Assignment 2:

Q1(a) and (b).

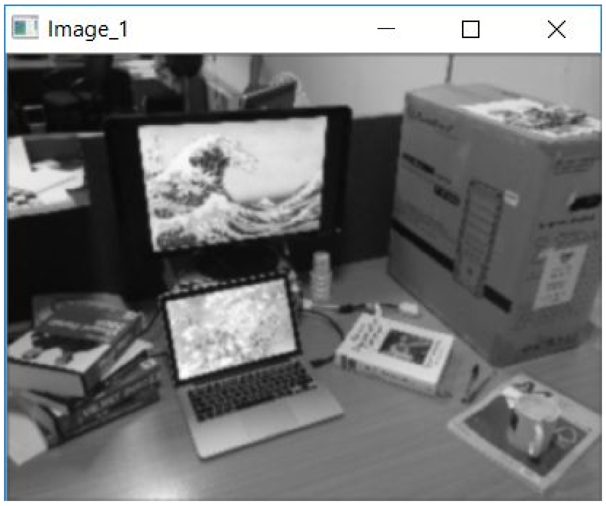
Methodology:

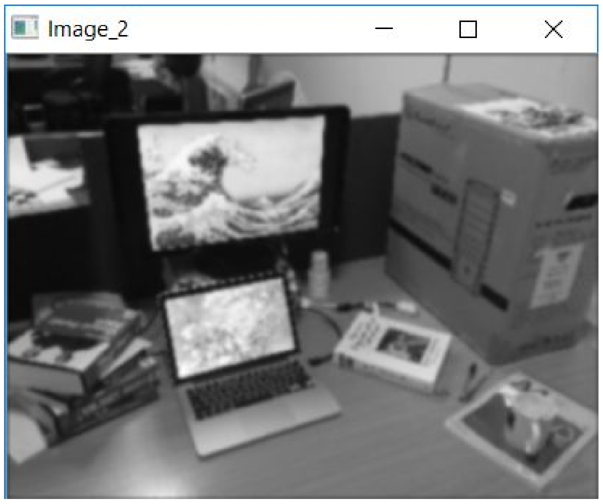
In this implementation the original image is convolved with a Gaussian Filter to obtain the Gaussian Blurred Images. The size of the Gaussian Kernel for this part of the implementation is fixed at 13. The images are then downsampled and then again convolved with the kernel of size 13. This process is repeated for 4 octaves. Note that the key points displayed in the first and fourth gaussian have been truncated; only 1 in 15 keypoints are being displayed

Sample Output:

The sample output is shown below. We have obtained the Gaussian and DOG scale space images. Note that each octave has been shown together to enable effective comparison between each of the images in each octave.

