
# SIGNAL AND SYTEMS DESIGN LAB

MURAT ORTA 200702046

UĞUR TOPLAR 200702044

ÖMER BAKİ YALÇIN 200702034

MEHMET NEBİH KARAOĞLAN 220702603

Date:24.12.2023

**EEE 3005 Signals and Systems**

# WHAT IS THE IMAGE PROCESSING?

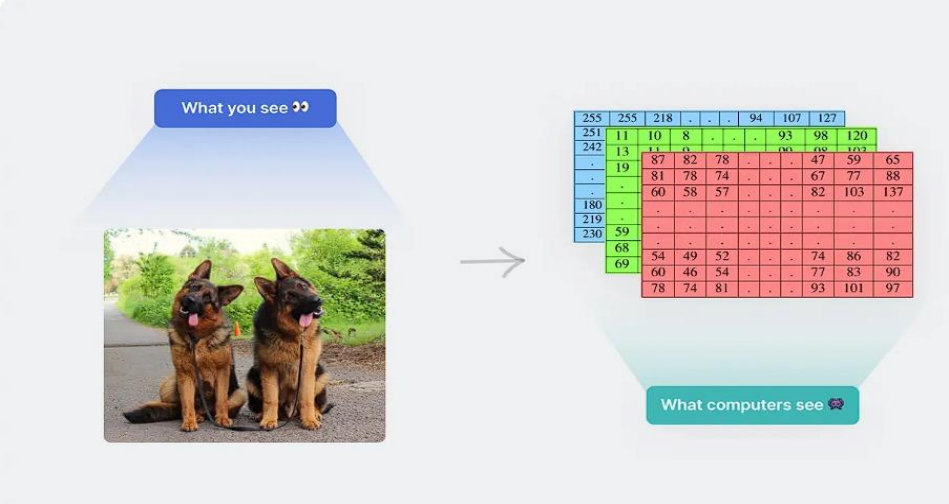



Image processing involves using computer algorithms to perform operations on an image to enhance it, extract information, or manipulate it in some way. It encompasses a wide range of techniques and methods, such as image filtering, segmentation, compression, enhancement, and pattern recognition. The goal is to improve the quality of an image, extract useful information, or perform specific tasks like object detection, image restoration, or recognition.

Image processing is a vast field with various techniques and processes. Here's a detailed breakdown:

**1. Image Acquisition:**
Capturing Images: Using devices like cameras, scanners, or other sensors to acquire images.
Digitization: Converting analog images into digital formats.
**2. Image Enhancement:**
Improving Visual Quality: Adjusting brightness, contrast, color balance, and sharpness.
Noise Reduction: Removing unwanted elements like random variations or artifacts.
**3. Image Restoration:**
Removing Defects: Repairing damaged or degraded images caused by blurring, noise, or compression.
Deblurring: Correcting images affected by motion blur or out-of-focus issues.
**4. Image Compression:**
Reducing File Size: Employing various algorithms to decrease the storage space required for images while preserving quality.

Lossless vs. Lossy Compression: Retaining all image data versus sacrificing some data for smaller file sizes.

**5. Image Segmentation:**

Dividing Images: Partitioning an image into meaningful segments or regions.
Object Extraction: Identifying and separating objects from the background.

**6. Feature Extraction:**

Identifying Patterns: Detecting edges, corners, shapes, or textures within an image.
Descriptor Generation: Creating numerical representations of image features for analysis.

**7. Object Detection and Recognition:**

Locating Objects: Finding and identifying specific objects or patterns within an image.
Classification: Categorizing objects based on predefined criteria.

**8. 3D Image Processing:**

Volume Rendering: Processing and visualizing images in three dimensions for applications in medical imaging, scientific visualization, etc.

**9. Machine Learning and AI in Image Processing:**

Deep Learning: Using neural networks to automate image analysis, classification, and generation tasks.

