Mátyás Gede*, Krisztián Kerkovits**, Lola Varga***

Automatic Georeferencing of the 1951–53 Topographic Map Series of Hungary

Keywords: automatic georeferencing, topographic maps, OpenCV, Python, Tesseract, GDAL

Summary: A fully automatized method was developed to georeference the scanned 1 : 25 000 map sheets of the Hungarian topographic survey carried out in 1951–53. The core of the process is the detection of the corners of the map content and the recognition of the sheet identifiers. The sheets depict geographic quadrangles whose extent can be derived from the sheet ID. The sheet corners are used as GCPs for the georeference.
The maps are using the Gauss–Krüger projection system. The geodetic base is dual: the Bessel ellipsoid in the western and southern part of the country and Krasovskiy ellipsoid in northeast. The datum transformation parameters were also adjusted to minimize the misplacement of the georeference. Special attention was paid for the sheets along the edge of the two parts as these sheets had irregular shape.
The whole process is implemented in Python, using only open-source tools: OpenCV for image processing, Tesseract for OCR and GDAL for georeferencing.
1147 map sheets were processed with an average speed of 4 seconds per sheet. False detection of the corners is automatically filtered by geometric analysis of the detected GCPs, while the sheet IDs are validated using regular expressions. The error of corner detection is under 1% of the sheet size for 89% of the sheets, under 2% for 99%.
Although the system is fine-tuned to this specific map series, it can be easily adapted to any other map series having approximately rectangular frame.

Introduction

Old topographic map series are keys when we need geographic information from the past. Surveys that were carried out in the 19th and 20th centuries are accurate enough to compare their content to the current situation. The information extracted from these maps can be used in several fields: analyzing changes of landscape or land use; in hydrological planning (as the course of waterways, the extent of lakes, swamps in the past carries important information about the present flood risks of an area), and so on. Making the map content comparable requires accurate georeferencing.
Manual georeferencing of a single map sheet is easy. Processing a whole series of large-scale topographic maps, however, is a cumbersome, tedious work. Luckily, there are various computer vision tools that can be used for creating an automated georeferencing workflow.
The authors of this paper present a solution that is able to perform this job on the scanned sheets of the 1951–53 (1 : 25 000 scale) topographic map series of Hungary.

Previous research

Naturally, this work is not unprecedented. Jatnieks (2010) developed a QGIS plugin called MapSheetAutoGeoRef, which – although makes mass georeferencing much faster – is only a semi-

* Associate professor, ELTE Eötvös Loránd University, Budapest, [saman@inf.elte.hu]
** Assistant professor, ELTE Eötvös Loránd University, Budapest, [kerkovits@map.elte.hu]
*** ELTE Eötvös Loránd University, Budapest, [vargalola24@gmail.com]

automatic approach. The user needs to manually mark the sheet conrers, and a grid reference data source is also required. Unfortunately, this plugin no longer works with the current versions of QGIS.

A different approach can be seen in Rus et al. (2010), using radon transformation for extracting the map coordinate grid lines and after that using the grid intersections as ground control points (GCPs), deriving their projection coordinates from the sheet identifier number. Herold et al (2011) also presents a similar solution, unfortunately giving little details about the processing steps and the software used.

Titova and Chernov (2009) determined the position of the map frame first, and then tried to detect local cross marks repeated throughout the map; both detections were performed using pattern matching. Coordinate information was supplemented based on the sheet ID, which was part of the file names.

Although some steps of the solution presented in this paper show similarities with the ones mentioned, there are important differences. The whole process was implemented using open-source tools. Frame detection is performed using Hough transformation and is refined by customized convolutional filters. Sheet identifiers are read by OCR. GCPs of georeferencing are the four corners of the map sheets whose coordinated can be derived from sheet ID. Jatnieks' (2010) approach is used to embed georeference data as GCPs in the intermediate GeoTIFF raster.

Preliminary results of this work had already been published at the ICC2019 (Gede, Varga 2021). Since then, additional research was done on the geodetic datum of the survey, and the corner finding algorithm was also refined, resulting in more accurate georeferencing.

Properties of the map series

The 1951–53 1 : 25 000 military topographic map series of Hungary (Figure 1), also named "Quick update" was based on previous topographic maps and aerial photographs, updated by field work. The geodetic base and the extent of map sheets was designed to match the Soviet system. Map sheets are bounded by 1/8° by 1/12° geographic quadrangles.

