Polygonal Building Extraction by Frame Field Learning

Nicolas Girard[1] Dmitriy Smirnov[2] Justin Solomon[2] Yuliya Tarabalka[3]
[1]Université Côte d'Azur, Inria [2]Massachusetts Institute of Technology [3]LuxCarta Technology

Abstract

While state of the art image segmentation models typically output segmentations in raster format, applications in geographic information systems often require vector polygons. To help bridge the gap between deep network output and the format used in downstream tasks, we add a frame field output to a deep segmentation model for extracting buildings from remote sensing images. We train a deep neural network that aligns a predicted frame field to ground truth contours. This additional objective improves segmentation quality by leveraging multi-task learning and provides structural information that later facilitates polygonization; we also introduce a polygonization algorithm that that utilizes the frame field along with the raster segmentation. Our code is available at https://github.com/Lydorn/Polygonization-by-Frame-Field-Learning.

1. Introduction

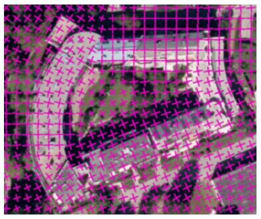


Figure 1: A frame field output by our network.

Due to their success in processing large collections of noisy images, deep convolutional neural networks (CNNs) have achieved state-of-the-art in remote sensing segmentation. Geographic information systems like Open Street Map (OSM) [29], however, require segmentation data in *vector* format (e.g., polygons and curves) rather than raster format, which is generated by segmentation networks. Additionally, methods that extract objects from remote sensing images require especially high throughput to handle the volume of high-resolution aerial images captured daily over large territories of land. Thus, modifications to the conventional CNN pipeline are necessary.

Existing work on deep building segmentation generally falls into one of two general categories. The first vectorizes the probability map produced by a network a posteriori, e.g., by using contour detection (marching squares [25]) followed by polygon simplification (Ramer–Douglas–Peucker [30, 13]). Such approaches suffer when the classification maps contain imperfections such as smoothed out corners, a common artifact of conventional deep segmentation methods. Moreover, as we show in Fig. 2, even perfect probability maps are challenging to polygonize due to shape information being lost from the discretization of the raster output. To improve the final polygons, these methods employ expensive and complex post-processing procedures. ASIP polygonization [20] uses polygonal partition refinement to approximate shapes from the output probability map based on a tunable parameter controlling the trade-off between complexity and fidelity. In [42], a decoder and a discriminator regularize output probability maps adversarially. This requires computing large matrices of pairwise discontinuity costs between pixels and involves adversarial training, which is less stable than conventional supervised learning.

Another category of deep segmentation methods learns a vector representation directly. For example, Curve-GCN [23] trains a graph convolutional network (GCN) to deform polygons iteratively, and PolyMapper [21] uses a recurrent neural network (RNN) to predict vertices one at a time. While these approaches directly predict polygon parameters, GCNs and RNNs suffer from several disadvantages. Not only are they more difficult to train than CNNs, but also their output topology is restricted to simple polygons without holes—a serious limitation in segmenting complex buildings. Additionally, adjoining buildings with common walls are common, especially in city centers. Curve-GCN and PolyMapper are unable to reuse the same polyline in adjoining buildings, yielding overlaps and gaps.

We introduce a building extraction algorithm that avoids the challenges above by adding a frame field output to a fully-convolutional network (see Fig. 1). While this has imperceptible effect on training or inference time, the frame field not only increases segmentation performance, e.g., yielding sharper corners, but also provides useful information for vectorization. Additional losses learn a valid frame field that is consistent with the segmentation. These losses regularize the segmentation, similar to [37], which includes

MRF/CRF regularization terms in the loss function to avoid extra MRF/CRF inference steps.

The frame field allows us to devise a straightforward polygonization method extending the Active Contours Model (ACM, or "snakes") [19], which we call the Active Skeleton Model (ASM). Rather than fitting contours to image data, ASM fits a skeleton graph, where each edge connects two junction nodes with a chain of vertices (i.e., a polyline). This allows us to reuse shared walls between adjoining buildings. To our knowledge, no existing method handles this case ([38] shows results with common walls but does not provide details). Our method naturally handles large buildings and buildings with inner holes, unlike end-to-end learning methods like PolyMapper [21]. Lastly, our polygon extraction pipeline is highly GPU-parallelizable, making it faster than more complex methods.

Our main contributions are:

(i) a learned frame field aligned to object tangents, which improves segmentation via multi-task learning;

(ii) coupling losses between outputs for self-consistency, further leveraging multi-task learning; and

(iii) a fast polygonization method leveraging the frame field, allowing complexity tuning of a corner-aware simplification step and handling non-trivial topology.

2. Related work

ASIP polygonization [20] inputs an RGB image and a probability map of objects (e.g., buildings) detected in the image (e.g., by a neural network). Then, starting from a polygonal partition that oversegments the image into convex cells, the algorithm refines the partition while labeling its cells by semantic class. The refinement process is an optimization with terms that balance fidelity to the input against complexity of the output polygons. The configuration space is explored by splitting and merging the polygonal cells. As the fidelity and complexity terms can be balanced with a coefficient, the fidelity-to-complexity ratio can be tuned. However, there does not exist a systematic approach for interpreting or determining this coefficient. While ASIP post-processes the output of a deep learning method, recent approaches aim for an end-to-end pipeline.

CNNs are successful at converting grid-based input to grid-based output for tasks where each output pixel depends on its local neighborhood in the input. In this setting, it is straightforward and efficient to train a network for supervised prediction of segmentation probability maps. The paragraphs below, however, detail major challenges when using such an approach to extract polygonal buildings.

First, the model needs to produce variable-sized outputs to capture varying numbers of objects, contours, and vertices. This requires complex architectures like recurrent neural networks (RNNs) [18], which are not as efficiently trained as CNNs and need multiple iterations at inference time. Such is the case for PolyMapper [21], Polygon-RNN [5], and Polygon-RNN++ [1]. Curve-GCN [23] predicts a fixed number of vertices simultaneously.

