CMPE 465: Intoduction to Computer Vision

Homework 2 Report

Efe Berk ERGULEC

Instructor: Venera ADANOVA

November 24, 2019

Introduction

Feature detection and matching is the process of detecting corners or similar points. Image alignment is the process of finding the elements in one image into meaningful correspondence with elements in a second image. Both of them include complex steps and cover subjects like homography matrix, feature-based alignment, image transformation and warping, translations and RANSAC. Current assignment will implement and provide solutions to all of these questions.

First question is about Harris Corner detector, which is useful to find corners of images. In second question, I will use Harris Corner Detection that I wrote in previous question, and other already existing routines in order to match features. Last question is RANSAC implementation to eliminate outliers and provide more accurate results. Each question will be explained with their results and these outcomes will be discussed.

Question 1

Harris Corner Detection is a corner detection algorithm that takes an image as input with σ and threshold values to find appropriate corners of input image. It gives corner locations as (x,y) and number of these corners. For this question, instructor asked to use this method to find corners and print them. Before discussing results, I want to explain Harris Corner Detection algorithm that I implemented.

Harris Corner Detector

As discussed above, my Harris Corner Detector algorithm function implements like below:

[cornerPts, numCornerPts] = HarrisCornerDetector(image, sigma, thres)

- *cornerPts* : Indices of matching corners of *image* picture.
- *numCornerPts* : Number of corner points that HarrisCornerDetector found.
- *image* : Input image.
- *sigma* (σ) : Sigma value. It is for gaussian filter.
- *thres* : Threshold value.

To sum up its functionality shortly, it takes image with σ and threshold and performs operations to earn matched points and number of these points. Algorithm of Harris Corner Detector shown as pseudocode at the next page.

Algorithm 1 Harris Corner Detector

Require: image, σ, thres

Ensure: cornerPts, numCornerPts

$img \leftarrow imread(image)$
if length(size(img)) > 2 **then**
$img \leftarrow rgb2gray(img)$
end if
$f_x \leftarrow [-1, 0, 1; -1, 0, 1; -1, 0, 1]$
$f_y \leftarrow [-1, -1, -1; 0, 0, 0; 1, 1, 1]$
$I_x \leftarrow filter2(f_x, img)$
$I_y \leftarrow filter2(f_y, img)$
$I_x^2 \leftarrow I_x.\hat{}2$
$I_y^2 \leftarrow I_y.\hat{}2$
$I_{xy} \leftarrow I_x. * I_y$
$h \leftarrow fspecial('gaussian', [7, 7], \sigma)$;
$I_{x_g}^2 \leftarrow filter2(h, I_x^2)$
$I_{y_g}^2 \leftarrow filter2(h, I_y^2)$
$I_{xy_g}^2 \leftarrow filter2(h, I_{xy})$
$height \leftarrow size(img, 1)$
$width \leftarrow size(img, 2)$
$result \leftarrow zeros(height, width)$
$R \leftarrow zeros(height, width)$
$R_{max} \leftarrow 0$
for i $\leftarrow$ 1:height **do**
for j $\leftarrow$ 1:width **do**
$H \leftarrow [I_{x_g}^2(i,j), I_{xy_g}^2(i,j); I_{xy_g}^2(i,j), I_{y_g}^2(i,j)]$;
$R(i,j) \leftarrow det(H) - thres * (trace(H))\hat{}2$
if R(i,j) $> R_{max}$ **then**
$R_{max} \leftarrow R(i,j)$
end if
end for
end for
$numCornerPts \leftarrow 0$
for i $\leftarrow$ 2:height-1 **do**
for j $\leftarrow$ 2:width-1 **do**
if R(i,j) $> 0.1 * R_{max}$ && R(i,j) $>$ R(i-1,j-1) && R(i,j) $>$ R(i-1,j) && R(i,j) $>$ R(i-1,j+1) && R(i,j) $>$ R(i,j-1) && R(i,j) $>$ R(i,j+1) && R(i,j) $>$ R(i+1,j-1) && R(i,j) $>$ R(i+1,j) && R(i,j) $>$ R(i+1,j+1) **then**
$result(i,j) \leftarrow 1$
$numCornerPts \leftarrow numCornerPts + 1$
end if
end for
end for
$[y, x] \leftarrow find(result == 1)$
$cornerPts \leftarrow [x, y]$

Functions Used

For this section, only new functions written down below.

