Building Footprint Regularization : From Vectorization to Deep Learning

This manuscript (permalink) was automatically generated from kshitijrajsharma/building-regularization-research@aa6e150 on June 7, 2025.

Authors

- **Kshitij Raj Sharma**
  0000-0002-2123-3917 · kshitijrajsharma · @kshitijrajsharma@mastodon.social
  Department of Geoinformatics, Paris Lodron University, Salzburg, Austria

✉ — Correspondence possible via GitHub Issues

Abstract

Geographic information systems (GIS) and cartographic applications typically require building footprints as precise vector polygons, rather than raster masks [1]. Building footprint regularization refers to the process of refining raw building outlines (e.g. from remotely sensed imagery or LiDAR) into clean polygon shapes that conform to expected geometric constraints (such as orthogonal corners or aligned edges). The goal is to eliminate irregular artifacts (noisy jags, misalignments) while preserving the true shape, so that the footprints are cartographically suitable for maps. A robust approach to obtain buildings in vector format is to first predict raster buildings using a neural network and then applying postprocessing that outputs polygons. The results achieved by conventional methods are either limited in terms of generalization capacity or are not restricted sufficiently to prior knowledge of regularity [2].This review traces the evolution of footprint regularization methods from early vectorization algorithms in the 1990s through modern deep learning approaches in the 2020s. We focus on 2D footprint outline techniques (planimetric building outlines) and exclude full 3D building reconstruction or roof modeling. Key developments and representative methods are discussed for each era, highlighting their algorithms, use cases, strengths, and limitations. We then compare traditional versus deep learning-based methods in terms of performance, flexibility, accuracy, and integration into GIS workflows. The review draws on peer-reviewed research and real-world implementations (including open-source tools and commercial pipelines) to provide a comprehensive perspective for remote sensing and GIS professionals.

Geometric and Heuristic Methods (1990s - 2000s)

Edge Detection and Line Fitting: Early building extraction in the 1990s relied on low-level image processing and geometric heuristics. For example, Huertas and Nevatia (1988) developed a system to detect buildings in aerial images by finding rectangular clusters of edges (lines) and using shadow cues to distinguish buildings from other structures [3] . Building polygons often consist of jagged lines. Guercke and Sester [4] use Hough-Transformation (Mathematically formalized by Duda, R.O., & Hart, P.E. (1972)[5]) to refine such polygons.

Those approach and similar ones could identify simple rectangular building footprints, but often produced polygons with jagged (bearing in mind they don't take into account the building shape itself rather the outline), noisy outlines. To clean such outlines, researchers applied line simplification algorithms from cartography, notably the Ramer–Douglas–Peucker algorithm : to remove small zig-zags and reduce vertex count while approximating the shape (which is still used to the date) [6/].

The Douglas–Peucker algorithm (originally from 1973) [7] became a common post-processing step to "compress" or simplify building polygon geometry.

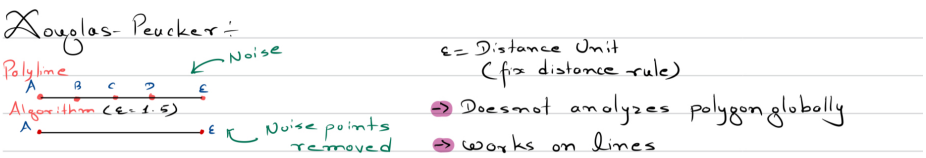

Figure 1: A simple illustration of Douglas-Peucker algorithm

Overall, early methods were largely rule-based: edges and corners were detected via image filters, and building shapes were assembled by connecting these primitives under geometric constraints defined by human experts.

Regularization via Hough Transform: By the 2000s, more sophisticated heuristics were introduced to enforce regularity in building outlines. A prominent tool was the Hough Transform for line detection. Hough transform is a feature extraction method used in image analysis. Hough transform can be used to isolate features of any regular curve like lines, circles, ellipses, etc. Hough transform in its simplest from can be used to detect straight lines in an image.[8/] For instance, Guercke and Sester (2011) [9] proposed a footprint simplification method that takes an initial digitized outline (which might be jagged) and uses a Hough Transform to identify the dominant line orientations; close-to-collinear segments are merged and adjusted by least-squares to align with those dominant directions [3].

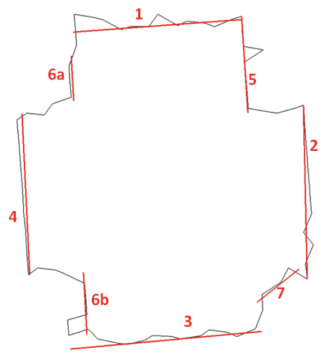

Figure 2: Initial hough transofrmation line segment explained by Guercke and Sester (2011)

The result is a cleaner, rectilinear footprint where spurious bends are straightened and most angles are ~90° or 180° [2]. Shiyong Cui et al. (2012) similarly applied the Hough transform to grouping line segments into two perpendicular families corresponding to a building's principal directions . They constructed an initial graph of line segments, pruned edges that lacked image contrast (assuming they were false boundaries), and then detected closed cycles in the graph to form building polygons [2].

This yielded neatly rectangular footprints for buildings aligned to the two main axes, although the method was inherently limited to rectilinear structures. Tian and Reinartz (2013) extended the idea to allow two arbitrary dominant orientations (not necessarily parallel/perpendicular to the image axes), enabling footprints with an oblique alignment (e.g. buildings rotated on the ground) [2].

