The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume L-4/W1-2026
Free and Open Source Software for Geospatial (FOSS4G) 2026 – Academic Track "Bridging Geospatial Technology and Humanity",
30 August–5 September 2026, Hiroshima, Japan

# DINO-EdgeQuery: Edge-First Polygon Decoding for Building Footprint Extraction from Satellite Imagery

Yuji Kobayashi , Yu Chun Lai

Orbitalnet, Inc., Nagoya, Aichi, Japan -
y-kobayashi@orbitalnet.jp, yuchun-lai@orbitalnet.jp

**Keywords:** Building footprint extraction, DINOv3, instance segmentation, polygon decoding, edge primitives, remote sensing.

**Abstract**

Building footprint extraction from high-resolution satellite imagery requires accurate building boundary raster masks and an effective shape reconstruction method to produce natural building footprints. Unlike vertex-centric graph approaches or mask contour tracing, we propose DINO-EdgeQuery, an edge-first paradigm that addresses this need. A frozen DINOv3 backbone, adapter neck, and an instance decoder inspired by Mask2Former provide building-aware boxes and masks, while a region of interest (ROI)-conditioned EdgeQuery decoder predicts edge activity, center, direction, length, and successor relations. By capturing dominant wall structures as edge primitives and reconstructing vertices purely through deterministic line intersections, we deliberately separate neural edge prediction from verifiable geometric assembly. Active edges are ordered by successor scores, and the P2 quality gate removes invalid or contained duplicate polygons. This geometric post-processing encourages sharp, line-intersection-based corners and enforces topology-safe polygons through deterministic validity checks. To improve generalization, we train segmentation and EdgeQuery branches separately, mask inactive losses to zero, and merge the trained weights for unified inference. Validation on the dataset publicly released on the website by the Wuhan University Geospatial Computer Vision Group demonstrates that our EdgeQuery framework, including the P2 quality gate, delivered high-quality, geometrically superior polygon outputs with zero self-intersections and triangle collapses, and simultaneously improved Intersection over Union (IoU) and suppressed duplicate building footprints.

## 1. Introduction

Building footprint extraction from high-resolution satellite imagery is fundamental to remote sensing, urban mapping, cadastral analysis, and geospatial database maintenance. Practical applications require not only raster masks but compact, editable, and topologically valid vector polygons. Despite progress in deep building segmentation, producing annotation-ready polygons remains difficult for dense scenes, small buildings, and footprints with long straight walls and sparse corners.

Existing raster-to-vector methods use mask polygonization, frame-field regularization, vertex connectivity estimation, or boundary-point sequence prediction (Girard et al., 2021; Zorzi et al., 2022; Lazarow et al., 2022). These approaches improve polygonal extraction, but their geometry is often tied to raster contours, variable-size vertex sets, or ordered boundary points. For buildings, the dominant structure is more naturally described by wall-like edge primitives and successor connectivity.

We propose DINO-EdgeQuery as the first stage of an edge-first building-footprint framework. The model uses instance segmentation as a visual prior rather than as final geometry. A frozen DINOv3 (Siméoni et al., 2025) backbone, adapter neck, and Mask2Former (Cheng et al., 2022)-style instance decoder provide building-aware boxes and masks. For each instance, region of interest (ROI)-conditioned tokens are passed to an EdgeQuery decoder that predicts candidate edge primitives, and a successor relation head estimates directed connectivity.

The current implementation does not use a learned neural polygon decoder after edge prediction. Instead, polygon vertices are reconstructed by a deterministic line-intersection assembler with quality gates and containment-based duplicate suppression. This keeps the geospatial geometry layer inspectable and stable, while the neural network focuses on instance priors, edge geometry, and successor relations. A second phase of DINO-EdgeQuery will investigate more tightly integrated polygon-level decoding.

Although the architecture defines a complete path from image input to final vector output, optimization was conducted branch-specifically. StageG1 updates the boundary-aware instance branch, while StageG2 trains EdgeQuery primitives and successor relations using contour-derived targets. During each stage, losses outside the target branch are masked to zero. The final inference model is produced by merging the trained branch weights and uses only predicted boxes and masks.

The contributions are: (1) a first-stage edge-first framework based on edge primitives and successor relations; (2) a boundary-aware DINOv3 instance prior with a backbone-agnostic adapter neck; (3) a deterministic, quality-gated line-intersection assembler with P2 contained-polygon suppression; and (4) a geospatial output path that converts image-space polygons into GeoJSON-compatible building footprints.

## 2. Related Work

### 2.1 Building Instance Segmentation

Deep learning has substantially advanced building extraction from aerial and satellite imagery, particularly through semantic and instance segmentation models that predict building regions, bounding boxes, or instance masks directly from image features. Early instance-oriented approaches adapted general-purpose instance segmentation architectures to remote-sensing imagery.

Zhao et al. (2018), for example, applied Mask R-CNN to satellite-image building extraction and showed that object-level masks are useful for separating adjacent buildings in dense urban scenes. However, they also observed that polygons derived from instance masks often contain irregular and noisy boundaries, motivating an additional boundary regularization step before the outputs can be used in cartographic or engineering applications.

Transformer-based segmentation models have also been introduced for building footprint extraction. Gibril et al. (2024) investigated Mask2Former with a Swin Transformer backbone for large-scale building footprint segmentation and reported strong generalization across urban areas with diverse building structures, heights, and sizes.

DINO-EdgeQuery builds on this line of work but does not aim to establish a new state of the art in building segmentation itself. Instead, it assumes that strong segmentation backbones and instance decoders can provide reliable building-aware boxes, masks, and query embeddings. The focus of this paper is the subsequent geometric decoding problem: how to convert these instance-level visual priors into compact, edge-structured, and GIS-ready building polygons.

