Assignment # 4

[Find errors and complete the steps]

[Note]

Use the PyCharm software if google colab is not working properly

Basic Operations with 1D and 2D NumPy Arrays

1. Install and import numpy library

```
#import numpy library

import numpy as np
```

2. Create 1D and 2D arrays

```
arr_1d = np.array([1, 2, 3, 4, 5])
arr_2d = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
```

3. Print 1D and 2D array outputs

```
print("1D Array:", arr_1d)
print("2D Array:\n", arr_2d)
```

4. Basic Array Operations

```
print("Sum of 1D Array:", np.sum(arr_1d))
print("Mean of 2D Array:", np.mean(arr_2d))
print("Transpose of 2D Array:\n", arr_2d.T)
```

[Your Task]

```
1. Create a function for this above program
2. Make the arr_1d and arr_2d to global variable
3. Print the output
```

Image Processing with NumPy (Indexing & Slicing in Action)

```
# Creating a grayscale image using a 2D NumPy array
    image = np.random.randint(0, 256, (5, 5), dtype=np.uint8)

    print("Original Image:\n", image)

    # Slicing a portion of the image
    cropped = image[1:4, 1:4]
    print("Cropped Section:\n", cropped)

    # Inverting colors
    inverted_image = 255 - image
    print("Inverted Image:\n", inverted_image)
```

[Your Task]

```
1. Create a function for this above program
2. Do not use global variable
3. Print the output
```

Augmented Reality Transformation – Perform linear algebra operations like scaling, rotation, and translation.

```
import numpy as np
import cv2

# Load an image
image = cv2.imread('image.jpg')  #Use your own image

# Scaling Transformation
```

```
def scale_image(image, scale_factor):
    # Create the scaling matrix
    scaling_matrix = np.array([[scale_factor, 0, 0],
                               [0, scale_factor, 0],
                               [0, 0, 1]])

    # Get the image dimensions
    rows, cols = image.shape[:2]

    # Apply the transformation
    scaled_image = cv2.warpPerspective(image, scaling_matrix, (cols, rows))
    return scaled_image

# Rotation Transformation
def rotate_image(image, angle):
    # Get the image dimensions
    rows, cols = image.shape[:2]

    # Get the rotation matrix
    rotation_matrix = cv2.getRotationMatrix2D((cols / 2, rows / 2), angle,
1)

    # Apply the rotation
    rotated_image = cv2.warpAffine(image, rotation_matrix, (cols, rows))
    return rotated_image

# Translation Transformation
def translate_image(image, tx, ty):
    # Create the translation matrix
    translation_matrix = np.array([[1, 0, tx],
                                   [0, 1, ty],
                                   [0, 0, 1]])

    # Get the image dimensions
    rows, cols = image.shape[:2]

    # Apply the transformation
    translated_image = cv2.warpPerspective(image, translation_matrix, (cols,
rows))
    return translated_image

# Example transformations
```

```
scaled_image = scale_image(image, 1.5)  # Scale by a factor of 1.5
rotated_image = rotate_image(image, 45)  # Rotate by 45 degrees
translated_image = translate_image(image, 50, 30)  # Translate by 50 pixels
along x and 30 pixels along y

# Show the results
cv2.imshow('Original', image)
cv2.imshow('Scaled', scaled_image)
cv2.imshow('Rotated', rotated_image)
cv2.imshow('Translated', translated_image)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

[Your Task]

```
1. Print and show the output
```

Face Detection from Image Arrays – Extract facial features by slicing a NumPy-based image array.

```
import cv2
import numpy as np

# Load an image and convert it to a NumPy array
image = cv2.imread('path_to_image.jpg')  # Replace with your image path
image_gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# Load OpenCV's pre-trained Haar Cascade for face detection
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades +
'haarcascade_frontalface_default.xml')

# Detect faces in the image
faces = face_cascade.detectMultiScale(image_gray, scaleFactor=1.1,
minNeighbors=5)

# Loop through the detected faces and extract facial features (regions)
for (x, y, w, h) in faces:
    # Slice the image array to extract the face region
    face_region = image[y:y+h, x:x+w]

    # Optional: Display the face region
    cv2.imshow('Face Region', face_region)

    # Extract additional facial features if required (e.g., eyes, nose)
    eyes_cascade = cv2.CascadeClassifier(cv2.data.haarcascades +
'haarcascade_eye.xml')
    eyes = eyes_cascade.detectMultiScale(face_region, scaleFactor=1.1,
minNeighbors=5)

    for (ex, ey, ew, eh) in eyes:
        eye_region = face_region[ey:ey+eh, ex:ex+ew]
        cv2.imshow('Eye Region', eye_region)

# Show the original image with detected faces
cv2.imshow('Detected Faces', image)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

[Your Task]

```
1. Print and show the output
```