Im);
figure(3),imshow(E),title(['Número de Elementos:' ...
    ,num2str(total_objetos)])
disp('El número total de objetos en la imagen es:')
disp(total_objetos)
%Se retiene el objeto grafico
hold on

%Se grafican uno a uno los centroides
for d=1:idx
    text(Pt(d,1),Pt(d,2),strcat('\color{magenta}' ...
        ,num2str(array(d))), 'FontSize',10);
end

aux=0;
for i=1:idx
    for j=1:idx-1
        if(array(j)>array(j+1))
            aux=array(j);
            array(j)=array(j+1);
            array(j+1)=aux;
        end
    end
    disp(array(j));
end
for d=1:idx
    arreglo1(d)=array(d);
    str = '-';
    [num2str(d) str num2str(arreglo1(d))]
end
for d=1:idx
    text(210,d*4.5,strcat('\color{red}A de:',num2str(d) ...
        , ' es:',num2str(array(d))), 'FontSize',10);
end

```

RESULTADOS

Para esta práctica fue elegida una foto con ladrillos apilados pero bien podría elegirse cualquier otro material como troncos , tuberías, vigas, etc.

Imagen original en blanco y negro

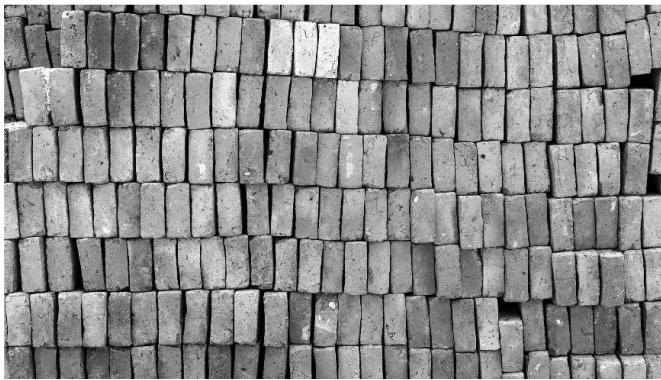


Figura 5. Imagen de plaquetas en blanco y negro.

Imagen binarizada

Hay que tomar en cuenta que para poder ver de una forma más clara la imagen es necesario modificar los valores al momento de ser la binarización, yo lo hice con distintas imágenes para comprobarlo y con distintos valores en el condicional if al momento de dar 0 y 255.



Figura 5. Imagen de plaquetas en blanco y negro.

Imagen erosionada y dilatada

Como ya se mencionó en el desarrollo de la práctica es necesario erosionar y dilatar la imagen para poder contabilizar material, también se mencionó la imagen se erosionó recursivamente este es el resultado de la operación realizada varias veces

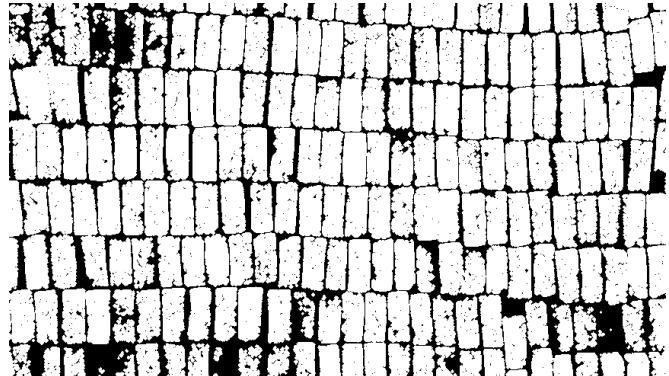


Figura 5. Imagen de plaquetas en blanco y negro.

Ladrillos con el relleno y aplicado y contabilizados

Como podemos observar en la imagen por un contabilizados 232 ladrillos qué es un número bastante exacto comparado a si los cuentas uno a uno.

Y con esto comprobamos el funcionamiento de nuestro programa.

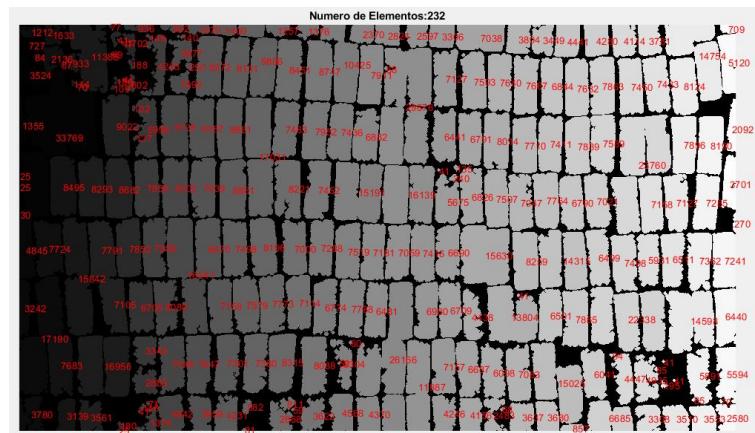


Figura 5. Imagen de plaquetas en blanco y negro.

CONCLUSIONES

En esta práctica se realizó un programa aplicación para filtros morfológicos capaz de contabilizar material de obra, evidentemente no sólo se pueden usar ladrillos como se mostró en nuestros resultados puede usarse cualquier otro tipo de material siempre y cuando se encuentre apilado invisible para la fotografía. Para obtener mejores resultados hay que hacer Ajustes a la erosión y a la dilatación según convenga, también hay que cuidar la calidad de la imagen y en una parte del algoritmo específicamente la que se encarga de la binarización es que tocar los valores para asignar los unos y ceros respectivamente.

BIBLIOGRAFIA

- [1] E. Cuevas, M. Diaz y J. Camarena, Tratamiento de imágenes con MATLAB. México: Alfaomega, 2017.
