Homework 7 – Features

Arya Araban

Abstract

In this Article we will explore the idea of feature points in digital images and try to estimate the geometry of an image given these feature points. We'll also be looking at how to construct an image that has a different geometry so that it matches an original image.

Finally, we'll implement a corner detection algorithm known as the Harris algorithm.

Report Info

Date: 1399/11/04

Keywords:

Feature Points

Geometry Estimation

Geomrty Transform

Corner Detection

1- Introduction

Feature point detection is one of the most important operations which we can perform on digital images. The way the detection works is that each pixel is separately checked, and if the conditions we base on them are true, it'll be considered as a feature point.

There are many different uses for these feature points, one of which is matching the feature points of two given images, and then approximating the geometry of the first image to the second image. Another use can be that once we find the feature points of two images, we can match them and then place the two images side-by-side to create a panaroma image.

2- Technical Description

Problem 7.1.1:

Estimate the geometry of the images in the Attacked1 folder to the haftone reference image. After finding the transformation matrix, apply it to the images in the Attacked2 folder to get a resulted geometry estimation.

Solution:

We already stated that in order to estimate the geometry of an image, we have to compare the feature points of two images and find their matches. After which, we can find a transformation matrix which will help us map the image with a geometry change to a reference image.

In order to do this, we follow several steps (note that we use the BRISK algorithm to find the feature points on an image)

Step 2 – Structure matrix setup:
Given the horizontal and vertical derivatives we can find the structure matix M which is defined as follows:

$$M = \sum_{(x,y)\in W} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} = \begin{bmatrix} \sum_{(x,y)\in W} I_x^2 & \sum_{(x,y)\in W} I_x I_y \\ \sum_{(x,y)\in W} I_x I_y & \sum_{(x,y)\in W} I_y^2 \end{bmatrix}$$

**note that this step can be optional, as long as we directly use the formulas given in the next step.*

Step 3 – Harris Response Calculation:
In this step, we compute the smallest eigenvalue of the structure tensor using the following approximation:

$$\lambda_{min} \approx \frac{\lambda_1 \lambda_2}{(\lambda_1 + \lambda_2)} = \frac{\det(M)}{\operatorname{tr}(M)}$$

with the trace $\operatorname{tr}(M) = m_{11} + m_{22}$

Step 4 – Non-Maximum Suppression:
in order to pick up the optimal values to indicate corners, we find the local maxima as corners within the window which is a 3 by 3 filter.

3- Describing the Results

Problem 7.1.1:
The original Lena image is as follows:

***Step1-** Find feature points of the geometry changed image*

***Step2-** Find feature points on the reference image*

***Step3-**Find the feature points which have a correspondence between the two images*

***Step4-**Find the Transformation matrix used to map the first image to the second image.*

If we denote the feature points from the source image as S(x,y) and the feature points from the target image T(x,y), the general definition for extracting the transformation between two images can be defined by the following equation:

$$S(x,y) \rightarrow T(x,y), \begin{bmatrix} x_t \\ y_t \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & t_x \\ c & d & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_s \\ y_s \\ 1 \end{bmatrix}$$

Note that we need at least three feature points to match in order to find the six unknown parameters.

Problem 7.1.2:
Implement the Harris corner detection algorithm for a given image

Solution:
In order to implement the Harris corner detection algorithm we follow a few steps:

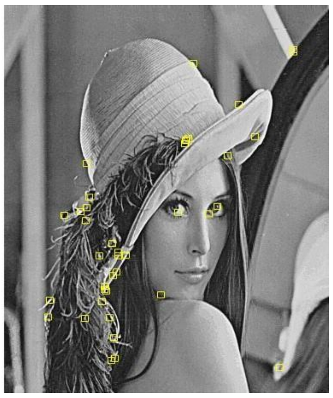
Step 1 - Spatial derivative calculation:
First we need to find the horizontal and vertical derivative's of a given image.
$I_x(x,y)$ and $I_y(x,y)$.