Due to the secrecy of military maps and the insufficient information provided by Soviets, the initial geodetic base was not exactly the same as the Soviet system: although the maps were constructed in the Gauss–Krüger projection system, the geodetic datum was a formerly used datum on the Bessel ellipsoid, not the Soviet standard Pulkovo 1942 datum on Krasovsky ellipsoid. As described later in detail, this difference caused misfit of the maps to sheets of the Soviet Union on the eastern borders of the country. To overcome this problem, map sheets east of 20°30' and north of 47° were recompiled in the new system. Content of sheets along the boundary longitude/latitude were extended to fill the gaps caused by the datum shift (Figure 2).

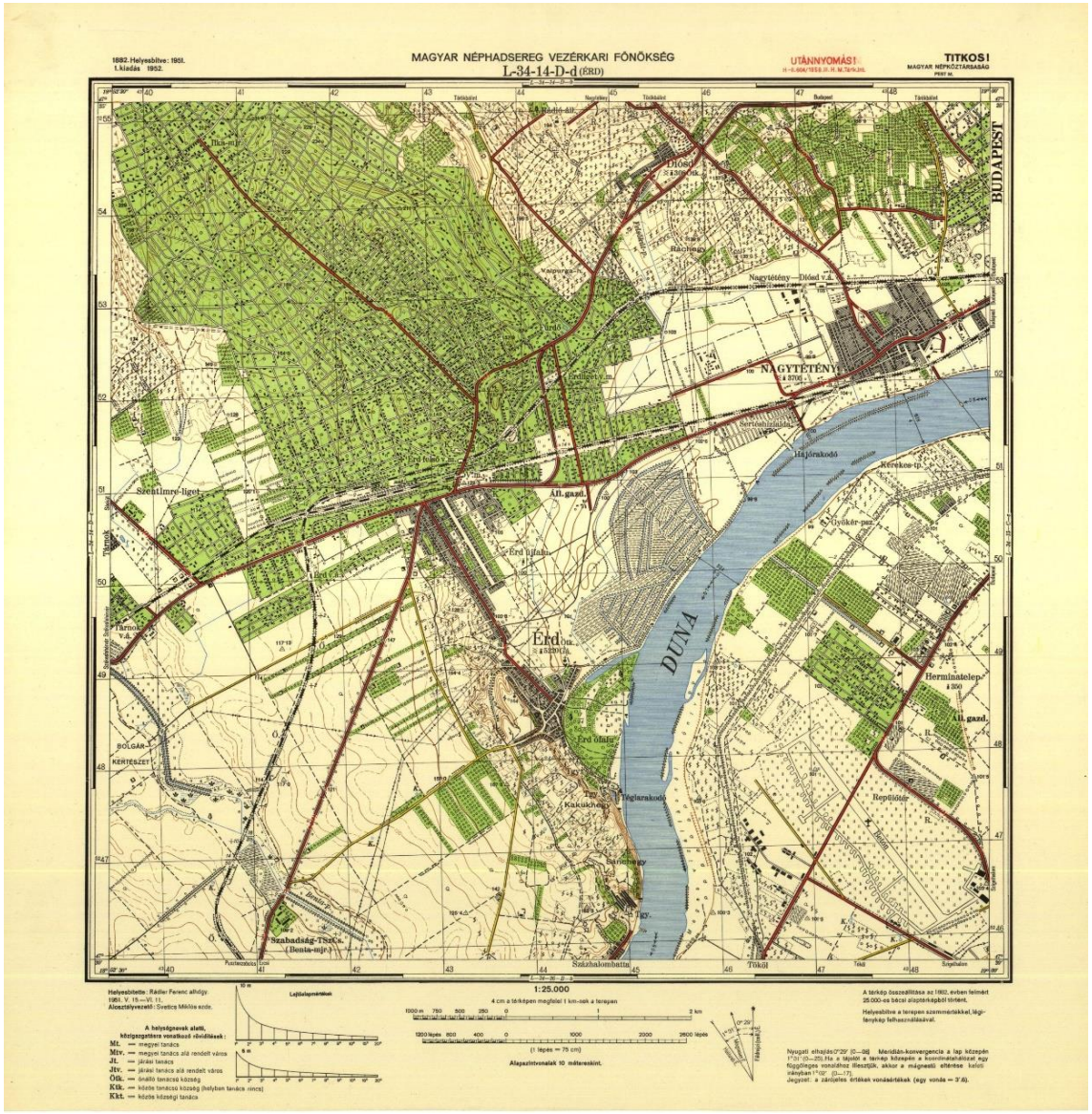

Figure 1: 1 : 25 000 scale map sheet of the 1951–53 military topographic map series of Hungary.