### 2.2 Raster-to-Vector Building Extraction

For many geospatial applications, building extraction cannot stop at raster segmentation. Urban mapping, cadastral analysis, spatial database maintenance, and FOSS4G workflows require vector polygons that are compact, editable, georeferenced, and topologically valid. A common pipeline first predicts a binary building mask and then converts it into polygons using contour tracing, Marching Squares, polygon simplification, snapping, or boundary regularization. This strategy is attractive because it can reuse strong segmentation models and conventional GIS algorithms. However, the visual prediction and the geometric reconstruction are decoupled. Small errors along raster boundaries can produce redundant vertices, jagged outlines, fragmented polygons, self-intersections, or unstable corner placement. As a result, the final footprint quality often depends not only on the learned model but also on post-processing parameters.

In the FOSS4G context, Šanca et al. (2023) proposed an end-to-end workflow that combines binary semantic segmentation, boundary regularization, vectorization, and a QGIS plugin for GIS-ready building footprints. Their work highlights the practical importance of converting deep learning predictions into open geospatial vector data. However, it follows a mask-first raster-to-vector strategy, whereas DINO-EdgeQuery predicts instance-conditioned edge primitives and successor relations before deterministic line-intersection polygon assembly.

Recent learning-based methods have attempted to reduce this gap between raster masks and vector footprints. Frame Field Learning (FFL) augments a segmentation network with a learned local frame field aligned with building contours, thereby providing directional cues that improve segmentation and facilitate downstream polygonization (Girard et al., 2021). This is an important step toward geometry-aware building extraction, because the network learns structural information beyond foreground-background classification. Nevertheless, the final polygon is still produced by a separate polygonization procedure that combines the raster segmentation and the predicted frame field. Therefore, polygon validity, vertex count, and boundary regularity remain partly dependent on the polygonization algorithm.

Another line of work addresses the reversibility problem between masks and polygons. HiSup formulates polygonal building mapping as a problem of learning masks that better preserve polygonal structure, using hierarchical supervision from vertices, line segments, and regional masks (Xu et al., 2023). This approach explicitly recognizes that high-quality raster masks do not always convert cleanly back into polygonal annotations. By injecting geometric supervision at multiple levels, it improves the compatibility between learned masks and downstream polygonization. However, the primary learned representation still remains closely tied to mask prediction and polygonization, rather than treating polygon edges as the main output primitives.

More direct vector-prediction approaches have been proposed to avoid purely post-processing-based polygon extraction. PolyWorld introduced a vertex-graph formulation for polygonal building extraction, in which building vertices are detected and connected using a graph neural network. Re:PolyWorld further improves this line of work by redesigning the graph model to exploit both vertex features and edge appearance, and by using an edge-aware graph neural network to predict pairwise vertex connectivity (Zorzi and Fraundorfer, 2023). These methods demonstrate the importance of structured polygon representations beyond raster masks. However, they remain primarily vertex-graph formulations: the polygon is constructed by detecting vertices and optimizing their pairwise connections. In contrast, DINO-EdgeQuery treats edges rather than vertices as the primary prediction unit. It predicts instance-conditioned edge primitives and successor relations over active edges, and reconstructs polygon vertices through deterministic line intersections.

BoundaryFormer further explores polygonal prediction by using a Transformer-based model that predicts object boundaries as polygons while retaining mask-space supervision through differentiable rasterization (Lazarow et al., 2022). Although this reduces the reliance on hand-crafted polygonization, its primary representation is still an ordered sequence of boundary points.

DINO-EdgeQuery differs from these approaches in where the primary geometric structure is represented. Rather than converting masks into polygons, connecting detected vertices, or regressing dense boundary-point sequences, it predicts instance-conditioned edge primitives and learns directed successor relations among active edges. Polygon vertices are then reconstructed as intersections of consecutive support lines in the predicted cycle. This edge-first formulation is particularly suitable for building footprints, which are often dominated by long wall segments and sparse corners. It also provides a natural path toward compact, regularized, and GIS-ready vector output.

## 3. Method

### 3.1 Overview

As shown in Figure 1, DINO-EdgeQuery is organized into a neural prediction stage and a deterministic polygon assembly stage. Modules 1-8 form the neural network: a frozen DINOv3 backbone and adapter neck produce building-aware features; a Mask2Former-like instance decoder predicts masks and boxes; and an ROI-conditioned EdgeQuery branch predicts edge primitives and successor relations for each building instance.

Modules 9-10 are not learned neural decoders. They perform quality-gated geometric post-processing and output formatting. This separation allows the network to learn stable edge geometry while enforcing polygon validity by deterministic rules at inference time.

The final model used in this work follows a segmentation-first curriculum. We first improve the instance branch with boundary-aware segmentation training, then train the EdgeQuery branch on contour-derived edge targets. During these branch-specific stages, losses outside the target branch are explicitly masked to zero. The final checkpoint is assembled by merging the boundary-aware segmentation weights with the trained EdgeQuery weights, after which the full 1-10 inference pipeline is executed using predicted masks and boxes only.

The current implementation uses DINOv3 as the frozen backbone, but the adapter neck is designed as a backbone-agnostic interface rather than a DINO-specific component. It converts a small set of intermediate transformer taps into the canonical F8/F16/F32 pyramid and the lightweight F4 boundary-residual path used by the downstream modules. Consequently, a future SOTA visual foundation model can replace DINOv3 by mapping its intermediate features into the same adapter contract, while leaving the instance decoder, ROI tokenizer, EdgeQuery decoder, and deterministic polygon assembler unchanged.

