



UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MÉXICO
FACULTAD DE INGENIERÍA
INGENIERIA EN COMPUTACIÓN

PROCESAMIENTO DE IMÁGENES DIGITALES

DOCUMENTACIÓN

PROFESORA: VIANEY MUÑOZ JIMÉNEZ

INTEGRANTES:

- **MEJIA RIVAS DANIELA.**
- **HERNÁNDEZ VALLEJO AURA QUETZALLI.**
- **ESCALERA JIMENEZ ENRIQUE.**
- **DIEGO ARMANDO GÓMEZ JIMÉNEZ.**
- **HERNANDEZ MARTINEZ EDUARDO AXEL.**

2023A

FECHA DE ENTREGA: 21-JUNIO-2023

Índice

Introducción	3
Marco teórico	5
Requerimientos de instalación y ejecución	7
Conclusión	19
Fuentes referenciales.....	21
Código Fuente.	22
Programa Compilado.	27

Introducción

Bienvenido a la documentación del proyecto de procesamiento de imágenes digitales desarrollado en MATLAB. Esta guía tiene como objetivo proporcionar una descripción detallada de las funcionalidades y características de este software, diseñado para realizar diversas operaciones en imágenes digitales.

El procesamiento de imágenes digitales es una disciplina que abarca una amplia gama de técnicas y algoritmos destinados a mejorar, analizar y manipular imágenes capturadas por dispositivos electrónicos. En este proyecto, hemos utilizado el poderoso entorno de programación de MATLAB para implementar un conjunto de funciones y herramientas que permiten realizar diferentes operaciones sobre imágenes digitales.

El software desarrollado ofrece una amplia variedad de funcionalidades, brindando a los usuarios la capacidad de aplicar operaciones morfológicas, ecualización de histograma, filtros, eliminación de ruido y segmentación, entre otras técnicas. Estas operaciones permiten mejorar la calidad de las imágenes, resaltar características de interés y extraer información relevante para su posterior análisis.

A través de una interfaz intuitiva y amigable, los usuarios pueden cargar imágenes en diferentes formatos, visualizarlas, aplicar las operaciones disponibles y guardar los resultados obtenidos.

En esta documentación, encontrarás información detallada sobre el uso de cada una de las operaciones disponibles, así como ejemplos prácticos y recomendaciones para obtener los mejores resultados. También se proporcionarán detalles técnicos sobre los algoritmos utilizados en la implementación de estas operaciones, lo que permitirá a los usuarios comprender el funcionamiento interno del software y adaptarlo a sus necesidades específicas.

Esperamos que esta documentación te sea de gran ayuda para aprovechar al máximo todas las capacidades y funcionalidades del software de procesamiento de imágenes digitales desarrollado en MATLAB. ¡Comencemos a explorar el

fascinante mundo del procesamiento de imágenes y descubre las infinitas posibilidades que ofrece este proyecto!

Marco teórico

El marco teórico del proyecto de procesamiento de imágenes digitales en MATLAB se basa en aspectos teóricos y conceptuales relevantes obtenidos a través de una exhaustiva investigación. Este marco proporciona la base teórica necesaria para comprender el problema abordado y desarrollar una alternativa de solución efectiva.

El procesamiento de imágenes digitales se basa en un conjunto de principios y técnicas que permiten manipular y mejorar imágenes capturadas por dispositivos electrónicos. Para lograr esto, es fundamental comprender conceptos clave desde mejoras en el contraste y manipulación de intensidades hasta detección de bordes y segmentación de objetos.

La ecualización de histograma es una técnica utilizada para mejorar el contraste de una imagen al redistribuir los niveles de intensidad de los píxeles de manera más uniforme. Esto resalta las características y detalles presentes en la imagen, mejorando su calidad visual.

La inversión binaria y fotográfica son operaciones básicas que permiten cambiar los valores de intensidad en una imagen. La inversión binaria invierte los píxeles oscuros en claros y viceversa, mientras que la inversión fotográfica invierte los valores de intensidad de manera proporcional, creando un efecto negativo de la imagen original.

Para combinar contenidos de píxeles de diferentes imágenes, se utilizan operaciones de adición y sustracción. La adición suma los valores de intensidad de los píxeles correspondientes, lo que puede ser útil para fusionar imágenes. La sustracción resta los valores de intensidad, permitiendo resaltar diferencias y detectar cambios entre imágenes.

Las operaciones de rotación y espejo se utilizan para modificar la orientación y perspectiva de una imagen. La rotación implica girar la imagen en un ángulo

determinado alrededor de un punto central, mientras que el espejo refleja la imagen horizontal o verticalmente.

Para reducir el ruido en una imagen, se emplean filtros como el de moda, media y mediana. El filtro de moda reemplaza cada píxel por el valor de intensidad más común en su vecindario, el filtro de media calcula el promedio de los valores de intensidad, y el filtro de mediana utiliza el valor mediano. Estos filtros mejoran la calidad visual y claridad de la imagen al reducir el impacto del ruido.

El filtro gaussiano suaviza una imagen al reducir las variaciones bruscas de intensidad entre píxeles vecinos. Se aplica una convolución con una función gaussiana, lo que resulta en una imagen más suave y sin detalles no deseados.

Para detectar bordes y contornos en una imagen, se utilizan operadores como Prewitt, Sobel y Roberts. Estos operadores aplican máscaras y cálculos de gradiente para identificar las transiciones bruscas de intensidad, resaltando las estructuras de borde en la imagen.

Las operaciones de erosión y dilatación son fundamentales en el procesamiento morfológico de imágenes. La erosión reduce el tamaño de los objetos y elimina pequeños detalles, mientras que la dilatación aumenta el tamaño de los objetos y rellena huecos en ellos.

Finalmente, la segmentación es una técnica que divide una imagen en regiones u objetos significativos. Permite identificar y separar diferentes elementos en la imagen, facilitando su análisis y posterior procesamiento.

Estos conceptos y técnicas forman la base teórica necesaria para comprender y aplicar de manera efectiva las operaciones de procesamiento de imágenes en MATLAB, en el contexto del proyecto de procesamiento de imágenes digitales desarrollado.