If we compare this image to the original image:

```
MSE=28.1423 --- SSIM=0.9159 --- MP= 6
```

Type 2:

The matched points of the second picture in attacked1 is as follows:

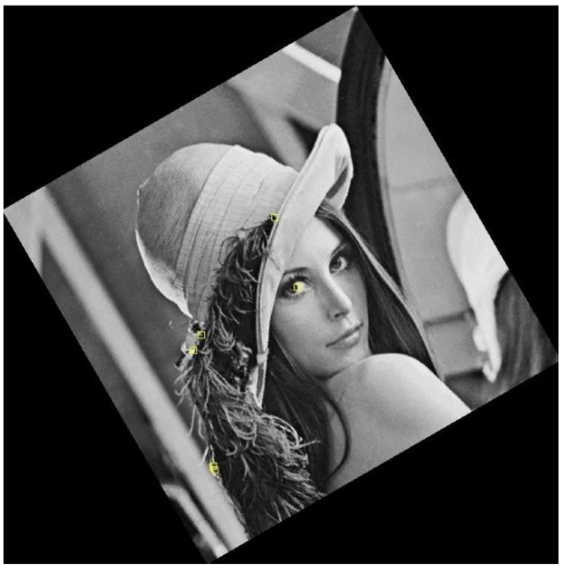


For the reference image, the matched points are as follows:

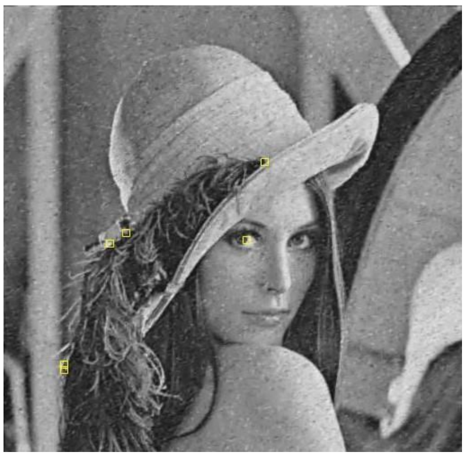
For explaining the results of this problem, we will look at each of the Types separately:

Type 1:

The matched points of the first picture in attacked1 is as follows:



For the reference image, the matched points are as follows:



Applying the obtained transformation on the first image of attacked2 gives the following:

For the reference image, the matched points are as follows:

Applying the obtained transformation on the third image of attacked2 gives the following:

If we compare this image to the cropped original image:

```
MSE=410.85 --- SSIM=0.61135 --- MP= 8
```

Applying the obtained transformation on the second image of attacked2 gives the following:

If we compare this image to the original image:

```
MSE=38.85 --- SSIM=0.93099 --- MP= 39
```

Type 3:

*note that since the third image of attacked1 image is cropped, we also have to crop the reference image and original image.

The matched points of the third picture in attacked1 is as follows:

Overall Results:

Mean:

```
MSE=132.25 --- SSIM=0.83875 --- MP= 19
```

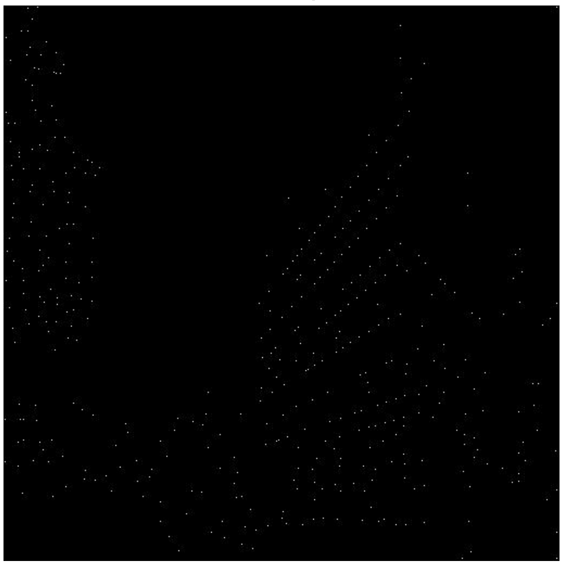
STD:

```
MSE=161.055 --- SSIM=0.132494 --- MP= 13.285
```

Problem 7.2.1:

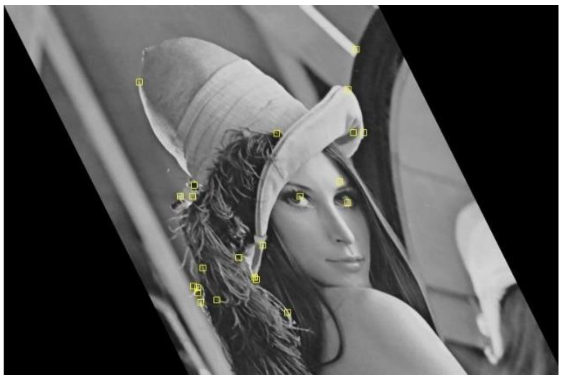
The original building image is as follows:

If we follow the Harris corner detection algorithm steps with a threshold size of 0.03, and a sigma size of 3 we can end up with the following binary image (The white color shows the corners):

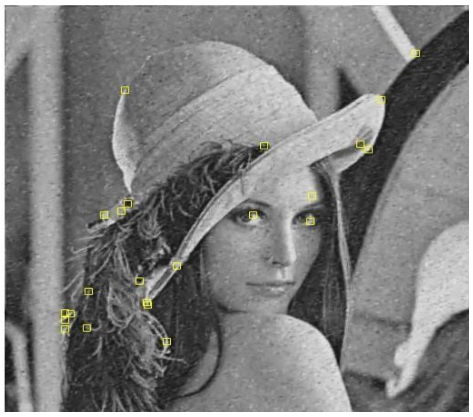


Type 4:

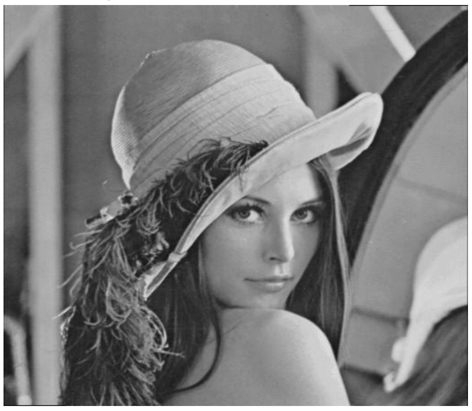
The matched points of the third picture in attacked1 is as follows:



For the reference image, the matched points are as follows:

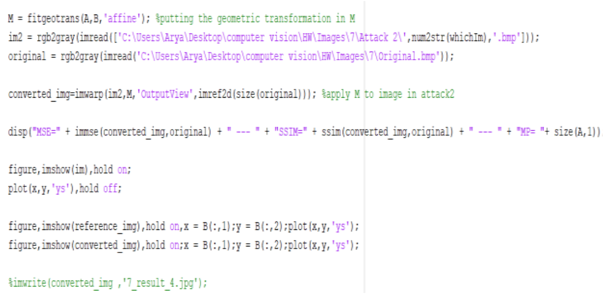


Applying the obtained transformation on the fourth image of attacked2 gives the following:



If we compare this image to the original image:

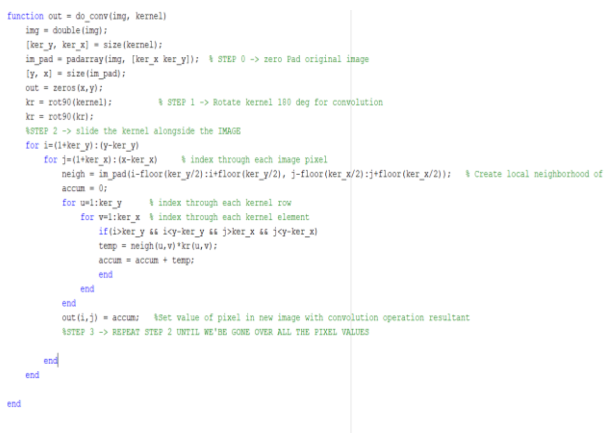
```
MSE=51.1584 --- SSIM=0.89678 --- MP= 23
```



```
M = fitgeotrans(A,B,'affine'); %putting the geometric transformation in M
im2 = rgb2gray(imread(['C:\Users\Arya\Desktop\computer vision\HW\Images\7\Attack 2\',num2str(whichIm),'.bmp']));
original = rgb2gray(imread('C:\Users\Arya\Desktop\computer vision\HW\Images\7\Original.bmp'));

converted_img=imwarp(im2,M,'OutputView',imref2d(size(original))); %apply M to image in attack2

disp("MSE=" + immse(converted_img,original) + " --- " + "SSIM=" + ssim(converted_img,original) + " --- " + "MP= "+ size(A,1))

figure,imshow(im),hold on;
plot(x,y,'ys'),hold off;

figure,imshow(reference_img),hold on,x = B(:,1);y = B(:,2);plot(x,y,'ys');
figure,imshow(converted_img),hold on;x = B(:,1);y = B(:,2);plot(x,y,'ys');

%imwrite(converted_img ,'7_result_4.jpg');
```

