Research Journal of Agricultural Science, 57 (4), 2025; ISSN: 2668-926X

http://doi.org/10.59463/RJAS.2025.4.23

DEVELOPEMENT OF A MULTIFUNCTIONAL TOPOGRAPHIC SOFTWARE USING OPEN-SOURCE RESOURCES

Vladimir IORDACHE[1] , Andrei ILIE[1]
[1]Technical University of Civil Engineering of Bucharest, Faculty of Geodesy
Corresponding author: vladimir.iordache@yahoo.com

Abstract. *This paper showcases the development and use of a multifunctional topographic software, which is entirely written and implemented in Python, using only open-source libraries and resources. The main objective of this paper is to create a reliable, flexible and fast tool that is able to transform the coordinates of a set of points into a new CRS (Coordinate Reference System) based on a set of control points (often called GCP), using various methods and algorithms, like the Gauss–Markov stochastic functional model for indirect observations. Beyond the aforementioned transformations, the present program is able to calculate and generate the 3D digital terrain model (DTM) from the elevation data of the provided set of points and then produce contour lines at a given interval by means of Delaunay triangulation, which can then be exported to DXF format.Currently, the market of such applications may appear oversaturated, but in reality, the most popular solutions require a substantial investment and often rely on proprietary solutions and algorithms, which may or may not be well documented. In contrast, the solution presented in this paper enhances reproducibility and transparency of the results by using open-source solutions and a versatile programming language, Python.The methodology involves the use of various numerical transformation techniques for coordinate compensation, geometric algorithms for coordinate transformation, interpolation methods for generating the DTM and others.The results obtained using the software demonstrate that the program can be, in various aspects, comparable to those obtained with commercial tools, and in some even better, the reason for this being the fact that the software was specifically designed with topographic situations in mind.*
The implications of this work are relevant, because it provides a transparent, well-documented and extensible alternative to popular commercial applications. This software may be unique, as it implements many different concepts from various fields that are close to topography, like GIS, photogrammetry, statistics, machine learning etc., within a full Python framework.

Keywords*: Python, CRS, DTM, Gauss-Markov, Delaunay, DXF, GIS, contour*

INTRODUCTION

Topography and geospatial data processing and manipulation play a fundamental role in engineering and in every one of its subdomains. Coordinate transformations, which may be performed manually or automatically, generation of DTMs and the subsequent contour lines represent key steps in almost every surveying workflow and project. This highlights the need for modern software solutions, which tackle these concepts in different and unique ways.
Despite the availability of numerous solutions, most applications are proprietary, expensive, resource-intensive, undocumented or nontransparent, and in some cases, even downright intrusive. The lack of open-source tools in this domain represents a significant gap in the field.
To address this limitation, this paper presents the development of a multifunctional topographic software fully written in Python, named TopoPro, which uses only open-source libraries and resources. This software resembles GIS applications like QGIS, as it tries to help visualize all the operations performed by implementing an interactive canvas and drawing points, lines and polygons that correspond to the input/modified data. To assist this, the program also uses an AutoCad-inspired layer system, which groups the vector data quickly and efficiently.
All of the aforementioned components were designed to assure compatibility with the DWG/DXF standard, which enables the export of all features and results from the program to a

http://doi.org/10.59463/RJAS.2025.4.23

CAD environment. In this sense, this paper showcases a program that aids a typical surveying workflow by implementing most of the functions and data processing tools needed prior to working with the data in a CAD environment.
By combining all of the previously mentioned aspects within a single Python framework, the software provides a transparent, extensible, and cost-effective alternative to commercial tools, while ensuring reproducibility and adaptability to various topographic use cases.

MATERIAL AND METHODS

In this section the main functions and methods of the TopoPro software will briefly be explained. The following explanations will include all the necessary steps and references needed in order to reproduce the obtained solutions. Also, some algorithms will be presented in a detailed manner.
Before going further, the figure below shows the main logic of the application and its main functions as objects (rectangles).

