A Segmentation-based Approach for Improving the Accuracy of Polygon Data

Alexey Noskov and Yerach Doytsher
Mapping and Geo-Information Engineering
Technion – Israel Institute of Technology
Haifa, Israel
emails: {noskov, doytsher}@technion.ac.il

***Abstract*—The suggested method enables us to improve the accuracy of city planning data by matching it with exact cadastral data. The existing approaches do not work well in the case of partial equality of polygon boundaries. The main idea of the presented algorithm in this paper is based on defining correspondent segments of polygon boundaries and further replacing polygon boundary segments of the non-accurate layer by segments of the accurate data set, segments without pairs are rectified using ground control points. The resulting data contain parts of the accurate data set polygon boundaries, whereas the remaining elements are rectified according to the replaced boundary segments. A review implemented by specialists enables us to say, that the results are satisfactory.**

Keywords-Polyline similarity; geometry matching; shape descriptor; topology.

I. INTRODUCTION

The same objects on a map, which are on an equal scale, could be presented with small differences because of a high diversity of data sources, organizations, users, or software. In an ideal situation, accurate geometries of exiting maps should be used for preparing new data sets or for updating. Usually, in the real world, new maps are digitized without respect to existing data sets. In many cases, data are unavailable, or available with significant restrictions, because of legal, technical, or other reasons. Additionally, even if an accurate data set is freely available, people often do not want to spend time using an existing data set; in most cases they prefer to digitize new geometries on a satellite image or scanned map. These data should be aligned using accurate data sets. This problem is especially sensitive for large-scale maps and plans [7].

Rectifying data using a set of ground control points is a popular way of improving the accuracy of a map [18]. The results of this approach are not satisfactory in many cases, because rectified objects could not be identical to directly measured accurate objects. Another possibility is based on defining correspondent objects on an accurate data set by geometry or attributes and replacing correspondent objects [15]. A serious problem with this approach follows from the fact that objects could be partially similar (e.g., segments of a polygon boundary are same, other parts are different). In contrast to existing approaches, the main idea presented in the paper, an algorithm is based on defining correspondent segments of polygon boundaries and further replacing polygon boundary segments of the non-accurate layer by segments of an accurate data set; segments without pairs are rectified by ground control points. The proposed algorithm could be applied for different polygon datasets with small boundary differences.

The problem is described in the paper using cadastral and city planning maps. A cadastral map is a comprehensive register of the real estate boundaries of a country. Cadastral data are product using quality large-scale surveying with total station, Differential Global Positioning System devices or other surveying systems with centimeter precision. Normally, the precision of maps based on non-survey large-scale data (e.g., satellite images) is lower. City planning data contain proposals for developing urban areas. Most city planning maps are developed by digitizing handmade maps, using space images. Almost all boundaries have small discrepancies in comparison to cadastral maps. It is very important to use exact boundaries or their segments on city planning data from a cadastral map. The approach described in the paper enables us to resolve the problem described.

The developed method consists of several stages: converting polygon layers into topological data format; splitting polylines (polygon boundaries) into segments; defining corresponding maximal segments of polylines; moving segments of land-use boundaries without pairs on cadastral map boundaries and moving centroids of planning data polygons according to surrounding boundaries.

This paper is structured as follows: the related work is considered in Section 2. The source datasets are described in Section 3. The process of defining initial variables is described in Section 4. The algorithm of defining correspondent polylines (main part of the approach) is presented in Section 5. The process of compiling of the final map is described in Section 6. The results are discussed in Section 7.

II. RELATED WORK

Discrepancy problems on digital maps can be resolved in different ways. Common shape matching techniques are currently used in the raster and vector fields, and sometimes in combination with each other. Several common techniques in the field of Shape Similarity or Pattern Recognition could be applied to the various needs of the matched objects and relevant research questions.

Vector matching techniques can be divided into three main categories.

A. Feature-based matching

This group of methods is based on an object's geometry and shape. The degree of compatibility of objects is

 ISBN: 978-1-61208-383-4

determined by their geometry, size, or area. The process is carried out by a structural analysis of a set of objects and comparing whether similar structural analysis of the candidates fits the objects of the other data set [2][13]. In [15], comparison of objects is based on analysis of a contour distribution histogram. A polar coordinates approach for calculating the histogram is used. A method based on the Wasserstein distance was published by Schmitzer et al. [6]. A special shape descriptor for defined correspondent objects on raster images was developed by Ma and Longin [22]. Feature-based matching approaches do not allow for resolving our problem, because they have been developed mainly for single shapes; but, we can use them as part of our approach.