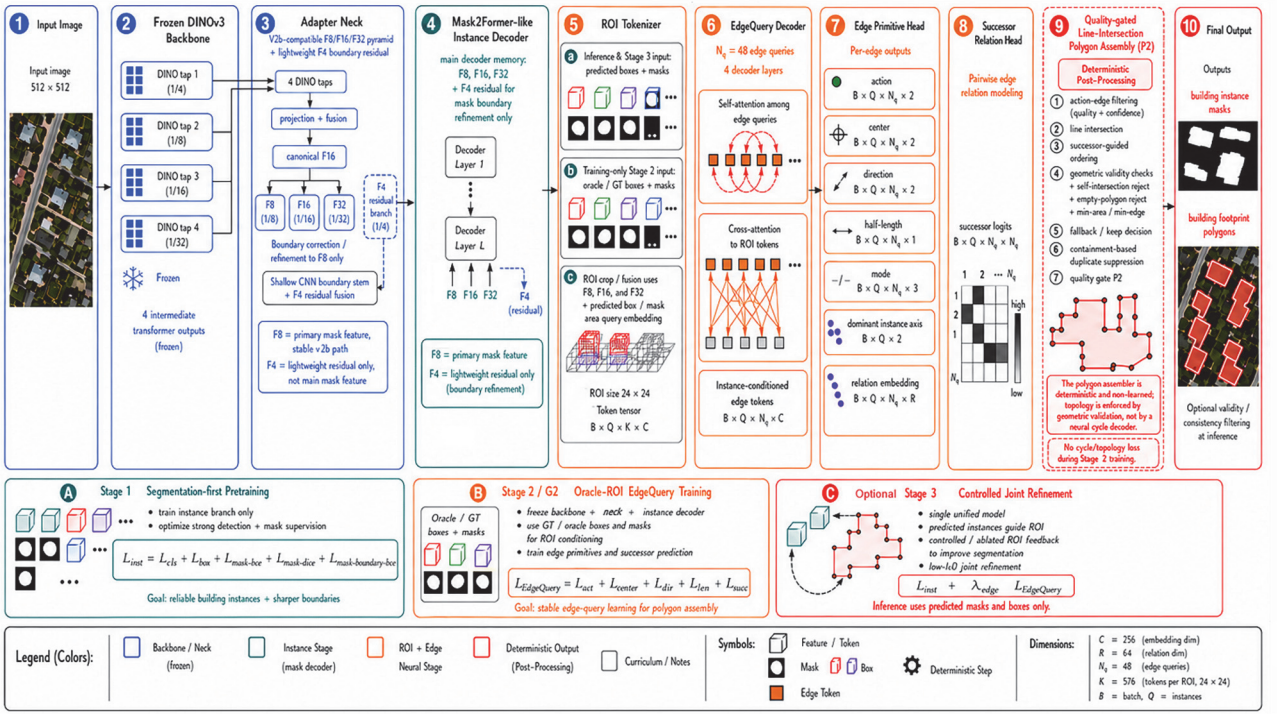


Figure 1. Overview of DINO-EdgeQuery.

### 3.2 Dataset Preparation for DINO-EdgeQuery

In this study, the target application is building footprint extraction from high-resolution satellite imagery, and the proposed method assumes paired training data in which each high-resolution satellite image is associated with corresponding building polygon annotations whose exterior winding directions are consistently normalized. Although the WHU Building Dataset is based on aerial imagery rather than satellite imagery, it is a public benchmark that closely satisfies these requirements and provides building annotations suitable for constructing the DINO-EdgeQuery training and validation data. Therefore, we use the WHU Building Dataset (Wuhan University Geospatial Computer Vision Group, n.d.), including its aerial imagery subset and COCO-format building labels at 0.2 m resolution, as the source dataset for this work.

Since DINO-EdgeQuery supervises both instance masks and edge-query targets, the original COCO annotations cannot be used directly.

We converted them into normalized 512 × 512 tile annotations suitable for instance segmentation, edge primitive learning, and successor-relation supervision. The preprocessing pipeline standardizes image size, clips polygons to tile boundaries, removes tiling artifacts, normalizes polygon geometry, and constrains polygon complexity to match the fixed EdgeQuery capacity.

This preprocessing produces aligned 512 × 512 image tiles, cleaned COCO-format instance masks, normalized exterior polygons, and edge-count-bounded building contours. The same preprocessing procedure is applied consistently to both the training and validation splits. The resulting annotations provide stable supervision for the segmentation branch, contour-derived EdgeQuery targets, and successor-relation learning. In particular, consistent exterior winding and bounded edge complexity are required because EdgeQuery learns ordered edge primitives and successor relations rather than unordered mask contours. This dataset preparation therefore serves as a bridge between standard COCO-style building annotations and the edge-first supervision format required by DINO-EdgeQuery. The preprocessing scripts are provided as supplementary material to support reproducibility.

| Step | Operation |
|---|---|
| Image tiling and annotation clipping | Each 1024 × 1024 WHU image is split into four non-overlapping 512 × 512 tiles. COCO polygon annotations are clipped by the tile window and translated into the local tile coordinate system. |
| Artifact and fragment removal | Polygon fragments generated by tiling are removed when their area is below a small pixel-area threshold. Tiny clipping-induced zig-zags and nearly degenerate fragments are suppressed to avoid unstable edge supervision. |
| Polygon cleaning and instance normalization | Invalid geometries are repaired when possible. MultiPolygon annotations are split into independent single-polygon building instances, and hole-like nested components are removed or flattened because the current EdgeQuery formulation assumes one exterior edge cycle per building. |
| Winding and coordinate normalization | Duplicate closing points and consecutive duplicate vertices are removed. Exterior winding direction is normalized under image coordinates, and bounding boxes are recomputed from the cleaned polygon coordinates. |
| Edge-count limitation | Since the model uses $N_v = 48$ edge queries per instance, polygons with more than 48 exterior edges are simplified while preserving topology whenever possible. Polygons that collapse during this simplification are discarded. |

Table 1. Preprocessing steps for constructing the DINO-EdgeQuery-ready dataset.

### 3.3 Problem Definition and Notation

Let $I$ denote an input overhead remote-sensing image and let $\mathcal{G}$ be the set of ground-truth building instances. Each instance consists of a class label, a bounding box, a binary instance mask, set as

$$I \in \mathbb{R}^{3\times H\times W}, \quad \mathcal{G} = \{(c_i^*, b_i^*, M_i^*, P_i^*)\}_{i=1}^{N},$$
$$P_i^* = \left(\mathrm{v}_{i,1}^*, \ldots, \mathrm{v}_{i,V_i}^*\right), \ \mathrm{v}_{i,t}^* \in \mathbb{R}^2 \quad (1)$$

Here, $c_i^*$, $b_i^*$, and $M_i^*$ denote the class label, bounding box, and binary instance mask of the $i$-th building, respectively. The polygon $P_i^*$ is represented as a variable-length sequence of image-space exterior vertices. This ground-truth polygon is used to derive contour-based edge supervision and to evaluate polygon quality, but DINO-EdgeQuery does not directly regress $P_i^*$ as a variable-length vertex sequence.