Octave 1:Gaussian Images





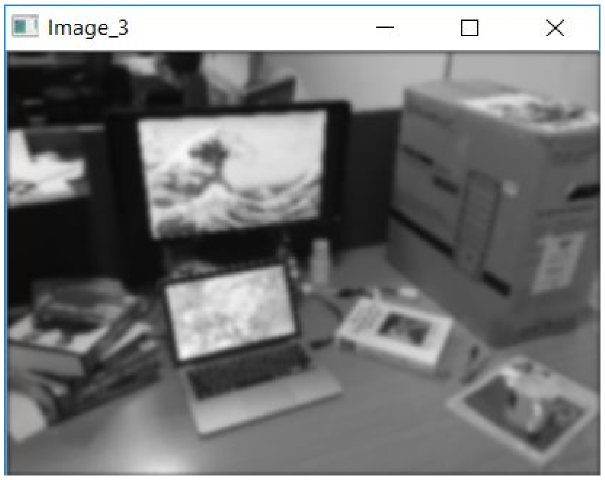


Image_3

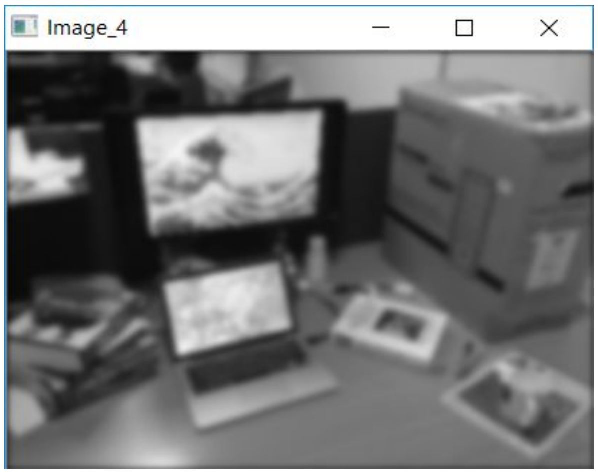


Image_4

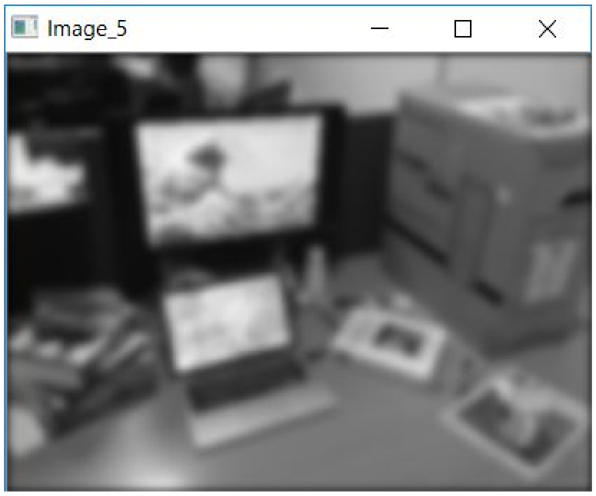
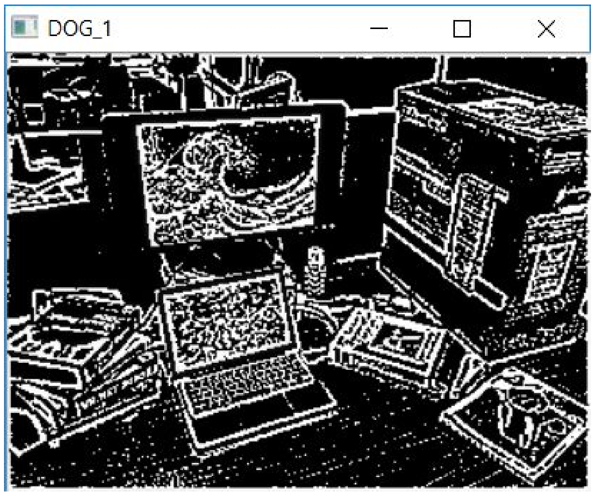
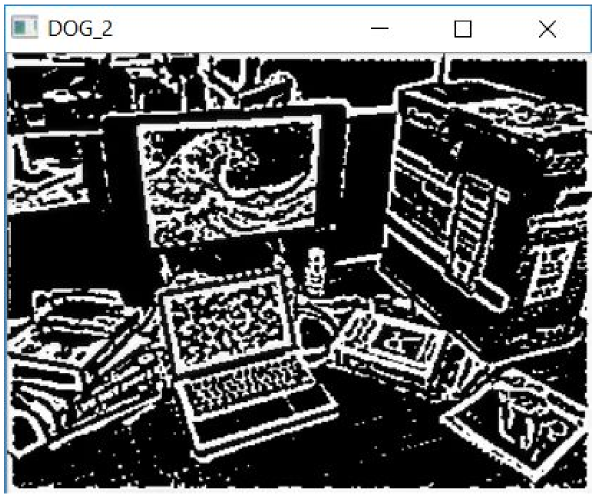
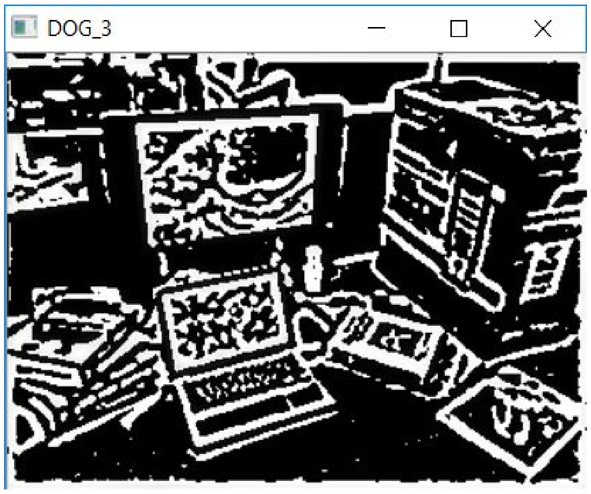