B. *Relational matching*

This group of methods takes objects' relationships into account. In [5], topological and spatial neighborly relations between two data sets, preserved even after running operations such as rotation or scale, were discovered. In relational matching, the comparison of the object is implemented with respect to a neighboring object. We can verify the similarity of two objects by considering neighboring objects. The problem of non-rigid shape recognition is studied by Bronstein et al. [4]; the applicability of diffusion distances within the Gromov-Hausdorff framework [4] and the presence of topological changes have been explored in this paper.

C. *Attributes-based matching*

Matching two data sets' objects by attributes could be very effective if a similar data model is used. Two types of attribute matching could be mentioned: Schema-based [11] and Ontology-based. In [16], an approach based on both types is presented. Attributes-based matching is a specific group of approaches; it can only be applied efficiently in special cases with special data. In most situations it is ineffective.

The merging and fusion of heterogeneous databases has been extensively studied, both spatially [10] and non-spatially [19]. The Map conflation method is based on data fusion algorithms; the aim of the process is to prepare a map which is a combination of two or more maps (often for updating an old map). Map conflation approaches have been presented in [12][7][18].

Computer Vision algorithms are popular in the field of data matching [17]. The Open Computer Vision (OpenCV) framework [3] is widely used today; it provides a number of "out-of-the-box" functions enabling us to detect and compare objects and bindings for popular programming languages (e.g., Python [9]). This makes the OpenCV framework very useful for data-matching tasks. Delaunay Triangulation [14] and Voronoi Polygons [1] are very useful techniques for working with discrete vector data and neighbor analysis. We should also note that in the practice - data is distributed in non-topological formats (e.g., Shape File format) and contains an embarrassment of data analysis, because of a surplus number of objects, duplication of primitives, e.g. polygon boundaries, unexpected gaps between objects etc. We need to use one of the topological data formats presented by Landa [21] to avoid these obstacles. Additionally, two perspective methods could be used in GIS data matching to reduce the time and computer resources required: Genetic Algorithms [20] help to avoid Brute-force operations in some cases; OpenCL technology [8] makes it possible to split a process into a huge number of parallel threads on a video card.

III. DATA SOURCE

For implementing and testing our approach, GIS data provided by Survey of Israel have been used. They contain cadastre and land-use city planning polygon shape files covering a part of Harish (a town in the Haifa District of Israel) (see Figure 1). From Figure 2, one can conclude that transformation of lines would not yield positive results, because the gaps are extremely variable - the curved parts of lines consist of different numbers of vertices; thus, even with correct parameters of transformation, the result would not be satisfactory.

IV. DEFINING INITIAL VARIABLES

Source shape files have been converted to GRASS GIS 7 topological data format [21]. Data preparation can be divided into 3 steps:

- Extracting polygon boundaries.
- Splitting polylines into a set of equidistant points. We have decided to use 2 meters between points. For depicting this parameter we will use the symbol d in the paper.
- Calculating an array of distances between the nearest points of two datasets. Setting of initial measures.

Several initial measures need to be calculated. Maximal distance (D_{max}) between the nearest points of two datasets and maximal standard deviation (σ_{max}) have been calculated. To calculate these parameters we need to create a list of 100 percentiles. Then we implement a loop from the first to the last percentile on the list. D_{max} equals percentile i and σ_{max} equals the double standard deviation of distance in the interval between percentiles number i and 100 if the standard deviation of distances between percentiles i-1 and i is more then 1. We calculate tail parameter (t) as follows: $t = D_{max}//d$. Tail defines a starting or ending segment of polyline which could be ignored.

We have developed a special shape descriptor (S), partially based on the descriptor presented in (Ma et al., 2011). The descriptor measures the similarity of polylines. Polylines are more similar if S is larger.

$$S=\sum(\exp(-(\log_{10}(1+dists\_a)-\log_{10}(1+dists\_b))^2)+ \exp(-(angles\_a-angles\_b)^2))/matrix\_size^2 \quad (1)$$

In Equation, 1 means matrix of ones, $\log_{10}$ - logarithm with base 10, dists_a – matrix of distances between all pairs of points laid on polyline a. dists_b – matrix of distances between all pairs of points laid on polyline b. If the number of points of a line is k, then matrix size is k×k. angles_a and

 ISBN: 978-1-61208-383-4

angles_b are matrices of angles in radians between all pairs of points of lines a and b, correspondingly.

