Instituto Politécnico Nacional
Unidad Profesional Interdisciplinaria en Ingeniería
y Tecnologías Avanzadas

SVAPF

Proyecto final

Sistemas de Visión Artificial

- García Olivares Carlos
- Hernández Orozco Cesar Gibran
- Hernández Rubio Sandra Stephany

- Profesor: Huitrón Ramírez Erick
- Grupo: 4MV6
- Fecha de entrega: 16 de enero del 2025

Introducción

En el proyecto solicitado, se desarrolló un sistema de visión artificial con actuadores, que combina múltiples técnicas, y algoritmos los cuales fueron usados para la clasificación de los productos. El sistema se diseñó mediante los conocimientos adquiridos a lo largo del curso, implementando los métodos de RGB y HSV, la firma de la imagen y los momentos invariantes de Hu para la búsqueda de patrones.

La detección de colores nos permitió clasificar los objetos en cinco colores predeterminados, así como el modelo HSV, el cual se encarga de la segmentación de imágenes, para detectar los colores de cada objeto. Para el caso de la identificación y confiabilidad de la forma de nuestro objeto se implementó la firma de la imagen, la cual nos permitió evaluar la similitud entre el contorno seleccionado y la plantilla ideal, a su vez, empleando los momentos invariantes de Hu, permitieron clasificar la pieza en las tres categorías ya establecidas anteriormente.

El proyecto también hizo uso de bases de datos para almacenar las plantillas y los resultados de los análisis, incluyendo también la creación de informes en Excel, los cuales documentarían las características de cada objeto que ha pasado por el análisis del sistema.
Por último, también se hizo uso de actuadores para poder separar los productos dañados de los correctos.

Código

```
clc,
close all,
clear;

% Ruta de la base de datos---------------------------------------
RUTA = "C:\Users\HP\Downloads\SVAPF\Base datos";
cam = webcam(1);
r=0;
g=0;
b=0;
%umbral
umbral=75;
% Definir objetos
dir_plantilla1="C:\Users\HP\Downloads\SVAPF\Plantillas\Plantilla_I.jpg";
dir_plantilla2="C:\Users\HP\Downloads\SVAPF\Plantillas\Plantilla_Pinza.jpg";
dir_plantilla3="C:\Users\HP\Downloads\SVAPF\Plantillas\Plantilla_X.jpg";

Producto1 = "Letra I";
Plantilla1 = imread(dir_plantilla1);
Producto2 = "Pinza";
Plantilla2 = imread(dir_plantilla2);
Producto3 = "Letra X";
Plantilla3 = imread(dir_plantilla3);

% Comunicación con Arduino----------------------------------------
arduinoPort = 'COM7';
% baudRate = 9600;
% arduino = serialport(arduinoPort, baudRate);
a = serialport(arduinoPort,115200);
```

```
% Definir colores----------------------------
colores = {'Rojo', 'Verde', 'Azul', 'Naranja', 'Rosa'};
Coloresrgb = {[255;0;0],[0;255;0],[0;0;255],[255,186,0],[87;35;100]};
rangoHue = [
    0.85 1.0; % Rojo
    0.45 0.56;  % verde
    0.59 0.66; % azul
    0.1 0.4; % naranja
    0.78 0.84; % rosa
];

%Inicializacion de variables-----------------------------
productosAnalizados = 0; % Contador de productos analizados

% Información del experimento---------
nombreAlumno1 = "García Olivarez Carlos Eduardo";
nombreAlumno2 = "Hernández Orozco Cesar Gibran";
nombreAlumno3 = "Hernández Rubio Sandra Stephany";
numeroLote = 'Lote 001';

%Creacion Excel-------------------------
inicioexcel= 6;
archivoExcel = 'informe.xlsx';
if ~isfile(archivoExcel)
    nombres = [nombreAlumno1;
        nombreAlumno2;
        nombreAlumno3;];
    writematrix(nombres, archivoExcel, 'Sheet', 1, 'Range', 'A1' );

    encabezados = ["No. Objeto","Fecha", "Hora", "Lote", "Tipo", "Color",
"Estado", "Confiabilidad"];
    writematrix(encabezados, archivoExcel, 'Sheet', 1, 'Range', 'A5');
end

while true
    % Capturar imagen-------------------------------------------
    IMA = snapshot(cam);
    IMA(:,:,1)=uint8(.5*(IMA(:,:,1)));
    imagen_gris = rgb2gray(IMA);
    %imshow(imagen_gris);
    imagen_binaria=(imagen_gris<umbral);
    %imshow(imagen_binaria);
    momento=Hu_moment(imagen_binaria);
    [vector, porcentaje, nombre]=Parecido(momento);
    nombre;
    porcentaje;
    porcentaje = (round((porcentaje),3));
    if porcentaje >90
        % Filtrado y detección de bordes----------------------------
            se = strel('disk', 5);

            IMfiltro = ~imclose(imagen_binaria,se);

        % Conteo de objetos -------------------
            IMfiltro1=bwareaopen(IMfiltro,18);
```

```
            imshow(IMfiltro1)
            [imagenes, numProductos_detectados]=bwlabel(IMfiltro1);
            numProductos_detectados;


            if numProductos_detectados>1
                disp('ERROR, Presione enter cuando solo haya un objeto en el
area de trabajo');
                input('','s');
            else
                productosAnalizados=productosAnalizados+1;
                IMAbordes = edge(double(IMfiltro), 'Canny', [0.1 0.2], 1);

              % Detección de color (HSV)---------------------------------------
              IMA(:,:,1)=uint8(2*(IMA(:,:,1)));
               imagen_filtrada=IMA;
               imagen_filtrada(~repmat(~IMfiltro, [1, 1, 3])) = 0;
                %imshow(imagen_filtrada);
              hsvImagen = rgb2hsv(imagen_filtrada);
                hue = hsvImagen(:,:,1); % Componente de tono
                hue = hue(hue>0);
                tonoPromedio = mean(hue); % Promedio color de toda la imagen
            IMA(:,:,1)=uint8(.5*(IMA(:,:,1)));
            % Determinar el color del objeto-----------------------------------
--
                colorDetectado = 'desconocido';
                for c = 1:length(colores)
                    if tonoPromedio >= rangoHue(c,1) && tonoPromedio <=
rangoHue(c,2)
                        colorDetectado = colores{c};
                        r=Coloresrgb{c}(1);
                        g=Coloresrgb{c}(2);
                        b=Coloresrgb{c}(3);
                        break;
                    end

                end
                colorDetectado;
            %Sobreposicion de contorno ideal y  contorno real------------
                %Obtener la firma de la imagen
                firma = Encadenado(IMAbordes);
                [ResFirma, ConfianzaFirma] = Firmado(IMAbordes, RUTA);

                porcentaje=ConfianzaFirma*100;
                porcentaje = (round((porcentaje),3));
                ResFirma=nombre;
                if nombre == Producto1
                    plantilla=Plantilla1;
                elseif nombre == Producto2
                    plantilla=Plantilla2;
                else
                    plantilla=Plantilla3;
                end
                %Procrustres

            %Confianza del producto -----------------------------
```

```
                if ConfianzaFirma>=.90
                        estado_contorno='Objeto en buen estado';
                        contorno=IMAbordes;
                        %writePosition(sd, .3);
                        writeline(a,"bueno");
                        readline(a)

                        disp("objeto completo");
                        texto = sprintf("Numero de productos detectados: %d\n" +
...
                "Color detectado: %s\n" + ...
                "Objeto: %s\n" + ...
                "Estado: %s\n" + ...
                "Confiabilidad: %.3f%%\n", ...  % Se agrega %.3f para mostrar el
porcentaje con 3 decimales
                numProductos_detectados, colorDetectado, nombre,
estado_contorno, porcentaje);

                        IMA(:,:,1)=uint8(2*(IMA(:,:,1)));
                        [f,c,~]=size(IMA);
                        for i=1:f
                            for j=1:c
                                if IMAbordes(i, j) > 0
                                    IMA(i, j, :) = [r, g, b];
                                end
                            end
                        end

                        IMA=Overlay_texto(IMA,texto,20,r,g,b);

                else
                       % ConfianzaFirma
                        estado_contorno='Objeto dañado';
                        contorno=plantilla;
                        writeline(a,"malo");
                        readline(a)
                        %writePosition(si, 1);
                        disp("objeto incompleto");
                        IMA(:,:,1)=uint8(2*(IMA(:,:,1)));
                        IMA = Centrar_Ima(IMA,umbral,1);

                        IMA=Overlay_incompleta(plantilla,IMA,umbral,r,g,b);

                        porcentaje=ConfianzaFirma*100;
                        porcentaje = (round((porcentaje),3));
                        texto = sprintf("Numero de productos detectados: %d\n" +
...
                "Color detectado: %s\n" + ...
                "Objeto: %s\n" + ...
                "Estado: %s\n" + ...
                "Confiabilidad: %.3f%%\n", ...  % Se agrega %.3f para mostrar el
porcentaje con 3 decimales
                numProductos_detectados, colorDetectado, nombre,
estado_contorno, porcentaje);
```

```
                        IMA=Overlay_texto(IMA,texto,20,r,g,b);
                end
                % %Estado del contorno con Hu
                % if porcentaje>=90
                %         estado_contorno='Objeto en buen estado';
                  % else
                %         estado_contorno='Objeto dañado';
                % end

            % Mostrar resultados (Overlay) ---------------------------------
----
                %figure(1);
                imshow(IMA);
                %imshow(~IMAbordes);


            %Registro de informe en Excel---------------------------------
---
                inicioexcel=inicioexcel+1;
                fila= strcat("A", num2str(inicioexcel));
                fecha = datestr(now, 'dd-mm-yyyy');
                hora = datestr(now, 'HH:MM:SS');
                %"No. Objeto","Fecha", "Hora", "Lote", "Tipo", "Color",
"Estado", "Confiabilidad"
                Informe = [productosAnalizados, fecha, hora, numeroLote, nombre,
colorDetectado, estado_contorno, porcentaje];
                writematrix(Informe, archivoExcel, 'Sheet', 1, 'Range', fila);


                drawnow;

                pause(9);
             end
    else
        disp('OBJETO NO RECONOCIDO O NO ENCONTRADO');
        pause(9);
    end

end
```