Requerimientos de instalación y ejecución

1. Requerimientos de Hardware:

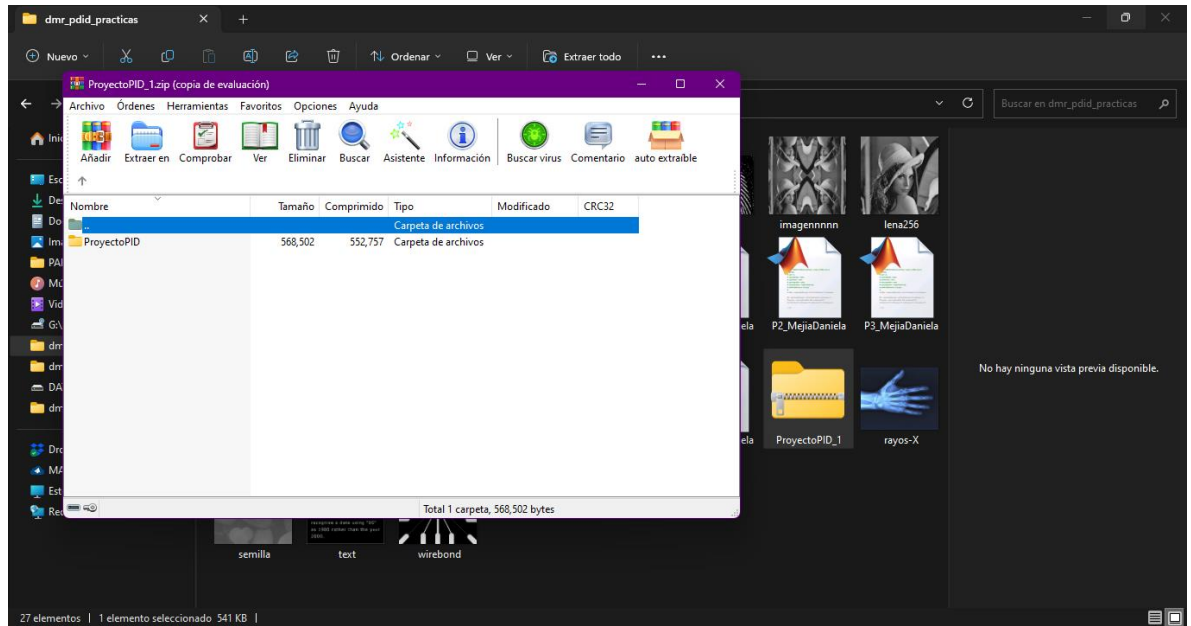
- Procesador: Se recomienda un procesador de al menos 2 GHz para un rendimiento óptimo.
- Memoria RAM: Se sugiere un mínimo de 4 GB de RAM, aunque la cantidad exacta puede variar según el tamaño y la complejidad de las imágenes procesadas.
- Almacenamiento: Espacio suficiente en disco para almacenar imágenes de entrada y salida, así como archivos auxiliares generados durante el procesamiento.
- Tarjeta gráfica: No es un requisito específico, pero una tarjeta gráfica compatible con MATLAB puede acelerar ciertas operaciones y visualizaciones.

2. Requerimientos de Software:

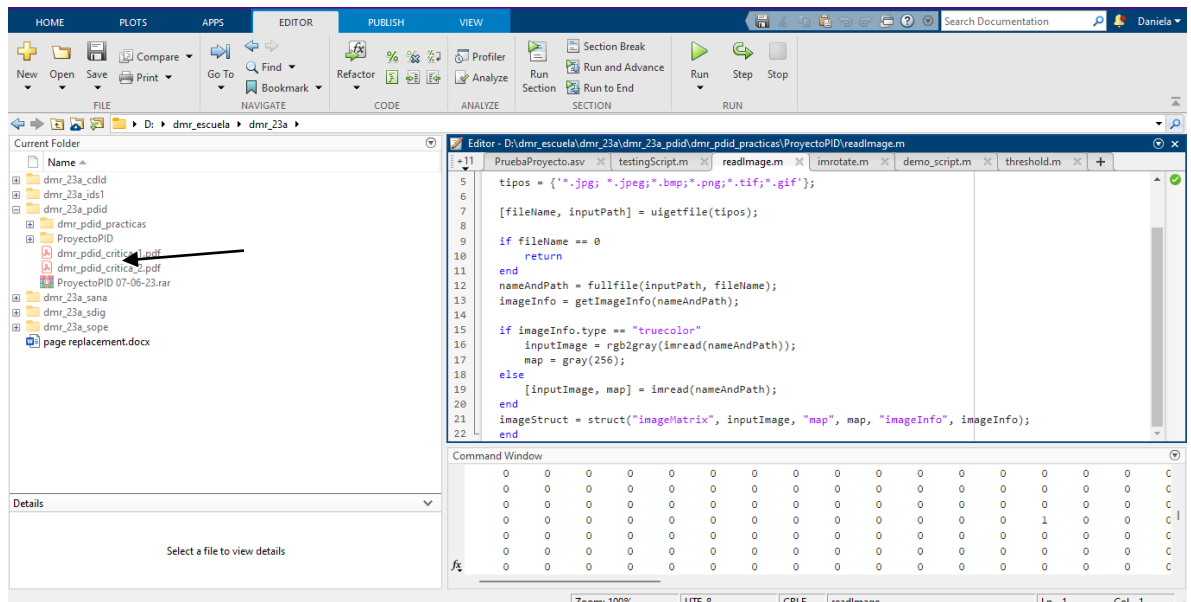
- MATLAB: Se debe tener una instalación de MATLAB cuya versión sea compatible con el toolbox de Image Processing. Se recomienda utilizar la versión más reciente disponible para aprovechar las últimas características y mejoras.
- Image Processing Toolbox: Asegúrate de tener instalada la toolbox de procesamiento de imágenes de MATLAB, ya que contiene funciones y herramientas específicas para el procesamiento y análisis de imágenes digitales.

3. Instrucciones de Instalación:

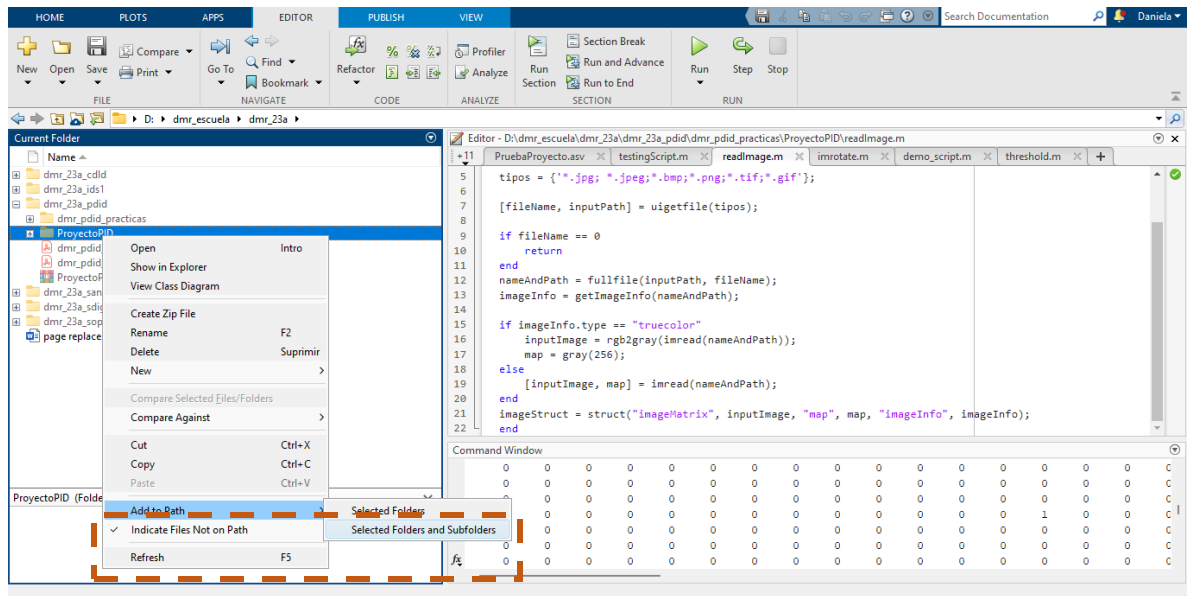
1. Descarga la carpeta del proyecto y descomprímela.



