POLYGONIZATION OF BINARY CLASSIFICATION MAPS USING MESH APPROXIMATION WITH RIGHT ANGLE REGULARITY

Onur Tasar, Emmanuel Maggiori, Pierre Alliez, Yuliya Tarabalka

Université Côte d'Azur, INRIA, TITANE team, 06902 Sophia Antipolis, France
Email: {firstname.lastname}@inria.fr

ABSTRACT

One of the most popular and challenging tasks in remote sensing applications is the generation of digitized representations of Earth's objects from satellite raster image data. A common approach to tackle this challenge is a two-step method that first involves performing a pixel-wise classification of the raster data, then vectorizing the obtained classification map. We propose a novel approach, which recasts the polygonization problem as a mesh-based approximation of the input classification map, where binary labels are assigned to the mesh triangles to represent the building class. A dense initial mesh is decimated and optimized using local edge and vertex-based operators in order to minimize an objective function that models a balance between fidelity to the classification map in ℓ_1 norm sense, right angle regularity for polygonized buildings, and final mesh complexity. Experiments show that adding the right angle objective yields better representations quantitatively and qualitatively than previous work and commonly used polygon generalization methods in remote sensing literature for similar number of vertices.

Index Terms— Polygonization, vectorization, remote sensing, classification maps, mesh approximation, right angles

1. INTRODUCTION

With the advent of deep neural networks and their applications to remote sensing data, it has already been shown that classification maps with high accuracy can be obtained [1, 2].

In order to generate a digitized representation the most straightforward approach is to vectorize the rasterized classification map, and simplify the polylines of the complex vectorized output, which is referred to as polygon generalization [3]. Among the polygon generalization methods, Radial distance [4] and Reumann-Witkam [5] are quite similar. The former removes vertices located inside tolerance circles centered at vertices of interest, and the latter computes the line passing through two consecutive vertices and removes the vertices that are closer to this line than a tolerance value. The Valingam-Whyatt approach [6] ranks all the vertices in accordance to their significance derived from their effective area, and iteratively removes the less significant vertices if their effective areas are lower than a tolerance value. The common Douglas-Peucker [7] approach computes the edge joining the first and last vertices, and finds the farthest vertex from this edge. If the distance between this vertex and the edge is larger than a threshold, the algorithm is repeated recursively with the first and farthest, then farthest and last vertices. If the distance between the farthest vertex and the edge is smaller than a threshold, all intermediate vertices are removed.

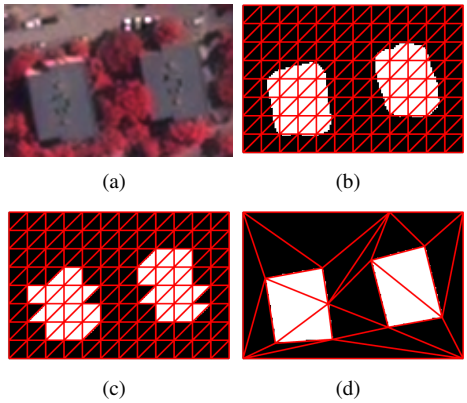

Fig. 1. Input image and example labeled meshes. (a) Input image, (b) Initial fine lattice, (c) Initial and (d) Optimized labeled triangle meshes. The triangles labeled as building are indicated by white.

In this paper, we extend a recent method [8], which uses a binary labeled triangle mesh to approximate the input classification map. In the recent approach, the objective function is designed to trade mesh complexity for fidelity to the classification map in ℓ_1 norm sense. While deep neural network approaches yield classification maps with high accuracy, a closer visual inspection reveals that such maps do not delineate the building contours perfectly, in particular near building corners that are overly rounded. Using only the classification map yields artifacts in the vectorized outputs. Based on a prior knowledge that most building edges meet at right angles, we add a novel geometric regularity term to the objective function, which favors angles of building corners being a factor of $\frac{\pi}{2}$ radians.