Problem 7.1.2:

The function used to apply convolution to an image is as follows:

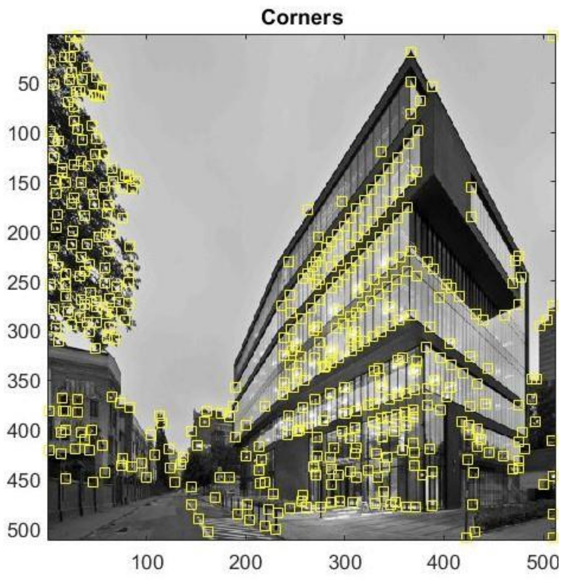


```
function out = do_conv(img, kernel)
    img = double(img);
    [ker_y, ker_x] = size(kernel);
    im_pad = padarray(img, [ker_x ker_y]);  % STEP 0 -> zero Pad original image
    [y, x] = size(im_pad);
    out = zeros(x,y);
    kr = rot90(kernel);          % STEP 1 -> Rotate kernel 180 deg for convolution
    kr = rot90(kr);
    %STEP 2 -> slide the kernel alongside the IMAGE
    for i=(1+ker_y):(y-ker_y)
        for j=(1+ker_x):(x-ker_x)     % index through each image pixel
            neigh = im_pad(i-floor(ker_y/2):i+floor(ker_y/2), j-floor(ker_x/2):j+floor(ker_x/2));   % Create local neighborhood of
            accum = 0;
            for u=1:ker_y      % index through each kernel row
                for v=1:ker_x  % index through each kernel element
                    if(i>ker_y && i<y-ker_y && j>ker_x && j<y-ker_x)
                    temp = neigh(u,v)*kr(u,v);
                    accum = accum + temp;
                    end
                end
            end
            out(i,j) = accum;   %Set value of pixel in new image with convolution operation resultant
            %STEP 3 -> REPEAT STEP 2 UNTIL WE'BE GONE OVER ALL THE PIXEL VALUES

        end
    end

end
```

The function used to apply the harris algorithm (with comments to show what's happening):

