

SLOVAK UNIVERSITY OF TECHNOLOGY IN BRATISLAVA  
FACULTY OF CIVIL ENGINEERING

Reg. No.: SvF-5342-104394

CORRECTION OF WATERCOURSES IN MAPS  
USING AIRBORNE LASER SCANNING DATA

Bachelor's thesis



SLOVAK UNIVERSITY OF TECHNOLOGY IN BRATISLAVA  
FACULTY OF CIVIL ENGINEERING

CORRECTION OF WATERCOURSES IN MAPS  
USING AIRBORNE LASER SCANNING DATA

Bachelor's thesis

Study programme: Mathematical and Computer Modelling  
Study field: 9.1.9. Applied Mathematics  
Training workplace: Department of Mathematics and Descriptive Geometry  
Supervisor: Ing. Mgr. Lukáš Tomek, PhD.





## BACHELOR THESIS TOPIC

Student: **Tomáš Homola**  
Student's ID: 104394  
Study programme: Mathematical and Computational Modeling  
Study field: Mathematics  
Thesis supervisor: Ing. Mgr. Lukáš Tomek, PhD.  
Head of department: Ing. Marek Macák, PhD.

Topic: **Correction of watercourses in maps using airborne laser scanning data**

Language of thesis: English

Specification of Assignment:

Vodné toky sú v mapách niekedy zakreslené len približne a nekopírujú skutočnú polohu koryta, pričom rozdiely môžu byť aj na úrovni desiatok metrov. Ako toto nesprávne trasovanie opraviť?

V súčasnosti (2017 – 2023) na Slovensku beží projekt, v ktorom sa pomocou leteckého laserového skenovania vytvára nový digitálny model reliéfu (DMR) celého územia Slovenskej republiky v rozlíšení 1 x 1 m. Takýto podrobný DMR je veľmi užitočný pre ľudí, ktorí sa venujú mapovaniu, lebo s jeho využitím dokážu manuálne korigovať nesprávne zakreslené objekty v mapách. Ako to však urobiť automaticky?

Hlavným cieľom bakalárskej práce je vytvoriť matematický model, ktorý dokáže nesprávne trasovanie vodného toku opraviť tak, aby tok v mape kopíroval skutočné koryto. Ústredným objektom bude vyvíjajúca sa krivka. Počiatočnú krivku (existujúce trasovanie v mape) chceme počas vývoja pritiahnúť do koryta, ktoré je viditeľné v DMR.

Na ceste za týmto cieľom študenta čaká množstvo úloh z rôznych oblastí:

### 1. Kartografia

- Pochopiť, ako sú vodné toky zakreslené v mapách (využívať sa bude voľne dostupná OpenStreetMap) a naučiť sa ich trasovanie exportovať a zobrazit' v iných softvéroch. Na export a editovanie kartografických dát sa využije softvér JOSM (Java OpenStreetMap Editor).

### 2. Geodézia

- Naučiť sa získavať dáta DMR ako aj klasifikovaných mračien bodov, na základe ktorých bol DMR vytvorený. Dáta sa budú získavať zo Základnej bázy údajov pre geografický informačný systém (ZBGIS).
- S geodetickými dátami zo ZBGIS sa treba naučiť pracovať a zobrazovať ich. Najskôr v geografickom informačnom systéme QGIS a neskôr v MATLAB-e.

### 3. Matematika

- Naštudovať úvod do diferenciálnej geometrie kriviek. Najprv statických kriviek, a následne vyvíjajúcich sa kriviek.
- Navrhnuť vhodné rýchlostné vektorové pole, ktoré krivku pritiahne do koryta. Využiť sa môže gradient DMR, ale aj gradient vzdialenostnej funkcie od tých bodov mračna, ktoré sú klasifikované ako „water“ (v prípade, že je takáto klasifikácia dostupná).

#### 4. Programovanie

- Študent najskôr musí zvládnuť niektoré MATLAB tutorialy.
- Práca nadväzuje na existujúci kód v MATLAB-e pre vývoj krivky riadený krivosťou, v ktorom je potrebné sa zorientovať.
- Implementovať do kódu potrebné rozšírenia.

Téma je náročná a je vhodná pre ambiciózneho študenta, ktorého zaujímajú praktické aplikácie matematiky a práca s reálnymi dátami.

Deadline for submission of Bachelor thesis: 05. 05. 2022

Approval of assignment of Bachelor thesis: 03. 05. 2022

Assignment of Bachelor thesis approved by: prof. RNDr. Karol Mikula, DrSc. – Study programme supervisor

## **Guidelines**

### **for Bachelor Thesis Writing**

#### **Preliminary provision**

In accordance with the Act No. 131/2002 Coll. on University Education and on Amending and Supplementing Certain Acts as Amended by Later Regulations, a bachelor thesis forms part of the studies in every study programme. The defence of a bachelor thesis is part of the state exam. A bachelor thesis is the final thesis in Bachelor's study programmes. The basis for writing a bachelor thesis is the assignment of a bachelor thesis.

#### **Thesis Structure**

- Title page
- Assignment of the bachelor thesis
- Guidelines for writing
- Author's declaration
- Title and abstract in English and Slovak (one page)
- Table of contents with chapter numbers
- List of appendices
- List of abbreviations and symbols
- Text of the thesis (recommended structuring),
  - Introduction
  - Current situation with respect to the thesis research problem
  - Goals of the thesis
  - Author's solution to the problem divided into chapters depending on the character of the thesis
  - Evaluation of the achieved results or of the proposed solutions
  - Conclusion
- Summary (concerns only a thesis written in a language other than Slovak)
- Bibliography
- Appendices (plans, tables, maps, sketches) including a poster of 1000x700 mm

#### **Length and Form**

1. The contents and the form of the graduation thesis must be handled in accordance with the regulation of the Ministry of Education, Science, Research and Sport of the Slovak Republic No. 233/2011 Coll., which is used to implement certain provisions of the Act No. 131/2002 Coll., and in accordance with the Methodical regulation No. 56/2011 on Essentials of Final Theses.
2. The required length of the thesis is 20-30 pages. The thesis must be submitted in two bound copies, at least one of them has to be in hardback (not in ring binding) so selected pages cannot be removed. With the approval of the supervisor it is possible to submit extensive graphic appendices in one copy.

3. The author is required to submit the thesis electronically in the information system. The author is responsible for the compliance of the paper and electronic version.
4. After the submission of the thesis in the information system, the author is obligated to deliver a draft of the licence agreement to the faculty. The draft of the licence agreement is generated by academic information system.
5. The recommended font type is Times New Roman, font size 12, and it should stay the same throughout the thesis. The recommended page settings are: spacing – 1.5; left margin – 3.5 cm; right margin – 2 cm; top and bottom margin – 2.5 cm; vertical page orientation; A4 format.
6. Figures and equations should be numbered within individual chapters, e.g. figure No. 3.1 is the figure No. 1 in chapter 3. Equations should be numbered at the right end of the line in parentheses, e.g. (3.1).
7. All the calculations should be clearly arranged, so that their correctness can be verified.
8. In case of borrowed equations, tables or cited parts of another text, the source must be indicated.
9. Quotations of other texts, including electronic materials, must be presented according to STN ISO 690 (01 0197) : 2012. *Information and Documentation. Guidelines for Bibliographic References and Citations to Information Resources.*
10. Examples of bibliographical entries:  
 BOWLES, J: *Foundation analysis and design*. Singapore, The McGraw-Hill Companies, Inc. 1997, ISBN 0-07-912247-7.  
 MICHALČÁK, O. – ADLER, E.: Výskum stability dunajských hrádzí. In: *Zborník vedeckých prác Stavebnej fakulty SVŠT*, Bratislava, Edičné stredisko SVŠT 1976, ISBN 0-3552-5214.  
 ŠÜTTI, J.: Určovanie priestorových posunov stavebných objektov. *Geodetický kartografický obzor*, 35 (77), 1987, č. 2, ISSN 0811-6900.  
 Article 18. Technical Cooperation. <http://www.lac.uk/iso/tc456> (2013-09-28)
11. The author of the thesis is responsible for its linguistic and terminological correctness.
12. The form of the poster (electronic or printed) is determined by the guarantor of each study programme.
13. A template of the poster can be found on the document server in the university's information system.

.....  
 study programme guarantor's signature

I have taken the provisions of the above-mentioned guidelines into consideration. I am aware of the fact that if my bachelor thesis is not written in conformity with these guidelines, it will not be accepted.

In Bratislava .....

.....  
 student's signature



## **Declaration**

I declare that this thesis has been composed solely by myself under supervision of my supervisor and using the literature stated in Bibliography.

Bratislava 5. 5. 2022

Tomáš Homola



## **Acknowledgement**

I would like to express my gratitude to my supervisor Ing. Mgr. Lukáš Tomek, PhD. Not only for his great guidance, feedback and time given to me, but also for making me genuinely enjoy working on this thesis.

Bratislava 5. 5. 2022

Tomáš Homola

## Abstract

**Title:** Correction of watercourses in maps using airborne laser scanning data

**Abstract:** In this thesis we deal with the use of evolving planar curves to create more accurate map depiction of watercourses. For this purpose we use data from airborne laser scanning and from the OpenStreetMap database. We introduce all the input data for our mathematical model which includes classified point cloud, digital terrain model and sequence of watercourse's nodal points. Then we describe and explain all components of our mathematical model for curve evolution in detail, as well as demonstrate its functionality in several numerical experiments using real data. All computations are implemented in MATLAB software.

**Keywords:** watercourse, planar curve evolution, classified point cloud, digital terrain model, MATLAB

## Abstrakt

**Názov práce:** Korekcia trasovania vodných tokov v mapách pomocou dát z leteckého laserového skenovania

**Abstrakt:** V tejto práci sa zaoberáme využitím vyvíjajúcich sa rovinných kriviek na vytvorenie presnejšieho vyobrazenia vodných tokov na mapách. Na to využívame dáta z leteckého laserového skenovania a OpenStreetMap databázy. Predstavíme všetky vstupné dáta pre náš matematický model, ktoré zahŕňajú klasifikované mračno bodov, digitálny model reliéfu a postupnosť uzlových bodov vodného toku. Následne detailne opíšeme a vysvetlíme všetky časti nášho modelu pre vývoj krivky a taktiež ukážeme jeho funkčnosť s využitím reálnych dát. Všetky výpočty sú implementované v softvéri MATLAB.

**Kľúčové slová:** vodný tok, vývoj rovinnnej krivky, klasifikované mračno bodov, digitálny model reliéfu, MATLAB

# Preface

Goal of this thesis is to create a mathematical model for fixing inaccurate depictions of watercourses on maps. For this model, we use planar curve evolution driven by a properly designed vector field using data from airborne laser scanning. We implement our model in MATLAB software [8] with the use of pre-existing code, which was developed as part of the project APVV-18-0247, titled "Automation of Building's Electronic Documentation Verification Using Innovative Data Collection Techniques and Virtual Models".

The code I was provided with from my supervisor included code for curve evolution driven by signed curvature which uses a stable semi-implicit IIOE (Inflow Implicit/Outflow Explicit) finite volume scheme, code for the tangential redistribution which uses a finite difference method applied to the equation (3.23) in section 3.4, code for the initial import of the terrain and OpenStreetMap data and the function for the change of coordinates as described in section 2.3. My task was to expand this code with new functionalities, including the design of the new normal speed for the curve evolution and the new stopping criterion, multiple plotting functions and code for the backward transformation of the new coordinates and subsequent export to GPX file.

The core of this thesis consists of four chapters. In chapter 1 we give a very basic introduction about this thesis. This includes a simple explanation of our suggested approach for adjustments of watercourses' mapping and the data we plan to use. We also state a few examples about the use of curve evolution in other fields.

In chapter 2 we first provide details about the project "Airborne Laser Scanning and DTM 5.0" launched by the Geodesy, Cartography and Cadastre Authority of the Slovak Republic. Then we explain how to export the terrain data (results of the airborne laser scanning) via the web application Map Client ZBGIS. In section 2.1.1, we briefly describe how airborne laser scanning is performed. Then we explain what precisely is a classified point cloud and how we can import it into MATLAB. We do the same for a digital terrain model in section 2.1.2. In section 2.2 we describe the OpenStreetMap database from which we download watercourse's nodal points (a sequence of GPS coordinates that create a watercourse map depiction). Since these nodal points are saved in a different coordinate reference system compared to the terrain data, we need to transform them first. This process is described in detail in section 2.3, together with some basics from geodesy regarding geodetic datums, different coordinates within a geodetic datum and transformations between them.

At the beginning of chapter 3 in section 3.1 we start by defining an evolving parametric curve, the backbone of our mathematical model. Then in section 3.2 we first introduce a very basic form of our model (3.1) which we subsequently rewrite into form (3.4). In section 3.3 we define the normal speed  $\beta$  by equation (3.3), then we describe how to construct a suitable velocity vector field  $\vec{V}$ . In section 3.4 we introduce the tangential speed  $\alpha$  which improves the mesh quality.

In chapter 4 we first introduce three watercourses we have chosen for the numerical experiments. In section 4.1 we explain the individual steps of our workflow, including the algorithm of the curve evolution and then stopping criterion in section 4.1.1. Results of our numerical experiments are presented in sections 4.2, 4.3 and 4.4.

Throughout this thesis, in its electronic version, we have included multiple hyperlinks referencing to the website [www.freemap.sk](http://www.freemap.sk). We encourage the reader to open each hyperlink to explore the map zoomed to the area of interest.

We have also created a playlist on YouTube where we have uploaded the videos of evolving curves from the numerical experiments. It is accessible via the link [https://bit.ly/evolvingCurves\\_playlist](https://bit.ly/evolvingCurves_playlist). The links to the specific evolution videos can be found in the corresponding sections of chapter 4.

# Contents

<b>1</b>	<b>Introduction</b>	<b>15</b>
<b>2</b>	<b>Data for the model</b>	<b>16</b>
2.1	Terrain data . . . . .	16
2.1.1	Classified point cloud . . . . .	17
2.1.2	Digital terrain model . . . . .	19
2.2	OpenStreetMap data . . . . .	20
2.3	Coordinate system transformation . . . . .	21
<b>3</b>	<b>Mathematical model</b>	<b>24</b>
3.1	Evolving parametric curve . . . . .	24
3.2	Formulation of the mathematical model . . . . .	25
3.3	Choice of the normal speed . . . . .	26
3.3.1	Terrain function . . . . .	26
3.3.2	Distance function . . . . .	29
3.3.3	Weighted combination . . . . .	30
3.4	Choice of the tangential speed . . . . .	31
<b>4</b>	<b>Numerical experiments</b>	<b>33</b>
4.1	Workflow description . . . . .	33
4.1.1	Stopping criterion . . . . .	34
4.2	Examples: Stream Starý potok . . . . .	35
4.2.1	Phase 1: Evolution driven by the distance function . . . . .	36
4.2.2	Phase 2: Evolution driven by the terrain function . . . . .	38
4.3	Examples: Stream Parná . . . . .	40
4.3.1	Phase 1: Evolution driven by the distance function . . . . .	40
4.3.2	Phase 2: Evolution driven by the terrain function . . . . .	42
4.4	Examples: Nameless stream in Tatras . . . . .	44
4.4.1	Evolution before subtracting the terrain's trend . . . . .	44
4.4.2	Evolution after subtracting the terrain's trend . . . . .	45
<b>5</b>	<b>Conclusions</b>	<b>46</b>
	<b>Resumé</b>	<b>47</b>
	<b>Bibliography</b>	<b>49</b>

# Chapter 1

## Introduction

Watercourses are often plotted incorrectly on maps. However, thanks to the new Digital Terrain Model, created as part of the project titled "Airborne Laser Scanning and DTM 5.0" launched by the ÚGKK SR<sup>1</sup> (see [18, 19]), we can manually fix the inaccurate mapping of watercourses. This is possible thanks to the high resolution (1 m × 1 m) of the digital terrain model, meaning, we can see where the real waterbed is located. Thus, we can contribute ourselves to having much more accurate maps for everyone.

On the other side, can we somehow automate this process with the use of mathematics and programming? The answer is yes, we most certainly can. We will use the knowledge from the field of differential geometry, specifically *curve evolution* to create a mathematical model which will do the fixing for us. Models using the curve evolution can be found in many other areas, for instance in medicine for virtual colonoscopy [12], for image segmentation [3, 13] or to model forest fire propagation [1, 2].

Our main focus will be to use the available terrain data (classified point cloud and digital terrain model, both described in chapter 2) for construction of a suitable velocity vector field  $\vec{V}$ . Then we will place the current mapping of some watercourse as the initial curve  $\gamma_0$  into the velocity field  $\vec{V}$  and let it guide  $\gamma_0$ , hopefully towards the real waterbed, see Figure 1.1.

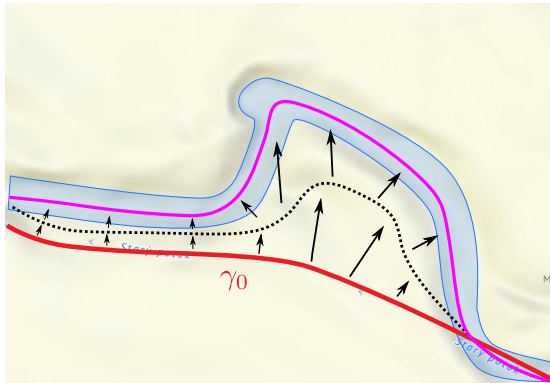


Figure 1.1: Part of the stream Starý potok with the highlighted original mapping  $\gamma_0$  (red). Via the curve evolution we then compute more accurate mapping (magenta).

Numerical part of this work is implemented in MATLAB software. Here we took advantage of the already programmed code for curve evolution driven by signed curvature which uses the stable *semi-implicit IIOE* (Inflow Implicit/Outflow Explicit) *finite volume scheme* [9, 13]. Additionally, to improve mesh quality, this code also includes the *tangential redistribution* of curve points during the evolution [2, 10, 14] as well as the *curvature regularization* described in section 3.3.3. Lastly, based on the results presented in chapter 4 we will evaluate the overall functionality of our model. Meaning, whether it has managed to sufficiently fix the inaccurate mapping of watercourses or if there is still some room for future improvements.

---

<sup>1</sup>Geodesy, Cartography and Cadastre Authority of the Slovak Republic.

# Chapter 2

## Data for the model

### 2.1 Terrain data

As the source of the input data for our model we used the results of the project titled "Airborne Laser Scanning and DTM 5.0" started in 2017 by the *Geodesy, Cartography and Cadastre Authority of the Slovak Republic* (ÚGKK SR). Its goal is to create a new Digital Terrain Model (DTM) of the whole territory of Slovakia based on the data obtained from airborne laser scanning (ALS). Entire territory of Slovakia is divided into 42 regions, also called *lots*. The ALS is carried out by the private sector and the lots are scanned separately, starting in the western Slovakia, then continuing eastwards. Current progress can be seen in Figure 2.1. Already scanned lots are highlighted in green.

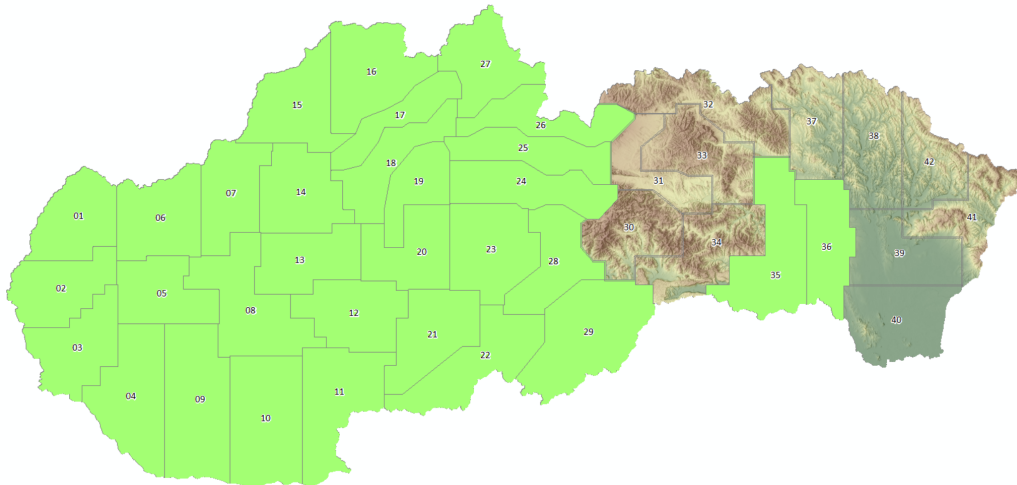


Figure 2.1: Division of Slovakia into individual lots. Source of the picture: [19].

