UiT

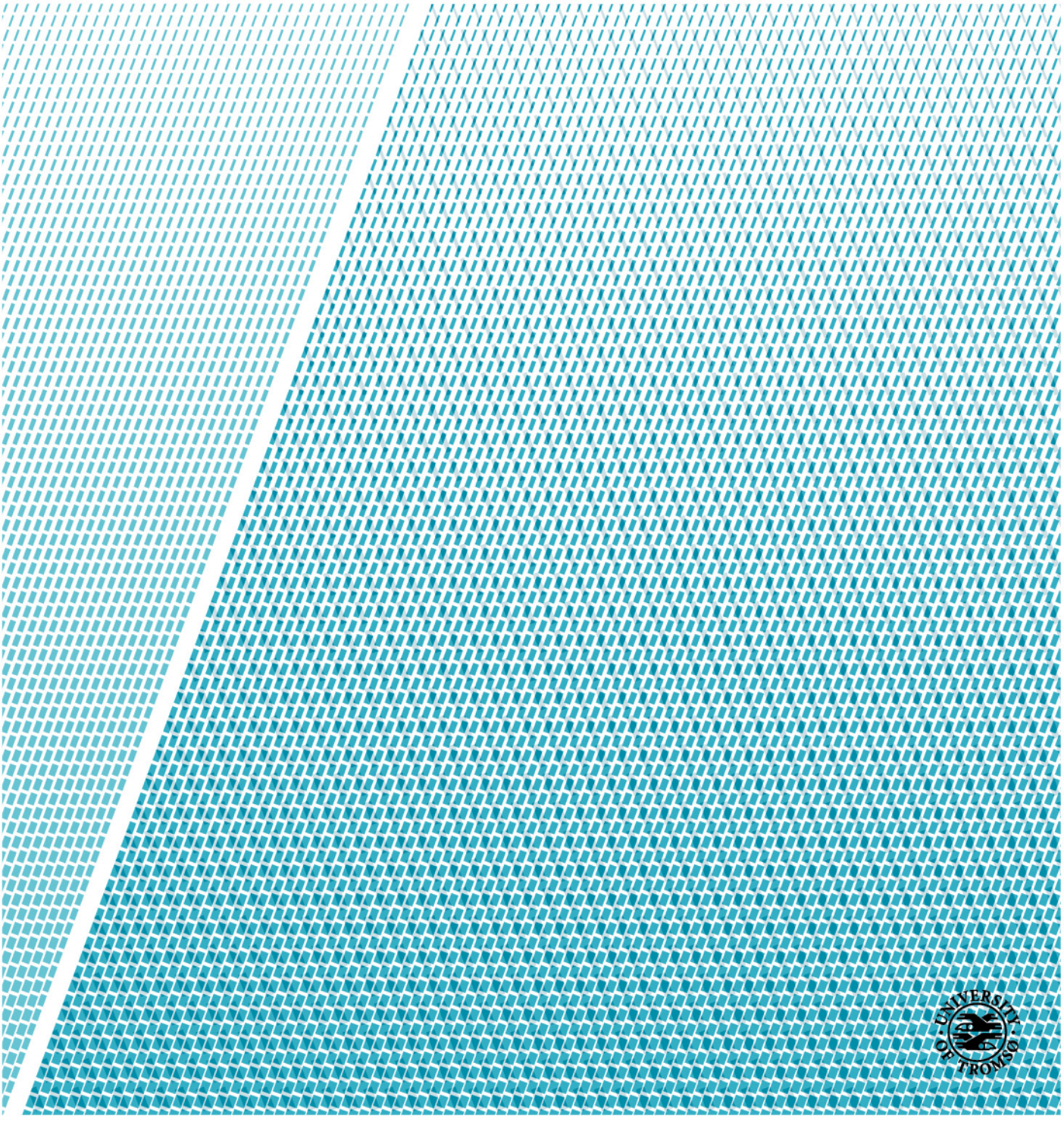
THE ARCTIC UNIVERSITY OF NORWAY

Faculty of Science and Technology, Department of Mathematics and Statistics

Automatic Building Structure Extraction from Aerial Photographs using Transformers

Sigbjørn Valde Foss

STA-3941 - Master's Thesis in Applied Physics and Mathematics 30 SP - June 2024



Abstract

This thesis explores a method for automatically extracting three-dimensional, geolocated building roof structure representations from non-orthorectified aerial photos, with a view to automate time-intensive tasks that are currently handled manually. A two stage framework for 3D roof structure inference from aerial photos is proposed. A transformer-based deep learning framework is used to extract the roof structure of buildings from non-orthorectified aerial photos, and a multi-view triangulation method is used to transform the extracted planar graphs to a 3D representation of the building roof without the use of a digital surface model. Additionally, a preprocessing method for producing training data for the roof structure extraction model using aerial images and existing 3D building graph representations from a national database is developed. The proposed framework is shown to successfully infer building roof structure graphs, achieving an F1-score of 0.77 on the test set, and a set of inferred graphs are successfully transformed to 3D with a mean average corner reconstruction error of 0.43 meters. The results are promising, and automatic extraction and 3D reconstruction is viable, but the process requires further development.

Contents

List of Tables

List of Figures

1 Introduction

1.1 Context

The demand for large amounts of accurate geographical data is ever-increasing, and traditional analysis methods are unsuited to process the data quickly enough. High-resolution aerial imagery can provide detailed and granular views of urban environments, but the volume of data poses significant challenges for manual analysis and interpretation. Traditional methods of manually surveying and documenting buildings are time-consuming, labor-intensive, and prone to errors. Recent advancements in deep learning-based computer vision offer avenues for automating such processes, and these methods are subject to the interest of an active research community (Gao, Peters, & Stoter, 2024).

This study is conducted as a part of the KartAI project, which is a collaboration between the Norwegian Mapping Authority (Kartverket), Kristiansand kommune, the University of Agder, and Norkart. KartAI aims to develop deep learning methods for handling building information in the shared database Sentral Felles Kartdatabase - Bygning (SFKB), a large shared database of building structural elements, with a view to automate time-intensive processes that are currently handled manually. The project has made strides in producing technology to detect buildings using segmentation among other things [1] (KartAI, 2021). This thesis adds to the project by attempting to develop a method for automatically extracting more detailed structural information about individual buildings.

The central aim of this study is to apply a deep learning planar graph[2] reconstruction method developed in J. Chen, Qian, and Furukawa (2021) in conjunction with multi-view triangulation (MVT) (Hartley & Zisserman, 2004) to produce 3D graph representations of buildings found in aerial images in Universal Transverse Mercator (UTM) coordinates[3], to automate the incorporation of vectorized building structures into the SFKB database. This is motivated by the availability of two large data sources in Karverkets possession: The building database SFKB and a collection of high-resolution aerial images collected during regular imaging missions ordered by Kartverket. The planar graph reconstruction method, HEAT (Holistic Edge Attention Transformer), is a supervised[4] deep learning model, which infers the graph structure of buildings from images. Resampled cutouts of buildings from the aerial images can be used as input data, and the SFKB building representations can be simplified and transformed to conform to the format of the HEAT model output, allowing them to be used as training labels. This makes possible the training of a powerful state-of-the-art deep learning model that can perform automated building structure extraction from aerial images.

The HEAT model produces 2D graph representations of single buildings in image coordinates, and a secondary objective of this thesis is to apply a method to construct 3D representations of the building graphs in world coordinates from the 2D inferences using MVT. MVT requires representations of the building graphs in multiple images taken from different viewpoints. The aerial images overlap in their coverage by 80% in the in-flight direction, which means that every building is represented in up to five separate images taken from different viewpoints. HEAT can be applied for structural inference in each of the different view images, yielding planar graph representations in the coordinates of images from multiple vantages and enabling the use of MVT for 3D reconstruction.

HEAT was selected for this project due to its recent development, previous successful applications beyond its original publication (Campoverde et al., 2024), and its supervised nature, which makes possible the use of the ample labeled data available in the SFKB database. Moreover, the failure

[1] Segmentation refers to the process of dividing an image into smaller regions based on certain characteristics or features. In this case, the regions are divided according to whether the algorithm believes there is a building there or not (binary semantic segmentation)

[2] A graph is an object consisting of nodes (points) connected by edges (lines). A planar graph is a graph embedded in a 2D plane where no edges intersect unless connected by a node (Barthelemy, 2017).

[3] UTM is the most commonly used map projection system for assigning coordinates to locations on the surface of the earth (Langley, 1998). Building representations in the SFKB database are in UTM coordinates.

[4] Supervised machine learning methods involve training a model on labeled data, where the algorithm learns the relationship between input features and corresponding output labels

cases of HEAT primarily occur in extracting large, uncommonly structured buildings, whereas this project's focus lies on detecting small, relatively uniform structures. Similar research has been performed in other projects (Campoverde et al., 2024), but the method presented in this thesis differs from earlier research in two ways: Firstly, the structural inference is applied to aerial photos that have not been orthorectified[5]. Non-rectified photos have the advantage of retaining a direct geometrical relation between real world objects and their image representation, which can be calculated if certain camera properties are known, but they display more of the non-roof structure of buildings, which may make inference more difficult. Secondly, the use of overlapping and non-rectified photographs enables the use of relatively simple MVT techniques for 3D reconstruction without the use of a digital surface model[6] (DSM).

1.2 Research Objectives

The objectives of this study can be broken down into four parts:

1. Find and implement a method for transforming SFKB building graphs from 3D UTM to the image coordinate systems of the aerial images.
2. Process the transformed SFKB graphs and aerial images to be used as labels and input data respectively for the HEAT model.
3. Train the HEAT model on the prepared data
4. Apply triangulation to the features extracted by the trained HEAT model in several different views of the same building to transform them back to UTM coordinates.

1.3 Paper structure

The following is an overview of the paper structure.

1.Introduction

An overview of the project, defining the research objectives and describing its context.

2. Related work

An overview of the development in the field of building structure extraction.

3. Theory

Theoretical foundation of the thesis. The central subject is the HEAT model (J. Chen et al., 2021) and the concepts it is built upon. The section also details the theoretical basis for the methods used for projecting the UTM data into the camera coordinate systems and for transforming the 2D graphs into the 3D UTM system. Finally, it introduces some terminology relating to graphs.

4. Methods

An overview of the methods used in this research. This includes a description of the input and label data, a description of the preprocessing method, details on training and applying the HEAT model, and the 3D reconstruction method.

5. Results

Description of the results of data preparation, model training and inference and 3D reconstruction.

[5] An orthophoto is an aerial or satellite photo which has been adjusted for topographic relief, lens distortion, and camera tilt (Wolf, DeWitt, & Wilkinson, 2013).