A list containing pairs of point sets has been prepared, where all points laid on line A are closest to points laid on line B of another dataset. For each element of the list, two shape descriptors of tails with t number of points have been calculated and collected into a list of shape descriptors of tails. St_min, St_max – minimal and maximal elements of the list. Also, we use maximal tail standard deviation of point distances (σt), and its (maximal tail) maximal value – σt_max.

The list of initial variables has been calculated: Dmax=10.21, σmax=2.31,t=10.21//2=5, St_min=0, St_max=0.25.

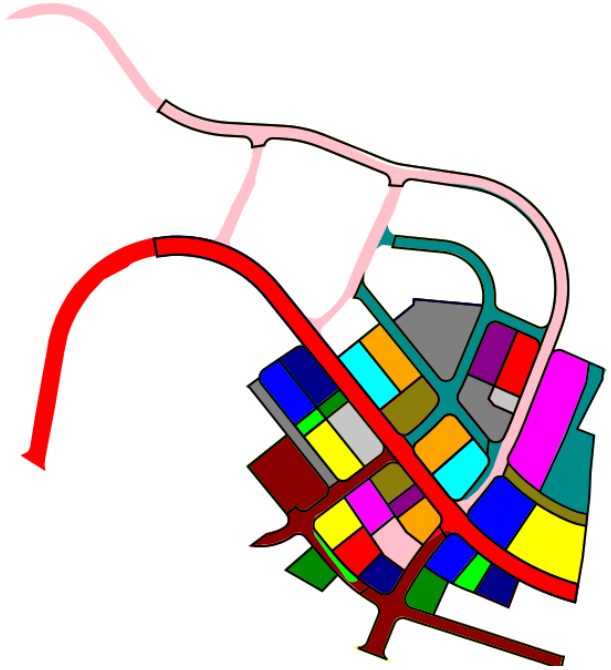

Figure 1. Source data: land-use city planning (color background) and cadastre (black outline) maps.

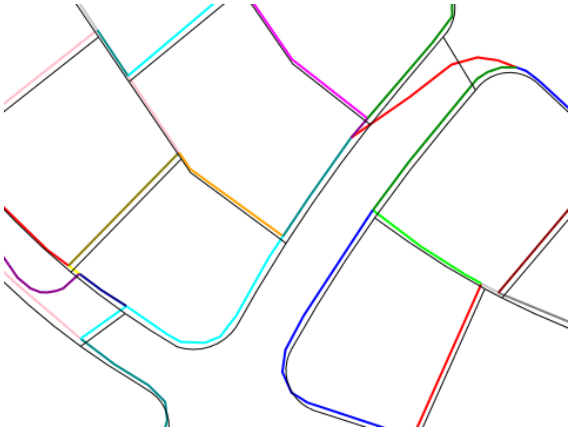

Figure 2. Positional discrepancies of city planning (color lines) and cadastre (black lines) datasets.

V. DEFINING CORRESPONDING LINES OF DATASETS

To define corresponding lines, we have developed a special descriptor based on several measures: distances between points, standard deviation of distances, shape descriptor. Figure 3 depicts the main idea – using equidistant points on a polyline to detect corresponding polylines, or segments of polylines. In the figure, a polyline of cadastral data set with nearest polylines of a city planning map are presented.

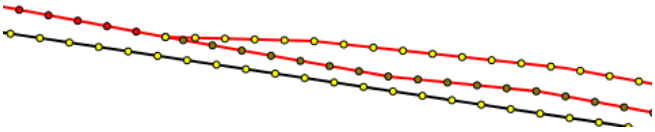

Figure 3. Equidistant points used to calculate similarity of polylines and polylines' segments. Red line – city planning dataset, black – cadastre.

The algorithm for line pairs searching is presented in pseudo code in Figure 4. An explanation follows the listing.