As stated in [19], the process begins with the gathering of data using the ALS. The output data set, called *Point Cloud* (PC), is then processed into a *Classified Point Cloud* (classified PC). Next step is to compute the digital terrain and surface models (DTM and DSM) and subsequently, the quality control is performed. As described in [18], the terrain data (DTM and classified PC) must meet the following mandatory criteria:

- scanning density at least 5 points per  $\text{m}^2$ ,
- lots with 250 m swath on 95% of their overlap,
- terrestrial and vertical reference systems:

$$- \text{S-JTSK(JTSK03)}^1 + \text{H}_{\text{Bpv}}^2,$$

<sup>1</sup>System of the Unified Trigonometrical Cadastral Network. In Slovak: Systém Jednotnej Trigonometrickej Siete Katastrálnej.

<sup>2</sup>Baltic vertical reference frame after adjustment. In Slovak: Baltský výškový systém po vyrovnaní.



$$- \text{ETRS89-TM34}^3 + h_{\text{ETRS89}},$$

- absolute vertical accuracy of point cloud at ellipsoidal heights:  $|\text{ETRS89} - m_h| \leq 0.15 \text{ m}$ ,
- absolute location accuracy of point cloud:  $|\text{ETRS89-TM34} - m_{XY}| \leq 0.30 \text{ m}$ ,
- absolute vertical accuracy of DTM:  $|h_{\text{ETRS89}} - m_H| \leq 0.20 \text{ m}$ ,
- absolute vertical accuracy of DTM:  $|B_{pv} - m_H| \leq 0.25 \text{ m}$ ,

where  $m_h$  and  $m_H$  stand for the measured elevation and  $m_{XY}$  for the measured location. The terrain data are freely available, but due to the substantial size, one has to provide own external hard-drive and either send or deliver it to the offices of ÚGKK SR in Bratislava or Prešov. Alternatively, as we have used in this work, smaller volumes of terrain data can be exported via the web application Map Client ZBGIS, then the data are delivered to the email address provided by the user. Selected regions for export are limited in size,  $400 \text{ km}^2$  for DTM and  $2 \text{ km}^2$  for classified PC. Source of all ALS products used in this work: ÚGKK SR.

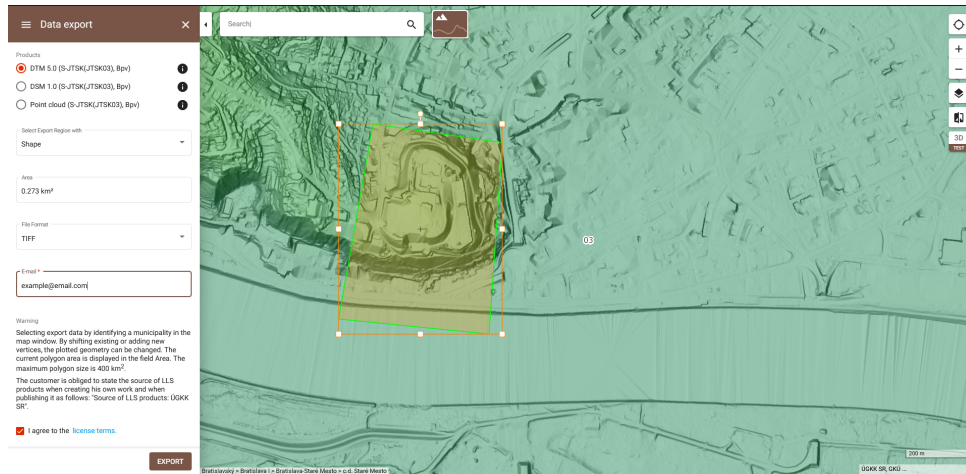


Figure 2.2: Map Client ZBGIS. For illustrative purposes we chose the area around Bratislava Castle. Individual visualizations of DTM, DSM and classified PC are shown in Figures 2.6a, 2.6b and 2.5, respectively.

### 2.1.1 Classified point cloud

The scanning is done by using an aircraft attached with a LiDAR<sup>4</sup> sensor, together with a GPS device and a computer processing all the data. The basic principle of ALS is as follows. The LiDAR sensor emits light energy (called a *light pulse*) towards the ground. Then the time it takes for the light pulse to bounce from the ground and return back is measured. Using basic physics, the time measured multiplied by the speed of light, then divided by two gives us the distance  $d_{GP}$  between the spot on the Ground the light pulse hit and the Plane. Last step is to subtract the distance  $d_{GP}$  from the plane's altitude to calculate the actual elevation  $z$  at the place the light pulse hit. This information is subsequently saved together with the corresponding GPS coordinates as a triplet  $(x, y, z)$  which is regarded as *one distinct point* in the PC. The intensity of the returning light pulse can also be recorded and saved.

To be more accurate, a few more variables are included in the calculation. For instance, the light pulses are emitted in more than one direction to cover larger area. Most pulses travel from the LiDAR sensor at an angle, so the computer needs to account for this angle during the calculations. Additionally, the computer also needs to compensate the changes recorded from the plane's inertial measurement unit (*yaw*, *roll* and *pitch*). This whole process is illustrated in Figure 2.3.

As stated in [18], the ALS is primarily performed during the winter season (from November to April). The only exceptions are four mountainous regions 17 (Little Fatra), 19 (Great Fatra), 24 (Low Tatras) and 26 (Tatras). In these regions the scanning is done from May to September. When the ALS is finished, the scanned PC is then processed into the classified PC according to:

- 1) Mandatory classification into classes: 01-*Unclassified* and 02-*Ground*.

<sup>3</sup>European Terrestrial Reference System 1989 zone 34.

<sup>4</sup>LiDAR is an acronym for Light Detection And Ranging.

- 2) Optional classification into classes: 01-*Unclassified*, 02-*Ground*, 03-*Low vegetation*, 04-*Medium vegetation*, 05-*High vegetation*, 06-*Buildings*, 07-*Low noise*, 09-*Water*, 11-*Road surface*, 17-*Bridge deck* and 18-*High noise*.

In this work we will focus only on the PC points classified as 09-*Water* (simply called water points).

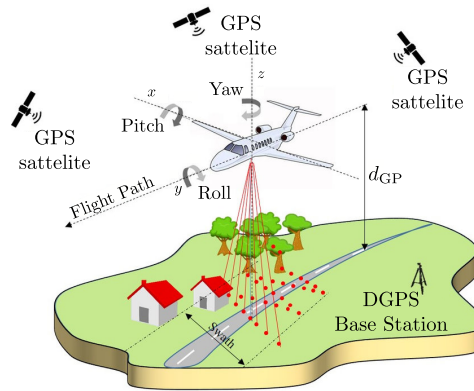


Figure 2.3: Airborne laser scanning. Source of the picture: [6].

Individual PC points with the corresponding properties are saved in LAS (LASer) file format, which is designed to store LiDAR PC data. In MATLAB Lidar Toolbox, the function `readPointCloud()` can read PC data from a LAS file, then returns it as a `pointCloud` object, see Figure 2.4.

Property	Value
Location	<code>1035543x3 single</code>
Count	<code>1035543</code>
XLimits	<code>[-5.7451e+05, -5.7413e+05]</code>
YLimits	<code>[-1.2812e+06, -1.2808e+06]</code>
ZLimits	<code>[134.4900, 276.4300]</code>
Color	<code>[]</code>
Normal	<code>[]</code>
Intensity	<code>1035543x1 uint8</code>

Figure 2.4: Example of the `pointCloud` object created from a loaded LAS file in MATLAB.

Under the `Location` there is an array of size  $n$  by 3 for storing the  $x, y, z$  coordinates of the individual PC points as single precision floating point values. The value of  $n$  (the total number of PC points stored in the loaded LAS file) is specified in the second row by the `Count`. Then there are also ranges for each coordinate, information about the light pulse intensity and color. Classification of each point is stored in a separate array as `uint8`. To visualize the classified PC we can use the function `pcshow()`, see Figure 2.5.

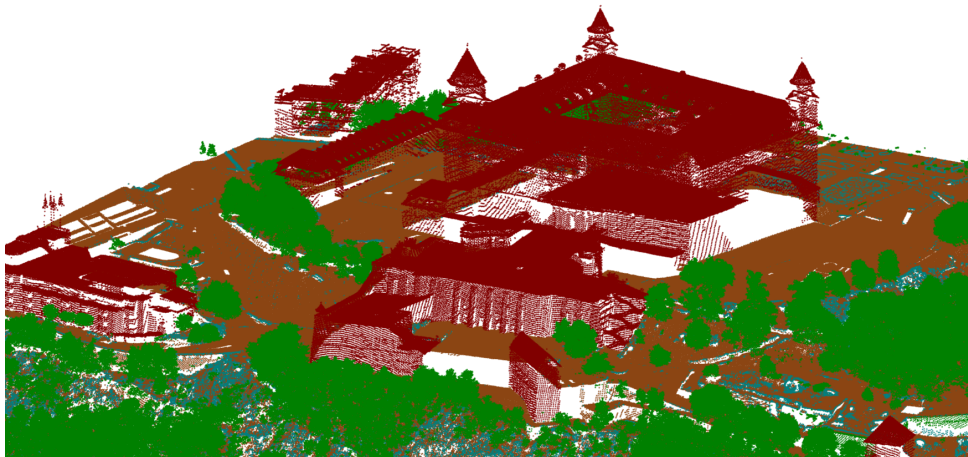


Figure 2.5: Classified PC scanned around Bratislava castle.

### 2.1.2 Digital terrain model

As described in [7], DTM is a digital representation of a specified reference terrain, which is commonly approximated by a set of discrete terrain points<sup>5</sup> arranged in a uniform rectangular grid (in a form of matrix). Each point is then prescribed with a single value of the terrain elevation<sup>6</sup> (height) at that specific place. DTMs have many applications, for instance in geodesy, civil engineering, hydrology or as an input data for topographic maps visualization.

DTM is calculated by interpolation over the PC points classified as 02-Ground. The output is a raster which according to [18] can be in ASC (Esri ASCII Grid file), TIFF (Tag Image File Format) or ESRI GRID format with high resolution, where 1 pixel = 1 m × 1 m. At the beginning of section 2.1 we also mentioned the DSM (Digital Surface Model). Very briefly, in comparison to DTM, DSM is a result of interpolation which takes into consideration also classified PC points from these additional classes: 01-Unclassified, 03-Low vegetation, 04-Medium vegetation, 05-High vegetation, 06-Buildings, 09-Water and 17-Bridge deck. It will therefore include heights of objects such as buildings or various vegetation. Differences between DTM and DSM can be seen in Figure 2.6. For the purposes of this work we only used DTM exported in TIFF format.

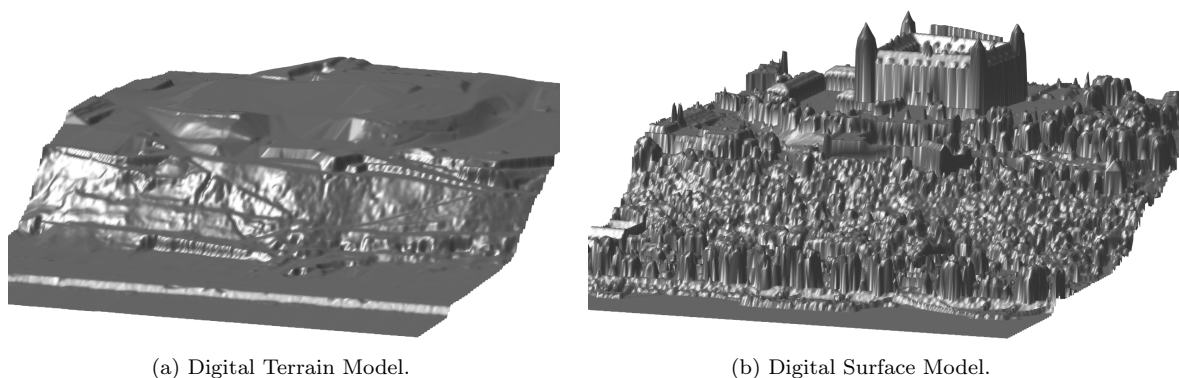


Figure 2.6: Comparison of DTM and DSM around Bratislava castle.

To load DTM into MATLAB, we can use the function `readgeoraster()` from the Mapping Toolbox. It returns two objects, first being the `heightMatrix` which is a 2-dimensional array containing the terrain elevation values. Second one is `rasterReference` (see Figure 2.7a) which is an object of type `MapCellsReference` and contains information such as the raster size, limits for  $x$  and  $y$  coordinates, type of projection, etc. To obtain additional information regarding the DTM we can use the function `georasterinfo()` which returns an object of type `RasterInfo` (see Figure 2.7b). This one is focused more on the details about the specified TIFF file.

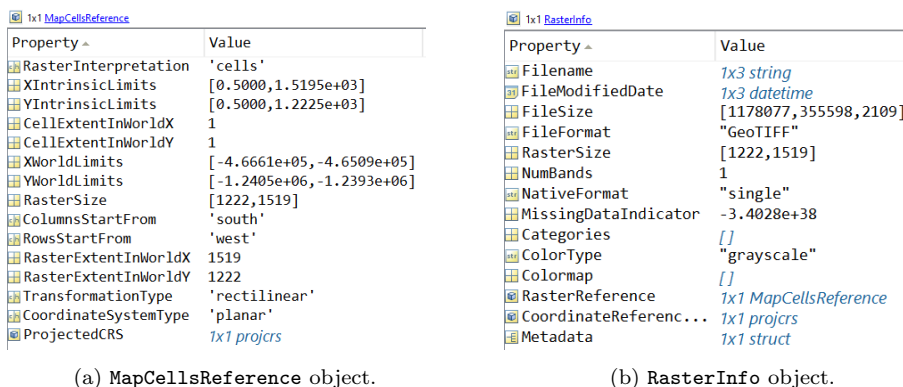


Figure 2.7: Objects containing information about DTM. Part of this specific TIFF file is shown in Figure 2.8.

If we look closely at the original DTM in Figure 2.8a, a certain raggedness is present which can be regarded as a form of noise. We can use, for instance, the *Gaussian blur filter* to smooth out the original

<sup>5</sup>Expressed in some coordinate system, in this case the S-JTSK(JTSK03) - Krovak East North.

<sup>6</sup>With respect to some vertical reference system, in this case the Bpv.

DTM. In MATLAB one can simply use the function `imgaussfilt()` from the Image Processing Toolbox. It filters an image using a 2-dimensional Gaussian smoothing kernel  $G_\sigma$  with specified standard deviation  $\sigma$ , the default value of standard deviation is set as  $\sigma = 0.5$ . In Figure 2.8 there is a comparison of the original DTM with several filtered DTMs with various parameters  $\sigma$ . However, setting the parameter  $\sigma$  too high may result in severe loss of DTM quality. Meaning, the waterbed, which is visible in Figures 2.8a, 2.8b and 2.8c, will be lost. This can be seen in Figure 2.8d where the waterbed has almost completely disappeared.

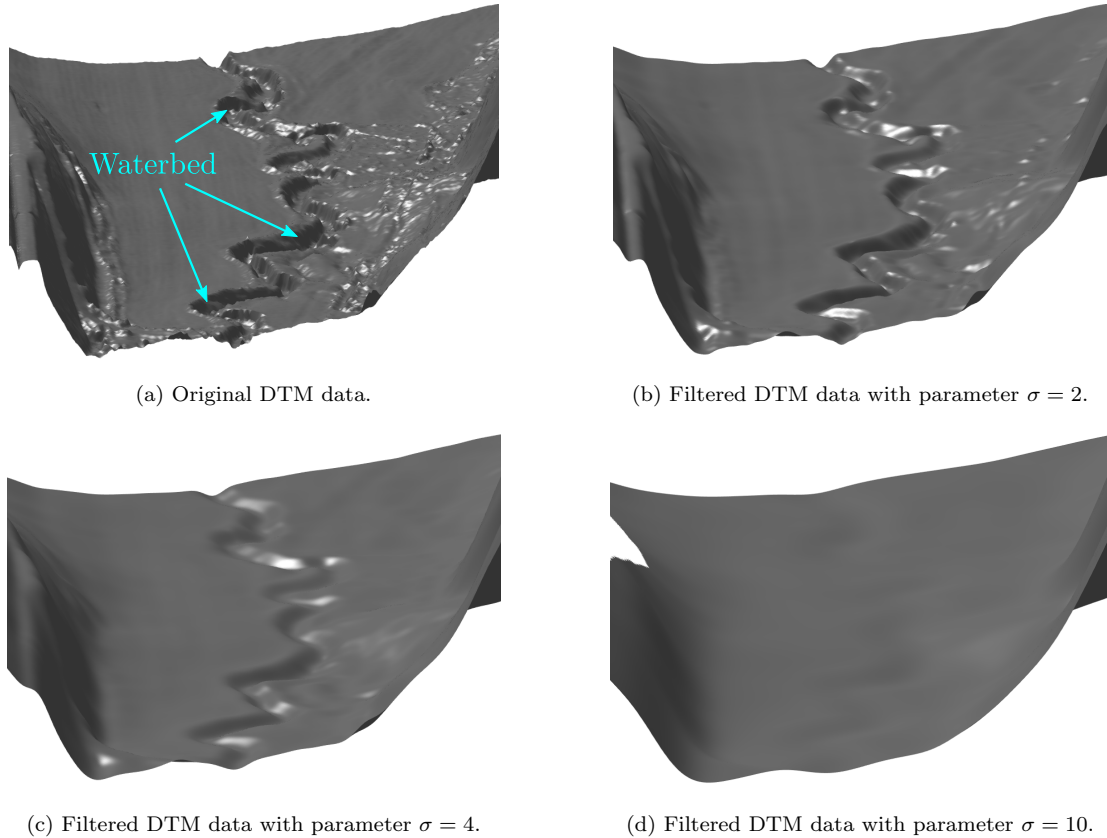


Figure 2.8: Comparison of the original DTM and filtered DTMs around stream Starý potok.

## 2.2 OpenStreetMap data

OpenStreetMap (OSM) is an open-content license database of maps which is largely developed by its users who do it voluntarily. It originates in Great Britain in 2004 when the raw map data were locked away at private companies and accessible only at great cost. Many people wanted some open-license map database so much that they were willing to go out with a GPS tracker and build a brand new database all by themselves. Since then the OSM community has grown world-wide, becoming very popular.

It is accessible via their website [www.openstreetmap.org](http://www.openstreetmap.org) where it can be edited as well. Another way to edit is to use a free editor called JOSM (Java Open Street Map). There we can download the data by selecting an area we are interested in, then start editing. As we can see in Figure 2.10, watercourses are represented in maps by a piece-wise linear curve defined as a sequence of its nodal points using the GPS coordinates.

Field	Value
node	1x1 struct
way	1x1 struct
bounds	[18.4997, 18.5148; 48.5963, 48.6031]
Attributes	1x1 struct

Figure 2.9: Example of `parsed_osm` structure in MATLAB.

To export these data we first select the chosen watercourse and move it to a new layer. Next we save it as an OSM file which can be then loaded into MATLAB using the function `parse_openstreetmap()` from OpenStreetMap Functions package<sup>7</sup>. This returns a structure `parsed_osm` (see Figure 2.9) where under the structure `node` we can find the coordinates of the watercourse's nodal points. These are expressed as  $(\lambda, \phi, h)$ , meaning, as longitude  $\lambda$ , latitude  $\phi$  and ellipsoidal height  $h$  in spatial reference system WGS84<sup>8</sup>. The value of the ellipsoidal height  $h$  will be in this case equal to zero, because OSM database does not store this information. Therefore, under the `node` structure, there are only the longitude  $\lambda$  and latitude  $\phi$  values saved in a  $2$  by  $n$  array where  $n$  is the number of the nodal points.

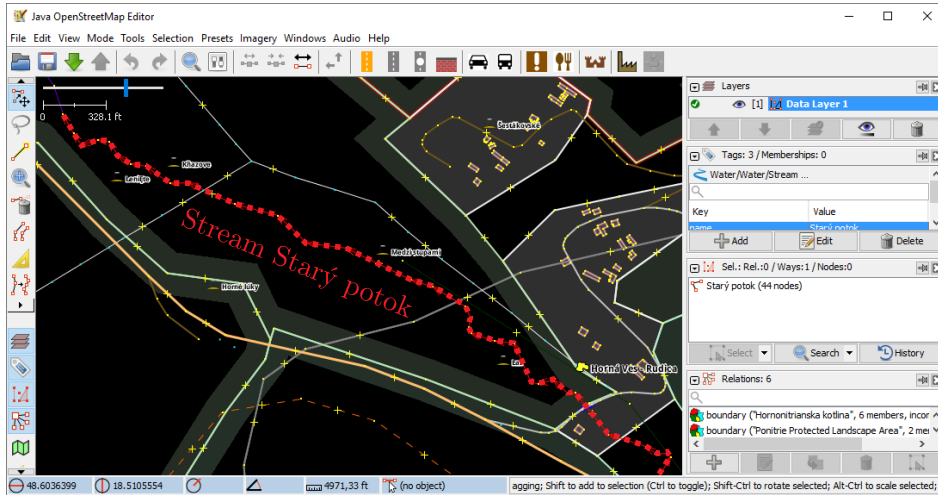


Figure 2.10: Java Open Street Map editor. Downloaded data show the area around the stream Starý potok.

