ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

FACULTAD DE INGENIERIA EN ELECTRICIDAD Y COMPUTACIÓN

"Detección de fallas en envases de vidrio no cilíndricos utilizando localización de bordes mediante la herramienta Matlab"

REPORTE DE MATERIA DE GRADUACIÓN

Previo a la obtención del Título de:

INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES

Presentado por:

Braulio Andrés Ruiz Flores

Freddy Daniel Ullauri Ulloa

GUAYAQUIL – ECUADOR

AÑO: 2009

AGRADECIMIENTO

A Dios por la culminación de este trabajo, de igual forma a todas las personas que colaboraron con la realización del mismo, especialmente a la Ing. Patricia Chávez por su invaluable ayuda y consejos.

DEDICATORIA

A Dios, mis padres,

mis hermanos y familiares

que me ayudaron sin dudar

para dar este gran paso en mi vida

Braulio Ruiz Flores

A Dios, A mis padres,

a mi hermana y mis abuelos

por su apoyo incondicional

en la culminación de mi carrera

Daniel Ullauri Ulloa

TRIBUNAL DE SUSTENTACIÓN

MSc. Patricia Chavez
PROFESOR DE LA MATERIA

Ing. Rebeca Estrada
PROFESOR DESIGNADO

Escaneado con CamScanner

DECLARACIÓN EXPRESA

"La responsabilidad del contenido de esta Materia de Graduación, me corresponde exclusivamente; y el patrimonio intelectual de la misma a la ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL".

(REGLAMENTO DE GRADUACIÓN DE LA ESPOL)

Braulio Andrés Ruiz Flores

Freddy Daniel Ullauri Ulloa

Escaneado con CamScanner

RESUMEN

El presente trabajo analiza los operadores para detectar bordes en imágenes y muestra una técnica basada en esto para encontrar alguna falla en envases de vidrio no cilíndricos ya sea fisura, envase quebrado o manchas en su superficie. El análisis se enfocará en el uso de 3 operadores que han ofrecido los mejores resultados en una verificación previa y que además son menos sensibles al ruido.

En el capitulo 1 se da a conocer ciertos conceptos que permitirán entender los distintos términos que se manejan en el proyecto.

El capitulo 2 describe brevemente los métodos para la detección de bordes y explica como estos operan en la imágenes a través de la convolución.

En el último capítulo se menciona como se desarrolló el proyecto, los inconvenientes que se tuvieron y como fueron solucionados. Adicionalmente se presenta una interfaz gráfica en la cual el usuario podrá seleccionar que operador utilizar para la detección de bordes, manejar ciertos parámetros para realizar el procesamiento y verificar si el envase presenta algún defecto.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Las plantas embotelladoras deben realizar un estricto control de calidad de sus operaciones por lo que no debe quedar de lado el estado de las botellas que emplean para depositar su producto. Muchas veces este control era llevado a cabo manualmente, es decir, que había una o varias personas que se encargaban de realizarlo lo cual no era un método seguro, se cometían muchos errores y llevaba demasiado tiempo realizarlo. Los defectos que se encontraban eran fisuras, botellas rotas o botellas con algún objeto en su interior.

Los avances en la computación han permitido que se desarrollen nuevos y modernos sistemas para la detección de fallas en distintos objetos por lo que esta revisión se la puede automatizar, es decir, utilizar una aplicación que permita detectar los defectos de una manera más eficiente, ágil y rápida.

Para este proyecto se presentará como alternativa el análisis de envases de vidrio a través de la detección de bordes empleando 3 algoritmos verificando el que permita obtener el mejor resultado para lo cual se hará un breve estudio de cada uno. Los operadores que se utilizarán son los de Sobel, Prewitt y Roberts.

Estos operadores son parte de la librería de Matlab 7.0.1 por lo cual no es necesario implementarlos, y se puede proceder directamente a una verificación visual de sus resultados.

Se verificará si la calidad de la imagen resultante permite utilizarla para hacer la comparación, caso contrario se utilizarán otros métodos que permitan mejorar los bordes detectados.

Se incluirá un manual de ayuda que permitirá a cualquier usuario manejar el programa teniendo conocimientos previos de procesamiento digital de señales ya que se permitirá escoger el algoritmo de detección de bordes a emplear.

CAPÍTULO 1

1. Conceptos Generales

El procesamiento digital de señales permite realizar análisis de imágenes, audio o video de tal forma que se pueda ejecutar diversas aplicaciones. Sus principales objetivos son:

- Mejorar la calidad visual para permitir la interpretación humana.
- Extraer información de las imágenes para que pueda ser entendida por el ordenador.

En el presente proyecto se utilizará detección de bordes para detectar fallas en envases de vidrio no cilíndricos.

1.1 Definición de Borde

Los bordes de una imagen se los distingue por los cambios bruscos entre valores de píxeles adyacentes. Se puede realizar una clasificación general según sea su dirección en:

- Bordes verticales: cuando los píxeles conectados verticalmente tienen valores diferentes respecto de los anteriores o posteriores.

- Bordes horizontales: cuando los píxeles conectados horizontalmente tienen valores diferentes respecto de los anteriores o posteriores.
- Bordes oblicuos: cuando se tiene una combinación de bordes horizontales y verticales.

