CSc 8830: Computer Vision

Assignment-3 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

Step 1: Capture the 10-second Video Footage

1. Use a camera of your choice to capture a 10-second video footage.
2. While recording, pan the camera slightly from left to right or right to left to capture different scenes during the duration.

Step 2: Extract Frames from the Video

1. Extract frames from the captured video footage.
2. Save these frames as individual images for further processing.

Step 3: Pick a Region of Interest (ROI) from an Image Frame

1. Select a frame from the extracted images.
2. Identify an object of interest in the selected frame.
3. Define a region of interest (ROI) around the object.

Step 4: Compare the ROI with Other Frames

1. Randomly select 10 frames from the dataset of 10-second video frames.
2. For each selected frame:
    - Extract the corresponding region of interest.
    - Compute the sum of squared differences (SSD) or normalized correlation between the ROI from the selected frame and the ROI from the original frame. Store the comparison result.

Step 5: Analyze the Comparison Results

1. Analyze the SSD or normalized correlation values obtained from the comparisons.
2. Determine if there is a match for the object in the scenes from the 10 images based on a predefined threshold or criteria.
3. Optionally, visualize the comparison results to identify any potential matches visually.

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)

Solved in a 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)*

Step 1: Setup Stereo Camera System

- Mount the stereo camera system on a stable platform such as a tripod.
- Ensure that the stereo cameras are calibrated and synchronized properly.

Step 2: Capture Images

- Capture an image with the marker registered from a distance D.
- Translate the camera horizontally by T units.
- Capture another image with the marker registered after the translation.

Step 3: Compute Disparity Maps

- Use stereo correspondence algorithms (e.g., block matching, semi-global matching) to compute the disparity maps for both images.
- Ensure that the disparity maps are properly rectified and aligned.

Step 4: Extract Disparity Values for Marker

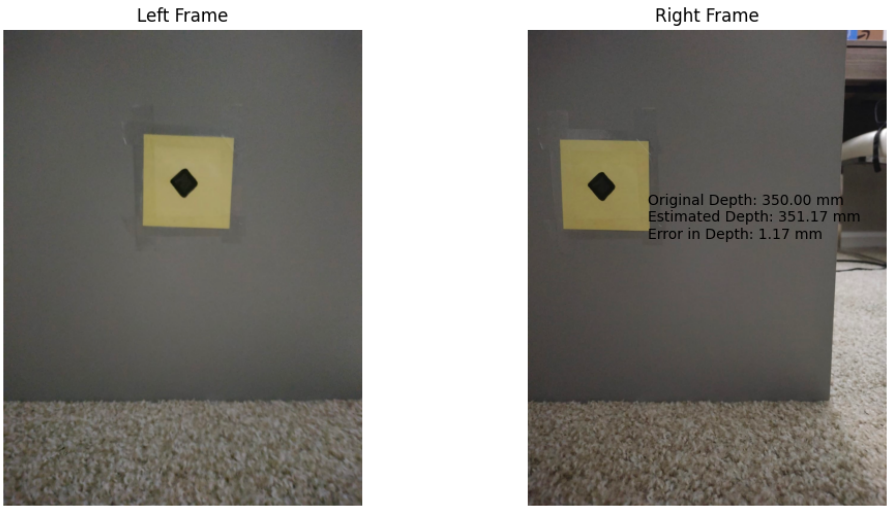
- Locate the region corresponding to the marker in both disparity maps.
- Extract the disparity values for the marker pixels in both maps.

Step 5: Compute Depth

- Compute the depth Z for each pixel using the formula: $Z=f\cdot T$/disparity

Step 6: Compute Distance

- Compute the distance D using the formula: $D=f\cdot TZ$
- The computed value of D represents the distance from the camera to the marker.



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

Step 1: Load the Video

1. Load the video file into MATLAB using the VideoReader function.

Step 2: Initialize Optical Flow Estimator

1. Choose an optical flow estimation method, such as Lucas-Kanade or Horn-Schunck.
2. Initialize an optical flow object using the chosen method.

Step 3: Compute Optical Flow

1. Loop through each frame of the video.
2. Compute the optical flow between the current frame and the reference frame.
3. Optionally, use different reference frames based on the specified criteria (every previous frame, every 11th frame, or every 31st frame).