Unfortunately, we have the watercourse's nodal points expressed in a different coordinate system in comparison to the terrain data. Therefore, we must first transform the data into one common coordinate system. The best choice will be to only transform the watercourse's nodal points from WGS84 to S-JTSK(JTSK03) - Krovak East North. This is further described in section 2.3.

## 2.3 Coordinate system transformation

To be able to express location of objects/places on Earth, we need to use some sort of coordinate system. As described in [16], such coordinate system is called the *Geodetic datum*, which is intended to geometrically represent Earth by approximating it by an ellipsoid. This is done by fitting the ellipsoid to the Earth by minimizing the differences between the ellipsoid itself and the geoid<sup>9</sup>. There are two types, *global* and *local* geodetic datums, depending on whether we want to describe the whole Earth with relatively good accuracy or some specific region with much greater accuracy. One of the most well-known global geodetic datums currently used is the previously mentioned WGS84 geodetic datum, which uses the ellipsoid with the same name. An example of a local geodetic datum is the S-JTSK(JTSK03) with Bessel 1841 ellipsoid.

For some chosen geodetic datum we can express the location of a point  $P$  in several different coordinates (see [16]). In this work we will focus on these three:

- (CS1) **Geodetic coordinates:** Coordinates of a point  $P$  are expressed in terms of longitude  $\lambda$ , latitude  $\phi$  and ellipsoidal height  $h$  as  $P = (\lambda, \phi, h)$ , similar to using spherical coordinates.
- (CS2) **Geocentric Cartesian coordinates:** Also known as ECEF (Earth-Centered, Earth-Fixed) coordinate system. It uses a fixed Cartesian coordinate system with its origin at the center of the Earth (see Figure 2.11). Coordinates of a point  $P$  are simply expressed by  $X, Y$  and  $Z$  values as  $P = (P_X, P_Y, P_Z)$ .
- (CS3) **Grid coordinates:** Coordinates of a point  $P$  are expressed as  $P = (P_x, P_y)$  in a projected coordinate reference system.

<sup>7</sup>Available from <https://www.mathworks.com/matlabcentral/fileexchange/35819-openstreetmap-functions>.

<sup>8</sup>World Geodetic System 1984.

<sup>9</sup>Geoid is the mathematical model of Earth based on global mean sea level used for precise measurements of elevation.

For more details about converting coordinates see [16] and [4]. Now, the change of coordinates from WGS84 ( $\lambda, \phi, h$ ) to S-JTSK(JTSK03) - Krovak East North ( $x, y$ ) will consist of five transformations in total. First, we convert coordinates from geodetic (CS1) to geocentric (CS2) coordinates within the WGS84 datum. Then we use transformations (T2) and (T3) to switch coordinates to the S-JTSK(JTSK03) datum. Then we convert coordinates from geocentric (CS2) back to geodetic (CS1) coordinates and subsequently project them to the grid coordinates S-JTSK(JTSK03) - Krovak East North. This whole process is visualized in Figure 2.11 and described in the following five steps:

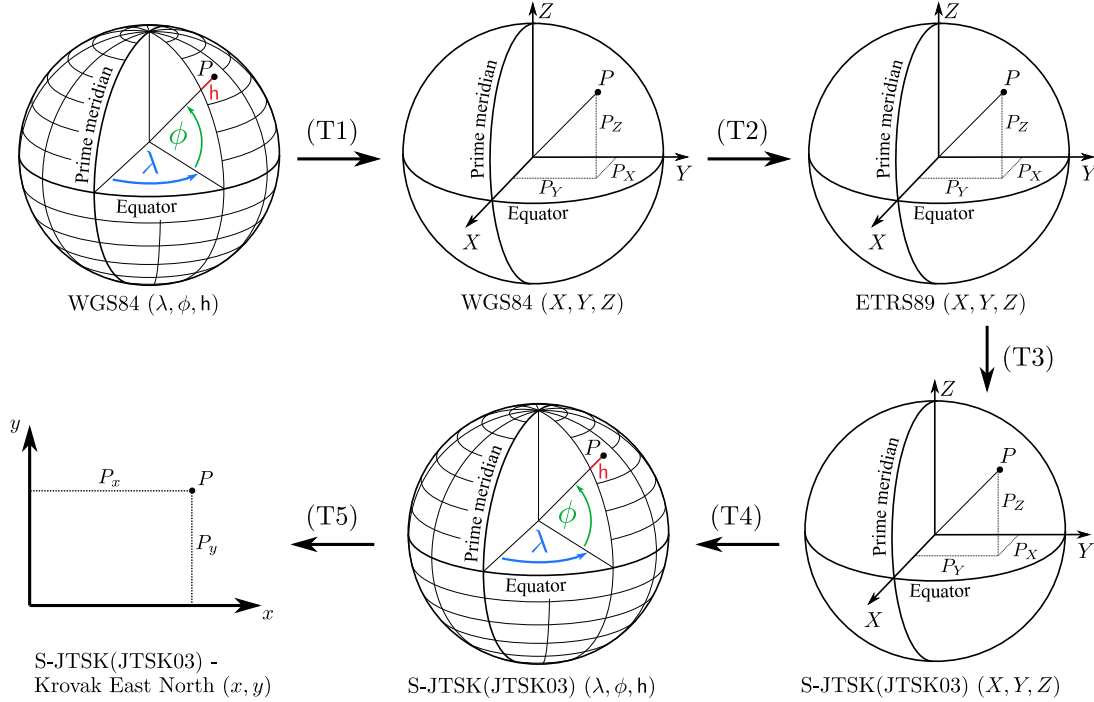


Figure 2.11: Transformation from WGS84 ( $\lambda, \phi, h$ ) to S-JTSK(JTSK03) - Krovak East North ( $x, y$ ).

- (T1) **Conversion from WGS84 ( $\lambda, \phi, h$ ) to WGS84 ( $X, Y, Z$ )**, meaning: from geodetic (CS1) to geocentric Cartesian coordinates (CS2). This is fairly straight forward, basically as if we were to convert spherical coordinates into Cartesian. However, slightly modified equations are used, see section 1.2.2 *Geocentric Cartesian coordinates* in [16]. In MATLAB, this transformation can be performed by the function `geodetic2ecef()` from the Mapping Toolbox. As its input arguments, the user must provide the `referenceEllipsoid` object<sup>10</sup> together with coordinates in geodetic format ( $\lambda, \phi, h$ ). This function then returns the transformed coordinates in ( $X, Y, Z$ ) format.
- (T2) **Datum transformation from WGS84 ( $X, Y, Z$ ) to ETRS89 ( $X, Y, Z$ ) (EPSG: 9225)**. Here we used the *7-parameter Helmert transformation* (or simply Helmert transformation). As described in [16], it consists of 3 translations:  $\Delta X$ ,  $\Delta Y$  and  $\Delta Z$ , 1 scaling:  $\Delta S$  and 3 rotations:  $R_X$ ,  $R_Y$  and  $R_Z$  (angles of rotation around each axis). Let us denote  $(X_A, Y_A, Z_A)$  the coordinates in some geodetic datum  $A$  and the  $3 \times 3$  rotation matrix  $\mathbf{R}$  created from the 3 rotations mentioned above (for more details about the rotation matrix  $\mathbf{R}$ , see [16]). Then the Helmert transformation can be expressed by the following equation

$$\begin{bmatrix} X_B \\ Y_B \\ Z_B \end{bmatrix} = \begin{bmatrix} \Delta X \\ \Delta Y \\ \Delta Z \end{bmatrix} + (1 + \Delta S) \mathbf{R} \begin{bmatrix} X_A \\ Y_A \\ Z_A \end{bmatrix}.$$

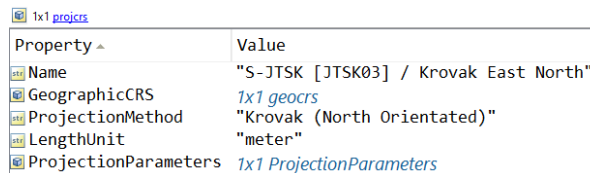
In this case, the geodetic datum  $A$  is WGS84 and datum  $B$  is ETRS89. Each transformation (or coordinate system, ellipsoid, etc.) is specified by its unique EPSG code in the EPSG Geodetic Parameter Dataset<sup>11</sup>. For example, this transformation (T2) is specified by the code 9225. With

<sup>10</sup>This object contains details about the ellipsoid, such as lengths of semi-major and semi-minor axes, eccentricity, flattening, mean radius, etc.

<sup>11</sup>Can be downloaded from <https://epsg.org/home.html>. Transformation parameters (or details about coordinate system, ellipsoid, etc.) can be looked up on the said website by entering the EPSG code into the search bar.

this code we can import the datum transformation parameters from the EPSG Dataset as a vector  $\vec{p} = (\Delta X, \Delta Y, \Delta Z, R_X, R_Y, R_Z, \Delta S)$  into MATLAB. To perform the Helmert transformation we used the function `d3trafo()` from the Geodetic Transformations package<sup>12</sup>. Naturally, as input arguments we give it the vector of parameters  $\vec{p}$  together with the coordinates which we want to transform. Last important step is to explain why we take a "little detour into ETRS89" and do not convert coordinates directly from WGS84 to S-JTSK(JTSK03). The reason is that currently the EPSG Dataset does not contain parameters for such direct datum transformation. Therefore, instead of just one, we had to use two subsequent different datum transformations (T2) and (T3).

- (T3) **Datum transformation from ETRS89 ( $X, Y, Z$ ) to S-JTSK(JTSK03) ( $X, Y, Z$ ) (EPSG: 8365)** is done the same way as transformation (T2), but different parameters are used.
- (T4) **Conversion from S-JTSK(JTSK03) ( $X, Y, Z$ ) to S-JTSK(JTSK03) ( $\lambda, \phi, h$ )**, meaning: from geocentric Cartesian (CS2) to geodetic coordinates (CS1). As stated in [16], backwards transformation (T4) is not as direct as the transformation (T1). Problematic is the calculation of the latitude  $\phi$ , that is, expressing it explicitly. Therefore, an iterative method is used to compute the approximate value of latitude  $\phi$  with some given precision, as described in detail in [16]. For this transformation we used the function `ecf2geodetic()` from MATLAB Mapping Toolbox. As input arguments we give it the coordinates in ( $X, Y, Z$ ) format and the `referenceEllipsoid` object for Bessel 1841 ellipsoid.
- (T5) **Forward projection from S-JTSK(JTSK03) ( $\lambda, \phi, h$ ) to S-JTSK(JTSK03) - Krovak East North ( $x, y$ )**, meaning: from geodetic (CS1) to the grid coordinates (CS3), using the *Krovak projection*<sup>13</sup>. For further details about this projection, see [4]. Yet again, in MATLAB this forward projection can be performed by the function `projfwd()`. As input arguments we provide it with values of the longitude  $\lambda$ , latitude  $\phi$  and the `projcrs` type object by which we specify the projection by its EPSG code 8353 (`projcrs` - PROJected Coordinate Reference System, see Figure 2.12). Subsequently, as the output of the function `projfwd()` we receive the coordinates of the watercourse's nodal points ( $x, y$ ) in the projected coordinate reference system S-JTSK(JTSK03) - Krovak East North.



Property	Value
Name	"S-JTSK [JTSK03] / Krovak East North"
GeographicCRS	1x1 <i>geocrs</i>
ProjectionMethod	"Krovak (North Orientated)"
LengthUnit	"meter"
ProjectionParameters	1x1 <i>ProjectionParameters</i>

Figure 2.12: Example of `projcrs` object containing information about Krovak (North oriented) projection.

After transforming all the watercourse's nodal points by the transformations (T1) - (T5) to the projected coordinate system S-JTSK(JTSK03) - Krovak East North, we have all data in one common coordinate system. Now we can visualize them in one image, see the Figure 2.13 bellow.

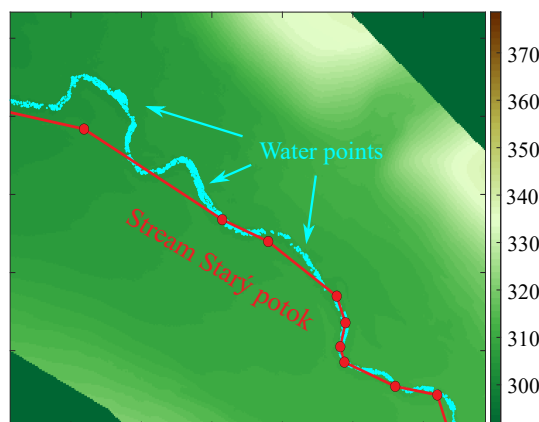


Figure 2.13: DTM from Figure 2.8 with the corresponding water points (from classified PC) and watercourse's nodal points (red points). In this Figure we can see part of stream Starý potok.

<sup>12</sup>Available from <https://www.mathworks.com/matlabcentral/fileexchange/9696-geodetic-transformations>.

<sup>13</sup>An oblique variant of the Lambert Conformal Conic projection.

# Chapter 3

## Mathematical model

### 3.1 Evolving parametric curve

First of all, we should choose a proper watercourse representation to describe it in the language of mathematics and then form a mathematical model with this representation at its core. Since a watercourse resembles a curved line, we will represent it by a *parametric curve*  $\gamma_0: U \rightarrow \mathbb{R}^2$ . As described in our workflow in section 4.1, we obtained  $\gamma_0$  from the *OpenStreetMap database* (see section 2.2) and we will use  $\gamma_0$  as the initial condition for the curve evolution.

Therefore, as the centerpiece will be considered the *evolving parametric curve*  $\gamma$  (see Figure 3.1), formally defined as a map

$$\gamma: U \times T \rightarrow \mathbb{R}^2: (u, t) \mapsto \gamma(u, t),$$

that for each value of parameter  $u \in U = [0, 1] \subset \mathbb{R}$  at time  $t \in T = [0, t_{\text{end}}]$  assigns a point in  $\mathbb{R}^2$  (or generally in some  $n$ -dimensional space  $\mathbb{R}^n$ ). In other words, for each fixed time  $t \in T$  there exists some static planar curve  $\gamma^t: U \rightarrow \mathbb{R}^2$ , where  $\gamma^t(u) \equiv \gamma(u, t)$ .

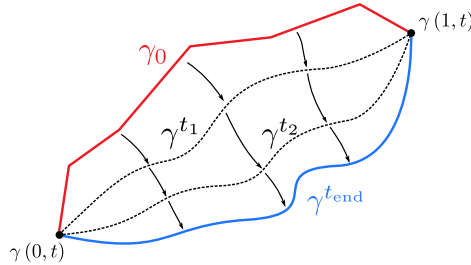


Figure 3.1: Evolving curve  $\gamma$  with fixed endpoints.

Furthermore, we will introduce a pair of vectors which are going to play an important role later. Using the tangent vector  $\partial_u \gamma(u, t)$  we first define the *unit tangent vector* as

$$T(u, t) = \frac{\partial_u \gamma(u, t)}{\|\partial_u \gamma(u, t)\|}$$

where  $\partial_u \equiv \frac{\partial}{\partial u}$  is a shortened notation of the partial derivative with respect to  $u$  and  $\|\cdot\|$  is the Euclidean norm. The second one  $N^+(u, t)$  will be called the *positively oriented unit normal vector*, which is simply  $T(u, t)$  rotated 90 degrees anti-clockwise (see Figure 3.2). With these tools at our disposal we can now focus our attention on the next task.

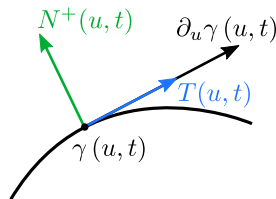


Figure 3.2: Vectors  $\partial_u \gamma$ ,  $T$  and  $N^+$ .



## 3.2 Formulation of the mathematical model

The core idea for the watercourse correction is to use curve evolution to compute more accurate watercourse shape using airborne laser scanning data (described in section 2.1). Therefore, we would like to describe how the curve  $\gamma$  should change shape in time. Mathematically speaking, this change will be naturally expressed as the partial derivative of  $\gamma(u, t)$  with respect to  $t$  of which the value should be somehow reasonably prescribed.

Let us prescribe for each point  $\gamma(u, t)$  a velocity vector  $\vec{v}(u, t)$  which will describe the direction in which it should move. We can express it by the following PDE [11, 13]

$$\partial_t \gamma(u, t) = \vec{v}(u, t) \quad (3.1)$$

where  $u \in ]0, 1[$ ,  $t \in ]0, t_{end}]$  and which is subject to the initial condition  $\gamma(u, 0) = \gamma_0(u)$ , together with the Dirichlet boundary condition  $\gamma(0, t) = G_0$ ,  $\gamma(1, t) = G_1$  where  $G_0, G_1 \in \mathbb{R}^2$ , meaning that endpoints of the curve  $\gamma$  will be fixed during the evolution.

Now we will focus our attention a bit more on the velocity vector  $\vec{v}(u, t)$  itself. It is a common practice to decompose a vector into two orthogonal components. First one being the component of  $\vec{v}(u, t)$  in the direction of  $T(u, t)$  - *Tangential velocity vector*  $\vec{v}_T(u, t)$  and the second in the direction of  $N^+(u, t)$  - *Normal velocity vector*  $\vec{v}_N(u, t)$ . We can express both of these as projections of  $\vec{v}(u, t)$  onto  $T(u, t)$  and  $N^+(u, t)$

$$\begin{aligned} \vec{v}_T(u, t) &= \underbrace{(\vec{v} \cdot T)}_{\alpha(u, t)} T(u, t) = \alpha(u, t) T(u, t), \\ \vec{v}_N(u, t) &= \underbrace{(\vec{v} \cdot N^+)}_{\beta(u, t)} N^+(u, t) = \beta(u, t) N^+(u, t), \end{aligned}$$

where  $\vec{v} \cdot T$  denotes the standard inner product of two vectors in two-dimensional Euclidean space. We have also introduced two new functions:

$$\alpha = \vec{v} \cdot T, \quad (3.2)$$

$$\beta = \vec{v} \cdot N^+, \quad (3.3)$$

where equations (3.2) and (3.3) define the *Tangential speed*<sup>1</sup>  $\alpha(u, t)$  and the *Normal speed*  $\beta(u, t)$ , respectively.

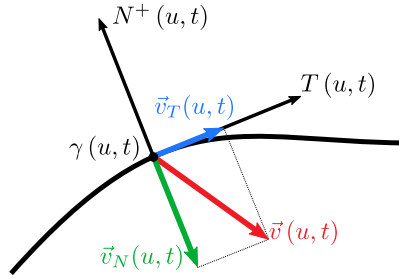


Figure 3.3: Decomposition of a velocity vector  $\vec{v}(u, t)$ .

**Remark 1.** We may think of this decomposition as a way of representing  $\vec{v}(u, t)$  in some other, let us call it a "local" coordinate system at the point  $\gamma(u, t)$  as its "local" origin with the orthonormal basis  $\{T(u, t), N^+(u, t)\}$ . In addition to that, we may regard  $\alpha(u, t)$  and  $\beta(u, t)$  as "coordinates" of  $\vec{v}(u, t)$  in that "local" coordinate system.

Thanks to the decomposition of  $\vec{v}(u, t)$  we managed to express it in terms of  $T(u, t)$  and  $N^+(u, t)$  as a linear combination

$$\vec{v}(u, t) = \alpha(u, t) T(u, t) + \beta(u, t) N^+(u, t)$$

and thus we can subsequently rewrite the original equation (3.1) into a new form [11, 20]

$$\partial_t \gamma(u, t) = \alpha(u, t) T(u, t) + \beta(u, t) N^+(u, t). \quad (3.4)$$

In the following subsections we are going to discuss the specifics of tangential and normal speed. For instance, their influence on  $\gamma$  during the evolution as well as how both should be prescribed.

<sup>1</sup>In English there is a difference between saying *velocity* and *speed*. The term *velocity* refers to a vector quantity - it has both magnitude and direction, whereas *speed* is just scalar quantity - it only has magnitude.

### 3.3 Choice of the normal speed

As the main evolution guiding component, we will use the normal speed  $\beta$ , because it causes the change of the shape of  $\gamma$  during the evolution (see [10]). What we seek to accomplish can be easily seen in Figure 3.4. Meaning, to attract  $\gamma_0$  (plotted in red color) using evolution into the waterbed where the actual watercourse flows.

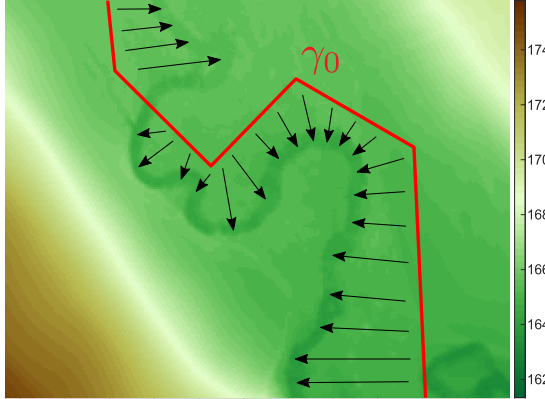


Figure 3.4: Part of stream Parná plotted with the corresponding terrain data.

