

Automatic Extraction of Ridge Lines from Digital Eleva- tion Models

Master Thesis

T. E. van Noppen

Automatic Extraction of Ridge Lines from Digital Elevation Models

Master Thesis

by

T. E. van Noppen

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on 17 February, 2022.

Student number: 4975197
Date: 3 February, 2022
Supervisors: Dr. ir. R. J. Van der Ent, TU Delft
Dr. ir. J. P. Aguilar-Lopez, TU Delft
Dr. ir. M. M. Rutten, TU Delft
Ir. B. J. A. de Graaff, HKV Lijn in Water
Dr. ir. A. van Dam, Deltares
Dr. ir. G. Donchyts, Deltares

Abstract

Second-order Gaussian kernels have been utilized to develop three algorithms that could automatically extract ridge lines for hydrodynamic modelling. Isotropic second-order Gaussian kernels produce inaccurate lines at crossings and junctions. To avoid the malfunctioning of Second-order Gaussian kernels, one default and two alternative algorithms were developed. The first, default algorithm is based on isotropic kernels and non-maximum suppression. For the first alternative algorithm, isotropic and anisotropic kernels have been applied for the filter process. The third algorithm uses skeletonization instead of non-maximum suppression. A verification was applied to analyze the performance of the algorithms. The Matthews correlation coefficient (MCC) of the default algorithm and the alternative algorithm that included anisotropic kernels was found to be 0.17. For the algorithm based on skeletonization a value of 0.08 was obtained. Hence it has been concluded that the algorithms that utilized non maximum suppression instead could more accurately detect ridge lines than the model based on skeletonization. However, the latter generated lines that contained less discontinuities. Furthermore this algorithm turned out to be computationally less demanding in comparison to the other two algorithms.

Contents

1	Introduction	1
1.1	Gaussian kernels	3
1.1.1	Crossings & Junctions	5
1.2	Research Objective	6
1.3	Thesis outline	6
2	Methodology	7
2.1	Algorithms	7
2.1.1	Algorithm I	7
2.1.2	NaN-filling	7
2.1.3	Filtering	8
2.1.4	Thresholding	11
2.1.5	Non-maximum suppression (NMS)	12
2.1.6	Linking broken ridge lines	13
2.1.7	Algorithm II	14
2.1.8	Algorithm III: Isotropy and Skeletonization	18
2.1.9	Overview algorithms	20
2.2	Post-processing in QGIS	20
3	Verification manually drawn ridge lines	23
3.1	Dimensions ridge structures	23
3.2	Parameters algorithms	23
3.3	Study site and data	24
3.4	Results algorithms	25
3.5	Results verification	28
3.5.1	Precision, Recall, F1-score and MCC	28
3.5.2	Positional accuracy analysis based on Goodchild and Hunter	30
4	Verification D-HYDRO	33
4.1	D-HYDRO model for De Roer	33
4.2	Development hydrodynamic models for validation	36
4.3	Results total inundated area	37
4.4	Results water depth	38
4.5	Analysis of the results	39
5	Discussion	45
5.1	Improvements algorithms	45
5.2	Verification method	46
5.2.1	Future research	47
6	Conclusion	49
	Bibliography	51
A	Python code algorithm 1	55
B	Python code algorithm 2	61
C	Python code algorithm 3	65
D	Post-processin steps in QGIS	69
E	Difference in water depth	71

Introduction

The catastrophic floods that occurred in mid-July 2021 in The Netherlands, Germany, Belgium and Luxembourg illustrate the destructive power of flood events. At least 224 fatalities and substantial damage to community, infrastructure and environment were reported [1]. Due to continued urban development in combination with altered precipitation patterns triggered by climate change, flood damage is projected to increase further the coming decades [2], [3]. Hence reinforcing the need for accurate models to simulate floods.

Throughout the past decades, numerous numerical models for flood simulation have been developed and applied for many engineering, planning and risk assessment studies [4], [5], [6]. Flood inundation modelling is usually based on simplified 2D models that solve one-dimensional Saint-Venant equations. These models are preferred over the full shallow water equations as they are computationally less demanding [7]. Traditionally, 2D-models have been constrained by the lack of high resolution topographic data [8]. Due to recent developments in data capture techniques high-resolution topographic data has become widely available. Hence it is possible to create high resolution 2D-meshes for flood modelling that include very accurately the elevation of the modelled area. Nevertheless, it remains common for 2D flood modelling to re-sample fine resolution topographic data to a coarser mesh. As such, computational stability can be achieved and computational costs can be reduced [7]. However, coarsening of fine topographic data can have a severe impact on the accuracy of a model [7], since relevant sub-grid variations in elevation might be suppressed and therefore not incorporated on the coarse mesh.

To overcome the loss of relevant small-scale topographic variations, one solution is to apply sub-grid modelling for abrupt changes in height [7] [8]. Therefore, a line element is placed along the crest level of a sudden change in elevation. Subsequently, the line element is placed on the sides, or faces, of grid-cells of the mesh. The elevation at the crest level of the high ground feature is attached to the line. As such, it can affect the local flow pattern by either acting as barrier by avoiding water flow from one cell to a neighboring cell or it can function as a weir when water exceeds the crest level of the feature. Hence the variability of sub-grid topography is taken into account in a model with a coarser mesh while the computational costs can be kept low [7]. Sub-grid modelling by means of such line elements has become part of various modelling software programs such as 3Di, HEC-RAS and D-HYDRO Suite [9], [10].

In this study, the line that is placed on top of a sudden change in elevation, is referred to as *ridge line* and has been defined as: a line element that connects the highest points that are located next to an abrupt change in elevation. Hence lines are placed on two types of landforms, which are visualized in figure 1.1. Here, (a) represents a ridge type of obstacle and the second (b) a shoulder type, for simplicity, both landforms will from now on be referred to as: *ridge structures*.

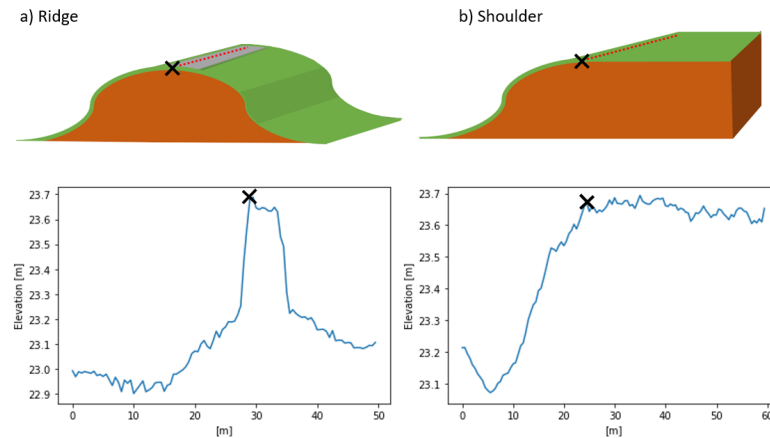


Figure 1.1: Fixed weir lines. (a) ridge type of obstacle, (b) shoulder type of obstacle.

Until recently, ridge lines were drawn manually, which is a time-consuming process. In addition, drawing lines manually is a process that is subject to the interpretation of the modeller. Therefore, this study aims to develop an algorithm that is capable in automatically extracting ridge lines, which could decrease this timely-process. Moreover, such an algorithm would make the detection of ridge lines an objective process, hence creating consistent lines. Various methods have been developed that can be applied for the purpose of automatically extract ridge lines based on a digital elevation map (DEM). These methods often rely on the analysis of the first or second order derivative of the DEM. For instance, automatic drainage pattern detection [11], [12] and the profile recognition and polygon breaking algorithm which was originally developed by Chang et al. [13] make use of the first or second order derivative of a DEM. The downside of these approaches is that the algorithms are highly sensitive to noise, hence irrelevant ridge lines are created [14] [15].

In this study the potential of image processing algorithms for the purpose of ridge structure detection, has been analyzed. In the field of image processing, numerous methods exist to extract features from images, among which are: ridges and edges [16]. In this context, ridges and edges represent a sudden change in average gray level of the image [17]. An edge symbolizes a transition between low- to high greyscale values or vice versa. A ridge consists of two parallel edges, which together form a line on an image [18]. Since grayscale images and DEMs are similar in the fact that they are both single-valued rasters [19], the algorithms that were initially developed for image processing purposes may also be applied to DEMs [20]. This has been done before, for instance in the work of Dai et al. [21] where an edge detection algorithm was used to automatically delineate agricultural terraces in the Loess Plateau in China. Edge and ridge detection algorithm rely on the process of filtering an image by performing a convolution with matrix or also known as: *kernel*. The convolution is carried out by 'moving' the kernel over an image, generally starting at the top left corner, while multiplying it locally with a central pixel and its surrounding pixels. An example of a convolution of an image of size 5 x 5 is visualized in figure 1.2.

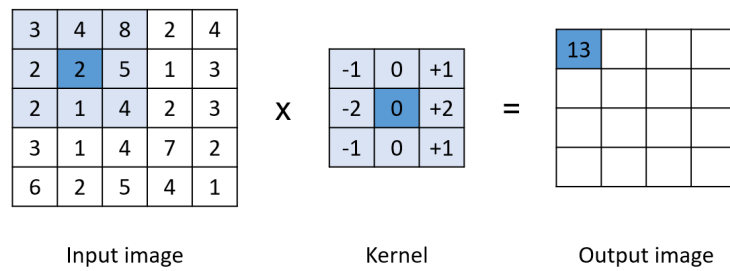


Figure 1.2: Image convolution with an input image of size 5×5 and a kernel of size 3×3 .

1.1. Gaussian kernels

Over the past decades, a lot of research has been carried out to the automatic detection of both edges and ridges based on a convolution with a kernel [22], [23]. A few well-known methods include the Roberts cross operator [24], the Sobel operator [25] and the Prewitt method [26]. These methods make use of very simple differential kernels that can compute local gradients of an image by performing a convolution, hence the regions where a large gradient is present, represent the edges. The advantage of using small differential kernels is that they are computationally inexpensive. The disadvantage is that they are highly susceptible to noise. The small kernels were designed to detect so-called step edges, which are edges with a very abrupt intensity change, ranging from one pixel to the next pixel. In real-world images, these type of edges do almost never occur. Usually an image consist of edges and ridges of which the discontinuity in grey level varies over different widths [16], [27]. Therefore other kernels have been developed which could detect edges and ridges of different sizes, including the Gaussian kernels [18] [23]. A Gaussian kernel is a 2D-representation of a Gaussian bell-curve function, which can be calculated according to:

$$G(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \quad (1.1)$$

where σ represents the standard deviation of the kernel, usually the mean μ is set to zero and is therefore often left out equation. Gaussian kernels are among the most employed tools for image processing, where they have proven useful for a number of different tasks, such as the automatic detection of line-like structures such as blood vessels [23], [28]. The derivative of the 2D-Gaussian kernel in one direction results in a first-order Gaussian kernel, which is used to extract edges. The second derivative in one direction creates the second-order Gaussian kernel, or (SOG-kernel). When a convolution is applied between a SOG-kernel and a DEM, the noise in the image is suppressed and the extreme lows in the output indicate the location of the ridge structures, as is visualized in figure 1.3. The Gaussian kernel in 2D and its first- and second-order derivative are illustrated in figure 1.4.

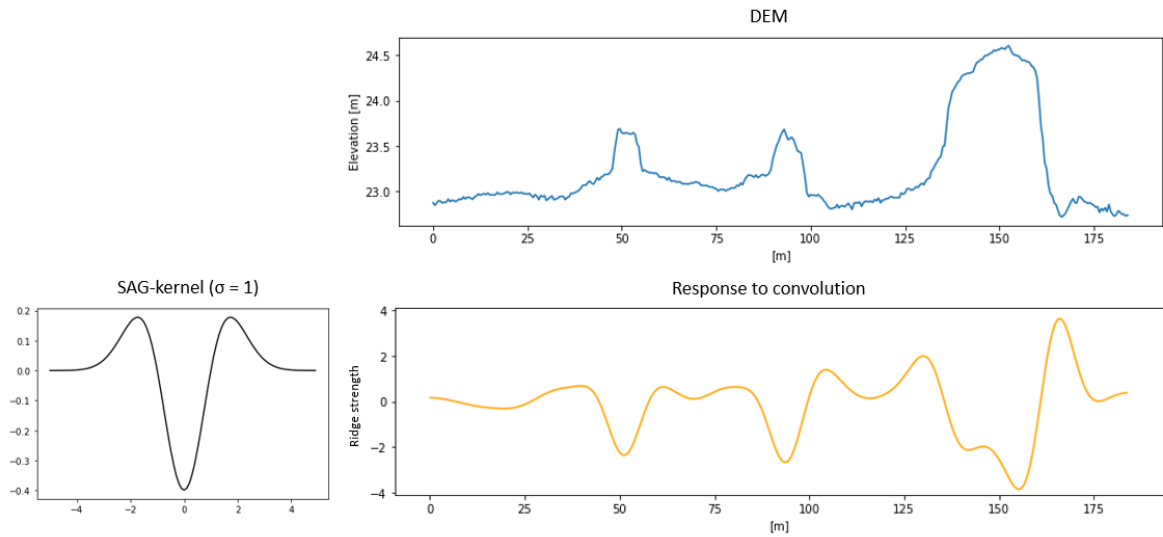


Figure 1.3: Convolution cross section and SOG-kernel with $\sigma = 1$.

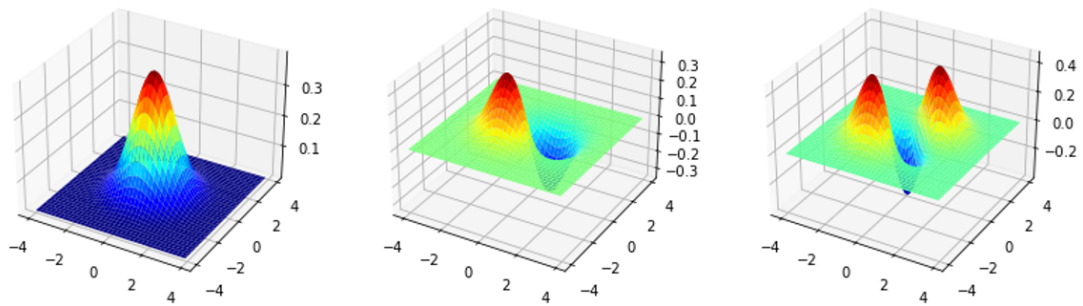


Figure 1.4: Gaussian kernel, its first and its second derivative.

In this study the potential of SOG-kernels for the purpose of automatically extracting ridge structures from DEM data will be analyzed. A second-order Gaussian kernel can be expressed as:

$$\hat{G}''(x, y) = \left(\frac{1}{\sigma\sqrt{2\pi}} + \frac{1}{\sigma^2} \right) \frac{1}{\sigma\sqrt{2\pi}} e^{-\left(\frac{\phi}{2\sigma}\right)^2} \quad (1.2)$$

with:

$$\phi = \begin{bmatrix} x & y \end{bmatrix} M^T \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} M \begin{bmatrix} x \\ y \end{bmatrix} \quad (1.3)$$

and:

$$M = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \quad (1.4)$$

here M^T is the transpose of the matrix M . Furthermore, the the standard deviation σ in the equation controls the size of the SOG-kernel. A larger standard deviation results in a wider kernel that can detect wider ridge lines, while a lower standard deviation results in a smaller kernel that is capable of extracting the smaller ridges [18]. The orientation of the 2D-kernel can be adjusted by changing the value of theta (θ), which can take a value of: $\theta \in [0, 2\pi]$. Since the responses at θ and $\theta + \pi$ are identical, the orientation interval can be set as: $\theta \in [0, \pi]$ [16]. The impact of different values for theta is visualized in figure 1.5. In this figure, two schematic DTMs are visualized in the two left figures. A convolution of the schematic DTMs with $\theta = 0$ would result in the two figures visualized in the second column where the yellow lines indicate pixels with high response values. Clearly, the kernel with only one orientation does not detect the horizontal line that is present in the upper left figure. However the combined output of two orientations, ensures that, both yellow lines are detected. This can be seen in the two right figures, which indicate the combined result of the two kernel orientations. A convolution with a SOG-kernel can be established by:

$$\frac{\partial^2 I}{\partial v_\theta^2} = -I * \left(\frac{x \cos \theta + y \sin \theta}{\sigma^2} \hat{G}''(x, y) \right) \quad (1.5)$$

where, ∂v_θ^2 represents the unit vector in direction θ . For simplicity a minus sign is added to the formula, hence ridge structures are indicated by high extremes, in stead of low extremes.

1.1.1. Crossings & Junctions

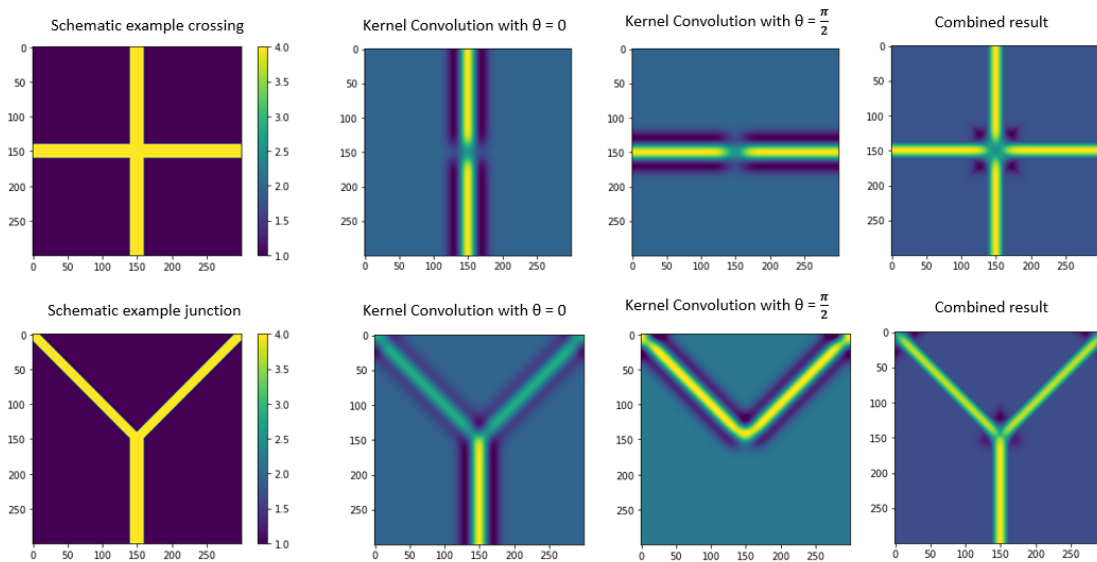


Figure 1.5: Schematic DEM representing a crossing and junction. The result of a kernel convolution with $\theta = 0$, $\theta = \frac{\pi}{2}$ and the combined result is visualized

Second-order Gaussian kernels rely on the idea that a substantial difference is present between the top and the toe of the ridge [18], which is unfortunately not always the case. Especially if a ridge is part of a crossing of two ridges, or if two ridges join to form a single ridge (a junction) [18]. For crossings and junctions there is no difference present between the top and the toe of the ridge and therefore, these areas will certainly lead to a near-zero output of the convolution hence creating gaps in the ride lines, as is visualized in figure 1.5. Here, the cross points of the joints and junctions are much less pronounced in the combined results shown in 1.5. Gaps in ridge lines at crossings and joints are very likely to have a large impact on the results of a hydrodynamic flood model. Therefore, the problem of crossings and junctions will play a significant role in the development of an algorithm that could automatically extract ridge structures.

1.2. Research Objective

The aim of this study is to develop and test an algorithm that can automatically detect ridge lines based on a digital elevation model (DEM). To do so, analysis has been conducted to the potential of SOG-kernels to detect ridge lines that can be used for hydrodynamic models, hence the following research question has been used:

How can second-order Gaussian kernels be utilized for the automatic extraction of ridge structures, and what are the implications for 2D-flood modelling?

To answer this question, three algorithms have been developed based on SOG-kernels. A validation of the algorithms was performed in two ways. Firstly the outcome of the algorithms has been compared to manually drawn ridge lines. And secondly, the outcome in the algorithms has been incorporated into a 1D/2D flood model. Subsequently the performance of the models has been analyzed with respect to a "ground truth" model.

Since the purpose of the extracted ridge lines is to be used for flood modelling, four requirements were set for the resulting lines of the algorithms:

- The algorithms should be able to detect ridge structures of various sizes. Usually ridge structures of various sizes can function as an obstacle in a flood model, therefore all relevant structures should be implemented in the hydrodynamic model.
- The algorithms should not be affected by noise. No lines should be created on abrupt changes that represent noise on the DEM.
- The ridge lines that are placed by the algorithms should be located exactly on, or at least in the vicinity of, the crest level of the ridge structure. A ridge line that is placed near the highest points of a ridge structure would probably obtain a lower elevation, hence the ridge structure would start to overflow much faster than would be the case in reality.
- The algorithms should produce continuous ridge lines, without unnecessary gaps. Gaps have a major impact on the inundation pattern of a flood model. Due to gaps the blocking function of the weir is not incorporated in the model, hence water can flow through the gap to the neighboring grid cell.

The performance of the algorithms is verified in two ways, firstly the automatically extracted lines are compared to manually drawn ridge lines, and secondly the resulting lines are incorporated into a hydrodynamic model developed for the river Roer in Limburg (The Netherlands). This model has been developed in the software program D-HYDRO and concerns a coupled 1D-2D flood model, thus both one dimensional as two dimensional input variables are used.

1.3. Thesis outline

The rest of this paper is organized as follows. In chapter 2, the methodology will be outlined, which includes a detailed explanation of three algorithms that were developed for this study. This chapter also contains information on the impact of the parameters that are used for the algorithms. The algorithms are validated in chapter 3 and 4. In chapter 3 a comparison is performed between the outcome of the algorithms and manually drawn ridge lines. Chapter 4 concerns the validation in the hydrodynamic model.

2

Methodology

In this section, firstly an elaboration on the procedure of the first algorithm is provided. Subsequently, the second and the third algorithm will be outlined. The generated ridge lines from the algorithms require post-processing steps in order to create the correct format for the D-HYDRO model. The post-processing steps will be outlined in the final section.

2.1. Algorithms

The first algorithm can be considered as the default algorithm. For the second algorithm a convolution is applied with an elongated version of the SOG-kernel. The third algorithm utilizes skeletonization in order to obtain line elements instead of zones that indicate ridge structures. In this section, every algorithm will be discussed in detail, whereby the impact of the various parameters will be outlined. For the development and testing of the algorithms a Digital Terrain Model (DTM) was used with a resolution of 0.5 x 0.5m. The output of the algorithms is a binary raster map where '1' indicates a pixel that is part of a ridge line and '0' a pixel that is not part of a ridge line.

2.1.1. Algorithm I

The first 'default' algorithm is visualized in 2.20. The algorithm consists of five steps, which are indicated by the upper boxes. The lower boxes show all the parameters that are required for each step.

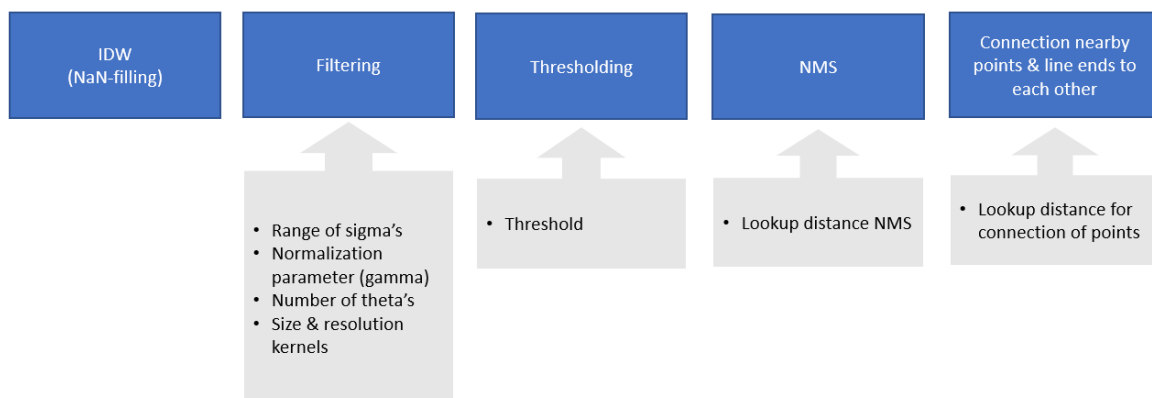


Figure 2.1: Procedure of algorithm I.

2.1.2. NaN-filling

An accurate digital representation of the terrain is of significant importance for the correct localization of ridge lines. LiDAR (Light detection and ranging) has become the main data source for producing high resolution DTMs [29]. Usually unwanted features, including buildings, vegetation and water bodies, are

removed from the DTM resulting in voids in the data [30]. The data points in these voids contain *Not a Number*-values (NaN-values). To fill these NaN-values, a pre-processing step is required. Various interpolation algorithms exist to fill NaN-values in elevation data [30] [31], whereof the most common are: inverse distance weighting, kriging and the moving average [31]. Kriging is considered as one of the best performing methods, however the procedure requires a long calculation time. On the contrary, the moving average is relatively fast but the resulting interpolated map is often less accurate [32]. In between is inverse distance weighting (IDW) which gives satisfactory accuracy with a reasonable calculation time [32]. Therefore, IDW has been selected in this study as NaN-filling mechanism.