2. METHODOLOGY

We generate a classification map from an input image and place a fine lattice on it (see Fig.1(b)). Then, we create an initial labeled triangle mesh, where label of each triangle is set to label of the class with most samples inside the triangle (see Fig.1(c)). We iteratively modify the mesh using a set of geometric operators that minimize an objective function. The optimized mesh for the input image in Fig.1(a), is depicted in Fig.1(d). The final vectorized output is generated by eliminating the edges that are not located on building borders.

We denote by $\mathcal{L}$ a set of binary labels, where $l \in \mathcal{L}$ is 1 for building and 0 for non-building class. We denote by T the labeled triangle mesh consisting of triangles $\{t_i\}$, $A(t)$ the area of triangle t, l_t its label and V_t its vertices. Θ_i denotes the wedge of a vertex $v_i \in V_t$, defined by summing the angles between the edges origination from v_i, of the consecutive faces with label 1 (building), starting from t and incident to v_i. Fig.2(a) depicts three wedges of a triangle t.

Our goal is to minimize the following objective function:

$$E(T) = \sum_{t \in T} \Bigg[(1 + \delta(l_t)) \left(\min_{l_t \in \mathcal{L}} \iint\limits_{x,y \in t} C_{prob}(l_t, x, y)\, dx\, dy \right) + \delta(1 - l_t) \left(\sum_{v_i \in V_t} C_{reg}(\Theta_i) \right) \frac{A(t)}{3} + \lambda \Bigg], \tag{1}$$

where $C_{prob}(l_t, x, y)$ denotes the cost of assigning label l_t to the pixel located at x, y, $C_{reg}(\Theta_i)$ is the regularity cost for wedge Θ_i, and λ provides a means to trade mesh complexity for fidelity.

$P(l, x, y)$ denotes the probability, estimated by a classifier, of assigning a label $l \in \mathcal{L}$ to a pixel located at (x, y) in the image. We define the probability cost in (1) as:

$$C_{prob}(l, x, y) = \|1 - P(l, x, y)\|_1, \tag{2}$$

which may be seen as the volume contained between the classifier's probability surface and the approximation surface delineated by the labeled mesh. Fig.3 illustrates C_{prob} in 1D.

The regularity cost function $C_{reg}(\Theta)$, designed to favor that edges of building corners meet at right angles (i.e., factor of $\frac{\pi}{2}$ radians) is defined as:

$$C_{reg}\left(\Theta\right) = \min_{k=\{1,2,3\}} \left\{ scale, skewness \left(\Theta - \frac{k\pi}{2} \right)^2 \right\}, \tag{3}$$

where *scale* is the maximum cost for a wedge, and *skewness* adjusts how tolerant the system would be to the distance from the closest factor of $\frac{\pi}{2}$ for a wedge. A large value for the *skewness* parameter reduces the tolerance to the distance from the closest factor of $\frac{\pi}{2}$, and vice-versa. We set *scale* parameter to 0.5 to ensure that probability and regularity costs

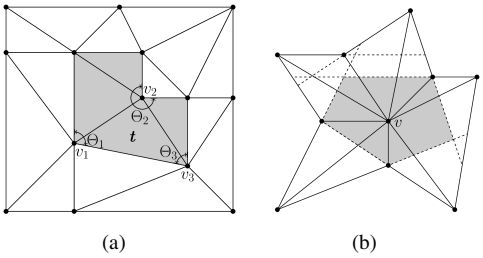

Fig. 2. Wedges of a triangle and kernel of a polygon. (a) Three wedges of triangle t. Θ_1, Θ_2 and Θ_3 correspond to wedges of vertices v_1, v_2 and v_3, respectively. Triangles with label 1 are depicted in gray. (b) Kernel of the polygon consisting of the triangles that are incident to v, is indicated by gray.

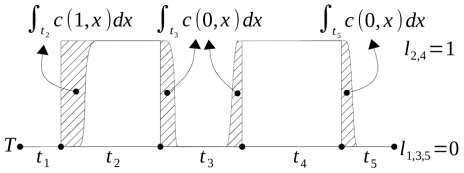

Fig. 3. C_{prob} in 1D.

range between 0 and $0.5A(t)$. The *skewness* parameter is set to 2 by default. A plot for $C_{reg}(\Theta)$ with default parameters is shown in Fig.4.