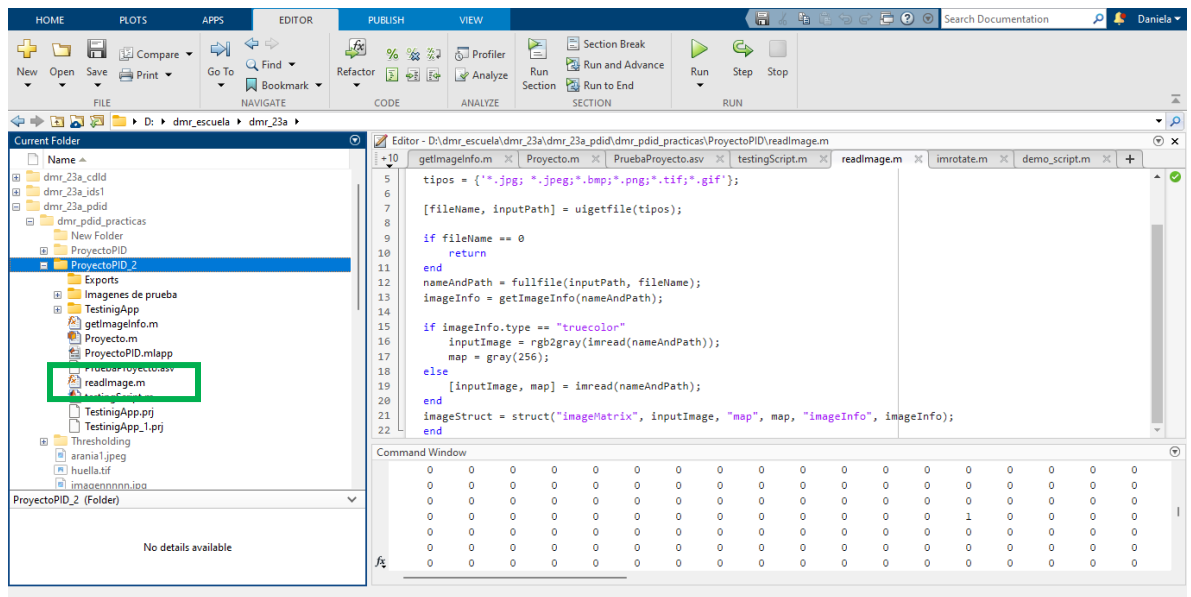
2. Una vez descomprimida, abre Matlab y busca en tus carpetas el folder que descomprimiste anteriormente.



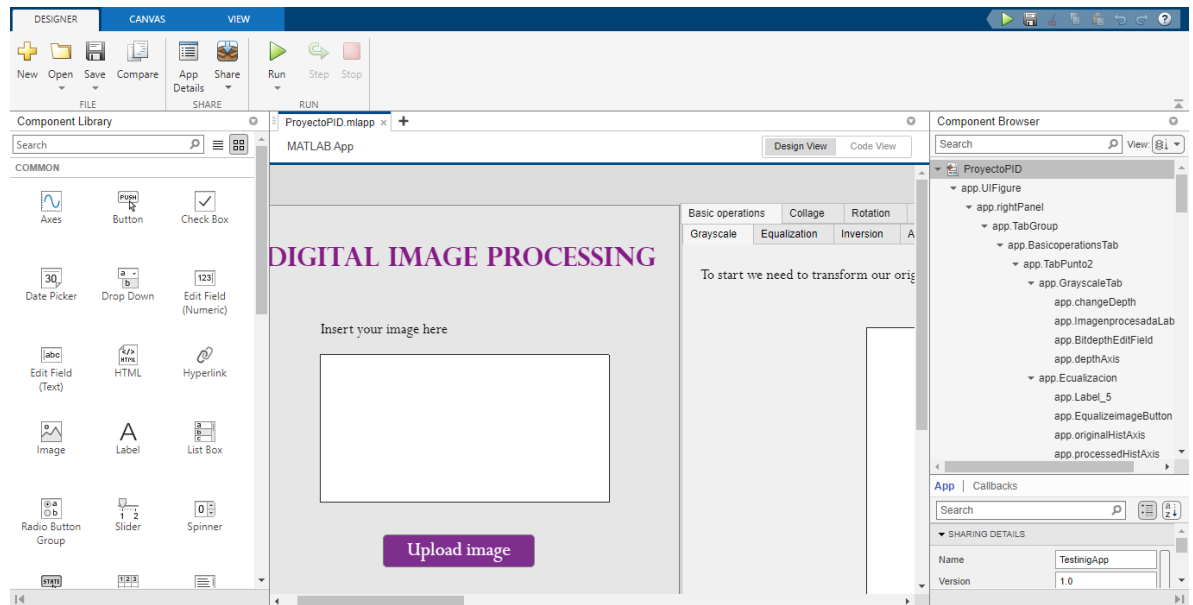
3. Tendrás que darle clic derecho y seleccionar Add to path>Selected Folders and Subfolders para que el folder se pueda abrir en Matlab.



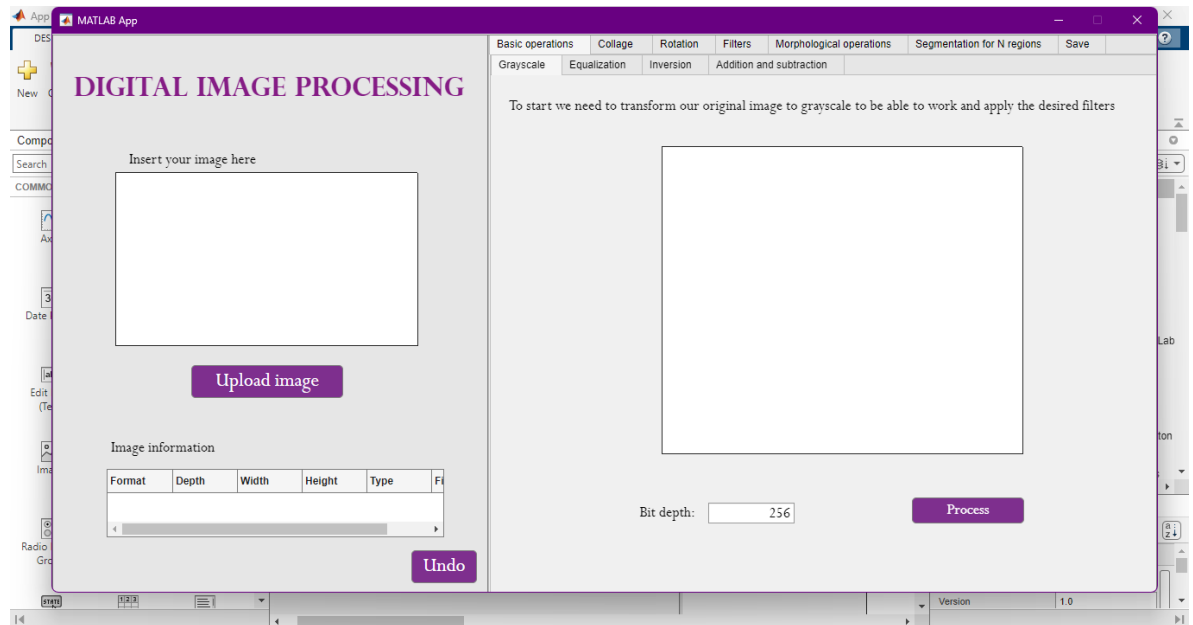
4. Selecciona el archivo que se llama ProyectoPID.mlapp