Función firma

```
function [ResRes,ResResnum]=Firmado(f2,RUTARaiz)
g=0;
h=0;
[dist1,cx,cy] = Encadenado(f2);
a=5;
fex=f2;
fex((cy-a):(cy+a),(cx-a):(cx+a))=255;
fex(cy,cx)=0;
%imshow(fex);
resultados(1)=0;
Resultados(1)="";
Carpeta_raiz=RUTARaiz;
Tdist1=size(dist1);
elementos = dir(RUTARaiz);
```

```
esCarpeta = [elementos.isdir] & ~ismember({elementos.name}, {'.', '..'});
Base_datos_carpetas = {elementos(esCarpeta).name};

for N=1:length(Base_datos_carpetas)
    Imagen_actual=Base_datos_carpetas(N);
    ruta2=strcat(Carpeta_raiz,"\", Base_datos_carpetas(N));
    archivosPNG = dir(fullfile(ruta2, '*.png'));
    nombresArchivosPNG = {archivosPNG.name};

    for IM=1:length(nombresArchivosPNG)
        ruta3=strcat(ruta2+"\"+nombresArchivosPNG(IM));
        f4=imread(ruta3);

        if size(f4, 3) == 1
            f4 = cat(3, f4, f4, f4);
        end

        f5 = rgb2gray(f4);
        f6 = f5 < 127;
        dist2 = Encadenado(f6);
        Tdist2=size(dist2);

        if Tdist1(2)>Tdist2(2)
            V1=dist1;
            V2=dist2;
        else
            V1=dist2;
            V2=dist1;
        end
        x_original = linspace(1, length(V1), length(V2));
        x_nuevo = 1:length(V1);
        V2_interpolado = interp1(x_original, V2, x_nuevo, 'linear');

        r = corrcoef(V1,V2_interpolado);
        %disp(N);
        Res = (r(1,2));

        % % Imagen_actual
        % % Res
        if Res>=.0 || Res<=-.0
            g=g+1;
            resultados(g)=Res;
            Resultados(g)=string(Imagen_actual);
            for y=1:length(V2_interpolado)
                Plot_res(g,y)=V2_interpolado(y);
            end
            Resultados_ruta(g)=nombresArchivosPNG(IM);
            %figure, plot(V2_interpolado);
        end
    end

end
    [maximus,imaximus]=max(abs(resultados));
    ResRes=Resultados(imaximus);
    %
    ResResnum=maximus;
```

```
end
```

Función de encadenamiento

```
function [dist, cx, cy] = Encadenado(f2)
    % Mostrar la imagen
    %imshow(f2)

    % Tamaño de la imagen
    [filas, columnas] = size(f2);
    x = 1:columnas;
    fx = sum(f2, 1);
    y = 1:filas;
    fy = sum(f2, 2)';

    % Centroide aproximado
    cx = round(sum((x .* fx) / sum(fx)));
    cy = round(sum((y .* fy) / sum(fy)));

    % Buscar el primer píxel activo
    encontrado = false;
    for i = 1:filas
        for j = 1:columnas
            if f2(i, j) == 1
                encontrado = true;
                break;
            end
        end
        if encontrado
            break;
        end
    end

    % Matriz de direcciones
    dir = [3 2 1;
           4 0 8;
           5 6 7];

    % Punto inicial
    inicio = [i, j];
    n = 0;
    dist = [];

    % Bucle para recorrer el contorno
    while true
        % Ventana local
        V = f2(i-1:i+1, j-1:j+1);

        % Marcar el píxel actual como procesado
        f2(i, j) = 0;

        % Guardar distancia al centroide
        n = n + 1;
        dist(n) = sqrt((cx - j)^2 + (cy - i)^2);

        % Dirección del siguiente píxel
```

```
        d = max(max(dir .* V));
        switch d
            case 1
                i = i - 1;
                j = j + 1;
            case 2
                i = i - 1;
            case 3
                i = i - 1;
                j = j - 1;
            case 4
                j = j - 1;
            case 5
                i = i + 1;
                j = j - 1;
            case 6
                i = i + 1;
            case 7
                i = i + 1;
                j = j + 1;
            case 8
                j = j + 1;
            otherwise
                break;
        end

        % Si regresamos al punto inicial, detener
        if i == inicio(1) && j == inicio(2)
            break;
        end
    end
end
```