These Hough-based methods exemplify how prior knowledge of building shape (e.g. most buildings have parallel opposite walls and right-angle corners) was hard-coded into algorithms well before machine learning became common. The advantage was that the output polygons were regular by design : straight lines, right or consistent angles; making them immediately usable for mapping. However, the success of these methods depended on reliable low-level edge detection. In practice, missing or spurious line segments could cause incomplete or incorrect polygons. Methods like Cui's required a clear dominance of two perpendicular directions; complex or curved buildings, or those with more than two prevailing orientations, fell outside their scope. Hough transform is considered as a computational complex in terms of algorithm itself & often require postprocessing techniques like snapping/merging lines or form cycles to create valid polygons[8]

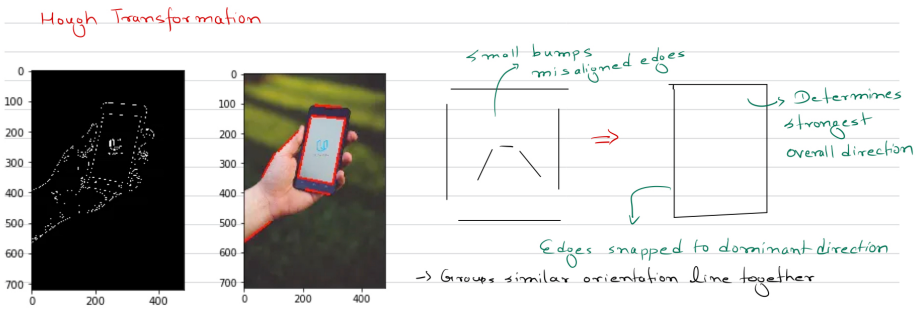

Figure 3: A simple Hough transformation explaination

Model-Based Fitting and Constraints: Beyond Hough transforms, researchers explored explicit shape fitting. Zebedin et al. (2008) introduced an approach to reconstruct building footprints by first detecting numerous line segments and then filtering and clustering these lines by orientation. Here initial lines are filtered by forming a histogram of orientation and then removing outliers. The filtered line directions are used to reconstruct the building with regular appearance. This approach is flexible, as it is not restricted to 90° angles.[2].

This flexibility to allow non-90° angles was a strength like the footprint could, in principle, follow a building that isn't perfectly orthogonal but it still assumed buildings have a limited set of principal directions (which may not hold for very irregular architectures).

Other methods employed *snakes/active contours* and energy minimization to refine building shapes. For example, Fazan and Dal Poz (2013) applied an active contour model (snakes) to building roof images, optimizing an energy that favored straight edges and right-angle corners. While this improved initial detections, A drawback of the proposed method is that the weighting functions favor right angles and therefore only work for buildings with simple rectangular shapes [10].

He et al. (2014) combined data-driven edge detection with a global regularization step: they used an alpha shape algorithm to get an initial footprint from LiDAR point data, then a variant of Douglas–Peucker that was formulated as an energy minimization focusing on polygon complexity (number of vertices). The output was further processed in two modes one maximizing geometric accuracy, another maximizing topological simplicity to balance detail vs. regularity[10].

Energy Formulation : (Basically way to formulate errors on those lines detected) $E = \alpha Edist + \beta Eangle + \gamma Elength$

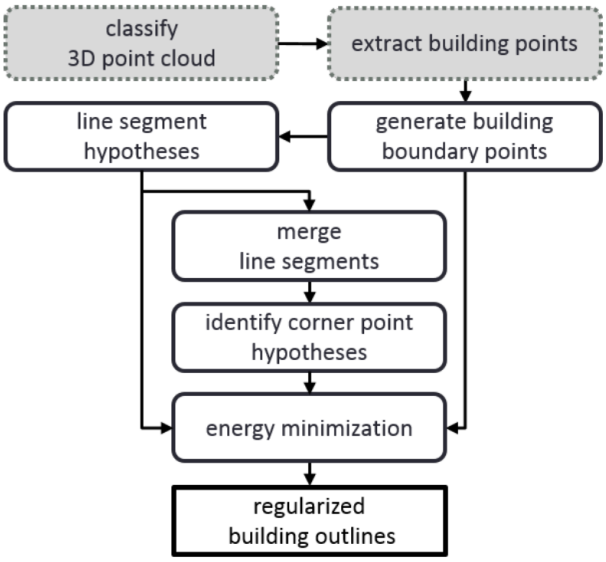

Figure 4: Workflow of building regularization using energy formulation by Albers (2016)

These model-fitting approaches introduced the idea of globally optimizing a footprint shape (e.g., via dynamic programming or least-squares) to satisfy regularity constraints.

Strengths and Limitations: Traditional methods were mostly computationally lightweight and interpretable. They often ran in a couple of sequential steps (edge detection, line grouping, polygon formation) and could be tuned by adjusting thresholds (for line length, angle tolerance, etc.) When assumptions held e.g., a building was clearly rectangular and image data was clean ,these methods

produced very clean footprints. For instance, a study by Guercke & Sester showed that applying Hough-based regularization removed minor zig-zag artifacts and yielded impressively straight building edges.

However, these approaches struggled as building shapes grew more complex or data quality worsened. Irregular or curved buildings (round towers, L- or T-shaped footprints, etc.) did not fit neatly into a two-orientation assumption or a single rectangle model. Many algorithms were fragile: failing to detect a single key edge could cause entire sides of a polygon to be missed. They were also scenario-specific often tailored to isolated buildings with simple roofs and would require retuning for different environments or data sources. It is often said that while such classical methods work in some cases, they are “not applicable to many complex building structures” and they rely heavily on human-engineered features and parameters [3].

In summary, the pre-2010s state-of-the-art could produce “regular” building outlines under favorable conditions, but lacked the robustness and generality needed for broad, automated mapping tasks. These limitations set the stage for machine learning, which promised to learn building shape patterns directly from data and reduce the need for ad hoc rules.

