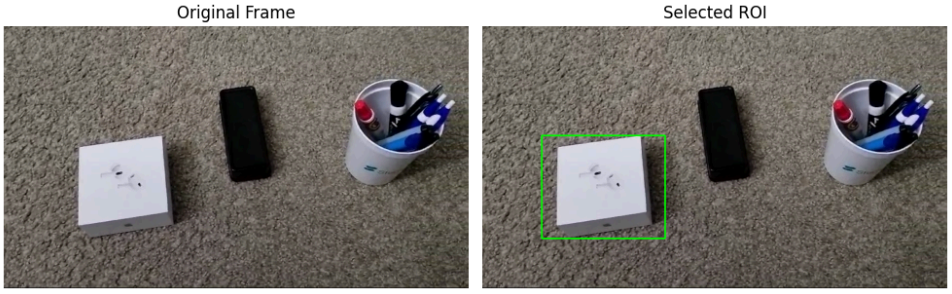
CSc 8830: CV Assignment-3 Solutions Report

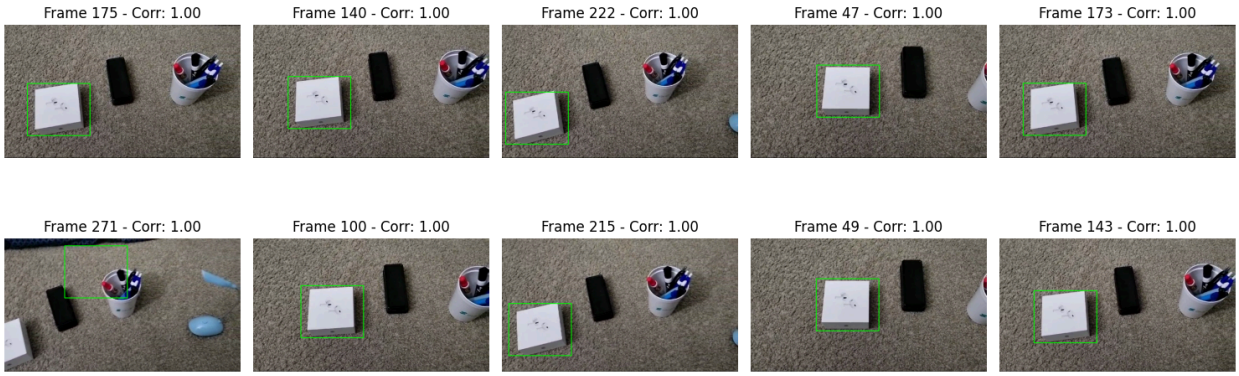
1. **Capture a 10 sec video footage using a camera of your choice. The footage should be taken with the camera in hand and you need to pan the camera slightly from left-right or right-left during the 10 sec duration. Pick any image frame from the 10 sec video footage. Pick a region of interest corresponding to an object in the image. Crop this region from the image. Then use this cropped region to compare with randomly picked 10 images in the dataset of 10 sec video frames, to see if there is a match for the object in the scenes from the 10 images. For comparison use sum of squared differences (SSD) or normalized correlation.**

`A 10 sec video has been taken from a mobile phone camera`

Steps Involved:

1. Setup the Camera: Set up your camera on a stable platform or hold it securely by hand.
2. Record Video: Start recording the video with the camera. Ensure that the camera captures the scene with a slight left-right or right-left panning motion during the 10-second duration.
3. Save Video: Once the recording is complete, save the video footage to your device.
4. Pick Image Frame: Select any frame from the recorded video footage that contains the object or region of interest you want to analyze.
5. Crop Region of Interest: Manually select and crop the region of interest from the selected image frame. Ensure that the cropped region contains the object you want to compare.
6. Compare with Dataset: Randomly pick 10 images from the dataset of 10-second video frames. For each selected image, perform the following steps:
    - Crop the same region of interest as in step 5 from the selected image.
    - Compute the sum of squared differences (SSD) or normalized correlation between the cropped region of interest from the selected image frame and the cropped region of interest from the dataset image.
    - Store the computed similarity metric (SSD or normalized correlation value) for comparison.