- [2] A. Juárez. "How To Make Hole Filling In Image Processing Work In C# - Epoch Abuse". Epoch Abuse. <https://epochabuse.com/hole-filling-in-image-processing/> (accedido el 28 de octubre de 2022).
- [3] R. Gonzalez y R. Woods, Digital Image Processing, 3a ed. Miami: Pearson, 2017.
- [4] A. Ramírez. "Reporte de Búsqueda, detección y conteo de objetos". cimat. http://www.cimat.mx/~alram/VC/ramirez_seg_mObjDetecc.pdf (accedido el 28 de octubre de 2022).
- [5] P. Gómez. "App para contar el material de obra y almacén en 5 minutos | OVACEN". OVACEN. <https://ovacen.com/app-contar-material/> (accedido el 31 de octubre de 2022).

Asignatura: Procesamiento de imágenes

Profesor: D. Sc. Gerardo García Gil

Alumno: Carlo Pinedo Suarez

Registro: 19310163 **Ciclo:** 2022-B

Ingeniería en Desarrollo de Software

Centro de Enseñanza Técnica Industrial (CETI)

INTRODUCCIÓN

El análisis y procesamiento de imágenes se realiza a través de computadoras, debido a la complejidad y el número de cálculos necesarios para realizarlo. Es por esto que, si bien la formulación matemática necesaria para su realización data de varios siglos atrás, la posibilidad real de utilizarla de forma cotidiana en la práctica clínica ha sido posible recién en las últimas décadas, gracias al avance en las tecnologías del hardware.

La proliferación de nuevos equipamientos con capacidad para realizar millones de operaciones por segundo y su extensión a la vida cotidiana y a todo tipo de usuario, ha hecho posible que el análisis y procesamiento de imágenes digitales se constituya en un gran campo de estudio. En la actualidad, esta tecnología se encuentra incorporada incluso en todo tipo de equipamiento doméstico, como cámaras digitales, scaners y teléfonos celulares, entre otros.

En términos históricos, la utilización de imágenes radiográficas para diagnóstico clínico data prácticamente desde el descubrimiento de los rayos X en 1895 (Röentgen). Incluso, las imágenes funcionales a partir de la emisión de fotones (rayos γ) por parte de radionucleidos ya cuenta con más de 90 años de antigüedad (Hevesy & Seaborg, 1924). Sin embargo, las imágenes eran adquiridas sobre films radiográficos o directamente in vivo, por lo que su correcto procesamiento no ha explotado su real potencialidad sino hasta la incorporación de la tecnología que permitió digitalizarlas.

El motivo principal de esta “aparición tardía” del procesamiento de imágenes ha sido entonces, debido a los requerimientos de hardware tanto para el procesamiento de las mismas como para la representación de estas en sistemas gráficos de alta performance. Paralelamente a este desarrollo, la formulación de algoritmos para el procesamiento ha seguido los avances tecnológicos logrando un alto

grado de sofisticación y manipulación de imágenes en tiempo casi real.

El procesamiento digital de imágenes es el conjunto de técnicas que se aplican a las imágenes digitales con el objetivo de mejorar la calidad o facilitar la búsqueda de información.