Therefore, the basic approach we propose is to use *advection* by a properly chosen velocity vector field

$$\vec{V}: \Omega \rightarrow \Omega: \vec{r} \mapsto \vec{V}(\vec{r}),$$

where  $\Omega \subset \mathbb{R}^2$  denotes our computational domain and  $\vec{r} = (x, y)$  is the position vector of a point in  $\Omega$ . The idea behind a vector field, is that to each point  $\vec{r} = (x, y)$  in space it assigns a vector  $\vec{V}(\vec{r})$ . As was stated before when introducing (3.1), this is exactly what we need. So to each curve point  $\gamma(u, t)$  we prescribe a vector  $\vec{V}(\gamma(u, t))$  which will be then projected onto  $N^+(u, t)$  to compute the normal speed

$$\beta(u, t) = \vec{V}(\gamma(u, t)) \cdot N^+(u, t). \quad (3.5)$$

Next step is to construct a suitable vector field  $\vec{V}$  to compute the normal speed  $\beta$ . We will use two gradient vector fields  $\nabla h$  and  $\nabla d$  computed from two functions based on the data for our model:

- The *terrain elevation function*  $h(x, y)$  (or simply terrain function).
- The *distance function*  $d(x, y)$  to the water points.

First, we are going to specify the vector field  $\vec{V}$  in the equation (3.5) using  $\nabla h$  and  $\nabla d$  separately. Then, a weighted combination using both gradient vector fields will be considered, together with a curvature regularization as well.

#### 3.3.1 Terrain function

In general, the terrain function  $h$  will be defined as a map

$$h: \Omega \rightarrow \mathbb{R}: (x, y) \mapsto h(x, y)$$

which assigns the terrain elevation  $h(x, y)$  to each point  $(x, y)$  in our computational region  $\Omega$ . The value of  $h(x, y)$  will be determined by the DTM data. However, the raw terrain data  $h_0$  usually contain some undesired distortions<sup>2</sup> which we would like to eliminate. This can be described by the following steps:

(H1) By applying the Gaussian filter  $G_\sigma$  to the raw terrain data  $h_0$ , we compute the filtered terrain function  $\tilde{h}$  (see Figure 2.8). Formally it can be expressed as  $\tilde{h} = G_\sigma * h_0$ , where  $*$  denotes the convolution.

<sup>2</sup>For instance, the raggedness of DTM as described in section 2.1.2 or a strong downhill trend, which is described in the problem (PH3).

- (H2) We subtract the trend<sup>3</sup> (if necessary) from the filtered terrain  $\tilde{h}$  to get the final version of the terrain function, denoted simply by  $h$ .

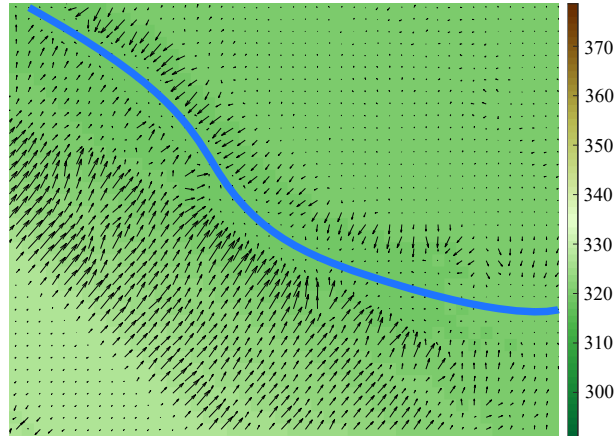


Figure 3.5: Negative gradient of the terrain  $-\nabla h$  with the highlighted waterbed of stream Starý potok.

Now, the idea behind using the negative terrain gradient  $-\nabla h$  is visualized in Figure 3.5. Intuitively, we can see that the negative terrain gradient  $-\nabla h$  is largely pointing in the direction to the waterbed marked by blue curve in Figure 3.5. We can take advantage of this property to prescribe the vector field as  $\vec{V} = -\nabla h$ , thus expressing the normal speed (3.5) in the following manner

$$\beta(u, t) = -\nabla h(\gamma(u, t)) \cdot N^+(u, t). \quad (3.6)$$

However, relying solely on the negative terrain gradient  $-\nabla h$  may not be sufficient enough to attract  $\gamma_0$  into the waterbed. The possible problems include for example:

- (PH1) In some regions around the watercourse the terrain may not be inclined enough. Thus the negative terrain gradient  $-\nabla h$  will be too small. If  $\gamma$  were to encounter such regions, the resulting normal speed  $\beta$  would not be powerful enough to attract it into the waterbed. This can be seen, for example, in the upper-right part of Figure 3.5 where the individual gradient vectors are barely visible at all.
- (PH2) An inconveniently inclined terrain might drive  $\gamma$  away from the waterbed. As it is visualized in Figure 3.6, there is an artificial barrier (embankment) along the watercourse to prevent flooding. In this case, such terrain obstacles would not permit  $\gamma$  to access the waterbed. Furthermore, if  $\gamma$  were to encounter some terrain depressions<sup>4</sup> (highlighted in red in Figure 3.6), it would end up trapped inside.

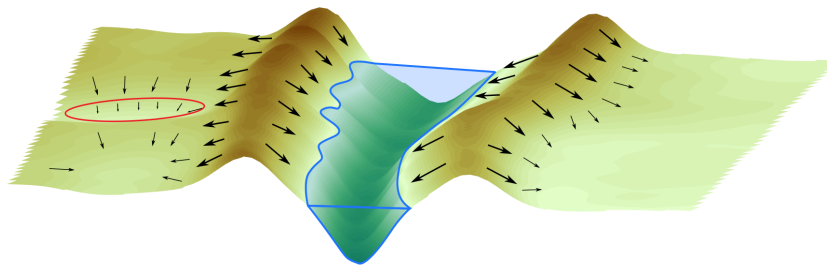


Figure 3.6: Example of the inconvenient terrain along stream Parná.

- (PH3) In mountainous regions, some watercourses may flow steeply downhill due to large elevation differences. The negative terrain gradient may then be influenced by the terrain's trend. Meaning, it would just "flush away"  $\gamma$  downhill, not necessarily in the direction to the waterbed.

To solve the problems (PH1) and (PH2), a different gradient vector field might be used which will be computed from the distance function  $d$  (see section 3.3.2). The solution for the problem (PH3) is described in the following section.

<sup>3</sup>This method will be described later.

<sup>4</sup>A low-point (hole) in the ground with higher ground surrounding it.

### Rotation of the fitting plane

To tackle the problem (PH3), we can express the downhill trend in the direction of the water flow by a fitting plane

$$f(x, y) = p_{00} + p_{10}x + p_{01}y \quad (3.7)$$

where  $p_{00}$ ,  $p_{10}$  and  $p_{01}$  are parameters computed from the filtered terrain  $\tilde{h}$  using linear regression. However, using the fitting plane  $f$  right away may not lead to the desired results, because sometimes the fitting plane  $f$  may be skewed due to the uneven terrain elevation (see the blue fitting plane in Figure 3.7). To solve this, we are going to rotate the fitting plane  $f$  to compensate this misalignment. This will give us a new fitting plane  $\bar{f}$  (see the red fitting plane in Figure 3.7) which describes the desired trend more properly.

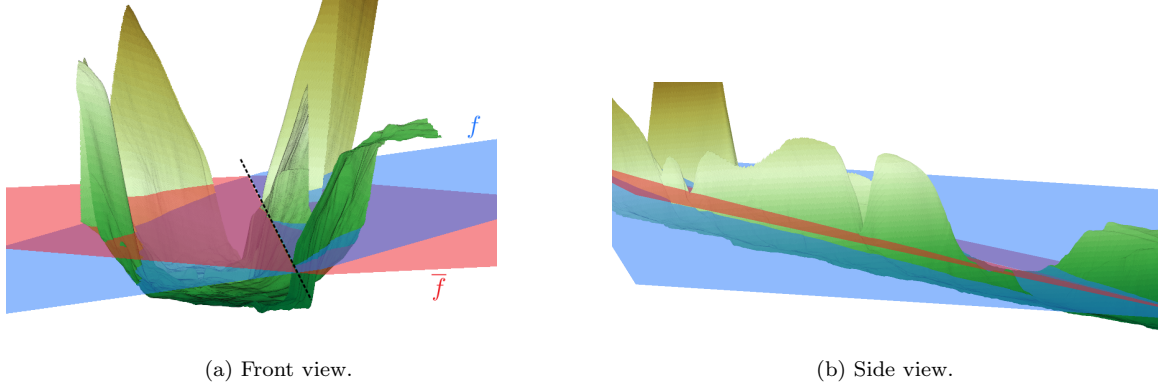


Figure 3.7: Fitting planes  $f$  (blue) and  $\bar{f}$  (red) computed from the DTM around stream Starý potok. In the side view 3.7b, the fitting plane  $\bar{f}$  is barely visible because it is no longer skewed by the uneven terrain. Whereas the fitting plane  $f$  is shifted upwards on the right.

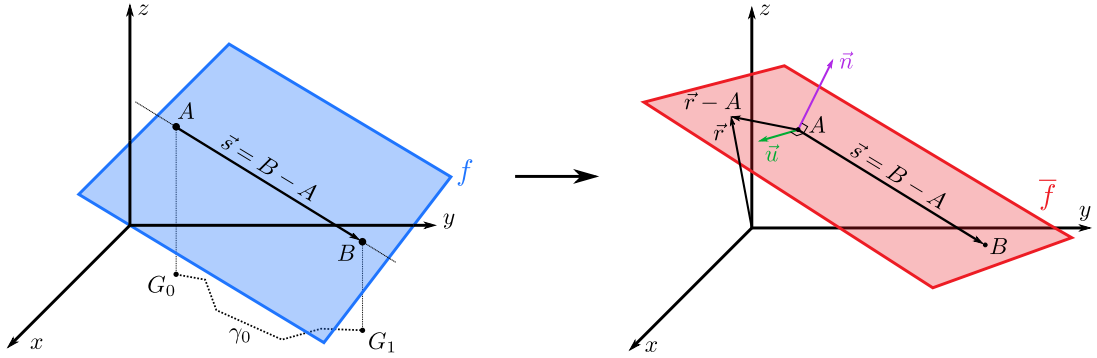


Figure 3.8: Rotation of the fitting plane  $f$  into  $\bar{f}$ .

First, to specify the line around which the rotation will be done, we will use the first and the last point of  $\gamma_0$ , denoted by  $G_0$  and  $G_1$ , respectively. Then we lift them onto the fitting plane  $f$  to get points

$$\begin{aligned} A &= (G_0, f(G_0)), \\ B &= (G_1, f(G_1)). \end{aligned}$$

With these points we can compute the vector  $\vec{s} = B - A = (s_x, s_y, s_z)$ . As the next step, we express the normal vector  $\vec{n}$  of the rotated fitting plane  $\bar{f}$  as the cross product of vectors  $\vec{u}$  and  $\vec{s}$ . The vector  $\vec{u}$  will be prescribed as  $\vec{u} = (s_y, -s_x, 0)$  and thus we can express the normal vector  $\vec{n}$  of the rotated fitting plane  $\bar{f}$  as

$$\vec{n} = \vec{u} \times \vec{s} = (-s_x s_z, -s_y s_z, s_x^2 + s_y^2) = (n_x, n_y, n_z).$$

To express the rotated fitting plane  $\bar{f}$  we will use the fact, that the following equation

$$(\vec{r} - A) \cdot \vec{n} = 0 \quad (3.8)$$

holds for each point  $\vec{r} = (x, y, z) \in \bar{f}$ . After some algebraic operations applied to the equation (3.8) we have

$$\begin{aligned} \vec{r} \cdot \vec{n} &= A \cdot \vec{n}, \\ x n_x + y n_y + z n_z &= A \cdot \vec{n}, \\ z &= \underbrace{\frac{A \cdot \vec{n}}{n_z}}_{\bar{p}_{00}} - \underbrace{\frac{n_x}{n_z}}_{\bar{p}_{10}} x - \underbrace{\frac{n_y}{n_z}}_{\bar{p}_{01}} y. \end{aligned}$$

We will now express the desired rotated fitting plane  $\bar{f}$  analogously to the fitting plane (3.7) as

$$\bar{f}(x, y) = \bar{p}_{00} + \bar{p}_{10}x + \bar{p}_{01}y. \quad (3.9)$$

Subsequently, we can formally define the final terrain function from the step (H2) as

$$h(x, y) = \tilde{h}(x, y) - \bar{f}(x, y). \quad (3.10)$$

Alternatively, in case the terrain is not steeply inclined, we define the final terrain function simply as

$$h(x, y) = \tilde{h}(x, y). \quad (3.11)$$

### 3.3.2 Distance function

As we mentioned in section 2.1.1, the Classified Point Cloud (PC) may contains point classified as water, which represent the water surface. If these water points are available, it is reasonable to make use of them. We will use them to compute the so called distance function  $d$ . We define it as a map

$$d: \Omega \rightarrow \mathbb{R}: \vec{r} \mapsto d(\vec{r})$$

where  $d(\vec{r}) = \min_{P \in \Gamma} \|\vec{r} - P\|$  represents the distance of each point  $\vec{r} = (x, y) \in \Omega$ , from the nearest point  $P \in \Gamma$ . Here, the "boundary"  $\Gamma$  represents the set of water points from the classified point cloud. The distance function  $d$  can be also seen as the solution to the *Eikonal equation*

$$\|\nabla d(\vec{r})\| = 1 \quad (3.12)$$

which is subject to the boundary condition  $d|_{\Gamma} = 0$ . To compute the distance function  $d$ , we solve the Eikonal equation (3.12) numerically using the *fast-marching method* developed by James Sethian (see [17]). However, to do this we first need to discretize the boundary  $\Gamma$ . The discretized boundary  $\hat{\Gamma}$  will be represented by the set of pixels from the DTM in which the water points are located (white pixels in Figure 3.9). For these pixels from  $\hat{\Gamma}$  we set  $d = 0$ .

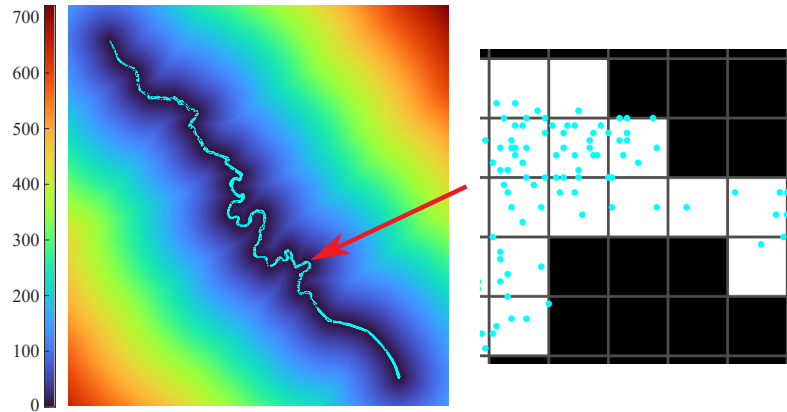


Figure 3.9: On the left side: 2D plot of distance function  $d$  around stream Parná computed using the corresponding set of water points  $\Gamma$ . On the right: zoomed in view of the discretized boundary  $\hat{\Gamma}$ .

**Remark 2.** The term *distance function* is quite self-explanatory. On the other hand, we could also think of it as a way of creating a sort of "virtual terrain" around the water points. Such "virtual valley" (see Figure 3.10) around the watercourse should resolve the first two problems (PH1) and (PH2) regarding the unsuitable real terrain.

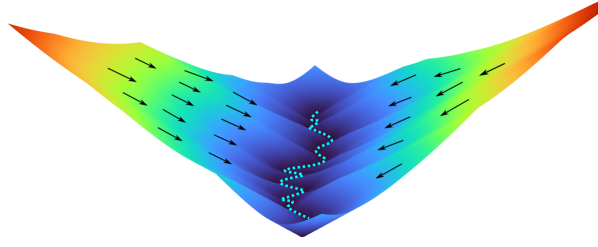


Figure 3.10: 3D plot of the distance function  $d$  from Figure 3.9.

If we set  $\vec{V} = -\nabla d$ , then we can define the normal speed  $\beta$  analogously to the definition (3.6) as

$$\beta(u, t) = -\nabla d(\gamma(u, t)) \cdot N^+(u, t). \quad (3.13)$$

However, using the negative distance function gradient  $-\nabla d$  also brings its own set of issues, for instance:

- (PD1) If the watercourse is wider, the negative distance function gradient  $-\nabla d$  can guide  $\gamma$  only to the shore-water interface.
- (PD2) As it will be discussed in section 4.3.1, using the vector field  $\vec{V} = -\nabla d$  will be problematic when  $\gamma$  encounters *meanders*<sup>5</sup>.

Therefore, to resolve as many problems as possible, we will use both gradient vector fields  $-\nabla h$  and  $-\nabla d$  to prescribe the final normal speed  $\beta$ .

### 3.3.3 Weighted combination

Since we are combining two gradient vector fields into one, we would like to control the weight each one contributes. For this we will use a weighted combination

$$\vec{V} = (1 - \theta(t))(-\nabla h) + \theta(t)(-\nabla d) \quad (3.14)$$

with the controlling parameter  $\theta(t)$ . This parameter will be defined as  $\theta: [0, t_{\text{end}}] \rightarrow [0, 1]$ . It means that  $\theta$  will be dependent on time, so that we can control the weights of the two gradient vector fields during the evolution. Depending on the value of  $\theta$ , we can split the resulting vector field  $\vec{V}$  into three cases:

$$\vec{V} = \begin{cases} -\nabla h, & \text{when } \theta = 0, \\ -(1 - \theta)\nabla h - \theta\nabla d, & \text{when } \theta \in ]0, 1[, \\ -\nabla d, & \text{when } \theta = 1. \end{cases}$$

If however, the classified PC does not contain any water points, we are restricted to  $\theta = 0$ . Meaning, we can only work with the negative terrain gradient  $-\nabla h$ .

Furthermore, it might happen that the curvature will become large in some sections of  $\gamma$  during the evolution. This will result in sharp edges when  $\gamma$  is discretized. Such behavior may cause issues for the numerical scheme. Another reason is related to the noise in DTM, for example larger rocks around the watercourse. If only the negative terrain gradient  $-\nabla h$  is available, some points would not be able to get past these obstacles (see Figure 3.11). This is similar to what we described in the problem (PH2).

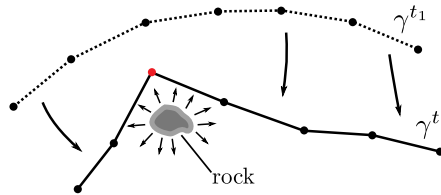


Figure 3.11: Example of noise in DTM in the form of a rock.

<sup>5</sup>Twisted parts of a watercourse, see the middle section of  $\Gamma$  in Figure 3.9.

For these reasons, to smooth out parts of  $\gamma$  with large curvature or to move the stranded points, we will use the *curvature regularization* term

$$\kappa^\pm(u, t) N^+(u, t) \quad (3.15)$$

where  $\kappa^\pm(u, t)$  denotes the *signed curvature* at point  $\gamma(u, t)$ .

**Remark 3.** If we used the signed curvature  $\kappa^\pm$  as the only guiding component for the evolution of  $\gamma$ , the equation (3.4) would be expressed as  $\partial_t \gamma(u, t) = \kappa^\pm(u, t) N^+(u, t)$ . As a result, the initial curve  $\gamma_0$  would eventually evolve into a line segment due to the smoothing effect of the signed curvature  $\kappa^\pm$ .

We can now express the final version of the normal speed  $\beta$  for our model by combining the components (3.14) and (3.15) into

$$\beta(u, t) = \delta_{\text{ext}} [-(1 - \theta(t)) \nabla h(\gamma(u, t)) - \theta(t) \nabla d(\gamma(u, t))] \cdot N^+(u, t) + \delta_\kappa \kappa^\pm(u, t). \quad (3.16)$$

Additionally, we also added two new parameters  $\delta_{\text{ext}}$  and  $\delta_\kappa$ . With the parameter  $\delta_{\text{ext}}$  we can control the overall influence of the resulting velocity vector field  $\vec{V}$ . This part of the normal speed  $\beta$  is usually also called the *external normal speed*. Similarly for  $\delta_\kappa$ , this one controls the influence of the curvature regularization. Using the normal speed  $\beta$  defined by (3.16), we express the final form of our model as

$$\partial_t \gamma(u, t) = [\delta_{\text{ext}} [-(1 - \theta(t)) \nabla h(\gamma(u, t)) - \theta(t) \nabla d(\gamma(u, t))] \cdot N^+(u, t) + \delta_\kappa \kappa^\pm(u, t)] N^+(u, t) + \alpha(u, t) T(u, t) \quad (3.17)$$

where the tangential speed  $\alpha$  will be described in the following section.