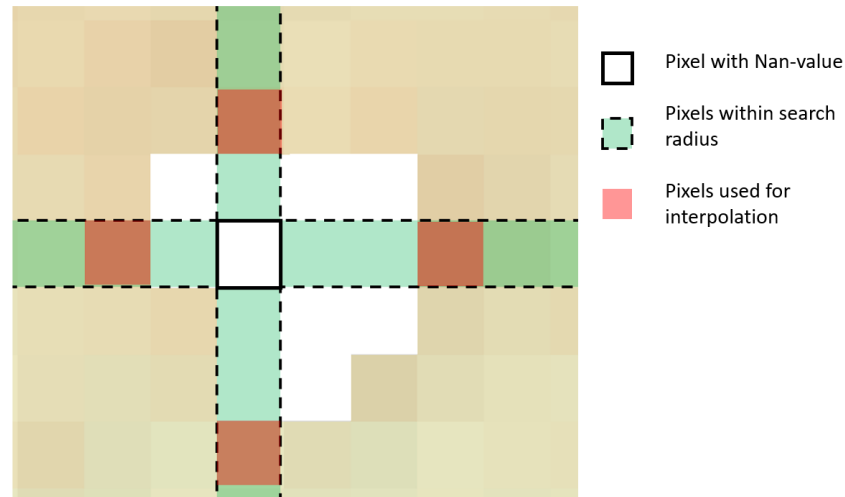


Figure 2.2: Schematic picture of Inverse Distance Weighting. In total four pixels are utilized for interpolation which are highlighted with the red boxes. The search radius is indicated by the green pixels

To carry out IDW, the module *fill no data* has been used, which is available in the python package *rasterio* (version 1.2.0). This module analyses every pixel containing NaN-values. For each pixel a search is performed in four direction (north, south, east, west) to find the closest non-NaN-pixels. The firstly encountered non-NaN pixels are used for interpolation. Thus up to four pixels are used for interpolation as is shown in 2.2. For the algorithm the default search distance has been used, as such, a maximum number of pixels that is analyzed to find values to interpolate from is 100.

2.1.3. Filtering

IDW is followed by a filter process in which a set of SOG-kernels utilized with a predefined range of sigma and theta. This process generates one map that contains different ridge response values at every pixel, which represent the combined outcome of the convolutions with various kernels. High ridge response values refer to the areas where ridges are detected. Firstly, the impact of the parameter sigma will be discussed.

Parameter: sigma

The impact of different sigma values on the outcome of the convolution is visualized in figure 2.3. In the upper graph of the figure, a cross section of a DTM is visualized. The other three graphs reveal the response of a convolution of the DTM when using SOG-kernels with a sigma value of 0.5, 1.0 and 2.0. As is shown, only the kernel with $\sigma=2$ correctly detects the largest ridge on the right side of the DTM. The convolution with $\sigma=0.5$ results in two peaks on each side of the ridge. The smaller two peaks on the left side of the image are detected by all three kernels.

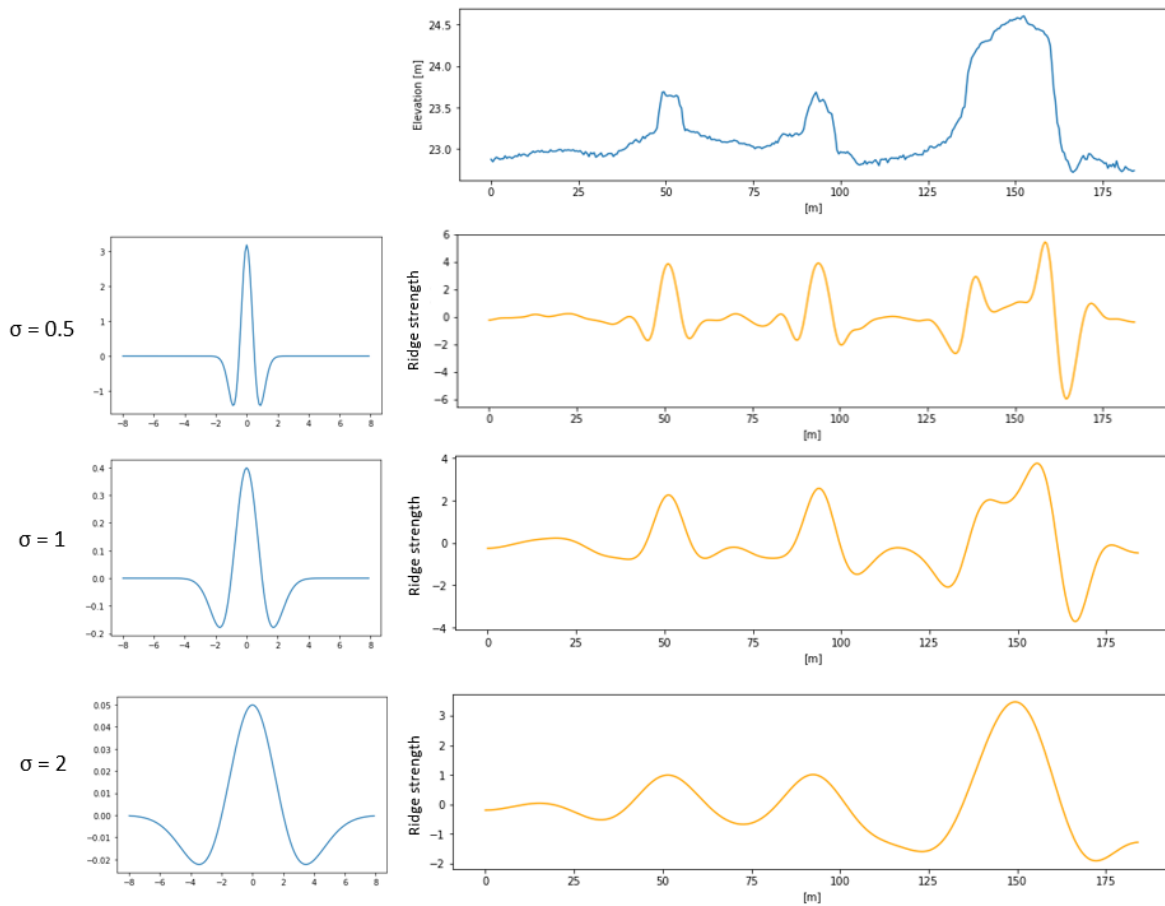


Figure 2.3: Kernel convolution of DTM with kernels of $\sigma = 0.5, 1.0$ and 2.0 . Upper graph: cross section DTM.

For an optimum result of the convolution, it is of major importance that the width of range of sigma values corresponds approximately to the width of the ridge structures that must be detected [33]. Based on this principle, the following equation was found for the relation between the width of the ridge structure and sigma:

$$\sigma = \frac{W_R * Res_k}{3.46 * Res_D} \quad (2.1)$$

where, W_R is the width of the ridge, Res_D and Res_k indicate the resolution of the DTM and the resolution used for the kernel. The value of 3.46 represents the distance between the two low extremes of the kernel for $\sigma = 1$, which has been established by taking the third derivative of the Gaussian function. Hence the distance between the zero-crossings was found to be 3.46. An examples of the selection of an appropriate value for sigma is provided in in figure 2.4. Here, the DTM contains a resolution of 0.5m. The kernel is created by using a step size for x of 0.1. The ridge structure consists of a width of approximately 8m, hence filling in the equation results in $\sigma = 0.46$.

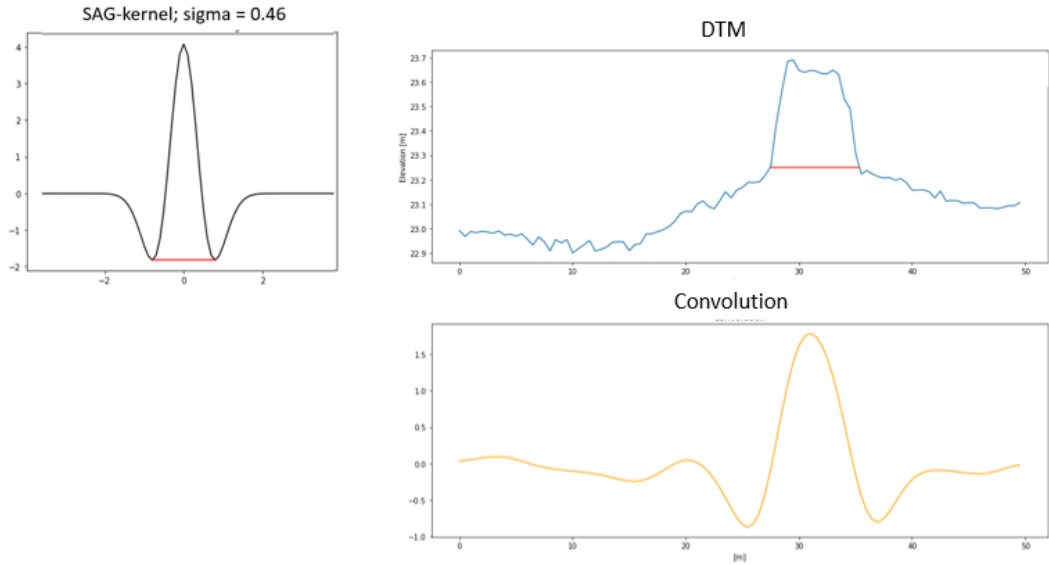


Figure 2.4: Determining appropriate value for sigma. Upper graph: cross section DTM. Middle: kernel. Lower graph: response to convolution.

Parameter: gamma

Figure 2.3, reveals that the output of the convolution becomes lower for wider kernels [28], [16], [33]. The kernel of $\sigma = 0.5$, resulted in a range of -6 up to 6, whereas the kernel with $\sigma = 0.5$, created an output with values ranging between -2 up to 3.5. In order to detect ridge structures of various sizes the output is combined by means of a normalization [16] [33][27] [28]. Hence, the kernels of multiple sizes can be used and the result to the convolution can be combined into one layer by extracting the maximum response value for every pixel [34], [28]. For the normalization an additional parameter is added to formula, referred to as the γ -parameter, which has been proposed by Lindeberg (1998) [33], [27]. With the normalization parameter, the convolution of an image (I) with a Gaussian kernel $G(x,y)$ can be expressed as:

$$\frac{\partial^2 I}{\partial v_\theta^2} = -I * \left(\sigma^{\gamma n} \frac{x \cos \theta + y \sin \theta}{\sigma^2} \hat{G}''(x, y) \right) \quad (2.2)$$

where the normalization parameter γ takes a value of $\gamma \in \mathbb{R}_+$. The variable n denotes the order of the derivative, which obtains a value of 2 when utilizing second-order Gaussian function. By increasing gamma, the kernels with a larger sigma value become more pronounced in the combined result, while for lower values for gamma, the smaller kernels overrule the larger kernels. Therefore, it can be stated that the value for gamma does affect the manner in which a ridge is detected [28]. For instance, if a lower value for gamma is applied, the smaller kernels have a more predominant effect on the outcome of the combined convolution. Consequently, it might occur that a wider ridge obtains two lines on each side of the ridge in stead of one line in the middle of the ridge line. For ridge detection the optimum value for gamma is established at 0.75 [33].

Parameter: step size and dimension

The value for sigma does also depend on the resolution used for the kernel, which is determined by the step size for x and y in formula 1.2. To illustrate the impact of the resolution of the kernel, two kernels have been plotted in figure 2.5. One kernel contains a σ -value of 10, the other kernel a σ -value of 1. As can be seen in the figure, decreasing the value of sigma, while reducing the resolution (step size $x = 1$ to $x = 0.1$) yields a kernel with exactly the same form, as can be seen on the x-axis. Except the range on the y-axis changed with three orders of magnitude. Therefore, a convolution of both kernels with a cross section of a DEM would yield a similar output, apart from the range of the output.

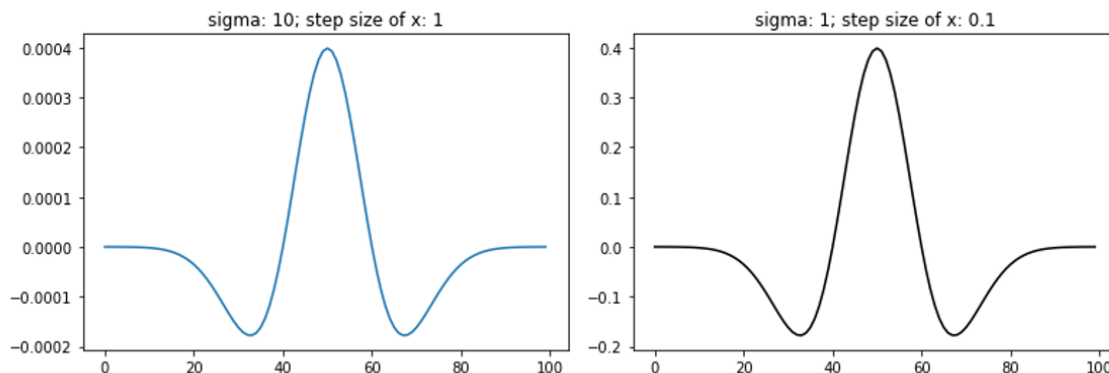


Figure 2.5: Impact step size

In addition to the step size, also the dimensions of the kernel must be set. It is of paramount importance that the SOG-function is entirely covered by the shape of the kernel, otherwise the output of the convolution will not oscillate around zero but around a value above zero. Hence when multiple kernels are used, the largest kernel will certainly dominate the results, regardless the normalisation that is applied.

Parameter: Number of theta's

Ideally, the kernel should be oriented normal to the direction of the ridge to be detected. Since the direction of the various ridges is unknown beforehand, a set of kernels covering a range of possible orientations should be used for the convolution [16]. The maximum value is then selected to compose the output of the filter process [16].

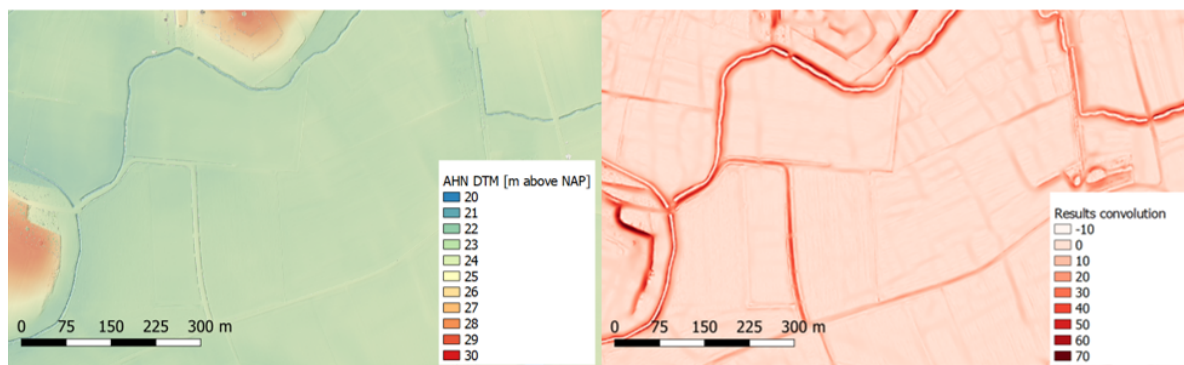


Figure 2.6: Result of filter process.

An example of the filter procedure is illustrated in 2.6. The left image shows the DTM that was used as input, while the right image reveals the response value to the convolution. Regarding the time consumption of the filter procedure, it should be noted that the number of kernels used for the filter process significantly affects the total calculation time of the algorithms. A larger number of kernels with different values for theta and sigma increases the accuracy but makes the process computationally expensive.

2.1.4. Thresholding

The output of the filter process is a map with the maximum response values of the convolutions with various kernels, referred to as the ridge strength. To extract only the relevant ridges a phase of thresholding should follow [18]. By using a specific threshold, the pixels that will be considered as part of a ridge structure can be extracted. The threshold determines the minimum required height of an abrupt change in elevation in order to be considered as a ridge structure. Therefore, the threshold is established by the

modeller of the hydrodynamic model as he/she defines the requirements for a ridge structure. To do so, the results of the filter process must be analyzed and compared to the DTM. Hence the threshold can be selected based on the response values that correspond to the smallest ridge structures that must be detected. Figure 2.7 illustrates the process of threshold selection, wherein three different thresholds were set for the region in 2.6. As shown in the figure, a lower threshold increases the amount of area that is considered as a ridge structure.

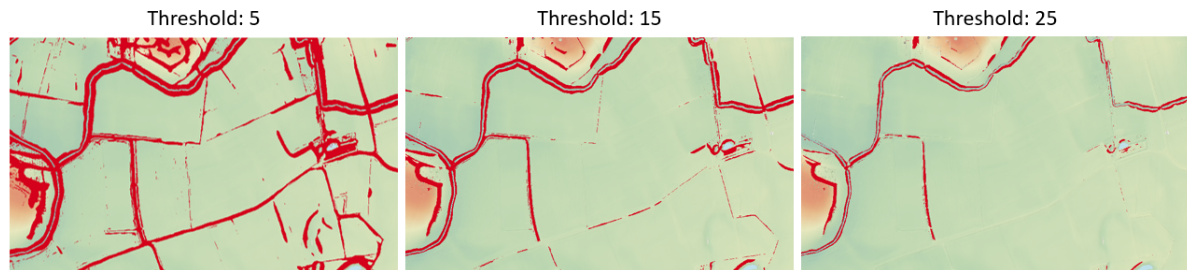


Figure 2.7: Impact of threshold = 5, 15, 25. The red zones indicate the areas that fall above the threshold

2.1.5. Non-maximum suppression (NMS)

Canny (1968) [35] developed a method for edge detection that could extract edge lines by suppressing all pixels whose intensity is not maximal within a specific local neighborhood. This method is nowadays commonly known as Non-Maximum Suppression (NMS), and has been implemented into the algorithms after the thresholding. The procedure of NMS is as follows: first, for every pixel that contains a value above the threshold, the orientation (θ) that gave the highest response value is established. Subsequently, an imaginary line of a predefined length is drawn perpendicular to this orientation. If any pixel on this line has a higher ridge response value than the central pixel, the central pixel is suppressed. NMS requires one parameter, the length of the imaginary line that is drawn perpendicular (L_{NMS}). In the work of Rosenfield and Thurston (1971) [17], the length of L_{NMS} is based on the size of the edges that should be detected by the edge detection algorithm. In this study a similar approach is used whereby the lookup length is set to the size of the smallest ridge structures that must be detected by the algorithm.

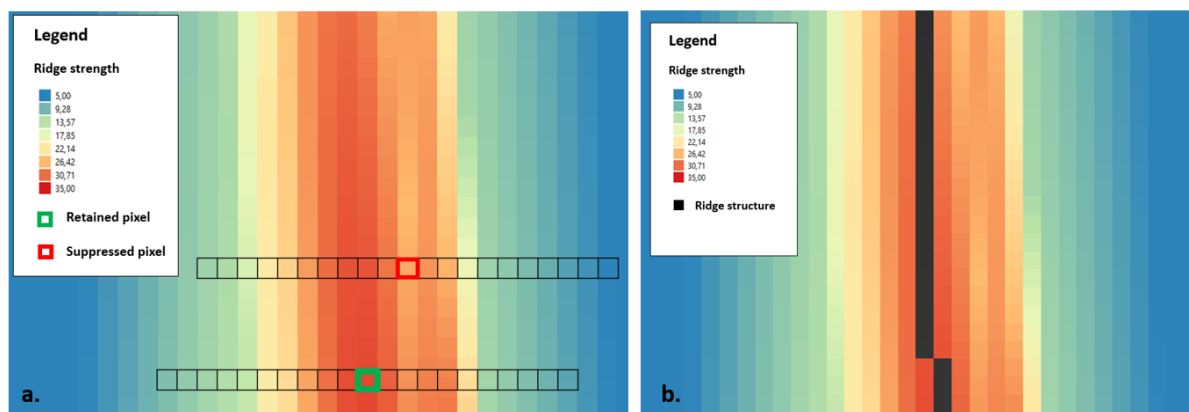


Figure 2.8: Non Maximum suppression. (a) an imaginary line is drawn perpendicular to the ridge structure. The central pixel is suppressed if a pixel is present on the line that contains a higher ridge strength, otherwise the pixels is retained. (b) the output of NMS.

It appeared that after NMS, often lines were created at the border of the DTM. Analysis revealed that the formation of lines at the borders had to do with the manner in which the convolution is carried out. During the convolution, the kernel moves from the upper left to the lower right part of the DTM. At the outer pixels the kernel extends beyond the dimensions of the DTM. To enable a convolution on

these parts of the DTM, the area that extends the DTM is treated as zeros, hence an abrupt change in elevation is created between the extended area and the values of the DTM at the borders. Therefore, the kernels detect 'ridge structures' at the borders of the DTM. To avoid the formation of ridge lines at the borders, all outer rows and columns are set to zero after NMS. The downside of suppressing all outer rows and columns is that ridge structures with their crest level exactly located at the borders of the DTM are not detected. Further research is required to determine how the convolution can be performed without the formation of ridge lines at the boundaries of the DTM.

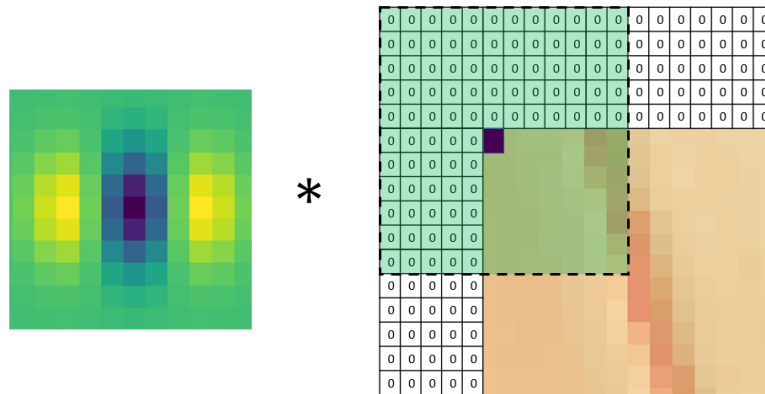


Figure 2.9: Kernel convolution at the borders of the DTM. As illustrated, the DTM is extended with additional rows and columns filled with zeros. Hence a convolution can be performed at the borders of the DTM.

2.1.6. Linking broken ridge lines

After NMS, discontinuities might appear on the ridge lines [36], as illustrated in figure 2.10. The small gaps can be related to various causes. In some cases there is indeed a gap between two ridge structures, hence the gaps are correctly indicated. However, often gaps appear due to the usage of a range of sigma values. For instance, when using a range of 1 to 2, with an interval of 1.2, a kernel of sigma = 1 could have given the highest response value at a specific point on a ridge structure, while a few meters down the ridge structure, a kernel with a sigma value of 1.2 could give the highest response value. During NMS, a gap is created in between the pixels where the highest response values are found. To connect these gaps, one final step is added to the algorithm: the linking of broken features. This final step is also meant to eliminate isolated ridge pixels that are not part of a ridge line or that are in the vicinity of a line.

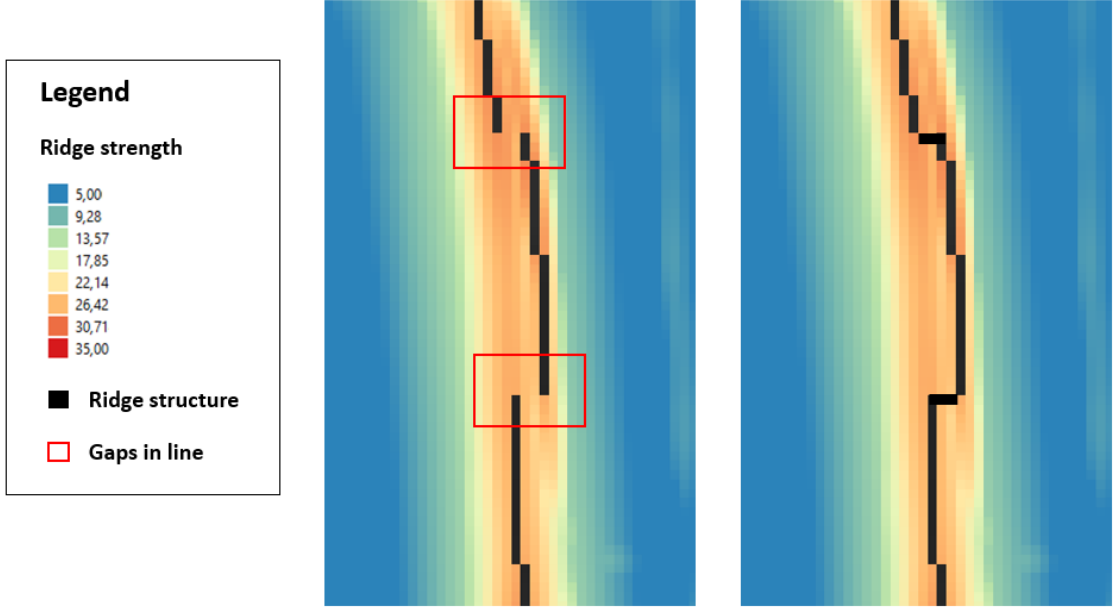


Figure 2.10: Connection discontinuities in ridge lines. The discontinuities are located in the red boxes.

The following processes are performed: isolated ridge pixels that are located in a distance L_{xx} of other ridge pixels are connected to each other, otherwise they are removed, and endpoints that are located in a distance L_{xx} of other ridge pixels are connected to each other. Isolated ridge pixels and endpoints are detected by analyzing the directly neighboring pixels. The distance L_{xx} is an adjustable parameter, that indicates the number of pixels that is considered for the connection process. For instance, when the L_{xx} is set to 4 whereby the DTM has a resolution of 0.5m, the lookup distance becomes 2m. The lookup distance L_{xx} can be set to a higher value, however when the value becomes too large, there is a higher probability that lines are created on places where no ridge structures are present. Thus the lookup distance should not be much larger than the step size that is used for the range of sigma values.

2.1.7. Algorithm II

In [37], a new type of SOG-kernels is introduced: the anisotropic second-order Gaussian kernel. This type of kernel is an elongated version of the isotropic second-order Gaussian kernel, hence it does more often resemble the rather elongated nature of ridges and edges. Therefore, it is argued that the usage of anisotropic kernels during the filter process significantly improves the performance of edge and ridge detectors [38], [37], [18]. The resulting ridge and edge lines become more contiguous, whereby the problem of crossings and junctions is reduced [18]. A second-order anisotropic Gaussian kernel is obtained by:

$$\hat{G}''(x, y) = \left(\frac{1}{\sigma\sqrt{2\pi}} + \frac{1}{\sigma^2} \right) \frac{1}{\sigma\sqrt{2\pi}} e^{\left(\frac{\phi}{2\sigma}\right)^2} \quad (2.3)$$

with:

$$\phi = [x \quad y] M^T \begin{bmatrix} \rho^2 & 0 \\ 0 & \rho^{-2} \end{bmatrix} M \begin{bmatrix} x \\ y \end{bmatrix} \quad (2.4)$$

and:

$$M = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \quad (2.5)$$

With this equation, the formula for the convolution can be established by:

$$\frac{\partial^2 I}{\partial v_\theta^2} = -I * \left(\frac{xcos\theta + ysin\theta}{\rho^{-2}\sigma^2} \hat{G}''(x,y) \right) \quad (2.6)$$

where rho (ρ) refers to the *anisotropy factor*. The default value for rho is 1, which results in an isotropic kernel. By increasing rho, the second-order Gaussian kernel becomes anisotropic, as is visualized in figure 2.11.