Un modelo de color es un modelo matemático abstracto que permite representar los colores numéricamente, normalmente utilizando tres o cuatro valores o componentes de color. Por ejemplo, el modelo de color define cómo el programa dibuja los píxeles que componen una imagen, texto o forma en particular. Como si de un aula de plástico se tratara, una parte periférica representa o crea imágenes acción digital, crea el mejor color de otros.

Es decir, los colores se representan mezclando colores primarios o primarios de forma secuencial, y dependiendo de la presencia e intensidad de los diferentes colores que se mezclen, se obtiene el color real.

Bueno, los modelos de color se definen por los colores que se consideran colores primarios, entre otras cosas. De ahí viene el acrónimo RGB: Red, Green, Blue.

La familia de espacios de color HSI se usa principalmente en gráficos por computadora para definir colores usando representaciones artísticas de tinte, tinte y tinte. Proviene del espacio de color RGB mediante la conversión de coordenadas. En los sistemas de imágenes por computadora, es necesario convertir las coordenadas de color de HSI a RGB para mostrarlas en la pantalla y viceversa para las operaciones de procesamiento.

DESARROLLO

El desarrollo de esta práctica consiste en modificar el color del objeto de una imagen utilizando la conversión de modelos de colores RGB y HSI, es por esto que hay que tener bien claro cómo funciona cada modelo. Así que empezaré explicando que son.

Modelo de color RGB

La suposición básica que subyace en la teoría colorimétrica moderna (cuando se trata de tareas de procesamiento de imágenes) que subyace en la visión del color humano es la estimulación de tres capas de células sensibles a la luz (llamadas conos) en la retina. Estos fotorreceptores, llamados rojo, verde y azul, definen un espacio de tres colores llamado RGB, cuyos colores primarios son colores puros en las frecuencias bajas, medias y altas del espectro electromagnético visible.

En el modelo de color RGB, cada color aparece en sus principales componentes espectrales: rojo, verde y azul (rojo, verde, azul) según el sistema de coordenadas cartesianas. El subespacio de color RGB es el cubo que se muestra en la (Fig. 1), en el que

los componentes rojo, verde y azul están en las tres esquinas, los componentes cian, magenta y amarillo en las otras tres esquinas, el componente negro en el punto de inicio [0 0 0] y el componente blanco en la esquina más alejada del punto de inicio [1 1 1]. Por conveniencia, se supone que los valores están normalizados al intervalo [0,1], lo que produce el cubo de la figura 1, que se denomina bloque unitario.

La imagen en el modelo de color RGB consta de tres planos de imagen independientes; uno para cada color base. Cuando se convierte a una pantalla RGB, estas tres imágenes se combinan en la pantalla de fósforo para producir una imagen de color compuesta. Por lo tanto, tiene sentido utilizar el modelo RGB para el procesamiento cuando las propias imágenes se representan de forma natural en estos planos de color.

