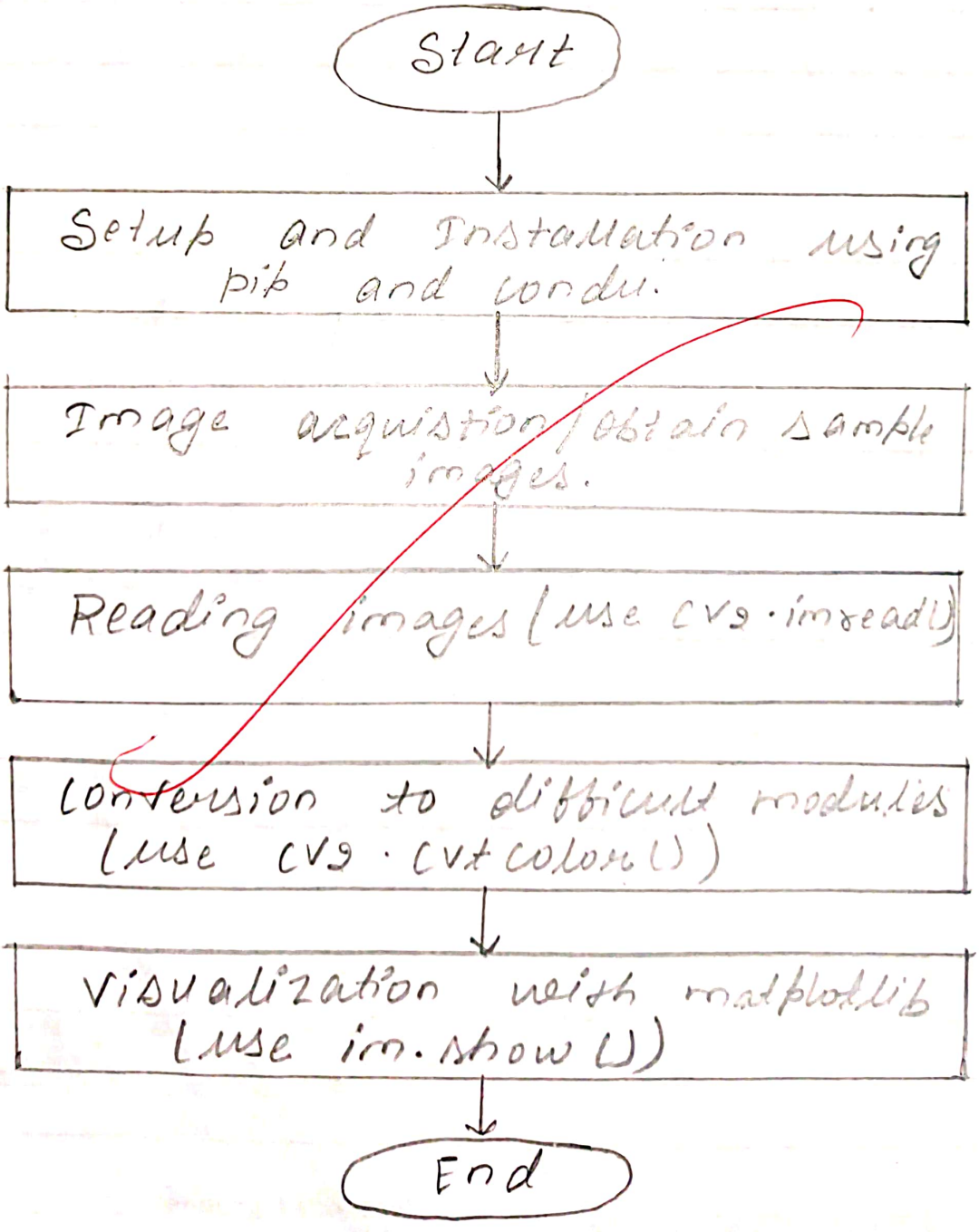
Experiment - 01

Aim : Display images in various formats using different color models.

Software Required : Google colab

Flowchart :



Date 1/8/24 Expt. No. 01

Expt. Name 01 Page No. 02

AIM: Display images in various formats using different color models.

Software Required: Google colab

Description: The experiment involves reading an image file in different formats like - JPG, PNG and displaying it using different color models, like RGB, GreyScale. Python's library like OpenCV and matplotlib can be utilised for image manipulation and visualization.

Color models are essential in digital image processing and Computer Graphics. They provide a framework for representing and manipulating colors in images. This report examines the most commonly used color models including RGB, CMYK and HSV and demonstrates their application in displaying images in different formats.

Teacher's Signature:

Understanding these color models is crucial for tasks, such as image editing, printing and computer vision.

The practical experiment involves selecting a set of sample images and converting them into different color models using image processing libraries. Each image is displayed in the following color models:

i) RGB (Red, Green, Blue):
The standard model for digital displays and computer graphics.

ii) CMYK (Cyan, Magenta, Yellow, Black):
The standard model for color printing.

iii) HSV (Hue, Saturation, Value):
A model that describes colors in terms of their shade, depth and brightness.

Teacher's Signature:_____

Date _____ Expt. No. _____

Expt. Name _____ Page No. 04

Libraries for the experiment :

i) Visualization with matplotlib →
It shows how to visualize the converted images using imshow() function.

ii) Visualization with OpenCV →
It provides instructions on how to save the processed images to the local filesystem using OpenCV's cv2.imwrite() function.

Algorithm :

1.) Start
2) Import cv2.
Import matplotlib.pyplot as plt.
3) Select image path.
4) image_rgb = cv2.cvtColor (image, cv2.COLOR_BGR2RGB)
5) image_hsv = cv2.cvtColor (image_rgb, cv2.COLOR_BGR2HSV)
6) image_grey = cv2.cvtColor.
7) Display image plot.
8) Exit.

