**Task (2)**

|  |  |  |
| --- | --- | --- |
| **Name** | **Section** | **Bench Number** |
| Amr Mohamed | 2 | 7 |
| Adham Mohamed | 1 | 9 |
| Belal Mohamed | 1 | 23 |
| Mahmoud Emad | 2 | 28 |
| Ezzat Hegazy | 1 | 56 |
| Mina Azer | 1 | 58 |

**Submitted To:**

**Prof. Dr. Ahmed Badawi**

**TA. Peter Emad**

**Systems and Biomedical Engineering Dept.**

**Cairo University**

**Corner Detection**: is an essential technique in computer vision used to identify distinct features or points in an image.

**The Harris Corner Detection**:

It is used to identify corners or key points in an image, and it is used for object recognition or image stitching.

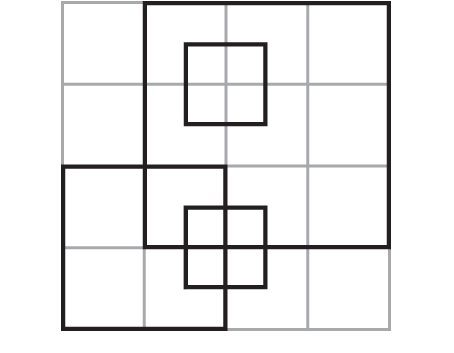
The steps involved in using the Harris Operator to detect corners in an image:

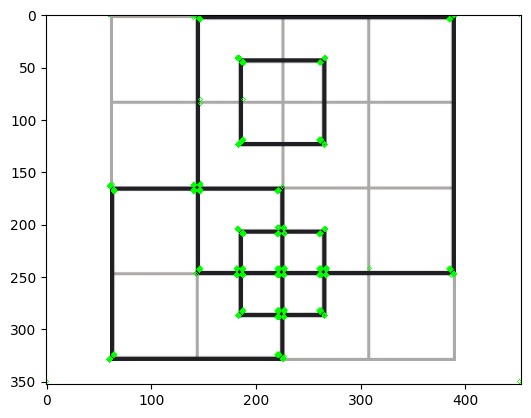
* The first step is to convert the input color image into grayscale.
* Compute Image Gradients The next step is to compute the image gradients. This can be done using a Sobel filter to calculate the x and y derivatives of each pixel in the grayscale image.
* Compute the Harris Response Function detect corners in the image. It is calculated using the following formula:

R = det(M) - k(trace(M)) ^2

where M is the second-moment matrix of image gradients, and k is an empirically determined constant. The eigenvalues of M are used to determine the degree of corner-like behavior at each pixel location.

* The next step is to perform non-maximum suppression on the Harris response function. This step removes all the points that are not local maxima in a given window size.
* Thresholding The final step is to apply a threshold to the Harris response function to eliminate any remaining points with low corner strength.

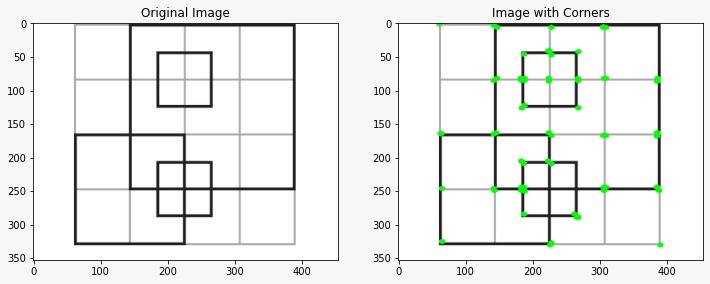




**The Lambda Operator**: It is a scale-invariant corner detection algorithm that aims to detect corners at different scales in an image. Here are the steps involved in the Lambda Operator:

* Image Grayscale Conversion Like the Harris Corner Detector
* The input image is convolved with a Gaussian filter at multiple scales to create a scale-space representation of the image. This step allows the algorithm to detect corners at different scales.
* Compute the Eigenvalues of the Structure Tensor The next step is to compute the eigenvalues of the structure tensor for each pixel in the scale-space image. The structure tensor is a matrix that describes the local image structure at a particular location in the image.
* The Lambda response function is calculated using the eigenvalues of the structure tensor. It measures the degree of corner-like behavior at each pixel location and is used to identify corners in the image.
* Non-Maximum Suppression and Thresholding The final step is to perform non-maximum suppression and thresholding to eliminate any remaining points with low corner strength.

**Note:** Harris Corner Detector is designed to detect corners at a fixed scale, while the Lambda Operator can detect corners at different scales. This makes the Lambda Operator more useful in applications that require the detection of corners at different scales.

**Top of Form**

**The computation time:** depends on several factors, such as the size of the input image, the processing power of the computer, and the implementation of the algorithms. In general, the Harris Corner Detector is faster and more efficient than the Lambda Operator due to its simpler algorithm and fewer computational steps.