Función de momentos de Hu

```
function [momentos_hu]=Hu_moment(imagen_bin)


    [y, x] = find(imagen_bin); % Coordenadas de los píxeles blancos
    m00 = sum(imagen_bin(:)); % Momento de orden 0
    m10 = sum(x); % Momento de orden 1 en X
    m01 = sum(y); % Momento de orden 1 en Y

    x_bar = m10 / m00; % Centroide en X
    y_bar = m01 / m00; % Centroide en Y

    mu20 = sum((x - x_bar).^2);
    mu02 = sum((y - y_bar).^2);
    mu11 = sum((x - x_bar) .* (y - y_bar));
    mu30 = sum((x - x_bar).^3);
    mu03 = sum((y - y_bar).^3);
    mu21 = sum((x - x_bar).^2 .* (y - y_bar));
    mu12 = sum((x - x_bar) .* (y - y_bar).^2);

    eta20 = mu20 / m00^(2);
    eta02 = mu02 / m00^(2);
```

```
    eta11 = mu11 / m00^(2);
    eta30 = mu30 / m00^(2.5);
    eta03 = mu03 / m00^(2.5);
    eta21 = mu21 / m00^(2.5);
    eta12 = mu12 / m00^(2.5);

    momentos_hu = [
        eta20 + eta02;
        (eta20 - eta02)^2 + 4 * eta11^2;
        (eta30 - 3 * eta12)^2 + (3 * eta21 - eta03)^2;
        (eta30 + eta12)^2 + (eta21 + eta03)^2;
        (eta30 - 3 * eta12) * (eta30 + eta12) * ((eta30 + eta12)^2 - 3 * (eta21
+ eta03)^2) + ...
        (3 * eta21 - eta03) * (eta21 + eta03) * (3 * (eta30 + eta12)^2 - (eta21
+ eta03)^2);
        (eta20 - eta02) * ((eta30 + eta12)^2 - (eta21 + eta03)^2) + 4 * eta11 *
(eta30 + eta12) * (eta21 + eta03);
        (3 * eta21 - eta03) * (eta30 + eta12) * ((eta30 + eta12)^2 - 3 * (eta21
+ eta03)^2) - ...
        (eta30 - 3 * eta12) * (eta21 + eta03) * (3 * (eta30 + eta12)^2 - (eta21
+ eta03)^2)
     ];

    momentos_hu=momentos_hu';

    % size(HU_letras);
    % size(momentos_hu);
    %  distancias = sum(abs(HU_letras - momentos_hu), 2);
    %  [~, idx_mas_similar] = min(distancias);
    %  letra=Letras(idx_mas_similar);
    %  if isnan(momentos_hu(1))==true
    %     letra='';
    %  end

end
```

Función para centrado de imagen

```
function Ima_centrada=Centrar_Ima(Ima_no_centrada,umbral,mostrar_centroide)
    Ima_no_centrada1=Ima_no_centrada;
    Ima_no_centrada=rgb2gray(Ima_no_centrada);
    Ima_no_centrada=(Ima_no_centrada<umbral);
    size(Ima_no_centrada);
    Ima_bordes = edge(double(Ima_no_centrada), 'Canny', [0.1 0.2], 1);
    [~, cx, cy] = Encadenado(Ima_bordes);
    [filas, columnas] = size(Ima_no_centrada);
    if mostrar_centroide == 1
        Ima_no_centrada1((cy-1):(cy+1),(cx-1):(cx+1),1)=0;
    end
    horizontalShift = -(cx-(columnas/2));
    verticalShift = -(cy-(filas/2));
    Ima_centrada = imtranslate(Ima_no_centrada1, [horizontalShift,
verticalShift],'FillValues',255);
end
```

Función para correlación de pieza

```
function [vector, porcentaje, nombre]= Parecido(v)
nombre_obj=["Pinza" "Letra I" "Letra X" "fondo"];
    HU_obj=[ 0.2680    0.0067    0.0057    0.0009   -0.0000    0.0001    0.0000;
             0.3415    0.0901    0.0001    0.0000    0.0000    0.0000   -0.0000;
             0.2584    0.0134    0.0005    0.0000   -0.0000    0.0000   -0.0000;
             0.1848    0.0077    0.0022    0.0002    0.0000    0.0000   -0.0000;

            ];

     distancias = sum(abs(HU_obj - v), 2);
     [~, idx_mas_similar] = min(distancias);
     HU_obj=HU_obj';
     vector=HU_obj(:, idx_mas_similar);

     size(vector);
     size(v);
    % Correlación
    correlationCoeff = corr(vector(:), v(:));
    vector=vector';
    porcentaje = abs(correlationCoeff * 100);
    if isnan(v(1))==true
        porcentaje=0;
    end
    nombre= nombre_obj(idx_mas_similar);
    if nombre=="fondo"
        porcentaje=0;
    end
end
```

Función para Overlay

```
 function Imagen_con_texto = Overlay_texto(img, texto,fontSize,r,g,b)
    % Configurar el texto
    overlayText = texto; % Texto a mostrar
    position = [size(img, 2) - 250, 10]; % Posición [x, y] ajustada manualmente
     % Tamaño de la fuente
    textColor = [r, g, b]; % Color del texto en RGB (blanco)

    % Fondo transparente
    boxColor = [0, 0, 0]; % Color del cuadro (se ignorará porque será
transparente)
    boxOpacity = 0; % Fondo completamente transparente

    % Insertar el texto en la imagen
    Imagen_con_texto = insertText(img, position, overlayText, ...
        'FontSize', fontSize, 'TextColor', textColor, ...
        'BoxColor', boxColor, 'BoxOpacity', boxOpacity, ...
        'AnchorPoint', 'RightTop');
end
```