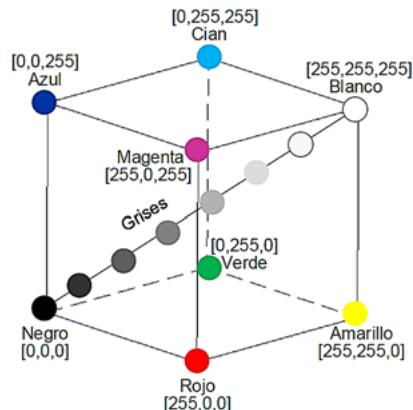


Figura 1.

Modelo de color HSI

La familia de espacios de color HSI se usa principalmente en gráficos por computadora para definir colores usando representaciones artísticas de sombras, matices y tonos. Proviene del espacio de color RGB mediante la conversión de coordenadas. En los sistemas de imágenes por computadora, es necesario convertir las coordenadas de color de HSI a RGB para mostrarlas en la pantalla y viceversa para las operaciones de procesamiento.

El espacio de color HSI se desarrolló para cuantificar el valor del tono (H), la saturación (S) y la intensidad (I) de un color. H es una medida del contenido espectral de un color. Se representa como un ángulo alrededor del eje vertical y varía de 0 a 360 grados, comenzando con rojo en 0 grados.

En este espacio de color, la saturación (S) se extiende desde 0 (a lo largo del eje I) radialmente hacia afuera, con un máximo de 1 en la superficie del cono. Este componente se refiere al porcentaje de luz pura con una longitud de onda dominante e indica qué tan lejos se compara el color con el gris de igual brillo. La intensidad es una medida del brillo relativo; esto es un rango [0, 1].

En la parte superior e inferior del cono, donde los valores I son 1 y 0, la saturación y el tono no tienen sentido ni están definidos. En cualquier punto del eje I, el componente de saturación es 0 y el matiz (H) no está definido. Esta singularidad ocurre en cualquier lugar R = G = B.

El modelo de color HSI deriva su utilidad de dos factores principales:

1. Al igual que en el modelo YIQ, el componente de intensidad I está separado de la información cromática.
2. Los componentes de tono (H) y saturación (S) están estrechamente relacionados. Tiene que ver con la forma en que los humanos percibimos los colores. Véase la figura 2 que muestra el espacio HSI.

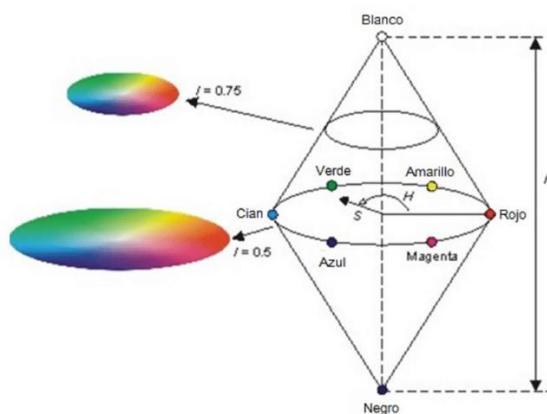


Figura 2. Modelo HSI.

Ventajas:

1. Buena compatibilidad con la intuición humana.
2. Separabilidad de los valores cromáticos y acromáticos.
3. Posibilidad de usar una característica de color como tonalidad, con propósitos de segmentación. Se puede usar H, S o I como característica única.

Desventajas:

1. Singularidades en la transformación. Por ejemplo tonalidad indefinida para puntos acromáticos.
2. Sensibilidad a pequeñas variaciones de valores RGB cerca de los puntos de singularidad.
3. Inestabilidad numérica al operar con valores de tonalidad (H) dada la naturaleza angular de esta característica.

Conversión RGB a HSI

La conversión de RGB (rango de [0,1]) a HSI con rango de [0,360] para H y [0,1] para S e I es la siguiente:

$$H = \cos^{-1} \frac{1}{2}[(R-G) + (R-B)] / [(R-G)^2 + (R-B)(G-B)]^{1/2}$$

$$S = 1 - [3/(R+G+B) * \min(R, G, B)]$$

$$I = 1/3 (R+G+B)$$

