Image Collage Maker

Course No: CSE 4128

Course Title: Image Processing and Computer Vision Laboratory

Department of Computer Science and Engineering

Presented To:

Dr. Sk. Md. Masudul Ahsan

Professor

Department of CSE, KUET

Dipannita Biswas

Lecturer

Department of CSE, KUET

Presented by

Lamisa Bintee Mizan Deya

Roll:1907049

Group: B1

Year: 4th

Semester: 1st

Department of CSE, KUET

Outline

- ✓ Objectives
- ✓ Introduction
- ✓ Demonstration
- ✓ Upload image
- ✓ Select & Drag Image
- ✓ Rotate Image
- ✓ Crop Image
- ✓ Background
- ✓ Filters
- ✓ Background Remove
- ✓ Paint & Erase
- ✓ Add Text
- ✓ Save Collage
- ✓ Conclusion

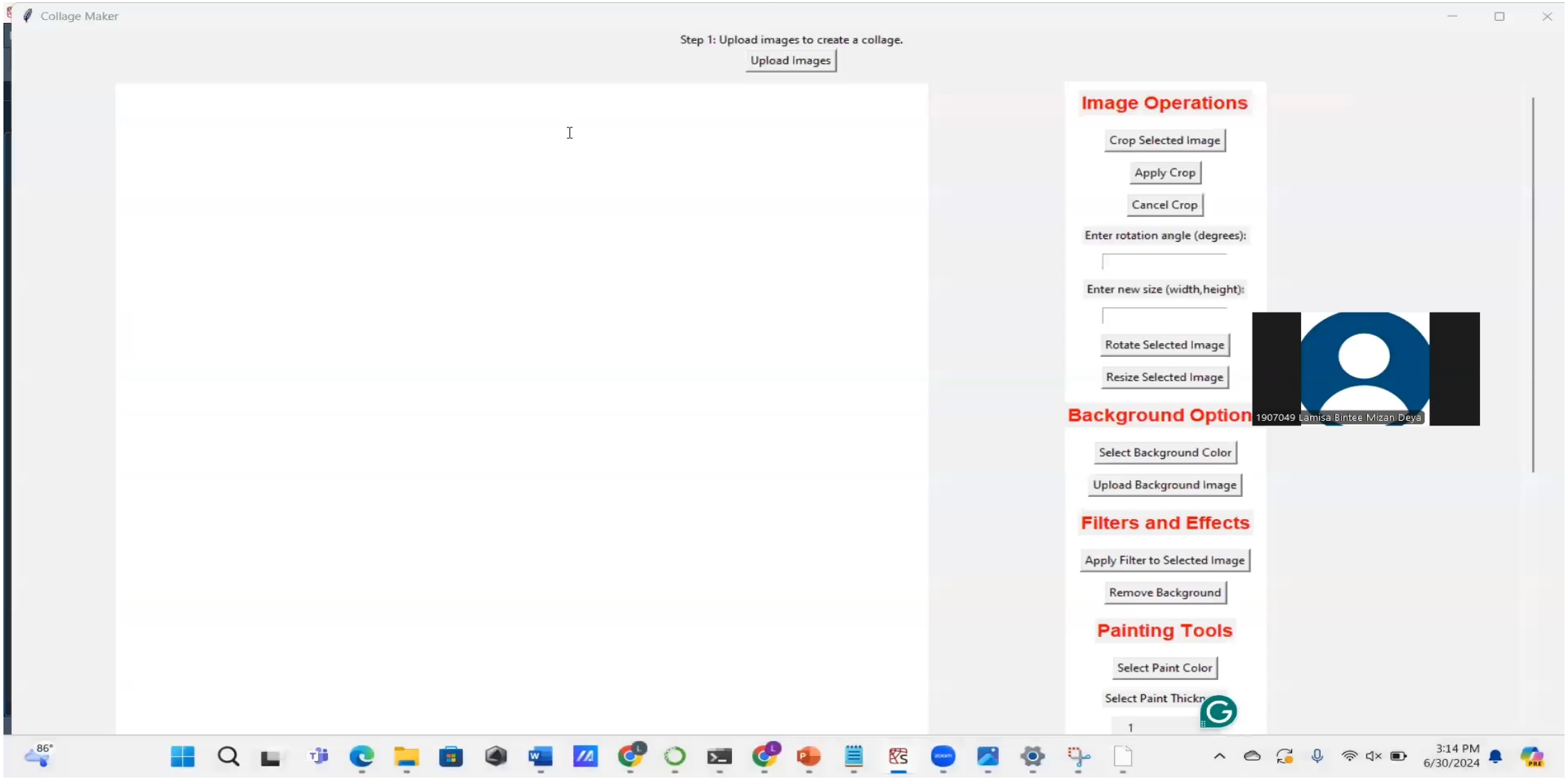
Objectives

- Learn about different libraries used for image processing
- Know about different geometrical operations
- To be familiar with how geometrical operations can be applied to an image
- To be able to use spatial filters
- To learn about the custom ‘u2net.onnx’ model.
- To know about how to paint and add text on an image

