Proyecto Primer Parcial

Visión por Computadora

Elaborado por:
José Baltazar Ramírez Rodríguez

Dra María Susana Ávila García

26 de marzo del 2019

I. INTRODUCCIÓN

La visión por computadora permite el procesamiento de imágenes para un sinfín de especialidades que van desde la medicina, pasando por la música, hasta el ámbito de seguridad; ya que este procesamiento nos permite mejorar los componentes de una imagen: el contraste, la intensidad de un píxel, la orientación de un píxel, etc. En el área de medicina, por ejemplo, sería fundamental lograr tener una clara percepción de ciertas regiones de una imagen donde se tenga alguna anomalía y poder así determinar un correcto diagnóstico al paciente.

Una de las primeras aplicaciones del procesamiento digital de imágenes, fue en la industria del periódico (Rafael C. González, 1977), cuando las imágenes eran enviadas primero por cable submarino entre Londres y Nueva York. Después se introdujo el sistema de transmisión de imágenes por cable Bartlane, que reducía el tiempo en el que se transportaba una imagen a través del Atlántico de una semana a menos de tres horas. En la actualidad, existen diferentes técnicas y métodos para el procesamiento digital de imágenes.

En este primer proyecto se describirán algunas funciones que fueron implementadas en una interfaz de usuario (GUI) y que permiten operaciones a nivel píxel -siendo este el elemento mínimo que compone una imagen – para modificar y tener mejores resultados en una imagen.

II. DESCRIPCIÓN DE LA APLICACIÓN

La aplicación consiste en permitir al usuario, aplicar distintos métodos que logren procesar una imagen. Los métodos que el usuario puede utilizar son:

- Transformación Lineal de Intensidad de Imagen (Negativo de una imagen)
- Transformación Logarítmica de Intensidad de Imagen
- Transformación de Ley de Potencia de Intensidad de Imagen
- Transformación lineal por partes: Estiramiento de Contraste (Contrast Stretching)
- Transformación lineal por partes: Segmentación de Nivel de Intensidad (Intensitylevel Slicing)
- Cálculo del Histograma de una imagen en escala de grises
- Ecualización de Histograma de una imagen en escala de grises.

- Ajuste de Histograma de una imagen en escala de grises con el de una imagen dereferencia. (Histogram Matching)

La aplicación, como se muestra en la figura 1, despliega una interfaz que permite mediante un pop-up menú, seleccionar entre los tipos de transformaciones que se pueden realizar a una imagen en el procesamiento de imágenes.

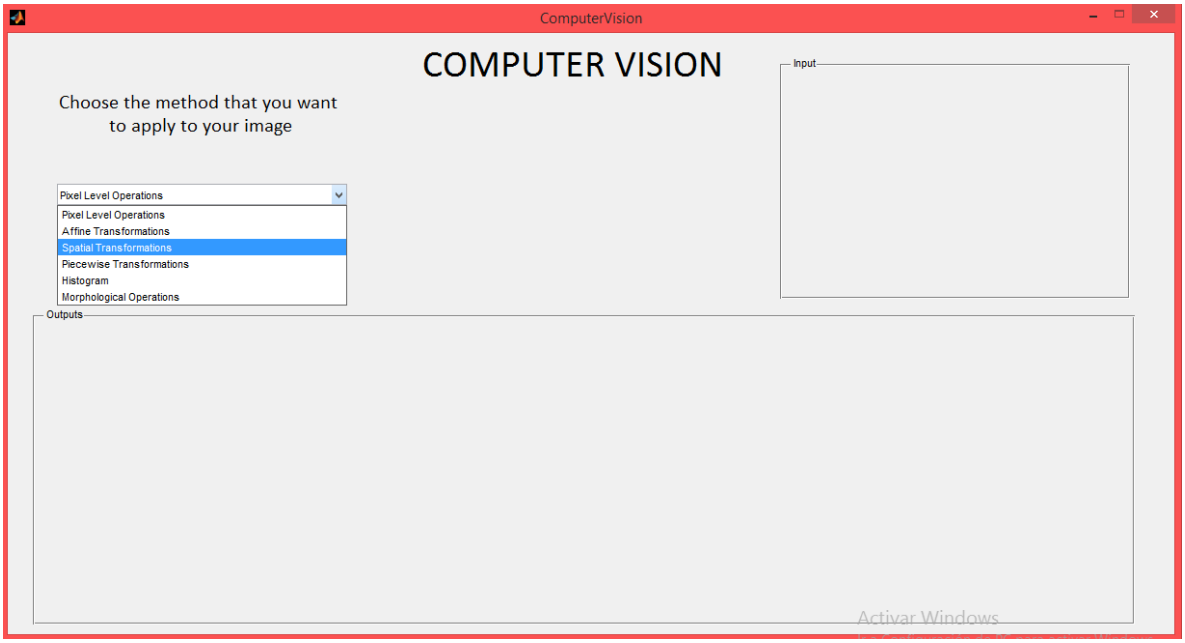


Fig 1. Elección del método a utilizar

Posteriormente la interfaz habilita un botón al usuario para poder cargar su imagen que será procesada. Se muestra en la figura 2.

