Assignment 3

Nalet Meinen
Introduction to Signal and Image Processing

May 8, 2019

Contents

1 Introduction

In this assignment, two methods will be analyzed that are both iterative methods. First RANSAC will be presented, which means random sample consensus. The algorithm is a good general parameter estimator approach and can get alow well with large proportions of outliners in input data. Imagine a line of points which we will draw a line if one point is of charts most likely the line will get in this direction. RANSAC will not put much weight on those outliners.

With texture synthesis, the goal is to fill an empty space in an image and reconstruct that with an SSD approach. SSD means Sum of squared differences. With the SSD approach, the algorithm select a pixel on the boundary of the space to fill and calculates the SSD. With this result, a pixel from a sample site can then be used to substitute one pixel from the empty space.

2 Methods

2.1 RANSAC

RANSAC (RANdom SAmple Consensus) is an iterative method for estimating the parameters of a mathematical model from a set of data with outliners. In this assignment, the goal is to detect sharp lines within images and plot them on it. The algorithm starts with a minimal subset of data points that are needed to fit the model randomly. Second, points with some distance threshold from the model are a consensus set. These points can then be used to determine if the model is supported best. In the first iteration, the first match is stored. If a better is found in a later iteration, the later one is stored. With enough interactions, a good solution can be found.

2.2 Texture Synthesis

The texture is a detail in an image that is at a scale too small to be resolved into its constituent elements and at a scale large enough to be apparent in the spatial distribution of image measurements. So the texture is a kind of a repetitive way to align images at each other but it's not. The goal of texture synthesis is not to see images aligned with each other. The viewer of this image should not see, that the whole image consists of multiple small ones. In complex textures, repetitive artifacts should not be visible e.g. for a people crowd.

The algorithm tries to predict how to fill the missing data in an image. Giving an empty space in an image, space should be filled with a sample texture from the image itself. To achieve this the algorithm starts at the corner of the missing data an analyzing every pixel with the weighted sum of all the pixel in the neighborhood. The size of the neighborhood is given by the patch size. The SSD (Sum of Square Differences) determines which pixels, or a variety of pixels, can be used for filling in the missing spot in the image.

3 Results and Discussion

3.1 RANSAC

3.1.1 fit_line

The function if the mathematical line back as a linear mathematical function consisting of the parameters m and c. To ensure that the line is not horizontal, we subtract the coordinate points. If the difference would be indeed zero, we use the value ϵ to make it not horizontal.

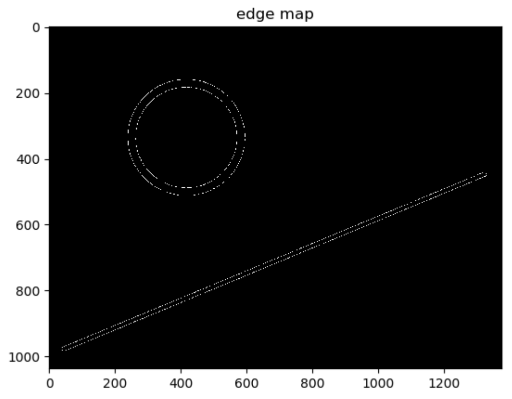
3.1.2 point_to_line_dist

The mathematical formula is used $d = \frac{|m \cdot x_0 - y_0 + c|}{\sqrt{m^2+1}}$.

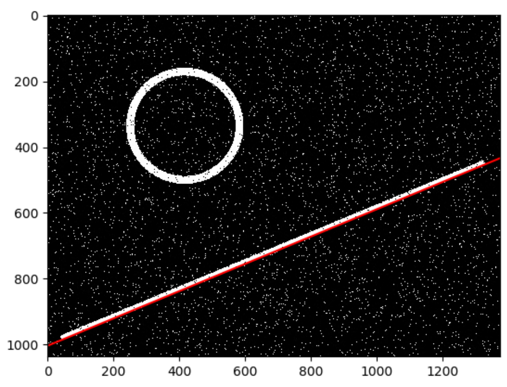
3.1.3 edge_map

With the knowledge from the previous assignments, a sigma of 3 is used. It was enough to get not any disturbing artifacts and did not push the calculation time up as lower sigma values would.

3.1.4 Iteration



(a) canny of image

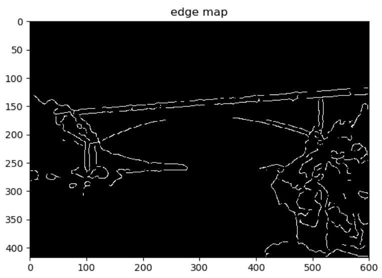


(b) result line

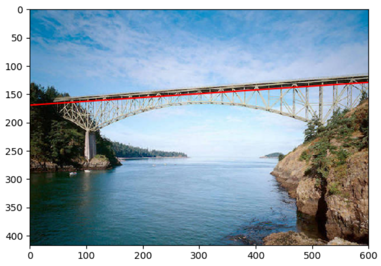
Figure 1: Calculations of synthetic.jpg

With this low point count, it was enough to match a line. Also, the calculations were fast.