- `filter2()` : Instead of `imfilter()` function that has used in previous assignment, I used this function because images that has given to us for this assignment are two-dimensional images. In addition, it gives way more efficient results than `imfilter()`.
- `trace()` : Calculates the sum of the diagonal elements of matrix A.
- `find()` : Find indexes of matrix that is expected from condition written inside its parenthesis.

From pseudocode above, outputs gained from every steps provided below (Figures 1, 2, 3 and 4).

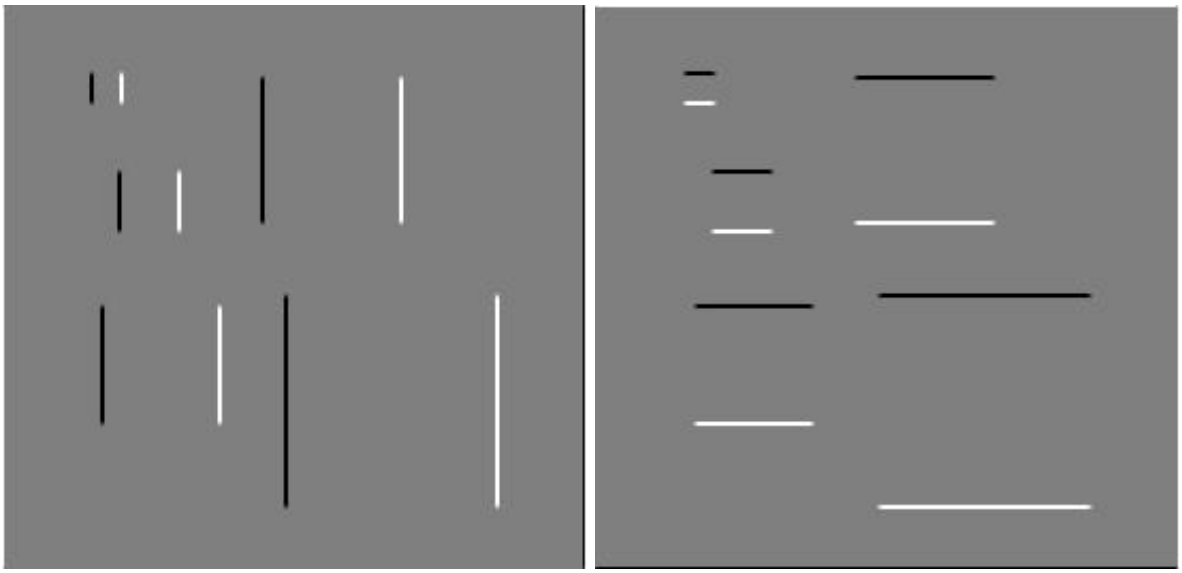

Figure 1: Results after first derivative ($I_x and I_y$).

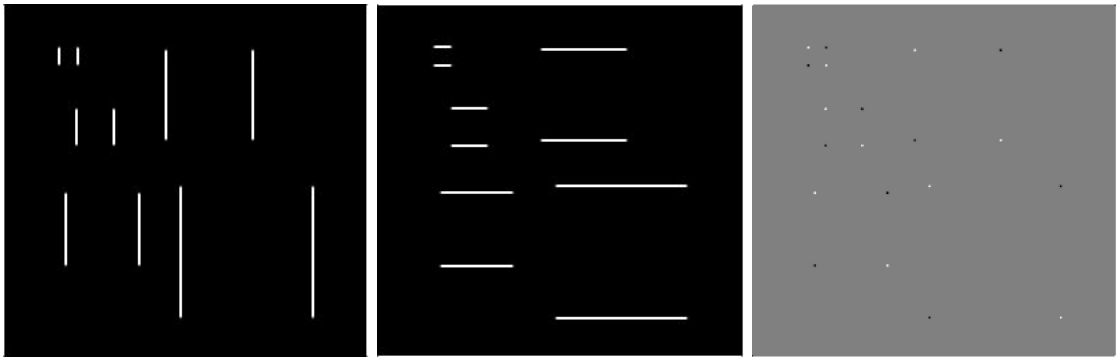

Figure 2: Results after second derivative ($I_x^2, I_y^2 and I_{xy}$).