The last term λ in (1), also summed over the mesh triangles, provides a means to control the balance with the final mesh complexity (a large value for λ decreases the number of triangles, and vice-versa). $\delta(\cdot)$ denotes the Dirac delta function, with value one at zero and zero elsewhere. The regularity cost for a triangle is computed only if it has been classified as building (i.e., l_t = 1) and if at least one of its vertices is located on building borders. Otherwise, its regularity cost is ignored and the probability cost in (1) is multiplied by 2 so that the total cost for each triangle consistently and ranges between 0 and $A(t)$.

We now detail the local mesh-based operators utilized to minimize the objective function. We use two edge-based operators: *edge flip* [9], which flips edges to improve edge alignment with building borders and *halfedge collapse*, which reduces complexity of the triangle mesh (see Fig.5).

The limitation of edge-based operators is that they work on only fixed vertices that have already been placed. In order to compensate this limitation and increase expressiveness of the labeled triangle mesh we utilize another operator, which relocates the vertices in continuous space. Here, the challenge is to determine the moving direction that reduces the objective defined in (1) for a vertex. Because the objective function is not differentiable w.r.t current location of the vertex, gradient based optimization approaches are not applicable to our problem. We adapt the simplex method for function minimization

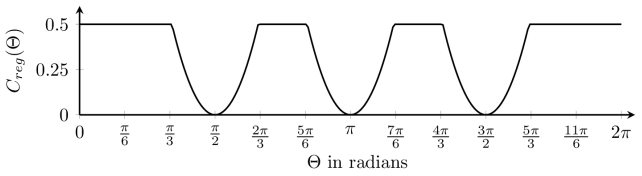

Fig. 4. $C_{reg}(\Theta)$ when *scale* is 0.5 and *skewness* is 2.

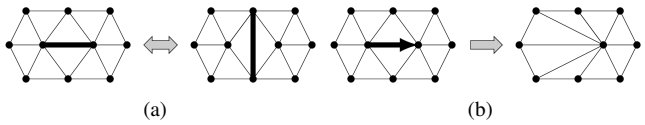

Fig. 5. Edge based operators. (a) Edge flip, (b) Halfedge collapse.

described in [10]. We denote the vertex that would be relocated by v. We generate two random vertices inside the kernel of the polygon (see Fig.2(b)) that consists of the faces incident to v. The triangle formed by v and the two random vertices is moved to find the new location for v, by using three different movement types: *reflection, expansion* and *shrink*. We define the best and worst points of the triangle as the points, to which if v is relocated, produce the lowest and highest values for objective (1). We denote these points by P_b and P_w, cost of a point P_i of the triangle by y_i, and distance between P_i and P_j by $[P_iP_j]$. The *reflection* of P_w is denoted by P^* and defined as:

$$P^* = (1+\alpha)\overline{P} - \alpha P_w, \tag{4}$$

where $\overline{P}$ is centroid of the triangle and α is the ratio of $[P^*\overline{P}]$ to $[P_w\overline{P}]$. If y^* is between y_w and y_b, P_w is replaced by P^* and the *reflection* process starts again. If $y^* < y_b$, the new minimum is found, P^* is expanded to P^{**} as:

$$P^{**} = \gamma P^* + (1-\gamma)\overline{P}, \tag{5}$$

where γ is the ratio of $[P^{**}\overline{P}]$ to $[P^*\overline{P}]$. If $y^{**} < y_b$, we replace P_w by P^{**} and restart the process. On the contrary, if $y^{**} > y_b$, that means it is a failed *expansion*, and P_w is replaced by P^* again. If $y^* > y_w$, P_w is either kept at its original location or replaced by P^* depending on which location gives the lower cost; then, the *contraction* operation is applied as:

$$P^{**} = \beta P_w + (1-\beta)\overline{P}, \tag{6}$$

where β is the *contraction* coefficient, which is the ratio of $[P^{**}\overline{P}]$ to $[P\overline{P}]$. We accept P^{**} for P_w and restart the process unless $y^{**} > min(y_w, y^*)$, which means the contracted point is worse than the better of P_w and P^*. For such a failed *contraction* movement, all the P_i's are replaced by $(P_i + P_b)/2$ and the process is started over.