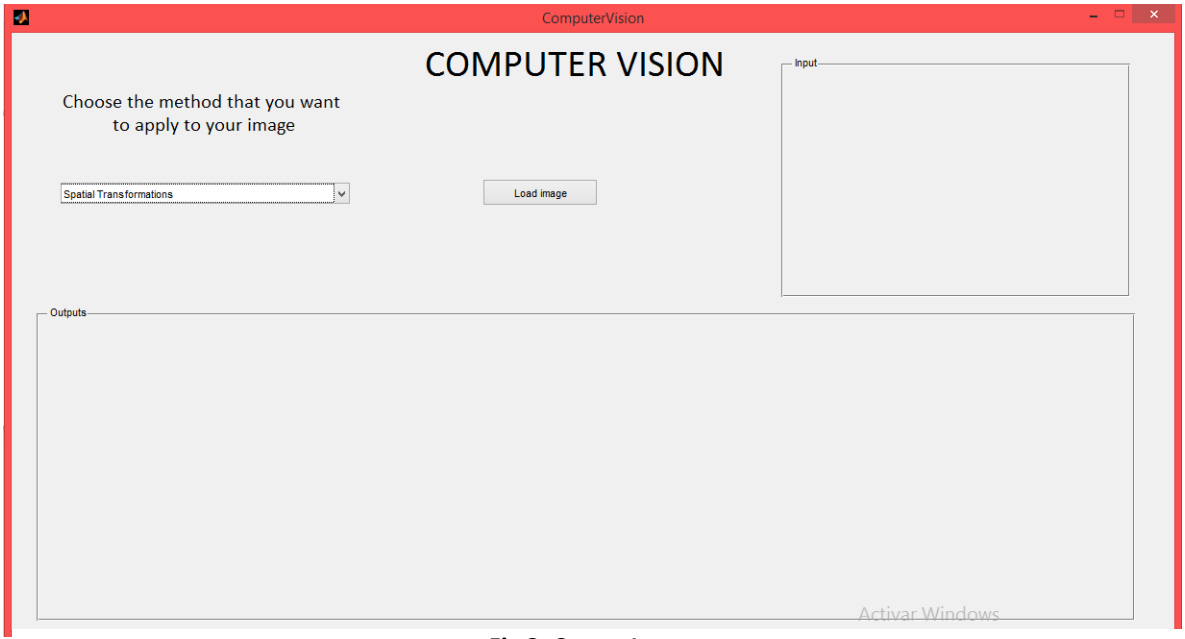


Fig 2. Cargar Imagen

Una vez que el usuario da clic en el botón, podrá seleccionar la imagen que desea y para luego dar clic en aceptar.

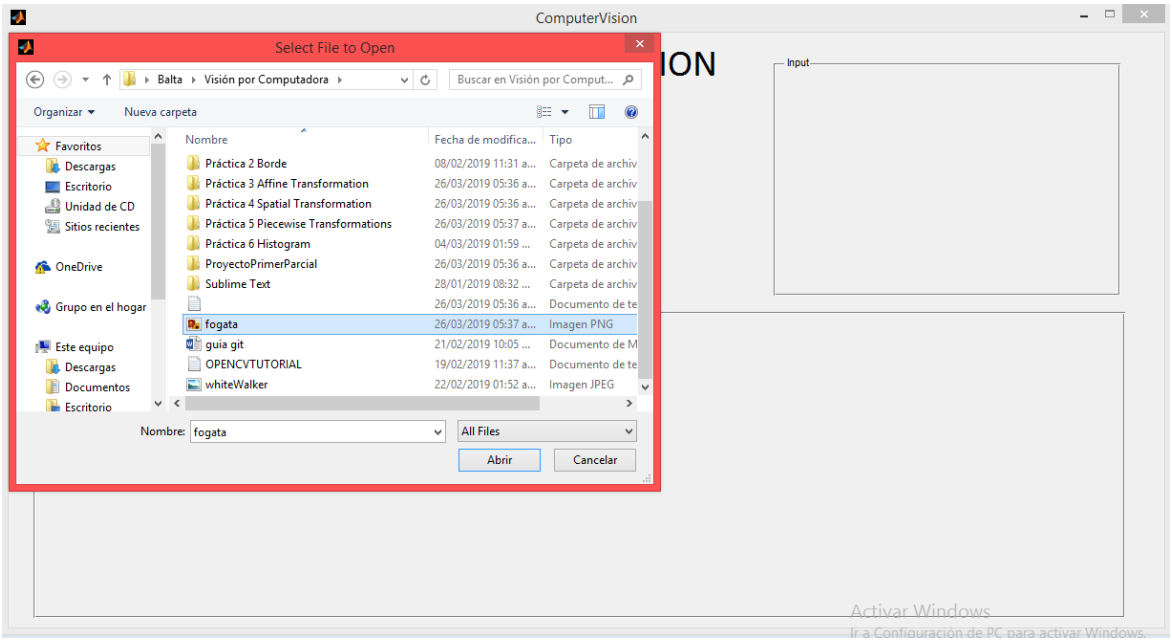


Fig 3. Seleccionar Imagen

El programa comenzará a procesar la imagen de acuerdo con el método que el usuario eligió con la imagen anteriormente cargada y mientras realiza este proceso, se desplegará un letrero de procesando datos.