2. Solve the following by hand (on paper or typed: Do not just copy it from literature)

(a) Derive the motion tracking equation from fundamental principles. Select any 2 consecutive frames from the set from problem 1 and compute the motion function estimates.

(b). Derive the procedure for performing Lucas-Kanade algorithm for motion tracking when the motion is known to be affine: u(x,y) = a1*x + b1*y + c1; v(x,y) = a2*x + b2*y + c2 (the numbers are subscripts, not power)

Attached Solution PDF

3. Fix a marker on a wall or a flat vertical surface. From a distance D, keeping the camera stationed static (not handheld and mounted on a tripod or placed on a flat surface), capture an image such that the marker is registered. Then translate the camera by T units along the axis parallel to the ground (horizontal) and then capture another image, with the marker being registered. Compute D using disparity based depth estimation in stereo-vision theory. ***(Note: you can pick any value for D and T. Keep in mind that T cannot be large as the marker may get out of view. Of course this depends on D)***

Steps Involved:

1. **Setup:** Fix a marker on a flat vertical surface and ensure it is well-lit and clearly visible. Place the camera on a tripod or a stable surface facing the marker.
2. **Calibration:** Calibrate the stereo camera setup to accurately estimate depth. This involves determining the intrinsic and extrinsic parameters of each camera in the stereo pair. You can use calibration patterns like checkerboards and dedicated calibration software.
3. **Capture Images:**
    - Capture an image of the marker with the camera stationed at a distance D from the marker.
    - Translate the camera horizontally by T units along the axis parallel to the ground.
    - Capture another image of the marker from the new position.

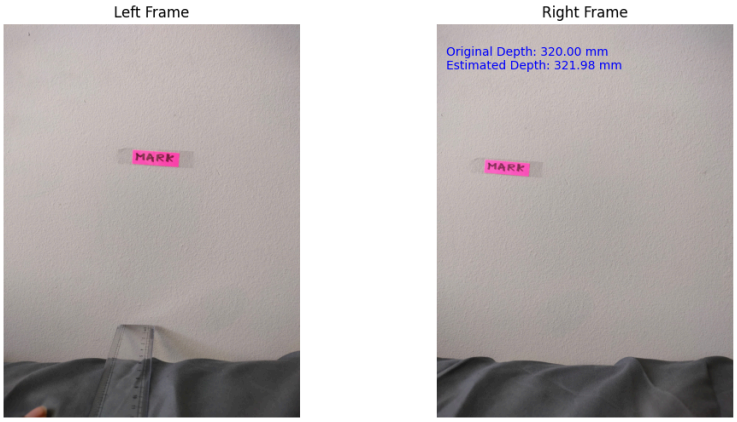
4. **Depth Estimation:**
    - Rectify the stereo images to ensure corresponding points in the two images lie on the same scanline.
    - Compute the disparity map between the rectified stereo images. Disparity represents the horizontal shift of image pixels between the left and right images and is inversely proportional to depth.
    - Use the disparity map and stereo calibration parameters to compute the depth map, which represents the distance of each pixel from the camera.
5. **Compute Distance:**
    - Select corresponding points on the marker in both images. These points should have matching features in both images.
    - Compute the average disparity value for these corresponding points in the disparity map.
    - Use the stereo baseline (distance between the camera centers) and the focal length of the cameras to compute the distance D to the marker using the formula:

$$D = \frac{f \cdot B}{\text{average disparity}}$$

where:

- f is the focal length of the camera.
- B is the baseline distance between the camera centers.

6. **Result:** The computed distance D represents the distance between the camera and the marker.




```
Original distance D: 320.0 mm
Estimated distance D: 321.98240319726756 mm
```

4. For the video (problem 1) you have taken, plot the optical flow vectors on each frame using MATLAB's optical flow codes. (i) treating every previous frame as a reference frame (ii) treating every 11th frame as a reference frame (iii) treating every 31st frame as a reference frame

Scenario (i): Treating every previous frame as a reference frame

1. Load the video into MATLAB.
2. Initialize an optical flow object using the opticalFlow function.
3. Loop through each frame of the video.
4. Compute optical flow between the current frame and the previous frame using the estimateFlow method of the optical flow object.
5. Plot the optical flow vectors on the current frame using the plot method of the optical flow object.
6. Repeat steps 4-5 for each frame.