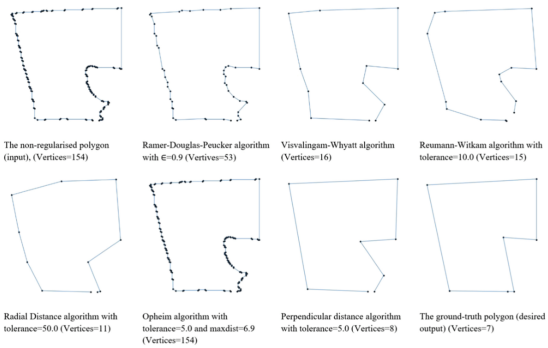

Figure 5: A comparison of traditional regularization algorithms on a noisy polygon in terms of node reduction, shape simplification, and edge smoothness [11]

Learning-Based Methods (2010s)

By the mid-2010s, the rise of deep learning fundamentally changed how building footprints were extracted. Instead of manually defining edges and shape rules, researchers began training convolutional neural networks (CNNs) to recognize buildings and output them in raster or vector form. The typical pipeline circa 2015–2017 was to use a semantic segmentation network (such as U-Net or DeepLab) to produce a binary mask of building pixels, then apply a vectorization algorithm to convert that mask into polygons [6].

This two-step approach : **CNN segmentation followed by geometric post-processing** was a direct evolution of earlier workflows, swapping out hand-coded image filters for learned CNN features. For example, Marmanis et al. (2016) and Maggiori et al. (2017) mentioned that fully convolutional networks could outperform traditional techniques in detecting building regions from aerial images [3].

Once a clean building mask was obtained, off-the-shelf polygonization (e.g., marching squares to trace outlines) and Douglas–Peucker simplification would yield a polygon vector. A problem with this approach is that semantic segmentation models are unable to delineate the boundaries between objects of the same class. This means that a single polygon will be drawn around a group of buildings that share walls, such as a block of rowhouses. To handle this case, the semantic segmentation model can be replaced with an instance segmentation model such as **Mask R-CNN**. This model generates a separate raster mask for each instance of a class that is detected [6]. Beyong which additional

smoothing or regularization was needed, and many practitioners continued to apply tolerance-based simplification or mild “squaring” adjustments to make the polygons map-ready.

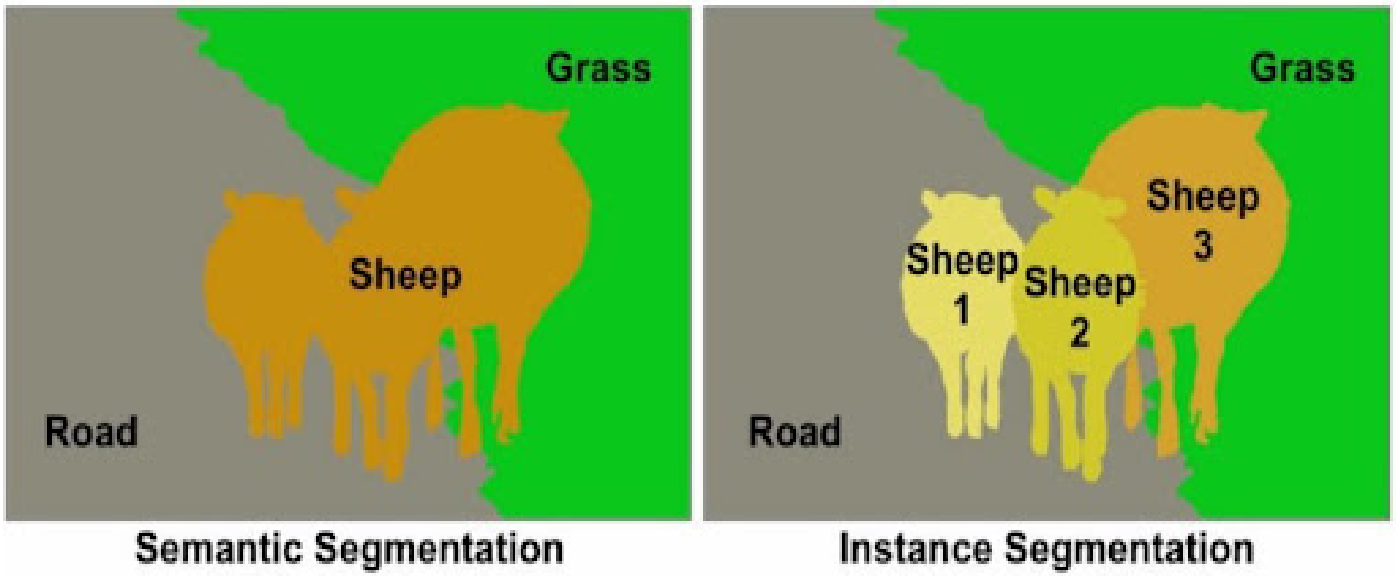

Figure 6: Semantic Segmentation to Instance Segmentation Aprooaches , source

Deep Structured Models (Active Contours)

A significant development in bridging classical regularization and deep learning was the integration of active contour models into neural networks. Marcos et al. (2018) introduced Deep Structured Active Contours (DSAC), a hybrid approach where a CNN learns to predict the parameters of an active contour that locks onto building edges . In their framework, the network output is not a raster, but rather coefficients that define the shape and tension of an active contour (snake) which then deforms to fit the building boundary.

Gur et al. (2019) extended this concept by iteratively updating a polygon outline in an end-to-end trainable manner. Their pipeline starts with an approximate polygon (like a coarse outline of the building) and uses a neural network to repeatedly adjust the vertices, analogous to how one would iteratively relax an active contour. While effective, the polygons produced by Gur et al. were not explicitly enforced to be rectilinear the focus was on aligning to image evidence, not necessarily making right angles [2].

Hatamizadeh et al. (2020) proposed a multi-building active contour model: a CNN first predicts initial contours for many buildings in a scene, and then a learned energy function refines all of them simultaneously [2]. This allowed processing dense urban scenes with many buildings at once, something earlier active-contour methods (which often assumed one building at a time) didn’t handle. Hatamizadeh’s model was end-to-end (it directly outputs vector polygons from an image), but like its predecessors, its regularization was implicit it preferred smooth, compact shapes but did not guarantee, say, all angles = 90°.