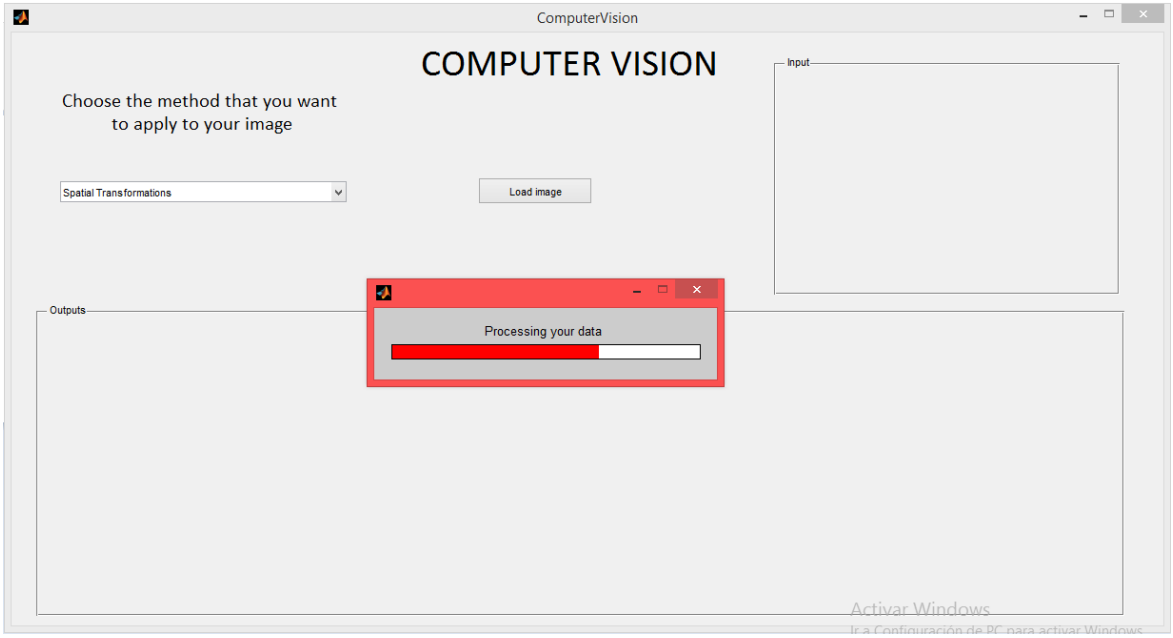


Fig 4. Procesando datos

Por último, de acuerdo con el método que el usuario eligió desplegará dos o tres imágenes de salida, o una tabla dependiendo si es el histograma el que se eligió, y en la parte superior derecha se mostrará la imagen de entrada en escala de grises y en la parte inferior las imágenes de salida. El resultado se muestra en la figura 5.

