Polygon Consensus: Smart Crowdsourcing for Extracting Building Footprints from Historical Maps

Benedikt Budig,* Thomas C. van Dijk, Fabian Feitsch
Chair for Computer Science I
Universität Würzburg
benedikt.budig@uni-wuerzburg.de

Mauricio Giraldo Arteaga
New York Public Library
mauriciogiraldo@nypl.org

ABSTRACT

Over the course of three years, the New York Public Library has run a crowdsourcing project to extract polygonal representation of the building footprints from insurance atlases of the 19th and early-20th century. As is common in crowdsourcing projects, the overall problem was decomposed into small user tasks and each task was given to multiple users. In the case of polygons representing building footprints, it is unclear how best to integrate the answers into a majority vote: given a set of polygons ostensibly describing the same footprint, what is the consensus? We discuss desirable properties of such a "consensus polygon" and arrive at an efficient algorithm. We have manually evaluated the algorithm on approximately 3,000 polygons corresponding to 200 footprints and observe that our algorithmic consensus polygons are correct for 96% of the footprints whereas only 85% of the (input) crowd polygons are correct.

CCS Concepts

•**Applied computing** → **Digital libraries and archives;** •**Information systems** → **Geographic information systems; Crowdsourcing; Information extraction;**

Keywords

Smart Crowdsourcing; Historical Maps; Polygon Consensus; Deep Georeferencing

1. INTRODUCTION

The New York Public Library (NYPL) has an extensive collection of insurance atlases from the 19th and early 20th century. One of their goals is to extract (vector) polygon and attribute data from these maps. The collection includes tens of thousands of sheets from 1853 to 1930 organized in 200 atlases. Feature extraction was originally based on staff and volunteer work to manually trace polygons in a custom web-based GIS. Using this manual process, it took three years to extract about 179,000 polygons across three atlases. At that pace, it would be impossible to extract the bulk of the data in any reasonable amount of time. In 2013, NYPL Labs started development of a semi-automatic pipeline to digitize the contents of these atlases, which includes a crowdsourcing website called *Building Inspector*.[1]

Figure 1: *Building Inspector* presents a polygon that needs to be "fixed". Users can add, move and delete vertices such that it matches the building footprint.

In this paper we consider an algorithmic challenge that arises in analyzing the Building Inspector data. We call it the *polygon consensus* problem and it is informally stated as follows: given a set of polygons that are supposed to represent the same object, compute a single polygon that represents the majority opinion. We present an algorithmic solution for this problem, based on a heuristic currently employed by the New York Public Library.

Crowdsourcing for geographic information is not new, and under the term *volunteered geographic information* (VGI) represents an expansive field of work, with the successful OpenStreetMap project [8] as its poster child. See for example Goodchild [7] for a general review of the concept, placing it within a context of more traditional citizen science and the role of the general public in geographic observation.

Particularly related to the current paper is the work of Squire et al. [13], who integrate multiple reports on the spatial extent of an environmental contamination. They use the term *consensus polygon* and arrive at a definition different from ours. This difference stems from the application, where they are most concerned with area coverage rather than with the shape. As a result, using their approach on the Building Inspector data would not give good results.

*Corresponding author.

SIGSPATIAL'16, October 31-November 03, 2016, Burlingame, CA, USA

ISBN 978-1-4503-4589-7/16/10...$15.00

DOI: http://dx.doi.org/10.1145/2996913.2996951

[1] http://buildinginspector.nypl.org/

Image segmentation and feature extraction from raster map images (including scanned historical maps) is an active field of research. Chiang et al. [3] present a comprehensive survey on digital map processing techniques. There are many approaches [10, 11, 12], but none of them solves the problem so well that our postprocessing is unnecessary. Indeed, their polygons could be used as input for a crowdsourcing project.

In computational geometry, there exists some research on consensus of geometric objects, such as median trajectories [14, 2] and mean consensus trajectories [4]. These have no direct bearing on our building footprint problem, since they do not conform to our desiderata in Section 2.1.