Source Code : DSAC , ACDRNet, DALS

Recurrent Vertex Prediction (Polygon RNNs) : PolyMapper

Instead of converting segmentation masks into polygons as a post-processing step, Recurrent Vertex Prediction models approach polygon extraction as a sequence prediction problem. In this framework, the model outputs a series of vertices one at a time, similar to writing out coordinate lists.

Polygon-RNN, first introduced by Castrejon et al. (2017) and later improved by Acuna et al. (2018), demonstrated that a recurrent neural network (RNN) could learn to draw object outlines by sequentially predicting polygon vertices, using image features to guide the process at each step.

A notable extension of this idea is PolyMapper, which integrates convolutional and recurrent modules in an end-to-end architecture. First, a CNN component (similar to Mask R-CNN) detects building instances and predicts coarse masks along with boundary and corner probability maps. Next, the most likely vertices from the corner map are selected and passed, together with image features, into an LSTM-based recurrent module. This module outputs vertices in sequence to trace the building outline, stopping when an end-of-sequence token is predicted, which signals the polygon should close.

Figure 7: Comparison of output generated by instance segmentation and Polymapper , source

This approach has distinct advantages: the RNN can learn to skip over minor irregularities, resulting in cleaner and simpler polygons with fewer vertices. It can also learn to favor structural regularities, such as right angles, due to its exposure to training data. PolyMapper demonstrated that such models produce more regular and human-like building footprints than traditional instance segmentation pipelines.

However, this modeling approach brings complexity. The network must learn when to terminate the sequence (when to stop adding vertexes), and the loss function must account for sequence prediction dynamics. Early Polygon-RNN models also faced issues such as generating self-intersecting polygons or incorrectly ordering vertices unless constraints were explicitly enforced.

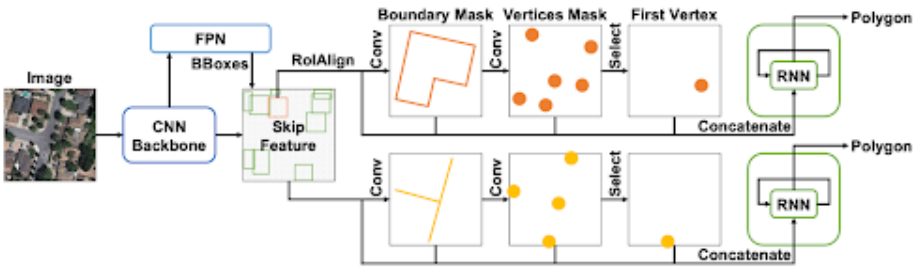

Figure 8: Overview of PolyMapper for Building and Roads : Source

By the end of the 2010s, two main deep learning approaches emerged for extracting building footprints from imagery:

1. **Segmentation-based methods** focused on generating accurate masks of buildings and then used advanced post-processing techniques such as learned active contours (“snakes”) to clean and regularize the shapes.

2. **Direct polygon prediction methods** aimed to output building outlines directly as sequences of vertices and edges, using models like recurrent neural networks (RNNs) or parameterized shape representations.

These approaches marked a significant improvement over older heuristic techniques. Convolutional neural networks (CNNs) could generalize better across diverse geographies and imaging conditions. For instance, a model trained on buildings in one city could often perform reasonably well in another, whereas hand-tuned algorithms often failed when conditions changed.

Despite this progress, early deep learning models still had limitations. The building shapes they produced were often *almost* clean but not perfectly geometric for example, a nearly straight wall might still have a slight jitter in the predicted vertices. This lack of geometric precision posed challenges for GIS applications that require clean vector shapes.

Source Code : NA

Modern Deep Learning Approaches (2020s)

Polygonal Building Segmentation by Frame Field Learning

In addition to direct polygon prediction, researchers also explored ways to inject geometric structure into the deep learning process. One notable approach by Girard et al. (2021) [12] involved predicting not only a segmentation mask for buildings, but also a frame field : a set of orthogonal vectors at each pixel along the boundary indicating local edge directions.

A **frame field** acts like a directional map around a building’s edges: it shows which way walls run and where corners should be. Using this directional information, the method first extracts a rough outline from the mask and then **snaps and refines it** by aligning it with the frame field and detected corner points. The post-processing pipeline includes multiple geometric steps such as skeletonization, corner detection, and line simplification each algorithmically defined rather than learned.

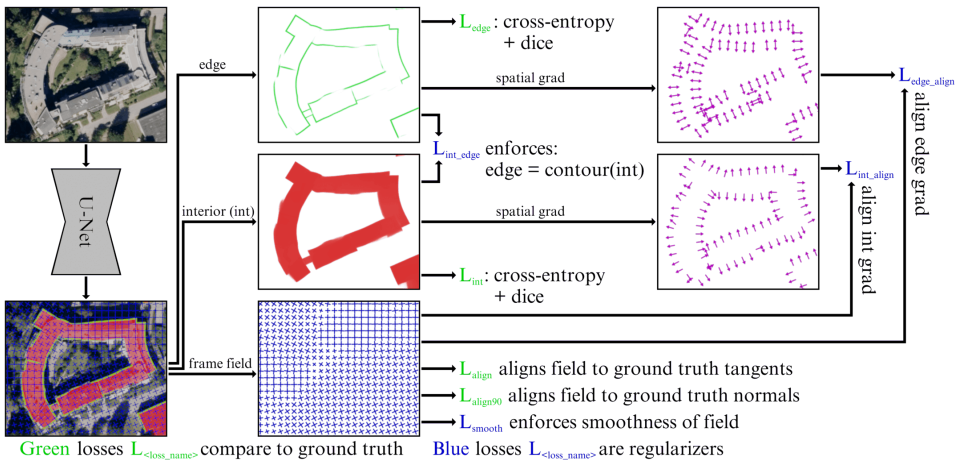

Figure 9: Explaination of framefield : Source

This method is able to handle buildings that are touching and buildings with courtyards by explicitly representing shared walls and generating polygons with holes. In addition, it runs about 10x faster

than PolyMapper at inference time. The downside of this method is that the polygon extraction routine is complex and lacks the elegance of a model trained end-to-end.[6/]

In the last few years, deep learning models for building footprint regularization have reached new levels of maturity. These models are characterized by end-to-end training (the network learns to output a final polygon with minimal post-processing) and by the integration of architectural elements that explicitly handle the polygon's structure (such as graph neural networks, transformers, or differentiable geometric algorithms). Below we highlight several state-of-the-art approaches, including CNN/RNN hybrids, graph-based models, and transformer-based models, and discuss how they improve upon prior methods.