Figure 4. Searching for equal polylines or polylines' segments.

```
Foreach idA,idB in get_ids_of_closest_lines(){
   Pts_A    = get_points('city planning',idA)
   Pts_B =    get_points('cadastre',idB)
   If min(len(Pts_A),len(Pts_B)) > tail {
      Foreach segm in get_segments(Pts_A,Pts_B){
         Pts_A_segm=segm['Pts_A']
         Pts_B_segm=segm[' Pts_B']
         Result_line_pair=find_pair(Pts_A_segm,Pts_B_segm)   } }}
Function find_pair(PtsA,PtsB) {
  If (distance(PtsA[0],PtsB[0]) >
   distance(PtsA[0], PtsB[-1])){
   PtsA=reverse(PtsA)   }
  Length=min(len(PtsA), len(PtsB))
  Global_measures=[ ]
  Foreach l in reverse([tail,…,length]){
    Local_measures=[ ]
    Foreach i in [0,…,len(PtsA)-tail]{
      Foreach j in [0,…,len(PtsB)-tail]{
        cur_measure=Calc_measures(PtsA,PtsB,i,j,l)
        if (cur_measure[0]<max_stand_dev and
            cur_measure[1]<max_distance){
          Local_measures.append(cur_measure)} } }
   Global_measures
.append(Find_local_indicator(Local_measures))
   If    Global_measures    and    (l==length    or
len(Global_measures)>tail){
      Gen_desc_list=[calculate_global_indicator(cur) for cur in
                     Global_measures]
      If max(Gen_desc_list[:-tail])> max(Gen_desc_list[-tail:]){
        Return
Global_measures[index_of_maximal(Gen_desc_list)]} }}
Function Calc_measures(PtsA,PtsB,i,j,l){
  cur_PtsA= PtsA[i:i+l]
  cur_PtsB= PtsB[j:j+l]
  dists=Distances(cur_PtsA,cur_PtsB)
  Return [stand_dev(dists),max(dists),
    min(dists),delta_x,delta_y,
   get_max_stddev_of_tailes(cur_PtsA,cur_PtsB),
   get_min_shape_descr_of_tails(cur_PtsA,cur_PtsB),
   get_shape_descriptor(cur_PtsA,cur_PtsB),
   i, j, l]}
```


 ISBN: 978-1-61208-383-4

The pseudo-function gets_the 'id's_of_closest_lines () and returns a pair of neighboring lines' ids, points which are closest. Usually, for one line A, several pairs of ids can be defined (idA1-idB1, idA1-idB2,..). All id pairs are processed. Pts_A – points of a city planning dataset are situated on a line with id idA, Pts_B – points of line idB (cadastral map). The pseudo function gets_segments (Pts_A, Pts_B) and splits lines into segments at intervals where the distance between nearest points is more than D_{max}. In the first line of the pseudo function - finding_pairs (PtsA,PtsB) - we test distances from start point of line A to start and end points of line B. If start-start distance is more than start-end, we invert the order of points in line A. Then we set l,i,j variables: l – length of line, i - number of starting point on line A, j - number of starting point on line B. The function Calc_measures (PtsA, PtsB, i, j, l) and calculates a set of parameters (standard deviation of distances, shape descriptor, minimal shape descriptor of line tails, minimal and maximal distance between points). This enables us to define similarity of line A segment from i to i+l and for line B - from j to j+l. Variables i and j which define the optimal segment (pseudo function Find_local_optimal (Local_measures)) have been found for each possible length l using (2).

$$Loc\_Ind=(d-D_{max})/(-D_{max})+(s-S_{t\_max})/(S_{t\_max}-S_{t\_min}) \quad (2)$$

The meaning of parameters in (2): d – maximal distance between points of lines A and B for (l,i,j), s - minimal tail shape descriptor. In this step we have a Global_measures list containing elements which correspond to some l and contain measures of line segments with maximal indicator Loc_Ind derived from the list (Local_measures) with variable (i,j). This process is illustrated in Figure 5 and 6 (segment length is 20 meters). The next stage is defining optimal segment length. In the previous stage we defined optimal segments i,j for some length l by calculating local indicator Loc_Ind. To define optimal segment length we use global indicator G_Ind; its formula is presented as (3).

$$G\_Ind=((\sigma_t-\sigma_{max})/(-\sigma_{max}))+(d-D_{max})/(-D_{max})+ \quad (3)$$
$$+(s-S_{t\_max})/(S_{t\_max}-S_{t\_min}))\cdot 2+(1-l)/whole\_segment\_length$$

In the Equation, σ_t means maximal standard deviation of point distances of line segments' tails; for more details see Section 4 and (2). The resulting optimal line length is defined by maximal global indicator G_Ind. The process is illustrated in Figures 7 and 8.