[6] A DSM is a digital spatial representation of the terrain surface (Hirt, 2016)

6. Discussion

A more in-depth discussion of the results obtained in section 5, and proposal of potential improvements.

7. Conclusion

Conlcusion of the thesis.

2 Related Work

2.1 Building detection

The problem of extracting buildings from remote sensing imagery has seen continuous research efforts for decades (Hu, Wang, Huang, & Liu, 2023). Traditional approaches typically rely on manually crafted systems that analyze various aspects of building appearances, including shadows, colors, and geometric features to extract 2D features like building boundaries or segmentation masks (Kim & Muller, 1999; Côté & Saeedi, 2013; Li, Femiani, Xu, Zhang, & Wonka, 2015). However, these methods are often tailored to specific scenarios and require meticulous tuning for new datasets (Yuan, 2016). With the emergence of deep neural networks (DNNs), techniques such as semantic segmentation (e.g., U-net (Ronneberger, Fischer, & Brox, 2015), Deeplab (L.-C. Chen, Papandreou, Kokkinos, Murphy, & Yuille, 2016), ViT (Dosovitskiy et al., 2020)) and instance segmentation (e.g., Mask R-CNN (He, Gkioxari, Dollár, & Girshick, 2017)) have shown notable advancements (Hu et al., 2023). Despite their effectiveness, these segmentation methods primarily yield simple raster representations of buildings. Recent trends have shifted towards extracting vectorized representations such as graphs or polygons, which offer more efficient ways of representing regular architectural features. These representations, characterized by straight connected lines, naturally capture the layout of rooms and buildings and are more adaptable for transitioning from 2D to 3D contexts.

2.2 Structure reconstruction

Classical methods for inferring vectorized structures from raster data include image processing techniques such as the Hough-transform (Macé, Locteau, Valveny, & Tabbone, 2010) and superpixel segmentation (Qin et al., 2018), which typically involve extensive parameter tuning. Techniques like these have been studied extensively within the field of computer vision with emphasis on the planar graph reconstruction and room layout estimation tasks (J. Chen et al., 2021).

Recently, deep learning techniques have come to dominate the domain, with models like L-CNN (Zhou, Qi, & Ma, 2019) achieving notable success in planar graph reconstruction tasks.

2.3 Roof structure extraction models

Roof structure extraction, a subset of structured reconstruction, involves inferring the graph structure of building rooftops from aerial or satellite images of individual buildings. Unlike the polygon extraction task tackled by PolyBuilding (Hu et al., 2023), this problem extends to inferring structure within the interior of the roof outline. Initially proposed by Nauata and Furukawa (2019), their two-stage architecture combined a convolutional neural network (CNN) with integer programming to construct the graph. However, the model struggles when handling scenarios such as missed corners or curved buildings.

ConvMPN (Zhang, Nauata, & Furukawa, 2019) introduced a solution utilizing a message-passing graph neural network, which is applied to pre-detected corner candidates. Despite its effectiveness, it suffers from high memory requirements and constraints on the number of corner candidates.

Recently, Zhao et al. (Zhao, Persello, Lv, & Stein, 2023) proposed RSGNN, integrating a multi-task learning module for primitives extraction and matching alongside a graph neural network. Similarly, the attention-based model HEAT (J. Chen et al., 2021) was utilized in this domain. Both RSGNN and HEAT require substantial labeled data and encounter challenges when applied to larger or more intricate buildings.

Two additional models have recently been proposed: Weixiao Gao and Stoter (2023), employing unsupervised line detection, and Lussange, Yu, Tarabalka, and Lafarge (2023), utilizing two Mask R-CNNs (He et al., 2017) for integrated segmentation and roof-to-ground height inference.

The more recent models were not chosen for use in this study for different reasons. Weixiao Gao and Stoter (2023) is unsupervised and therefore unable to leverage the large amount of labeled data accessible in the SFKB databases. The Lussange et al. (2023) model uses an integrated end-to-end detection and 3D reconstruction approach, employing a fundamentally different method for 3D

reconstruction compared to the one outlined in this paper. Hence, HEAT is deemed more suitable for this particular dataset.

3 Theory

The following section details the theoretical foundation of the thesis. The central subject is the HEAT model (J. Chen et al., 2021) and the concepts it is built upon, which is covered by sections 3.1 - 3.4. HEAT leverages transformer-based object detection so the following will provide some background on transformers, as well as later developments of the transformer architecture which are related to the HEAT model. Specifically, HEAT utilizes deformable-DETR modules (Zhu et al., 2020), which are a development of transformer-based object detection introduced by the DETR model (Carion et al., 2020). DETR is again an adaptation of the vision transformer (ViT) (Dosovitskiy et al., 2020) for object detection tasks, so features relevant to all these models will be presented in the following.

Section 3.5 contains the technical details on the camera transformation used to convert 3D UTM representations of the SFKB data to the 2D image frames, and the multi-view triangulation used for the reverse transformations.

Section 3.6 contains a short overview of some of the terminology used when discussing graphs. This is a somewhat tangential subject to the subject of this thesis, but the terminology is used to describe features of building representations.

3.1 Transformers

Transformers, initially introduced in the paper "Attention is all you need" (Vaswani et al., 2017), revolutionized machine translation and have since become the standard in sequence-to-sequence tasks. The key innovation was the direct application of the attention mechanism, avoiding recurrence entirely in sequence-to-sequence tasks. The attention mechanism has been shown to have applications far beyond the original scope of the paper, and adaptations have extended to tasks like segmentation (Dosovitskiy et al., 2020) and object detection (Carion et al., 2020) among others.

In sequence-to-sequence tasks, the attention mechanism enables the model to focus on relevant parts of the input sequence, while aggregating information from the whole input sequence. There are two types of attention: self-attention and cross-attention. Self-attention is used in encoding to create a meaningful representation of the input, considering only the current input. Cross-attention is usually utilized in a decoder to reason over a new sequence using previously encoded representations. In self-attention, the model assumes the input is a 2D matrix (ignoring batching), such as an n x m matrix representing a sentence of length m with a word embedding of length n.
The transformer does not process input sequentially, so for tasks where order matters, positional encoding is used. This involves applying a periodic trigonometric function with a position-dependent frequency to the input sequence, representing positional relations. Next, the input passes through three linear networks in parallel to generate Query (Q), Key (K), and Value (V) matrices. The attention weight matrix is computed by

$$W(Q, K, V) = \text{Softmax}\left(\frac{QK^T}{d_k}\right).$$

The operation is illustrated in the left image of figure 1. Essentially, it calculates a similarity score between two representations of the input data (Q and K), scales it, and generates a matrix representing the amount of attention to be applied to each input pairing. This attention matrix is then multiplied with the value matrix to extract a salient feature representation,

$$A = WV.$$

In multi-head self-attention, the attention mechanism operates simultaneously on different learned linear projections of Q, K, and V. The resulting outputs are concatenated and reprojected to match the desired output shape, enabling the model to process information in varying representation subspaces.
In cross-attention, the Q and K matrices are derived from a separate input than the V matrix. The attention weight matrix is in this way computed based on prior information and used to reason over

new information contained in the V matrix. For instance, in translation tasks, the Q and K matrices are typically computed using the entire sentence, while the V matrix is computed word by word. This setup allows the model to process each new word within the context of the entire sentence.

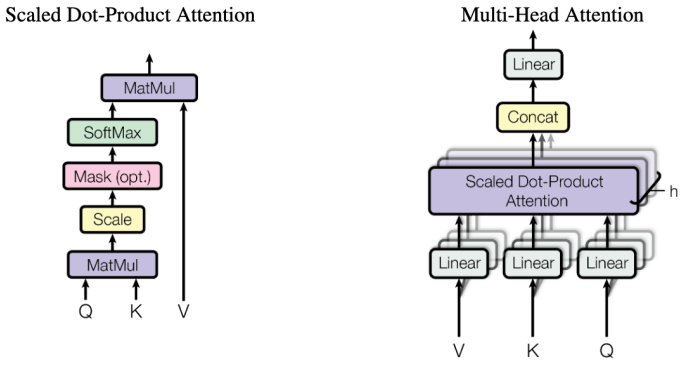


Figure 1: *Self-attention block from (Vaswani et al., 2017)*

3.2 Vision Transformers

Vision Transformer (ViT) applies a transformer directly to images with minimal modifications to the model architecture (Dosovitskiy et al., 2020). It utilizes the same encoder as the original transformer but incorporates a straightforward image embedding technique to represent image data in a sequential format suitable for processing by the transformer (figure 2). Image patches are extracted and flattened using a trainable linear projection with a fixed length. A positional embedding is then added to the tokens, and they are sent to the transformer encoder, which is identical to the one described in (Vaswani et al., 2017). The orange parts and extra embedding in figure 2 are specifically used for the classification task and are not relevant for this thesis.

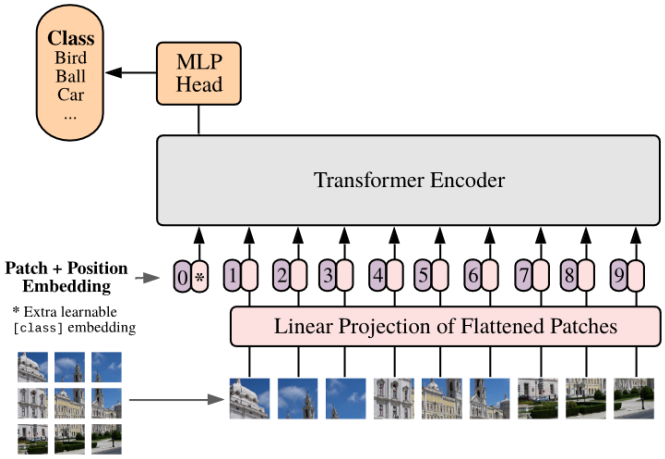


Figure 2: *The vision transformer (Dosovitskiy et al., 2020). The extra learnable embedding and MLP head shown in the figure are not relevant in this context, as they are specifically used for classification.*

3.3 Deformable DETR

Deformable DETR, an object detection transformer derived from DETR (DEtection TRansformer) (Carion et al., 2020), introduced a new attention module relevant to the HEAT model. The specifics of the DETR model are not pertinent here, as its main contribution lies in constructing a cost function for object detection. However, the deformable decoder introduced in Zhu et al. ((Zhu et al., 2020)), is used in the HEAT model. Aspects of the model related to object detection will not be addressed here.

The deformable attention module samples keys at a small set of points near a reference point, rather than from entire feature map. This approach enables processing of higher-resolution images and reasoning over smaller image features, while also reducing training time compared to previous models. In earlier cross-attention modules with global attention, a behavior of progressively refining to attend sparsely to more specific areas further along in the training period is observed. By focusing on specific areas at initialization, this process is partially completed before attention is applied, potentially improving convergence time.