Scenario (ii): Treating every 11th frame as a reference frame

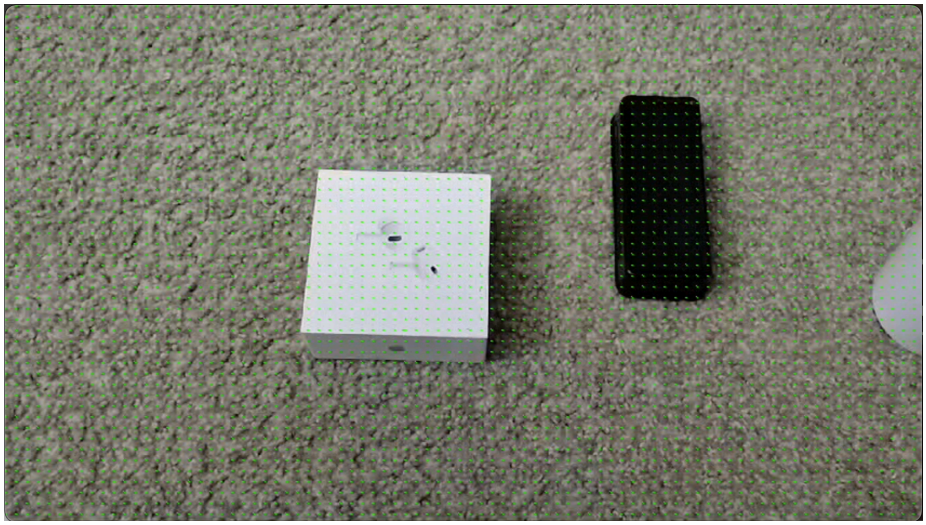
1. Load the video into MATLAB.
2. Initialize an optical flow object using the opticalFlow function.
3. Loop through every 11th frame of the video.
4. Compute optical flow between the current frame and the 10th previous frame (as a reference frame) using the estimateFlow method of the optical flow object.
5. Plot the optical flow vectors on the current frame using the plot method of the optical flow object.
6. Repeat steps 3-5 for each 11th frame.

Scenario (iii): Treating every 31st frame as a reference frame

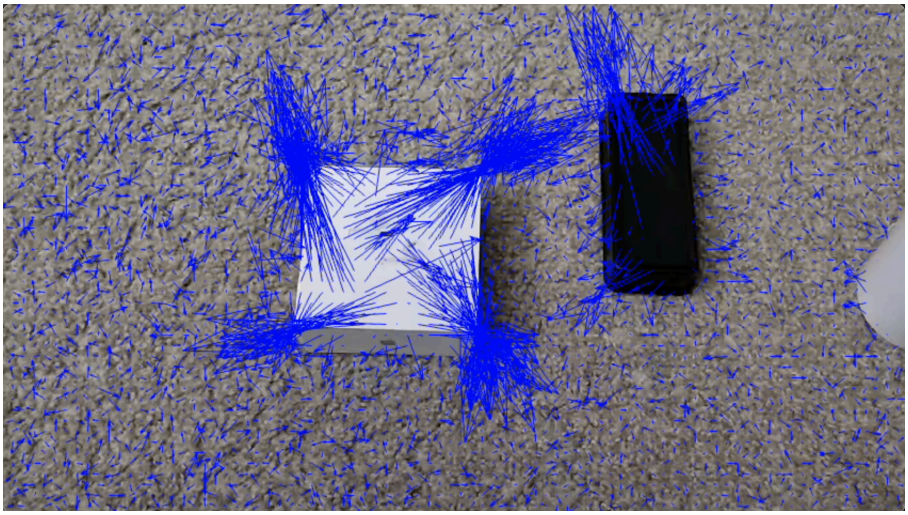
1. Load the video into MATLAB.
2. Initialize an optical flow object using the opticalFlow function.
3. Loop through every 31st frame of the video.
4. Compute optical flow between the current frame and the 30th previous frame (as a reference frame) using the estimateFlow method of the optical flow object.
5. Plot the optical flow vectors on the current frame using the plot method of the optical flow object.
6. Repeat steps 3-5 for each 31st frame.

