





What is Image Processing?

The class of techniques known as "digital image processing" deals with applying computer algorithms to manipulate digital images. For many applications, including image compression, object detection, and face recognition, it is a necessary preprocessing step.

Image processing is the process of improving an already-existing image or collecting important data from it. This is crucial for a number of Deep Learning-based Computer Vision applications, as these kinds of preprocessing can significantly increase a model's performance. Another use, particularly in the entertainment sector, is manipulation of images, such as the addition or removal of objects

How Computers Understand?

A computer understands digital images as 2D or 3D matrices, where each value or pixel in the matrix represents the amplitude, sometimes referred to as the "intensity," of the pixel. We often work with 8-bit pictures, in which the amplitude value is between 0 and 255. Various "types" of images are handled by computers according to their function representations.

1.Binary images

Binary images are those that have just two possible pixel intensity values: 0 for black and 1 for white. These pictures are usually used to draw attention to a specific area of a colorful picture.

2. Grayscale Image

Pictures in grayscale, often known as 8-bit pictures, are made up of 256 different colors, with black represented by a pixel intensity of 0 and white by a pixel intensity of 255. The various colors of gray represent the remaining 254 values

3. RGB Color Image

RGB color images consist of three 16-bit matrices representing the Red, Green, and Blue channels. Each pixel is defined by three coordinates, with values ranging from 0 to 255. Black is represented as (0, 0, 0), white as (255, 255, 255), and various combinations create different colors, such as red (255, 0, 0), green (0, 255, 0), and blue (0, 0, 255).

4.RGBA images

RGBA photos are RGB images that have been colored, with an additional channel called "alpha" that shows how opacity is applied to the RGB image. Opacity is basically a "see-through" quality, with a value ranging from 0% to 100%.

In physics, opacity describes how much light can flow through an item. For example, wood is opaque, frosted glass is translucent, and cellophane paper is transparent (100 percent opacity). In RGBA photos, the alpha channel attempts to replicate this characteristic.

Image Processing Techniques

Image processing can be applied to improve an image's quality, eliminate unneeded elements from an image, or even start from scratch and produce new images. Image processing, for instance, can be used to eliminate the background from a portrait picture so that the subject is the only thing in the the center.

Image processing is a vast and complex field, with many different algorithms and techniques that can be used to achieve different results. Here are some of the techniques that can be used.

1: Image Enhancement

2: Image Restoration

3: Object Detection

4: Image Compression

5: Image Manipulation

6: Image Generation

7: Image-to-Image Translation

Phases of Image Processing

1.Image Acquisition

An analogue-to-digital converter is used to digitalize the image after it is taken by a camera and, if the camera output is not already digital, so that it can be processed further in a computer.

2. Image Enhancement

In this step, the acquired image is manipulated to meet the requirements of the specific task for which the image will be used. Such techniques are primarily aimed at highlighting the hidden or important details in an image, like contrast and brightness adjustment, etc.

3. Image Restoration

This step, which focuses on improving a picture's appearance, is objective since it may be connected to a mathematical or probabilistic model that explains why an image decreases. For instance, reducing image noise or blur.

4. Color Image Processing

This step aims at handling the processing of colored images (16-bit RGB or RGBA images), for example, peforming color correction or color modeling in images.

5. Wavelets and Multi-Resolution Processing

The fundamental units used to represent images at different resolutions are called wavelets. Images are successively divided into smaller parts for pyramidal representation and data compression.

6. Image Compression

Images are often compressed to transfer or due to storage limits. Online, platforms like Google use thumbnails for efficiency, only showing the full resolution when clicked, saving server bandwidth.

7. Morphological Processing

To represent and describe shape, image components are extracted using mathematical operations. For instance, erosion sharpens while dilation blurs object edges in images.

8. Image Segmentation

In this step image segmentation divides an image into key parts for a clearer analysis. It helps computers focus on vital areas, enhancing automated system performance.

9. Representation and Description

After image segmentation, the next step determines if a segmented area is shown as a boundary or a full region. Description focuses on extracting attributes for relevant quantitative data or distinguishing object classes.

10. Object Detection and Recognition

Once objects are segmented and described, the system labels them, identifying them as "vehicle," "person," etc., for user clarity.

11. Knowledge Base

The bounding box coordinates and object label of an object of interest located in the image could be all that is needed to know something. Anything that can be added to the knowledge base to help in the solution of the particular task at hand can be coded.

**Number\_plate Function :**

**FUNCTİON NAME : WHAT IT DOES**

**rgb2gray:** Converts a color image (RGB) to grayscale, reducing it to a single channel representing intensity.

**graythresh:** Computes a threshold value using Otsu's method, aiming to separate the foreground from the background in a grayscale image.

**im2bw:** Converts a grayscale or color image to a binary image by thresholding.

**bwareaopen:** Removes small objects (or noise) from a binary image based on their area.

**bwlabel:** Labels connected components in a binary image, assigning a unique label to each connected region.

**find:** Returns the linear indices of nonzero elements in an array. Useful for locating specific values or regions in an image.

**corr2:** Computes the 2-D correlation coefficient between two matrices. In this context, it's likely used to measure similarity between template images and portions of the main image.

**cell2mat:** Converts a cell array to an ordinary array. This function is not directly related to image processing but is useful for reshaping or converting data structures.

