Image Processing

Index

Introduction

Image processing is the field of study and application that deals with modifying and analyzing digital images using computer algorithms. The goal of image processing is to enhance the visual quality of images, extract useful information, and make images suitable for further analysis or interpretation.

Fundamentals of Image Formation

Image formation is an analog to digital conversion of an image with the help of 2D Sampling and Quantization techniques that is done by the capturing devices like cameras.

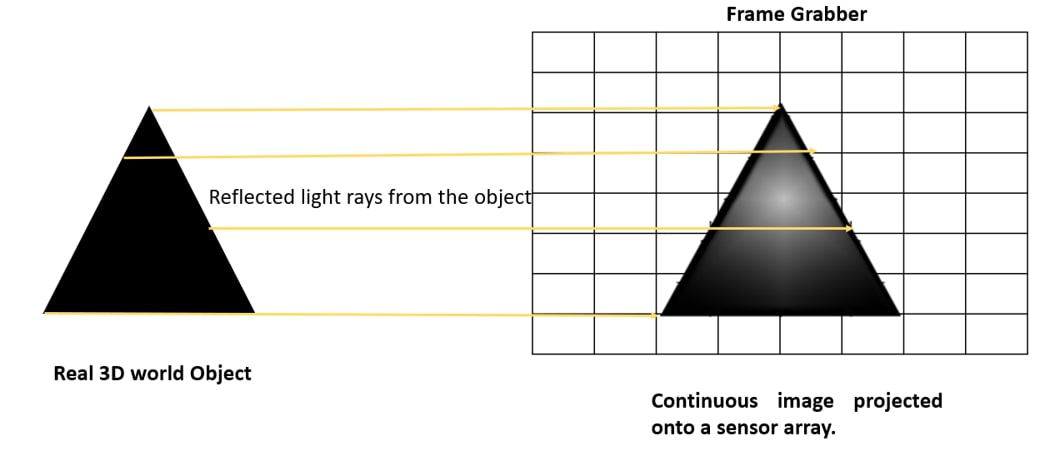
The formation of the analog image took place. It is basically a conversion of the 3D world that is our analog image to a 2D world that is our Digital image.

Generally, a frame grabber or a digitizer is used for sampling and quantizing the analog signals.

Imaging:

The mapping of a 3D world object into a 2D digital image plane is called imaging. In order to do so, each point on the 3D object must correspond to the image plane. We all know that light reflects from every object that we see thus enabling us to capture all those light-reflecting points in our image plane.

Various factors determine the quality of the image like spatial factors or the lens of the capturing device.



Color and Pixelation:

In digital imaging, a frame grabber is placed at the image plane which is like a sensor. It aims to focus the light on it and the continuous image is pixelated via the reflected light by the 3D object. The light that is focused on the sensor generates an electronic signal.

Each pixel that is formed may be colored or grey depending on the intensity of the sampling and quantization of the light that is reflected and the electronic signal that is generated via them.

All these pixels form a digital image. The density of these pixels determines the image quality. The more the density the more the clear and high-resolution image we will get.

Forming a Digital Image:

In order to form or create an image that is digital in nature, we need to have a continuous conversion of data into a digital form. Thus, we require two main steps to do so:

Sampling (2D): Sampling is a spatial resolution of the digital image. And the rate of sampling determines the quality of the digitized image. The magnitude of the sampled image is determined as a value in image processing. It is related to the coordinates values of the image.

Quantization: Quantization is the number of grey levels in the digital image. The transition of the continuous values from the image function to its digital equivalent is called quantization. It is related to the intensity values of the image.

The normal human being acquires a high level of quantization levels to get the fine shading details of the image. The more quantization levels will result in the more clear image.

