Assignment A6: Image Features

Clinton Fernandes

u1016390

09/01/2016

A6

CS 6320

1. Introduction

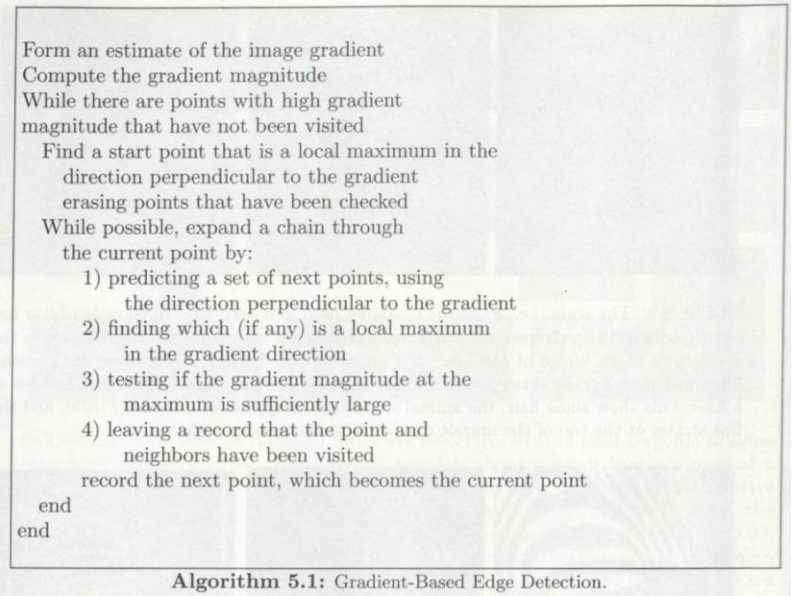
The purpose of this assignment is to Implement Algorithms 5.2 and 5.3 from the text and test and compare them as feature detectors on the given images. Through the algorithms, we have to obtain the locations, radius and the orientations of the patches. This includes examining the stability of feature detection over translation, rotation and scale.

A question that can be answered from this exercise is which of the algorithms, using Harris corner detector or Laplacian of Gaussian is better and in which cases?

2. Method

Algorithm 5.2

The Algorithm for finding the patches with the help of Harris corner detector is as follows:



```
Form an estimate of the image gradient
Compute the gradient magnitude
While there are points with high gradient
magnitude that have not been visited
  Find a start point that is a local maximum in the
    direction perpendicular to the gradient
    erasing points that have been checked
  While possible, expand a chain through
    the current point by:
      1) predicting a set of next points, using
           the direction perpendicular to the gradient
      2) finding which (if any) is a local maximum
           in the gradient direction
      3) testing if the gradient magnitude at the
           maximum is sufficiently large
      4) leaving a record that the point and
           neighbors have been visited
      record the next point, which becomes the current point
  end
end
```

Algorithm 5.1: Gradient-Based Edge Detection.

A more elaborate description of the Algorithm, as used in the code can be found below:

```
Use harris Corner detector
Find the corner regions or use all the points for R>threshold value from the
above function.

Now run a local maxima function over the thresholded values of R which gives
a more accurate location of the points representing the corners

Initialize the list of patches

For each corner do the following

    Assign xc and yc  as the location of the interest point.

      Compute radius for this patch by using simple argmax on the
      Convolution of Laplacian of Gaussian with the image
      Do this for every sigma from 0.6:0.01:3.

      Compute the orientation histogram H(Q) for the gradient orientations a
      radius k*r of (xc,yc). i.e., for all pixels surrounding (xc,yc) find
      the histogram.

    Compute orientation of the patch by using simple aragmax of H(Q)
    If there are multiple theta that maximizes the histogram then make
    multiple copies of the patch.

    Now attach a patch to patch list with (xc,yc,r,Q)
    if multiple theta, attch multiple patches.

END_of_code
```

Algorithm 5.2

The Algorithm for finding the patches with the help of Laplacian of Gaussian is as follows:

Assume a fixed scale parameter k
Find all locations and scales which are local extrema of
  $\nabla^2_\sigma \mathcal{I}(x, y)$ in location (x, y) and scale σ forming a list of triples (x_c, y_c, r)
For each such triple
  Compute an orientation histogram $H(\theta)$ for gradient orientations within
    a radius kr of (x_c, y_c).
  Compute the orientation of the patch θ_p as
    $\theta_p = \underset{\theta}{\text{argmax}}\, H(\theta)$. If there is more than one θ that
    maximizes this histogram, make one copy of the patch for each.
  Attach (x_c, y_c, r, θ_p) to the list of patches for each copy

Algorithm 5.3: Obtaining Location, Radius, and Orientation of Pattern Elements Using the Laplacian of Gaussian.