5. Se va a abrir el App Designer



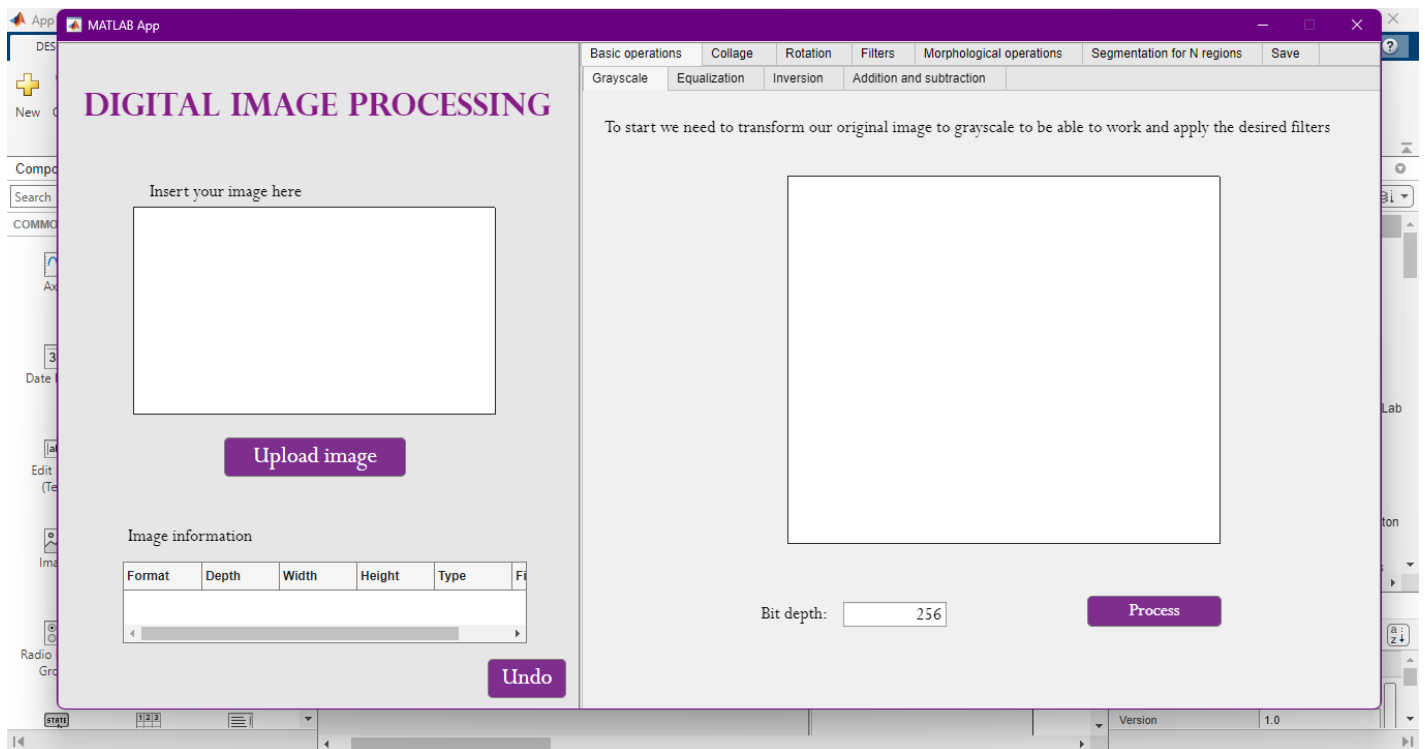
6. Dar clic en el botón Run que se encuentra en la parte posterior y así es como se va a empezar a ejecutar el programa.



- Detalla cómo ejecutar el programa o los scripts en MATLAB. Puede ser a través de la línea de comandos de MATLAB o mediante la carga y ejecución de archivos de script específicos.
- Proporciona instrucciones claras sobre cómo cargar imágenes de entrada, cómo ajustar los parámetros y cómo obtener los resultados esperados.

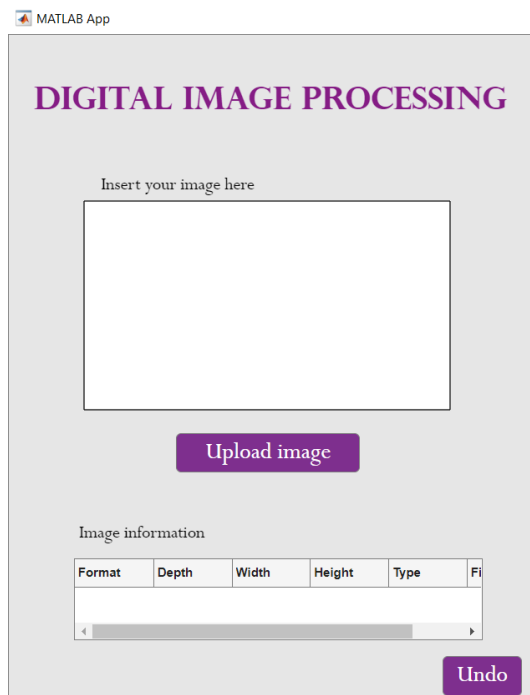
4. Instrucciones de Ejecución:

Pantalla principal de la aplicación



La aplicación es capaz de abrir, guardar y trabajar con imágenes digitales en diversos formatos estándares: JPG, TIF, BMP, PPM. Además, permite realizar diferentes tipos de procesamiento de imágenes

Inserción de imágenes



En esta parte de la aplicación tenemos esta funcionalidad que permite a los usuarios seleccionar una imagen de su dispositivo y cargarla en la aplicación.

Además, le permite al usuario conocer información de la imagen como:

- formato
- profundidad
- anchura
- altura
- tipo
- tamaño del archivo

Menú de operaciones

Basic operations	Collage	Rotation	Filters	Morphological operations	Segmentation for N regions	Save	
Grayscale	Equalization	Inversion	Addition and subtraction				

El menú de operaciones se encuentra en la parte superior de la interfaz este menú nos proporciona el acceso a las funcionalidades o acciones que los usuarios pueden realizar.

Las operaciones a las que el usuario tiene disposición son:


Basic operations (operaciones básicas)

- Escala de grises

MATLAB App

DIGITAL IMAGE PROCESSING

Insert your image here



Upload image

Image information

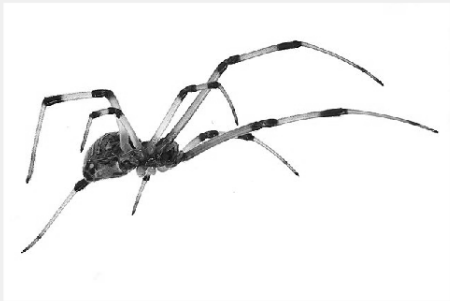
Format	Depth	Width	Height	Type	Fi
jpg	24	640	427	truecolor	

Undo

Basic operations Collage Rotation Filters Morphological operations Segmentation for N regions Save

Grayscale Equalization Inversion Addition and subtraction

To start we need to transform our original image to grayscale to be able to work and apply the desired filters



Bit depth: 256


Process

- Ecualización.

MATLAB App

DIGITAL IMAGE PROCESSING

Insert your image here



Upload image

Image information

Format	Depth	Width	Height	Type	File
jpg	24	640	427	truecolor	

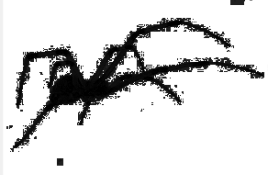
Undo

Basic operations | Collage | Rotation | Filters | Morphological operations | Segmentation for N regions | Save


Grayscale | **Equalization** | Inversion | Addition and subtraction

Automatically adjusts the lightness of the colors in the active layer so that the value channel's histogram is as smooth as possible, that is, each possible lightness value appears at the same number of pixels as for the other values.


Original grayscale image




Equalized image



Original image histogram



Equalized Image Histogram




Equalize image

- Inversión (binaria y fotográfica)

MATLAB App

DIGITAL IMAGE PROCESSING

Insert your image here



Upload image

Image information

Format	Depth	Width	Height	Type	File
jpg	24	640	427	truecolor	

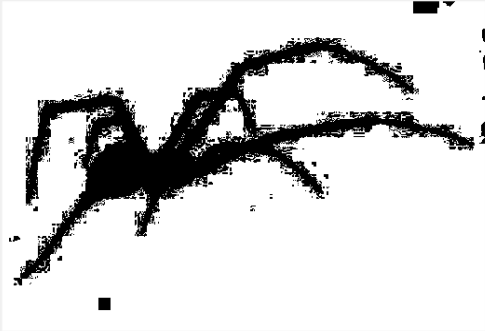
Undo

Basic operations | Collage | Rotation | Filters | Morphological operations | Segmentation for N regions | Save

Grayscale | Equalization | **Inversion** | Addition and subtraction

The determination of the opposite color in that color model is called inversion of a color. It is an operation that affects each pixel, regardless of its neighborhood. The reverse image can be thought of as "the reverse" or the "negative" of an image.