Figura 3. Fórmulas general de conversión RGB HSI.

La conversión de HSI (rango de [0,360] para H y [0,1] para S e I) a RGB con rango de valores en [0,1] es la siguiente:

Si H está en el sector RG ($0 < H \leq 2\pi/3$):

$$b = 1/3 (1-S)$$

$$r = 1/3 (1 + [S * \cos H / \cos (\pi/3 - H)])$$

$$g = 1 - (r + b)$$

Figura 4. Fórmulas de conversión RGB HSI para el sector RG.

Si H está en el sector GB ($2\pi/2 < H \leq 4\pi/3$):

$$H = H - 2\pi/3$$

$$r = 1/3 (1-S)$$

$$g = 1/3 (1 + [S * \cos H / \cos (\pi/3 - H)])$$

$$b = 1 - (r + g)$$

Figura 5. Fórmulas de conversión RGB HSI para el primer sector GB

Si H está en el sector BR ($4\pi/3 < H \leq 2\pi$):

$$H = H - 4\pi/3$$

$$g = 1/3 (1-S)$$

$$b = 1/3 (1 + [S * \cos H / \cos (\pi/3 - H)])$$

$$r = 1 - (g + b)$$

Figura 6. Fórmulas de conversión RGB HSI para el primer sector BR

La manera más fácil de seleccionar superficies de luminancia constantes es a través de planos. Una versión simplificada de la luminancia percibida en términos de valores RGB es $L = (R + G + B)/3$.

Otros miembros de la familia HSI, como el doble cono hexagonal HLS, pueden ser obtenidos directamente del HSI modificando las superficies de luminancia constante, al definirse $L = (\max(R, G, B) + \min(R, G, B))/3$.

Si los valores máximos y mínimos coinciden, entonces $S = 0$ y la tonalidad es indefinida. En caso contrario, la saturación se define como:

$$\begin{aligned} \text{Si } L \leq 0.5 \text{ entonces } S &= (\text{Max} - \text{Min}) / (\text{Max} + \text{Min}) \\ \text{Si } L > 0.5 \text{ entonces } S &= (\text{Max} - \text{Min}) / (2 - \text{Max} - \text{Min}) \end{aligned}$$

Figura 7. Tonalidad si los valores no coinciden.

Donde Max = máximo (R, G, B) y Min = mínimo (R, G, B). La tonalidad (Hue) se calcula de la siguiente manera.

Si R = Max entonces:

$$H = (G' - B') / (\text{Max-Min})$$

Figura 8. H en caso de que R sea el valor máximo.

Si G = Max entonces:

$$H = (B - R) / (\text{Max-Min})$$

Figura 9. H en caso de que G sea el valor máximo.

Si B = Max entonces:

$$H = 4 + (R - G) / (\text{Max-Min})$$

Figura 10. H en caso de que B sea el valor máximo.

Conversión HSI a RGB

Hasta ahora lo que tenemos sería una imagen en formato HSI, pero para poder visualizarla nosotros tenemos que convertirla a formato RGB.

Nos ocuparemos de muchos más ángulos y trigonometría, por lo que debemos tener cuidado de calcular todo en grados o radianes.

Depende del ángulo de matiz, cómo calcularemos los valores RGB. Echemos un vistazo a las fórmulas y veamos a qué me refiero.

Para un ángulo de matiz entre 0 y 120 grados o $2/3$ de pi en radianes, usamos las siguientes fórmulas.

$$B = I(1 - S)$$

$$R = I \left[1 + \frac{S \cos H}{\cos(60^\circ - H)} \right]$$

$$G = 3I - (R + B)$$

Figura 11. Fórmulas de conversión de HSI a RGB para el sector RG

Y para ángulo de tonalidad entre 120 y 240 grados. Pero primero debemos restar 120 grados del ángulo antes de ingresarlos en las siguientes fórmulas.