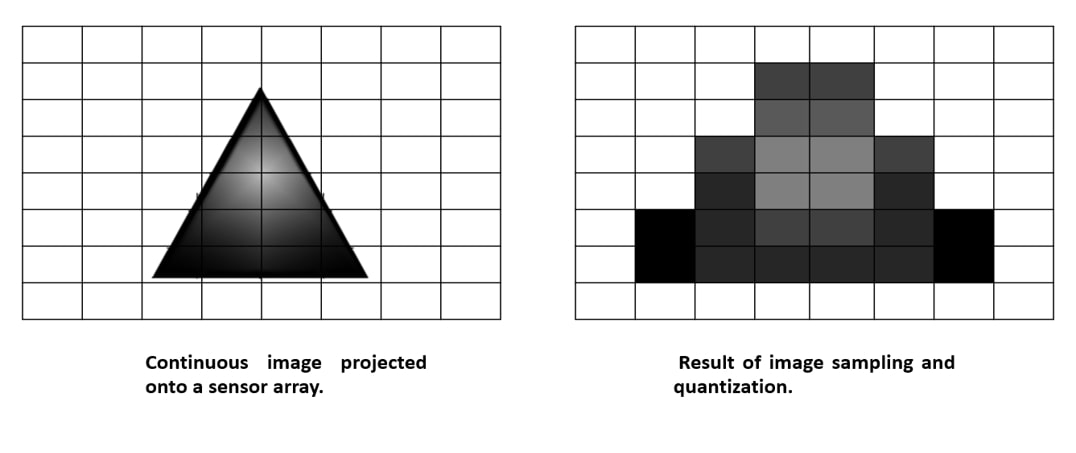


Image Processing Techniques

To achieve the aforementioned goals, several techniques are generally used in image processing.

1. Filtering and Convolution

Filtering operations such as blurring, sharpening and noise reduction are applied to images using convolution. Convolution involves sliding a filter or kernel over an image and performing mathematical operations on each pixel. This process enables various improvements such as anti-aliasing, edge detection and texture removal.

2. Histogram Equalization

Histogram smoothing is a technique used to improve image contrast by redistributing pixel intensity values. Extending the histogram maximizes the dynamic range of the image, improving visual quality and detail visibility. This technique is particularly useful in scenarios where images suffer from poor lighting conditions or limited contrast.

3. Edge Detection

Edge detection algorithms identify image boundaries and important transitions. They emphasize areas where the intensity changes quickly, such as edges, curves or contours. Edge detection plays an important role in object detection, shape analysis and feature extraction. Popular edge detection algorithms are Canny edge detector and Sobel operator.

4. Image Transformation

Image transformation techniques involve manipulating the geometric properties of images to achieve specific goals. Common transformations include rotation, scaling, translation and displacement. These functions enable operations such as image alignment, image registration, and perspective correction. By transforming images, we can adjust their orientation, size and spatial relationships to facilitate later analysis and interpretation.

5. Feature Detection and Extraction

Feature recognition algorithms aim to identify certain visual patterns or structures in images. These features can be corners, edges, patches or textures. Once identified, these features are extracted and described using numerical representations such as histograms of oriented gradients (HOG) or scale transformations (SIFT). Feature detection and extraction are essential for many applications, including object recognition, image fusion, and image-based localization.

6. Image Segmentation

Image segmentation involves dividing an image into several distinct regions based on their visual characteristics. This technique allows you to separate objects from the background or divide the image into significant regions. Common segmentation methods include thresholding, region growing, clustering, and graph-based algorithms. Image segmentation is an essential step in many computer vision tasks, including object detection, image annotation, and semantic understanding.

7. Object Detection and Recognition

Object detection and recognition refers to the recognition and classification of certain patterns of objects or images. This task typically involves training machine learning models using identified datasets to learn the visual properties of target objects. Technologies such as convolutional neural networks (CNN) and deep learning architectures have revolutionized object detection, enabling robust and accurate recognition in various fields, including autonomous driving, surveillance systems and facial recognition.

What is OpenCV?

OpenCV is a Python open-source library for computer vision in artificial intelligence, machine learning, facial recognition, etc.

The term "computer vision" (abbreviated as "CV") in OpenCV refers to a branch of research that enables computers to comprehend the content of digital images like pictures and movies.

To comprehend the content of the images is the goal of computer vision. It takes the description of the images-which may be of an object, a text description, a three-dimensional model, etc.-and extracts it from the images. Computer vision, for instance, can help cars by enabling them to recognize various roadside items, such as pedestrians, traffic signs, and traffic lights, and then respond appropriately.