A more elaborate description of the Algorithm, as used in the code can be found below:

```
Use laplacian of Gaussian interest function.
Through this, find all the locations and the scale forming the list of
triples of (xc, yc, radius).

Initialize the list of patches

For each interest point do the following

    Assign xc and yc  as the location of the interest point.

      Compute the orientation histogram H(Q) for the gradient orientations a
      radius k*r of (xc,yc). i.e., for all pixels surrounding (xc,yc) find
      the histogram.

    Compute orientation of the patch by using simple aragmax of H(Q)
    If there are multiple theta that maximizes the histogram then make
    multiple copies of the patch.

    Now attach a patch to patch list with (xc,yc,r,Q)
    if multiple theta, attch multiple patches.

END_of_code
```

function R = CS5320_Harris(im,k)

This function take sin the image and the Harris windows size as the input and it's output is R, the corner response.

The following matrix gives a good idea of the behavior of the orientation in a window. In a window of constant gray level, both eigenvalues of this matrix are small because all the tcrm8 are small. In an edge window, we expect to see large eigenvalue associated with gradients at the edge and one small eigenvalue because few gradients run in other directions. But in a corner window, both eigenvalues should be large.

Hence the Harris corner detector;

$$\mathcal{H} = \sum_{window} \{(\nabla I)(\nabla I)^T\}$$

$$\approx \sum_{window} \begin{Bmatrix} (\frac{\partial G_\sigma}{\partial x} * *\mathcal{I})(\frac{\partial G_\sigma}{\partial x} * *\mathcal{I}) & (\frac{\partial G_\sigma}{\partial x} * *\mathcal{I})(\frac{\partial G_\sigma}{\partial y} * *\mathcal{I}) \\ (\frac{\partial G_\sigma}{\partial x} * *\mathcal{I})(\frac{\partial G_\sigma}{\partial y} * *\mathcal{I}) & (\frac{\partial G_\sigma}{\partial y} * *\mathcal{I})(\frac{\partial G_\sigma}{\partial y} * *\mathcal{I}) \end{Bmatrix}$$

The *Harris corner detector* looks for local maxima of

$$\det(\mathcal{H}) - k(\frac{\text{trace}(\mathcal{H})}{2})^2$$

this is the value of R

function [interest_pts,scale] = CS5320_LoG_interest(im,p)

We could center a blob of fixed appearance (say, dark on a light background) on the corner, and then choose the scale to be the radius of the best fitting blob. An efficient way to do this is to use a Laplacian of Gaussian filter.

The LoG_interest function uses the image as its input along with a variable p, percentage of max response to return.

The Laplacian of a function in 2D is defined as $$(\nabla^2 f)(x,y) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}.$$

Algorithm:

For all the values of the sigma in the given range, do the following:

Create a Gaussian filter for the considered sigma

Create a laplacian template

LoG = filter2 (Laplacian, Gaussian)

vals (i,j,sigma range) = filter2(LoG, image)

End

For all the pixel coordinates:

Find the Response at current pixel coordinate, along all vals planes

max_response = max(max(vals(i,j,:)));

max_index = find(vals(i,j,:)==max_response);

max_scale = s(max_index);

if (max_scale == sigma(1) | max_scale == sigma(last))

max_response(i,j) = max_response(i,j)-1000; %neglect this point

end

temp_scale(i,j) = max_scale*sqrt(2);

end

end

MAX_RESP = max(max(max_response));

for i = 1:rows

for j = 1:cols

```
                if max_response(i,j) > MAX_RESP*p
                        old_scale(i,j)=temp_scale(i,j) ;
                        new_max_response(i,j) = max_response(i,j);
                 end
        end
end
logic_scale = double(CS5320_local_max(new_max_response));
for i = 1:rows
        for j = 1:cols
                if logic_scale(i,j) ==1
                        scale(i,j) = old_scale(i,j);
                        interest_pts(i,j) = logic_scale(i,j);
                  end
        end
end
```

function H = CS5320_gradient_histogram(im,r,c,radius,w,thresh)

Algorithm:

For this function, calculate the gradients for all the points in the template surrounding the pixel in consideration, within certain radius.

