Building Footprint Regularization : From Vectorization to Deep Learning

This manuscript (permalink) was automatically generated from kshitijrajsharma/building-regularization-research@0a2f786 on May 30, 2025.

Authors

- **Kshitij Raj Sharma**
  0000-0002-2123-3917 · kshitijrajsharma · @krschap@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 (Zebedin et al., 2008; Cui et al., 2012; Tian and Reinartz, 2013) or are not restricted sufficiently to prior knowledge of regularity (Marcos et al., 2018; Gur et al., 2019; Hatamizadehet al., 2020; Zhao et al., 2021; Zorzi and Fraundorfer, 2023) [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)) 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) [5/].

The Douglas–Peucker algorithm (originally from 1973) [6] became a common post-processing step to "compress" or simplify building polygon geometry.

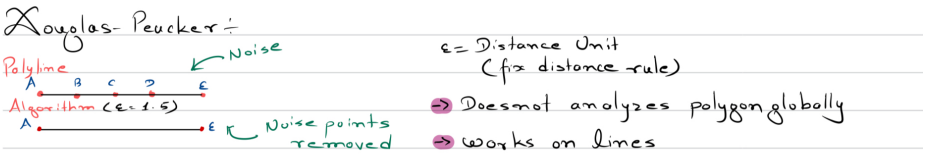

Figure 1: A simple illustration of Douglas-Peucker algorithm

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.[st1739/hough-transform-287b2dac0c70?] For instance, Guercke and Sester (2011) 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 [8].

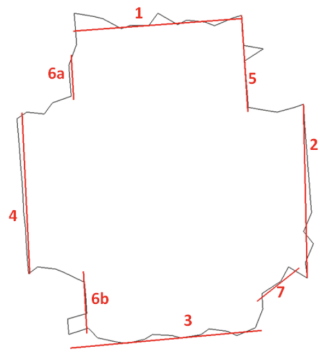

Figure 2: Initial hough transofrmation line segment explained by Guercke and Sester (2011)

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[st1739/hough-transform-287b2dac0c70?]

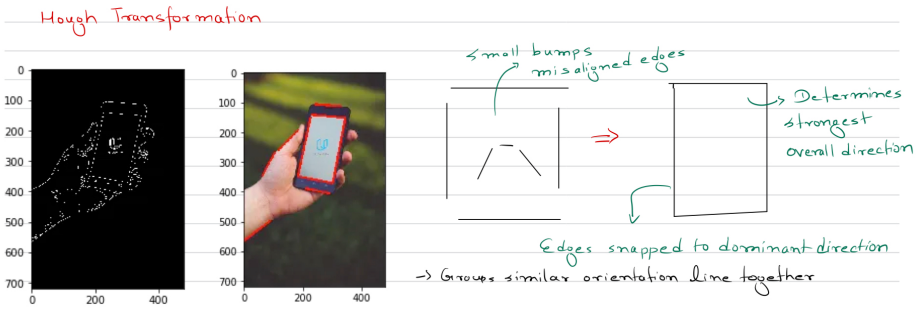

Figure 3: A simple Hough transformation explaination

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 [9].

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[9].

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

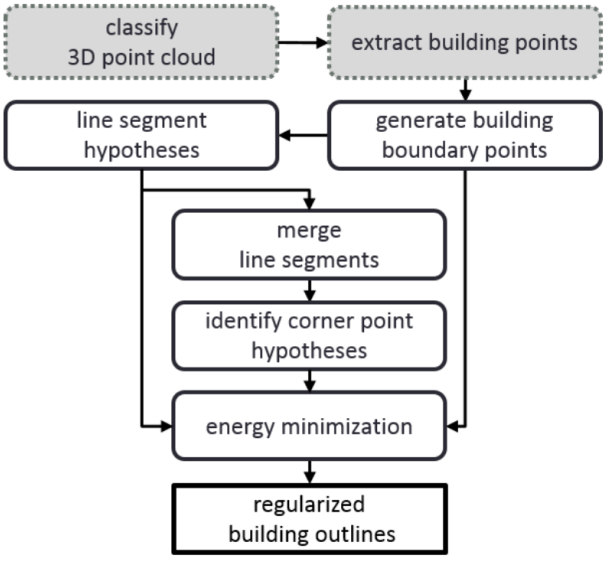

Figure 4: Workflow of building regularization using energy formulation by Albers (2016)

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.

Transition to 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 [5/].

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 [5/]. 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.

