Experiment- 10

Student Name: Kush Bhasin
UID: 20BCS7677
Branch: BE(CSE)
Section/Group: 20BCS-1/B
Semester: 7th
Subject Name: Computer Vision Lab
Subject Code: 20CSP- 422

1. **Aim:** Write a program to interpret the effectiveness of template matching techniques for video stabilization tasks

2. **System Requirements:**
   - Python 3.9
   - Visual Studio Code

3. **Description:**

Template matching techniques can be used for video stabilization tasks to estimate and compensate for camera motion in a sequence of frames. Here's how template matching can be applied in video stabilization:

1. **Template Selection:** A template is selected from an initial frame in the video sequence. The template represents a distinct and stable feature or region in the scene that will be used as a reference for stabilization. This can be, for example, a prominent object or a textured region with high contrast.
2. **Template Tracking:** The selected template is then tracked across subsequent frames using template matching techniques. Template matching involves comparing the template with regions in each frame to find the best match based on a similarity measure. Common similarity measures include sum of squared differences (SSD) and normalized cross- correlation (NCC).
3. **Motion Estimation:** The template matching process provides an estimate of the motion between frames by determining the displacement that results in the best match. This motion estimation represents the camera motion or global motion in the scene.
4. **Stabilization Transformation:** Based on the estimated motion, a stabilization transformation is applied to the frames. This transformation compensates for the camera motion and aligns the frames to stabilize the video. The transformation can be a translation, rotation, or affine transformation, depending on the nature of the motion.
5. **Warping and Interpolation:** The stabilized frames are warped according to the stabilization transformation to remove the effects of camera motion. This involves resampling the pixels in the frames to create a stabilized output. Interpolation techniques, such as bilinear or bicubic interpolation, can be used to fill in the gaps and generate smooth transitions between the frames.
6. **Fine-tuning and Refinement:** Template matching may have limitations in handling large or complex camera motions or occlusions. Therefore, additional techniques like feature- based

tracking or optical flow can be integrated to refine the stabilization results and handle challenging scenarios.

4. Steps:

1. Read the input video and select a template region.
2. Extract frames from the video and convert them to grayscale.
3. Apply template matching between the template region and each frame to find the best match.
4. Calculate the motion vectors between consecutive frames based on the template matching results.
5. Estimate the global motion from the motion vectors to stabilize the video.
6. Apply the estimated motion to the frames to obtain the stabilized video.
7. Save the stabilized video to an output file.

5. Code:

```
import cv2
import numpy as np

# Load the video file
video_capture = cv2.VideoCapture('cars.mp4')

# Define the reference template
template = cv2.imread('template.png', cv2.IMREAD_COLOR)

# Define the output video writer
fourcc = cv2.VideoWriter_fourcc(*'XVID')
output = cv2.VideoWriter('stabilized_video.avi', fourcc, 30.0,
(int(video_capture.get(3)), int(video_capture.get(4))))

# Initialize variables for tracking
prev_frame = None
prev_template_match = None

while True:
    ret, frame = video_capture.read()
    if not ret:
        break

    # Convert the frame to grayscale for template matching
    gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # Perform template matching
    result = cv2.matchTemplate(gray_frame, cv2.cvtColor(template,
cv2.COLOR_BGR2GRAY), cv2.TM_CCOEFF_NORMED)
    _, _, _, max_loc = cv2.minMaxLoc(result)
```

```
        # Calculate the displacement from the previous frame
        if prev_template_match is not None:
            displacement = (max_loc[0] - prev_template_match[0], max_loc[1] -
    prev_template_match[1])
        else:
            displacement = (0, 0)

        # Apply the displacement to stabilize the frame
        if prev_frame is not None:
            stabilized_frame = np.roll(prev_frame, displacement, axis=(0, 1))
        else:
            stabilized_frame = frame

        # Update the previous frame and template match location
        prev_frame = stabilized_frame
        prev_template_match = max_loc

        # Write the stabilized frame to the output video
        output.write(stabilized_frame)

        # Display the stabilized frame
        cv2.imshow('Stabilized Frame', stabilized_frame)

        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

    video_capture.release()
    output.release()
    cv2.destroyAllWindows()
```

6. Output:


Properties: adobe1_hcrop_4940x4320_v1_v2.mov
Rate = 48000 Hz
Compressor = mp4a
VR Projection: Equirectangular
VR Layout: Stereoscopic - Over/Under
VR Captured View: 191° Horizontal by 180° Vertical
Data Rate Analysis
Data Rate/Sec
Sample Size
3.4MB
0.0MB
4.8MB
0.0MB
00,00,00,00
00,00,00,10
00,00,00,20
00