

Name: Arooj shareef

Class: BS AI (4A)

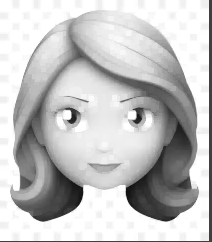
Roll no: su92-bsaim-f23-047

Lab task: 05

Submitted to: Sir Rasikh Ali

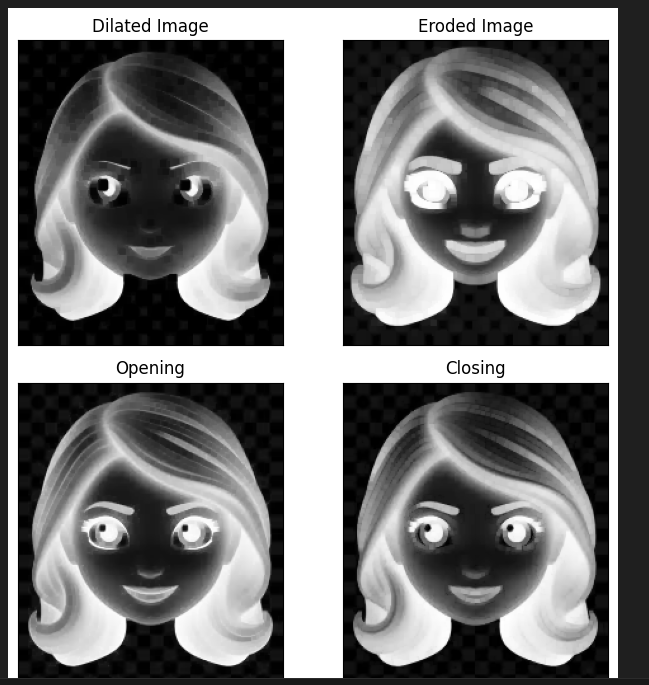
**Description of opencv.**

1. **Image Loading:**  
     
   The image is read in grayscale mode (0) from the given file path using cv2.imread.  
     
   **Kernel Creation:**  
     
   A kernel of size 5x5 (structuring element) is generated using np. ones. This kernel is employed for morphological operations such as erosion and dilation.  
     
   **Morphological Operations:**  
     
   **Erosion:**  
     
   The cv2.erode function is used on the image, which erodes the edges of foreground objects. This operation eliminates small white noises and reduces the objects in the image.  
     
   **Dilation:**  
     
   cv2.dilate is performed on the image, which dilates the object boundaries. This increases the object size and fills gaps in small objects.  
     
   **Display Results:**  
     
   The original grayscale image, the eroded image, and dilated image are each shown in a different window using cv2.imshow.

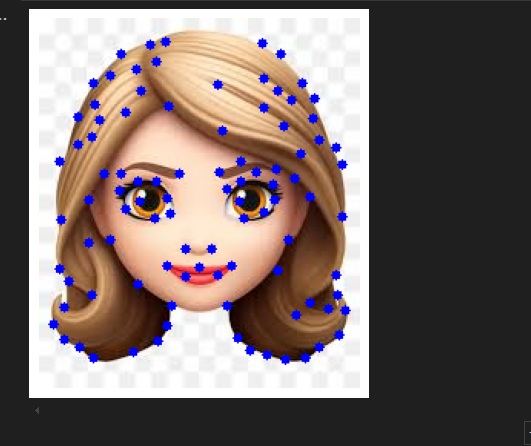
  

1. This **morphological image processing** operations (dilation, erosion, opening, and closing) on a grayscale image using OpenCV and visualizes the results using Matplotlib.

**Image Loading:**  
  
Loads an image from the given file path and converts it to grayscale via cv2.cvtColor.  
  
