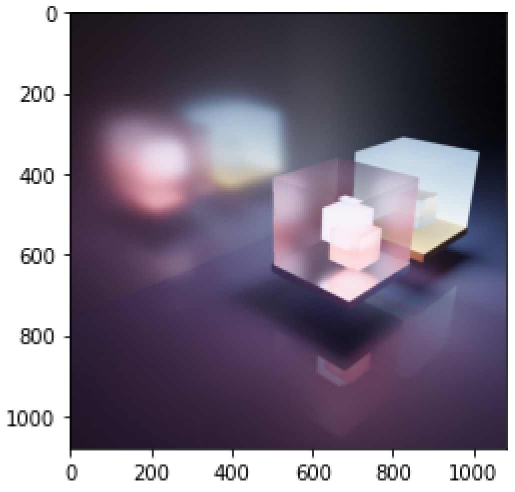
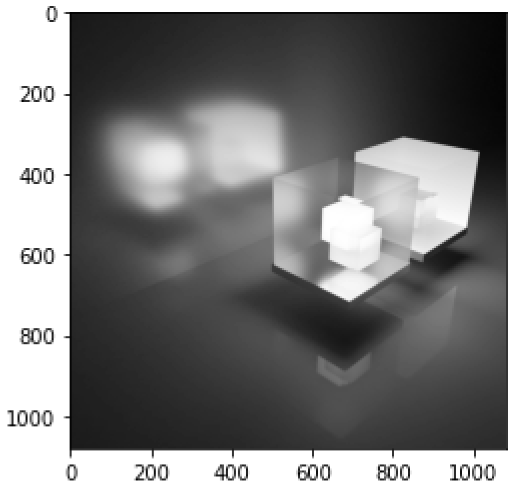
P2 Assignment Write Up
Computer Vision CS682
Mithilaesh Jayakumar (G01206238)

2.1 Harris Corner Detection

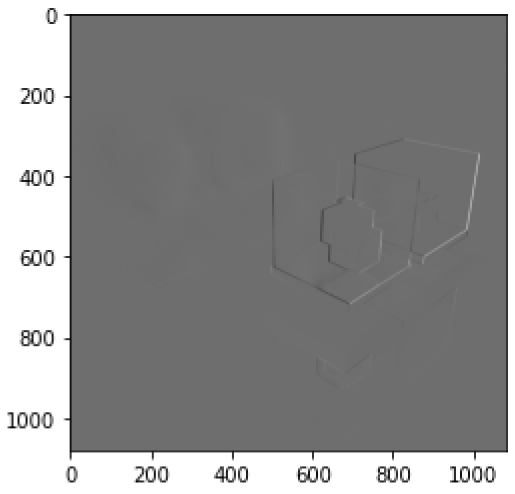
The original image which we are using for our Harris Corner detection is the one shown below.

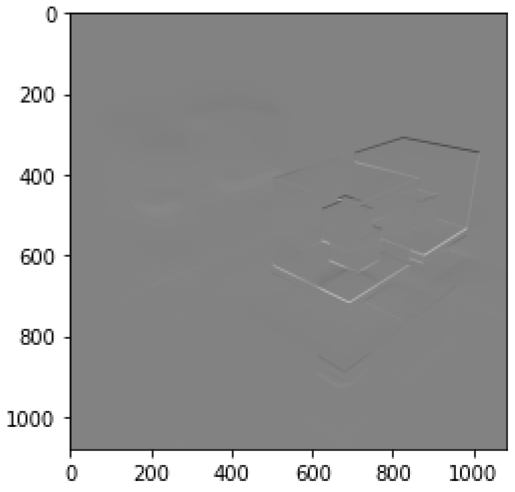


In order to make the computations faster we are converting this image into a grayscale image as shown below.