$$R = I(1 - S)$$

$$G = I \left[1 + \frac{S \cos H}{\cos(60^\circ - H)} \right]$$

$$B = 3I - (R + G)$$

Figura 12. Fórmulas de conversión HSI RGB para el sector GB

Y para ángulo de tonalidad entre 120 y 240 grados. Pero primero debemos restar 120 grados del ángulo antes de ingresarlos en las siguientes fórmulas.

$$R = I(1 - S)$$

$$G = I \left[1 + \frac{S \cos H}{\cos(60^\circ - H)} \right]$$

$$B = 3I - (R + G)$$

Figura 13. Fórmulas de conversión HSI RGB para el sector GB

Y por último, para ángulo de tonalidad entre 240 y 360 grados. Necesitamos restar 240 grados del ángulo antes de comenzar.

$$G = I(1 - S)$$

$$B = I \left[1 + \frac{S \cos H}{\cos(60^\circ - H)} \right]$$

$$R = 3I - (G + B)$$

Figura 14. Fórmulas de conversión de HSI a RGB para el sector BR

Asignar color falso a un objeto en la imagen

Hasta ahora lo que tenemos es una conversión de RGB a HSI, lo que sigue es describir el algoritmo con el cual cambiaremos el color del objeto deseado.

Una imagen digital en color consta de tres matrices $m \times n$. Una matriz contiene los valores R, la segunda contiene los valores G y la tercera contiene los valores B. Cada píxel corresponde a la misma posición en cada una de estas tres matrices, por lo que el color depende del valor de la matriz en esa ubicación. En resumen, trabajar en el espacio RGB para imágenes digitales es trabajar con píxeles que contienen información R, G y B. De manera similar, una imagen digital en el espacio HSI consta de tres matrices de $m \times n$ que contienen los valores de H, S e I respectivamente. Así, cada píxel de la imagen digital en este espacio de color contiene tres datos: tono, saturación e intensidad.

Pasos para la realización del algoritmo:

1. Seleccionar una región del objeto en la imagen, hallar tono mínimo y tono máximo.

Para todos los pixeles de la imagen:

2. hallar los respectivos valores de H, S e I. (conversión del sistema RGB al HSI).
3. Elegir un nuevo tono. Si se encuentra que un píxel tiene un tono que se encuentre entre el tono mínimo y máximo, le asignamos el nuevo tono. Esto se hace modificando el valor de H, S e I.
4. Aplicar las transformaciones inversas. (conversión del sistema HSI a RGB).

Hasta aquí ya tenemos todo lo necesario para empezar a programar así que proseguiré a explicar el código utilizado en este proyecto.

La manera en la que obtuve los valores máximos y mínimos para los pixeles fue usando la función imtool, y buscando entre los pixeles tuve que encontrar manualmente los valores.

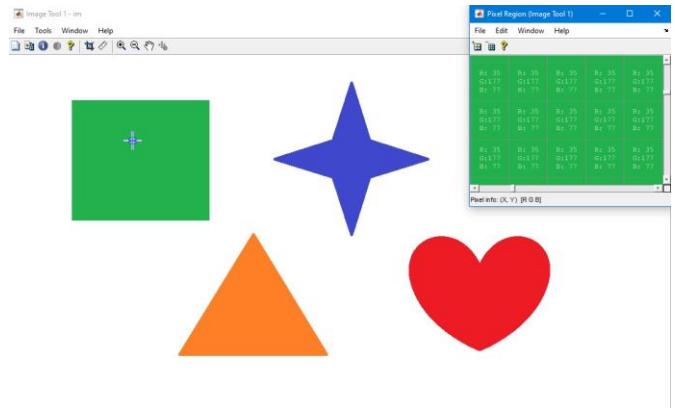


Figura 15. Imagen abierta en imtool.

Voy a hacer una pequeña anotación y es que para modificar el valor de H, S e I, yo busqué una página que me hiciera la conversión de color RGB a HSI y así saber de qué color iba a quedar pintado mi objeto. Como vemos en la siguiente imagen.

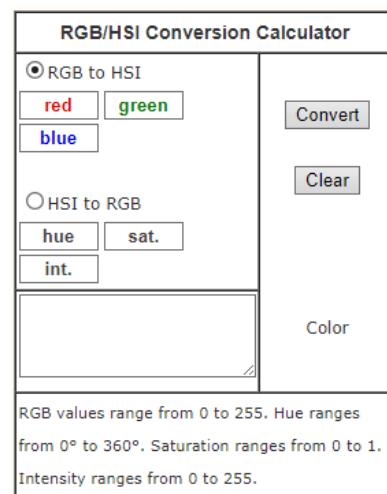


Figura 16. Conversor RGB A HSI.

Código

