Matching 3D surfaces using Wasserstein distance

Ewa Bednarczuk[2] Rafał Bieńkowski[2] Robert Kłopotek[1] Jan Kryński[3] Krzysztof Leśniewski[2] Krzysztof Rutkowski[2] Małgorzata Szelachowska[3]

- **Introduction:** The Earth features are described by a set of discreet points which coordinates are necessary to be determined. Before the era of GPS technology, the horizontal coordinates of such points were usually read from maps, which was associated with their low accuracy. Other quantities were measured at these points with an accuracy, quite high even for the present time. The improvement of the horizontal coordinates of such points by using computational algorithms and additional available information leads to obtain much valuable data sets without the necessity of repeating long lasting and expensive measurement campaigns.
- **Motivation:** In 1957–1979 the whole territory of Poland was roughly homogeneously covered with gravity stations at which gravity was measured. The set of over 1 million gravimetric points surveyed at that time is up to now extremely valuable and unique, and it is widely used in Earth sciences, especially in geophysics, geodesy, and geology. The locations of gravity stations were determined from printed maps. Thus, it can be expected that the errors of horizontal coordinates of gravimetric stations are over hundred meters. In the case of the gravimetric points, for which the slopes of the terrain in their immediate vicinity exceeded 6 degrees, the slopes were measured with an inclinometer in four or, in the case of a more complex topography, in eight directions in the radius of 100 m (Kryński, 2007). In recent decades, the information about the terrain heights has become available in the form of digital terrain models (DTM). The DTM is a discrete representation of the terrain surface in the grid form. Within this investigation, the DTED2 model was used (NGA, 1996).
- **Goal:** The aim of this research is to match 3D surfaces, i.e. first one represented by the digital terrain model and the second one in the form of the so-called crosses simulating land slopes measured in four directions around the gravity station, using approximations of Wasserstein distance.
- **Main tool:** From the mathematical point of view the problem reduces to comparing surfaces. We will discretize both surfaces and treat them as discrete measures. Our main tool is the Wasserstein distance, also known as Monge-Kantorovich-Rubinstein distance, earth-mover's distance or optimal transport distance. For example, *"we want to transport goods between producers and consumers, whose respective spatial distributions are modelled by probability measures. The further producers and consumers are from each other, the more difficult will be our job"* (2018, Villani).

Main tool: Optimal transport (Wasserstein distance) (1)

Problem: Imagine we have a pile of sand (represented as measure μ). What would be an efficient way to move all of that mass into a second pile of sand (represented as measure ν)?

Fig. 1: Picture generated by Adobe Firefly AI

Let μ, ν be given probability measures. Let $\Pi(\mu,\nu)$ be the set of all joint probability measures on $\mathbb{R}^d \times \mathbb{R}^d$ whose marginals are μ, ν, i.e. for all subset $B \subset \mathbb{R}^d$ we have $\pi(B \times \mathbb{R}^d) = \mu(B)$ and $\pi(\mathbb{R}^d \times B) = \nu(B)$.

Wasserstein distance (for $p = 1$) is given by :

$$W(\mu,\nu) = \inf_{\pi \in \Pi(\mu,\nu)} \int d(x,y)\, d\pi(x,y).$$

We can say that Wasserstein is considered as a "horizontal" distance (Santambrogio, 2015). For discrete measures, i.e. $\mu = \sum_{i=1}^{n} \alpha_i \delta_{x_i}, \nu = \sum_{i=1}^{n} \beta_i \delta_{y_i}$, formula W simplify to linear problem:

$$W(\mu,\nu) = \min_{\Pi} \left(\underbrace{\sum_{i,j} \Pi_{i,j} d(x_i, y_j)}_{work} : \Pi_{i,j} \geq 0, \sum_i \Pi_{i,j} = \beta_j, \sum_j \Pi_{i,j} = \alpha_i \right).$$

Visualization of optimal transport (2)

