Polygonal Building Segmentation by Frame Field Learning

Nicolas Girard, Dmitriy Smirnov, Justin Solomon, Yuliya Tarabalka

Polygonal Building Segmentation by Frame Field Learning

Nicolas Girard[1], Dmitriy Smirnov[2], Justin Solomon[2], and Yuliya Tarabalka[1,3]

[1] Université Côte d'Azur, Inria
[2] Massachusetts Institute of Technology
[3] LuxCarta Technology
nicolas.girard@inria.fr

Abstract. While state of the art image segmentation models typically output segmentations in raster format, applications in geographic information systems often require vector polygons. We propose adding a frame field output to a deep image segmentation model for extracting buildings from remote sensing images. This improves segmentation quality and provides structural information, facilitating more accurate polygonization. To this end, we train a deep neural network, which aligns a predicted frame field to ground truth contour data. In addition to increasing performance by leveraging multi-task learning, our method produces more regular segmentations. We also introduce a new polygonization algorithm, which is guided by the frame field corresponding to the raster segmentation.

Keywords: polygonization, segmentation, regularization, remote sensing, frame field, PolyVector field

1 Introduction

Due to their success in processing large collections of noisy images, deep convolutional neural networks (CNNs) have become state-of-the-art for segmentation models in remote sensing. For example, the fully-convolutional U-Net [18] is a popular model for pixel-wise classification, with favorable performance on segmenting buildings in overhead images. Geographic information systems like Open Street Map (OSM) [15], however, require segmentation data to be in *vector* format (i.e., polygons and curves) rather than raster, which object segmentation networks output. Thus, modifications to the conventional raster-based pipeline are necessary.

Existing work on deep building segmentation generally falls into one of two general categories. The first involves vectorizing the classification map produced by a network, e.g., by using contour detection (marching squares [13]) followed by polygon simplification (RamerDouglasPeucker [17]). This approach suffers from artifacts when the classification map is not perfect, especially because conventional deep segmentation methods are often unable to produce sharp

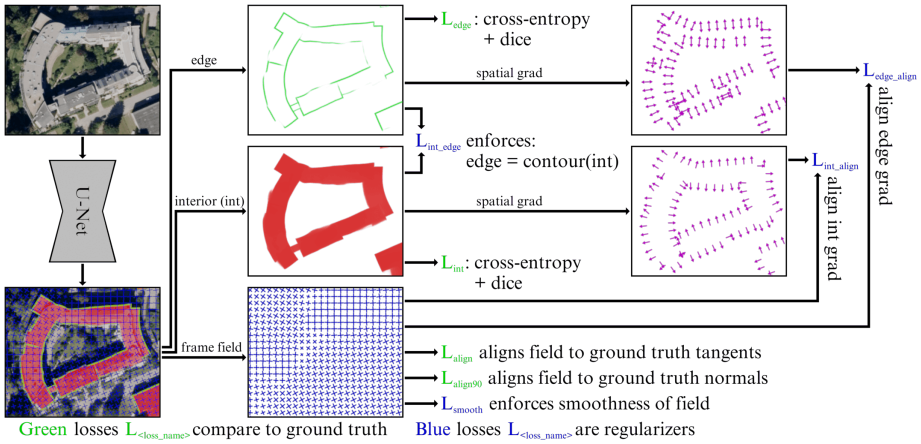


Fig. 1: Overview of the training procedure for our model. Given an overhead image as input, the model outputs an edge mask, an interior mask, and a frame field for the predicted segmentation. We train the model by optimizing a loss function consisting of terms that align the masks and frame field to ground truth data as well as regularizers to enforce smoothness of the frame field and consistency between the outputs.

corners. To improve the final polygons, these methods employ expensive and complex post-processing procedures. In [8], polygonal partition refinement is used to approximate shapes in the output classification map. This introduces a tunable parameter to control the tradeoff between complexity and fidelity. In [23], two distinct models—a shared decoder and a discriminator—are trained to produce cleaner buildings by regularizing the output segmentation maps in an adversarial fashion. This method requires computing large matrices of pairwise discontinuity costs between pixels and involves adversarially training a system of networks, which is less stable than conventional supervised learning.

The other main category of deep segmentation methods attempts to learn a vector representation directly. For example, Curve-GCN [11] trains a graph convolutional network (GCN) to iteratively deform a polygon to fit each object, and PolyMapper [9] uses a recurrent neural network (RNN) to predict polygon vertices one at a time. While these approaches directly predict the parameters of a polygon, 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. This is a serious limiting factor, since buildings often exhibit more complex topologies. Additionally, buildings with shared walls are also common, especially in city centers. To the best of our knowledge, no previous work can output such interior walls. While Curve-GCN and PolyMapper may detect adjacent buildings as distinct polygons, the shared edges are unlikely to be aligned.

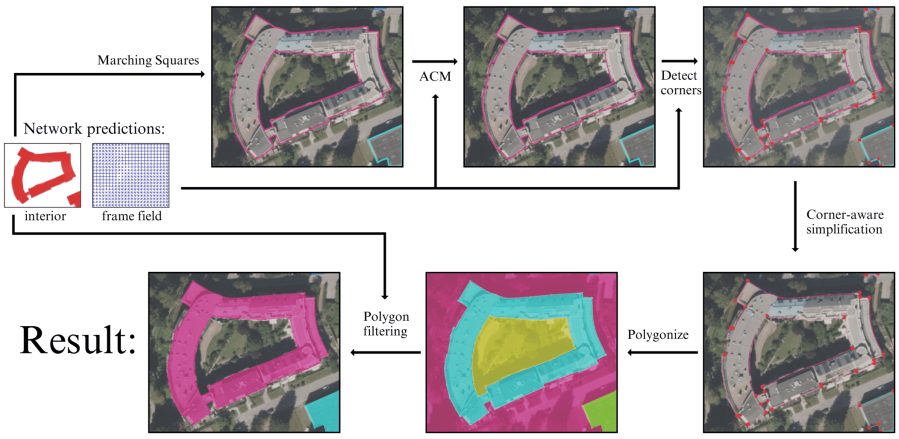


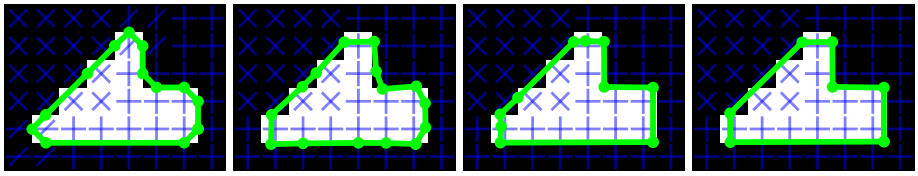
Fig. 2: Overview of our post-processing polygonization algorithm. Given an interior classification map and frame field (see Fig. 1) as input, we optimize the contour to align to the frame field using an active contour model (ACM) and detect corners using the frame field, simplifying non-corner vertices.