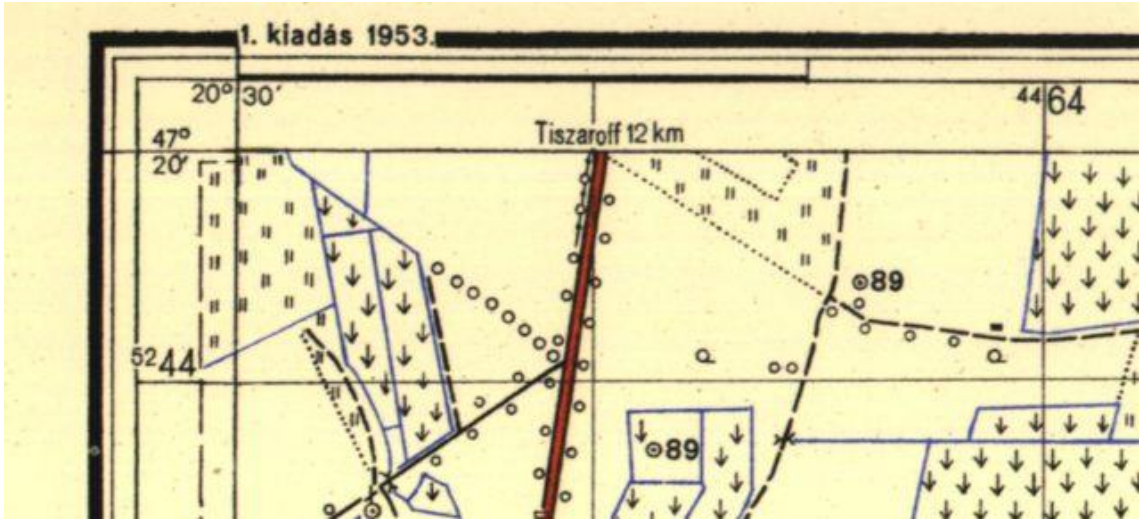

Figure 2: Extension of map content beyond the sheet boundary (20°30') to avoid gap caused by the datum shift.

Unfortunately, most information about this inconsistency of datums is available as oral tradition, because the political situation in the 1950s did not let cartographers to explain the issue. It turned out to be that the most trustful source of information are the hidden references of a contemporary paper (Homoródi 1952). In this paper, Homoródi vaguely refers to a "Central European system" based on the Bessel ellipsoid established in the 1940s. Furthermore, he mentions a large geodesic work to connect regional datums in a larger, unified system. Calculations were done on the Hayford ellipsoid, but Homoródi stated that they plan to use these results urgently and unmodified for the Krasovskiy ellipsoid.

By careful reading, one may extract information that had to be hidden from the Soviet censors. The passages suggest that the "Central European system" is the geodetic base of the German Reich using the Potsdam datum referred to as DHG by Mugnier (2015). This system was probably introduced in Hungary at the end of 1940. We found a few topographic map sheets in the library using this system between 1940 and 1945, which strengthened our assumptions. We concluded that map sheets printed until the end of 1952 (Southern and Western Hungary) simply used the German DHG coordinate system.

The paper of Homoródi (1952) states that the base points in Hungary were transformed to the Hayford ellipsoid for international use, and lists the coordinates for several Laplace points. This suggests that Hungarian base points were probably used for the determination of datum ED50 in Hungary during World War II. However, this system was never used to our best knowledge, as in 1952, the Soviet commanders compared the map sheets already produced to their maps, and found a few kms of difference in the projected coordinates (due to the different size of the ellipsoid), and geographic coordinates were off by ca. 100 m, furthermore the two grid systems were not even parallel to each other. This led to a scandal, and later sheets were to be produced in the Pulkovo 1942 system. Homoródi (1952) suggests that the transformation of the existing system to the Krasovskiy ellipsoid must be done in a rush, and they simply plan to use the values of the vertical deflection calculated on the Hayford ellipsoid. This means that the map sheets produced in 1953 using the Pulkovo 1942 system did not have a well-established geodetic base.

The reference system of the earlier maps using DHG coordinates was easy to define as a PROJ string. Homoródi (1952) listed ellipsoidal coordinates for a handful of points, and Timár et al. (2004) have already calculated Burša–Wolf parameters for these set of points. Using this, the appropriate definition is:

+proj=tmerc +lat_0=0 +lon_0=21 +k=1 +x_0=4500000 +y_0=0 +ellps=bessel
+towgs84=566.54,108.52,487.93,2.2867,2.6409,-1.5194,-0.7365 +units=m +axis=neu
+no_defs

Note, that West of meridian 18°, *lon_0=15* and *x_0=3500000*.

