Homework 3

Stefan Weber

Introduction to Signal and Image Processing

May 06, 2020

1 RANSAC

The goal of the random sample consensus (RANSAC) algorithm is to find the parameters of a mathematical function which has the most inliers. To find the best fitting line the parameters m and c have to be determined; therefore two points must be chosen.

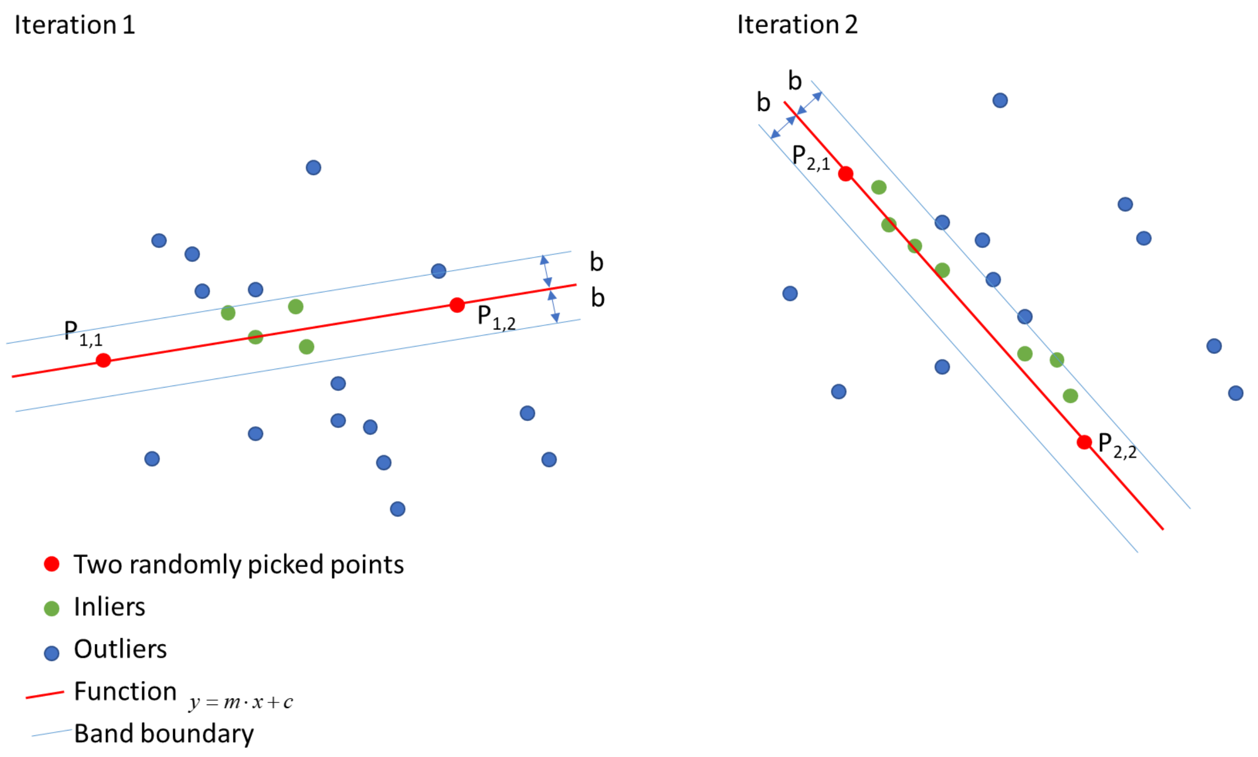


Figure 1: Graphical illustration of two steps of the RANSAC algorithm.

Figure 1 illustrates two iteration steps of the RANSAC algorithm. To find edges in a given image, this image has to be pre-processed. The RANSAC algorithm to find the function of the best fitting edge includes following steps:

1. Pre-process the image. This includes the change from RGB to greyscale, filtering the image and computing canny edge detection. The output is an edge map of the image.
2. Randomly choose two points from the edge map.
3. Fit-in a line $y = m \cdot x + c$ between the two points.
4. Compute all distances from the leftover points to this line: $d_i = \frac{|m \cdot x_i - y_i + c|}{\sqrt{m^2+1}}$, if $d_i < b$ → count it as inlier.
5. Repeat step 2-4 n-times and update the parameters m and c, if more inliers were counted.