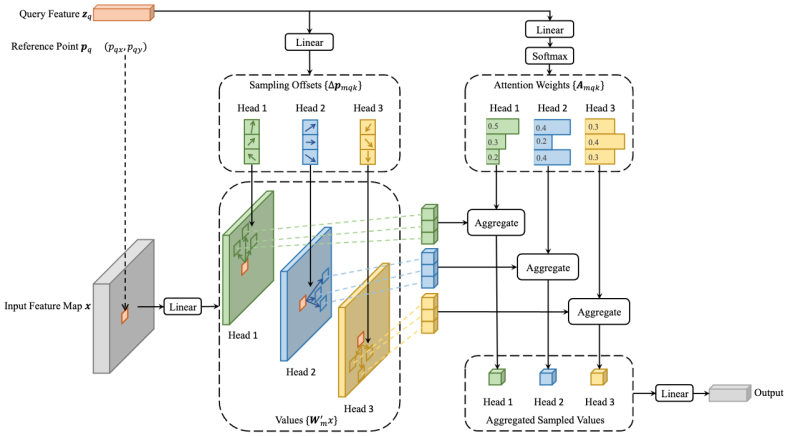


Figure 3: *Deformable attention module (Zhu et al., 2020)*

Figure 3 depicts the deformable attention module. Alongside the feature map, it takes a fixed set of reference points as input, generated upstream for instance by the HEAT model as described later. A query matrix is generated for each reference point, as in the regular transformer. A set number of sampling offsets is generated for each attention head. Attention weights at these offsets are extracted, normalized, and aggregated with the reprojected query features. The query reprojection is a linear transformation followed by a softmax layer.

Deformable-DETR can be adapted to process multi-scale feature maps. The deformable attention feature with a multi scale feature map input $\{\mathbf{x}^l\}$ with depth L is described by

$$\text{MSDeformAttn}(\mathbf{p}_q, \{\mathbf{x}^l\}) = \sum_{m=1}^{M} \mathbf{W}_m \left[\sum_{l=1}^{L} \sum_{k=1}^{K} A_{mlqk} \cdot \mathbf{W}'_m \mathbf{x}^l(\phi_l(\hat{\mathbf{p}}_q) + \Delta \mathbf{p}_{mlqk}) \right]. \tag{1}$$

m, q, and k represent the indices for attention heads, queries, and sampling points. $\mathbf{W}'_m$ and $\mathbf{W}_m$ are weight matrices learned during training. A_{mlqk} and $\Delta \mathbf{p}_{mlqk}$ denote sampling offsets and attention weights at the sampling point. The reference point coordinate is normalized to 1 and denoted $\hat{\mathbf{p}}_q$. This coordinate is then processed through a function ϕ_l, which rescales it to the feature map level. $\phi_l(\hat{\mathbf{p}}_q)$ represents a reference point location, while $\Delta \mathbf{p}_{mlqk}$ signifies an offset vector location, so

$$\mathbf{x}^l(\phi_l(\hat{\mathbf{p}}_q) + \Delta \mathbf{p}_{mlqk})$$

is a set of points from the l-level feature map x^l sampled at offsets from a reference point. Regular multi-head attention is given by

$$MHA(\mathbf{x}) = \sum_{m=1}^{M} \mathbf{W}_m \left[\sum_{k \in \Omega_k} A_{mqk} \cdot \mathbf{W}'_m \mathbf{x}_k \right].$$

Note that the inner sum iterates over all sampling points Ω_k, and the trainable weights $\mathbf{W}'_m$ are directly multiplied with the input map. Compare this to equation 1 where the inner sum is over a sample from Ω_k for each feature level, applying the weight map to these samples, rather than to the entire set of input features.

3.4 HEAT: Holistic Edge Attention Transformer for Structured Reconstruction

HEAT (Holistic Edge Attention Transformer for Structured Reconstruction) is an attention-based network designed to extract a planar graph from an image of a building to depict its underlying geometric structure (J. Chen et al., 2021) (figure 4). HEAT requires a corner detector at its head, and subsequent modules are trained end-to-end, allowing for joint training with the corner detector. The model relies on two core innovations.

Firstly, it employs two parallel transformer decoders: one decoder processes both image and geometric features, while the other processes only geometric features. "Geometric features" refer to elements of the graph representation, such as distances between nodes and node positions. This dual-decoder approach ensures that the model incorporates both visual and structural information into its reasoning.

Secondly, it integrates deformable attention transformers to facilitate the processing of higher-resolution images within reasonable training times.

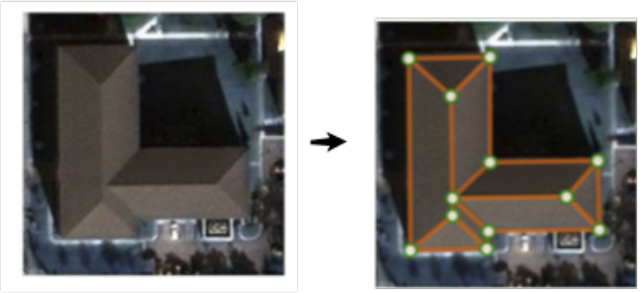


Figure 4: *Illustration of the graph inference performed by a trained HEAT model (J. Chen et al., 2021).*

3.4.1 Model architecture

The model structure is shown in figure 5. The model consists of three modules performing distinct tasks: 1) Edge node initialization, 2) fusion of edge image features and edge filtering, and 3) holistic structural reasoning using two transformer decoders with shared weights. The specifics of these modules will be discussed in the subsequent sections. In the original paper, corner detection is performed by an adaptation of the HEAT architecture, which will be addressed towards the end of the section. In the following, any matrices labeled with a subscripted or superscripted large M are trainable by backpropagation.

1) Edge node initialization: Initially, the model detects corner candidates using some corner detector. Each possible pair of corners forms a candidate edge, which is then encoded and used as

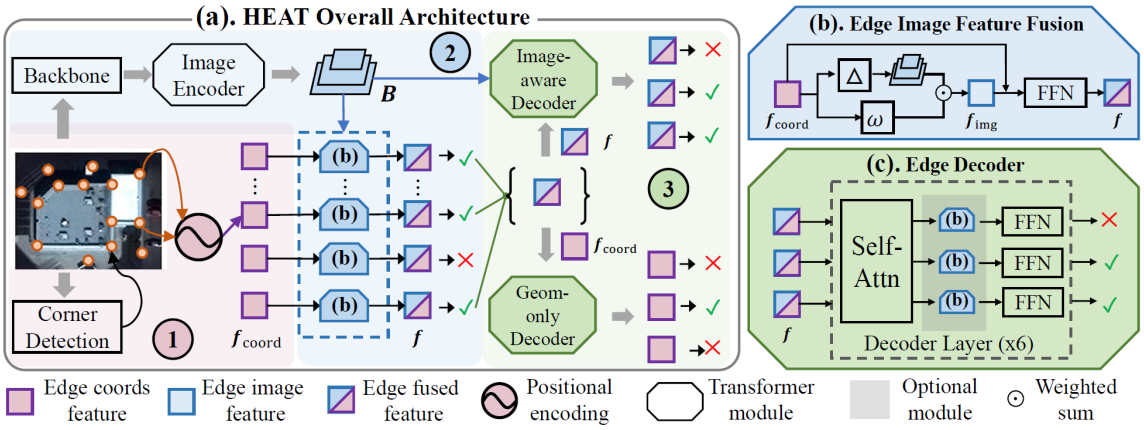


Figure 5: *Overview of the HEAT model (J. Chen et al., 2021)*

nodes in the transformer. For two corners with coordinates (e_1^x, e_1^y) and (e_2^x, e_2^y), the nodes f_{coord} are initialized with the positional encoding

$$f_{\text{coord}} = M_{\text{coord}} z^T,$$

where

$$z = \gamma(e_1^x) \oplus \gamma(e_1^y) \oplus \gamma(e_2^x) \oplus \gamma(e_2^y),$$

and $\oplus$ denotes concatenation. The γ-function is a frequency based positional encoding,

$$\gamma(t) = (\sin(\omega_0 t), \cos(\omega_0 t), ..., \sin(\omega_{31} t), \cos(\omega_{31} t)),$$
$$\omega_i = (1/10000)^{2i/32} \quad , \quad i = 0, 1, ..., 31.$$

The matrix M_{coord} is of dimension 256×256, and $z \in \mathbb{R}^{256}$.

2) Image feature fusion and edge filtering: The objective of this module (depicted in figure 5b) is to extract image features in the vicinity of edge candidate locations within the image. These features are used to filter out bad candidates and are also forwarded through the network for inference purposes. Using a ResNet [7] (He et al., 2016) backbone, three image feature maps B^l, where $l = 1, 2, 3$, are extracted at various scales (forming a feature pyramid) from the input image. These feature maps then undergo processing via a deformable-DETR transformer encoder (Zhu et al., 2020) to generate the image feature representation.

$$f_{\text{img}} = \sum_{l=1}^{3} \sum_{i=1}^{4} w^l(i) \left[M_{\text{img}}^l B^l \left(\frac{e_1 + e_2}{2^{l+2}} + \frac{\Delta^l(i)}{2^{l+1}} \right) \right]. \tag{2}$$

Here,

$$\Delta^l = M_{\text{loc}}^l f_{\text{coord}},$$

contains four 2D sampling offset w.r.t to the image center at feature scale level l, and

$$w^l = \text{softmax}(M_{\text{agg}}^l f_{\text{coord}})$$

comprises the attention weights for level l. Note the resemblance of equation 2 to the deformable attention mechanism (equation 1) in section 4.3. The scaled reference point corresponds to the scaled

[7] ResNet, or Residual Network, is a type of convolutional network architecture which facilitates much deeper networks by using "skip connections", where the input to a sequence of layers is added to the output (He, Zhang, Ren, & Sun, 2016).

coordinates of corner candidates $(e_1 + e_2)/(2^{l+2})$, while the scaled sampling offsets correspond to Δ^l. The fused feature vector is obtained by

$$f = \text{FFN}(\text{Add and norm}(f_{\text{img}}, f_{\text{coord}})).$$

Bad edge candidates are filtered out by a 2-layer MLP with a sigmoid output activation.

3) Holistic edge decoders: The edge decoder structure is depicted in figure 5 c). The decoders share weights, but the geometry-only decoder lacks the supplementary image feature fusion modules, and the two encoders receive separate inputs. Each decoder consists of six layers featuring 8-way multi-head self-attention and feed-forward networks. The image-aware decoder receives the fused feature maps f, whereas the geometry-only decoder receives only the coordinate feature map f_{coord}.