Introduction

- Image Collage maker
  - Combines multiple images into a single image
  - Provides a creative way to showcase memories, highlight a series of photos, or present a visual story.
- Key Features:
  - ✓ User-Friendly Interface
  - ✓ Editing Tools (cropping, rotating, resizing, and applying filters)
  - ✓ Text and Graphics
  - ✓ Background Options

Demonstration



Upload Image

- ❖ Opens a file dialog for the user to select image files.
- ❖ For each selected image, it opens the image, converts it to RGBA format.
  - The Alpha channel represents the transparency level of each pixel.
  - It allows for smoother blending and overlay effects
- ❖ Places the image on the canvas at coordinates (50, 50) with the top-left corner

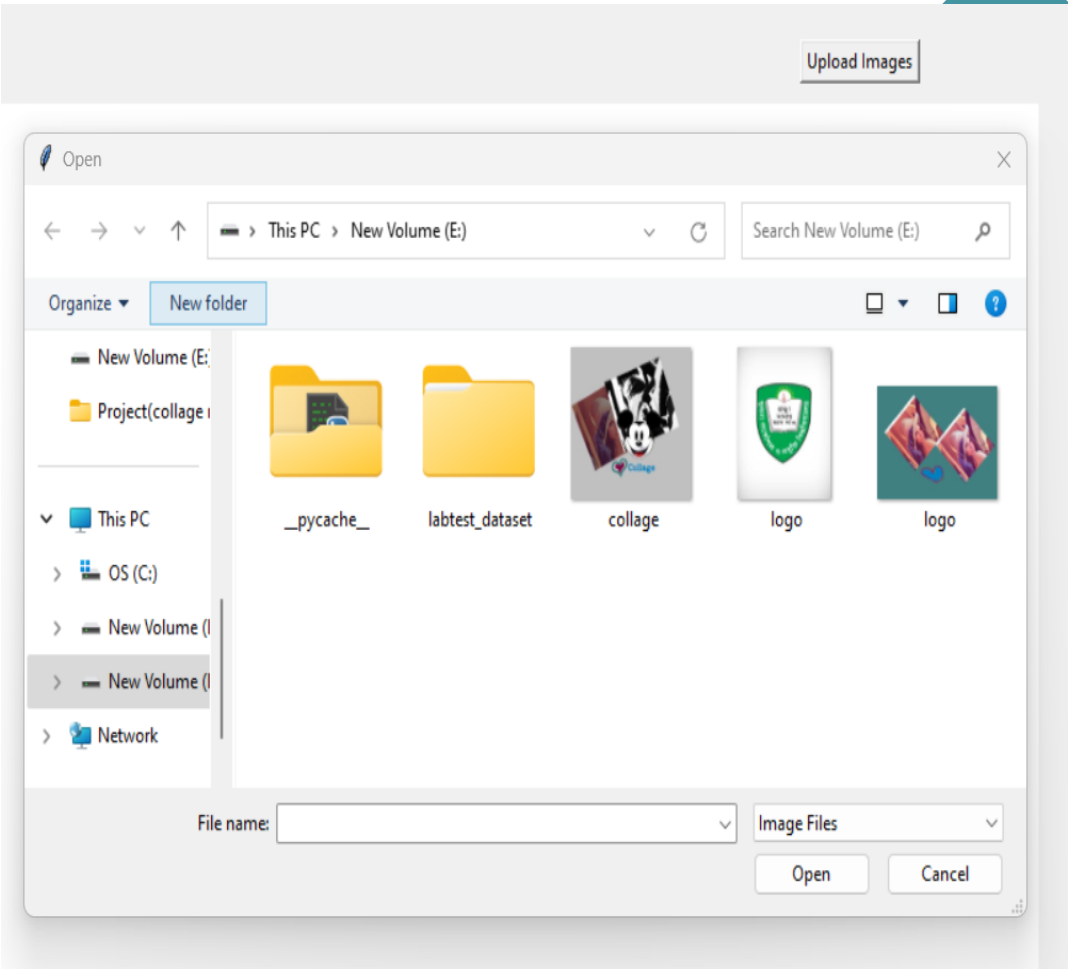


Figure 1: Upload Image

Select & Drag Image

❖Select

- Selects an image on the canvas based on the mouse click location.
- Calculate the closest canvas image item as the selected item