1.1

The parameters m and c were found as follows: $m = \frac{\Delta y}{\Delta x}$, $c_i = y_i - m \cdot x_i$

To ensure no division by 0, the smallest possible floating-point number is added to the denominator.

1.2

The shortest distance from a point to a line is perpendicular to the line itself and can be calculated as follows:

$$d_i = \frac{|m \cdot x_i - y_i + c|}{\sqrt{m^2+1}}$$

1.3

To compute the edge map the image is pre-processed with a gaussian filter to smoothen the image and reduce noise. With a high sigma less details are left in the image. If the sigma is too high no edges can be detected because the intensity of the gradients is below the threshold. The computing speed is faster with a higher sigma because less distances to the line must be computed.

Different sigma values were tried. For all images, a sigma of 3 was finally chosen. The computing time was exactable, and the results were satisfying.

1.4 Results

In all figures a sigma of 3 was chosen and 500 iterations were performed.

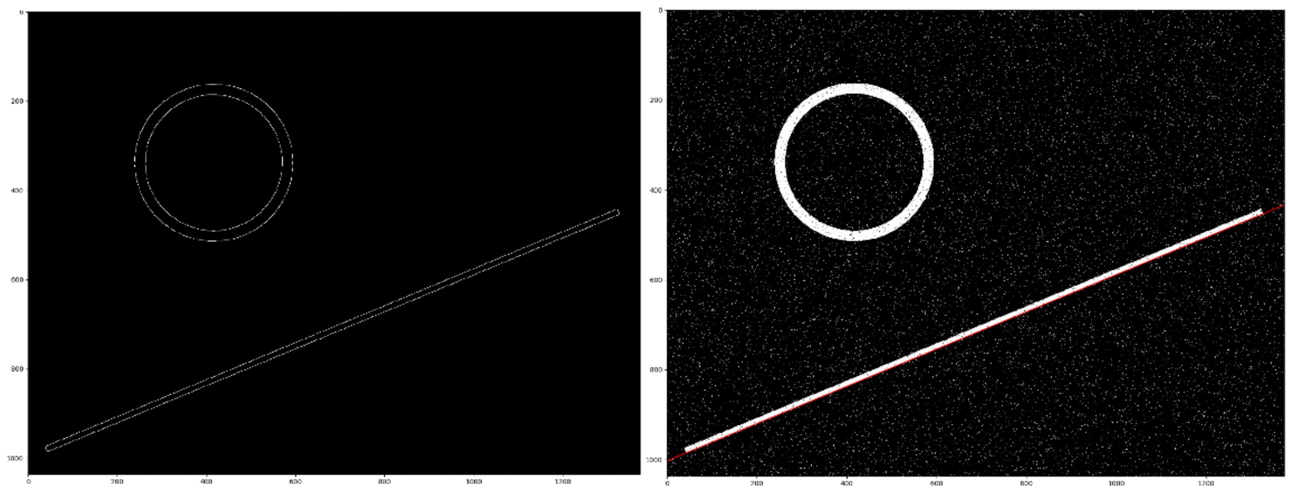


Figure 2: Left: Edge map, right: Original image with the found line

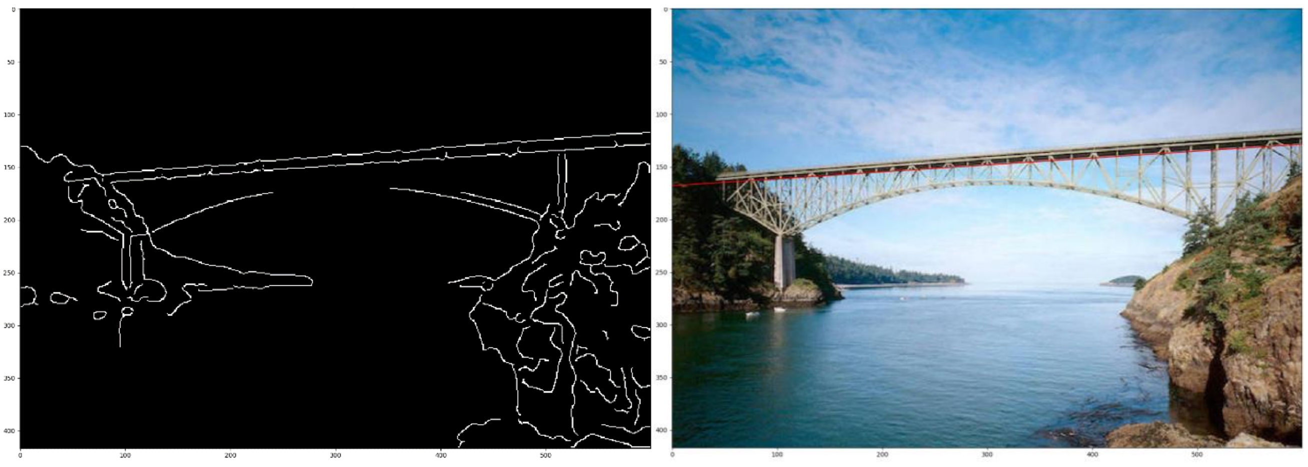


Figure 3: Left: Edge map, right: Original image with the found line

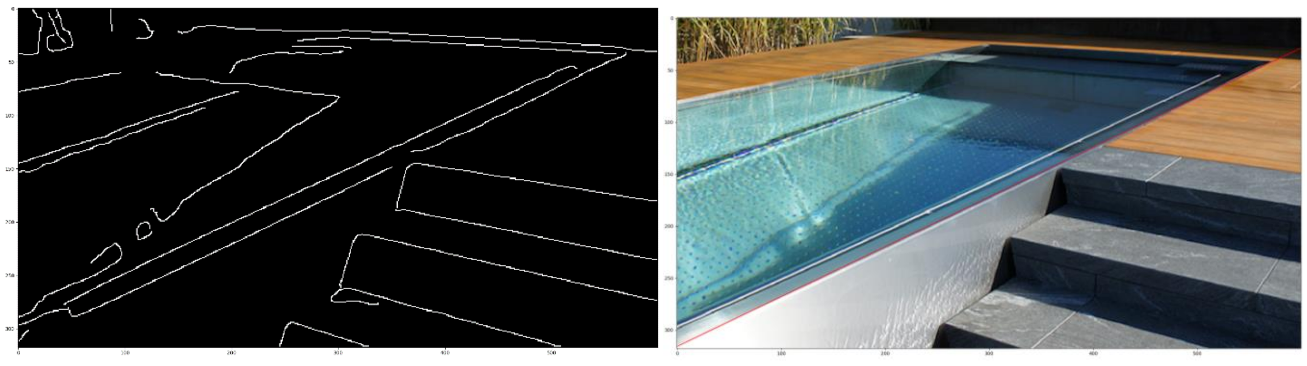


Figure 4: Left: Edge map, right: Original image with the found line

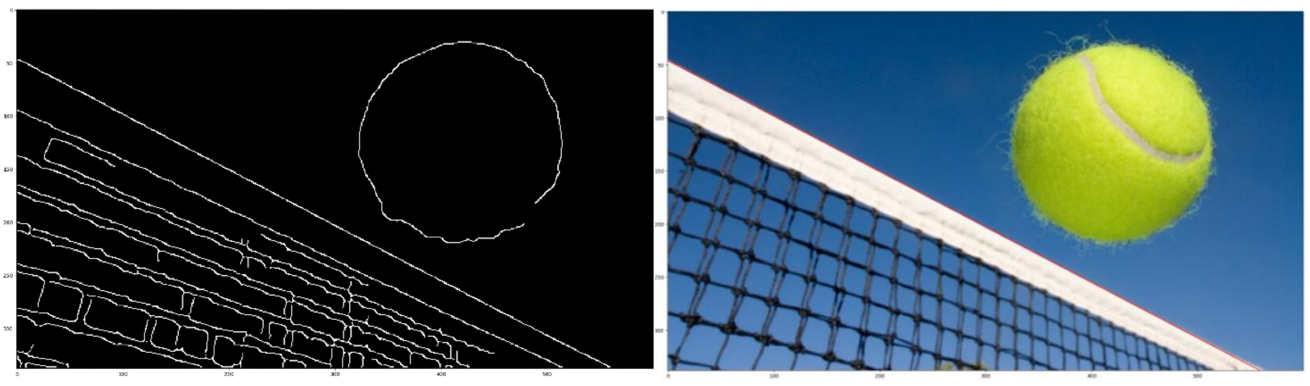


Figure 5: Left: Edge map, right: Original image with the found line

All Figures show good results. The applied filters reduce the number of points in the image drastically and the longest edge in the image can be found fast.

2 Texture Synthesis

The steps of texture synthesis are shortly explained.