The reference system of the sheets produced in 1953 were not straightforward to define. Using reference system EPSG:3334 (with standard parameters for the Pulkovo 1942 datum), the map sheets were misaligned by ca. 40 m. It turned out that the Burša–Wolf parameters listed by Timár et al. (2003) provided way more acceptable result. Thus, the definition used is:

+proj=tmerc +lat_0=0 +lon_0=21 +k=1 +x_0=4500000 +y_0=0 +ellps=krass
+towgs84=17.20,-84.03,-60.97,1.085,0.682,-0.473,-3.185 +units=m +axis=neu +no_defs

Calculating the difference between the two datums, the gap between adjacent sheets at meridian 20°30' is 75 m wide, and at parallel 47°, it is 30.5 m wide. Dividing this by the map scale, new sections produced in 1953 and bounded by 20°30' are 3 mm wider, and new sections bounded by parallel 47° are 1.2 mm taller (cf. Figure 2). This difference will be considered by the corner detection algorithm described below.

Software implementation

The workflow of the automatic georeference is built up of several steps, performed by two Python scripts. The first one is responsible for the corner and sheet ID detection:

- extracting black(ish) pixels,
- straight line detection using probabilistic Hough transformation,
- finding outermost horizontal and vertical lines and their intersections – the outer frame corners,
- finding inner corners,
- performing OCR on the top middle area of the map sheet to find sheet ID,
- applying a regex match on recognized text to extract the sheet ID only,
- writing the file name, the corner pixel coordinates and the sheet ID to a CSV file.

The second script performs the actual georeferencing of the images. This is separated from the detection part for a practical reason: detection settings needed several rounds of refining. Mass georeferencing of the sheets makes sense only when the control points are already detected with sufficient accuracy. Steps of the georeferencing are:

- calculating sheet bounding latitudes and longitudes from sheet ID,
- transforming these coordinates to Gauss–Krüger projection,
- merging GCP information to the yet unmodified image,
- transforming the image into Plate Carrée projection,
- cropping and saving actual map content (within frame borders).

The working environment was Python 3.7, using Numpy 1.18.1, OpenCV 4.1.2, GDAL (OSGeo version 3.1.4) and Tesseract OCR (pytesseract), version 4.1.0. The test computer is a 64-bit Windows 10 desktop computer having i7-4770 CPU, 16GB RAM and Nvidia GeForce GTX 1050 graphical card.

Corner detection

These map sheets have a complex frame. A thick outer frame and a thin inner one with grid lines, coordinates, and other content between them. The frame is printed in black, so the first step is the separation of black pixels on the scanned image. Unfortunately, some of the prints are rather faded, so on many sheets “black” is actually gray. Therefore, a pixel was considered black, if the difference of the R, G, B channels is under 27% while their intensity is under 70%. (These are experimental limits for this map series. The authors also tried using the hue component from HSV as well as simply filtering the darkest pixels of the grayscale image, but those methods were not as successful as the chosen one.) The result of the black pixel extraction is a binary mask (Figure 3).

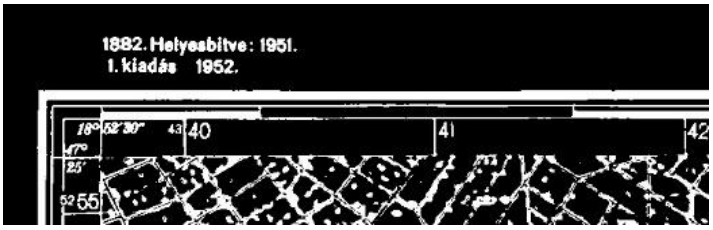

Figure 3: Extracting black pixels.

Long straight lines are detected using OpenCV's probabilistic Hough transformation (OpenCV, 2020). We only needed the frame borders so only lines longer than the quarter of the image width were taken into consideration. Naturally, frame lines are much longer than that, but due to imperfect printing, there are very often gaps breaking these lines, so if the minimum length limit is set higher, sometimes the frame lines are also skipped.

The next step is the separation of (nearly) horizontal and vertical lines. The angular tolerance is set to 5° here because some maps were scanned a little bit rotated. The leftmost, rightmost vertical and the topmost horizontal lines define the left, right and top outer frame. The bottom frame is a bit harder to find because there are long scale bars at the bottom and if the line length filtering limit is set high enough to always exclude them, it often skips the frame lines as well. Instead, the rough position of the bottom line is estimated based on the three other frame lines, and lines below a specified limit are omitted when looking for the bottommost horizontal line. Once all the frame lines are found, their intersections give the rough position of the outer frame corners.

The outer corner positions are refined by applying a convolutional filter using the kernel seen on Figure 4 in the neighborhood of the rough corner positions. The location of the maximum value after convolution gives the exact position of the corner.