## Applications

**Medical Imaging:** Diagnosis, analysis, and treatment planning.
Remote Sensing: Satellite imagery analysis for mapping, weather forecasting, etc.
**Security and Surveillance:** Object detection, face recognition, etc.
Entertainment: Image editing, special effects in movies, etc.
**Robotics and Automation:** Vision-based tasks for robots, quality control in manufacturing, etc.

Each of these processes involves specific algorithms, mathematical operations, and tools to manipulate and analyze images to achieve desired outcomes.

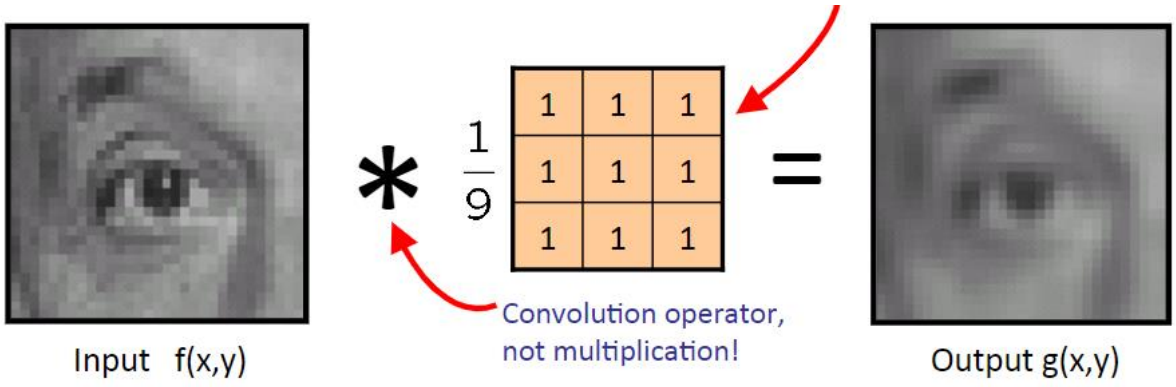
Image processing is the utilization of computer-based algorithms and techniques to perform various operations on digital images. It involves a wide array of methods and techniques used to manipulate images. Here are some fundamental methods used in image processing:

# IMAGE PROCESSING TECHNIQUES

## 1. Filtering:

Convolution and Filters: Applying different filters to emphasize specific features or reduce noise in an image.

Edge Detection: Using edge detection filters to identify sharp transitions or edges in an image.




## 2. Enhancement and Correction:

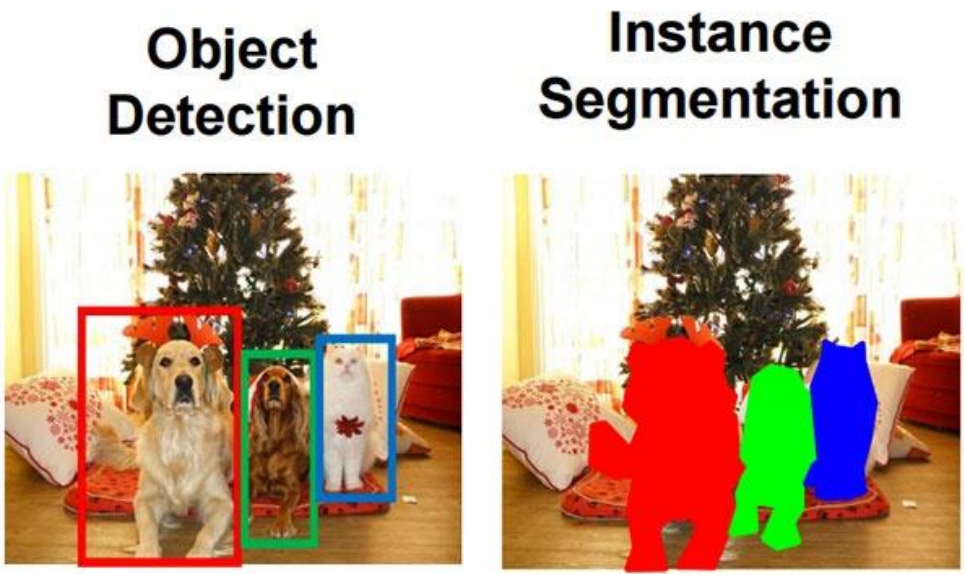
Contrast and Brightness Adjustment: Increasing or decreasing contrast, adjusting brightness in an image.