Función para Overlay de pieza incompleta

```
function Ima_con_overlay = Overlay_incompleta(f1, f0, umbral, R, G, B)
    for g=1:5
% Convert images to grayscale
    original = rgb2gray(f0);
```

```
    original_color = f0;
    original1 = rgb2gray(f1);
    % Detect and extract features
    ptsOriginal = detectSURFFeatures(original);
    ptsDistorted = detectSURFFeatures(original1);

    [featuresOriginal, validPtsOriginal] = extractFeatures(original,
ptsOriginal);
    [featuresDistorted, validPtsDistorted] = extractFeatures(original1,
ptsDistorted);

    % Match features between original and distorted images
    indexPairs = matchFeatures(featuresOriginal, featuresDistorted);
    matchedOriginal = validPtsOriginal(indexPairs(:, 1));
    matchedDistorted = validPtsDistorted(indexPairs(:, 2));

    % Check if there are enough matched points
    if size(indexPairs, 1) < 3
        warning('Not enough matched points to estimate geometric transform.
Returning original image.');
        Ima_con_overlay = original_color;
        return;
    end

    % Estimate geometric transform
    try
        [tform, ~] = estimateGeometricTransform2D(matchedDistorted,
matchedOriginal, 'similarity');
    catch
        warning('Error estimating geometric transform. Returning original
image.');
        Ima_con_overlay = original_color;
        return;
    end



    % Warp the distorted edges
    outputView = imref2d(size(original));
    original1 = (original1 > umbral);
    edges1 = edge(double(original1), 'Canny', [0.1 0.2], 1);
    edges1 = removeOuterEdge(edges1);
    recovered = imwarp(edges1, tform, 'OutputView', outputView);
    %imshow(edges1);
    %imshow(recovered);
    % Overlay the edges onto the original image
    [filas, columnas] = size(original);
    for j = 1:filas
        for k = 1:columnas
            if recovered(j, k) > 0
                original_color(j, k, :) = [R, G, B];
            end
        end
    end

    % Return the final overlaid image
```

```
    Ima_con_overlay = original_color;
    end
    % Helper function to remove outer edges
    function img = removeOuterEdge(edges)
        [rows, cols] = size(edges);
        [X, Y] = meshgrid(1:cols, 1:rows);
        radius = min([rows, cols]) / 2; % Radius for the mask
        centerX = cols / 2;
        centerY = rows / 2;
        mask = (X - centerX).^2 + (Y - centerY).^2 <= radius^2;

        % Apply the mask to the edges
        edges = edges .* mask;
        img = edges;
    end

end
```

Desarrollo:
1.- Elaboración de la maqueta.

La maqueta se planteo basándonos en los sistemas de calidad de la industria, donde se coloca una pieza en el área de inspección, y dependiendo del estado que se detecte se desliza hacia la izquierda, si es pieza en buen estado, o hacia la derecha, si la pieza se encuentra dañada.

Figura 1: Escenario para la inspección de calidad.

En el interior de la maqueta se colocaron tiras led que producían luz fría, esto para tener un ambiente controlado con la misma iluminación en todo momento. A su vez se coloco una lámpara extra para obtener más iluminación.

Para la selección de productos se optó por escoger letras poco complicadas que no se llegaran a confundir, esta selección se hizo de un paquete de juguetes para niños.

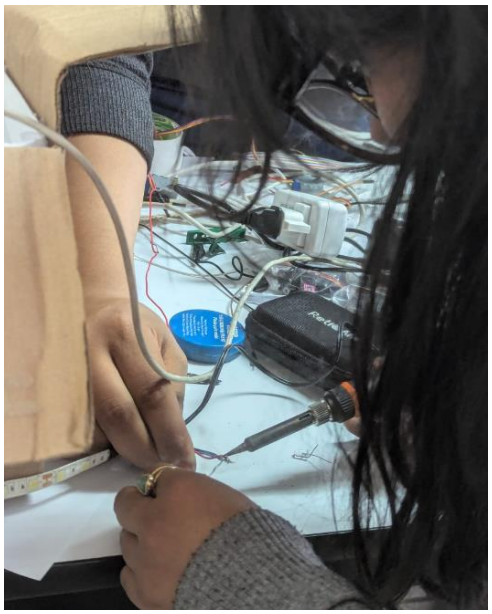


Figura 2: La tira led fue soldada a una etapa de potencia de 12v.

Figura 3: Alfabeto para niños, de estos mismos se seleccionaron los colores a usar

Antes de continuar se decidieron realizar las primeras pruebas con el código de hasta ese momento, esto con el fin de identificar si era necesario realizar alguna corrección a futuro.

Figura 4: Escenario de pruebas para calidad, aún no están los actuadores para facilitar los muestreos.

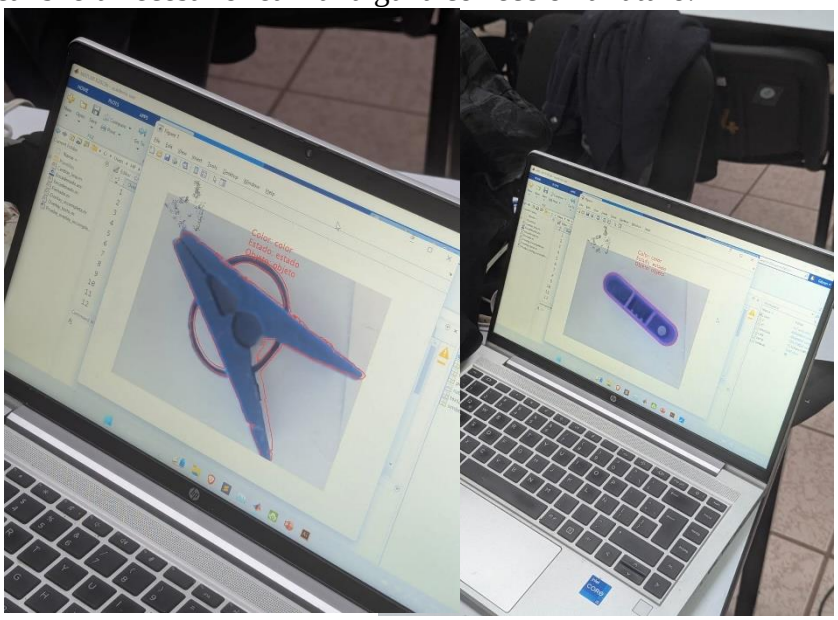
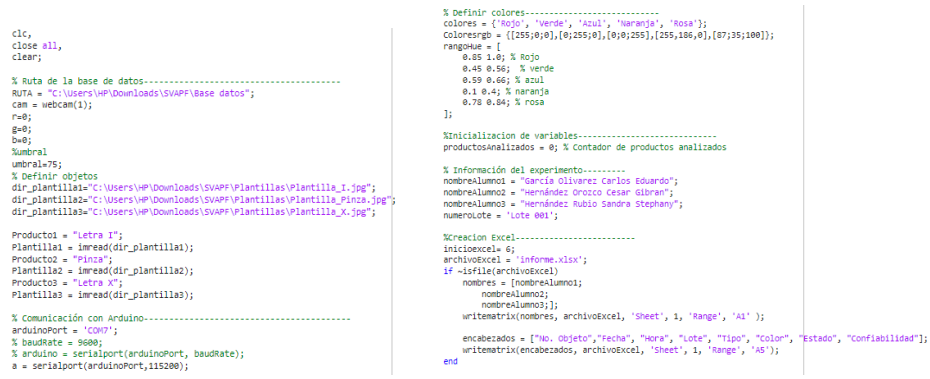


Figura 5: Para nuestras primeras pruebas ya se detectaba el objeto, forma y overlay, aunque aún no había texto que mostrar la clasificación.