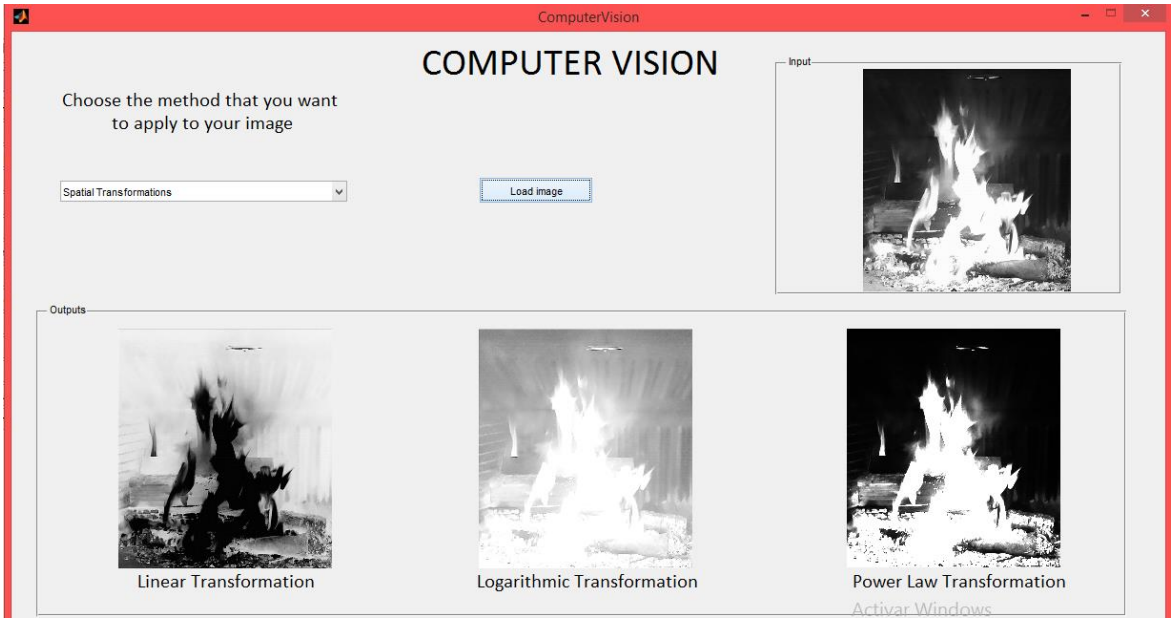


Fig 5. Resultado

III. DESCRIPCIÓN DE LOS MÉTODOS

Los métodos que se utilizaron son los que se trabajaron en clase a través de las prácticas, a diferencia que esta vez serían transcritos y adaptados al lenguaje MATLAB.

El primero, es el método para las Transformaciones Espaciales. Dentro de este se habilita un botón, el cual, al ser presionado, lee la imagen que va a ser seleccionada y la convierte a escala de grises para comenzar a ser procesada.

```
[File_Name, Path_Name] = uigetfile();
img = imread([Path_Name, File_Name]);


imagenEntrada = handles.imagenEntrada;      %el objeto que se va a manejar
%img = imread('fogata.png');                 %leer la imagen y asignarla en una matriz
imagenGris = rgb2gray(img);                  %convertir la imagen a escala de grises
imagenGrisLogaritmo = imagenGris;           %imagen para transformación logarítmica
imagenGrisPowerLaw = imagenGris;            %imagen para power law operación
```

Además, se crean un par de copias de las matrices para cada método a realizar, en estos casos: *imagenGris*, *imagenGrisLogaritmo* e *imagenGrisPowerLaw* para transformación lineal, logarítmica y de potencia respectivamente.

```
%----Diálogo de carga-----------------------------------------------
f = waitbar(0,'Please wait...');
pause(.5)

waitbar(.67,f,'Processing your data');
pause(.5)
close(f)
%-------------------------------------------------------------------
imshow(imagenGris, 'Parent', imagenEntrada); %mostrar la imagen en escala de grises en el axes imagenEntrada

%Activar las vistas de las etiquetas de imágenes de salida
set(handles.linearTransformation,'Visible','on');
set(handles.logarithmicTransformation,'Visible','on');
set(handles.powerLawTransformation,'Visible','on');
```

Después se agrega un *waitbar* de MATLAB para generar tiempo de espera en lo que se carga la imagen y se procesa, además que se habilitan los *static text* para señalar las imágenes de salida. A partir de esa sección del código comienzan los métodos para procesar las imágenes. El primero de ellos es la transformación lineal donde se define un limite máximo de 256 llamado *l* y se comienza a recorrer la matriz *imagenGris*. Se obtiene la intensidad de cada pixel y se aplica la operación pertinente para obtener el negativo de la imagen.

```
%---------------Linear Transformation-------------------------------
l = 256;    %el valor maximo en escala de grises que puede alcanzar la imagen

for i = 1:size(imagenGris,1)
    for j = 1:size(imagenGris,2)
        r = imagenGris(i,j);    %obtener el valor de cada pixel de la imagen
        s = l - 1 - r;          %aplicar la operacion
        imagenGris(i,j) = s;    %asignar la salida a cada pixel de la imagen
    end
end
imshow(imagenGris,'Parent',handles.salida1);    %mostrar la salida ya transformada
```

Luego se asigna la salida a cada píxel de la imagen y se muestra en uno de los axes de salida de la interfaz de MATLAB.

El siguiente método es el de Transformación logarítmica. Al igual que en el proceso anterior, se recorre la matriz y se obtiene la intensidad asociada a cada píxel, la diferencia radica en la fórmula que hay que utilizar. Se realiza un cast a double para poder realizar la operación de *log* en MATLAB y se regresó el valor a uint8 para asignar valores enteros a la imagen de salida. Luego se verifica que la salida esté dentro del rango de 0-255. Si excede este límite el valor máximo permitido será de 255 y se muestra la imagen de salida en otro axes diferente en la interfaz.

```
%---------------Logarithmic Transformation-------------------------
c = 50;     %constante de operacion


for i = 1:size(imagenGrisLogaritmo,1)
    for j = 1:size(imagenGrisLogaritmo,2)
        r = imagenGrisLogaritmo(i,j);    %obtener el valor de cada pixel de la imagen
        s =  c * log(double(1+r));       %aplicar la operacion casteando a doble. Matlab no deja operar logaritmo con tipo uint8
        s = uint8(s);                   %castear a entero para evitar los decimales del resultado del logaritmo

        if s <= 255     %¿está dentro del rango?
            imagenGrisLogaritmo(i,j) = s;    %asignar la salida a cada pixel de la imagen
        elseif s > 255
            imagenGrisLogaritmo(i,j) = 255;    %asignar la salida a cada pixel de la imagen
        end
    end
end
imshow(imagenGrisLogaritmo,'Parent',handles.salida2);    %mostrar la salida ya transformada
```