**Morphological Operations:**  
  
**Dilation**: Dilates object boundaries via cv2.dilate.  
  
**Erosion**: Thins object boundaries via cv2.erode.  
  
**Opening**: Eliminates noise via cv2.morphologyEx with cv2.MORPH\_OPEN.  
  
**Closing**: Fills small holes via cv2.morphologyEx with cv2.MORPH\_CLOSE.  
  
**Visualization:**  
  
Utilizes Matplotlib to show the original and processed images in a 2x2 subplot grid:  
  
Dilated Image  
  
Eroded Image  
  
Opening  
  
Closing  
  
**Plot Customization:**  
  
Titles are added to each subplot, and axis ticks are hidden for better visualization.

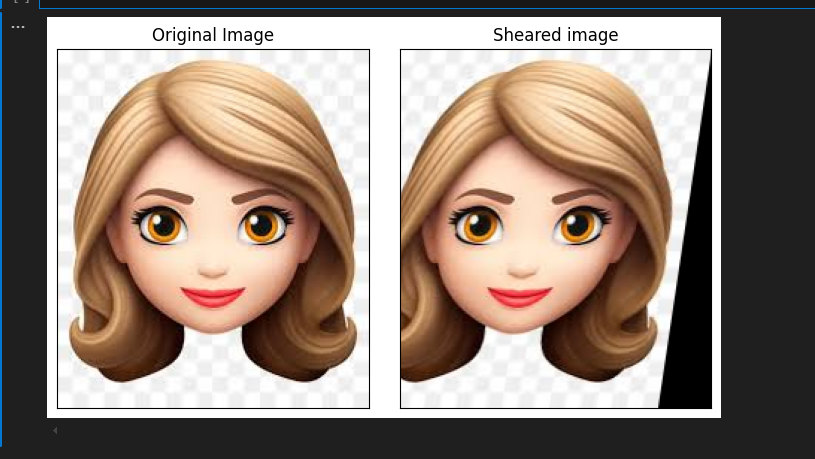


3.This code reads an image, converts it to grayscale, and detects up to 100 strong corners using the Shi-Tomasi method. It then marks these corners with blue dots on the original image and displays the result using Matplotlib. The image path is checked for validity, and the script exits if the image is not found.



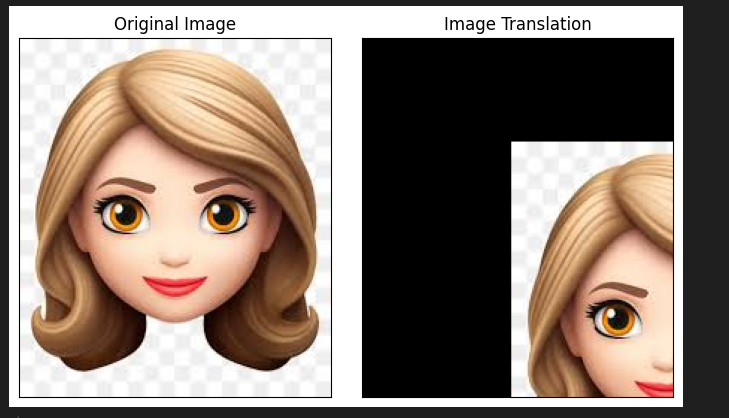
**4**-A **shear transformation** on an image using OpenCV and displays the original and sheared images side by side using Matplotlib. Here's a brief description of the code:

1. **Image Loading**: The image is loaded from the specified file path using cv2.imread.
2. **Color Conversion**: The image is converted from BGR (default in OpenCV) to RGB format using cv2.cvtColor for proper display with Matplotlib.
3. **Shear Transformation**: A 2D shear transformation matrix is defined with a horizontal shear factor (shear = -0.15) and no vertical shear (shear = 0). The cv2.warpAffine function applies this transformation to the image.
4. **Display**: The original and sheared images are displayed side by side using Matplotlib. The axes ticks are removed for a cleaner visualization.
5. **Output**: The script shows the original image and the sheared image, demonstrating the effect of the shear transformation.