2.- Inicialización del programa

En esta parte de nuestro programa se establecen las rutas y variables necesarias para que nuestro sistema funcione, así como la comunicación serial con Arduino para enviar y recibir los datos entre MATLAB y Arduino, los colores a detectar, y datos iniciales de nuestro Excel.



```
clc,
close all,
clear;

% Ruta de la base de datos----------------------------------------
RUTA = "C:\Users\HP\Downloads\SVAPF\Base datos";
cam = webcam(1);
r=0;
g=0;
b=0;
%umbral
umbral=75;
% Definir objetos
dir_plantilla1="C:\Users\HP\Downloads\SVAPF\Plantillas\Plantilla_I.jpg";
dir_plantilla2="C:\Users\HP\Downloads\SVAPF\Plantillas\Plantilla_Pinza.jpg";
dir_plantilla3="C:\Users\HP\Downloads\SVAPF\Plantillas\Plantilla_X.jpg";

Producto1 = "Letra I";
Plantilla1 = imread(dir_plantilla1);
Producto2 = "Pinza";
Plantilla2 = imread(dir_plantilla2);
Producto3 = "Letra X";
Plantilla3 = imread(dir_plantilla3);

% Comunicación con Arduino----------------------------------------
arduinoPort = 'COM7';
% baudRate = 9600;
% arduino = serialport(arduinoPort, baudRate);
a = serialport(arduinoPort,115200);
```

```
% Definir colores----------------------------
colores = {'Rojo', 'Verde', 'Azul', 'Naranja', 'Rosa'};
Coloresrgb = ([255;0;0],[0;255;0],[0;0;255],[255,186,0],[87;35;100]);
rangoHue = [
    0.85 1.0; % Rojo
    0.45 0.56;  % verde
    0.59 0.66; % azul
    0.1 0.4; % naranja
    0.78 0.84; % rosa
];

%Inicializacion de variables----------------------------
productosAnalizados = 0; % contador de productos analizados

% Información del experimento---------
nombreAlumno1 = "García Olivarez Carlos Eduardo";
nombreAlumno2 = "Hernández Orozco Cesar Gibran";
nombreAlumno3 = "Hernández Rubio Sandra Stephany";
numeroLote = 'Lote 001';

%Creacion Excel-----------------------
inicioexcel= 6;
archivoExcel = 'informe.xlsx';
if ~isfile(archivoExcel)
    nombres = [nombreAlumno1;
        nombreAlumno2;
        nombreAlumno3;];
    writematrix(nombres, archivoExcel, 'Sheet', 1, 'Range', 'A1' );

    encabezados = ["No. Objeto","Fecha", "Hora", "Lote", "Tipo", "Color", "Estado", "Confiabilidad"];
    writematrix(encabezados, archivoExcel, 'Sheet', 1, 'Range', 'A5');
end
```

3.- Captura de imagen

En este paso se hace la toma de la imagen y se convierte a escala de grises y a binaria aplicando el umbral que se escogió como el más adecuado.

```
IMA = snapshot(cam);
IMA(:,:,1)=uint8(.5*(IMA(:,:,1)));
imagen_gris = rgb2gray(IMA);
%imshow(imagen_gris);
imagen_binaria=(imagen_gris<umbral);
```

4.- Cálculo y comparación de los momentos de Hu, y filtrado

Se realiza el cálculo con la función de Hu_moment y con Parecido, se compara el objeto actual con las plantillas para de esta forma determinar el porcentaje de similitud y que objetos es el identificado

En el caso de los momentos de Hu, se hace uso de la imagen binaria, y se extraen las propiedades, como el área, la excentricidad y el perímetro.

En la función de parecido se comparar los momentos calculados del objeto fotografiado con los momentos de las plantillas de referencia, como salida tenemos el vector de los momentos calculados del objeto detectado, el porcentaje de similitud y el nombre de la plantilla más similar.

En el caso del filtrado y la detección de bordes se hace uso de las operaciones de imclose y bwareopen, también con bwlabel, se hace el conteo de los productos detectados, en este caso nuestro programa funciona cuando se detecta un solo pbjeto en el area de trabajo.

```
if porcentaje >90
        % Filtrado y detección de bordes-----------------------------
            se = strel('disk', 5);

            IMfiltro = ~imclose(imagen_binaria,se);

        % Conteo de objetos --------------------
            IMfiltro1=bwareaopen(IMfiltro,18);
            imshow(IMfiltro1)
            [imagenes, numProductos_detectados]=bwlabel(IMfiltro1);
            numProductos_detectados;

if numProductos_detectados>1
```

```
                disp('ERROR, Presione enter cuando solo haya un objeto en el area
de trabajo');
                input('','s');
            else
                productosAnalizados=productosAnalizados+1;
                IMAbordes = edge(double(IMfiltro), 'Canny', [0.1 0.2], 1);
```

4.- Detección de color

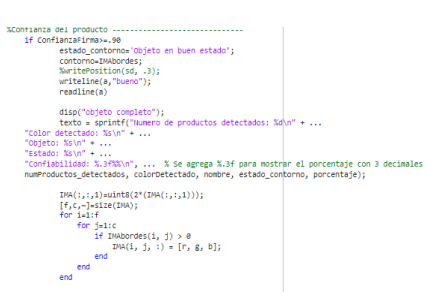
El método que se uso para la identificación de color, consistía en transformar la imagen a un espacio HSV, calculando el tono promedio del objeto para determinar su color y de esta forma compararlo con los rangos ya definidos anteriormente.

```
IMA(:,:,1)=uint8(2*(IMA(:,:,1)));
               imagen_filtrada=IMA;
               imagen_filtrada(~repmat(~IMfiltro, [1, 1, 3])) = 0;
                %imshow(imagen_filtrada);
              hsvImagen = rgb2hsv(imagen_filtrada);
                hue = hsvImagen(:,:,1); % Componente de tono
                hue = hue(hue>0);
                tonoPromedio = mean(hue); % Promedio color de toda la imagen
            IMA(:,:,1)=uint8(.5*(IMA(:,:,1)));
            % Determinar el color del objeto-------------------------------------
--
                colorDetectado = 'desconocido';
                for c = 1:length(colores)
                    if  tonoPromedio  >=  rangoHue(c,1)  &&  tonoPromedio  <=
rangoHue(c,2)
                        colorDetectado = colores{c};
                        r=Coloresrgb{c}(1);
                        g=Coloresrgb{c}(2);
                        b=Coloresrgb{c}(3);
                        break;
                    end

                end
                colorDetectado;
```