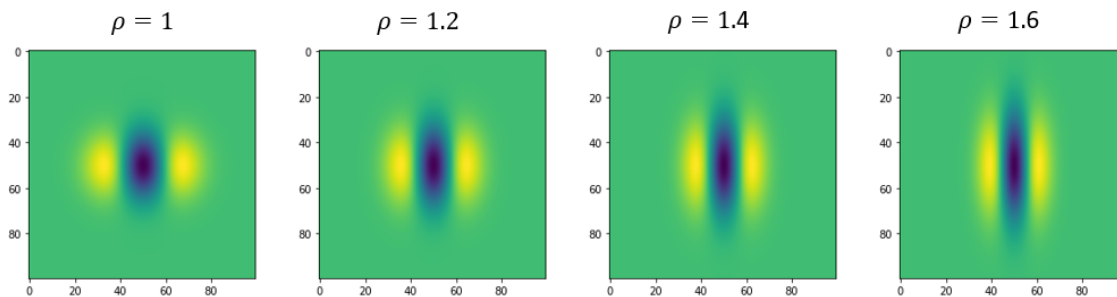


Figure 2.11: Impact of adjusting rho (ρ) on kernel

In order to test whether anisotropic kernels would also increase the performance of ridge detection based on DEM-data, a second algorithm has been developed. The proposed procedure for this algorithm is visualized in figure 2.12. In general, the second algorithm resembles the first algorithm, the only difference is that both isotropic and anisotropic second order Gaussian kernels are applied during the filter process.

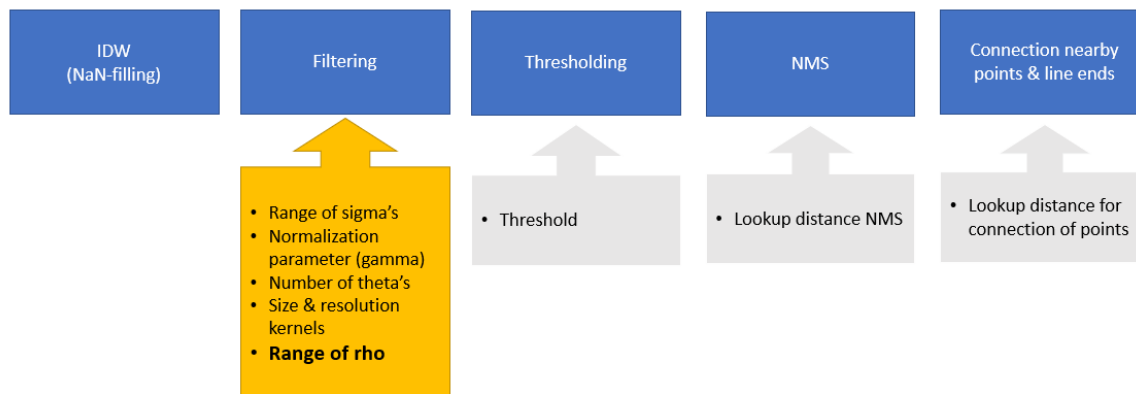


Figure 2.12: Procedure of algorithm II.

To visualize the impact of rho, a convolution with a schematic DEM and two values for rho is carried out. The results are shown in 2.13. The schematic DEM has a dimension of 100 by 100 steps. Two ridge

lines are created in the middle with a value of '1', which together form a cross. The rest of the DEM is set to '0'. A convolution is performed with $\rho = 1$ and $\rho = 1.5$. For both convolutions, a sigma value of '1' has been used and six different orientations for theta were applied. The result of the convolution is shown in the upper two figures. The figures reveal that a value of 1.5 for rho creates thinner lines in the response values compared to a value of 1 for rho. Furthermore, the center of the crossing is slightly brighter for $\rho = 1.5$ than for $\rho = 1$, thus the response values at the center are relatively higher for $\rho = 1.5$ than for $\rho = 1$. Therefore, it seems that the center of a crossing is more likely to be considered as a ridge structure when a larger value for rho is utilized. To analyze whether this is indeed the case, the entire procedure of the algorithm is used to produce ridge lines based on $\rho = 1$ and $\rho = 1.5$. Here, it should be noted that the range of response values varies significantly for the two values of rho, as becomes apparent in the legend of the upper two figures of 2.13. For rho equals 1, a response value ranging between approximately -10 to 35 is obtained, while for rho equals 1.5, the response values end up between -10 and 100. To compare the ridge lines based on the different values for rho, a threshold has been set at the 90th percentile of the response values. Hence ridge lines are developed based on the upper 10 percent of the response values. The resulting ridge lines are shown in the lower two plots of figure 2.13. Apparently, the kernels with $\rho = 1.5$ are able to detect the ridges at the crossing, while for $\rho = 1$ there are no ridge lines composed at the crossing.

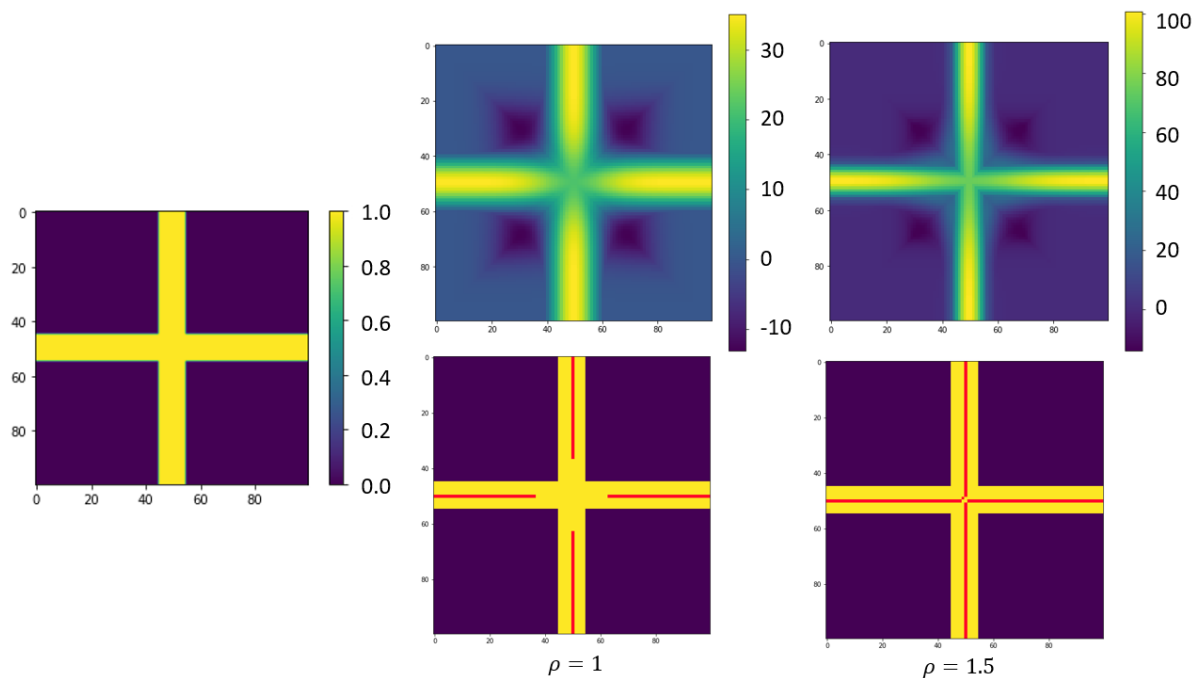


Figure 2.13: Impact of $\rho = 1.0$ and $\rho = 1.5$ on schematic DEM. Upper two figures: output of the convolution. Lower two figures: detected ridge lines.

A similar analysis has been performed on DEM-data, as such the performance of anisotropic kernels on actual ridge structures can be examined. Therefore, ridge structures were selected that together form a junction. The junction is located on a floodplain of the river Waal, in the surroundings of Nijmegen. The selected region encompasses an area of 4680 m² and is visualized in 2.14. A filter process has been carried out with a set of kernels of size: $\sigma = 1$ and six different orientations (θ). The entire procedure as shown in 2.12 was performed while using three different values for rho: $\rho = 1$, $\rho = 1.3$ and $\rho = 1.5$. The threshold was set at the 90th percentile of the outcome of the convolutions. The extracted ridge lines and the outcome of the three filter processes are visualized in 2.14.

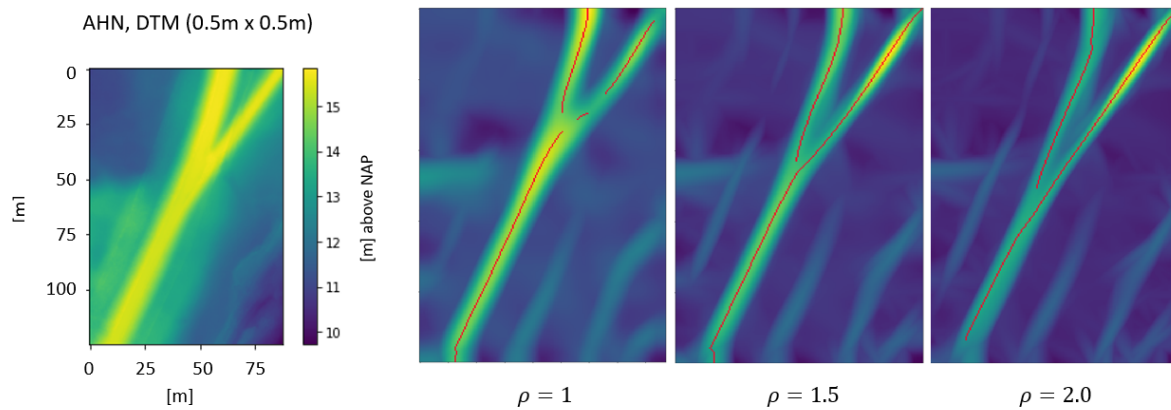


Figure 2.14: Impact of ρ , $\rho = 1.0$, $\rho = 1.5$ and $\rho = 2.0$ for DTM located in floodplain of river De Waal.

The figure reveals that a larger value for rho does not create connected ridge lines at the center of the junction. However, it does increase the length of the ridge lines close to the junction, hence reducing the length of the gap on the junction. Lopez-Molina et al. (2015)[18], propose to use a range of values for rho, and retain only the maximum value at each pixel. Based on the outcome visualized in figure 2.13, one would assume that a normalization must be applied to combine the results of various values for rho, as the figure is almost entirely covered by the kernel with $\rho = 1.6$. However, in the work performed by Lopez-Molina et al.(2015) [18] a normalization was been applied. Further research did not provide insight on the usage of a normalisation to combine the outcome of various anisotropic kernels for the purpose of ridge detection. Further research must be carried out to determine whether a normalization should be applied in order to correctly combine the outcome of various values for rho.

To analyze whether the kernel with the highest anisotropy factor does indeed overrule the other kernels, several test, a plot has been made that shows which kernel gave the highest response value to the convolution. For this plot, the same area has been used as in 2.14 and a range of $\rho = 1.0, 1.2, 1.4, 1.6$ has been used. The results, shown in 2.15, reveal that the kernels with the largest value for rho (1.6) do make up the largest part of the results, but the kernels with lower values for rho are not entirely overruled.

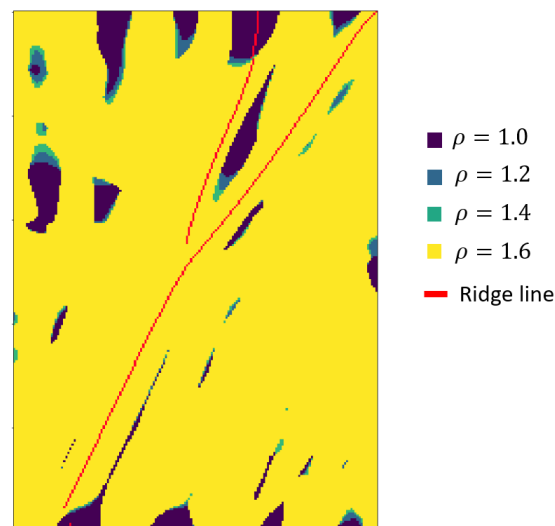


Figure 2.15: Analysis on which value for rho produced the maximum response value.

A downside of detecting ridges based on second-order anisotropic kernels is the so-called: *anisotropy stretch effect* [16], [39], [37]. The anisotropy stretch effect appears when large values for rho are used.

Due to a larger value for rho, line ends become a bit further extended compared to results for kernels with a lower value of rho. An example of the anisotropy stretch effect is shown in 2.16. Here, some unwanted spurious ridge pixels are generated around the schematic ridge structure in the center of the figure. The extended line ends pose a problem when two ridge structures are located in line with each other with a significantly lower part in between. In this case, the algorithm based on anisotropic kernels will probably place a single ridge line over the two ridge structures, whereby the gap is not taken into account. To avoid a large stretch effect, the kernels should not contain an anisotropic factor that is too large. In this study, the range for the anisotropy factor has been based on the work of Lopez-Molina et al. (2015) [18]. In their work a range of $\rho = 1.0, 1.2, 1.4$ and 1.6 has been used, which showed a considerable improvement in comparison with a model without anisotropic kernels. To analyze the impact on the resulting ridge lines, the algorithm has been tested on various regions with the selected range. Based on this analysis, it has been decided that the range of 1.0 to 1.6 , with a step size of 0.2 , result in ridge lines that were better connected on crossings and junction while the lines were not too much disturbed by the anisotropy stretch effect.

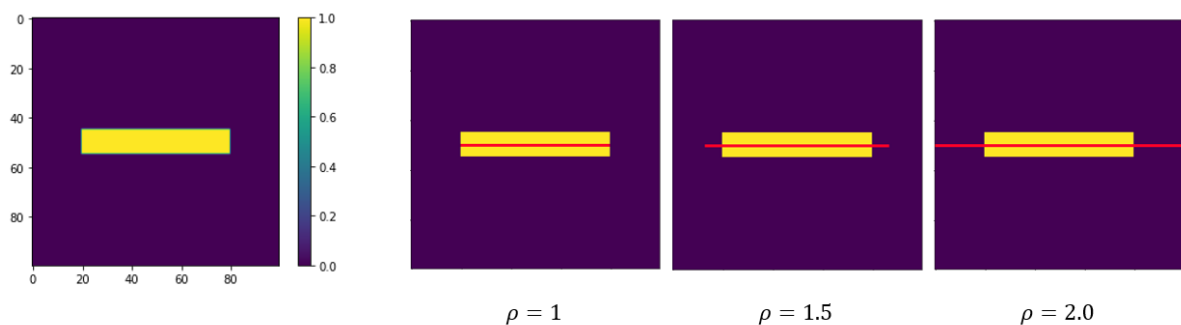


Figure 2.16: Stretch effect due to convolution with anisotropic kernels.

2.1.8. Algorithm III: Isotropy and Skeletonization

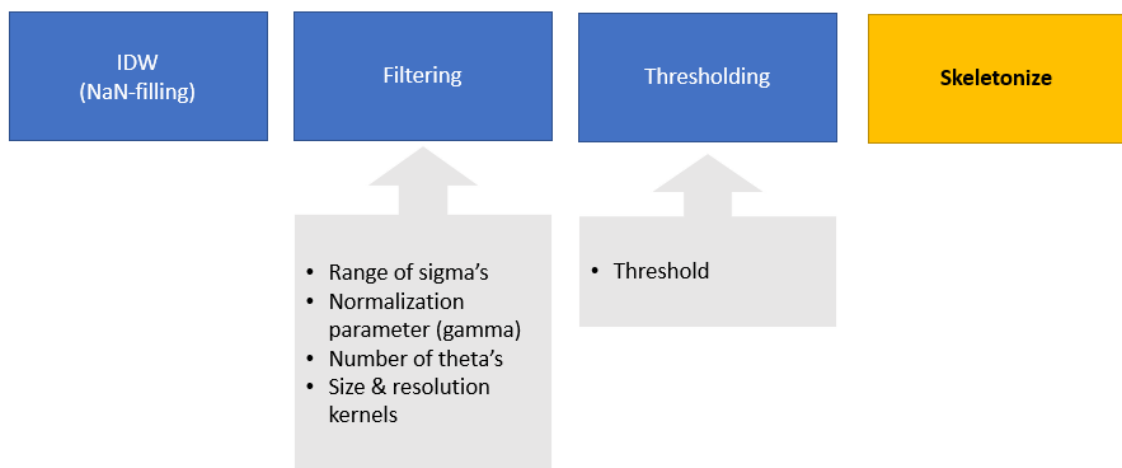


Figure 2.17: Procedure of algorithm III.

The third algorithm uses only isotropic kernels and instead of NMS skeletonization is applied to create lines from the pixels that fall above the threshold. The procedure for this algorithm is visualized in 2.17. Skeletonization creates a 1-pixel wide representations of binary objects. After the threshold process, a map binary map is obtained where '1' indicates the areas where ridge structures are detected and

'0' the regions where no ridge has been found. Skeletonization is able to identify the outer points of a binary object, thus in this case the regions where ridge structure has been detected. Subsequently the pixels on the boundaries are suppressed, which continues without breaking the connectivity of the corresponding object. An example of the procedure of skeletonization is shown in 2.18, where the binary image of a horse is thinned.

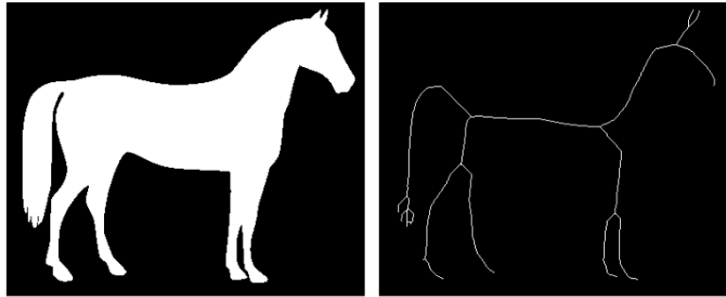


Figure 2.18: Skeletonize of image.

The method used for skelenization was proposed by Zhang and Suen in 1984 [40]. For this method the 3x3 local neighborhood of each pixel is examined to decide whether a pixel should be deleted, and thus set to '0', or retained. Two cycles are used to determine if a pixel should be deleted. For the first cycle a pixel is removed if the neighboring pixels, as shown in 2.19, satisfy all the following conditions:

- $2 \leq B(P) \leq 6$
- $A(P) = 1$
- $P_2 \vee P_4 \vee P_6 = 0$
- $P_4 \vee P_6 \vee P_8 = 0$

Here, $B(P)$ is the number of neighboring pixels with a value of '1', $A(P)$ is the number of transitions from '0' to '1' in the sequence $P_2, P_2, \dots, P_8, P_9, P_2$

P9 (x-1, y-1)	P2 (x-1, y)	P3 (x-1, y+1)
P8 (x, y-1)	P1 (x, y)	P4 (x, y+1)
P7 (x+1, y-1)	P6 (x+1, y)	P5(x+1, y+1)

Figure 2.19: Skeletonize pixel neighbors.

In the second cycle, pixels are removed when the local neighbors satisfy the following conditions:

- $2 \leq B(P) \leq 6$
- $A(P) = 1$
- $P_2 \vee P_4 \vee P_8 = 0$
- $P_2 \vee P_6 \vee P_8 = 0$

The two cycles are repeated until no more pixels are removed, hence only thin lines remain.

Skeletonization appears to be significantly faster than NMS. Analysis on different parts of varying sizes of the DTM revealed that skeletonization is approximately 100 times faster than NMS. However for the purpose of ridge line extraction, skeletonization suffers from a considerable drawback, as the procedure does not take into account where the highest areas are located on the DEM, neither does it consider the location of the maximum response values to the filter process. Ridge lines are often placed in the middle of regions that fall above the threshold. Therefore, lines are often placed slightly off the crest level of the ridge structure. Especially for large areas that fall above the threshold, the ridge lines are likely to be placed not exactly on the highest part of the ridge.

2.1.9. Overview algorithms

An overview of the three algorithms is visualized in 2.20. The differences between the algorithms are highlighted. To summarize, the first algorithm is based on isotropic second-order Gaussian kernels and Non-maximum suppression (NMS). The second algorithm is the algorithm utilizes one extra type of kernel: the anisotropic Gaussian kernel during the filter procedure. The third algorithm is the algorithm that makes use of skeletonization instead of non-maximum suppression. The python-code of the three algorithms can be found in A, B and C.




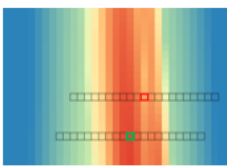
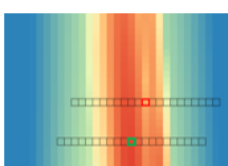
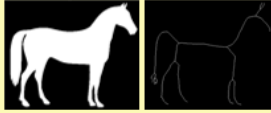
	A1	A2	A3
Kernel types	Isotropic 	Anisotropic and isotropic 	Isotropic 
Formation lines	NMS 	NMS 	Skeletonization 

Figure 2.20: Overview three algorithms (A1: algorithm 1, A2: algorithm 2 and A3: algorithm 3). The highlighted boxes indicated the difference compared to the first default algorithm.

2.2. Post-processing in QGIS

The output of all three algorithms is a binary raster layer, where '0' indicates the pixels that are not classified as ridge line, and '1' the pixels that are classified as ridge line. To implement the raster files into D-HYDRO, the files must be transformed into a shape file and subsequently a pliz-file. Various post-processing steps are necessary in order to transform the output of the algorithms into the correct format for D-HYDRO. In figure 2.21, only the most relevant post-processing steps are visualized, the entire process can be found in the appendix D. In this section the most relevant steps are explained, starting with the removal of the pixels in water bodies. In a DTM, water bodies are filtered out of the data, resulting in large gaps in the DTM. In this case, a water body refers to any body of surface water such as a river or lake. Interpolation based on Inverse Distance Weighting does fill up these holes. However, the interpolation results in the creation of sudden changes in height, which are interpreted as ridge lines by all algorithms. To ensure that the ridge lines that are located in water bodies are not used for further analysis, all ridge pixels in water bodies are set to '0'.

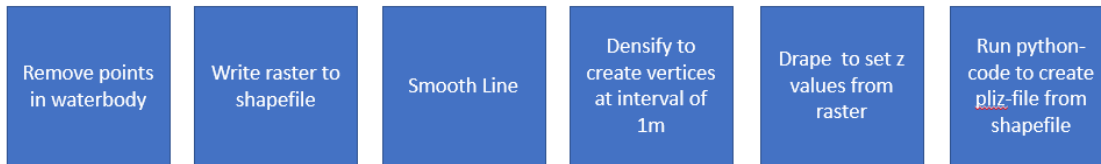


Figure 2.21: Procedure post-processing in QGIS.

The second step in the post-processing process is the transformation from raster to shape file, hence creating lines from the pixels with the value '1'. The lines will be used to extract the elevation at the top of the detected ridge lines. To extract the data, points (vertices) are created on the lines with an interval of 1 meter as is visualized in figure 2.22. To ensure this interval is approximately even, a smoothing step is applied. Subsequently, the lines are densified to create vertices at every 1 meter of the line. In the next step, the elevation is extracted from the DTM at every vertice. After the processing in QGIS the ridge lines contain data on the elevation and exact location at all vertices.

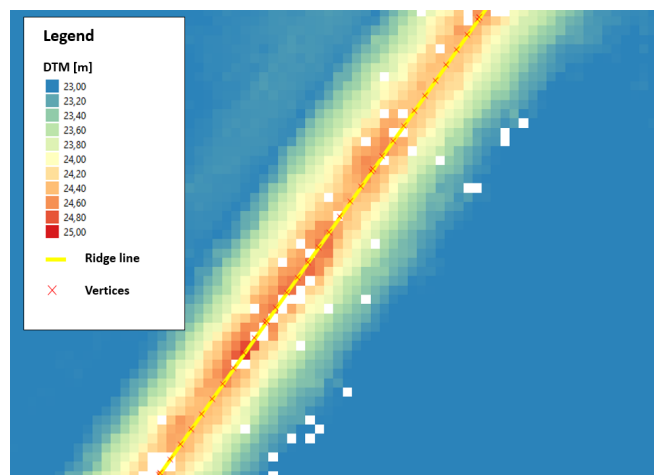


Figure 2.22: Ridge line with vertices that indicate the location where elevation is extracted.

It should be mentioned that some vertices are located in no-data regions, which is for instance the case for the vertices in the lower left corner of figure 2.22. These vertices obtain a value of '0' if no further processing is carried out. Therefore, a final step is performed to remove all vertices with a value of '0'. The drawback of removing vertices with a value of '0' is that some lines might shift a bit to another place, as is visualized in figure 2.23. Consequently, in D-HYDRO ridge lines might end up on the wrong location. Aside from the removal of NaN-values from the data, the python code does also create a pliz-file from the shape file. The pliz-file can be used as input for D-HYDRO.

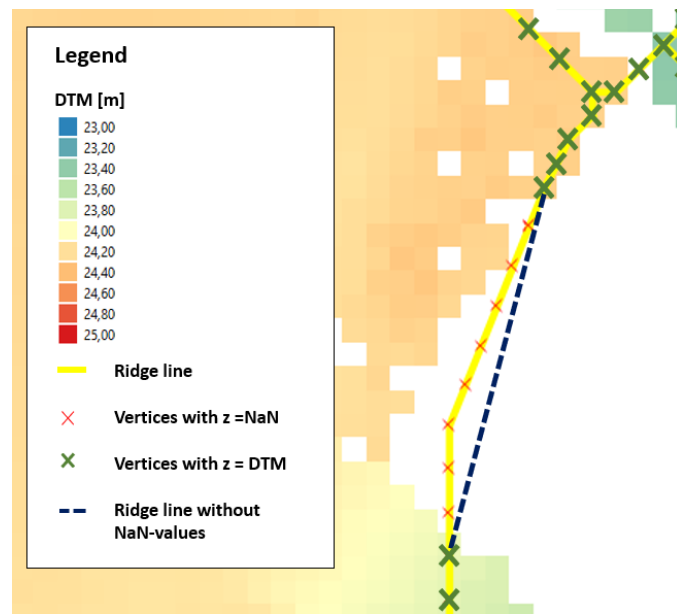


Figure 2.23: Impact of removal NaN-values on location of ridge line.

