Being an Art Restorer

L. Bundrock, Y. Cao, T. Dombrovskij, T. Emunds, A. Kristof, S. Tafazoli

&

Dr. Andreas Kleefeld

Abstract
In the world of art, there exist some valuable artworks which get deteriorated over time and in these cases, a professional art restorer is hired to reconstruct the corrupted parts of the image who follows a certain procedure for restoration that takes a lot of effort and is expensive. In order to eliminate this issue, this process is modeled and automated using computer, which is the goal of this project.

1 Motivation

There exist several paintings and masterpieces from famous artists which have been deteriorated partially and for example in the museum world, they are very valuable. Usually, in the traditional way, a skilled art restorer is hired to reconstruct the corrupted parts of the painting; this process of fixing the missing areas of the image is called inpainting which is time consuming and expensive. For that reason, the digitalization of this process could play a significant role to save time and costs which is the task in this group project. In this project, our advisor assigned us the task of restoring the missing parts in the “The Starry Night” work of art by Van Gogh seen in Figure 1.

Figure 1: “The Starry Night” by Van Gogh.

2 Introduction

The usual methodology and ideas that a professional art restorer would follow are to use the global image in order to fill in the corrupted areas in the image. In that way, the unity of the artwork will be maintained. Another main idea is that the structure of missing parts should be continued by the surrounding pixels, e.g. the contour lines in the available neighboring pixels should be followed. Another challenge is the restoring of colors in the missing areas, since there are a wide variety of color shades available. The final and most complicated challenge is to add texture to the missing

area that is a more detailed and complicated task. Considering this normal process of art restoration, the Van Gogh's image shown in Figure 1 is a really complex challenge for inpainting.

2.1 Problem Statement

The task of this project is to model the process of art restoration and implement it using MATLAB. There already exist several inpainting algorithms like Laplace method that uses the PDEs which could be implemented. In this task, the corrupted image is given and it is assumed that the position of corruptions is known. The basic challenge of this project would be to reconstruct missing parts in a grayscale image. The real challenge is to extend the procedure to colored images like the Van Gogh's image shown in Figure 1.

3 Methods

In order to start making the art restoration digital, we first came up with the idea of using the average value of the pixels surrounding the corrupted region. So, first a local part of the image around the missing area, including the missing region, is considered and then using a 3×3 structural element, the local information is averaged and the middle pixel of the structural element is replaced by the average value.
Unfortunately, the results were not satisfying in this case and resulted in a full blurred image, hence mentioning the results is avoided here. The next idea was to use the median value of the structural element instead of the average value which is explained in the following.

3.1 Median Filter

The idea of using the median value is that a local part of the image is considered that includes the missing area, then a 3×3 filter moves from the upper left corner through the local image and in each step, the median of the 9 values is substituted in the middle pixel. This process is shown in Figure 2, where the median of the shown values corresponds to 213; the gray region represents the defected region, while the white area corresponds to the available pixels of the local image. This

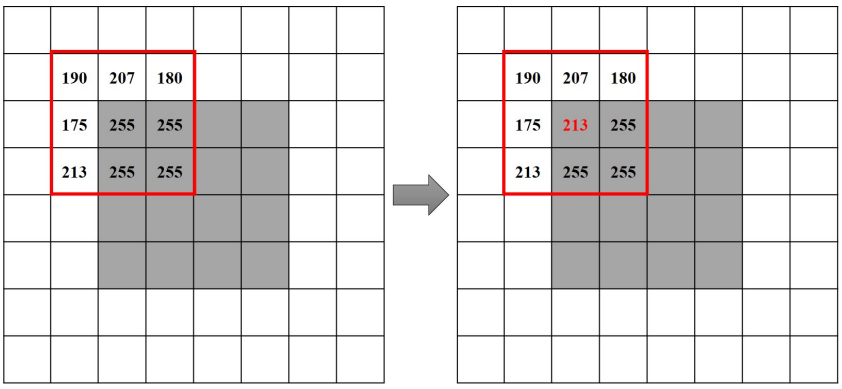


Figure 2: Median$\{175, 180, 190, 207, 213, 255, 255, 255, 255\} = 213$.