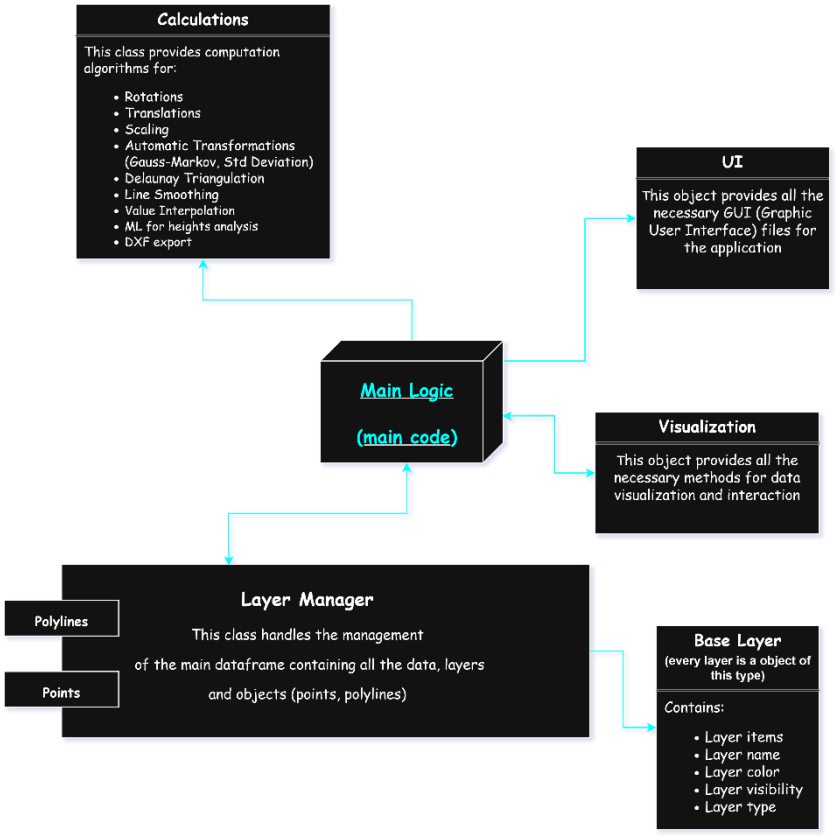


Figure 1. Main Logic of the Application

http://doi.org/10.59463/RJAS.2025.4.23

The application is based on the imported CSV data, which must contain the following fields in any order: Point ID, North Coordinate, East Coordinate, Height and optionally Point Code. Two different CSV files are required: one for the local coordinates (source CRS) and one for the ground control points (GCPs) in the destination CRS, although most functions can also operate with only the source dataset.

After the import, a central Pandas Dataframe is created that stores all the data. A DataFrame is a two-dimensional data structure, similar to a table with rows and columns, which allows storing heterogeneous information about the dataset (*10 minutes to pandas — pandas 2.3.3 documentation* [no date]; *pandas - Python Data Analysis Library* [no date]).

The imported dataset can then be edited through the CSV Edit function. Using this function, all the data and metadata associated with the local (source CRS) and national (destination CRS) coordinates can be modified according to our needs. In addition, it provides tools for searching points by any field, exporting data as a CSV file, manually adding points, assigning layers, and performing height analysis.

One of the most interesting function is the height analysis, which combines several machine learning algorithms to detect potential outliers in the heights dataset that may significantly influence the 3D model. In total, there are 6 used algorithms that try to detect possible outliers. A point will be considered an outlier if it was detected by at least 3 algorithms.

The used algorithms can be found in the Scikit-Learn and SciPy libraries.(*SciPy* [no date]; *scikit-learn: machine learning in Python — scikit-learn 1.8.0 documentation* [no date]) These are:

Linear Regression
Isolation Forest
Gaussian Process Regression
Nearest Neighbors
Decision Tree Regression
Z-Score (Standard Deviation)

Analiza cotelor punctelor suspecte

Numar total puncte: 235
Deviatie standard cote: 5.341
Numar total puncte suspecte: 79
Numarul punctelor dupa eliminarea celor detectate o data sau de 2 ori: 39
Număr puncte detectate prin regresie liniara: 34
Număr puncte detectate prin Isolation Forest: 36
Număr puncte detectate prin Gaussian Process Regression: 23
Număr puncte detectate prin K-Nearest Neighbor: 64
Număr puncte detectate prin Decision Tree: 45
Număr puncte detectate prin eliminarea valorilor aberante (95%): 33

Numarul total de detectari pentru puncte

ID	N	E	Z	lin_reg	iso_f	gp	knn	dtree_reg	zscore	total_detectari
33	5011.262	3011.122	466.402	1	1	1	1	1	1	6
34	5024.935	3017.935	466.563	1	1	1	1	1	1	6
57	4977.021	2993.442	463.766	1	1	1	1	1	1	6
58	4959.067	3004.771	464.148	1	1	1	1	1	1	6
81	4989.828	2936.778	466.015	1	1	1	1	1	1	6
86	5017.328	3014.192	469.401	1	1	1	1	1	1	6
89	5052.432	3006.335	461.879	1	1	1	1	1	1	6
79	5020.334	2931.304	462.401	1	1	0	1	1	1	5
80	4998.323	2920.274	462.518	1	1	0	1	1	1	5
146	5031.327	2936.909	461.597	1	1	0	1	1	1	5
87	5035.647	3023.142	466.848	1	1	1	1	0	1	5
84	5022.828	2989.501	463.567	1	0	1	1	1	1	5
85	5017.694	3003.409	463.737	1	0	1	1	1	1	5
56	4998.672	2999.219	463.763	1	0	1	1	1	1	5
11	5009.963	2996.766	463.358	1	0	1	1	1	1	5
158	4984.197	2975.222	463.616	1	0	1	1	1	1	5
184	5060.118	2972.509	462.607	1	1	1	1	0	1	5
165	4976.485	2971.447	465.896	1	1	0	1	1	1	5
161	4982.472	2971.878	463.623	1	0	1	1	1	1	5
163	4976.467	2976.128	471.356	1	1	0	1	1	1	5
159	4983.902	2973.327	463.608	1	0	1	1	1	1	5
144	5020.355	2931.233	461.561	1	1	0	1	1	1	5