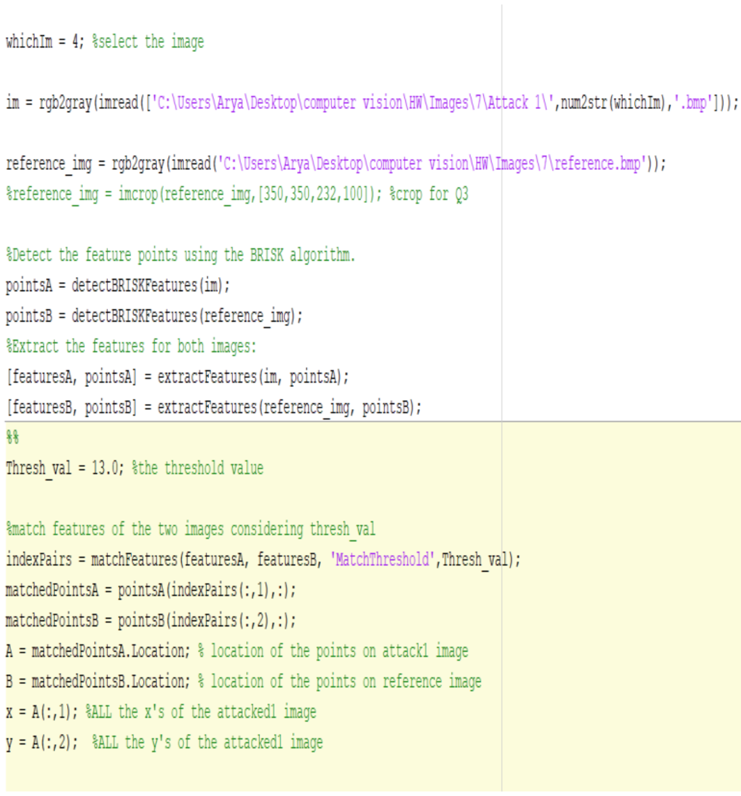
```
function [res, r, c] = harris(im)

    sigma = 3; thresh = 0.03;

    dx = [-1 0 1; -1 0 1; -1 0 1]; % Derivative masks
    dy = dx';

    Ix = do_conv(im, dx);     % Image derivatives
    Iy = do_conv(im, dy);

    % Generate Gaussian filter of size 6*sigma (+/- 3sigma) and of
    % minimum size 1x1.
    g = fspecial('gaussian',max(1,fix(6*sigma)), sigma);

    % Smoothed squared image derivatives:
    Ix2 = do_conv(Ix.^2, g);
    Iy2 = do_conv(Iy.^2, g);
    Ixy = do_conv(Ix.*Iy, g);

    % Find the Harris corner measure based on found derivatives:
    res = (Ix2.*Iy2 - Ixy.^2)./(Ix2 + Iy2 + eps);

    % Extract local maxima by performing a grey scale morphological
    % dilation and then finding points in the corner strength image that
    % match the dilated image and are also greater than the threshold.
    sze = 2+1;                    % Size of mask.
    mx = ordfilt2(res,sze^2,ones(sze)); % Grey-scale dilate.
    res = (res==mx)&(res>thresh);       % Find maxima.

    [r,c] = find(res);                  % Find row,col coords.

    %show points on the original image:
     figure, imagesc(im), axis image, colormap(gray), hold on
     plot(c,r,'ys'), title('Corners');


end
```

And finally, if we want to visualize the corners we found on the original image, we have the following:



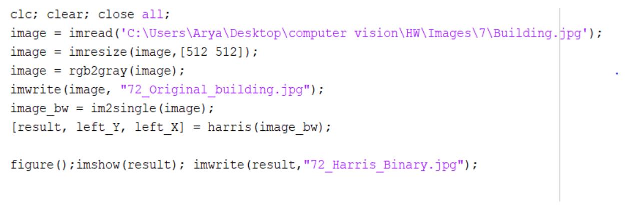
4- Appendix(code)

Problem 7.1.1:



```
whichIm = 4; %select the image

im = rgb2gray(imread(['C:\Users\Arya\Desktop\computer vision\HW\Images\7\Attack 1\',num2str(whichIm),'.bmp']));

reference_img = rgb2gray(imread('C:\Users\Arya\Desktop\computer vision\HW\Images\7\reference.bmp'));
%reference_img = imcrop(reference_img,[350,350,232,100]); %crop for Q3

%Detect the feature points using the BRISK algorithm.
pointsA = detectBRISKFeatures(im);
pointsB = detectBRISKFeatures(reference_img);
%Extract the features for both images:
[featuresA, pointsA] = extractFeatures(im, pointsA);
[featuresB, pointsB] = extractFeatures(reference_img, pointsB);
%%
Thresh_val = 13.0; %the threshold value

%match features of the two images considering thresh_val
indexPairs = matchFeatures(featuresA, featuresB, 'MatchThreshold',Thresh_val);
matchedPointsA = pointsA(indexPairs(:,1),:);
matchedPointsB = pointsB(indexPairs(:,2),:);
A = matchedPointsA.Location; % location of the points on attack1 image
B = matchedPointsB.Location; % location of the points on reference image
x = A(:,1); %ALL the x's of the attacked1 image
y = A(:,2);  %ALL the y's of the attacked1 image
```

And finally, the main program’s code:



```
clc; clear; close all;
image = imread('C:\Users\Arya\Desktop\computer vision\HW\Images\7\Building.jpg');
image = imresize(image,[512 512]);
image = rgb2gray(image);
imwrite(image, "72_Original_building.jpg");
image_bw = im2single(image);
[result, left_Y, left_X] = harris(image_bw);

figure();imshow(result); imwrite(result,"72_Harris_Binary.jpg");
```