

Task (3)

Computer Vision

(Features Detection)

Name	Section	Bn
Sama Mostafa	1	44
Misara ahmed	2	43
Yousr asharf	2	54
Rahma Abdelkader	1	31
Youssef essam	2	57

Doctor supervision:	Ahmed M. badawo
TA supervision:	Eng/laila abbas - Eng/peter salah

1 Harris Corner Detection:

1.1 Principle

The Harris corner detection algorithm is a popular algorithm for detecting corners in an image “where corners are points in an image where the intensity changes in more than one direction” and it works by:

- computing a "corner response" score for each pixel in the image, where the score is based on the change in intensity that occurs when the image is shifted by a small amount in any direction.

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \quad R = \det M - k(\text{trace } M)^2$$

1.2 The steps to match image sets

1. Compute the **gradient** of the image at each pixel.
2. Compute the sum of squared differences between the pixel intensities in a small window around each pixel, and the intensities in a shifted version of the window. The shift can be in any direction, and is typically small (e.g., 1 or 2 pixels).
3. Compute the corner response score for each pixel as the determinant of a matrix M (I_x and I_y are the x and y components of the image gradient).
4. The sums are taken over the window around each pixel. The weights are typically chosen to be a function of the window size and the noise level in the image.
5. Threshold the corner response scores R to keep only the strongest corners.

1.3 Application

Harris operators are useful in object recognition, Image stitching, tracking, and 3D reconstruction.

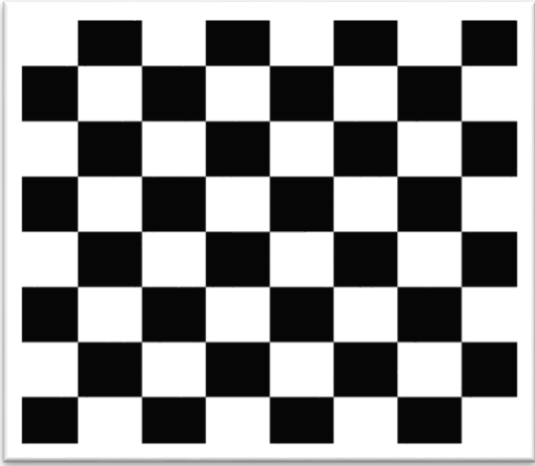
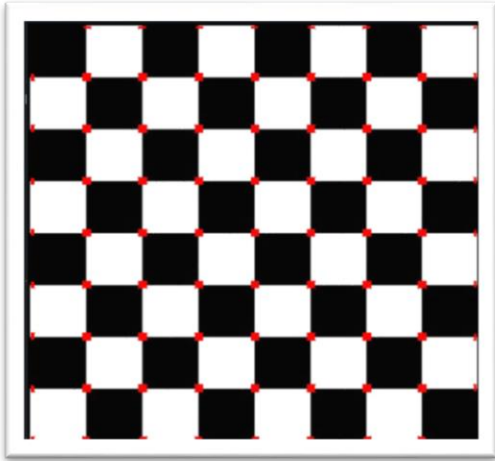
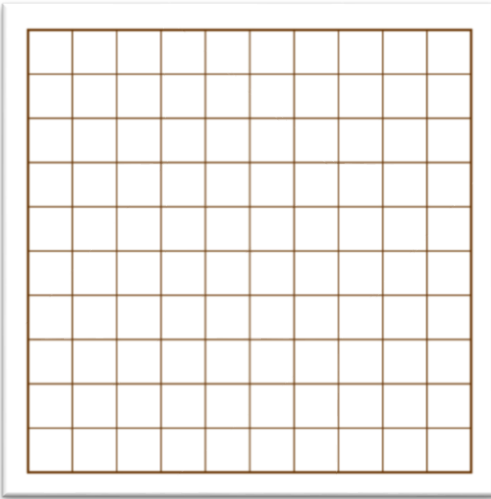
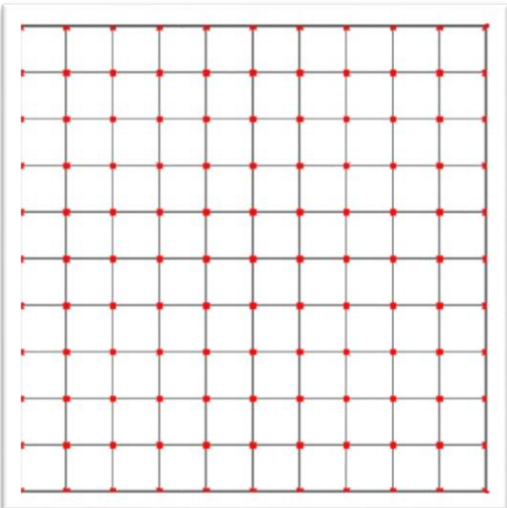
1.4 Algorithm

The function takes as input an image, a constant k which is window size, and a threshold value.