Correction of Faulty or Corrupted Images: Removing blurriness, noise, or defects.


## 3. Segmentation:

Object Detection: Separating or recognizing different objects or regions within an image.

Partitioning: Dividing the image into different regions to facilitate analysis.



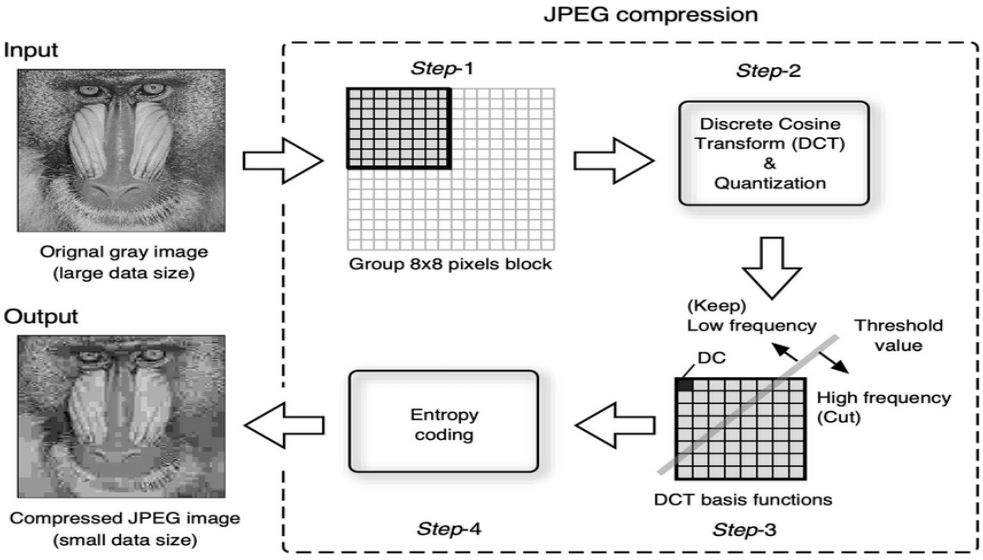

**4. Feature Extraction:**

Edges, Corners, and Patterns: Extracting specific features from the image, such as edges, corners, or certain patterns.

Color and Texture Analysis: Analyzing color intensity, textures, or patterns.

**5. Compression:**

Data Compression: Using compression algorithms to store or transmit images in smaller sizes.




**6. Deep Learning and Artificial Intelligence:**

Deep Learning: Automating complex tasks, such as object recognition, using artificial neural networks and deep learning technique.

## How Convolution Works


Convolution in image processing involves the application of a kernel matrix (also known as a filter or mask) onto an image. Here's an overview of how the convolution process works:

**Kernel Matrix:** Firstly, a kernel matrix is defined. This matrix is typically small in size and determines the filtering or processing function desired. For example, an edge detection filter used for convolution can be represented as a kernel matrix.

**Sliding Operation:** The kernel matrix is slid over the input image with specific strides. This sliding operation involves multiplying the kernel matrix with the pixels of the image at each position and calculating the resulting pixel value.

**Summing Pixel Values:** products obtained from the multiplication are summed up to determine the effect of the kernel matrix at that position on the image. This process continues until the next pixel position is reached.

**Resultant Image:** Once the sliding operation completes, an output image is generated. This image represents the result of using the kernel matrix to emphasize specific features or filter the image.

# Project Title: Image Processing and Interface Design

**Introduction:**
This project focuses on employing image processing techniques using kernel matrices and convolution methods within MATLAB. Additionally, an interface has been designed to facilitate image alterations through this process.