Figure 2. Heights Analysis Report

http://doi.org/10.59463/RJAS.2025.4.23

After the detection process is finished, the program generates a report that lists all points identified as outliers by three or more algorithms, together with the methods that flagged them. An example of such a report can be seen in the following figure:
After importing all the required data, the user can proceed with the coordinate transformation. This can be done manually, by using the provided translation, rotation and scaling functions or automatically by applying one of the provided transformation models.
The first model, referred to as the **Simple Model**, represents a custom implementation developed within this project, while the second one is an implementation of the **Gauss-Markov** algorithm. The Simple Model operates in all three dimensions (N, E, Z), whereas the Gauss–Markov implementation processes only the planar coordinates (N, E). Both of these models require the correct correspondence between the local and national points. To determine the accurate correspondence we can either specify the ID of the corresponding local point in the "Statie" column of the national dataset or by assigning the corresponding IDs as point codes or point names for the national coordinate file.
The translation, rotation and scaling functions are provided by the Shapely library, which is a Python package for manipulation and analysis of planar geometric objects.
Furthermore, the 2 provided transformation models will be briefly presented below.
The Simple Model automatically identifies the most stable reference pair of points based on the maximum horizontal distance, computes the translation vector, estimates the average orientation difference while removing outliers using statistical filtering, and finally applies a rotation around the most reliable reference point. This approach ensures geometric consistency between the two coordinate systems and improves the robustness of the transformation in practical topographic use cases.

This model proposes the following:
Let the corresponding points in the two coordinate systems be defined as:

$$P_i = (N_i, E_i, Z_i), \qquad P'_i = (N'_i, E'_i, Z'_i), \qquad for\ i = 1, 2, \dots, n$$

where:
P_i are the coordinates in the local (source) system
P'_i are the coordinates in the national (target/GNSS) system
For each pair of corresponding local points (P_i, P_j), the horizontal distance is computed as:

$$d_{ij} = \sqrt{(N_j - N_i)^2 + (E_j - E_i)^2}$$

Then:

$$d_{max} = max(d_{ij})$$

The point P_{r2} of the selected reference pair (longest distance/baseline) (P_{r1}, P_{r2}), corresponding to the second point of the longest baseline, is then used as the reference point for translation and rotation.
The translation vector between the two systems is determined as:

$$\Delta N = E'_{r2} - E_{r2}$$
$$\Delta E = E'_{r2} - E_{r2}$$

http://doi.org/10.59463/RJAS.2025.4.23

$$\Delta Z = Z'_{r2} - Z_{r2}$$

All local points are translated according to:

$$\begin{pmatrix} N_t \\ E_t \\ Z_t \end{pmatrix} = \begin{pmatrix} N \\ E \\ Z \end{pmatrix} + \begin{pmatrix} \Delta N \\ \Delta E \\ \Delta Z \end{pmatrix}$$

Following this, for every pair of corresponding points (P_i, P_j) and (P'_i, P'_j), the line orientation angles are computed as:

$$\theta_{ij} = \arctan\left(\frac{E_j - E_i}{N_j - N_i}\right)$$

$$\theta'_{ij} = \arctan\left(\frac{E'_j - E'_i}{N'_j - N'_i}\right)$$

The orientation difference is:

$$\Delta\theta_{ij} = \theta'_{ij} - \theta_{ij}$$

The mean orientation difference is:

$$\Delta\bar{\theta} = \frac{1}{n} * \Sigma\,(\Delta\theta_{ij})$$

Outliers are removed using the Z-score method:

$$Z_{ij} = \frac{\Delta\theta_{ij} - \Delta\bar{\theta}}{s\theta}$$

where:

$s\theta$ is the standard deviation of the orientation differences

Furthermore, values with $|Z_{ij}| <$ threshold are kept, and the final rotation angle is computed as:

$$\bar{\theta}_{final} = \frac{1}{m} * \Sigma\,(\Delta\theta_{ij}), \qquad (for\ |Z_{ij}| < threshold)$$

In the end, the translated points are rotated around the reference point P'_{r2} using the angle $\bar{\theta}_{final}$:

$$\begin{pmatrix} E_r \\ N_r \end{pmatrix} = \begin{pmatrix} cos(\bar{\theta}_{final}) & -sin(\bar{\theta}_{final}) \\ in(\bar{\theta}_{final}) & cos(\bar{\theta}_{final}) \end{pmatrix} * \begin{pmatrix} E_t - E'_{r2} \\ N_t - N'_{r2} \end{pmatrix} + \begin{pmatrix} E'_{r2} \\ N'_{r2} \end{pmatrix}$$

The final transformed coordinates in the destination CRS are:

http://doi.org/10.59463/RJAS.2025.4.23

$$\begin{pmatrix} N_f \\ E_f \\ Z_f \end{pmatrix} = \begin{pmatrix} N_r \\ E_r \\ Z_t \end{pmatrix}$$