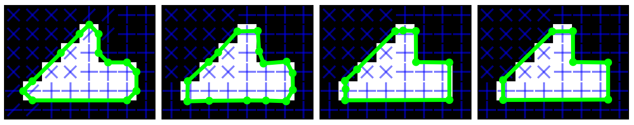
A second challenge is that the model must make discrete decisions of whether to add a contour, whether to add a hole to an object, and with how many vertices to describe a contour. Adding a contour is solved by object detection: a contour is predicted for each detected object. Adding holes to an object is more challenging, but a few methods detect holes and predict their contours. One model, BSP-Net [8], circumvents this issue by combining predicted convex shapes for the final output, producing shapes in a compact format, with potential holes inside. To our knowledge, the number of vertices is not a variable that current deep learning models can optimize for; discrete decisions are difficult to pose differentiably without training techniques such as the straight-through estimator [3] or reinforcement learning [35, 28, 27].

A third challenge is that, unlike probability maps, the output structure of polygonal building extraction is not grid-like. Within the network, the grid-like structure of the image input has to be transformed to a more general planar graph structure representing building outlines. City centers have the additional problem of adjoining buildings that share a wall. Ideally, the output geometry for such a case would be a collection of polygons, one for each individual building, which share polylines corresponding to common walls. Currently, no existing deep learning method tackles this case. Our method solves it but is not end-to-end. PolyMapper [21] tackles the individual building and road network extraction tasks. As road networks are graphs, they propose a novel sequentialization method to reformulate graph structures as closed polygons. Their approach might work in the case of adjoining buildings with common walls. Their output structure, however, is less adapted to GPU computation, making it less efficient. RNNs such as PolyMapper [21], Polygon-RNN [5], and Polygon-RNN++ [1] perform beam search at inference to prune off improbable sequences, which requires more vertex predictions than are used in the final output and is inefficient. The DefGrid [14] module is a non-RNN approach where the network processes polygonal superpixels. It is more complex than our simple fully-convolutional network and is still subject to the rounded corner problem.

The challenges above demand a middle ground between learning a bitmap segmentation followed by a hand-crafted polygonization method and end-to-end methods, aiming to be easily-deployable, topologically flexible w.r.t. holes and common walls, and efficient. A step in this direction is the machine-learned building polygonization [41] that predicts building segmentations using a CNN, uses a generative adversarial network to regularize building boundaries, and learns a building corner probability map, from which vertices are extracted. In contrast, our model predicts a frame field both as additional geometric information (instead of a building corner probability map) and as a way to regularize building boundaries (instead of adversarial training). The addition of this frame field output is similar in spirit to DiResNet [12], a road extraction neural network that outputs road direction in addition to road segmentation, first introduced in [2]. The orientation is learned for each road pixel by a cross-entropy classification loss whose labels are orientation bins. This additional geometric feature learned by the network improves the overall geometric integrity of the extracted objects (in their case road connectivity). The differences to our method include the following: (1) our frame fields encode two orientations instead of one (needed for corners), (2) we use a regression loss instead of a classification loss, and (3) we use coupling losses to promote coherence between segmentation and frame field.

3. Method

Our key idea is to help the polygonization method solve ambiguous cases caused by discrete probability maps by asking the neural network to output missing shape information in the form of a frame field (see Fig. 2). This practically does not increase training and inference time, allows for simpler and faster polygonization, and regularizes the segmentation—solving the problem of small misalignments of the ground truth annotations that yield rounded corners if no regularization is used.



(a) Iter. 0 (b) Iter. 50 (c) Iter. 250 (d) Result

Figure 2: Even a perfect classification map can yield incorrect polygonization due to a locally ambiguous probability map, as shown in (a), the output of marching squares. Our polygonization method iteratively optimizes the contour (b-d) to align to a frame field, yielding better results as our frame field (blue) disambiguates between slanted walls and corners, preventing corners from being cut off.

3.1. Frame fields

We provide the necessary background on frame fields, a key part of our method. Following [39, 11], a frame field is a *4-PolyVector field*, which assigns four vectors to each point of the plane. In the case of a frame field, however, the first two vectors are constrained to be opposite to the other two, i.e., each point is assigned a set of vectors $\{u, -u, v, -v\}$. At each point in the image, we consider the two directions that define the frame as two complex numbers $u, v \in \mathbb{C}$. We need two directions (rather than only one) because buildings, unlike organic shapes, are regular structures with sharp corners, and capturing directionality at these sharp corners requires two directions. To encode the directions in a way that is agnostic to relabeling and sign change, we represent them as coefficients of the following polynomial:

$$f(z) = (z^2 - u^2)(z^2 - v^2) = z^4 + c_2 z^2 + c_0 \,. \quad (1)$$

We denote (1) above by $f(z; c_0, c_2)$. Given a (c_0, c_2) pair, we can easily recover one pair of directions defining the corresponding frame:

$$\begin{cases} c_0 = u^2 v^2 \\ c_2 = -(u^2 + v^2) \end{cases} \iff \begin{cases} u^2 = -\frac{1}{2}\left(c_2 + \sqrt{c_2^2 - 4c_0}\right) \\ v^2 = -\frac{1}{2}\left(c_2 - \sqrt{c_2^2 - 4c_0}\right) \,. \end{cases} \quad (2)$$

In our approach, inspired by [4], we learn a smooth frame field with the property that, along building edges, at least one field direction is aligned to the polygon tangent direction. At polygon corners, the field aligns to *both* tangent directions, motivating our use of PolyVector fields rather than vector fields. Away from polygon boundaries, the frame field does not have any alignment constraints but is encouraged to be smooth and not collapse to a line field. Like [4], we formulate the field computation variationally, but, unlike their approach, we use a neural network to learn the field at each pixel, which is also explored in [36]. To avoid sign and ordering ambiguity, we learn a (c_0, c_2) pair per pixel rather than (u, v).