Instead, the proposed method learns a fixed set of instance-conditioned edge primitives and a successor relation over active edges. Each predicted building instance is first represented by an instance mask and box, which provide local visual priors for ROI-conditioned EdgeQuery decoding. The EdgeQuery branch then predicts edge activity, edge geometry, and successor connectivity. Polygon vertices are reconstructed geometrically from the ordered edge cycle by deterministic line-intersection assembly.

The predicted output for an image consists of building instance masks and topology-safe footprint polygons in image coordinates. These image-space polygons can later be transformed into geospatial coordinates using the affine georeference of the input raster. The key design choice of this work is to learn local edge geometry and directed connectivity while leaving final polygon assembly and validity enforcement to deterministic geometric post-processing.

### 3.4 Instance Segmentation Branch

The input image is processed by a frozen DINOv3 backbone and an adapter neck. The neck converts intermediate DINO features into a V2b-compatible multi-scale pyramid with F8, F16, and F32 features. In the final model, F8 remains the primary and stable mask-decoding feature. To improve boundary localization, the adapter neck additionally includes a shallow CNN boundary stem that extracts F4-like low-level boundary features from the RGB image. These F4-like features are not used as direct mask or ROI inputs; instead, they are projected and fused as a lightweight residual correction into the stride-8 feature path.

$$T_k = \phi_{\mathrm{DINO}}^{(k)}(I), \quad k = 1, \ldots, 4,$$
$$\mathbf{T} = (T_1, T_2, T_3, T_4),$$
$$(F_8, F_{16}, F_{32}) = \phi_{\mathrm{neck}}(\mathbf{T}),$$
$$F_8' = F_8 + \phi_{F_4\to F_8}\left(\phi_{\mathrm{stem}}(I)\right),$$
$$\mathcal{F}_{\mathrm{inst}} = (F_8', F_{16}, F_{32}). \quad (2)$$

A Mask2Former-like instance decoder predicts class logits, bounding boxes, masks, and instance query embeddings for Q building queries. StageG1 trains the adapter neck and instance decoder with boundary-aware supervision while the DINOv3 backbone and EdgeQuery branch are frozen. This stage is intended to improve generalization of building masks and sharpen building boundaries before EdgeQuery training.

### 3.5 ROI Tokenizer

For each predicted building instance, the ROI tokenizer extracts local multi-scale tokens using the instance box and mask. These ROI tokens provide instance-conditioned context for EdgeQuery decoding. During oracle-ROI training, ground-truth boxes and masks are used only to stabilize local conditioning. During final inference, no oracle boxes or masks are used; the tokenizer receives predicted masks and boxes from the instance branch.

The ROI tokenizer is part of the neural EdgeQuery branch and is trained during StageG2 together with the EdgeQuery decoder and prediction heads. In contrast, the backbone, adapter neck, and instance decoder are frozen during StageG2 so that the EdgeQuery branch learns to use a fixed instance prior.

$$\mathcal{F}_{\mathrm{roi}} = (F_8', F_{16}, F_{32}), \quad \widehat{M}_q = \sigma\left(\widehat{m}_q\right),$$
$$\mathbf{Z}_q^{\mathrm{pred}} = \phi_{\mathrm{roi}}\left(\mathcal{F}_{\mathrm{roi}}, \hat{b}_q, \widehat{M}_q\right),$$
$$\mathbf{Z}_i^{\mathrm{oracle}} = \phi_{\mathrm{roi}}(\mathcal{F}_{\mathrm{roi}}, b_i^*, M_i^*). \quad (3)$$

### 3.6 EdgeQuery Decoder and Edge Primitive Parameterization

Given ROI tokens, the EdgeQuery decoder predicts a fixed set of edge slots for each building instance. Each edge slot is conditioned on the local instance context and represents a potential boundary segment. Instead of predicting a variable-length vertex sequence, DINO-EdgeQuery represents each footprint through active edge primitives.

For each edge slot, the Edge Primitive Head predicts activity, center, direction, half-length, mode, dominant instance axis, and a relation embedding. The activity score determines whether a slot is used. The center, direction, and half-length define a support line segment, and the relation embedding supports successor prediction.

### 3.7 Successor Relation and Polygon Assembly

The Successor Relation Head predicts directed connectivity among active edge primitives. Instead of constructing a graph over detected vertices, DINO-EdgeQuery models the next-edge relation over predicted edge primitives. The successor logits define candidate directed cycles that can be decoded into ordered edge sequences.

In the final implementation, successor relation learning remains part of the neural objective, but polygon assembly itself is not a learned neural decoder. The neural network produces edge geometry and successor scores; the subsequent cycle selection, line intersection, validity checks, and duplicate suppression are deterministic post-processing steps.

Let $S_q$ denote the successor-score matrix over active edge primitives, and let $C_q$ denote the selected directed successor cycle for instance q.