The Gauss-Markov Model is used to determine the four parameters of a conformal transformation, where at least two common points between the local and national coordinate systems are required. When the number of common points exceeds the minimum necessary, the parameters of the transformation are estimated using the least squares adjustment method, which minimizes the sum of the squared residuals between observed and computed coordinates. In this context, one of the most frequently applied approaches is the Gauss–Markov model, also known as the method of indirect observations. In this model, only the coordinates of the points in the national (target) coordinate system are treated as observations, while the local (source) coordinates are assumed to be error-free.
Consequently, the stochastic component of the model considers that only the observed coordinates are affected by random errors — an assumption that, although mathematically convenient, does not always fully correspond to real measurement conditions.(Ionescu 2005)
The coordinates of the points in the national system are affected by measurement errors. To address this, we introduce small quantities called corrections:

$$v_{Xi} = a * x_i - b * y_i + c - X_i$$
$$v_{Yi} = b * x_i + a * y_i + d - Y_i$$

To determine the transformation parameters, the least squares method is applied:

$$[vv] \rightarrow min$$

The aforementioned corrections system is:

$$V = A * \Delta + l$$

where:

$$A_{(2p,4)} = \begin{pmatrix} x_1 & -y_1 & 1 & 0 \\ y_1 & x_1 & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ x_i & -y_i & 1 & 0 \\ y_i & x_i & 0 & 1 \end{pmatrix}$$

$$V_{(2p,1)} = \begin{pmatrix} V_{X1} \\ V_{Y1} \\ \vdots \\ V_{Xi} \\ V_{Yi} \end{pmatrix}$$

http://doi.org/10.59463/RJAS.2025.4.23

$$\Delta_{(4,1)} = \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix}$$

$$l_{(2p,1)} = \begin{pmatrix} -X_1 \\ -Y_1 \\ \vdots \\ -X_i \\ -Y_i \end{pmatrix}$$

For estimating the transformation parameters, the Gauss–Markov adjustment model is applied:

$$N = A^T * A$$

$$\Delta = -N^{-1} * A^T * l$$

From the adjusted parameters, the translation, rotation, and scale can be derived as follows:

$$T_X = c$$

$$T_Y = d$$

$$\varphi = arctan\left(\frac{b}{a}\right)$$

$$m = \sqrt{a^2 + b^2}$$

Following this, the obtained precision can be estimated:

$$S = \pm\sqrt{\frac{(V^T * V)}{(2p - 4)}}$$

$$S_a = S * \sqrt{N_{11}{}^{-1}}$$

$$S_b = S * \sqrt{N_{22}{}^{-1}}$$

$$S_c = S * \sqrt{N_{33}{}^{-1}}$$

$$S_d = S * \sqrt{N_{44}{}^{-1}}$$

Research Journal of Agricultural Science, 57 (4), 2025; ISSN: 2668-926X

http://doi.org/10.59463/RJAS.2025.4.23

The coordinates of non-common points in the national system are computed using the functional model.

To evaluate the precision of the transformed coordinates, the error propagation law is applied:

$$Q_{ff} = f^{T} * N^{-1} * f$$

$$S_f = \pm S * \sqrt{Q_{ff}}$$

where:

$$f^{T} = \begin{pmatrix} x_1 & -y_1 & 1 & 0 \\ y_1 & x_1 & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ x_i & -y_i & 1 & 0 \\ y_i & x_i & 0 & 1 \end{pmatrix}, \qquad for\ uncommon\ points$$

After the point transformation and analysis, the dataset is ready to be used as the basis for the 3D Model and contour lines generation. The software allows the creation of a Triangulated Irregular Network (TIN) using Delaunay triangulation, where triangles can be generated over the entire dataset or within a user-defined polygon, which can interactively be drawn.(*matplotlib.tri — Matplotlib 3.10.8 documentation* [no date])

The resulting DTM can then be interpolated using linear or cubic interpolation, enhancing the accuracy and density of the model.(Dinas, Martínez 2020)

Furthermore, to enhance the quality of the contours a Gaussian filter can be applied for smoothing (*(PDF) DSP -Gaussian Filtering in Image Filtering - Report* [no date]), and optionally, the curves can be further refined using NURBS (Non-Uniform Rational B-Splines), which grant fine control over the contour creation (JOHANNESSEN and FONN, 2020).

The last function that remains to be used is the Export DXF function, which writes all/only the specified data to a DXF file so that it can be used later in a CAD project/environment.