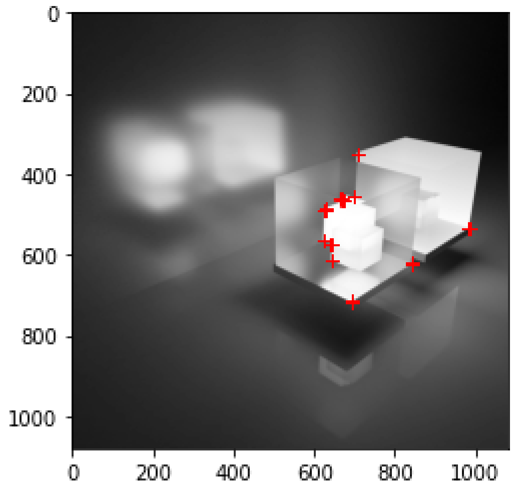
Now using the Sobel filter we are computing the image derivatives. The First one shows the x derivatives of our image and the second one shows the y derivatives of our image.



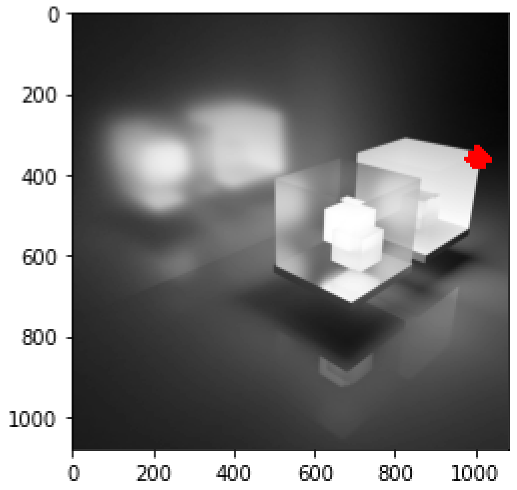


For the computation of the Harris corner we are going to use different weighted matrices of different sizes. Here we need to select an optimal threshold value so that the output image has considerable corner features marked. Also the cost of computation also depends on the threshold value. The cost of computation as well as the number of detected features both increase as the threshold value decreases. So we have varied the threshold value for each case of the weighted matrix so that a considerable amount of features are shown in the output image.

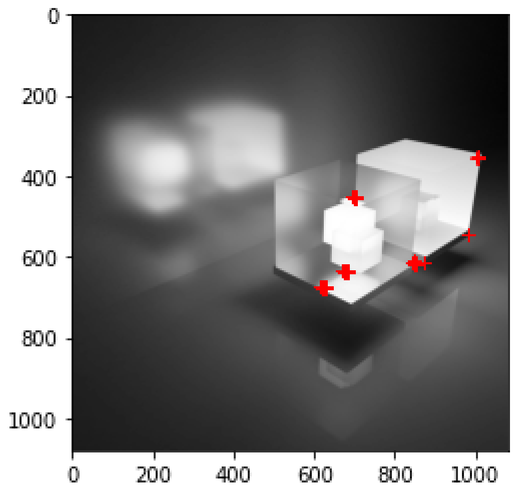
For the (1) 5x5 weighted matrix the resulting f value is 36 (approx) and selecting a threshold value of 5 results in the following output. We could see a few corners were not considered but still a good amount of corners were identified.



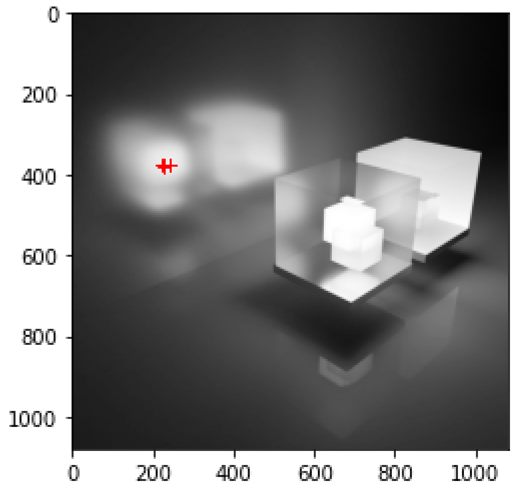
For the (2) 25x25 weighted matrix the resulting f value is 3543 (approx) and selecting a threshold value of 1300 results in the following output. We could see that all the corners identified were present at the same point of the image.



For the (3) gaussian weighted matrix of sigma = 5 the resulting f value is 13 (approx) and selecting a threshold value of 0.5 results in the following output.We could see that only a few corners were identified repeatedly.