3.1.5 Results

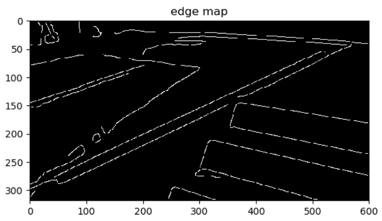


(a) canny of image

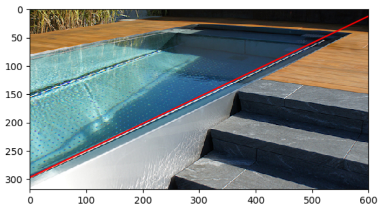


(b) result line

Figure 2: Calculations of bridge.jpg

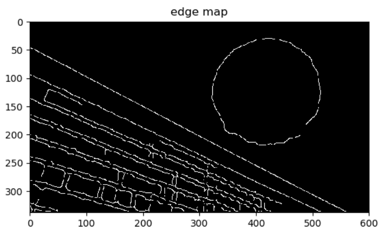


(a) canny of image

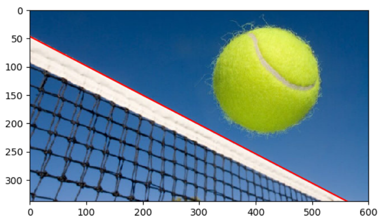


(b) result line

Figure 3: Calculations of pool.jpg



(a) canny of image



(b) result line

Figure 4: Calculations of tennis.jpg

It is impressing with sigma 3 resulting in a small number of points, that so an accurate solution can be achieved. RANSAC is indeed a robust method. That is also why it can be used for real-time applications, such as self-steering cars.

3.2 Texture Synthesis

3.2.1 compute_ssd

It is in the computation of the SSD where a performance bottleneck can be easily created. Using the build in the function of the Numpy library and using only one loop is the key to success. Also very important is the preparation of the data, as it is for the whole exercise.

3.2.2 copy_patch

Implementing this function was quite easy, but better results are achieved if the mask is staying in the range from 0 - 255, even if only the min and max value existed.

3.2.3 Results

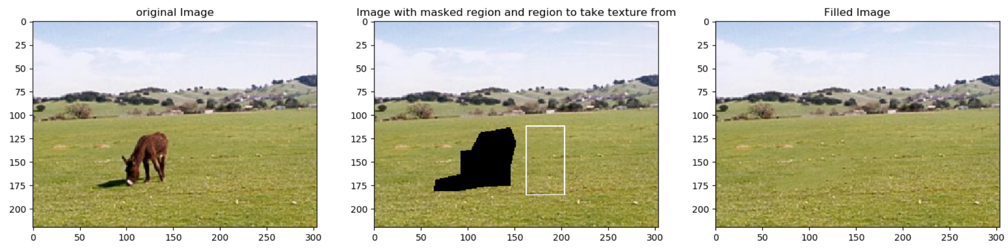


Figure 5: Calculations with donkey

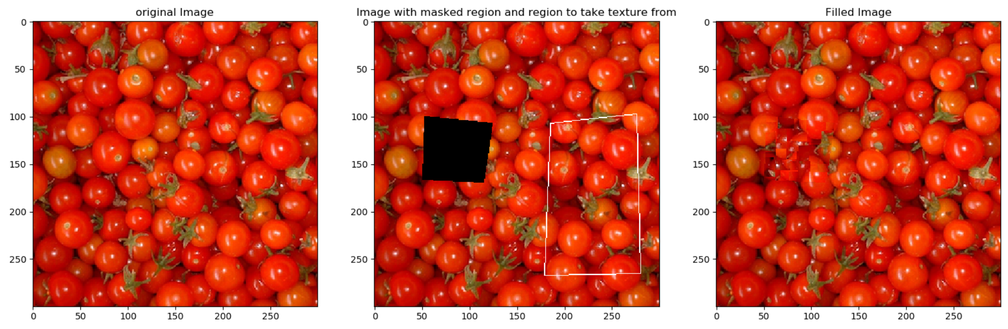


Figure 6: Calculations with donkey

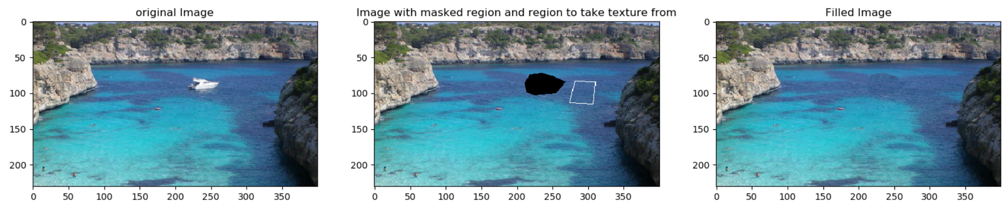


Figure 7: Calculations with donkey

Changing the half patch size variable to 5 and 10 did not really help to achieve better results. However, choosing the best texture patch is based on a random gauss, so the image looks from calculation to calculation a bit better or worse. With simple textures with less detail, the results are amazingly good. Several optimizations could be made in the choosing of the textures samples for the empty spot in the image.

3.3 Conclusion

The work in this assignment was demanding and made a lot of fun. Only if everything is comprehended a solution could be rudimentary created. Lot of work was made in the preparation of the incoming data for the right structures for the calculations.

References

[1] Alasdair McAndrew
A Computational Introduction to Digital Image Processing (2015). (English)

[2] Alexei A Efros and Thomas K Leung.
Texture synthesis by non-parametric sampling. In Computer Vision 1999. The Proceedings of the Seventh IEEE International Conference on, volume 2, pages 10331038. IEEE, 1999. (English)

[3] Random sample consensus
https://en.wikipedia.org/wiki/Random_sample_consensus (2019). (English)