This question is solved using both Python & Matlab

Python:



Matlab:



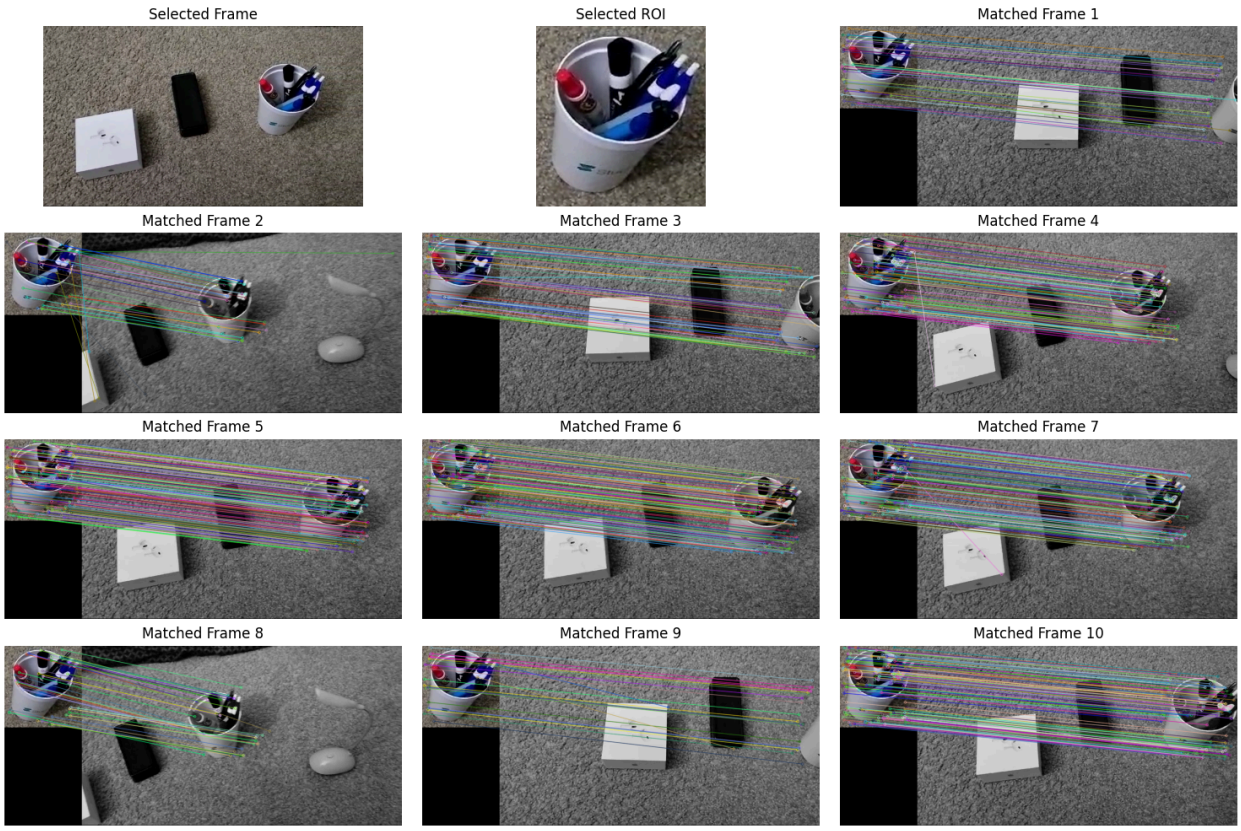
5. Run the feature-based matching object detection on the images from problem (1). ***MATLAB (not mandatory for this problem)*** *Tutorial for feature-based matching object detection is available here:* ***https://www.mathworks.com/help/vision/ug/object-detection-in-a-cluttered-scene-using-point-feature-matching.html***

1. **Load Images:** Load the images from problem (1) into Python using OpenCV's cv2.imread() function.
2. **Feature Detection:** Detect keypoint features in the images using a feature detection

algorithm such as ORB, SIFT, or SURF. For example, you can use the ORB (Oriented FAST and Rotated BRIEF) algorithm, which is efficient and free to use.