In this paper, we introduce a building segmentation algorithm that avoids some of the challenges above by adding a frame field output to a fully-convolutional network. As noted above, only learning the pixel-wise classification of objects makes polygonization challenging. We thus train a network to additionally learn a frame field aligned with the object outlines. This frame field not only increases segmentation performance, e.g., yielding sharper corners, but also provides useful information for vectorization. Thus, while our work generally falls into the first category of approaches described above, we use the learned frame field to bridge the gap between the segmentation and polygonization steps. To this end, we propose a polygonization method leveraging the frame field. Our algorithm uses a tunable parameter to control the output complexity and supports nontrivial building topology. Additionally, our trained model provides information to split adjacent buildings in the form of a wall classification map.

The main contributions of this paper are:

- learning a frame field aligned to object tangents, which improves segmentation via multi-task learning;
- applying coupling losses between outputs so that they are consistent with one another, further leveraging multi-task learning;
- providing additional structural information in the form of a frame field for subsequent polygonization;
- leveraging the frame field as part of a novel polygonization method, allowing complexity tuning of our corner-aware simplification step and handling non-trivial building topology.

2 Frame fields



(a) Step 0 (b) Step 50 (c) Step 250 (d) Result

Fig. 3: Even a perfect classification map can admit a wrong polygonization due to locally ambiguous segmentation maps, as we illustrate above in (a), the output of marching squares. We thus iteratively optimize the contour (b-d) to align to a frame field, yielding better results as our frame field (blue) disambiguate between slanted walls and corners, preventing corners from being cut off.

We provide the necessary background on frame fields, a key part of our method. Following [22, 3], a frame field is a *4-PolyVector field*, which is locally defined by a frame, i.e., two symmetric line fields. At each point in the image, we consider the two directions that define the frame as two complex numbers $u, v \in \mathbb{C}$. To encode the directions in a way that is agnostic to relabelling and change of sign, we represent them as coefficients of the following complex polynomial:

$$f(z) = (z^2 - u^2)(z^2 - v^2) = z^4 + c_2 z^2 + c_0. \tag{1}$$

The constants $c_0, c_2 \in \mathbb{C}$ uniquely determine an equivalence class corresponding to a frame in the set of pairs of directions. 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:

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

In our approach, inspired by [1], we aim to 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, we would like the field to align to *both* tangent directions, motivating our use of PolyVector fields rather than vector fields. We illustrate our motivation for using frame fields in Fig. 3. Away from polygon boundaries, the frame field does not have any alignment constraints but is encouraged to be smooth and not collapse into a line field. Like [1], we formulate the field computation variationally, but unlike their approach we use a neural network to learn the field at every pixel of the image, a representation

also explored in [21]. Since learning a (u, v) pair per pixel induces challenging issues involving labeling and sign, we instead learn a (c_0, c_2) pair, which has no sign or ordering ambiguity.

3 Frame field learning

We describe our method, which is illustrated in Fig. 1. Our network takes an $H \times W$ RGB image I as input and outputs a classification map and a frame field. The classification map is made up of two channels, y_{int} corresponding to building interiors and y_{edge} to building boundaries. The frame field consists of four channels corresponding to the two coefficients $c_0, c_2 \in \mathbb{C}$, as described in §**??** above.

3.1 Starting with basic image segmentation

Our method can be used with any deep segmentation model as a backbone. We show experiments using U-Net [18] and DeepLabV3 [2] architectures with a two-channel output corresponding to object interiors and contours. Our training is supervised, where each input image is labelled with ground truth $\hat{y}_{int}$ and $\hat{y}_{edge}$, corresponding to rasterized polygon interiors and edges, respectively. We then use a linear combination of the cross-entropy loss and the Dice loss [**?**] for the loss L_{int} applied on the interior output as well as the loss L_{edge} applied on the contour (edge) output. We define loss functions below:

$$L_{BCE}(\hat{y}, y) = \frac{1}{HW} \sum_{x \in I} \hat{y}(x) \cdot \log(y(x)) + (1 - \hat{y}(x)) \cdot \log(1 - y(x)), \tag{3}$$

$$L_{Dice}(\hat{y}, y) = 1 - 2 \cdot \frac{|\hat{y} \cdot y| + 1}{|\hat{y} + y| + 1}, \tag{4}$$

$$L_{int} = \alpha \cdot L_{BCE}(\hat{y}_{int}, y_{int}) + (1 - \alpha) \cdot L_{Dice}(\hat{y}_{int}, y_{int}), \tag{5}$$

$$L_{edge} = \alpha \cdot L_{BCE}(\hat{y}_{edge}, y_{edge}) + (1 - \alpha) \cdot L_{Dice}(\hat{y}_{edge}, y_{edge}), \tag{6}$$

where $0 < \alpha < 1$ is a hyperparemeter. In practice, we set $\alpha = 0.25$

3.2 Adding a frame field output

In addition to the segmentation masks, our network outputs a frame field. We add another head to the fully-convolutional network via a module consisting of a sequence of one convolution, a batch normalization layer, an ELU nonlinearity, another convolution, and a tanh nonlinearity. That frame field module takes as input the concatenation of the segmentation output and the output features of the backbone network layer. The output has four channels, two each for $c_0, c_2 \in \mathbb{C}$.

The 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:

$$L_{align} = \frac{1}{HW} \sum_{x \in I} \hat{y}_{edge}(x) f(e^{i\theta_\tau}; c_0(x), c_2(x))^2, \tag{7}$$

$$L_{align90} = \frac{1}{HW} \sum_{x \in I} \hat{y}_{edge}(x) f(e^{i\theta_{\tau^\perp}}; c_0(x), c_2(x))^2, \tag{8}$$

$$L_{smooth} = \frac{1}{HW} \sum_{x \in I} \left(\|\nabla c_0(x)\|^2 + \|\nabla c_2(x)\|^2 \right), \tag{9}$$

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

- L_{align} enforces alignment of the frame field with the tangent directions. This term is small when the polynomial $f(\cdot; c_0, c_2)$ 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 into a line field by expressing a slight preference for the field to be also aligned with $\tau^\perp$.
- L_{smooth} is a Dirichlet energy measuring the smoothness of the functions $c_0(x)$ and $c_2(x)$ as a function of the location x in the image. Smoothly-varying c_0 and c_2 yield a smooth frame field.