### 3.4 Choice of the tangential speed

From the analytical point of view, tangential speed  $\alpha$  is not as important since it has no effect on the shape of the curve during the evolution [14, 10]. However, when dealing with the evolution numerically, we have to discretize  $\gamma^t$  using a finite amount of points  $\gamma_i^n$ ,  $i = 1, \dots, m$  and  $n = 0, \dots, \mathbf{nMax}$ . In this case, having the tangential speed  $\alpha = 0$  may lead to the instability of the numerical scheme [20], or even crash of the computation.

To prevent this, we can use the tangential speed  $\alpha$  to our advantage, to suitably shift points  $\gamma_i^n$  along the curve  $\gamma^t$  to achieve *uniform distribution* of points. This will in return improve both quality of the mesh and precision of the numerical solution. Before we discuss it further, we first need to define the arc length function

$$s(\hat{u}) = \int_0^{\hat{u}} \|\partial_u \gamma(u, t)\| \, du$$

which measures the distance traveled between points  $\gamma(0, t)$  and  $\gamma(\hat{u}, t)$ . In addition to that, if we integrate  $\|\partial_u \gamma(u, t)\|$  over the whole interval  $U = [0, 1]$ , the result will give us the total length of the curve  $\gamma^t$

$$L[\gamma^t] = \int_0^1 \|\partial_u \gamma(u, t)\| \, du.$$

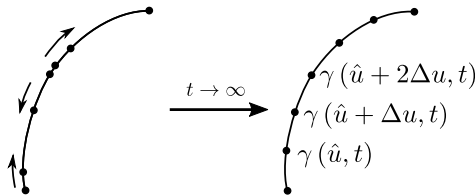


Figure 3.12: Asymptotically uniform distribution of points.

To achieve uniform distribution of points, so called asymptotically uniform redistribution will be used [14]. It means that as  $t$  approaches  $\infty$ , the points  $\gamma_i^n$  will become uniformly distributed. Analytically it can be described as

$$\lim_{t \rightarrow \infty} \frac{\int_{\hat{u}}^{\hat{u} + \Delta u} \|\partial_u \gamma(u, t)\| \, du}{L[\gamma^t]} = \mathcal{C} \quad \implies \quad \lim_{t \rightarrow \infty} \frac{\|\partial_u \gamma\|}{L} = \mathcal{C}, \quad (3.18)$$

usually choosing value of the constant as  $\mathcal{C} = \frac{1}{b-a}$ , if  $U = [a, b]$ . However, in section 3.1 we considered  $U = [0, 1]$ , therefore  $\mathcal{C} = 1$ . To bypass using the limit in (3.18), we can use a *relaxation* (see [14]) to reformulate (3.18) in the form of the differential equation

$$\partial_t \left( \frac{\|\partial_u \gamma\|}{L} \right) = \left( 1 - \frac{\|\partial_u \gamma\|}{L} \right) \omega, \quad (3.19)$$

where  $\omega \in \mathbb{R}_0^+$  is called *the relaxation parameter* and it controls the speed of the relaxation.

**Remark 4.** Intuitively, the equation (3.19) is simply a prescription of how to guide the ratio  $\frac{\|\partial_u \gamma\|}{L}$  in time to the value 1. If  $\frac{\|\partial_u \gamma\|}{L} > 1$ , then its derivative will be negative and thus decreasing its value. On the contrary, when  $\frac{\|\partial_u \gamma\|}{L} < 1$ , the derivative will come out positive, increasing the value of the ratio.

Next, we can apply the quotient rule for derivatives to the left-hand side of the equation (3.19)

$$\frac{(\partial_t \|\partial_u \gamma\|) L - \|\partial_u \gamma\| \partial_t L}{L^2} = \left( 1 - \frac{\|\partial_u \gamma\|}{L} \right) \omega \quad (3.20)$$

and to express the partial derivatives  $\partial_t \|\partial_u \gamma\|$  and  $\partial_t L$ , we can use the following relations (see [14])

$$\partial_t \|\partial_u \gamma\| = -\|\partial_u \gamma\| \kappa^\pm \beta + \partial_u \alpha, \quad (3.21)$$

$$\partial_t L = -\int_0^1 \kappa^\pm \beta \underbrace{\|\partial_u \gamma\|}_{ds} du + \underbrace{\int_0^1 \partial_u \alpha}_{=0} du = -\int_0^L \bar{\kappa}^\pm \bar{\beta} ds, \quad (3.22)$$

where we expressed  $\kappa^\pm$  and  $\beta$  in terms of the arc-length parameter  $s \in [0, L]$  instead of  $u \in [0, 1]$  as

$$\bar{\kappa}^\pm(s(u), t) = \kappa^\pm(u, t),$$

similarly for  $\bar{\beta}$  and  $\bar{\alpha}$  as well. Now we can use (3.21) and (3.22) to rewrite the equation (3.20) further into

$$\frac{(-\|\partial_u \gamma\| \bar{\kappa}^\pm \bar{\beta} + \partial_u \alpha) L + \|\partial_u \gamma\| \int_0^L \bar{\kappa}^\pm \bar{\beta} ds}{L^2} = \left( 1 - \frac{\|\partial_u \gamma\|}{L} \right) \omega.$$

Then we multiply it by  $\frac{L}{\|\partial_u \gamma\|}$  and after some algebraic operations we obtain the following PDE

$$\partial_s \bar{\alpha} = \bar{\kappa}^\pm \bar{\beta} - \frac{1}{L} \int_0^L \bar{\kappa}^\pm \bar{\beta} ds + \left( \frac{L}{\|\partial_u \gamma\|} - 1 \right) \omega, \quad (3.23)$$

where  $\partial_s \bar{\alpha} = \frac{\partial_u \alpha}{\|\partial_u \gamma\|}$ . Since both endpoints of  $\gamma$  are fixed during the evolution, the equation (3.23) will be coupled with the Dirichlet boundary conditions  $\bar{\alpha}(0, t) = \bar{\alpha}(L, t) = 0$ . As the solution, we will obtain the desired tangential speed  $\alpha$ .



# Chapter 4

## Numerical experiments

In this chapter, we are going to test our evolution model (3.17) from section 3.3.3. First, we will describe steps of our workflow in section 4.1 below. Then we will present several numerical experiments we have performed using real-world data. For the numerical experiments, we have chosen these three watercourses:

- (W1) Part of stream Osliansky potok called Starý potok located south-east from village Horná Ves.
- (W2) Part of stream Parná located between villages Košolná and Suchá nad Parnou.
- (W3) Part of Nameless stream located south-east from Zbojnická chata in Tatras.

### 4.1 Workflow description

In this section we will describe individual steps of our workflow in detail. First the process of acquiring all necessary terrain and OSM (OpenStreetMap) data, processing of all data in MATLAB and then the algorithm of the evolution itself.

- I) **Finding a watercourse that needs adjustment** using the website [www.freemap.sk](http://www.freemap.sk), because it uses the available DTM (Digital Terrain Model) as its background texture (hillshading). Therefore, the waterbed can be clearly visible and thus we can conclude, whether a watercourse is correctly plotted on the map or needs adjustment.
- II) **Download of the necessary input data.** Classified PC (Point Cloud) and DTM are downloaded via the web application Map Client ZBGIS. Using the "Shape" option in "Select Export Region with" we select the area around the chosen watercourse. Then we specify the desired file formats, specifically, LAS file format for classified PC and TIFF file format for DTM. OSM data are acquired in the OSM file format via the map editor JOSM (JavaOpenStreetMap) as described in section 2.2.
- III) **Data import and processing in MATLAB.** Terrain data are imported into MATLAB using the functions described in sections 2.1.1 and 2.1.2. Similarly, import of the watercourse's nodal points is described in section 2.2. Subsequently, these nodal points are transformed from geodetic WGS84 coordinates (CS1) to S-JTSK(JTSK03) - Krovak East North grid coordinates (CS3) via the transformations (T1) - (T5) described in section 2.3. Processing of the terrain data includes:
  - (1) Computing the filtered terrain data as described in section 2.1.2 and visualized in Figure 2.8. In MATLAB, we saved the original terrain data in the variable  $H$  ( $h_0$  in section 3.3.1) and the filtered terrain data in the variable  $H_{\text{filt}}$  ( $\tilde{h}$  in section 3.3.1). Both variables are 2-dimensional arrays.
  - (2) If necessary, computing and rotating the fitting plane as described at the end of section 3.3.1 to eliminate the potential strong downhill trend described in section 3.3.1 in problem (PH3). The terrain data after subtracting the rotated fitting plane (or simply subtracted terrain) are stored in the variable  $H_{\text{subtracted}}$ , also a 2-dimensional array.
  - (3) If classified PC data are available, computing the distance function  $d$  numerically as we described in section 3.3.2. It is also saved as a 2-dimensional array  $\text{distFun}$ .

- (4) Lastly, we compute the terrain gradient  $\nabla h$  and the distance function gradient  $\nabla d$  numerically using MATLAB function `gradient()`. As input argument we put in either of the above mentioned arrays. As output we receive two arrays `Hx` and `Hy`, where the  $x$  and  $y$  coordinates of gradients are saved.

All the necessary information regarding the terrain data will be stored in the object `DMR` which is an instance of our own `terrain` class in MATLAB. Similarly, watercourse's nodal points will be used to create the object `EC` which is also an instance of our own `evolvingCurve` class. The object `EC` will be used to store all information about the evolving curve  $\gamma$  introduced in section 3.1. We will, however, select only those nodal points which are located in such pixels where the terrain function  $h$  is defined by DTM. Otherwise we would not be able to compute the normal speed  $\beta$ , because the value of the resulting vector field  $\vec{V}$  would be computed using `NaN` values.

IV) **Computation of the new mapping** using the curve evolution, as described by these steps:

- (1) We specify the DTM data we would like to use for the evolution. This can be either the original terrain data `H`, the filtered terrain data `H_filt` or the subtracted terrain data `H_subtracted`. We also specify the parameters  $\theta$ ,  $\delta_{\text{ext}}$ ,  $\delta_{\kappa}$  for the normal speed  $\beta$  described in section 3.3.3, the relaxation parameter  $\omega$  described in section 3.4 and the size of the time step `dt`.
- (2) We compute the desired edge length `hd`, according to which we subdivide  $\gamma_0$  into  $m-1$  segments between points  $\gamma_i^{n=0}$ ,  $i = 1, \dots, m$ .
- (3) Inside a `for` loop we perform up to `nMax`<sup>1</sup> evolution steps. Each evolution step:
  - (i) We save the curve points' coordinates (or simply coordinates) from the last step  $\gamma_i^{n-1}$  and also from the second to the last step  $\gamma_i^{n-2}$ .
  - (ii) After a certain number of evolution steps we add or delete curve points using an algorithm from [1].
  - (iii) We compute curve geometry: current number of curve points `EC.Npts`, edge lengths `EC.h` and the curve length `EC.L`, then for each curve point  $\gamma_i^{n-1}$  we compute the unit tangent vector  $T_i^{n-1}$ , positively oriented unit normal vector  $(N^+)_i^{n-1}$  and the signed curvature  $(\kappa^\pm)_i^{n-1}$ .
  - (iv) We compute indices of the DTM pixels (`EC.curvePixels`) where the curve points  $\gamma_i^{n-1}$  are located.
  - (v) Based on the indices `EC.curvePixels` from the previous step (iv) we compute the normal speed  $\beta$  for each curve point  $\gamma_i^{n-1}$  according to the equation (3.16). Then we compute the tangential speed  $\alpha$  using finite difference method according to the equation (3.23) from section 3.4.
  - (vi) Construction of the system matrix and right hand side using the semi-implicit IIOE (Inflow Implicit/Outflow Explicit) finite volume scheme, see [9, 13]. Then we solve this system to compute the new coordinates  $\gamma_i^n$ .
  - (vii) At each specified evolution step according to the value of `EC.ptsControlStep` we check, whether to stop the evolution based on the stopping criterion explained in section 4.1.1 below. If the condition (4.4) is satisfied, we stop the evolution.

V) **Export of the new mapping**, if it is sufficiently accurate.

- (1) Reduction of the curve points using the MATLAB function `reducepoly()`, which uses the *Ramer–Douglas–Peucker algorithm* [5, 15] to reduce the number of points.
- (2) Backward transformation of the new coordinates from S-JTSK(JTSK03) - Krovak East North grid coordinates (CS3) to the geodetic WGS84 coordinates (CS1).
- (3) Export of the new watercourse's nodal points to the OpenStreetMap database.

### 4.1.1 Stopping criterion

Stopping criterion is based on the percentage of curve points (in this section simply called points) which no longer move. By *not moving points* we understand such points which are either moving too slow or "are stuck jumping" between two pixels (blue and green points in Figure 4.1). Before we explain both, we will first define two displacement vectors  $\vec{r}_{i,1}$  and  $\vec{r}_{i,2}$  for each point  $\gamma_i^n$

$$\vec{r}_{i,1} = \gamma_i^n - \gamma_i^{n-1} \quad \text{and} \quad \vec{r}_{i,2} = \gamma_i^{n-1} - \gamma_i^{n-2},$$

<sup>1</sup>Maximal allowed number of evolution steps after which the evolution is stopped, if not stopped by the stopping criterion described in section 4.1.1.

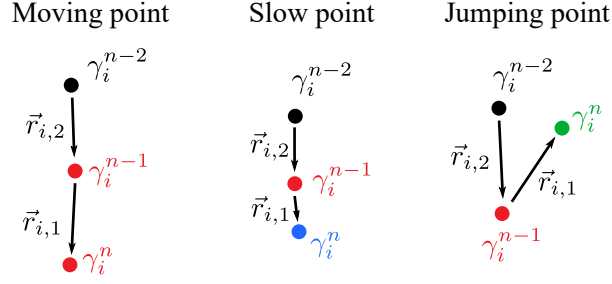


Figure 4.1: Examples of still moving points (red), a slow point (blue) and a jumping point (green).

subsequently normalizing both

$$\hat{r}_{i,1} = \frac{\vec{r}_{i,1}}{\|\vec{r}_{i,1}\|} \quad \text{and} \quad \hat{r}_{i,2} = \frac{\vec{r}_{i,2}}{\|\vec{r}_{i,2}\|}.$$

The term *slow point* is quite self-explanatory. For each point  $\gamma_i^n$  we compute the norm of the vector  $\vec{r}_{i,1}$ . Then we check, whether the norm  $\|\vec{r}_{i,1}\|$  is small enough to label a point  $\gamma_i^n$  as "slow". In this case we use the following condition

$$\|\vec{r}_{i,1}\| < 0.1 \text{ m}. \quad (4.1)$$

So if a point  $\gamma_i^n$  has moved less than 10 cm (one tenth of the DTM's pixel edge length), then we label it as a *slow point*.

On the other hand, to decide, whether a point  $\gamma_i^n$  "jumped back" in the opposite direction (see the green point in Figure 4.1), we compute the dot product between vectors  $\hat{r}_{i,1}$  and  $\hat{r}_{i,2}$ . Then we check the following condition

$$(\hat{r}_{i,1} \cdot \hat{r}_{i,2}) < \text{DPT}. \quad (4.2)$$

In a nutshell, what we are looking for are the dot products which fall inside the interval  $[-1, \text{DPT}[$ , where DPT stands for the Dot Product Threshold. That is when the vectors  $\hat{r}_{i,1}$  and  $\hat{r}_{i,2}$  are pointing in "sufficiently opposite" directions. In the numerical experiments we have chosen the threshold as  $\text{DPT} = -0.5$ , meaning, angles greater than  $120^\circ$ . So, if the dot product is smaller than the threshold DPT, then we label a point  $\gamma_i^n$  as a *jumping point*.

**Remark 5.** Intuitively speaking, this situation occurs, for instance, when two neighboring pixels have negative gradient vectors pointing at each other (for example DTM pixels on the bottom of the waterbed). Therefore, a point  $\gamma_i^n$  will end up moving back and forth between such pixels.

Next, we combine conditions (4.1) and (4.2) into

$$\|\vec{r}_{i,1}\| < 0.1 \text{ m} \quad \vee \quad (\hat{r}_{i,1} \cdot \hat{r}_{i,2}) < \text{DPT}, \quad (4.3)$$

by which we decide, whether we label a point  $\gamma_i^n$  as a *not moving point*. Using the condition (4.3) we compute the Number of Not Moving Points NNMP and subsequently check, if the percentage of these points is higher than the value NMPSC (Not Moving Points Stopping Criterion)

$$\frac{\text{NNMP}}{\text{EC.Npts}} > \text{NMPSC}. \quad (4.4)$$

If the condition (4.4) is met, we stop the evolution.

## 4.2 Examples: Stream Starý potok

As the first numerical experiment, we tried to fix mapping of the stream Starý potok. The red line in Figures 4.2a and 4.3a represents  $\gamma_0$ , the original mapping of the watercourse (W1). In this case the waterbed is quite simple and most of the original mapping is relatively accurate, aside from a few parts inside the *Zoomed area* in Figure 4.2a which can be better seen in Figure 4.3.

In all figures we highlighted the water points from the classified PC with cyan color, so that we can visually check if our model is working correctly. Furthermore, by red we plotted *still moving curve points* and by magenta we plotted *not-moving curve points*.

For the initial phase of the evolution we will use only the negative distance function gradient  $-\nabla d$  (setting  $\theta = 1$  in equation (3.17)) to attract  $\gamma_0$  as close to the waterbed as possible. Then in the second phase we will switch to the negative terrain gradient  $-\nabla h$  (setting  $\theta = 0$  in equation (3.17)) to push  $\gamma$  to the bottom of the waterbed.

**Remark 6.** The way we use the gradient vector fields  $-\nabla d$  and  $-\nabla h$  can be compared to sharpening a knife using whetstones. First a rough whetstone is used to do the hard work faster, then a finer one for polishing. In our case we initially use the negative distance function gradient  $-\nabla d$  to attract  $\gamma_0$  at least to the edge of the waterbed, bypassing any potential terrain obstacles (as described in section 3.3.1). This is why we also use a greater time step  $dt$ . Then we switch to the negative terrain gradient  $-\nabla h$  and a smaller time step  $dt$  to do the "final polish" of  $\gamma$ .

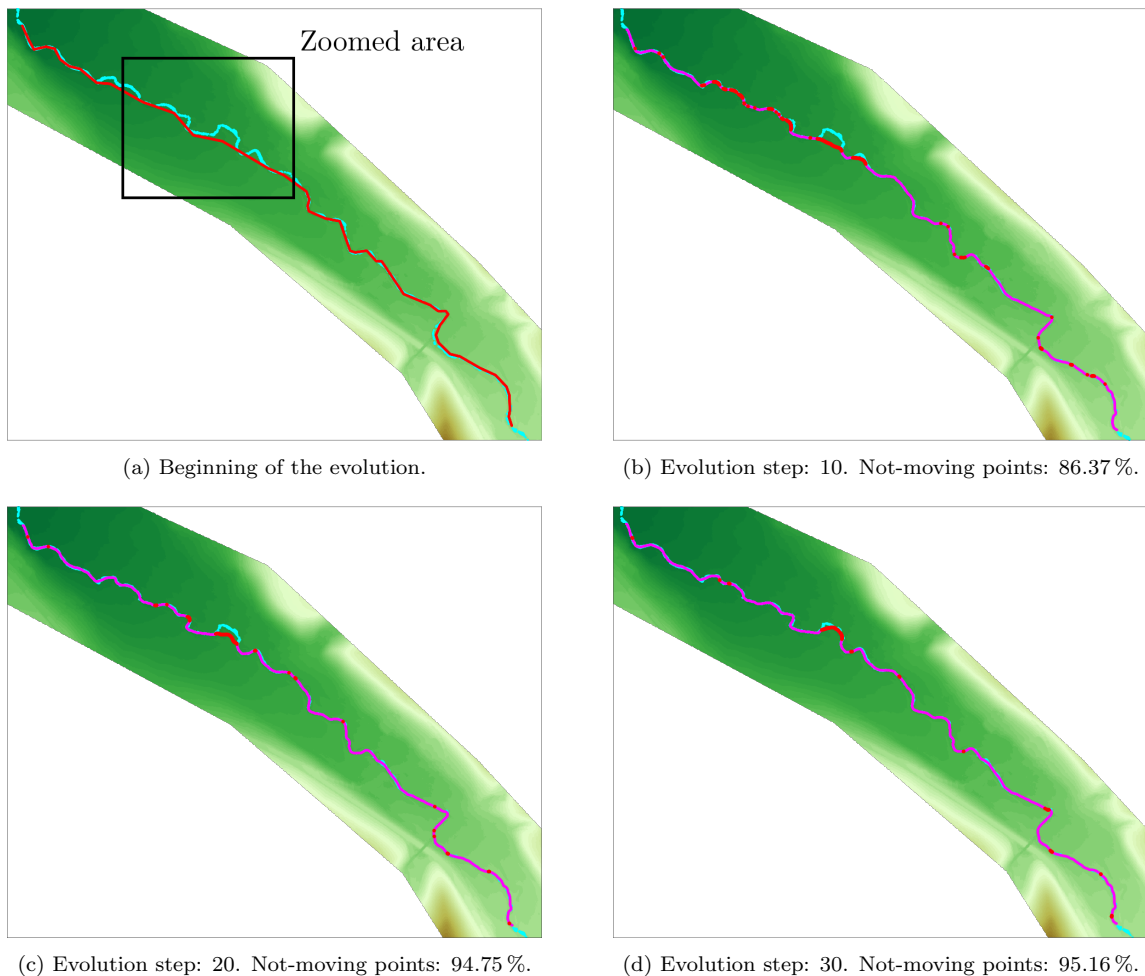
#### 4.2.1 Phase 1: Evolution driven by the distance function

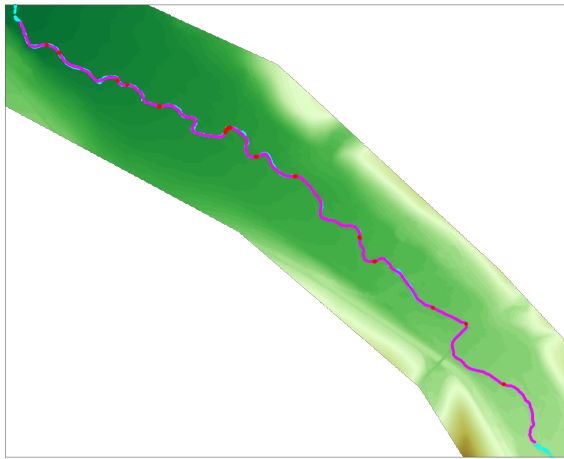
For this phase of the evolution we have chosen the parameters as follows:

Table 4.1: Parameters for the first phase of the evolution.

