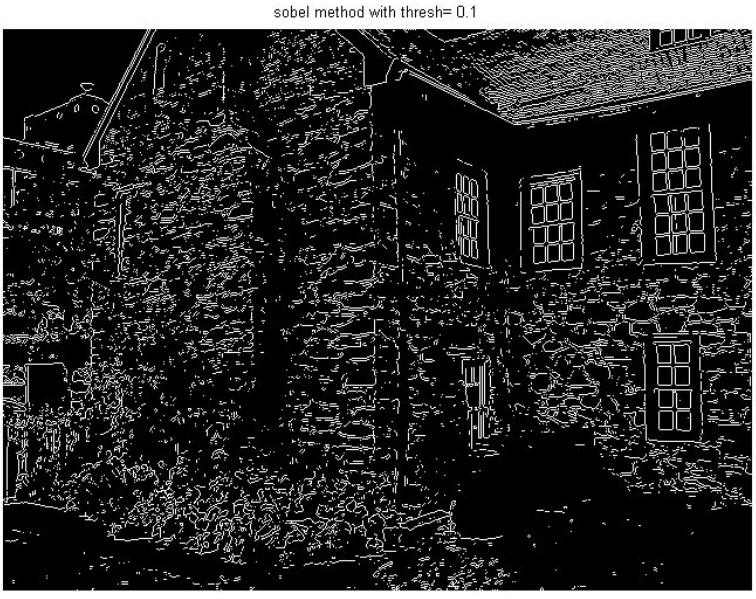
Image Features and Descriptors

Edge Detection:

The edge-detection methods detect edges by first computing a measure of edge strength, usually a first-order derivative expression such as the gradient magnitude, and then searching for local directional maxima of the gradient magnitude using a computed estimate of the local orientation of the edge, usually the gradient direction. Another way are zero-crossing based methods that search for zero crossings in a second-order derivative expression computed from the image in order to find edges, usually the zero-crossings of the Laplacian or the zero-crossings of a non-linear differential expression.

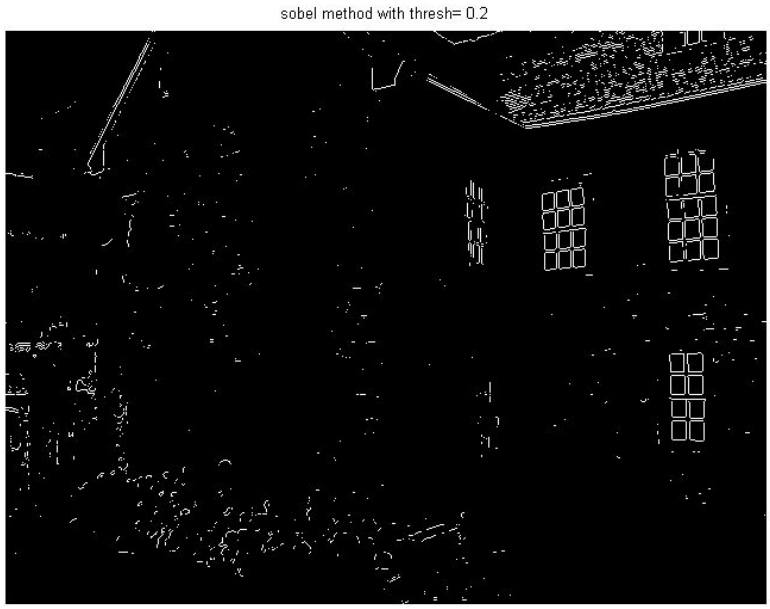
These measure of edge strength are determined by the **Threshold**. The lower the threshold, the more edges will be detected, and the result will be increasingly susceptible to noise and detecting edges of irrelevant features in the image. Conversely a high threshold may miss subtle edges, or result in fragmented edges.

a). Sobel Edge Detection Method with thresh= 0.1




```
>> thresh= 0.1;
>> d1 = edge(i,'sobel',thresh);
```

b). Sobel Edge Detection Method with thresh= 0.2



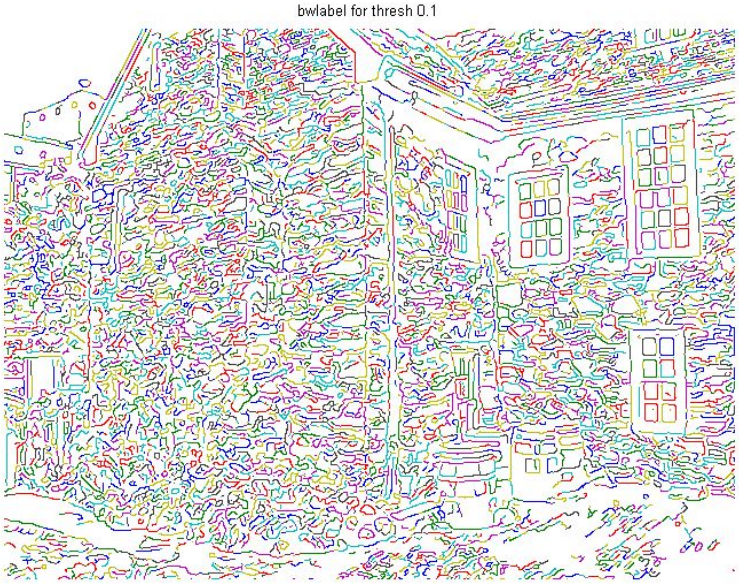

```
>> thresh= 0.2;
>> d1 = edge(i,'sobel',thresh);
```