3.3 Adding coupling losses between different outputs

Given that the outputs of our network (the interior and boundary segmentation masks as well as the frame field) should all be compatible with one another, we add coupling losses mutua consistency:

$$L_{int\ align} = \frac{1}{HW} \sum_{x \in I} f(\nabla y_{int}(x); c_0(x), c_2(x))^2 \tag{10}$$

$$L_{edge\ align} = \frac{1}{HW} \sum_{x \in I} f(\nabla y_{edge}(x); c_0(x), c_2(x))^2 \tag{11}$$

$$L_{int\ edge} = \frac{1}{HW} \sum_{x \in I} \max\left(1 - y_{int}(x), \|\nabla y_{int}(x)\|_2\right) \cdot \left| \|\nabla y_{int}(x)\|_2 - y_{edge}(x) \right| \tag{12}$$

- $L_{int\ align}$: Aligns the spatial gradient of the predicted interior map y_{int} with the frame field (analogous to (7)).
- $L_{edge\ align}$: Aligns the spatial gradient of the predicted edge map y_{edge} with the frame field (analogous to (7)).

– $L_{int\ edge}$: 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 $1 - y_{int}(x)$ term in the max) and along building contours (hence the $\|\nabla y_{int}(x)\|_2$ term in the max) and is not applied inside buildings, so that shared walls between adjacent buildings can still be detected by the edge map.

3.4 Handling numerous heterogeneous losses

We linearly combine our eight losses using eight coefficients, which can be challenging to balance. Because the losses have different scales, we first compute a normalization coefficient $N_{\langle loss\ name \rangle}$ by computing the average of each loss on a random subset of the training dataset using a randomly-initialized network. The total loss is then combined by balancing main losses and regularization losses with one parameter $\lambda \in [0, 1]$:

$$\begin{aligned} &\lambda \left(\frac{L_{int}}{N_{int}} + \frac{L_{edge}}{N_{edge}} + \frac{L_{align}}{N_{align}} \right) \\ &+ (1 - \lambda) \left(\frac{L_{align90}}{N_{align90}} + \frac{L_{smooth}}{N_{smooth}} + \frac{L_{int\ align}}{N_{int\ align}} + \frac{L_{edge\ align}}{N_{edge\ align}} + \frac{L_{int\ edge}}{N_{int\ edge}} \right) \end{aligned} \tag{13}$$

This normalization aims to rescale each loss term such that it contributes equally to the overall loss. However, since we would like our model to be influenced more by the main losses L_{int}, L_{edge} and L_{align} than the regularizers, we introduce this bias by setting $\lambda = 0.75$.

4 Polygonization

An overview of the polygonization method in Fig. 2 shows the main steps. We first polygonize the contours of buildings using an active contours model (ACM) [7] approach to optimize the contour to align to the learned frame field, resulting in clean contours, especially at corners. Then, we apply a corner-aware simplification step to extract low-complexity polygons without simplifying away corner vertices.

4.1 Active Contours Model

The contours are initialized by applying marching squares [13] to the interior probability segmentation map y_{int} with an isovalue l (set to 0.5 in practice). Contours are collections of polylines made up of vertices $V \in \mathbb{R}^2$ and edges E, i.e., line segments connecting pairs of vertices. The total energy of the model is a

linear combination of three terms:

$$E_{probability} = \sum_{v \in V} (y_{int}(v) - l)^2, \tag{14}$$

$$E_{frame\ field\ align} = \sum_{e \in E} f(e_{dir}; c_0(e_{center}), c_2(e_{center}))^2, \tag{15}$$

$$E_{length} = \sum_{e \in E} |e|^2, \tag{16}$$

where e_{dir} is the direction vector of edge e, e_{center} is the center point of edge e, $|e|$ is the length of edge e, and l is the isovalue used for initializing the contours by the marching squares method. These objective terms can be interpreted as follows:

- $E_{probability}$: Forces contours to stay close to the isovalue l of the segmentation map. The way contours are initialized makes this energy equal to zero at the beginning of optimization.
- $E_{frame\ field\ align}$: Aligns each edge of the contours to the frame field, resulting in regular-looking contours.
- E_{length}: Regularizes contour length to avoid small local overlaps.

In practice, the final result is robust to different values of coefficients for each of these three energy terms, and we determine them using a small cross-validation set. Since contours are initialized to lie on building boundaries and are not expected to move much because of the $E_{probability}$ energy, the optimization converges quickly (we run it for 500 steps on all experiments in practice).

4.2 Corner-aware polygon simplification

We now have a collection of polylines that forms a planar graph. Before simplifying them, we detect building corner vertices, which should not be removed during simplification. A given contour vertex has two associated edges as well as $\pm u$ and $\pm v$ directions of the frame field at its location. If each edge aligns to a different field direction, we consider that vertex to be a corner. We split the polylines at such corner vertices and apply simplification on each component separately.

4.3 Detecting building polygons in planar graph

To obtain our final output of building polygons, we polygonize the collection of polylines giving us a list of polygons (possibly with holes) that partition the entire image. The last step computes a building probability value for each polygon using the predicted interior probability map and removes low-probability polygons.

5 Experimental setup

5.1 Datasets

We perform experiments on the CrowdAI Mapping Challenge dataset [19] and the Inria Aerial Image Labeling dataset [14]. We provide details about each dataset below. For both datasets, the ground truth angle for the frame field is precomputed from polygon line segments.

Our full method uses a U-Net backbone with 16 starting hidden features (instead of 64 in the original), which we refer to as U-Net16. We perform ablation experiments on the CrowdAI dataset by changing one parameter at a time starting from our full method.

CrowdAI dataset. This dataset originally has 280741 training images, 60317 validation images, and 60697 test images; each image is 300×300 pixels. As the ground truth annotations of the test set are unreleased [9, 8], we use the original validation set as our test set and discard the original test images. We then use 75% of the original training images as our initial training set and 25% for validation. Out final models are then trained on the entire original training set with hyperparameters selected using our validation test.