Step 4: Visualize Optical Flow Vectors

1. Plot the optical flow vectors on each frame.
2. Optionally, overlay the optical flow vectors on the original frame for visualization.

Step 5: Display Results

1. Show or save the frames with the plotted optical flow vectors.

This question is solved using 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* . – This problem is solved using Python

Step 1: Load the Video and Select ROI

1. Load the video file.

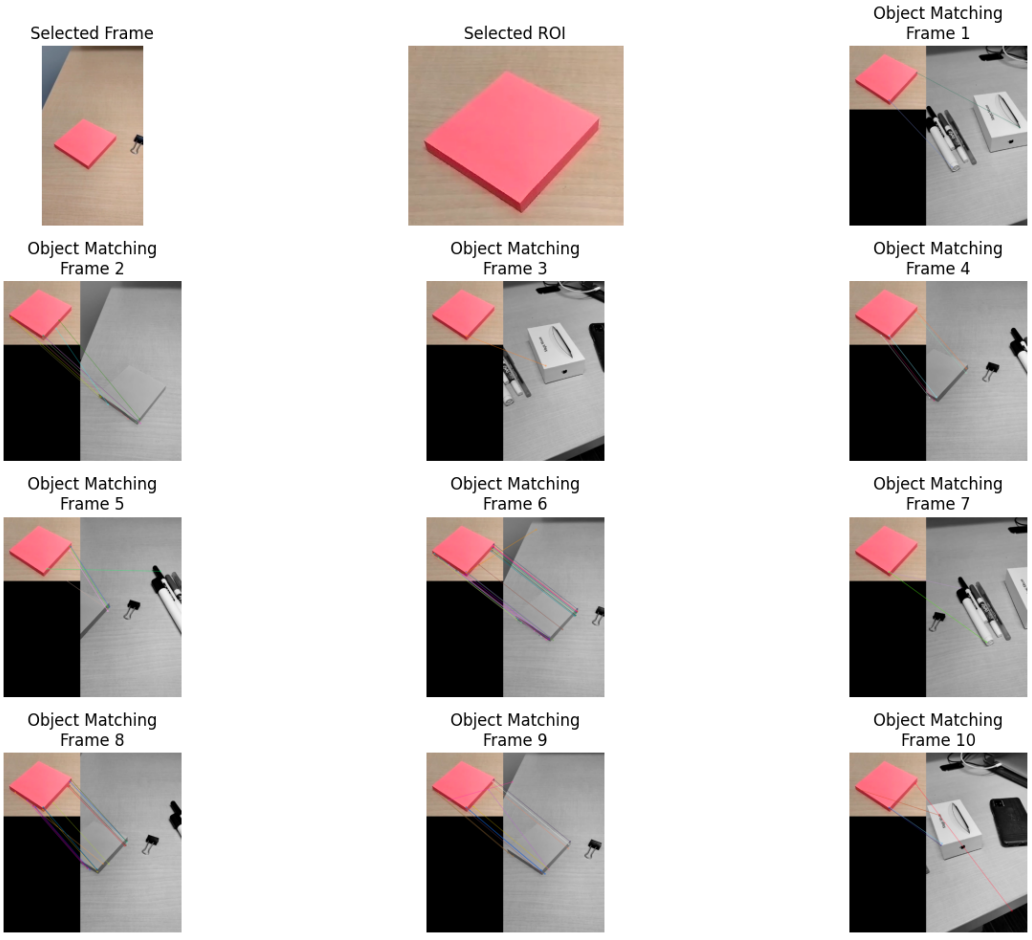
2. Randomly select a frame from the video.
3. Allow the user to select a region of interest (ROI) in the selected frame.

Step 2: Detect Objects Using SIFT

1. Initialize the SIFT detector.
2. Detect keypoints and compute descriptors for the selected ROI.
3. Loop through randomly chosen frames from the video.
4. Detect keypoints and compute descriptors for each frame.
5. Perform feature matching between the selected ROI and each frame.
6. Apply the ratio test to select good matches.
7. Calculate the sum of squared differences (SSD) for each match.