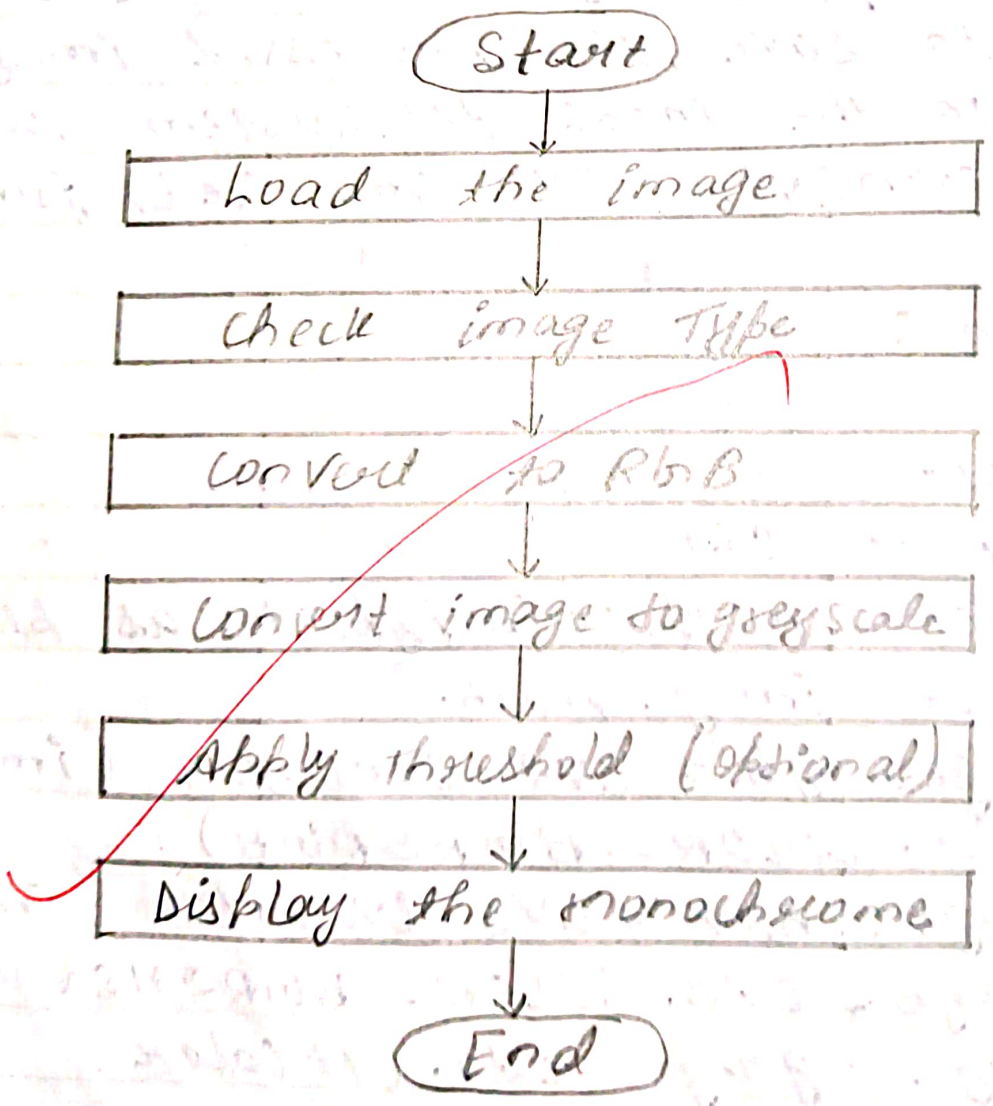
Teacher's Signature: _____

Experiment → 1.2

Aim : Convert color images into monochrome images and demonstrate image color conversion techniques.

Software Required : Google colab

Flowchart :



Date 8/8/24 Expt. No. 02

Expt. Name _____ Page No. 05

Aim : Convert color images into monochrome images and demonstrate image color conversion techniques.

Software Required: Google colab

Description : Converting color image to grey scale is essential in image processing. It aims to understand & apply various techniques for simplifying images, highlighting specific features, and preparing images for further analysis.

→ Color models –

RGB : (Red, Green, Blue) the Standard color model for digital images.

HSV : (Hue, Saturation, Value) A model that separates color information from intensity.

Grayscale : A single-channel representation of image intensity.

→ Image Conversion –

Grayscale Conversion : Transforming a color image into a greyscale image.

Teacher's Signature: _____

Date_____ Expt. No._____

Expt. Name_____ Page No. 06

HSV Conversion : Transforming an image from the RGB color space to the HSV color space.

Formula for conversion of image :

Gray = 0.299 x R + 0.587 x G + 0.114 x B Gray

This formula is derived from the way human perceive color. Human eyes are more sensitive to green light and less sensitive to blue light, which is why the green component has a higher weight in the formula.

Implementation Steps :

1. First, we load the image using the Pillow library.
2. Convert the loaded color image to greyscale using Pillow library. The greyScale image will have intensity values ranging from 0 (black) to 255 (white).
3. Convert the colour image to HSV color model.

Teacher's Signature:_____

Date _____ Expt. No. _____

Expt. Name _____ Page No. 07

4.) Convert color image to CMYK.
5.) Using matplotlib, display the original RGB image alongside with converted greyscale, HSV, and CMYK images.
6.) Display the images.

Algorithm:

1.) Start
2.) Load the color image
3.) Convert the color image to greyscale using formula.
4.) Save or display the greyscale image.
5.) End.

Conclusion:

This experiment demonstrates the process of converting color images into monochrome images & highlights the differences between various color models (RGB, HSV, CMYK). By visualizing these conversions, one can understand the impact of each model on processing of images & graphics design.

Teacher's Signature: _____

Date _____ Expt. No. _____

Expt. Name _____ Page No. 08

Learning outcomes :-

→ Learnt how images are represented in different color spaces (RGB vs Grayscale).

→ Gain practical experience using image processing libraries such as OpenCV.

→ Understand the mathematical formula and function used for color-to-greyscale conversion.

Experiment → 1.3

Aim : Apply image enhancement techniques using grey level transformations.

Software Requirement : Google colab

Flowchart :