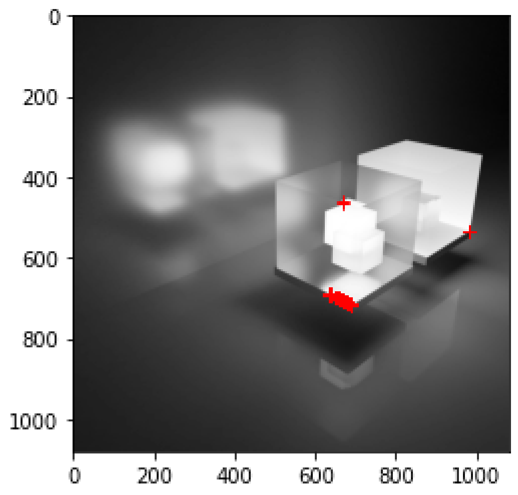


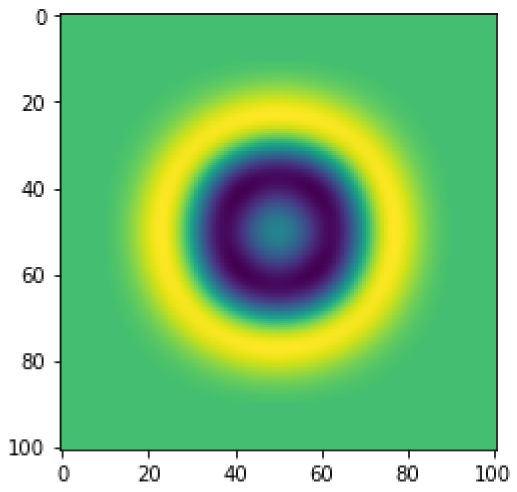
For the (4) gaussian weighted matrix of sigma = 50 the resulting f value is 0.0013 (approx) and selecting a threshold value of 0.5 results in the following output.We could see that the corners were not identified properly.



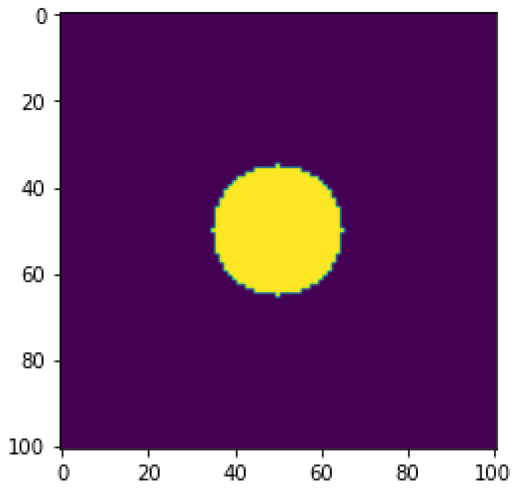
When we use the weighted matrix of size 1x1 the resulting scoring function becomes 0.

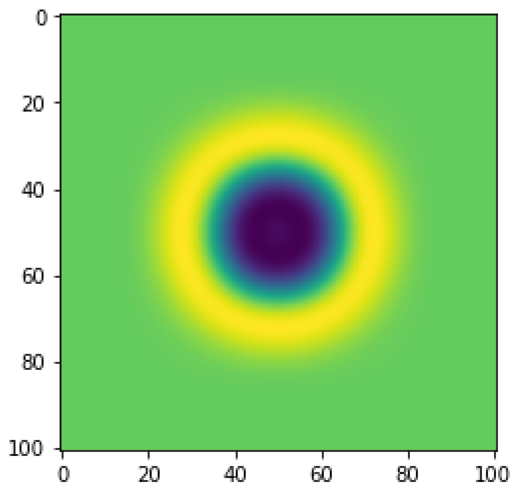
Now we are going to modify the scoring function as f = A + C. The resulting scoring function value is 13.52 and the generated output image has features marked on the edges instead of the corners for threshold value of 11. This scoring function identifies the edges instead of the corners.