1.2 Ruido

Durante el procesamiento de imágenes se debe tener en cuenta algunos errores que pueden degradar la imagen a la cual se la conoce como ruido. El ruido influye notablemente en el proceso de análisis de las imágenes por lo que al realizar cualquier tarea, los resultados deben ser filtrados.

Las causas del ruido son la transmisión de la imagen, adquisición e inclusive el procesamiento, los cuales causan una disminución en la calidad de la imagen y sus efectos en muchos casos suelen ser devastadores.

Para la eliminación del ruido se suelen usar diversos filtros, es decir, no existe una solución fija debido a que el ruido es aleatorio. Muchas veces los filtros que se emplean lo suprimen pero se

recomienda usar filtros espaciales ya que no consumen demasiados recursos del ordenador.

Los tipos de ruido que se suelen emplear para hacer experimentos con imágenes son: ruido blanco y ruido gaussiano.

1.2.1 Ruido Blanco

Es uno de los ruidos que es más empleado debido a la que posee una densidad espectral de potencia constante por lo que su intensidad no decrece mientras aumenta su frecuencia. Además su media es cero y no está correlacionado.

1.2.2 Ruido Gaussiano

Otro ruido usado con frecuencia es el ruido Gaussiano cuya función de densidad de probabilidad está expresada por la curva gaussiana la cual unidimensionalmente está definida como:

$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}} \quad (1)$$

De la ecuación (1), μ es la media σ es la desviación típica de la variable aleatoria. Este ruido es muy usado ya que se asemeja mucho a los casos prácticos.

Al tener una valor de media cero se utiliza para generar ruido aditivo blanco (Figura 1.1).

Figura 1.1: Comparación entre imagen con ruido y sin ruido

CAPÍTULO 2

2. Operadores para la detección de bordes

La detección de esquinas y líneas se basa en los operadores de detección de bordes, mismos que, mediante el cálculo de primeras y segundas derivadas (Figura 2.1) permiten determinar los puntos de principal importancia para poder realizar las mediciones necesarias.

En el análisis de objetos dentro de las imágenes resulta esencial poder distinguir entre el objeto de interés y el resto de la imagen. Las técnicas utilizadas para determinar los objetos de interés son conocidas como técnicas de segmentación. Una de las más comunes es la segmentación mediante la detección de bordes.

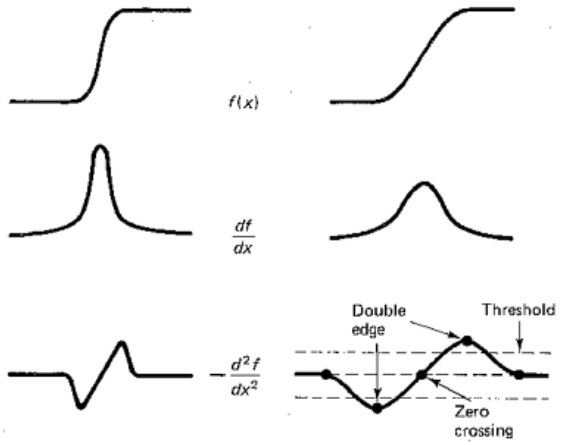

Figura 2.1: Aplicación de primera y segunda derivadas a una función

Existen varios métodos para la detección de bordes los mismos que se refieren a los límites de una imagen. Los métodos de detección de bordes utilizan para sus fines diversos operadores que marcan puntos de acuerdo a discontinuidades en los niveles de gris, los colores o las texturas.

Al mencionar detección de bordes, el término sugiere que la aplicación de un operador con este propósito dará como resultado un contorno. Sin embargo, el objetivo real de detectar bordes es obtener imágenes de mayor intensidad en los valores que detecten transiciones cercanas. Ya que los bordes son encontrados en zonas de la imagen donde el nivel de intensidad cambia bruscamente. Cuanto más rápido se produce el cambio de intensidad, el borde es más fuerte, es por esto que a veces se trata de resaltarlos antes de aplicar esta detección.

En base a varias fuentes consultadas, al momento de elegir un operador se debe tener en cuenta algunos parámetros durante el proceso ya que se puede alterar el resultado esperado, por ello se deben realizar varias pruebas antes de escoger un operador.

Para encontrar los puntos en los que se produce la variación de intensidad, se emplean métodos basados en los operadores derivada.

Básicamente se tienen dos posibilidades: aplicar la primera derivada (gradiente) o la segunda derivada (laplaciana). En el primer caso se buscarán grandes picos y en el segundo, pasos de respuesta positiva a negativa o viceversa (cruces por cero).

Algunos de los algoritmos de detección de bordes más comunes son los siguientes:

- Técnicas basadas en el gradiente:
  - Operador de Roberts
  - Operador de Sobel
  - Operador de Prewitt

Los operadores basados en el gradiente asumen que los bordes de una imagen son píxeles con un alto gradiente. Un rápido índice de cambio de intensidad en alguna dirección dada por el ángulo del vector gradiente puede observarse en los píxeles de los bordes. En la Figura 2.2 se muestra un píxel de borde ideal con su correspondiente vector de gradiente. En el píxel, la intensidad cambia de 0 a 255 en dirección del gradiente. La magnitud del gradiente indica qué tan marcado está el borde. Si calculamos el gradiente en regiones uniformes obtendremos un vector de valor 0, lo que significa que no hay píxeles de borde.