### 3.8 Quality-gated line-intersection polygon assembly

For each predicted instance, active edge primitives are converted into support lines using their predicted centers and directions. A successor-guided cycle $C_q = (k_1, \ldots, k_V)$ orders the selected edges. Consecutive support lines are then intersected to reconstruct polygon vertices. This formulation is well suited to building footprints, which are often composed of straight wall segments and sparse corners. It also avoids directly regressing a variable-length vertex sequence.

$$\hat{\mathbf{n}}_{q,k} = R_{\pi/2}\frac{\hat{\mathbf{d}}_{q,k}}{\left\|\hat{\mathbf{d}}_{q,k}\right\|_2}, \quad \hat{\rho}_{q,k} = \hat{\mathbf{n}}_{q,k}^{\top}\hat{\mathbf{c}}_{q,k},$$
$$\mathcal{L}_{q,k} = \left\{\mathbf{x} \in \mathbb{R}^2 \mid \hat{\mathbf{n}}_{q,k}^{\top}\mathbf{x} = \hat{\rho}_{q,k}\right\},$$
$$\hat{\mathbf{v}}_{q,t} = \mathrm{intersection}\left(\mathcal{L}_{q,k_t}, \mathcal{L}_{q,k_{t+1}}\right),$$
$$k_{V+1} = k_1. \quad (4)$$

Here, $\hat{\mathbf{c}}_{q,k}$ and $\hat{\mathbf{d}}_{q,k}$ denote the predicted edge center and direction of the $k$-th edge primitive for instance $q$, respectively. $R_{\pi/2}$ rotates the predicted direction by 90 degrees to obtain the line normal $\hat{\mathbf{n}}_{q,k}$, and $\hat{\rho}_{q,k}$ is the corresponding support-line offset. The ordered edge cycle $C_q$ determines the sequence of

support lines to be intersected.
The final inference profile uses a quality-gated line-intersection assembler. Quality Gate P1 rejects polygons that fail mask-consistency, coverage, outside-ratio, area-ratio, or geometric-validity checks. Quality Gate P2 extends P1 with containment-based duplicate suppression. Specifically, a polygon $P_a$ is removed when another polygon $P_b$ satisfies the containment conditions in Eq. (5): at least 50% of $P_a$ is covered by $P_b$, $P_b$ is at least as large as $P_a$, and the centroid of $P_a$ lies inside $P_b$.

$$\frac{|P_a \cap P_b|}{|P_a|} \geq 0.50, \quad |P_b| \geq |P_a|, \\ \text{centroid}(P_a) \in P_b. \quad (5)$$

Here, $|\cdot|$ denotes polygon area. This containment criterion suppresses duplicated footprints that are difficult to remove by Intersection over Union (IoU)-based non-maximum suppression alone, especially when a smaller polygon is nested inside a larger prediction for the same building. The polygon assembler is deterministic and non-learned; topology is enforced by geometric validation rather than by a neural cycle decoder.

### 3.9 Training Supervision and Segmentation-First Curriculum

DINO-EdgeQuery is trained with a segmentation-first curriculum. The purpose of the curriculum is not to train all losses simultaneously from the beginning, but to improve each component under stable conditioning. In practice, the full 1-10 inference graph is retained, while optimization is decomposed into branch-specific stages. Losses outside the current target branch are masked to zero, preventing unstable gradients from non-target modules from degrading the branch being optimized.

**StageG1: Boundary-aware segmentation fulltrain.** In the first stage, the model is optimized as an instance segmentation model on the full training set. The DINOv3 backbone and EdgeQuery branch are frozen, while the adapter neck and instance decoder are updated. The objective is to improve building localization, masks, and boundary sharpness before EdgeQuery training. In addition to classification, box, mask BCE, and mask Dice terms, we add a boundary-focused mask BCE loss computed on a narrow ground-truth boundary band.

$$L_{\text{inst}} = \lambda_{\text{cls}} L_{\text{cls}} + \lambda_{\text{box}} L_{\text{box}} + \lambda_{\text{giou}} L_{\text{giou}} \\ + \lambda_{\text{bce}} L_{\text{mask-bce}} + \lambda_{\text{dice}} L_{\text{mask-dice}} \\ + \lambda_{\text{bd}} L_{\text{mask-boundary-bce}}. \quad (6)$$

In the architecture figure, this is written in the simplified form L_inst = L_cls + L_box + L_mask-bce + L_mask-dice + L_mask-boundary-bce. The boundary term encourages sharper building outlines and reduces the tendency of interior-dominated mask losses to ignore small boundary errors.

**StageG2: EdgeQuery fulltrain with contour-derived targets.** After StageG1, the segmentation branch is frozen. The ROI tokenizer, EdgeQuery decoder, Edge Primitive Head, and Successor Relation Head are trained on nonempty training images using contour-derived edge targets. Ground-truth masks and boxes are used for ROI conditioning during this training stage, but not during inference. The objective supervises edge activity, center, direction, half-length, and successor relation only.

$$L_{\text{EdgeQuery}} = \lambda_{\text{act}} L_{\text{act}} + \lambda_{\text{center}} L_{\text{center}} + \lambda_{\text{dir}} L_{\text{dir}} + \\ \lambda_{\text{len}} L_{\text{len}} + \lambda_{\text{succ}} L_{\text{succ}} \quad (7)$$

Earlier design variants included vertex, raster, topology, regularity, and cycle-consistency losses. In the final model, these terms are retained only as zero-masked placeholders for interface compatibility and future controlled joint refinement. They are not active in the reported StageG2 objective. Topology is instead enforced by the deterministic line-intersection assembler and its geometric validity checks at inference time.
After branch-specific training, the StageG1 segmentation weights and the StageG2 EdgeQuery weights are merged into a single checkpoint. The merged model executes the full pipeline at inference: instance prediction, ROI tokenization, edge primitive prediction, successor relation prediction, deterministic polygon assembly, and final output formatting.

**Optional controlled joint refinement.** The original DINO-EdgeQuery design also allows controlled joint refinement in which predicted instances guide ROI conditioning and the instance and EdgeQuery branches are refined in a single model. In the final experiments reported here, this stage is treated as an optional extension rather than the primary training path. The reported final model uses StageG1 boundary-aware segmentation training followed by StageG2 EdgeQuery training and deterministic P2 polygon assembly.

$$L_{\text{joint}} = L_{\text{inst}} + \lambda_{\text{edge}} L_{\text{EdgeQuery}}, \quad (8)$$

with non-target losses masked when a branch is trained separately.
This formulation keeps the accepted system design intact while reflecting the implementation used for the final model: neural modules predict instance priors, edge primitives, and successor relations, while polygon validity and duplicate suppression are handled by deterministic geometric post-processing.
At inference time, no oracle boxes or masks are used. The model first predicts building instances, then decodes edge primitives and successor relations for each predicted instance, and finally reconstructs building footprint polygons using the quality-gated deterministic line-intersection assembler.
The polygon assembler is deterministic and non-learned. Topology is enforced by geometric validity checks, self-intersection rejection, empty-polygon rejection, mask-consistency thresholds, and P2 containment-based duplicate suppression rather than by a neural cycle decoder.
This curriculum separates the main difficulties of the problem: boundary-aware instance recognition, edge-query learning under stable ROI conditioning, and topology-safe geometric polygon assembly at inference time. It also provides a practical path to improve each component independently without destabilizing the full system.