The computation time for Harris= 3.672309160232544sec

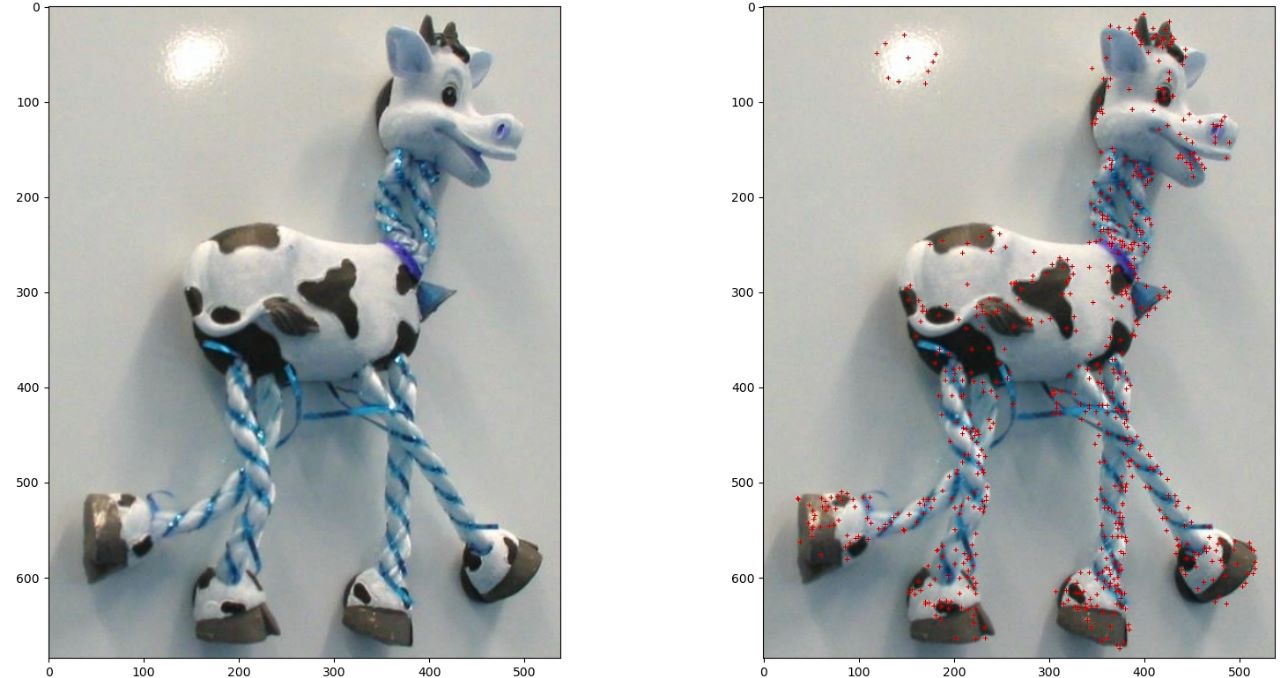
The computation time for lambda= 5.672309160232544sec

**The Scale-Invariant Feature Transform (SIFT):** is a widely used method for detecting and describing local features in an image. The algorithm is designed to be invariant to scale, rotation, and affine distortion. Here are the steps involved in generating feature descriptors using SIFT:

* The first step is to detect the local extrema in the image's scale space. This is achieved by convolving the image with a Gaussian filter at multiple scales and computing the difference of Gaussians (DoG) between adjacent scales. Local extrema are identified in the resulting scale space by comparing each pixel's value to its eight neighbors in the same scale and the corresponding pixels in the adjacent scales.
* Next, the detected extrema are refined using a technique known as key point localization. This involves fitting a quadratic function to the DoG scale space at each extremum and determining its location and scale based on the function's maximum or minimum value. Key points with low contrast or poor localization accuracy are discarded.
* Each key point is assigned one or more dominant orientations based on the gradient magnitudes and orientations of the image pixels around the key point. This is achieved by constructing a histogram of gradient orientations and selecting the orientation(s) with the highest count(s).
* Finally, a descriptor is generated for each key point using the gradient magnitudes and orientations of the pixels in the key point’s surrounding region. The descriptor consists of a vector of feature values that encode the local image structure around the key point. The descriptor is designed to be invariant to scale, rotation, and affine distortion, making it useful for matching features across images.
* The computation time for lambda= 103.48301197102sec

The SIFT algorithm is used to extract distinctive features from images then used to match corresponding points in different images.

**The sum square difference and normalized cross-correlation:** metrics are used to compare the feature descriptors and find the best match between the features in different images. Here is an algorithm for matching s:

* Extract SIFT features from both images in the image set.
* For each feature in the first image, compare it with all features in the second image using sum square difference (SSD) and normalized cross-correlation (NCC) metrics. For SSD, calculate the sum of the squared differences between the descriptors of the two features. For NCC, calculate the dot product of the two normalized feature descriptors.
* Store the SSD and NCC scores for each comparison.
* For each feature in the first image, select the best match in the second image by finding the feature with the lowest SSD and the highest NCC score.
* Apply a threshold to the SSD and NCC scores to filter out poor matches.
* Store the matched features and their corresponding SSD and NCC scores.