```
clc;
clear all;
im=imread('Formas.jpg');

figure(1)
imshow(im);
imtool(im);

I=double(im)/255;
```

```

%CONVERTIR RGB A HSI

%Obtenemos valores RGB maximos y minimos
%de la figura a detectar
xmin=186;
ymin=333;
xmax=75;
ymax=106;

%Extraemos los canales RGB de la imagen.
R = I(:,:,1);
G = I(:,:,2);
B = I(:,:,3);

%Obtenemos de los canales HSI de toda la
imagen
numi=1/2*((R-G)+(R-B));
denom=((R-G).^2+((R-B).*(G-B))).^0.5;
H=acosd(numi./ (denom+0.00001));
H(B>G)=360-H(B>G);
S=1-(3./ (sum(I,3)+0.00001)).*...
min(I,[],3);
I=sum(I,3)./3;

%Cambiar el color del objeto en la imagen
Hmin=H(xmin,ymin);
Hmax=H(xmax,ymax);
[m,n,t]=size(im);

for i=1:m
    for j=1:n
        if H(i,j)>=Hmin && H(i,j)<=Hmax
            H(i,j) = 185.1019;
            S(i,j) = 1;
            I(i,j) = 0.63399;
        end
    end
end
imgHSI = cat(3,H,S,I);
figure(2)
imshow(im2uint8(imgHSI));

%CONVERTIR HSI A RGB
HSI=zeros(size(im));
HSI(:,:1)=H;
HSI(:,:2)=S;
HSI(:,:3)=I;

R1=zeros(size(H));
G1=zeros(size(H));
B1=zeros(size(H));
RGB1=zeros([size(H),3]);

B1(H<120)=I(H<120).* (1-S(H<120));
R1(H<120)=I(H<120).* (1+((S(H<120).* ...
    cosd(H(H<120)))./cosd(60-H(H<120))));;
G1(H<120)=3.*I(H<120)(R1(H<120)+...
    B1(H<120));

H2=H-120;

R1(H>=120&H<240)=I(H>=120&H<240).* ...
    (1-S(H>=120&H<240));
G1(H>=120&H<240)=I(H>=120&H<240).* ...
    (1+((S(H>=120&H<240).* ...
        cosd(H2(H>=120&H<240)))./...
        cosd(60-H2(H>=120&H<240))));;
B1(H>=120&H<240)=3.*I(H>=120&H<240)- ...
    (R1(H>=120&H<240)+G1(H>=120&H<240));

H2=H-240;

G1(H>=240&H<=360)=I(H>=240&H<=360).* ...
    (1-S(H>=240&H<=360));
B1(H>=240&H<=360)=I(H>=240&H<=360).* ...
    (1+((S(H>=240&H<=360).* ...
        cosd(H2(H>=240&H<=360)))./...
        cosd(60-H2(H>=240&H<=360))));;
R1(H>=240&H<=360)=3.*I(H>=240&H<=360)- ...
    (G1(H>=240&H<=360)+B1(H>=240&H<=360));