method seems to work well at the first glance, but the problem occurs when the filter is moved further towards the corrupted region, where the median of the values remains constant and equal to 255, i.e. only the corners of the image are filled with a value other than 255. To fix this issue, we applied the filter several times and each time increased the filter size to avoid filling in the corrupted regions with a constant value. Afterwards, the new filter is applied to a sample grayscale image. The results are shown in Figure 4. The left image in the figure represents the whole viewpoint of the corrupted shoulder part, the right upper image shows the same image from a closer viewpoint and the right lower image shows the result of restoration after applying the modified median filter.

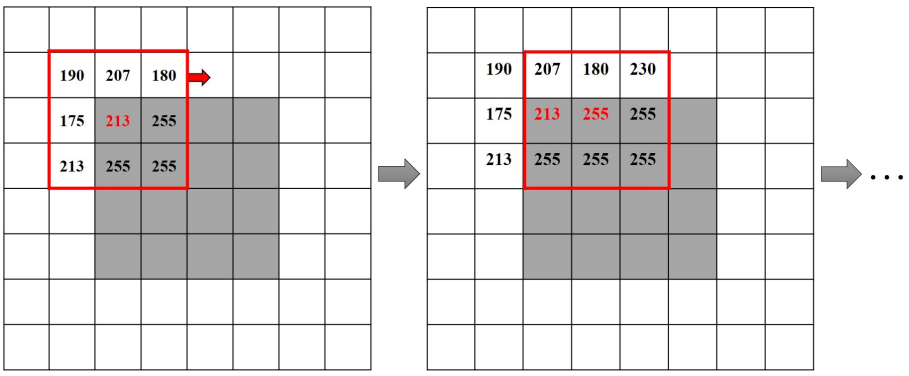


Figure 3: Median$\{180, 207, 213, 230, 255, 255, 255, 255, 255\} = 255$.

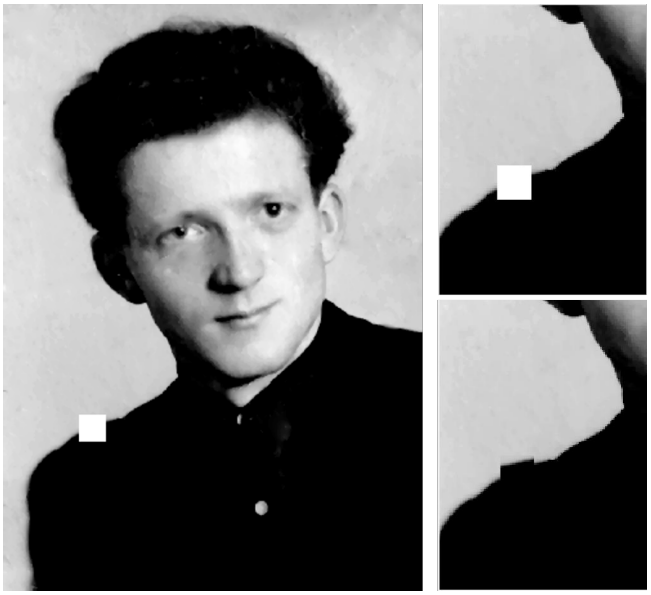


Figure 4: Grayscale test case. The left image: Corrupted in the region of the shoulder (Original image taken from [3]); the right upper image: Corrupted region in a closer viewpoint. The right lower image: Restored image after applying the modified median filter.