c). Canny Edge Detection Method with thresh= 0.2


```
>> thresh= 0.2;
>> d3 = edge(i,'canny',thresh);
```

bwlabel:

a). bwlabel for thresh= 0.1




```
>>thresh= 01;
>> d3 = edge(i,'canny',thresh);
>> l = bwlabel(d3,4);
>> rgb=label2rgb(l,'lines','w','shuffle');
```

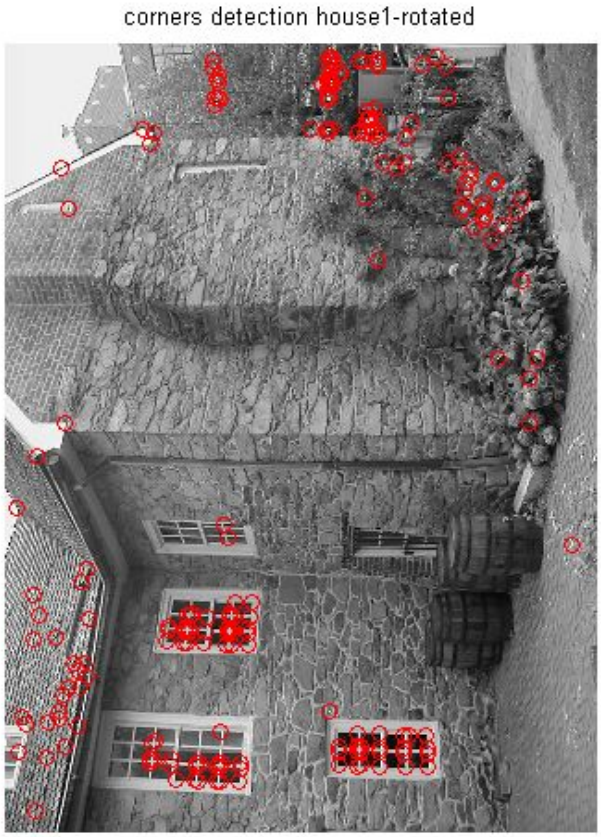
b). bwlabel for thresh= 0.2


```
>>thresh= 02;
>> d4 = edge(i,'canny',thresh);
>> l = bwlabel(d4,4);
>> rgb=label2rgb(l,'lines','w','shuffle');
```

Corner Detection

a).When the house1 image is rotated:


```
>> im =rgb2gray(imread('house1.jpg'));
>> corners = harris_corners(im, 7,1.5);
>> imshow(im); title('corners house1');
hold on;
>> plot(corners(:,1), corners(:,2), 'ro');
```




```
im2rgb2gray(imread('house1-rotated.jpg'));
>> corners2 = harris_corners(im2, 7,1.5);
>> figure, imshow(im2);title('corners
house1-rotated');
 hold on;
>> plot(corners2(:,1), corners2(:,2), 'ro');
```

From initial observations, the detected corner positions rotate by the same amount as the image. As the corner detection algorithm does not consider the orientation of the image but the relative positions of the neighbouring pixels in the image to check if its a corner or not, which remains the same even after the image is rotated. In our case, for every detected corner at the position (x, y) in the original image, there is a corresponding detected corner at (y, x) in the rotated image.
But on comparison, we find that not all of the 200 corners match(there are 12 mismatches).

b).When the house1 image is scaled down 2 times, and then 4 times:


```
>>im3 =rgb2gray(imread('house1-2down.jpg'));
>>corners2 = harris_corners(im3, 7,1.5);
>>figure, imshow(im3), title('corners detection
house1-2down');
>> hold on;
>>plot(corners2(:,1), corners2(:,2), 'ro');
```


```
>>  im4 = imread('house1-4down.jpg');
>>  im4= rgb2gray(im4);
>>  corners2 = harris_corners(im4, 7,1.5);
>>  imshow(im4), title('corners detection
house1-4down');
>>  hold on;
>>  plot(corners2(:,1), corners2(:,2), 'ro');
```

It can clearly be seen that number of corners detected in the 1-2 down image decreased considerably. But those that have been detected correctly have been are scaled in accord with the image scaling. Again the corner detection algorithm takes into account the relative position of the neighbouring pixels to find a corner,which however, may change for the reason that, with scaling intensities of the neighbouring pixels changes which may lead the algorithm to detect the wrong corner which may not be there in the actual image. But as far as the correct corners are detected their relative positions get scaled as the image.