Inverted image




Inversion | Binary investment | Photo investment

- Adición y sustracción (para imágenes en escala de gris)

MATLAB App

DIGITAL IMAGE PROCESSING

Insert your image here



Upload image

Image information

Format	Depth	Width	Height	Type	Fi
jpg	24	640	427	truecolor	

Undo

Basic operations Collage Rotation Filters Morphological operations Segmentation for N regions Save

Grayscale Equalization Inversion Addition and subtraction

The operation adds and subtracts between images consists of merging the values of 2 consists of merging the values of 2 input images

Image 1






Image 2



Upload image

Processed image



Addition

Subtraction

Collage (tapiz)


Rotation and mirror(rotación y espejo)

- Rotación

MATLAB App

DIGITAL IMAGE PROCESSING

Insert your image here



Upload image

Image information

Format	Depth	Width	Height	Type	File
jpg	24	640	427	truecolor	

Undo

Basic operations

Collage

Rotation

Filters


Morphological operations

Segmentation for N regions

Save

A rotation is a transformation where a figure is rotated around a fixed point to create an image.

Insert multiples of 45 degrees



Rotation angle

Rotate image


Mirror

▪ Espejo

MATLAB App

DIGITAL IMAGE PROCESSING

Insert your image here



Upload image

Image information

Format	Depth	Width	Height	Type	File
jpg	24	640	427	truecolor	

Undo

Basic operations

Collage

Rotation

Filters

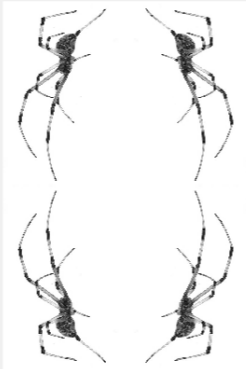
Morphological operations

Segmentation for N regions

Save

A rotation is a transformation where a figure is rotated around a fixed point to create an image.

Insert multiples of 45 degrees



Rotation angle

Rotate image

Mirror

15


Filters (Filtros).

- Filtro de moda, media y mediana.
- Máximos y mínimos (para $n=1 \dots 9$)

MATLAB App

DIGITAL IMAGE PROCESSING

Insert your image here



Upload image

Image information

Format	Depth	Width	Height	Type	Fi
jpg	24	640	427	truecolor	

Undo

Basic operations Collage Rotation Filters Morphological operations Segmentation for N regions Save

Smoothing filters Statistical order Edge detection

They are useful when the image is supposed to have a lot of noise and you want to remove it. They can also be used to highlight information corresponding to a certain scale (size of the filter matrix).

Image average Filter

Image fashion filter

Image median filter


Image max filter

Image min filter

N Order filter:

Filter

Processed image

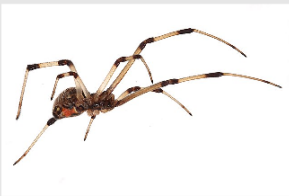


- Filtro Gaussiano.
- Filtro Laplaciano para 4 y 8 vecinos.

MATLAB App

DIGITAL IMAGE PROCESSING

Insert your image here



Upload image

Image information

Format	Depth	Width	Height	Type	Fi
jpg	24	640	427	truecolor	

Undo

Basic operations Collage Rotation Filters Morphological operations Segmentation for N regions Save

Smoothing filters Statistical order Edge detection

Smoothing filters are intended to reduce noise and/or effects that may appear in an image as a result of the capture, digitization, and transmission process.

The Gaussian filter preserves edges, reduces noise by smoothing the image but the image is not sharp.


Gaussian Filter

The Laplacian filter is recommended for enhancing linear features in urban environments.

8-Laplacian

4-Laplacian

Processed image




- Detección de contornos (Prewitt, Sobel y Roberts)

MATLAB App

DIGITAL IMAGE PROCESSING

Insert your image here



Upload image

Image information

Format	Depth	Width	Height	Type	Fi
jpg	24	640	427	truecolor	

Undo

Basic operations Collage Rotation Filters Morphological operations Segmentation for N regions Save

Smoothing filters Statistical order Edge detection

They look for the edges between different colors and thus can detect the outlines of objects. They are used to make selections and for other artistic proposals. Most of them are based on gradient calculation methods and generate thick lines.

It has the ability to show edge points but not their orientation.

Roberts


Gets the gradient for the edges. Although the mask is different

Prewitt

Detects the horizontal and vertical edges separately on a grayscale image.

Sobel

Processed image




Morphological operations (Operaciones morfológicas)

- Erosión
- Dilatación

MATLAB App

DIGITAL IMAGE PROCESSING

Insert your image here



Upload image

Image information

Format	Depth	Width	Height	Type	Fi
jpg	24	640	427	truecolor	

Undo

Basic operations Collage Rotation Filters Morphological operations Segmentation for N regions Save

Structuring element

0	0	0
0	1	0
0	0	0


Apply dilation

*Dilation adds pixels to the boundaries of objects in an image

Apply erosion

*Erosion remove pixels from the boundaries of objects

Processed image

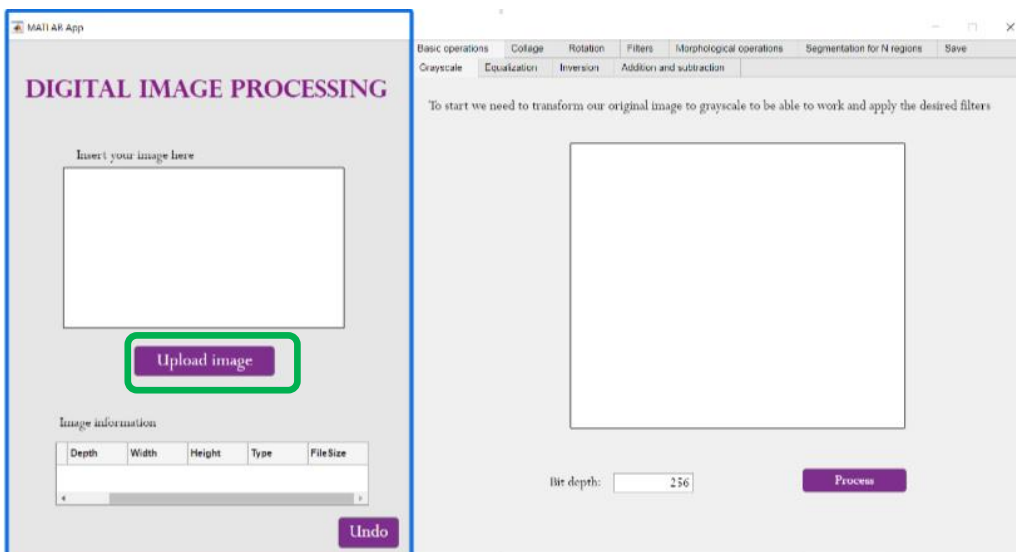


Segmentation for N regions (Segmentación para N regiones)