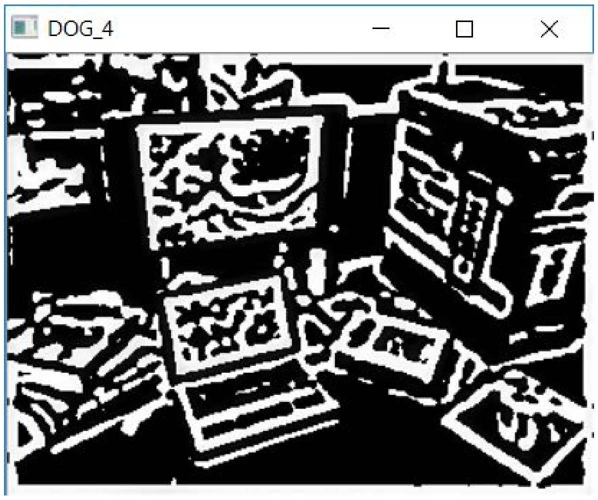
Image_5

Octave 1: DOG Images:



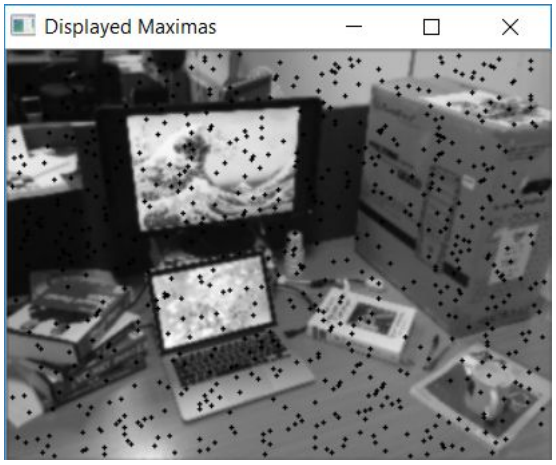




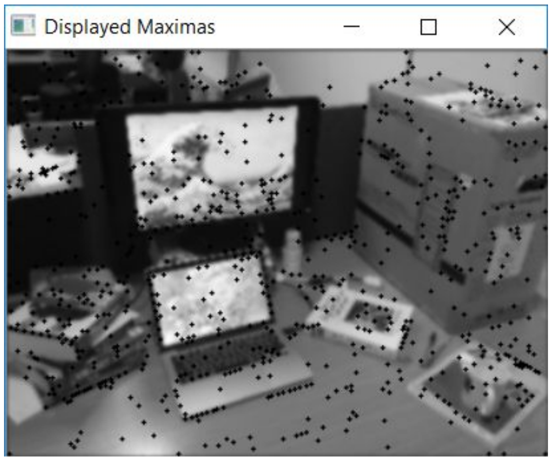


Octave 1: Detected Keypoints:

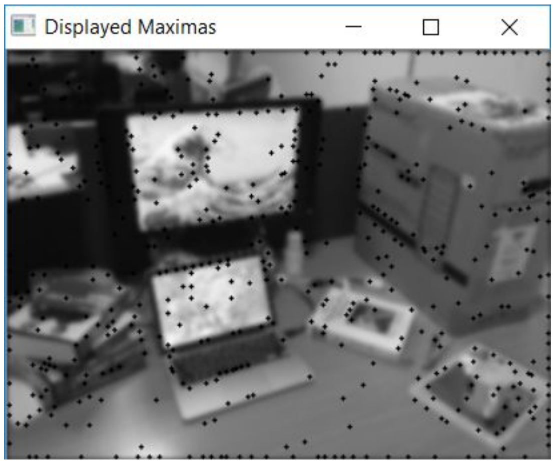
Gaussian 1:



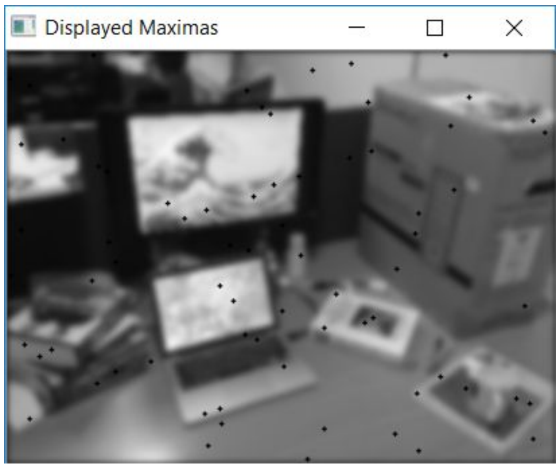
Gaussian 2:



Gaussian 3:



Gaussian 4:



Octave 2: Gaussian Images:

Image 1:

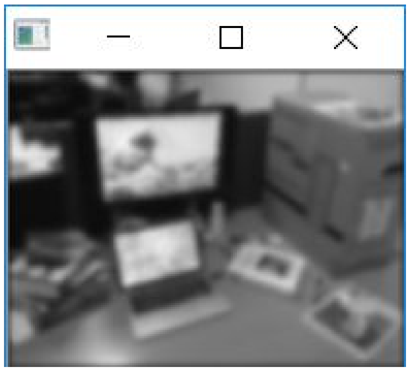


Image 2:

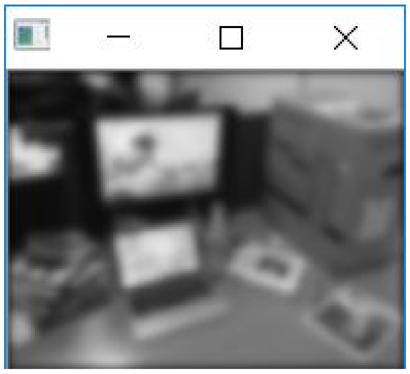


Image 3:

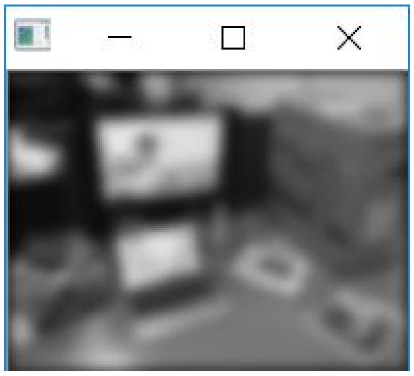


Image 4:

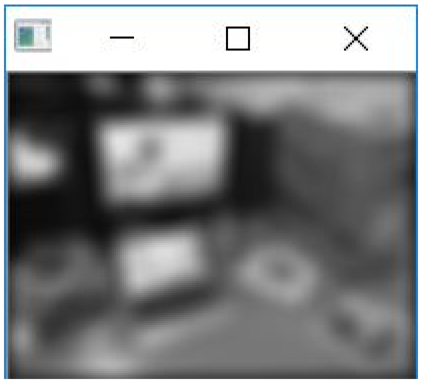
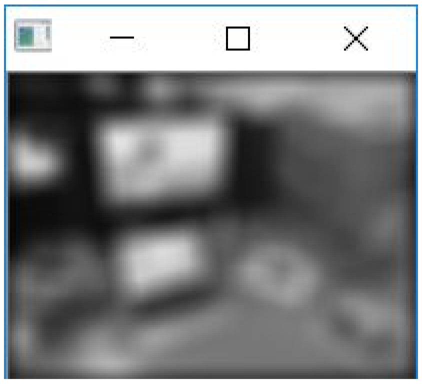
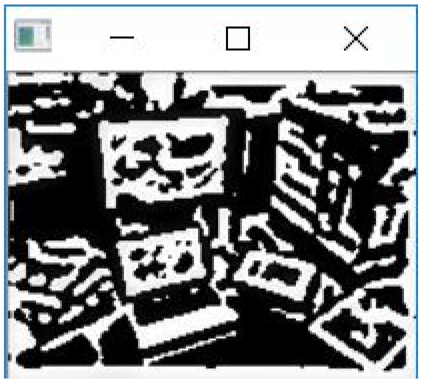


Image 5:

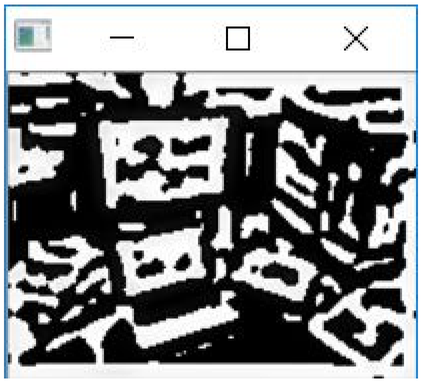


Octave 2: DOG Images:

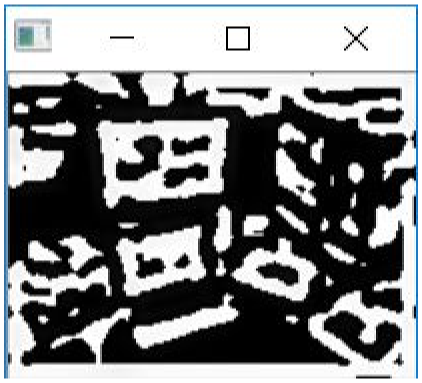
DOG1:



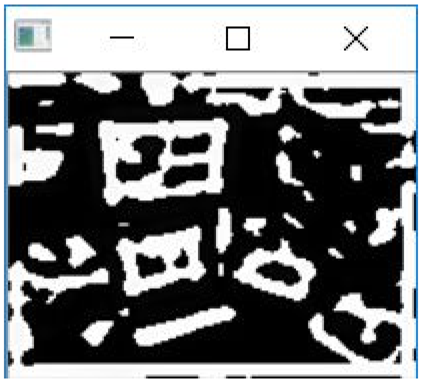
DOG2:



DOG3:

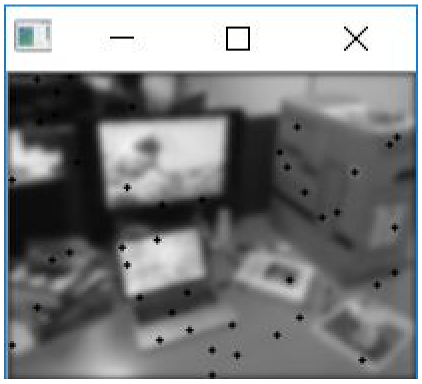


DOG4:

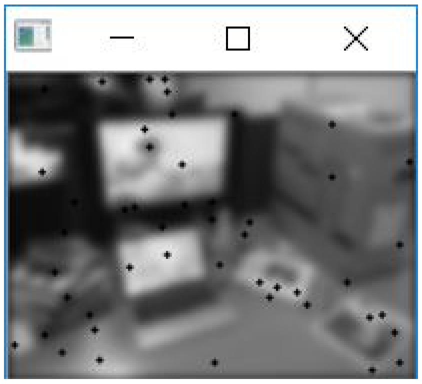


Octave 2: Detected Maximas:

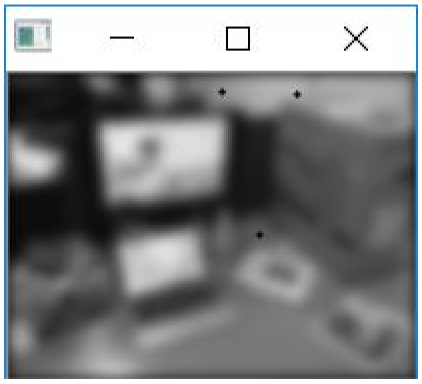
Gaussian 1:



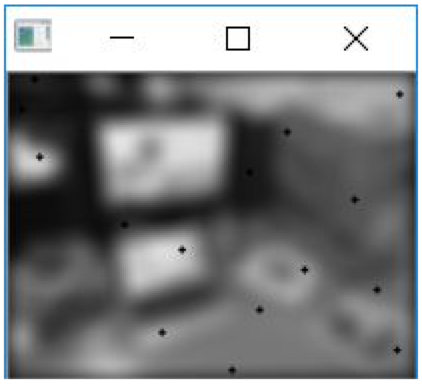
Gaussian 2:



Gaussian 3:

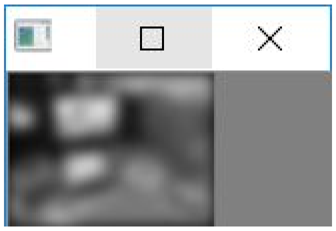


Gaussian 4:



Octave 3: Gaussian Images:

Gaussian 1:



Gaussian 2:

Gaussian 3:

Gaussian 4:

Gaussian 5:

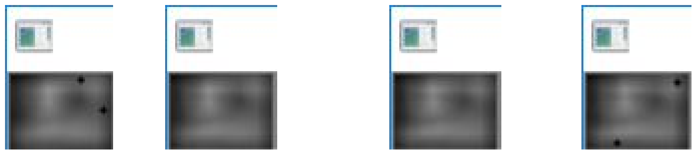
Octave 3: DOG Images

Octave 3: Detected Maximas:

Octave 4: Gaussian Images

Octave 4: DOG Images

Detected Keypoints



Q1(c) Orientation Assignment of Keypoints:

Methodology:

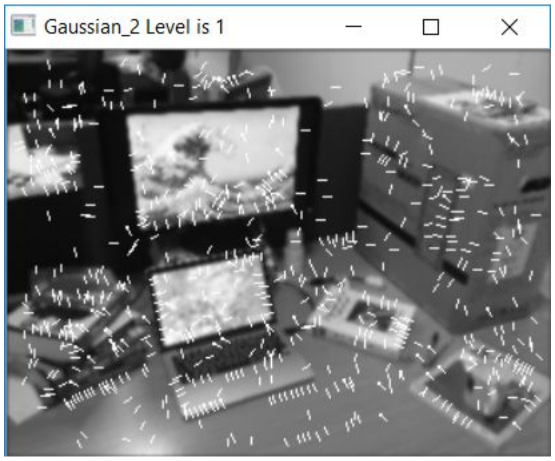
Orientation assignment is carried out for each octave Gaussian Scale Space .The orientations are assigned locally on each of the images in each Gaussian. The orientations are combined and displayed on the original image.