Figure 3: Results after gaussian filter ($g(I_x^2), g(I_y^2) and g(I_{xy})$).

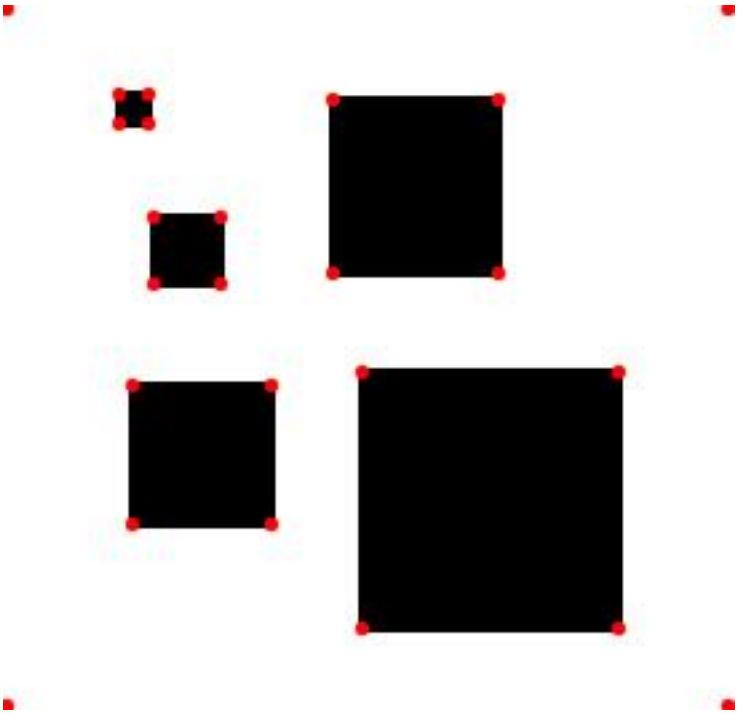

Figure 4: Output of Harris Corner Detector (result).

Discussion of Results

Harris Corner Detector follows concise steps for detection. In first step, it takes derivative of image with respect to horizontal and vertical direction. After that it takes second derivative with respect to images and after that it implements gaussian filter. Last step is to implement cornerness function. After these steps output obtained.

From experiment results, I can clearly say that my output gives accurate results and I am satisfied with my work done.

NOTE: Before passing through second question, I want to point out an important detail. I used HarrisCornerDetector function in all questions. That's why I put this function in a separate file named *HarrisCornerDetector.m*.

Question 2

Matching features between two different images is the subject of this question. From previous question, there are already Harris Corner Detection function exists. Additionally, functions implemented in this question already given in assignment paper. From these information, I followed steps that is given from instructor.

Firstly, I implemented an image that have two different aspects. These images provided below (Figure 5).

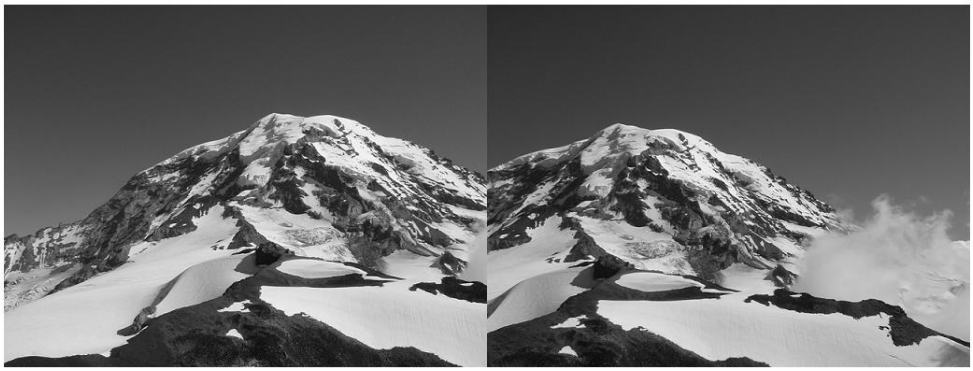

Figure 5: Pair of images used for this question.