RGB1 = cat(3,R1,G1,B1);
RGB1=im2uint8(RGB1);
figure(3)
imshow(RGB1);

```

RESULTADOS

Para esta práctica se buscó una imagen que tuviera distintos objetos separados y que fueran distinguibles entre sí, la imagen muestra distintas figuras geométricas con contrastes muy evidentes.

Imagen original

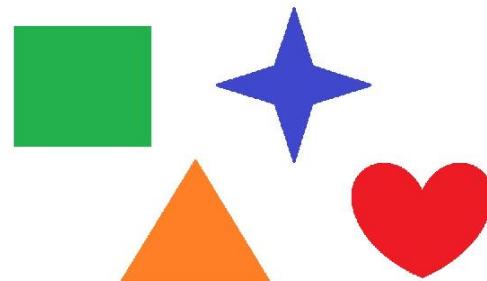


Figura 17. Imagen original con contrastes notorios.

Imagen convertida de RGB a HSI

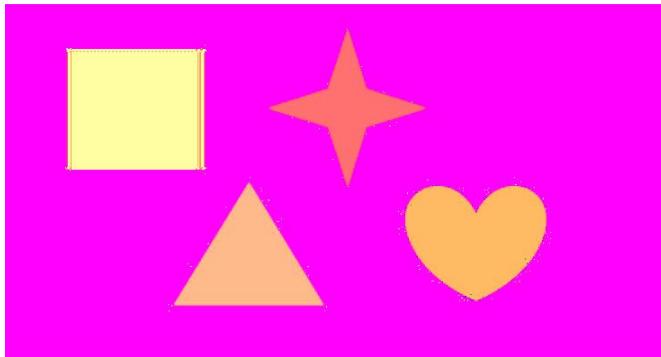


Figura 18. Imagen RGB convertida a HSI

Resultado final después de aplicar el color falso

Como podemos ver en la imagen de abajo la figura a la que se le aplicó el color falso fue el rectángulo verde superior izquierdo.

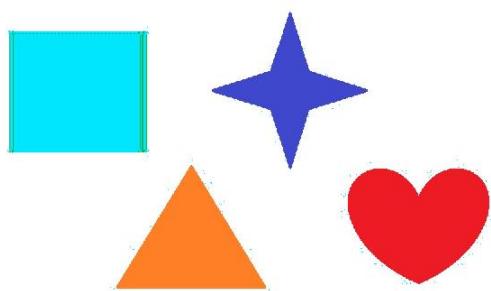


Figura 19. Imagen final con rectángulo coloreado.

CONCLUSIONES

Como podemos ver aplicar el algoritmo usado en este proyecto es bastante sencillo, lo complicado viene al momento de localizar los pixeles con los valores máximos y mínimos ya que sin el debido tratamiento o si la imagen tiene una pésima calidad los resultados al cambiar el color no serán los deseados. pasará que se dibujen otros pixeles en la imagen. Y claro cuando se utilizan imágenes distintas hay que estar jugando con los valores dentro del código.

El modelo HSI es bastante interesante ya que la perspectiva en que son vistos los colores nos da posibilidades que el modelo RGB no puede darnos como mencionó en las ventajas y desventajas. El hecho de que lo que estemos modificando sea un ángulo en el parámetro H da una idea como un modelo matemático puede ser aplicado en el software.

BIBLIOGRAFIA

- [1] E. Cuevas, M. Diaz y J. Camarena, Tratamiento de imágenes con MATLAB. México: Alfaomega, 2017.
- [2] R. Alvarado, "Segmentación de imágenes digitales sobre la base de la información integral del color", tesis doctoral, Universidad Nacional Autónoma de México, México, 2017.
- [3] R. Gonzalez y R. Woods, Digital Image Processing, 3a ed. Miami: Pearson, 2017.
- [4] A. Krzisnik. "How To - RGB To HSI And HSI To RGB Color Model In C# - Epoch Abuse". Epoch Abuse. <https://epochabuse.com/rgb-to-hsi/> (accedido el 10 de noviembre de 2022).
- [5] A. Pérez, "Uso del sistema HSI para asignar falso color a objetos en imágenes digitales", REVISTA MEXICANA DE FISICA, vol. 54, n.º 2, pp. 186–192, 2008.
- [6] O. Cuevas. "RGB to HSI, HSI to RGB Conversion Calculator". Had2Know - Free Online Calculators and How-to Guides. <https://www.had2know.org/technology/hsi-rgb-color-converter-equations.html> (accedido el 10 de noviembre de 2022).