Inria dataset. This dataset contain 360 5000×5000 pixel aerial images of 10 cities in Europe and the USA. Ground truth building polygons are taken from Open Street Map (OSM) instead of using the original ground truth mask, since the latter are only available in raster format, and vector format is needed to compute ground truth frame field angles. Because the OSM annotations are not always aligned, we align them using [4]. We randomly split the images into train (50%), validation (25%), and test (25%) sets. Because the OSM annotations have a lot of missing buildings in certain images, our test results on this dataset are somewhat skewed. Thus, for the test images we manually select those with few missing buildings in the annotations, giving us 54 test images in total.

5.2 Ablation studies

We perform an ablation study to validate various components of our pipeline using a U-Net16 backbone:

- **No field:** Remove the frame field output for comparison to pure segmentation. Only interior segmentation L_{int}, edge segmentation L_{edge} and interior $\leftrightarrow$ edge coupling $L_{int\ edge}$ losses remain.
- **No field-aligned polygonization:** Use a baseline simplification algorithm (marching-squares followed by RamerDouglasPeucker) on the interior classification map learned by our full method to study the improvement of our polygonization method leveraging the frame field.

– **No coupling losses:** All coupling losses ($L_{int\ align}$, $L_{edge\ align}$, $L_{int\ edge}$) removed to determine whether enforcing consistency between outputs has an impact. We report the result of this experiment in the supplementary materials.

The results are shown in Table 1 below.

5.3 Different backbone

Our frame field learning method can be added to any segmentation network. Thus, we also train our full method with a DeepLabV3 [2] model that utilizes a ResNet-101 [6] backbone (which we refer to as DeepLab101). We also perform the "no field" ablation study using this backbone. We report these experiments in the supplementary materials.

5.4 Polygonization method

We perform an ablation study to demonstrate the effect on the tradeoff between complexity and fidelity of the tolerance parameter. We also perform this study on the simple polygonization algorithm—marching squares followed by topology-preserving simplification—for comparison.

5.5 Metrics

The main metric for our task is Intersection over Union (IoU) which computes the overlap between a predicted segmentation and the ground truth annotation. We also show the MS COCO [10] Average Precision (AP and its variants AP_{50}, AP_{75}, AP_S, AP_M, AP_L) and Average Recall (AR and its variants AR_{50}, AR_{75}, AR_S, AR_M, AR_L) evaluation metrics. Precision and recall are computed for a certain IoU threshold: detections with an IoU above the threshold are counted as true positives whiles others are false positives and ground truth annotations with an IoU below the threshold are false negatives. Each object is also given a score value representing the model's confidence in the detection. In our case it is the mean value of the interior probability map inside the detection. The Precision-Recall curve can be obtained by varying the score threshold that determines what is counted as a model-predicted positive detection. Average Precision (AP) is the average value of the precision across all recall values and Average Recall (AR) is the maximum recall given a fixed number of detections per image (100 in our case). Finally the mean Average Precision (mAP) is calculated by taking the mean AP over multiple IoU thresholds (from 0.50 to 0.95 with a step of 0.05). Likewise for the mean Average Recall (mAR). Following MS COCO's convention, we make no distinction between AP and mAP (and likewise AR and mAR) and assume the difference is clear from context. The AP_{50} variant is AP computed with a single IoU threshold of 50% (similarly for AP_{75}, AR_{50} and AR_{75}). The AP_S, AP_M and AP_L variants are AP computed for small ($area < 32^2$), medium ($32^2 < area < 96^2$) and large ($area > 96^2$) objects respectively (like-wise for the AR equivalents). To study the polygon complexity/fidelity trade-off , we plot the AP and AR scores for difference simplification tolerance values.

5.6 Training and active contours details

We do not heavily tune our hyperparameters: once we find a value that works based on validation performance, we keep the same value for all experiments, models, and datasets. We only select a different total number of epochs for the U-Net and DeepLabV3 backbones (25 and 15 respectively) chosen by first training the full method on the training set of the CrowdAI dataset and choosing the epoch number of the highest IoU on the validation fold.

Segmentation losses L_{int} and L_{edge} are both a combination of 25% cross-entropy loss and 75% Dice loss. To balance the losses, we set $\lambda = 0.75$ to give more weight to the main losses. We train the model on 4 GTX 1080Ti GPUs in parallel on 512×512 patches and a batch size of 16 per GPU (effective batch size 64). We compute for each loss its normalization coefficient $N_{\langle loss\_name \rangle}$ on 1000 batches before optimizing the network.

Our method is implemented in PyTorch [16]. On the CrowdAI dataset, training takes 2 hours per epoch on 4 1080Ti GPUs for the small U-Net backbone and 3.5 hours per epoch for the DeepLabV3 backbone on 4 2080Ti GPUs. Inference on a 5000×5000 image (requires splitting into 1024×1024 patches) takes 7 seconds on a Quadro M2200 (laptop GPU).

The active contours optimization is implemented in PyTorch for efficient parallel computation. For a full-size 5000×5000 image containing thousands of buildings, 500 optimization steps are computed in 20 seconds on a Quadro M2200 (laptop GPU), while the initialization step for the marching squares algorithm takes 12 seconds.

6 Results

6.1 CrowdAI dataset

We report MS COCO metrics on the original validation set of the CrowdAI dataset in Table 1 for the various ablation studies. Each ablation is evaluated on three outputs: mask (classification map thresholded at 0.5), baseline (simple polygonization algorithm), and ours (our field-aligned polygonization method). For completeness, Table 1 also shows metrics of previous works.

We perform an analysis of the polygonization complexity/fidelity tradeoff by changing the tolerance value of the baseline simplification method and our corner-aware method. Fig. 5 shows the advantage of preserving detected corners and also plots the AP and AR values of both methods while increasing the tolerance value. Our method does not have a drop of score unlike the baseline method.

Running times: We compare running times of our method compared to others in Table 2. Since each comparison method is proposed in a different research paper, each row notes the different hardware used and whether or not each timing includes the whole pipeline. Overall, we find that our polygonalization time is competitive with previous work.

Table 1: Ablation studiy on the CrowdAI dataset [19] of our full method with and without frame fields. Both polygonization methods have a small simplification tolerance of 0.125.