After that Harris Corner Detector defined and features extracted. Output of Harris Corner Detector function for both questions provided below (Figure 6).

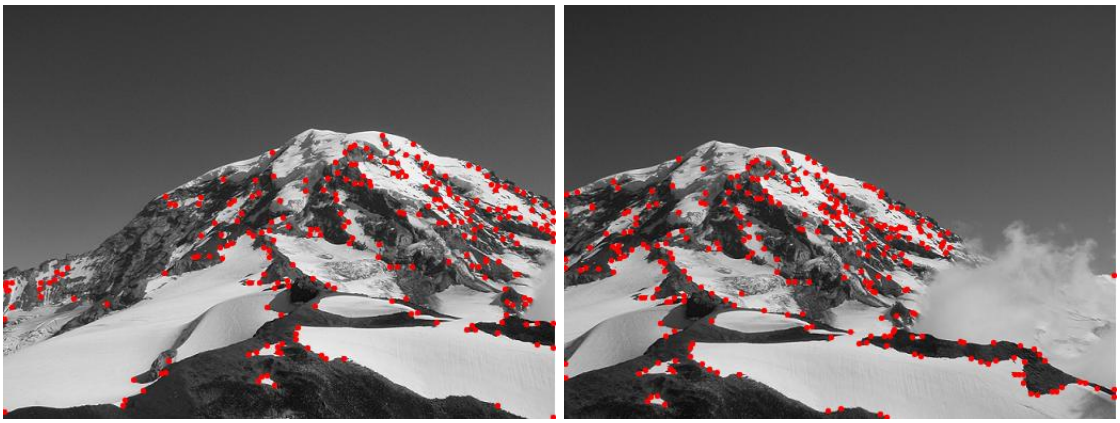

Figure 6: Corners of both images.

Feature matching follows these steps. After feature matching, matched features picked from that data. Output obtained from this process provided below (Figure 7).