3

Verification manually drawn ridge lines

Quality evaluation of the binary outcome of the algorithms is not a straightforward task, although it comes down to comparing a given ground truth to the resulting raster files. The difficulty is in the construction of the ground truth, which is subject to the interpretation of the composer. Since no single method is able to evaluate all requirements of the ridge lines, a combination of verification methods has been applied. In this chapter, manually drawn ridge lines have been compared to the outcome of the algorithms. Therefore, a small area in the surroundings of the Roer is selected. Subsequently two verifications were performed. Firstly the location of the lines was directly compared to the manually drawn lines and secondly a buffer zone was used to verify the performance of the ridge lines with respect to the manually drawn lines.

3.1. Dimensions ridge structures

Specific requirements of a ridge structure such as a minimum height or slope are in general not clearly defined. Usually these requirements are established by the modeller of the hydrodynamic model. However, in order to verify the performance of the algorithms, several requirements must be set. For this study, the following requirements were established: the algorithms should be able to detect ridge structures with dimensions of 3.5 m to 15 m in width. These dimensions were selected based on two aspects. Firstly, the value of 3.5 m equals the resolution of the hydrodynamic model that functions as an approximation of the ground truth for the verification in D-HYDRO. Secondly, the value of 15 m was selected based on analysis on the kernel dimensions that would result in an acceptable calculation time. For larger ridge structures, a larger value for sigma must be used, hence the dimensions of the SOG-kernel should increase. Furthermore, the minimum difference in elevation that should be present is used of 1:7. As such, a ridge structure with a width of about 3.5 m must have a difference in elevation of at least 25 cm in order to be detected by the algorithm. For ridge lines wider than 3.5 m meter the minimum elevation increases according to the required minimum slope.

3.2. Parameters algorithms

The dimensions of the ridge structures that must be detected by the algorithms form the basis of the selection of the majority of the parameters. An overview of all parameters is shown in 3.1. Dimensions that must be detected were set to set to 3.5 m to 15 m. Based on 2.1, it was found that a kernel size σ of 0.2 to 1.0 should be used. An interval of $\sigma = 0.2$, has been applied thus in total 5 kernel sizes were used for every algorithm. The selection of some parameters is already discussed in the previous chapter. For instance the value of the normalization parameter (γ), which has been set to 0.75. The number of orientations (θ) is a trade-off since a larger number increases the accuracy but does also increase the calculation time. In order to obtain a reasonable calculation time, while obtaining accurate ridge lines, the number of orientations has been set to 6. The resolution of the kernels and the size of the largest kernel prescribes the required dimensions of the kernels. Here a resolution has been used of 0.1, since the largest kernel has a value of σ , the dimensions were set to -5 up to 5. As such, the Gaussian function is covered by the dimensions of the kernel. For the second algorithm a range of 1.0 to 1.6 has been utilized with an interval of 0.2 for ρ , as was proposed in the work of Lopez et al.

(2015) [18]. The lookup distance for NMS has been set to 4, with a DTM resolution of 0.5 x 0.5m, the lookup distance becomes 2 m. Larger values for the NMS might suppress ridge lines which are located close to each other. The lookup distance for closing broken lines was established on 3 pixels, thus broken ridge lines with a distance in between of 1.5 m were reconnected. Finally, the thresholds were established by analyzing some initial results. Several cross sections were extracted and compared to the outcome of the algorithms, which led to the conclusion that a threshold of 15 had to be used for the first and third algorithm. For the second algorithm a value of 35 that was required to detect ridge structures of 3.5 m in width with a minimum difference in height of approximately 25cm.

Table 3.1: Parameters.

	Algorithm 1	Algorithm 2	Algorithm 3
Size (σ)	0.2, 0.4, 0.6, 0.8, 1.0	0.2, 0.4, 0.6, 0.8, 1.0	0.2, 0.4, 0.6, 0.8, 1.0
Normalization (γ)	0,75	0,75	0,75
Number of orientations (θ)	6	6	6
Dimensions	[-5:5]	[-5:5]	[-5:5]
Resolution	0.1	0.1	0.1
Anisotropy (ρ)	1.0	1.0, 1.2, 1.4, 1.6	1.0
L_{NMS}	4	4	4
L_{CBL}	3	3	3
Threshold	15	35	15

3.3. Study site and data

The data used for this study refer to the surroundings of the river Roer in the province Limburg (The Netherlands). The river basin of the Roer in The Netherlands is a unique area since it is one of the few rivers that is able to meanders freely through the landscape. The meandering created at some locations old and recent cut-off bends and steep river banks, which are visible on the DTM of the area [41].



Figure 3.1: Area utilized for verification.

For the comparison between the automatically detected ridge lines and the manually drawn lines, an

area of 1 x 1.4 km has been selected which is located south of the village Melick. The elevation in the selected area varies between 10 and 35m above NAP. Various ridge structures of different sizes were encountered on the DTM of the region, hence making it perfectly suited to analyze the performance of the algorithms. Data for the selected region was obtained from the national DTM (Actueel Hoogtebestand Nederland, AHN3), that is created from LiDAR data at a 0.5 x 0.5m resolution, which is available at [42]. It should be noted that the data has a stochastic error and standard error of 5cm [43].

In table 3.2, an overview is provided of the reference and test dataset. The ridge lines resulting from the algorithms are referred to as the test datasets. The reference data represents an approximation of the ground truth. Two files were used as reference dataset: (1) ridge lines that were specifically drawn for this study for the selected region and (2) ridge lines that were composed by the water authority of Limburg. The former has been drawn manually based on empirical analysis of the DTM. Several cross sections were analyzed to ensure that the local difference in height fulfilled the required dimensions that were set. In total, all drawn lines summed up to a length of 20.3 km. The second reference data set, with the lines that were composed by the water authority, was composed for the entire D-HYDRO model that has been utilized for this study. The lines have been clipped for the selected region, hence resulting in a total length of 10.9 km. An inspection of the lines composed by the water authority led to some conclusions. Firstly, on several locations no lines were drawn, while a significant difference in elevation on these locations was present. And secondly, a number of lines appear to be drawn purely based on the presence of roads, which were not necessarily higher than the surrounding area. Despite these deficiencies, the data set will be taken into account, since a significant portion of the drawn lines appears to be correctly drawn. The five data sets from table 3.2, were transformed into binary raster files that contained a resolution of 0.5 x 0.5m, which resembles the resolution of the DTM that was utilized. In the binary raster files, '1' indicated a pixel that is part of a ridge line and '0' represented a pixel that is not part of a ridge line.

Table 3.2: Datasets MD.

Reference data	Test data
1. Manually drawn ridge lines	1. Ridge lines Algorithm 1
2. Ridge lines composed by Water Authority	2. Ridge lines Algorithm 2
	3. Ridge lines Algorithm 3

3.4. Results algorithms

After running the algorithms for the selected region an empirical analysis was carried out to get a general idea of the outcome. The four requirements that were set for the algorithms have been used for this analysis.

1. The algorithms should not be affected by noise.

In general the algorithms did not generate a large amount of lines due to the presence of noise on the DTM. However, some responses to noise were observed. For instance the ridge lines from algorithm 1, which are visualized in 3.2, show some response to abrupt change in the DTM that are not part of a ridge structure. The response to noise can be omitted by avoiding the usage of kernels with a very small sigma value. Furthermore, a larger threshold could also be applied to suppress noise on the resulting ridge lines.

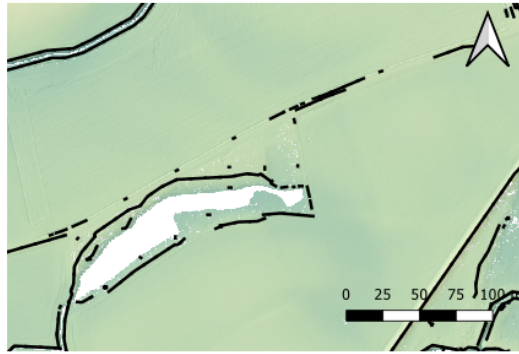


Figure 3.2: Ridge lines algorithm 1.

2. The algorithms should be able to detect ridge structures of various sizes.

Due to the usage of a range of sigma values and by applying a normalization, the outcome of the algorithms were all capable in detecting ridges of different sizes. An example is given in figure 3.3. An embankment is present on the left part of the figure, which contains a difference in height of 2.5 m. While the ditch visualized in the middle of the figure consist of a difference in height of about 0.5 m. Lines were drawn on both structures by all the algorithms.

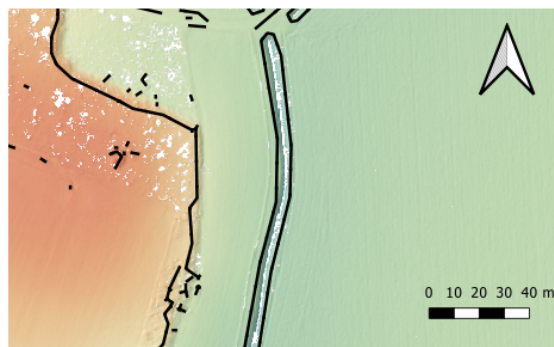


Figure 3.3: Ridge lines algorithm 1.

3. The ridge lines that are placed by the algorithms should be located exactly on, or at least in the vicinity of, the crest level of the ridge structure.

As expected the lines of the third algorithm did not always end up at the crest of the ridge structure since skeletonization has been applied in stead of NMS. An example is visualized in 3.4. A cross section has been made at the red dotted line in figure 3.4 to analyze the elevation at this location (figure 3.5). The red dot in the figure visualizes the location of the ridge line from the first algorithm, hence revealing that the line has been placed exactly on top of the ridge structure. The line of the second algorithm, indicated by the blue dot, is placed 1m off the crest level. The line of the third algorithm is placed approximately 5m next to the highest part of the ridge structure. The difference in elevation found at this location is relatively high, therefore a large amount of pixels obtained a value above the threshold. The process of NMS does take into consideration the height of the response values to the convolution, whereas skeletonization puts a line in the middle of an area that fell above the threshold and does not account for the height of the response values.

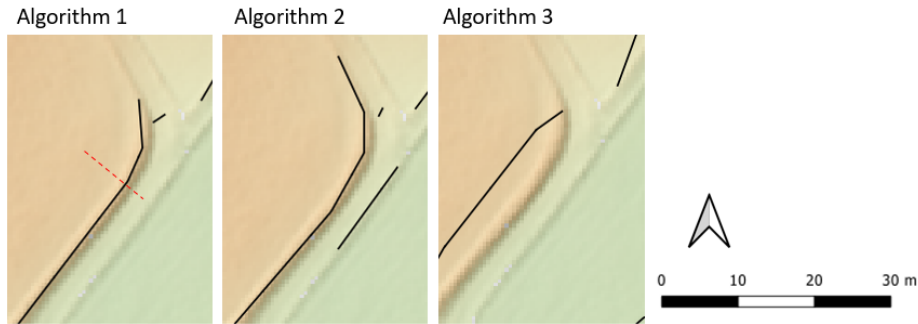


Figure 3.4: Analysis outcome of algorithm 1, 2 and 3. Red dotted line indicates cross section visualized in 3.5.

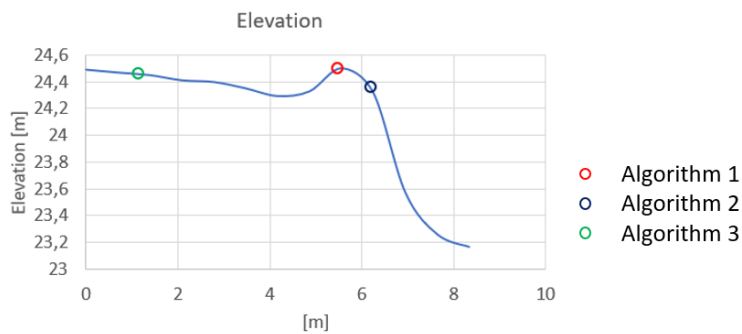


Figure 3.5: Cross section, with dots indicating the location of the ridge lines.

The difference between the first and the second algorithm originates from the usage of anisotropic kernels in the second algorithm. Due to the elongated form of the anisotropic kernels the second algorithm performs slightly worse on bended ridge structures, as is also visualized in figure 3.6. The figure shows that the first algorithm does better follow the bend of the ditch, while the second algorithm cuts off a part. Hence, the lines of the second algorithm are situated on a lower part in comparison with the lines of the first algorithm.



Figure 3.6: Bended ridge structures of outcome of algorithm 2.

4. The algorithms should produce continuous ridge lines, without unnecessary gaps.

Analysis on the outcome of the algorithms revealed that principally the second algorithm generated continuous ridge lines. Figure 3.7, reveals the difference between the outcome, whereby only the second algorithm detected the entire ridge structure. As shown, a gap was found on the left side of the figure in the lines of algorithm 1 and 3, which was not encountered in the lines of the second algorithm. The difference in elevation at this location was found to be 30 cm, hence the structure should have been detected by the algorithms. Based on the first evaluation of the outcome of the algorithms, it was

stated that the second algorithm did more often produce continuous ridge lines in comparison with the other two algorithms. Hence suggesting that the elongated form of the anisotropic kernel does increase the performance of filter process.

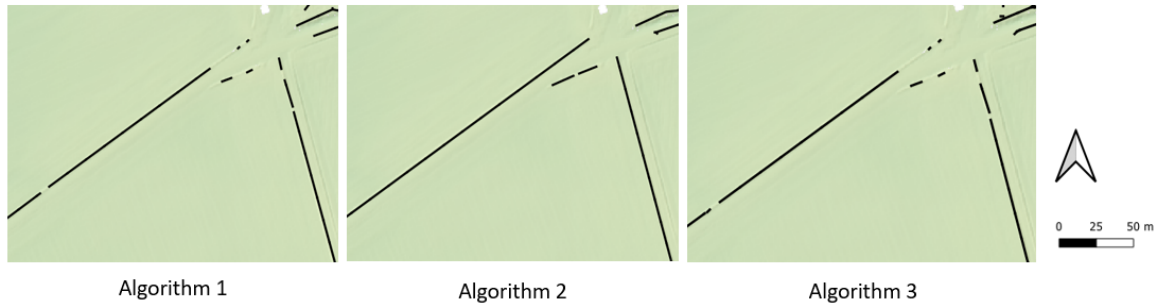


Figure 3.7: Ridge lines algorithm 1, 2 and 3.

3.5. Results verification

3.5.1. Precision, Recall, F1-score and MCC

The performance of the algorithms was first verified by calculating four measures: the precision, recall, F1-score and Matthews correlation coefficient (MCC). Therefore, firstly a confusion matrix was composed, which is shown in table 3.8. In figure 3.9, an example of the classification procedure for the confusion matrix is visualized. The red pixels indicate the false negative pixels, which are the ridge pixels that were not detected by the algorithm. The orange pixels represent the pixels that were incorrectly classified as ridge pixels. The green pixels indicate the true positive pixels that were correctly assigned by the algorithm as ridge pixel. The true negatives are not visualized in the figure but are represented by all the other pixels.

	Predicted no ridge	Predicted ridge
Actual no ridge	True negative (TN)	False positive (FP)
Actual ridge	False negative (FN)	True positive (TP)

Figure 3.8: Convolution matrix

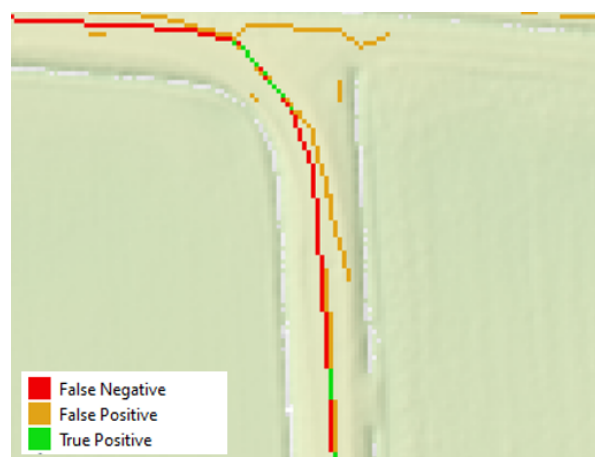


Figure 3.9: Pixel classification of ridge lines for convolution matrix.

The four measures are computed by:

$$Precision = \frac{TP}{TP + FP} \quad (3.1)$$

$$Recall = \frac{TP}{TP + FN} \quad (3.2)$$

$$F_1Score = 2 \frac{Precision * Recall}{Precision + Recall} \quad (3.3)$$

$$MCC = \frac{TP * TN - FP * FN}{\sqrt{(TP + FP) * (TP + FN) * (TN + FP) * (TN + FN)}} \quad (3.4)$$

where precision represents the number of actual positive cases out of all predicted positives. Recall is the percentage of detected positive cases out of all actual true cases. Usually the accuracy can also be calculated to evaluate the classification performance. However, since the binary raster files are severely imbalanced it has been decided to leave accuracy out of the analysis. The F_1Score is the weighted average of precision and recall, hence taking both positive and negative predictions into account. F_1Score is often considered more useful than the accuracy, especially for large data sets, such as the raster files used in this evaluation. However, the true negative values (TN) are not considered in the F_1Score , therefore a final metric has been calculated: the MCC. This metric does also account for the true negatives (TN) and is especially suited for unbalanced data sets [44].

Results for the comparison between the manually drawn ridge lines and the outcome of the algorithms is shown in in table 3.3. The table reveals a significant difference between the first two algorithms and the third algorithm. The first two algorithms score higher on every measure. Thus, the process of skeletonization results in ridge lines that are not located exactly on the crest of the ridge structure. The results for the first and the second algorithm are relatively similar. While the first algorithm obtained higher results for the precision and F_1Score , the second algorithm scored higher on the recall measure. The MCC is approximately similar for both algorithms. Hence no significant difference was found when also the true negatives (TN) were considered.

Table 3.3: Manually drawn.

	Algorithm 1	Algorithm 2	Algorithm 3
Precision	0,16	0,15	0,08
Recall	0,19	0,20	0,10
F_1 -Score	0,18	0,17	0,08
MCC	0,17	0,17	0,08

In table 3.4 the comparison between the algorithms and the ridge lines that were composed by the water authority is shown. The values in the table indicate that the performance of the three algorithms is significantly lower when comparing the outcome to the dataset provided by the water authority. No significant differences can be observed in the resulting values, hence based on these values no firm conclusions can be made regarding the performance of the algorithms. The low results can be ascribed to the mediocre quality of the lines that were composed by the water authority.

Table 3.4: Water authority.

	Algorithm 1	Algorithm 2	Algorithm 3
Precision	0,02	0,02	0,02
Recall	0,05	0,06	0,05
F_1 -Score	0,03	0,04	0,03
MCC	0,03	0,03	0,03

3.5.2. Positional accuracy analysis based on Goodchild and Hunter

A very fine raster resolution has been used for the calculation of the precision, recall, F1-score and MCC. Hence the chance that ridge pixels of the algorithm were located exactly at the same place as ridge pixels from the reference data sets was very low. Therefore, an addition analysis is applied: the positional accuracy method for linear features as proposed by Goodchild and Hunter (1997) [45]. This method uses buffer zones around the reference data sets to establish the distance between the test and reference data set.



Figure 3.10: Buffer area with buffer distance of 5 m around manually drawn ridge line.

For this method, the following procedure is applied: firstly a target percentile y is established. The target percentile refers to the percentage of line length that should end up in the buffer zone. Subsequently the buffer zone is created. Therefore a certain buffer distance is (x) is used. If the target percentage is not reached, thus if the percentage of line that falls within the buffer zone is larger or smaller than the target percentage, a new buffer distance is set. This process continues until the buffer distance determined for which the target percentage is obtained. In 3.10, an example is given of ridge pixels that were determined by one of the algorithms and a buffer zone around a reference data set with a buffer distance of 5 m.

Figure 3.11 shows the results of the Goodchild and Hunter analysis for the comparison between the manually drawn lines and the outcome of the three algorithms. The graph shows that a significantly higher buffer distance is required for the third algorithm to obtain the same target percentile as for the first and the second algorithm. This confirms the conclusion that was drawn based on the analysis of the precision, recall, F1-score and MCC. Apparently the ridge lines of the third algorithm have not been located exactly on top of the ridge structures. A small dissimilarity is observed between the first two algorithms, whereby the second algorithm required a lower buffer distance in order to obtain the target percentages. This indicates that the lines of the second algorithm outperforms are placed closer to the lines that were drawn manually. However, the difference is very small, therefore is it difficult to state whether the second algorithm is performing better than the first.

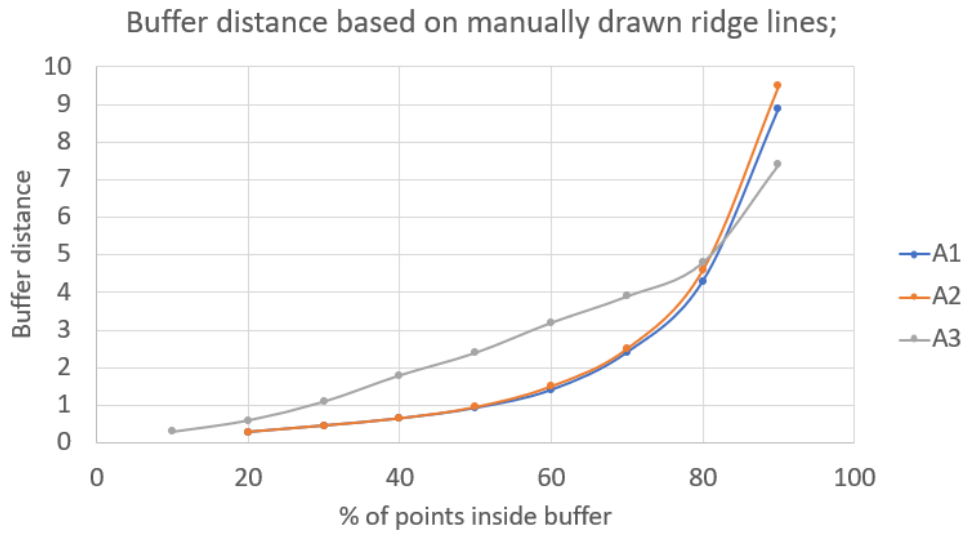


Figure 3.11: Results Goodchild and Hunter analysis. Comparison manually drawn lines vs outcome algorithms.

The graph in figure 3.12 shows the comparison between the lines composed by the water authority and the outcome of the three algorithms. A significantly larger buffer distance was utilized to obtain similar target percentages as for the comparison with the manually drawn lines. Hence, it can be stated that the automatically extracted ridge lines are situated much further away from the ridge lines composed by the water authority than from the lines that were drawn manually. Furthermore, only marginal differences are observed between the algorithms, therefore no firm conclusions can be drawn on figure 3.12.

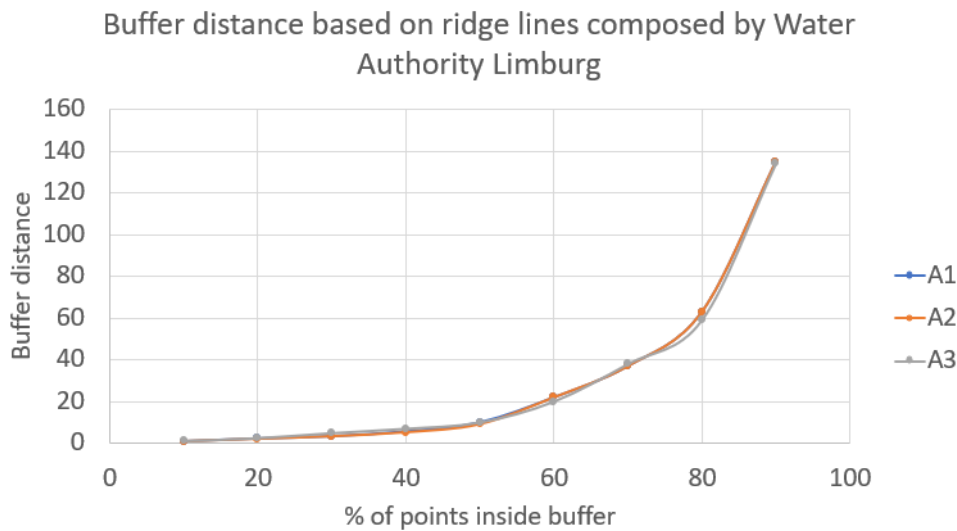


Figure 3.12: Results Goodchild and Hunter analysis. Comparison ridge lines composed by Water Authority Limburg vs outcome algorithms.

Verification D-HYDRO

For the second part of the verification an analysis is carried out in D-HYDRO. The verification in D-HYDRO is principally meant to analyze the performance of the automatically extracted ridge lines in a hydrodynamic model. For this part, a flood model is used that has been developed in D-HYDRO to simulate an extreme discharge event for the river Roer. The automatically extracted ridge lines were included into the model and validated by comparing the outcome to aerial photos and to a model which is an approximation of the ground truth. The latter is created by using a much finer resolution compared to the other models. Before going into detail about the formation of the different models, first background information of the D-HYDRO model is provided.

4.1. D-HYDRO model for De Roer

The D-HYDRO model that is utilized for the verification is developed for the river De Roer. The river Roer is a tributary to the Meuse river and is partly situated in The Netherlands, Germany and Belgium. The length of the river is approximately 165 km and its catchment area is about 2400 km². The model covers an area of 25 km². Only the Dutch part of the model is utilized for analysis, which encompasses an area of 18 km².

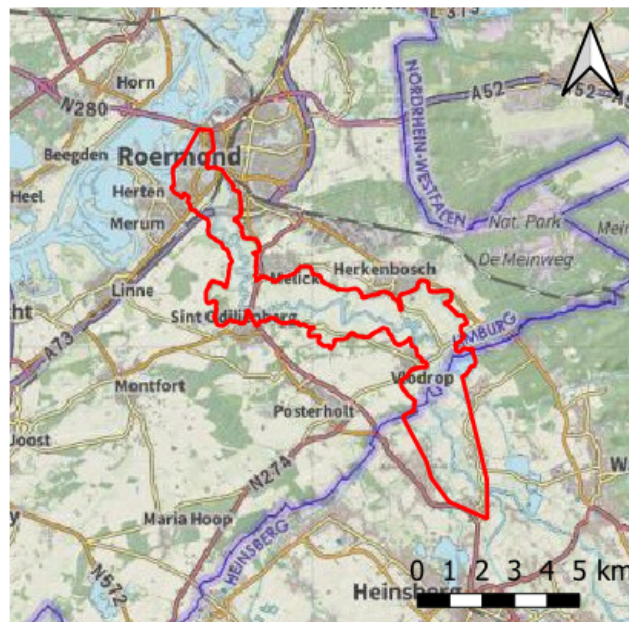


Figure 4.1: Model De Roer.