3. **Feature Description:** Compute descriptors for the detected keypoints to describe their local image regions. This step is essential for matching keypoints between images.
4. **Feature Matching:** Match keypoints between pairs of images using a feature matching algorithm. OpenCV provides functions like cv2.BFMatcher() for brute-force matching or cv2.FlannBasedMatcher() for FLANN-based matching.
5. **Filter Matches:** Apply a matching ratio test or other filtering techniques to eliminate incorrect matches and retain only reliable matches.
6. **Draw Matches:** Draw the matched keypoints on the images to visualize the detected objects and their corresponding matches.
7. **Object Detection:** Determine the presence and location of the object of interest in the scene based on the matched keypoints.

This is Solved using Python



6. Refer to the Bag of Features example MATLAB source code provided in the classroom's classwork page. In your homework, pick an object category that would be commonly seen in any household (e.g. cutlery) and pick 5 object types (e.g. for cutlery pick spoon, fork, butter knife, cutting knife, ladle). Present your performance evaluation.

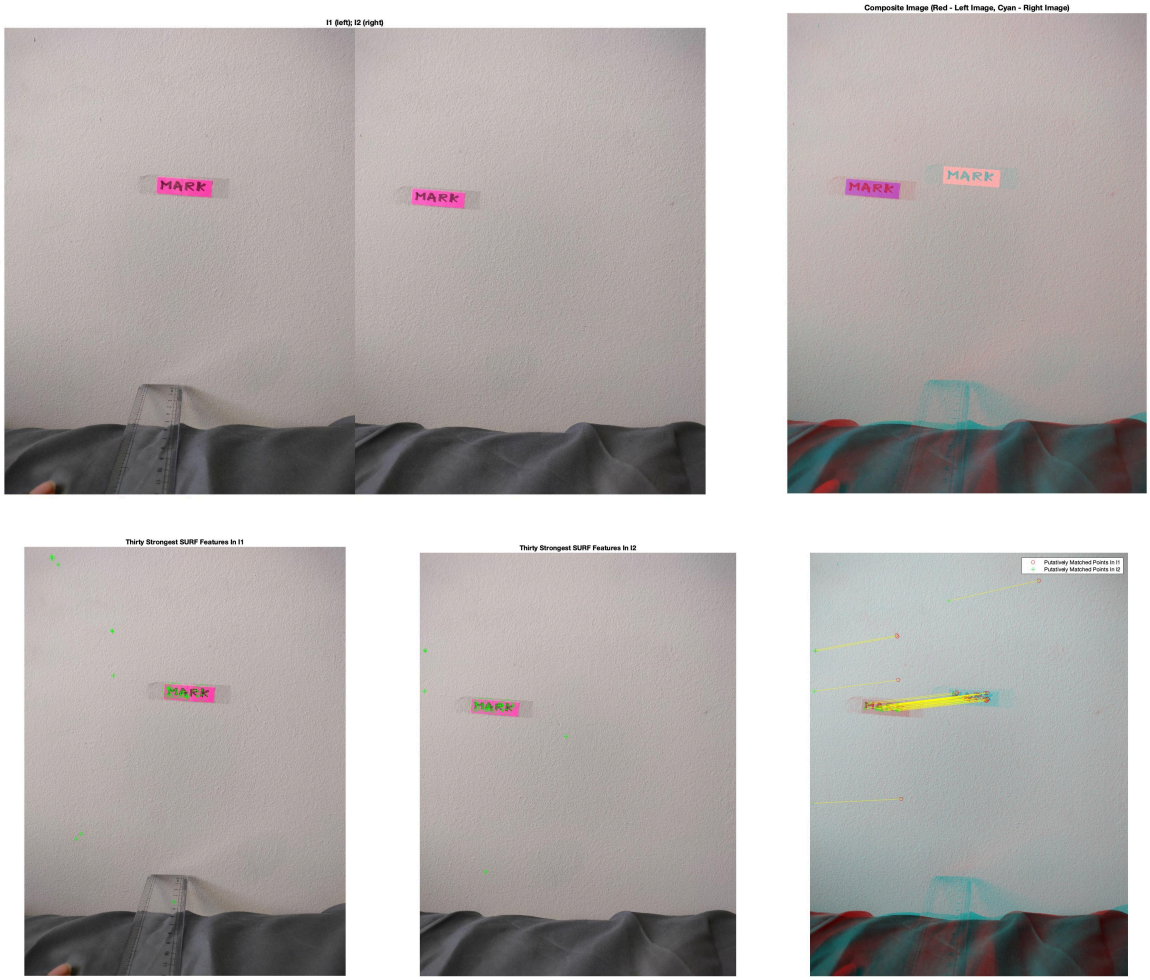
Followed the steps directly provided in the tutorial.

```
Encoding images using Bag-Of-Features.
----------------------------------------
* Encoding an image...done.

Encoding images using Bag-Of-Features.
----------------------------------------
* Encoding an image...done.

Encoding images using Bag-Of-Features.
----------------------------------------
* Encoding an image...done.

Encoding images using Bag-Of-Features.
----------------------------------------
* Encoding an image...done.

Encoding images using Bag-Of-Features.
----------------------------------------
* Encoding an image...done.

Encoding images using Bag-Of-Features.
----------------------------------------
* Encoding an image...done.

Encoding images using Bag-Of-Features.
----------------------------------------
* Encoding an image...done.

Encoding images using Bag-Of-Features.
----------------------------------------
* Encoding an image...done.
Accuracy: 1
```