Source Code : GitHub

PolyWorld: End-to-End Polygon Extraction via CNN and GNN

PolyWorld [13] introduces a novel end-to-end deep learning architecture for extracting vector building footprints directly from satellite imagery. Unlike earlier methods such as Polygon-RNN or PolyMapper, which rely on sequential vertex prediction or post-processing of segmentation masks, PolyWorld formulates the problem as a graph-based polygon matching task.

The pipeline involves three main stages:

1. **Vertex Detection**: A fully convolutional neural network outputs a vertex confidence map from which likely building corners are identified. Each vertex is paired with a learned visual descriptor encoding local image features.

2. **Graph-Based Learning**: Detected vertices are embedded in a fully connected graph. An attentional Graph Neural Network (GNN) evaluates pairwise relationships between vertices to learn "connection strengths" i.e., the likelihood that a pair of vertices should be connected by an edge.

3. **Polygon Assembly via Differentiable Matching**: The final polygon structure is determined by solving a graph matching problem, formulated as an optimal cycle through the vertices. This is achieved using a differentiable relaxation of the Hungarian algorithm (Sinkhorn algorithm), enabling gradient-based learning.

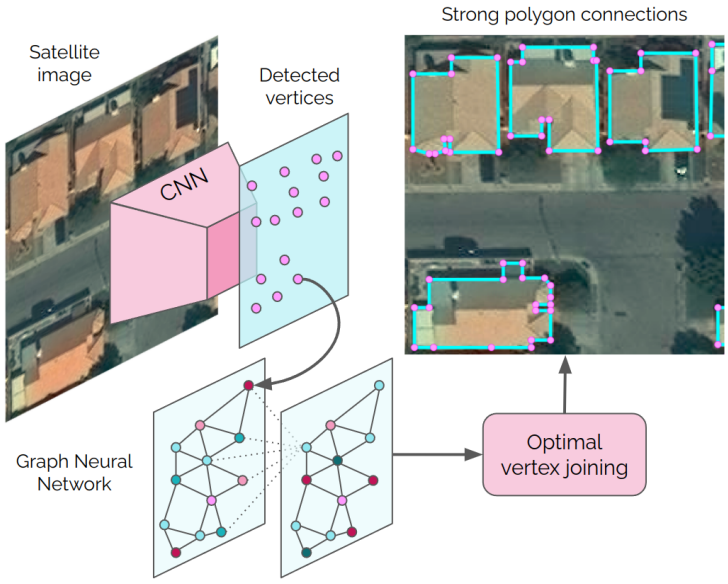

Figure 10: Explanation of how PolyWorld works: source

Despite having better performance than the frame fields models on the CrowdAI dataset, PolyWold does not have the ability to generate polygons with holes, or handle buildings with shared walls. However, the authors offer some ideas for how the model could be modified to handle these cases. The model and inference (but not the training) source code is open source, but has a restrictive license that only permits its use for research.[6/]