Method	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	AR	AR_{50}	AR_{75}	AR_S	AR_M	AR_L
Mask (full)	53.6	77.8	62.8	25.1	69.4	69.5	57.6	79.0	66.4	29.7	74.1	75.2
Baseline (full)	49.6	73.8	58.1	21.2	65.5	67.0	53.8	75.6	62.2	25.5	70.5	72.5
Ours (full)	50.5	76.6	59.3	20.4	67.4	69.0	55.3	78.1	64.0	25.7	72.8	75.0
Mask (no field)	50.9	74.3	59.5	24.5	65.6	66.3	55.9	77.9	64.7	29.8	71.2	74.6
Baseline (no field)	50.5	76.6	59.1	22.6	66.2	69.3	54.8	78.5	63.5	26.8	71.2	75.2
Mask R-CNN [5] [20]	41.9	67.5	48.8	12.4	58.1	51.9	47.6	70.8	55.5	18.1	65.2	63.3
PANet [12]	50.7	73.9	62.6	19.8	68.5	65.8	54.4	74.5	65.2	21.8	73.5	75.0
PolyMapper [9]	55.7	86.0	65.1	30.7	68.5	58.4	62.1	88.6	71.4	39.4	75.6	75.4
Li et al. [8]	65.8	87.6	73.4	39.3	87.0	91.9	78.7	94.3	86.1	57.2	91.2	97.6

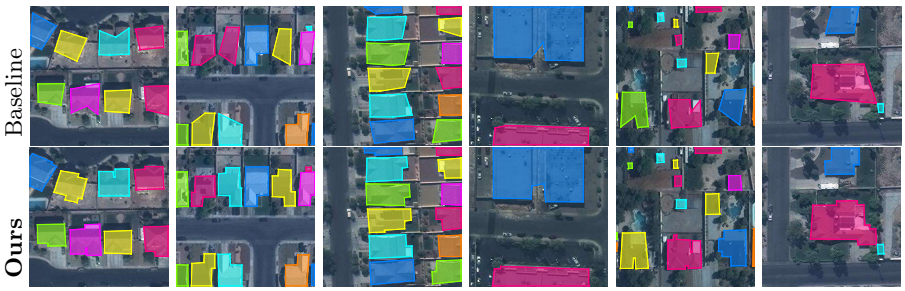


Fig. 4: Polygonization comparison between our method and the baseline with a tolerance of 16 pixels on random samples. Both take the same classification map as input, but the baseline does not use the frame field.

6.2 Inria dataset

The Inria dataset allows us to show the ability of our method to handle hard cases of non-trivial building topologies, with, e.g., one or more holes, as shown in Fig. 6. 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. 7, we can evaluate the effect of the regularization.

Table 2: Average times to polygonize a 300×300 pixel patch

Method	Time (sec)	Hardware
PolyMapper [9]	0.38	GTX 1080Ti
Li et al. [8]	1-3 (not including model inference)	Laptop CPU
Ours (U-Net16)	0.21	Quadro M2200 (laptop GPU)

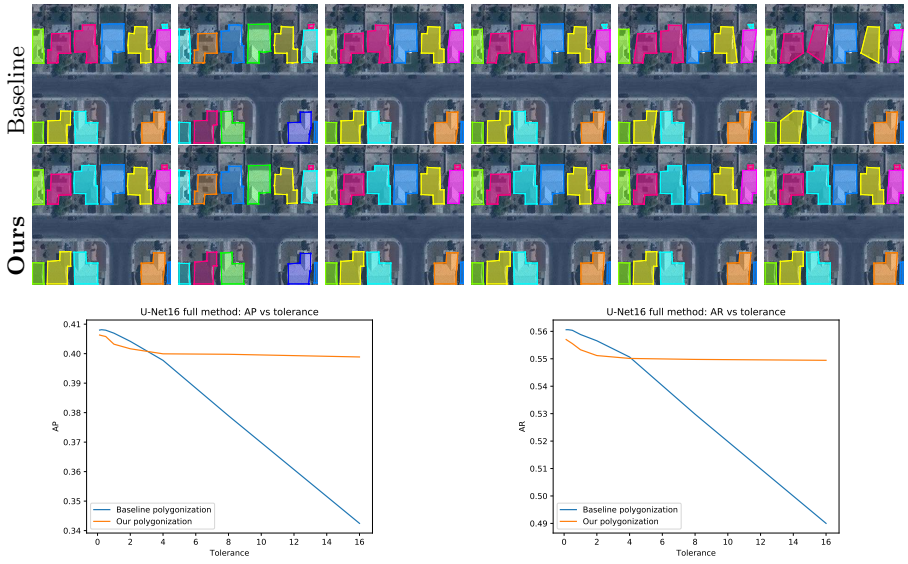


Fig. 5: Comparison between the baseline simplification algorithm with our corner-aware one. Both take the same classification map as input, but the baseline does not use the frame field. We also compare AP (left) and AR (right) scores of the two. The corner-aware simplification guarantees that no corners will be simplified, regardless of the tolerance value.

7 Discussion

Our goal is to demonstrate the benefit of learning a frame field output in addition to the standard classification maps for the task of building extraction in the form of polygons. Because we use a simple network architecture and training scheme, to which we add the frame field, we do not match state-of-the-art in Table 1 for the CrowdAI dataset. We show, however, that adding a frame field improves the final score (see Table 1) for both the classification mask as well as the polygonization. We also visually observe in Fig. 7 the regularization effect on the predicted classification maps, increasing corner sharpness and wall straightness. More importantly, our method provides the missing information needed to resolve ambiguous cases for polygonization, as shown in Fig. 3, and outputs more regular polygons. Finally, the frame field allows for a robust detection of building corners, which is useful for simplification, as preventing the removal of building corners ensures key points of the contours and the global shape of the building remain intact even with extreme simplification tolerance values, as can be seen in Fig. 4.

While the CrowdAI dataset features trivial buildings topology-wise, the Inria dataset better shows the advantage our method, which can handle non-trivial topologies such as holes in buildings as in Fig. 6, contrary to PolyMapper [9] which can only extract buildings having exactly one contour.