2. POLYGON CONSENSUS

This paper concerns a two-stage crowdsourcing project. Its web-based software supports various annotation tasks on scans of the maps in the system. Before the crowd gets involved, a pipeline of image processing tools is used to extract polygons from the map image [6]. The set of extracted polygons unfortunately contains many errors, since this problem is truly hard for computer vision. In the first crowd task, the user is shown an automatically detected polygon overlaid on the map image and has to decide whether it matches a building (YES), matches at least part of a building but needs fixing (FIX) or does not match a building at all (NO). Polygons for which a majority of at least three users vote YES are assumed to be correct; majority-NO polygons discarded as useless. In this paper we focus on the interesting case of FIX polygons. These are processed in a second crowd task, where users are shown a polygon and are instructed to make it match the underlying building footprint by adding, deleting and moving vertices (see Figure 1).

As we will see in Section 3, only about 85% of the "fixed" polygons we get from the crowd are indeed correct. In order to increase reliability, multiple independent users are given the same polygon to fix. Hence as output from this step we obtain, for every automatically detect polygon, a set of polygons created by different users that are each supposed to be a fixed version of the detected polygon. We call this a *group* of polygons, and its elements *user polygons*. All of these polygons will differ, at least slightly in the exact position of manually-edited vertices, and possibly also in the user's judgment of *how* to fix the polygon. This is our problem: given a set of user-created polygons, how can we find one *consensus* polygon that represents best what the majority of the users intended?

2.1 Modeling

We start with two general considerations about the properties a "consensus polygon" should have – these are vague for now. First, a consensus polygon should be shaped similarly to the majority of the polygons in the group, even in the presence of outliers. Second, if vertices from different polygons are near each other and, structurally within their polygon, serve a similar function, then a consensus polygon should consider these as noisy representations of an actual vertex. Such a set of vertices should result in a *single* vertex of a consensus polygon.

We do not give a formal definition of polygon consensus as an optimization problem; this would be an interesting avenue for future work, particularly because the appropriate definition of consensus may depend on the application and spelling out explicit optimization criteria and constraints is good practice. At present, we give a reasonable heuristic algorithm and show that it gives good results on the Building Inspector data. This algorithm was previously only described online;[2] we describe it here with minor improvements and clarifications as compared to the version available online.

2.2 Vertex Voting

The basic vertex voting algorithm takes a set of polygons as input and computes a consensus polygon (or decides that there is no consensus): first it clusters the vertices of the input polygons and then finds a cycle through the clusters that is supported by many input polygons.

The algorithm starts by determining a set of candidate vertices for the consensus polygon. For this purpose, it takes the set of all vertices from the input polygons and clusters it using DBSCAN [5]. It requires, as a parameter, the distance threshold ε for points to be considered "near." We set its *MinPts* parameter to 1 unless otherwise noted. Let $\mathcal{C}$ be the resulting set of clusters. In the second step, the algorithm searches for a cycle through the clusters, which will define the edges of the consensus polygon. This cycle is heuristically constructed in the following way, after picking a consistent orientation of the polygons.

Let (u, v) be any arc in an input polygon, and let C be the cluster that v is in: then we say that u votes for C. The algorithm starts in an arbitrary cluster containing the most vertices, then it iteratively goes to the cluster that has the most votes among the vertices in the current cluster (breaking ties arbitrarily). This process is terminated when a cluster is visited for the second time, and the resulting cycle of clusters is taken as the combinatorial structure of the consensus polygon. (Note that we may return to a different cluster than the first.) If the resulting cycle contains at least three clusters, we construct the output by connecting the centroid of each cluster along the cycle. Otherwise, the algorithm concludes there is no consensus.

Let n be the number of vertices in all input polygons combined. Without a data structure to speed up region queries, the runtime of DBSCAN is $\Theta(n^2)$. The general observation that a runtime of $\mathcal{O}(n \log n)$ can be expected in many cases [5] does not hold for our data: if at least a constant fraction of the polygons contribute a vertex to at least a constant fraction of the clusters, each individual region query is likely to return $\Omega(n)$ vertices. In this way, the runtime of clustering dominates that of tallying the votes and performing the graph search and the total runtime is $\mathcal{O}(n^2)$.