3.2. Frame field learning

We describe our method, illustrated in Fig. 3. Our network takes a $3{\times}H{\times}W$ image I as input and outputs a pixel-wise classification map and a frame field. The classification map contains two channels, $\widehat{y}_{int}$ corresponding to building interiors and $\widehat{y}_{edge}$ to building boundaries. The frame field contains four channels corresponding to the two complex coefficients $\widehat{c}_0, \widehat{c}_2 \in \mathbb{C}$, as in §3.1 above.

Segmentation losses. Our method can be used with any deep segmentation model as a backbone; in our experiments, we use the U-Net [31] and DeepLabV3 [7] architectures. The backbone outputs an F-dimensional feature map $\widehat{y}_{backbone} \in \mathbb{R}^{F \times H \times W}$. For the segmentation task, we append to the backbone a fully-convolutional block (taking $\widehat{y}_{backbone}$ as input) consisting of a 3×3 convolutional layer, a batch normalization layer, an ELU nonlinearity, another $3{\times}3$ convolution, and a sigmoid nonlinearity. This segmentation head outputs a segmentation map $\widehat{y}_{seg} \in \mathbb{R}^{2 \times H \times W}$. The first channel contains the object interior segmentation map $\widehat{y}_{int}$ and the second contains the contour segmentation map $\widehat{y}_{edge}$. Our training is supervised—each input image

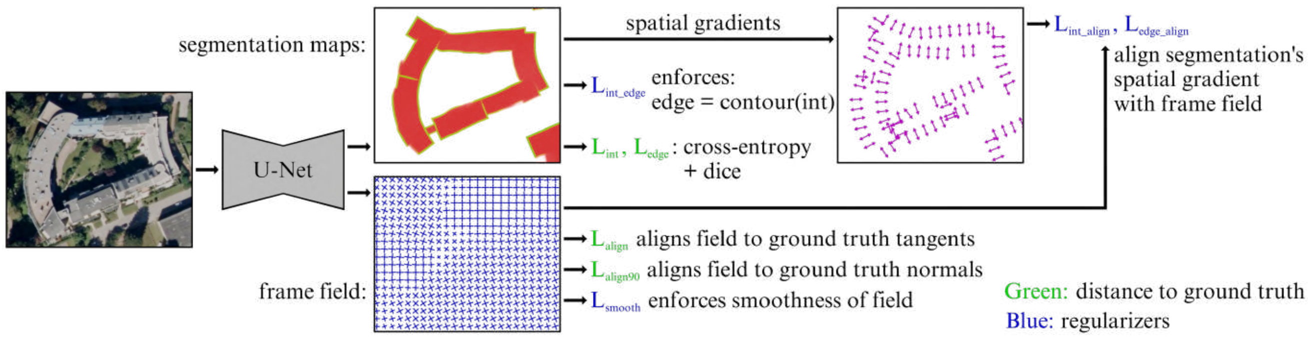

segmentation maps:
spatial gradients
L_{int_align}, L_{edge_align}
align segmentation's spatial gradient with frame field
L_{int_edge} enforces: edge = contour(int)
U-Net
L_{int}, L_{edge}: cross-entropy + dice
L_{align} aligns field to ground truth tangents
$L_{align90}$ aligns field to ground truth normals
L_{smooth} enforces smoothness of field
frame field:
Green: distance to ground truth
Blue: regularizers

Figure 3: Given an overhead image, our model outputs an edge mask, interior mask, and frame field. The loss aligns the masks and field to ground truth data, enforces smoothness of the frame field, and ensures consistency between the outputs.

is labeled with ground truth y_{int} and y_{edge}, corresponding to rasterized polygon interiors and edges, respectively. We then use a linear combination of the cross-entropy loss and Dice loss [34] for loss L_{int} applied on the interior output as well as loss L_{edge} applied on the contour (edge) output.

Frame field losses. In addition to the segmentation masks, our network outputs a frame field. We append another head to the backbone via a fully-convolutional block consisting of a 3×3 convolutional layer, a batch normalization layer, an ELU nonlinearity, another 3×3 convolution, and a tanh nonlinearity. This frame field block inputs the concatenation of the output features of the backbone and the segmentation output: $[\widehat{y}_{backbone}, \widehat{y}_{seg}] \in \mathbb{R}^{(F+2)\times H\times W}$. It outputs the frame field with $\widehat{c}_0, \widehat{c}_2 \in \mathbb{C}^{H\times W}$. The corresponding ground truth label is an angle $\theta_\tau \in [0,\pi)$ of the unsigned tangent vector of the polygon contour. We use three losses to train the frame field:

$$L_{align} = \frac{1}{HW}\sum_{x\in I} y_{edge}(x)|f(e^{i\theta_\tau}; \widehat{c}_0(x), \widehat{c}_2(x))|^2, \quad (3)$$

$$L_{align90} = \frac{1}{HW}\sum_{x\in I} y_{edge}(x)|f(e^{i\theta_{\tau^\perp}}; \widehat{c}_0(x), \widehat{c}_2(x))|^2, \quad (4)$$

$$L_{smooth} = \frac{1}{HW}\sum_{x\in I} \left(\|\nabla\widehat{c}_0(x)\|^2 + \|\nabla\widehat{c}_2(x)\|^2\right), \quad (5)$$

where θ_w is the direction of w ($w = \|w\|_2 e^{i\theta_w}$), and $\tau^\perp = \tau - \frac{\pi}{2}$. Each loss measures a different property of the field:

- L_{align} enforces alignment of the frame field to the tangent directions. This term is small when the polynomial $f(\cdot; \widehat{c}_0, \widehat{c}_2)$ has a root near $e^{i\theta_\tau}$, implicitly implying that one of the field directions $\{\pm u, \pm v\}$ is aligned with the tangent direction τ. Since (1) has no odd-degree terms, this term has no dependence on the sign of τ, as desired.
- $L_{align90}$ prevents the frame field from collapsing to a line field by encouraging it to also align with $\tau^\perp$.
- L_{smooth} is a Dirichlet energy measuring the smoothness of $\widehat{c}_0(x)$ and $\widehat{c}_2(x)$ as functions of location x in the image. Smoothly-varying $\widehat{c}_0$ and $\widehat{c}_2$ yield a smooth frame field.