Also calculate the orientations and magnitude of the gradient

Now we have to sought the pixel locations into bins according to the angle. Such that there are 9 bins with an angle range of 20 degrees and the max range is 0 to 180 degrees.

We can assign the value of the bin in two ways, by the magnitude of gradient of each point of the template of by the count of the template.

The output of the function is H (9x1 vector): orientation counts in 20-degree bins

function im_lm = CS5320_local_max(im)

This function gives the value one at the locations in the image where the local maxima exsts. For this function to be implemented, matlab's imregionalmax function was used.

3. Verification

function R = CS5320_Harris

The input to this function is the image(mxn) and the window size of the Harris matrix. Output of the function is the R(mxn) matrix whose contents are the corner response

<=0: homogeneous >0 and small: edge large: corner

To verify whether this function works, we can consider an image, for example, 'glass-box.jpg'. Giving a window size as 1, we can use this function to detect the corners of an image.

The values of pixel locations are to be marked red in the original image for all the values of R greater than a threshold value.

```
imshow(im)
[rows,cols] = size(im);
hold on;
for i = 1+k:rows-k
    for j = 1+k:cols-k
        if R(i,j)>max(max(R))/15
            plot(j,i,'r.')
        end
    end
end
```

Fig: code to display corners via Harris corner detector

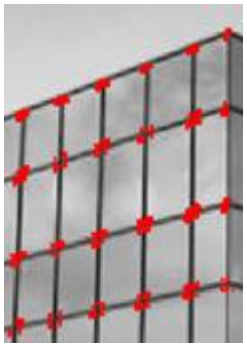


Fig: Detected corners in image

function im_lm = CS5320_local_max()

This function gives non-zero values at local maxima as its output. The function can be verified by taking a simple image having its local maxima at some points, which are known. Then, after running the function on the same image we can check if the function gives the required local maxima.

```
A = 10*ones(10,10);
A(2:4,2:4) = 22;
A(6:8,6:8) = 33;
A(2,7) = 44;
A(3,8) = 45;
A(4,9) = 44;
A(3,3) = 60;
A(8,8) = 70;
```

```
A =

    10    10    10    10    10    10    10    10    10    10
    10    22    22    22    10    10    44    10    10    10
    10    22    60    22    10    10    10    45    10    10
    10    22    22    22    10    10    10    10    44    10
    10    10    10    10    10    10    10    10    10    10
    10    10    10    10    10    33    33    33    10    10
    10    10    10    10    10    33    33    33    10    10
    10    10    10    10    10    33    33    70    10    10
    10    10    10    10    10    10    10    10    10    10
    10    10    10    10    10    10    10    10    10    10

>> A_lm = CS5320_local_max(A);
>> imshow(A_lm)
```

 <= This image displays the places where the local maxima exists.

function H = CS5320_gradient_histogram(im,r,c,radius,w,thresh)

To verify this function. Let us consider a small image, whose gradient orientations angles we know at the different locations.