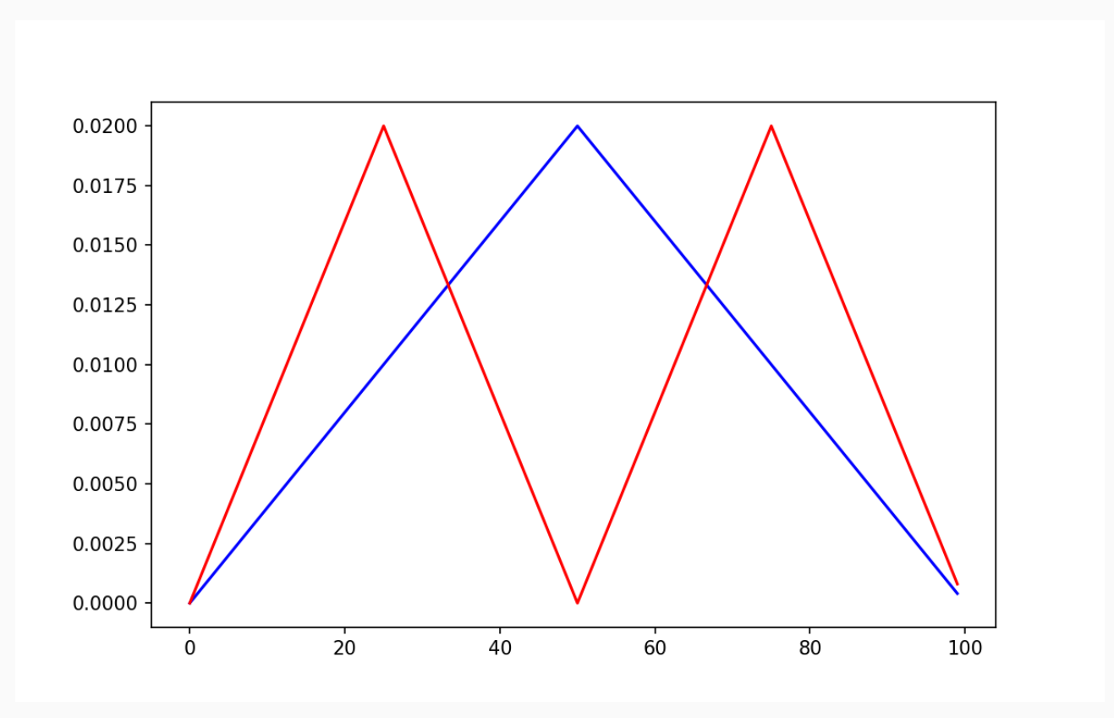

Fig. 2: Two continuous measures

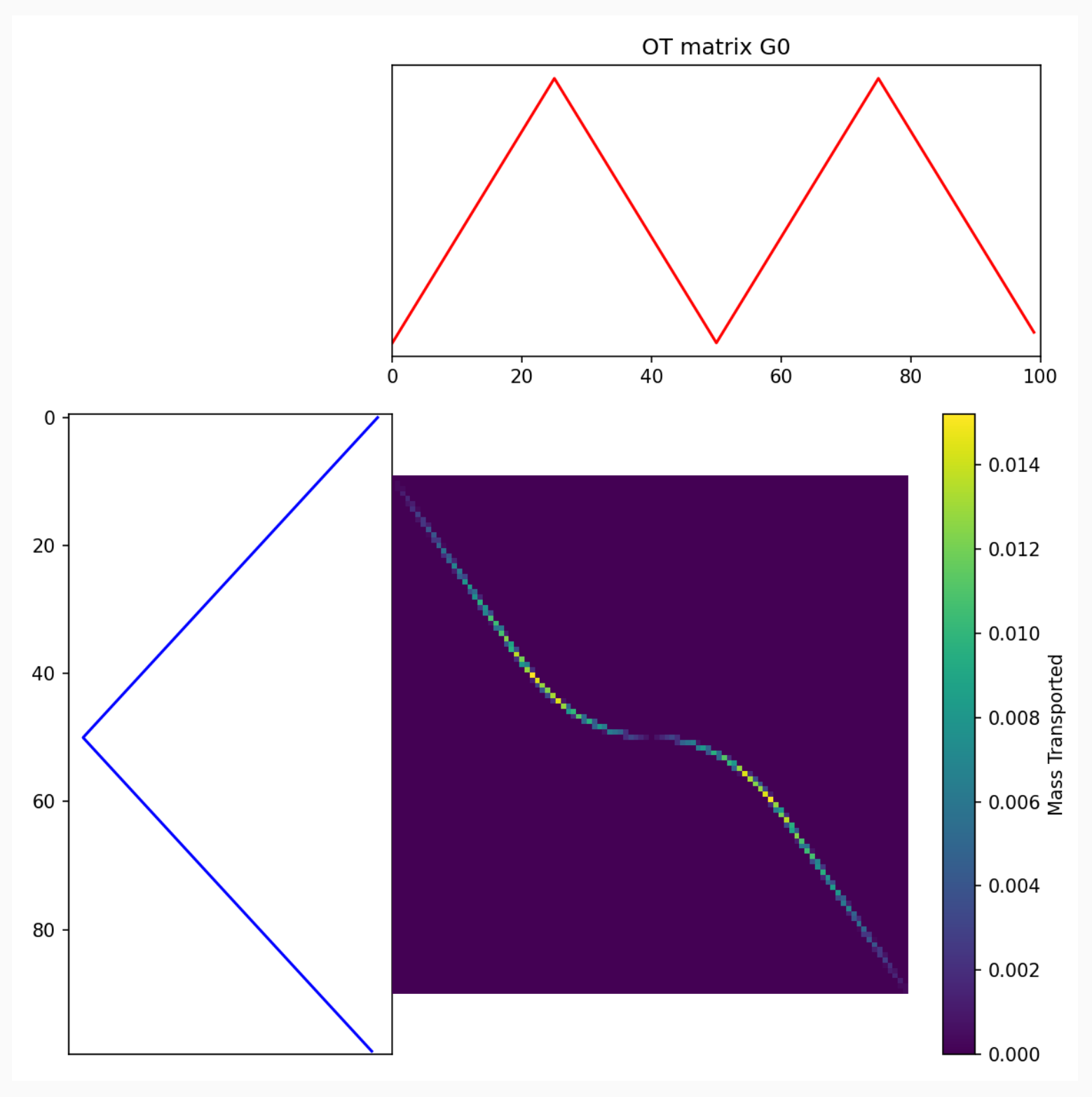

Fig. 3: Optimal transport, lighter color means more mass was transported

An example using the POT library (Flamary R., 2021). Fig. 3 provides visualizes the OT (Optimal Transport) matrix G0, which represents the optimal transportation plan between two distributions *red* and *blue* (Fig. 2). Each cell in the plot corresponds to the transportation amount between an element in *red* and an element in *blue*. The color intensity in each cell indicates the amount of mass being transported - lighter color means more mass was transported.

Goal: Matching two 3D surfaces (3)

First, we calculate Wasserstein distances for each set of crosses (160 points given in 4 perpendicular directions) on the DTM with approx 900 000 points. We are using "*scipy.stats.wasserstein_distance*" given in (SciPy Wasserstein)

Second, we are using linear assignment programming (SciPy linear) with the cost given by Wasserstein distances, i.e.

$$\min \sum_i \sum_j C_{i,j} X_{i,j},$$

where C is cost matrix (size of 160×900000) and $X[i,j] = 1$ iff row i is assigned to column j.

contact: lesniews@ibspan.waw.pl
[1]Institute of Computer Science, Cardinal Stefan Wyszynski University in Warsaw, Poland
[2]Systems Research Institute, Polish Academy of Sciences, Warsaw, Poland
[3]Institute of Geodesy and Cartography, Warsaw, Poland

Results (4)

We want to find *magenta* crosses, solutions are given by *green* points. Wasserstein distance cost matrix is computed about 20 times slower than Euclidean distance cost matrix, but the results have similarly bad solution location (*green* points).