Output coupling losses. We add coupling losses to ensure mutual consistency between our network outputs:

$$L_{int\ align} = \frac{1}{HW}\sum_{x\in I} f(\nabla\widehat{y}_{int}(x); \widehat{c}_0(x), \widehat{c}_2(x))^2, \quad (6)$$

$$L_{edge\ align} = \frac{1}{HW}\sum_{x\in I} f(\nabla\widehat{y}_{edge}(x); \widehat{c}_0(x), \widehat{c}_2(x))^2, \quad (7)$$

$$L_{int\ edge} = \frac{1}{HW}\sum_{x\in I} \max\left(1 - \widehat{y}_{int}(x), \|\nabla\widehat{y}_{int}(x)\|_2\right) \cdot \left|\|\nabla\widehat{y}_{int}(x)\|_2 - \widehat{y}_{edge}(x)\right|. \quad (8)$$

- $L_{int\ align}$ aligns the spatial gradient of the predicted interior map $\widehat{y}_{int}$ with the frame field (analogous to (3)).
- $L_{edge\ align}$ aligns the spatial gradient of the predicted edge map $\widehat{y}_{edge}$ with the frame field (analogous to (3)).
- $L_{int\ edge}$ makes the predicted edge map be equal to the norm of the spatial gradient of the predicted interior map. This loss is applied outside of buildings (hence the $1 - \widehat{y}_{int}(x)$ term) and along building contours (hence the $\|\nabla\widehat{y}_{int}(x)\|_2$ term) and is not applied inside buildings, so that common walls between adjoining buildings can still be detected by the edge map.

Final loss. Because the losses (L_{int}, L_{edge}, L_{align}, $L_{align90}$, L_{smooth}, $L_{int\ align}$, $L_{edge\ align}$, and $L_{int\ edge}$) have distinct units, we compute a normalization coefficient for each loss by averaging its value over a random subset of the training dataset using a randomly-initialized network. Losses are then normalized by this coefficient before being linearly combined. This normalization aims to rescale losses such that they are easier to balance. More details are in the supplementary materials.

3.3. Frame field polygonization

The main steps of our polygonization method are shown in Fig. 4. It is inspired by the Active Contour Model (ACM) [19]. ACM is initialized with a given contour and

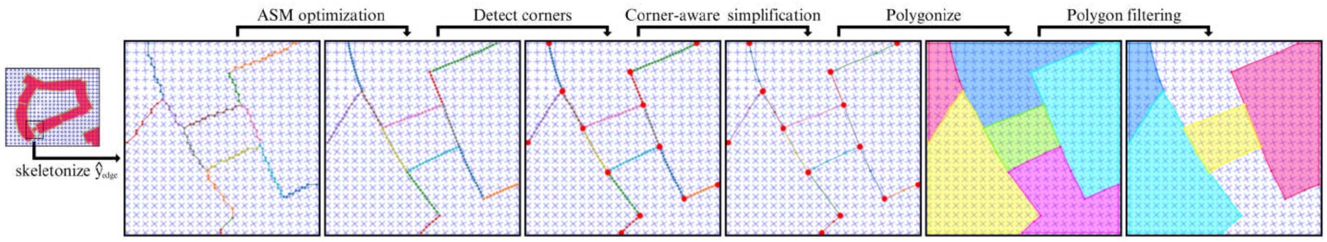


Figure 4: Overview of our post-processing polygonization algorithm. Given an interior classification map and frame field (Fig. 3) as input, we optimize the contour to align to the frame field using an Active Skeleton Model (ASM) and detect corners using the frame field, simplifying non-corner vertices.

minimizes an energy function $E^*_{contour}$, which moves the contour points toward an optimal position. Usually this energy is composed of a term to fit the contour to the image and additional terms to limit the amount of stretch and/or curvature. The optimization is performed by gradient descent. Overall the ACM lends itself perfectly for parallelized execution on the GPU, and the optimization can be performed using an automatic differentiation module included in deep learning frameworks. We adapt ACM so that the optimization is performed on a skeleton graph instead of contours, giving us the Active Skeleton Model (ASM). We call the skeleton graph the graph of connected pixels of the skeleton image obtained by the thinning method [40] applied on the building wall probability map y_{edge}. The following energy terms are used:

- $E_{probability}$ fits the skeleton paths to the contour of the building interior probability map $y_{int}(v)$ at a certain probability threshold ℓ (set to 0.5 in practice).
- $E_{frame\ field\ align}$ aligns each edge of the skeleton graph to the frame field.
- E_{length} ensures that the node distribution along paths remains homogeneous as well as tight.

Details about our data structure (designed for GPU computation), definition and computation of our energy terms, and explanation of our corner-aware simplification step can be found in the supplementary materials.

4. Experimental setup

4.1. Datasets

Our method requires ground truth polygonal building annotations (rather than raster binary masks) so that the ground truth angle for the frame field can be computed by rasterizing separately each polygon edge and taking the edge's angle. Thus, for each pixel we get a θ_τ value, which is used in L_{align}.

We perform experiments on these datasets (more details in the supplementary material):

- CrowdAI Mapping Challenge dataset [32] (*CrowdAI dataset*): 341438 aerial images of size 300×300 pixels with associated ground truth polygonal annotations.
- Inria Aerial Image Labeling dataset [26] (*Inria dataset*): 360 aerial images of size 5000×5000 pixels. Ten cities are represented, making it more varied than the *CrowdAI dataset*. However, the ground truth is in the form of raster binary masks. We thus create the *Inria OSM dataset* by taking OSM polygon annotations and correcting their misalignment using [15]. We also create the *Inria Polygonized dataset* by converting the original ground truth binary masks to polygon annotations with our polygonization method (see supplementary materials).
- *Private dataset*: 57 satellite images for training with sizes varying from 2000×2000 pixels to 20000×20000 pixels, captured over 30 different cities from all continents with three different types of satellites. This is our most varied and challenging dataset. However, the building outline polygons were manually labeled precisely by an expert, ensuring the best possible ground truth. Results for this private dataset are in the supplementary material.