Es importante destacar que aquí el valor de la constante *c* es fijo, por lo que el usuario sólo podrá visualizar cambios bajo este valor en la fórmula de $s = c * \log(1 + r)$

Lo mismo ocurre con la siguiente función y los valores de sus constantes, *c* y *gamma* para la Transformación de Potencias. La fórmula para este caso es $s = c * r^{gamma}$

```
%---------------Powe Law Transformation----------------------------
c1 = 0.001;     %constante de operacion
gamma =  2.5;


for i = 1:size(imagenGrisPowerLaw,1)
    for j = 1:size(imagenGrisPowerLaw,2)
        r = imagenGrisPowerLaw(i,j);    %obtener el valor de cada pixel de la imagen
        s =  c1 * power(double(r),gamma);    %aplicar la operacion
        s = uint8(s);                   %castear a entero para evitar los decimales del resultado del logaritmo

        if s <= 255     %¿está dentro del rango?
            imagenGrisPowerLaw(i,j) = s;    %asignar la salida a cada pixel de la imagen
        elseif s > 255
            imagenGrisPowerLaw(i,j) = 255;    %asignar la salida a cada pixel de la imagen
        end
    end
end
imshow(imagenGrisPowerLaw,'Parent',handles.salida3);    %mostrar la salida ya transformada
```

La siguiente opción del Pop-Up Menu es para las Transformaciones por partes que incluye las funciones de *contrast stretching* e *intensity-level slicing*.

Igual que en los casos anteriores, se utilizan constantes ya definidas en el código. Para el contrast stretching se declaran los valores de r1, r2, s1 y s2 que se pueden observar en el código. También se declara un *waitbar* para esperar en lo que procesa la imagen y se recorre la matriz, obtiene intensidad de cada píxel. Se aplica la fórmula dependiendo del valor de *r* para determinar sobre cuál rango estará la nueva salida, todo esto siempre menor a 255 y mayor a 0.

```
%---------ContrastStretching-------------------------------------------------

r1 = 50;
r2 = 100;
s1 = 50;
s2 = 200;


for i = 1:size(imagenGrisContrastStretching,1)
    for j = 1:size(imagenGrisContrastStretching,2)
        r = imagenGrisContrastStretching(i,j);    %obtener el valor de cada pixel de la imagen
        if 0 <= r && r <= r1
            s = (s1/r1)*r;
           imagenGrisContrastStretching(i,j) = s;
        elseif r1 < r && r <= r2
            s = ((s2-s1)/(r2-r1))*(r-r1)+s1;
            imagenGrisContrastStretching(i,j) = s;
        elseif r2 < r && r <= 255
            s = ((255-s2)/(255-r2))*(r-r2)+s2;
            imagenGrisContrastStretching(i,j) = s;
        end
    end
end
imshow(imagenGrisContrastStretching,'Parent',handles.salida1);    %mostrar la salida ya transformada
```

En el método de intensity-level slicing se obtiene el valor de la intensidad del píxel y si se encuentra dentro del rango de A y B, se asigna un valor de 255, de lo contrario se manda a 0. Estos valores de A y B también son definidos en el código.

```
%---------Intensity-Level Slicing-------------------------------------------
A = 15;
B = 50;

for i = 1:size(imagenGrisIntensitySlicing,1)
    for j = 1:size(imagenGrisIntensitySlicing,2)
        r = imagenGrisContrastStretching(i,j);    %obtener el valor de cada pixel de la imagen
        if A < r && r < B
            imagenGrisIntensitySlicing(i,j) = 255;
        else
            imagenGrisIntensitySlicing(i,j) = 0;
        end
    end
end
imshow(imagenGrisIntensitySlicing,'Parent',handles.salida2);    %mostrar la salida ya transformada
imshow(imagenGris, 'Parent', imagenEntrada); %mostrar la imagen en escala de grises en el axes imagenEntrada

%Activar las vistas de las etiquetas de imágenes de salida
set(handles.contrastStretching,'Visible','on');
set(handles.intensityLevelSlicing,'Visible','on');
```

Por último, se muestran las imágenes en los axes correspondientes y se activan las etiquetas de texto para señalar a qué transformación corresponde.

El siguiente método es el histograma. En este igual que en todos, se recorre la matriz y se obtiene la intensidad de cada píxel. Se define un arreglo de 1 a 256 para guardar la cantidad de veces que ocurre un nivel de gris en la imagen y poder sacar los valores de la tabla que se muestra en la interfaz.

```
%para sacar la cantidad de ocurrencias de cada nivel de gris
for i = 1:size(imagenGris,1)
    for j = 1:size(imagenGris,2)
        r = imagenGris(i,j);
        if r == 0
            r = 1;
            escalaGris(r) = escalaGris(r)+1;
        %else
         %   escalaGris(r) = escalaGris(r)+1;
        end
        escalaGris(r) = escalaGris(r)+1;
    end
end

totalPixeles = numel(imagenGris);

cdf = 0;
pdfArreglo = 1:256;
cdfArreglo = 1:256;
salidaNivelGris = 1:256;
filas = 1:256;
```

