International Cartographic Association
Association Cartographique Internationale

Automatic Georeferencing of Topographic Map Sheets Using OpenCV and Tesseract

Mátyás Gede [a,*], Lola Varga [a]

[a] *Institute of Cartography and Geoinformatics, ELTE Eötvös Loránd University – saman@map.elte.hu, vargalola24@gmail.com*

* Corresponding author

Abstract: The authors developed a pipeline for the automatic georeferencing of older 1 : 25 000 topographic map sheets of Hungary. The first step is the detection of the corners of the map content, then the recognition of the sheet identifier. These maps depict geographic quadrangles whose extent can be derived from the sheet ID. The sheet corners are used as GCPs for the georeference.

The whole process is implemented in Python, using various open source libraries: 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%. The sheet ID recognition success rate is 75.9%.

Although the system is finetuned to a specific map series, it can be easily adapted to any other map series having approximately rectangular frame.

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

1. Introduction

Old maps are great source of information about the past. This particularly true for map sheets of accurate country surveys: since the end of the 19th century, mapping methods became accurate enough to get reliable positional information, while the introduction of standardized map legends facilitate efficient information extraction. The information retrieved can be used in studies needing time series of georeferenced data such as landscape change or urban growth analysis.

Using these maps in GIS environment requires their accurate georeferencing. While it is quite straightforward for one map sheet, georeferencing a whole series of map sheets may take tremendous time. The authors' goal was to automatize this job using open source computer vision tools.

1.1 Similar research

There are several previous projects aiming automatic georeferencing of maps. Jatnieks (2010) developed a QGIS plugin called MapSheetAutoGeoRef which – although makes mass georeferencing much faster – is only a semi-automatic approach. The user has to manually mark the sheet conrers, and a grid reference data source is also required.

Rus et al. (2010) used radon transformation for extracting the map coordinate grid lines and used the grid intersections as ground control points (GCPs), deriving their projection coordinates from the sheet identifier number. A similar approach is presented by Herold et al (2011) although rather little details are given 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.

The solutions presented in this paper have similarities to the ones above, but there are important differences as well: only open source software was used; the frame detection is based on Hough transformation and is refined by a simple convolutional filter to locally detect corners of thick frames; and the sheet ID detection is performed by OCR (although the sheet ID is also available in the file names of the test map series – this information was used for evaluating the accuracy of 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.

1.2 Software and hardware environment

The software implementation was realized in 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.

30th International Cartographic Conference (ICC 2021), 14–18 December 2021, Florence, Italy. This contribution underwent single-blind peer review based on submitted abstracts. https://doi.org/10.5194/ica-proc-4-38-2021

1.3 Maps processed

The methods described below was tested on the 1950–52 1 : 25 000 military topographic map series of Hungary (Figure 1). Map sheets are bounded by 1/8° by 1/12° geographic quadrangles; the boundary latitudes and longitudes can be calculated based on the sheet ID. The map frame consists of a thicker outer and a thin inner frame, the total frame width is 10mm. The sheet ID is right above the top frame, centered (Figure 2). The maps are in the Gauss–Krüger projection system, using the Pulkovo 1942 datum on Krasovsky ellipsoid.

Figure 1. Sample map sheet.

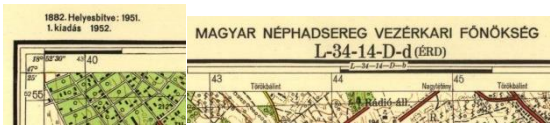

Figure 2. Map frame structure (left) and location of the sheet ID (right)

2. Processing steps

2.1 Corner detection of map sheets

The first step of image processing was the extraction of "blackish" pixels from the RGB image. The frame and the sheet ID are both printed in black, which appears as dark grey on the scanned image. Its extraction was performed by creating a mask of those pixels where the difference of the R, G, B channels is under a specific limit (25%) and the average of them is under another one (50%). These limits were determined by experiments; other map sets may require different settings. The result of the filtering is a binary mask (Figure 3).

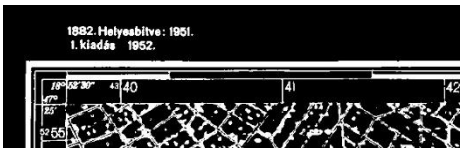

Figure 3. Result of black masking

The next step is straight line extraction on this mask. OpenCV provides a convenient method using the probabilistic Hough transformation for this task (OpenCV 2020). As only the long lines of the frames are really needed, the parameters of line extraction were set to find only lines longer than the quarter of the image width. (Although frame lines are much longer than that, in reality the print is often imperfect and there are gaps so if the minimum line length is set too high, sometimes the frame lines are also filtered out.)

Based on the azimuth of the lines, the nearly horizontal and the nearly vertical ones are extracted into separate lists. The angular tolerance at this step is set to 5 degrees in order to successfully process map sheets that were scanned with minor rotation.