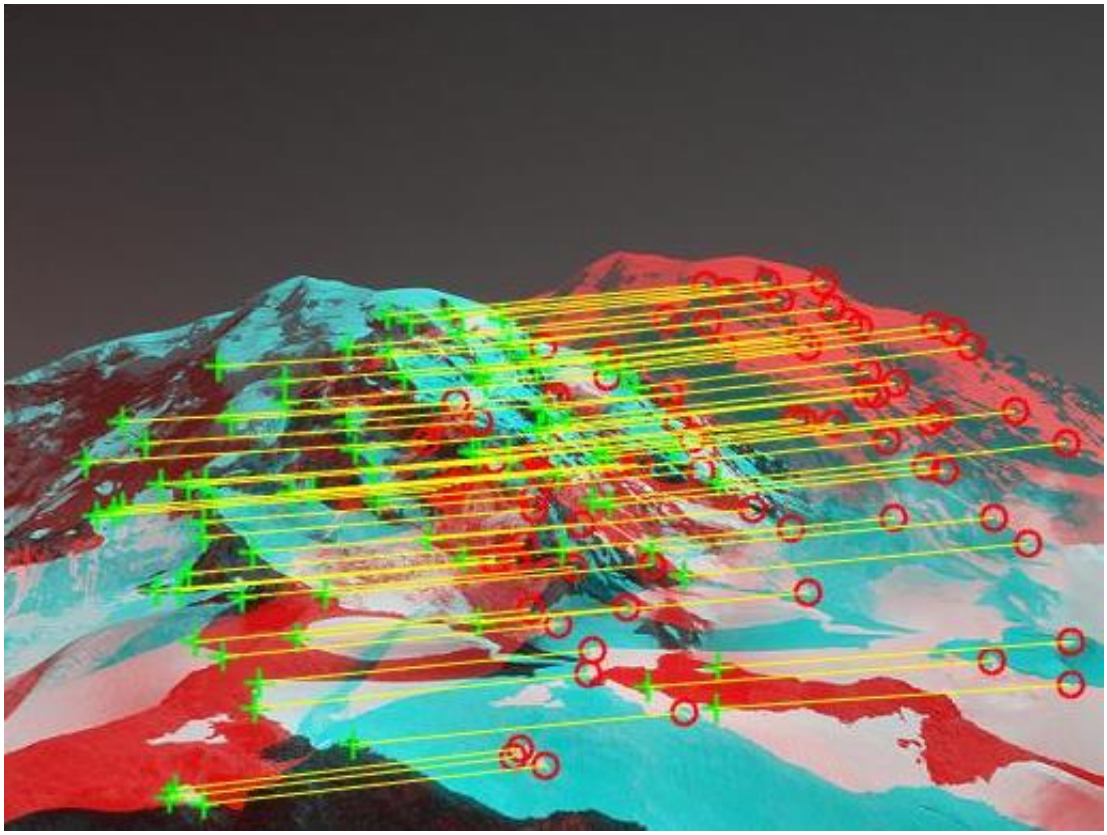

Figure 7: Output

Discussion of Results

To conclude, there was only several simple steps to apply for this question. Its aim is to help students to percept how feature matching has done. I strongly believe that I have achieved this task with success and my results supports my belief. Hence, this assignment is very practical to observe feature matching and its steps.

Question 3

Last question covers homography calculation between images with RANSAC method. This question expects RANSAC algorithm with supporting functions. So that, I prefer to explain functions rather than to explain whole code. Expected functions summarized below.

Functions Used

First function is projection function that takes a point in an image and homography to produce projected point. Function explanation and pseudocode of function provided below.

$$[x_2, y_2] = Project(x_1, y_1,\ h)$$

- x_1, y_1 : Project points.
- h : Homography matrix.
- x_2, y_2 : Projected points.

Algorithm 2 Project

Require: x_1, y_1, h
Ensure: x_2, y_2
$res \leftarrow h * [x_1; y_1; 1]$
$x_2 \leftarrow res(1)$
$y_2 \leftarrow res(2)$

Second function is ComputeInlierCount taking homography matrix, matching points, number of matching points and inlier treshold number. Function definition provided below.

$$inlierNum = ComputeInlierCount(h,\ matches,\ numMatches,\ inlierThreshold)$$

- h : Homography matrix.
- *matches* : Matching points between two images.
- *numMatches* : Number of matching points between two image.
- *inlierThreshold* : Threshold numbers.

It takes these inputs to calculate threshold numbers. Firstly, it takes random numbers and after that it makes calculations to count inliers. If a point is not inside this inlier, it eliminates that point.

Last function of this question is RANSAC implementation. RANSAC is the key function for this question because it covers both of the functions above. It includes complex steps. Function definition and pseudocode of function given below.

[hom, homInv]= RANSAC (matches, numMatches, iterations, inlierThreshold)

- *matches* : Matching points between two images.
- *numMatches* : Number of matching points between two image.
- *iterations* : Iteration number.
- *inlierThreshold* : Threshold numbers.
- *hom* : Homography matrix.
- *homInv* : Inverse of homography matrix.

Algorithm 3 RANSAC

Require: matches, numMatches, iterations, inlierTreshold
Ensure: hom, homInv
  $ha \leftarrow eye(3,3)$
  for i $\leftarrow$ 1:iterations **do**
    $[p1, p2] \leftarrow randsample(numMatches, 4)$;
    $h \leftarrow findHomography(p1, p2)$
    $inlierNum \leftarrow ComputeInlierCount(h, matches, numMatches, inlierThreshold)$
    if inlierNum $>$ inlierTreshold **then**
      $ha \leftarrow h$
      $inlierTreshold \leftarrow inlierNum$
    end if
  end for
  $hom \leftarrow ha$
  $homInv \leftarrow inv(hom)$

In addition to functions expected, I wrote additional funcitonal for efficiency. Inside ComputeInlierCount and RANSAC, I need to find homography for the flow of algorithm. Because of that, i wrote a supporting function FindHomography that takes matching points from two different images to produce a homography matrix. Function definition and pseucode provided below.

$$[h] = findHomography(p_1, p_2)$$

- h : Homography matrix.
- p_1 : First image.
- p_2 : Second image.