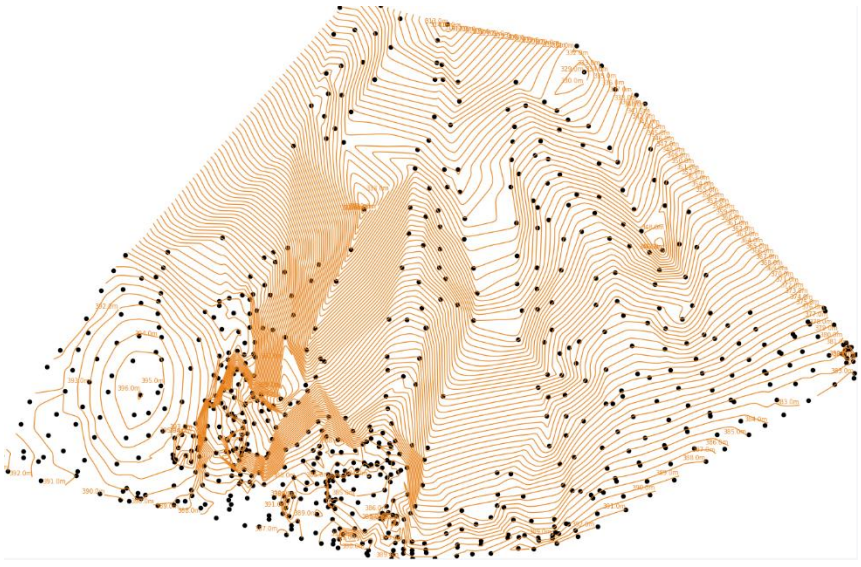


Figure 3. Example of generated contour lines using the Gaussian Filter

http://doi.org/10.59463/RJAS.2025.4.23

RESULTS AND DISCUSSIONS

After showcasing the main functions of the developed software, a typical workflow can also be presented.

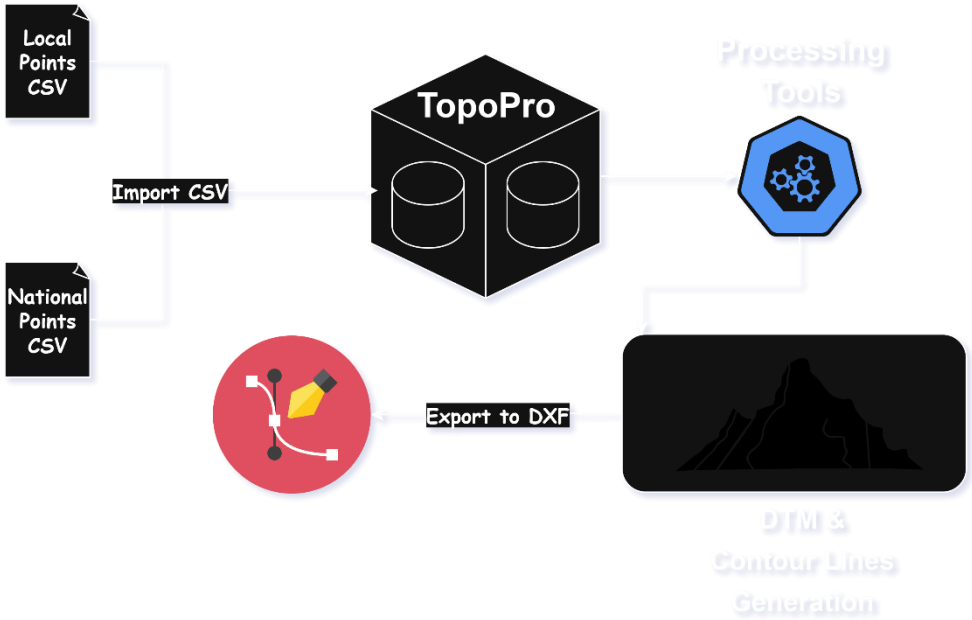


Figure 4. Typical Workflow

Following the figure from above, we can showcase the ease of use of the software by transforming a local dataset to a national one, presenting each step through a series of figures taken directly from TopoPro.

The present dataset stems from a real cadastre project and includes various points with wrong-determined heights, which can be reviewed using the Heights analysis function.

First, we import the local and national dataset (Statie, GNSS).

Research Journal of Agricultural Science, 57 (4), 2025; ISSN: 2668-926X
http://doi.org/10.59463/RJAS.2025.4.23

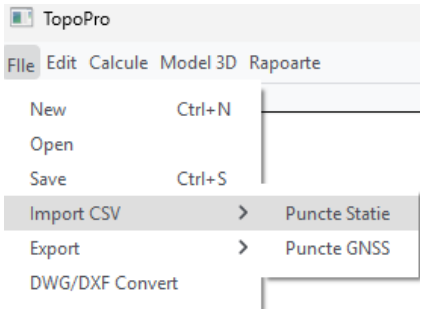


Figure 5. Import CSV function

After importing the local and national points the dataset will be displayed on the canvas.