### 3.10 Open Geospatial Vector Data Output

Let $A$ denote the affine geotransform of the input GeoTIFF. For the $i$-th predicted footprint, each image-space vertex $(x_{i,t}, y_{i,t})$ is transformed into the georeferenced coordinate $\boldsymbol{g}_{i,t}$ by applying $A$. The geospatial polygon $P_i^{geo}$ is then constructed from the transformed vertices and exported as a GeoJSON feature.

$$\mathbf{g}_{i,t} = A\left[x_{i,t}, y_{i,t,1}\right]^T, \; P_i^{\text{geo}} = \text{Polygon}\left(\mathbf{g}_{i,1}, \ldots, \mathbf{g}_{i,V_i}\right). \quad (9)$$

The predicted image-space polygon vertices are converted into map coordinates using the affine georeference stored in the input GeoTIFF. In our test-time export pipeline, we use Rasterio, which is built on GDAL, to read the GeoTIFF affine transform and coordinate reference system. Each predicted pixel-space vertex is transformed by the affine mapping from image

coordinates to projected map coordinates. When reprojection is required, pyproj/PROJ is used to transform the output coordinates to a target CRS such as EPSG:4326. The resulting building footprints are exported as GeoJSON features, preserving prediction attributes such as confidence score, polygon quality measures, and filtering status.

## 4. Experiments

### 4.1 Dataset and Evaluation Splits

The experimental evaluation uses the DINO-EdgeQuery-ready dataset constructed from the WHU Building Dataset as described in Section 3.2. StageG1 is trained on all available training tiles, including tiles without buildings, because the instance branch must learn both building recognition and background rejection. StageG2 is trained only on nonempty training tiles, because contour-derived edge targets require at least one building polygon.

For validation, we report results on three splits. The nonempty split contains only tiles with at least one annotated building and is used to measure polygon quality, high-IoU recall and precision, and boundary accuracy. The empty split contains tiles without buildings and is used to measure false-positive robustness. The all split combines both subsets and reflects deployment-like behavior, but its mean image-level IoU values should be interpreted carefully because empty tiles do not define building-recall targets

### 4.2 Evaluation Metrics

We evaluate both raster-level and polygon-level behavior. Raster-level measures include union IoU, union precision, union recall, and boundary IoU at 2 px and 3 px tolerances. Polygon matching is summarized by Recall@75, Precision@75, Recall@80, and Precision@80, where predicted polygons are matched to ground-truth instances by IoU thresholds. We also report pred_per_gt, fp_per_gt, and duplicates_per_gt to quantify over-prediction and duplicate footprints.

Because DINO-EdgeQuery explicitly outputs vector geometry, we further report validity-oriented measures: self-intersection count, triangle-collapse count, failed images, no-valid outputs per ground-truth instance, polygon-to-mask IoU, coverage, and outside ratio. For empty validation tiles, the main metrics are the total number of predicted polygons and the empty-image false-positive rate.

### 4.3 Implementation and Inference Settings

The final model is obtained by StageG1 boundary-aware segmentation fulltrain followed by StageG2 EdgeQuery fulltrain on contour-derived edge targets. The deployed checkpoint is the StageG2 best training-loss checkpoint. During inference, the instance branch predicts masks and boxes, the ROI tokenizer conditions the EdgeQuery decoder, and the Edge Primitive and Successor Relation heads predict edge activity, geometry, and directed connectivity. The polygon assembler is deterministic and non-learned.

The final inference profile uses conf_threshold = 0.60, mask_threshold = 0.50, pre-mask NMS = 0.75, post-polygon NMS = 0.75, successor-cycle assembly, edge_active_threshold = 0.15, and 32 edge candidates per instance. Boundary support is enabled, and collinear merging is disabled.

### 4.4 Quality Gate Profiles

Quality Gate P1 filters line-intersection polygons using mask-consistency and geometric constraints: minimum mask IoU = 0.15, minimum coverage = 0.35, and maximum outside ratio = 0.65. P2 extends P1 with containment-based duplicate suppression. A polygon A is removed when at least 50% of A is covered by another polygon B, B is at least as large as A, and the centroid of A lies inside B. This containment criterion removes nested duplicate predictions that are not reliably removed by IoU-based NMS.

## 5. Results

### 5.1 Validation Results on Nonempty Tiles

Table 2 compares P1 and P2 on the nonempty validation split. P2 reduces the number of predicted polygons and false positives, while increasing high-IoU precision. The trade-off is a modest decrease in recall, which is expected because the containment filter intentionally removes nested or redundant polygons.

| Metric | P1 | P2 |
|---|---|---|
| Predicted polygons | 21,554 | 19,203 |
| Union IoU | 0.7107 | 0.7064 |
| Boundary IoU 2 px | 0.3746 | 0.3710 |
| Boundary IoU 3 px | 0.4535 | 0.4496 |
| Recall@75 | 0.6459 | 0.6213 |
| Precision@75 | 0.5486 | 0.5924 |
| Recall@80 | 0.5442 | 0.5258 |
| Precision@80 | 0.4622 | 0.5013 |
| Predictions/GT | 1.1774 | 1.0489 |
| FP/GT | 0.2997 | 0.2295 |
| Duplicates/GT | n.r. | 0.0007 |
| Contained removed/GT | 0 | 0.1284 |
| Self-int. / triangle | 0 / 0 | 0 / 0 |

Table 2. Effect of Quality Gate P2.