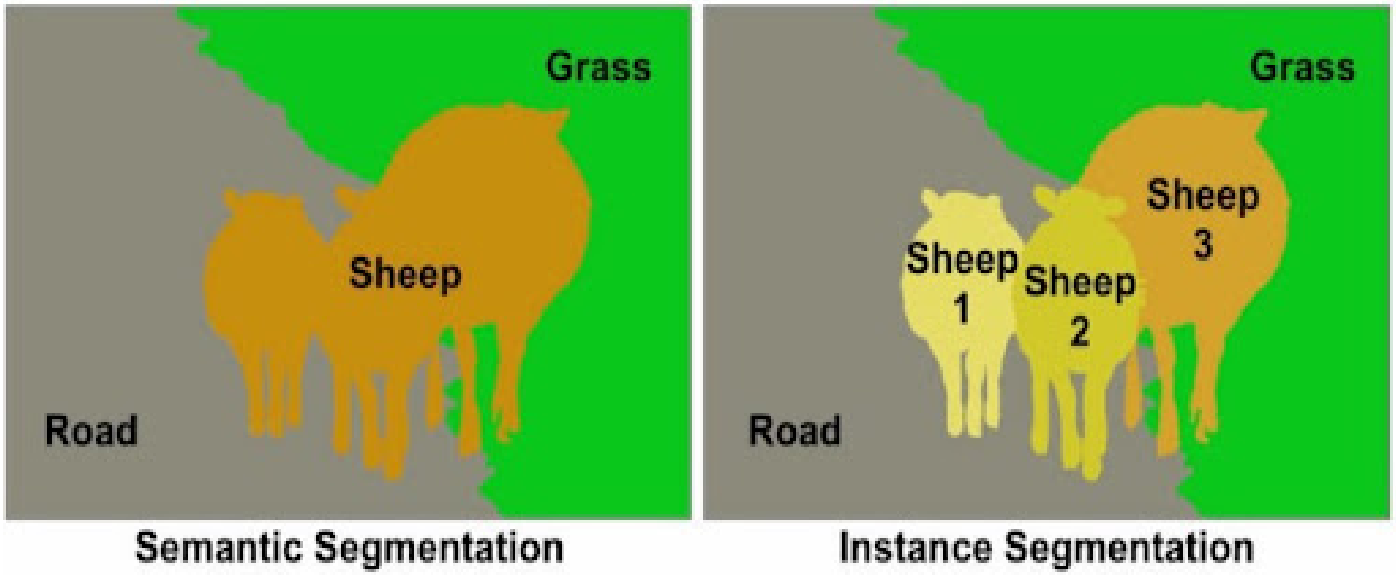

Figure 5: Semantic Segmentation to Instance Segmentation Aprooaches , Source

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°.

Recurrent Vertex Prediction (Polygon RNNs)

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) [10,**doi:10.1109/ECCV.2018.00229?**], 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** [11], 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 6: 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.

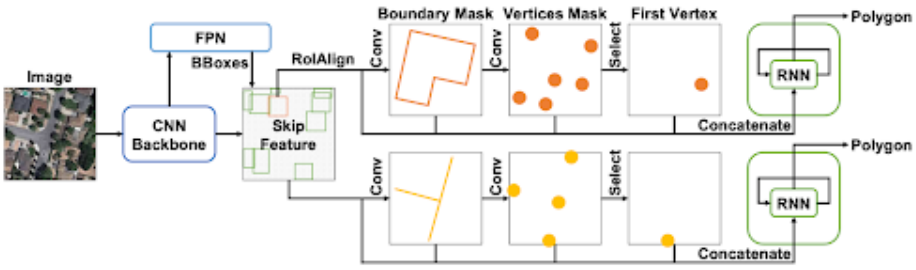

Figure 7: Overview of PolyMapper for Building and Roads : Source

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. **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/

6. **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)

7. **Medium: Read and write stories.**
Medium
https://medium.com

8. https://www.mdpi.com/2220-9964/8/4/191

9. **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

10. **Convex Global 3D Registration with Lagrangian Duality**
Jesus Briales, Javier Gonzalez-Jimenez
2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017-07)
https://doi.org/g9mrks
DOI: 10.1109/cvpr.2017.595

11. **Douglas-Rachford Networks: Learning Both the Image Prior and Data Fidelity Terms for Blind Image Deconvolution**
Raied Aljadaany, Dipan K Pal, Marios Savvides
2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2019-06)
https://doi.org/gk2q6b
DOI: 10.1109/cvpr.2019.01048