$-\frac{1}{15}$	$-\frac{1}{15}$	$-\frac{1}{15}$	$-\frac{1}{15}$	$-\frac{1}{15}$	$-\frac{1}{15}$	$-\frac{1}{15}$
$-\frac{1}{15}$	$-\frac{1}{15}$	$-\frac{1}{15}$	$-\frac{1}{15}$	$-\frac{1}{15}$	$-\frac{1}{15}$	$-\frac{1}{15}$
$-\frac{1}{15}$	$-\frac{1}{15}$	$-\frac{1}{15}$	$-\frac{1}{15}$	$-\frac{1}{15}$	$-\frac{1}{15}$	$-\frac{1}{15}$
$-\frac{1}{15}$	$-\frac{1}{15}$	$-\frac{1}{15}$	0	$\frac{1}{15}$	$\frac{1}{15}$	$\frac{1}{15}$
$-\frac{1}{15}$	$-\frac{1}{15}$	$-\frac{1}{15}$	$\frac{1}{15}$	$\frac{1}{15}$	$\frac{1}{15}$	$\frac{1}{15}$
$-\frac{1}{15}$	$-\frac{1}{15}$	$-\frac{1}{15}$	$\frac{1}{15}$	$\frac{1}{15}$	$\frac{1}{15}$	$\frac{1}{15}$
$-\frac{1}{15}$	$-\frac{1}{15}$	$-\frac{1}{15}$	$\frac{1}{15}$	$\frac{1}{15}$	$\frac{1}{15}$	$\frac{1}{15}$

Figure 4. Kernel used for refining top left corner position. Kernels for other corners are mirrored versions of this.

The north–south extent of a sheet is 1/12° in latitude. This, considering the 1 : 25 000 scale gives the inner height of the map as roughly 371 mm (calculating with an Earth radius of R = 6373 km):

$$h = R\frac{1}{12}\frac{\pi}{180}$$

The frame width is 10 mm (described in the map legend booklet, see Figure 5). Based on this, the exact resolution of the scan can be calculated as the pixel distance between the top and bottom frames divided by 391 (i.e. 371 + 2*10). Using the raster resolution, the position of the inner corners can also be estimated – 10 mm both horizontally and vertically from the outer corners. (If a map sheet is bounded by the 20°30' meridian from the west or by the 47° latitude from the south, a larger distance is used on that side, because of the extended map content, see Figure 2.) These positions are also refined by a local search for intersecting horizontal and vertical lines. The results of corner detection can be observed on Figure 5.

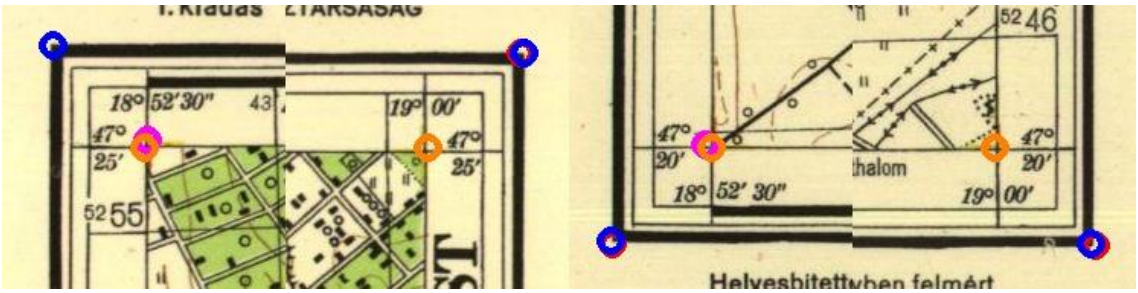

Figure 5. Estimated (red and purple) and refined (blue and orange) positions of the outer and the inner corners of a map sheet

Map sheet ID detection

Map sheet identifiers, along with other texts, are located centered above the top map frame (Figure 6). It is followed by the name of the largest settlement on the map in parentheses, so its horizontal position is varying.

Figure 6. Location of map sheet identifier above the top frame

A larger part of the scanned map, above the outer top frame is processed by the *image_to_string* function of Pytesseract, a Python binding of the open-source OCR software Tesseract (Zelic & Sable, 2021). This function returns the raw recognized text, such as *"MAGYAR NÉPHADSEREG VEZÉRKARI FŐNÖKSÉG L-34-14-D-d (ÉRD)"*. The sheet identifier is extracted using a regular expression match on this text. Hungarian military topographic maps followed the Soviet sheet nomenclature, a 1 : 25 000 sheet having the *{letter}-{number}-{number}-{letter}-{letter}* pattern; the first letter and number determine a 6° by 4° quadrangle. This quadrangle is divided into 12×12 (144) small quadrangles; the second number identifies one of these. The following letter is one of A, B, C, D and specifies one quarter while the last letter (from a, b, c, d) again identifies a quarter (War Department and Bolin, 1946). Figure 7 shows and overview map of Hungarian sheets.