3.2 Restoring Color Images

In the previous section, we discussed the application of the median filter on grayscale images and the problem of color shift on grayscale images. Now in this section, we will extend our method to color images by avoiding false colors. The difficulty is that a color image consists of vector-valued data, i.e. a pixel consists of three channels, a red, a green, and a blue channel.
Therefore the whole data matrix of pixel value we extracted from color image is a $size \times size \times 3$ tensor matrix, where the size is the resolution of the image. Since the data matrix consist of three color channels, our method is to split the data into R(red)-G(green)-B(blue) channels and each channel returns a 2-D data matrix of pixel values ranging from 0 to 255.
After the splitting, the work procedure is exact the same as what has been done in the grayscale images, i.e. the median filter is applied to each channel to fill in the corrupted region. And then we combine the data from R-G-B channels back to one tensor matrix, which gives us the results of restoration of color images.
It can be observed from Figure 5 that the original color image with white corrupted region is split into red, green and blue channels. Moreover, the corrupted region of each channel is shown as pure red, green and blue respectively, which results in white color in colored image. From the color image result in Figure 6, the method returns correct colors after we apply the median filter to each channel and combine the three channels together. However, there is still a color shift problem on the

Figure 5: Color image split.

Figure 6: Color image result.

boundary of the corrupted region. Therefore, it is necessary to combine the existing method with the contour extraction as constraint to our filter.

3.3 Contour Extraction

Before we are able to think about connecting any contours, we need to detect them. To do so, we have been looking at two different operators. On one hand, there was the Sobel operator. It is a discrete differentiation operator. To detect edges, an approximation of the gradient is calculated. On the other hand, there is the Laplace operator which is, in two dimensions given by

$$\Delta f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \tag{1}$$

Since an image is represented as a set of discrete pixels, we have to find a discrete convolution kernel that can approximate the second derivatives in the definition of the Laplace operator. This results in the Matrix $\begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix}$ which can be used as a kernel in image processing. Looking at the second derivatives highlights rapid changes in the color. After applying one of these operations, we

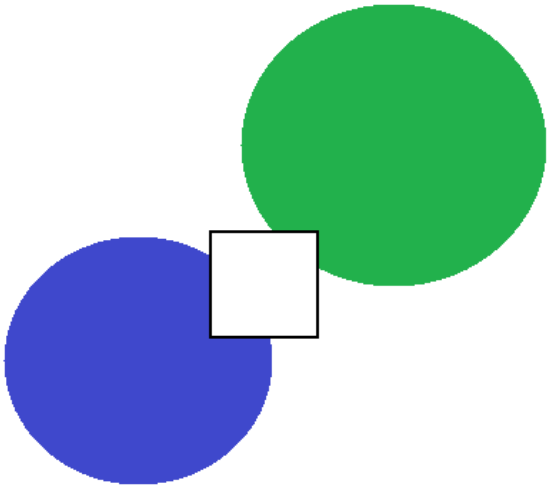
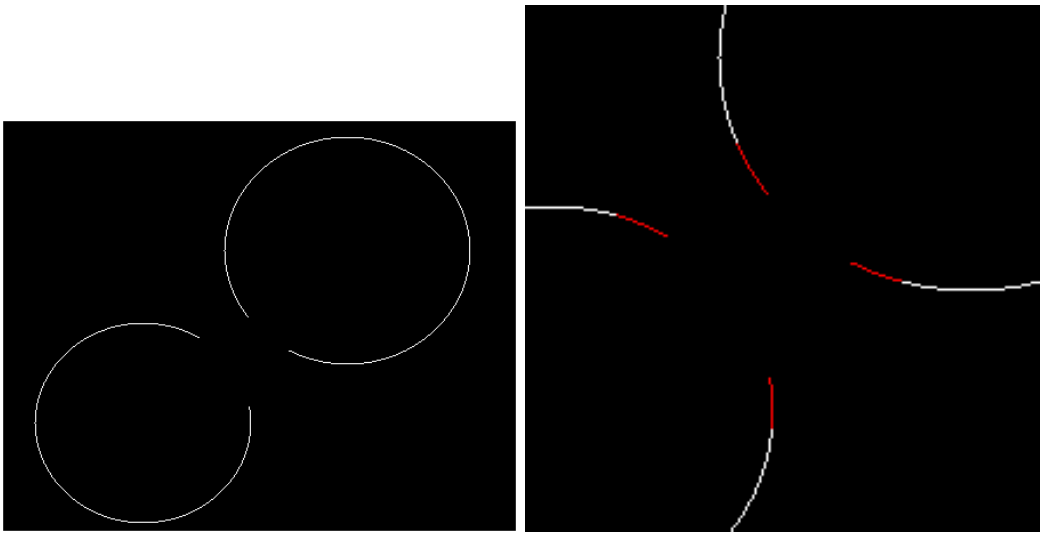