It is unfortunate that the parameter ε makes the algorithm scale dependent. However, for a given data set we may well be able to pick a good value for ε, as it reflects a rather direct property of the input data: the amount of "noise" we expect on vertices of the input polygons. For cases where the scale of the input polygons is unknown or varies by group, a single fixed ε is problematic and future work could consider picking ε automatically.

2.3 Preclustering

Assume that the input group contains outliers. This can lead the basic vertex voting algorithm to unreasonable solutions. We therefore add a filtering step before the clustering.

[2] `http://nbviewer.jupyter.org/gist/mgiraldo/a68b53175ce5892531bc`

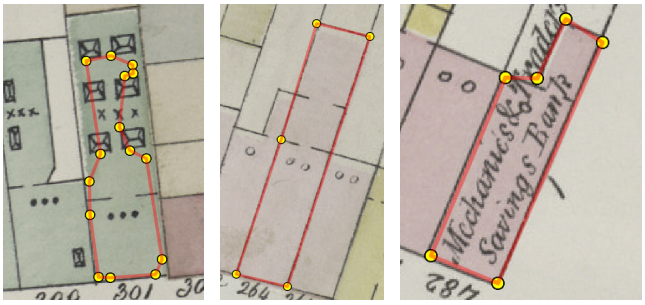

Figure 2: The polygon on the left is semantically incorrect because it does not represent a building footprint. The polygon in the middle is also incorrect, because it has one vertex too many. The polygon on the right has inaccurate vertex positions, but is semantically correct because it covers the right shape and has exactly one vertex for each corner.

For each polygon in the group, we consider the centroid of its vertices. If two centroids are far apart, it is unlikely that their polygons describe the same shape (building footprint). Hence we cluster the centroids: we use DBSCAN with another distance threshold φ. We continue using only the polygons in the largest cluster.

Unfortunately, the parameter φ does not have a clean interpretation in the application domain and is less intuitive than ε. Still, this variant of the algorithm is currently used in production in the Building Inspector, where it slightly improves the quality of results (see Section 3).

3. EXPERIMENTS

We have evaluated the above algorithms on data from the Building Inspector project. These data are publicly available through an open API.[3] At the time of our experiments, there were 5,834 groups labeled FIX, containing 58,651 user polygons. We sampled a random subset of 200 groups for this paper. We have run the two variants of our algorithm on all 200 groups, resulting in 400 consensus polygons. Together with the corresponding 1,278 user polygons and the 200 polygons detected by the computer vision step, this makes a total of 1,878 polygons assessed in this section.

3.1 Ground Truth

We call a polygon *correct* if it traces a single building footprint and has exactly one vertex for each corner of the footprint (and no vertices where there are no corners). See Figure 2 for examples of correct and incorrect polygons. Three of the authors independently evaluated all 1,878 polygons by manual inspection. To avoid influencing our judgment, all polygons were randomly shuffled and it was unclear to the judges which kind was which. For polygons with conflicting votes, the majority answer was taken.

Note that this evaluation only addresses the correctness of individual polygons: it does not take into account whether that footprint represents a reasonable consensus for its group or not. This way we we do not have to define what consensus means: all user polygons are supposed to mark the same object, so if an output polygon is a correct, then it is likely to be an appropriate consensus.

[3] `http://buildinginspector.nypl.org/data`

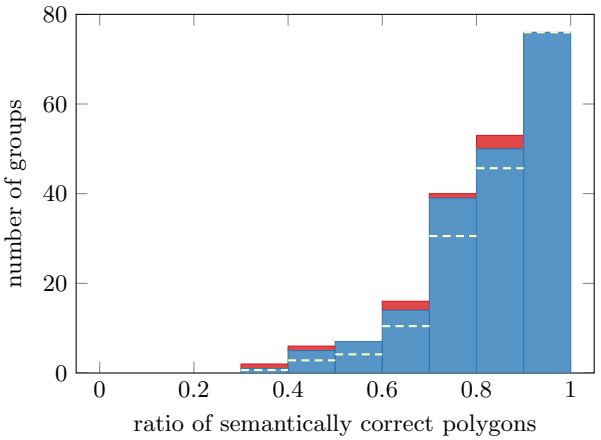

Figure 3: Ratio of correct user polygons (in 10 bins). The dashed yellow lines indicate the expected number of correct groups for the baseline. Blue indicates the number of correct groups for `Voting`; red is the number of incorrect groups. The blue area above the yellow line reflects the gain from our algorithm.