The leftmost and the rightmost vertical lines define the outer side frames, while the topmost horizontal one is the outer top frame. The intersections of the side frames and the top frame define the rough position of the outer corners of the map sheet.

Detection of the bottom frame proved to be troublesome for this map series as there are long scale bars below it, and the authors were unable to find a set of parameters for Hough line extraction that reliable detects the bottom frame but not the scale bars. Therefore, after finding the top corners and determining the actual map frame width in pixels, this information can be used to estimate the rough position of the bottom frame and lines much lower than that are dropped before finding the bottommost horizontal line. Once we have the bottom line, the bottom outer corner positions are also calculated as the intersections of the frame lines.

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.

$$\begin{bmatrix} -\frac{1}{15} & -\frac{1}{15} & -\frac{1}{15} & -\frac{1}{15} & -\frac{1}{15} & -\frac{1}{15} & -\frac{1}{15} \\ -\frac{1}{15} & -\frac{1}{15} & -\frac{1}{15} & -\frac{1}{15} & -\frac{1}{15} & -\frac{1}{15} & -\frac{1}{15} \\ -\frac{1}{15} & -\frac{1}{15} & -\frac{1}{15} & -\frac{1}{15} & -\frac{1}{15} & -\frac{1}{15} & -\frac{1}{15} \\ -\frac{1}{15} & -\frac{1}{15} & -\frac{1}{15} & 0 & \frac{1}{15} & \frac{1}{15} & \frac{1}{15} \\ -\frac{1}{15} & -\frac{1}{15} & -\frac{1}{15} & \frac{1}{15} & \frac{1}{15} & \frac{1}{15} & \frac{1}{15} \\ -\frac{1}{15} & -\frac{1}{15} & -\frac{1}{15} & \frac{1}{15} & \frac{1}{15} & \frac{1}{15} & \frac{1}{15} \\ -\frac{1}{15} & -\frac{1}{15} & -\frac{1}{15} & \frac{1}{15} & \frac{1}{15} & \frac{1}{15} & \frac{1}{15} \end{bmatrix}$$

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

Once the outer corner positions are known, it's possible to calculate the exact resolution of the raster image. The inner height of the map can be calculated as the length of an 1/8° meridian section divided by 25 000 (the map scale), which gives 372mm. As the frame width is 10mm, the resolution (in mm/pixel units) can be calculated as the pixel distance of the top and bottom frames divided by 392.

Using the raster resolution, the rough position of the inner frames can also be estimated (10mm horizontally and vertically from the outer frames). These positions are also refined by locally searching for intersecting horizontal and vertical lines. Figure 5 shows the results of corner detection on a sample sheet.

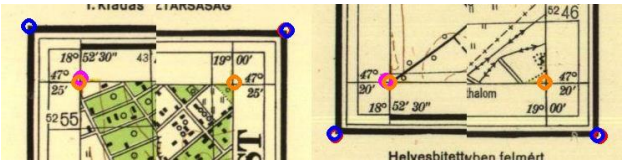

Figure 5. Estimated (red and purple) and refined (blue and orange) positions of the outer and the inner corners of a map sheet

2.2 Detecting map sheet ID

The map sheet ID is located centered above the top frame. As it is followed by the sheet name (name of the largest settlement on the map), its horizontal position is varying from sheet to sheet. Therefore a larger part of the image is cropped and processed by pytesseract's *image_to_string* function (Zelic & Sable, 2021). The resulting text is then tested against a regular expression in order to extract the sheet identifier only. The 1 : 25 000 map IDs follow the *{letter}-{number}-{number}-{letter}-{letter}* pattern; the first letter and number determine a 6° by 4° quadrangle. This quadrangle is divided into 12x12 (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 6 shows and overview map of Hungarian sheets.

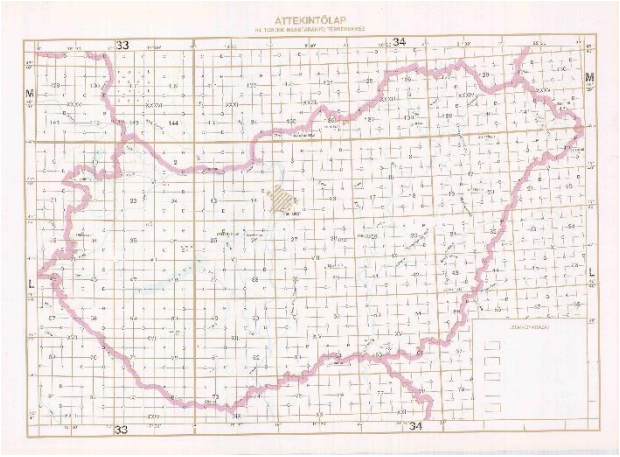

Figure 6. Overview map of Hungarian Gauss–Krüger sheets. Courtesy of Institute of Cartography and Geoinformatics, Eötvös Loránd University

2.3 Georeferencing

With the sheet ID and the map corners detected we have all information required for georeferencing the map. The Python bindings of GDAL were used for this task. First the bounding latitudes and longitudes are calculated from sheet ID. Then the coordinates of the map corners are transformed to Gauss–Krüger projection (keeping in mind that the actual projection zone depends on the longitude). GCP information with the projected coordinates is merged to the unmodified raster image using GDAL's Translate function into an intermediate GeoTIFF file.