Figure 7: Example 1.

converted our image into a binary image, setting every value above a certain threshold to one and every other pixel to zero.
Considering Figure 7, you can see another problem. Obviously there is an edge along the missing

part. Because we are not interested to detect lines along those borders, we have to remove them which leads to Figure 8a. At every contour touching the defected area, we start an algorithm. This



(a) Applying Laplace filter. (b) Contours marked.

one recognizes which white pixels belong to this contour. The pixels, connected to the starting pixel (x_i, y_i) are called P_n. This is done by checking for $k, l \in \{-1, 0, 1\}$ the pixels $(x_i + k, y_i + l) \notin P_n$ to be white as well. If there is one, we continue with this pixel. If there is no further white pixel or we reached a certain number of iterations, we stop the algorithm. If there are existing $p, q \in \mathbb{N} : P_p \cap P_q \neq \emptyset$, we have to look at it more detailed. If $P_p \subset P_q \vee P_q \subset P_p$, we are ignoring the dominated one in the further steps.
Continuing in this manner, results in one set of pixels P_i, for every line hitting the defected area.

4 Contour Mapping

After tracking the contours that enter the corrupted area of the image, the next challenge is to map these contours to one another. In order to generate separate domains to apply the color filling algorithm to, it is necessary to consider the shape of the contours. Figure 9 shows an example of four contours entering a corrupted area at points 1 to 4 (green). A human can intuitively continue

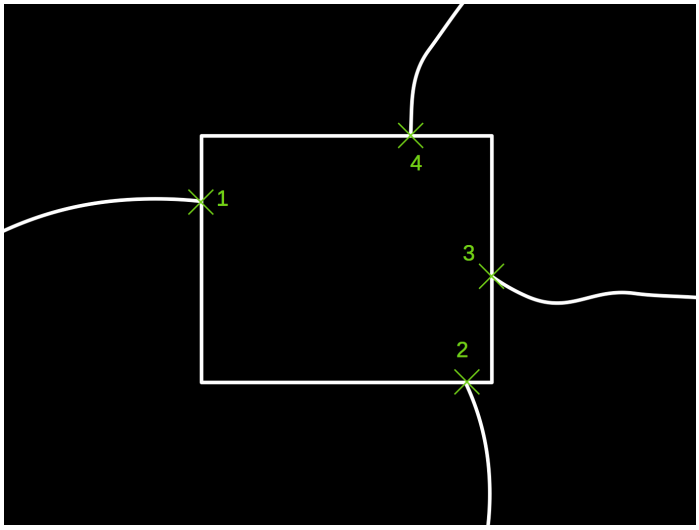


Figure 9: Contours entering the corrupted region.

these contours and tell that contour 1 is most likely connected to contour 2 and contour 3 most likely

to contour 4. For a machine to be able to do this, we developed a method to make an educated guess on which contour matches with which.
The first three points are marked on the traced part of one contour at equal distance to each other, including the starting point. Using these three points a circle can be constructed. The circle consists of a center point with coordinates x_m and y_m and a radius r. For all three points the following equation holds:

$$(x - x_m)^2 + (y - y_m)^2 = r^2$$

Which can be written as:

$$x_m^2 + y_m^2 - r^2 - 2x_m x - 2y_m y = -(x^2 + y^2)$$

With substitution we get a linear system of three equations that can be solved for $x_m^2 + y_m^2 - r^2 - 2x_m x - 2y_m y$, which in turn gives us the required coordinates x_m, y_m and the radius r.
The idea is, that a circle preserves the curvature of the contour, and will roughly point towards the starting point of the correct other contour. This is illustrated in Figure 10. A red circle is constructed through the 3 green points on contour 1. The distance of the starting point of every contour (orange) is then used as a measure for the likelihood of the respective contour to be part of a correct mapping. The distance to the circle is calculated as the difference in distance to the center coordinates, x_m and y_m, and the radius r. For each starting point a ranking of the all starting points is then created.