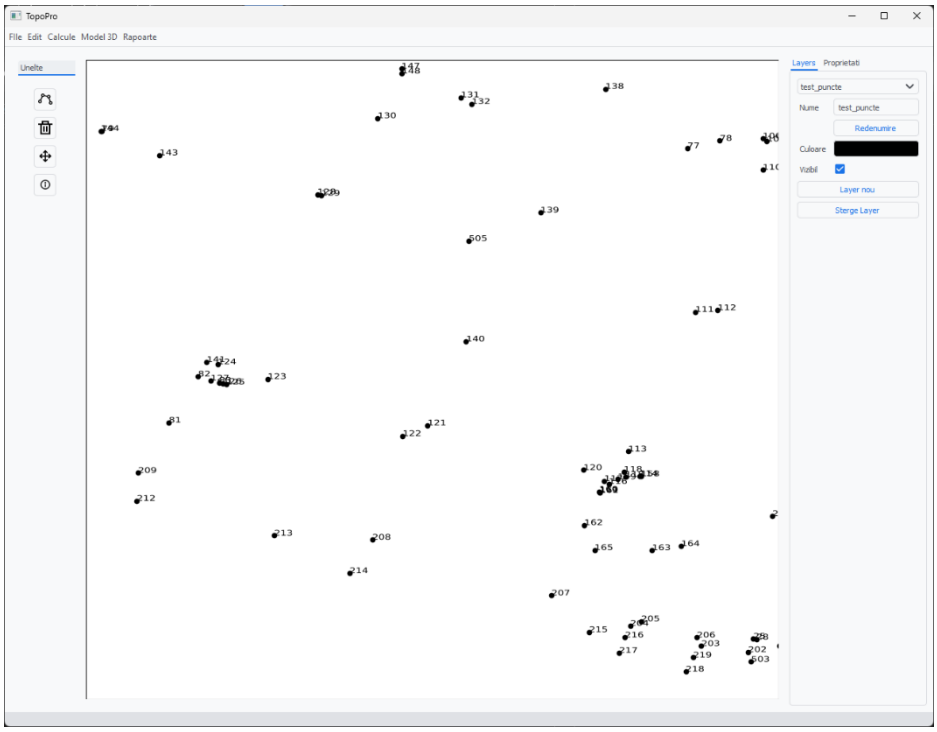


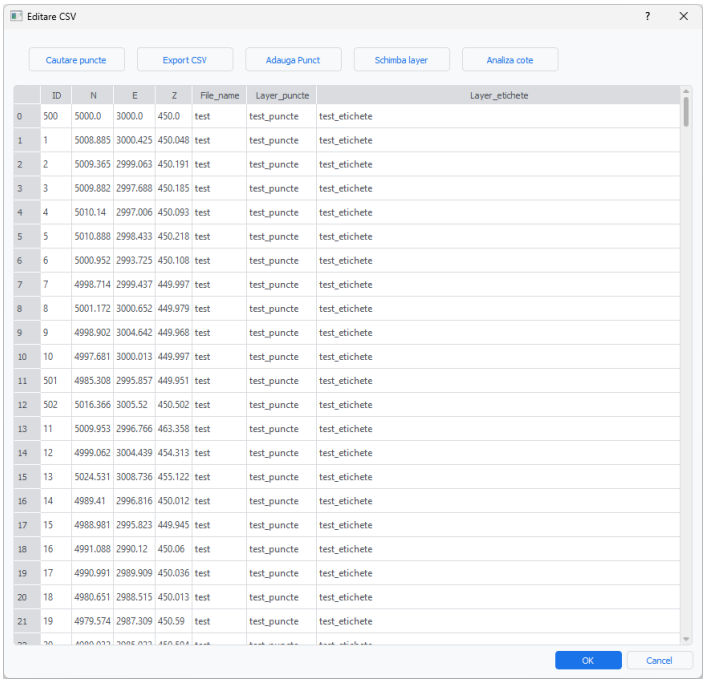
Figure 6. The imported dataset

http://doi.org/10.59463/RJAS.2025.4.23

Once the points are displayed on the canvas any function provided by the TopoPro software can now be called. In the following figure we can see the local imported dataset and all the associated functions that can be immediately applied to it.

Research Journal of Agricultural Science, 57 (4), 2025; ISSN: 2668-926X

http://doi.org/10.59463/RJAS.2025.4.23



	ID	N	E	Z	File_name	Layer_puncte	Layer_etichete
0	500	5000.0	3000.0	450.0	test	test_puncte	test_etichete
1	1	5008.885	3000.425	450.048	test	test_puncte	test_etichete
2	2	5009.365	2999.063	450.191	test	test_puncte	test_etichete
3	3	5009.882	2997.688	450.185	test	test_puncte	test_etichete
4	4	5010.14	2997.006	450.093	test	test_puncte	test_etichete
5	5	5010.888	2998.433	450.218	test	test_puncte	test_etichete
6	6	5000.952	2993.725	450.108	test	test_puncte	test_etichete
7	7	4998.714	2999.437	449.997	test	test_puncte	test_etichete
8	8	5001.172	3000.652	449.979	test	test_puncte	test_etichete
9	9	4998.902	3004.642	449.968	test	test_puncte	test_etichete
10	10	4997.681	3000.013	449.997	test	test_puncte	test_etichete
11	501	4985.308	2995.857	449.951	test	test_puncte	test_etichete
12	502	5016.366	3005.52	450.502	test	test_puncte	test_etichete
13	11	5009.953	2996.766	463.358	test	test_puncte	test_etichete
14	12	4999.062	3004.439	454.313	test	test_puncte	test_etichete
15	13	5024.531	3008.736	455.122	test	test_puncte	test_etichete
16	14	4989.41	2996.816	450.012	test	test_puncte	test_etichete
17	15	4988.981	2995.823	449.945	test	test_puncte	test_etichete
18	16	4991.088	2990.12	450.06	test	test_puncte	test_etichete
19	17	4990.991	2989.909	450.036	test	test_puncte	test_etichete
20	18	4980.651	2988.515	450.013	test	test_puncte	test_etichete
21	19	4979.574	2987.309	450.59	test	test_puncte	test_etichete

Figure 7. The imported local dataset

The next step involves the local dataset transformation. In this example we will use the Gauss-Markov model.