1. Selection of a point *r* on the boundary of the masked region.

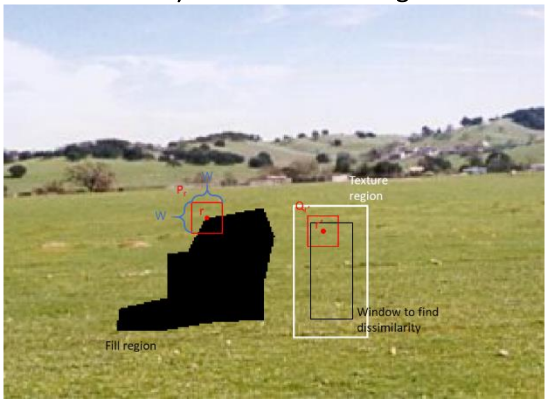
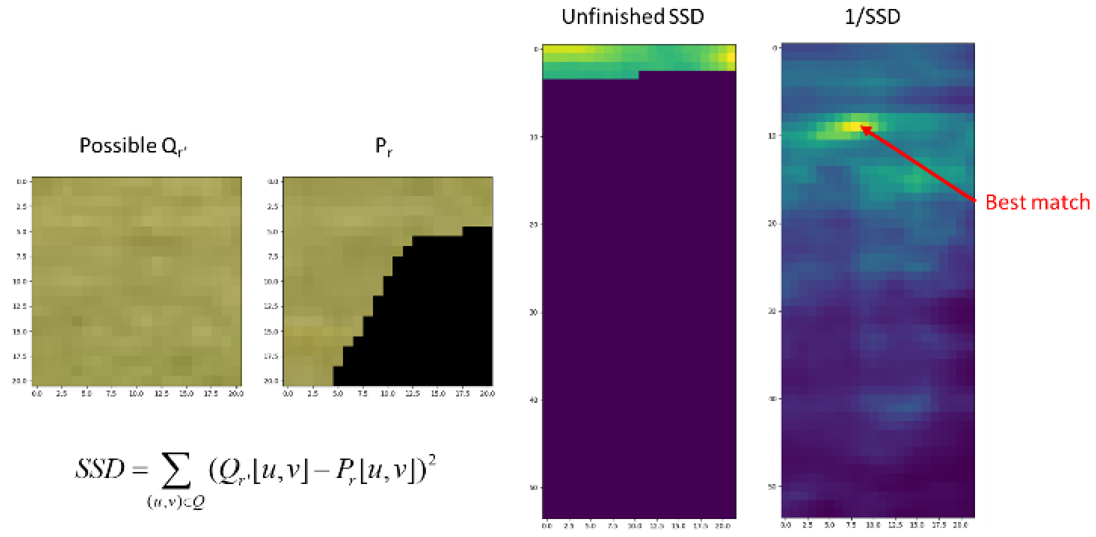


Figure 6: Donkey.png with the to-be-filled region in black, the texture region (white square), the region where the SSD can be computed (black square), the patch P_r, and the patch $Q_{r'}$

2. Create a patch P_r of size WxW with *r* as its centre. The aim is to find another patch Q_{r}' in the texture region centred on *r*' and of equal size, such that a dissimilarity value of the SSD (P_r; Q_{r}') is minimized. Unfilled regions are ignored when computing the SSD.



$$SSD = \sum_{(u,v)\in Q} (Q_{r'}[u,v] - P_r[u,v])^2$$

Figure 7: Illustration of one step of the SSD and the final computation result, 1/SSD, with the best matched pixel. On the bottom left the formula to compute the SSD is displayed.

