Proyecto: filtrado de imagenes implementación de vision articial

Integrantes
Jose Yahriel Uribe Rosas
Sebastian Martinez Muñiz

May 2023

Universidad de Guadalajara
Centro Universitario de Ciencias Exactas e Ingenierías
Procesamiento digital de señales

Profesor:Stewart Rene Santos Arce

Contenido

1 Introducción

1.1 resumen general

El procesamiento de imágenes es una técnica utilizada para manipular y analizar imágenes digitales. Es una rama de la visión por computadora que se enfoca en el análisis y mejora de imágenes digitales mediante la utilización de algoritmos y técnicas matemáticas, una de las técnicas más comunes de procesamiento de imágenes es el filtrado de color. Este proceso implica seleccionar una región específica de una imagen que se distingue por su color y eliminar todo lo demás. para aplicar un filtro de color, se debe primero definir el rango de color que se desea eliminar o aislar. Esto se hace mediante la selección de un espacio de color, como RGB o HSV, y definiendo los valores de los canales de color que se desean filtrar. Por ejemplo, si se desea aislar un objeto rojo en una imagen RGB, se podrían filtrar todos los píxeles que tengan valores de rojo alto y valores de verde y azul bajos.

Una vez que se ha definido el rango de color deseado, se pueden aplicar varios tipos de filtros de color. Uno de los más comunes es el filtro binario, que asigna un valor 0 o 1 a cada píxel dependiendo de si cae dentro del rango de color deseado. Esto permite aislar la región de interés y eliminar todo lo demás. Otro tipo de filtro común es el filtro de convolución, que aplica una matriz ponderada a cada píxel de la imagen para suavizar o resaltar las características de interés. Por ejemplo, se podría aplicar un filtro de convolución para resaltar los bordes de un objeto de un color determinado. para poder desarrollar esta practica necesitamos tener un buen entendimiento sobre filtros , las operaciones realizadas,un manejo de matlab decente, además de tener conocimiento de control junto con visión artificial.por este motivo considero importante presentar un marco teórico de los conceptos tomados en esta practica para que cualquier lector logre entender de una manera más clara como se esta realizando esta practica.

pasos para realizar esta practica

1. conectar arduino con matlab
2. creas un objeto con tu arduino
3. declaramos los pines que vamos a utilizar para los servomotores
4. establecemos un valor inicial a los ángulos
5. damos un objeto de cámara y definimos su resolución
6. capturamos una imagen de inicio
7. abrimos vídeo
8. Capturamos el objeto que vamos a seguir
9. realizamos segmentación
10. creamos y aplicamos un filtro pasa bajas
11. desarrollamos código para que pueda calcular los errores de posición y mover la cámara de acuerdo a donde
12. se encuentre el objeto
13. mostramos resultados

1.2 Marco teórico

1.2.1 Visión artificial y visión computacional

¿que es visión artificial?
La visión artificial es una parte de la inteligencia artificial (IA) . Se centra en el desarrollo y el perfeccionamiento de técnicas que permiten a las máquinas ver, identificar y procesar imágenes de la misma manera que lo hace la visión humana. Un sistema de visión artificial es una combinación de hardware y software que tiene la capacidad de capturar y procesar datos de imágenes Los sistemas de visión artificial suelen estar compuestos por un conjunto de sensores digitales insertados en cámaras industriales capaces de ofrecer imágenes y datos. El software puede procesar, analizar y medir diferentes datos que sirven a los ingenieros para supervisar los procesos y tomar decisiones acertadas. Estos sistemas son uno de los recursos tecnológicos industriales en los que se han desarrollado un mayor número de avances en los últimos años.

¿que es la visión computacional?
La visión computacional es una de las ramas de la inteligencia artificial y el machine learning que busca replicar la capacidad de visión humana, mediante la enseñanza de patrones en algoritmos almacenados en software. El objetivo de la visión computacional no solo se centra en imitar la vista, sino en tener la percepción de lo que se ve y comprender asociando el sentido que le damos los humanos a lo que vemos. El procesamiento de la visión computacional se realiza mediante píxeles en imágenes y vídeos, para lo cuál se enseñan miles de imágenes relacionadas con etiquetas para enseñarle al modelo lo que queremos que aprenda, desarrollando así mediante el entrenamiento la creación de patrones individuales que luego puedan seguir perfeccionándose con mas datos y una vez realizado el proceso de aprendizaje cuando se esta en pruebas con imágenes y vídeos nuevos o diferentes a las de test pueda aplicar lo aprendido.

Mucha gente suele confundir estas dos ramas y pensar que son sinónimos sin embargo existen ciertas diferencias entre estas dos,la visión computacional se enfoca en la adquisición, procesamiento y análisis de imágenes y vídeos por computadora, con el objetivo de extraer información útil y significativa de estas señales visuales. En otras palabras, se trata de utilizar la tecnología informática para entender y trabajar con las imágenes y vídeos. Por otro lado, la visión artificial es un término más amplio que abarca no solo la visión por computadora, sino también otros tipos de sensores,

como sensores infrarrojos o láser, y su aplicación en campos como la robótica o la vigilancia. La visión artificial se enfoca en desarrollar sistemas que puedan percibir y comprender el mundo físico de la misma manera que lo hacen los seres humanos.

estas ramas en la actualidad aplican muchos algoritmos modernos y redes neuronales sin embargo años atrás dependían completamente de los filtros y transformadas tema que se hablará más adelante por que gracias a esto podemos hacer maravillas con la información obtenida y manipularla de tal manera que nos permita obtener la información que deseamos tener y depurar de alguna manera la información no deseada o requerida

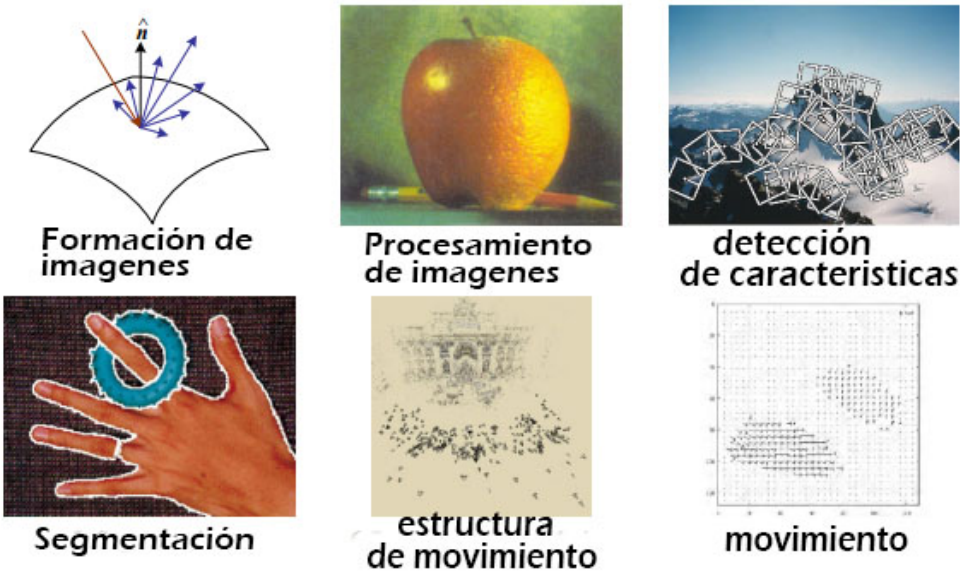


Figure 1: Procedimientos fundamentales .