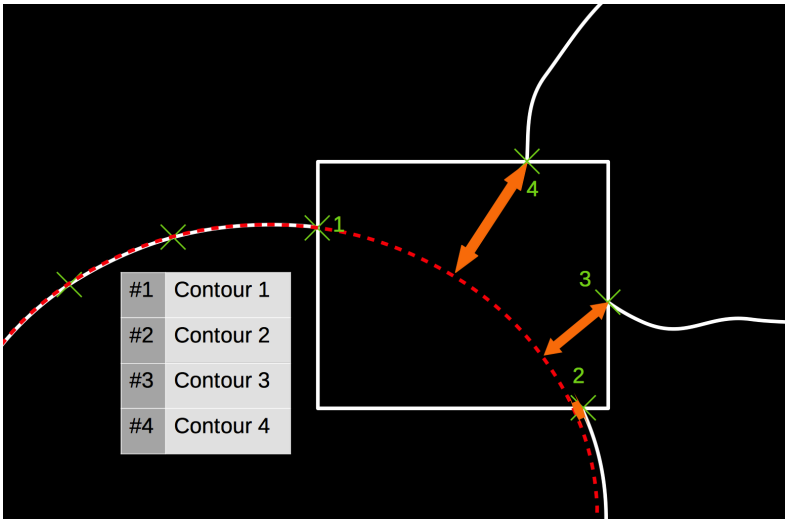


Figure 10: Matching points ranking.

The first place is trivially the initial starting point itself, since it has a distance of 0 to the circle. It is followed by contour 2 since it is almost a perfect match.
This process is then repeated for all starting points. Using the resulting tables of rankings the matching is then computed.
This method returned acceptable results for our test cases, however it is not guaranteed to succeed. Difficulties are more likely to occur the less the curvature of a contour resembles a circle. The location of the three points used to construct the circle can be critical in that matter. If they by chance happen to be perfectly aligned, the equation system for the circle coordinates becomes unsolvable, because the circle would be infinitely large. However even for real world images this is very unlikely to happen, even on seemingly straight lines.
Another problem could be several contours entering the corrupted area parallel. This can result in multiple contours being matched with the same starting point. Other approaches than circles where considered but were ultimately dismissed for being more problematic.

5 Connecting Matched Contourlines

After we established which contourlines we wish to connect, we have to decide how to connect those. The filtered image is given as a matrix $A \in \{0,1\}^{n\times m}$, therefore, to be able to form a continuous function, we model it as a function

$$I : [1,m] \times [1,n] \to [0,1], (x,y) \mapsto t. \tag{2}$$

A curve C in the picture can be parameterized as an image of a Path

$$\Phi : [a,b] \to \mathbb{R}^2, t \mapsto \begin{pmatrix} x(t) \\ y(t) \end{pmatrix}, \tag{3}$$

with $C = \Phi([a,b])$. To get a curve $\tilde{C}$ of the to be continued lines, we need to perform an interpolation with the grid points in a set G, with elements for the first m pixels in each of the contourlines P_1 and P_2 (see section above), to get the coordinates (x,y) for each of n query points.