Un píxel de borde se describe mediante dos características importantes:

- Intensidad del borde, que es igual a la magnitud del gradiente.
- Dirección del borde, que es igual al ángulo del gradiente.

El gradiente se estima por medio del uso de operadores. Algunos de estos operadores se describen a continuación.

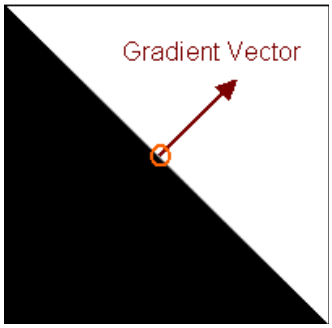

Figura 2.2: Gradiente de un píxel de borde.

2.1 Operadores

Muchas técnicas basadas en la utilización de máscaras para la detección de bordes utilizan máscaras de tamaño 3x3 o incluso más grandes con el fin de reducir los errores producidos por efectos del ruido mediante medias locales tomadas en los puntos en donde se superpone la máscara. Por otro lado, las máscaras normalmente tienen tamaños impares, de forma que los operadores se encuentran centrados sobre los puntos en donde se calculan los gradientes.

Los operadores de gradiente común encuentran bordes horizontales y verticales. Estos operadores trabajan mediante convolución. Los operadores de Prewitt**,** Sobel y Roberts son operadores dobles o de dos etapas. La detección de bordes se realiza en dos pasos, en el primero se aplica una máscara para buscar bordes horizontales, y en el segundo paso se busca bordes verticales, el resultado final es la suma de ambos.

2.1.1 Operador de Roberts

Obtiene una buena respuesta ante bordes diagonales. El gran inconveniente de este detector es que es sensible al ruido. En la Tabla 1 se aprecian las matrices de Roberts.

2.1.2 Operador de Prewitt

Se involucra a los vecinos de las filas y las columnas adyacentes para proporcionar mayor inmunidad al ruido. En la Tabla 1 se aprecian las matrices de Prewitt.

2.1.3 Operador de Sobel

Es más sensible a los bordes diagonales que el de Prewitt, sin embargo la diferencia entre ambos es mínima. En la Tabla 1 se aprecian las matrices de Sobel.

A continuación se detallan las máscaras de convolución de cada uno de estos operadores, los detectores de fila (horizontales) son Hh y los detectores de columna (verticales) son Hv:

	Roberts			Prewitt			Sobel		
Hh	0	0	-1	1	0	-1	1	0	-1
	0	1	0	1	0	-1	2	0	-2
	0	0	0	1	0	-1	1	0	-1
Hv	-1	0	0	-1	-1	-1	-1	-2	-1
	0	1	0	0	0	0	0	0	0
	0	0	0	1	1	1	1	2	1

Tabla 2.1: Matrices de los operadores

CAPÍTULO 3

3. Desarrollo del proyecto

A lo largo del desarrollo del proyecto surgieron una serie de inconvenientes que fueron resueltos para obtener imágenes adecuadas y que permitan ser verificadas para detectar alguna falla.

3.1 Preparación de las imágenes

Parte fundamental de este proyecto son los envases a examinar, se seleccionaron envases casi rectangulares pues el análisis es un poco más sencillo ya que si se utilizaban circulares el análisis requería de un proceso más complejo para lograr analizar todos los detalles.

Figura 3.1: Envases empleados en el proyecto

Las imágenes capturadas para su posterior análisis deberían tener la mejor calidad posible para evitar ruido en las imágenes. Entre los

distintos problemas existentes al momento de capturar las imágenes era el fondo, al realizar las fotografías con un fondo muy oscuro y luego de realizar el análisis ocurría que la imagen tenía demasiado ruido por lo que no se apreciaban los detalles de los envases y además si es que alguno tenía un defecto.

Para realizar las fotografías era importante realizarlas sin flash ya que esta luz era reflejada en el envase y por eso al analizarlas, habían muchos errores ya que los bordes de la luz la tomaba como un defecto.

Otro parámetro importante a corregir fue el reflejo de la fuente de luz en los envases el cual nos permitía capturar la imagen sin flash, en las imágenes se creaba demasiado brillo en ciertas partes por lo que había que colocarlas correctamente y así mismo evitar cualquier sombra ya que podía alterar el resultado ya que al final del análisis eran tomados como defectos por el programa.

Para corregir estos detalles no deseados en las imágenes existen diferentes técnicas basadas en la fotografía. En nuestro caso realizamos un pequeño cajón con fondo de cartulina blanca y las paredes y techo cubiertos de papel calco. La cartulina blanca en el

fondo y en el piso nos ayudó a manejar el ruido detrás del envase, además el papel calco permitió que la fuente de luz que iluminaba el cajón no se vea reflejada en el envase. Cabe recalcar que el piso y el fondo del cajón se unían en una especie de curva para evitar así tener una sombra en esa unión y no tener una línea al momento de filtrar las imágenes.