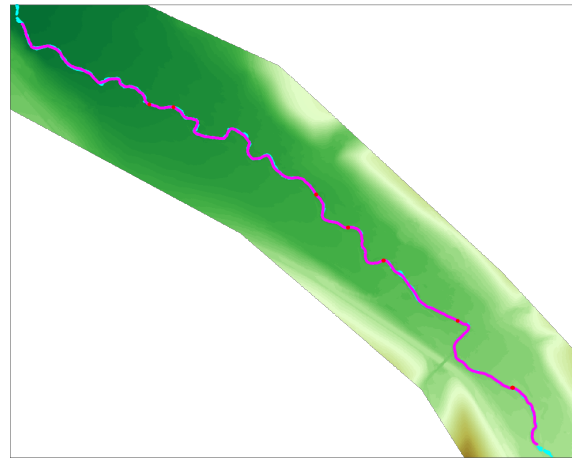
$\theta$	$\delta_{\text{ext}}$	$\delta_{\kappa}$	$\omega$	DPT	NMPSC	EC.nMax	dt
1.0	1.5	2.0	0.0	-0.5	99%	201	1.0

Since most of the original mapping was very close to the waterbed, majority of the curve points converged to the waterbed quickly after just 10 evolution steps, see Figure 4.2b. Stopping criterion (4.4) was satisfied after 50 evolution steps. Video available at [https://bit.ly/StaryPotok\\_Phase1](https://bit.ly/StaryPotok_Phase1).





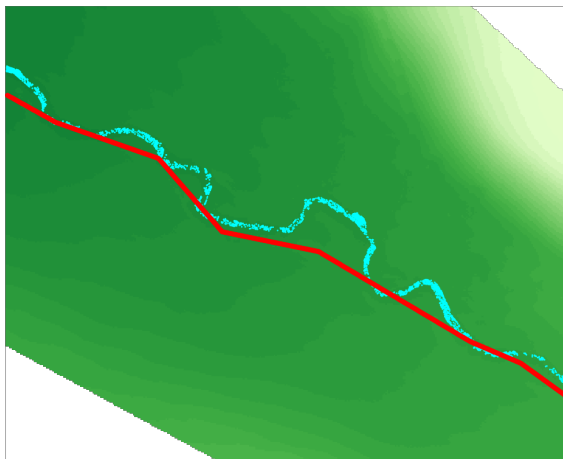
(e) Evolution step: 40. Not-moving points: 98.02%.



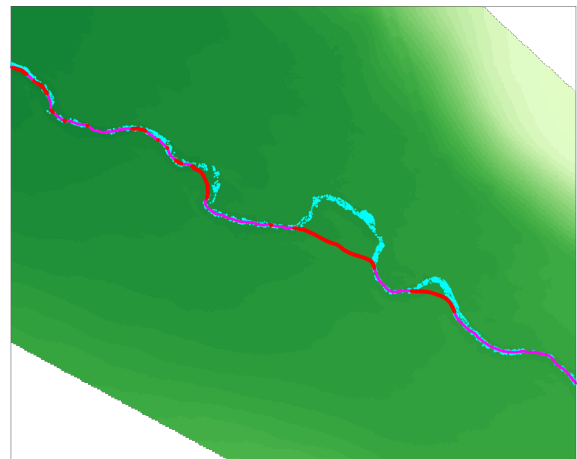
(f) Evolution step: 50. Not-moving points: 99.54%.

Figure 4.2: Evolution of the watercourse (W1) driven by the negative distance function gradient  $-\nabla d$  first.

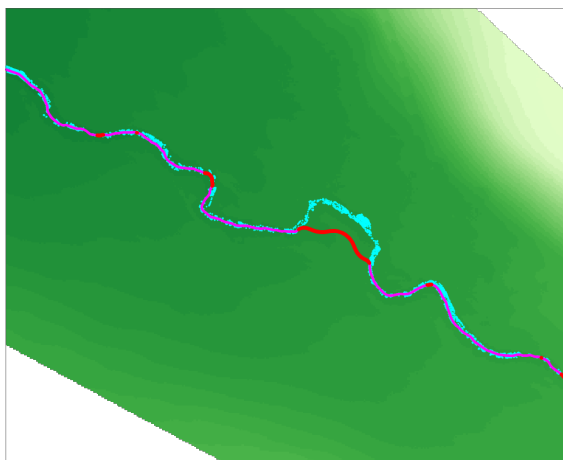
The evolution of  $\gamma$  in places where the original mapping of the watercourse (W1) was already accurate is not very interesting. Therefore, a better view of the evolution progress can be seen in Figure 4.3 below. Here we focused on the section of the watercourse (W1) where the original mapping was inaccurate. Video available at [https://bit.ly/StaryPotok\\_Zoom\\_Phase1](https://bit.ly/StaryPotok_Zoom_Phase1).



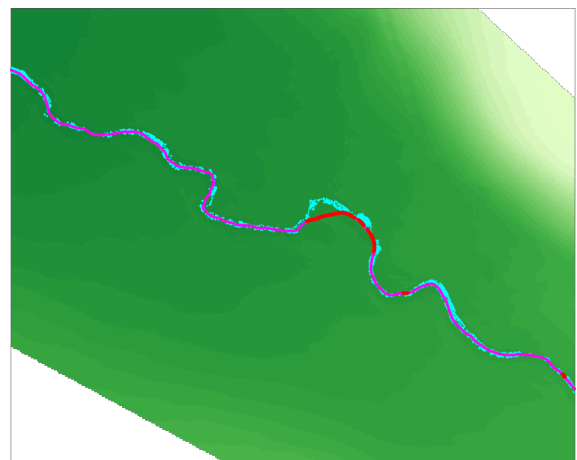
(a) Beginning of the evolution.



(b) Evolution step: 10. Not-moving points: 86.37%.

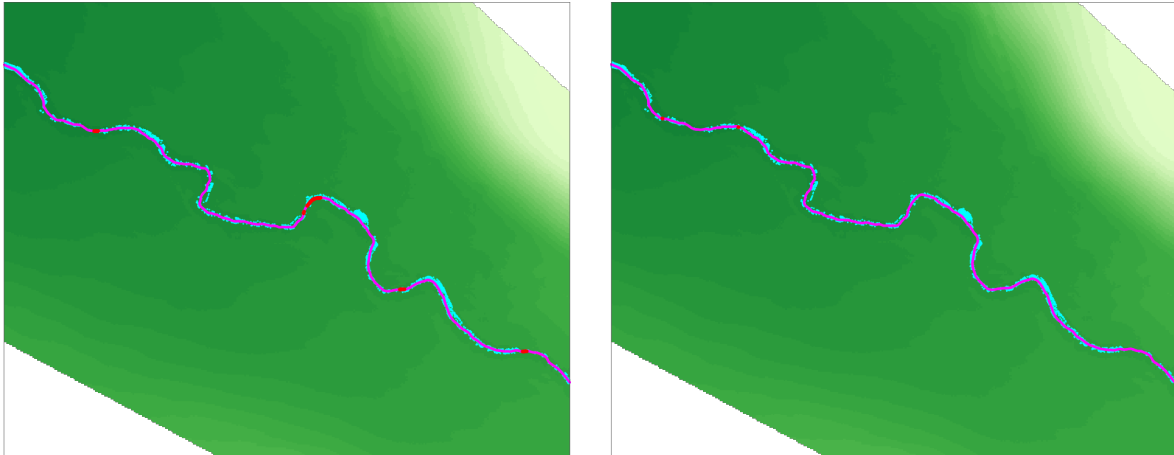


(c) Evolution step: 20. Not-moving points: 94.75%.



(d) Evolution step: 30. Not-moving points: 95.16%.

Figure 4.3: Zoomed in view on the section of the watercourse (W1) where the mapping was not as accurate.



(e) Evolution step: 40. Not-moving points: 98.02%.

(f) Evolution step: 50. Not-moving points: 99.54%.

Figure 4.3: Zoomed in view on the section of the watercourse (W1) where the mapping was not as accurate.

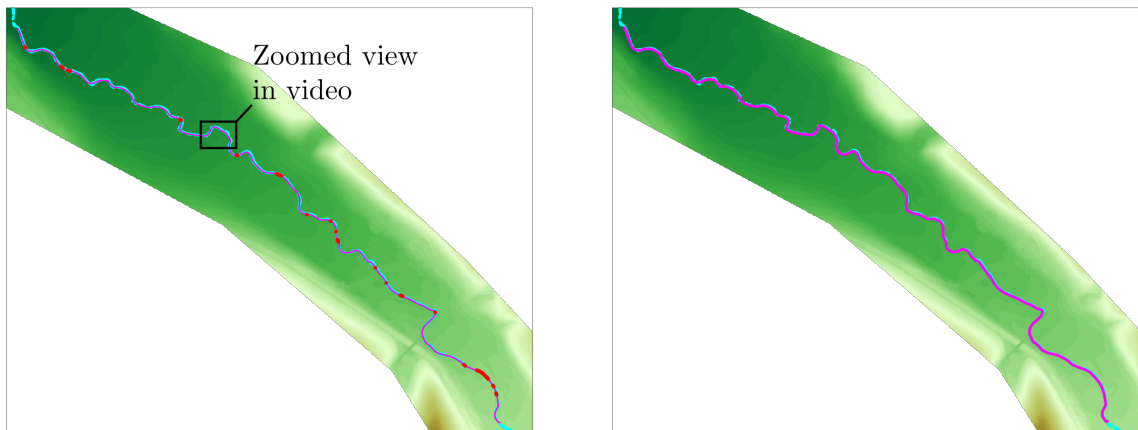
### 4.2.2 Phase 2: Evolution driven by the terrain function

In this phase of the evolution we also decreased the strength of the curvature regularization (parameter  $\delta_\kappa$ ) to prevent  $\gamma$  from leaving the parts of the waterbed with high curvature. Only very minor adjustments were applied to  $\gamma$ , as there are basically almost no visible differences between Figures 4.2f, 4.4a and 4.4b.

Table 4.2: Parameters for the second phase of the evolution.

$\theta$	$\delta_{\text{ext}}$	$\delta_\kappa$	$\omega$	DPT	NMPSC	EC.nMax	dt
0.0	3.0	0.1	0.0	-0.5	99%	201	0.5

Here we used the negative terrain gradient  $-\nabla h$  to attract  $\gamma$  to the bottom of the waterbed. It took only 20 evolution steps until 100% of the curve points were labeled as *not-moving*. Evolution progress during this phase can be seen much better in the video at [https://bit.ly/StaryPotok\\_Zoom\\_Phase2](https://bit.ly/StaryPotok_Zoom_Phase2) where we focused on the small area of interest highlighted in Figure 4.4a.



(a) Evolution step: 60. Not-moving points: 96.23%.

(b) Evolution step: 70. Not-moving points: 100.00%.

Figure 4.4: Subsequent evolution of the watercourse (W1) driven by the negative terrain gradient  $-\nabla h$ .

In Figure 4.5 we can see the original mapping of the watercourse (W1) compared to our much more accurate result. Main differences can be seen in Figures 4.5b and 4.5c where the original mapping insufficiently described the shape of the waterbed.

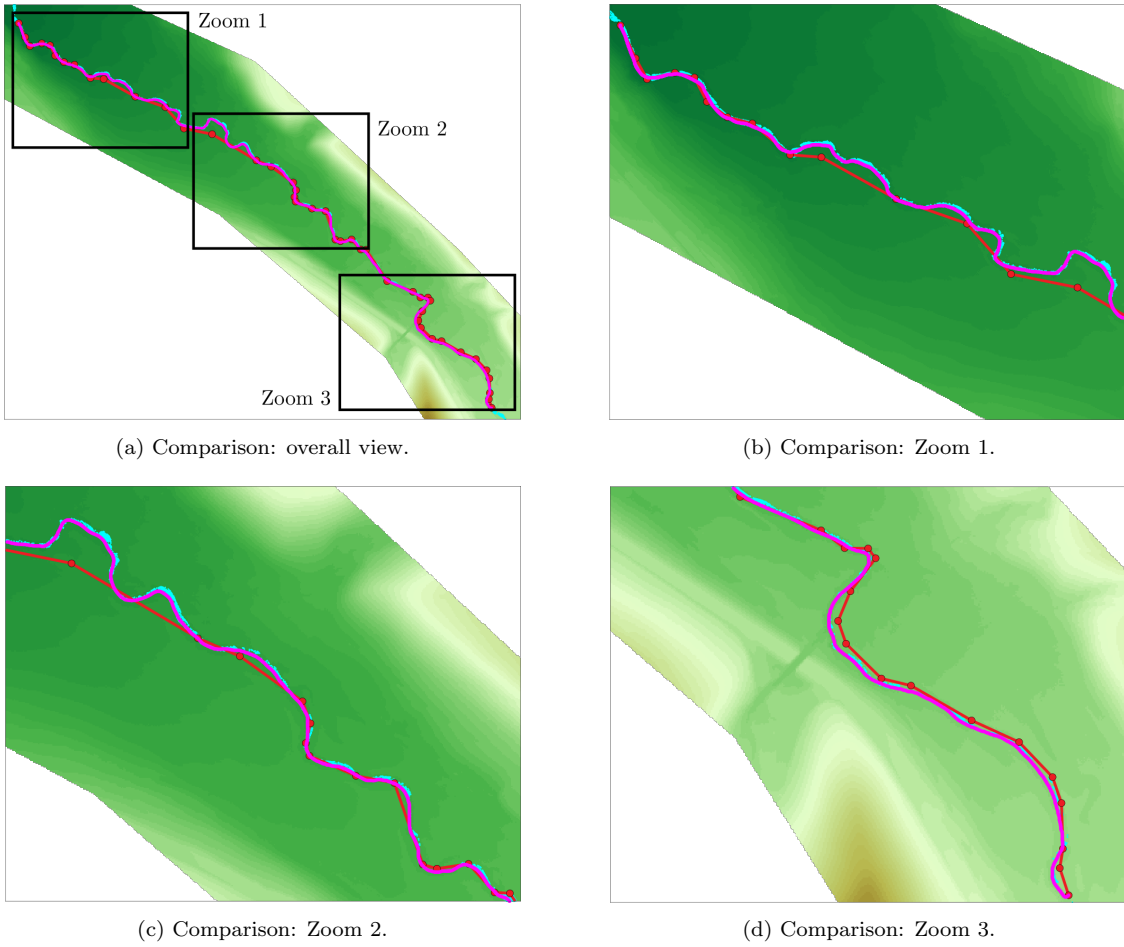


Figure 4.5: Comparison of the original mapping (red) vs. mapping computed by our model (magenta).

Furthermore, to not only show the accuracy of our model visually, we can also express our model's accuracy via the *Hausdorff distance*  $d_H(A, B)$ . It describes how far two sets of points  $A$  and  $B$  are from each other by finding the greatest distance between the two sets. It is defined as

$$d_H(A, B) = \max(d_{A,B}, d_{B,A}), \quad (4.5)$$

where

$$d_{A,B} = \max_{a \in A} \left( \min_{b \in B} \|a - b\| \right), \quad (4.6)$$

$$d_{B,A} = \max_{b \in B} \left( \min_{a \in A} \|b - a\| \right). \quad (4.7)$$

The distance  $d_{A,B}$  can be simply explained as follows: for each point  $a \in A$  we find the distance to the closest point  $b \in B$  and then choose the greatest of these distances. For the distance  $d_{B,A}$  the whole process is the same, but in the opposite direction. In general, however, the distances  $d_{A,B}$  and  $d_{B,A}$  do not need to be necessarily the same, see Figure 4.6. Therefore, as the Hausdorff distance  $d_H$ , we take the maximum as defined by the equation (4.5).

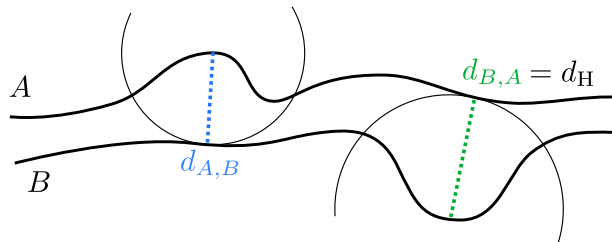


Figure 4.6: Hausdorff distance  $d_H$  between sets  $A$  and  $B$ , which are represented by two curves.

Additionally, we can use the so called *average Hausdorff distance*  $\bar{d}_H$  which is defined as

$$\bar{d}_H(A, B) = \frac{1}{2} (\bar{d}_{A,B} + \bar{d}_{B,A}), \quad (4.8)$$

where

$$\bar{d}_{A,B} = \frac{1}{|A|} \sum_{a \in A} \min_{b \in B} \|a - b\| \quad \text{and} \quad \bar{d}_{B,A} = \frac{1}{|B|} \sum_{b \in B} \min_{a \in A} \|b - a\|,$$

where  $|A|$  and  $|B|$  denote number of points in sets  $A$  and  $B$ , respectively. Here, instead of taking the maximum as in the equation (4.6), we compute the average of all distances across all points  $a \in A$ , similarly for  $\bar{d}_{B,A}$ . Then we compute the average of  $\bar{d}_{A,B}$  and  $\bar{d}_{B,A}$  as defined by the equation (4.8) to obtain the average Hausdorff distance  $\bar{d}_H$ . For our numerical experiments we will use the following sets of points:

- $\gamma_0$  - the set of the original nodal points.
- $\gamma_{N1}, \gamma_{N2}$  - the sets of the new nodal points computed by our model after phase 1 and phase 2.
- $\gamma_E$  - the set of the nodal points which represent the *ground truth*.<sup>2</sup>

Individual results of the Hausdorff distance for the watercourse (W1) are written down in Tables 4.3a, 4.3b and 4.3c bellow.

Table 4.3: Comparison of the computed Hausdorff distances.

(a) Before evolution.		(b) After phase 1.		(c) After phase 2.	
$d_H(\gamma_0, \gamma_E)$	34.35 m	$d_H(\gamma_{N1}, \gamma_E)$	4.65 m	$d_H(\gamma_{N2}, \gamma_E)$	3.52 m
$\bar{d}_H(\gamma_0, \gamma_E)$	4.20 m	$\bar{d}_H(\gamma_{N1}, \gamma_E)$	0.84 m	$\bar{d}_H(\gamma_{N2}, \gamma_E)$	1.35 m

As we can see from Table 4.3b, after the first evolution phase the accuracy has increased dramatically, the value of the distance  $\bar{d}_H$  is comparable to the resolution of DTM (1 m). DTM we used here has dimensions  $1222 \times 1519$  pixels ( $1222 \text{ m} \times 1519 \text{ m}$  in real world). After the second phase (Table 4.3c), the maximal distance  $d_H$  has improved approximately by 1 m and the average distance  $\bar{d}_H$  has slightly worsened by 0.5 m. However, compared to the resolution of DTM, the differences between computed distances shown in Tables 4.3b and 4.3c are insignificant. By comparing the results from Tables 4.3a and 4.3c, we can conclude that our model managed to compute a much more accurate new mapping of the watercourse (W1).