Figure 9. The transformed dataset

http://doi.org/10.59463/RJAS.2025.4.23

The last phase of this example is represented by the contour line generation. Using this dataset, the obtained DTM will be deficient, as many points included in the 3D model have an incorrect height, which can however be corrected through the Heights Analysis feature.

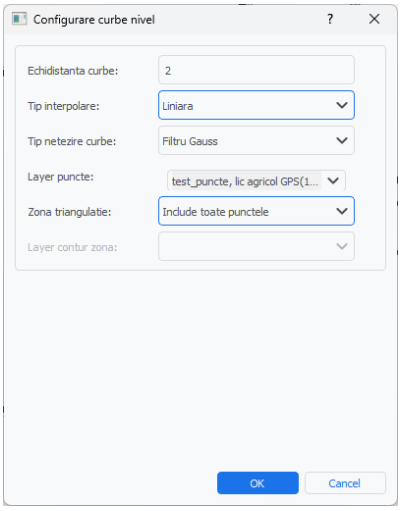


Figure 10. Contour lines generation configuration.

http://doi.org/10.59463/RJAS.2025.4.23

Figure 11. The uncorrected 3D model.

After performing the height analysis and deleting the found outliers, the following DTM was obtained.

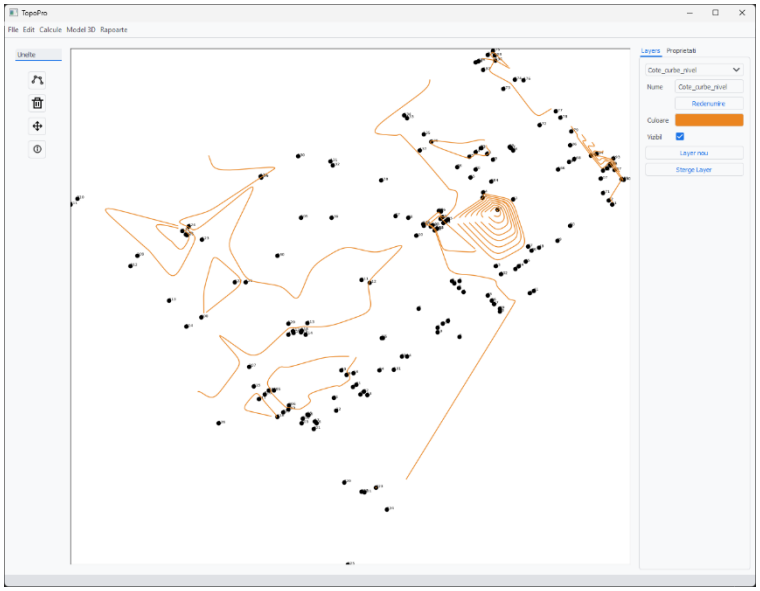


Figure 12. The corrected 3D model

http://doi.org/10.59463/RJAS.2025.4.23

With the contour lines generated, the only step needed to finish the proposed workflow is to export the generated dataset to DXF.

As a last comparison, the software was used to generate the 3D model and DTM for the Plumbuita Park area, which represents the study site used during the second-year field practice at the Faculty of Geodesy, UTCB. The resulting DTM and contour lines were then exported to DXF and compared with a model created using a widely adopted commercial topographic software.

In the comparative visualization, the contour lines generated by the commercial application were represented in yellow, while those produced by the proposed TopoPro software were represented in brown.

The visual comparison revealed a high level of consistency between the two datasets, confirming the accuracy and reliability of the developed software for practical topographic applications.