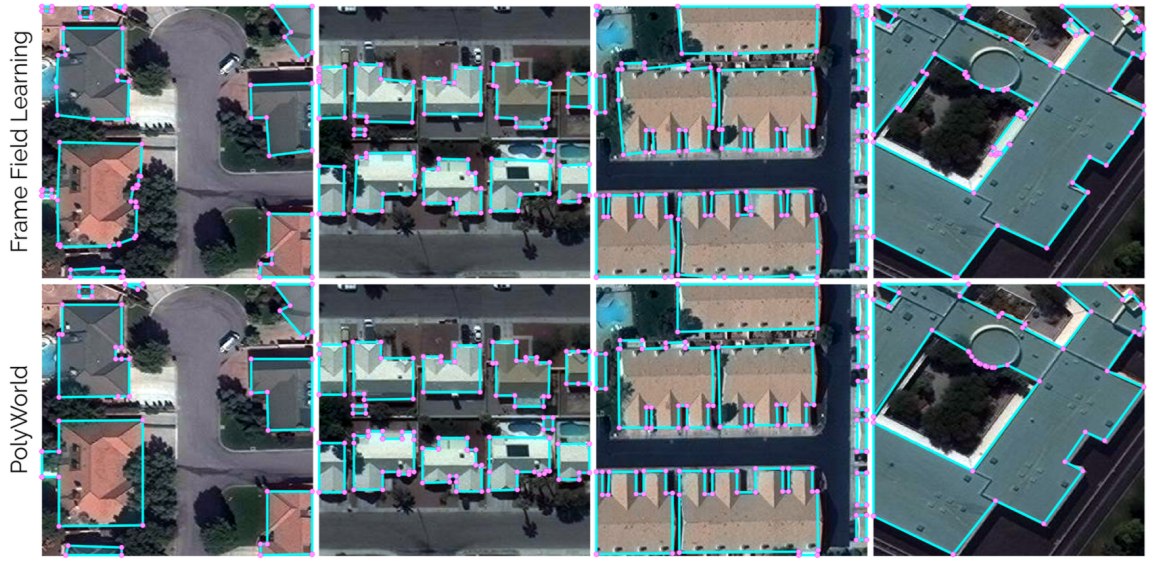

Figure 11: PolyWorld vs Frame Field Learning on CrowdAI test dataset : source

Method	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	AR	AR_{50}	AR_{75}	AR_S	AR_M	AR_L
Mask R-CNN [13]	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 [21]	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 [16]	55.7	86.0	65.1	30.7	68.5	58.4	62.1	88.6	71.4	39.4	75.6	75.4
FFL (no field), mask	57.8	84.0	66.9	33.8	74.1	80.7	67.0	90.4	76.9	46.2	79.7	85.7
FFL (no field), simple poly	61.1	87.4	71.2	35.1	74.5	82.3	64.7	89.4	74.1	41.7	77.9	85.7
FFL (with field), mask	57.7	83.8	66.3	33.8	73.8	81.0	68.1	91.0	77.7	47.5	80.0	86.7
FFL (with field), simple poly	61.7	87.6	**71.4**	35.7	74.9	83.0	65.4	89.8	74.6	42.5	78.6	85.8
FFL (with field), ACM poly [10]	61.3	87.4	70.6	33.9	75.1	83.1	64.9	89.4	73.9	41.2	78.7	85.9
PolyWorld (offset off)	58.7	86.9	64.5	31.8	80.1	85.9	71.7	92.6	79.9	47.4	85.7	94.0
PolyWorld (offset on)	**63.3**	**88.6**	70.5	**37.2**	**83.6**	**87.7**	**75.4**	**93.5**	**83.1**	**52.5**	**88.7**	**95.2**

Figure 12: Comaprison results oon CrowdAI test dataset by PolyWorld

Figure represents MS COCO results on the CrowdAI test dataset for all the building extraction and polygonization experiments. The results of PolyWorld are calculated discarding the correction offsets (offset off), and refining the vertex positions (offset on). FFL refers to the Frame Field Learning method. The results are computed with and without frame field estimation. “mask” refers to the pure segmentation produced by the model. “simple poly” refers to the Douglas–Peucker polygon simplification, and “ACM poly” refers to the Active Contour Model polygonization method [13]

Metric	Meaning
AP	Average Precision (overall) – higher is better. General performance measure combining precision and recall.
AP50	AP at IoU threshold 0.5 – more lenient match condition.
AP75	AP at IoU threshold 0.75 – stricter match condition.
APS / APM / APL	AP for small, medium, and large buildings, respectively.
AR	Average Recall (overall) – measures how well true objects are detected.
AR50 / AR75	AR at IoU thresholds 0.5 and 0.75.

Metric	Meaning
ARS / ARM / ARL	AR for small, medium, and large objects, respectively.

Source Code : GitHub

Improved version , Re:PolyWorld (2023)

Following PolyWorld, Zorzi and Fraundorfer (2023)[14] introduced Re:PolyWorld, which is claimed to be an improved multi-stage version of the framework . Re:PolyWorld added a second refinement stage where an initial polygon prediction is further optimized and made even more regular by an additional GNN module.

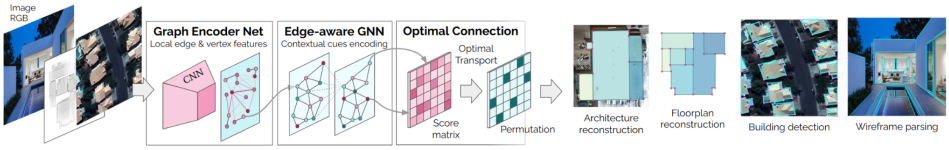

Figure 13: Re:PolyWorld Methodology

With these enhancements, Re:PolyWorld achieved new state-of-the-art scores on the CrowdAI dataset, improving both the precision and the shape quality of footprints. For example, it improved the mean intersection-over-union (IoU) and corner angle error metrics beyond what PolyWorld and a strong frame-field baseline had achieved.

Method	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	AR	AR_{50}	AR_{75}	AR_S	AR_M	AR_L
Mask R-CNN [7]	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 [10]	55.7	86.0	65.1	30.7	68.5	58.4	62.1	88.6	71.4	39.4	75.6	75.4
FFL, simple poly [6]	61.7	87.6	71.4	35.7	74.9	83.0	65.4	89.8	74.6	42.5	78.6	85.8
FFL, ACM poly [6]	61.3	87.4	70.6	33.9	75.1	83.1	64.9	89.4	73.9	41.2	78.7	85.9
PolyWorld [30]	63.3	88.6	70.5	37.2	83.6	87.7	75.4	93.5	83.1	52.5	88.7	95.2
Re:PolyWorld	**67.2**	**89.8**	**75.8**	**42.9**	**85.3**	**89.4**	**78.6**	**94.1**	**86.7**	**58.3**	**90.3**	**96.2**

Figure 14: Benchmark dataset of Re:PolyWorld

he continued success of these GNN-based methods demonstrates the value of treating polygon formation as a graph problem (where deep networks ensure the graph forms nice cycles with desired properties) rather than a pixel-by-pixel segmentation problem

Transformer-Based Sequence Models : Pix2Poly

Very recently, researchers have applied transformers the sequence modeling architecture behind advances in NLP to polygon extraction. Pix2Poly [15] is an attention-based model that casts building footprint delineation as a sequence prediction problem, handled entirely by a transformer encoder-decoder.

The key idea is to avoid the multi-step detour that graph-based models take (e.g., detect vertices → match into polygon). Instead, Pix2Poly's transformer directly outputs an ordered list of vertex coordinates in sequence, one vertex after another, in a single forward pass. To do this, it discretizes continuous image coordinates into a sequence of tokens (similar to how one might tokenize words or subwords in language) and trains the network to emit the token sequence corresponding to the building outline.