**picture:** In the provided code, picture is a variable holding the image data after various processing steps. It's not a function but a variable name.

function np=number\_plate(img)

%image correlation method

%matches 2 matrix

load('imgfildata.mat');

[~,cc]=size(img);

picture=imresize(img,[300 500]);

if size(picture,3)==3

picture=rgb2gray(picture);

%grey values are btwn 0 to 1 or 0 to 255

end

threshold = graythresh(picture); %greythresh gives the threshold value of greyscale image

picture =~im2bw(picture,threshold); %black nd white values are 0 or 1 and values greater thn threshold=1,rest=0 and invert white and black ie 1 to 0 and 0 to 1

picture = bwareaopen(picture,30); % those things that have less than 30 pixels are removed

if cc>2000

picture1=bwareaopen(picture,3500); %those things that have less than 3500 pixels are removed ie excluding nmbr plate

else

picture1=bwareaopen(picture,3000); %those things that have less than 3000 pixels are removed ie excluding nmbr plate

end

picture2=picture-picture1; %only number plate is left

picture2=bwareaopen(picture2,200); %only text is there in the nmbr plate

[L,Ne]=bwlabel(picture2);

%l gives matrix which has info of

nmbr plate and

Ne gives number of digits or characters

final\_output=[];

t=[];

for n=1:Ne

[r,c] = find(L==n);

n1=picture(min(r):max(r),min(c):max(c));

%picture command crops nth object from L

n1=imresize(n1,[42,24]);

%in database size is 42,24 so it is

resized so that

we can match it with the database

x=[ ];

totalLetters=size(imgfile,2);

for k=1:totalLetters

y=corr2(imgfile{1,k},n1);

x=[x y];

end

% t=[t max(x)];

if max(x)>.35

z=find(x==max(x));

out=cell2mat(imgfile(2,z));

final\_output=[final\_output out];

end

end

np=final\_output;

end

**FUNCTİON NAME : WHAT IT DOES**

**size:** Returns the dimensions of an array or matrix.

* It can be used to determine the number of rows and

columns in a matrix or the size along a specific dimension.

**imresize:** Resizes an image to a specified dimension or scale.

* It can be used to enlarge or reduce the dimensions of an image. The function takes the image and the desired dimensions or scale factor as inputs.Formun Üstü

**imread:** Reads an image from a file into MATLAB.

* It reads image data from various file formats (e.g., JPG, PNG, BMP) and returns the image as a matrix. The function requires the path to the image file as an input.

**The Main Code:**

clc

close all;

clear;

load imgfildata;

[file,path]=uigetfile({'\*.jpg;\*.bmp;\*.png;\*.tif'},'Choose an image');

s=[path,file];

picture=imread(s);

[~,cc]=size(picture);

picture=imresize(picture,[300 500]);

if size(picture,3)==3

picture=rgb2gray(picture);

end

se=strel('rectangle',[5,5]);

a=imerode(picture,se);

figure,imshow(a);

b=imdilate(a,se);

threshold = graythresh(picture);

picture =~im2bw(picture,threshold);

picture = bwareaopen(picture,30);

imshow(picture)

if cc>2000

picture1=bwareaopen(picture,3500);

else

picture1=bwareaopen(picture,3000);

end

figure,imshow(picture1)

picture2=picture-picture1;

figure,imshow(picture2)

picture2=bwareaopen(picture2,200);

figure,imshow(picture2)

[L,Ne]=bwlabel(picture2);

propied=regionprops(L,'BoundingBox');

hold on

pause(1)

for n=1:size(propied,1)

rectangle('Position',propied(n).BoundingBox,'EdgeColor','g','LineWidth',2)

end

hold off

figure

final\_output=[];

t=[];

for n=1:Ne

[r,c] = find(L==n);

n1=picture(min(r):max(r),min(c):max(c));

n1=imresize(n1,[42,24]);

imshow(n1)

pause(0.2)

x=[ ];

totalLetters=size(imgfile,2);

for k=1:totalLetters

y=corr2(imgfile{1,k},n1);

x=[x y];

end

t=[t max(x)];

if max(x)>.45

z=find(x==max(x));

out=cell2mat(imgfile(2,z));

final\_output=[final\_output out];

end

end

file = fopen('number\_Plate.txt', 'wt');

fprintf(file,'%s\n',final\_output);

fclose(file);

winopen('number\_Plate.txt')

**bwlabel: Labels connected components in a binary image.**

**regionprops: Measures properties of image regions.**

**rectangle: Draws a rectangle on a figure.**

**find: Finds indices of non-zero elements.**

**corr2: Computes the 2-D correlation coefficient.**

**cell2mat: Converts a cell array to an ordinary array.**

**FUNCTİON NAME : WHAT IT DOES**

**uigetfile: Opens a dialog box to select files interactively in MATLAB, making it easier to work with specific data files or images during the execution of a script or function.**

**rgb2gray: Converts an RGB image to grayscale.**

**strel: Creates a structuring element for morphological operations.**

**imerode: Erodes an image using a structuring element.**

**imshow: Displays an image.**

**imdilate: Dilates an image using a structuring element.**

**graythresh: Computes a global image threshold using Otsu's method.**

**im2bw: Converts an image to binary.**

**bwareaopen: Removes small objects from a binary image.**

**SOURCES(KAYNAKÇA) :**

**Codes:**

[**https://uk.mathworks.com/matlabcentral/fileexchange/54456-licence-plate-recognition**](https://uk.mathworks.com/matlabcentral/fileexchange/54456-licence-plate-recognition)

**Function explanations:**

[**https://uk.mathworks.com/help/matlab/ref**](https://uk.mathworks.com/help/matlab/ref)

**Image Processing explanations:**

[**https://www.v7labs.com/blog/image-processing-guide**](https://www.v7labs.com/blog/image-processing-guide)