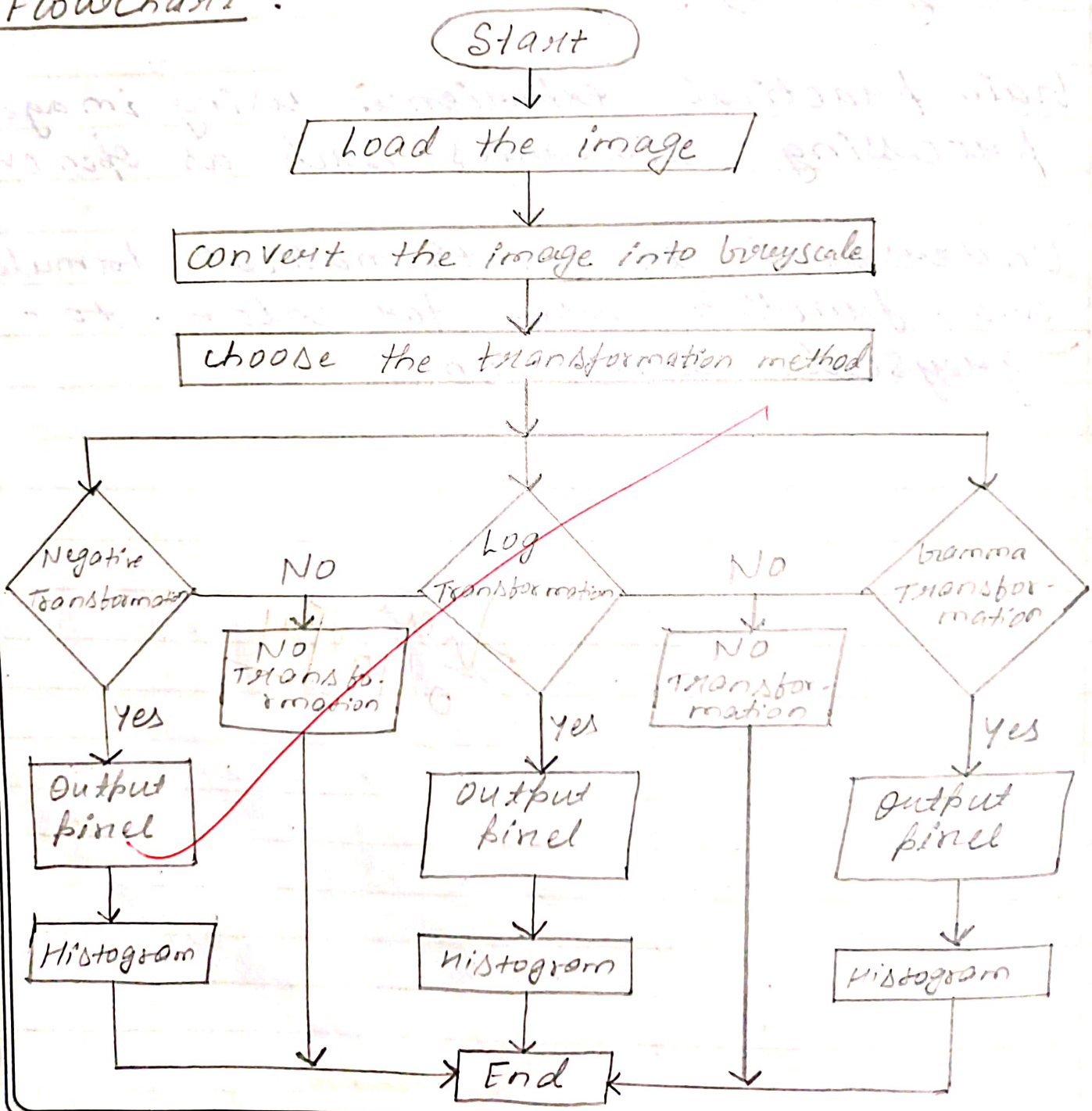


fig.1

Date 22/8/24 Expt. No. 1.3

Expt. Name _____ Page No. 09

Aim: Apply image enhancement techniques using grey level transformation.

Software Required: Google Colab

Description: A greyscale image is a type of image where each pixel represents a shade of gray, typically ranging from black (0 intensity) to white (maximum intensity). Grayscale images only carry intensity information, not color, and are represented by a single channel.

Common types of grey level Transformations:

1) Linear Transformation:

In image processing, it involves a direct, proportional mapping of pixel intensity values.

It's most common type is –

Negative Transformation:

This inverts the intensity levels of an image, making bright areas dark and dark areas bright. It is represented as:

Teacher's Signature: _____

$S = L - 1 - r$

where L is the number of intensity levels in the image (e.g., 256 for an 8-bit image).

2) Logarithmic Transformation :

It is used to compress the dynamic range of an image. They are particularly used for enhancing details in the darker regions of an image without saturating the brighter regions.

formula :- $S = C \cdot \log(1 + r)$

where; $S \rightarrow$ output pixel value
$r \rightarrow$ Input pixel value
$C \rightarrow$ Scaling constant, often choose to normalize the output.

The transformation enhances lower intensity values more than higher ones, making it effective for images with large variations in intensity.

3) Power-law (Gamma) Transformations :

It is also known as gamma corrections, are used to correct the brightness and contrast of an image.

The transformation is defined by the eqn :-

Teacher's Signature:_____

Date_____ Expt. No._____

Expt. Name_____ Page No. 11

$$S = C \cdot r^{\gamma}$$

where:

S → output pixel value
r → Input pixel value
c → constant
γ → controls the shape of the transformation curve.

Implementation Steps / Algorithm:

Step i) Start
Step ii) Load the greyscale image
Step iii) Choose the transformation method
Step iv) Apply the chosen transformation:
a) Negative Transformation:
output_pixel = 265 - input_pixel
b) Log Transformation:
output_pixel = c * log (1 + input_pixel)
c) Gamma Transformation:
output_pixel = c * (input_pixel ^ gamma)
d) Histogram Equalization:
Use histogram equalization function.
Step v) Save or display the enhanced image.
Step vi) End.

Teacher's Signature:_____

Date _____ Expt. No. _____

Expt. Name _____ Page No. 12

Learning outcome:

i) Understanding Image Enhancement.

ii) Learn different techniques to enhance image quality using grey level transformations.

iii) Learn the importance of image preprocessing in computer vision and image analysis tasks.

iv) Learnt to enhance the appearance of images for better visualization.

v) Understanding essential preprocessing step in computer vision and image processing tasks to ensure algorithms work more effectively.

good

22/08/24

Teacher's Signature: _____

Experiment 1.4

Aim: Perform histogram matching and Specifications on images.

Software Required: Google Colab

Flowchart:

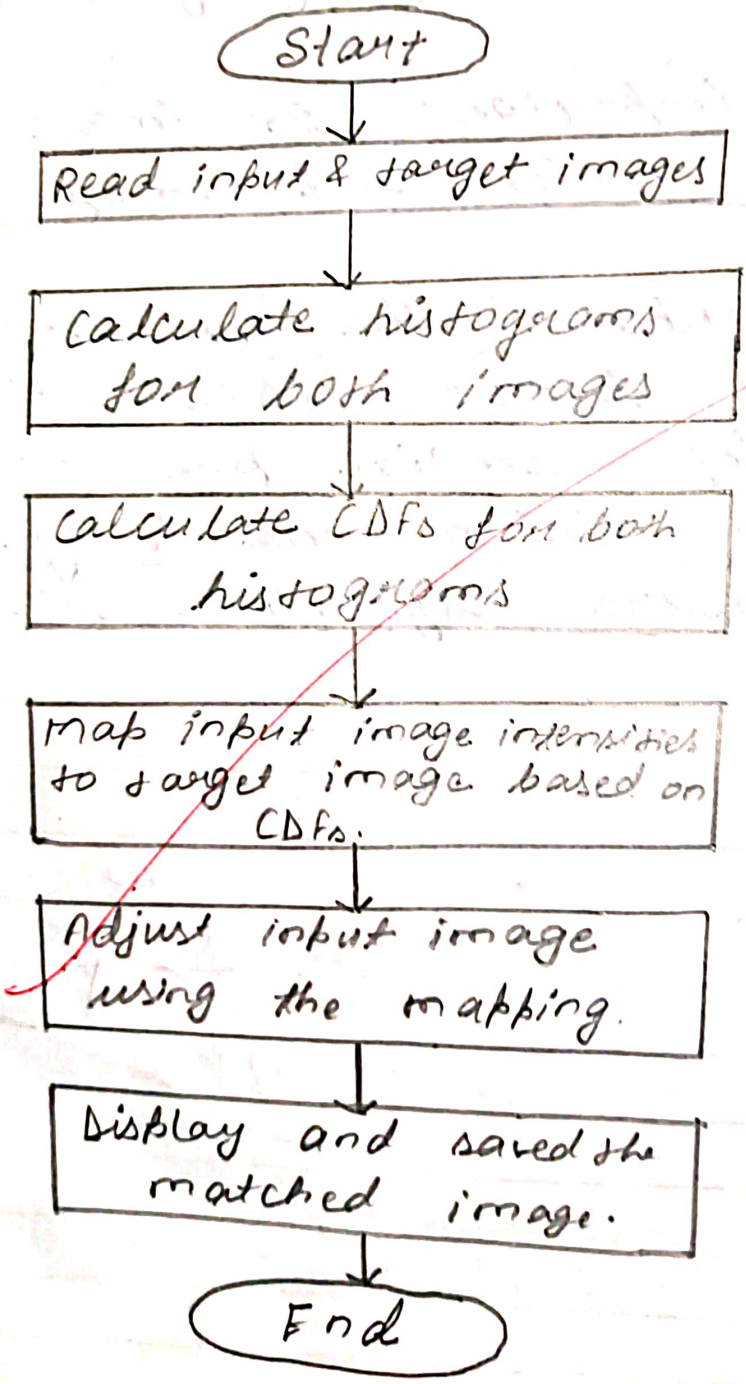


Fig 1: flowchart of histogram matching.