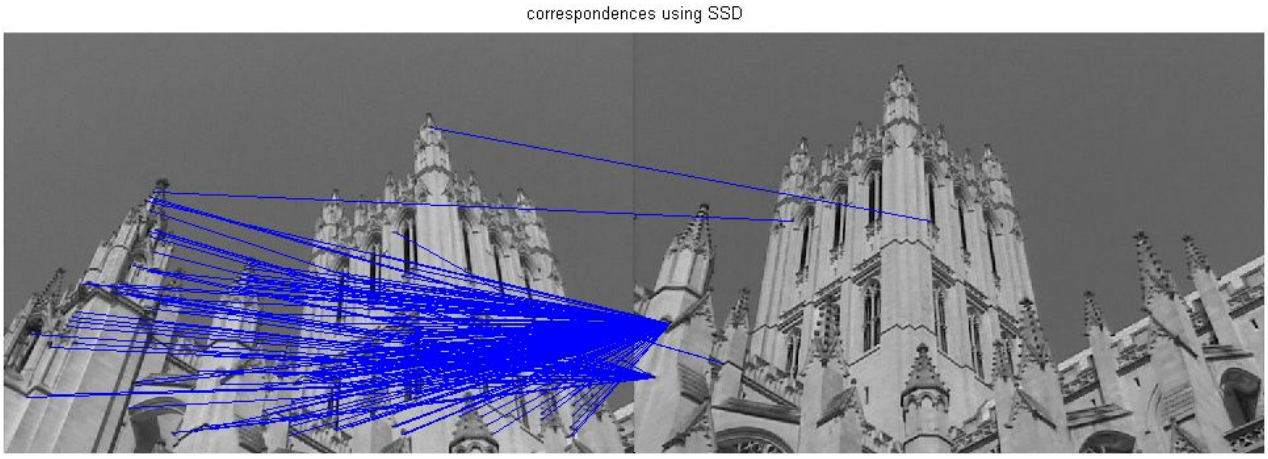
Correspondence and Matching two Images

a).SSD Algorithm.

* Corner Detection of Cathedral1:

*Corner Detection of Cathedral2:

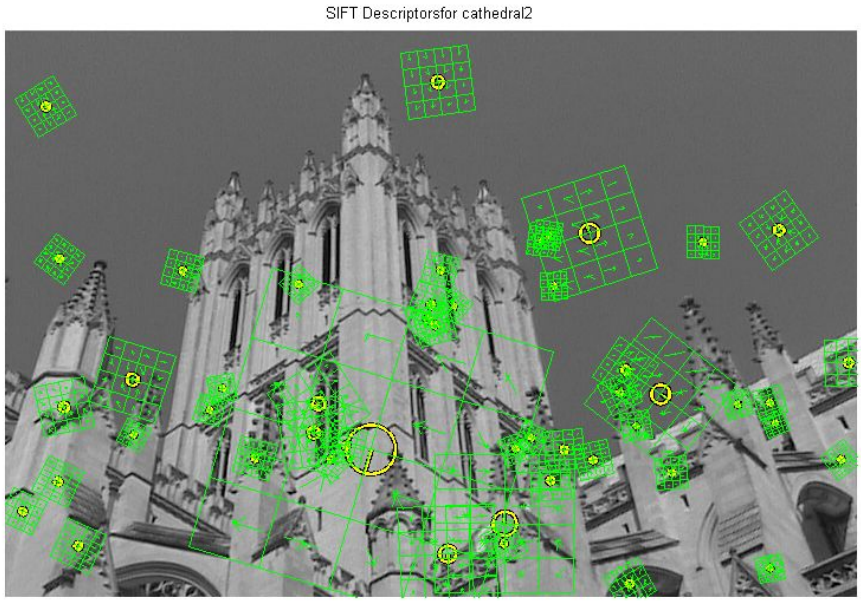
Correspondences Mapped using SSD Algorithm:



b).