### 5.2 Empty-Tile False Positive Robustness

The empty validation subset measures whether the model hallucinates buildings in tiles without annotated structures. P2 reduces the total number of predictions on empty tiles compared with P1, indicating that containment-based filtering also suppresses some false positives in background-only scenes.

| Metric | P1 | P2 |
|---|---|---|
| Empty tiles | 980 | 980 |
| Predicted polygons | 90 | 81 |
| Mean pred./tile | 0.0918 | 0.0827 |

Table 3. Empty-tile false-positive summary.

### 5.3 Qualitative Image-Space and Geospatial Results

Figure 2 shows the image-space inference pipeline, including the instance mask predicted from the validation data, EdgeQuery primitives selected successor-cycle edges, and the final P2 polygon overlay.

Figure 3 shows the geospatial output path using the test image. To verify that the predicted footprints can be used in a practical FOSS4G workflow, we exported the final P2 polygon outputs to GeoJSON using the georeference stored in the input GeoTIFF. Rasterio/GDAL was used to read the affine transform and CRS, and pyproj/PROJ was used for optional CRS transformation. The generated GeoJSON was then loaded into QGIS and overlaid on an OpenStreetMap basemap. This confirms that the proposed pipeline produces not only image-space predictions but also geospatial vector outputs that can be inspected in common open-source GIS software.

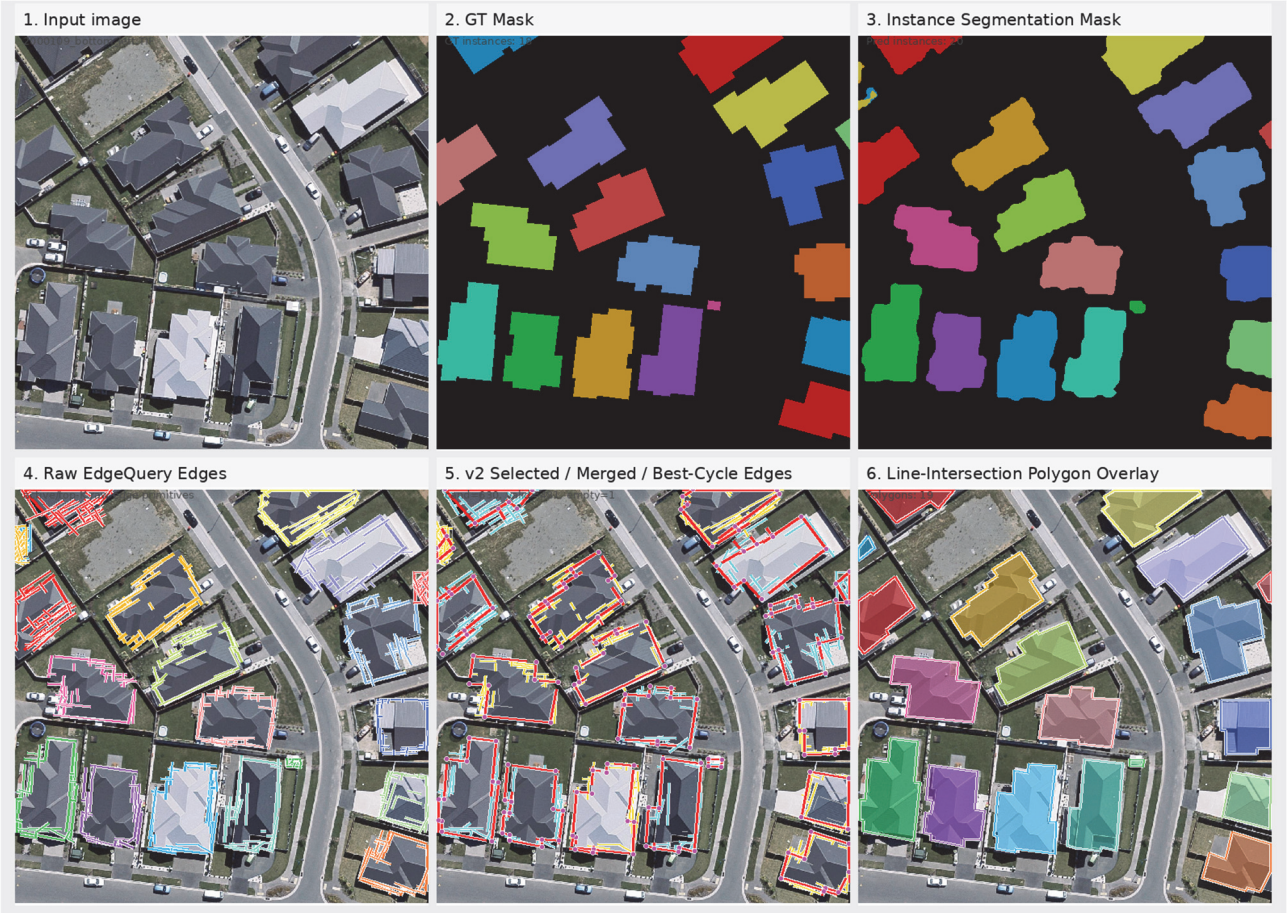


Figure 2. Image-space inference visualization of DINO-EdgeQuery.


Figure 3. Geospatial overlay of DINO-EdgeQuery GeoJSON footprints on OpenStreetMap.

### 5.4 Effect of Containment-Based Duplicate Suppression

Containment-based duplicate suppression addresses the failure mode in which multiple line-intersection polygons are produced for the same building footprint. IoU-based polygon NMS is not sufficient for this case: when a smaller polygon is almost fully nested inside a larger footprint, the intersection-over-union can remain low because the union area is dominated by the larger polygon. The final P2 profile removes a polygon A when at least 50% of A is covered by another polygon B, B is at least as large as A, and the centroid of A lies inside B. This rule is applied after the P1 quality gate and post-polygon NMS, and it uses only predicted polygon geometry and mask-consistency information; no ground-truth annotation is used at inference time.