Date _____ Expt. No. 1-4

Expt. Name _____ Page No. 13

Aim : Perform histogram matching and Specifications on images.

Software Required : Google colab

Description :

Histogram matching involves transforming the histogram of one image to match the histogram of another image. This technique is used to modify the contrast of an image by mapping a desired virtual appearance or facilitating better comparison between images.

* Histogram → It is the image represents the frequency distribution of its intensity values. It provides a graphical representation of the number of pixels at each intensity level.

* Cumulative Distribution Function (CDF) → The CDF of a histogram is a cumulative sum of the histogram values, normalized to the range [0, 1]. It represents the

Teacher's Signature: _____

Date _____ Expt. No. _____

Expt. Name _____ Page No. 14

probability distribution of the intensity values in the image.

* mapping function → To achieve histogram matching, a mapping function is created. The function transforms the intensity values of the source image so that its histogram matches the histogram of the reference image.

Algorithm :

Step 1 → Start
Step 2 → Load the source and reference images.
Step 3 → Compute the histogram of both images.
Step 4 → Compute the Cumulative Distribution functions (CDF)
Step 5 → map input intensity levels to Target levels.
Step 6 → Create a Transformation function.
Step 7 → Apply the Transformation function.

Teacher's Signature: _____

Date _____ Expt. No. _____

Expt. Name _____ Page No. 15

Step 8 → Create and Apply mapping function.

Step 9 → Save or display the resulting image.

Step 10 → End.

Learning outcome:

1) Learn what histogram represent in the context of digital images.
2) Understand the concept of histogram matching and its purpose in image processing.
3) Learn how to compute and interpret cumulative distribution functions (CDFs) from image histogram.

30/8/24

Experiment - 9.1

Aim: Enhance images using spatial and frequency domain filters.

Software Requirement: Google colab

Flowchart:

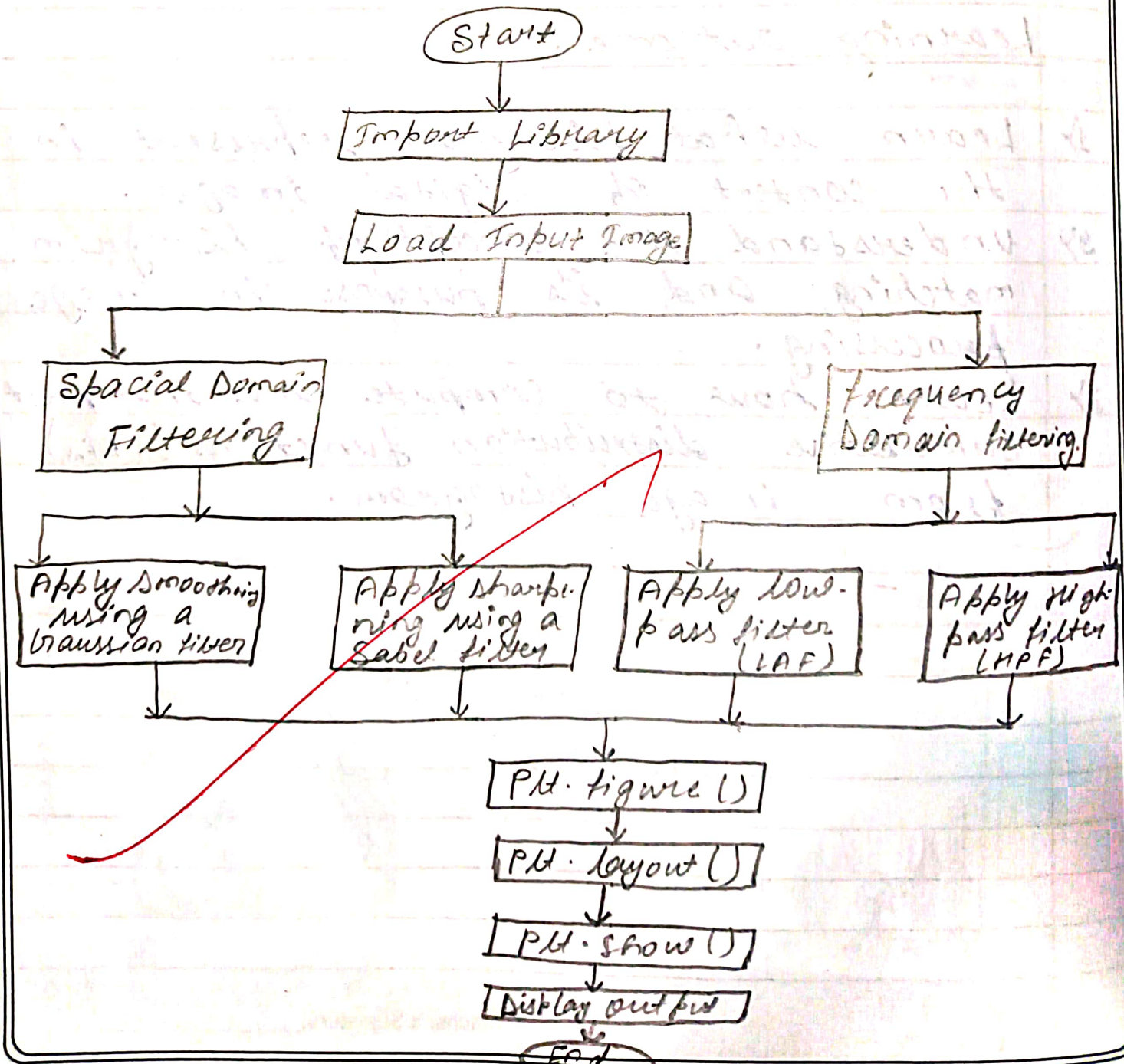


fig.1 : flowchart of filters

Date 20/9/24 Expt. No. 2.1

Expt. Name _____ Page No. 16

Aim : Enhance images using Spatial and frequency domain filters.

Software Requirement : Python IDE (Google colab)

Description : Enhancing images using Spatial and frequency domain filters involves applying techniques to improve image quality. Image enhancement is the process of improving the visual appearance of an image or making it more suitable for analysis by human eyes or machine perception. the two main techniques used for enhancement are :

- Spatial Domain filtering
- frequency Domain filtering

Spatial domain filtering manipulates an image by applying operations directly to the pixel values.