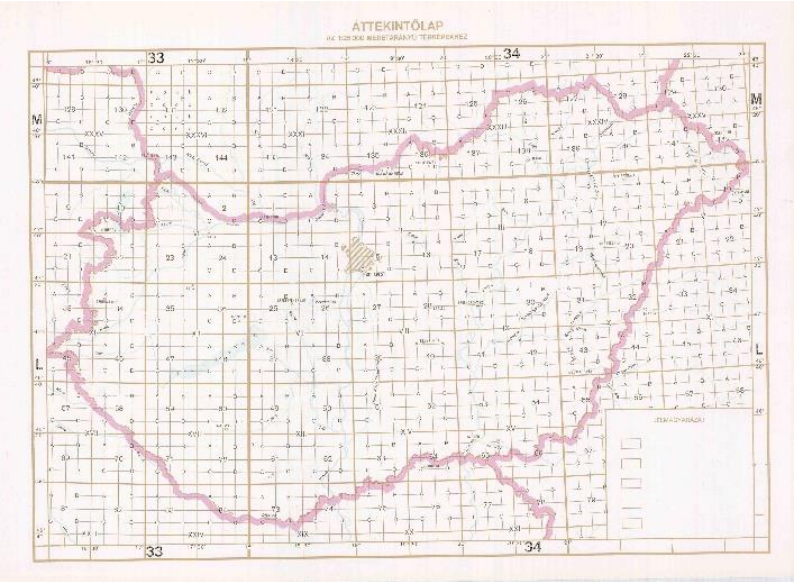

Figure 7. Overview map of Hungarian Gauss–Krüger sheets. Courtesy of Institute of Cartography and Geoinformatics, Eötvös Loránd University

In the case of the currently processed maps, sheet identifiers were also available as parts of the filenames of the scanned maps (e.g. *bmp_L-33-011-B-c.jpg*). Automatic recognition of sheet IDs are still an important part of this project, as lessons learned here might be useful in cases where there is no other source of this information. Additionally, having the sheets IDs in the file names made it simple to automatically evaluate the accuracy of the recognition.
The coordinates of the map corners as well as the extracted sheet ID is saved in a CSV file in the following format:

```
bmp_L-33-011-C-c.jpg,365,281,2731,280,353,2614,2728,2611,L-33-11-C-e
bmp_L-33-011-C-d.jpg,362,289,2722,293,350,2622,2723,2627,L-33-11-C-d
bmp_L-33-011-D-a.jpg,367,286,2728,286,356,2619,2727,2620,L-33-11-D-a
bmp_L-33-011-D-b.jpg,296,293,2656,295,285,2626,2651,2623,L-33-11-D-b
```

Georeferencing

Once we have the coordinates of the corners and the sheet identifier, georeferencing is rather simple. The authors used the Python bindings of GDAL for this task. The geographic latitude and longitude of the corner points is calculated from the sheet ID. These coordinates are then transformed to Gauss–Krüger projection (zone depending on the longitude). Control points (the map corners) with the projected coordinates are merged to the unmodified raster image using GDAL's Translate function into an intermediate GeoTIFF file. This is already a georeferenced raster, but still containing the map frame and all the content of the scanned map outside the frame. If the goal is to create a seamless mosaic of the map sheets, one final step is needed: The map is reprojected into Plate Carrée projection and cropped along the boundary latitudes and longitudes. This image can be saved using

the Warp function of GDAL (Warmerdam, Rouault et al., 2021). Figure 8 shows the seamless mosaic of georeferenced sheets. The few holes visible are places of sheets that are missing from our collection.