3.2 Semantic Correctness

Of the 200 polygons that were detected through computer vision, we found that none are semantically correct. This is expected, since we only look at polygons that the crowd marked FIX, and confirms that our notion of correctness is consistent with the crowd. Of the user polygons, 84.7% are correct. In many groups (129 of 200), the users were able to solve the problem consistently, in the sense that more than 80% of the user polygons were correct in the group. The remaining groups raised more difficulties; for 15 of them, less than half of the user polygons were correct (see Figure 3).

The accuracy of 84.7% is a reasonable quality level for individual tasks in a crowd sourcing project and gives the hope that integrating multiple user answers for the same footprint can be used to increase the success rate by taking some kind of majority vote: this is what we do in this paper.

We consider an algorithm *successful* if its output polygon is correct. As baseline, consider the algorithm that takes a random user polygon from the group. This could be considered "consensus" in the sense that if many polygons agree, one of those is likely to be picked. In expectation, this algorithm has success rate 85.2% on our data. (This is different from the base accuracy because of the grouping.)

We now consider the voting algorithm with preclustering (`Voting`). It is successful on 96% of the groups. This can be considered a success for the whole project: the 11.3 percentage-point increase in correct polygons over raw crowd data represents more than $6,600$ additional footprints when extrapolated to the full data set. Figure 3 shows more detailed data. However, this variant of the algorithm requires two parameters, which can be problematic depending on the data set to be processed.

Next we consider the voting algorithm without preclustering (`VotingRaw`). It has a parameter fewer since it has no filtering step. Its success rate is 94%: still well above the users' base rate of 84.7%. The `Voting` algorithm is not strictly better, though: on 7 groups only `Voting` was successful, and on 3 only `VotingRaw` was.

In these first two experiments, we have set ε (and also φ in the case of `Voting`) to effective values by hand. The experiments have shown we can remove the filtering step at only a minor cost. This leaves ε, for which a reasonable value can be picked in many cases, based on manual inspection of the map images and the user polygons.

3.3 Geometric Precision

We also evaluate the geometric precision of our consensus polygons. Consider the underlying map images and the brightness of the pixels under the edges of the polygon: if the edges indeed follow the ink marking the outline of a building footprint, these pixels can be assumed to be relatively dark. In the following, we consider brightness on a scale from 0 (black) to 1 (white).

The polygons detected by computer vision (and marked as FIX) have an average brightness of 0.68, which we will see is high. This means that these polygons do not align well with the underlying building footprints in the maps, because a significant fraction of their edges cross over background paper rather than inked lines. The correct user polygons have a much better score (0.49), which suggests that assessing the brightness is actually a sensible measure of goodness. However, the semantically incorrect user polygons also have a good score (0.54): this suggests that a brightness analysis alone will not solve the semantics question of correctness. Indeed, in 15 of the 200 groups in our data set, the brightest polygon is incorrect: this is only slightly better than the base rate.

Statistics confirm that brightness cannot be considered a good feature for determining the semantic correctness of a polygon. Consider a classifier that, given a threshold t, assumes that all polygons with brightness lower than t are correct. The quality of a classifier can be evaluated using a *receiver operating characteristic* curve (ROC curve). Following standard practice, we calculated the *area under the curve* (AUC), which is 0.643. Generally, an AUC value between 0.5 and 0.7 is considered "poor discrimination, not much better than a coin toss" [9]. This shows that the correctness of a polygon can not reasonably be judged by its brightness alone.