Corner detection

The edge node initialization module receives corner candidates represented as 2D point coordinates. Any corner detector could be employed, but J. Chen et al. (2021) utilizes a modified version of the HEAT architecture up to the edge-filtering module (figure 6). The authors of the paper found no

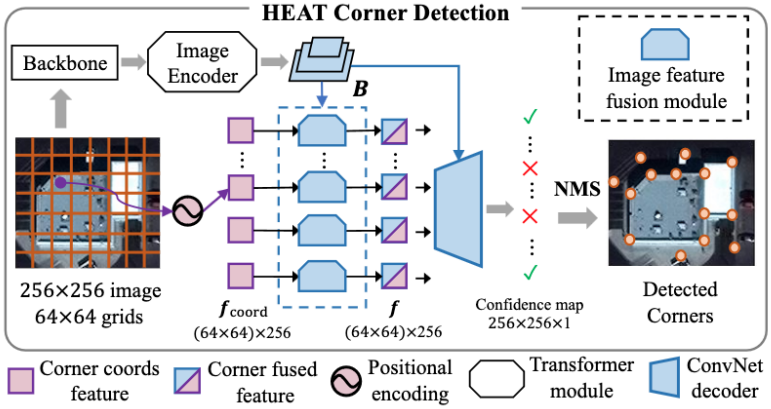


Figure 6: *HEAT corner detector*

performance improvement with self-attention, so it was omitted. The image is divided into 4×4 frames, each serving as a node. An MLP in front of the positional encoder processes the 16-pixel slices to generate a single number f_{coord}. The Image feature fusion module operates as described in 4.4.1.2). The fused features are passed through a convolutional network, producing a confidence map with the same dimensions as the input image. Non-maximum suppression is applied to the ConvNet output to generate corner candidates. The corner detector shares the ResNet backbone with the full HEAT model, while the rest of the model is jointly trained.

3.4.2 Training

J. Chen et al. (2021) adopt a masked training approach inspired by the large language model BERT (Devlin, Chang, Lee, & Toutanova, 2019) to encourage learning of structural patterns. Edge training labels are assigned truth values: True (T) or False (F), while a subset of edge labels are randomly designated as unknown (U). During training, the model is tasked with inferring the labels of these unknown samples. In practice, the truth values are encoded as a one-hot vector, processed through a linear layer, concatenated with f_{coord}, and then downscaled to a length of 256 using another layer.

3.4.3 Iterative inference

The truth values are also employed during testing. An iterative process is employed for inference where predictions are sequentially updated from the most confident to the least over three iterations utilizing a decoder processing the image content. Initially, all items are labeled as (U), then updated in the second iteration based on a confidence prediction score, with edges below 0.01 labeled as (F) and those above 0.9 as (T), while retaining (U) for others, and finally, in the last iteration, a confidence threshold of 0.5 is applied for final predictions. This process facilitates the refinement of predictions over multiple passes to potentially enhance accuracy.

3.4.4 Loss function

The cost function used in training HEAT is a weighted sum of four binary cross entropy (BCE) losses,

$$L_{\text{tot}} = L_{\text{EF}} + L_{\text{IAD}} + L_{\text{GOD}} + \alpha L_{\text{CD}}. \tag{3}$$

The first three terms in the sum are edge detection losses for the edge-filtering (EF), image-aware decoder (IAD), and geometry-only decoder (GOD) respectively. The final term is the weighted corner detection (CD) loss, where α is a weight parameter to allow adjusting the impact on the gradients. The cross-entropy loss is defined as

$$L = \frac{1}{N}\sum_{i=1}^{N} \left[y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i) \right],$$

where N is the number of samples (in this case $N = 256^2$ as a prediction is made on every pixel in an image), y_i is the ground truth label, and $\hat{y}_i$ is the prediction on sample i.

3.5 Camera transformations

The SFKB data is transformed from UTM coordinates to the image coordinate system using a projective camera transform (Hartley & Zisserman, 2004). The camera transforms maps of 3D points in the world coordinate system to 2D points in the image plane. The transformation is performed by multiplying the 3×4 camera projection matrix P with the homogeneous[8] world coordinate vector. Let $\mathbf{x} = (x, y, z, 1)^T$ be a world coordinate point in the view of a camera described by the camera matrix P. Let $\mathbf{x}' = (x', y', z')$ be the result of the projective transformation,

$$\mathbf{x}' = P\mathbf{x}.$$

The image representation of $\mathbf{x}$ is then given by

$$\mathbf{x}_c = \left(\frac{x'}{z'}, \frac{y'}{z'} \right)^T.$$

3.5.1 The Camera matrix

The camera matrix is given by

$$P = KE, \tag{4}$$

where K is the intrinsic matrix that describes the camera's internal characteristics, and E is the extrinsic matrix that encodes the rotation and position of the camera. The intrinsic matrix is by

$$K = \begin{bmatrix} f_x & s & x_{\text{ppa}} & 0 \\ 0 & f_y & y_{\text{ppa}} & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{5}$$

[8] Homogeneous coordinates are an extension to Cartesian coordinates which simply append an additional coordinate to the Cartesian vector (Hartley & Zisserman, 2004). It is typical to set the value of the extra coordinate to 1, and that convention is used in this paper. Homogeneous coordinates facilitate many common operations in computer vision, including the projective transform discussed here.

Here, f_x and f_y are representations of the focal length[9] in terms of pixel dimensions. There are two values because the pixel width and height are not necessarily equal. s is a skew factor, based on the characteristics of the sensor, and x_{ppa} and y_{ppa} constitute the coordinate of the principal point[10] in image coordinates.
The extrinsic matrix defines the rotation and translation to be applied during the camera transform without taking into account the intrinsic camera characteristics. In regular Cartesian coordinates, the extrinsic transformation of a point $\mathbf{x}$ in world coordinates can be written in terms of a translation and rotation,

$$\mathbf{x}' = R(\mathbf{x} - \mathbf{x}_c),$$

where $\mathbf{x}_c$ is the position of the camera, and R is a regular 3×3 right-handed rotation matrix

$$R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos r_x & -\sin r_x \\ 0 & \sin r_x & \cos r_x \end{bmatrix} \begin{bmatrix} \cos r_y & 0 & \sin r_y \\ 0 & 1 & 0 \\ -\sin r_y & 0 & \cos r_y \end{bmatrix} \begin{bmatrix} \cos r_z & -\sin r_z & 0 \\ \sin r_z & \cos r_z & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

and r_x, r_y and r_z are rotations about the x, y and z axes respectively (Hartley & Zisserman, 2004). Equivalently, in homogeneous coordinates,

$$\begin{aligned} \mathbf{x}' &= \begin{bmatrix} & R & & -R\mathbf{x}_c \\ 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{x} \\ &= E\mathbf{x}. \end{aligned} \tag{6}$$

3.5.2 Multi-view triangulation

Multi-view triangulation refers to the determination of a 3D point in world coordinates given a set of images of the point from different viewpoints (Hartley & Zisserman, 2004). Let $\mathbf{x}$ be the 3D world coordinate point in homogeneous coordinates. The point is represented in n images by the points $\mathbf{x}'_i$, $i = 1, 2, ..., n$, where

$$\mathbf{x}'_i = P_i\mathbf{x}, \tag{7}$$

and P_i is the camera matrix for the i^{th} image. The multi-view triangulation problem then consists of inferring $\mathbf{x}$ when $\mathbf{x}'_i$ and P_i are known. The simplest method for performing multi-view triangulation is using the direct linear transformation (DLT). There are more advanced methods, but the DLT performs well when the noise is reasonably small considering its simplicity, and has a smaller run time (J. Chen et al., 2020).

Equation 7 is equivalent to

$$\mathbf{x}'_i \times (P_i\mathbf{x}) = 0,$$

which again is equivalent to

$$\begin{aligned} y'_i(P_i^3\mathbf{x}) - P_i^2\mathbf{x} &= 0 \\ P_i^1\mathbf{x} - y'_i(P_i^3\mathbf{x}) &= 0 \\ x'_i(P_i^2\mathbf{x}) - y'_i(P_i^1\mathbf{x}) &= 0, \end{aligned}$$

where $\mathbf{x}'_i = (x'_i, y'_i, 1)^T$ and P_i^k is the k^{th} row of P_i. The third equation can be disregarded as it is a linear combination of the first two. Using the first two equations for every image, a $2n \times 4$ matrix

[9] The focal length is the distance from the point where rays of light converge after passing through the camera lens to the imaging surface.
[10] The principal point is the point in the image plane where it is intersected by the optical axis.

A which fulfills $A\mathbf{x} = 0$ can be constructed by

$$A = \begin{bmatrix} x'_1 P_1^3 - P_1^1 \\ y'_1 P_1^3 - P_1^2 \\ x'_2 P_2^3 - P_2^1 \\ y'_2 P_2^3 - P_2^2 \\ \vdots \\ x'_n P_n^3 - P_n^1 \\ y'_n P_n^3 - P_n^2 \end{bmatrix} \tag{8}$$

In the presence of noise, $A\mathbf{x} = \mathbf{w}$, which constitutes an over-determined least-squares problem: Finding $\mathbf{x}$ such that $\mathbf{w}$ is minimized. This can be solved using the singular value decomposition (SVD). Let $A = USV^T$ be the SVD of A. The minimum value of $\mathbf{w}$ is obtained by choosing $\mathbf{x}$ as the last column of V^T (Corless & Fillion, 2013). If $\mathbf{x} = (x, y, z, h)$, the 3D world coordinate point in regular Cartesian coordinates is

$$\mathbf{x}_{\text{UTM}} = \left(\frac{x}{h}, \frac{y}{h}, \frac{z}{h}\right).$$

3.6 Graph terminology

A graph is a set of *nodes* connected by *edges*. The graphs used to describe buildiding roofs in this thesis are *numbered* and *undirected*: The nodes are labeled to be distinguishable from each other and the edges do not point to a specific node.

The concept of a *subgraph* will be used throughout this thesis. A subgraph a subset of a larger graph. A *simple cycle* in a graph is a sequence of nodes through which the graph can be traversed, where only the first and last nodes are equal. A *chordless* cycle is a cycle such that no two nodes in the cycle are connected by an edge that is not a part of the cycle.

