CourseName: Computer Vision Lab **Course Code:** CSP-422

Experiment:1.4

Aim: Write a program to evaluate the efficacy of human-guided control point selection for image alignment.

Software Required: Any Python IDE

Description:

Human-guided control point selection is a technique used in image alignment tasks where a human operator manually selects a set of control points to guide the alignment process. These control points are typically landmarks or distinctive features in the images that can be easily identified by a human observer. The overview of the process:

· Image Pair Selection: Choose a pair of images that need to be aligned. These images can be overlapping or have a known transformation between them, such as images of the same scene taken from different viewpoints.

· Control Point Selection: Display the two images side by side and allow the human operator to interactively select control points in both images. The operator can use a mouse or other input device to click on corresponding points in the two images. Ideally, the control points should be distinctive features that can be easily identified in both images.

· Control Point Matching: After the operator selects control points in both images, the corresponding control points need to be identified automatically. This is done by matching the selected control points in one image to their corresponding points in the other image. Various matching techniques can be employed, such as feature-based matching algorithms like SIFT or SURF, or geometric matching methods like RANSAC.

· Transformation Estimation: Once the control point correspondences are established, a transformation model can be estimated to align the images. Common transformation models include translation, rotation, scaling, and affine transformations. More complex transformations, such as projective transformations or non-linear deformations, can also be considered depending on the alignment requirements.

· Alignment and Image Warping: Using the estimated transformation model, the images can be aligned by warping one image to match the other. The transformation can be applied to individual pixels or regions of the image using techniques like interpolation or resampling.

CourseName: Computer Vision Lab **Course Code:** CSP-422

· Evaluation and Refinement: It is important to evaluate the quality of the alignment and visually inspect the results. If necessary, the human operator can adjust or add more control points to refine the alignment. Iterative refinement can be performed until the desired alignment accuracy is achieved.

Steps:

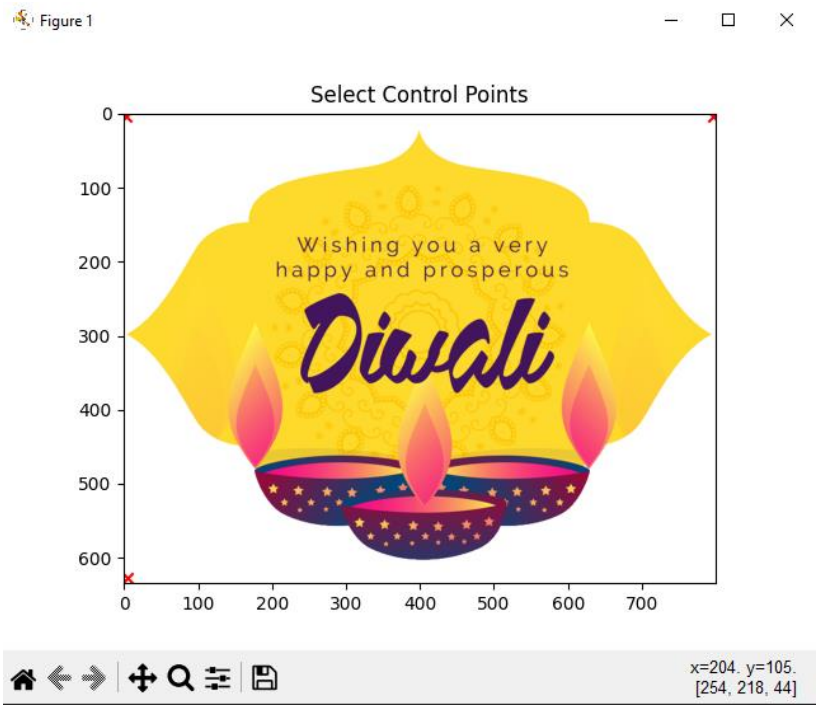
1. Load the source and target images.
2. Display the source and target images to the user.
3. Prompt the user to select corresponding control points on both images.
4. Store the selected control points for alignment.
5. Perform image alignment using the control points.
6. Display the aligned image to the user.
7. Evaluate the alignment quality using metrics such as mean square error or structural similarity index.
8. Calculate the efficacy of human-guided control point selection by comparing the alignment results with and without user intervention.
9. Output the evaluation results and efficacy metrics.

Implementation/Output:

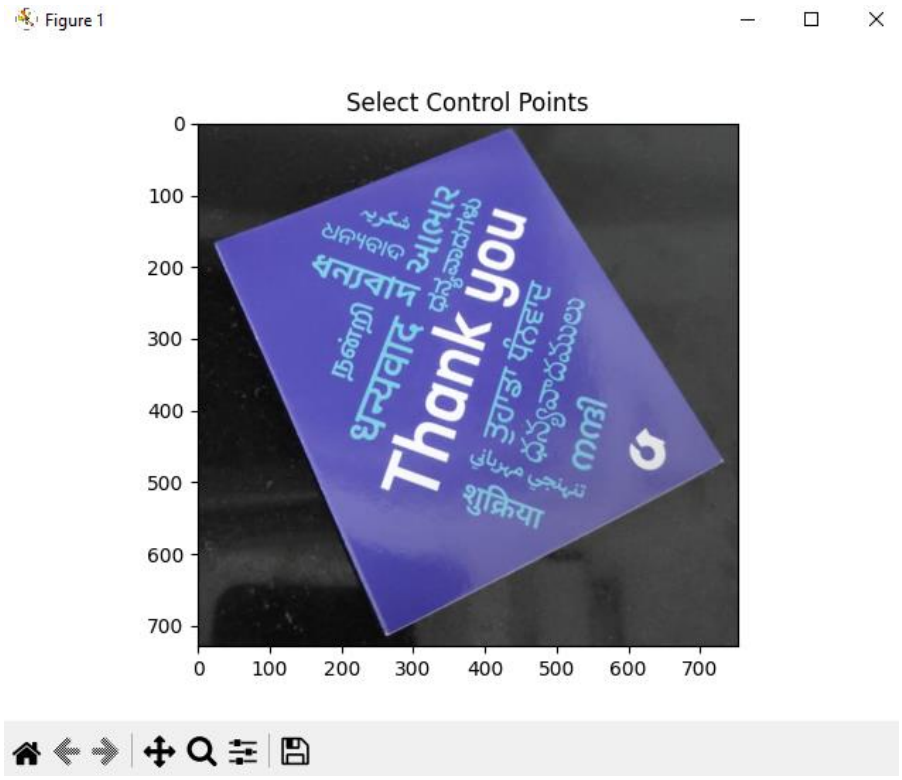
```
import cv2
import numpy as np
import matplotlib.pyplot as plt

def select_control_points(image, num_points=4):
    control_points = []

    plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
    plt.title("Select Control Points")

    for i in range(num_points):
        point = plt.ginput(1, timeout=0)
        control_points.append(point[0])
        plt.scatter(point[0][0], point[0][1], c='r', marker='x')

    plt.close()
    return control_points
```

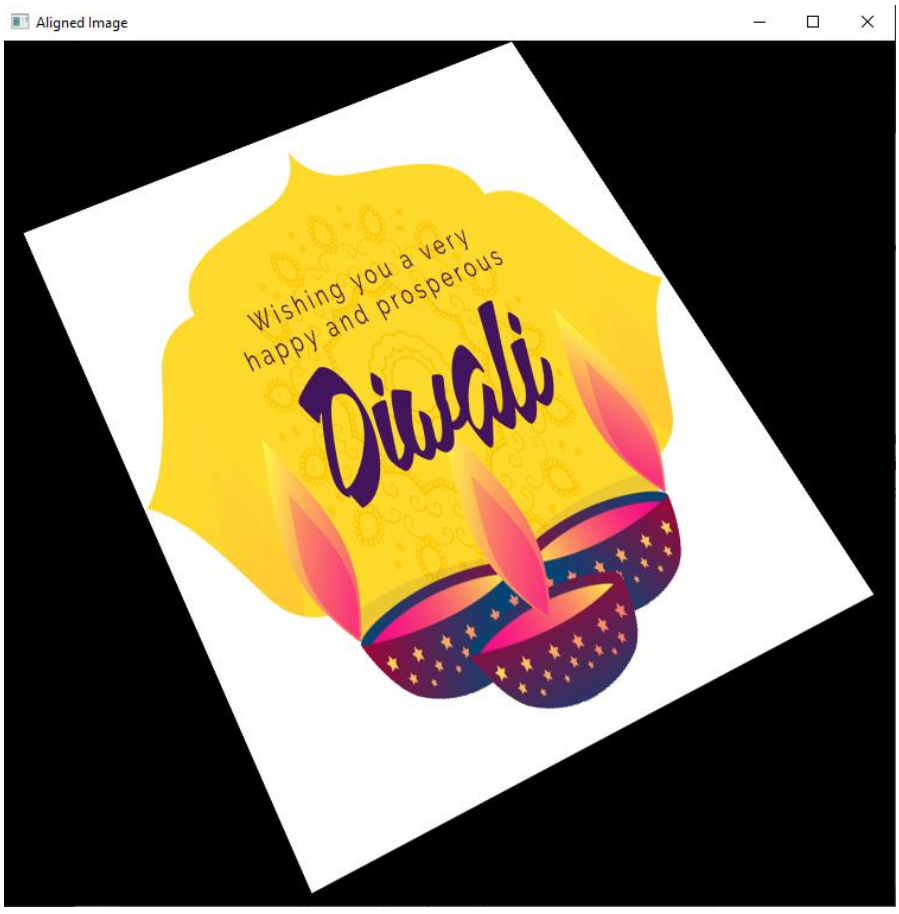
CourseName: Computer Vision Lab **Course Code:** CSP-422

```
def main():
    target_image = cv2.imread('C:/Users/Yash Gupta/PycharmProjects/DSA/image2.png')
    source_image = cv2.imread('C:/Users/Yash Gupta/PycharmProjects/DSA/pngwing.png')

    source_points = select_control_points(source_image)
    target_points = select_control_points(target_image)

    # Calculate the perspective transformation matrix
    M, _ = cv2.findHomography(np.array(source_points), np.array(target_points))

    # Warp the source image
    aligned_image = cv2.warpPerspective(source_image, M, (target_image.shape[1], target_image.shape[0]))

    # Save the aligned image
    cv2.imwrite('aligned_image.jpg', aligned_image)

    cv2.imshow("Aligned Image", aligned_image)
    cv2.waitKey(0)
    cv2.destroyAllWindows()

main()
```



CourseName: Computer Vision Lab **Course Code:** CSP-422





Name: Yash Gupta **UID:** 20BCS5009