4.2. Backbones

The first backbone we use is *U-Net16*, a small U-Net [31] with 16 starting hidden features (instead of 64 in the original). We also use *DeepLab101*, a DeepLabV3 [7] model that utilizes a ResNet-101 [17] encoder. Our best performing model is *UResNet101*—a U-Net with a ResNet-101 [17] encoder (pre-trained on ImageNet [10]). We observed that the pre-trained ResNet-101 encoder achieves better final performance than random initialization. For the UResNet101, we additionally use distance weighting for the cross-entropy loss, as done for the original U-Net [31].

4.3. Ablation study and additional experiments

We perform an ablation study to validate various components of our method (results in Tables 1, 2, and 4):

- "No field" removes the frame field output for comparison to pure segmentation. Only interior segmentation L_{int}, edge segmentation L_{edge} and interior/edge coupling $L_{int\ edge}$ losses remain.
- "Simple poly." uses a baseline polygonization algo-

rithm (marching-squares contour detection followed by the Ramer–Douglas–Peucker simplification) on the interior classification map learned by our full method. This allows us to study the improvement of our polygonization method from leveraging the frame field.

Additional experiments in the supplementary material include: "no coupling losses" removes all coupling losses ($L_{int\ align}$, $L_{edge\ align}$, $L_{int\ edge}$) to determine whether enforcing consistency between outputs has an impact; "no $L_{align90}$," "no $L_{int\ edge}$," "no $L_{int\ align}$ and $L_{edge\ align}$," and "no L_{smooth}" all remove the specified losses; "complexity vs. fidelity" varies the simplification tolerance parameter ε to demonstrate the trade-off between complexity and fidelity of our corner-aware simplification procedure.

4.4. Metrics

The standard metric for image segmentation is Intersection over Union (IoU), which is then used to compute other metrics such as MS COCO [22], Average Precision (AP), and Average Recall (AR)—along with variants AP_{50}, AP_{75}, AR_{50}, AR_{75}. Since we aim to produce clean geometry, it is important to measure contour regularity, not captured by the area-based metrics IoU, AP, and AR. Moreover, as annotations are bound to have some alignment noise, only optimizing IoU will favor blurry segmentations with rounded corners over sharp segmentations, as the blurry ones correspond to the shape expectation of the noisy ground truth annotation; segmentation results with sharp corners may even yield a lower IoU than segmentations with rounded corners. We thus introduce the *max tangent angle error* metric that compares the tangent angles between predicted polygons and ground truth annotations, penalizing contours not aligned with the ground truth. It is computed by uniformly sampling points along a predicted contour, computing the angle of the tangent for each point, and comparing it to the tangent angle of the closest point on the ground truth contour. The *max tangent angle error* is the maximum tangent angle error over all sampled points. More details about the computation of these metrics can be found in the supplementary material.

5. Results and discussion

5.1. CrowdAI dataset

We visualize our polygon extraction results for the *CrowdAI dataset* and compare them to other methods in Fig. 5. The ASIP polygonization method [20] inputs the probability maps of a U-Net variant [9] that won the CrowdAI challenge. All methods perform well on common building types, e.g., houses and residential buildings, but we can see that results of ASIP are less regular than PolyMapper and ours. For more complex building shapes (e.g., not rectangular or with a hole inside), ASIP outputs reasonable results, albeit still not very regular. However, the PolyMapper approach of object detection followed by polygonal outline regression does not work in the most difficult cases. It does not support nontrivial topology by construction, but also, it struggles with large complex buildings. We hypothesize that PolyMapper suffers from the fact that there are not many complex buildings and does not generalize as well as fully-convolutional networks.

Method	Mean *max tangent angle errors* ↓
UResNet101 (no field), simple poly.	51.9°
UResNet101 (with field), simple poly.	45.1°
U-Net variant [9], ASIP poly. [20]	44.0°
UResNet101 (with field), ASIP poly. [20]	38.3°
U-Net variant [9], UResNet101 **our** poly.	36.6°
PolyMapper [21]	33.1°
UResNet101 (with field), **our** poly.	**31.9°**

Table 1: Mean *max tangent angle errors* over all the original validation polygons of the *CrowdAI dataset* [32].

We report results on the original validation set of the *CrowdAI dataset* for the *max tangent angle error* in Table 1 and MS COCO metrics in Table 2. "(with field)" refers to models trained with our full frame field learning method, "(no field)" refers to models trained without any frame field output, "mask" refers to the output raster segmentation mask of the network, "our poly." refers to our frame field polygonization method, and "simple poly." refers to the baseline polygonization of marching squares followed by Ramer-Douglas-Peucker simplification. We also applied our polygonization method to the same probability maps used by the ASIP polygonization method (U-Net variant [9]) for fair comparison of polygonization methods.

In Table 1, "simple poly." performs better using "(with field)" segmentation compared to "(no field)" because of a regularization effect from frame field learning. PolyMapper performs significantly better than "simple poly." even though it is not explicitly regularized. Our frame field learning and polygonization method is necessary to decrease the error further and compare favorably to PolyMapper.

In Table 2, our UResNet101 (with field) outperforms most previous works, except "U-Net variant [9], ASIP poly. [20]" due to the U-Net variant being the winning entry to the challenge. However our polygonization applied after that same U-Net variant achieves better max tangent angle error and AP than ASIP but worse AR. The same is true when applying ASIP to our UResNet101 (with field): it has slightly worse AP, AR, and max tangent angle error. However, the ASIP method also results in better max tangent angle error when using our UResNet101 (with field) compared to using the U-Net variant.