It is obvious that optimal segment length is 41 meters (element with maximal G_Ind, according to Figure 8).

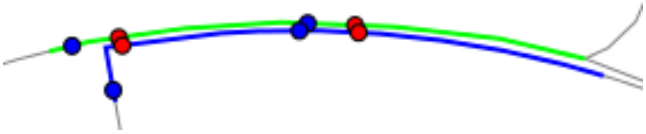

Figure 5. Segment of line A (city planning) – green; segment of line B (cadastre) – blue. Start and end point of the most similar line segments are red points (i=6,j=6, Loc_ind=0.86); blue points – i=2, j=1, Loc_Ind=0.026.

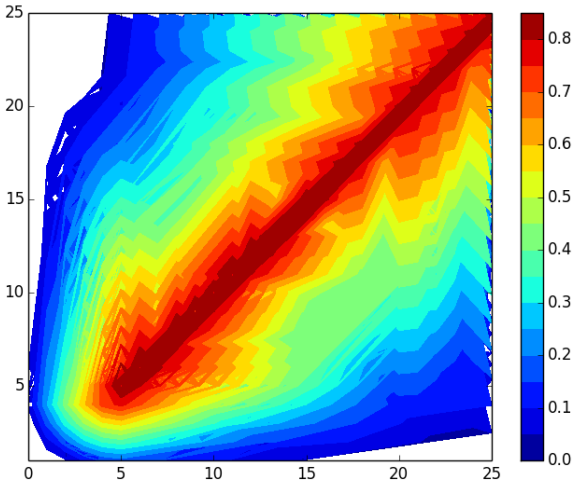

Figure 6. Plot of indicator Loc_Ind: X axis – i, Y axis - j.

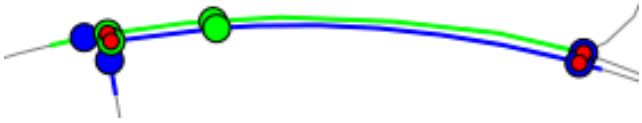

Figure 7. Segment of line A (city planning) – green; segment of line B (cadastre) – blue. Nodes of the most similar line segments with different lengths of segment: red points – l=41,i=5,j=5,G_Ind=2.35; green points – l=10,i=5,j=5 G_Ind=1.69; blue points – l=43,i=3,j=3 G_Ind=1.64.

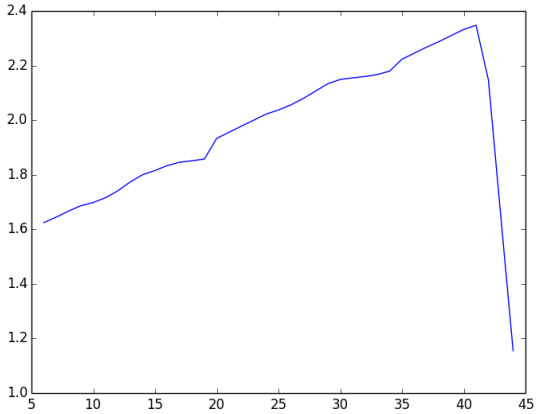

Figure 8. Plot of indicator G_ind: X axis – segment length (in meters), Y axis – G_Ind.

VI. COMPILING A FINAL MAP

At this point, we have the pairs of corresponding segments. Some segments are overlapped; to resolve conflicts, a special parameter was developed:

$$P= (l-min\_len)/range\_len+\sigma/(-\sigma_{max}) \quad (4)$$

where, l is length of line of one of the lines in a line pair, min_len – minimal length of line of all line pairs, range_len – range of length of all line pairs. A line pair with maximal P will be saved, others will be removed. The process is shown in Figure 9.

 ISBN: 978-1-61208-383-4

After removing overlapping line pairs, we can use a correspondent line segment of the cadastral dataset instead of the city-planning dataset.

The lines and line segments of the city-planning dataset without the corresponding lines of the cadastral dataset have been moved. Delta X and delta Y have been calculated as average delta X and delta Y of neighboring nodes of line pairs. Unclosed boundaries of polygons have been closed by moving nodes of an unclosed line to the nearest node of a neighboring line (see Figure 10). Centroids of polygons of the city-planning dataset have been moved according to average delta X and delta Y of those boundaries weighted by lengths.