Two primary tasks are listed below and are defined as follows:

Object Classification: In this process, new objects are classified as belonging to one or more of your training categories by a trained model on a dataset of specific objects.

Object identification: In the object identification phase, our model will pinpoint a specific instance of an object. For instance, it may parse two faces in an image and identify Rohit Sharma and Virat Kohli, respectively.

Installation OpenCV in Windows

|  |
| --- |
| pip install opencv-python |

Reading an Image

OpenCV allows us to perform multiple operations on the image, but to do that it is necessary to read an image file as input, and then we can perform the various operations on it. OpenCV provides following functions which are used to read and write the images.

|  |
| --- |
| import cv2    img = cv2.imread(r'C:\Users\IOCT\Pictures\image.png',1)  #This is using for display the image  cv2.imshow('image',img)  cv2.waitKey(3) # This is necessary to be required so that the image doesn't close immediately.  cv2.destroyAllWindows() |

Saving an Image

OpenCV imwrite() function is used to save an image to a specified file. The file extension defines the image format.

|  |
| --- |
| import cv2  # read image as color  img = cv2.imread(r'C:\Users\IOCT\Pictures\image.png', 1)  # convert to grayscale  gray\_img = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)  # save grayscale image  status = cv2.imwrite(r'C:\Users\IOCT\Pictures\image\_gray.png', gray\_img)  if status:  print("Image saved successfully.")  else:  print("Failed to save image.") |

Extracting Height and Width

|  |
| --- |
| import cv2  img = cv2.imread(r'C:\Users\IOCT\Pictures\image.png', 1)  h, w = img.shape[:2]  print("Height = {}, Width = {}".format(h, w)) |

Extracting the RGB Values of a Pixel

|  |
| --- |
| import cv2  img = cv2.imread(r'C:\Users\IOCT\Pictures\image.png', 1)  (B, G, R) = img[100, 100]  print("R = {}, G = {}, B = {}".format(R, G, B))  B = img[100, 100, 0]  print("B = {}".format(B)) |

Extracting the Region of Interest (ROI)

Sometimes we want to extract a particular part or region of an image. This can be done by slicing the pixels of the image.

|  |
| --- |
| import cv2  img = cv2.imread(r'C:\Users\IOCT\Pictures\image.png', 1)  # by slicing the pixels of the image  roi = img[100 : 500, 200 : 700]  cv2.imshow("ROI", roi)  cv2.waitKey(0) |

Resizing the Image

We can also resize an image in Python using resize() function of the cv2 module and pass the input image and resize pixel value.

|  |
| --- |
| import cv2  img = cv2.imread(r'C:\Users\IOCT\Pictures\image.png', 1)  resize = cv2.resize(img, (500, 500))  cv2.imshow("Resized Image", resize)  cv2.waitKey(0) |

Drawing a Rectangle

We can draw a rectangle on the image using rectangle() method. It takes in 5 arguments:

* Image
* Top-left corner co-ordinates
* Bottom-right corner co-ordinates
* Color (in BGR format)
* Line width

|  |
| --- |
| import cv2  img = cv2.imread(r'C:\Users\IOCT\Pictures\image.png', 1)  output = img.copy()  rectangle = cv2.rectangle(output, (10, 20),(200, 300), (255, 0, 0), 2)  cv2.imshow("Image",rectangle) |

Displaying text

It is also an in-place operation that can be done using the putText() method of OpenCV module. It takes in 7 arguments:

* Image
* Text to be displayed
* Bottom-left corner co-ordinates, from where the text should start
* Font
* Font size
* Color (BGR format)
* Line width

|  |
| --- |
| import cv2  img = cv2.imread(r'C:\Users\IOCT\Pictures\image.png', 1)  output = image.copy()  text = cv2.putText(output, 'OpenCV Demo', (100, 100), cv2.FONT\_HERSHEY\_SIMPLEX, 4, (255, 0, 0), 2)  cv2.imshow(“Image”,text) |