Runtimes. We compare runtimes in Table 3. ASIP does not have a GPU implementation. In their paper they give

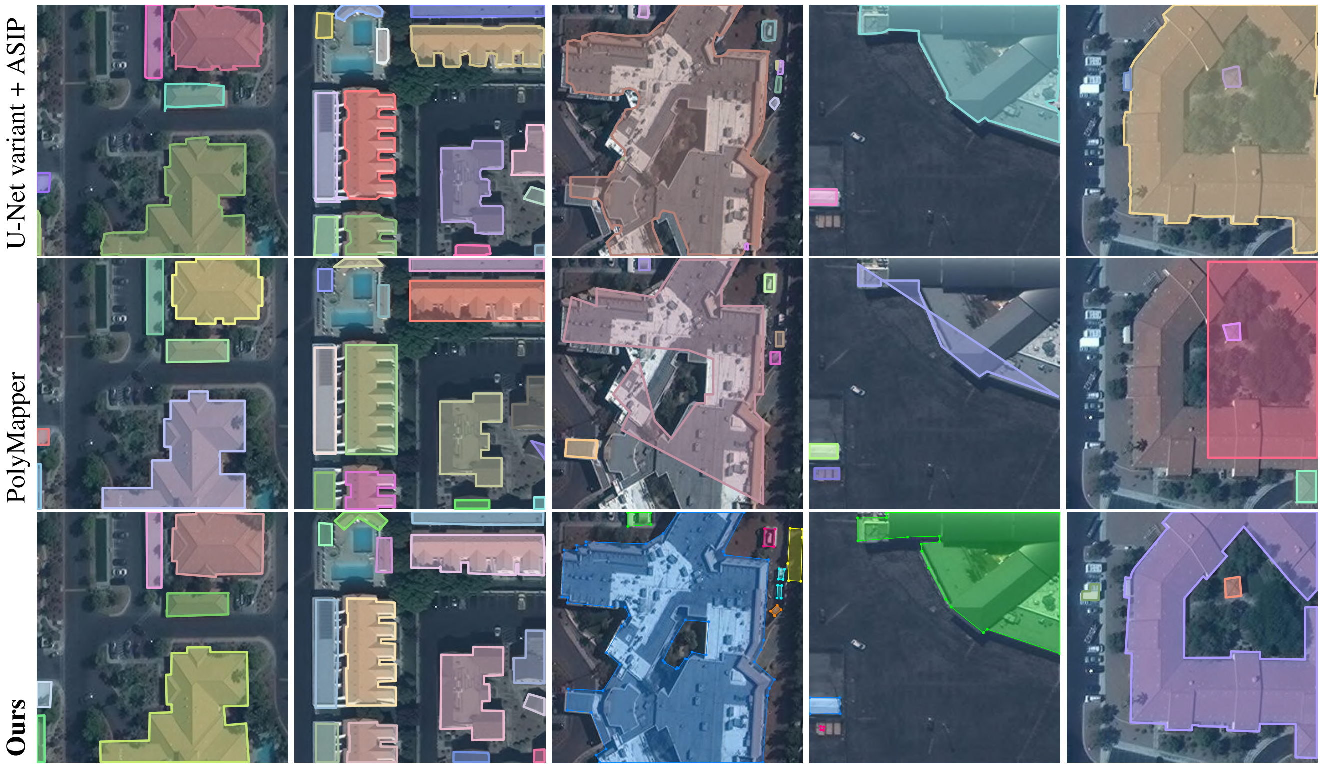


Figure 5: Example building extraction results on CrowdAI test images. Buildings become more complex from left to right. (top) U-Net variant [9] + ASIP [20], (middle) PolyMapper [21], and (bottom) **ours**: UResNet101 (full), frame field polygonization.

Method	$AP \uparrow$	$AP_{50} \uparrow$	$AP_{75} \uparrow$	$AR \uparrow$	$AR_{50} \uparrow$	$AR_{75} \uparrow$
UResNet101 (no field), mask	62.4	86.7	72.7	67.5	90.5	77.4
UResNet101 (no field), simple poly.	61.1	87.4	71.2	64.7	89.4	74.1
UResNet101 (with field), mask	64.5	89.3	74.6	68.1	91.0	77.7
UResNet101 (with field), simple poly.	61.7	87.7	71.5	65.4	89.9	74.6
UResNet101 (with field), **our** poly.	61.3	87.5	70.6	65.0	89.4	73.9
UResNet101 (with field), ASIP poly. [20]	60.0	86.3	69.9	64.0	88.8	73.4
U-Net variant [9], UResNet101 **our** poly.	**67.0**	**92.1**	**75.6**	73.2	93.5	81.1
Mask R-CNN [16] [33]	41.9	67.5	48.8	47.6	70.8	55.5
PANet [24]	50.7	73.9	62.6	54.4	74.5	65.2
PolyMapper [21]	55.7	86.0	65.1	62.1	88.6	71.4
U-Net variant [9], ASIP poly. [20]	65.8	87.6	73.4	**78.7**	**94.3**	**86.1**

Table 2: AP and AR results on the *CrowdAI dataset* [32] for all polygonization experiments.

Method	Time (sec) $\downarrow$	Hardware
PolyMapper [21]	0.38	GTX 1080Ti
ASIP [20]	0.15	Laptop CPU
Ours	**0.04**	GTX 1080Ti

Table 3: Average times to extract buildings from a 300×300 pixel patch. **Ours** refers to UResNet101 (with field), our poly. ASIP's time does not include model inference.

an average runtime of 1-3s on CPU with ~10% CPU utilization. Assuming perfect parallelization, they estimate their average runtime to be 0.15s with 100% CPU utilization. Their method uses a priority queue for optimizing the polygonal partitioning with various geometric operators and is harder to implement on GPU. Our efficient data structure makes our building extraction competitive with prior work.

5.2. Inria OSM dataset

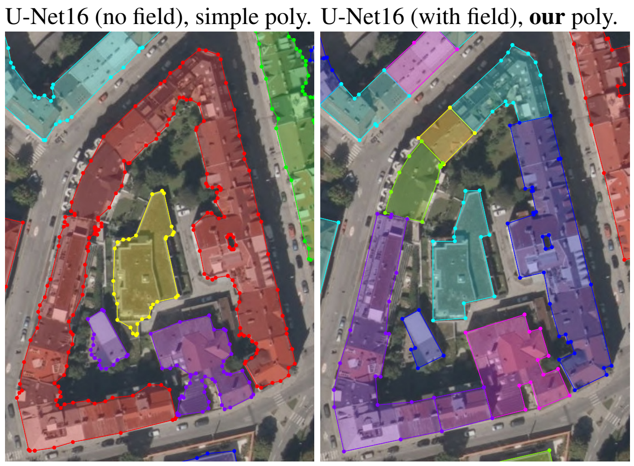

U-Net16 (no field), simple poly.
U-Net16 (with field), our poly.