En la formulación y resolución de problemas de visión por computadora, a menudo es útil inspirarse en tres enfoques de alto nivel: para aplicar estos conocimientos se necesita tener tres enfoques

• Científico: construir modelos detallados del proceso de formación de imágenes y desarrollar técnicas matemáticas para invertirlos y recuperar las cantidades de interés (si es necesario, haciendo suposiciones simplificadoras para hacer las matemáticas más manejables).

• Estadístico: utilizar modelos probabilísticos para cuantificar la probabilidad previa de las incógnitas y los procesos de medición ruidosos que producen las imágenes de entrada, luego inferir las mejores estimaciones posibles de las cantidades deseadas y analizar sus incertidumbres resultantes. Los algoritmos de inferencia utilizados a menudo están estrechamente relacionados con las técnicas de optimización utilizadas para invertir los procesos de formación de imágenes científicas.

• Ingeniería: desarrollar técnicas que sean simples de describir e implementar, pero que también se sabe que funcionan bien en la práctica. Pruebe estas técnicas para comprender sus limitaciones y modos de falla, así como sus costos computacionales esperados (rendimiento en tiempo de ejecución).

1.2.2 Convolución

En matemáticas, y en particular análisis funcional, una convolución es un operador matemático que transforma dos funciones f y g en una tercera función que en cierto sentido representa la magnitud en la que se superponen f y una versión trasladada e invertida de g. Una convolución es un tipo muy general de media móvil, como se puede observar si una de las funciones se toma como la función característica de un intervalo. en palabras simples podemos decir que la convolución es una operación matemática que hace la integral del producto de 2 funciones(señales), con una de las señales volteada

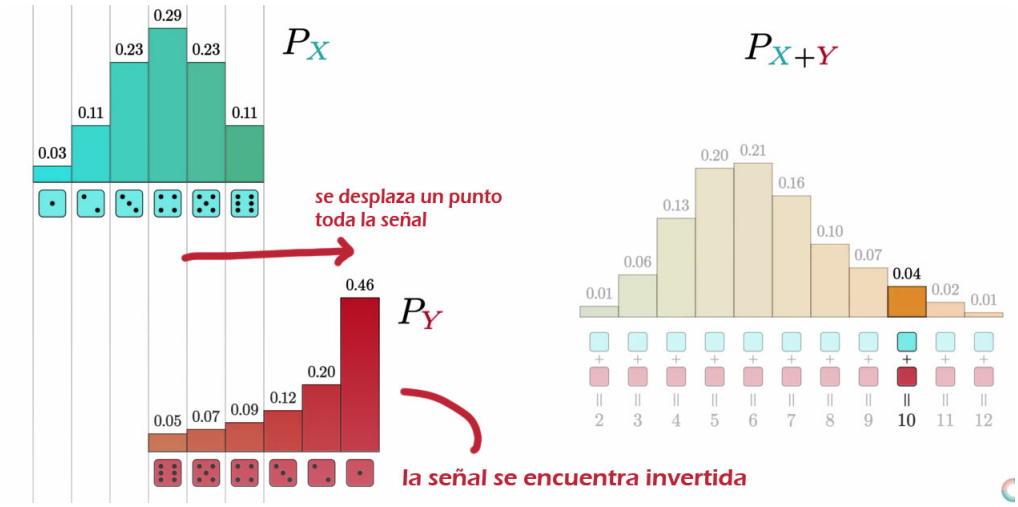


Figure 2: Representación gráfica de la convolución

explicando la representación gráfica de la figura 2 de como se hace una operación en la convolución esta operación es una combinación de dos funciones o conjunto de datos que darán como resultado una tercera función que representara como se están superponiendo o mezclando entre sí. Un ejemplo práctico es una señal de entrada como un audio o una imagen y una función como lo son los filtros, esta operación combina estos dos para crear una nueva señal que tendrá las características del filtro aplicado a la señal de audio en este caso a la señal de entrada

1.2.3 Filtros

para aplicar un filtro necesitamos la convolución pero la pregunta es ¿qué es un filtro?
Un filtro digital es una herramienta que se utiliza para procesar señales digitales. Se trata de una operación matemática que toma una secuencia de números (la señal de entrada) y la transforma en otra secuencia de números (la señal de salida), con el propósito de resaltar o atenuar determinadas características. El objetivo principal de un filtro digital es modificar una señal digital de entrada de manera controlada. Puede resaltar componentes de frecuencia específicas, eliminar ruido no deseado o suavizar la señal. Esto se logra mediante la aplicación de algoritmos y técnicas de procesamiento de señales. Al utilizar un filtro digital, se pueden obtener resultados precisos y repetibles, ya que se basa en operaciones matemáticas exactas. Además, ofrece flexibilidad, ya que los parámetros del filtro se pueden ajustar según las necesidades específicas de la aplicación.

filtros más comunes

1. Filtros pasa bajas: la información que permanece son las frecuencias más bajas y atenúan o eliminan las frecuencias más altas. suelen utilizarse para suaviza o quitar ruido de alta frecuencia

2. Filtros pasa altas: permanecen las frecuencias altas, disminuyen las frecuencias más bajas. son muy utilizados para destacar características de alta frecuencia y eliminar ruido de baja frecuencia en visión computacional se usa para detectar bordes

3. Filtros pasa bandas: pasan un rango específico de frecuencias y atenúan las frecuencias fuera de ese rango.

4. Filtros rechaza bandas: funcionan de manera inversa a los pasa banda donde anulan un rango especifico de señales indicadas y las que no pasan

5. Filtros adaptativos: Estos filtros se ajustan automáticamente para adaptarse a las características cambiantes de la señal de entrada. Son utilizados en aplicaciones como cancelación de eco o reducción de ruido adaptativa.

1.2.4 Transformada de fourier

La transformada de Fourier es una operación matemática fundamental para algunas disciplinas como las telecomunicaciones o la física. Sin ella no existirían las telecomunicaciones modernas, no solo Internet o la telefonía móvil, sino la propia telefonía convencional, que no habría podido evolucionar más allá de una forma de comunicación local y no habrían existido las llamadas de larga distancia.

$$F[f(t)] = F(\omega) = \hat{f}(\omega) = \int_{-\infty}^{\infty} f(t) e^{-i\omega t} dt$$

Figure 3: Transformada de fourier

$$F^{-1}[F(\omega)] = f(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(\omega) e^{i\omega t} d\omega$$

Figure 4: Transformada de fourier inversa

Aunque esta operación matemática figura 3 debe su nombre al matemático Joseph Fourier, lo cierto es que muchos han contribuido a su invención, entre ellos Euler, Bernoulli, Lagrange y Gauss. Fourier tuvo un papel esencial, al inventar las series de Fourier, donde una función periódica se podía descomponer en la suma de funciones trigonométricas. La transformada de Fourier generaliza este concepto.

se usa esta herramienta matemática para analizar señales y descomponerlas en sus distintas componentes de frecuencia, prácticamente lo que realiza es tomar una función o una señal en el dominio del tiempo y cambia al plano de la frecuencia un ejemplo que podemos tomar de una aplicación es el siguiente tienes una canción y su transformada te permitirá analizar esta canción y brindarte la información sobre qué notas musicales se encuentran, en que frecuencias están y que tan fuerte se escuchan estas. es como si fragmentaras la canción en distintos componentes por cierto, es importante recordar que la transformada de Fourier no solo se aplica a señales periódicas, sino también a señales no periódicas. Esto significa que puedes analizar cualquier tipo de señal, ya sea un sonido, una imagen o incluso datos numéricos, y obtener los datos sobre las frecuencias que la componen

es importante a la hora de hacer su análisis conocer estos conceptos

1. Amplitud: es lo alta o baja que es
2. La frecuencia: es el ritmo que lleva. O es el número de ondulaciones por segundo
3. El desfase: es cuando empieza la onda.