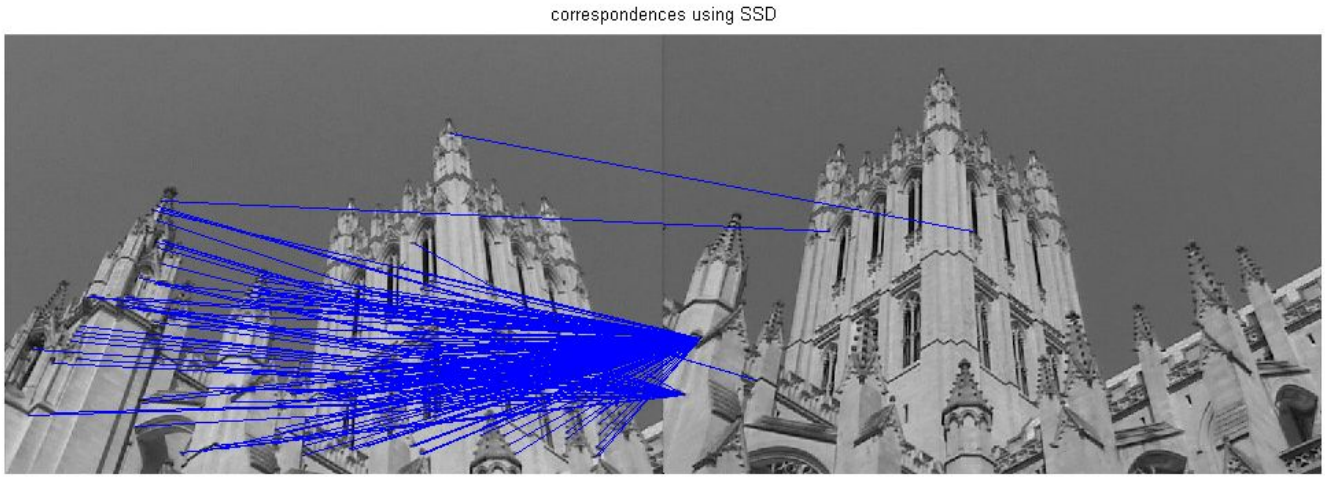
SIFT :

```
>>  I2= single(rgb2gray(imread('cathedral2.bmp')));
>> [f,d] = vl_sift(I) ;
>> I2b= (rgb2gray(imread('cathedral2.bmp')));
>> figure, imshow(I2b);
>> perm = randperm(size(f,2)) ;
sel = perm(1:50) ;
h1 = vl_plotframe(f(:,sel)) ;
h2 = vl_plotframe(f(:,sel)) ;
set(h1,'color','k','linewidth',3) ;
set(h2,'color','y','linewidth',2) ;
>>  h3 = vl_plotsiftdescriptor(d(:,sel),f(:,sel)) ;
set(h3,'color','g') ;
>> title('SIFT Descriptorsfor cathedral2');
```



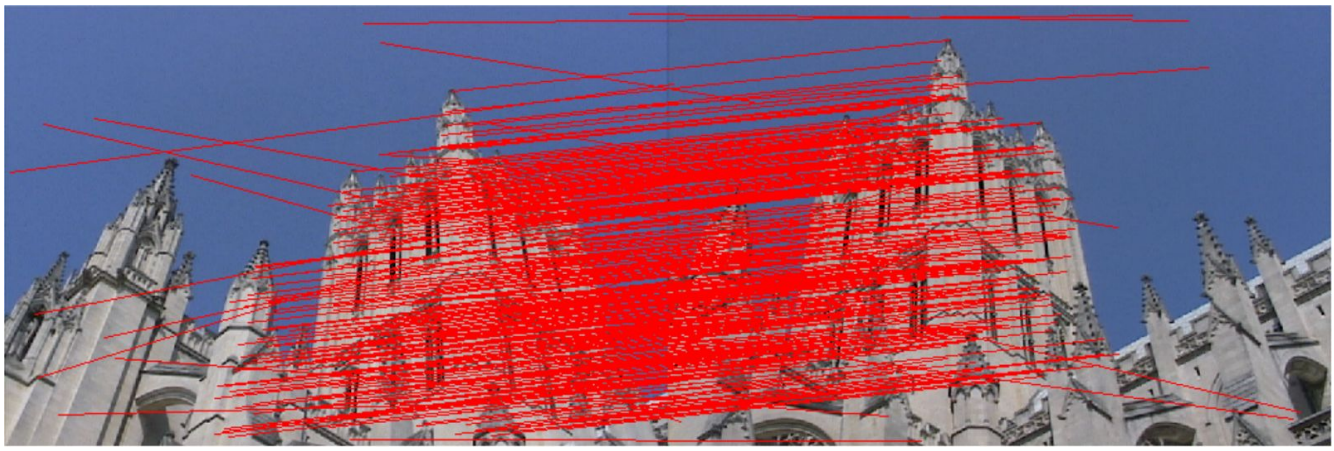
SSD vs SIFT Matching

Correspondences Mapped using SSD Algorithm:



VS

Correspondences Mapped using SIFT



It can be seen clearly that SIFT matching is better than the one using just SSD. The better performance of SIFT matching lies on the fact that it uses SIFT Descriptors which provide more and accurate information about the feature points than the simple SSD.