For radius = 15,



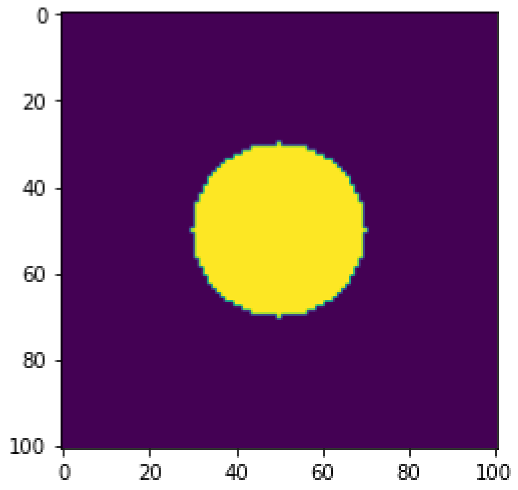


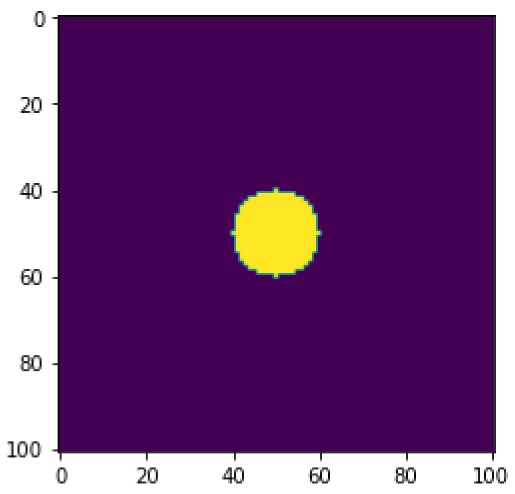
For radius = 10,

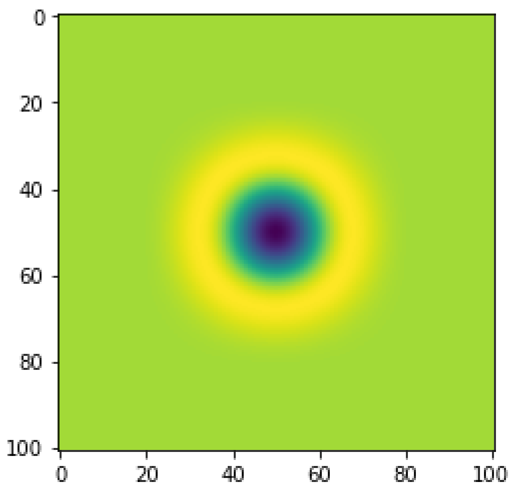
2.2 Multiscale Blob Detection

We have implemented a normalized LoG filter and we apply that filter to our circle image. We used a sigma value of 7 and a kernel size of 42 for this LoG filter. The below are the output images.