**Methodology:**
The project is centered around fundamental principles of image processing, utilizing convolution operations with specially designed kernel matrices. These matrices are used to apply various filtering techniques to the image. The convolution process in MATLAB involves manipulating pixel values in the image based on a specific kernel matrix to generate a modified image.

## IMAGE PROCESSING TECHNIQUES WITH KERNEL MATRIX

Kernel matrices are fundamental tools used in various image processing techniques to achieve different results. These matrices serve as a crucial component for performing diverse operations in the field of image processing. Here are some types of photo processing that can be achieved using kernel matrices:

This list represents just a few examples; kernel matrices can be applied across a wide range of image processing techniques. The types of operations performed vary depending on the size, values, and sequence of the used matrix, allowing for diverse applications in image processing.

Here are some commonly used kernel matrices in image processing along with their functions:

**Edge Detection:**

Edge detection is used to determine boundaries of objects in an image by enhancing contrasts.

- Sobel Kernel:

$$A = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

- Prewitt Kernel:

$$A = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

**Sharpening:**

Sharpening aims to make edges or details in an image more pronounced.

- Laplacian Kernel:

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

- Kernel Matrix

$$A = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

**Blurring:**

Blurring reduces detail in an image, typically used to reduce noise or create a softer effect.

- Gaussian Filter:

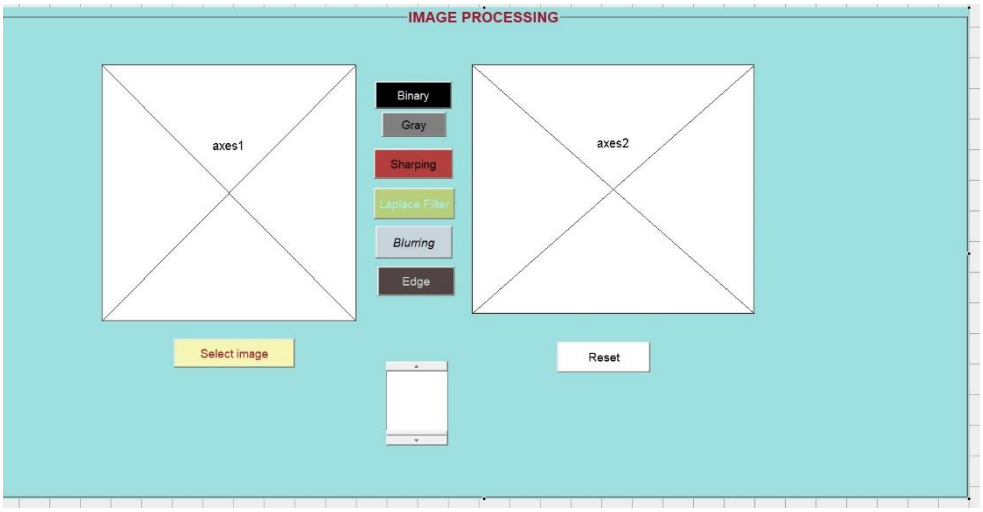
$$A = 1/16 \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

- Mean Filter:

$$A = 1/9 \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

## Interface Design:

As part of the project, an interface was designed to visualize the results obtained from applying image processing techniques. This interface provides users with the ability to apply different filters to the image and visualize the results in a user-friendly manner.




