Name – Anuj Koli

Roll no. 22102A0013

MV Assignment 1

1. Implement basic image manipulation and enhancement techniques using Python.

```
import cv2

import numpy as np

# Load image

img = cv2.imread('C:\\Users\\anujk\\Desktop\\New_Start\\MV assignments\\1\\image.jpg')

# Resize image

resized_img = cv2.resize(img, (300, 300))

# Convert to grayscale

gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# Increase brightness

bright_img = cv2.convertScaleAbs(img, alpha=1.2, beta=50)

# Alpha controls contrast, beta controls brightness

# Save output

cv2.imwrite('resized_image.jpg', resized_img)

cv2.imwrite('gray_image.jpg', gray_img)

cv2.imwrite('bright_image.jpg', bright_img)
```

Original Image

Resized Image

Gray Image

Bright Image

1. Study and Implement advanced Image Manipulation and Enhancement techniques like Image Sharpening, Edge Enhancement, Noise Removal, Image Restoration, adding text and Watermarking in an image using python.

```
import cv2

import numpy as np

# Load image

img = cv2.imread('C:\\Users\\anujk\\Desktop\\New_Start\\MV assignments\\2\\image.jpg')

# Sharpen image using a kernel

kernel = np.array([[0, -1, 0], [-1, 5, -1], [0, -1, 0]])

sharpened_img = cv2.filter2D(img, -1, kernel)
```

```
# Noise removal using Gaussian Blur

denoised_img = cv2.GaussianBlur(img, (5, 5), 0)

# Add text watermark

watermarked_img = img.copy()

cv2.putText(watermarked_img, 'Watermark', (40, 30), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 0), 2)

# Save output

cv2.imwrite('sharpened_image.jpg', sharpened_img)

cv2.imwrite('denoised_image.jpg', denoised_img)

cv2.imwrite('watermarked_image.jpg', watermarked_img)
```

Original Image

Sharpened Image

Denoised Image

Watermarked Image

Watermark

1. Study and Implement advanced Image Manipulation and Enhancement techniques like Image Sharpening, Edge Enhancement, Noise Removal, Image Restoration, adding text and Watermarking in an image using python.

```
import cv2

import numpy as np

# Load image

img = cv2.imread("C:\\Users\\anujk\\Desktop\\New_Start\\MV assignments\\3\\image.jpg")

# Translation

rows, cols = img.shape[:2]

M_translate = np.float32([[1, 0, 50], [0, 1, 100]])  # Shift by 50, 100

translated_img = cv2.warpAffine(img, M_translate, (cols, rows))

# Rotation

M_rotate = cv2.getRotationMatrix2D((cols / 2, rows / 2), 45, 1)  # 45 degrees

rotated_img = cv2.warpAffine(img, M_rotate, (cols, rows))
```

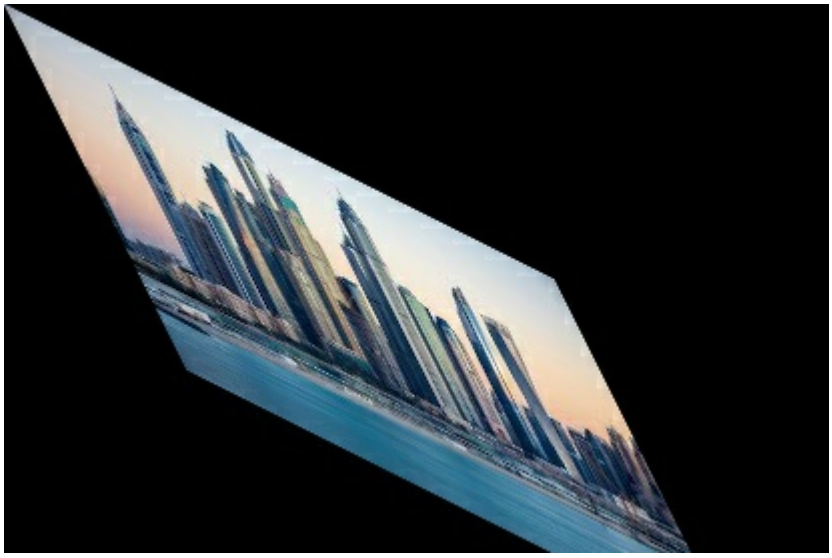
```
# Scaling

scaled_img = cv2.resize(img, None, fx=1.5, fy=1.5)

# Shearing

M_shear = np.float32([[1, 0.5, 0], [0.5, 1, 0]])

sheared_img = cv2.warpAffine(img, M_shear, (int(cols * 1.5), int(rows * 1.5)))

# Flip horizontally

flipped_img = cv2.flip(img, 1)

# Affine transformation

pts1 = np.float32([[50, 50], [200, 50], [50, 200]])

pts2 = np.float32([[10, 100], [200, 50], [100, 250]])

M_affine = cv2.getAffineTransform(pts1, pts2)

affine_img = cv2.warpAffine(img, M_affine, (cols, rows))

# Save output

cv2.imwrite('translated_img.jpg', translated_img)

cv2.imwrite('rotated_img.jpg', rotated_img)

cv2.imwrite('scaled_img.jpg', scaled_img)

cv2.imwrite('sheared_img.jpg', sheared_img)

cv2.imwrite('flipped_img.jpg', flipped_img)

cv2.imwrite('affine_img.jpg', affine_img)
```

Original Image

Rotated Image

Scaled Image

Sheared Image



Flipped Image

Affine Image