DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING
Discover. Learn. Empower.

NAAC GRADE A+
Accredited University

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

Experiment:3.3

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

Software Required: Any Python IDE

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.

Name- Yash Gupta   UID- 20BCS5009

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.

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.

Implementation/Output:

```
import cv2
import numpy as np


def stabilize_video(input_video, template_region, output_video):
    cap = cv2.VideoCapture(input_video)
    template = cv2.imread(template_region, 0)
    template_height, template_width = template.shape

    fourcc = cv2.VideoWriter_fourcc(*'XVID')
    out = cv2.VideoWriter(output_video, fourcc, 30.0, (template_width, template_height))
    ret, prev_frame = cap.read()
    prev_gray = cv2.cvtColor(prev_frame, cv2.COLOR_BGR2GRAY)
    while True:
        ret, frame = cap.read()
        if not ret:
            break
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        result = cv2.matchTemplate(gray, template, cv2.TM_CCOEFF_NORMED)
        min_val, max_val, min_loc, max_loc = cv2.minMaxLoc(result)
        top_left = max_loc
        bottom_right = (top_left[0] + template_width, top_left[1] + template_height)
```

```python
        motion_vector = (top_left[0] - template_width // 2, top_left[1] - template_height // 2)
        out.write(stabilized_frame)
    cap.release()
    out.release()
    cv2.destroyAllWindows()
stabilize_video('input_video.mp4', 'template_region.png', 'output_video.mp4')
```