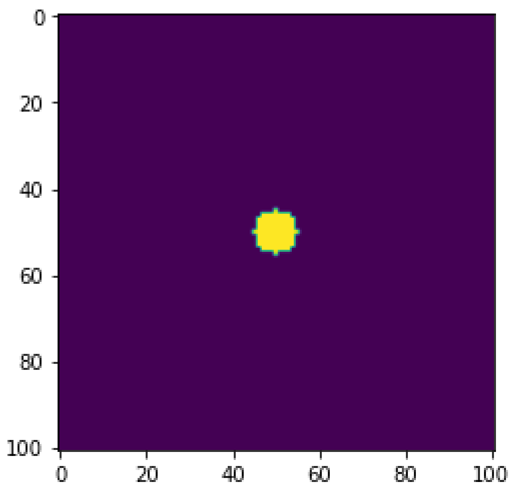
For radius = 20,

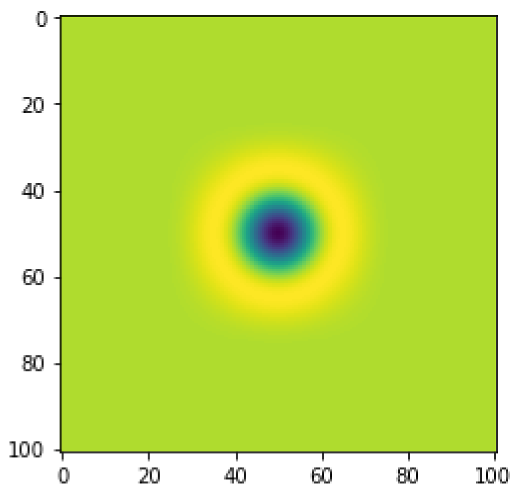






For radius = 5,

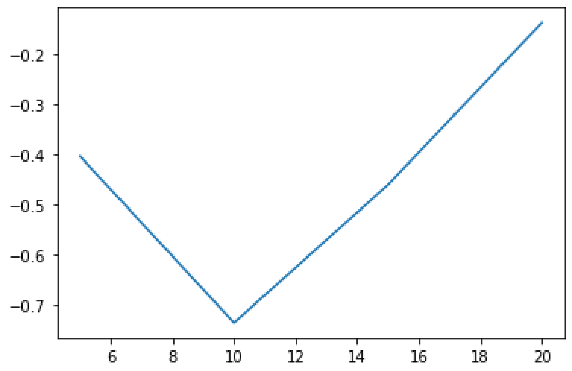




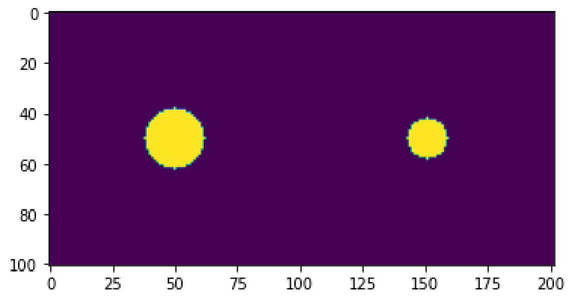
The relationship between the sigma and the radius of the circle is that the filter response will be maximized if the zero-crossing of the Laplacian filter centered at the origin occurs at the boundary of the blob.

Sigma = radius/sqrt(2)

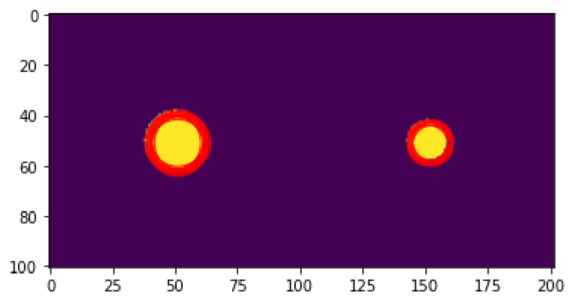
Therefore, for our case the maximum response occurs for radius = 10.



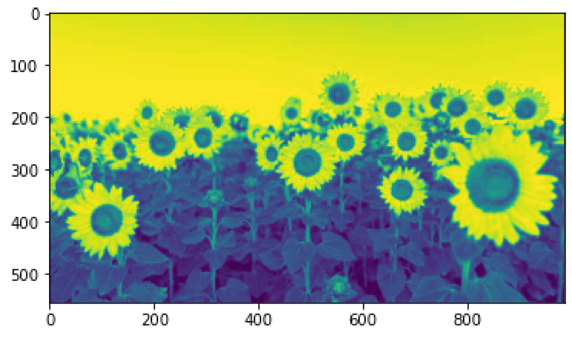
Now we are going to annotate an image with multiscale detections. The following were the images that were tested for multiscale detections.