Figure 6: Small crop of *Inria dataset* results.

Method	mIoU ↑	Mean *max tangent angle errors* ↓
Eugene Khvedchenya[1], simple poly.	**80.7**%	52.2 °
ICTNet [6], simple poly.	**80.1**%	52.1°
UResNet101 (no field), simple poly.	73.2%	52.0°
Zorzi et al. [41] poly.	74.4%	34.5°
UResNet101 (with field), **our** poly.	74.8%	**28.1**°

Table 4: IoU and mean *max tangent angle errors* for polygon extraction methods on the *Inria polygonized dataset*.

The *Inria OSM dataset* is more challenging than the *CrowdAI dataset* because it contains more varied areas (e.g., countryside, city center, residential, and commercial) with different building types. It also contains adjacent buildings with common walls, which our edge segmentation output can detect. The mean IoU on test images of the output classification maps is 78.0% for the U-Net16 trained with a frame field compared to 76.9% for the U-Net16 with no frame field. The IoU does not significantly penalize irregular contours, but, by visually inspecting segmentation outputs as in Fig. 6, we can see the effect of the regularization. Our method successfully handles complex building shapes which can be very large, with blocks of buildings featuring common walls and holes. See the supplementary materials for more results.

5.3. Inria polygonized dataset

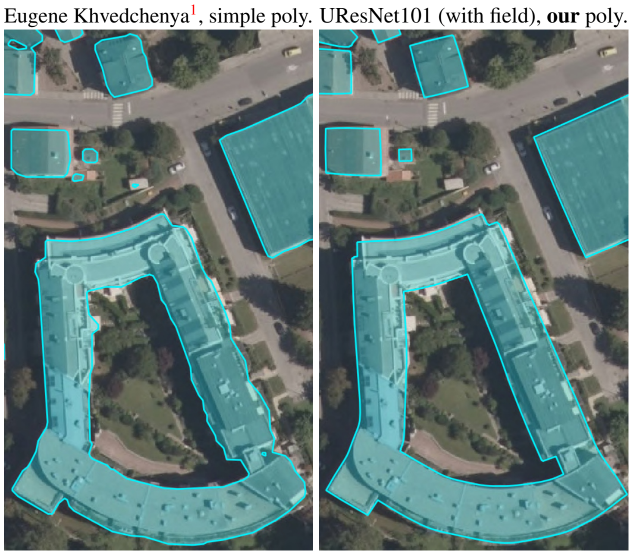

Eugene Khvedchenya[1], simple poly.
UResNet101 (with field), **our** poly.

Figure 7: Crop of an *Inria polygonized dataset* test image.

The *Inria polygonized dataset* with its associated challenge[1] allows us to directly compare to other methods trained on the same ground truth, even though it does not consider learning of separate buildings. In Table 4, our method matches [41] in terms of mIoU, with lower *max tangent angle error*. The two top methods on the leaderboard (ICTNet [6] and "Eugene Khvedchenya") achieve a mIoU over 80%, but they lack contour regularity with high *max tangent angle error*; they also only output segmentation masks, needing *a posteriori* polygonization to extract polygonal buildings. Fig. 7 shows the cleaner geometry of our method. The ground truth of the *Inria polygonized dataset* has misalignment noise, yielding imprecise corners that produce rounded corners in the prediction if no regularization is applied. See the supplementary materials for more results.

[1]https://project.inria.fr/aerialimagelabeling/leaderboard/

6. Conclusion

We improve on the task of building extraction by learning an additional output to a standard segmentation model: a frame field. This motivates the use of a regularization loss, leading to more regular contours, e.g., with sharp corners. Our approach is efficient since the model is a single fully-convolutional network. The training is straightforward, unlike adversarial training, direct shape regression, and recurrent networks, which require significant tuning and more computational power. The frame field adds virtually no cost to inference time, and it disambiguates tough polygonization cases, making our polygonization method less complex. Our data structure for the polygonization makes it parallelizable on the GPU. We handle the case of holes in buildings as well as common walls between adjoining buildings. Because of the skeleton graph structure, common wall polylines are naturally guaranteed to be shared by the buildings on either side. As future work, we could apply our method to any image segmentation network, including multi-class segmentation, where the frame field could be shared between all classes.

7. Acknowledgements

Thanks to ANR for funding the project EPITOME ANR-17-CE23-0009 and to Inria Sophia Antipolis - Méditerranée "Nef" computation cluster for providing resources and support. The MIT Geometric Data Processing group acknowledges the generous support of Army Research Office grant W911NF2010168, of Air Force Office of Scientific Research award FA9550-19-1-031, of National Science Foundation grant IIS-1838071, from the CSAIL Systems that Learn program, from the MIT–IBM Watson AI Laboratory, from the Toyota–CSAIL Joint Research Center, from a gift from Adobe Systems, from an MIT.nano Immersion Lab/NCSOFT Gaming Program seed grant, and from the Skoltech–MIT Next Generation Program. This work was also supported by the National Science Foundation Graduate Research Fellowship under Grant No. 1122374.