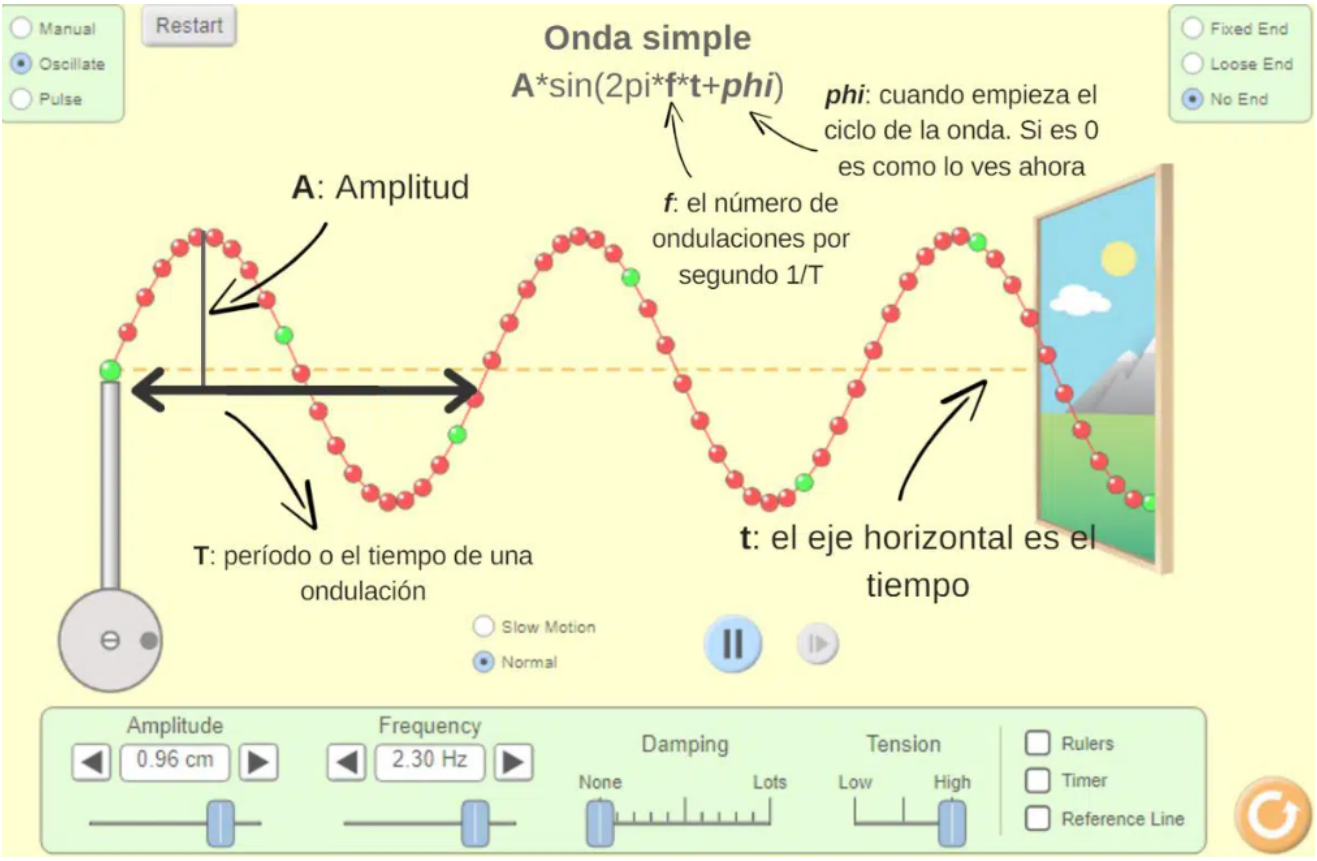


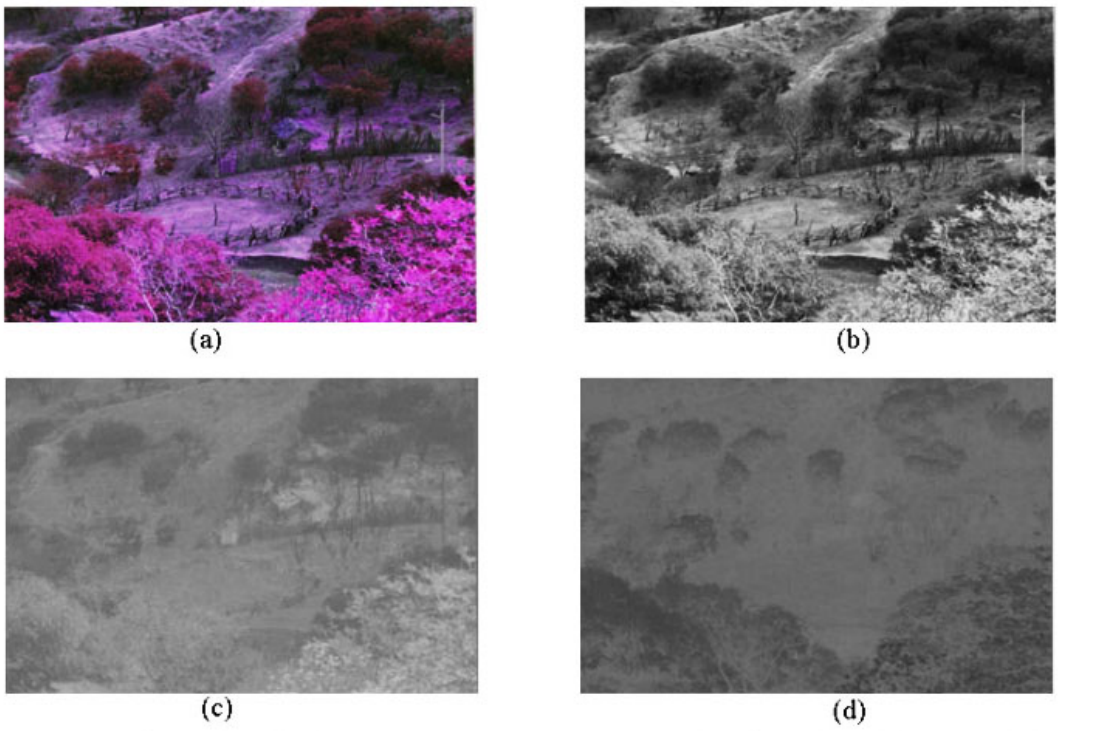
Figure 5: Onda sinusoidal

¿Cual es la diferencia entre la transformada de fourier y la transformada rápida de fourier?

