Homework 3

Thomas Buchegger
Introduction to Signal and Image Processing

May 8, 2019

Contents

1 Introduction

This assignment is about the RANSAC algorithm and texture synthesis with SSD.

RANSAC (random sample consensus) is a rather simple algorithm designed to cope with a large proprtion of outliners in input data. It is a resampling technique which generates different solutions by using the minimum number of data points required to estimate the underlying model parameters. Unlike other estimation techniques which use as much data as possible, RANSAC only uses the smallest possible set and proceeds to enlarge this set with consistent data points.

Texture Synthesis is used to fill an empty section of an image with a reconstruction based on a SSD (Sum Squared Difference) approach. This means, the algorithm selects pixel on be boundary of the to be filled space and then calculates the SSD. So a pixel from the sample set is used to substitute a pixel from the empty space.

2 Ransac

2.1 Introduction

As mentioned, RANSAC is a method for iteratively estimate the parameters of a mathematical model from a set of data with outliners. The goal of this assignment was to detect lines on an image and plot them. The algorithm works as follows:

1. Select randomly minimum number of points to determine the model parameters
2. Solve for the parameter of the model
3. Determine how many points from the set of all points fit with a predefinded tolerance
4. If fraction of number of inliners over the total number points in the set exceed a predefined threshold, re-estimate model parameters using all identified inliners and terminate
5. Else, repeate steps 1 to 4 (N iterations)

The number of iterations should be chosen high to ensure that at least one set of random samples does not include any outliners.

2.2 Exercises

2.2.1 edge_map

To get an edge map out of the picture, I used a canny edge algorithm with sigma 1.5. The results where satisfying and calculation time was within acceptance.

2.2.2 fit_line

A linear function has the two parameters m and c. The task was to get the slope of the fitted line m and the y-intersect of the fitted line. This could be achieved with subtracting the coordinate points. To avoid division by zero, we add some noise ϵ.

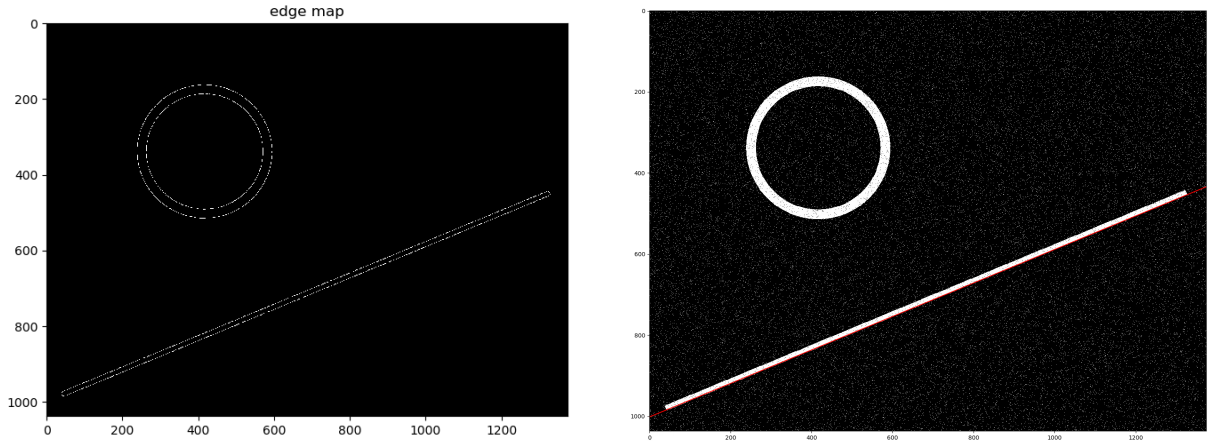
2.2.3 point_to_line_dist

As for the distribution, I used the following formula: $dist = \frac{|m \cdot x_0 - y_0 + c|}{\sqrt{m^2+1}}$

2.3 Results

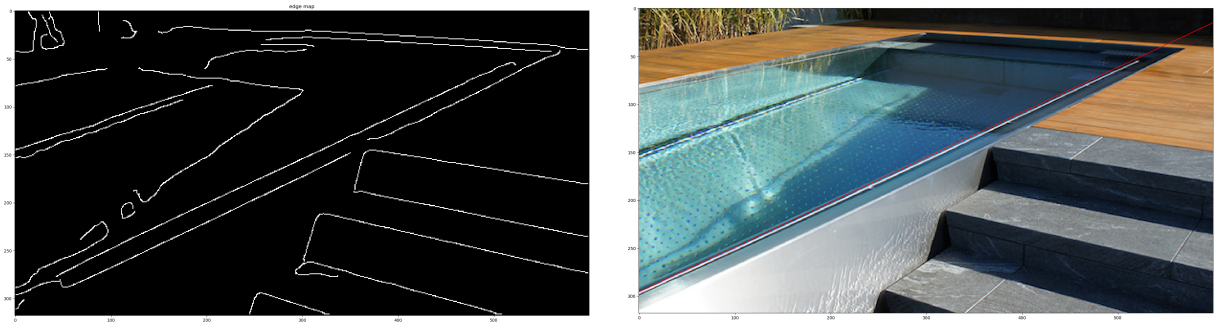
The following pictures are the results with the ransac algorithm. The used parameters are:

- Iterations = 500
- Threshold = 2
- n samples = 2



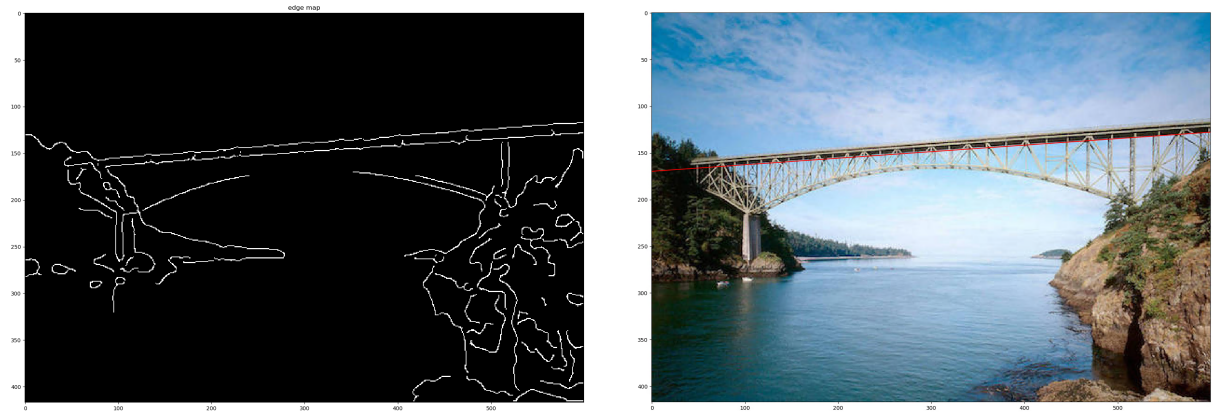
(a) edge map (b) result

Figure 1: Calculations of synthetic.jpg



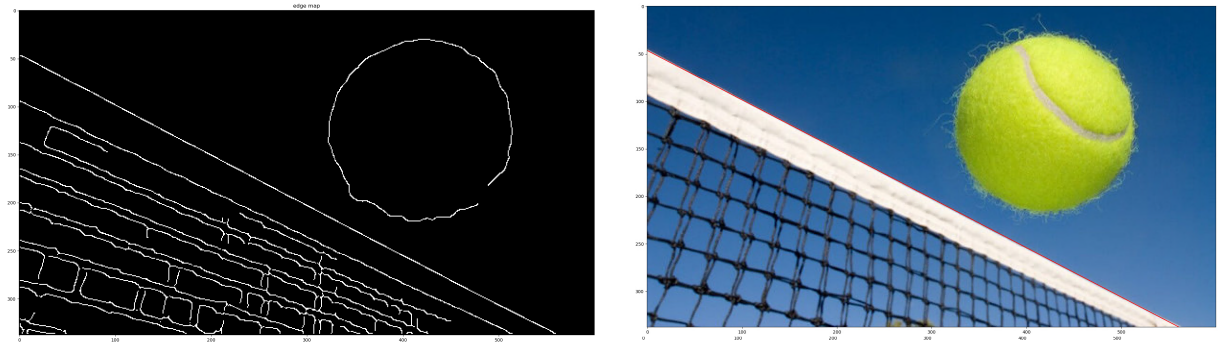
(a) edge map (b) result

Figure 2: Calculations of pool.jpg



(a) edge map (b) result

Figure 3: Calculations of bridge.jpg



(a) edge map (b) result

Figure 4: Calculations of tennis.jpg

The results are identical with the to be achieved goals in the assignment. Canny with sigma 1.5 was a good choice, as it returns well recognizable edges to be detected by the RANSAC algorithm.

3 Texture Synthesis

3.1 Introduction

The algorithm writen for this assignment tries to predict how to fill the empty space in the image based on a sample texture from the image itself. This is done by firstly analyzing every pixel at the corner of the missing part and with the weighted sum of all the neigbourhood pixels. The neigbourhood is given by the patch size. The SSD determines which pixel can be used to fill the missing space in the image.

3.2 Exercises

3.2.1 compute_ssd

For all possible locations of the patch in texture_img computes the SSD for all pixels where mask is 0.
It was difficult to achieve a satisfying performance. Different approaches were tried before finding the best result in a single loop solution. This might seem obvious as a nested loop always requires more time, but here it was difficult to break the code down to a single one.