The score of the consensus polygons calculated by our algorithms is each better than the average score of USER polygons (0.440 for the correct polygons). This shows that the consensus polygons outperform the users' base rate not only in terms of semantic correctness, but also in geometric precision: manual inspection confirms that the algorithmic consensus polygons are generally better aligned to the underlying image.

4. CONCLUDING REMARKS

We considered a data set gathered in a specific crowdsourcing project at the New York Public Library. Algorithmic analysis of this data results in higher-quality output, increasing the value of the gathered data. We propose the general usefulness of such *smart crowdsourcing*, where the user task is not simply a multiple-choice question, and where the integration of the responses may be nontrivial. This will be important for the development of successful crowdsourcing projects for spatial information, since such information can be hard to capture in discrete multiple-choice questions. The algorithms and techniques in this paper should be generally applicable, though the image processing to extract the initial polygons was bespoke and might not readily generalize to other maps.

The map sheets in the Building Inspector project were already georectified as part of their digitization, prior to our involvement. In a sense, the feature extraction can be seen as deep georeferencing, where we do not just have the map image in a known coordinate system, but also know about the object-level semantic elements in the map. This enables rich data experiences such as `http://spacetime.nypl.org/` and virtual reality applications [1].

Acknowledgments.

We thank Michael Resig and Alexander Wolff for fruitful discussion and helpful comments.

5. REFERENCES

[1] C. Balletti, L. Galeazzo, C. Gottardi, F. Guerra, and P. Vernier. New technologies applied to the history of the Venice Lagoon. In *Proc. 11th ICA Conf. on Digital Appr. to Cart. Heritage*, pages 182–190, 2016.

[2] K. Buchin, M. Buchin, M. van Kreveld, M. Löffler, R. I. Silveira, C. Wenk, and L. Wiratma. Median Trajectories. *Algorithmica*, 66(3):595–614, 2013.

[3] Y.-Y. Chiang, S. Leyk, and C. A. Knoblock. A Survey of Digital Map Processing Techniques. *ACM Comp. Surv.*, 47(1):1:1–1:44, 2014.

[4] L. De La Cruz, S. Kobourov, S. Pupyrev, P. S. Shen, and S. Veeramoni. Computing Consensus Curves. In *Proc. SEA'14*, volume 8504, pages 223–234, 2014.

[5] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *Proc. KDD'96*, pages 226–231, 1996.

[6] M. Giraldo Arteaga. Historical Map Polygon and Feature Extractor. In *Proc. MapInteract'13*, pages 66–71, 2013.

[7] M. F. Goodchild. Citizens as sensors: the world of volunteered geography. *GeoJournal*, 69(4):211–221, 2007.

[8] M. Haklay and P. Weber. OpenStreetMap: User-Generated Street Maps. *Pervasive Comp.*, 7(4):12–18, 2008.

[9] D. W. Hosmer Jr and S. Lemeshow. *Applied Logistic Regression*. John Wiley & Sons, 2004.

[10] S. D. Laycock, P. G. Brown, R. G. Laycock, and A. M. Day. Aligning Archive Maps and Extracting Footprints for Analysis of Historic Urban Environments. *Computers & Graphics*, 35(2):242–249, 2011.

[11] Y. Liu. An automation system: generation of digital map data from pictorial map resources. *Pattern Recognition*, 35(9):1973–1987, 2002.

[12] T. Miyoshi, W. Li, K. Kaneda, H. Yamashita, and E. Nakamae. Automatic Extraction of Buildings Utilizing Geometric Features of a Scanned Topographic Map. In *Proc. ICPR'04*, volume 3, pages 626–629, 2004.

[13] S. Squire, M. H. Ramsey, M. J. Gardner, and D. Lister. Sampling proficiency test for the estimation of uncertainty in the spatial delineation of contamination. *Analyst*, 125:2026–2031, 2000.

[14] M. van Kreveld and L. Wiratma. Median trajectories using well-visited regions and shortest paths. In *Proc. 19th ACM SIGSPATIAL'11*, pages 241–250, 2011.