La transformada de Fourier(TF) y su transformada rápida(TFF o FFT), por sus siglas en ingles fast Fourier transform están familiarmente relacionadas, pero son distintas en la forma que se realiza su cambio de plano(su calculo) y su eficiencia computacional La transformada rápida de Fourier es un algoritmo especifico que permite calcular de una manera eficaz su transformada , este algoritmo es una versión optimizada que aprovecha su estructura matemática de la transformada para disminuir su cantidad de operaciones que necesita realizar , en lugar de hacer todos los cálculos ,FFT divide la señal en segmentos más pequeños y utiliza la recursividad en sus cálculos para estos segmentos , y como bien sabemos que la recursividad es una herramienta computacional de eficiencia reduce radicalmente cuantas operaciones se deben hacer y por ende los cálculo son mucho más rápidos.

1.2.5 Segmentación de colores

La segmentación de color se enfoca en dividir una imagen en distintas áreas basándose en las propiedades de cada píxel que por lo general tienen distintas propiedades de color que básicamente quiere decir que son elementos diferentes que hacen variar a un determinado color , este método se utiliza mucho en el procesamiento de imágenes y visión artificial con la finalidad de extraer un objeto en especial o separar ciertas regiones las cuales se tenga a partir de su color ,aquí se definen umbrales que se utilizan para identificar y separar objetos de su fondo , estos se pueden establecer de manera manual o automática todo depende del algoritmo que se utilice y como se analiza el histograma del color de la imagen



(a) Es la imagen original de color. (b) Componente Y, esta componente, contiene información sobre los tres canales de color RGB. (c) Crominancia I. (d) Crominancia Q. Estas imágenes resultan de la transformación NTSC.

Figure 6: Segmentación

histograma: Un histograma es la representación gráfica en forma de barras, que simboliza la distribución de un conjunto de datos. Sirven para obtener una "primera vista" general, o panorama, de la distribución de la población, o de la muestra, respecto a una característica, cuantitativa y continua.

este método tiene bastantes aplicaciones la principal es la detección de objetos ,seguimiento de objetos en movimiento y reconocimiento de patrones

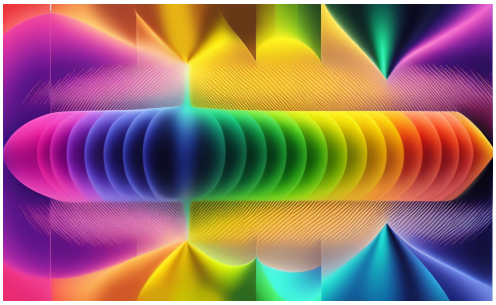


Figure 7: variación de colores

es importante tener en cuenta que hay una gran variedad de colores figura 7 y estos colores que se muestran no son todos en realidad la gama y el rango de variaciones es muy grande por esto hay distintos modelos colores que tratan de representar un mapeo de un rango de colores entre estos modelos esta RGB,HSV y CYMK

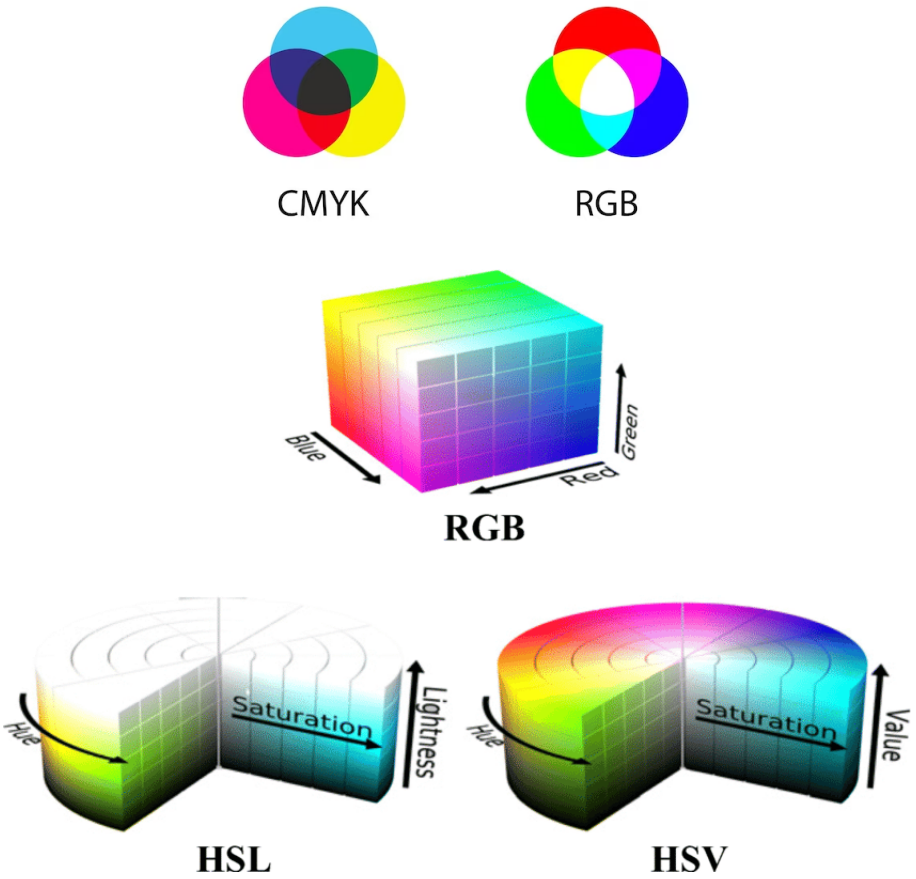


Figure 8: Modelos de colores

1.2.6 Fenómeno de Gibbs

El fenómeno de Gibbs es una consecuencia del truncamiento de la respuesta impulso de un filtro realizable. Este fenómeno se caracteriza por la aparición de oscilaciones en el dominio de la frecuencia. Cuanto mayor sea el truncamiento, más oscilaciones de Gibbs se presentarán en el dominio de la frecuencia. Sin embargo, se puede minimizar el fenómeno de Gibbs al aumentar la cantidad de muestras tomadas (mayor M + 1).
Cuando la función que se está desarrollando en Serie de Fourier tiene discontinuidades (señales de variación rápida) no es posible que haya una buena convergencia en los entornos de las mismas.
En tales entornos, las sumas parciales muestran tanto sobrevalores como subvalores alrededor del valor real de la función, que pueden llegar a un 18% del salto en la discontinuidad.