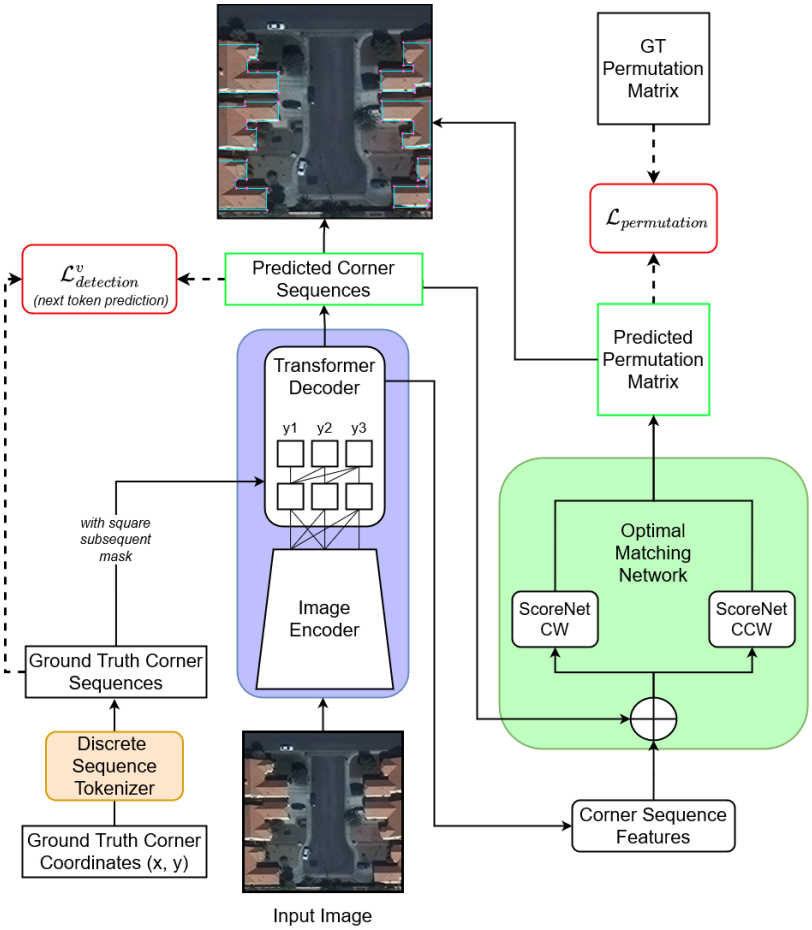

Figure 15: Overview of Pix2Poly Architecture

Because the transformer's self-attention can attend globally to the image, Pix2Poly can, in theory, capture the global shape of the building while placing each vertex. The authors highlight that it avoids certain bottlenecks of earlier methods: for example, it doesn't require a non-maxima suppression step to select vertices (which was non-differentiable in many prior pipelines), nor does it need a separate graph matching module, since the sequence inherently encodes the connectivity.

The entire model is differentiable end-to-end, making training more straightforward and cohesive. In their experiments, Pix2Poly achieved state-of-the-art results not only for building footprints but also for road network extraction, indicating the versatility of the approach.

Essentially, Pix2Poly represents the convergence of transformer-based detection with graph learning: it uses a transformer as a "vertex sequence detector" and still incorporates an optimal matching network (similar to PolyWorld's assignment module) to ensure the predicted sequence forms closed polygons. This model claimed to be less complex as compared to FLL , PolyWorld as it has total parameter count of (31.9M) [15]

Figure 16: Example of Pix2poly output

Source Code : Github

Other Noticable Advances

Alongside the above, there have been other notable modern approaches. PolyBuilding (2022) [1] introduced a similar concept of a “polygon transformer” that directly predicts vector representations of buildings. It emphasizes fully end-to-end training and shows that a transformer can outperform CNN+RNN hybrids on benchmark aerial image datasets.

Generative models have also been explored: for instance, RegGAN (2022) [16] used a generative adversarial network to refine building masks such that their boundaries look more like real building shapes. In RegGAN, a generator CNN outputs a building mask and a discriminator network critiques it, especially focusing on boundary regularity. This adversarial training leads to output masks with sharper, straighter edges than a standard segmentation network.

Similarly, another study proposed Poly-GAN (2023) to post-process OpenStreetMap building footprints, adjusting vertices via a GAN to better align and orthogonalize them [11]. These GAN-based approaches can be seen as learned versions of the old heuristic regularization rather than applying a Hough transform, they apply a discriminator that has learned what a “correct” building outline looks like and thus encourages the output to conform to those learned patterns.

Comparison: Traditional vs. Deep Learning Methods

Accuracy and Performance

Aspect	Traditional Methods	Deep Learning Methods
Detection Accuracy	Moderate; struggles with small/faint buildings	High; CNNs and transformers achieve state-of-the-art IoU and recall
Processing Speed	Very fast (per building) on CPU	Slower per image but GPU-accelerated; parallelization possible
Scalability	Needs tuning for new regions	Scales to large areas
Example	Hough Transform, DP simplification	PolyWorld, Pix2Poly, Frame Field Learning

Flexibility and Generalization

Aspect	Traditional Methods	Deep Learning Methods
Adaptability	Manual reprogramming required	Retrainable and fine-tunable on new data
Shape Handling	Biased to rectilinear structures	Learns to detect irregular, curved, or complex forms
Data Sensitivity	Edge-based; poor in low contrast	Learns semantic cues (shadows, context)
Example	Thresholding, edge detectors	CNNs, Transformers trained on diverse imagery

Cartographic Quality

Aspect	Traditional Methods	Deep Learning Methods
Output Regularity	Hard constraints (e.g. 90° angles)	Learned regularity (polygon loss, angle constraints, GANs)
Visual Quality	Very clean, stylized	Near finer details as compared from hand-crafted results
Limitations	May snap overly aggressively	May allow some deviation; occasional noise
Example	Regularize Footprint tool	PolyWorld, Pix2Poly with angle loss, GAN refinement