Figura 3.2: Imagen del cajón

Con todos estos detalles se realizaron la toma de las imágenes y recalcando que al momento de tomar las fotos tanto la cámara como los envases estaban en posiciones fijadas previamente.

3.2 Manipulación de las imágenes

Una vez capturadas las imágenes de los envases debemos analizarlas para así poder determinar si el envase tiene defectos o no. En nuestro proyecto optamos el uso de MATLAB, herramienta computacional que entre sus prestaciones básicas se hallan: la manipulación de matrices,

la representación de datos y funciones, la implementación de algoritmos, la creación de interfaces de usuario (GUI) y la comunicación con programas en otros lenguajes y con otros dispositivos de hardware (Ver ANEXO A).

El proceso que se realizó para analizar las imágenes en Matlab empieza con una reducción del tamaño a la mitad de las mismas que lo más probable es que se pierdan algunos detalles pero en este caso esa pérdida es mínima ya que el pixeleado es grande (2304x1728).

Luego realizamos un filtrado de la imagen por medio del comando *medfilt2* el cual nos ayuda a eliminar el ruido conocido como “sal y pimienta” que se refiere a los píxeles blancos que suelen haber en las imágenes. Una vez filtrado procedemos a realizar la detección de bordes en las imágenes para lo cual utilizaremos una máscara definida por el usuario entre Prewitt, Sobel o Roberts.

El siguiente paso es la convolución de la resultante en la detección de bordes con una matriz definida previamente por nosotros y así suavizar la imagen para reducir el número de componentes conectados en la imagen. Una ultima manipulación a la imagen se la realiza por medio del comando *bwareaopen* que nos permite eliminar

objetos en la imágenes con un número de píxeles definido por el usuario, este número mediante la experimentación que tuvimos para nuestro caso el valor a utilizar será 60 píxeles.

Una vez que hemos realizado estos pasos, seleccionaremos la parte central de la imagen que corresponde a la parte central del envase y luego de esto utilizamos el comando *bwlabel* para etiquetar cada unos de los objetos que se detectan en esta sección del envase. Si el envase llegase a tener algún defecto esta función etiquetará el mismo y lo cual nos permitirá descartar este envase para algún proceso posterior.

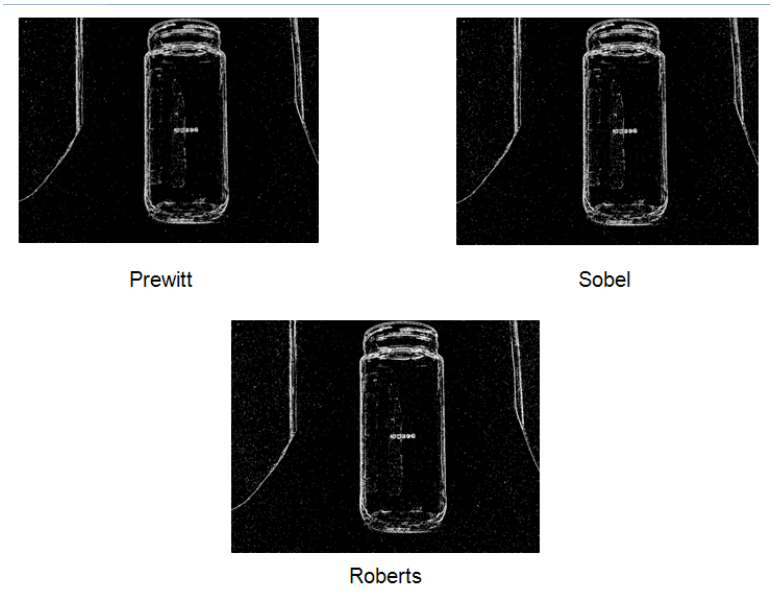

Figura 3.3: Imágenes aplicada los tres operadores

3.3 Funcionalidad del Programa

El programa está diseñado para realizar el análisis de los envases ya sea imagen por imagen, o sino realizar un análisis de todas las imágenes que estén almacenadas en una carpeta. Además el usuario podrá configurar los parámetros de píxeles para el caso del filtro para suavizar las imágenes y el tipo de operador que desea utilizar en el programa.

El programa le presentará al usuario la imagen que está siendo analizada y debajo de ella el estado del envase (Envase con fallas o Envase en Buen estado), si el envase llegara a tener defectos el programa abrirá una ventana adicional en la cual mostrará la imagen con los bordes detectados en la cual se podrá apreciar el defecto que tiene dicho envase. El código de esta interfaz se muestra en el ANEXO B.

3.4 Resultados obtenidos

Se analizaron 38 imágenes a las cuales se aplicaron los operadores de Sobel, Prewitt y Roberts. Con esto se verificó que 3 frascos fueron detectados incorrectamente.

Al hacer el análisis con los 3 operadores para detectar bordes, las imágenes de los bordes diferían en detalles mínimos pero esto no incidió para que el resultado se alterara.