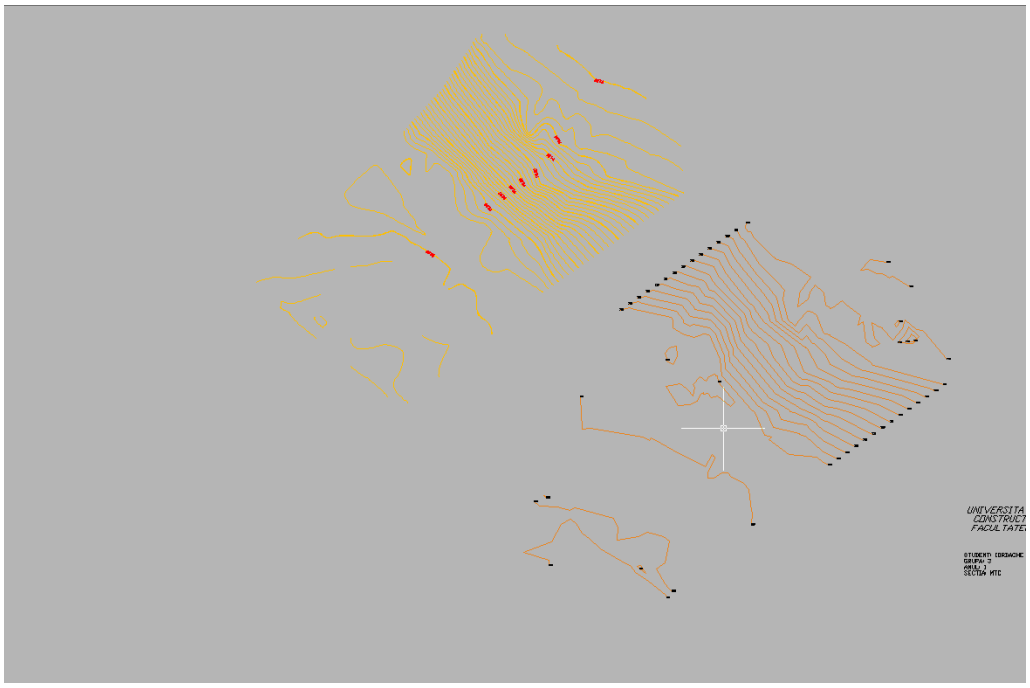


Figure 13. Comparison between commercial software and TopoPro.

CONCLUSIONS

This paper presented the design and implementation of a multifunctional topographic software developed entirely in Python, called TopoPro.

The program integrates essential surveying and geospatial operations, such as coordinate transformation, 3D terrain modeling and contour line generation, all within a transparent and extensible computational framework.

The obtained results demonstrated that the software performs comparably to well-known commercial applications, while maintaining full reproducibility, flexibility and cost efficiency.

http://doi.org/10.59463/RJAS.2025.4.23

All this is done using advanced computational algorithms like Delaunay triangulation, linear, cubic interpolation and smoothing techniques (Gaussian filtering, NURBS curves), which can also be found in commercial alternatives.

In conclusion, TopoPro represents a viable alternative to costly commercial programs, which is transparent and well-documented. Although the program may still present certain logical or computational inconsistencies, these can be easily identified and corrected due to the transparent structure of the code and the comprehensive documentation provided.

Such minor issues do not significantly affect the overall functionality or accuracy of the software.

BIBLIOGRAPHY

DANCIU, V., 2004. Prelucrarea măsurătorilor geodezice în: Rețele geodezice de sprijin. 2004. București: Conspres.

DE BERG, MARK, 2005. Computational Geometry. Algorithms and Applications. 2005. Springer.

DINAS, SIMENA AND MARTÍNEZ, HECTOR, 2020. Delaunay Triangulation. In: . p. 1–6. ISBN 978-3-319-08234-9.

GARVEY, EVAN JAMES, 2020. DSP -Gaussian Filtering in Image Filtering - Report. ResearchGate. Online. 19 April 2020. DOI 10.13140/RG.2.2.22274.90569.

HARTLEY, RICHARD AND ZISSERMAN, ANDREW, 2004. Multiple View Geometry in computer vision. 2004. Cambridge University.

IONESCU, ION, 2005. Fotogrammetrie Inginereasca. 2005. Matrix Rom Bucuresti.

JOHANNESSEN, K. A. AND FONN, Eivind, 2020. Splipy: B-Spline and NURBS Modelling in Python. ResearchGate. Online. 2020. Available from: https://www.researchgate.net/publication/356717298_Splipy_B-Spline_and_NURBS_Modelling_in_Python

SZELISKI, RICHARD, 2022. Computer Vision: Algorithms and Applications. 2022. Springer.

***10 minutes to pandas — pandas 2.3.3 documentation, [no date]. . Online. Available from: https://pandas.pydata.org/docs/user_guide/10min.html#basic-data-structures-in-pandas [Accessed 4 October 2025].

***matplotlib.tri — Matplotlib 3.10.8 documentation, [no date]. . Online. Available from: https://matplotlib.org/stable/api/tri_api.html [Accessed 13 January 2026].

***pandas - Python Data Analysis Library, [no date]. . Online. Available from: https://pandas.pydata.org/ [Accessed 13 January 2026].

***PORTLAND STATE UNIVERSITY, [no date]. TIN & Surface Interpolation.. Online. Available from: https://web.pdx.edu/~jduh/courses/geog493f13/Week06.pdf [Accessed 13 January 2026].

***Qt for Python, [no date]. . Online. Available from: https://doc.qt.io/qtforpython-6.9/ [Accessed 13 January 2026].

***scikit-learn: machine learning in Python — scikit-learn 1.8.0 documentation, [no date]. . Online. Available from: https://scikit-learn.org/stable/index.html [Accessed 13 January 2026].

***SciPy, [no date]. . Online. Available from: https://scipy.org/ [Accessed 13 January 2026].