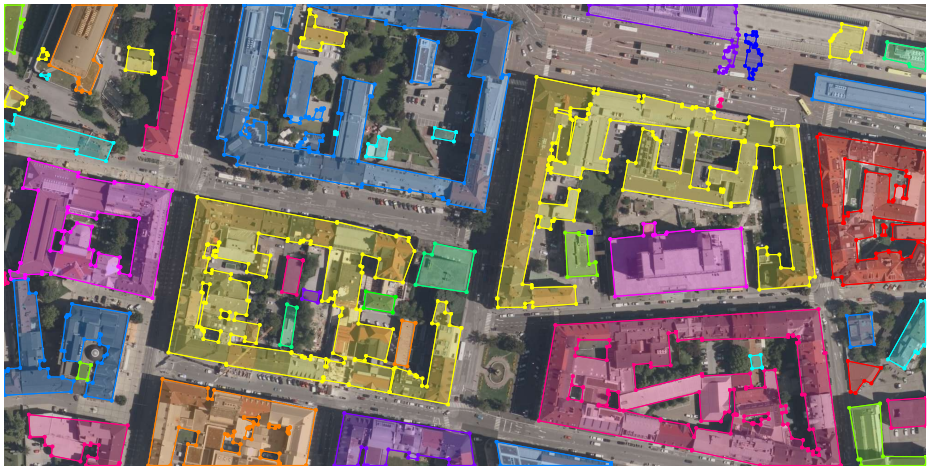


Fig. 6: Example crop on a challenging test image of the Inria dataset using the U-Net16 backbone and a 16 pixel tolerance for polygonization. Our method handles nontrivial topologies.

Our method is also faster (see Table 2) than previous work as our network is fully-convolutional and the Active Contours is optimized in parallel on a batch of images on the GPU.

Our polyonization method allows us to tune the complexity to fidelity ratio with the easy-to-interpret tolerance value of the Ramer-Douglas-Peucker algorithm, unlike Li et al. [8], which uses a non-interpretable parameter λ to balance complexity and fidelity energies during polygonal partition optimization. Finally, PolyMapper [9] does not have the ability to tune the complexity to fidelity ratio.

8 Conclusion

We improve on the task of image segmentation by learning an additional output to a standard segmentation model, a frame field. Because the network learns an additional highly correlated task, the segmentation performance is increased. Additionally, the use of coupling losses between outputs forces them to be correlated with one another, regularizing the segmentation (see Fig. 7).

We also introduce a new polygonization algorithm that makes use of the frame field to solve ambiguous cases of contour detection in segmentation maps. Additionally the ability to use the frame field to detect building corners allows for their preservation during the simplification step, regardless of the tolerance value used, keeping the global shape of buildings.

Our approach is efficient, since the trained model is a single fully-convolutional network that is optimized by local supervision—all outputs for a pixel only require image information in a neighborhood around that pixel, just like conventional image segmentation. The training is straightforward, unlike adversarial training,

Fig. 7: Predicted classification maps (red: interior, green: boundary) on a few test patches (zoom for details). First row is the baseline U-Net16 with no frame field, second row is our U-Net16 with frame field learning.

direct shape regression, and recurrent networks, which require significant tuning. Our method also potentially can be used as a plugin for any any existing image segmentation network, including in a multi-class segmentation setting, where the frame field could be shared between all classes.

As future work, we will add a building splitting step to the polygonization method, made possible by the segmentation boundary map learned by our model. We also aim to fully realize the potential of our method by augmenting a stronger network architecture/training scheme with our frame field learning and polygonization methods.

References

1. Bessmeltsev, M., Solomon, J.: Vectorization of line drawings via polyvector fields. ACM Trans. Graph. (2019)
2. Chen, L., Papandreou, G., Schroff, F., Adam, H.: Rethinking atrous convolution for semantic image segmentation. CoRR **abs/1706.05587** (2017), `http://arxiv.org/abs/1706.05587`
3. Diamanti, O., Vaxman, A., Panozzo, D., Sorkine-Hornung, O.: Designing N-polyvector fields with complex polynomials. Eurographics SGP (2014)
4. Girard, N., Charpiat, G., Tarabalka, Y.: Noisy Supervision for Correcting Misaligned Cadaster Maps Without Perfect Ground Truth Data. In: IGARSS (2019)
5. He, K., Gkioxari, G., Dollar, P., Girshick, R.: Mask r-cnn. In: The IEEE International Conference on Computer Vision (ICCV) (Oct 2017)
6. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. CoRR **abs/1512.03385** (2015), `http://arxiv.org/abs/1512.03385`
7. Kass, M., Witkin, A., Terzopoulos, D.: Snakes: Active contour models. INTERNATIONAL JOURNAL OF COMPUTER VISION **1**(4), 321–331 (1988)
8. Li, M., Lafarge, F., Marlet, R.: Approximating shapes in images with low-complexity polygons. In: CVPR (2020)
9. Li, Z., Wegner, J.D., Lucchi, A.: Topological map extraction from overhead images. ICCV (2019)

10. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) Computer Vision – ECCV 2014. pp. 740–755. Springer International Publishing, Cham (2014)
11. Ling, H., Gao, J., Kar, A., Chen, W., Fidler, S.: Fast interactive object annotation with curve-gcn. In: CVPR (2019)
12. Liu, S., Qi, L., Qin, H., Shi, J., Jia, J.: Path aggregation network for instance segmentation. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2018)
13. Lorensen, W., Cline, H.E.: Marching cubes: A high resolution 3d surface construction algorithm. SIGGRAPH (1987)
14. Maggiori, E., Tarabalka, Y., Charpiat, G., Alliez, P.: Can semantic labeling methods generalize to any city? the Inria aerial image labeling benchmark. In: IGARSS (2017)
15. OpenStreetMap contributors: Planet dump retrieved from https://planet.osm.org (2017)
16. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: Pytorch: An imperative style, high-performance deep learning library. In: Advances in Neural Information Processing Systems 32, pp. 8024–8035. Curran Associates, Inc. (2019), http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf
17. Ramer, U.: An iterative procedure for the polygonal approximation of plane curves (1972)
18. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. MICCAI (2015)
19. Sharada Prasanna Mohanty: Crowdai dataset. https://www.crowdai.org/challenges/mapping-challenge/dataset_files (2018)
20. Sharada Prasanna Mohanty: Crowdai mapping challenge 2018: Baseline with mask rcnn. https://github.com/crowdai/crowdai-mapping-challenge-mask-rcnn (2018)
21. Taktasheva, M., Matveev, A., Artemov, A., Burnaev, E.: Learning to approximate directional fields defined over 2d planes. In: van der Aalst, W.M.P., Batagelj, V., Ignatov, D.I., Khachay, M., Kuskova, V., Kutuzov, A., Kuznetsov, S.O., Lomazova, I.A., Loukachevitch, N., Napoli, A., Pardalos, P.M., Pelillo, M., Savchenko, A.V., Tutubalina, E. (eds.) Analysis of Images, Social Networks and Texts. pp. 367–374. Springer International Publishing, Cham (2019)
22. Vaxman, A., Campen, M., Diamanti, O., Panozzo, D., Bommes, D., Hildebrandt, K., Ben-Chen, M.: Directional field synthesis, design, and processing. In: Computer Graphics Forum. vol. 35, pp. 545–572. Wiley Online Library (2016)
23. Zorzi, S., Fraundorfer, F.: Regularization of building boundaries in satellite images using adversarial and regularized losses. IGARSS (2019)