References

[1] David Acuna, Huan Ling, Amlan Kar, and Sanja Fidler. Efficient interactive annotation of segmentation datasets with polygon-rnn++. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2

[2] Anil Batra, Suriya Singh, Guan Pang, Saikat Basu, C.V. Jawahar, and Manohar Paluri. Improved road connectivity by joint learning of orientation and segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 3

[3] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. 2013. arXiv:1308.3432. 2

[4] Mikhail Bessmeltsev and Justin Solomon. Vectorization of line drawings via polyvector fields. *ACM Trans. Graph.*, 2019. 3

[5] Lluıs Castrejón, Kaustav Kundu, Raquel Urtasun, and Sanja Fidler. Annotating object instances with a polygon-rnn. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2

[6] Bodhiswatta Chatterjee and Charalambos Poullis. On building classification from remote sensor imagery using deep neural networks and the relation between classification and reconstruction accuracy using border localization as proxy. In *2019 16th Conference on Computer and Robot Vision (CRV)*, pages 41–48, 2019. 8

[7] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. 2017. arXiv:1706.05587. 3, 5

[8] Zhiqin Chen, Andrea Tagliasacchi, and Hao Zhang. Bsp-net: Generating compact meshes via binary space partitioning. *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2

[9] Jakub Czakon, Kamil A. Kaczmarek, Andrzej Pyskir, and Piotr Tarasiewicz. Best practices for elegant experimentation in data science projects. *EuroPython*, 2018. 6, 7

[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009. 5

[11] Olga Diamanti, Amir Vaxman, Daniele Panozzo, and Olga Sorkine-Hornung. Designing N-polyvector fields with complex polynomials. *Eurographics SGP*, 2014. 3

[12] Lei Ding and Lorenzo Bruzzone. Diresnet: Direction-aware residual network for road extraction in vhr remote sensing images. 2020. arXiv:2005.07232. 3

[13] David H. Douglas and Thomas K. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *The Canadian Cartographer*, 10(2):112–122, 1973. 1

[14] Jun Gao, Zian Wang, Jinchen Xuan, and Sanja Fidler. Beyond fixed grid: Learning geometric image representation with a deformable grid. In *ECCV*, 2020. 2

[15] Nicolas Girard, Guillaume Charpiat, and Yuliya Tarabalka. Noisy Supervision for Correcting Misaligned Cadaster Maps Without Perfect Ground Truth Data. In *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, 2019. 5

[16] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick. Mask r-cnn. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. 7

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. 2015. arXiv:1512.03385. 5

[18] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997. 2

[19] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision (IJCV)*, 1(4):321–331, 1988. 2, 4

[20] Muxingzi Li, Florent Lafarge, and Renaud Marlet. Approximating shapes in images with low-complexity polygons. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 1, 2, 6, 7

[21] Zuoyue Li, Jan Dirk Wegner, and Aurélien Lucchi. Topological map extraction from overhead images. In *The IEEE International Conference on Computer Vision (ICCV)*, 2019. 1, 2, 6, 7

[22] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *The European Conference on Computer Vision (ECCV)*, pages 740–755, Cham, 2014. Springer International Publishing. 6

[23] Huan Ling, Jun Gao, Amlan Kar, Wenzheng Chen, and Sanja Fidler. Fast interactive object annotation with curve-gcn. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1, 2

[24] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 7

[25] William Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. 1987. 1

[26] Emmanuel Maggiori, Yuliya Tarabalka, Guillaume Charpiat, and Pierre Alliez. Can semantic labeling methods generalize to any city? the inria aerial image labeling benchmark. In *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, 2017. 5

[27] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. 2013. arXiv:1312.5602. 2

[28] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, Feb. 2015. 2

[29] OpenStreetMap contributors. Planet dump retrieved from https://planet.osm.org , 2017. 1

[30] Urs Ramer. An iterative procedure for the polygonal approximation of plane curves. *Computer graphics and image processing*, 1(3):244–256, 1972. 1

[31] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. 2015. arXiv:1505.04597. 3, 5

[32] Sharada Prasanna Mohanty. Crowdai dataset. https://www.crowdai.org/challenges/mapping-challenge/dataset_files, 2018. 5, 6, 7

[33] Sharada Prasanna Mohanty. Crowdai mapping challenge 2018: Baseline with mask rcnn. https://github.com/crowdai/crowdai-mapping-challenge-mask-rcnn, 2018. 7

[34] Carole H Sudre, Wenqi Li, Tom Vercauteren, Sebastien Ourselin, and M Jorge Cardoso. Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations. In *Deep learning in medical image analysis and multimodal learning for clinical decision support*, pages 240–248. Springer, 2017. 4

[35] Richard S. Sutton and Andrew G. Barto. Reinforcement learning: An introduction, 2020. 2

[36] Maria Taktasheva, Albert Matveev, Alexey Artemov, and Evgeny Burnaev. Learning to approximate directional fields defined over 2d planes. In Wil M. P. van der Aalst, Vladimir Batagelj, Dmitry I. Ignatov, Michael Khachay, Valentina Kuskova, Andrey Kutuzov, Sergei O. Kuznetsov, Irina A. Lomazova, Natalia Loukachevitch, Amedeo Napoli, Panos M. Pardalos, Marcello Pelillo, Andrey V. Savchenko, and Elena Tutubalina, editors, *Analysis of Images, Social Networks and Texts*, pages 367–374, Cham, 2019. Springer International Publishing. 3

[37] Meng Tang, Federico Perazzi, Abdelaziz Djelouah, Ismail Ben Ayed, Christopher Schroers, and Yuri Boykov. On regularized losses for weakly-supervised cnn segmentation. In *The European Conference on Computer Vision (ECCV)*, 2018. 1

[38] S. Tripodi, L. Duan, F. Trastour, V. Poujad, Lionel Laurore, and Yuliya Tarabalka. Automated chain for large-scale 3d reconstruction of urban scenes from satellite images. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLII-2/W16:243–250, 09 2019. 2

[39] Amir Vaxman, Marcel Campen, Olga Diamanti, Daniele Panozzo, David Bommes, Klaus Hildebrandt, and Mirela Ben-Chen. Directional field synthesis, design and processing. *Computer Graphics Forum*, 35:545–572, 05 2016. 3

[40] T. Y. Zhang and C. Y. Suen. A fast parallel algorithm for thinning digital patterns. *Commun. ACM*, 27(3):236–239, Mar. 1984. 5

[41] Stefano Zorzi, Ksenia Bittner, and Friedrich Fraundorfer. Machine-learned regularization and polygonization of building segmentation masks, 2020. arXiv:2007.12587. 2, 8

[42] Stefano Zorzi and Friedrich Fraundorfer. Regularization of building boundaries in satellite images using adversarial and regularized losses. In *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, 2019. 1