Step 3: Display Results

1. Display the selected frame and ROI.
2. Display the matched frames with the detected objects and average SSD value.



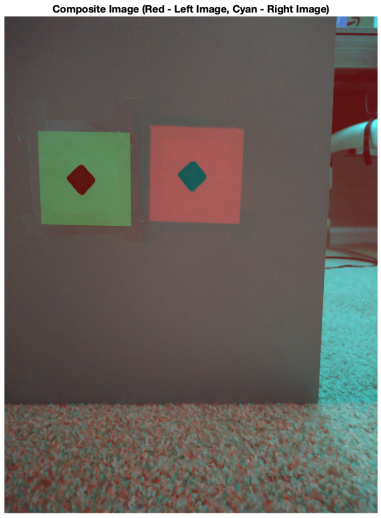
6. Refer to the Bag of Features example MATLAB source code provided in the classroom's classwork page. In your homework, pick an object category that would be commonly seen in any household (e.g. cutlery) and pick 5 object types (e.g. for cutlery pick spoon, fork, butter knife,

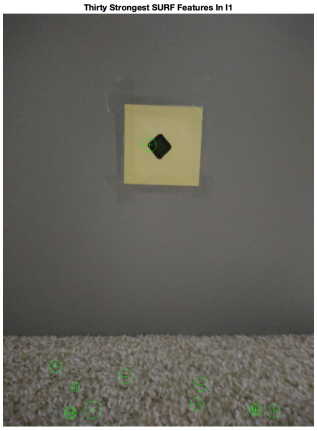
cutting knife, ladle). Present your performance evaluation.

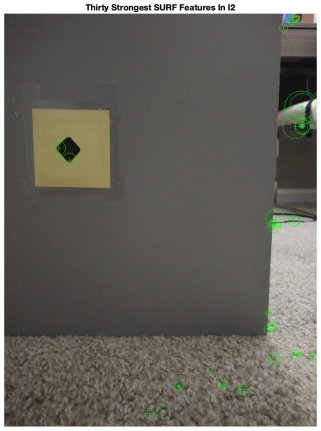
```
Encoding images using Bag-Of-Features.
-------------------------------------
* Encoding an image...done.

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
```

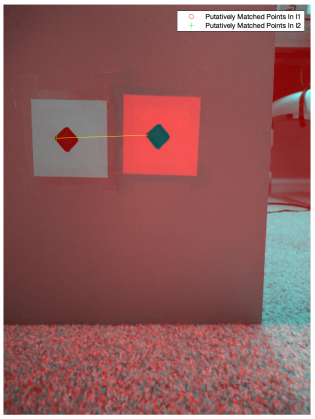
7. Repeat the image capture experiment from problem (3), however, now also rotate (along the ground plane) the camera 2 (right camera) towards camera 1 position, after translation by T. Make sure the marker is within view. Note down the rotation angle. Run the tutorial provided for uncalibrated stereo 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).*

Translated by 150 mm & rotation Angle is 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. Real-Time Object Tracking with QR Marker:

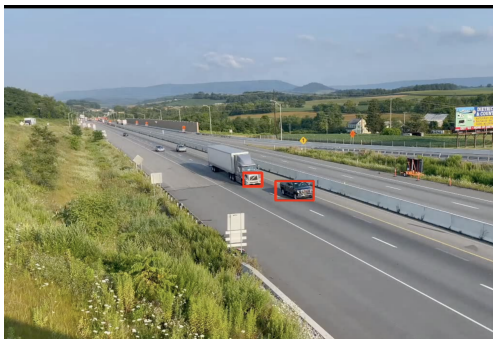
- Approach: This approach detects and tracks QR codes in real-time using the cv2.QRCodeDetector() class.
- Implementation:
  - The script captures video frames from the camera in real-time.
  - It detects QR codes in each frame using the detectAndDecodeMulti() function of the QRCodeDetector class.
  - If a QR code is detected, it draws a rectangle around it and displays the decoded information.
  - The process continues until the user exits the program by pressing the 'q' key.

2. Real-Time Object Tracking without Marker:

- Approach: This approach tracks objects without using any markers by detecting and updating the positions of objects based on their movement using the Euclidean distance.
- Implementation:
  - The script captures video frames from the camera in real-time.
  - It applies background subtraction to detect moving objects in the scene.
  - Detected objects are tracked based on their centroids using the Euclidean distance.
  - It draws bounding boxes around the detected objects and updates their positions.
  - The process continues until the user exits the program by pressing the 'Esc' key.

With QR Marker:

Without Marker:



Repo:
https://github.com/TejaCherukuri/CSC8830_ComputerVision/tree/main/Assignment-3