Fig: Image

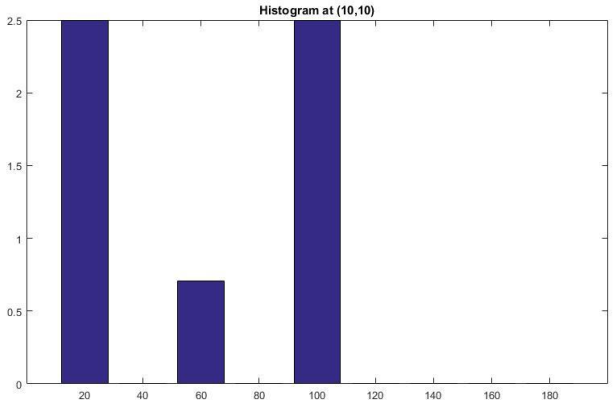


Fig: Histogram at 10,10

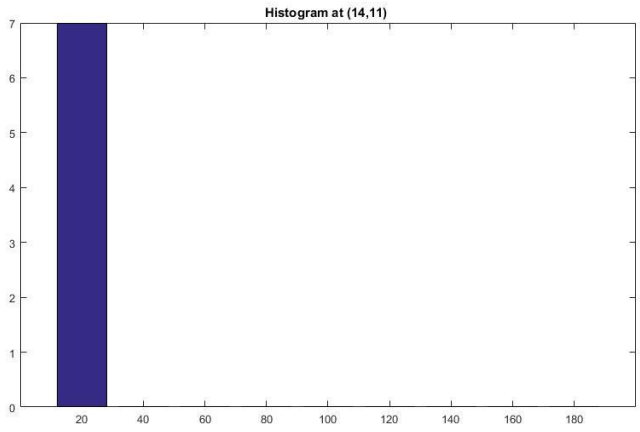


Fig: Histogram at 14,11

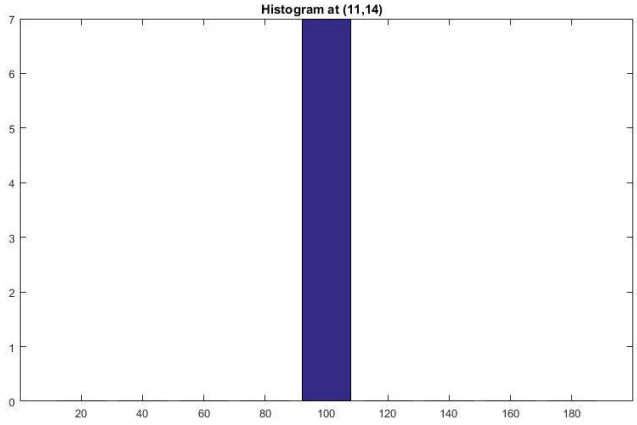


Fig: Histogram at 11,14

function [interest_pts,scale] = CS5320_LoG_interest(im,p)

To test this function and check if it gives the interest point and radius at the required point, consider the following image. As the blob is at the center, having radius 1, we should expect a similar function output.

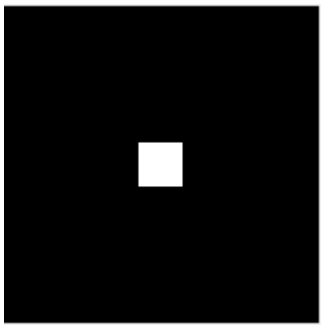


Figure: 20x20 image

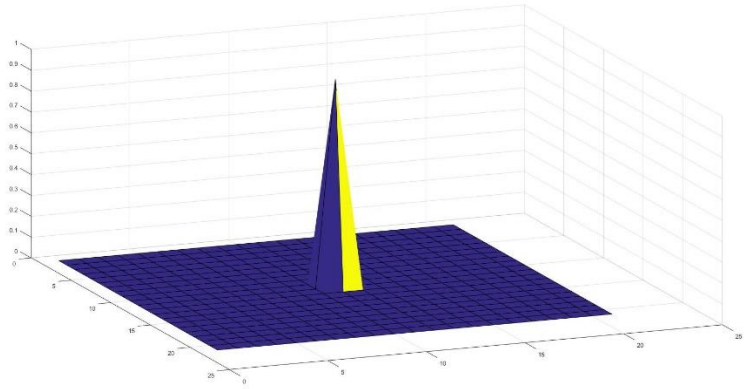


figure: interest point is the center

	10	11	12
10	0	0	0
11	0	1.0889	0
12	0	0	0
13	0	0	0

Figure: the scale/radius of the blob at the center (11,11)

4. Data

For this section,

First we check the response of the Algorithms 5.2 and 5.3 on the images.

Then we will study how the performance of the Algorithms 5.2 and 5.3 differ when we translate and rotate the image.

Fig: CS5320_corner_patches on 'glass-box.jpg'

Number of patches = 26 computation time = 8.88 sec

Interest points by visual inspection seem to be at the corners.