## Writing Cod On Matlab:

In the project, customized codes in MATLAB were utilized to implement image processing techniques. Special kernel matrices were defined for convolution operations, enabling various filtering techniques on the image. Additionally, an interface was designed in MATLAB to visually present the obtained results to the user.

```
function varargout = image_processing_project(varargin)

gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @image_processing_project_OpeningFcn, ...
                   'gui_OutputFcn',  @image_processing_project_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
```

```
function image_processing_project_OpeningFcn(hObject, eventdata, handles, varargin)

handles.output = hObject;

guidata(hObject, handles);


function varargout = image_processing_project_OutputFcn(hObject, eventdata, handles)

varargout{1} = handles.output;

function load_Callback(hObject, eventdata, handles)

global image
[filename pathname] = uigetfile({'*.jpg'},'file selector');
fullpathname = strcat(pathname,filename);
axes(handles.axes1);
image = imread(fullpathname);
imshow(image);

function filter_Callback(hObject, eventdata, handles)% Laplace filter


global image

axes(handles.axes2);
kernel = [0 1 0; 1 -4 1; 0 1 0];

filtered_r = conv2(double(image(:,:,1)), kernel, 'same');
filtered_g = conv2(double(image(:,:,2)), kernel, 'same');
filtered_b = conv2(double(image(:,:,3)), kernel, 'same');

filtered_image = cat(3, uint8(filtered_r), uint8(filtered_g), uint8(filtered_b));
imshow(filtered_image);

function slider1_Callback(hObject, eventdata, handles)


global image
axes(handles.axes2);

val = get(handles.slider1,'Value')+0.5;
gray_image = rgb2gray(image);
brightness = val*gray_image;
imshow(brightness);

function slider1_CreateFcn(hObject, eventdata, handles)

if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end


function reset_Callback(hObject, eventdata, handles)

global image
```

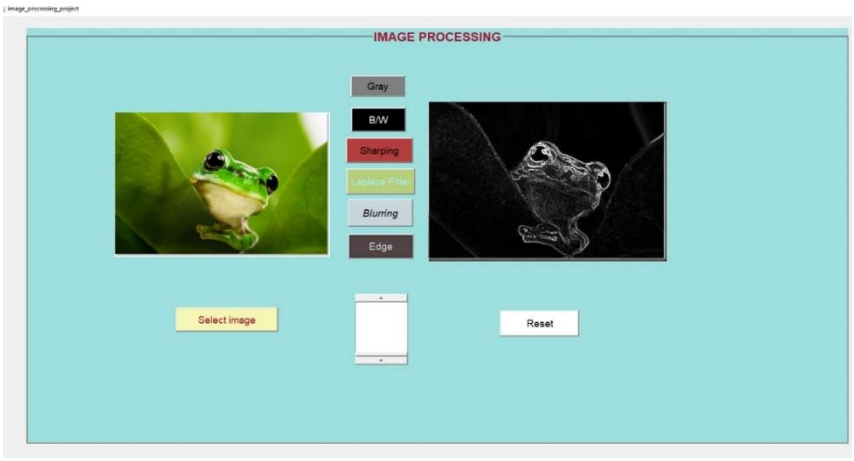
```
axes(handles.axes2);
imshow(image);


function sharp_Callback(hObject, eventdata, handles)

global image


axes(handles.axes2);

sharpeningKernel = [0 -1 0; -1 5 -1 ; 0 -1 0];
%[-1 -1 -1; -1 9 -1; -1 -1 -1];

sharpenedImage = convn(double(image), sharpeningKernel, 'same');



sharpenedImage = uint8(sharpenedImage);
imshow(sharpenedImage);

function Blurring_Callback(hObject, eventdata, handles)

global image


axes(handles.axes2);

sigma = 1.5;
size = 5;
[X, Y] = meshgrid(-size:size, -size:size);
kernel = exp(-(X.^2 + Y.^2) / (2*sigma^2)) /(2*pi*sigma^2);

blurred_image = convn(double(image), kernel, 'same');

blurred_image = uint8(blurred_image);
imshow(blurred_image);


function edge_Callback(hObject, eventdata, handles)


global image


axes(handles.axes2);
Gx = [-1 0 1; -1 0 1; -1 0 1];
Gy = [-1 -1 -1; 0 0 0; 1 1 1];
edge_karel = sqrt(conv2(double(image(:,:,1)), double(Gx), 'same').^2 +
conv2(double(image(:,:,1)), double(Gy), 'same').^2);
edge_karel = uint8(edge_karel);
imshow(edge_karel);


function gray_Callback(hObject, eventdata, handles)

global image
axes(handles.axes2);
[rows, cols, ~] = size(image);
    gray_image = zeros(rows, cols);
kernelMatrix = [0.2989, 0.5870, 0.1
```