3.2.2 copy_patch

This task was not as difficult as the previous one. The best results could be achieved with the image being in range 0 to 255.

3.3 Results

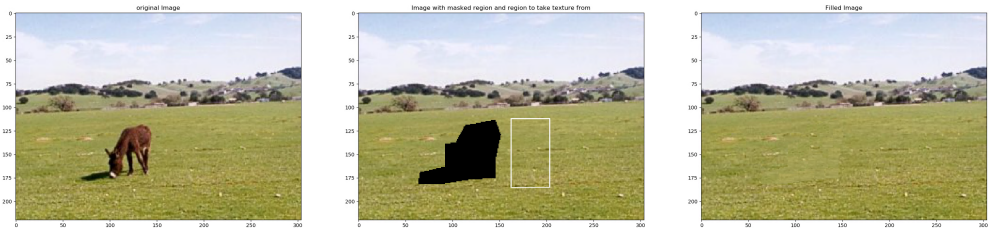


Figure 5: With patch of size 10

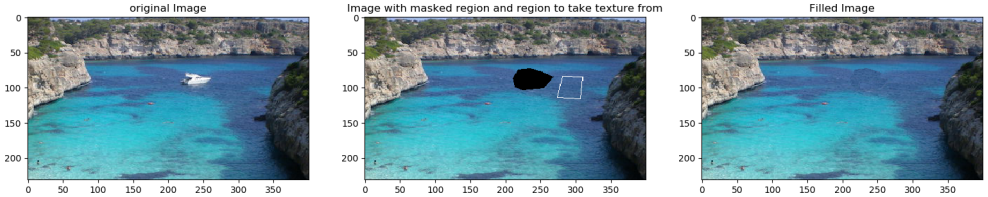


Figure 6: With patch of size 10

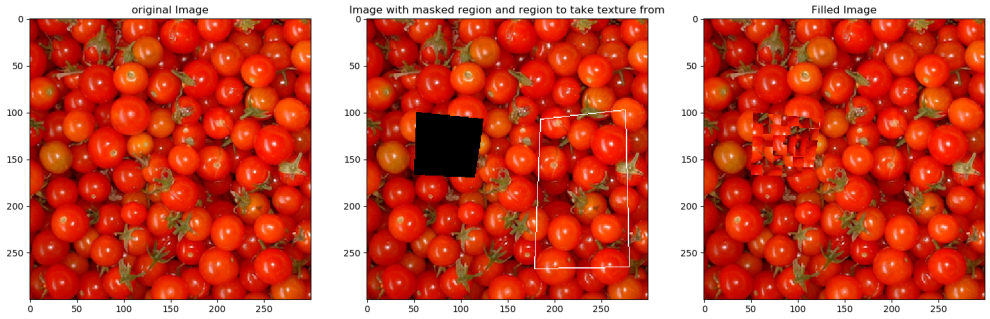


Figure 7: With patch of size 10

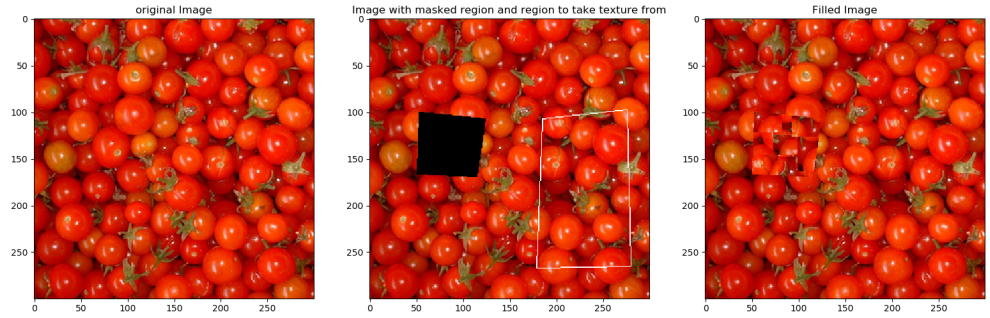


Figure 8: With patch of size 20

Both for the donkey and the yacht image, a patch size of 10 yields really good resulsts. In the tomato picture, however, one can clearly see that the patterns do not match. Trying a patch of size 20 for the tomato returns a slightly better result, though not satisfactory.
The results for donkey and yacht are better, because the textrue is consistent throughout the empty space as well as in the patch area. Therefore it doesn't really matter which part of the patch is taken to fill.
What is different to the tomato picture is, that there is no consistent texture which makes it a lot harder to fill the empty space with a good result.

4 Conclusion

It was an interesting assignment, especially since before I haven't heard of the RANSAC algorithm or SSD. So to not only get to know them, but also implement in a project surely was difficult though a fun thing to do.