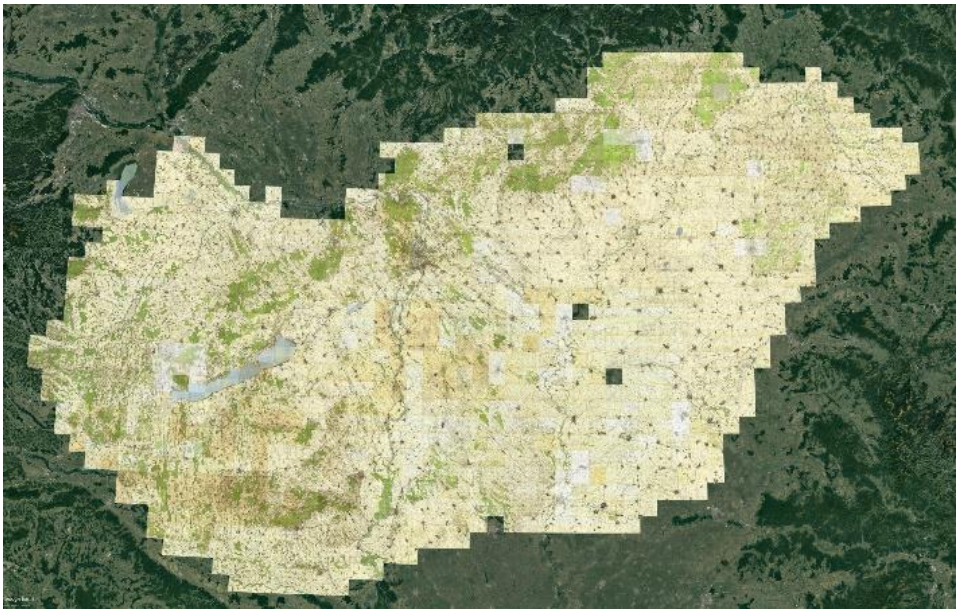

Figure 8. Seamless mosaic of georeferenced sheets in displayed in Google Earth.

Results, discussion

The method described above was able to detect map corners on all the 1148 sheets. The average time needed for corner and sheet ID detection is 3.5 s per sheet, while projection transformation and saving the cropped, georeferenced map is only 0.6 s.

Accuracy of corner detection

The accuracy of detected map corner positions was evaluated by simple mathematical methods. There ratios are calculated: the ratio of the length of left and right frames; the ratio of the length of top and bottom frames, finally the ratio of the length of horizontal and vertical frames corrected by 2/3 of the cosine of the latitude. These ratios should optimally be 1; their error is the difference from 1. The corner detection error of a sheet is described by the largest of the three differences. The histogram chart (Figure 9) of these errors indicates that the displacement is typically 2mm.

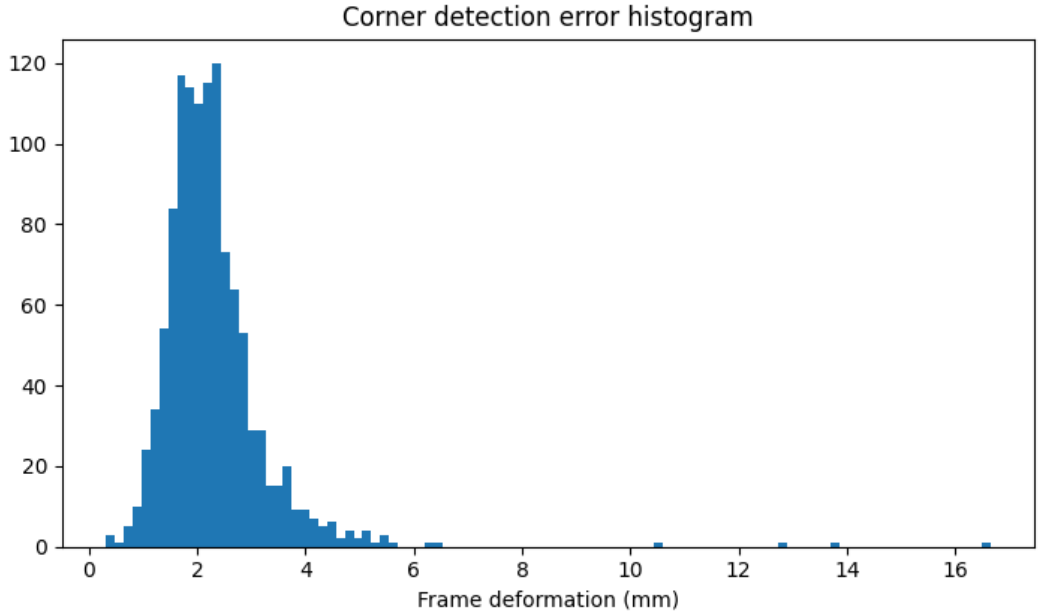

Figure 9. Histogram of frame deformation caused by inaccuracy of corner detection

There were two sheets with the map content "breaking out" beyond the map frame (The country border is only a few centimeters beyond the frame, so editors decided not to create an extra sheet for it, see Figure 10). Although corner detection worked on these sheets too, due to the extra content they had to be processed manually.