3. Update the initial image with the found patch $Q_{r'}$.

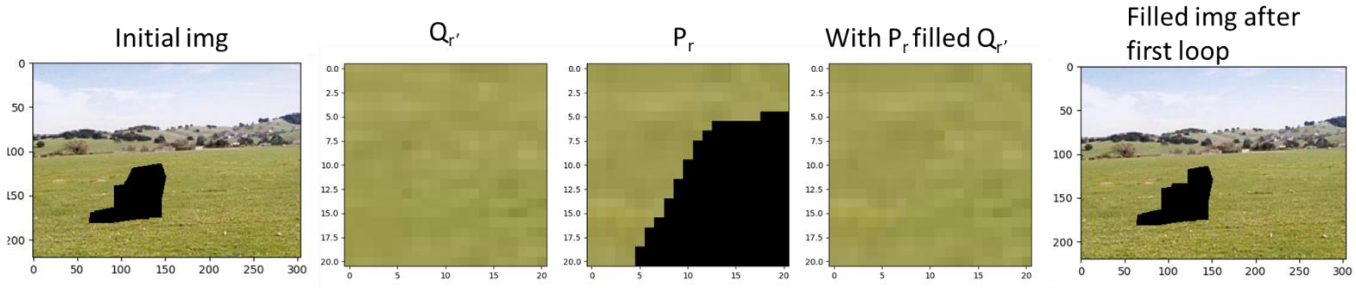


Figure 8: All necessary steps to fill the Image with the patch $Q_{r'}$.

4. Update the fill region and the fill region edge.

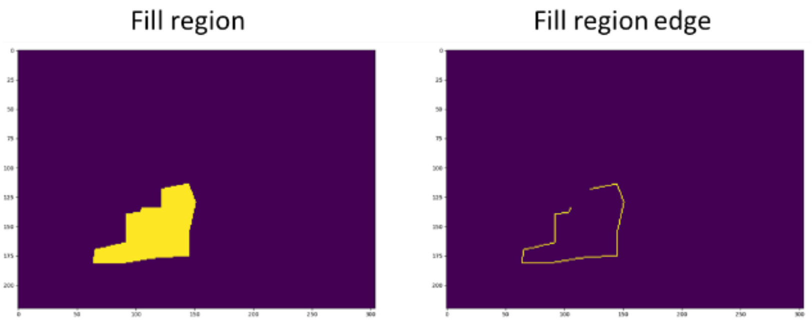


Figure 9: Updated fill region and fill region edge after one step.

5. Start over until fill region is completely filled.

2.1

SSD is already explained in the paragraph above and is illustrated in Figure 7.

2.2

How to replace values in the masked region with the best match is already explained in the paragraph above and illustrated in Figure 8.

2.3

The results of the texture synthesis are illustrated and shortly discussed in this paragraph. The texture synthesis was performed on three different images.

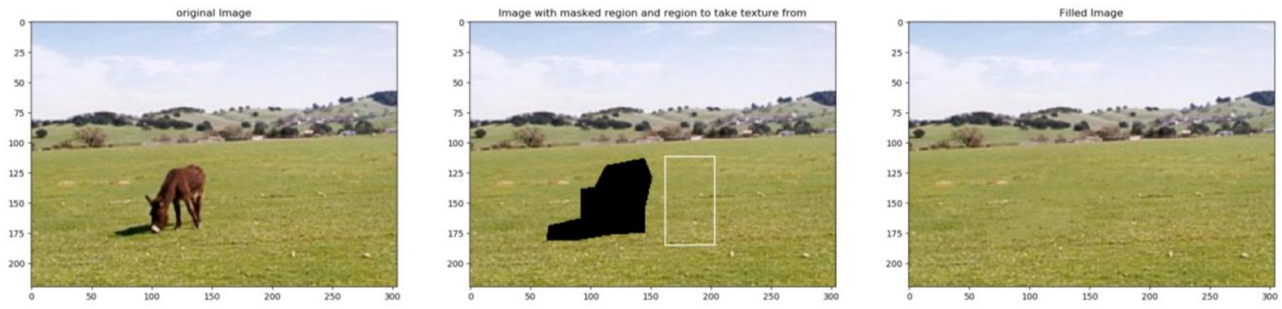


Figure 10: Result of the texture synthesis with a patch half size of 10 performed on the picture donkey.png.

Figure 10 shows really good results. It needs a close look and the knowledge which part was synthesised in order to find the boundaries where the image was filled with which texture region.

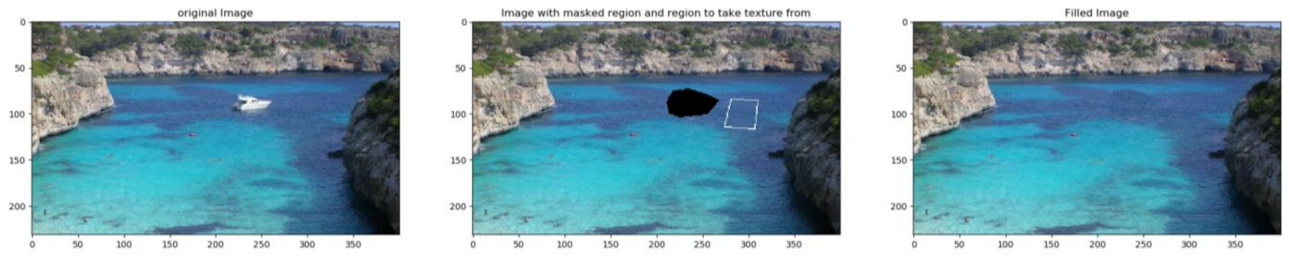


Figure 11: Result of the texture synthesis with a patch half size of 10 performed on the picture yacht.png.