❖Drag

- Updates the image's coordinates to follow the mouse movement.
- Ensures the selected image stays on top of other canvas items using Tkinter library’s tag_raise method

Figure 2: Drag Image

Crop Image

- Binds left mouse button click, drag, and release events to initiate, update, and finalize the crop rectangle.
- Saves the initial x and y coordinates when the user clicks to start the crop.
- Draws a rectangle from the initial click position to the current mouse position
- Updates the coordinates of the cropping rectangle based on the mouse's current position
- Determines the coordinates of the crop box.
- Crops the selected image based on the calculated crop box.

Crop

Figure 3: Before Crop

Figure 4: After Crop

Rotate Image

- Image is rotated by the user's specified angle
- Define rotation matrix

$$R = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$$

- Define corner matrix

$$\text{Corner matrix} = \begin{bmatrix} 0 & 0 \\ W & 0 \\ W & H \\ 0 & H \end{bmatrix}$$

Rotate Image (Cont.)

- Dot product of these two to find new corners
- Subtract max and min of new corner in x-direction to get new width.
- Subtract max and min of new corner in y-direction to get new height.
- For each pixel (x, y) in the new image, translate the coordinates using the formula:

$$x_{\text{centered}} = x - \text{new\_center\_x}$$

$$y_{\text{centered}} = y - \text{new\_center\_y}$$

- Translate the coordinates back to the original image's coordinate system:

$$\begin{bmatrix} x_{\text{original}} \\ y_{\text{original}} \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix}^{-1} \begin{bmatrix} x_{\text{centered}} \\ y_{\text{centered}} \end{bmatrix}$$

- Copy the pixels

Rotate Image (Cont.)

- Apply the inverse rotation matrix to the centered coordinates to find the corresponding coordinates in the original image

$$\begin{bmatrix} x_{\text{original}} \\ y_{\text{original}} \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix}^{-1} \begin{bmatrix} x_{\text{centered}} \\ y_{\text{centered}} \end{bmatrix}$$

- Translate the coordinates back to the original image's coordinate system:

$$x_{\text{original\_translated}} = x_{\text{original}} + \text{center\_x}$$

$$y_{\text{original\_translated}} = y_{\text{original}} + \text{center\_y}$$

- Copy the pixels

Rotate

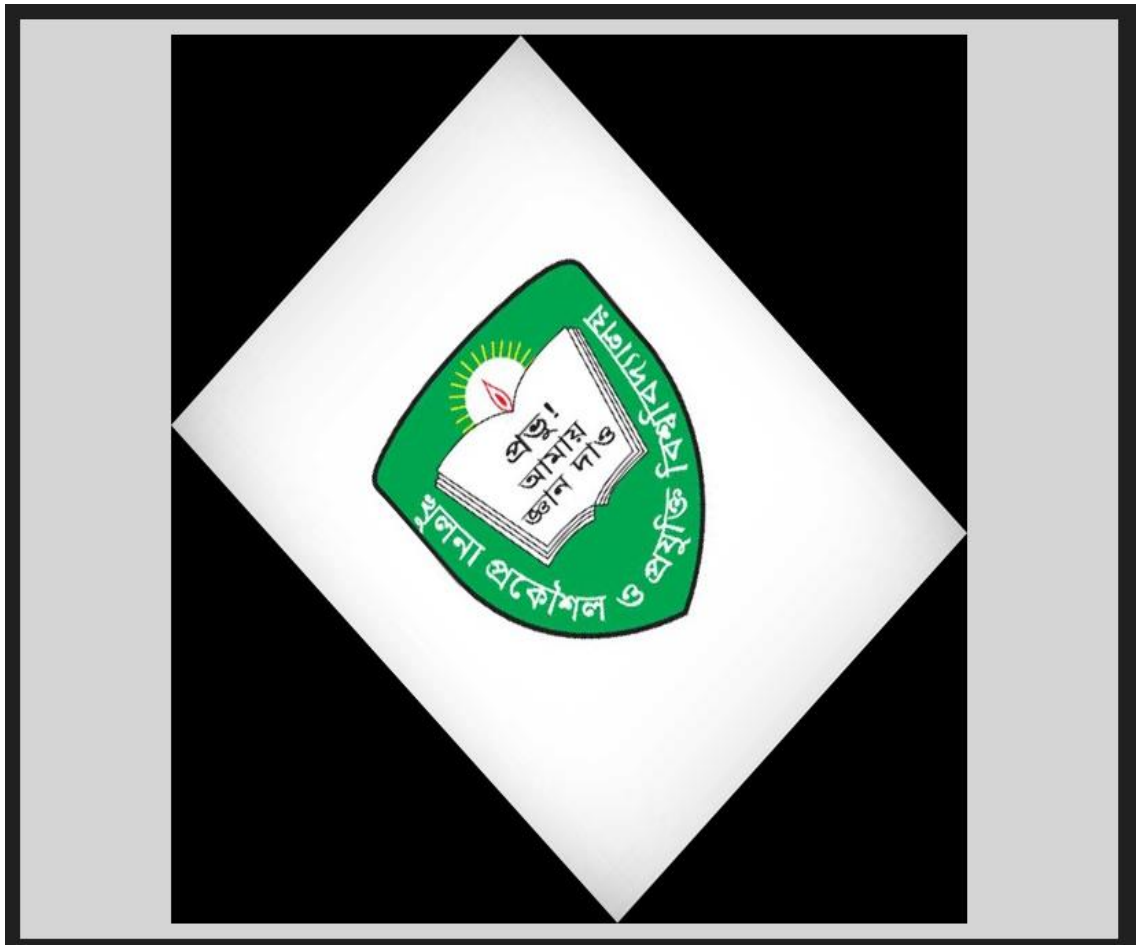


Figure 5: Rotate using default rotate function

Figure 6: Rotate using manual function

Background

- 2 options
  - ✓ Background Color
  - ✓ Background Image
- For background color:
  - Opens a color chooser dialog to select a background color
- For background image:
  - Opens a file dialog to select an image file
  - Resizes the background image to fit the canvas
  - Use canvas.tag_lower() to ensure that the background image is at the lowest layer

Background Image

Figure 7: Select Background Image

Filters

- 3 options
  - ✓ Brightening
  - ✓ Contrast change
  - ✓ Sharpening
- Adjusts brightness by multiplying each pixel value by the user-defined factor.
- Adjusts contrast by shifting pixel values relative to the mean brightness.
- Adjusts sharpness by blending the original image with a blurred version.
  - Apply a Gaussian blur to the image using a manually created kernel.

Filters

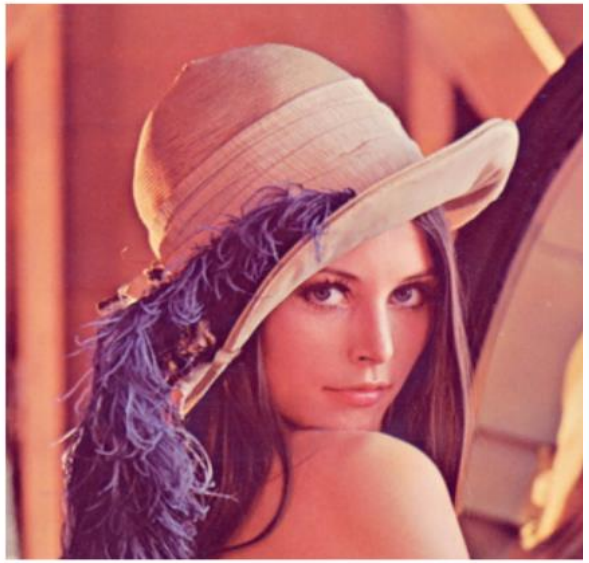


Figure 8: Input Image

Figure 9: Brightening

Figure 10: Apply Contrast

Figure 11: Apply Sharpness

Figure 12: Apply all Filters

Background Remove

- Select an image on the canvas.
- Convert the image to a byte stream.
- Load the custom ‘u2net.onnx’ model.
  - ✓ Deep learning model designed for image segmentation, specifically for accurately extracting objects from their backgrounds.
- Use ‘rembg’ library to remove the background.