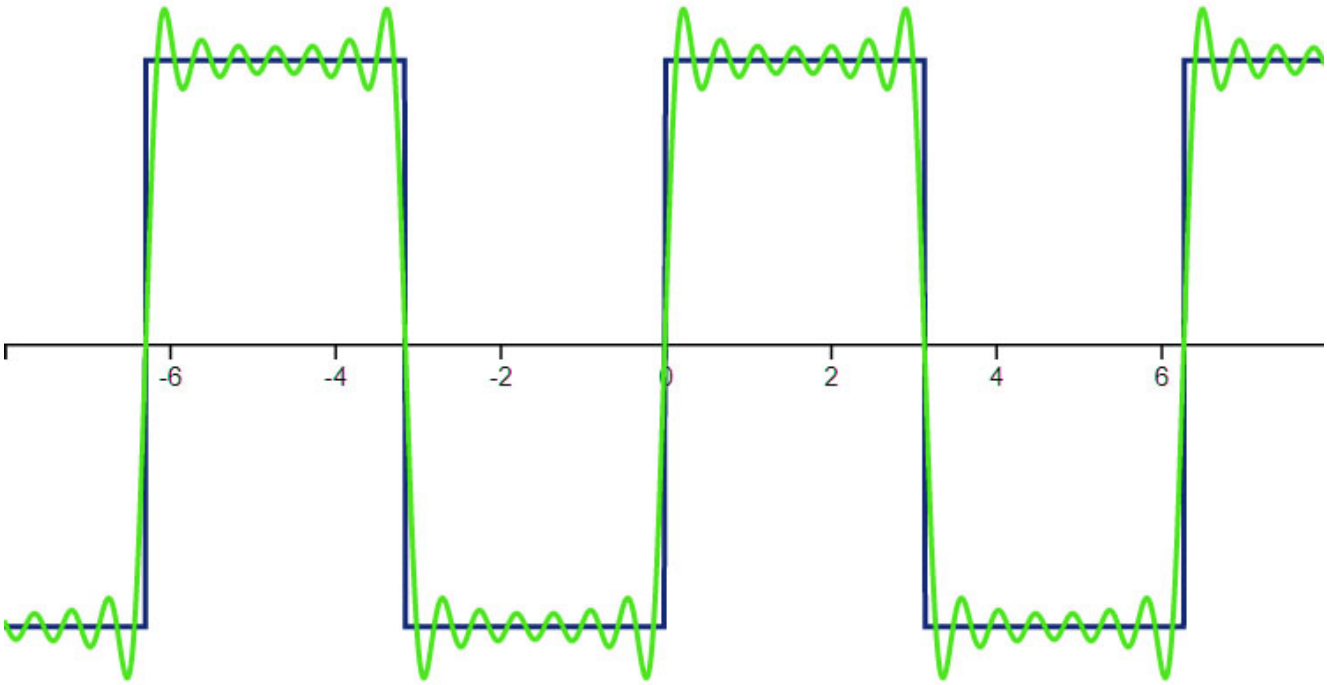


Figure 9: Representación gráfica del fenómeno de Gibbs

Las condiciones de Dirichlet sugieren que las señales discontinuas pueden tener una representación de la Serie de Fourier siempre que haya un número finito de discontinuidades. Esto parece contrario a la intuición, sin embargo, ya que los exponenciales complejos son funciones continuas. No parece posible reconstruir exactamente una función discontinua a partir de un conjunto de continuas. De hecho, no lo es. No obstante, puede ser si relajamos la condición de 'exactamente' y la reemplazamos con la idea de 'casi en todas partes'. Es decir que la reconstrucción es exactamente la misma que la señal original excepto en un número finito de puntos. Estos puntos, no necesariamente de manera sorprendente, ocurren en los puntos de discontinuidades.

2 Desarrollo

Para el desarrollo de nuestro código primero crearemos el objeto de arduino para poder tener comunicación con el desde matlab para esto necesitamos instalar la siguiente toolbox figura 10 .

Figure 10: Matlab Toolbox arduino

esta librería nos permitirá poder controlar al Arduino a partir de los scripts de Matlab haciendo que lo podamos programar directamente , el código siguiente además de crear el objeto de arduino que nos permite poder declarar sus entradas y salidas, nos permite también como estas se van a comportar desde que pueden ser pwm,digitales,continuas o en este caso se hacen servo

Inicializar arduino

```
a = arduino('COM7', 'Mega2560', 'Libraries', 'Servo');  % creamos el
    objeto arduino
servo_x = servo(a,'D7'); % creamos el servo x que estara en el
   puerto digital 7
servo_y = servo(a,'D8'); % creamos el servo y que estara en el
   puerto digital 8
```

Una vez viendo que este paso se hizo correctamente procedemos a armar nuestra base de la cámara para que pueda moverse y seguir el objeto, nosotros decidimos armarla basto con comprar dos placas y añadirle dos servos a cada placa, para el movimiento en x el servo es 110 y para el movimiento en y es 100 es importante colocar de manera correcta los servos ya que si desde la estructura partimos mal complicará a la hora de controlarlo ,también es recomendable establecer una condición inicial de 90 grados en cada eje para que sean ortogonales entre sí estos mismos (podemos ver la carcasa en la siguiente pagina figura 11)

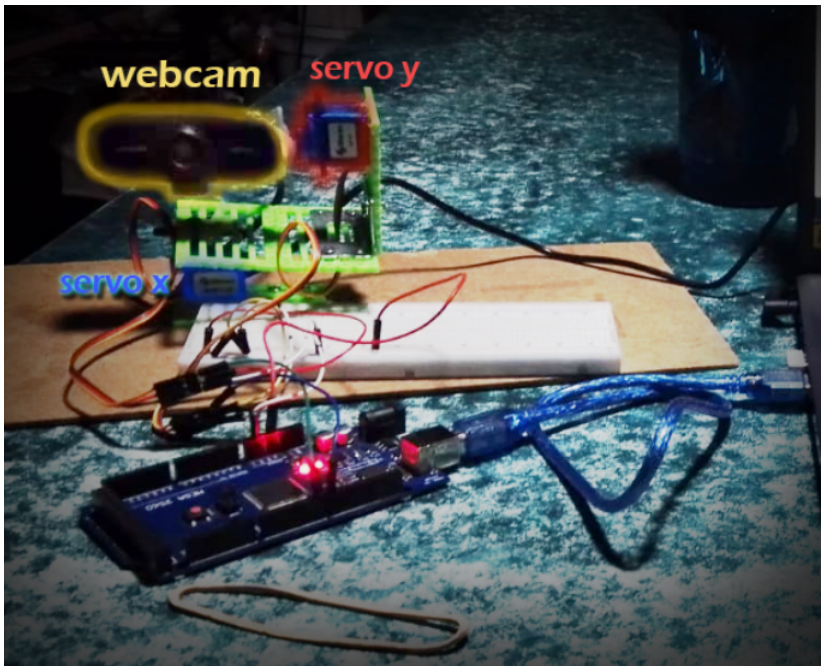


Figure 11: Estructura de cámara

para poder inicializar la camara,tomar fotos y video necesitamos descargar la siguiente toolbox

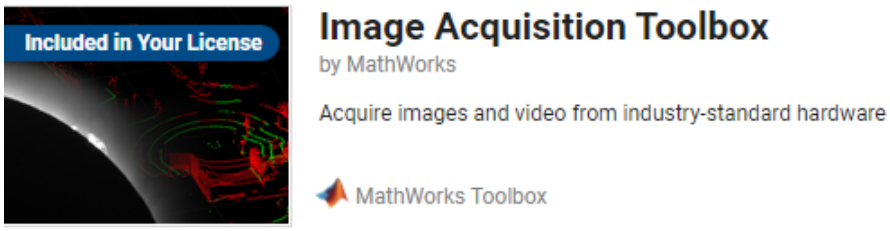


Figure 12: Toolbox para cámaras

Inicializar cámara(explicación)

entonces mandamos la cámara a una posición inicial ,declaramos dos vectores para que almacenen el error para poderlo graficar después, el crear el objeto de cámara en el código nos permitirá poder utilizar las funciones que ofrece esta tolbox sobre nuestra cámara como snapshot que nos permite capturar una foto también podremos decidir qué resolución utilizar que en este caso nos interesa utilizar una resolución baja ya que no nos importa el detalle si no que logre diferenciar un color esto se hace porque mientras más resolución tenga la imagen más costo computacional tendrá y será más lento en el procesamiento.