The extreme discharge event that was simulated for this study occurred between the 6th to the 16th of January 2011. Figure 4.2, shows the inflow hydrograph that was used as upstream boundary condition. As shown in the graph, two peak discharges took place during this period, whereby the first peak reached a maximum discharge of 114 m³/s and the second a maximum of 135 m³/s.

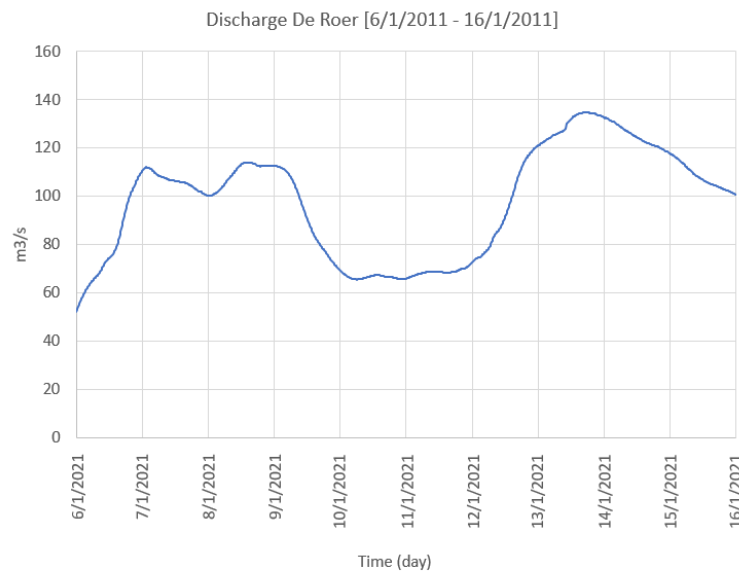


Figure 4.2: Upstream discharge Q [m³/s] during the 6th until the 16th of January 2011.

The D-HYDRO model utilized for this study, concerns a coupled 1D- 2D model, whereby 1D-shallow water equations are solved. Among the one dimensional input variables is the upstream river discharge visualized in figure 4.2. Other boundary conditions include the water depth downstream of the river and the discharge of tributary rivers. Furthermore, cross sections of the riverbed and structures, such as culverts, weirs and orifices were added as one-dimensional components to the model. The operation of the weirs over time has also been included. The two dimensional input variables of the model include the elevation, the surface friction and the ridge lines, which are referred to as fixed weirs in the model. Elevation data was obtained from the AHN3 (resolution: 0.5 x 0.5 m). The elevation is integrated into the model by determining the average for every grid cell. The time interval used for the model was set to 60 seconds. Precipitation has not been included into the model. The manner in which the ridge lines are implemented into the model is outlined in the next section.

Fixed weirs

The functioning of the ridge lines in the model depends largely on the manner on which the lines are implemented into the model. In D-HYDRO the lines are aligned on the lines on the cell edges of the computational grid network. This processes is referred to as grid snapping. Two examples of grid snapping are visualized in figure 4.3. A ridge line is only incorporated into a model when it traverses a flow link, where flow links represent the connections between the cell centers. A ridge line is not aligned on the grid cell when it does not intersect with a flow link, as is visualized in the right grid cell of figure 4.3. Hence the line is not taken into account in the model. When the ridge line does intersect with a flow link, a weighted average of the height is taken. Therefore only the two nearest elevation measurements are taken into account for the calculation of the height of the projected ridge line. For example, in the left grid cell of figure 4.3 only the green points with a value of 1.0 and 2.0 are considered. Subsequently a weighted average of these values will result in a value of 1.1 for the projected ridge line. If multiple ridge lines intersect with a flow link, then the highest value of the interpolated values is taken. In D-HYDRO the aligned ridge lines are referred to as fixed weirs. At low water levels the fixed weirs form an obstruction, hence the weirs block the flow from one grid cell to the neighboring cell. The fixed weir commences to function as a weir When the water level exceeds the height of the crest level of the fixed weir.

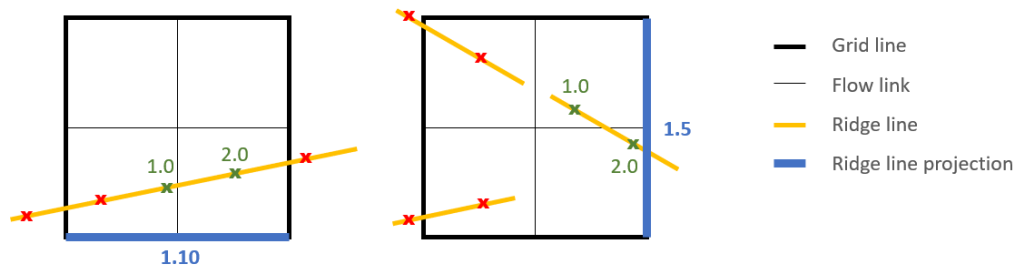


Figure 4.3: Projection of height on grid faces by means of calculating weighted average.

Discharge over fixed weir

To determine the discharge over a fixed weir, two approaches can be used in D-HYDRO: the so-called "Lookup-table" ("Tabellenboek") approach, and the 'Villemonthe' model, which is the default option. For this study the default option has been selected for the calculation of the discharge over fixed weirs. The Villemonthe approach is an entirely empirical based formulation, which is based on the analysis of a large number of measurements. The proposed formula by Villemonthe is:

$$Q = C_{d0} Q_c(E_1) \sqrt{1 - \max(0, \min(1, (\frac{E_2}{E_1})^p)} \quad (4.1)$$

Here, Q represents the discharge across the weir [m^3/s], C_{d0} is the resistance coefficient of the weir [-] which depends on the weir's vegetation and flow conditions over the weir. The variable p is the power coefficient [-] that is based on experimental data and takes into account the geometry of the weir. The variable $Q_c(E_1)$ is the theoretical value for discharge over the weir in case of critical flow [m^2/s], which is calculated according to:

$$Q_c = \frac{2}{3} E_1 \sqrt{\frac{2g}{3} E_1} \quad (4.2)$$

The variables of E_1 and E_2 represent the upstream and downstream energy head, which can be calculated according to:

$$E_1 = h_1 + (\frac{U_1^2}{2g})^p; \quad (4.3)$$

$$E_2 = h_2 + (\frac{U_2^2}{2g})^p \quad (4.4)$$

Here h_1 and h_2 represent respectively the upstream and downstream water level measured from the crest level of the weir, as is shown in figure 4.4, and U_1 and U_2 respectively the upstream and downstream flow velocity [m/s].

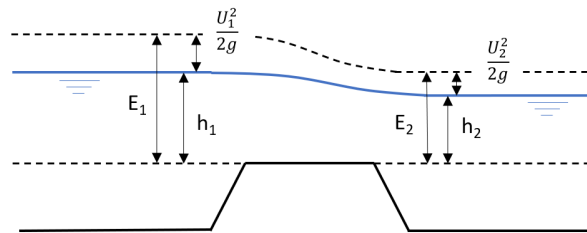


Figure 4.4: Flow over weir

Further research could be performed on the relation between abrupt changes in height and energy loss. Insight on the energy loss would help defining the dimensions of the ridge structures that must be incorporated into the hydrodynamic model. Hence, small differences in elevation that have a negligible impact on the water flow would not be detected and incorporated into the model. The relation between energy loss and ridge structure dimensions has been left outside the scope of this study.

4.2. Development hydrodynamic models for validation

Six variations of the hydrodynamic model have been developed for the verification of the ridge lines. An overview of the six models is shown in table 4.1. The models were based on an angular grid network, of which five models contained a resolution of 25 x 25m and one model that contained a finer resolution of 3.57 x 3.57 m, as such, exactly 7 x 7 grid cells could fit in one coarse grid cell. The latter model could function as an approximation of the ground truth. One out of the five coarse grid models did not contain any ridge lines. Three coarse grid models contained the ridge lines of one of the three algorithms. The final coarse grid model obtained the ridge lines that were manually drawn by the water authority of Limburg. The outcome of the five coarse grid models has been used as test data, while the outcome of the fine grid model was used as reference data. In addition to the six developed models, data has been used on the total inundated area. This dataset was developed by utilizing aerial photos that were taken between 9:00 and 11:00 at the 15th of January 2011.

Table 4.1: Datasets.

Reference data	Test data Model [25 x 25m]
1. Model [3.57 x 3.57m] without ridge lines	1. Ridge lines Algorithm 1
2. Total inundated area (9:00 - 11:00 at the 15th of January 2011)	2. Ridge lines Algorithm 2
	3. Ridge lines Algorithm 3
	4. Ridge lines Water Authority
	5. Without ridge lines

For the production of the first three models from the test data, a significant difference in calculation time was observed between them. The lines were produced for a clipped area of about 34 km². The third algorithm required only 14 hours to generate lines for the clipped region, whereas the first and the second algorithms needed at least 17 days. Fortunately all algorithms were able to run parallel, hence the calculation time was significantly reduced. The difference in calculation time between the first two algorithms and the third algorithm can be ascribed to the process of non-maximum suppression. Furthermore, the difference between the first and the second algorithm is probably the result of the usage of a different threshold, hence slightly less pixels had to pass the NMS-procedure for the second algorithm.

The test and reference data sets were compared on two aspects: the total inundated area and the water depth. For the latter only the model with the fine resolution was used as reference data. For the

comparison, some pre-processing was required. Therefore, the outcome fine-grid model was transformed into large grids. To do so, the sum of the water depth was calculated over 7 x 7 grids, hence a 25 x 25 m grid network was created that exactly resembled the coarse mesh. As such, the comparison between the reference and test data could be carried out.

4.3. Results total inundated area

Firstly, the total inundated area according to the aerial photos was compared to the outcome of the fine grid simulation. At 10:00 on the 15th of January, approximately 6.600 km² was inundated according to the fine grid model, while according to the dataset based on the aerial photos an area of 6.644 km² was inundated. Analysis revealed that 5,904 km² was predicted to be inundated according to both datasets. An area of 0,70 km² was only inundated in the prediction of the fine grid model, while 0,73 km² was only inundated in the dataset based on the aerial photos.

Subsequently the reference datasets were compared to the five test datasets from table 4.1. Therefore the total inundated area at 10:00 on the 15th of January was compared by calculating the total over- and underestimated area. The overestimated area indicates the area that was predicted to be inundated according to one of the five algorithms but not according to the reference dataset, while the underestimated area indicates the area that was inundated in the reference dataset but not in the test dataset.

Results for the comparison of between the five test datasets and the dataset created from the aerial photos is shown in figure 4.5. Regarding the overestimated area, the models with ridge lines perform better than the model without lines. Furthermore, the model with the ridge lines that were composed by the water authority obtained the lowest score on the overestimated area, however it received the highest value for the underestimated area. Thus, the model did more often correctly assign an area as inundated, but it also missed out on a significant part by classifying it as non-inundated. The difference between the outcome of the models with automatically generated ridge lines is for both overestimated as underestimated area relatively low. The model with ridge lines from the third algorithm outperformed the other two algorithms, in terms of overestimated area. However, the second model obtained a lower value for the underestimated area.

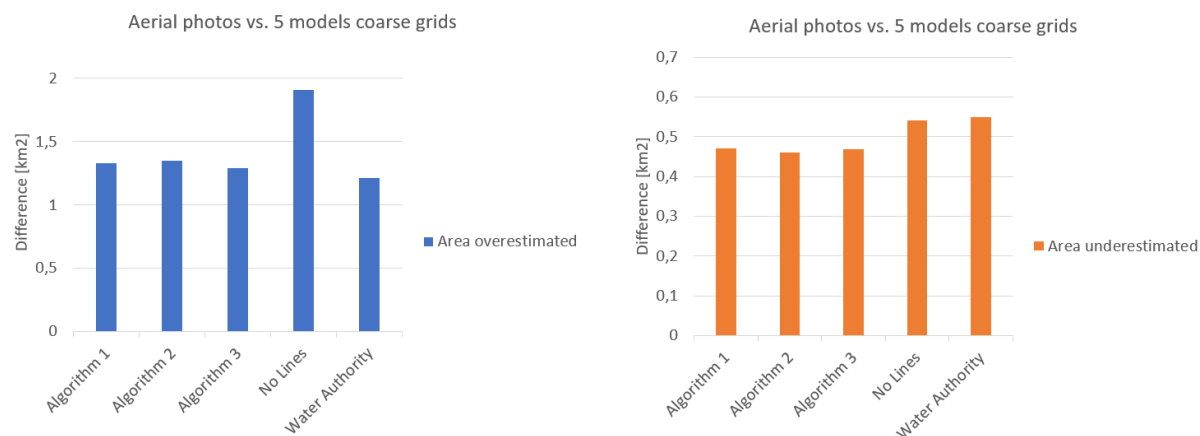


Figure 4.5: Inundated area according to data based on aerial photos versus output reference models (15 January 10:00).

Results for the comparison between the fine grid model and the five test datasets is shown in figure 4.5. The results in the figure reveal approximately similar differences between the five models in terms of overestimated area. Regarding the underestimated area, the third algorithm reveals the lowest score compared to the outcome of the other four models.

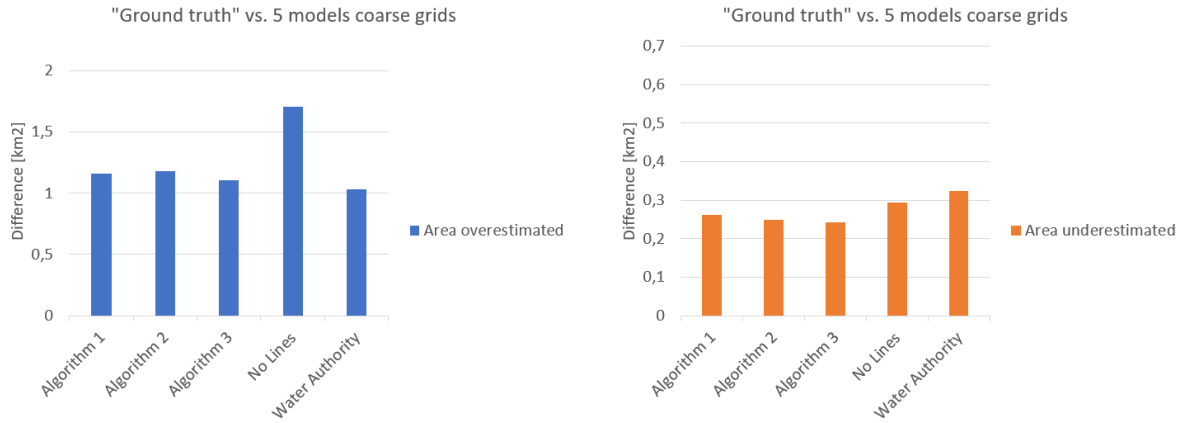


Figure 4.6: Inundated area according to fine grid model versus output reference models (15 January 10:00).

To verify whether similar results are obtained at another time in the simulation, the total inundated area was also compared for the fourth day of the simulation, thus at the 10th of January 2011 at 00:00. At this time the first peak discharge was observed in the upstream discharge of the Roer. The total inundated area at this time according to the fine grid model comprises 4,94km². Results of the comparison are shown in figure 4.7. In general the amount of overestimated area does not show a major increase or decrease for all models compared to the outcome at day 9 of the simulation. Regarding the underestimated area, the difference has become smaller. Furthermore, the figure reveals that the model with ridge lines from the third algorithm does not only outcompete the other models at day 9 of the simulation but also at day 4. Therefore, it appears that the total inundated area resulting from model based on the third algorithm does better resemble the output of the fine grid model for two time steps of the simulation.

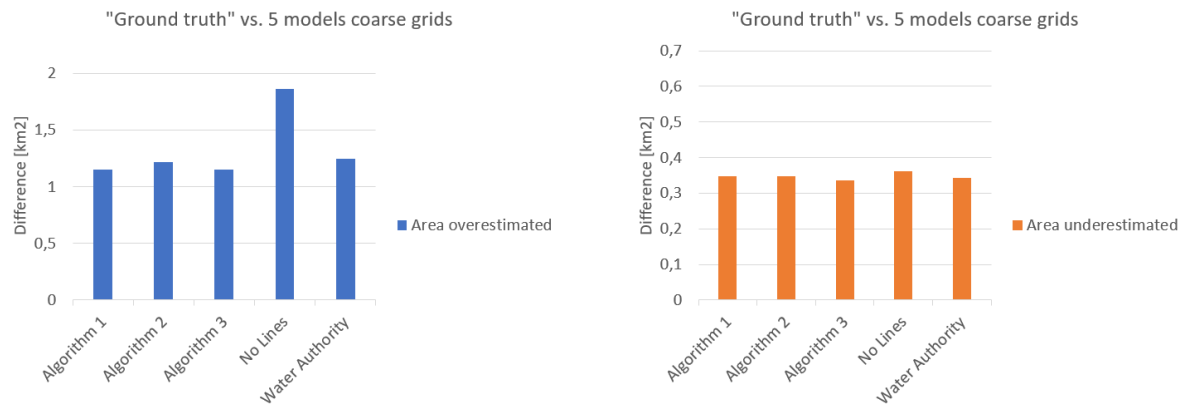


Figure 4.7: Inundated area according to fine grid model versus output reference models (10 January 00:00).

4.4. Results water depth

The water depth has been evaluated by calculating the root mean square error (RMSE) between the fine grid model and the five test data sets. The RMSE has been established by:

$$RMSE = \sqrt{\frac{\sum_{i=1}^N (x_i - \hat{x}_i)^2}{N}} \quad (4.5)$$

where N represents the total number of data non-missing data points, x refers to the values of the "ground truth" data set, and \hat{x} refers to the predicted values of the test data sets. In this study, only the non-zero data points were considered for the calculation of the RMSE. Otherwise the RMSE might return very low values as a significant part of the data files contains a water depth of zero.

Results of the RMSE are visualized in 4.8. The RMSE of the difference in water depth between the fine grid model and the five models reveals a similar pattern as was observed in the upstream discharge of the Roer. Hence suggesting that a larger upstream discharge does increase the error of the predictions of the five models. The graph reveals that the RMSE of the models varies between 0 and 0.26 m. At day 4 of the simulation during peak discharge, the water depth of the fine grid model was on average: 0.33 m, with a maximum value of 2.5 m. Furthermore, the figure shows that the the model without ridge lines obtained the highest RMSE for the entire simulation, which is not surprising as the model does not account for sub-grid variations in topography. Hence water is not retained due to fixed weirs in the model.

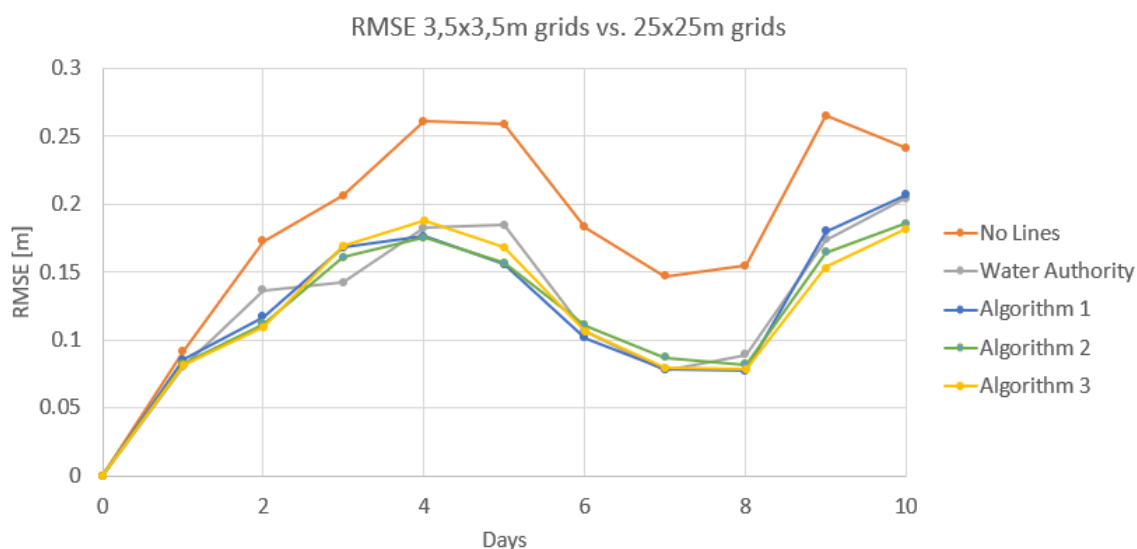


Figure 4.8: RMSE comparison water depth; fine grid model versus reference models.

In general, no extreme differences for the RMSE were obtained between the four models with ridge lines. Hence suggesting that the lines from the three algorithms perform as well as the lines that were composed by the water authority. Except for day two and five, the model based on the composed lines by the water authority obtained a higher RMSE, while on day three the RMSE was lower. Focusing only on the three algorithms, during the first peak discharge the model based on lines from algorithm 1 scored slightly lower than the other two algorithms. Thus, the outcome of this model did obtain water levels that were closer related to the water level values of the fine grid model. During the second peak, the lowest values for the RMSE were obtained for the comparison between the fine grid model and the third algorithm.

4.5. Analysis of the results

The difference in total inundated and water depth has been analyzed by zooming in on some locations. The algorithms have been compared by analyzing the difference in water depth at day four of the simulation. In figure 4.9, the difference between the results based on the algorithms and the outcome of the fine grid model is visualized. The blue and the red areas indicate respectively the regions that have a higher and lower water depth compared to the fine grid model. The two highlighted regions in the plots will be analyzed in more detail. As shown in the figure, the water depth in box 1 was lower for the outcome based on algorithms 1 and 2 than for the fine grid model. While the model based on ridge lines from the third algorithm did not show a significant difference at this location. At box 2, the model

based on the third algorithm, overestimates the water depth, while the model with ridge lines from the first and second obtains a water depth that is closer related to the outcome of the fine grid model.

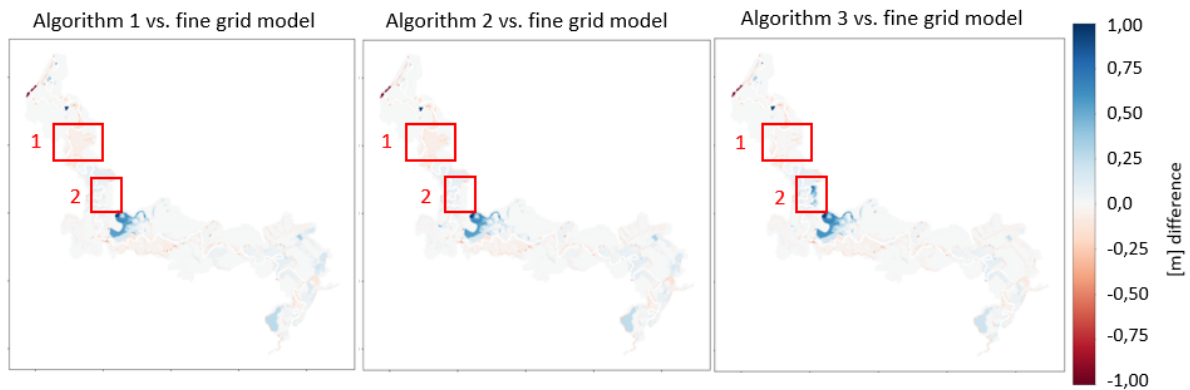


Figure 4.9: Difference in water depth fine grid model versus algorithm 1, 2 and 3. Blue areas indicated zones where a higher water depth was observed in the outcome of the models based on the algorithms, while red zones highlight the areas where a lower water depth was obtained (10 January 00:00). More detailed figures can be found in the appendix E.

The water depth in the region situated in the first red box is visualized in figure 4.10. For this analysis only the second and third algorithm are considered because the first algorithm did not reveal a major difference compared to the second algorithm. At day 3 of the simulation of the model based on algorithm 3, water started to enter the grid cells in the outlined area in the figure. In the simulation of algorithm 1 and 2 these grid cells were not inundated at that specific time interval.

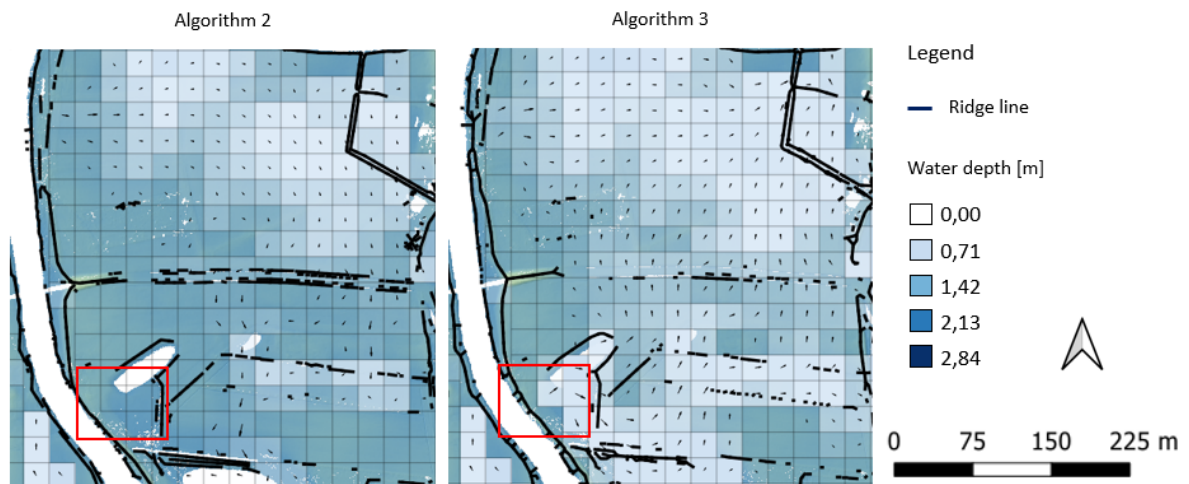


Figure 4.10: Water depth algorithm2 and 3 (9 January 2011, 3:00).