GIS Integration

Aspect	Traditional Methods	Deep Learning Methods
Output Format	Vector-ready (Polygons)	Historically : Raster masks , Now outputs GeoJSON/Shapefiles directly
Workflow Fit	Compatible with legacy GIS	Integrated in QGIS, ArcGIS Pro via plugins, OpenCV
Post-Processing	Quite sophisticated	Often Minimal with latest pipelines
Example	Manual digitization, vector tools	Microsoft's global footprint pipeline, Google Open Buildings

Quality Control

Aspect	Traditional Methods	Deep Learning Methods
Failure Visibility	Obvious errors, easy to flag	May generate plausible but wrong results
Correction	Manual re-runs or inspection	Hybrid review: DL + regularization + optional human validation
Robustness	Deterministic but brittle , Easy to explain	Robust to noise, generalizes well across geographies, Hard to explain

Conclusion

The transition from traditional methods to deep learning marks a major step forward in footprint extraction. Classical techniques were interpretable and rule-based, but lacked scalability and flexibility. Modern deep learning models like PolyWorld and Pix2Poly are accurate, fast (with GPU), generalize well, and produce GIS-ready vector outputs that rival manual digitization.

Although traditional methods remain valuable for strict cartographic constraints, deep learning has become the default for global-scale mapping. These tools now support highly automated workflows, reducing manual effort and enabling rapid, consistent extraction of building footprints worldwide.

The fusion of geometric logic with learning-based systems has established a new standard: fast, precise, and regularized footprints at scale, not just for research, but for operational, production-ready geospatial data pipelines.

References

1. **PolyWorld: Polygonal Building Extraction With Graph Neural Networks in Satellite Images**
Stefano Zorzi, Shabab Bazrafkan, Stefan Habenschuss, Friedrich Fraundorfer
(2022)
https://openaccess.thecvf.com/content/CVPR2022/html/Zorzi_PolyWorld_Polygonal_Building_Extraction_With_Graph_Neural_Networks_in_Satellite_CVPR_2022_paper.html

2. **Rectilinear Building Footprint Regularization Using Deep Learning**
Philipp Schuegraf, Zhixin Li, Jiaojiao Tian, Jie Shan, Ksenia Bittner
ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences (2024-06-10) https://doi.org/g9k82r
DOI: 10.5194/isprs-annals-x-2-2024-217-2024

3. **Automatic Building Footprint Extraction from Multi-Resolution Remote Sensing Images Using a Hybrid FCN**
Philipp Schuegraf, Ksenia Bittner
ISPRS International Journal of Geo-Information (2019-04-12) https://doi.org/ggwr8s
DOI: 10.3390/ijgi8040191

4. **Building Footprint Simplification Based on Hough Transform and Least Squares Adjustment**
Richard Guercke, Monika Sester
Proceedings of the 14th Workshop of the ICA Commission on Generalisation and Multiple Representation (2011-07)

5. **Use of the Hough transformation to detect lines and curves in pictures**
Richard O Duda, Peter E Hart
Communications of the ACM (1972-01) https://doi.org/b4t7tv
DOI: 10.1145/361237.361242

6. **Automated Building Footprint Extraction (Part 3): Model Architectures • Element 84** (2022-11-09) https://element84.com/software-engineering/automated-building-footprint-extraction-part-3-model-architectures/

7. **Algorithms for the reduction of the number of points required to represent a digitized line or its caricature**
David H Douglas, Thomas K Peucker
The Canadian Cartographer (1973)

8. **Hough Transform**
Surya Teja Karri
Medium (2019-09-27) https://medium.com/@st1739/hough-transform-287b2dac0c70

9. **Aggregation of LoD 1 building models as an optimization problem**
R Guercke, T Götzelmann, C Brenner, M Sester
ISPRS Journal of Photogrammetry and Remote Sensing (2011-03) https://doi.org/fbs43j
DOI: 10.1016/j.isprsjprs.2010.10.006

10. **AUTOMATIC EXTRACTION AND REGULARIZATION OF BUILDING OUTLINES FROM AIRBORNE LIDAR POINT CLOUDS**
Bastian Albers, Martin Kada, Andreas Wichmann
The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences (2016-06-09) https://doi.org/gcc7nx

DOI: 10.5194/isprs-archives-xli-b3-555-2016

11. **Poly-GAN: Regularizing Polygons with Generative Adversarial Networks**
Lasith Niroshan, James D Carswell
Lecture Notes in Computer Science (2023) https://doi.org/g9m2vh
DOI: 10.1007/978-3-031-34612-5_13

12. **Polygonal Building Extraction by Frame Field Learning**
Nicolas Girard, Dmitriy Smirnov, Justin Solomon, Yuliya Tarabalka
2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2021-06)
https://doi.org/gnt53r
DOI: 10.1109/cvpr46437.2021.00583

13. **PolyWorld: Polygonal Building Extraction with Graph Neural Networks in Satellite Images**
Stefano Zorzi, Shabab Bazrafkan, Stefan Habenschuss, Friedrich Fraundorfer
2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2022-06)
https://doi.org/gtmwjv
DOI: 10.1109/cvpr52688.2022.00189

14. **Re:PolyWorld - A Graph Neural Network for Polygonal Scene Parsing**
Stefano Zorzi, Friedrich Fraundorfer
2023 IEEE/CVF International Conference on Computer Vision (ICCV) (2023-10-01)
https://doi.org/g9mrpk
DOI: 10.1109/iccv51070.2023.01537

15. **Pix2Poly: A Sequence Prediction Method for End-to-end Polygonal Building Footprint Extraction from Remote Sensing Imagery** https://arxiv.org/html/2412.07899v1

16. **RegGAN: An End-to-End Network for Building Footprint Generation with Boundary Regularization**
Qingyu Li, Stefano Zorzi, Yilei Shi, Friedrich Fraundorfer, Xiao Xiang Zhu
Remote Sensing (2022-04-11) https://doi.org/gqhkzx
DOI: 10.3390/rs14081835