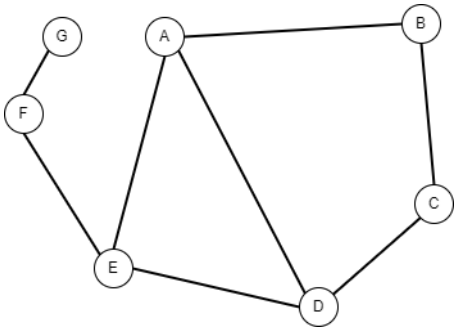


Figure 7: *Example of an undirected, numbered graph*

Figure 7 shows an example of a numbered and undirected graph, which contains the graph features defined above. The subgraphs $A \to B \to C \to D \to A$ and $A \to D \to E \to A$ are chordless simple cycles. $A \to B \to C \to D \to E \to A$ is cyclic but not chordless. $E \to F \to G \to$ is non-cyclic.

4 Methods

The development of the building structure extraction process consists of three main stages:

1. The **preprocessing stage**, in which image and label data is prepared for being used in the HEAT model.
2. The **training stage**, where the HEAT model is trained on the data.
3. The **Inference and reconstruction stage**, where HEAT is applied to a sample of the test data, and the results are transformed back to 3D coordinates.

These steps and the datasets used in the process are described in detail in the following.

4.1 Description of the data

This project uses two primary data sources: Aerial images from missions performed by external partners of Kartverket (as input data), and 3D graph representations of every registered building in the country from SFKB (as label data).

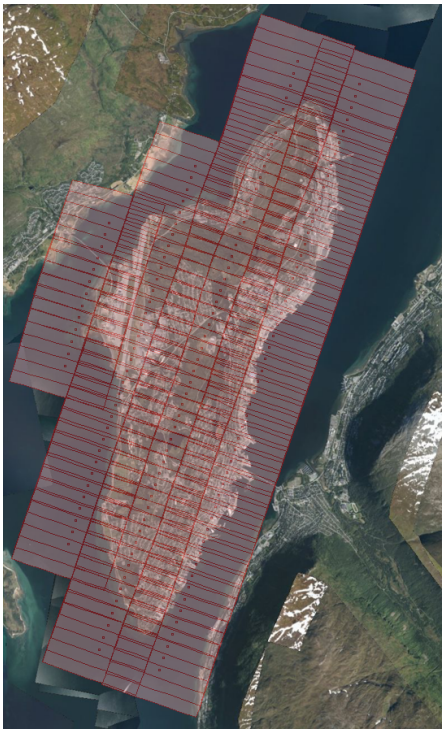


Figure 8: *Aerial coverage of images over Tromsø.*

4.1.1 Aerial images

The aerial images come from a mission executed in 2021, depicting the Tromsø island. Several such missions are performed every year around the country, with more populated areas covered more frequently. Any of the recent missions could have been chosen as the interest area for this study, and the Tromsø area includes a sufficient amount of buildings. The images were taken by an optical camera mounted on an aircraft. The dataset contains 151 RGB images of size 26460×17004, overlapping by approximately 80% in the in-flight direction (figure 8), and also includes detailed metadata including calibration metrics, position, and direction of the camera. The images and associated metadata are of very high quality and contain very little distortion. These images are un-rectified, i.e. they are not orthorectified , which means that the relationships between objects in the image and their real word position in relation to the camera can be described by the equations detailed in section 3.5.

4.1.2 SFKB data

The SFKB data is more detailed than what the HEAT model is meant to process (figure 9). A tailor-made SQL routine has been used to retrieve the relevant data and to prune data points that do not approximately conform to Level of detail (LoD) 2 as defined by Biljecki, Ledoux, and Stoter (2016). This LoD incorporates the overall structure of the roof, and implicitly represents the wall elements by drawing vertical lines down from exterior roof vertices, but excludes more detailed features such as windows and different classes of edges. Only the graph describing the roof is needed, as the wall elements can be derived. The house representations are contained in JSON files with a dictionary structure where the keys contain the nodes and the items associated with every key represent the edges connected to the node. This representation conforms to the format expected by the HEAT model.

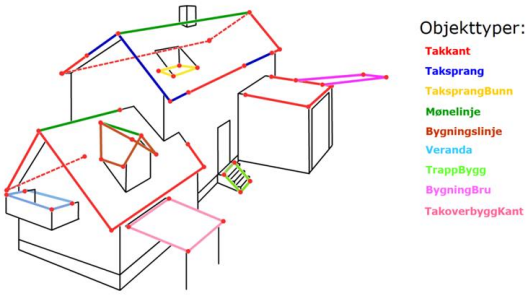


Figure 9: *Level of detail in the SFKB database*

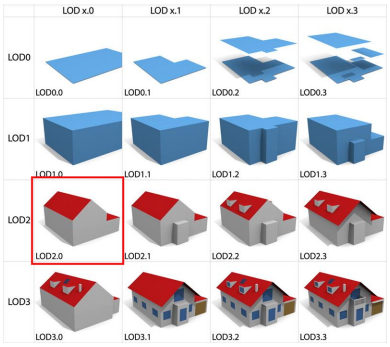


Figure 10: *Standard classification of level of detail in representations of buildings (Biljecki et al., 2016). LoD 2.0, outlined by a red box, is used in this study.*

4.2 Data Preprocessing

HEAT requires three sources of data for training: 256 × 256 images of the buildings whose structures it is supposed to infer, graph representations of the building structures to be used as training labels, and a separate representation of just the graph nodes to be used as training labels for the corner detector (see section 3.4.1 on corner detection). The preprocessing pipeline is designed to extract these from the aerial images, image metadata, and the SFKB data, and transform the SFKB graph representations into the coordinate system of the aerial images so that the position of the graphs corresponds to the position of the building they represent.

The pipeline is illustrated in figure 11. The coverage area for each image (figure 8) is defined by corner coordinates in 2D UTM coordinates. House graphs extracted from SFKB are determined to be inside or outside the coverage area using a simple geometric test. Only buildings within a certain distance from the edge of the image are included to avoid extracting partial buildings at the edge of the coverage area.

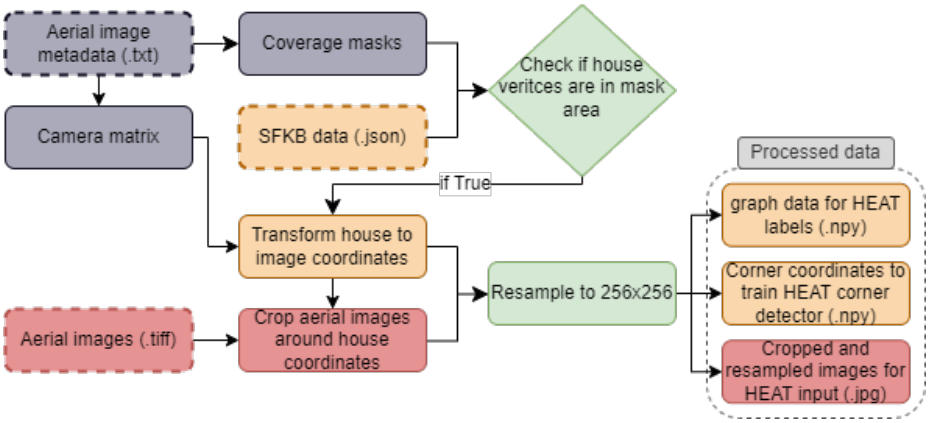


Figure 11: *Flowchart of the preprocessing pipeline. The yellow elements refer to operations relating to the graph data, blue refers to the image metadata, red denotes image data, and green refers to operations using several sources. Boxes with dashed outlines indicate input data.*

The camera position $\mathbf{x}_c$ and rotational elements r_x,r_y and r_z used to construct the extrinsic matrix (equation 6), and the focal length f and principal point $(x_{\text{ppa}}, y_{\text{ppa}})$ used to construct the

intrinsic matrix (equation 5) are all extracted from the metadata. The skew s is zero and the camera pixels are square so $f_x = f_y = f$. The rotational elements are converted from degrees to radians. Using these parameters, the camera matrix (equation 4) can be constructed for each image. Image coordinates are obtained by applying the camera transform to the homogenous UTM coordinates. Using the transform, image coordinate representations of the building graphs w.r.t. every aerial image in which they appear can be constructed. To avoid letting the model see multiple representations of a single house during training, only one representation is used to train the HEAT model, so a random one is chosen for every house.

To extract the image containing the building, a bounding box around each transformed building graph is constructed in the image plane, and the image it encloses is clipped from the full aerial photo(figure 12). The coordinates of the corners of the bounding box enclosing the graph in image coordinates are:

$$
\begin{aligned}
b^x_{\min} &= x_{\min} - d_{\text{pad}} - d^x_{\text{sq}} \\
b^x_{\max} &= x_{\max} + d_{\text{pad}} + d^x_{\text{sq}} \\
b^y_{\min} &= y_{\min} - d_{\text{pad}} - d^y_{\text{sq}} \\
b^y_{\max} &= y_{\min} + d_{\text{pad}} + d^y_{\text{sq}}.
\end{aligned}
$$

$x_{\min}$, $x_{\max}$, $y_{\min}$ and $y_{\max}$ denote the minimal and maximal coordinate elements on the building graph. d_{pad} is a constant minimal padding distance from the building corners to the edge of the image. d^x_{sq} and d^y_{sq} are the elements of $\mathbf{d}_{\text{sq}}$ which denotes extra padding applied to the dimension of the shortest picture length to ensure squareness, given by

$$
\mathbf{d}_{\text{sq}} = \begin{cases} \left(0, \frac{1}{2}(x_{\text{ext}} - y_{\text{ext}})\right)^T & \text{, if } x_{\text{ext}} > y_{\text{ext}} \\ \left(\frac{1}{2}(y_{\text{ext}} - x_{\text{ext}}), 0\right)^T & \text{, otherwise,} \end{cases}
$$

where $x_{\text{ext}} = x_{\max} - x_{\min}$ and $y_{\text{ext}} = y_{\max} - y_{\min}$.

The graph elements are transformed in the same way. Let (x, y) be the coordinate of a node on the graph in image coordinates. The coordinates in the cropped image are then given by

$$
x_{\text{cropped}} = x - x_{\min} + d_{\text{pad}} + d^x_{\text{sq}}, \tag{9}
$$

$$
y_{\text{cropped}} = y - y_{\min} + d_{\text{pad}} + d^y_{\text{sq}}. \tag{10}
$$

Finally, the image is resampled to 256×256 using linear interpolation. To retain the correct correspondence between the images and graphs, the graph x- and y- coordinates are multiplied by $256/x_{\text{ext}}$ and $256/y_{\text{ext}}$ respectively. The 256×256 images are saved as jpg files, and the graph nodes for the corner model are saved as Numpy arrays. The full graph representations are saved as dictionaries where the key is a corner node and the values are the coordinates of other nodes connected by an edge 13.