Although the result is already a georeferenced raster, additional steps are required to produce a seamless mosaic of map sheets. The map is transformed to latitude/longitude projection and is cropped by the bounding latitudes and longitudes. (In this projection the geographic graticule appears as horizontal/vertical lines, which makes this cropping easy.) The cropped, georeferenced maps are saved as GeoTIFF images using GDAL's Warp function (Warmerdam, Rouault et al., 2021). Figure 7 shows the seamless mosaic of georeferenced sheets. The few holes visible are places of sheets that are missing from our collection.

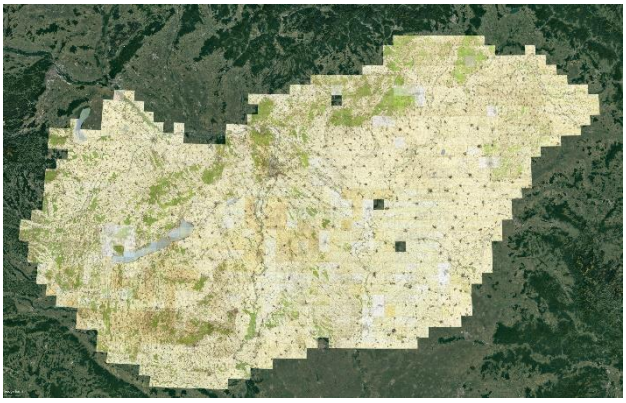

Figure 7. Seamless mosaic of georeferenced sheets in displayed in Google Earth.

3. Results

The solution presented above managed to find map corners on all the 1147 map sheets it was tested. The average processing time of one sheet was around 4 seconds on the testing computer. Line detection was the most time-consuming part, while projection transformation and saving the final georeferenced image war surprisingly fast, requiring about 0.1 seconds per sheet. For practical reasons, two separate scripts were created: one for corner and sheet ID detection and another one for georeferencing.

3.1 Evaluating corner detection accuracy

The accuracy of detected map corner positions were checked 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 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 8) of

these errors indicates that the displacement is typically 0.5%, which, considering the 372mm sheet height, is under 2mm.

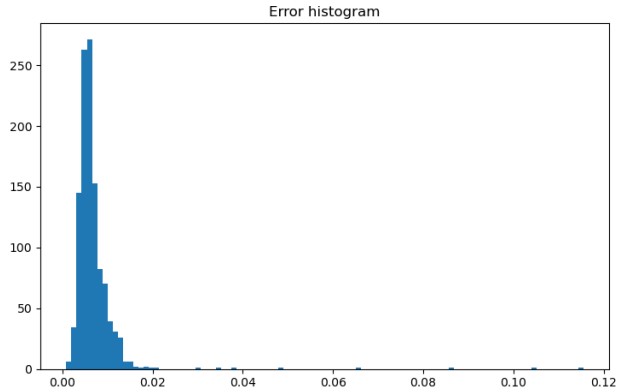

Figure 8. Histogram of corner detection error

It is worth noticing, that a considerable part of these errors is caused by the fact that during the production of these maps the geodetic base of the survey was changed (from the German system based on the Bessel ellipsoid to the Soviet one based on the Krasovsky one). The sheets north of 46°40' and east of 20°30' were compiled in the soviet system, therefore those sheets that are on the border of the two systems have non-standard boundaries.

3.2 Evaluating OCR accuracy

As the file names of the scanned maps already contained the sheet IDs, it was a great opportunity to easily test the accuracy of OCR. For approximately quarter of the sheets (261 out of 1147) 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 recognised 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.

4. Conclusions

The results show that the method described above is suitable for automatically georeferencing large amount of scanned topographic map sheets. At present the weakest point is the accuracy of sheet ID detection, but it can surely be improved by refining OCR settings (or training OCR on the font the given map series is using). Additionally, as the sheet id had been usually assigned to the raster image during the scanning process, in those cases sheet ID detection is not a crucial step of the method.

By minor adjustments, this solution can be used on any topographic map series where

- the sheet ID is either available or machine readable,
- the map frame is rectangular, and
- the coordinates of the sheet corners can be calculated based on the sheet ID.

5. Acknowledgements

EFOP-3.6.3-VEKOP-16-2017-00001: Talent Management in Autonomous Vehicle Control Technologies – The project is supported by the Hungarian Government and co-financed by the European Social Fund.

6. References

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.

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

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

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/