Funcionamiento

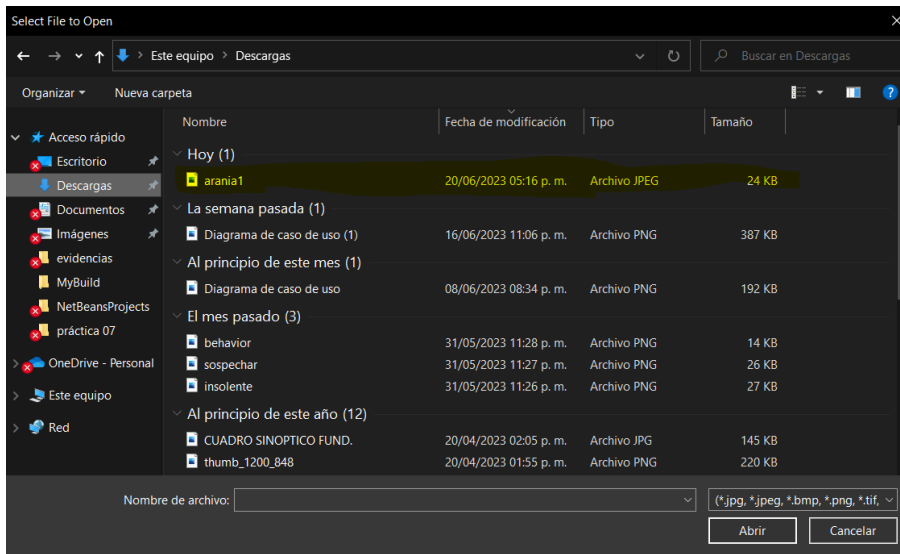
Inserción de imagen

la aplicación permite a los usuarios seleccionar una imagen de su dispositivo y cargarla en la aplicación.



- 1 Buscar en el área de inserción el botón Upload image
- 2 Hacer clic en el botón Upload image

3. Seleccionar una imagen y dar click en abrir



Conclusión

En conclusión, la documentación del proyecto de procesamiento de imágenes digitales desarrollado en MATLAB ha proporcionado una visión completa de las funcionalidades y capacidades de este software. A lo largo de esta guía, hemos explorado las diversas operaciones que pueden realizarse, como operaciones morfológicas, ecualización de histograma, filtrado, eliminación de ruido y segmentación, entre otras.

El software desarrollado en MATLAB ha demostrado ser una herramienta potente y versátil para el procesamiento de imágenes digitales. Su interfaz intuitiva y amigable facilita a los usuarios la carga, visualización y manipulación de imágenes de manera eficiente. Además, la optimización del rendimiento ha permitido tiempos de procesamiento rápidos y eficientes, incluso para imágenes de gran tamaño.

A lo largo de esta documentación, se han proporcionado ejemplos prácticos y recomendaciones para el uso adecuado de cada operación, lo que permite a los

usuarios aprovechar al máximo el potencial del software y obtener resultados de alta calidad en sus tareas de procesamiento de imágenes.

Además, la explicación técnica de los algoritmos utilizados en la implementación de estas operaciones ha brindado a los usuarios una comprensión más profunda de los fundamentos y principios detrás del procesamiento de imágenes digitales. Esto les ha permitido adaptar y personalizar el software según sus necesidades específicas, así como explorar nuevas técnicas y algoritmos en el campo del procesamiento de imágenes.

En resumen, este proyecto de procesamiento de imágenes digitales en MATLAB ha proporcionado una solución completa y eficiente para mejorar, analizar y manipular imágenes digitales. Esperamos que esta documentación haya sido una guía útil y valiosa para aprovechar al máximo todas las capacidades del software y desatar la creatividad y el potencial de los usuarios en el emocionante campo del procesamiento de imágenes digitales.

Fuentes referenciales

1. González, R. C., Woods, R. E., & Eddins, S. L. (2009). Digital Image Processing Using MATLAB. Gatesmark Publishing.
2. Yaroslavsky, L. P. (2001). Digital Image Processing Using MATLAB. Springer.
3. Burger, W., & Burge, M. J. (2016). Digital Image Processing: Concepts, Algorithms, and Scientific Applications. Springer.
4. González, R. C., & Woods, R. E. (2008). Digital Image Processing (3rd ed.). Pearson.
5. MathWorks Documentation. Recuperado de: <https://www.mathworks.com/help/images/index.html>
6. MATLAB Central. Recuperado de: <https://www.mathworks.com/matlabcentral/>
7. IEEE Xplore. Recuperado de: <https://ieeexplore.ieee.org/Xplore/home.jsp>
8. ACM Digital Library. Recuperado de: <https://dl.acm.org/>

Código Fuente.

```
Editor - C:\Users\dgome\OneDrive\Documentos\ProcesamientoDeImágenes\ProyectoPID\testingScript.m
getImageInfo.m  Proyecto.m  readImage.m  testingScript.m  +
1  % [treesImage, treesMap] = imread("trees.tif");
2  % indexedTree = treesImage;
3  %
4  % [M1, M2, M3] = imread("corn.tif");
5  % imread("corn.tif")
6  % M1(1,1,3);
7  % imshow(M1);
8  % imageInfo1 = imfinfo("trees.tif");
9  % imageInfo2 = imfinfo("corn.tif");
10 % %imageInfo = struct(bitDepth, imageInfo1.BitDepth, size, )
11 % %indexedTrees = [treesImage, treesMap];
12 % %indexedTrees
13 % %imshow(indexedTrees[1], indexedTrees[2]);
14 %
15 % someStruct = getImageInfo("corn.tif");
16 % [anotherImage, anotherMap] = imread("trees.tif");
17 % convertedImage = ind2gray(anotherImage, anotherMap);
18 % anotherMap = gray(256);
19 % imshow(convertedImage, anotherMap);
20
21 firstImage = Proyecto;
22
```

```
Editor - C:\Users\dgome\OneDrive\Documentos\ProcesamientoDeImágenes\ProyectoPID\readImage.m
getImageInfo.m  Proyecto.m  readImage.m  testingScript.m  +
1  function [nameAndPath, imageStruct] = readImage()
2  %readImage() use this to retrieve an image's full path and a structure
3  %containing the image
4  %
5  tipos = {'*.jpg; *.jpeg;*.bmp;*.png;*.tif;*.gif'};
6
7  [fileName, inputPath] = uigetfile(tipos);
8
9  if fileName == 0
10     return
11 end
12 nameAndPath = fullfile(inputPath, fileName);
13 imageInfo = getImageInfo(nameAndPath);
14
15 if imageInfo.type == "truecolor"
16     inputImage = rgb2gray(imread(nameAndPath));
17     map = gray(256);
18 else
19     [inputImage, map] = imread(nameAndPath);
20 end
21 imageStruct = struct("imageMatrix", inputImage, "map", map, "imageInfo", imageInfo);
22 end
```

```
Editor - C:\Users\dgome\OneDrive\Documentos\ProcesamientoDeImágenes\ProyectoPID\getImageInfo.m
getImageInfo.m x Proyecto.m x readImage.m x testingScript.m x +
1 function [imageInfoStruct] = getImageInfo(fileName)
2 %UNTITLED Summary of this function goes here
3 % Detailed explanation goes here
4 imageInfo = imfinfo(fileName);
5 imageFileSize = imageInfo.FileSize;
6 imageFormat = imageInfo.Format;
7 imageBitDepth = imageInfo.BitDepth;
8 imageWidth = imageInfo.Width;
9 imageHeight = imageInfo.Height;
10 imageType = imageInfo.ColorType;
11 imageInfoStruct = struct("format", imageFormat, "depth", imageBitDepth, ...
12     "width", imageWidth, "height", imageHeight, "type", imageType, ...
13     "FileSize", imageFileSize);
14 end
```

```
Editor - C:\Users\dgome\OneDrive\Documentos\ProcesamientoDeImágenes\ProyectoPID\Proyecto.m
getImageInfo.m x Proyecto.m x readImage.m x testingScript.m x +
1 classdef Proyecto
2     %Proyecto App de PID
3
4     properties
5         nameAndPath
6         name
7         imageMatrix
8         imageMap
9         binaryImage
10        grayScaleImage
11        imageHistogram
12        imageInfo
13        equalizedImage
14        invertedImage
15        addImage
16        susImage
17        erImage
18        dilImage
19        eqHistogram
20    end
21
22    methods
23        function obj = Proyecto()
```