Después tomamos una foto inicial,obtenemos la medida de la imagen,comenzamos video y abrimos una ventana que nos permita ver que esta capturando la camara , las medidas serán importantes más adelante a la hora de aplicar los filtros por lo que no se deben dejar de lado.

Inicializar cámara(parte de código)

```
angulo_x = 90 %posicion inicial x;
angulo_y = 90 %posicion inicial y;

vx_error=[];
vy_error =[];
v_iteracio = [];
iteracion = 1;

% creacion de los objetos
cam = webcam(); % damos un objeto camara
cam.Resolution='720x480';
foto_objeto = snapshot(cam); %tomamos un imagen inicial
frame_size = size(foto_objeto); %obtenemos sus medidas
videoPlayer = vision.VideoPlayer('Position',[10 30 [frame_size(2),
   frame_size(1)]]); %creamos un visualizador de video
% capturamos la imagen de nuestro objeto a seguir
runloop=true; %creamos un control para el primer video en la
   capturadel objeto
while runloop
    foto_objeto=snapshot(cam); %tomamos la foto de nuestro objeto
    step(videoPlayer,foto_objeto); % al mostramos en nuestro video
    runloop = isOpen(videoPlayer);
end
```

Control y filtrado (explicación)

Primero debemos tener en cuenta que se aplicará un control PID y que hay distintas maneras de obtener la integral y derivada del error a la hora de implementarla en un algoritmo en este caso la integral es una suma del anterior y el siguiente mientras que la función derivativa es una diferencia

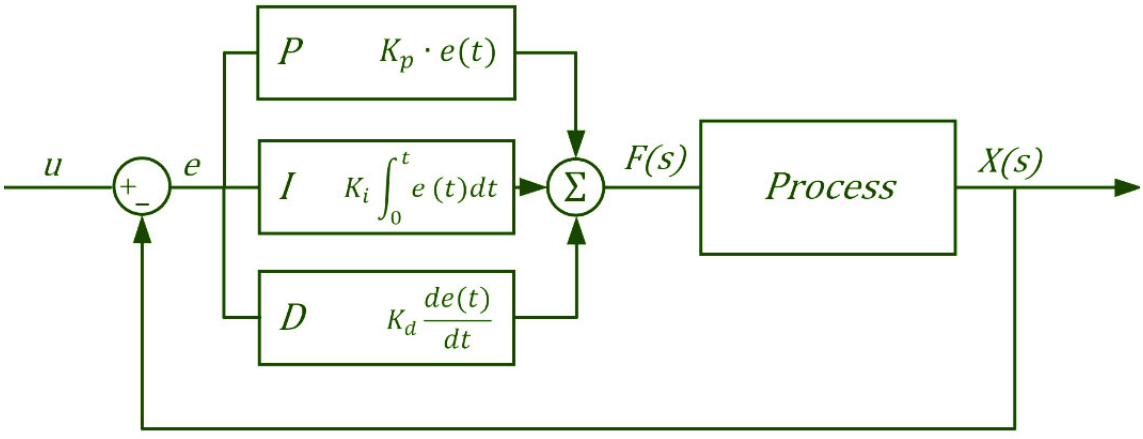


Figure 13: Controlador PID

la sintonización de las ganancias fue a prueba y error , se le asignaron diferentes ganancias a x y y

Ganancias

```
%Ganancias en x

kp_x = 0.33; % 0.33
kd_x = 0.0019; % 0.15
ki_x = 0.00085;  % 0.11
x_derivada = 0;
cx_anterior = set_x;
x_integral = 0;
%
   ------------------------------------------------------------------

%ganancias en y

% inicializacion de las ganancias en y
kp_y = 0.19; %0.19
ki_y = 0.00065; % 0.0006
kd_y = 0.0016;% 0.0012
y_derivada = 0;
y_integral = 0;
cy_anterior = set_y;
```

procesamiento de imagen ,filtrado y control(explicación)