Algorithm 4 FindHomography

Require: p_1, p_2
Ensure: h
  $A \leftarrow []$
  for i = 1: length(p_1) **do**
    $x_i \leftarrow p_1(i, 1)$;
    $y_i \leftarrow p_1(i, 2)$;
    $x_j \leftarrow p_2(i, 1)$;
    $y_j \leftarrow p_2(i, 2)$;
    $t \leftarrow [x_i, y_i, 1, 0, 0, 0, (-x_j * x_i), (-x_j * y_i), (-x_j); 0, 0, 0, x_i, y_i, 1, (-y_j * x_i), (-y_j * y_i), (-y_j)]$
    $A \leftarrow [A; T]$
  end for
  $h \leftarrow A * A'$
  $h \leftarrow eig(h)$
  $h \leftarrow reshape(h, 3, 3)$
  $h \leftarrow h'$

With the help of these functions and algorithm design, output that has huge accuracy obtained. Comparison of output without RANSAC and with RANSAC provided below (Figure 8).

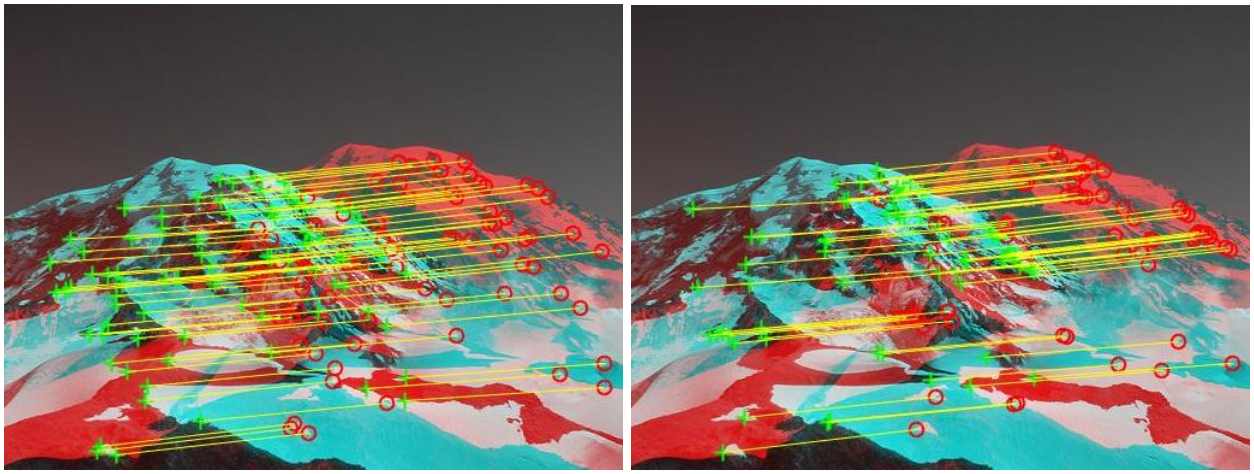

Figure 8: Result without RANSAC vs. Result with RANSAC

Discussion of Results

This question was an absolute advanture. It covers comprehensive and deep mathematical steps. In order to acquire accurate output, pure and true implementation is necessary for all steps of RANSAC and other functions. I believe that I became successful to get this accurate output by mentioning Figure 8 above. It shows clear example of how RANSAC approach affects accuracy of output. Hence, I believe that RANSAC implementation is very successful and earned a good output.

Conclusion

To sum up, three differrent question asked in this assignment in order to reinforce knowledge of students for two different subjects. First question expects an appropriate feature detection algorithm called Harris Corner Detection. Second question wants an example of Harris Corner Detection algorithm that has written in previous question to make feature matching. Last question wants RANSAC algorithm to reduce outliers and create an accurate output. After implementing all questions, good outputs obtained for me and I believe that I got experienced too. Hence, this assignment is a great application to percept fundamental concepts of feature detection and matching, and image alignment.