7. Repeat the image capture experiment from problem (3), however, now also rotate (along the ground plane) the camera 2 (right camera) towards camera 1 position, after translation by T. Make sure the marker is within view. Note down the rotation angle. Run the tutorial provided for uncalibrated stereo rectification in here: https://www.mathworks.com/help/vision/ug/uncalibrated-stereo-image-rectification.html (MATLAB is mandatory for this exercise). Exercise this tutorial for the image pairs you have captured. You can make assumptions as necessary, however, justify them in your answers/description. ***(Note: you can print out protractors from any online source and place your cameras on that when running experiments: http://www.ossmann.com/protractor/conventional-protractor.pdf).***

Followed the steps directly provided in the tutorial.

Rotation Angle: 15 degrees

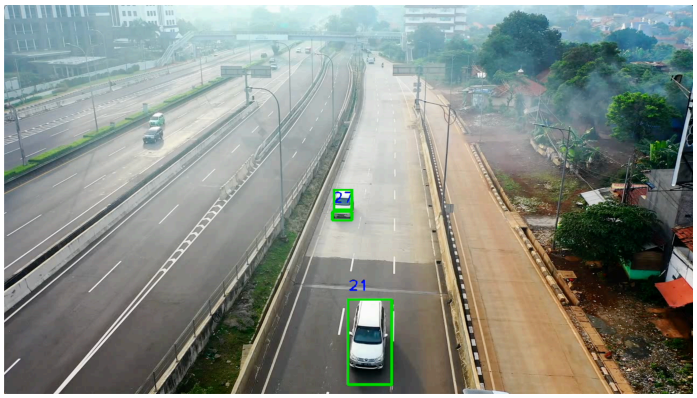


8. Implement a real-time object tracker (two versions) that (i) uses a marker (e.g. QR code or April tags), and (ii) does not use any marker and only relies on the object

1. **With QR Marker:**
    - This version detects and decodes QR codes from the camera feed using the cv2.QRCodeDetector() class. It draws rectangles around detected QR codes and displays the decoded information.
2. **Without Marker:**
    - This version uses background subtraction for object detection.
    - It tracks objects by updating the center points of bounding boxes and assigning unique IDs to them.
    - Each detected object is represented by a bounding box with an ID, which is displayed on the frame.
    - Objects are tracked based on their movement between consecutive frames.

i) 

ii) 



Repo:
https://github.com/ShaikNagurShareef/CSc8830-Computer-Vision/tree/main/Assignment-3