primero procedemos a transformar en double la foto ya que double proporciona la capacidad de tener magnitudes más pequeñas o más grandes y esto nos viene bien a la hora de realizar las sumás para después esto promediarlo con roipoly , el double evita que cantidades muy pequeñas o muy grandes sean redondeadas o truncadas,es importante tener en cuenta que esta función roipoly sacará un promedio con las tres capaz una vez obtenido este promedio se comenzará a realizar una búsqueda donde se hará en cada capa la capa red,green y blue donde obtendremos sus referencias que serán los pixeles deseados que la camara seguirá, implementamos la transformada rápida de Fourier ya que como explicamos al inicio este algoritmo es menos costoso computacionalmente hablando, después procedemos a crear un filtro pasa bajas para suavizar los bordes ya que nuestro interés esta en la segmentación del color no en tener todo a detalle , una vez aplicado el filtro se regresa al espacio correspondiente y procedemos a buscar el color esta búsqueda nos proporcionara una mascara que ya aplicándola al frame original nos dará el objeto ya teniendo este objeto basta con calcular el centroide ya que este punto será donde la cámara estará enfocada para seguir el objeto y en ese punto se realizarán los cálculos del error en los dos ejes estos errores los mandamos a una función que modificara el alguno y la velocidad de cambio de acuerdo que tanto error se tenga

```
% procesamos la imagen de nuesro obejto para obtener los colores
foto_objeto = im2double(foto_objeto); % convertimos a double para
   poder reliazar las sumas
seleccion = roipoly(foto_objeto); % seleccionamos el color del
   objeto


% obtenemos las referencias en rgb
ref_r= sum(sum(foto_objeto(:,:,1).*seleccion))/sum(seleccion(:));  %
   obtenemos la referencia en red
ref_g= sum(sum(foto_objeto(:,:,2).*seleccion))/sum(seleccion(:)); %
   obtemos la referencia en green
ref_b= sum(sum(foto_objeto(:,:,3).*seleccion))/sum(seleccion(:)); %
   obtenemos la referencia en blue
close all % cerramos todo
clear cam

% %% creacion de filtro pasa bajas
m = frame_size(1); %esto es y
n=frame_size(2); %esto es x
% %creamos el filtro pasa bajas para suavizar
filtro=zeros(m,n);
sigma=0.04;
for y=1:m
        dy=(y-m/2)/(m/2);
        for x=1:n
            dx=(x-n/2)/(n/2);
            dxy=sqrt(dx^2+dy^2);
            filtro(y,x)=exp(-(dxy^2)/(2*sigma^2));
        end
end

% busqueda de color
% creacion nueva de los objetos
cam2 = webcam();
cam2.Resolution = '720x480';
frame = snapshot(cam2);
frame = im2double(frame);

frame_size = size(frame);
videoPlayer = vision.VideoPlayer('Position',[10 30 [frame_size(2),
   frame_size(1)]]); %creamos un visualizador de video
runloop=true; %creamos un control para el primer video en la
   capturadel objeto

umbral=30/255;
writePosition(servo_x,angulo_x/180);
writePosition(servo_y,angulo_y/180);
```

```
% Declaramos nuestras ganancias de control
set_x = frame_size(2)/2;
set_y = frame_size(1)/2;
profundidad = frame_size(3);


kp_x = 0.33; % 0.33
kd_x = 0.0019; % 0.15
ki_x = 0.00085;  % 0.11
x_derivada = 0;
cx_anterior = set_x;
x_integral = 0;

% inicializacion de las ganancias en y
kp_y = 0.19; %0.19
ki_y = 0.00065; % 0.0006
kd_y = 0.0016;% 0.0012
y_derivada = 0;
y_integral = 0;
cy_anterior = set_y;


while runloop
    frame=snapshot(cam2); %tomamos la foto de nuestro objeto
    frame = im2double(frame); % cambiamos el frame a double para
       realizar los calculos
    frame=frame+.20*randn(m,n,3);
    frame2=frame;

    for z=1:profundidad
        %pasamos la imagen a la frecuencia
        frame_f(:,:,z) = fftshift(fft2(frame(:,:,z)));
        %filtramos
        frame_ff(:,:,z) = filtro.*frame_f(:,:,z);
        %regresamos al espacio
        frame(:,:,z) = ifft2(ifftshift(frame_ff(:,:,z)));
    end

    %realizamos la busqueda
    b_r= frame(:,:,1)>ref_r-umbral & frame(:,:,1)<ref_r+umbral;
    b_g= frame(:,:,2)>ref_g-umbral & frame(:,:,2)<ref_g+umbral;
    b_b= frame(:,:,3)>ref_b-umbral & frame(:,:,3)<ref_b+umbral;
    busqueda = b_r.*b_g.*b_b;
    busqueda = medfilt2(busqueda);

    for i = 1:3
        frame_encontrado(:,:,i) = frame(:,:,i).*busqueda;
    end
```

```
%calculos de los momento sobre la busqueda para encontrar el
   centroide
m_00 = momentos(busqueda,0,0);
m_01 = momentos(busqueda,0,1);
m_10 = momentos(busqueda,1,0);
c_x=m_10/m_00;
c_y=m_01/m_00;

x_error = set_x - c_x;
y_error= set_y - c_y;

if (isnan(x_error))
    angulo_x=90;
    cx_anterior = set_x;
else
    if (x_error <= -20) || (x_error >= 20)
        if x_error < 0
            paso_x = -(abs(x_error)*10)/(124);
        else
            paso_x = (abs(x_error)*10)/(124);
        end
        x_integral = x_integral + paso_x;  %actualizamos la
           integral del error en grados
        x_derivada = c_x - cx_anterior;  %calculamos la derivada
            del error
        x_derivada = x_derivada*10/124; %convertimos o
           normalizamos
        angulo_x = angulo_x + kp_x*paso_x + kd_x*x_derivada +
           ki_x*x_integral;  %calculamos la senal de control
        cx_anterior = c_x;  %actualizamos la varible
        vx_error(iteracion-1) = x_error;
    end
end
writePosition(servo_x,angulo_x/180); %escribimos el nuevo angulo
    calculado



if (isnan(y_error))
    angulo_y =90;
    cy_anterior = set_y;
else
    if (y_error <= -14) || (y_error >= 14 )
        if y_error < 0
            paso_y = -((10*abs(y_error))/89);
        else
            paso_y = ((10*abs(y_error))/89);
        end
```

```
            y_integral = y_integral+paso_y;
            y_derivada = c_y - cy_anterior;  %calculamos la derivada
                del error
            y_derivada = y_derivada*10/89; %convertimos o
               normalizamos
            angulo_y = angulo_y + kp_y*paso_y + kd_y*y_derivada+ki_y
               *y_integral; %calculamos la senal de control
            cy_anterior = c_y;  %actualizamos la varible
            vy_error(iteracion-1) = y_error;
        end
    end
    writePosition(servo_y,angulo_y/180);

    v_iteracio(iteracion) = iteracion;
    iteracion = iteracion+1;


    step(videoPlayer,real(frame_encontrado)) % al mostramos en
       nuestro video
    runloop = isOpen(videoPlayer);
    if runloop == false
        clear cam2 a
    end
end
v_iteracio = 0:1:size(length(vy_error))
figure
hold on
plot(vx_error);

figure
hold on
plot(vy_error);
```

3 Resultados

3.1 Resultados de video

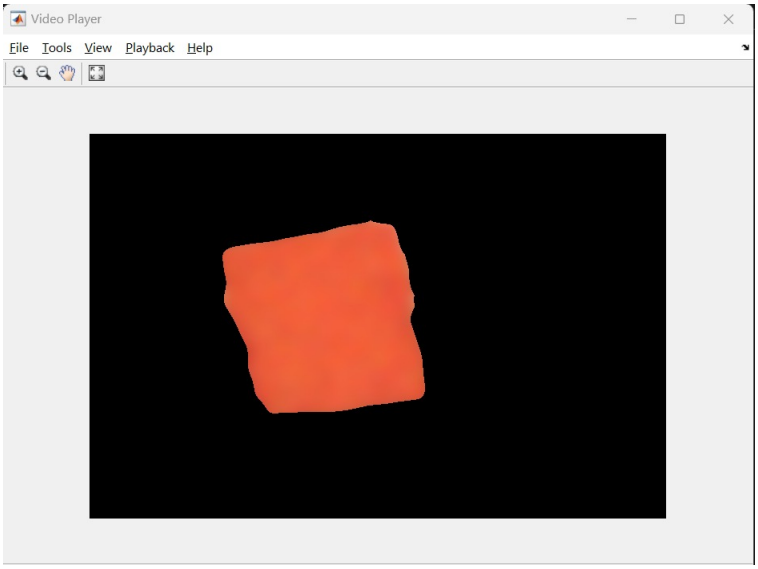


Figure 14: Detección de objeto

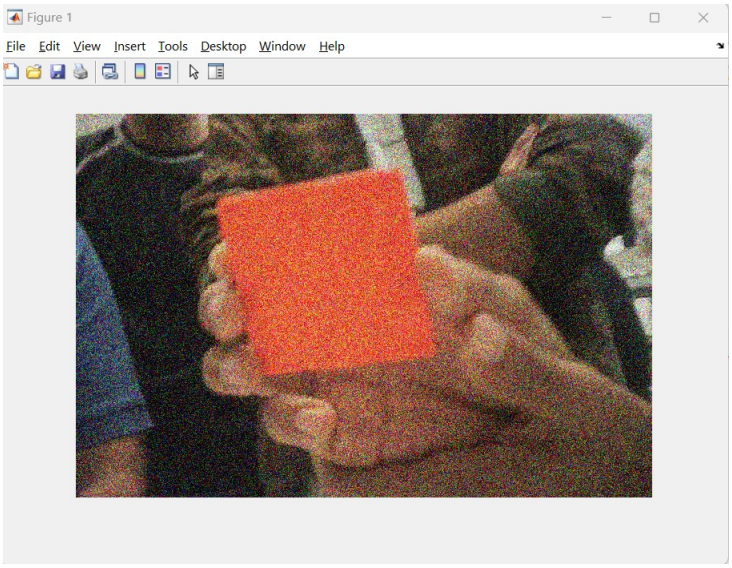


Figure 15: Aplicacion de ruido

decidimos utilizar un postick diferente al que conocemos era como tipo carton ya que los de papel reflejaban demasiado la luz y esto afectaba bastante la detección del objeto