5.- Evaluación y clasificación del objeto

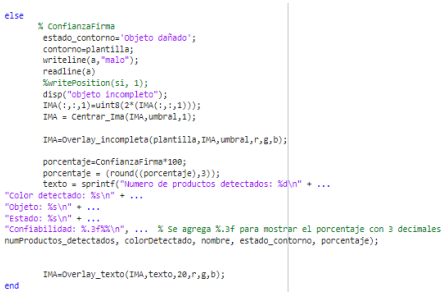
En este caso hacemos el calculo de la firma del objeto con la función de encadenado, y con firmado obtenemos la similitud entre el contorno del objeto y las plantillas, obtenemos la confianza de nuestro producto para posteriormente clasificarlos, mandando los resultados en pantalla e indicando a Arduino que acción realizar para que los motores se muevan en la posición indicada, de esta forma podrá ser enviado al área correspondiente, ya sea de productos desechados o aceptados.



```
%Confianza del producto ----------------------------
    if ConfianzaFirma>=.90
            estado_contorno='Objeto en buen estado';
            contorno=IMAbordes;
            %writePosition(sd, .3);
            writeline(a,"bueno");
            readline(a)

            disp("objeto completo");
            texto = sprintf("Numero de productos detectados: %d\n" + ...
    "Color detectado: %s\n" + ...
    "Objeto: %s\n" + ...
    "Estado: %s\n" + ...
    "Confiabilidad: %.3f%%\n", ...  % Se agrega %.3f para mostrar el porcentaje con 3 decimales
    numProductos_detectados, colorDetectado, nombre, estado_contorno, porcentaje);

            IMA(:,:,1)=uint8(2*(IMA(:,:,1)));
            [f,c,~]=size(IMA);
            for i=1:f
                for j=1:c
                    if IMAbordes(i, j) > 0
                        IMA(i, j, :) = [r, g, b];
                    end
                end
            end
```



```
else
        % ConfianzaFirma
         estado_contorno='Objeto dañado';
         contorno=plantilla;
         writeline(a,"malo");
         readline(a)
         %writePosition(si, 1);
         disp("objeto incompleto");
         IMA(:,:,1)=uint8(2*(IMA(:,:,1)));
         IMA = Centrar_Ima(IMA,umbral,1);

         IMA=Overlay_incompleta(plantilla,IMA,umbral,r,g,b);

         porcentaje=ConfianzaFirma*100;
         porcentaje = (round((porcentaje),3));
         texto = sprintf("Numero de productos detectados: %d\n" + ...
"Color detectado: %s\n" + ...
"Objeto: %s\n" + ...
"Estado: %s\n" + ...
"Confiabilidad: %.3f%%\n", ...  % Se agrega %.3f para mostrar el porcentaje con 3 decimales
numProductos_detectados, colorDetectado, nombre, estado_contorno, porcentaje);


         IMA=Overlay_texto(IMA,texto,20,r,g,b);
end
```

6.- Registro en Excel

Los datos del análisis (número de objeto, fecha, hora, tipo, color, estado, etc.) se guardan en un archivo Excel para llevar un registro.

	A	B	C	D	E	F	G	H	I	J
1	García Olivarez Carlos Eduardo									
2	Hernández Orozco Cesar Gibran									
3	Hernández Rubio Sandra Stephany									
4										
5	No. Objeto	Fecha	Hora	Lote	Tipo	Color	Estado	Confiabilidad		
6										
7	1	15-01-2025	18:34:48	Lote 001	Letra X	Rosa	Objeto en buen estado	99.877		
8	2	15-01-2025	18:31:45	Lote 001	Letra X	naranja	Objeto en buen estado	99.553		
9	3	15-01-2025	18:29:41	Lote 001	Letra X	Rojo	Objeto dañado	69.985		
10	4	15-01-2025	18:30:37	Lote 001	Letra X	Verde	Objeto en buen estado	99.94		
11	5	15-01-2025	18:30:55	Lote 001	Letra X	Verde	Objeto en buen estado	99.931		
12	6	15-01-2025	17:41:51	Lote 001	Pinza	Rojo	Objeto dañado	83.457		
13	7	15-01-2025	17:42:02	Lote 001	Pinza	Rojo	Objeto dañado	85.195		
14	8	15-01-2025	17:42:12	Lote 001	Pinza	Rojo	Objeto dañado	86.046		
15	9	15-01-2025	17:42:23	Lote 001	Pinza	Rojo	Objeto dañado	82.473		
16	10	15-01-2025	17:42:33	Lote 001	Letra X	Azul	Objeto dañado	83.569		
17	11	15-01-2025	17:42:44	Lote 001	Letra X	Azul	Objeto dañado	84.625		
18	12	15-01-2025	17:42:54	Lote 001	Letra X	Azul	Objeto dañado	83.425		
19	13	15-01-2025	17:43:05	Lote 001	Letra X	Naranja	Objeto dañado	78.029		
20	14	15-01-2025	17:43:15	Lote 001	Letra X	Azul	Objeto dañado	86.72		
21										
22										
23										

7. Visualización y sobreposición de resultados

En este paso se sobrepone el texto con la información obtenida y los bordes del objeto detectado en la imagen original utilizando las funciones Overlay_texto y Overlay_incompleta.

El programa está ejecutándose continuamente dentro de un bucle while true, capturando y procesando imágenes en tiempo real.

Resultados

Como final, es esta sección se incorporan los resultados obtenidos, donde se presentan ejemplos de los 5 colores registrados en el sistema, así como el uso de los distintos productos y el caso de daño de alguno de estos.