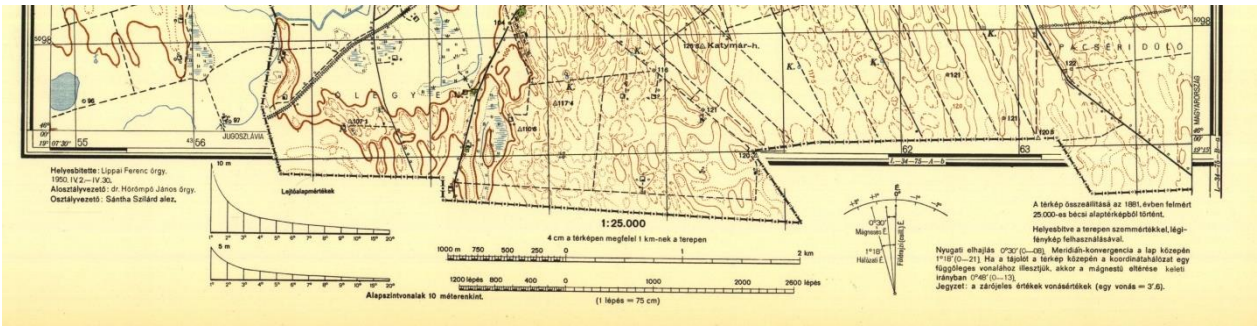

Figure 10. Irregular map sheet with map content beyond the frame

Accuracy of sheet ID recognition

As the file names of the scanned maps already contained the sheet IDs, automatic recognition of sheet identifiers was not a crucial part of the project. Still, it was a great opportunity to easily test the accuracy of OCR. For approximately quarter of the sheets (261 out of 1148) there was no match for the testing regular expression, mostly because some characters of the ID were not digitized. When the regular expression had a match, it was usually correct, or only had minor, automatically correctable errors: e.g. the last character was recognized as an 'e' while only 'a', 'b', 'c', 'd' is allowed there, which means it should have read as a 'c'. There were only 16 cases (1,4%) where the detected sheet ID was misread beyond the possibility of automatic correction.

It is worth mentioning that Tesseract OCR was used with its default settings, which means that text recognition accuracy most probably can be improved by finetuning the settings of OCR, and/or training it on the fonts used on the maps.

References

Gede M. and Varga L. (2021). Automatic Georeferencing of Topographic Map Sheets Using OpenCV and Tesseract. Proc. Int. Cartogr. Assoc., 4, 38, https://doi.org/10.5194/ica-proc-4-38-2021, 2021.

Herold, H., Roehm, P., Hecht R. and Meinel, G. (2011). Automatically georeferenced maps as a source for high resolution urban growth analyses. In: Proceedings of the 25th ICA International Cartographic Conference, July 3 - 8, 2011, Paris, France.

Homoródi L. (1952). Vizsgálatok új háromszögelési hálózatunk elhelyezésére és tájékozására. Földméréstani közlemények, vol. 4, no. 1. pp. 2–10, continued in no. 2. pp. 61–71.

Jatnieks, J. (2010) Extended Poster Abstract: Open Source Solution for Massive Map Sheet Georeferencing Tasks for Digital Archiving. In: Chowdhury G., Koo C., Hunter J. (eds) The Role of Digital Libraries in a Time of Global Change. ICADL 2010. Lecture Notes in Computer Science, vol 6102. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-13654-2_33

Mugnier, C.J. (2015). Grids & Datums: Federal Republic of Germany. Photogrammetric Engineering & Remote Sensing vol. 81. no. 6. pp. 435–440.

OpenCV (2020). Open Source Computer Vision documentation. https://docs.opencv.org/4.1.2/

Rus, I., Balint, C., Craciunescu, V., Constantinescu, S., Ovejanu, I., Bartos-Elekes, Zs. (2010). Automated Georeference of the 1:20 000 Romanian Maps Under Lambert-Cholesky (1916–1959) Projection System

Titova, O.A., Chernov, A.V. (2009). Method for the automatic georeferencing and calibration of cartographic images. Pattern Recognit. Image Anal. 19, 193–196. https://doi.org/10.1134/S1054661809010325

Timár G., Kubány Cs., Molnár G. (2003). A magyarországi Gauss–Krüger-vetületű katonai topográfiai térképek dátumparaméterei [Datum parameters of the Hungarian Gauss–Krüger military topographic maps]. Geodézia és Kartográfia vol. 55. no. 7. pp. 18–22.

Timár G., Lenkei P., Molnár G., Varga J. (2004). A második világháború német katonai térképeinek koordinátarendszere [GIS integration of the German Army Grid (DHG) and its geodetic datums]. Geodézia és Kartográfia vol. 56. no. 6. pp. 28–35.

War Department (USA) and Bolin, R. L., Depositor (1946) "Handbook on USSR Military Forces, Chapter XII: Maps, Conventional Sign, and Symbols" (1946). DOD Military Intelligence. 29.

http://digitalcommons.unl.edu/dodmilintel/29

Warmerdam, F., Rouault, E. et al., (2021) GDAL documentation. https://gdal.org/

Zelic, F, Sable, A. (2021) A comprehensive guide to OCR with Tesseract, OpenCV and Python. 2021. https://nanonets.com/blog/ocr-with-tesseract/