Types of Spatial filters :

1) Smoothing (Low-Pass) filters :- Used to reduce noise or blur details.
   - mean filter
   - Gaussian filter

Teacher's Signature: _____

Date _____ Expt. No. _____

Expt. Name _____ Page No. 17

11.) Sharpening (High-pass) filters: Enhance edges and fine details in the image.
- Laplacian Filter → Highlights regions of rapid intensity change (edges).
- Sobel filter → Detect-edges by calculating gradients in the image intensity.

Frequency domain filtering operates on an image after transforming it from the spatial domain into the frequency domain using fourier Transform.
The fourier transform decomposes an image into sinusoidal components, making it easier to manipulate frequencies.
i) Low-Pass filters → removes high frequency components (e.g., noise, edges) and are useful for smoothing.
ii) High-Pass filters → removes low frequency components (e.g, large structures) and enhance edges and fine details.

Algorithm:
Step.01: Load the image (Grayscale for Simplicity)

Teacher's Signature: _____

Date _____ Expt. No. _____

Expt. Name _____ Page No. 18

Step 02: Apply Spatial Domain filtering
- Smoothing using a Gaussian filter (Low-Pass)
- Sharpening using a Laplacian filter (High-pass).

Step 03: Apply frequency Domain filtering
- Fourier Transform.

Step 04: Create a mask with a square window for low frequencies.

Step 05: Apply the mask and inverse DFT.

Step 06: Display all results.

Learning outcome:

1) Understand the difference between Spatial domain filtering and frequency domain filtering.
2) Gain practical knowledge of implementing Smoothing and sharpening filters.

20/09/24

Teacher's Signature: _____

Experiment → 2.2

Aim: Remove noise from image and apply inverse filtering.

Software Required: Google colab

Flowchart:

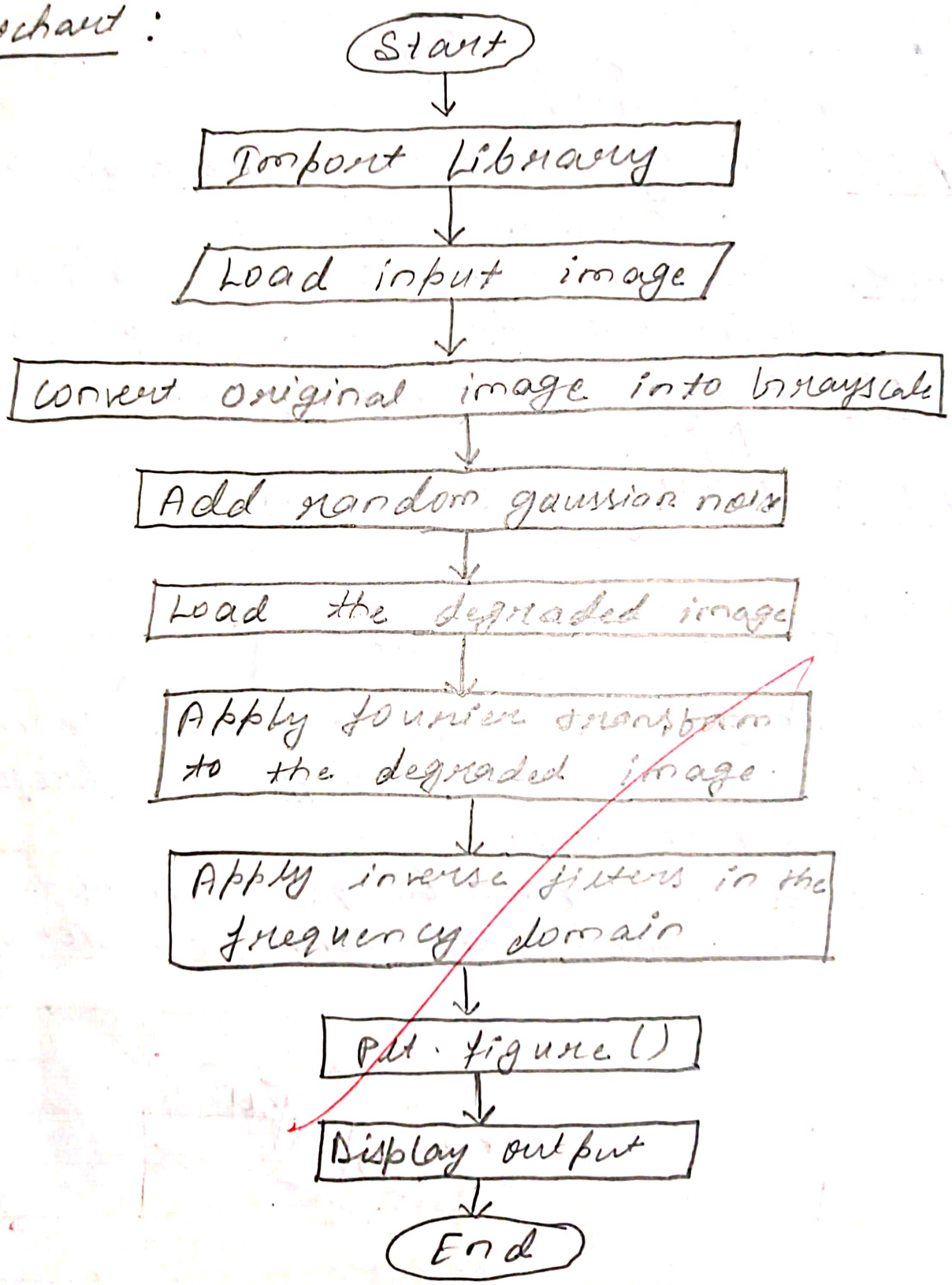


Fig. 1: Flowchart of inverse filtering

Date 27/9/24 Expt. No. 2.2

Expt. Name _____ Page No. 19

Aim :- Remove noise from images and apply inverse filtering.

Software Required : Any Python IDE, Google Colab

Description: Image noise removal involves the application of filters or techniques to reduce or eliminate noise in images. Common types of noise include Gaussian noise, Salt-and-pepper noise, and Speckle noise. Inverse filtering, on the other hand, aims to recover the original image from a degraded version caused by blurring or other linear distortions. This experiment focuses on implementing techniques for both image noise removal and inverse filtering.

Image noise removal is a crucial step in image processing that aims to enhance the quality of images by reducing or unwanted artifacts, caused by noise. Noise can occur due to various factors such as electronic

Teacher's Signature: _____

interference, sensor limitations, or transmission errors, common types of noise include gaussian noise, which is characterized by a bell-shaped distribution of intensity values; Salt-and-pepper noise, which manifests as randomly occurring which appears as grainy textures in the image.

Inverse filtering, on the other hand, is employed to recover the original image from a degraded version caused by blurring or other linear distortions. It is particularly useful in image restoration tasks.