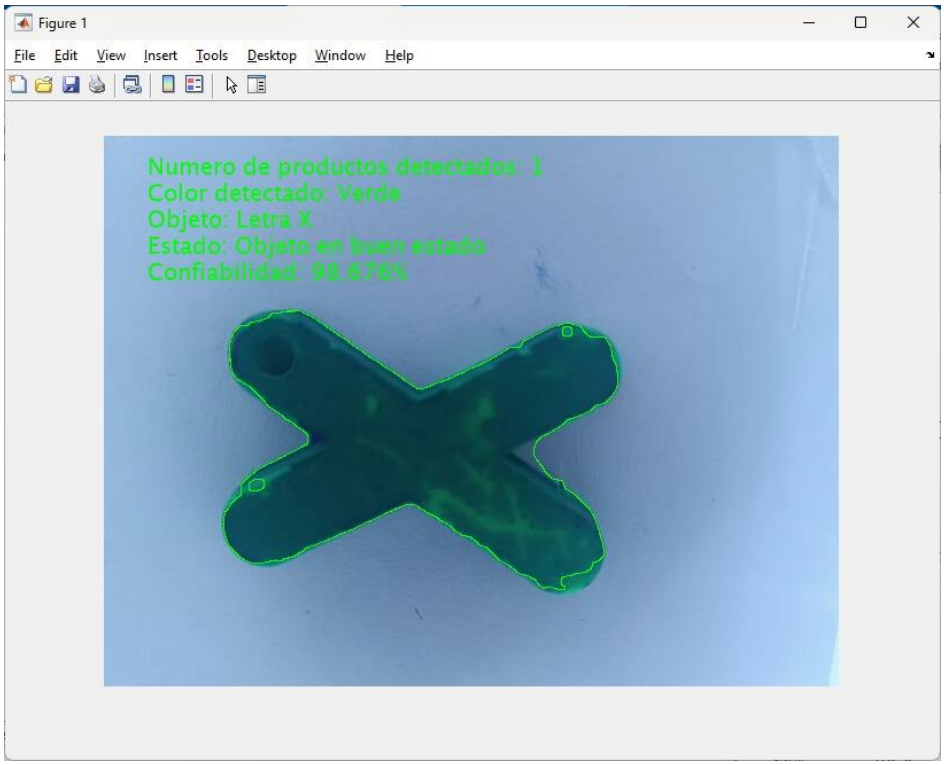


Figura R.1: Resultado para una letra X de color verde

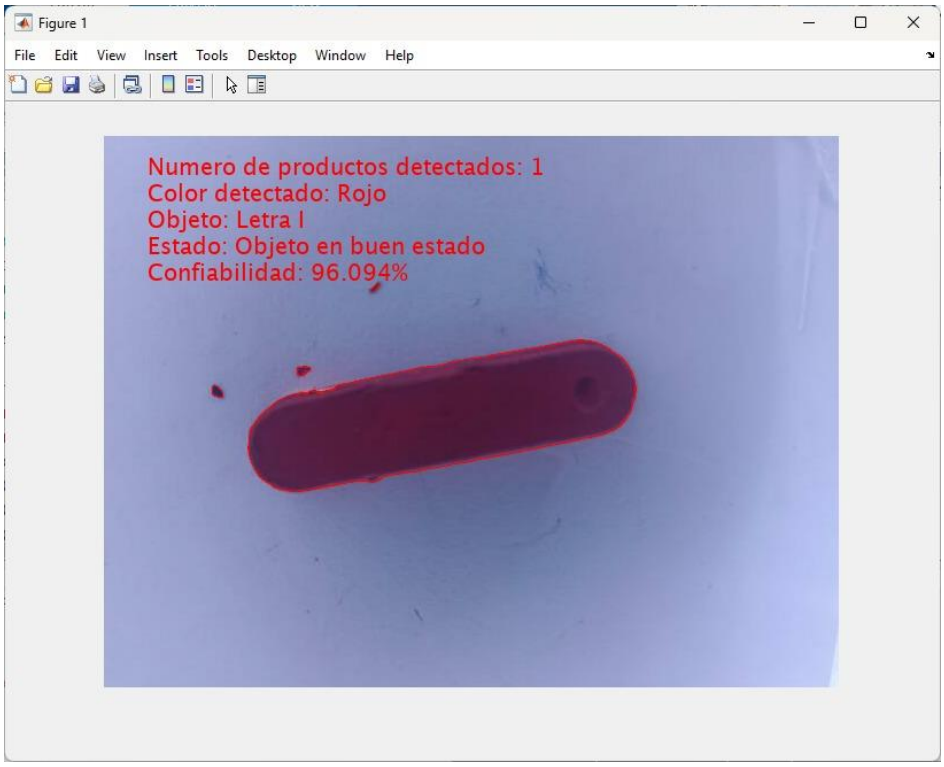


Figura R.2: Resultado para una letra I de color rojo

- Resultados afirmativos:

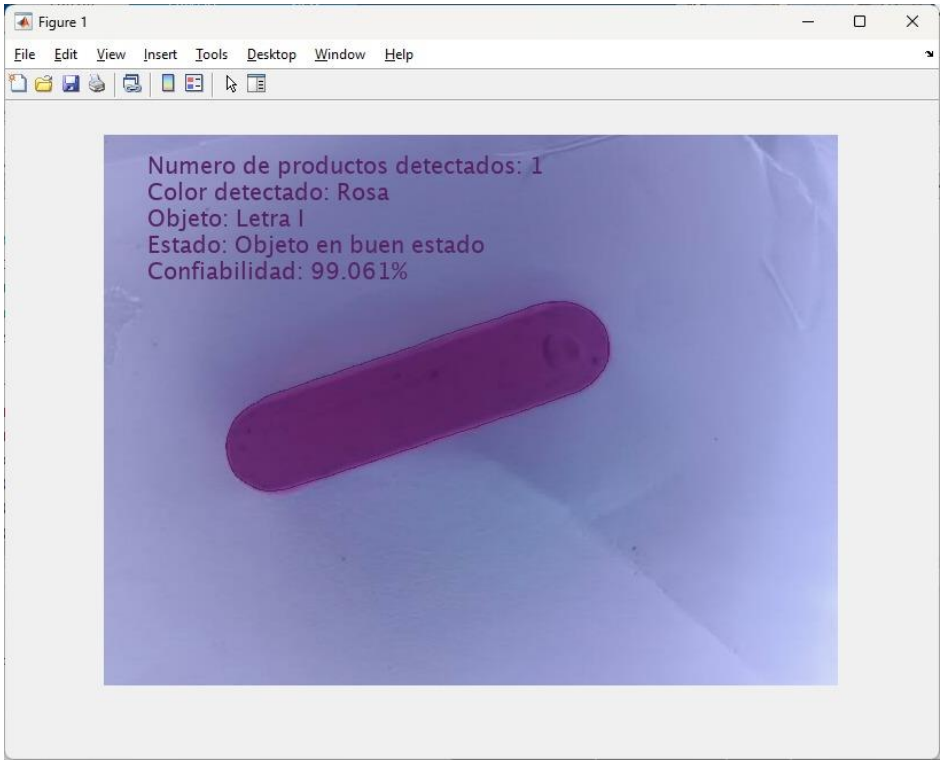


Figura R.3: Resultado para una letra I de color rosa

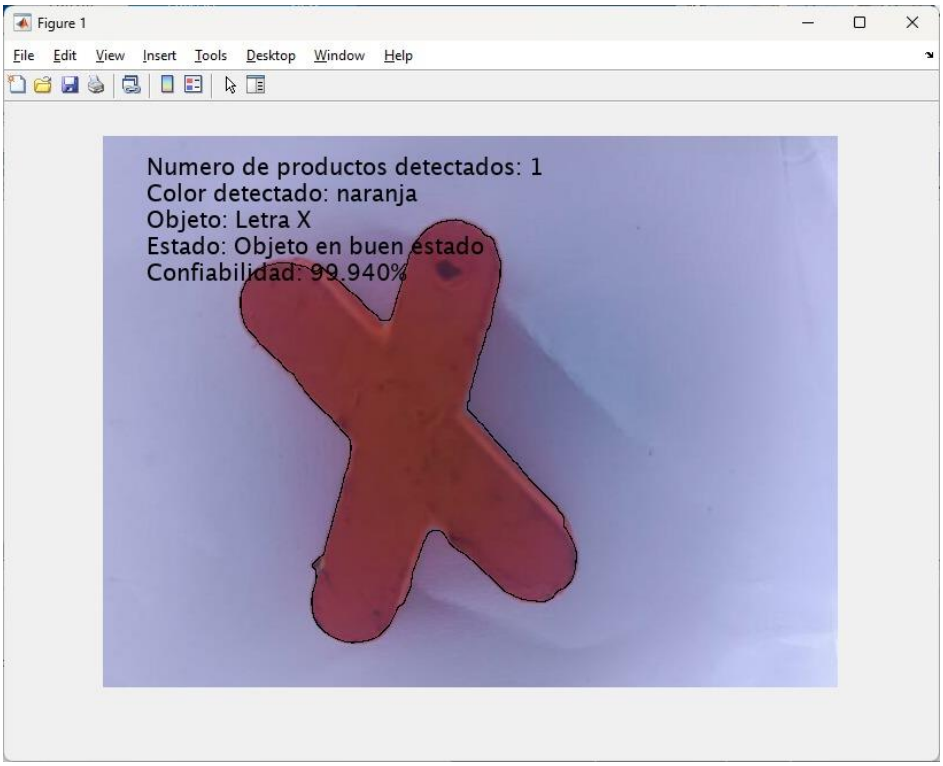


Figura R.4: Resultado para una letra X de color naranja

- Resultados negativos:

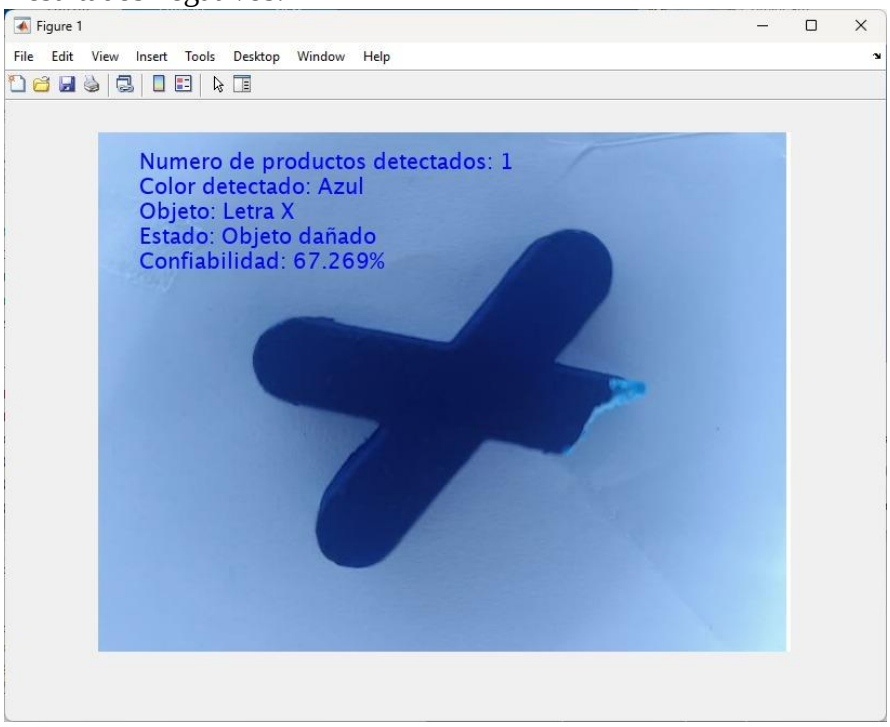


Figura R.5: Resultado para una letra X dañada color azul

Además, se agrega una imagen la cual muestra el resultado de la maqueta final, en la cual se identifican aspectos como la zona de inspección, los actuadores que tiraban las piezas dañadas o buenas respectivamente a su zona, la iluminación controlada e incluso el celular usado como cámara en tiempo real.

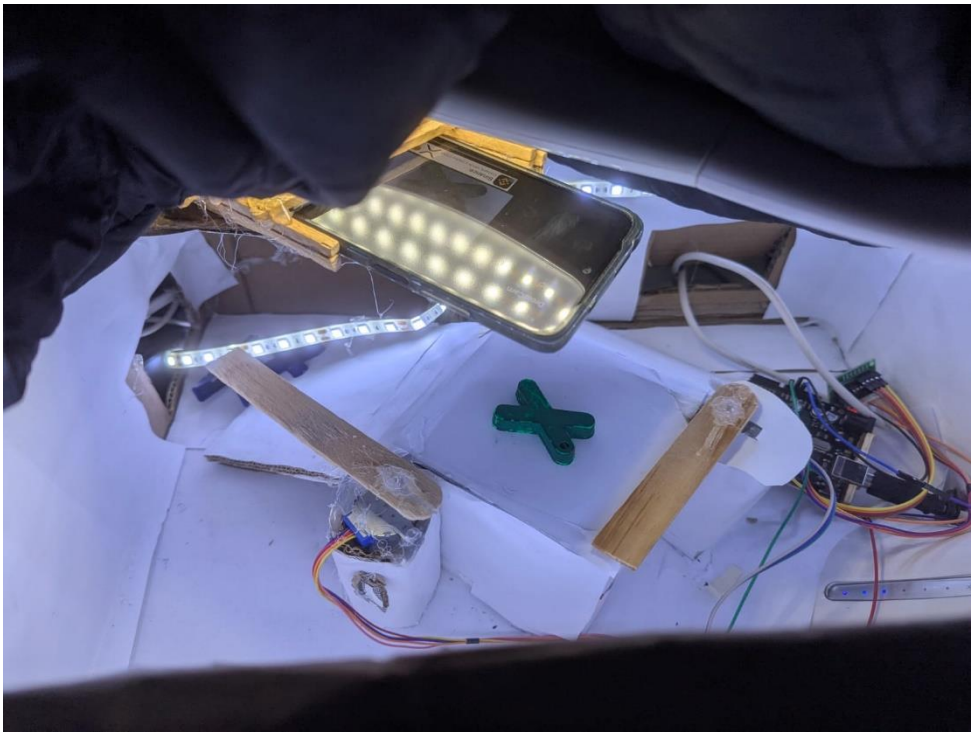


Figura R.7: Escenario de inspección de calidad

Conclusión

El proyecto solicitado nos permitió integrar los diferentes conocimientos adquiridos a lo largo de la materia para el reconocimiento y la clasificación de objetos, pudimos hacer uso de funciones ya utilizadas, y observar como se desempeñan al aplicarlas a un entorno real, viendo como influyen diversos factores, ya no solo de programación sino de el entorno generado para el análisis de los objetos, alterando muchas veces los resultados, aprendimos sobre la importancia de ajustar adecuadamente los parámetros para el procesamiento, como los umbrales o los rangos de color, que sometidos ante distintos entornos y métodos de iluminación podían cambiar drásticamente, vimos la efectividad de los momentos de Hu, que sirven para cosas más simples como la clasificación, mientras que con la firma observamos que pueden ser más adecuadas cuando detalles más específicos son requeridos, en este caso para ver la confiabilidad del producto.