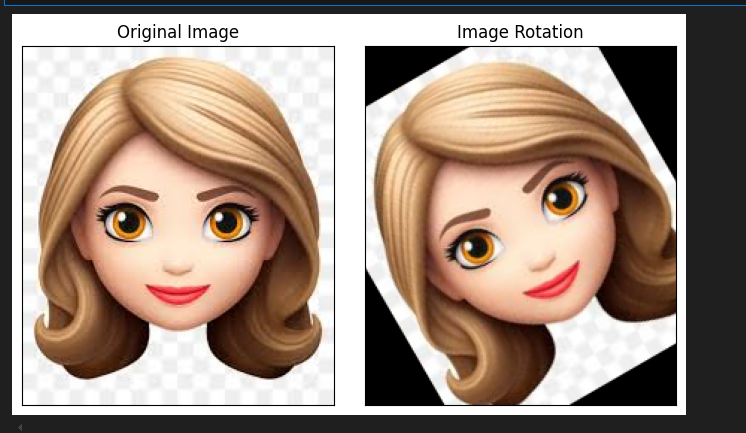
**5-** A **image translation** (shifting the image along the x and y axes) using OpenCV and displays the original and translated images side by side using Matplotlib. Here's a brief description of the code:

1. **Image Loading**: The image is loaded from the specified file path using cv2.imread.
2. **Color Conversion**: The image is converted from BGR (default in OpenCV) to RGB format using cv2.cvtColor for proper display with Matplotlib.
3. **Translation Transformation**: A 2D translation matrix is defined with horizontal (Tx = 100) and vertical (ty = 70) shift values. The cv2.warpAffine function applies this transformation to the image.
4. **Display**: The original and translated images are displayed side by side using Matplotlib. The axes ticks are removed for a cleaner visualization.
5. **Output**: The script shows the original image and the translated image, demonstrating the effect of the translation transformation.



6- **image rotation** using OpenCV and displays the original and rotated images side by side using Matplotlib. Here's a brief description of the code:

1. **Image Loading**: The image is loaded from the specified file path using cv2.imread.
2. **Color Conversion**: The image is converted from BGR (default in OpenCV) to RGB format using cv2.cvtColor for proper display with Matplotlib.
3. **Rotation Transformation**:
   * The center of rotation is calculated as the midpoint of the image.
   * A rotation matrix is created using cv2.getRotationMatrix2D with the specified angle (30 degrees) and scale (1, meaning no scaling).
   * The cv2.warpAffine function applies the rotation transformation to the image.
4. **Display**: The original and rotated images are displayed side by side using Matplotlib. The axes ticks are removed for a cleaner visualization.
5. **Output**: The script shows the original image and the rotated image, demonstrating the effect of the rotation transformation.



A **image scaling** (zooming in and out) using OpenCV and displays the original, zoomed, and scaled-down images side by side using Matplotlib. Here's a brief description of the code:

1. **Image Loading**: The image is loaded from the specified file path using cv2.imread.
2. **Color Conversion**: The image is converted from BGR (default in OpenCV) to RGB format using cv2.cvtColor for proper display with Matplotlib.
3. **Scaling (Zooming In)**:
   * A scale factor of 3.0 is applied to zoom into the image.
   * The image is resized using cv2.resize with cv2.INTER\_CUBIC interpolation for smoother results.
4. **Scaling (Zooming Out)**:
   * A scale factor of 1/3.0 is applied to scale down the image.
   * The image is resized using cv2.resize with cv2.INTER\_AREA interpolation, which is suitable for shrinking images.
5. **Display**: The original, zoomed, and scaled-down images are displayed side by side using Matplotlib. The title of each subplot includes the shape (dimensions) of the corresponding image. The axes ticks are removed for a cleaner visualization.
6. **Output**: The script shows the original image, the zoomed-in image, and the scaled-down image, demonstrating the effects of different scaling operations.