Envases con fallas analizados	24
Envases sin fallas analizados	14
Verdaderos positivos	21
Verdaderos negativos	13
Falsos negativos	3
Falsos positivos	1

Tabla 3.1 Tabla de resultados

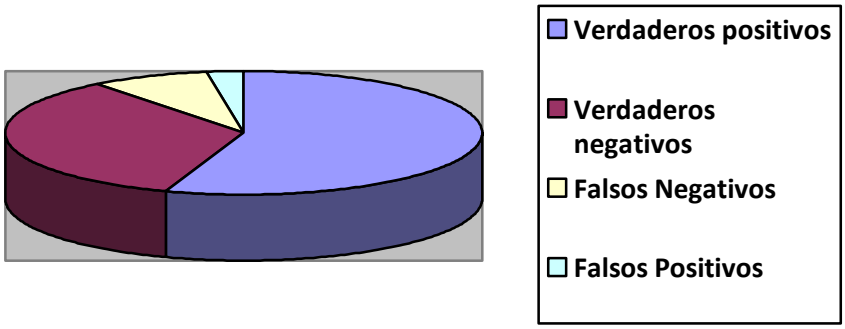

Figura 3.4 Resultado del análisis

CONCLUSIONES

- Al utilizar cualquiera de los 3 operadores se determinó que todos detectaban el mismo número de envases con defectos por lo que depende del criterio del usuario con cual desea realizar el proceso de análisis de los envases.

- A pesar de que el programa es bastante simple, el análisis tiene su retardo debido al tamaño de la imagen, complejidad del proceso o recursos del ordenador.

- En los 3 falsos negativos en la detección de fallas, fueron manchas leves en los envases, mientras que los envases con fisuras todas fueron detectados.

- Se deja como un futuro proyecto mejorar el proceso de análisis para que el tiempo en verificar los envases se reduzca ya que al hacerlo con un paquete de imágenes el tiempo es mayor aún.

- Se debe buscar las mejores alternativas posibles para que las fotografías obtenidas muestren la mayor cantidad de detalles de modo que los resultados sean más precisos.

- El sistema puede ser automatizado con la toma de las imágenes en tiempo real ya que para el traslado de los frascos se utilizaría una banda transportadora.

RECOMENDACIONES

- Al capturar las imágenes de los envases se debe tener cuidado con el reflejo de la luz en el envase, existen diferentes técnicas para poder minimizar el efecto de esta.

- Procurar elegir un fondo adecuado de modo que no interfiera con el proceso de análisis de la imagen ya que los envases al ser transparentes están expuestos a que cualquier objeto aparezca.

- Al tomar las fotografías se debe fijar un lugar tanto para la cámara como para los envases para que el análisis esté centrado en un solo lugar.

- Para evitar cualquier pérdida de información con respecto a la intensidad del color se deben hacer las fotografías con luz blanca de para que se pueda tener una intensidad de acuerdo al color del envase.

- Al ser envases no cilíndricos, se debe procurar hacer tomas de cada lado del envase de modo que cualquier defecto la cámara lo pueda captar al estar lo más cerca posible a las caras del mismo.

ANEXOS

ANEXO A

MANUAL DEL USUARIO

El programa está diseñado para detectar defectos en las paredes envases de vidrio ya sean estos fisuras, envases sucios o manchas.

Se divide en dos partes:

- Se puede verificar defectos en una sola imagen o,
- Detectar fisuras en un grupo de imágenes previamente almacenadas en una carpeta

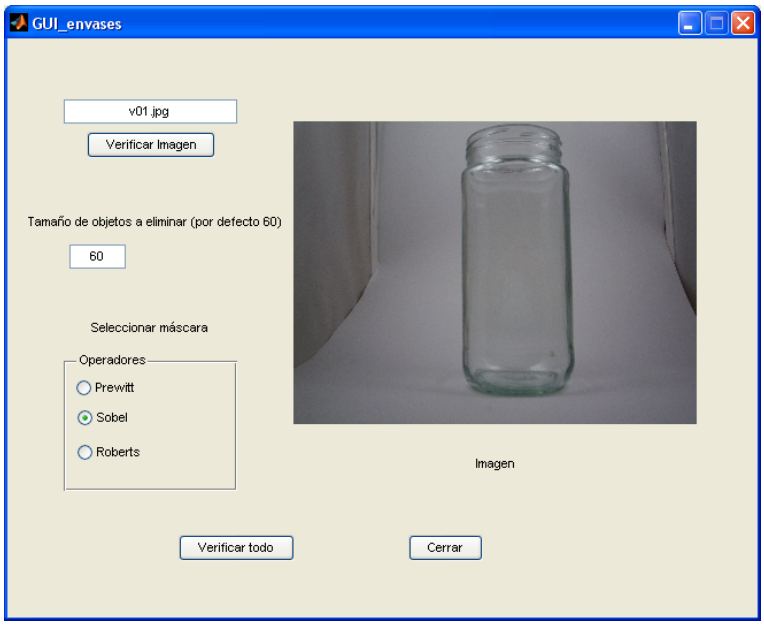

Figura A-1.- Menu principal del programa

Parámetros configurables:

Tamaño de objeto a eliminar: mediante un filtro podemos eliminar objetos (ruido) q tienen hasta cierto número de píxeles existentes en la imagen filtrada y mediante este parámetro podemos configurarlo. Mediante la experimentación consideramos q el valor q nos conviene para este proyecto es de 60 píxeles.