Background Remove

Figure 13: Image with Background

Figure 14: Image without Background

Paint & Erase

- Open a color chooser dialog to select a color.
- Calculate painting coordinates based on mouse position.
- Use the ‘ellipse’ method to draw an ellipse in the canvas.
- The ellipse method takes a bounding box as input, specified by two points: the top-left corner and the bottom-right corner.
- (x1-thickness, y1-thickness, x2+thickness, y2+thickness)
- Fill the ellipse with the user defined color.
- To erase, draw a transparent circle.

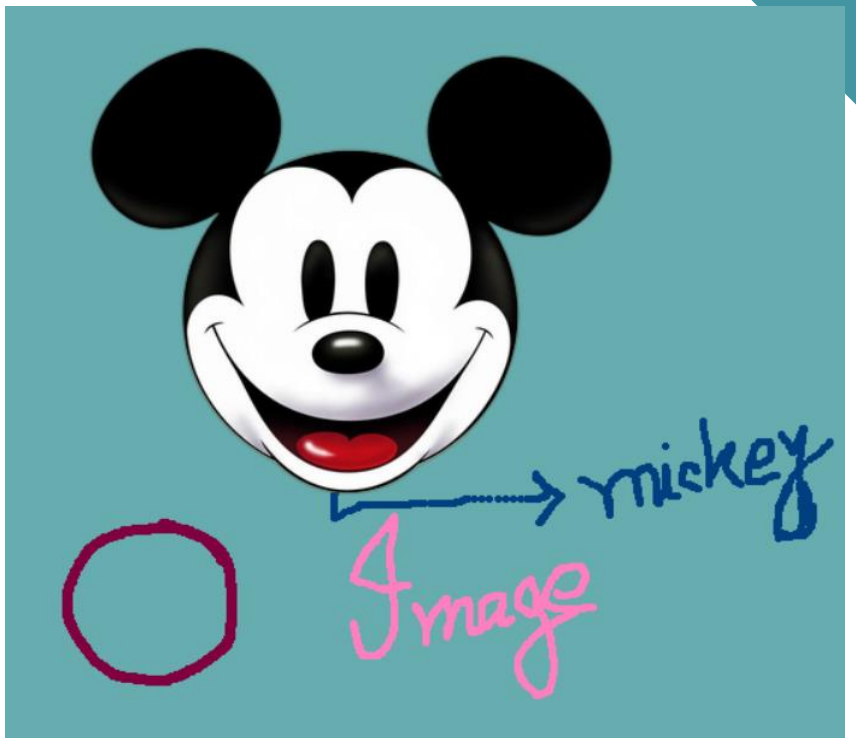


Figure 15: Apply Paint

Add Text

- Open a color chooser dialog to select text color.
- Retrieve the text, font size, and font name from user input fields.
- Set default text color(black) if none is selected.
- Retrieve the coordinates of the mouse click event.
- Create the text item on the canvas at the clicked position by canvas.create_text() method
- Store the ID of the created text item for future reference

Figure 16: Apply Text

Save Collage

- Initializes a blank canvas (numpy array) with dimensions 800x800
- Take 4 channel RGBA and set them to 255
- If the background is an image
  - Blend it with the canvas using result = cv2.addWeighted(src1, alpha, src2, beta, gamma)
- If the background is color
  - Set the bg color to the full canvas with full opacity
- Add Paint
  - The temporary paint buffer is applied to the main canvas using alpha blending.
- Add Image
  - Image converts to a NumPy array
  - Placed on the canvas, with their positions and dimensions
  - Alpha blending is used to correctly overlay the images.

Save Collage(Cont.)

- Add text
  - Converts the color from hexadecimal to BGR format.
  - The appropriate OpenCV font type is selected based on the font name.
  - The text is drawn on the canvas using ‘cv2.putText’
- Save Collage
  - Select a directory to save the collage.
  - The canvas is converted from RGBA to BGR format for compatibility with JPEG.
  - The final image is saved as a JPEG file

Collage

Figure 17: Collage of the images

Conclusion

- Successfully combines user-friendly features with advanced image processing techniques.
- Have implemented basic features like filtering
- Have applied cropping, rotating, resizing
- Also do the removal of background
- Have successfully added text and paint
- Set different background images and color
- Have implemented the ideas that have been taught in the lab.

Thank you