Si el valor de la intensidad es 0, se suma un 1 porque MATLAB no permite los índices iguales a 0. Se determina de una vez el valor del total de píxeles con la función *numel*. Se declaran arreglos para almacenar los valores en la tabla del histograma, todos de 1 a 256. Se calcula el histograma con las fórmulas proporcionadas en clase.

```
%para calcular el histograma
for i = 1:256
    pdf = escalaGris(i)/totalPixeles;
    pdfArreglo(i) = pdf;
    cdf = cdf + pdf;
    cdfArreglo(i) = cdf;
    salidaNivelGris(i) = l * cdf;
    salidaNivelGris(i) = round(salidaNivelGris(i));
    if salidaNivelGris(i) > 255
        salidaNivelGris(i) = 255;
    end
       % salida = sprintf('Input Level: %d | Output Level: %d', i, salidaNivelGris(i));
        %disp(salida)
    %else
        %salidaNivelGris(i);
        %salida = sprintf('Input Level: %d | Output Level: %d', i, salidaNivelGris(i));
        %disp(salida)
end
```

Si la salida de nivel gris es mayor al permitido, se define que el valor máximo sea de 255. Se guardan todos los valores de manera ordenada de 1 a 256.

```
  %para llenar la tabla
   datos = [filas',escalaGris',pdfArreglo',cdfArreglo',salidaNivelGris'];
   set(handles.tablaHistograma,'Data',datos);
   set(handles.tablaHistograma,'ColumnName',{'InputGrayLevel';'Ocurrence';'PDF';'CDF';'OutputGrayLevel'});
   set(handles.tablaHistograma,'Visible','on');

imagenGris1 = imagenGris;

%para ecualizar el histograma
 for i = 1:size(imagenGris1,1)
    for j = 1:size(imagenGris1,2)
        r = imagenGris1(i,j);
        for k = 1:256
            if r == k
                imagenGris1(i,j) = salidaNivelGris(r);
            end
        end
    end
 end
```

Se llena la tabla de la interfaz creando una matriz con los arreglos que se obtuvieron en el cálculo del histograma y se activa la vista de la tabla. También se muestra la ecualización del histograma, que recorre la matriz y el arreglo de la salida de nivel de Gris, en donde haga match el valor de la intensidad con el valor del nivel de la salida de gris, ahí se 'pintará' con esa nueva intensidad de salida el valor de *r*.

Para el histogram matching, se repite el proceso anterior con un nuevo vector de 1 a 256 para almacenar las ocurrencias y obtener la nueva salida de nivel de gris, esto con la imagen que ha sido ya ecualizada anteriormente.

```
%------------------Histogram Matching-----------------------------------
escalaGrisHM = 1:256;
%para sacar la cantidad de ocurrencias de cada nivel de gris
for i = 1:size(imagenGris1,1)
    for j = 1:size(imagenGris1,2)
        r = imagenGris1(i,j);
        if r == 0
            r = 1;
            escalaGrisHM(r) = escalaGris(r)+1;
        %else
         %   escalaGris(r) = escalaGris(r)+1;
        end
        escalaGrisHM(r) = escalaGris(r)+1;
    end
end
```

Luego se obtiene el histograma de la imagen que se ecualizó en el paso anterior, el proceso es el mismo.

```
salidaNivelGris1 = 1:256;
%para calcular el histograma nuevo
for i = 1:256
    pdf = escalaGrisHM(i)/totalPixeles;
    cdf = cdf + pdf;
    salidaNivelGris1(i) = l * cdf;
    salidaNivelGris1(i) = round(salidaNivelGris1(i));
    if salidaNivelGris1(i) > 255
        salidaNivelGris1(i) = 255;
    end
end
imagenGris2 = imagenGris1;
 %para ecualizar el histograma nuevo
 for i = 1:size(imagenGris2,1)
    for j = 1:size(imagenGris2,2)
        r = imagenGris2(i,j);
        for k = 1:256
            if r == k
                imagenGris2(i,j) = salidaNivelGris(r);
            end
        end
    end
 end
```

Se obtiene el histograma de la misma forma, pero ahora la nueva entrada, será la salida de la imagen ecualizada y se ecualiza ahora la nueva para que haga match con el histograma de la otra imagen.

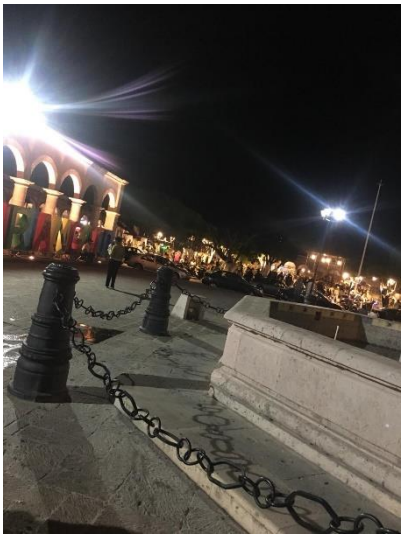
Por último, se muestran las imágenes en los axes de la interfaz y se activa las etiquetas correspondientes.

```
imshow(imagenGris, 'Parent', imagenEntrada); %mostrar la imagen en escala de grises en el axes imagenEntrada
imshow(imagenGris1,'Parent',handles.salida1);    %mostrar la salida ya ecualizada
imshow(imagenGris2,'Parent',handles.salida3);    %mostrar la salida ya haciendo matching
set(handles.histogram,'Visible','on');
set(handles.histogramMatching,'Visible','on');
```