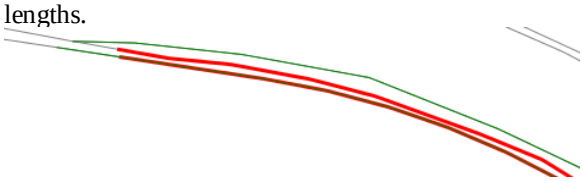

Figure 9. Overlaped line pairs: red line pair – P=1.23, green line pair – P=1.09 . Green line pair will be removed.

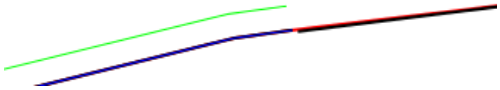

Figure 10. Moving segments without pairs and closing boundaries: green – lines that do not have pairs in cadastral dataset, blue – moved green lines, red – closing boundary by moving nodes, black – cadastral pair of city-planning line segments.

VII. RESULTS AND CONCLUSION

Results are presented in Figure 11 and Figure 12. We can conclude that most line segments have been taken from the cadastral dataset; others have been transformed to correspond with cadastral polyline segments. The result looks satisfactory; the final map is holistic and does not contain significant deficiencies. In the future, we need to test the approach with more datasets and different parameters, to compare with other approaches, to reduce calculation speed (the execution currently requires about one hour, too long for such a small dataset), and to investigate the reasons for deficiencies and unexpected geometries on the final map.

An approach for improving the accuracy of polygons' data is presented. The land-use city planning dataset locations have been corrected according to the cadastral dataset. The polylines' segments along the polygons have been split by equidistant points. Analysis has been performed using statistics based on the points of the neighboring polylines of the two datasets. A set of parameters has been used: shape descriptor of polyline segments, standard deviation of point distances, minimal and maximal point distances, standard deviation of segment tails, etc. A set of correspondent polyline segments has been found using special indicators, which enables us to find optimal segments from the list of polyline segments with different length and starting point. The polyline segments of the city planning data with similar/identical parameters to the segments of the cadastral data were linked to these segments (defining counterpart segments). Segments without a counterpart have been transformed.

To implement the approach, we used Python 2.7 programming language (with numpy, scipy and matplotlib additional libraries), GRASS GIS 7.1, and Debian GNU/Linux 8 operating system.