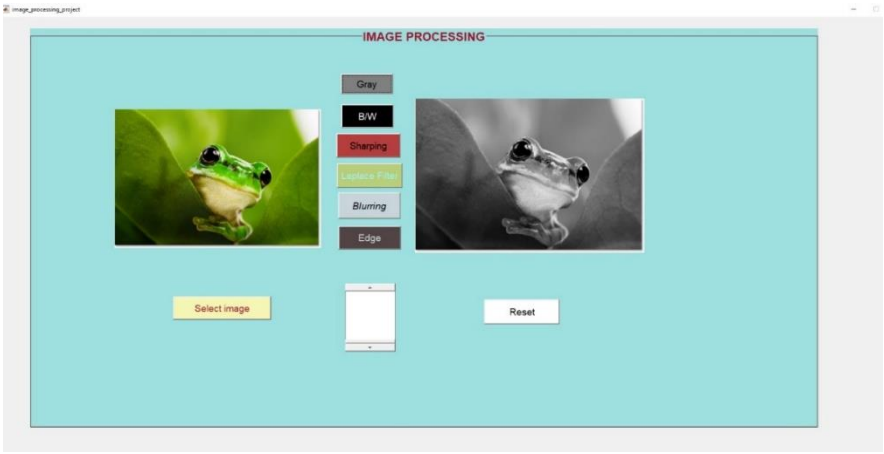
```
    for i = 1:rows
        for j = 1:cols
            gray_pixel_value = 0;
            for k = 1:3
                gray_pixel_value = gray_pixel_value + kernelMatrix(k) * image(i, j, k);
            end
            gray_image(i, j) = gray_pixel_value;
        end
    end


    gray_image = uint8(gray_image);
imshow(gray_image);


function binary_Callback(hObject, eventdata, handles)

global image
axes(handles.axes2);

kernel = [1 1 1; 1 1 1; 1 1 1] / 9;

binary_img = conv2(double(image(:,:,1)), kernel, 'same');
binary_img = uint8(binary_img);

threshold = 128;
binary_imggg = binary_img > threshold;

imshow(binary_imggg);
```

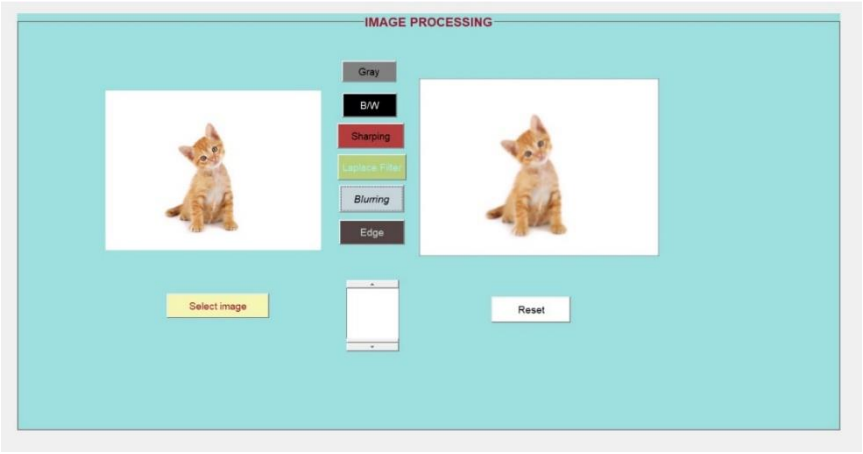
*Published with MATLAB® R2021a*

## Results and Evaluation:

The designed interface effectively visualizes the application of image processing techniques and their results. The outcomes provide valuable insight into understanding how image processing algorithms can be practically applied and visualized.






**Conclusion:**

This project marks a significant step in understanding the use of image processing techniques and their integration into a user interface. The achieved outcomes serve as an important example of how these techniques can be practically applied.