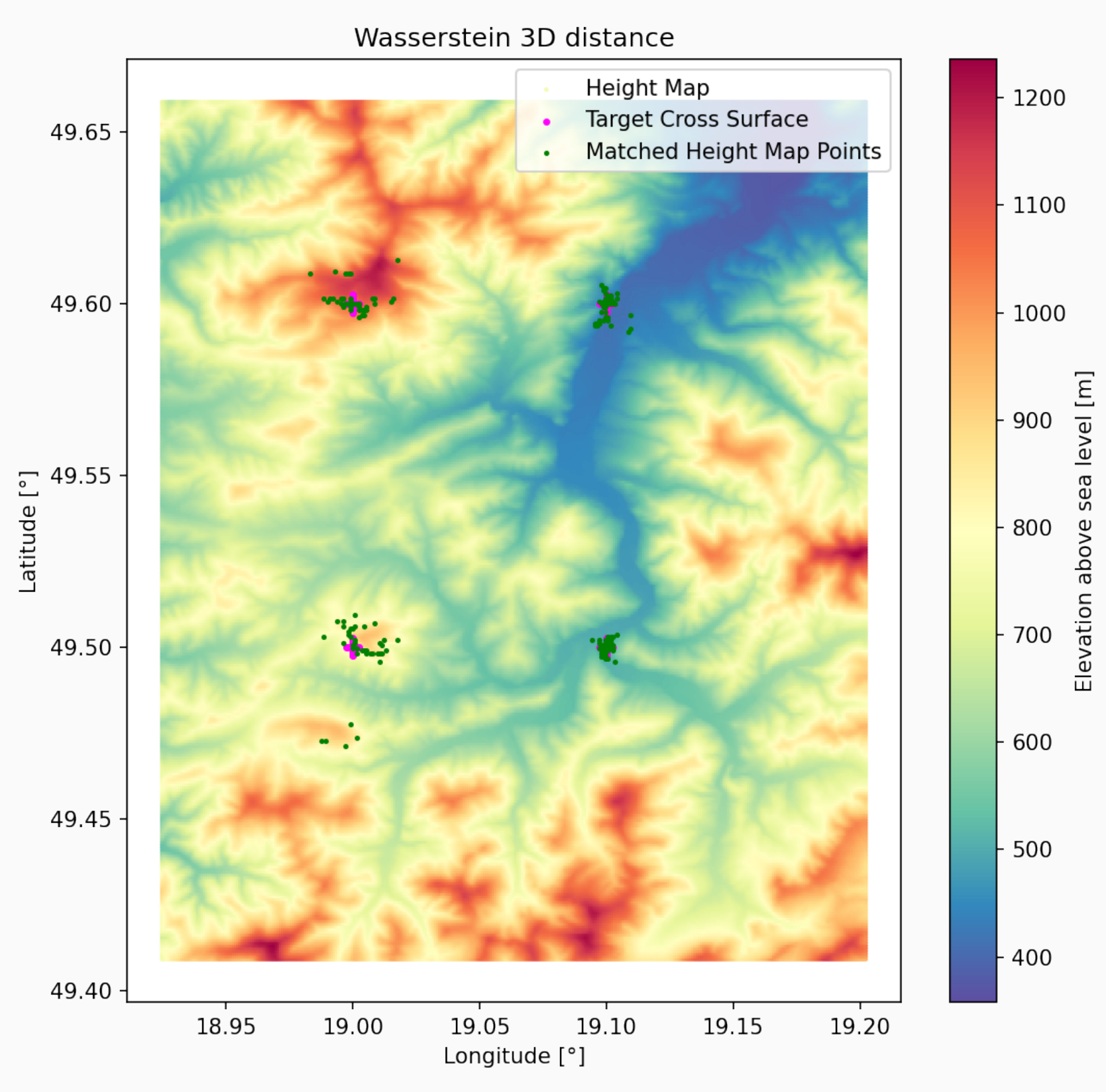

Fig. 4: Assignment programming with Wasserstein distance cost matrix

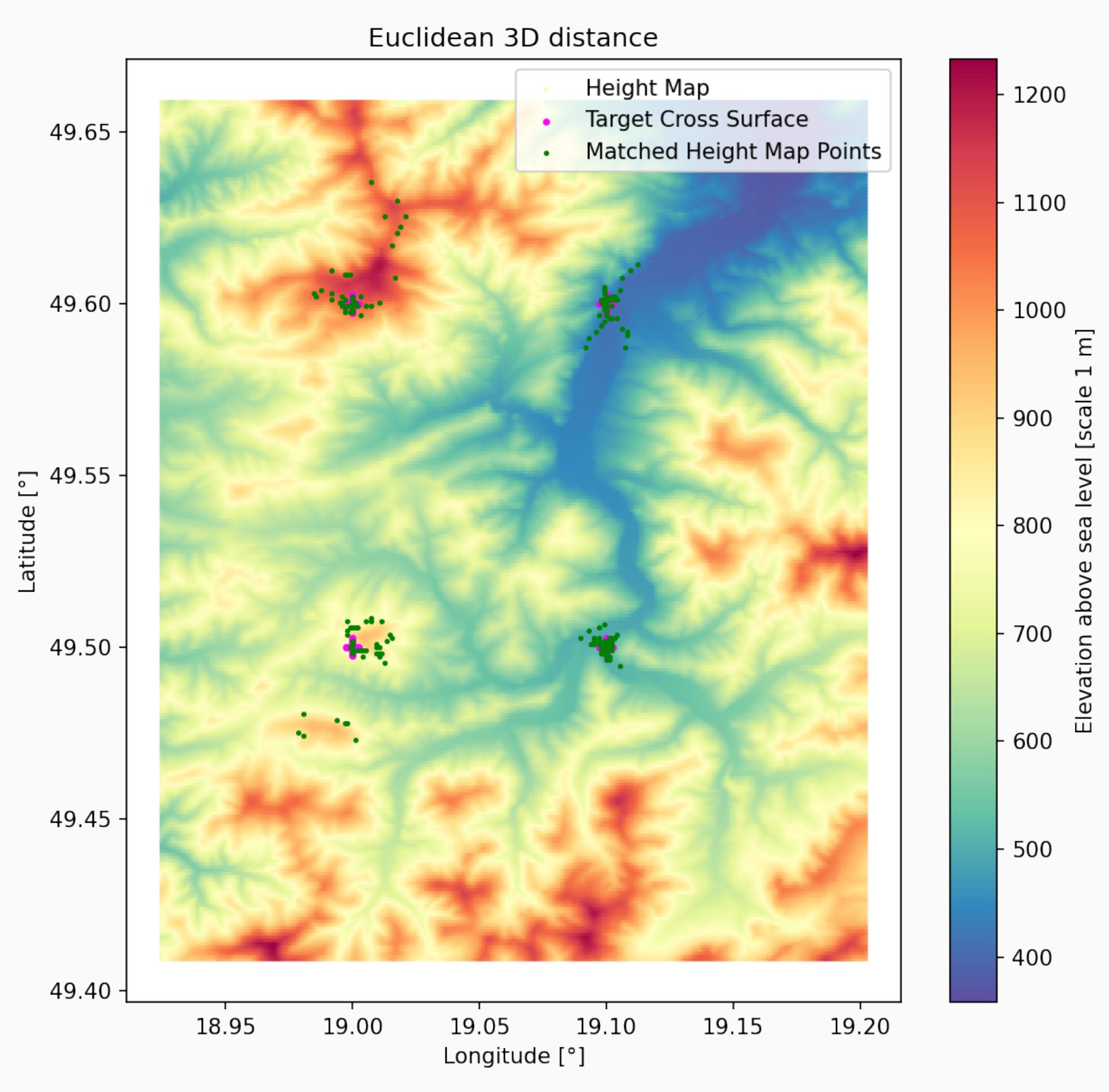

Fig. 5: Assignment programming with Euclidian distance cost matrix

To simulate a data from field work we use a DTM of spatial resolution $300m \times 300m$. Based on this data crosses have been calculated. For our solution we used a DTM of spatial resolution $30m \times 30m$. The test area is located near Bielsko-Biała in Poland.

Scaling of height (5)

Due to the much faster Euclidean distance cost matrix, now we look at some improvements to our method. Looking at numerical values of longitude and latitude, those are more or less in the (0;0.3) range while the height is in the (0;1000) range, so the solution fit is more likely to be skewed to minimize the height first (greater differences on z axis), and the other coordinates later. Below you can see results with linear assignment programming with Euclidean distance as a cost. We downscaled the z axis (height) by fixed values: 100 (Fig. 6) and 10000 (Fig. 7). Scaling by 100 gives the same results as in Fig. 5. Scaling by 10000 to the same range as longitude and latitude gives much better results of solution location (*magenta* crosses are covered by *green* solution points).