Once the moving triangle includes the new optimum location for v, it keeps shrinking, i.e. its area is getting smaller and smaller in each iteration. As soon as the triangle area or total number of iterations reaches the predefined threshold values, the process is stopped and v is relocated to P_b. Note that v is relocated only inside the polygon kernel in order to keep the triangulation valid. During the relocation process, we relocate only the vertices that are positioned at building borders.

Starting from the initial lattice, we iteratively optimize the labeled triangle mesh. We simulate each change, caused by the operators, that transforms the mesh T to T' and calculate the objective difference $\Delta E = E(T') - E(T)$ incurred by the change. We push all the operators to a priority queue, where they are sorted according to their ΔE value in ascending order. The first element in the queue is popped first and applied if its associated ΔE is negative. In each iteration, we relabel all the affected triangles, recalculate their costs, and update the priority queue accordingly. The iterations continue until there is no operator left in the queue.

Although the labeled mesh is optimized by the operators according to (1), in some cases results may not be visually appealing because of the topology change. We observe the topology change clearly when neighboring buildings are very close to each other. In such a case, multiple objects are merged into one. To describe the topology, we use Euler characteristic $\chi = V - E + F$, in which V, E and F are number of vertices, edges and faces that are adjacent to triangles, labeled as building in the mesh. In order to preserve the topology, we compute difference $\Delta V, \Delta E$, and ΔF between in T and T', and transform T to T' only if $\chi = \Delta V - \Delta E + \Delta F = 0$.

3. EXPERIMENTS

We use a 1180×1030 Pléiades image that has been captured over Santiago and has near infrared, red, and green bands. A ground-truth, in which each pixel is labeled as either building or non-building has been manually prepared. We generate the classification map using the network presented in [1].

We first create a fine lattice by generating a vertex at every 10 pixels. We then apply mesh operators in batches, where each batch fills the queue with one type of operator and applies all the operators in the queue to the mesh. We modify the initial labeled triangle mesh by proceeding with this sequence of batches: *edge flip*, *vertex relocation*, *halfedge collapse*, and *vertex relocation*.

We set the parameters for *reflection, expansion* and *contraction* movements in vertex relocation operator to the following values; $\alpha = 1, \gamma = 2$, and $\beta = 0.5$. We also set $\#$ of iterations and area thresholds for the simplex optimization method to 100 and 0.1 respectively. In addition, we ignore relocation movements when magnitude of displacement is below 0.01 pixels.

We compare our approach with a mesh approximation algorithm presented in [8] and the commonly used polygon generalization algorithms in GIS applications, namely Radial distance [4], Reumann-Witkam [5], Valingam-Whyatt [6],

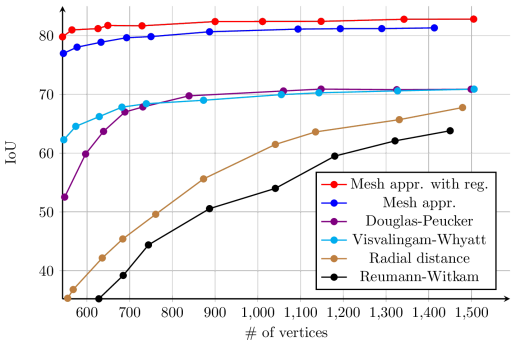

Fig. 6. IoU vs. # of vertices plot

and Douglas-Peucker [7]. For the generalization algorithms, we vectorize the classification map by using *gdal_polygonize* function of the GDAL library, and simplify the complex vectorized output by the generalization algorithms. We use intersection over union (IoU) between the vectorized classifications and ground-truth for the building class as the performance measurement. We compare the results for different # of vertices by changing the value for λ parameter in (1) and in the objective function in [8], and the value for tolerance threshold in generalization methods. IoU vs. # of vertices plot is shown in Fig. 6. The plot shows that our approach and the mesh approximation [8] outperform the generalization algorithms significantly. We also observe that mesh approximation with right angle regularity yields better results than [8]. Example building contours generated by our method and [8] are illustrated in Fig.7, which proves that even if the classification map is blobbly, building borders with right angles can be obtained with the help of regularity term in (1).