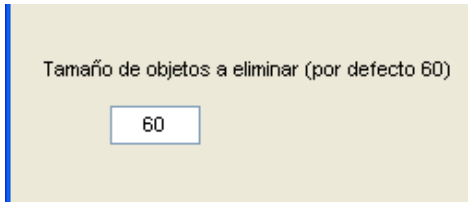

Figura A-2.- Configuracion de tamaño de objeto a elminar

Selección de máscara: Podemos seleccionar la máscara con la cual deseamos detectar los bordes en las imágenes ya sea Prewitt, Sobel o Roberts.

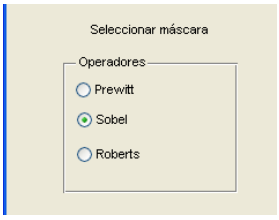

Figura A-3.- Selección de mascara

Examinar una sola imagen

Para examinar una sola imagen se debe cargar el nombre del archivo en el recuadro que se muestra en la imagen y dar clic en el botón “Verificar Imagen”.

Figura A-4.- Configuración de nombre de imagen y boton para examinar

Examinar un grupo imagen

También podemos examinar un grupo de imágenes previamente almacenadas en una carpeta, el programa ira mostrando cada imagen e indicará si el envase tiene o no fallas. Si el envase llegara a tener defectos se abrirá una ventana en la cual mostrará la imagen de los bordes detectados en dicha imagen e indicando a que archivo pertenece esta imagen.