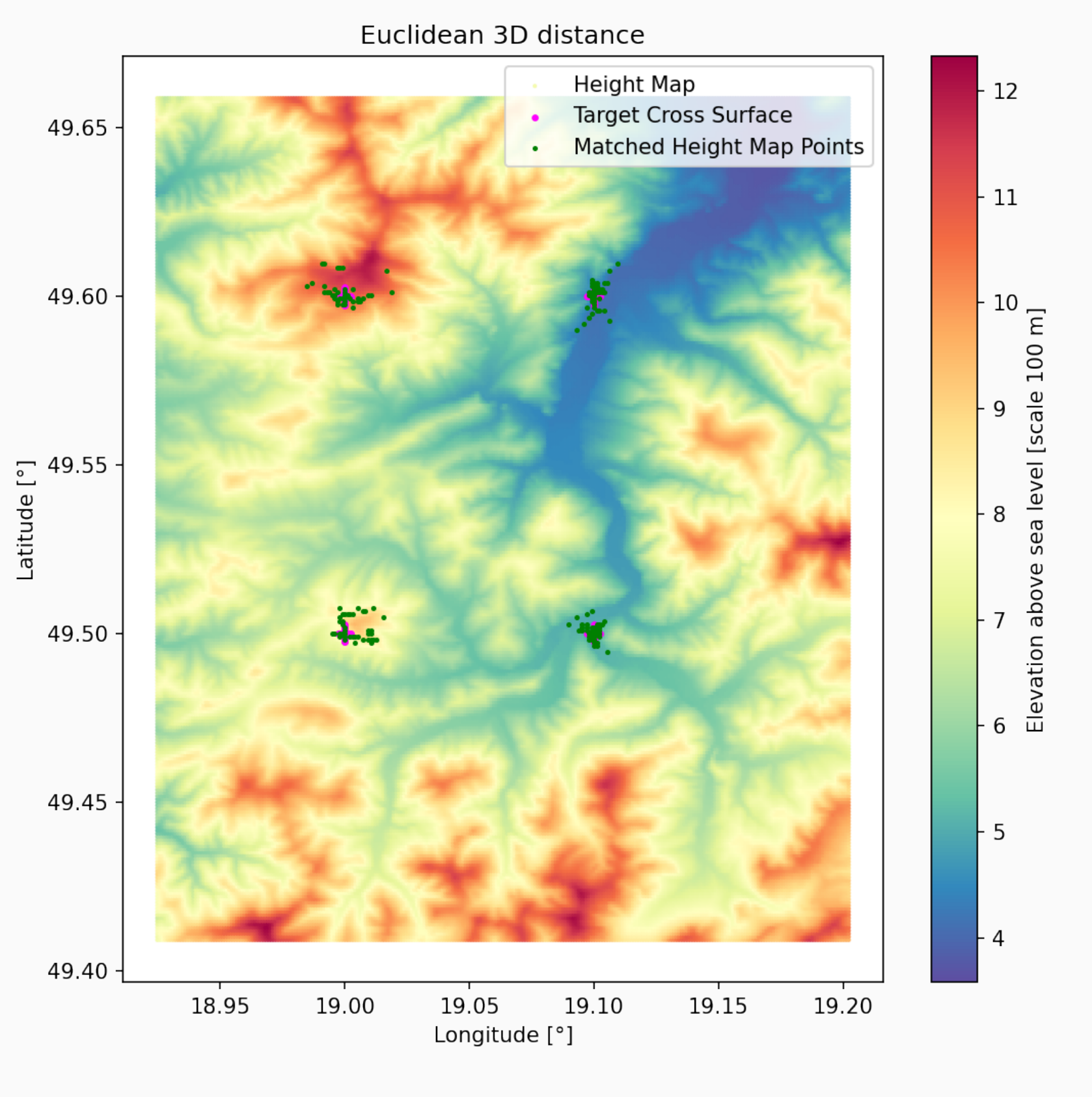

Fig. 6: Height (z axis) downscaled by 100

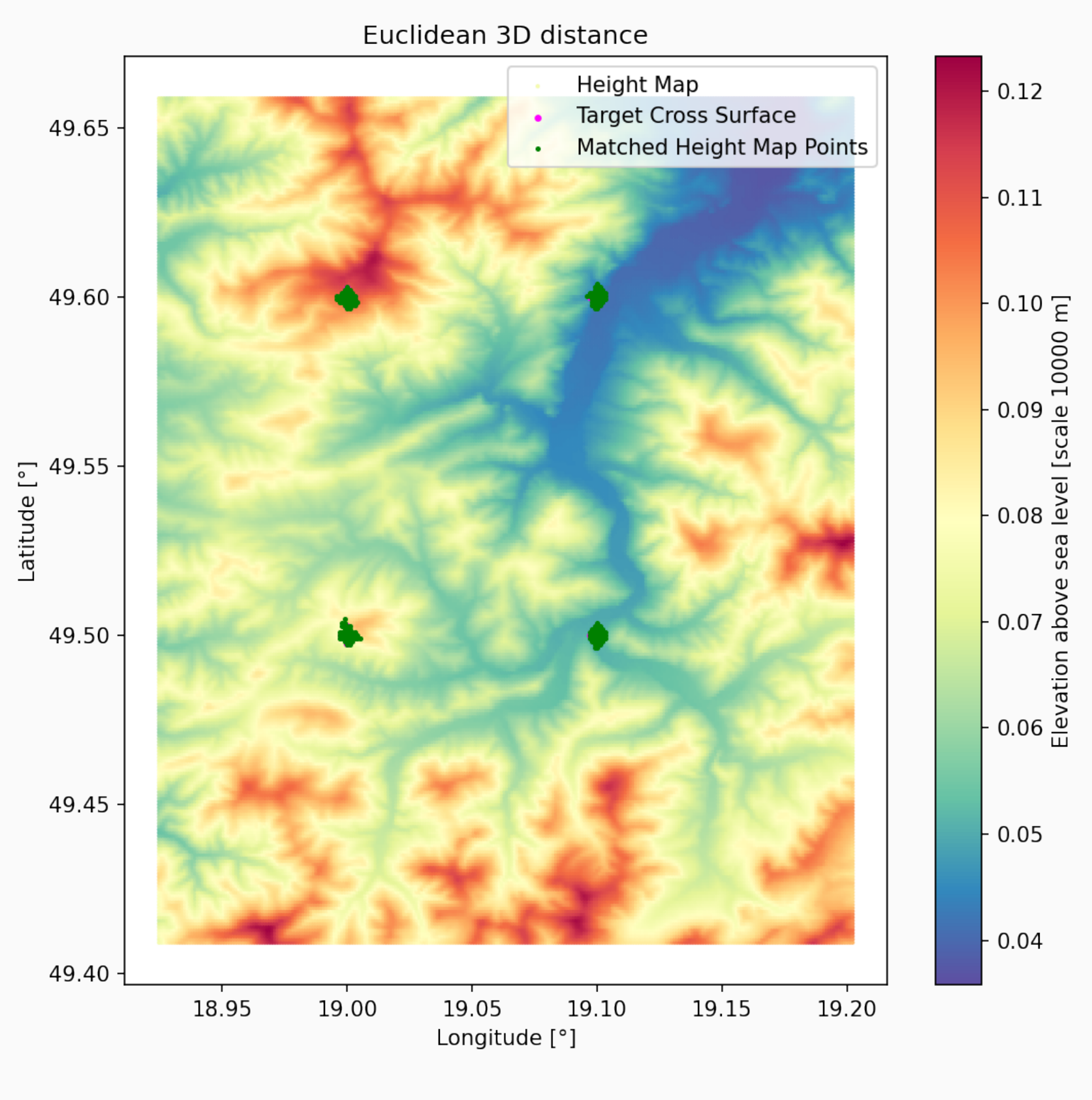

Fig. 7: Height (z axis) downscaled by 10000

Future work: In the future it is planned to apply the approach of the linear assignment programming with the cost given by Wasserstein distances for improving the horizontal coordinates of gravimetric points measured in 1957–1979 using information about terrain slopes around those points and the modern digital terrain models.

Kryński J., (2007): Precyzyjne modelowanie quasigeoidy na obszarze Polski - wyniki i ocena dokładności, Monographic series of the Institute of Geodesy and Cartography, Nr 13, Warsaw 2007, (266 pp).
NGA, (1996): Performance specification Digital Terrain Elevation Data (DTED), National Geospatial Intelligence Agency, Document MIL-PRF-89020A.
Flamary R. ,(2021): "POT: Python Optimal Transport". In: Journal of Machine Learning Research 22.78 (2021), pp. 1–8. `http://jmlr.org/papers/v22/20-451.html`
Santambrogio, 2015: Optimal Transport for Applied Mathematicians, Progress in Nonlinear Differential Equations and Their Applications, Springer
Villani, 2008: Optimal transport, old and new, Springer
SciPy Wasserstein distance v.1.10.1 `https://docs.scipy.org/doc/scipy-1.10.1/reference/generated/scipy.stats.wasserstein_distance.html`
SciPy linear sum assignment solver v.1.10.1 `https://docs.scipy.org/doc/scipy-1.10.1/reference/generated/scipy.optimize.linear_sum_assignment.html`