1. converts the image it to grayscale.
2. Computes the partial derivatives of the image in the x and y directions using the Sobel operator.
3. The implementation then computes the elements of the Harris matrix for each pixel in the image using a sliding window of size window_size.
4. The Harris matrix is a 2x2 matrix that describes the local structure of the image around each pixel. It is computed using the partial derivatives of the image and is used to detect corners in the image.

5. Checks if the Harris corner response is above the threshold value and marks the pixel as a corner in the output image if it is.
6. stores the location and Harris corner response of each detected corner in a vector called `corner_list`. This vector can be used for further processing, such as corner matching or feature extraction.

1.5 Example

Image	Operation
	
	

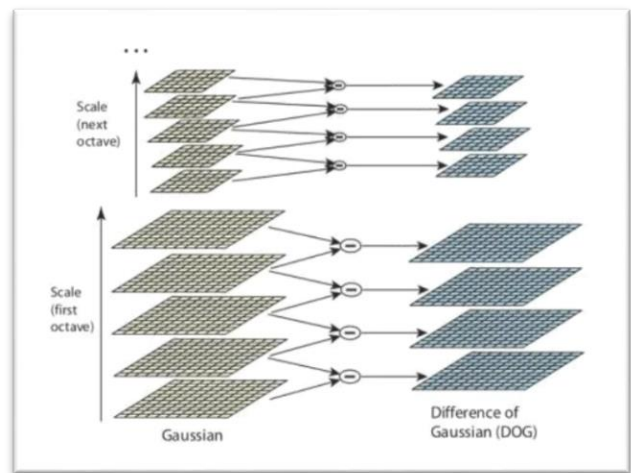
2 Scale Invariant Feature Transform:

2.1 What is Active contour:

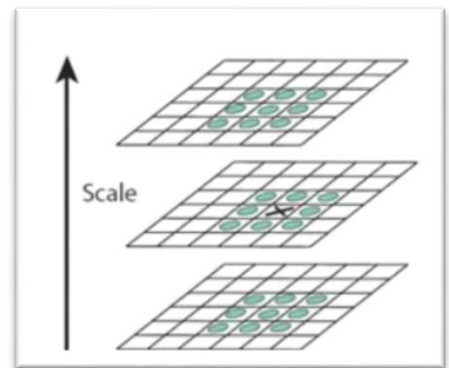
SIFT is a feature extraction method that reduces the image content to a set of points used to detect similar patterns in other images. This algorithm is usually related to computer vision applications, including image matching and object detection.

2.2 The steps

- **Scale-space peak selection:** ensure that we can identify the distinct points and ignore the noise through 'Gaussian Blur' technique then ensure that the features are not scale-dependent through using different scales of the original image, which is called **Building the Scale-Space**, then At each scale level, the image is analyzed to detect local extrema, which are points where the difference of Gaussian (DoG) function is maximum or minimum in both space and scale dimensions.



- **Keypoint Localization:** locate the key points in the image by comparing the pixel values with other pixels in their locality. So, each pixel value is now compared with all 8 neighboring pixels along with 9 pixels in the different scales as defined in the octaves. So, a total of 26 comparisons are made to check if a point is a local extremum, and only then is it classified as a potential key point.
- **Orientation Assignment:** to make the set of legitimate key points invariant to rotation, we must assign them a magnitude and orientation. The formula used



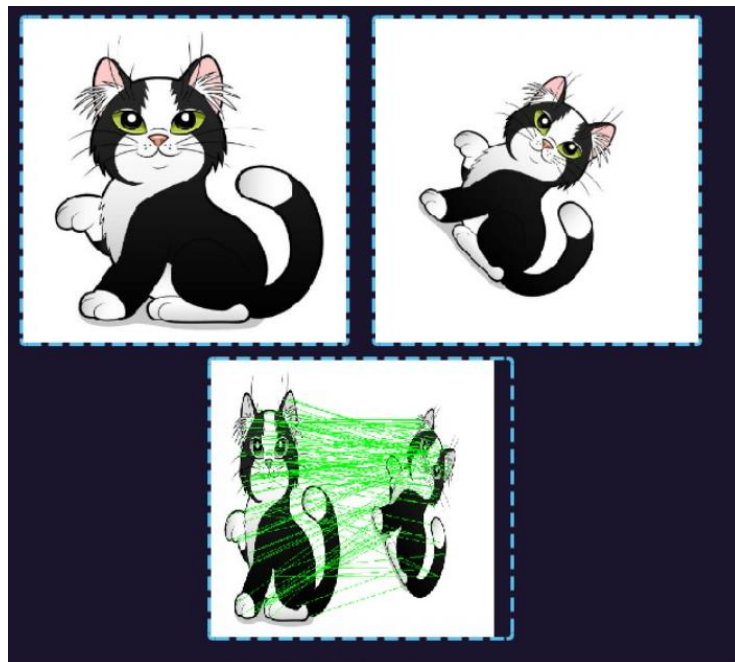
for calculating the magnitude and orientation is as follows:

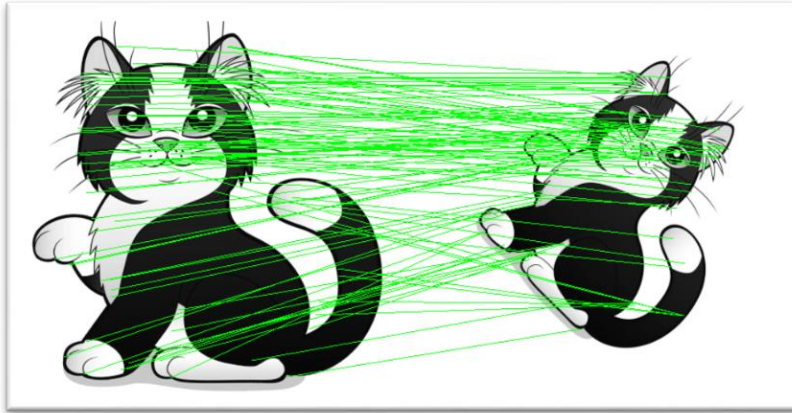
$$Magnitude = \sqrt{(G_x^2 + G_y^2)}$$

$$Orientation = \arctan(G_y/G_x)$$

- **Keypoint descriptor:** to compute a descriptor for the local image region about each keypoint that is highly distinctive and invariant as possible to variations such as changes in viewpoint and illumination.
- **Keypoint Matching:** keypoints between two images are matched by identifying their nearest neighbors. But in some cases, the second closest-match may be very near to the first. It may happen due to noise or some other reasons. In that case, the ratio of closest-distance to second-closest distance is taken. If it is greater than 0.8, they are rejected. It eliminates around 90% of false matches while discards only 5% correct matches, as per the paper.

2.3 Example





3 Matching Image Features

3.1 Sum of Squared Differences (SSD):

3.1.1 Principle

Matching image sets using sum of squared differences involves

- comparing the pixel values of each image in one set to the corresponding pixel values of each image in another set, and calculating the sum of the squared differences between the pixel values of the two images, where the set of images with the smallest sum of squared differences is considered to be the best match.

$$s = \sum_{(u,v) \in \mathbf{I}} (\mathbf{I}_1[u, v] - \mathbf{I}_2[u, v])^2$$

3.1.2 Steps

1. For each Image, compare it to each pixel in the second image using sum of squared differences.
2. The sum of squared differences between the two images by subtracting the pixel values of the first image from the pixel values of the second image, then square the result, and sum the squared differences across all pixels.
3. For each image in the first set, find the image in the second set with the smallest sum of squared differences, and note its index.
4. If there are any duplicate matches, choose the one with the smallest sum of squared differences.
5. The resulting set of matched images is the set of images from the second set identified from the first image.

3.1.3 Algorithm

1. First initializes a variable `sum_square` to 0.0, used to accumulate the sum of squared differences between the two descriptors.
2. Loop through each column of `desc_image2`, and calculates the squared difference between the corresponding elements of `desc_image1` and `desc_image2`, then adds it to `sum_square`.
3. Finally, take the square root of `sum_square`, and returns it as the SSD between the two descriptors.

3.1.4 Example



3.2 Normalized Cross Correlation(NCC):

3.2.1 Principle

Matching image set features using normalized cross-correlation involves

- finding the similarity between two images by comparing the pixel intensities at each location by sliding a template image over a reference image, then compute the cross-correlation between the two images at each position, where the position with the highest cross-correlation value corresponds to the best match between the two images.

$$\gamma = \frac{\sum_{x,y} (f(x,y) - \bar{f}_{u,v}) (t(x-u, y-v) - \bar{t})}{\sqrt{\sum_{x,y} (f(x,y) - \bar{f}_{u,v})^2 \sum_{x,y} (t(x-u, y-v) - \bar{t})^2}}$$

3.2.2 Steps

1. Compute the mean and standard deviation of the pixel intensities in the **reference** image.
2. Compute the mean and standard deviation of the pixel intensities in the **template** image, then subtract the mean of the reference image from the reference image and the template image.
3. Normalize the reference image and the template image by dividing them by their standard deviation.
4. Compute the cross-correlation between the normalized reference image and the normalized template image, then find the position with the highest cross-correlation value.

The NCC is a measure of the similarity between two feature vectors, and ranges from -1 to 1,

- 1 indicating perfect correlation, while 0 indicating no correlation, and -1 indicating perfect anti-correlation.

3.2.3 Algorithm

1. First computes the mean and standard deviation of desc_image1, then normalize desc_image1 by subtracting the mean and dividing by the standard deviation, and stores the result in norm_desc_image1.
2. Repeat the same process for desc_image2 and storing the result in norm_desc_image2.
3. Computes the element-wise product of the two normalized feature vectors, and stores the result in correlation_vector.
4. Finally, the function computes the mean of correlation_vector and returns it as the NCC between the two descriptors.

3.2.4 Example