Command Window

```

Editor - C:\Users\dgome\OneDrive\Documentos\ProcesamientoDeImágenes\ProyectoPID\Proyecto.m
getImageInfo.m  Proyecto.m  readImage.m  testingScript.m  +

23     function obj = Proyecto()
24         %UNTITLED Construct an instance of this class
25         % Detailed explanation goes here
26         [locFile, imageStruct] = readImage();
27         obj.nameAndPath = locFile;
28         obj.imageMatrix = imageStruct.imageMatrix;
29         obj.imageMap = imageStruct.map;
30         obj.imageInfo = imageStruct.imageInfo;
31
32         % ----- Codigo en prueba -----
33
34         if isempty(obj.imageMap) % image is RGB or grayscale
35             if (size(obj.imageMatrix, 3) == 1) % image is grayscale
36                 obj.grayScaleImage = obj.GraytoGray.grayScaleImage;
37             else
38                 obj.grayScaleImage = obj.RGBtoGrayScale.grayScaleImage;
39             end
40         else % image is indexed
41             obj.grayScaleImage = obj.toGrayScale.grayScaleImage;
42         end
43
44         %-----vvv SECCION MODIFICADA 7-6-23 (LECTURA DE IMG BINARIAS) vvv-
45         if(islogical(obj.imageMatrix))

```

```

Editor - C:\Users\dgome\OneDrive\Documentos\ProcesamientoDeImágenes\ProyectoPID\Proyecto.m
getImageInfo.m  Proyecto.m  readImage.m  testingScript.m  +

44         %-----vvv SECCION MODIFICADA 7-6-23 (LECTURA DE IMG BINARIAS) vvv-
45         if(islogical(obj.imageMatrix))
46             obj.grayScaleImage = uint8(255*obj.imageMatrix);
47             obj.imageMatrix = uint8(255*obj.imageMatrix);
48
49         end
50
51         %-----^^^ SECCION MODIFICADA 7-6-23 (LECTURA DE IMG BINARIAS) ^^^-
52
53         % ----- Codigo en prueba -----
54
55         %obj.grayScaleImage = obj.toGrayScale.grayScaleImage;
56
57         obj.binaryImage = obj.binarizeImage.binaryImage;
58
59         %Modificado 7/6/23
60         obj.imageHistogram = obj.getHistogram(obj.grayScaleImage).imageHistogram;
61
62         A = split(obj.nameAndPath, '\');
63         A = A(length(A));
64         A = split(A, '.');
65         A = A(1);
66         A = [A, ' Processed'];

```

Command Window


```
Editor - C:\Users\dgome\OneDrive\Documentos\ProcesamientoDeImágenes\ProyectoPID\Proyecto.m
getImageInfo.m  Proyecto.m  readImage.m  testingScript.m  +
66     A = [A, '_Processed'];
67     A = join(A);
68     A = string(A);
69     obj.name = A;
70     end
71
72     function [Proyecto] = binarizeImage(Proyecto)
73         binImage = imbinarize(Proyecto.imageMatrix);
74         %binImage = imbinarize(cast(Proyecto.imageMatrix,'uint8'));
75         Proyecto.binaryImage = binImage;
76     end
77
78     function [Proyecto] = toGrayScale(Proyecto)
79         Proyecto.grayScaleImage = ind2gray(Proyecto.imageMatrix, Proyecto.imageMap);
80     end
81
82     function [Proyecto] = GraytoGray(Proyecto)
83         Proyecto.grayScaleImage = Proyecto.imageMatrix;
84     end
85
86     function [Proyecto] = RGBtoGrayScale(Proyecto)
87         Proyecto.grayScaleImage = rgb2gray(Proyecto.imageMatrix);
88     end
```

```
Editor - C:\Users\dgome\OneDrive\Documentos\ProcesamientoDeImágenes\ProyectoPID\Proyecto.m
getImageInfo.m  Proyecto.m  readImage.m  testingScript.m  +
88     end
89
90     function [Proyecto] = changeDepth(Proyecto,value)
91         Proyecto.imageMap = gray(value);
92         Proyecto.imageInfo.depth = value;
93     end
94
95     function [Proyecto] = getHistogram(Proyecto,img)
96         Proyecto.imageHistogram = imhist(img);
97     end
98
99     function [Proyecto] = getEqHist(Proyecto)
100         Proyecto.eqHistogram = imhist(Proyecto.equalizedImage);
101     end
102
103     function [Proyecto] = updateGrayScaleWithMap(Proyecto, map)
104         Proyecto.grayScaleImage = ind2gray(Proyecto.imageMatrix, map);
105     end
106
107     function [Proyecto] = equalizeImage(Proyecto,img)
108         Proyecto.equalizedImage = histeq(img);
109     end
110
```

```

Editor - C:\Users\dgome\OneDrive\Documentos\ProcesamientoDeImagenes\ProyectoPID\Proyecto.m
getImagelInfo.m  Proyecto.m  readImage.m  testingScript.m  +
109         end
110
111     function [Proyecto] = invertImage(Proyecto)
112         Proyecto.invertedImage = imcomplement(Proyecto.grayScaleImage);
113     end
114     function [Proyecto] = photoInvertImage(Proyecto)
115         Proyecto.invertedImage = 255 - (Proyecto.grayScaleImage);
116     end
117     function [Proyecto] = AddToImage(Proyecto,img2,w,h)
118         img2 = imresize(img2,[w h]);
119         Proyecto.addImage = Proyecto.grayScaleImage + img2;
120     end
121     function [Proyecto] = SusToImage(Proyecto,img2,w,h)
122         img2 = imresize(img2,[w h]);
123         Proyecto.susImage = Proyecto.grayScaleImage - img2;
124     end
125     function [Proyecto] = DilateImage(Proyecto,img,src)
126         Proyecto.dilImage = imdilate(img,src);
127     end
128     function [Proyecto] = ErodeImage(Proyecto,img,src)
129         Proyecto.erImage = imerode(img,src);
130     end
131     function [Proyecto] = Update(Proyecto, img)

```

Command Window

```

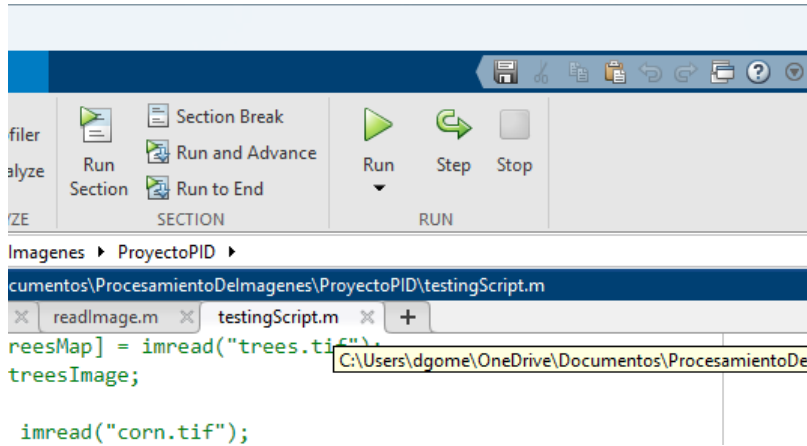
Editor - C:\Users\dgome\OneDrive\Documentos\ProcesamientoDeImagenes\ProyectoPID\Proyecto.m
getImagelInfo.m  Proyecto.m  readImage.m  testingScript.m  +
113         end
114     function [Proyecto] = photoInvertImage(Proyecto)
115         Proyecto.invertedImage = 255 - (Proyecto.grayScaleImage);
116     end
117     function [Proyecto] = AddToImage(Proyecto,img2,w,h)
118         img2 = imresize(img2,[w h]);
119         Proyecto.addImage = Proyecto.grayScaleImage + img2;
120     end
121     function [Proyecto] = SusToImage(Proyecto,img2,w,h)
122         img2 = imresize(img2,[w h]);
123         Proyecto.susImage = Proyecto.grayScaleImage - img2;
124     end
125     function [Proyecto] = DilateImage(Proyecto,img,src)
126         Proyecto.dilImage = imdilate(img,src);
127     end
128     function [Proyecto] = ErodeImage(Proyecto,img,src)
129         Proyecto.erImage = imerode(img,src);
130     end
131     function [Proyecto] = Update(Proyecto, img)
132         Proyecto.grayScaleImage = img;
133     end
134     end
135     end

```

Programa Compilado.