The observed difference in water depth seemed to be related to a ridge structure that was not correctly detected by the third algorithm. Focusing on the outlined region in figure 4.10, it appeared that the third algorithm placed the ridge line 2.5 m off the crest level of the ridge structure. A cross section has been made of the ridge structure indicated by the red dotted line in figure 4.10. Based on the cross section, which is visualized in figure 3.5, a difference in elevation was measured of about 0.5 m between the location of the ridge line of algorithm 2 and 3. Hence, the following conclusion has been drawn. Analysis of the lines showed that the first two algorithms did more accurately represent the topology at this location. Therefore, the observed difference in water depth in box 2 of figure 4.9 is probably related

to the manner in which the elevation of the ridge lines is determined in D-HYDRO. A grid size of 25 x 25 m has been used for the hydrodynamic models based on the output of the algorithms. Since the elevation of the ridge lines for the model is only determined by analyzing the height at the intersection of the flow links with the grids, there is a major chance that gaps or lower parts of a ridge structure are missed. Thus, the observed difference in water depth compared to the fine grid model can be related to the manner in which the height of the ridge structures is determined rather than a deficiency of the first two algorithms.

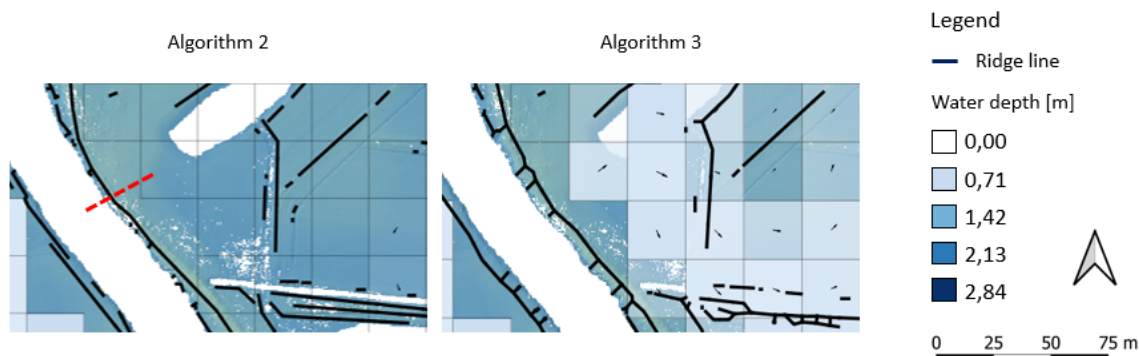


Figure 4.11: Water depth and ridge lines algorithm 2 versus algorithm 3 (9 January 2011, 3:00). Red dotted line indicates the location of the cross section in figure 4.12.

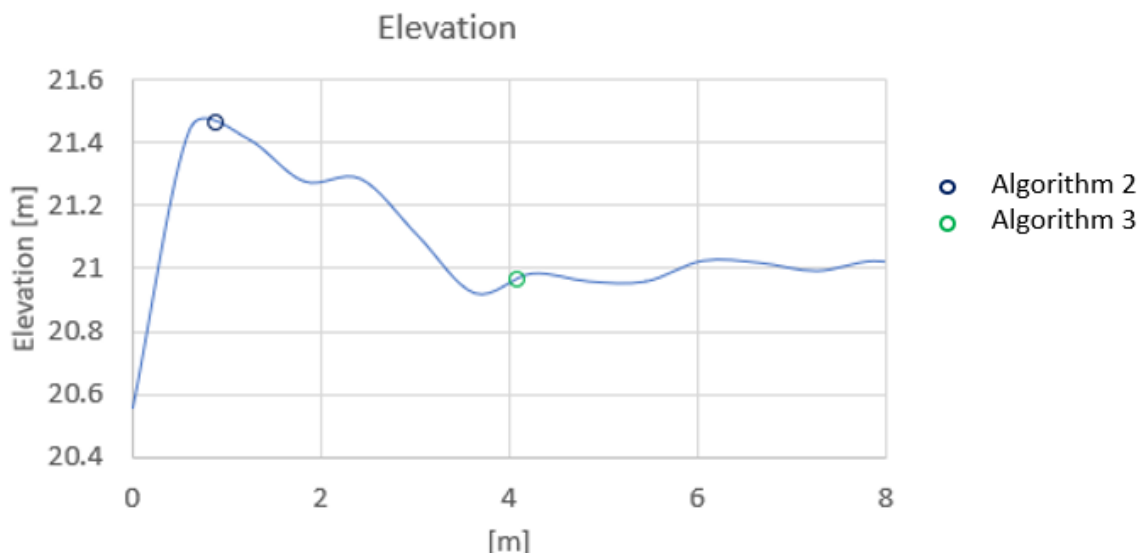


Figure 4.12: Cross section. Blue and green dot represent respectively the location of ridge line from algorithm 2 and algorithm 3.

The ridge lines located in the second red box in figure 4.9, were also analyzed to establish the cause of the difference in water depth in this region. Zooming in on the region in box 1, it appeared that the automatically detected ridge lines of the third algorithm were not placed exactly at the top of the ridge structure (figure 4.13). Hence water could enter the grid cell that intersects with the red box of 4.13 at an earlier stage of the simulation. The ridge lines of algorithm 2 are exactly placed at the crest level of the ridge structure, hence blocking the flow into the grid cells. The outcome of algorithm 1 resembles the outcome of the second algorithm, however the line is not placed exactly on top of the ridge structure, but

slightly off, nevertheless the water does not enter the grid cell in the red box. Apparently the elongated kernels of the second algorithm result in more continuous ridge lines that are placed exactly on top of the ridge structure.

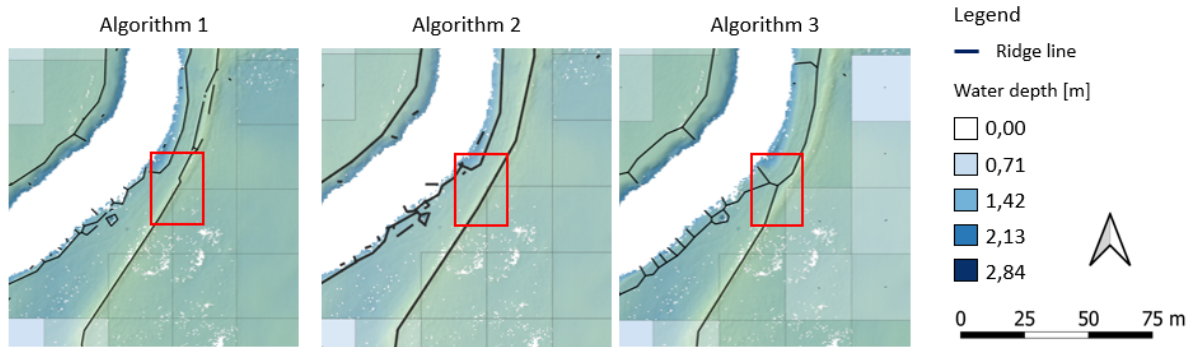


Figure 4.13: Water depth algorithm 1, 2 and 3 (8th of January 2011 at 8:00).

A final analysis is performed on the outcome of the models based on algorithm 1 and 2. Both algorithms revealed similar scores for the RMSE during the entire simulation (4.8). Except during the last days a notable difference has been observed. To establish the cause of the difference in RMSE for this day, the difference in water depth has been visualized in figure 4.14. Since the hydrodynamic model is relatively complex as it covers a large area and it contains numerous input variables, it is difficult to explain all observed differences between the outcome of the models. Which is for instance the case for the regions in box 1 and 2. For both regions no apparent cause could be determined for the difference in water depth between algorithm 1 and 2. The situation at box 1 is visualized in detail in 4.14 and will be explained below.

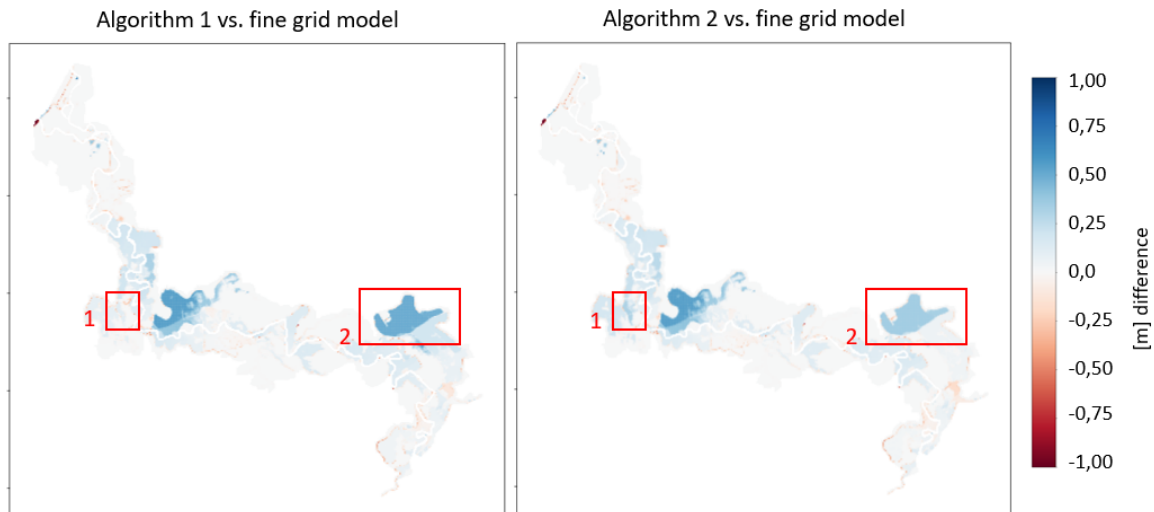


Figure 4.14: Difference in water depth at the 16th of January 2011 at 23:00 between fine grid model and results based on algorithm 1 and 2 (blue areas indicated zones where a higher water depth was observed in the outcome of the models based on the algorithms, while red zones highlight the areas where a lower water depth was obtained). More detailed figures can be found in the appendix E.

Another discrepancy in water depth was found in box 1. At this location already at day 8 of the simulation, water was passing the ridge line of algorithm 2, hence flowing from the right side to the left of the ridge structure. In the model based on algorithm 1 water did not pass the ridge structure the entire simulation, which was also observed in the fine grid model. Zooming in on the ridge lines (figure 4.16),

it appeared that the lines were placed only 1 meter apart. Analysis was performed to establish the elevation that has been used for the ridge structures. The red dots in figure 4.16, indicate the points that were used to determine the elevation of the ridge line. In total about 10 cm of difference between the ridge line of algorithm 1 and 2 was found. It seems highly unlikely that such a small difference in elevation could create such a significant difference in water depth in box 1 of figure 4.14. Moreover, the water depth at the moment water starts passing the ridge line in algorithm 2, is not significantly higher compared to the outcome of the model based on algorithm 1. Hence, it is impossible to state whether the ridge lines of algorithm 1 or 2 perform better in a D-HYDRO model.

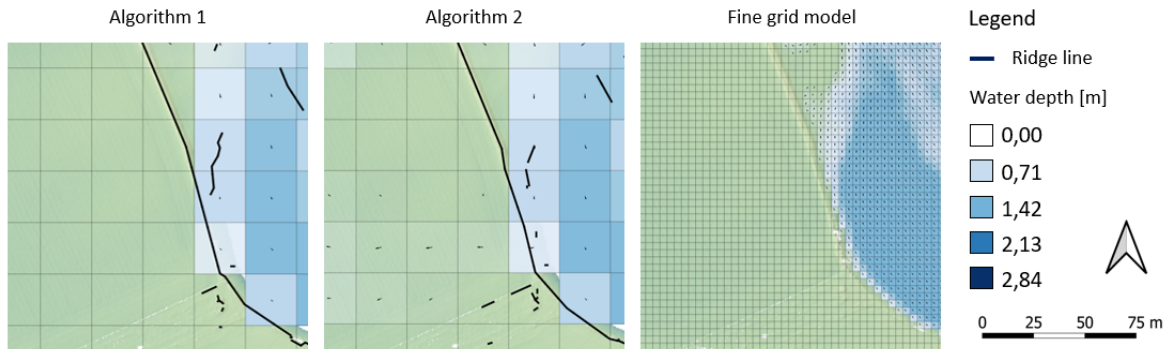


Figure 4.15: Water depth algorithm 1 and 2 and fine grid model (14th of January 2011 at 13:00).



Figure 4.16: Determination of elevation. Points that are taken into account for the weighted average are indicated by red dots.

5

Discussion

In this paper, the potential of Gaussian kernels has been tested on the automatic detection of ridge structures. A verification was performed by comparing the outcome of three different algorithms to manually drawn ridge lines. Furthermore, the automatically detected ridge lines were incorporated into a 1D-2D model, whereby the outcome has been compared to aerial pictures and a model representing an approximation of the ground truth. Various aspects of this study could be improved, and numerous remarks could be made regarding the verification method. The most prominent will be discussed in this chapter. Firstly, remarks on the algorithms will be provided, followed by a discussion of the verification method. Finally, suggestions for further research will be outlined.

5.1. Improvements algorithms

The three developed algorithms could all be improved in different ways. A major drawback of the first and the second algorithm was the time consumption. The process of skeletonization, which has been applied in the third algorithm, requires a fraction of the time that is needed in order to carry out NMS. Adjustments to the code of the first two algorithms might reduce the computing time. However, the lines that were detected by these algorithms were often placed exactly at the crest level of the ridge structure. On the contrary, the third algorithm created lines that were often slightly off the top of the ridge structure, which was confirmed by the results from the comparison between the outcome of the algorithms and the manually drawn lines. Especially on regions with a large amount of pixels above the threshold, inaccurate ridge lines were generated by the third algorithm. Further research could be performed on the procedure of skeletonization. The results of the third algorithm would certainly increase if skeletonization would take into account the height of the response values. However, such an adjustment comes probably at the expense of the calculation time of the algorithm.

Gaps were often encountered on the lines that resulted from the first and second algorithm. The formation of gaps can be, amongst other causes, ascribed to the process of NMS. Taking in consideration the procedure of NMS, it is not surprising that gaps appeared in the ridge lines appear. For instance, it occurred that the crest level of a ridge structure was located on the right side of a ridge structure, while one meter down the ridge, the crest level was located on the other side. Hence it is inevitable that a gap is created in the middle of the wider ridges. Moreover, the usage of multi-scale ridge detection did also induce the formation of gaps in the ridge lines. Perpendicular to a ridge structure, small and large kernels can obtain equally high results when a convolution is applied. Due to NMS, only the maximum value is retained. However, for wider ridge structures and when a low lookup value is utilized for NMS, parallel ridge lines are created resulting in the formation of gaps. Gaps were also created on ridge structures that were part of a crossing or junction. In this study, the anisotropic kernels did not completely solve the problem of crossings and junctions. The response value to the convolution at the crossings and junctions was still lower than at locations. A solution might be to develop a different type of kernel that has the same form as a crossing or junction. The result of a convolution with such a kernel could be added to the filter process. As such, crossings and junctions might be correctly extracted by the algorithms.

To counteract the formation of discontinuities on the ridge lines, a procedure was added to the algorithms whereby broken ridge lines were linked. However, it appeared that the majority of the gaps in the lines did not have any impact on the performance in D-HYDRO. This can be ascribed to the manner in which the lines are incorporated into a D-HYDRO model. The ridge lines are implemented to the model by means of a weighted average. The weighted average is determined at the intersection of the flow links and the ridge line. Therefore, in a model with a grid resolution of 25 x 25 m, only a very small part of the ridge line is considered for the weighted average. Further research must be performed to establish a manner in which the elevation of the ridge lines is more accurately added to the hydrodynamic model. Analysis must also be executed on the usefulness of the procedure of linking line ends and single ridge pixels. Perhaps this procedure does not ameliorate the performance of the ridge lines in a D-HYDRO model.

The procedure of IDW, resulted in the creation of abrupt changes in the interpolated regions. The algorithms recognized these abrupt changes as ridges, hence lines were frequently created in and around these zones. A different type of interpolation would certainly increase the performance in these regions. In addition, the malfunctioning of the algorithms at the boundaries could be improved by adjusting the manner in which the convolution is carried out. Currently, the algorithms detect ridge lines at the boundaries of the DTM, since imaginary rows and columns filled with zeros are added at the border of the DTM. Hence an abrupt change is encountered at the borders.

Finally, additional research on the parameter settings might also improve the performance of the algorithms. Research could be performed on the required dimensions of the ridge structures that must be used for certain types of hydrodynamic models. Hence only the relevant ridge structures can be extracted by the algorithms. Furthermore, research on threshold selection could also improve the performance of the algorithms. Currently, the threshold is manually determined by analyzing the output of the filter process. An automatic selection of the threshold would accelerated the procedure of ridge structure detection.

5.2. Verification method

Verifying the outcome of the algorithms has been a major challenge in this study. The ridge lines that were composed by the water authority of Limburg made it possible to use data that was less biased than the lines that were specifically drawn for this study. However, an inspection of the ridge lines provided by the water authority suggested that quality of the lines was low. The most prominent ridge structure were covered by the lines, but ridge lines were often not exactly placed on the crest level of the ridge structure. The poor quality of the lines could not be traced back in the results of the verification in D-HYDRO. The total inundated area and the RMSE was approximately similar to the outcome of the models based on the algorithms. This might be linked to the manner in which the fine grid model has been developed. The fine grid model has been developed with a resolution of 3.5 x 3.5m. The topography is incorporated into the grids by taking the average of of the elevation for every grid cell. As such, the maximum elevation of a ridge structure is not taken into account. Hence a model where ridge lines were placed not exactly on top of a ridge structure, might generate similar water depths as the fine grid model. This might also explain the high scores in RMSE for the third algorithm, whereby lines are also often generated not exactly on top of a ridge structure.

The fine grid model was developed with the intention to function as a ground truth model. However, the model was created by re-sampling the DTM into a coarser resolution, hence it would never be capable to accurately represent the inundation that took place in January 2011. This became already apparent when the outcome of the fine grid model was compared to the data that was composed based on aerial photos. A significant difference was observed between the total inundated area. The data that was developed by utilizing aerial photos might better reflect reality, however it is unavoidable that also here inundated areas were missed or were overestimated.

The comparison between the outcome of the algorithms and the manually drawn ridge lines was performed by transforming the lines into a binary raster file. For this process, a resolution of 0.5 x 0.5 m has been used, which is similar to the resolution of the DTM that was used as input for the algorithms. However, due to such a fine resolution the chance is very small that the lines of the test and reference dataset are located at the same place. Therefore, relatively low values were obtained for the precision,

recall, F1-score and MCC. Higher values would probably be generated when a coarser resolution was utilized to transform the ridge lines into rasters.

A first remark on the D-HYDRO model was already made in the previous section about the manner in which the ridge lines are incorporated into the model. The quality of ridge lines in coarse grid models might be severely damaged when only the elevation at the intersection of the ridge lines and the flow links is considered. Another remark on the D-HYDRO model can be made about the parameters that were used for ridge structure, among which the slope on both sides and the width of structure. These parameters are used for the calculation of the flow over a ridge structure. For this study, the default values have been applied, hence inaccurate flows over the ridge structures are established. For future research these parameters should be correctly set, hence the flow over the ridge structure does better reflect reality. Another remark on the model must be made about the assumption that the ridge structures do not collapse during the inundation. In reality it is likely that a ridge structure would collapse as a consequence of the flood. Determining which ridge structures would collapse and which structures remain intact is a complex, if not, a nearly impossible task. Nevertheless research devoted to this topic could provide more insight on the optimum way to model flow around ridge structures. Lastly, the model that was utilized is rather complex as it covers a large area and involves numerous input variables. Therefore, not every difference in water depth and inundated area could be explained.

Lastly, it must be noted that the performance of the algorithms has only been verified for the surroundings of the Roer. Several runs were carried out on a floodplain of the river Waal, near the city Nijmegen. These tests revealed some promising results, inferring that the algorithms could also be applied to other regions. More research is required, for which regions with larger elevation differences are also included.

5.2.1. Future research

For future research, the algorithms could be adjusted in order to detect not only ridges but also the lowest areas on the DTM. The extreme lows could be implemented in the hydrodynamic model to determine patterns in overland flow. Further research could also be carried out to verify the performance of the algorithms in comparison with methods that are to some extent capable to determine ridge structures, such as Deterministic-8 or D-infinity. Another option for further research concerns the application of a neural network. The output of the algorithms combined with other data could be used to learn a neural network to detect ridge structures.

To summarize, various aspects of the study could be ameliorated. Several aspect on the algorithms could be improved and the verification method could have been more extensive, for example by analyzing the performance of the algorithms at other locations. Finally, despite the promising results, it should be underlined that a final check on the outcome of the algorithms remains necessary.

6

Conclusion

For this research, three algorithms have been developed to automatically extract ridge lines based on DEM data. The algorithms were developed with the intention to produce ridge lines that can be utilized for hydrodynamic flood modelling. Therefore, the outcome of the algorithms had to fulfill four requirements. Firstly, the algorithm should be insensitive to noise and, secondly, should be able to detect ridge structures of various sizes. Thirdly, at the extracted ridge structures a line had to be placed exactly on, or at least in the vicinity of, the crest level of the ridge structure. And fourthly, the algorithms should produce continuous ridge lines, without unnecessary gaps. Based on these criteria, the second order Gaussian kernel was selected for the development of the algorithms as it revealed promising results in the detection of line features on images.

A deficiency of the SOG-kernel is that it performs poorly at ridge structures that are part of a crossing or junction. Therefore, three algorithms were developed with a specific focus on solving the issue of crossings and junctions. The first, default, algorithm uses isotropic kernels during the filter process followed by NMS. The second algorithm utilizes anisotropic and isotropic kernels followed by NMS and the third algorithm uses only isotropic kernels but instead of NMS, skeletonization is applied. The three algorithms return a binary raster, which could be implemented, after some post-processing steps, into a D-HYDRO model.

The performance of the algorithms has been verified in different manners. Firstly the outcome of the algorithms has been compared to manually drawn lines by computing the precision, recall and F_1 -score and by analyzing position accuracy based on the procedure as proposed by Goodchild and Hunter. Secondly the quality of the ridge lines is evaluated in a coupled 1D/2D-model developed for The Roer. Therefore, the ridge lines were implemented into the hydrodynamic model. The resulting water depth and total inundated area has been compared to the outcome of a fine grid model with a resolution of 3.5 x 3.5m. Furthermore, the results of the models based on the outcome of the algorithms, was compared to data that was composed based on aerial photos.

Several conclusions can be drawn based on the results of the verification. The comparison of between the manually drawn ridge lines and the automatically detected lines revealed that the first and the second algorithms obtained significantly higher values for precision, recall and the F_1 -score. Hence it can be stated that these algorithms are more capable in placing the line exactly on the crest level of the ridge structure. This was confirmed when the outcome of the D-HYDRO model was analyzed by zooming in on some locations. No significant discrepancy was discovered between the outcome of algorithm 1 and 2 for to comparison to the manually drawn ridge lines. However, an inspection of the resulting lines revealed that the second algorithm did produce more continuous lines than the first algorithm. Furthermore, it turned out that the quality of the composed ridge lines by the water authority too poor for analyzing the performance of the algorithms.

Regarding the second part of the evaluation, it can be concluded that a model based on the lines from the algorithms perform as well as a model based on the ridge lines that were composed manually. In general, the models that included ridge lines performed significantly better than the model without lines, as a higher RMSE was obtained for the former. However, it should be noted that the verification was

carried out by utilizing a fine grid model as an approximation of the ground truth. The fine grid model did represent the overall pattern of the topography, nevertheless it could not take into account the exact height of the crest level of a ridge structure. Therefore, no firm conclusions can be made about the algorithm that, except for the fact that a model with ridge lines performs better than a model without ridge lines.

Regarding the four requirements that were set, the following conclusions can be drawn. The three algorithms could all detect ridge structures of various sizes. The algorithms did reveal some high responses to noise, however this could be suppressed by applying kernels with a larger sigma value or by utilizing a higher threshold. Solely the first and the second algorithms could extract ridge lines exactly on the crest level of the ridge structure. Finally, mainly the second algorithm was capable to generate continuous ridge lines.

Considering the research question:

How can second-order Gaussian kernels be utilized for the automatic extraction of ridge structures, and what are the implications for 2D-flood modelling?

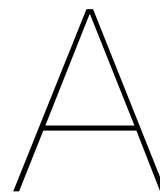
the following can be stated. An algorithm based on non maximum suppression and both isotropic and anisotropic kernels, produces the most accurate results. However, an algorithm for which skeletonization is applied can produce a quite neat output in a fraction of the calculation time that is required for an algorithm based on non maximum suppression. Regarding the implications for 2D-flood modelling, the output of a model based on automatically extracted ridge lines performs as well as a model based on manually drawn ridge lines. However, due to the verification method it is impossible to state which algorithm should be applied to automatically detect ridge lines in order to generate the best performance of the hydrodynamic model.