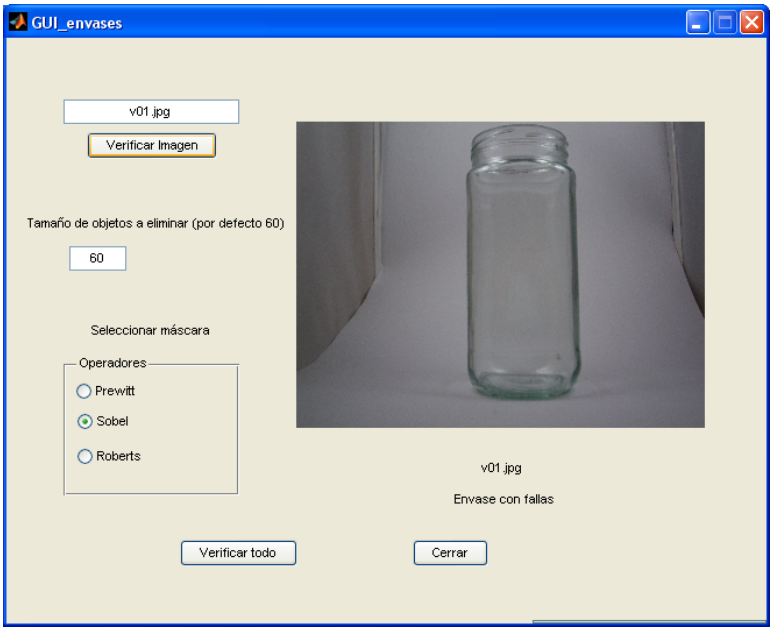

Figura A-5.- Respuesta del programa despues de examinar imagen

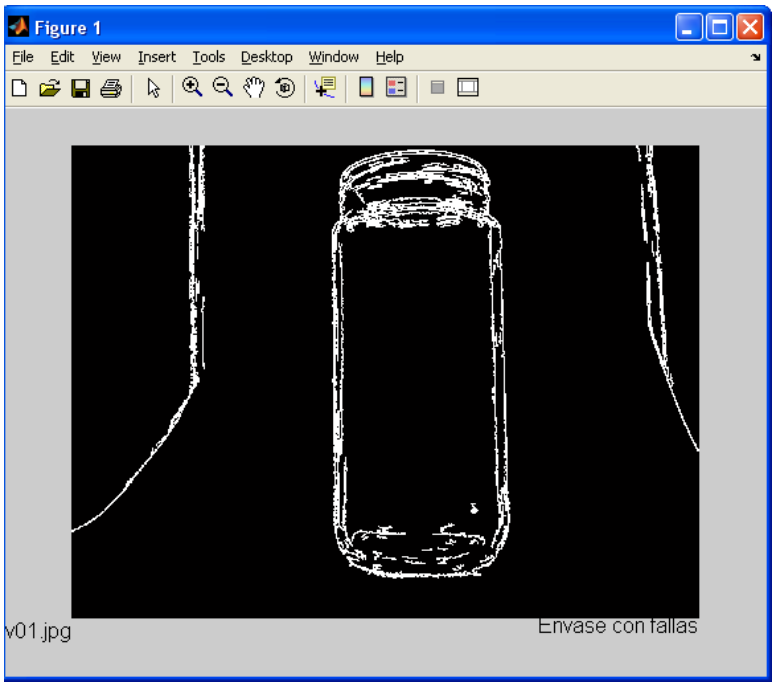

Figura A-6.- Imagen de bordes de un envase con fallas

ANEXO B

Código del programa

```
function varargout = GUI_envases(varargin)
%GUI_ENVASES M-file for GUI_envases.fig
%      GUI_ENVASES, by itself, creates a new GUI_ENVASES or raises
the existing
%      singleton*.
%
%      H = GUI_ENVASES returns the handle to a new GUI_ENVASES or
the handle to
%      the existing singleton*.
%
%      GUI_ENVASES('Property','Value',...) creates a new GUI_ENVASES
using the
%      given property value pairs. Unrecognized properties are
passed via
%      varargin to GUI_envases_OpeningFcn.  This calling syntax
produces a
%      warning when there is an existing singleton*.
%
%      GUI_ENVASES('CALLBACK') and
GUI_ENVASES('CALLBACK',hObject,...) call the
%      local function named CALLBACK in GUI_ENVASES.M with the given
input
%      arguments.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows
only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help GUI_envases

% Last Modified by GUIDE v2.5 25-Feb-2009 17:39:06

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @GUI_envases_OpeningFcn, ...
                   'gui_OutputFcn',  @GUI_envases_OutputFcn, ...
                   'gui_LayoutFcn',  [], ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end
```

```
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before GUI_envases is made visible.
function GUI_envases_OpeningFcn(hObject, eventdata, handles,
varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   unrecognized PropertyName/PropertyValue pairs from the
%            command line (see VARARGIN)

im_original=imread('blanco.jpg');
set(handles.orgIm,'HandleVisibility','ON');
axes(handles.orgIm);
image(im_original);
axis equal;
axis tight;
axis off;
set(handles.orgIm,'HandleVisibility','OFF');


% Choose default command line output for GUI_envases
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes GUI_envases wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% inicializa parametros por defecto
initialize_gui(hObject, handles, false);
set(handles.uipanel1,'SelectionChangeFcn',@uipanel1_SelectionChangeF
cn);


% --- Outputs from this function are returned to the command line.
function varargout = GUI_envases_OutputFcn(hObject, eventdata,
handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;



function inEdit_Callback(hObject, eventdata, handles)
% hObject    handle to inEdit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of inEdit as text
%        str2double(get(hObject,'String')) returns contents of
inEdit as a double


% --- Executes during object creation, after setting all properties.
function inEdit_CreateFcn(hObject, eventdata, handles)
% hObject    handle to inEdit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
%%%%

if ispc
    set(hObject,'BackgroundColor','white');
else

set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor
'));
end


% --- Executes on button press in loadPush.


function Pixeles_Callback(hObject, eventdata, handles)
% hObject    handle to Pixeles (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Pixeles as text
%        str2double(get(hObject,'String')) returns contents of
Pixeles as a double

pixeles = str2double(get(hObject, 'String'));   % Captura el numero
de pixeles de los objetos que se desea filtrar en la imagen
```

```
if isnan(pixeles)
    set(hObject, 'String', 60);
    errordlg('Input must be a number','Error');
end

handles.metricdata.pixeles = pixeles;
guidata(hObject,handles)

% --- Executes on button press in appPush.
function appPush_Callback(hObject, eventdata, handles)
% hObject    handle to appPush (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Esta funcion verifica si existen fallas en las imagenes de los
envases
% almacenados en una carpeta


% Se almacena en una matriz todas la imagenes que hay en la carpeta
fileFolder = fullfile(matlabroot,'work');
dirOutput = dir(fullfile(fileFolder,'v*.jpg'));
fileNames = {dirOutput.name}'
tam = size(fileNames);

i=1;
if tam(1)>1
    while i<=tam(1)                % Recorre la carpeta que
contiene las imagenes de los envases

        im = imread(fileNames{i});
        %Presenta la imagen del envase en el Menu principal
        set(handles.orgIm,'HandleVisibility','ON');
        axes(handles.orgIm);
        image(im);
        axis equal;
        axis tight;
        axis off;
        set(handles.orgIm,'HandleVisibility','OFF');

        set(handles.P_imagen, 'String', fileNames{i});

        im2=im(1:2:end, 1:2:end, 1:1:end); % Se reduce a la mitad la
imagen
        im1=rgb2gray(im2);
        im1=medfilt2(im1,[3 3]); %Remueve el ruido de la imagen
        BW = edge(im1,handles.metricdata.operador); %Detecta los
bordes segun el operador seleccionado
        msk=[0 0 0 0 0;
             0 1 1 1 0;
             0 1 1 1 0;
             0 1 1 1 0;
             0 0 0 0 0;];
```

```
        B=conv2(double(BW),double(msk)); %Suavizar imagen para
reducir el número de componentes conectados
        wnoise=bwareaopen(B,handles.metricdata.pixeles); %Remueve
objetos de la imagen con un numero menor de pixeles determinado
        Im_cut=wnoise(185:685,515:765);
        [L,num] = bwlabel(Im_cut,4);