## 4.3 Examples: Stream Parná

For the next numerical experiment we have chosen part of stream Parná. This time the waterbed is much more complicated compared to the watercourse (W1). As it can be seen in Figure 4.7a, the original mapping is also quite inaccurate in most places. In this case, we will use the same strategy regarding the choice of the parameter  $\theta$  as we used for the watercourse (W1) in section 4.2.

### 4.3.1 Phase 1: Evolution driven by the distance function

For the first phase of the evolution we have chosen the parameters as follows:

$\theta$	$\delta_{\text{ext}}$	$\delta_{\kappa}$	$\omega$	DPT	NMPSC	EC.nMax	dt
1.0	1.5	2.0	0.0	-0.5	98 %	201	1.0

Due to the complicated shape of the watercourse (W2) this phase of the evolution needed almost three times the amount of steps compared to the first phase of the evolution of the watercourse (W1). As can be seen in places highlighted by the (PD2) arrows in Figure 4.7j, not all parts of  $\gamma$  were attracted to the waterbed, as if  $\gamma$  was not able "to enter few problematic meanders."

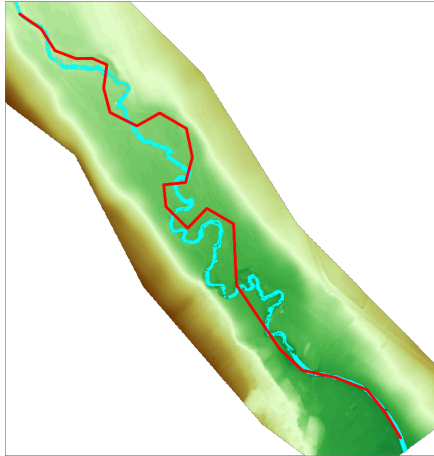
<sup>2</sup>In other words the "exact solution" which we manually created to our best abilities and exported using the website [www.freemap.sk](http://www.freemap.sk). We used both, DTM and water points as reference.



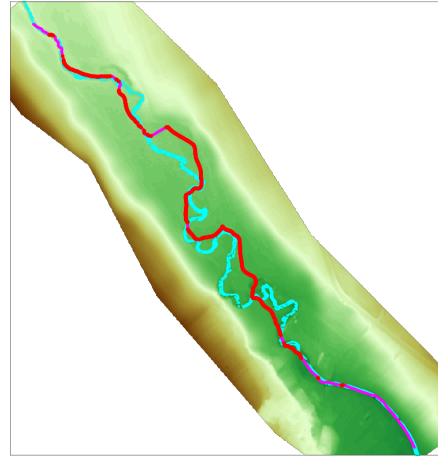
The problem is that the negative distance function gradient  $-\nabla d$  is mostly tangent to  $\gamma$  in these places, see Figure 4.10. Therefore, there is little to none normal speed  $\beta$  to push  $\gamma$  forward. But sometimes a few curve points make it through, see the evolution progress at the place marked in Figure 4.7e.

Also, upon further investigation we have found out one additional inconvenience regarding the section of  $\gamma$  highlighted by the (PH2) arrow in Figure 4.7j. First,  $\gamma$  was unable to get inside the meander due to the same reason as described above. Then, because of the old waterbed being present there,  $\gamma$  has become trapped inside, unable to reach the actual waterbed.

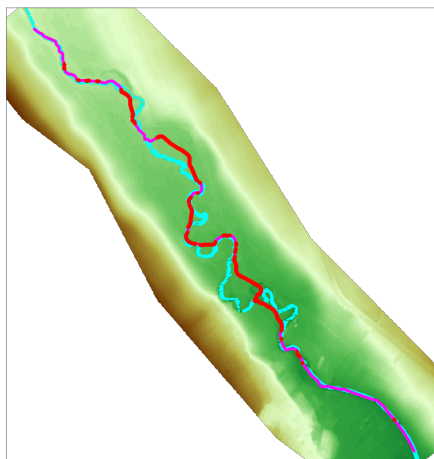
Video of this phase of the evolution is available at [https://bit.ly/Parna\\_Phase1](https://bit.ly/Parna_Phase1).



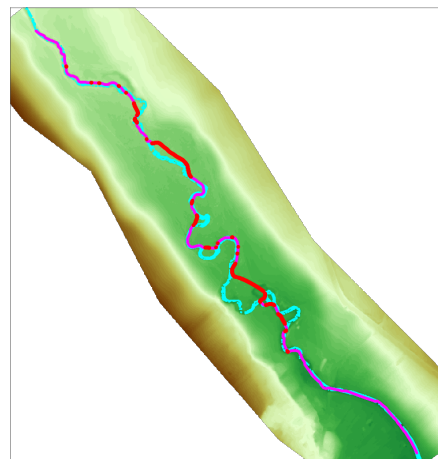
(a) Beginning of the evolution.



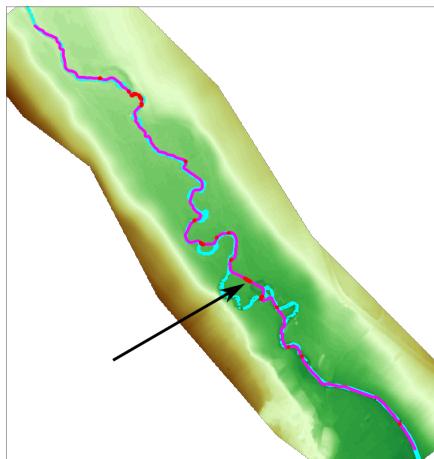
(b) Evolution step: 10. Not-moving points: 38.72%.



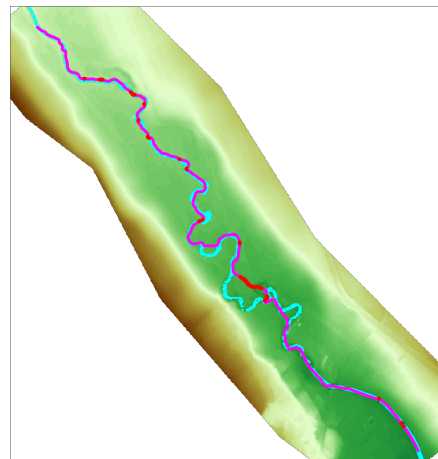
(c) Evolution step: 20. Not-moving points: 58.73%.



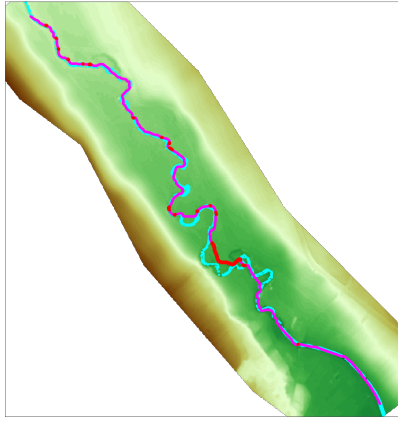
(d) Evolution step: 30. Not-moving points: 75.51%.



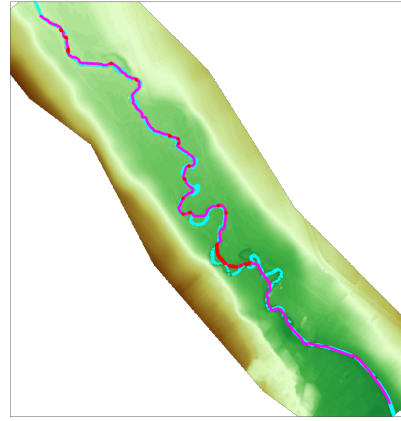
(e) Evolution step: 60. Not-moving points: 96.04%.



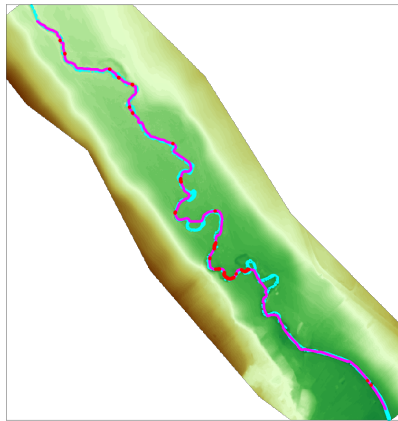
(f) Evolution step: 80. Not-moving points: 93.86%.



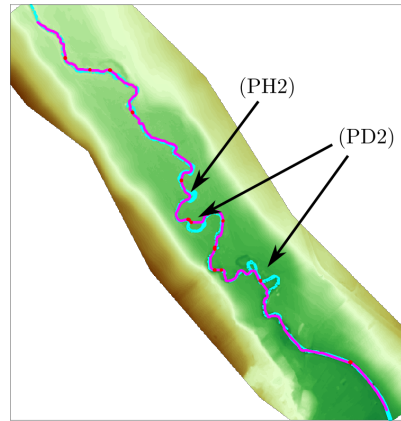
(g) Evolution step: 100. Not-moving points: 91.65%.



(h) Evolution step: 120. Not-moving points: 93.48%.



(i) Evolution step: 150. Not-moving points: 94.70%.



(j) Evolution step: 170. Not-moving points: 98.64%.

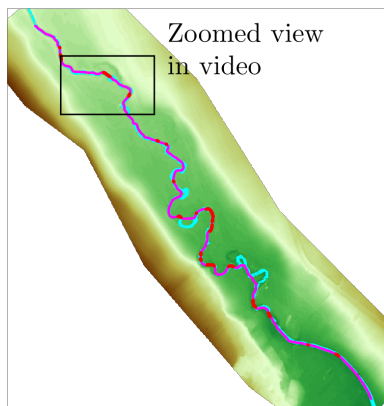
Figure 4.7: Evolution of the watercourse (W2) driven by the negative distance function gradient  $-\nabla d$  first.

### 4.3.2 Phase 2: Evolution driven by the terrain function

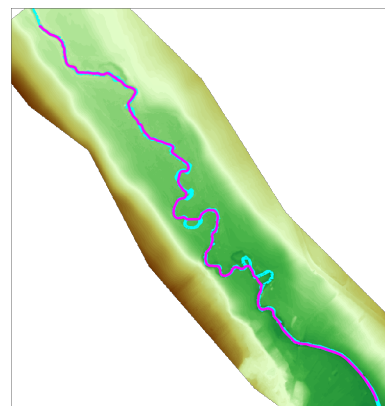
For the second phase we have chosen the parameters as follows:

$\theta$	$\delta_{\text{ext}}$	$\delta_{\kappa}$	$\omega$	DPT	NMPSC	EC.nMax	dt
0.0	3.0	0.1	0.0	-0.5	99%	201	0.5

As can be seen in Figure 4.8b, even the negative terrain gradient  $-\nabla h$  was unable to push  $\gamma$  into the waterbed. On the other hand, it had some smoothing effects, a better view of which can be seen in the following video at [https://bit.ly/Parna\\_Zoom\\_Phase2](https://bit.ly/Parna_Zoom_Phase2).



(a) Evolution step:180. Not-moving points: 90.19%.



(b) Evolution step:190. Not-moving points: 100.00%.

Figure 4.8: Subsequent evolution of the watercourse (W2) driven by the negative terrain gradient  $-\nabla h$ .

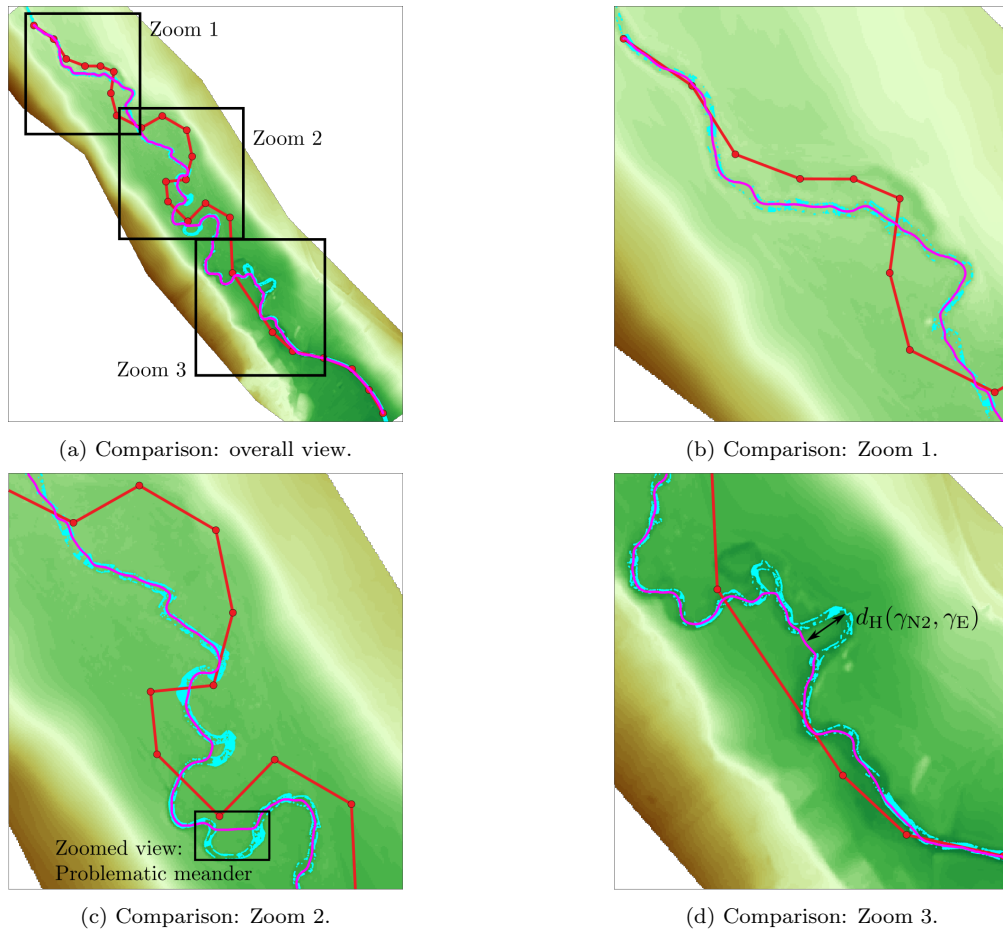


Figure 4.9: Comparison of the original mapping (red) vs. mapping computed by our model (magenta).

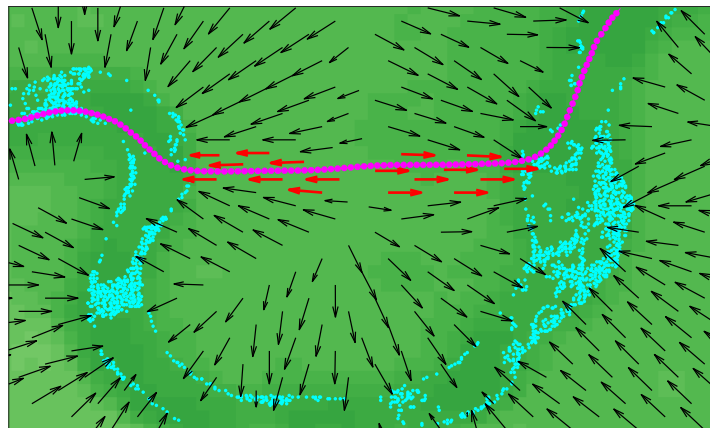


Figure 4.10: Zoomed view of the place where  $\gamma$  was unable to enter the meander due to  $-\nabla d$  being mostly tangent to  $\gamma$  (see the red arrows). For this reason, there is almost no normal speed  $\beta$  to push  $\gamma$  towards the waterbed.

For the most part, however, the mapping computed by our model is much more accurate than the original one, see Figures 4.9b, 4.9c, 4.9d and the results of  $\bar{d}_H$  in Tables 4.4b and 4.4c. The greatest inaccuracy was measured inside one of the problematic meanders, which is shown in Figure 4.9d.

Table 4.4: Comparison of the computed Hausdorff distances.

(a) Before evolution.		(b) After phase 1.		(c) After phase 2.	
$d_H(\gamma_0, \gamma_E)$	64.69 m	$d_H(\gamma_{N1}, \gamma_E)$	34.52 m	$d_H(\gamma_{N2}, \gamma_E)$	34.79 m
$\bar{d}_H(\gamma_0, \gamma_E)$	18.06 m	$\bar{d}_H(\gamma_{N1}, \gamma_E)$	2.59 m	$\bar{d}_H(\gamma_{N2}, \gamma_E)$	1.86 m

## 4.4 Examples: Nameless stream in Tatras

For the last numerical experiments we have chosen Nameless stream<sup>3</sup>, one of many small streams flowing throughout Tatras. Unfortunately, the classified PC for the watercourse (W3) had no water points, therefore, we were forced to use only DTM. Meaning, choice of the parameter  $\theta$  in equation (3.17) is restricted to  $\theta = 0$ .

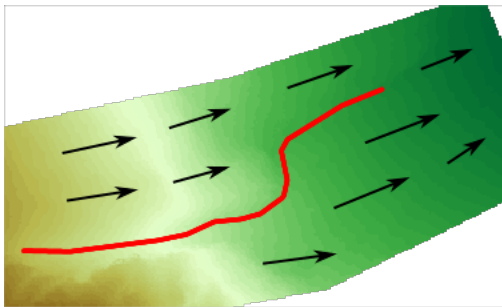
In section 4.4.1 we will show the effects of the steep terrain by using the terrain function given by the equation (3.11). And then in section 4.4.2 we used DTM after subtracting the terrain's trend as defined by the equation (3.10) in section 3.3.1.

### 4.4.1 Evolution before subtracting the terrain's trend

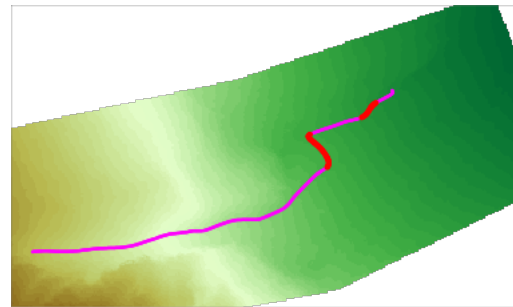
For the evolution in this section, we have chosen the parameters as follows:

$\theta$	$\delta_{\text{ext}}$	$\delta_{\kappa}$	$\omega$	DPT	NMPSC	EC.nMax	dt
0.0	2.0	1.0	0.0	-0.5	99%	350	0.5

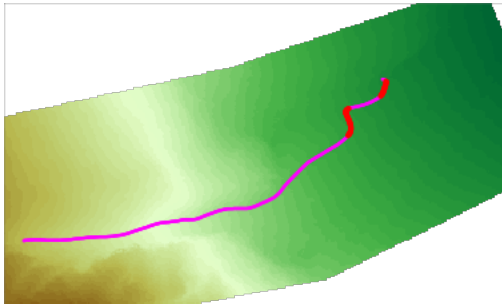
In Figure 4.11a, the terrain's trend is highlighted by the black arrows. As was expected,  $\gamma$  has been just "flushed downhill". Given a higher value of EC.nMax,  $\gamma$  would eventually move outside the current DTM, thus causing a crash of the computation. Video of the evolution is available at this link [https://bit.ly/NamelessStream\\_withTrend](https://bit.ly/NamelessStream_withTrend).



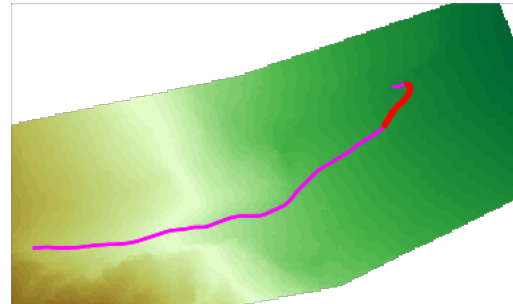
(a) Beginning of the evolution.



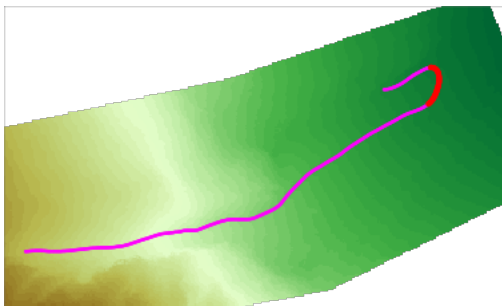
(b) Evolution step: 50. Not-moving points: 85.50%.



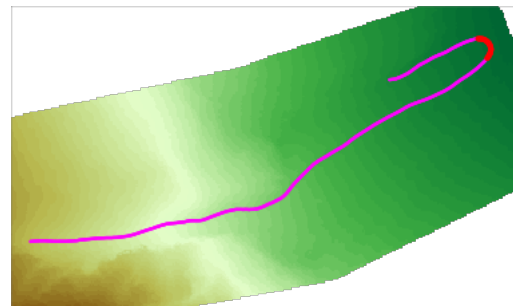
(c) Evolution step: 100. Not-moving points: 88.81%.



(d) Evolution step: 150. Not-moving points: 88.71%.



(e) Evolution step: 250. Not-moving points: 92.16%.



(f) Evolution step: 350. Not-moving points: 95.60%.