Supplementary Materials

We present ablation experiment results on the CrowdAI [19] dataset and show more qualitative results for both the CrowdAI and Inria [14] dataset. Additionally, we validate our method on a third private dataset consisting of more challenging satellite images.

1 CrowdAI dataset

When computing metrics, the MS COCO evaluation code takes into account a single score value for each segmentation detection representing the methods confidence for that segment. In the paper, we included that score only for the mask detection (averaging the interior classification map covered by the predicted mask) and set it to 1 for the polygonal detections which negatively affected their metrics. Here, we revise the computation by computing a score value for each polygon that averages the interior classification map covered by the polygon. This results in accurate computation of MS COCO metrics in Table S1 and S2 for all ablation studies.

Each ablation is evaluated on three outputs: mask (classification map thresholded at 0.5), baseline (simple polygonization algorithm), and ours (field-aligned polygonization). Both backbones perform better with the frame field in terms of mask metrics, with the DeepLab101 backbone performing better than U-Net16. Removing coupling losses does not impact AP and AR metrics. We observe a slight drop in performance after polygonization, because of the 1 pixel simplification tolerance.

Table S1: Ablation studies on the CrowdAI dataset [19] with the *U-Net16* backbone. Ablation studies are: (full) method, (no field) and (no coupling losses). Both polygonization methods have a small simplification tolerance of 1 pixel.

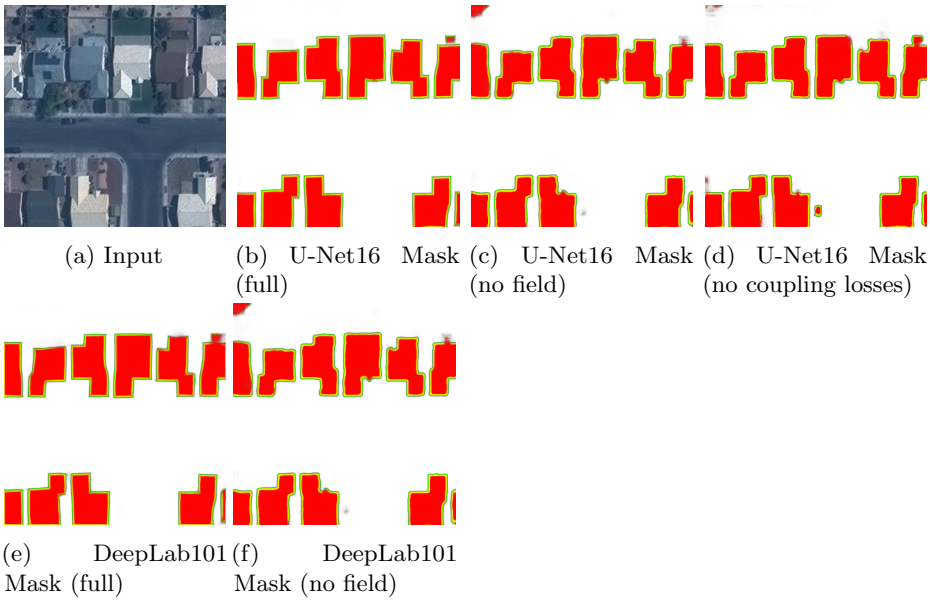
Method	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	AR	AR_{50}	AR_{75}	AR_S	AR_M	AR_L
Mask (full)	53.6	77.8	62.8	25.1	69.4	69.5	57.6	79.0	66.4	29.7	74.1	75.2
Baseline (full)	49.6	73.8	58.1	21.2	65.5	67.0	53.8	75.6	62.2	25.5	70.5	72.5
Ours (full)	50.5	76.6	59.3	20.4	67.4	69.0	55.3	78.1	64.0	25.7	72.8	75.0
Mask (no field)	50.9	74.3	59.5	24.5	65.6	66.3	55.9	77.9	64.7	29.8	71.2	74.6
Baseline (no field)	50.5	76.6	59.1	22.6	66.2	69.3	54.8	78.5	63.5	26.8	71.2	75.2
Mask (no coupling losses)	53.7	77.7	62.8	25.7	69.0	68.9	57.7	79.2	66.4	31.0	73.4	74.4
Baseline (no coupling losses)	52.0	77.7	60.6	22.8	68.3	69.8	56.1	79.2	64.8	27.2	73.2	75.1
Ours (no coupling losses)	51.0	76.6	59.5	20.6	67.8	69.5	55.7	78.8	64.3	26.5	72.8	75.2

We visualize the predicted classification maps from each ablation study for an example test sample in Fig. S1. Both for the U-Net16 and DeepLab101 backbones, the (full) method yields more regular classification maps with sharper corners compared to (no field). Additionally, only learning the frame field with (no coupling losses) is insufficient, as can be seen in Fig. S1d.

Table S2: Ablation studies on the CrowdAI dataset [19] with the *DeepLab101* backbone. Ablation studies are: (full) method, (no field) and (no coupling losses). Both polygonization methods have a small simplification tolerance of 1 pixel.