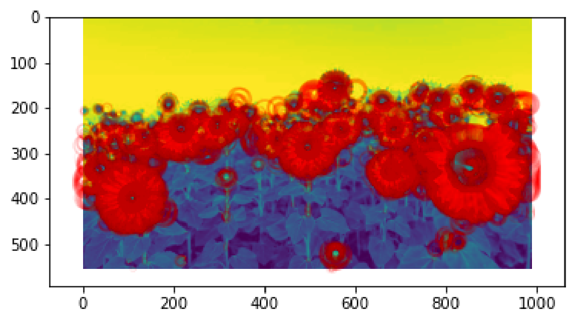


The detections for the above image were identified using the threshold value 0.5

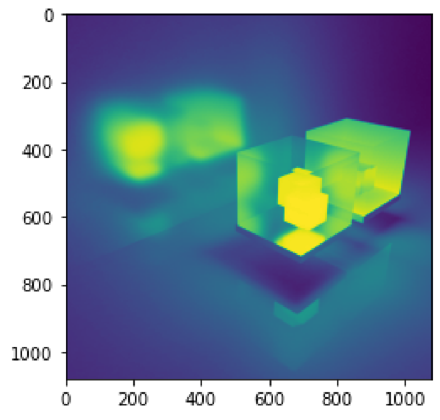


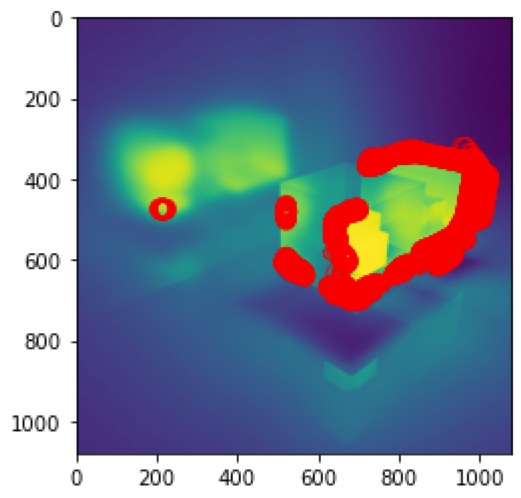
The detections for the below image were identified using the threshold value 0.02



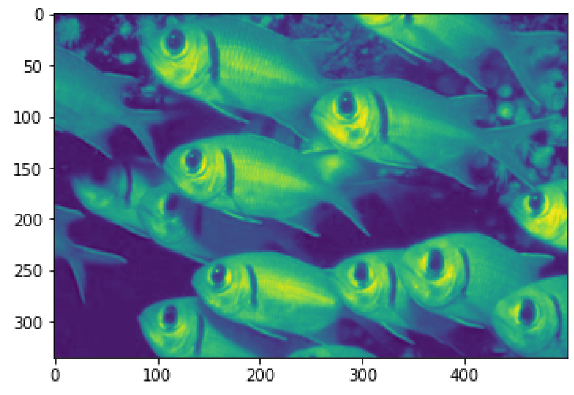


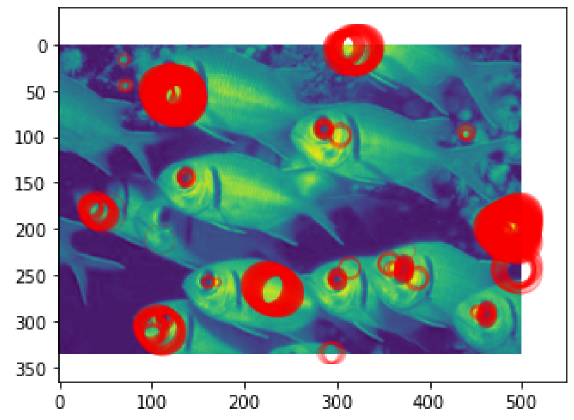
The detections for the below image were identified using the threshold value 0.02





The detections for the below image were identified using the threshold value 0.05





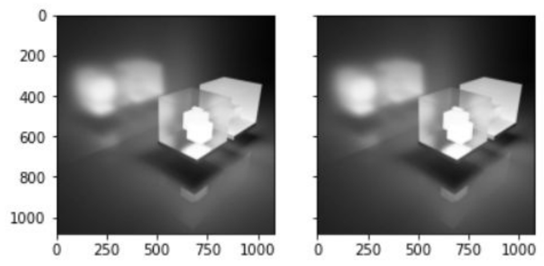
2.3 Image Warping

The following are the outcomes of the image warping implementation.

Identity Transformation,

```
[[1 0 0]
 [0 1 0]
 [0 0 1]]
```

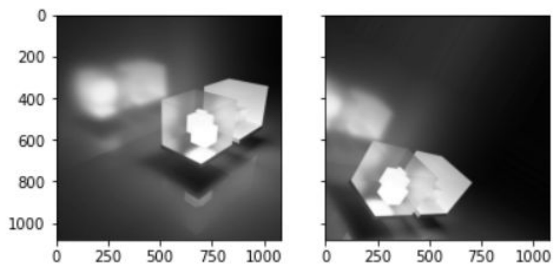
```
<matplotlib.image.AxesImage at 0x191cd55dcd0>
```



Rotate by 30,

```
[[ 0.8660254 -0.5        0.        ]
 [ 0.5        0.8660254  0.        ]
 [ 0.         0.         1.        ]]
```