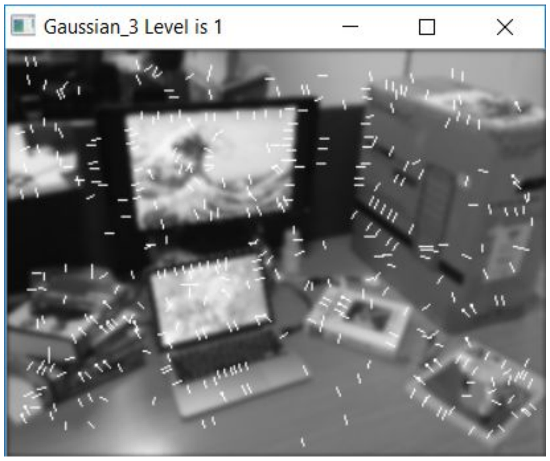
Sample Output:

Level indicates the pyramid level in which the key points were detected. . We are assigning orientations only to specific key points; the ones that are detected in the 2nd and 3rd Gaussian in each scale space. There were no key points detected in the 2nd and 3rd Gaussian of the third and the fourth octave.

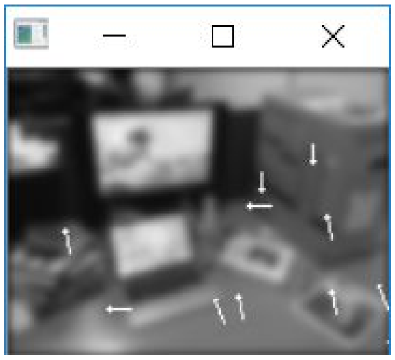
1. **Orientation in each octave:**

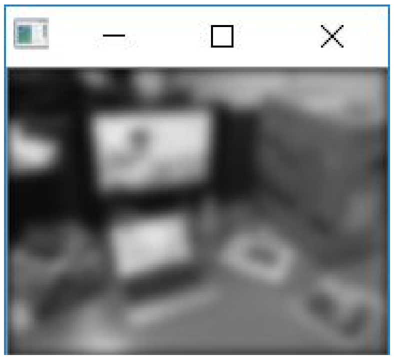
Octave 1:





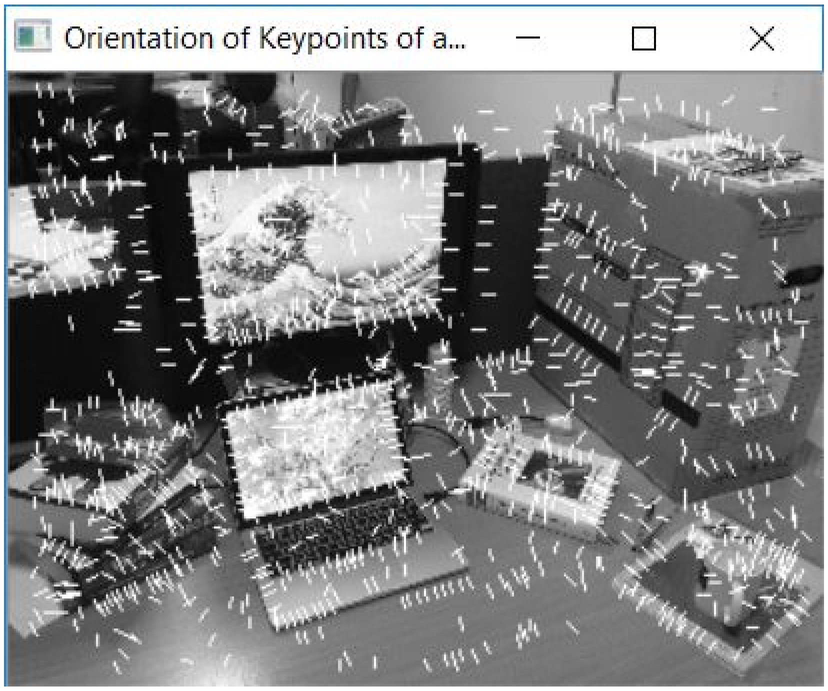
Octave 2:





2. Orientation on Original Image:

Combining all the local orientations and displaying on the original image



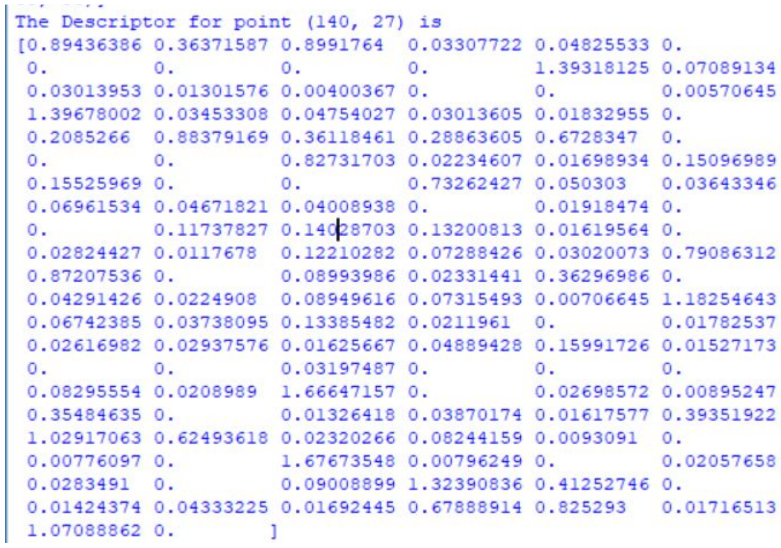
Q1(d) Creating the Descriptor for the Keypoints:

Methodology:

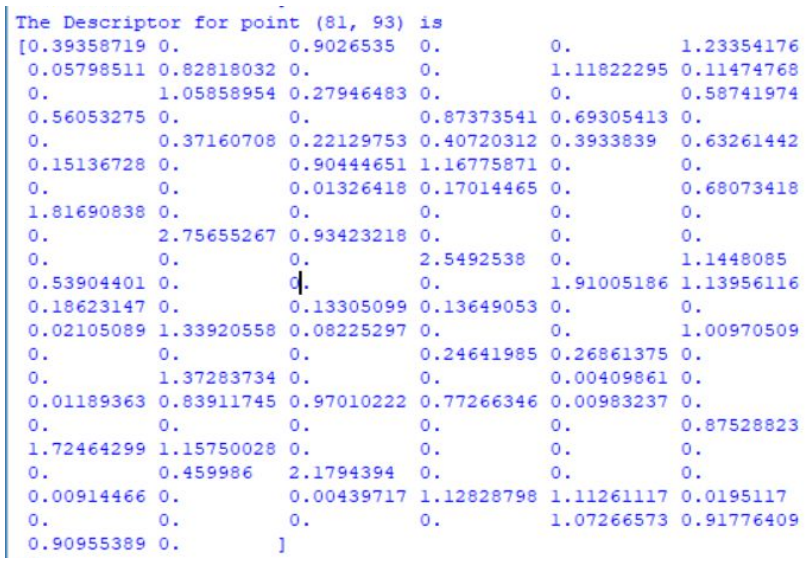
The descriptor is created for each keypoint detected in all the octaves. Note that the descriptor is being defined in each of the local gaussian images according to which key points belong to that gaussian. The lower octave images are being upsampled into the original by multiplying their coordinates by 2^{i-1} where i is the octave number. $i \in (1,2,3,4)$. If one keypoint is being assigned multiple descriptors then the first one is retained.

Sample Output:

The sample output has been shown for one particular keypoint (140,27) and (81,93) has been detected.



```
The Descriptor for point (140, 27) is
[0.89436386 0.36371587 0.8991764  0.03307722 0.04825533 0.
 0.         0.         0.         0.         1.39318125 0.07089134
 0.03013953 0.01301576 0.00400367 0.         0.         0.00570645
 1.39678002 0.03453308 0.04754027 0.03013605 0.01832955 0.
 0.2085266  0.88379169 0.36118461 0.28863605 0.6728347  0.
 0.         0.         0.82731703 0.02234607 0.01698934 0.15096989
 0.15525969 0.         0.         0.73262427 0.050303   0.03643346
 0.06961534 0.04671821 0.04008938 0.         0.01918474 0.
 0.         0.11737827 0.14028703 0.13200813 0.01619564 0.
 0.02824427 0.0117678  0.12210282 0.07288426 0.03020073 0.79086312
 0.87207536 0.         0.08993986 0.02331441 0.36296986 0.
 0.04291426 0.0224908  0.08949616 0.07315493 0.00706645 1.18254643
 0.06742385 0.03738095 0.13385482 0.0211961  0.         0.01782537
 0.02616982 0.02937576 0.01625667 0.04889428 0.15991726 0.01527173
 0.         0.         0.03197487 0.         0.         0.
 0.08295554 0.0208989  1.66647157 0.         0.02698572 0.00895247
 0.35484635 0.         0.01326418 0.03870174 0.01617577 0.39351922
 1.02917063 0.62493618 0.02320266 0.08244159 0.0093091  0.
 0.00776097 0.         1.67673548 0.00796249 0.         0.02057658
 0.0283491  0.         0.09008899 1.32390836 0.41252746 0.
 0.01424374 0.04333225 0.01692445 0.67888914 0.825293   0.01716513
 1.07088862 0.        ]
```



```
The Descriptor for point (81, 93) is
[0.39358719 0.         0.9026535  0.         0.         1.23354176
 0.05798511 0.82818032 0.         0.         1.11822295 0.11474768
 0.         1.05858954 0.27946483 0.         0.         0.58741974
 0.56053275 0.         0.         0.87373541 0.69305413 0.
 0.         0.37160708 0.22129753 0.40720312 0.3933839  0.63261442
 0.15136728 0.         0.90444651 1.16775871 0.         0.
 0.         0.         0.01326418 0.17014465 0.         0.68073418
 1.81690838 0.         0.         0.         0.         0.
 0.         2.75655267 0.93423218 0.         0.         0.
 0.         0.         0.         2.5492538  0.         1.1448085
 0.53904401 0.         0.         0.         1.91005186 1.13956116
 0.18623147 0.         0.13305099 0.13649053 0.         0.
 0.02105089 1.33920558 0.08225297 0.         0.         1.00970509
 0.         0.         0.         0.24641985 0.26861375 0.
 0.         1.37283734 0.         0.         0.00409861 0.
 0.01189363 0.83911745 0.97010222 0.77266346 0.00983237 0.
 0.         0.         0.         0.         0.         0.87528823
 1.72464299 1.15750028 0.         0.         0.         0.
 0.         0.459986   2.1794394  0.         0.         0.
 0.00914466 0.         0.00439717 1.12828798 1.11261117 0.0195117
 0.         0.         0.         0.         1.07266573 0.91776409
 0.90955389 0.        ]
```

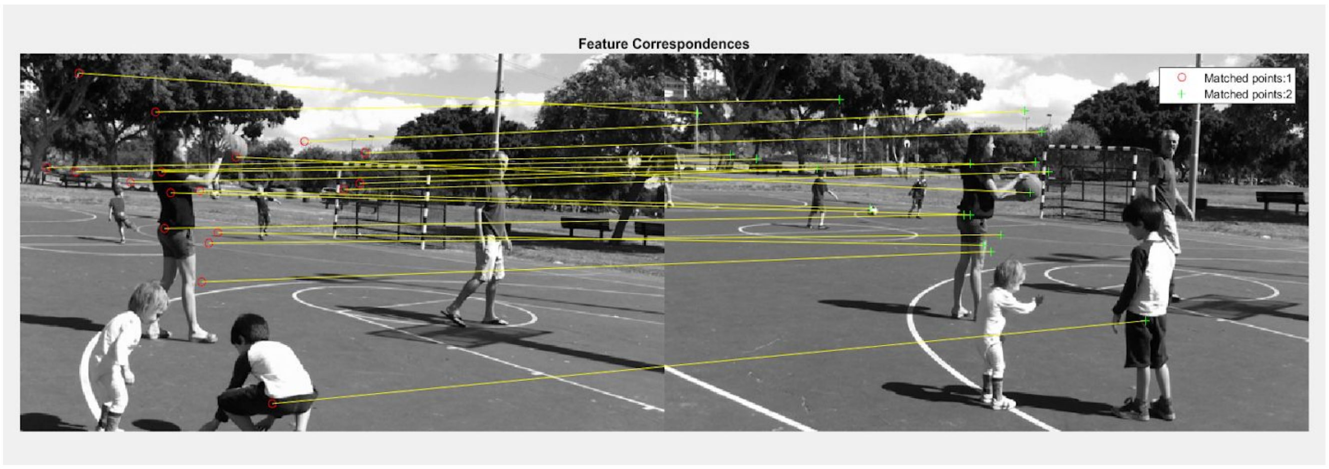
Q1(e): Matching Two Images.

For the results in matching it was observed that a kernel size of 6σ gave the best results so that size has been used to compute the DOG for this implementation.
An alternative function PyramidFast has been provided for faster implementation.
The image size is fixed to (320,640) in this part.

MATLAB has been used to plot feature correspondences. The feature correspondences are detected in Python. The obtained result is used in MATLAB to plot the feature correspondences.

Methodology: We compare the key points detected in two images and using minimum norm approach match those keypoints whose descriptors are the closest in value. The display is done by using MATLAB for the detected points.

Matching Results:



Note: A filter function using numpy dot has been created as the ordinary function takes time to execute. Please make the corresponding change in the code for faster speed. It is only an alternative and not the actual implementation.