The results in Figure 11 are also really good but not as good as in Figure 10. On the top of the patch a clear unrealistic boundary can be seen. This happens because the brighter part of the water does not appear in the texture region.

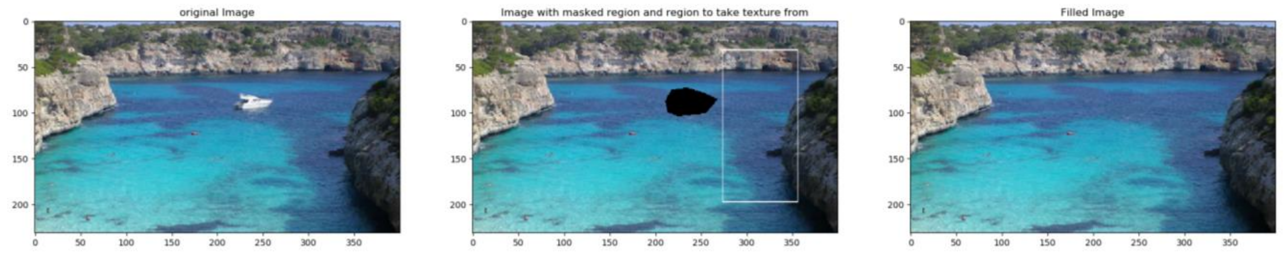


Figure 12: Result of the texture synthesis with a patch half size of 5 performed on the picture yacht.png *and with an increased texture region.*

To remove this clear boundary the texture region can be increased. As one can see in Figure 12 the result with an increased texture region shows a better result than in Figure 11.

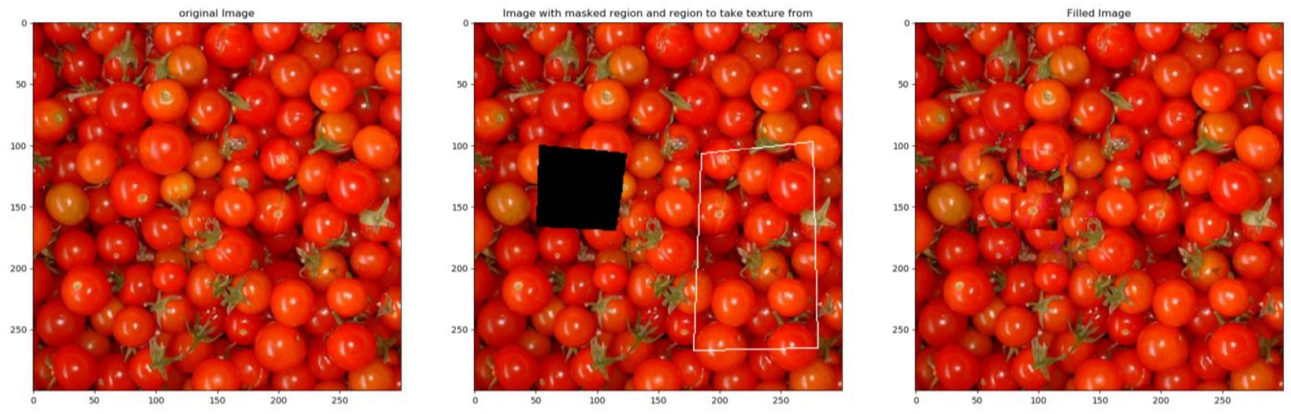


Figure 13: Result of the texture synthesis with a patch half size of 30 performed on the picture tomato.png.

The algorithm applied on the *tomato.png* fails as can be seen in Figure 13. The algorithm performs well on stochastic textures and less good on repeated pattern. The *tomato.png* includes stochastic and repeated textures, which is hard to handle with sharp boundaries. Better results may be achieved with an implemented minimum error boundary cut.