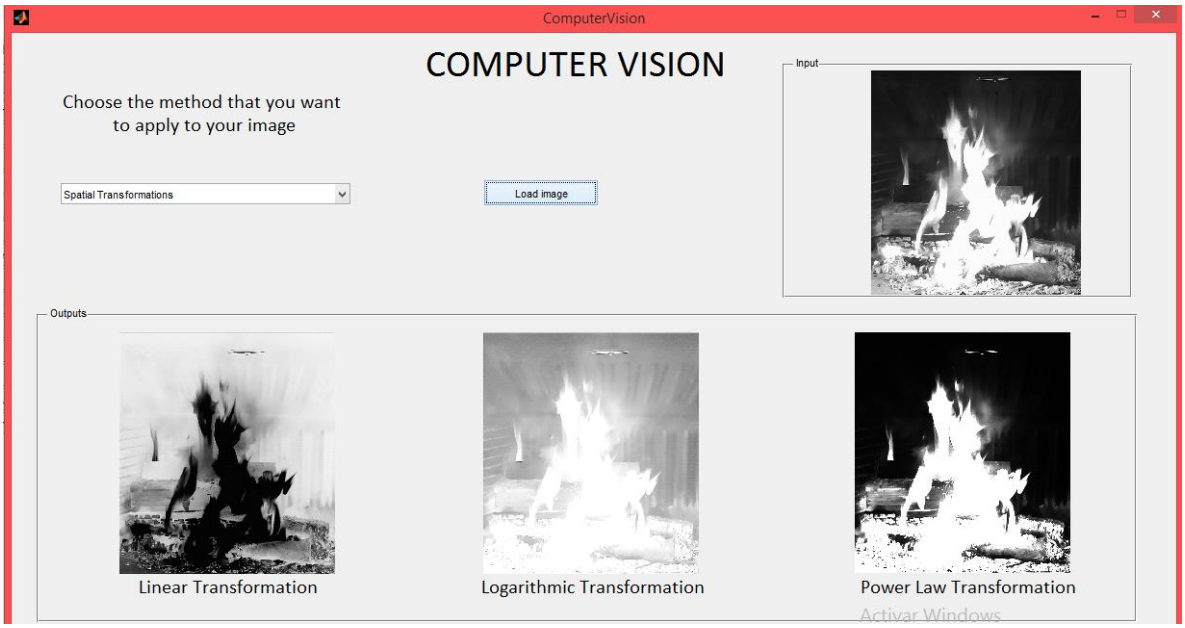
IV. PRUEBAS

Las pruebas que se realizaron fueron con las siguientes 3 imágenes que se muestran son 2 que han sido proporcionadas por la profesora en clase y otra que fue tomada con un teléfono celular para esta práctica.

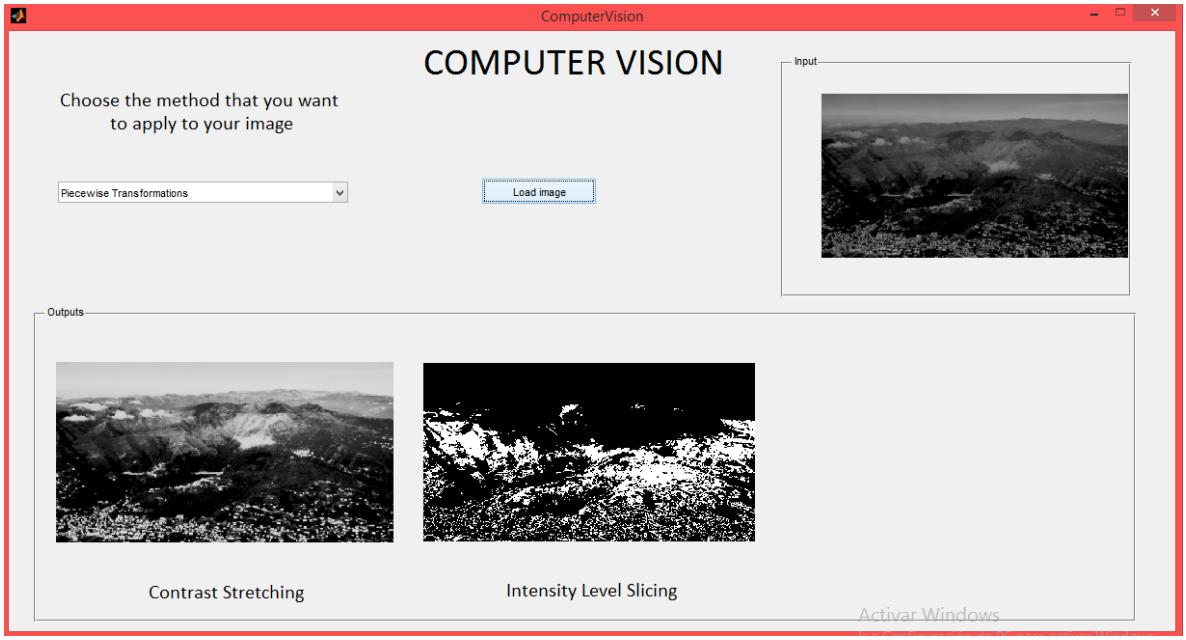
Las imágenes se muestran a continuación:



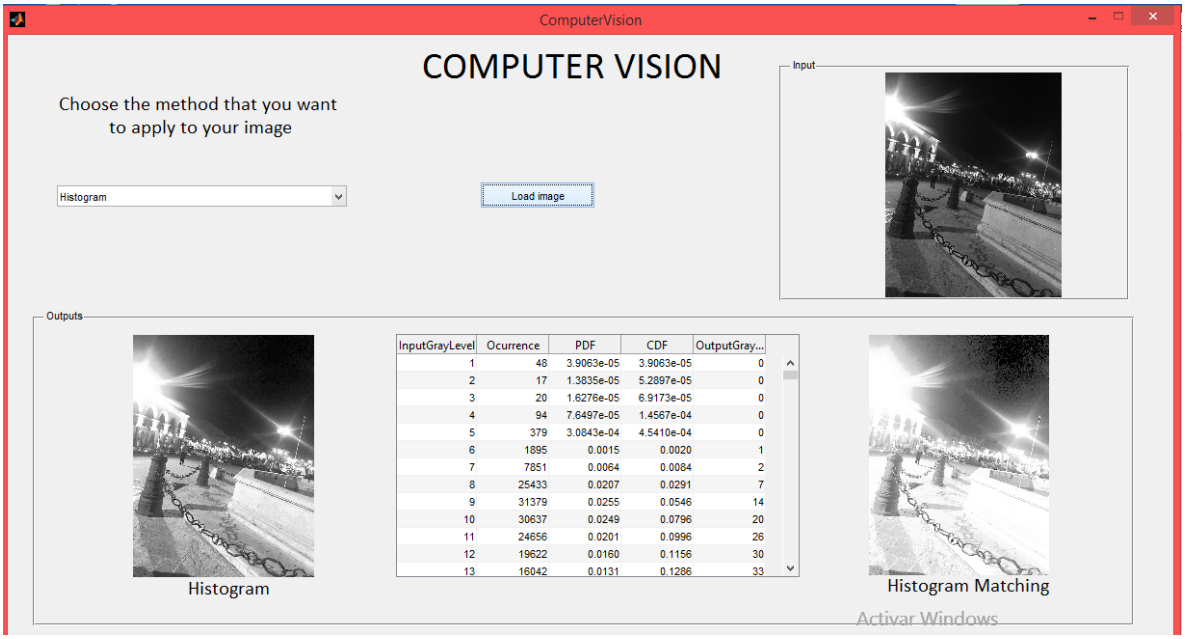
Se mostrará una el resultado de cada proceso para una imagen. Es decir, una imagen corresponderá a cada método. El resultado fue el siguiente:



El resultado para transformaciones por partes es:



Y el resultado para el histograma es:



V. ANÁLISIS DE LOS RESULTADOS

Los resultados que se obtuvieron de las imágenes que se muestran en las pruebas son bastante buenos. Por ejemplo, en el caso de histograma la imagen original, en escala de grises se percibe muy oscura y sufre un cambio en contraste. Se observa ya más iluminada y al recibir esta como entrada al hacer match con el histograma actual, esta se percibe aún más clara.

Para el caso de las transformaciones por partes, el constrast stretching resulta bastante eficiente ya que, al ampliar el rango de forma dinámica, se logran visualizar las nubes y las montañas de una forma más clara. Evidentemente esto trajo consigo una mejora en la calidad de la imagen de entrada. Se puede observar si se compara con la imagen original, que el contraste aumentó y que se distingue de mejor manera el cielo de la imagen, las montañas y las nubes en la parte izquierda de la imagen también pueden apreciarse mejor. Aunque existe una zona en la imagen que está muy oscura que es casi en la parte central, debajo de las montañas. Definitivamente este es la mejor opción para mejorar esta imagen, ya que el constrast stretching se dice que normaliza la imagen, porque "estira" el rango de valores de intensidad que contiene para abarcar un rango de valores deseado. Los puntos (r1, s1) y (r2, s2) controlan la forma del contraste.

Y por último para el primer caso de las transformaciones espaciales, aunque logran su objetivo, se considera que con un caso de estudio de rayos x, o médico se pudiera aplicar de mejor forma estos algoritmos de transformaciones espaciales. El mejor de estos tres procesos, de forma personal, es la transformación lineal.

VI. REFERENCIAS

[1] Rafael C. González-Richard E. Woods, 1977, "Digital Image Processing", Pearson-Prentice Hall.