Before being passed to the HEAT model for training, Random flipping and rotation are applied for data augmentation, and Gaussian blur with $\sigma = 2$ is applied to the label map.

4.3 Model training

Before training, the HEAT model is initialized with the pre-trained weights from the J. Chen et al. (2021) model, and then it is trained with the same hyperparameters except for the learning rate scheduling rate. The original model was trained for 800 epochs, an reduced the learning rate by 25% after 600 epochs. In contrast, the model trained in this study is trained for less than 200 epochs, and so the learning rate is reduced much earlier, after 95 epochs. This schedule was set after observing a lot of oscillatory behaviors of the loss in earlier training tests. The data set consists of 256×256 images with corresponding building graphs and corner labels. The dataset is split into a training, testing, and validation set containing 80%, 15%, and 5% of the total samples, respectively. The model is trained on the training set, and the accuracy of the model on the validation set is reported. The validation set is used for regularization during training. The final model is chosen by its performance on the validation set, which should reduce the chance of overfitting on the training set.

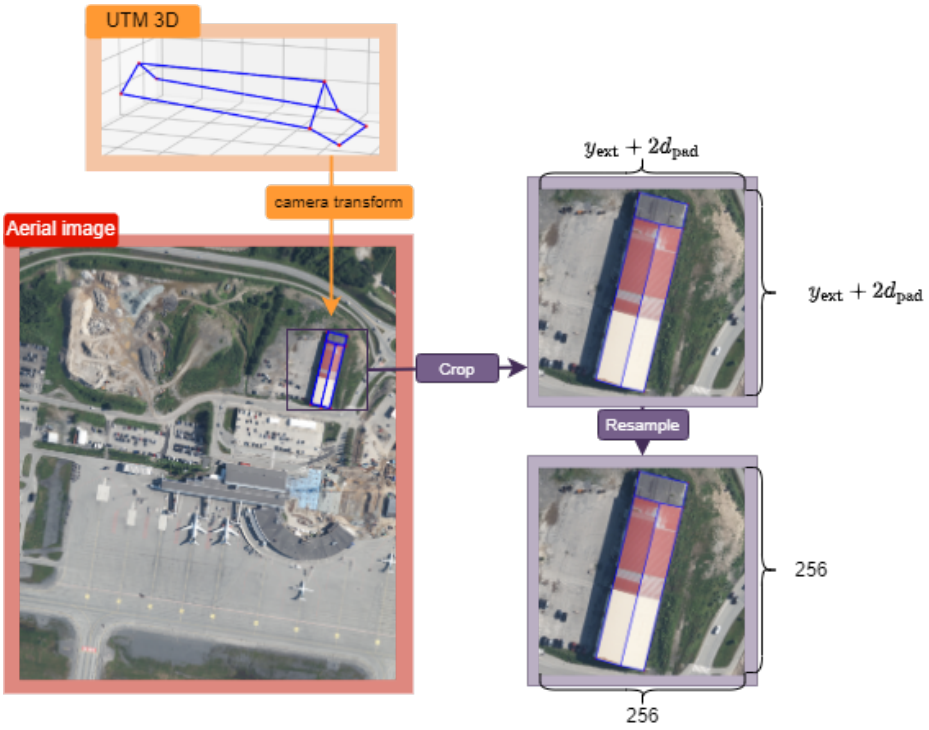


Figure 12: *Illustration of the process of extracting building graphs and associated images in the same coordinate system*

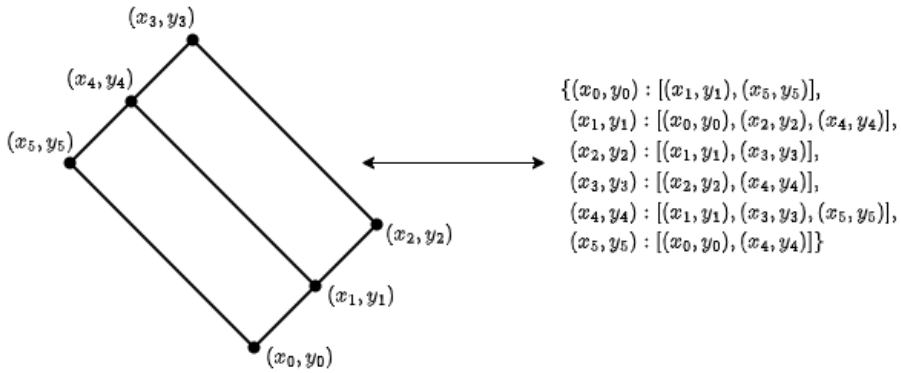


Figure 13: *Dictionary representation of 2D building graph*

4.3.1 Training metrics

During training, in addition to the four losses in equation 4.3.1, five additional metrics are reported:

- Corner detection recall on the training set
- Edge detection F1-scores for the three edge decoders on the training set
- Mean of the F1-scores for the image-aware decoder and the corner detection on the validation set. This will be referred to as the validation performance in the following.

The F1 score is defined as

$$F_1 = 2\frac{P \cdot R}{P + R},$$

where P and R are the precision and recall,

$$P = \frac{TP}{TP + FP},$$
$$R = \frac{TP}{TP + FN}.$$

TP, FP, and FN are the number of true positive, false positive, and false negative predictions respectively. These are defined for the corners, edges, and area as follows

Corners: A corner is successfully predicted (TP) if a ground-truth corner is located within an 8-pixel around the prediction. If there are multiple predictions within this radius, the nearest one is labeled as successful while the rest are labeled as false positives.

Edges: And edge detection is true positive if both corners were detected by the corner detector and the edge matches a ground-truth edge. A detected edge is labeled false positive if any of its connecting corners were not detected, regardless of whether it matches a ground truth edge.

Regions: Regions are defined as chordless cyclic (as defined in 3.6) sub-graphs within the detected building graph. The areas of any regions calculated for both the inferred graph and the ground truth graph and the Intersect over Union (IoU) (equation 11) between overlapping regions are found. An area is labeled a true positive if it has an IoU over 0.7 with a ground truth region that does not already have a positive matching with another detected area.

The area metric is only used for evaluating the model performance in the testing phase, while the corner and edge metrics are reported during training.

$$\text{IoU} = \frac{\text{Area}(A_{\text{pred}} \cap A_{\text{GT}})}{\text{Area}(A_{\text{pred}} \cup A_{\text{GT}})}, \tag{11}$$

4.4 HEAT inference and 3D reconstruction

After the HEAT model is trained, it is used for inference on the test set. In the testing phase the precision, recall, and F1-score for the corner, edge and region detection is calculated on the test set as defined in the previous section.

To test the 3D reconstruction, a small set of buildings from the test set where the graph inference was successful is selected for reconstruction. Reconstruction requires graph representations from different views of the building. Therefore, for each building, a set of five 256×256 images from each aerial image that depicts it is extracted by the method in section 4.2. The graphs inferred by HEAT are transformed back into the coordinate system of the large aerial photo by applying the inverse of the transformation in equations 9 and 10 and the resampling, i.e.

$$x = \frac{x_{\text{ext}}}{256} x_{\text{cropped}} + x_{\min} - d_{\text{pad}} - d^x_{\text{sq}},$$
$$y = \frac{y_{\text{ext}}}{256} y_{\text{cropped}} + y_{\min} - d_{\text{pad}} - y^x_{\text{sq}}.$$

The matrix A from equation 8 can then be constructed, and the system is solved to find an approximation of the 3D UTM coordinates of the building graph (figure 14).

Successful 3D reconstruction relies on precise graph inference and perfect correspondence of the overall graph structure—any missing or extra edges or nodes in a graph in the set lead to a failed reconstruction. Only buildings with identical and complete inferred graph structures in at least 2 projections are used in the 3D reconstruction. "Identical structure" means that the graphs should have the same number of nodes, and the list of connections for each node (defining the edges) should be identical under permutation. The reconstruction process is applied to 10 sets of graphs inferred by HEAT, where at least two of the inferred graphs match in structure. The accuracy of the 3D reconstruction is simply quantified by the distances of the reconstructed corners to the SFKB corners.

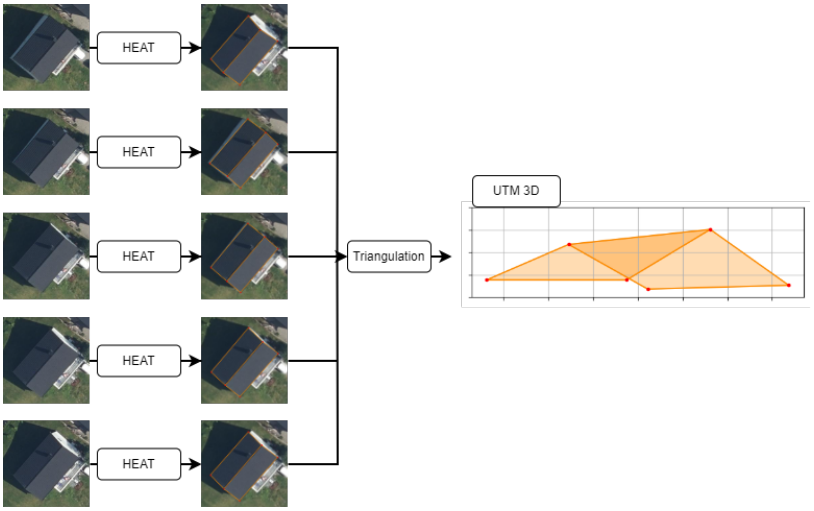


Figure 14: *Illustration of graph inference and 3D reconstruction. Each building image is clipped from a different aerial image.*

5 Results

The following presents the results of the experiments. Firstly, the results of the pre-processing is presented. This includes the coordinate transformation from UTM to image coordinates, and sample selection for the data set used to train HEAT. Secondly, details of the training process for the HEAT model is presented. Thirdly, the results of applying the trained HEAT model to the test set is presented. Finally, the results of 3D reconstruction of the HEAT inferences are presented.

5.1 Pre-processing

A total of 21376 3D building graphs within the target area were extracted from the SFKB dataset using the SQL routine detailed in Section 4.1.2. These graphs were transformed to image coordinates in the frame of each image in whose coverage area they were located. The SFKB graph data and aerial images are both accurate enough that the correspondence between the image and graph representations in image coordinates of the house correspond well in many cases (Figure 15).