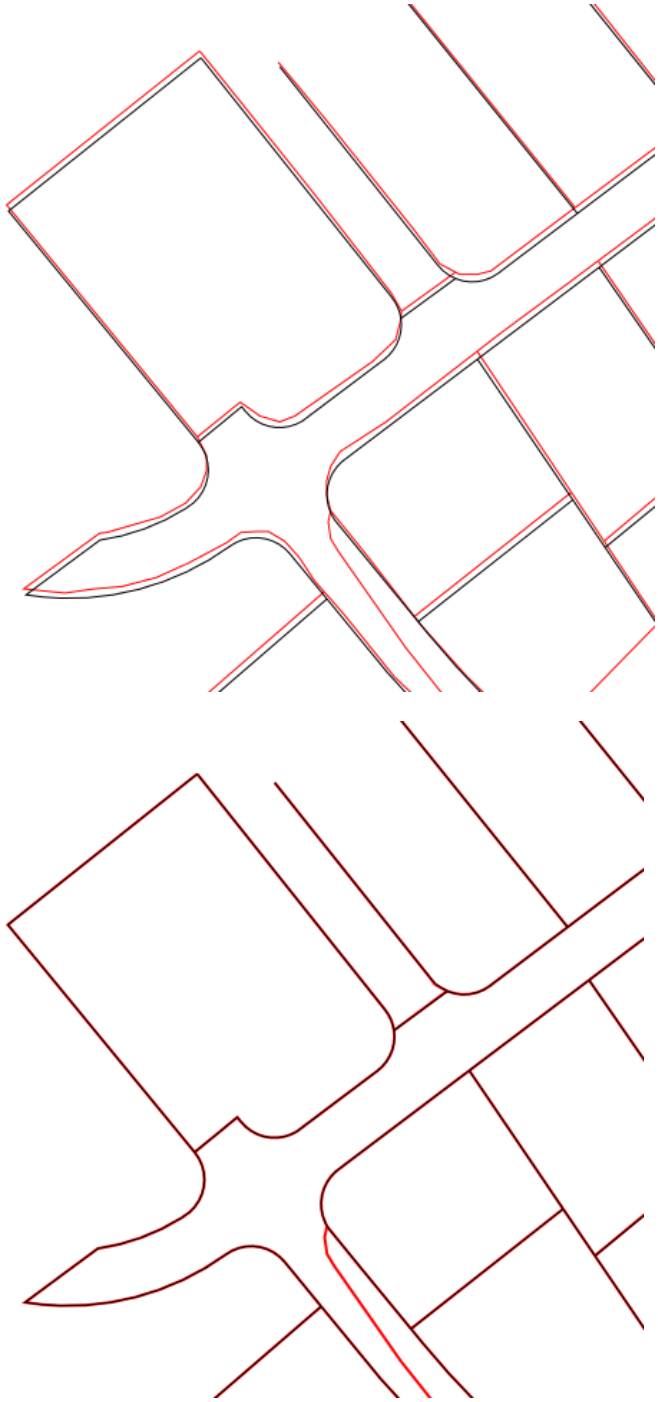

Figure 11. Results. Extent 1. Red – land-use city planning dataset, black – cadastral dataset. Upper – original data, lower –result.

 ISBN: 978-1-61208-383-4

ACKNOWLEDGEMENT

This research was supported by the Survey of Israel as a part of Project 2019317. The authors would like to thank the Survey of Israel for providing the financial support and data for the purpose of this research.

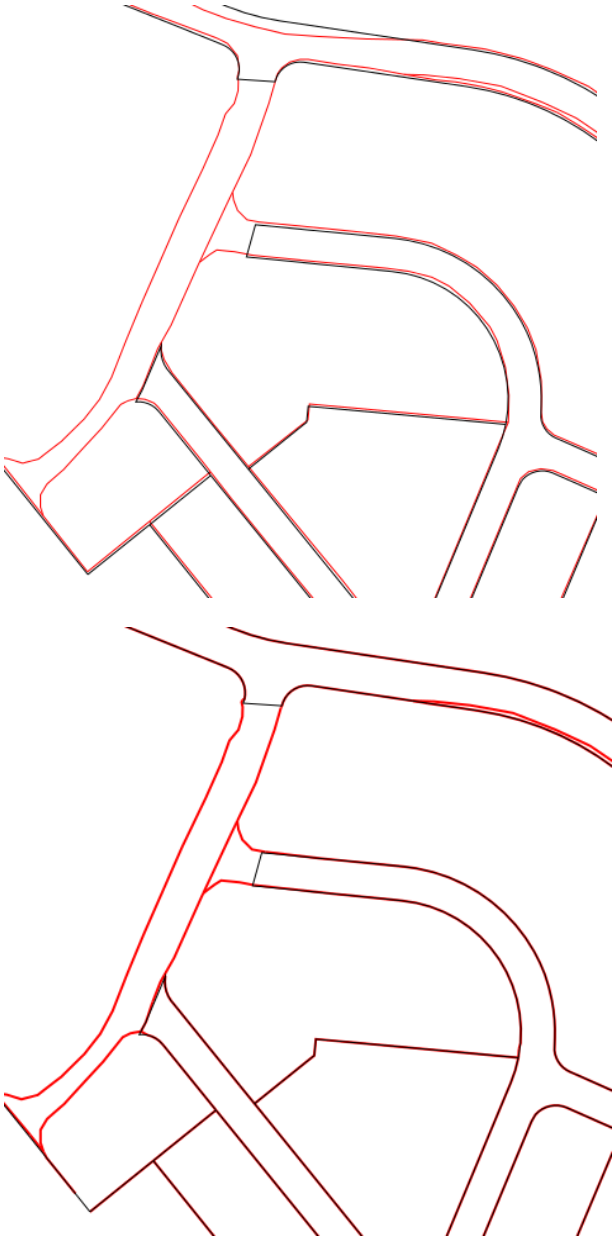

Figure 12. Results. Extent 2. Red – land-use city planning dataset, black – cadastral dataset. Upper – original data, lower –result.