Bibliography

- [1] F. Kreienkamp, S. Y. Philip, J. S. Tradowsky, S. F. Kew, P. Lorenz, A. Belleflamme, T. Bettmann, S. Caluwaerts, S. C. Chan, A. Ciavarella, L. D. Cruz, H. D. Vries, N. Demuth, A. Ferrone, M. Fischer, H. J. Fowler, K. Goergen, D. Heinrich, Y. Henrichs, F. Kaspar, E. Nilson, F. E. L. Otto, F. Ragone, I. Sonia, R. K. Singh, P. Termonia, L. Thalheimer, J. V. D. Bergh, H. V. D. Vyver, B. V. Schaeybroeck, R. Vautard, D. Vonk, and N. Wanders, "Rapid attribution of heavy rainfall events leading to the severe flooding in western europe during july 2021 contributors," *World Weather Attribution*, no. July, 2021.
- [2] J. Poussin, P. Bubeck, J. Aerts, and P. Ward, "Potential of semi-structural and non-structural adaptation strategies to reduce future flood risk: case study for the meuse," *Natural Hazards and Earth System Sciences*, vol. 12, no. 11, pp. 3455–3471, 2012.
- [3] N. Ongdas, F. Akiyanova, Y. Karakulov, A. Muratbayeva, and N. Zinabdin, "Application of hec-ras (2d) for flood hazard maps generation for yesil (ishim) river in kazakhstan," *Water*, vol. 12, no. 10, p. 2672, 2020.
- [4] N. V. Dung, B. Merz, A. Bárdossy, T. D. Thang, and H. Apel, "Multi-objective automatic calibration of hydrodynamic models utilizing inundation maps and gauge data," *Hydrology and Earth System Sciences*, vol. 15, no. 4, pp. 1339–1354, 2011.
- [5] U. Pasquier, Y. He, S. Hooton, M. Goulden, and K. M. Hiscock, "An integrated 1d–2d hydraulic modelling approach to assess the sensitivity of a coastal region to compound flooding hazard under climate change," *Natural Hazards*, vol. 98, no. 3, pp. 915–937, 2019.
- [6] M. Campolo, P. Andreussi, and A. Soldati, "River flood forecasting with a neural network model," *Water resources research*, vol. 35, no. 4, pp. 1191–1197, 1999.
- [7] D. Yu, "Parallelization of a two-dimensional flood inundation model based on domain decomposition," *Environmental Modelling & Software*, vol. 25, no. 8, pp. 935–945, 2010.
- [8] D. Yu and S. N. Lane, "Urban fluvial flood modelling using a two-dimensional diffusion-wave treatment, part 1: mesh resolution effects," *Hydrological Processes: An International Journal*, vol. 20, no. 7, pp. 1541–1565, 2006.
- [9] Deltares, Delft, *D-Flow Flexible Mesh Technical Reference Manual*, 1.1.0 ed., February 2021.
- [10] G. Henckens and W. Engel, *Benchmark inundatiemodellen - modelfunctionaliteiten en testbank berekeningen*. STOWA, 2017.
- [11] J. F. O'Callaghan and D. M. Mark, "The extraction of drainage networks from digital elevation data," *Computer vision, graphics, and image processing*, vol. 28, no. 3, pp. 323–344, 1984.
- [12] E. Pardo-Igúzquiza, J. J. D. Valsero, and P. A. Dowd, "Automatic detection and delineation of karst terrain depressions and its application in geomorphological mapping and morphometric analysis," *Acta Carsologica*, vol. 42, no. 1, 2013.
- [13] Y.-C. Chang, G.-S. Song, and S.-K. Hsu, "Automatic extraction of ridge and valley axes using the profile recognition and polygon-breaking algorithm," *Computers & Geosciences*, vol. 24, no. 1, pp. 83–93, 1998.
- [14] G. Fatih and T. Gökgöz, "A new algorithm for extraction of continuous channel networks without problematic parallels from hydrologically corrected dems," *Boletim de Ciências Geodésicas*, vol. 16, no. 1, pp. 20–38, 2010.

- [15] H. Zhang, Z. Ma, Y. Liu, X. He, and Y. Ma, "A new skeleton feature extraction method for terrain model using profile recognition and morphological simplification," *Mathematical Problems in Engineering*, vol. 2013, 2013.
- [16] G. Wang, C. Lopez-Molina, and B. De Baets, "Multiscale edge detection using first-order derivative of anisotropic gaussian kernels," *Journal of Mathematical Imaging and Vision*, vol. 61, no. 8, pp. 1096–1111, 2019.
- [17] A. Rosenfeld and M. Thurston, "Edge and curve detection for visual scene analysis," *IEEE Transactions on computers*, vol. 100, no. 5, pp. 562–569, 1971.
- [18] C. Lopez-Molina, G. V.-D. De Ulzurrun, J. M. Baetens, J. Van den Bulcke, and B. De Baets, "Unsupervised ridge detection using second order anisotropic gaussian kernels," *Signal Processing*, vol. 116, pp. 55–67, 2015.
- [19] J. Jasiewicz and T. F. Stepinski, "Geomorphons—a pattern recognition approach to classification and mapping of landforms," *Geomorphology*, vol. 182, pp. 147–156, 2013.
- [20] P. K. Bhola, B. B. Nair, J. Leandro, S. N. Rao, and M. Disse, "Flood inundation forecasts using validation data generated with the assistance of computer vision," *Journal of Hydroinformatics*, vol. 21, no. 2, pp. 240–256, 2019.
- [21] W. Dai, J. Na, N. Huang, G. Hu, X. Yang, G. Tang, L. Xiong, and F. Li, "Integrated edge detection and terrain analysis for agricultural terrace delineation from remote sensing images," *International Journal of Geographical Information Science*, vol. 34, no. 3, pp. 484–503, 2020.
- [22] G. Shrivakshan and C. Chandrasekar, "A comparison of various edge detection techniques used in image processing," *International Journal of Computer Science Issues (IJCSI)*, vol. 9, no. 5, p. 269, 2012.
- [23] A. M. Lopez, F. Lumbreras, J. Serrat, and J. J. Villanueva, "Evaluation of methods for ridge and valley detection," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 21, no. 4, pp. 327–335, 1999.
- [24] L. Roberts, "Machine perception of three dimensional solids, optical and electro-optical information processing. j. tippet et al. eds, cambridge, mass. m," 1965.
- [25] I. Sobel, "On calibrating computer controlled cameras for perceiving 3-d scenes," *Artificial intelligence*, vol. 5, no. 2, pp. 185–198, 1974.
- [26] J. M. Prewitt, "Object enhancement and extraction," *Picture processing and Psychopictorics*, vol. 10, no. 1, pp. 15–19, 1970.
- [27] T. Lindeberg, "Feature detection with automatic scale selection," *International journal of computer vision*, vol. 30, no. 2, pp. 79–116, 1998.
- [28] P. Majer, "On the influence of scale selection on feature detection for the case of linelike structures," *International Journal of Computer Vision*, vol. 60, no. 3, pp. 191–202, 2004.
- [29] Z. Ismail, M. Abdul Khanan, F. Omar, M. Abdul Rahman, and M. Mohd Salleh, "Evaluating error of lidar derived dem interpolation for vegetation area.," *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences*, vol. 42, 2016.
- [30] S. Crema, M. Llana, A. Calsamiglia, J. Estrany, L. Marchi, D. Vericat, and M. Cavalli, "Can inpainting improve digital terrain analysis? comparing techniques for void filling, surface reconstruction and geomorphometric analyses," *Earth Surface Processes and Landforms*, vol. 45, no. 3, pp. 736–755, 2020.
- [31] K. Gavriil, G. Muntingh, and O. J. Barrowclough, "Void filling of digital elevation models with deep generative models," *IEEE Geoscience and Remote Sensing Letters*, vol. 16, no. 10, pp. 1645–1649, 2019.

- [32] W. Maleika, "Inverse distance weighting method optimization in the process of digital terrain model creation based on data collected from a multibeam echosounder," *Applied Geomatics*, vol. 12, no. 4, pp. 397–407, 2020.
- [33] T. Lindeberg, "Edge detection and ridge detection with automatic scale selection," *International journal of computer vision*, vol. 30, no. 2, pp. 117–156, 1998.
- [34] T. M. Koller, G. Gerig, G. Szekely, and D. Dettwiler, "Multiscale detection of curvilinear structures in 2-d and 3-d image data," in *Proceedings of IEEE International Conference on Computer Vision*, pp. 864–869, IEEE, 1995.
- [35] J. Canny, "A computational approach to edge detection," *IEEE Transactions on pattern analysis and machine intelligence*, no. 6, pp. 679–698, 1986.
- [36] C. Sun and P. Vallotton, "Fast linear feature detection using multiple directional non-maximum suppression," *Journal of Microscopy*, vol. 234, no. 2, pp. 147–157, 2009.
- [37] P.-L. Shui and W.-C. Zhang, "Noise-robust edge detector combining isotropic and anisotropic gaussian kernels," *Pattern Recognition*, vol. 45, no. 2, pp. 806–820, 2012.
- [38] J.-M. Geusebroek, A. W. Smeulders, and J. Van De Weijer, "Fast anisotropic gauss filtering," *IEEE transactions on image processing*, vol. 12, no. 8, pp. 938–943, 2003.
- [39] D. Wang, J. Yin, C. Tang, X. Cheng, and B. Ge, "Color edge detection using the normalization anisotropic gaussian kernel and multichannel fusion," *IEEE Access*, vol. 8, pp. 228277–228288, 2020.
- [40] T. Y. Zhang and C. Y. Suen, "A fast parallel algorithm for thinning digital patterns," *Communications of the ACM*, vol. 27, no. 3, pp. 236–239, 1984.
- [41] H. Winteraeken, "De roer: over meanders en overstromingen," *Natuurhistorisch Maandblad*, vol. 103, no. 8, pp. 201–204, 2014.
- [42] "Ahn3 downloads."
- [43] Ahn, "Kwaliteitsbeschrijving," Mar 2020.
- [44] D. Chicco and G. Jurman, "The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation," *BMC genomics*, vol. 21, no. 1, pp. 1–13, 2020.
- [45] M. F. Goodchild and G. J. Hunter, "A simple positional accuracy measure for linear features," *International journal of geographical information science*, vol. 11, no. 3, pp. 299–306, 1997.



Python code algorithm 1

```
1 # -*- coding: utf-8 -*-
2 """
3
4 Created on Wed Apr 21 11:38:09 2021
5
6 @author: noppen
7 """
8
9 import numpy as np
10 import rioarray as rio
11 import os
12 import time
13 from skimage.morphology import skeletonize
14 from scipy.ndimage import rotate
15 from scipy.ndimage import convolve
16 from scipy.ndimage.morphology import binary_hit_or_miss
17 from rasterio.fill import fillnodata
18 from PIL import Image, ImageDraw
19 # from matplotlib import pyplot as plt
20
21 start = time.time()
22
23 #%% Parameters
24
25 # Parameters kernels
26 # Number of orientations of the kernel (theta)
27 N_theta = 6
28 variable_theta = np.arange(0, np.pi, np.pi/N_theta)
29 # Range of the size of the kernels (sigma)
30 variable_sigma = np.arange(0.2, 1.1, 0.2)
31
32 # Threshold after the convolution
33 threshold = 15 # Threshold value
34 # Lookup (l_nms) for non-maximum suppression
35 # Here the l_nms indicates the number of pixels that are considered when drawing a line
   perpendicular to the ridge structure
36 l_nms = 4
37
38 # Lookup (l_c) for connection points and line ends
39 l_c = 3
40
41 #%% PART I: PRE-PROCESSING DATA
42 # Import DTM
43 data_path = os.chdir('../Data')
44 z = rio.open_rasterio('Clip_3.tif')
45
46 # Select data AHN
47 tif = z[0, :, :]
48 sample = np.where(tif > 1E5, np.nan, tif.values)
49
50 # IDW > Fill nan values
```

```

51 s_mask = np.where(np.isnan(sample), 0, 1)
52 sample = fillnodata(sample, mask = s_mask)
53
54 #%% PART II: KERNEL CONVOLUTION
55 def kernel_conv (DEM, s = 1, t = 0, r = 1, lowest = -5, highest = 5, step = 0.1, gamma =
    0.75):
56     x = np.arange(highest, lowest, -step)
57     y = np.arange(lowest, highest, step)
58     x, y = np.meshgrid(x, y)
59
60     # Create kernel based on sigma, theta & rho
61     M_theta = np.array([ [np.cos(t), np.sin(t)], [-np.sin(t), np.cos(t)] ])
62     rho_sqrt = np.array([ [r**2, 0], [0, r**-2] ])
63     z_second = np.zeros((len(x), len(y)))
64
65     for i in range(len(x)):
66         for j in range(len(y)):
67             xy = np.array([x[i, j], y[i, j]])
68             x_y = np.array([ [x[i, j]], [y[i, j]]])
69             phi = xy@ M_theta.transpose()@ rho_sqrt@ M_theta@ x_y
70             G_hat = (1/ (2*np.pi*s**2)) * np.exp(-phi/(2*s**2))
71             SAG = (((x[i, j] * np.cos(t)) + (y[i, j] * np.sin(t)))**2)/(r**4 * s**4) - (r**2
                / s**2) * G_hat
72             z_second[i, j] = SAG
73
74     # Convolution
75     DEM_conv = -(s**2)**gamma * convolve(DEM, z_second)
76     return (DEM_conv)
77
78 # Sigma van 1: resulteert bij step=0.1 in een positieve piek van 20 stappen, omgerekend naar
    AHN(0.5m) geeft detectie ridge van: 10 meter
79 # Keuze voor detecteren ridge van 3 tot 15 meter breed geeft sigma: 0.3 tot 1.5
80
81 Direction = np.zeros(np.shape(sample))
82 Sigma_final = np.zeros(np.shape(sample)) + variable_sigma[0]
83
84 for j in range (len (variable_sigma)):
85     print(j)
86     for k in range (len(variable_theta)):
87         DTM_conv = kernel_conv(DEM = sample,
88                               s = variable_sigma[j],
89                               t = variable_theta[k])
90
91         if k==0 and j==0:
92             Result_conv = DTM_conv
93
94             # Wordt
95             uiteindelijke resultaat in opgeslagen van convolutie
96
97         else:
98             Sigma_final = np.where(DTM_conv>Result_conv, variable_sigma[j], Sigma_final)
99             Direction = np.where(DTM_conv>Result_conv, variable_theta[k], Direction)
100             Result_conv = np.where(DTM_conv>Result_conv, DTM_conv, Result_conv)
101
102 #%% PART III: THRESHOLDING
103 Result_binary = np.where(Result_conv>threshold, 1, 0)
104
105 #%% PART IV: NON-MAXIMA SUPPRESSION
106 x, y = np.shape(sample)
107 DTM_extra = np.empty ((2*l_nms+sample.shape[0], 2*l_nms+sample.shape[1]))
108 DTM_extra[:] = np.nan
109 DTM_extra[l_nms:-l_nms, l_nms:-l_nms] = Result_conv
110
111 NMS = np.empty (np.shape(DTM_extra))
112 NMS[:] = np.nan
113 NMS[l_nms:-l_nms, l_nms:-l_nms] = Result_conv
114
115 Line = np.zeros(np.shape(DTM_extra))
116
117 for i in range (x):
118     print(i)
119     for j in range (y):
120         if Result_binary[i, j]==1:
121             theta = Direction[i, j]

```



```

117     a_1 = np.round(l_nms*np.cos(theta)) + l_nms
118     a_2 = np.round(l_nms*-np.cos(theta)) + l_nms
119     b_1 = np.round(l_nms*-np.sin(theta)) + l_nms
120     b_2 = np.round(l_nms*np.sin(theta)) + l_nms
121
122     im = np.asarray(Line, dtype=np.uint8)
123     im = Image.fromarray(im, mode='L')
124     draw = ImageDraw.Draw(im)
125     draw.line(((a_1+j),(b_1+i),(a_2+j),(b_2+i)), fill = 1)
126     c = np.array(im)
127     arr = DTM_extra[np.where(c==1)]
128
129     if DTM_extra[i+l_nms, j+l_nms]<(np.nanmax(arr)):
130         NMS[i+l_nms, j+l_nms]=np.nan
131
132
133 Result_NMS = NMS[l_nms:-l_nms, l_nms:-l_nms]
134 Result_NMS = np.where(Result_binary==0, np.nan, Result_NMS)
135 Result_binary = np.where(Result_NMS>0, 1, 0)
136
137 # Outside to np.nan
138 Result_binary[0,:]=0
139 Result_binary[-1,:]=0
140 Result_binary[:,0]=0
141 Result_binary[:, -1]=0
142
143
144 #%% PART V: CONNECTION POINTS TO LINES
145 a = 2
146 b = 3
147 ridge_extra = np.zeros ((2*l_c+x),(2*l_c+y))
148 endpoints = np.zeros ((2*l_c+x),(2*l_c+y))
149 ridge_extra[l_c:-l_c, l_c:-l_c] = Result_binary
150
151 # Function to connect lines from point i,j to point x, y
152 # Input is neighbors (NB), coordinates point 1 (i,j), Binary result (RE) and distance wherein
    lines/points are connected
153 def connection(NB, i, j, RE, n):
154     a = np.zeros(np.shape(RE))
155     [x_coor, y_coor] = np.where(NB>0)
156     x_coor = x_coor + i-n
157     y_coor = y_coor + j-n
158
159     for k in range(len(x_coor)):
160         a = np.asarray(a, dtype=np.uint8)
161         im = Image.fromarray(a, mode='L')
162         draw = ImageDraw.Draw(im)
163
164         draw.line((j, i, y_coor[k], x_coor[k]), fill=1)
165         np.array(im)
166         RE = im + RE
167         RE = np.where(RE > 0, 1, 0) # Binary
168
169         a = np.zeros(np.shape(RE))
170     return(RE)
171
172 # Create 4 options of 3x3 kernel that can be used to find the end lines
173 arr1 = np.array([[1,0,0], [1, 1, 0], [0,0,0]])
174 arr2 = np.array([[0,0,0], [1, 1, 0], [1,0,0]])
175 arr3 = np.array([[0,1,0], [0, 1, 0], [0,0,0]])
176 arr4 = np.array([[1,0,0], [0, 1, 0], [0,0,0]])
177
178 # Find all line ends by hit or miss with created 3x3 kernel
179 for i in range(4):
180     endpoints = binary_hit_or_miss(ridge_extra, arr1) + endpoints
181     arr1 = rotate(arr1, angle=90)
182 for i in range(4):
183     endpoints = binary_hit_or_miss(ridge_extra, arr2) + endpoints
184     arr2 = rotate(arr2, angle=90)
185 for i in range(4):
186     endpoints = binary_hit_or_miss(ridge_extra, arr3) + endpoints

```

```

187     arr3 = rotate(arr3 , angle=90)
188 for i in range(4):
189     endpoints = binary_hit_or_miss(ridge_extra , arr4) + endpoints
190     arr4 = rotate(arr4 , angle=90)
191
192 # Connect all line ends and separate point that are within a distance of length l
193 # values a and b are used to create local neighbors
194
195 for i in range(l_c,x+l_c):
196     for j in range(l_c,y+l_c):
197         if (ridge_extra[i,j]==1):
198
199             # Neighboring points
200             neighbors = ridge_extra[i-1:i+2,j-1:j+2]
201             if np.sum(neighbors)==1:
202                 a = 2
203                 b = 3
204
205             if np.sum(ridge_extra[(i-l_c):(i+(l_c+1)) ,(j-l_c):(j+(l_c+1))])==1:
206                 ridge_extra[i,j]=0
207
208             for k in range(l_c):
209                 if np.sum(ridge_extra[(i-a):(i+b) ,(j-a):(j+b)])>1:
210                     neighbors = ridge_extra[(i-a):(i+b) ,(j-a):(j+b)]
211                     ridge_extra = connection(neighbors , i , j , ridge_extra , n=a)
212                     a = a+1
213                     b = b+1
214
215             # Connection line ends
216             if (endpoints[i,j]==1):
217                 a = 2
218                 b = 3
219             for k in range(l_c):
220                 if np.sum(endpoints[(i-a):(i+b) ,(j-a):(j+b)])==2:
221                     neighbors = endpoints[(i-a):(i+b) ,(j-a):(j+b)]
222                     endpoints = connection(neighbors , i , j , endpoints , n=a)
223                     a = a+1
224                     b = b+1
225
226 ridge_extra = ridge_extra + endpoints
227 Result_binary = np.where(ridge_extra[l_c:-l_c,l_c:-l_c] > 0 ,1 ,np.nan)
228                                     # Binary
229 Result = np.where(Result_binary==1, Result_conv , np.nan)
230                                     # Fill with values convolution
231
232 end = time.time()
233 print(end - start)
234
235 # Thinning lines for further processing in qgis
236 Result_binary = skeletonize(Result_binary)
237 Result_binary = np.where(Result_binary==1,1,np.nan)
238
239 # %% "" SAVE RESULTS ""
240 data_path = os.chdir('../Results')
241
242 tif = z
243 tif[0, :, :] = Result_binary
244 tif.rio.to_raster("A1_Results_binary.tif")
245
246 tif = z
247 tif[0, :, :] = Result
248 tif.rio.to_raster("A1_Results.tif")
249
250 tif = z
251 tif[0, :, :] = Sigma_final
252 tif.rio.to_raster("A1_sigma.tif")
253
254 tif = z
255 tif[0, :, :] = Result_conv

```

```
256 | tif.rio.to_raster("A1_results_convolution.tif")
```


B

Python code algorithm 2

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Wed Apr 21 11:38:09 2021
4
5 @author: noppen
6 """
7
8 import numpy as np
9 import rioarray as rio
10 import os
11 import time
12 from skimage.morphology import skeletonize
13 from scipy.ndimage.morphology import binary_hit_or_miss
14 from scipy.ndimage import convolve
15 from scipy.ndimage import rotate
16 from rasterio.fill import fillnodata
17 from PIL import Image, ImageDraw
18 from matplotlib import pyplot as plt
19
20 start = time.time()
21
22 #%% Parameters
23
24 # Parameters kernels
25 N_theta = 6
26 variable_theta = np.arange(0, np.pi, np.pi / N_theta)
27 variable_rho = np.arange(1, 1.6, 0.2)
28 variable_sigma = np.arange(0.2, 1.1, 0.2)
29
30 threshold = 30 # Threshold value
31 # Lookup (l_nms) for non-maximum suppression
32 # Here the l_nms indicates the number of pixels that are considered when drawing a line
33 # perpendicular to the ridge structure
34 l_nms = 4
35
36 # Lookup (l_c) for connection points and line ends
37 l_c = 3
38
39 #%% PART I: PRE-PROCESSING DATA
40 # Import DTM
41 data_path = os.chdir('../Data')
42 z = rio.open_rasterio('Clip_3.tif')
43
44 # Select data AHN
45 tif = z[0,:,:]
46 sample = np.where(tif > 1E5, np.nan, tif.values)
47
48 # Inverse Distance Weighting
49 s_mask = np.where(np.isnan(sample), 0, 1)
50 sample = fillnodata(sample, mask = s_mask)
```

```

51 #%% PART II : KERNEL CONVOLUTION
52 def kernel_conv (DEM, s = 1, t = 0, r = 1, lowest = -6, highest = 6, step = 0.1, gamma=0.75):
53     x = np.arange(highest, lowest, -step)
54     y = np.arange(lowest, highest, step)
55     x, y = np.meshgrid(x, y)
56
57     # Create kernel based on sigma, theta & rho
58     M_theta = np.array([ [np.cos(t), np.sin(t)],[-np.sin(t),np.cos(t)] ])
59     rho_sqrt = np.array([ [r**2, 0],[0, r**2] ])
60     second = np.zeros((len(x),len(y)))
61     for i in range(len(x)):
62         for j in range(len(y)):
63             xy = np.array([x[i, j],y[i, j]])
64             x_y = np.array([ [x[i, j]], [y[i, j]]])
65             phi = xy@ M_theta.transpose()@ rho_sqrt@ M_theta@ x_y
66             G_hat = (1/ (2*np.pi*s**2)) * np.exp(-phi/(2*s**2))
67             SAG = (((x[i, j] * np.cos(t)) + (y[i, j] * np.sin(t)))**2)/(r**4 * s**4) - (r**2
                / s**2)) * G_hat
68             second[i, j] = SAG
69
70     # Convolution
71     DEM_conv = - s**(2*gamma) * convolve(DEM, second)
72     return (DEM_conv)
73
74 Direction = np.zeros(np.shape(sample))
75 Sigma_final = np.zeros(np.shape(sample)) + variable_sigma[0]
76 Rho_final = np.ones(np.shape(sample)) + variable_rho[0]
77
78 for i in range (len(variable_rho)):
79     print(i)
80     for j in range (len (variable_sigma)):
81
82         for k in range (len(variable_theta)):
83             DTM_conv = kernel_conv(DEM = sample,
84                                     s = variable_sigma[j],
85                                     t = variable_theta[k],
86                                     r = variable_rho[i])
87
88             if k==0 and j==0:
89                 Result_conv = DTM_conv
90             else:
91                 Direction = np.where(DTM_conv>Result_conv, variable_theta[k], Direction)
92                 Sigma_final = np.where(DTM_conv>Result_conv, variable_sigma[j], Sigma_final)
93                 Rho_final = np.where(DTM_conv>Result_conv, variable_rho[i], Rho_final)
94                 Result_conv = np.where(DTM_conv>Result_conv, DTM_conv, Result_conv)
95
96 #%% PART III : THRESHOLDING
97 Result_binary = np.where(Result_conv>threshold, 1, 0)
98
99 #%% PART IV : NON-MAXIMA SUPPRESSION
100 x,y = np.shape(sample)
101 DTM_extra = np.empty ((2*l_nms+sample.shape[0]),(2*l_nms+sample.shape[1]))
102 DTM_extra[:] = np.nan
103 DTM_extra[l_nms:-l_nms,l_nms:-l_nms] = Result_conv
104
105 NMS = np.empty (np.shape(DTM_extra))
106 NMS[:] = np.nan
107 NMS[l_nms:-l_nms,l_nms:-l_nms] = Result_conv
108
109 Line = np.zeros(np.shape(DTM_extra))
110
111 for i in range (x):
112     print(i)
113     for j in range (y):
114         if Result_binary[i, j]==1:
115             theta = Direction[i, j]
116             a_1 = np.round(l_nms*np.cos(theta)) + l_nms
117             a_2 = np.round(l_nms*-np.cos(theta)) + l_nms
118             b_1 = np.round(l_nms*-np.sin(theta)) + l_nms
119             b_2 = np.round(l_nms*np.sin(theta)) + l_nms
120
121             im = np.asarray(Line, dtype=np.uint8)

```