Algorithm:

Step 01: Load the input image.
Step 02: Apply noise removal filter.
Step 03: Save or display the denoised.
Step 04: Load the degraded image
Step 05: Apply fourier transform to

Teacher's Signature:_____

Date _____ Expt. No. _____

Expt. Name _____ Page No. 21

the degraded image.

Step 06: Apply inverse filter in the frequency domain.

Step 07: Perform inverse fourier transform to obtain the restored image.

Step 08: Display restored image.

Learning outcome:

i) Learn various noise removal filters.

ii) Learn how to apply inverse filtering techniques.

iii) Understand the principles of image restoration.

Experiment → 2.3

Aim: Conduct image morphological Operation.

Software Required: Google colab.

Flowchart:

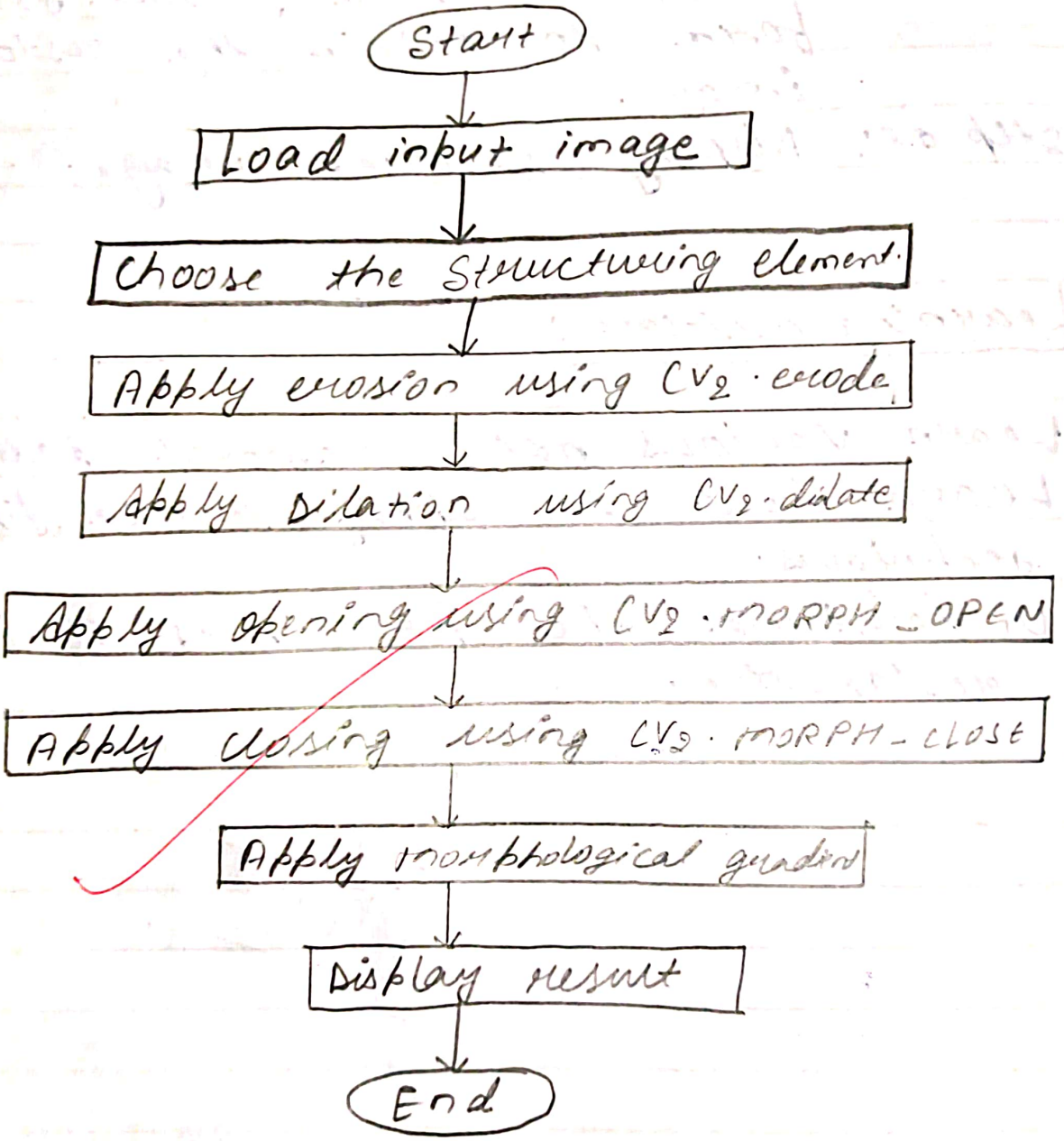


Fig. 1: Flowchart to show morphological Operation.

Date 3/10/24 Expt. No. 2.3

Expt. Name _____ Page No. 22

Aim: Conduct image morphological operation.

Software Required: Google colab

Description: Image morphological operations are a set of Non-Linear Image processing techniques used to modify or enhance the property used to modify or enhance the property used enhance the image. these operations are based on mathematical concepts like set theory and Geometry. These are 3 primary morphological operations:
Erosion, Dilation and opening/closing. Combines these two operations in a specific results. morphological operations can be utilized for noise reduction, object segmentation, shape manipulation and image filtering.
most basic morphological operations are:

Teacher's Signature: _____

Date _____ Expt. No. _____

Expt. Name _____ Page No. 93

i) Dilation : Adds pixels to the boundary of an object, making objects more visible and filtering in small holes.

ii) Erosion : Removes pixels from the boundary of an object; removing islands and small objects.

iii) Opening : Erosion followed by dilation, which removes small-scale details from the circle's border.

iv) closing : Dilation followed by erosion, which fills small holes in the circle's body.

Algorithm :

Step 01 : Start

Step 02 : Load the input image

Step 03 : Choose the structure element (kernel).

Teacher's Signature: _____

Date _____ Expt. No. _____

Expt. Name _____ Page No. 94

Step 04: Apply the chosen morphological operation (erosion, dilation, opening and closing).

Step 05: Save or display the resulting image.

Step 06: End.

Learning outcome:

i) Learnt how to implement and apply morphological operations using python and Open CV.

ii) Develop skills in selecting appropriate morphological operations.

iii) Identify real-world applications of this operation.

Experiment → 3.1

Aim: Detect points, lines, edges & boundaries in images.

Software Required: Google colab.