4. CONCLUDING REMARKS

In this paper, we extended a recent polygonization method [8] by adding a novel regularity term to the objective function, as we know that most building edges meet at right angles. Qualitative results confirmed that with the help of such a regularity term, the mesh delineates building contours better. We also showed that our method performs better quantitatively than the recent approach [8] as well as commonly used polygon generalization algorithms in remote sensing literature.

For the future work, we plan to add richer geometric regularities such as parallelism, symmetries and orbits. We also plan to measure fidelity against the input image itself, in addition to the classification map. We also wish to extend our approach to multiple classes.

5. REFERENCES

[1] E. Maggiori, Y. Tarabalka, G. Charpiat, and P. Alliez, "High-resolution semantic labeling with convolutional neural networks," *IEEE TGRS*, vol. 55/12, 2017.

[2] M. Volpi and D. Tuia, "Dense semantic labeling of sub-decimeter resolution images with convolutional neural networks," *IEEE TGRS*, vol. 55/2, 2017.

[3] M. Galanda, "Automated polygon generalization in a multi agent system," *PhD. Dissertation, University of Zurich, Switzerland*, 2003.

[4] W. Shi and C. Cheung, "Performance evaluation of line simplification algorithms for vector generalization," *The Cartographic Journal*, vol. 43, no. 1, pp. 27–44, 2006.

[5] K. Reumann and A. P. M. Witkam, "Optimizing curve segmentation in computer graphics," in *Proceedings of the International Computing Symposium*, 1974.

[6] M. Visvalingam and J. D. Whyatt, "Line generalisation by repeated elimination of the smallest area," *Cartographic Journal*, vol. 30, no. 1, pp. 46–51, 1992.

[7] D. H. Douglas and T. K. Peucker, "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature," *Cartographica: The International Journal for Geographic Information and Geovisualization*, vol. 10, no. 2, pp. 112–122, 1973.

[8] E. Maggiori, Y. Tarabalka, G. Charpiat, and P. Alliez, "Polygonization of remote sensing classification maps by mesh approximation," in *IEEE ICIP*, 2017, p. 5.

[9] M. Botsch, L. Kobbelt, M. Pauly, P. Alliez, and B. Lévy, *Polygon mesh processing*, CRC press, 2010.

[10] J. A. Nelder and R. Mead, "A simplex method for function minimization," *The computer journal*, 1965.

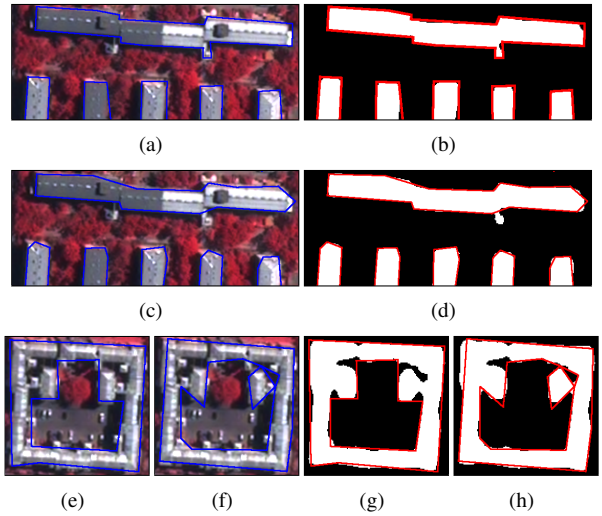

Fig. 7. Visual comparison of mesh appr. with and without regularity when # of vertices are similar. Input image and classification map are depicted in the left and right column respectively. (a, b, e, g): with, (c, d, f, h) without regularity.