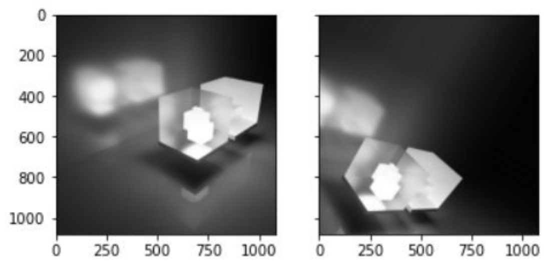
```
<matplotlib.image.AxesImage at 0x191cd7eed90>
```



Rotate by 30 and Translate to center,

```
[[ 0.8660254 -0.5        0.        ]
 [ 0.5        0.8660254  0.        ]
 [ 0.         0.         1.        ]]
```

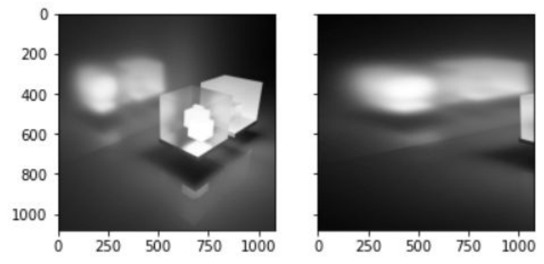
```
<matplotlib.image.AxesImage at 0x191ced214c0>
```



Scale along x by 2,

```
[[2 0 0]
 [0 1 0]
 [0 0 1]]
```

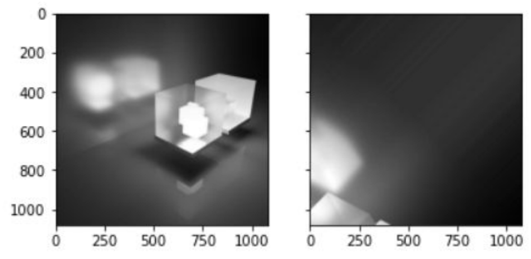
```
<matplotlib.image.AxesImage at 0x191cf00f280>
```



The kernel given in the last description used to shear the image,

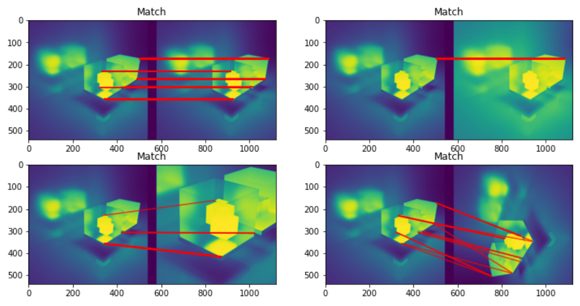
```
[[ 1 -1  0]
 [ 1  1  0]
 [ 0  0  1]]
```

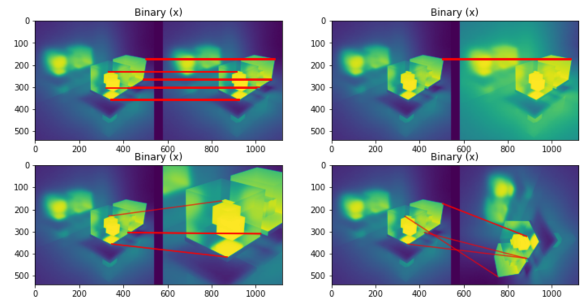
```
<matplotlib.image.AxesImage at 0x191cf2bbd60>
```

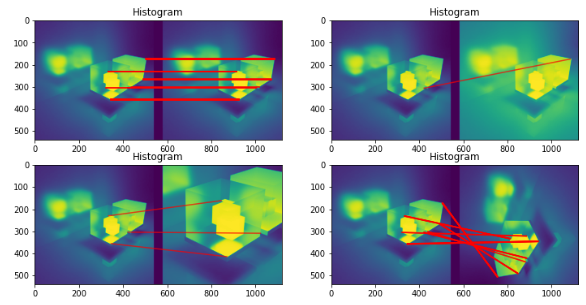


2.4 Image Descriptors

The following are the output images generated for Image descriptor implementation.







The Histogram performs poor on the img_contrast and the same performs well on the img_transpose.