REFERENCES

[1] F. Aurenhammer, "Voronoi diagrams—a survey of a fundamental geometric data structure," ACM Computing Surveys (CSUR), vol. 23(3), 1991, pp. 345-405.

[2] S. Belongie, J. Malik, and J. Puzicha, "Shape matching and object recognition using shape contexts," IEEE Trans. on Pattern Analysis and Machine Intelligence, 24(4), 2002, pp. 509—522.

[3] G. Bradski and A. Kaehler, "Learning OpenCV: Computer vision with the OpenCV library," O'Reilly Media, Inc, 2008.

[4] A. Bronstein, R. Kimmel, M. Mahmoudi, and G. Sapiro, "A Gromov-Hausdorff framework with diffusion geometry for topologically-robust non-rigid shape matching," International Journal of Computer Vision, vol. 89(2-3), 2010, pp. 266-286.

[5] X. Chen, "Spatial relation between uncertain sets," International archives of Photogrammetry and remote sensing, vol. 31(B3), Vienna, 1996, pp. 105-110.

[6] Schmitzer, Bernhard, and S. Christoph, "Object segmentation by shape matching with Wasserstein modes," Energy Minimization Methods in Computer Vision and Pattern Recognition, Springer Berlin Heidelberg, 2013.

[7] S. Filin and Y. Doytsher, "The detection of corresponding objects in a linear-based map conflation," Surveying and land information systems, vol. 60(2), 2000, pp. 117-127.

[8] B. Gaster, L. Howes, D. Kaeli, P. Mistry, and D. Schaa, "Heterogeneous Computing with OpenCL: Revised OpenCL1," Newnes, 2012.

[9] J. Howse, "OpenCV Computer Vision with Python," Packt Publishing Ltd, 2013.

[10] C. Parent and S. Spaccapietra, "Database integration: the key to data interoperability," Advances in Object-Oriented Data Modeling, M. P. Papazoglou, S. Spaccapietra, Z. Tari (Eds.), The MIT Press, 2000.

[11] E. Rahm and P. Bernstein, "A survey of approaches to automatic schema matching," The International Journal on Very Large Data Bases (VLDB), vol. 10(4), 2001, pp. 334–350.

[12] A. Saalfeld, "Conflation-automated map compilation," International Journal of Geographical Information Science (IJGIS), vol. 2 (3), 1988, pp. 217–228.

[13] E. Safra, , Y. Kanza, Y. Sagiv, C. Beeri, and Y. Doytsher, "Ad-hoc matching of vectorial road networks," International Journal of Geographical Information Science, iFirst, 2012, pp. 1–40, ISSN: 1365-8816, ISSN: 1362-3087.

[14] J. Shewchuk, "Triangle: Engineering a 2D quality mesh generator and Delaunay triangulator," Applied computational geometry towards geometric engineering, Springer Berlin Heidelberg,1996, pp. 203-222.

[15] X. Shu and X. Wu. "A novel contour descriptor for 2D shape matching and its application to image retrieval", Image and vision Computing, vol. 29.4, 2011, pp. 286-294.

[16] P. Shvaiko and J. Euzenat, "A survey of schema-based matching approaches," Journal on Data Semantics IV, Springer Berlin Heidelberg, 2005, pp. 146-171.

[17] C. Steger, M. Ulrich, and C. Wiedemann, Machine vision algorithms and applications, Weinheim: wiley-VCH, 2008, pp. 1-2.

[18] V. Walter and D. Fritsch, "Matching spatial data sets: a statistical approach," International Journal of Geographical Information Science (IJGIS), vol. 13 (5), 1999, pp. 445–473.

[19] G. Wiederhold, "Mediation to deal with heterogeneous data sources," Interoperating Geographic Information System, 1999, pp. 1–16.

[20] I. Wilson, J.M. Ware, and J.A. Ware, "A genetic algorithm approach to cartographic map generalisation" Computers in Industry, vol. 52(3), 2003, pp. 291-304.

[21] M. Landa, "GRASS GIS 7.0: Interoperability improvements," GIS Ostrava, Jan. 2013, pp.21-23.

[22] T. Ma and J. Longin, "From partial shape matching through local deformation to robust global shape similarity for object detection," Computer Vision and Pattern Recognition (CVPR), IEEE Conference on. IEEE, 2011, pp. 1441-1448.

 ISBN: 978-1-61208-383-4