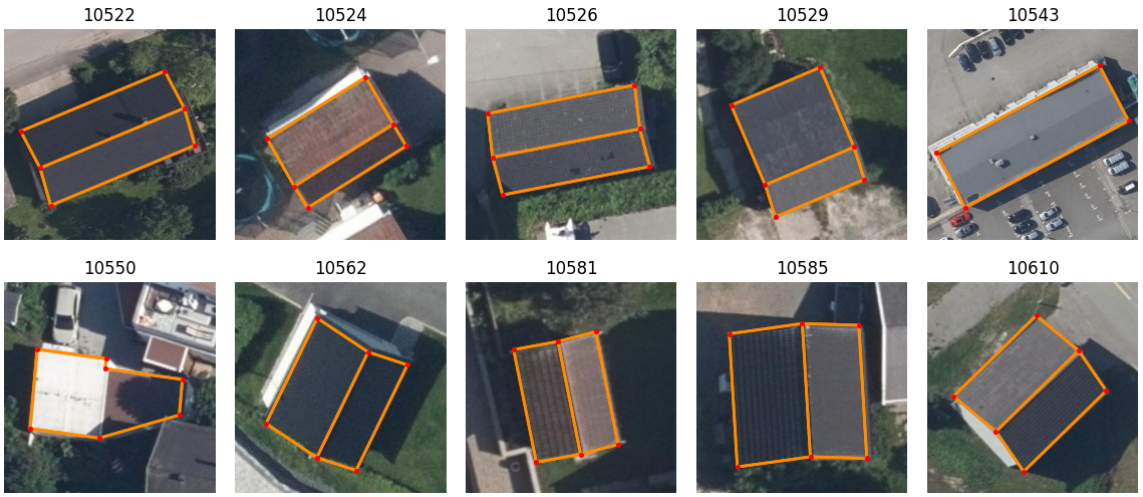


Figure 15: *Ten examples of SFKB building graph data transformed into an image coordinate system.*

However, manual inspection revealed that many nodes and, in particular, edges are missing in the extracted SFKB data. Much of the interior roof structure is often missing. Some examples of unusable graph-image pairs are shown in Figure 16. This is most prominent in more complex structures but also applies to many of the smaller buildings. Such incomplete graph representations cannot be used for training the HEAT model, and the samples that are missing elements had to be removed from the data set. Notably, since almost all larger and more complex building graphs are missing structural elements, every building with more than 10 nodes was removed from the set. The remaining 6843 graph-image combinations were analyzed manually to remove unusable data among the remaining ones. The result is a small dataset of 1635 samples, which is split into a training set of size 1308, a testing set of size 245, and a validation set of size 82.

5.2 Training the HEAT model

The HEAT model was trained over 192 epochs with a learning rate of $2 \cdot 10^{-4}$, batch size of 16, and initialized with weights from the model trained by J. Chen et al. (2021). The only change from the original paper is the rate at which weight decay is applied. In the original paper, the learning rate is reduced by 25% after 600 epochs, while in this experiment the training schedule reduces the learning rate by the same amount after 95 epochs. This is because the learning curve flattens much earlier in this training cycle than it did in the original paper, and the training duration is much shorter overall. The model was trained locally on a computer with an i7-9750H, 2.60GHz core, and

Figure 16: *Eight examples of image-graph pairs that were discarded. They exhibit several shortcomings that preclude them from being included in the data set: Missing structural elements, hidden features due to the viewing angle, poor alignment, and graph elements that do not enclose a surface (non-cyclic graph elements).*

dual GPUs (Intel UHD 630 and NVIDIA GeForce GTX 1990) with a total of 16 GB RAM using Python 3.8 and Pytorch 1.12.1.

An overview of the training metrics is shown in Figure 17. The loss curve flattens out after around 100 epochs. Some additional gains could likely be made if the training was continued as the learning curve is still diminishing, although at a low rate. The model was trained on very limited hardware resources, and could have been trained for much longer with access to better hardware. In general, the total loss is dominated by the corner detection loss, with $\alpha = 0.05$ (see Equation 4.3.1). The evolution of the metrics during training is closely correlated and follows the same trajectory with the exception of the geometry-only decoder at the start. This decoder overall receives less information (it does not see the image information at all), and does not start noticeably adapting to the data until approximately epoch 25. There is a marked drop in the loss, and correspondingly an increase in the accuracies after the learning rate is reduced after epoch 95. The maximal validation accuracy of 0.802 was reached after 187 epochs, near the end of the training session. The corner detection recall remains low, reaching a maximum of 0.37 after 185 epochs, indicating that the model often makes many clustered corner predictions around the ground-truth corners.

5.3 Inference

The results of the inference on the test set are shown in Table 1 together with the results of inference on the same set using the initial weights from J. Chen et al. (2021). As expected, the trained model sees an improvement in every metric compared to the initial model. The recall is higher than the precision in all three categories, indicating that the model in general emphasizes making more correct predictions over reducing the number of false positives. Notice that the corner prediction recall during inference (0.768) on the testing set is much higher than on the training set during training. These metric aren't comparable, as the recall during testing is calculated after edge inference, and corners that aren't attached to an edge are filtered out, leaving fewer false positive corner predictions.

Table 1: *Results from inference on the testing set using the HEAT model.*

	Corners			Edges			Regions		
	Precision	Recall	F-score	Precision	Recall	F-score	Precision	Recall	F-score
Model trained on Tromsø data	0.768	0.872	0.816	0.667	0.753	0.707	0.767	0.813	0.789
Pre-trained model	0.516	0.771	0.618	0.372	0.540	0.441	0.552	0.606	0.578

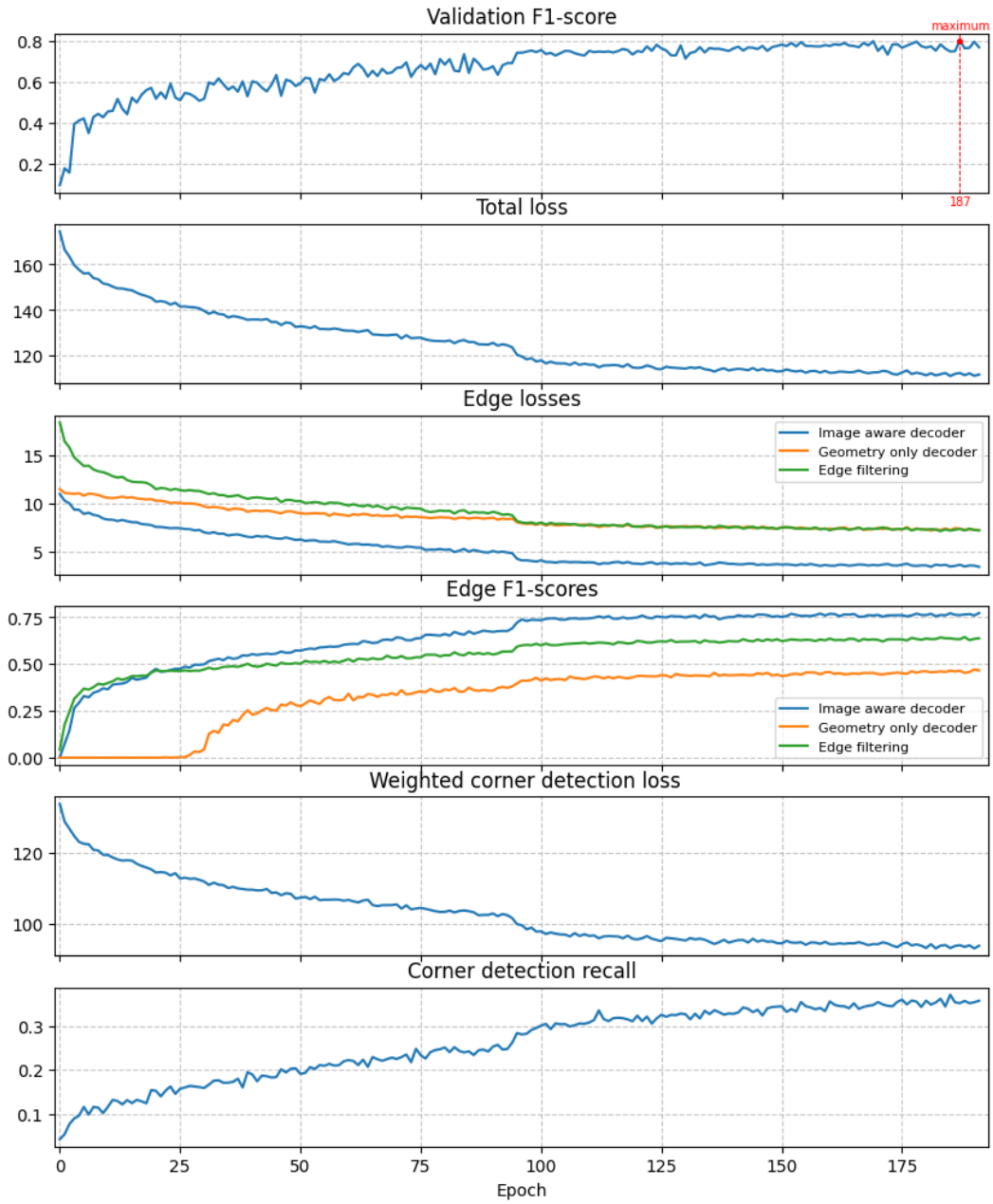


Figure 17: *Training metrics for the HEAT model on the Tromsø dataset.*

5.4 3D reconstruction

Ten buildings from the test set with successful HEAT graph inferences were chosen for reconstruction on the basis of having some degree of variation in structure, size and overall appearance (figure 18). The inferred graphs were transformed from the frame of the 256 × 256 single house images to the image frame of the full aerial photos (figure 19). Finally, the corners of the resampled graphs were used to triangulate the 3D coordinates of the inferred building graph in world coordinates (figure 20).

The mean absolute error (MAE) for the three processes; inference, resampling, and reconstruction, were recorded and are shown in table 2. The normalized errors are visualized in Figure 21. There is no clear correlation between either the size of the building or the number of images used in

Figure 18: *Five buildings used in the 3D reconstruction with HEAT structure inferences. The images of the five remaining houses can be found in Appendix 8, Figure 29. Upper panels: Source images. Bottom panels: Inferred graphs. precision and recall for edge and region prediction is shown.*

reconstruction and the 3D reconstruction error. As will be discussed in section 6, a large amount of the error likely comes from inaccurate data in the SFKB database. A standard deviation of 0.12 in the 3D reconstruction errors indicate a uniform distribution. Correlations between the error in the three representations are shown in table 3. The correlations indicate a strong correlation between adjacent processing steps, i.e. between the HEAT inference error and the resampled error and the resampled error and the 3D reconstruction error. However, the correlation is not very strong between the inference and the 3D reconstruction errors.