Method	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	AR	AR_{50}	AR_{75}	AR_S	AR_M	AR_L
Mask (full)	54.9	78.1	64.9	25.6	71.2	76.8	58.7	79.8	68.1	29.5	75.8	81.6
Baseline (full)	52.7	78.0	62.4	22.7	69.3	76.0	56.8	79.1	66.0	26.6	74.5	80.8
Ours (full)	51.9	77.1	61.2	21.3	68.6	75.8	56.3	78.8	65.5	25.9	73.9	80.7
Mask (no field)	52.8	75.2	61.8	26.1	67.7	75.0	57.8	78.4	66.7	30.3	73.7	81.8
Baseline (no field)	53.7	79.1	63.1	24.2	70.0	79.3	57.3	80.1	66.5	27.8	74.5	83.4

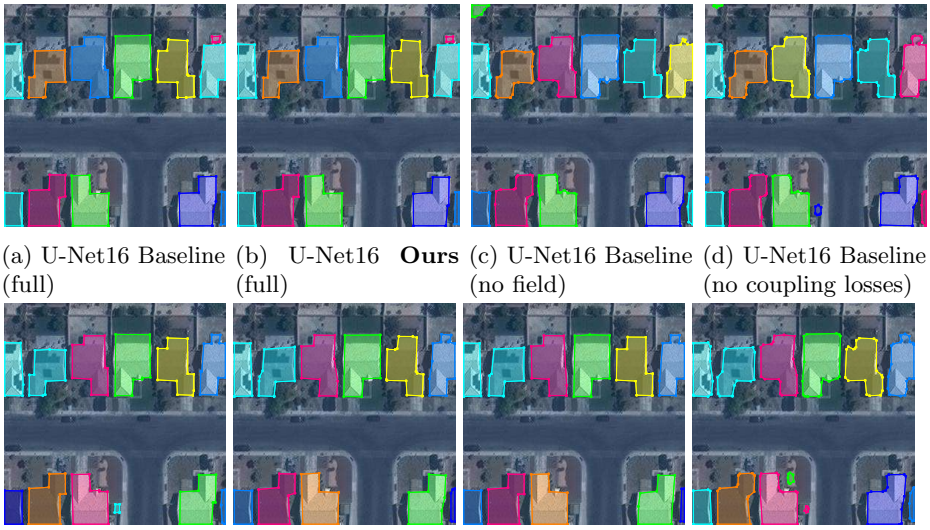
We compare the simple baseline to our polygonization (when the frame field is computed) in Fig. S2. The simple baseline polygonization with no frame field or coupling losses (Fig. S2c, S2d and S2h) gives the worst results with smoothed out corners. The baseline polygonization applied to the classification maps learned alongside a frame field gives much better results (Fig. S2a and S2f). Additionally by using our polygonization method on classification maps learned without coupling losses, we are able to correct the lack of sharp corners in the classification maps (Fig. S2e).



(a) Input (b) U-Net16 Mask (full) (c) U-Net16 Mask (no field) (d) U-Net16 Mask (no coupling losses)

(e) DeepLab101 Mask (full) (f) DeepLab101 Mask (no field)

Fig. S1: Classification predictions on a test sample for all training ablation studies.



(a) U-Net16 Baseline (full) (b) U-Net16 **Ours** (full) (c) U-Net16 Baseline (no field) (d) U-Net16 Baseline (no coupling losses)

(e) U-Net16 **Ours** (no coupling losses) (f) DeepLab101 Baseline (full) (g) DeepLab101 **Ours** (full) (h) DeepLab101 Baseline (no field)

Fig. S2: Extracted polygons on a test sample for all ablation studies including using the simple polygonization (Baseline) versus our polygonization method (**Ours**). For both polygonization, a tolerance of 1 pixel was chosen.

2 Inria dataset

We show an additional result on an a larger image from the Inria dataset in Fig. S3. Our method successfully handles complex buildings. Though we can observe some misclassifications, our method requires a much simpler training procedure (in contrast to, e.g., [**?**] for example, which trains a U-Net variant). We compare to the baseline polygonization method with no frame field learning in Fig. S4.

3 Private dataset of satellite images

We also validate our method on a private large-scale dataset of optical satellite images. The images in this dataset were acquired using three types of satellites (Pleiades, WorldView, and GeoEye) over different types of cities (dense, industrial, residential areas, and city centers). We uniformized the image sampling at 50 cm/pixel spatial resolution, with 3-band RGB images. 57 images of 30 cities across 5 continents are present in the training dataset. The size of images vary from around 2000×2000 pixels to 20000×20000 pixels. The total dataset covers an area spanning around 700 sq. km.. The building outline polygons were manually labeled.

Satellite images are more challenging than aerial images (such as the CrowdAI and Inria images) because they are less clear due to atmospheric effects. This dataset also contains much more varied images compared to CrowdAI and Inria, making up for its smaller size. We preprocess the images by splitting them into smaller 512×512 pixel patches. We then keep 90% of patches for training and 10% for validation. We train two models: U-Net16 (full) and U-Net16 (no field) until validation loss converges (around 1500 epochs). All other hyperparameters are exactly the same as in all previous experiments.

We show results on four test images from different cities not present in the training dataset, a challenging test of generalisation capabilities, in Fig. S5-S12. For each test image, we show our method followed by the baseline. The frame field regularization effect is more pronounced on these satellite images.

Fig. S3: U-Net16 **Ours** (full): our full method on a crop of the vienna36 test image from the Inria dataset with a simplification tolerance of 1 pixel.

Fig. S4: U-Net16 baseline (no field): the baseline polygonization method with no frame field learning on a crop of the vienna36 test image from the Inria dataset with a simplification tolerance of 1 pixel.

Fig. S5: U-Net16 **Ours** (full): our full method on the "Egypt" test image from the private dataset with a simplification tolerance of 1 pixel.

Fig. S6: U-Net16 Baseline (no field): the simple polygonization method with no frame field learning on the "Egypt" test image from the private dataset with a simplification tolerance of 1 pixel.

Fig. S7: U-Net16 **Ours** (full): our full method on the "Bangkok" test image from the private dataset with a simplification tolerance of 1 pixel.

Fig. S8: U-Net16 Baseline (no field): the simple polygonization method with no frame field learning on the "Bangkok" test image from the private dataset with a simplification tolerance of 1 pixel.

Fig. S9: U-Net16 **Ours** (full): our full method on the "Chile" test image from the private dataset with a simplification tolerance of 1 pixel.

Fig. S10: U-Net16 Baseline (no field): the simple polygonization method with no frame field learning on the "Chile" test image from the private dataset with a simplification tolerance of 1 pixel.

Fig. S11: U-Net16 **Ours** (full): our full method on the "PortHarcourt" test image from the private dataset with a simplification tolerance of 1 pixel.

Fig. S12: U-Net16 Baseline (no field): the simple polygonization method with no frame field learning on the "PortHarcourt" test image from the private dataset with a simplification tolerance of 1 pixel.