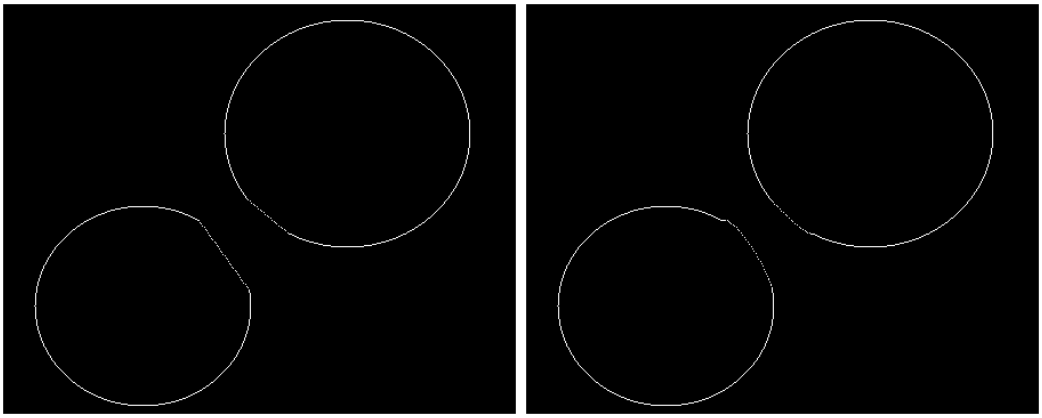
The first step is to evaluate the size of n, which we do by taking the distance of the starting points $\begin{pmatrix} x_1 \\ y_1 \end{pmatrix}, \begin{pmatrix} x_2 \\ y_2 \end{pmatrix}$ of the to be continued contours and multiply it by a factor $c \in \mathbb{R}$

$$n = c \cdot \left\| \begin{pmatrix} x_1 - x_2 \\ y_1 - y_2 \end{pmatrix} \right\|_2. \tag{4}$$

The next step is to interpolate the two functions

$$\Phi_x : [a,b] \to \mathbb{R}, t \mapsto x(t) \tag{5}$$
$$\Phi_y : [a,b] \to \mathbb{R}, t \mapsto y(t) \tag{6}$$

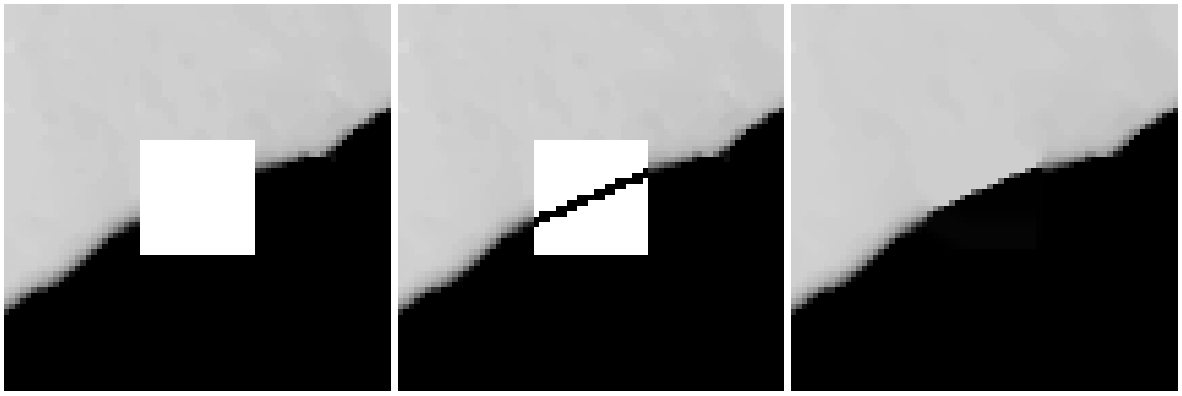
separately with the supporting points (t_i, x_i) for Φ_x and (t_i, y_i) for Φ_y, with $i \in \{-(m-1), ..., 0, (n+1), ..., (n+m)\}$ and $\Phi(t_i) = (x_i, y_i)$, where $(x_i, y_i) \in P_1$, for $i \in \{-(m-1), ..., 0\}$, and $(x_i, y_i) \in P_2$, for $i \in \{n+1, ..., n+m\}$, to get the query points (t_i, x_i) (or (t_i, y_i)), with $i \in \{1, ..., n\}$.
Our first approach was to do a linear interpolation in both coordinates. When fitted to the 7 Example (from the section above) the result displayed in 11a is being computed.
A different approach was to interpolate with a cubic spline, with the use of the 'not-a-knot' boundary condition, which demands that the third derivative is continuous at the grid points t_{-m+2}, t_{n+m-1}(see [1] Chapter 9). The result is displayed in 11b. As we can see, the spline interpolation is creating significant better results in reconnecting the circles.