On the nonempty validation split, P2 reduced predicted polygons from 21,554 to 19,203 and FP/GT from 0.2997 to 0.2295, while increasing Precision@75 from 0.5486 to 0.5924 and Precision@80 from 0.4622 to 0.5013. The trade-off is a modest decrease in recall, which is expected because the filter intentionally removes contained or redundant footprint hypotheses. We therefore adopt P2 as the final inference profile for cleaner GIS-ready polygon output.

## 6. Discussion

### 6.1 Edge-First Polygon Decoding with Instance Priors

DINO-EdgeQuery demonstrates that instance segmentation and vector polygon decoding do not have to be treated as competing formulations. In the proposed design, instance segmentation

provides building-aware visual priors, while the EdgeQuery branch learns structured edge geometry and successor relations. This hybrid design avoids directly polygonizing masks, yet still benefits from the localization strength of modern instance segmentation.

### 6.2 Deterministic Geometry as a Validity Layer

A key design decision is to separate neural prediction from geometric validity enforcement. The neural network predicts edge primitives and successor relations, whereas line intersections, self-intersection rejection, empty-polygon rejection, mask-consistency filtering, and contained-polygon suppression are implemented as deterministic post-processing. This is particularly suitable for FOSS4G workflows, where reproducible geometry validation and GIS-ready vector output are as important as raw neural prediction accuracy.

### 6.3 Current Limitations

The current model remains dependent on the quality of the instance branch. If a building is missed by instance segmentation, the EdgeQuery decoder cannot recover its footprint. P2 also introduces a precision-recall trade-off: it removes redundant and low-quality polygons, but it may also remove some valid predictions in difficult cases. In addition, the quality-gate thresholds are currently hand-designed and may require adaptation to other domains, image resolutions, or building styles. Another limitation is that the current formulation assumes one exterior edge cycle per building instance. Hole-like structures, courtyards, and multipart buildings are flattened or excluded during dataset preparation. Finally, the WHU Building Dataset is aerial rather than satellite imagery; although it closely matches the overhead-image and building-polygon requirements of this study, broader cross-domain validation is needed before operational deployment.

## 7. Conclusions

Future work will focus on stronger and more domain-robust instance priors, adaptive polygon-quality prediction, and differentiable feedback from geometric assembly to neural edge prediction. The deterministic P2 thresholds could be replaced or complemented by a learned polygon-quality head. Another important direction is to extend EdgeQuery to multi-cycle and hole-aware polygons, enabling explicit representation of courtyards, annexes, and multipart building footprints. Because the adapter neck defines a backbone-agnostic interface, future visual foundation models can also replace DINOv3 without changing the downstream EdgeQuery formulation.

Overall, DINO-EdgeQuery provides an edge-first framework for building footprint extraction from overhead imagery. The method uses instance segmentation as a visual prior, predicts edge primitives and successor relations, and converts the learned edge structure into topology-safe footprint polygons through deterministic quality-gated line-intersection assembly. The final P2 profile further suppresses contained duplicate polygons and produces cleaner GIS-ready outputs. Although practical deployment still requires stronger generalization and adaptive quality control, the results indicate a promising direction for connecting deep visual prediction with verifiable geospatial vector geometry.

## Acknowledgements

AI tools were used to assist with the preparation of schematic figures and with translation and language editing. All scientific content, results, interpretations, and final decisions were reviewed and verified by the author.

## References

Cheng, B., Misra, I., Schwing, A. G., Kirillov, A., Girdhar, R., 2022. Masked-attention mask transformer for universal image segmentation. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1290–1299.

Gibril, M.B.A., Al-Ruzouq, R., Shanableh, A., Jena, R., Bolcek, J., Shafri, H.Z.M., Ghorbanzadeh, O., 2024. Transformer-based semantic segmentation for large-scale building footprint extraction from very-high resolution satellite images. *Advances in Space Research*, 73(10), pp. 4937–4954.

Girard, N., Smirnov, D., Solomon, J., Tarabalka, Y., 2021. Polygonal building extraction by frame field learning. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5891–5900.

Lazarow, J., Xu, W., Tu, Z., 2022. Instance segmentation with mask-supervised polygonal boundary transformers. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4382–4391.

Šanca, S., Jyhne, S., Gazzea, M., and Arghandeh, R., 2023. An end-to-end deep learning workflow for building segmentation, boundary regularization and vectorization of building footprints. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLVIII-4/W7-2023, 169–175. Available at: isprs-archives.copernicus.org/articles/XLVIII-4-W7-2023/169/2023/ (11 Jun 2026).

Siméoni, O., Vo, H. V., Seitzer, M., Baldassarre, F., Oquab, M., Jose, C., Khalidov, V., Szafraniec, M., Yi, S., Ramamonjisoa, M., Massa, F., Haziza, D., Wehrstedt, L., Wang, J., Darcet, T., Moutakanni, T., Sentana, L., Roberts, C., Vedaldi, A., Tolan, J., Brandt, J., Couprie, C., Mairal, J., Jégou, H., Labatut, P., Bojanowski, P., 2025. DINOv3. *arXiv preprint arXiv:2508.10104*.

Wuhan University Geospatial Computer Vision Group, n.d. *WHU Building Dataset*. Available at: gpcv.whu.edu.cn/data/building_dataset.html (11 Jun 2026).

Xu, B., Xu, J., Xue, N., Xia, G.-S., 2023. HiSup: Accurate polygonal mapping of buildings in satellite imagery with hierarchical supervision. *ISPRS Journal of Photogrammetry and Remote Sensing*, 198, pp. 284–296.

Zhao, K., Kang, J., Jung, J., Sohn, G., 2018. Building extraction from satellite images using Mask R-CNN with building boundary regularization. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 242–246.

Zorzi, S., Bazrafkan, S., Habenschuss, S., Fraundorfer, F., 2022. PolyWorld: Polygonal building extraction with graph neural networks in satellite images. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1848–1857.

Zorzi, S., Fraundorfer, F., 2023. Re:PolyWorld - A Graph Neural Network for Polygonal Scene Parsing. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 16762–16771.