3.2 Graficas de error en X y Y

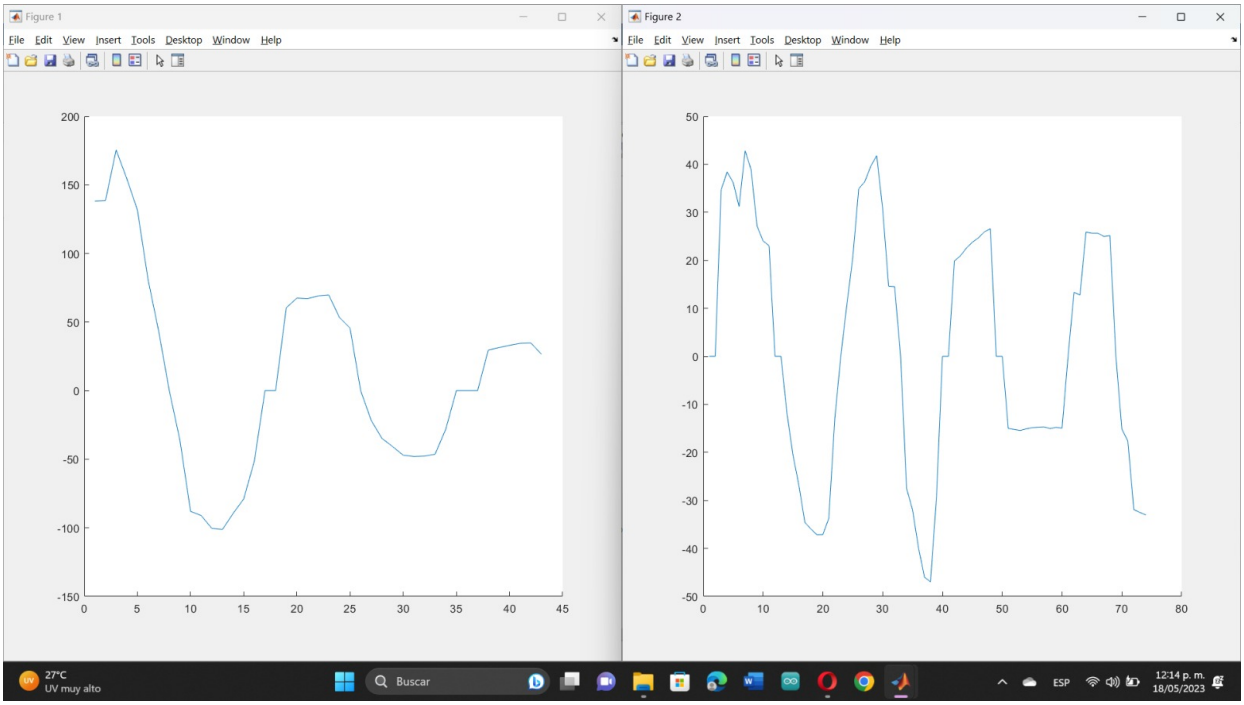


Figure 16: Graficas de error uno

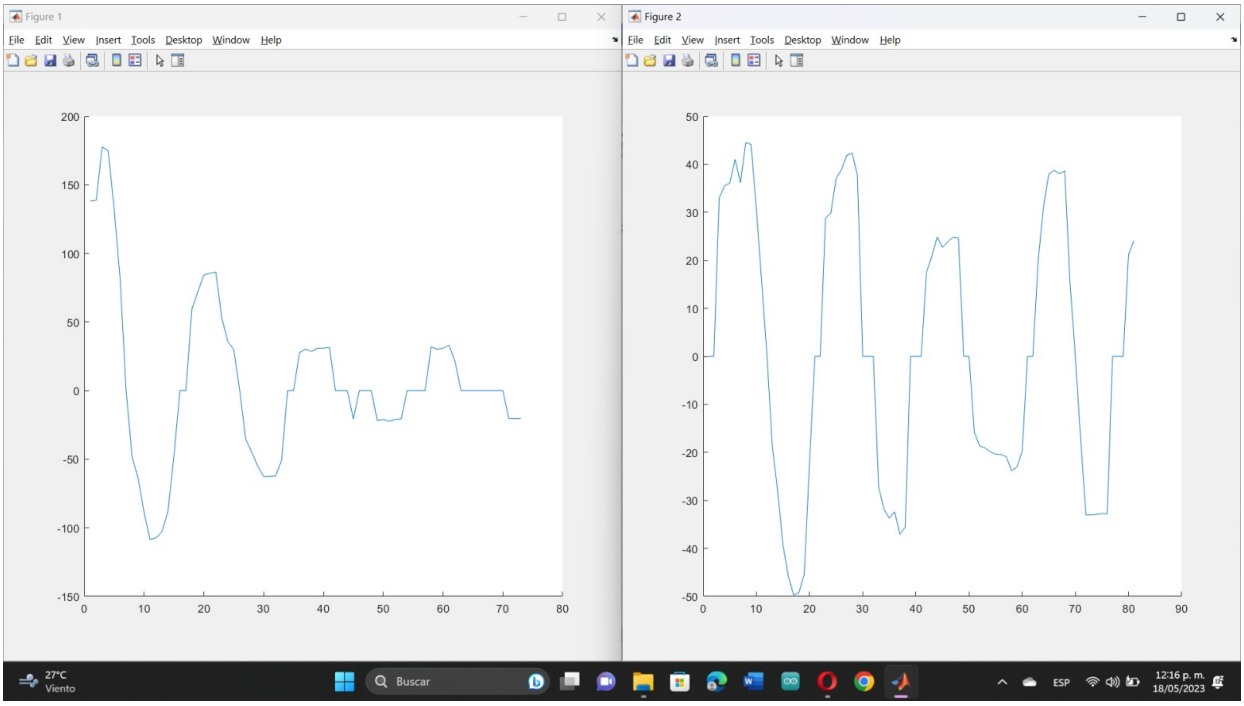


Figure 17: Graficas de error dos

4 Conclusiones

Nos dimos cuenta que la rezón por la que la cámara vibraba tanto era por histéresis que representa en sí la activación y desactivación constante por lo que le añadimos un margen de respuesta con una condicional para así evitar que estuviera moviéndose así constantemente también tuvimos que cambiar de motoreductores a servo por que los primeros a pesar de que facilitaron calculos después se empezo a complicar por lo que optamos por cambiarlos

5 Referencias

https://la.mathworks.com/matlabcentral/fileexchange/47522-matlab-support-package-for-arduino-h
https://la.mathworks.com/products/image-acquisition.html
http://profesores.elo.utfsm.cl/~mzanartu/ELO313/Docs/ELO313_2012_FFT.pdf
https://azimadli.com/vibman-spanish/latransformadarpidadefourier.htm
https://www.uv.es/soriae/tema_5_pds.pdf
https://la.mathworks.com/help/images/color-based-segmentation-using-k-means-clustering.
html
http://bibliotecavirtual.dgb.umich.mx:8083/xmlui/handle/DGB_UMICH/3540
https://atlas-robots.com/que-es-la-vision-artificial/
https://la.mathworks.com/discovery/convolution.html
https://datascience.eu/es/matematica-y-estadistica/convolucion/
https://www.youtube.com/watch?v=FeZnQdgrX9c&ab_channel=DomingoMery
https://biblus.us.es/bibing/proyectos/abreproy/70077/fichero/capitulo4.pdf
https://clasesdemecanica.net/index.php/centroides/
https://uapas1.bunam.unam.mx/matematicas/funcion_senoidal_amplitud/