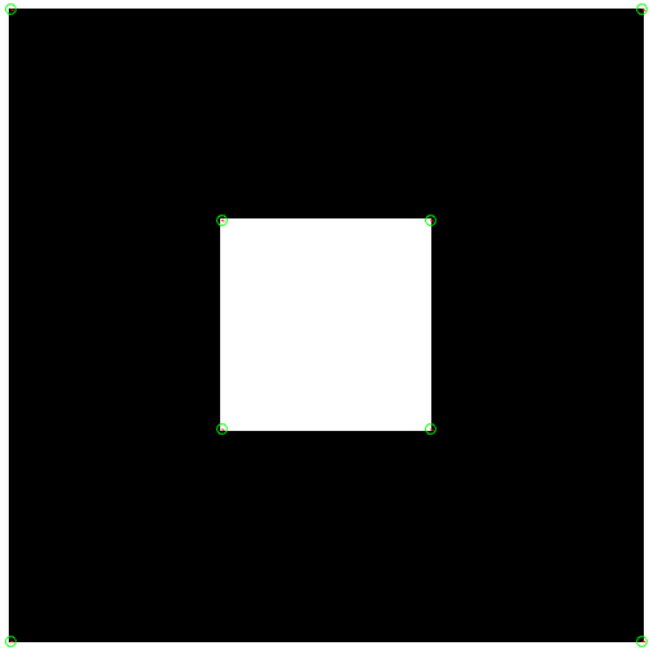


Fig: CS5320_corner_patches on 'square.jpg'

Number of patches = 8 computation time = 8.88 sec

Fig: CS5320_LoG_patches on 'glass-box.jpg'

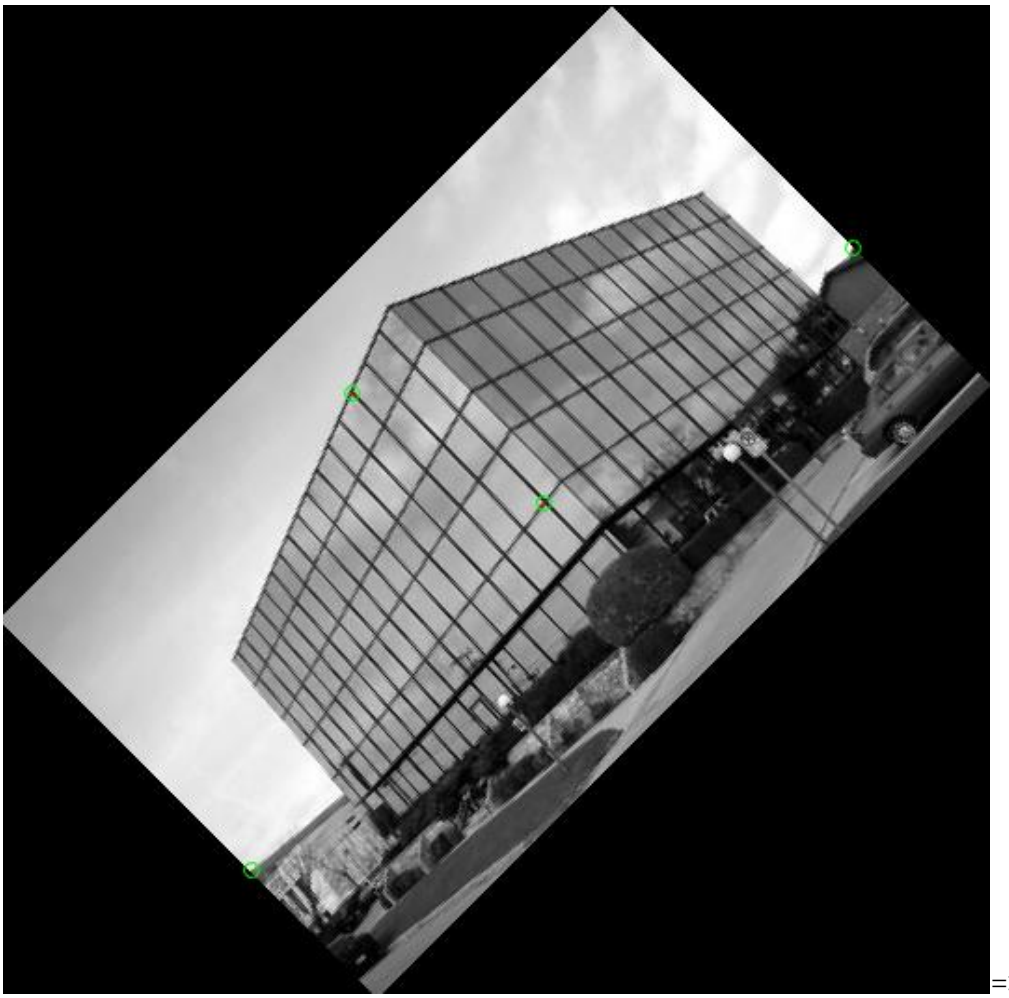
Interest points by visual inspection are at the corners.

