Faculty of Engineering
Systems & Biomedical Engineering Department

Computer Vision-Task 3

A Third Year Systems & Biomedical Engineering Task Report

Done by Team 5

Name	Section	BN
Ibrahim Mohamed	1	2
Omnia Sayed	1	14
Marina Nasser	2	12
Mahmoud Yaser	2	30
Maye Khaled	2	40

Submitted in Partial Fulfilment of the Requirements for
Computer Vision (SBE3230)

Submitted to:

Eng. Peter Emad

Eng. Laila Abbas

April 2023

A. Harris Operator for Corner Detection

- **Algorithm**

1. Take the grayscale of the original image.
2. Apply a Gaussian filter to smooth out any noise.
3. Apply Sobel operator to find the x and y gradient values for every pixel in the grayscale image.
4. For each pixel p in the grayscale image, consider a 3×3 window around it and compute the corner strength function. Call this its Harris value.
5. Find all pixels that exceed a certain threshold and are the local maxima within a certain window (to prevent redundant dupes of features)
6. For each pixel that meets the criteria in 5, compute a feature descriptor.

- **Output Sample**

B. Scale-Invariant Feature Transform

- **Algorithm**

  1. Load the image and store it in mat object.

  2. Detect keypoints: we detect the keypoints of the image and store them inside a vector, draw the detected keypoints on the image, and then display the image with the keypoints.

  3. Compute descriptors: we compute a descriptor for each keypoint; which is a vector that describes the local appearance of the image around each keypoint.

  4. Match keypoints: we match the keypoints by loading two images to compare, detect keypoints and compute descriptors for both images, match descriptors between the two images, sort the matches by their distance and select only the best matches, and finally draw the best matches on the images and display them.

- **Output Sample**

C. SSD and Normalized Cross Correlations

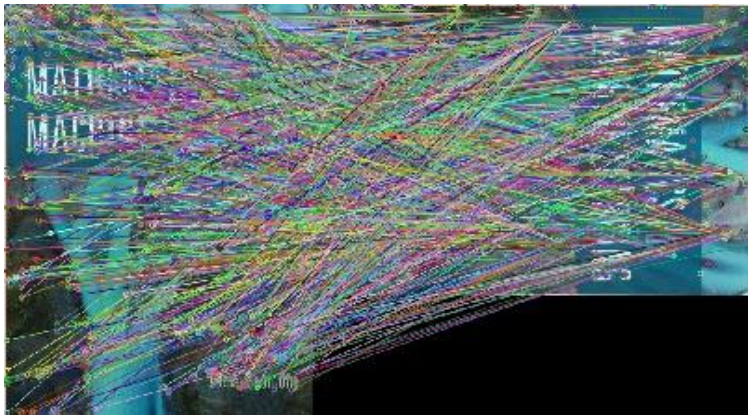
- **Algorithm**

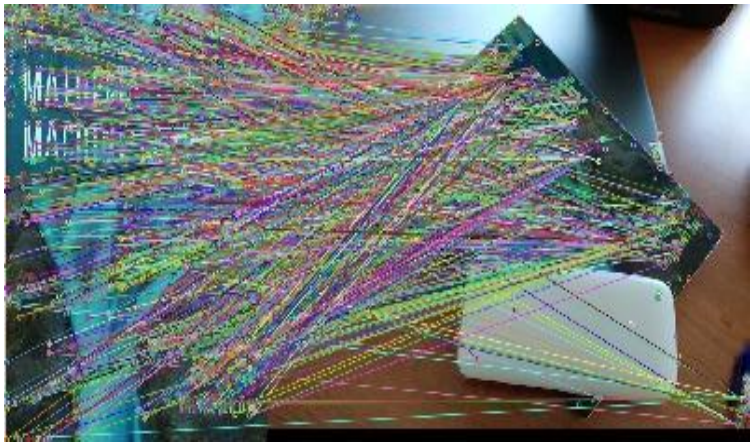
  1. This function matches two sets of feature vectors using SSD and NCC. The function takes as input two sets of feature vectors, each containing any number of feature vectors, and returns a list of indices indicating which feature vectors in the second set match each feature vector in the first set.

  2. The implementation uses two helper functions, ssd and ncc, to compute the distance between two feature vectors using SSD and NCC, respectively. The ssd function computes the sum of squared differences between two feature vectors, while the ncc function computes the normalized cross-correlation between two feature vectors.

  3. The main function, matchFeatures, takes two sets of feature vectors and a boolean indicating whether to use SSD or NCC for distance computation. It then computes the distances between all pairs of feature vectors using either SSD or NCC, and finds the best match for each feature vector in the first set by selecting the feature vector in the second set with the smallest distance.

- **Output Sample**





Due to the fact that the SSD method just uses addition and multiplication, it is often quicker than the NCC algorithm. The NCC approach, on the other hand, necessitates the computation of standard deviations as well as a division operation, which can be computationally demanding.

Here are 3 comparisons on 3 trials with SSD and NCC measured in nanoseconds.