```

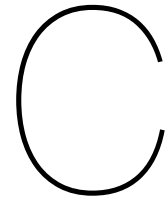
121         im = Image.fromarray(im, mode='L')
122         draw = ImageDraw.Draw(im)
123         draw.line(((a_1+j),(b_1+i),(a_2+j),(b_2+i)), fill = 1)
124         c = np.array(im)
125         arr = DTM_extra[np.where(c==1)]
126
127         if DTM_extra[i+l_nms, j+l_nms]<(np.nanmax(arr)):
128             NMS[i+l_nms, j+l_nms]=np.nan
129
130
131 Result_NMS = NMS[l_nms:-l_nms, l_nms:-l_nms]
132 Result_NMS = np.where(Result_binary==0, np.nan, Result_NMS)
133 Result_binary = np.where(Result_NMS>0, 1, 0)
134
135 ##% PART V: CONNECTION POINTS TO LINES
136 a = 2
137 b = 3
138 ridge_extra = np.zeros ((2*l_c+x),(2*l_c+y))
139 endpoints = np.zeros ((2*l_c+x),(2*l_c+y))
140 ridge_extra[l_c:-l_c, l_c:-l_c] = Result_binary
141
142 # Function to connect lines from point i, j to point x, y
143 # Input is neighbors (NB), coordinates point 1 (i, j), Binary result (RE) and distance wherein
144 # lines/points are connected
145 def connection(NB, i, j, RE, n):
146     a = np.zeros(np.shape(RE))
147     [x_coor, y_coor] = np.where(NB>0)
148     x_coor = x_coor + i-n
149     y_coor = y_coor + j-n
150
151     for k in range(len(x_coor)):
152         a = np.asarray(a, dtype=np.uint8)
153         im = Image.fromarray(a, mode='L')
154         draw = ImageDraw.Draw(im)
155
156         draw.line((j, i, y_coor[k], x_coor[k]), fill=1)
157         np.array(im)
158         RE = im + RE
159         RE = np.where(RE > 0, 1, 0) # Binary
160
161         a = np.zeros(np.shape(RE))
162     return(RE)
163
164 # Create 4 options of 3x3 kernel that can be used to find the end lines
165 arr1 = np.array([[1,0,0], [1, 1, 0], [0,0,0]])
166 arr2 = np.array([[0,0,0], [1, 1, 0], [1,0,0]])
167 arr3 = np.array([[0,1,0], [0, 1, 0], [0,0,0]])
168 arr4 = np.array([[1,0,0], [0, 1, 0], [0,0,0]])
169
170 # Find all line ends by hit or miss with created 3x3 kernel
171 for i in range(4):
172     endpoints = binary_hit_or_miss(ridge_extra, arr1) + endpoints
173     arr1 = rotate(arr1, angle=90)
174 for i in range(4):
175     endpoints = binary_hit_or_miss(ridge_extra, arr2) + endpoints
176     arr2 = rotate(arr2, angle=90)
177 for i in range(4):
178     endpoints = binary_hit_or_miss(ridge_extra, arr3) + endpoints
179     arr3 = rotate(arr3, angle=90)
180 for i in range(4):
181     endpoints = binary_hit_or_miss(ridge_extra, arr4) + endpoints
182     arr4 = rotate(arr4, angle=90)
183
184 # Connect all line ends and separate point that are within a distance of length l
185 # values a and b are used to create local neighbors
186 for i in range(l_c, x+l_c):
187     for j in range(l_c, y+l_c):
188         if (ridge_extra[i, j]==1):
189
190             # Neighboring points

```

```

191     neighbors = ridge_extra[i-1:i+2,j-1:j+2]
192     if np.sum(neighbors)==1:
193         a = 2
194         b = 3
195
196         if np.sum(ridge_extra[(i-l_c):(i+l_c+1),(j-l_c):(j+l_c+1)])==1:
197             ridge_extra[i,j]=0
198
199         for k in range(l_c):
200             if np.sum(ridge_extra[(i-a):(i+b),(j-a):(j+b)])>1:
201                 neighbors = ridge_extra[(i-a):(i+b),(j-a):(j+b)]
202                 ridge_extra = connection(neighbors,i,j,ridge_extra,n=a)
203                 a = a+1
204                 b = b+1
205
206     # Connection line ends
207     if (endpoints[i,j]==1):
208         a = 2
209         b = 3
210         for k in range(l_c):
211             if np.sum(endpoints[(i-a):(i+b),(j-a):(j+b)])==2:
212                 neighbors = endpoints[(i-a):(i+b),(j-a):(j+b)]
213                 endpoints = connection(neighbors,i,j,endpoints,n=a)
214                 a = a+1
215                 b = b+1
216
217 ridge_extra = ridge_extra + endpoints
218 Result_binary = np.where(ridge_extra[l_c:-l_c,l_c:-l_c] > 0 ,1 ,np.nan)
219                               # Binary
219 Result = np.where(Result_binary==1, Result_conv, np.nan)
220                               # Fill with values convolution
221
221 end = time.time()
222 print(end - start)
223
224 # Thinning lines for further processing in qgis
225 Result_binary = skeletonize(Result_binary)
226 Result_binary = np.where(Result_binary==1,1,np.nan)
227
228 plt.figure(figsize=(8,8))
229 plt.imshow(Result_binary)
230
231
232 #%% " "SAVE RESULTS" "
233
234 data_path = os.chdir('../Results')
235
236 tif = z
237 tif[0,:,:] = Result_binary
238 tif.rio.to_raster("A2_Results_binary.tif")
239
240 tif = z
241 tif[0,:,:] = Result
242 tif.rio.to_raster("A2_Results.tif")
243
244
245 tif = z
246 tif[0,:,:] = Rho_final
247 tif.rio.to_raster("A2_rho.tif")
248
249 tif = z
250 tif[0,:,:] = Sigma_final
251 tif.rio.to_raster("A2_sigma.tif")
252
253 tif = z
254 tif[0,:,:] = Result_conv
255 tif.rio.to_raster("A2_v2_results_convolution.tif")

```

Python code algorithm 3

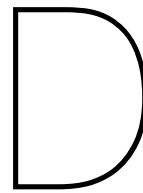
```
1 # -*- coding: utf-8 -*-
2 """
3
4 Created on Wed Apr 21 11:38:09 2021
5
6 @author: noppen
7 """
8
9 import numpy as np
10 import rioarray as rio
11 import os
12 from scipy.ndimage import convolve
13 from rasterio.fill import fillnodata
14 from skimage.morphology import skeletonize
15 import time
16
17 start = time.time()
18
19 ### Parameters
20
21 # Parameters voor kernels
22 N_theta = 6 # Number of orientations
23 variable_theta = np.arange(0,np.pi,np.pi/N_theta)
24 variable_sigma = np.arange(0.2,1.1, 0.2) # Size kernel
25
26 threshold = 15 # Threshold value
27
28 ### PART I: PRE-PROCESSING DATA
29 # Import DTM
30 data_path = os.chdir('../Data')
31 z = rio.open_rasterio('Clip_3.tif')
32
33 # Select data AHN
34 tif = z[0,:,:]
35 sample = np.where(tif > 1E5, np.nan, tif.values)
36
37 # IDW > Fill nan values
38 s_mask = np.where(np.isnan(sample), 0, 1)
39 sample = fillnodata(sample, mask = s_mask)
40
41
42 ### PART II: KERNEL CONVOLUTION
43
44 def kernel_conv (DEM, s = 1, t = 0, r = 1, lowest = -5, highest = 5, step = 0.1, gamma =
45 0.75):
46     x = np.arange(highest, lowest, -step)
47     y = np.arange(lowest, highest, step)
48     x, y = np.meshgrid(x, y)
49
50     # Create kernel based on sigma, theta & rho
51     M_theta = np.array([ [np.cos(t), np.sin(t)], [-np.sin(t), np.cos(t)] ])
```

```

51 rho_sqrt = np.array([ [r**2, 0],[0, r**-2] ])
52 z_second = np.zeros((len(x),len(y)))
53 for i in range(len(x)):
54     for j in range(len(y)):
55         xy = np.array([x[i,j],y[i,j]])
56         x_y = np.array([ [x[i,j]], [y[i,j]]])
57         phi = xy@ M_theta.transpose()@ rho_sqrt@ M_theta@ x_y
58         G_hat = (1/ (2*np.pi*s**2)) * np.exp(-phi/(2*s**2))
59         SAG = (((x[i,j] * np.cos(t)) + (y[i,j] * np.sin(t)))**2)/(r**4 * s**4) - (r**2
        / s**2)) * G_hat
60         z_second[i,j] = SAG
61
62 # Convolution
63 DEM_conv = -(s**2)**gamma * convolve(DEM, z_second)
64 return (DEM_conv)
65
66 Direction = np.zeros(np.shape(sample))
67 Sigma_final = np.zeros(np.shape(sample)) + variable_sigma[0]
68
69 for j in range (len (variable_sigma)):
70     print(j)
71     for k in range (len(variable_theta)):
72         DTM_conv = kernel_conv(DEM = sample,
73                                 s = variable_sigma[j],
74                                 t = variable_theta[k])
75
76         if k==0 and j==0:
77             Result_conv = DTM_conv # Wordt uiteindelijke resultaat in
78             opgeslagen van convolutie
79
80         else:
81             Sigma_final = np.where(DTM_conv>Result_conv, variable_sigma[j], Sigma_final)
82             Direction = np.where(DTM_conv>Result_conv, variable_theta[k], Direction)
83             Result_conv = np.where(DTM_conv>Result_conv, DTM_conv, Result_conv)
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999

```

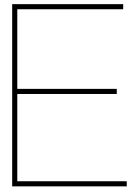




Post-processin steps in QGIS

1. To remove line elements in water bodies: (1) multiply with mask layer (0 for water 1 for no water)	2. To remove line elements in water bodies: (2) Set 0 to nan-values	3. Run gdal translate (int32)	4. Run r.thin in order to perform raster to vector tool.	5. Write raster to shapefile r.to.vect (GRASS)
6. Merge lines	7. Dissolve lines Vector > geoprocessing tools > Dissolve	8. Smooth Vector > geoprocessing tools > Simplify	9. Densify by interval (create vertices at interval of 1m)	10. Split line by specific distance with v.Split (GRASS)
11. Intersect with shape file D-HYDRO model	12. Drape (set z values from raster) . To attach elevation to vertices	13. Multipart to singleparts . To separate line elements	14. Export layer > geometry > include z-value	15. Run python-code to create pliz-file from shape file

Figure D.1: Post-processing steps QGIS.



Difference in water depth

Algorithm 1 vs. fine grid model
10 January 00:00

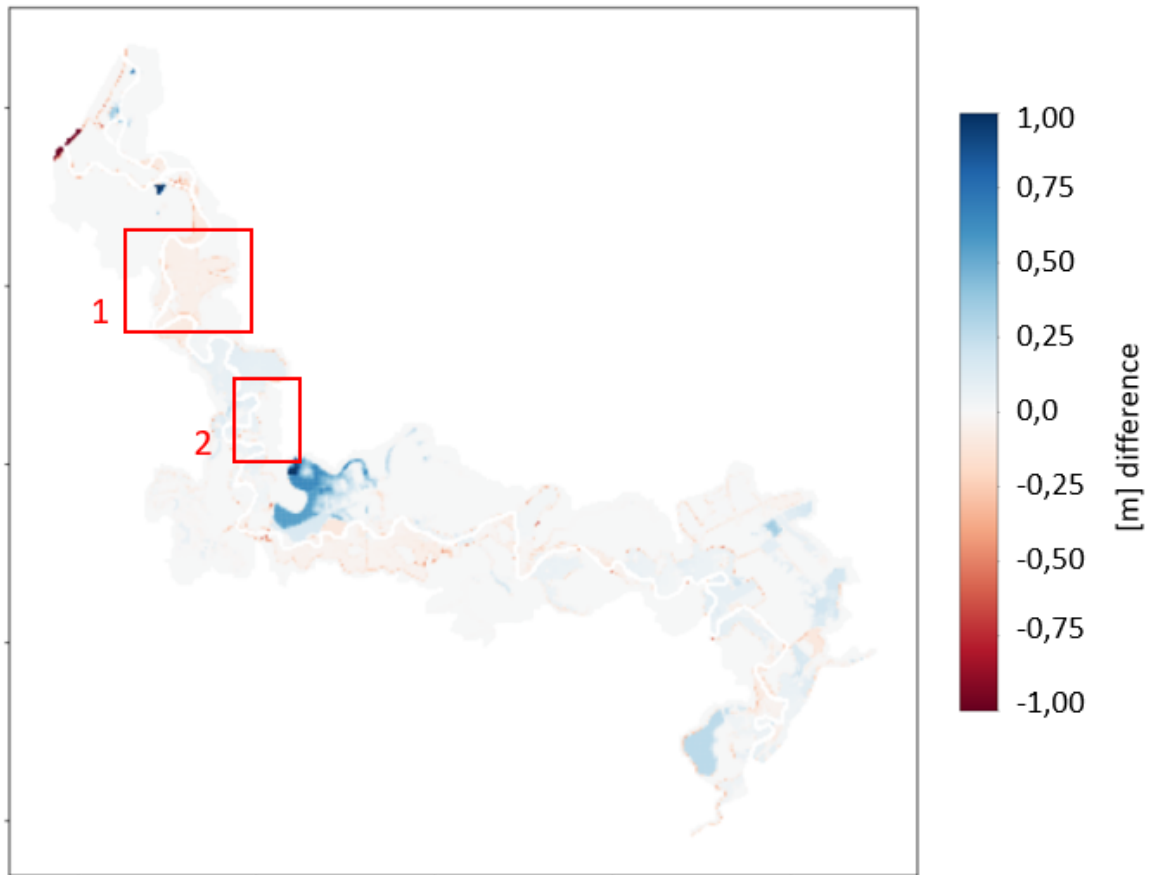


Figure E.1: Difference in water depth fine grid model versus algorithm 1.

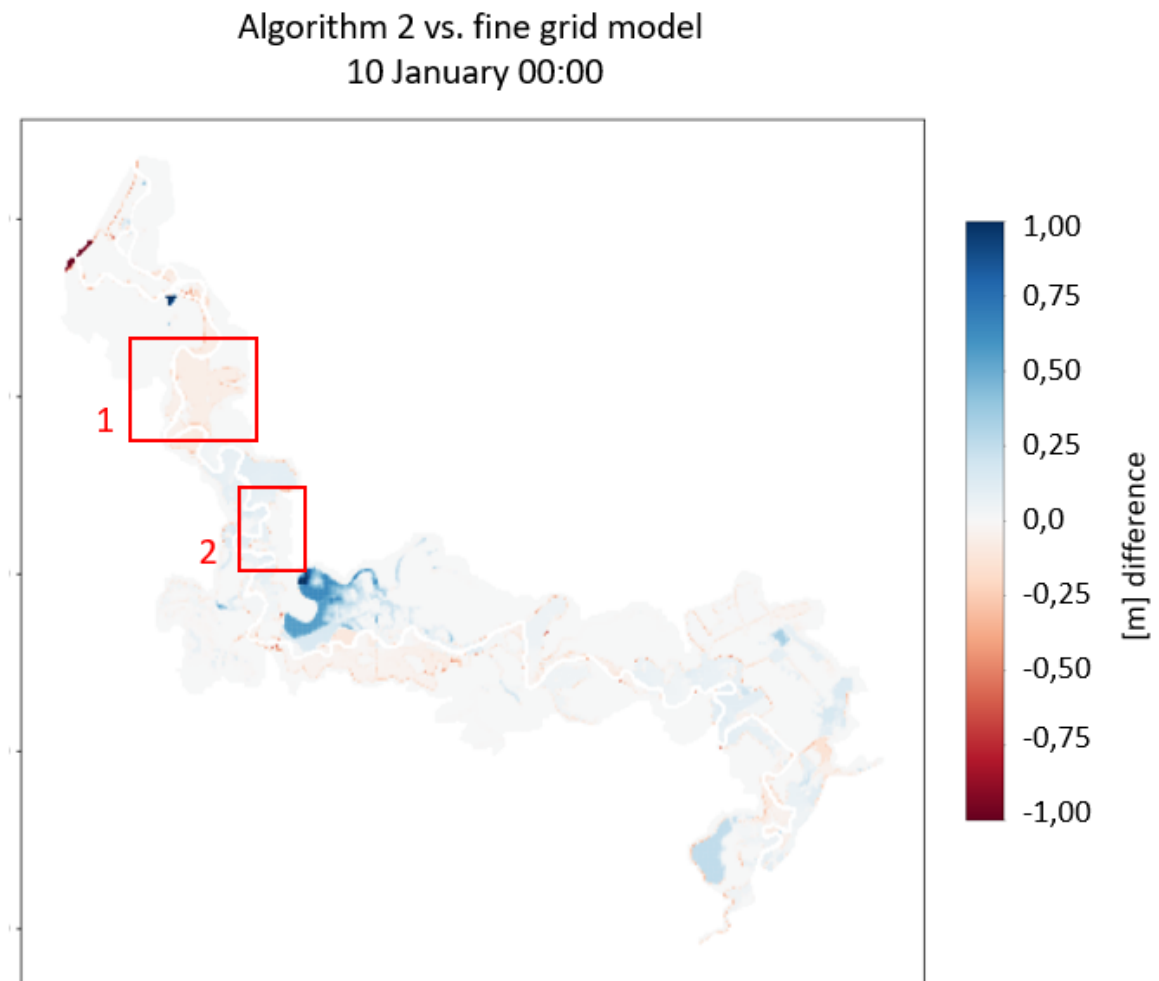


Figure E.2: Difference in water depth fine grid model versus algorithm 2.

Algorithm 3 vs. fine grid model
10 January 00:00

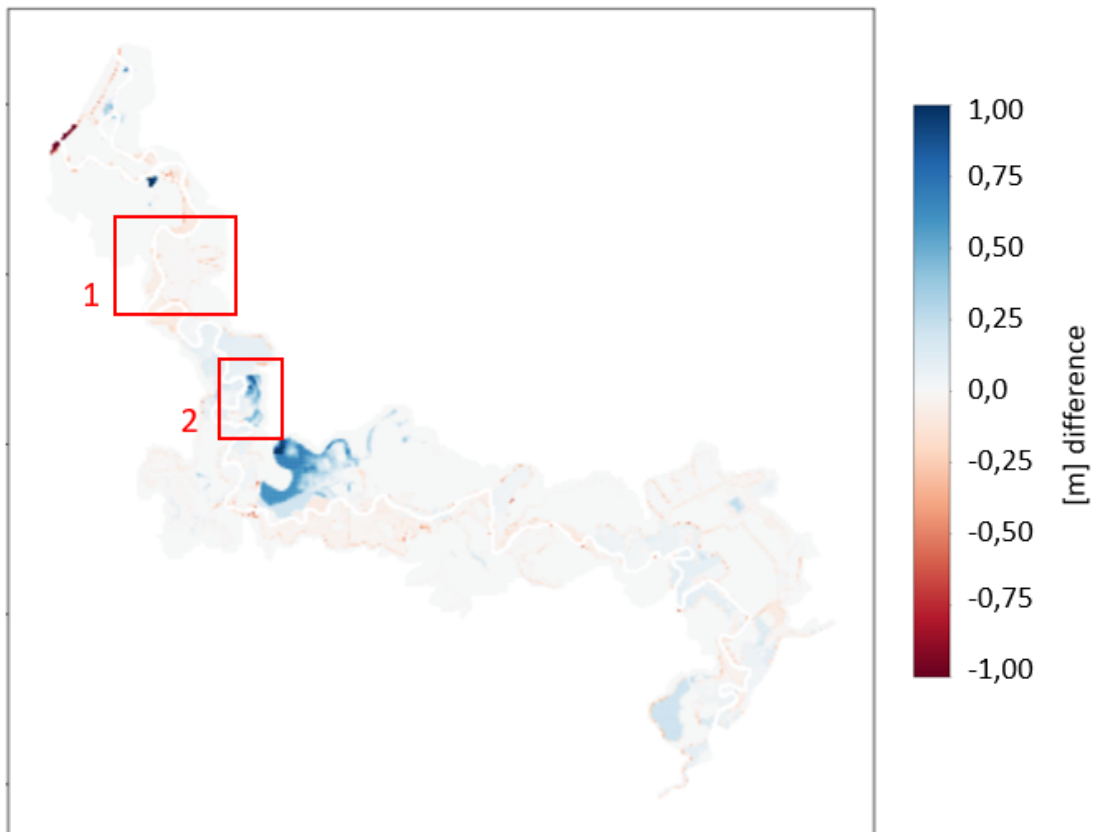


Figure E.3: Difference in water depth fine grid model versus algorithm 3.

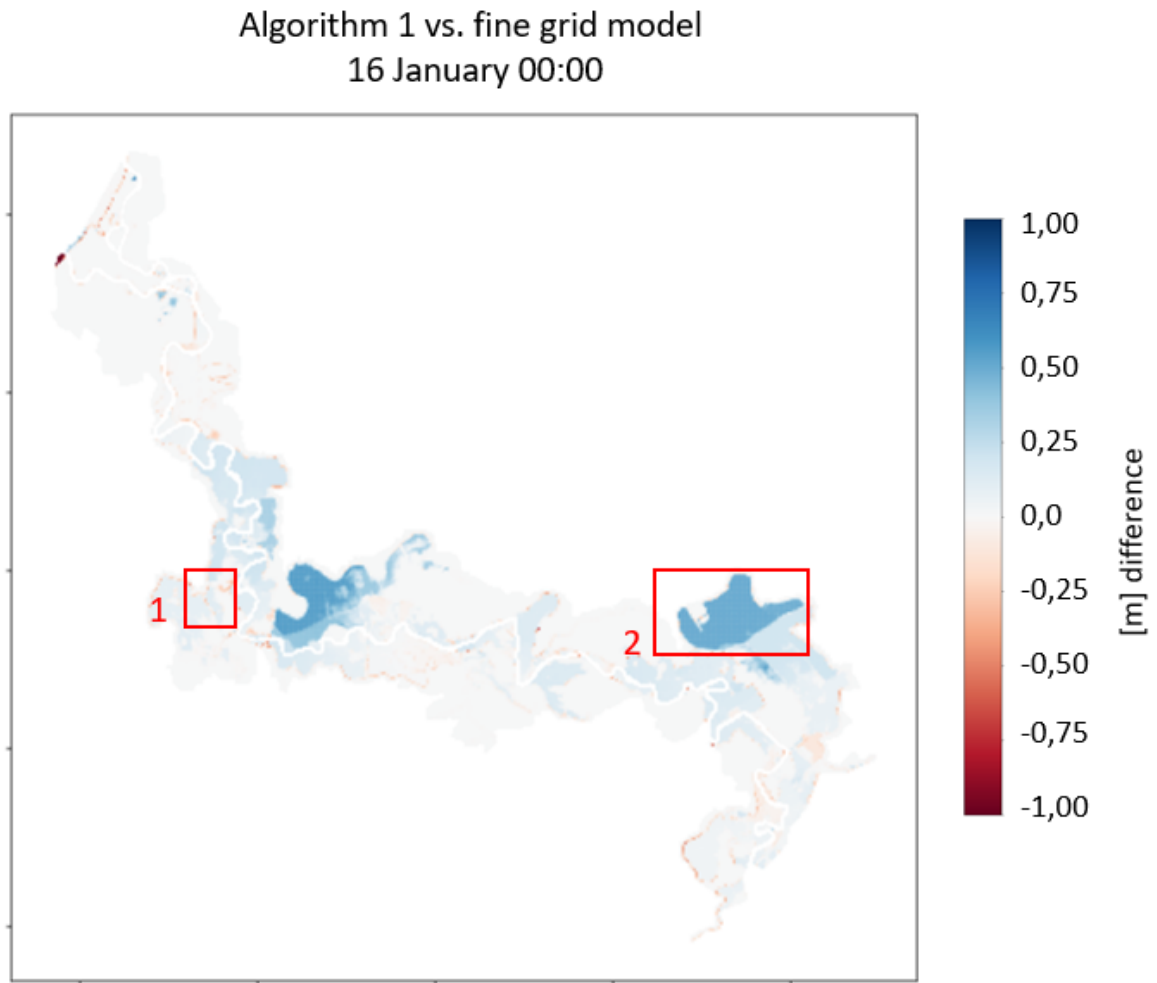


Figure E.4: Difference in water depth fine grid model versus algorithm 1.

Algorithm 2 vs. fine grid model
16 January 00:00

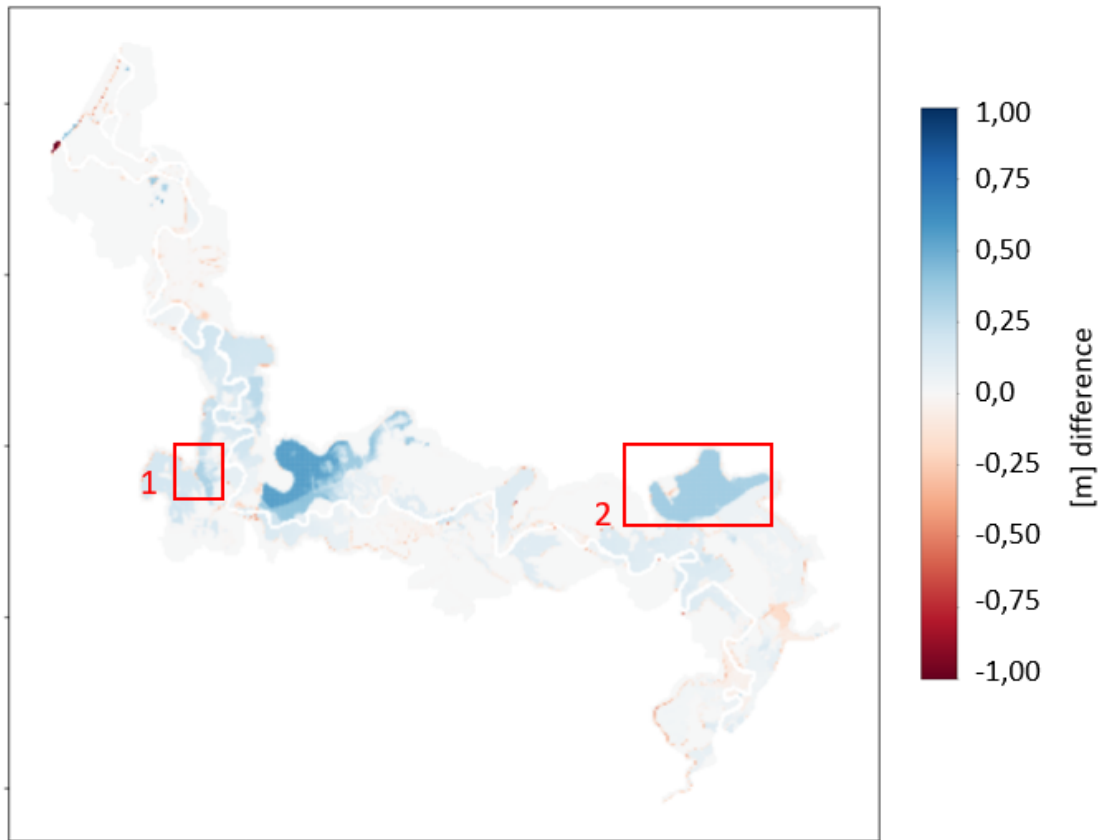


Figure E.5: Difference in water depth fine grid model versus algorithm 2.