            if num ~= 0 % Si el envase tiene fallas se presenta los
bordes de la imagen muestreada
                figure, imshow(wnoise);
                text(size(wnoise,2),size(wnoise,1)+15, ...
                    'Envase con fallas', ...
                    'FontSize',12,'HorizontalAlignment','right');
                text(size(wnoise,3),size(wnoise,1)+25, ....
                    fileNames(i), ...
                    'FontSize',12,'HorizontalAlignment','right');
                set(handles.Estado_envase, 'String', 'Envase con
fallas');
            else
                set(handles.Estado_envase, 'String', 'Envase en buen
estado');
            end
         i=i+1;
    end
else
        im=imread ('no.jpg');
        set(handles.orgIm,'HandleVisibility','ON');
        axes(handles.orgIm);
        image(im);
        axis equal;
        axis tight;
        axis off;
        set(handles.orgIm,'HandleVisibility','OFF');
        set(handles.P_imagen, 'String', '');
        set(handles.Estado_envase, 'String', 'No se encuentran
imagenes almacenadas');
end
% --- Executes on button press in closePush.
function closePush_Callback(hObject, eventdata, handles)
% hObject    handle to closePush (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close all;




% --- Executes during object creation, after setting all properties.
function Pixeles_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Pixeles (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called
```

```
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function initialize_gui(fig_handle, handles, isreset)
% If the metricdata field is present and the reset flag is false, it
means
% we are we are just re-initializing a GUI by calling it from the
cmd line
% while it is up. So, bail out as we dont want to reset the data.
if isfield(handles, 'metricdata') && ~isreset
    return;
end

%Configuracion de parametros por defecto
handles.metricdata.pixeles = 60;
handles.metricdata.operador = 'Sobel';
set(handles.Pixeles, 'String', handles.metricdata.pixeles);
set(handles.uipanel1, 'SelectedObject', handles.sobel);

set(handles.orgIm,'HandleVisibility','ON');
axes(handles.orgIm);
axis equal;
axis tight;
%axis off;
set(handles.orgIm,'HandleVisibility','OFF');

% Update handles structure
guidata(handles.figure1, handles);

function uipanel1_SelectionChangeFcn(hObject, eventdata)
% hObject    handle to the selected object in unitgroup
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%Coloca el nombre correspondiente a la variable segun el operador
%seleccionado

handles = guidata(hObject);
if (hObject == handles.prewitt)
    handles.metricdata.operador='Prewitt';
else if (hObject == handles.sobel)
    handles.metricdata.operador='Sobel';
else
    handles.metricdata.operador='Roberts';
    end
end
guidata(hObject,handles)
```

```
function Nombre_envase_Callback(hObject, eventdata, handles)
% hObject    handle to Nombre_envase (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Nombre_envase as
text
%        str2double(get(hObject,'String')) returns contents of
Nombre_envase as a double


% --- Executes during object creation, after setting all properties.
function Nombre_envase_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Nombre_envase (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in VerificarImagen.
function VerificarImagen_Callback(hObject, eventdata, handles)
% hObject    handle to VerificarImagen (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Esta funcion verifica si existen fallas en el envase



image_file = get(handles.Nombre_envase,'String');
if ~isempty(image_file)
im_original=imread(char(image_file));
set(handles.orgIm,'HandleVisibility','ON');
axes(handles.orgIm);
image(im_original);
axis equal;
axis tight;
axis off;
set(handles.orgIm,'HandleVisibility','OFF');
end;

set(handles.P_imagen, 'String', image_file);
```

```
im2=im_original(1:2:end, 1:2:end, 1:1:end);
im1=rgb2gray(im2);
im1=medfilt2(im1,[3 3]); %Median filtering the image to remove
noise%
prueba=handles.metricdata.operador;
BW = edge(im1,prueba); %finding edges
msk=[0 0 0 0 0;
     0 1 1 1 0;
     0 1 1 1 0;
     0 1 1 1 0;
     0 0 0 0 0;];
B=conv2(double(BW),double(msk)); %Suavizar imagen para reducir el
número de componentes conectados
wnoise=bwareaopen(B,handles.metricdata.pixeles);

Im_cut=wnoise(185:685,515:765);
[L,num] = bwlabel(Im_cut,4);

        if num ~= 0
            figure, imshow(wnoise);
            text(size(wnoise,2),size(wnoise,1)+15, ...
                'Envase con fallas', ...
                'FontSize',12,'HorizontalAlignment','right');
            text(size(wnoise,3),size(wnoise,1)+25, ....
                image_file, ...
                'FontSize',12,'HorizontalAlignment','right');
            set(handles.Estado_envase, 'String', 'Envase con
fallas');
        else
            set(handles.Estado_envase, 'String', 'Envase en buen
estado');
        end
```

BIBLIOGRAFÍA

- MADISETTI Vijay K., WILLIAMS Douglas B., “Digital Signal Processing Handbook”, Chapman & Hall/CRCnetBase, 1999.

- FERNANDEZ Nicolas, “Contribución al reconocimiento de objetos 2D mediante detección de bordes en imágenes a color”, 2002.

- VASEGHI Sabed V. “Advanced Digital Signal Processing and Noise Reduction”, John Wiley & Sons Ltda., Second Edition, 2000.

- MATLAB, “Image Processing Toolbox User’s Guide”, Mathworks Inc., 2008.