**German University in Cairo**

**Faculty of Digital Media Engineering and Technology**

**Spring 2017**

**Lab 5 + Mini Project II**

**Scale Invariant Feature Transform (SIFT)**

|  |  |
| --- | --- |
| **Experiment** | **Lab 5 + Mini Project II** |
| **Submitter** | **Amr Abu Greedah** |
| **Submitter ID** | **25-2284** |
| **Tutorial** | **T-17** |
| **Submission Date** | **4-8-2017** |

1. **SIFT Function**

**Input:** image, maximumSigma and threshold.

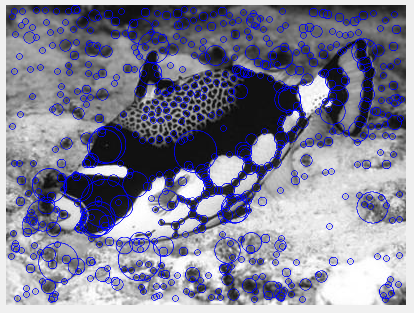
**Output:** sigmaAndKeyPoints (a matrix contains key points and their sigma).

**Steps:**

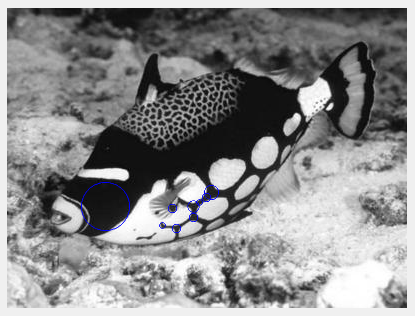
1. We save the filtered images in array called filtImgArray.
2. Then check the pixels of the 2nd to maximumSigma-1 layers and check their pixels if they are local maximum and greater than threshold we take them as key points and the current sigma in sigmaAndKeyPoints which is the output of the Sift function.
3. We use the sigmaAndKeyPoints to draw blobs on the input image.

**Output blobbed images**

Maximum sigma 20 and threshold 0



Maximum sigma 20 and threshold 0.3



1. **Get\_Descriptor Function**

**Input:** image and sigmaAndKeyPoints (output of the **SIFT** **Function**).

**Output:** descPartOfKeyPoints and positionOfKeyPoints.

**Steps:**

1. First double for loops compute number of keypoints in sigmaAndKeyPoints mat and saves it in **counter**.
2. descPartOfKeyPoints of size (**counter**, 128) is initialized with zero values.
3. Second double for loops compute the descPartOfKeyPoints output.
4. Current sigma is saved, then the condition checks if it’s not equal to zero.
5. matrixGVGH which is the 16x16 mat contains the tan-1(GV/GH) values is initialized.
6. filtImg contains the filtered image.
7. windowGV and windowGH contain the 16x16 around the indices of the current keypoint in each of them.
8. The next double for loops with variables Gi and Gj are to compute the matrixGVGH mat.
9. The final double for loops with variables gResX and gResY compute the histc which gives as an output bincounts (1, 8).
10. The next for loop with variable binJ saves the bincounts mat in descPartOfKeyPoints.
11. The current indices of the keypoint which are i and j are saved in positionOfKeyPoints.
12. **Snitch Function**

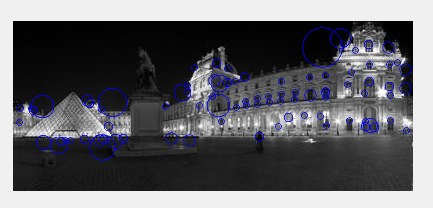
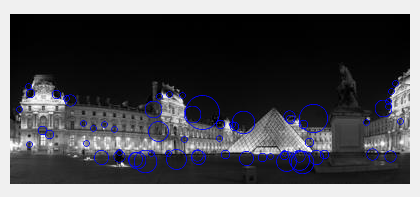
**Input:** Img1, descPartOfKeyPointsImg1, positionOfKeyPointsImg1, Img2, descPartOfKeyPointsImg2, positionOfKeyPointsImg2, threshold.

**Output:** shows the stitched image.

**Steps:**

1. Euclidean distance is calculated for each cell in the descPartOfKeyPointsImg1 mat and all the cells in the descPartOfKeyPointsImg2 mat.
2. If the current calculated Euclidean distance is less than threshold then it’s saved in arrayRes mat.
3. That minimum Euclidean distance in the arrayRes mat is retrieved with it’s position.
4. Then the position of the two key points is saved in sameKeyPoints mat.
5. A simple operation is done on the two images to stitch them.
6. outputImg contains the stitched image ;)
7. Done! :D
8. Almost forgot! Use mainCaller.m file to stitch the images, Louvre1 is Img1 and Louvre2 in Img2. Threshold is 1000.

**Blobbed Louvers:**



**Stitched Louvers:**