Flowchart:

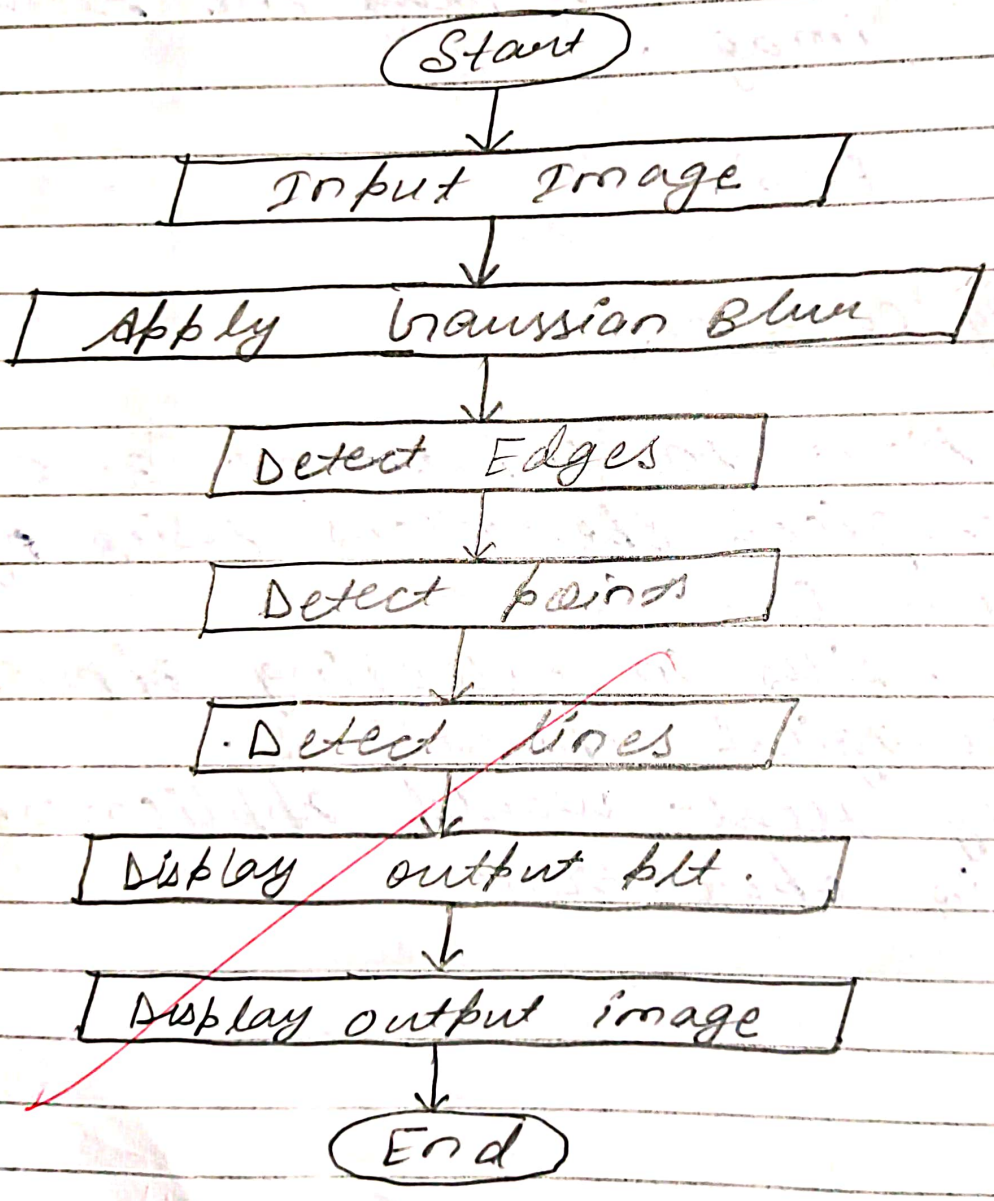


fig. 1 : flowchart to display process.

Date 10/10/24 Expt. No. 3.1

Expt. Name _____ Page No. 25

Aim: Detect points, lines, edges and boundaries in images.

Software Required: Google Colab.

Description: Point, line, edge and boundary detection are fundamental techniques in image processing used to identify significant changes in intensity that correspond to points, lines, edges or boundaries within an image.

i) **Point Detection**: It aims to identify individual pixels in an image that stand out due to a significant change in intensity compared to their neighbours. This technique is often used for identify features like corners or points of interest.

ii) **Line Detection**: Line detection identifies lines within an image by detecting

Teacher's Signature: _____

Date _____ Expt. No. _____

Expt. Name _____ Page No. 26

linear patterns of intensity changes. It is commonly used in applications like road detection in autonomous driving or text line detection in document analysis.

iii) **Edge Detection**: Edge detection identifies significant changes in intensity, making the boundaries between different regions. Edges are critical in defining object boundaries within an image.

Algorithm:

Step 01: Start

Step 02: Load the image

Step 03: convert the image to grayscale.

Step 04: Apply the Gaussian blur to reduce noise.

Teacher's Signature: _____

Date _____ Expt. No. _____

Expt. Name _____ Page No. 27

Step 05: Compute Harris response for each pixel.

Step 06: Apply the Harris edge detection algorithm.

Step 07: Display the edges.

Step 08: End.

Learning outcomes:

i) Identify the importance of points, lines, edges and boundaries in image analysis and computer vision.

ii) Apply specific algorithms (e.g., Canny edge detection) to identify edges.

iii) Use image processing libraries.

Experiment - 3.2

Aim: Implement Boundary linking, representation and description techniques on images.

Software Required: Google colab.

Flowchart:

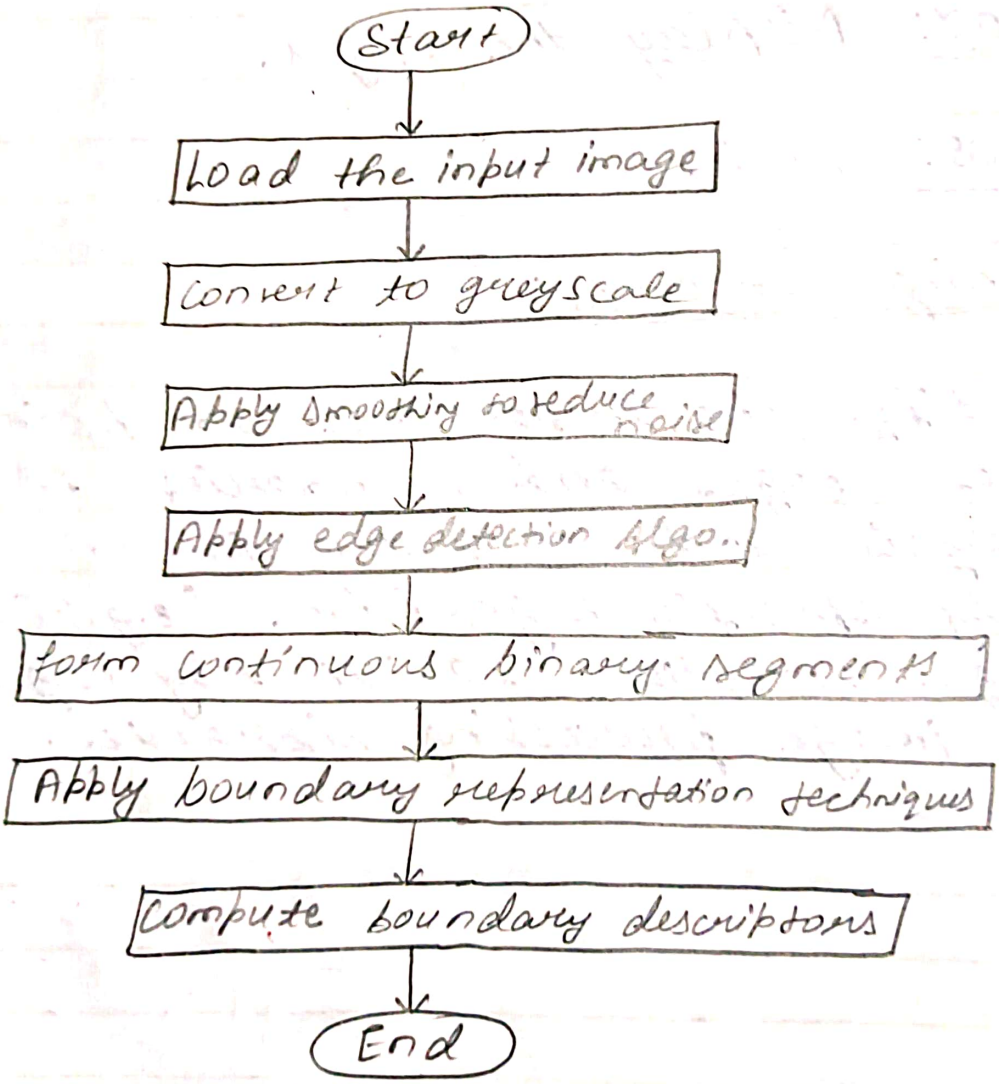


fig. 3.2.1 : flowchart to display techniques on images.

Date 17/10/24 Expt. No. 3.2

Expt. Name Page No. 28

Aim: Implement Boundary linking, representation and description techniques on images.

Software Required: Google Colab

Description:

→ Boundary Linking → Boundary linking connects discrete edge pixels to form continuous boundaries, facilitating the identification and analysis of object shapes in an image.

→ Boundary Representation → It transforms linked boundaries into a compact from for efficient storage and analysis. Chain codes represent the boundary as a sequence of directional moves.

→ Boundary Description → It extracts features from the represented boundaries to facilitate tasks such as shape analysis, object recognition and classification.

Teacher's Signature:

Date _____ Expt. No. _____

Expt. Name _____ Page No. 29

Algorithm :

Step 01: Start.
Step 02: Initialize an empty list to store boundaries.
Step 03: Create a visited array to mark visited pixels.
Step 04: Define 8-connected neighborhood.
Step 05: Iterate through each pixel in the edge image.
Step 06: Display image.
Step 07: Stop.

Learning Outcomes :

i) Implement algorithms to link edge pixels.
ii) Implement and compare various boundary representation.
iii) Learnt different methods of boundary representation.

Teacher's Signature: _____

Experiment - 3.3

Aim: Develop an application of Computer vision using a convolutional neural network.

Software Required: Google Colab.

Flowchart:

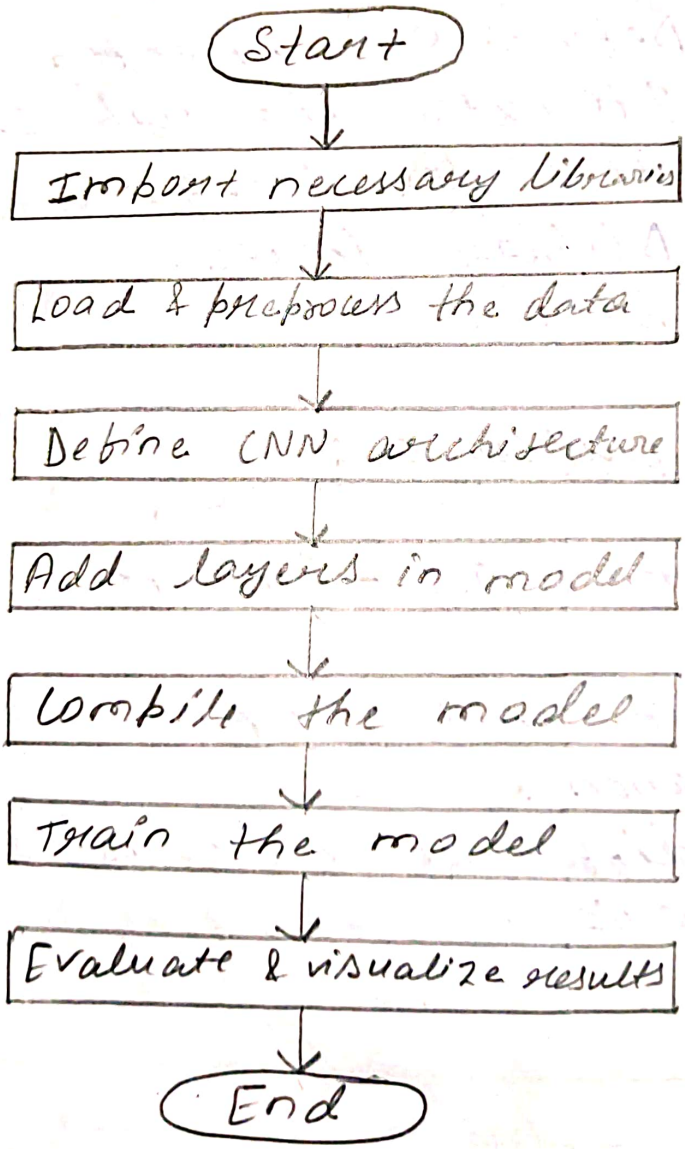


fig. 3.3.1 : Flowchart to classify images

Date _____ Expt. No. 3.3

Expt. Name _____ Page No. 30

Aim: Develop an application of computer vision using a convolutional neural network.

Software Required: Google colab

Description: The Experiment focuses on the development of a computer vision application using a convolutional Neural Network (CNN). The goal is to classify images from a dataset, such as identifying different objects, animals, or digits. CNNs are ideal for this task due to their ability to capture spatial hierarchies and local patterns from images through convolutional filters. This experiment includes training the CNN, validating its performance, and testing the model on unseen data to evaluate its accuracy.

For this experiment, we'll use CIFAR-10 dataset, a standard dataset consisting of 60,000 32 x 32 color images

Teacher's Signature: _____

Date _____ Expt. No. _____

Expt. Name _____ Page No. 31

in 10 classes, with 6,000 images per class. The classes include airplanes, cars, birds, cats, and others.
It's application can be used in various real-world scenarios, such as:

- Object Detection
- Autonomous vehicles
- medical imaging
- Retail

The experiment shows how to train a CNN model to classify images, a core component of many AI-driven computer vision applications.

Algorithm :

Step 01: Load the CIFAR-10 dataset.

Step 02: Define a CNN model using layers like Conv2D, maxpooling 2D.

Step 03: Compile the model.

Teacher's Signature: _____

Date _____ Expt. No. _____

Expt. Name _____ Page No. 32

Step 04: Train the model on training set.

Step 05: Evaluate the model on test set.

Step 06: Use techniques like data augmentation to improve model generalization.

Step 07: Analyse the Result.

Learning outcomes:

i) Gain knowledge of how convolutional, pooling work in image classification tasks.

ii) Learnt techniques for normalizing and preparing datasets for training models.

iii) Understand real-world applications for CNN.