(a) Linear interpolation. (b) Cubic spline interpolation.

6 Results

The methods described in this work will now be applied to a grayscale example. The illustration of the single steps can be seen in Figure 12. For this case we generated a missing area in the location of the shoulder of Figure 4 as seen in Figure 12a. After the extraction of the contours and detection of the contour entry-points, we interpolate linearly, as seen in 13b. Consequently the median filter is used to blend in the colors while accounting for the contour position. The comparison of the corrupted and restored image context of the complete image can be seen in Figure 13.



(a) Missing area with with surrounding information.

(b) Contour extraction and linear interpolation.

(c) Colour blending with the help of a median filter.

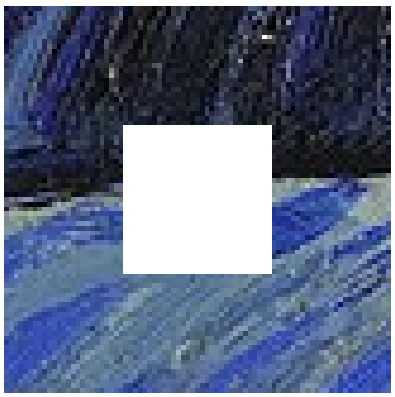
Figure 12: Resulting steps for the grayscale example

(a) Corrupted gray scale image.

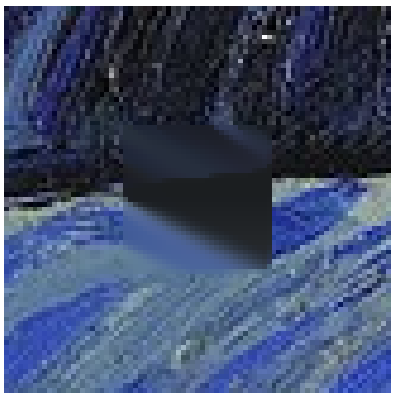
(b) Corrupted gray scale image [3].

Figure 13: Comparison of corrupted and restored image.

Applying the method to a colored image turned out to be quite challenging for the example that was given to us by our advisor. The biggest problem in this case is the extraction of the contours, since the texture has a high level of detail. Without taking the contours into account as shown in Figure 14, the colors are just blending in the direction the filter is moving. Contrary to our expectation there were no extreme cases of false colors. In summary we received the most satisfying results for corruptions consisting of a small area with little texture detail.



(a) Missing area with with surrounding information.



(b) Colour blending with the help of a median filter.

Figure 14: Resulting steps for the colored example.

Figure 15: Restored colored image.

7 Outlook

One of the major problems that our group encountered was the contour extraction. Due to time limitation we were not able to implement a better alternative and thereby make our proposed method applicable to the van Gogh example. Therefore, in future work one could try to implement methods proposed in e.g. [4]. Furthermore there are more sophisticated methods of filtering described in [2]. Another possible extension is to also include texture or restore more detailed areas in a picture. For that reason one would require methods from the field of machine learning, e.g. an auto-encoder.

References

[1] Wolfgang Dahmen and Arnold Reusken. *Numerik für Ingenieure und Naturwissenschaftler* -. Springer-Verlag, Berlin Heidelberg New York, 2. aufl. edition, 2008.

[2] C. Guillemot and O. Le Meur. Image inpainting: Overview and recent advances. *IEEE Signal Processing Magazine*, 31:127–144, 2014.

[3] P. Peter. *Understanding and advancing PDE-based image compression*. PhD thesis, Saarland University, Mathematical Image Analysis Group, 2016.

[4] Emmanuel J. Candè s and David L. Donoho. Curvelets - A Surprisingly Effective Nonadaptive Representation For Objects with Edges. *Saint-Malo Proceedings*, pages 1–10, 2000.