On translating the image

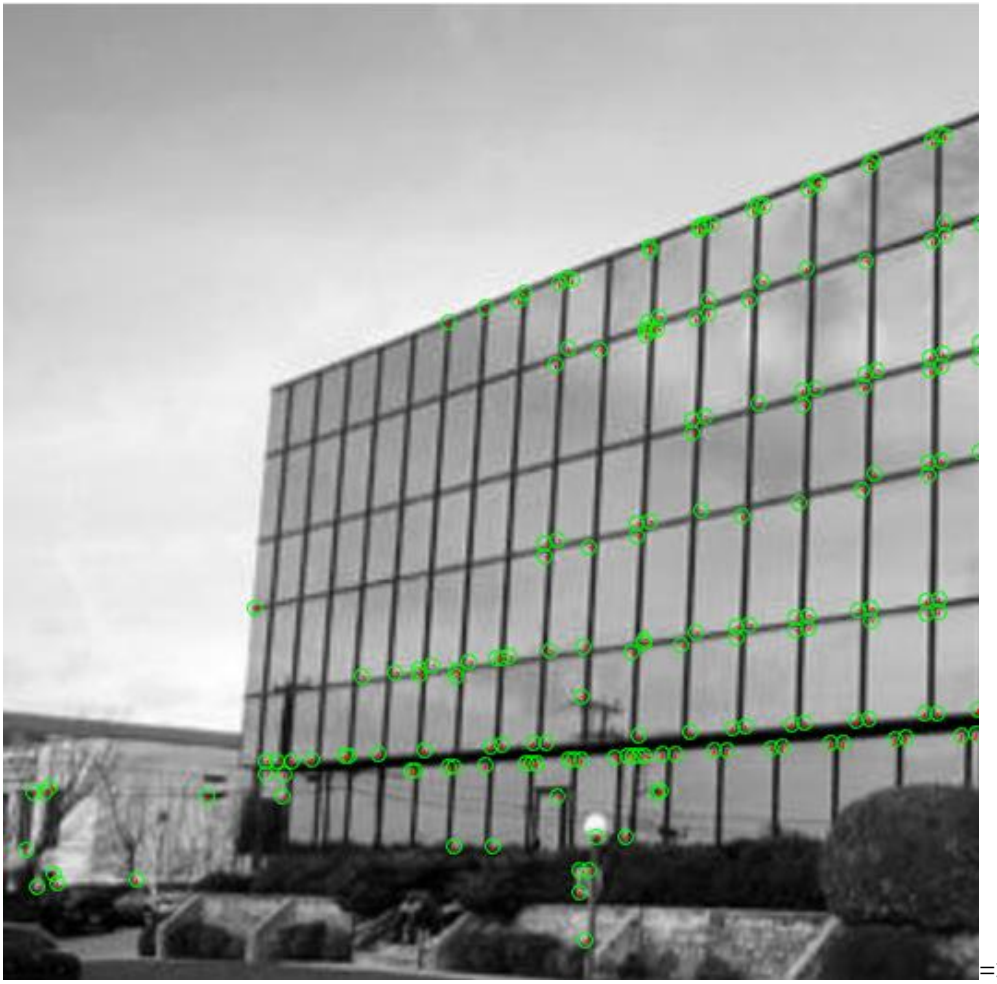
 => Algorithm 5.1

 => Algorithm 5.2

On rotating image


=> Algorithm 5.1

On scaling image by 2


=> Algorithm 5.1

5. Analysis

Number of patches detected on image ‘glass-box.jpg’ for Algorithm 5.1:

Original image = 224

Translated image = 81

Rotated image = 4

Scaled image = 221

Number of patches detected on image ‘glass-box.jpg’ for Algorithm 5.1:

Original image = 150

Translated image = 322

Rotated image = ~

Scaled image = 4

6. Interpretation

The algorithm 5.2 is stable for scaling, fairly stable to translation and unstable for rotation.

The algorithm 5.3 is unstable for scaling, stable to translation and unstable for rotation.

Algorithm 5.2 depends on Harris corner detector which tend to produce neighborhoods where the estimate of the center is very accurate, but the scale estimate is poor. Algorithm 5.3 depends on Laplacian of Gaussian methods produce neighborhoods where the estimate of the center is less accurate, but the scale estimate is better.

7. Critique

Due to lack of time (catching up for a midterm exam), I could not do the experimentation with the algorithms properly.

If I had to write the report properly and hence do the experiments, I would have compared the performance of both algorithm 5.2 and 5.3 more thoroughly on both the given images.

8. Log

Coding time: 23 hrs

Report: 3.5 hrs