1. Se ejecuta el programa.

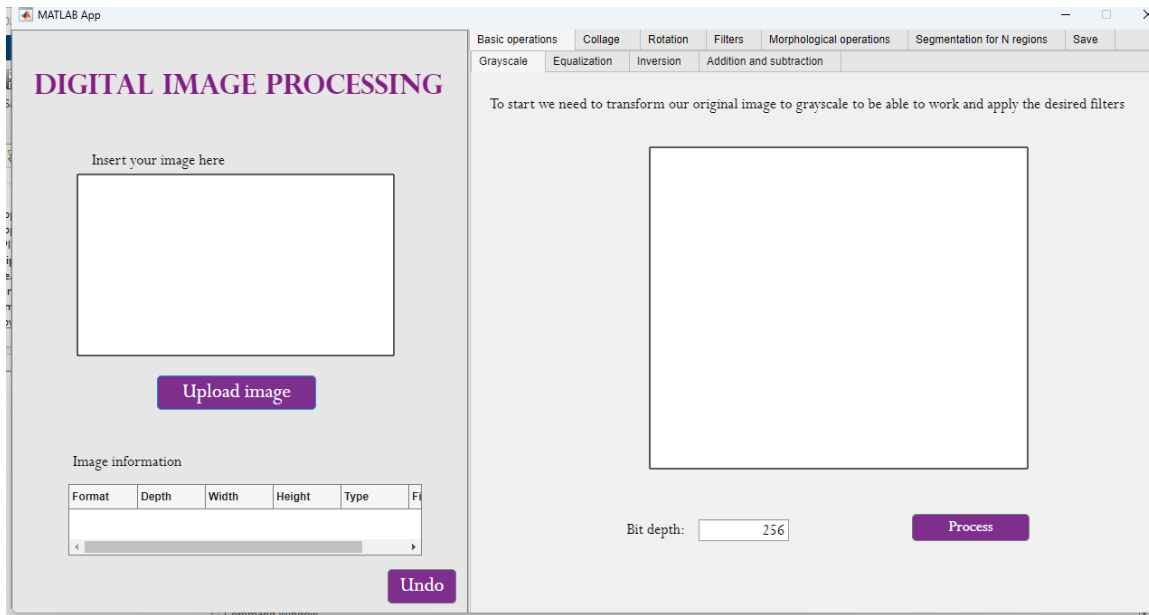
Se acciona el botón de Run para que se inicialice el programa.



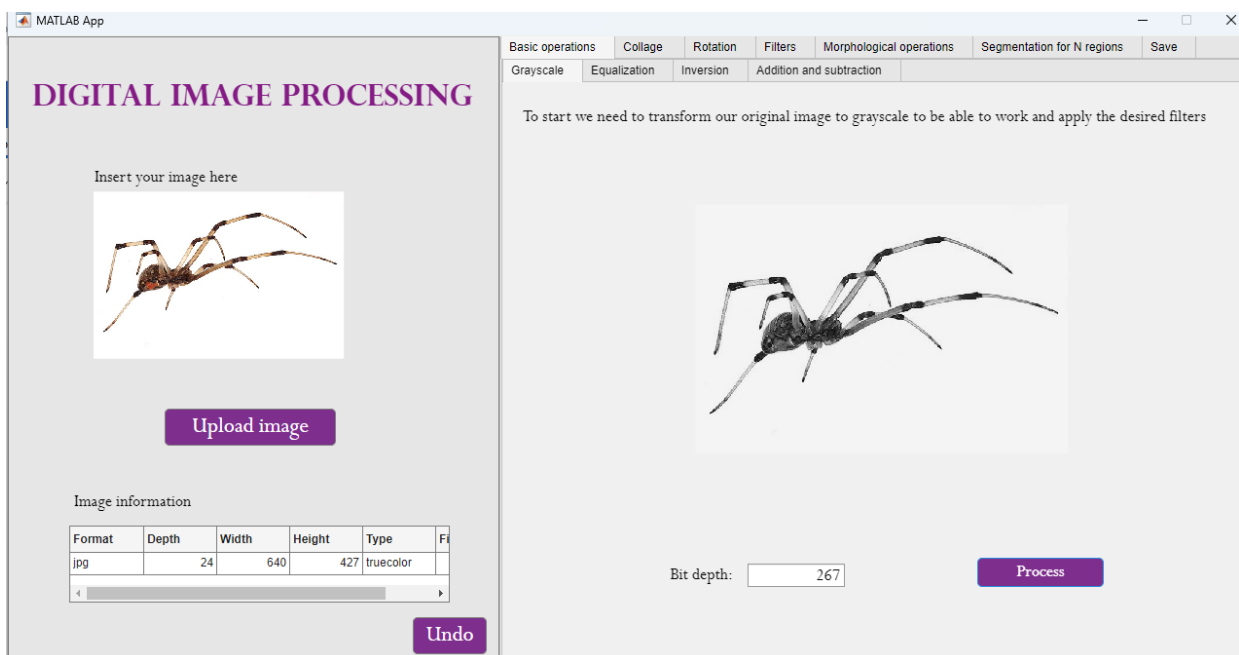
2. Posteriormente se abre la interfaz llamada ProyectoPid.

nombre	fecha de modificación	tipo	tamaño
Exports	18/06/2023 10:02 p. m.	Carpeta de archivos	
Imágenes de prueba	20/06/2023 04:21 p. m.	Carpeta de archivos	
TestinigApp	23/05/2023 09:02 p. m.	Carpeta de archivos	
getImageInfo	31/05/2023 01:17 p. m.	MATLAB Code	1 KB
Proyecto	07/06/2023 10:10 p. m.	MATLAB Code	5 KB
ProyectoPID	18/06/2023 10:00 p. m.	MATLAB App	431 KB
PruebaProyecto.asv	23/05/2023 11:00 p. m.	Archivo ASV	3 KB
readImage	31/05/2023 01:19 p. m.	MATLAB Code	1 KB
testingScript	31/05/2023 01:26 p. m.	MATLAB Code	1 KB
TestinigApp.prj	31/05/2023 01:35 p. m.	Archivo PRJ	3 KB
TestinigApp_1.prj	31/05/2023 01:36 p. m.	Archivo PRJ	6 KB

En breve se abrirá una pestaña que será la ya mencionada interfaz.



3. Una vez abierta la interfaz se podrá interactuar con las diversas imágenes que se tendrá para ejecutar cada función.



En upload image, se coloca la imagen deseada, para que después se realice el proceso como se muestra en la imagen superior.

4. En la parte de arriba habrá una barra de tareas en donde se podrá seleccionar la función que se desee, así como en cada una de estas, se especifica como opera cada proceso en las imágenes seleccionadas.