Figure 4.11: Evolution of the watercourse (W3) using the terrain data BEFORE subtracting the trend.

<sup>3</sup>Even though it is depicted on map, it has no name assigned. Therefore we will simply call it "Nameless stream."

#### 4.4.2 Evolution after subtracting the terrain's trend

In this section, the parameters for the evolution were the same as in section 4.4.1. The only difference being that DTM with subtracted trend was used instead. Right away we can see a huge improvement. As can be seen in Figure 4.12c,  $\gamma$  managed to converge to the potential waterbed after only 20 evolution steps when the stopping criterion (4.4) was satisfied. Video of the evolution from this section is available at link [https://bit.ly/NamelessStream\\_withoutTrend](https://bit.ly/NamelessStream_withoutTrend).

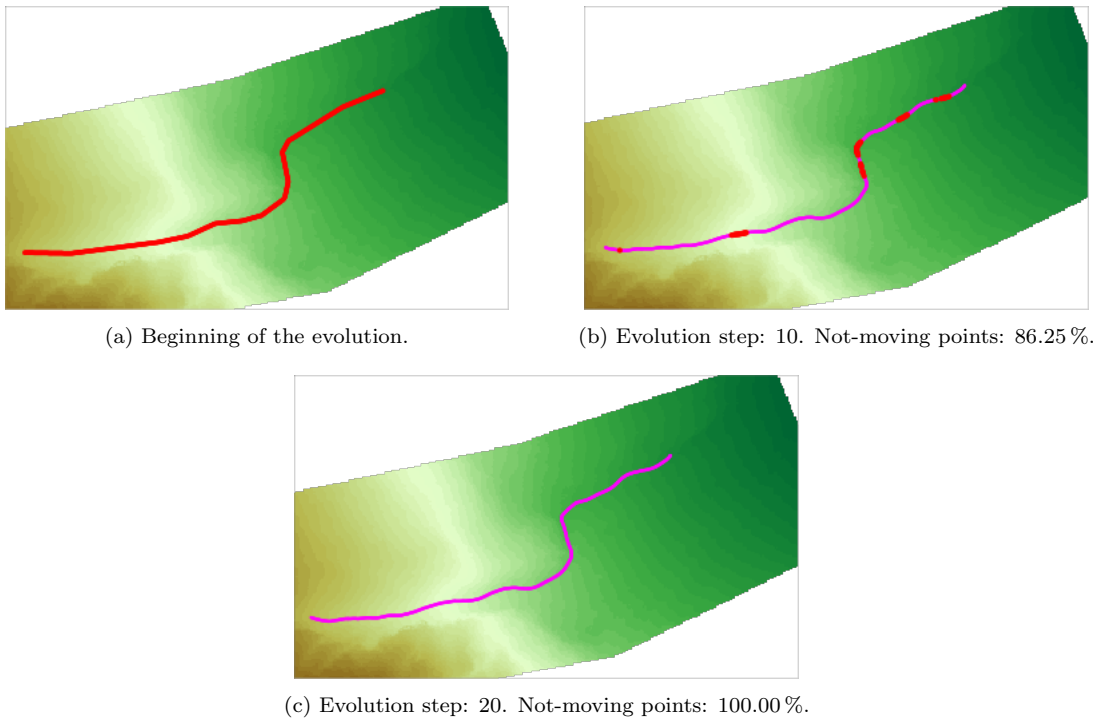


Figure 4.12: Evolution of the watercourse (W3) using the terrain data AFTER subtracting the trend.

Unfortunately, without the water points from the classified PC, we can only assume that it is the actual waterbed. Therefore, in this case, we will not compute the Hausdorff distances  $d_H$  and  $\bar{d}_H$ . We have also tried to export several classified PCs around a few other streams from Tatras region. However, none of the classified PCs had water points.

On the other side, in these numerical experiments we wanted to demonstrate the functionality of our suggested solution to the problem (PH3) from section 3.3.1. With the use of a fitting plane we managed to remove the problem regarding the terrain's steep incline in mountainous regions.

We have also performed one numerical experiment with longer part of the watercourse (W3). Obviously, for this experiment we had to export a larger DTM around the watercourse (W3). Here we have encountered another problem. With more complex terrain shape, a simple linear fitting plane may not be sufficient enough to describe the actual terrain trend. Therefore, more complex fitting surfaces will need to be used. This can be a part of possible future improvements of our model.

# Chapter 5

## Conclusions

The goal of this thesis was to create a mathematical model that would automatically correct the mapping of watercourses on maps, with the use of the airborne laser scanning data from ÚGKK SR.

Import of the terrain and OpenStreetMap data into MATLAB and subsequent processing was successful, mainly thanks to the availability of numerous useful MATLAB functions, all of which we described in sections 2.1.1, 2.1.1 and 2.2. Transformation of coordinates from section 2.3 was also implemented into MATLAB successfully. The implementation of our model given by the equation (3.17) in MATLAB was relatively straight forward thanks to the pre-existing code for curve evolution driven by signed curvature which was developed as part of the project APVV-18-0247, titled "Automation of Building's Electronic Documentation Verification Using Innovative Data Collection Techniques and Virtual Models".

Based on the results shown in Figures 4.5 and 4.9 we can say that our model has done an acceptable work. For a simpler waterbed of the stream Starý potok we managed to create a new, more accurate mapping without any significant issues. On the other hand, in case of the stream Parná our model ran into some problems. Both of the two gradient vector fields that we used were unable to attract  $\gamma_0$  (original mapping of the watercourse) into the waterbed in places with problematic meanders, as can be seen in Figures 4.9c and 4.9d. Solution to this problem may be the subject of the future works dealing with improvements of our model. However, the majority of the new mapping computed by our model is very accurate, also confirmed by the average Hausdorff distances in Tables 4.4b and 4.4c. Therefore, we can conclude its functionality as sufficiently capable. In section 4.4, we have also demonstrated the functionality of our suggested solution to the problem (PH3) regarding strong terrain's trend, even though there is also a place for future improvements. Either to use higher order fitting surfaces or a piece-wise fitting plane.

Reduction of the curve points was accomplished easily and successfully thanks to the MATLAB function `reducepoly()`. Next, the backward transformation of the new nodal points' coordinates to the geodetic WGS84 coordinates (CS1) was also successfully implemented in our own MATLAB function. In this case we also managed to decrease the number of the needed transformations by one, thanks to the direct datum transformation EPSG:5239 from geocentric S-JTSK(JTISK03) to geocentric WGS84 coordinates (CS2). For the export of the new coordinates into GPX file format we had to create our own MATLAB function. This part was also very straight forward and done relatively quickly.

All that remains is to use the exported GPX file with the new coordinates and replace the old mapping via the free map editor JOSM. Furthermore, we also plan to apply our model on other watercourses to uncover more of any potential deficiencies, perhaps include some additional input data that might help improve our model.

# Resumé

Vodné toky sú v mapách niekedy zakreslené len približne a nekopírujú skutočný tvar koryta. Tieto nepresnosti môžu byť často aj na úrovni desiatok metrov. Ako ale vieme takéto nesprávne trasovania opraviť? Vďaka novému Digitálnemu Modelu Reliéfu (DMR) s vysokým rozlíšením  $1 \times 1$  m, ktorý poskytuje Ústav Geodézie, Kartografie a Katastra Slovenskej republiky, vieme manuálne takéto nesprávne trasovania vodných tokov napraviť. Ako však tento proces urobiť automaticky?

Hlavným cieľom tejto bakalárskej práce bolo vytvoriť matematický model, ktorý by vedel nesprávne trasovanie vodného toku opraviť tak, aby kopírovalo skutočný tvar koryta. Náš matematický model využíva znalosti z oblasti diferenciálnej geometrie, konkrétne sme sa zamerali na vývoj parametrickej krivky  $\gamma$  v rýchlostnom vektorovom poli  $\vec{V}$ . Základnou myšlienkou za naším modelom je zobrať aktuálne trasovanie vodného toku  $\gamma_0$  a vložiť ho do vhodne skonštruovaného rýchlostného vektorového poľa  $\vec{V}$ , ktoré by malo počiatočnú krivku  $\gamma_0$  dotlačiť do skutočného koryta. Náš model sme implementovali v programe MATLAB s využitím už existujúceho kódu, ktorý bol naprogramovaný pre evolúciu kriviek riadenú krivosťou, s využitím *semi-implicitnej IIOE schémy konečných objemov*.

Prvým krokom bolo získanie dát o teréne okolo vodného toku pomocou Mapového klientu ZBGIS. Konkrétne sa jedná o DMR a klasifikované Mračno Bodov (MB), z ktorého sme vybrali iba tie body, ktoré sú klasifikované ako *Voda*. Tieto dáta boli vyjadrené v súradnicovom systéme S-JTSK(JTSK03) - Křovákovo zobrazenie. Jednotlivé dáta sme detailne popísali v sekcii 2.1. Aktuálne zakreslenie vodného toku sme získali z voľne dostupnej mapovej databázy OpenStreetMap pomocou mapového editora JOSM. Ide o sekvenciu uzlových bodov, ktoré reprezentujú vodný tok na mape. Avšak súradnice týchto uzlových bodov boli v geodetických WGS84 súradniciach, teda sme ich potrebovali najprv transformovať do súradnicového systému S-JTSK(JTSK03) - Křovákovo zobrazenie. Tento proces sme opísali v sekcii 2.3. Načítanie všetkých dát do prostredia MATLAB-u sme taktiež popísali v prislúchajúcich sekciách kapitoly 2.

Detaily nami navrhovaného matematického modelu sme podrobne opísali v kapitole 3. Ako prvé sme predstavili *vyvíjajúcu sa parametrickú krivku*  $\gamma$ . Následne sme v sekcii 3.2 opísali evolúciu krivky  $\gamma$  pomocou rovnice (3.1). Táto rovnica hovorí, že každému bodu krivky  $\gamma(u, t)$  predpíšeme vektor rýchlosti  $\vec{v}(u, t)$ , ktorý bude danému bodu krivky  $\gamma(u, t)$  určovať smer pohybu počas evolúcie. Rýchlostný vektor  $\vec{v}$  sme si následne rozložili na dve zložky v smere jednotkových vektorov  $T$  a  $N^+$  a definovali dve funkcie: *normálovú rýchlosť*  $\beta$  a *tangenciálnu rýchlosť*  $\alpha$ . Tieto dve funkcie môžeme rozumieť ako riadiace prvky, pomocou ktorých budeme riadiť evolúciu krivky  $\gamma$ . Na výpočet normálovej rýchlosti  $\beta$  sme využili záporné gradienty dvoch funkcií, konkrétne *funkcie terénu*  $h$  (definovaná pomocou DMR a detailne opísaná v sekcii 3.3.1) a *vzdialenostnej funkcie* od tých bodov z MB klasifikovaných ako *Voda* (detailne opísaná v sekcii 3.3.2). Taktiež sme dali do pozornosti niekoľko potenciálnych problémov, ktoré by sa mohli počas evolúcie objaviť. Napríklad problém (PH3), ktorý hovorí o príliš naklonenom teréne v horských oblastiach. Riešenie na tento problém sme opísali v sekcii 3.3.1 a jeho funkcionalitu demoštrovali v numerickom experimente v sekcii 4.4.2. Pre elimináciu problémov (PH1), (PH2) a (PD1) sme v sekcii 3.3.3 navrhli komplexnejšiu verziu normálovej rýchlosti  $\beta$ , kde sme navrhli váženú kombináciu vektorových polí  $-\nabla h$  a  $-\nabla d$  pomocou rovnice (3.14) s parametrom  $\theta$ . Taktiež sme pridali člen *krivosťnej regularizácie* definovaný pomocou vzťahu (3.15), ktorého úlohou je prekonanie šumu a zhladzovanie častí krivky  $\gamma$  s prívelmi vysokou krivosťou počas evolúcie. Pri tangenciálnej rýchlosti  $\alpha$  sme využili tzv. *asymptoticky rovnomernú redistribúciu* bodov, o ktorej píšeme v sekcii 3.4. Jej implementácia v MATLAB-e bola taktiež súčasťou už existujúceho kódu, ktorý mi bol poskytnutý. Finálna verzia nášho modelu je definovaná pomocou rovnice (3.17).

V poslednej kapitole 4 sme najprv predstavili tri vodné toky (W1), (W2) a (W3), ktorých nesprávne trasovanie sme chceli opraviť. Ďalej sme v sekcii 4.1 opísali postup práce od nájdenia vodného toku, ktorý potrebuje korekciu, ďalej načítanie a spracovanie dát v MATLAB-e, popis jednotlivých krokov algoritmu pre evolúciu krivky a čo následne spravíme s novým trasovaním vodného toku, ktorý vypočítal

náš model. V sekcii 4.1.1 sme opísali nami navrhnuté zastavovacie kritérium, ktoré je vyjadrené pomocou podmienky (4.4). Výsledky numerických experimentov sú prezentované v sekciiach 4.2, 4.3 a 4.4. V týchto sekciiach si môže čitateľ pozrieť priebeh evolúcie na množstve obrázkov z jednotlivých krokov evolúcie. Avšak pre lepší vizuálny zážitok silne odporúčame v jednotlivých sekciiach využiť odkazy na videá, v ktorých je priebeh evolúcie omnoho lepšie viditeľný. Presnosť nového mapovania vodných tokov (W1) a (W2) sme overovali jednak vizuálne na obrázkoch 4.5 a 4.9, ale aj s pomocou *Hausdorfovej vzdialenosti* (4.5) a *priemernej Hausdorfovej vzdialenosti* (4.8) (pozri Tabuľky 4.3 a 4.4). Pri vodnom toku (W3) sme nanešťastie nemali v klasifikovanom MB žiadne body klasifikované ako *Voda*, takže sme boli nútený použiť ako vstupné dáta iba DMR. V tomto prípade sme sa teda zamerali na demonštráciu funkcionality nami navrhovaného riešenia problému (PH3) ohľadom priveľmi nakloneného terénu. V sekcii 4.4.1 sme ukázali, aké následky na evolúciu môže mať príliš strmý terén - krivka  $\gamma$  "odtiekla preč" dole kopcom. Následne v sekcii 4.4.2 sme pri evolúcii využili vyrovnaný terén. Z tohto numerického experimentu sme dostali oveľa lepší výsledok, ako môžeme vidieť, ak porovnáme výsledky na Obrázkoch 4.11f a 4.12c.

Implementácia jednotlivých krokov zo sekcie 4.1 do MATLAB-u bola v podstate bezproblémová, hlavne vďaka obrovskému množstvu užitočných funkcií z rôznych rozširujúcich balíkov, ktoré MATLAB ponúka. Na záver môžeme zhodnotiť, že náš navrhovaný model (3.17) si svoju úlohu splniť celkom dobre. Avšak taktiež sme narazili na niekoľko ďalších problémov. Napríklad pri komplikovanejšom tvare koryta vodného toku (W2) si náš model nevedel poradiť pri niekoľkých problematických meandroch. Tento problém je dobre vysvetlený na Obrázku 4.10. Taktiež, ako sme vysvetľovali na konci sekcie 4.4.2, pri komplikovanejšom tvare terénu okolo vodného toku v horských oblastiach, jednoduchá fitovacia rovina nemusí stačiť na potrebné odstránenie trendu. Riešenia na tieto problémy môžu byť teda náplňou budúcich prác študentov, ktorých by zaujímala oblasť aplikovanej matematiky s využitím reálnych dát.



# Bibliography

1. AMBROZ, M.; BALAŽOVJECH, M.; MEDĽA, M.; MIKULA, K. Numerical modeling of wildland surface fire propagation by evolving surface curves. *Advances in Computational Mathematics*. 2019, vol. 45. Available from DOI: 10.1007/s10444-018-9650-4.
2. BALAŽOVJECH, M.; MIKULA, K.; ŠIBÍKOVÁ, M.; URBÁN, J. Lagrangean method with topological changes for numerical modelling of forest fire propagation. *Proceedings of ALGORITMY 2012, 19th Conference on Scientific Computing*. 2012, pp. 42–52. Available also from: <https://www.researchgate.net/publication/258291381>.
3. BENNINGHOFF, H.; GARCKE, H. Segmentation and Restoration of Images on Surfaces by Parametric Active Contours with Topology Changes. *Journal of Mathematical Imaging and Vision*. 2016, vol. 55, no. 1, pp. 105–124. ISSN 1573-7683. Available from DOI: 10.1007/s10851-015-0616-6.
4. *Coordinate Conversions and Transformations including Formulas*. 2021-12. Report, 373-07-02. International Association of Oil & Gas Producers. Available also from: <https://drive.tiny.cloud/1/4m326iu12oa8re9cjiadxonharclteqb4mumfxj71zsttwkx/41bad341-1dea-429f-9fad-1e68666da5e5>. Geomatics Guidance Note number 7, part 2.
5. DOUGLAS, D. H.; PEUCKER, T. K. ALGORITHMS FOR THE REDUCTION OF THE NUMBER OF POINTS REQUIRED TO REPRESENT A DIGITIZED LINE OR ITS CARICATURE. *Cartographica: The International Journal for Geographic Information and Geovisualization*. 1973, vol. 10, no. 2, pp. 112–122. Available from DOI: 10.3138/FM57-6770-U75U-7727.
6. HARIKUMAR, A. *Advanced methods for tree species classification and biophysical parameter estimation using crown geometric information in high density LiDAR data*. 2020. Available from DOI: 10.13140/RG.2.2.10395.69922. PhD thesis.
7. HIRT, C. Digital Terrain Models. *Encyclopedia of Geodesy*. 2016. Available from DOI: 10.1007/978-3-319-02370-0\_31-1.
8. MATLAB. *Release 2021b*. Natick, Massachusetts: The MathWorks Inc., 2021.
9. MIKULA, K.; OHLBERGER, M. Inflow-Implicit/Outflow-Explicit Scheme for Solving Advection Equations. *Springer Proceedings in Mathematics*. 2011, vol. 4. ISBN 978-3-642-20670-2. Available from DOI: 10.1007/978-3-642-20671-9\_72.
10. MIKULA, K.; REMEŠÍKOVÁ, M.; SARKOCI, P.; ŠEVČOVIČ, D. Manifold Evolution with Tangential Redistribution of Points. *SIAM Journal on Scientific Computing*. 2014, vol. 36, no. 4, A1384–A1414. Available from DOI: 10.1137/130927668.
11. MIKULA, K.; URBÁN, J. New fast and stable Lagrangean method for image segmentation. In: 2012. Available from DOI: 10.1109/CISP.2012.6469852.
12. MIKULA, K.; URBÁN, J. A new tangentially stabilized 3D curve evolution algorithm and its application in virtual colonoscopy. *Advances in Computational Mathematics*. 2014, vol. 40. Available from DOI: 10.1007/s10444-013-9328-x.
13. MIKULA, K.; URBÁN, J.; KOLLÁR, M.; AMBROZ, M.; JAROLÍMEK, I.; SIBIK, J.; ŠIBÍKOVÁ, M. An automated segmentation of NATURA 2000 habitats from Sentinel-2 optical data. *Discrete & Continuous Dynamical Systems - S*. 2021, vol. 14, no. 3, pp. 1017–1032.
14. MIKULA, K.; ŠEVČOVIČ, D.; BALAŽOVJECH, M. A Simple, Fast and Stabilized Flowing Finite Volume Method for Solving General Curve Evolution Equations. *Communications in Computational Physics*. 2008, vol. 7. Available from DOI: 10.4208/cicp.2009.08.169.

15. RAMER, U. An iterative procedure for the polygonal approximation of plane curves. *Computer Graphics and Image Processing*. 1972, vol. 1, no. 3, pp. 244–256. ISSN 0146-664X. Available from DOI: [https://doi.org/10.1016/S0146-664X\(72\)80017-0](https://doi.org/10.1016/S0146-664X(72)80017-0).
16. RUFFHEAD, A.; WHITING, B. *Introduction to geodetic datum transformations and their reversibility*. School of Architecture, Computing and Engineering, University of East London, 2020. Available also from: <https://www.researchgate.net/publication/339887497>.
17. SETHIAN, J. A. *Level set methods and fast marching methods: evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science*. 2nd ed. Cambridge: Cambridge University Press, 1999. Cambridge monographs on applied and computational mathematics 3. ISBN 0521642043cased.
18. ÚGKK SR. *Airborne Laser Scanning and DTM 5.0 - about the project*. 2015. Available also from: <https://www.geoportal.sk/sk/zbgis/lis-dmr/o-projekte/>.
19. ÚGKK SR. *Airborne Laser Scanning and DTM 5.0*. 2019. Available also from: [https://www.geoportal.sk/en/zbgis/als\\_dmr/](https://www.geoportal.sk/en/zbgis/als_dmr/).
20. ŠEVČOVIČ, D.; YAZAKI, S. Evolution of plane curves with a curvature adjusted tangential velocity. *Japan Journal of Industrial and Applied Mathematics*. 2011, vol. 28, pp. 413–442. Available from DOI: 10.1007/s13160-011-0046-9.