Table 2: *MA errors in HEAT inferred graphs, resampled graphs and reconstructed 3D graphs. The two right-most columns describe the mean and standard deviations of the errors*

House ID	11549	11898	12322	14296	16618	18412	18708	22084	29757	30647	Mean	STD
HEAT inference error	3.044	7.374	2.544	3.174	2.340	5.233	3.877	4.882	4.638	5.053	4.22	1.46
Resampled error	4.231	9.286	5.496	7.885	4.267	8.003	8.129	11.025	8.472	9.708	7.65	2.17
3D reconstruction error	0.260	0.501	0.357	0.447	0.327	0.357	0.483	0.692	0.335	0.499	0.43	0.12

Table 3: *Correlation matrix for the MA errors*

	Inference error	Resampled error	3D reconstruction error
Inference error	1.00	0.74	0.47
Resampled error	0.74	1.00	0.84
3D reconstruction error	0.47	0.84	1.00

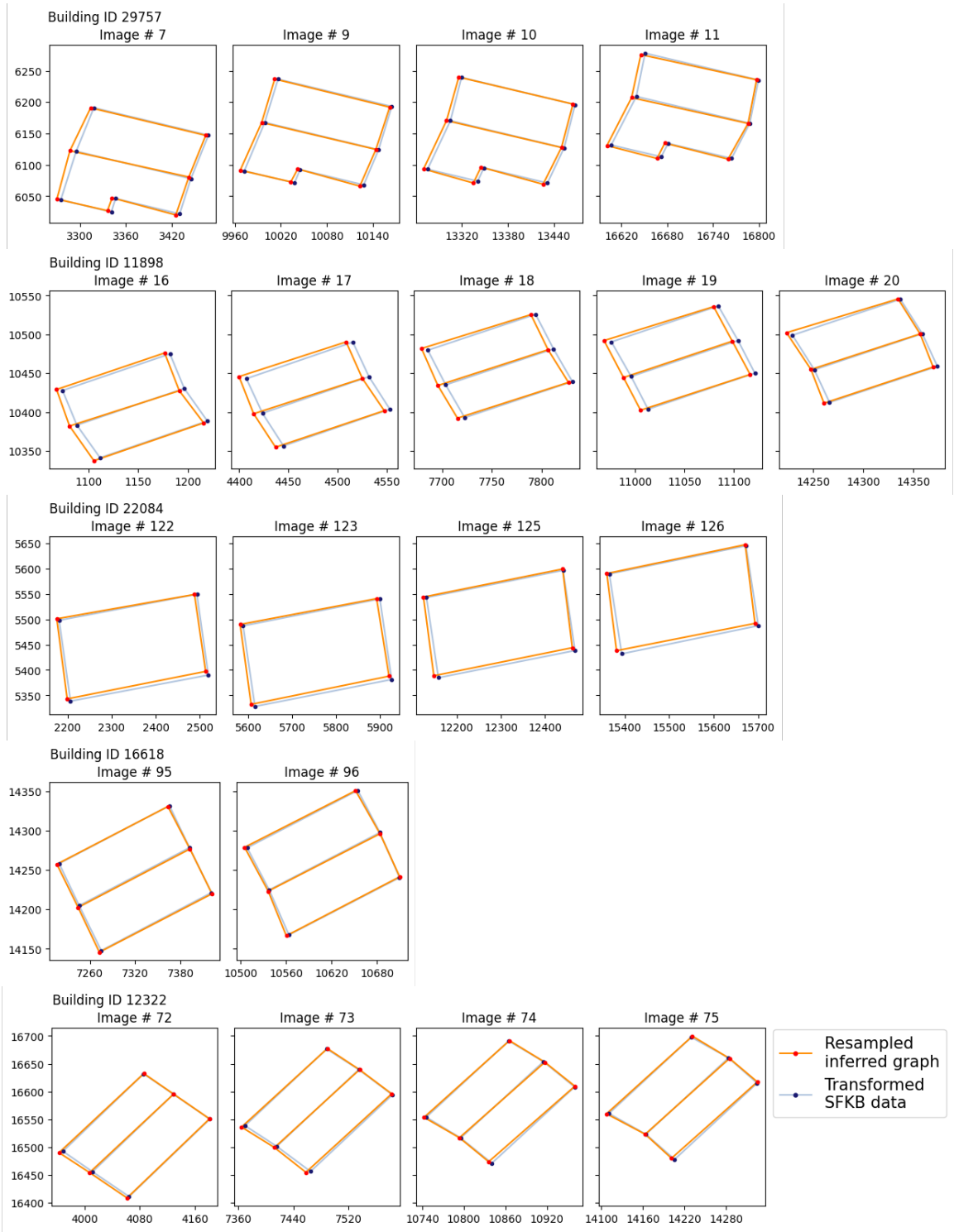


Figure 19: *Resampled graphs of the buildings from figure 18 in the large aerial image frames with the transformed label data from SFKB. The tiles of the sub-figures refer to the ID of the specific larger aerial image. Graphs of the five remaining images can be found in the Appendix (Section 8, Figure 30).*

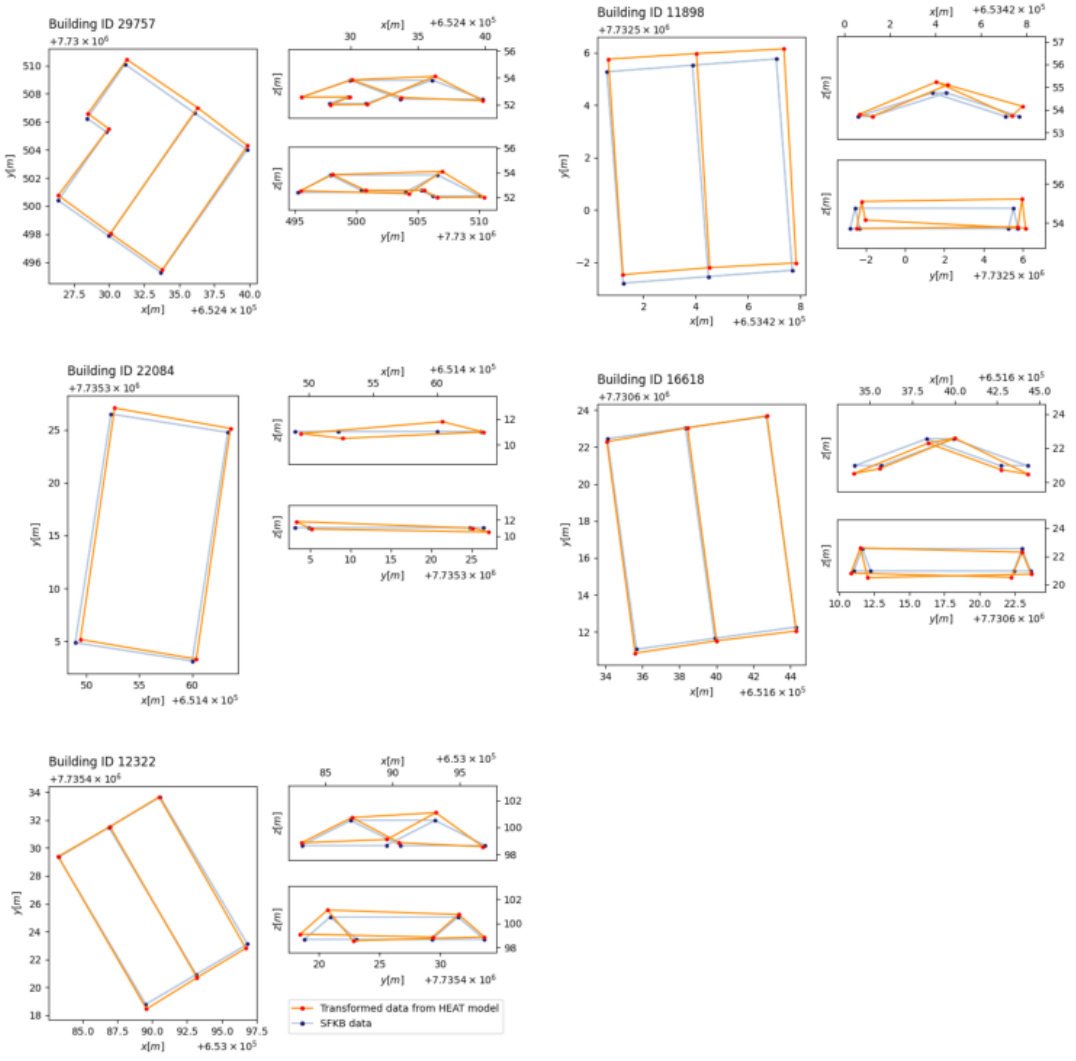


Figure 20: *3D reconstructions of the 5 buildings. The buildings are represented with one view along each of the spatial axes. 3D reconstructions of the remaining five buildings are shown in the Appendix (Section 8, Figure 30).*

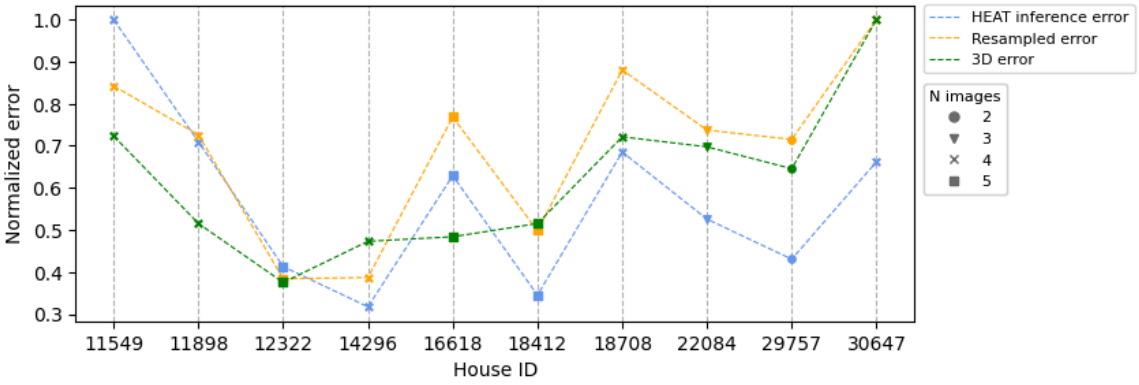


Figure 21: *Relative errors compared to the SFKB data throughout the reconstruction process. The marker styles indicate the number of views of the house that was used in the reconstruction, and the houses are ordered by size.*

6 Discussion

The results presented in the previous section are discussed in the following. First, the results about the data preparation method are examined. Secondly, the results and performance of the HEAT model and the data preparation methodology are considered. Thirdly, the 3D reconstruction process is analyzed. Finally, some suggestions for potential avenues of improvement are discussed.

6.1 Data preparation

The camera transformation to project the SFKB data to image coordinates yields good results in general. However, the end result of the transformation is still dependent on the quality of the SFKB data, which is not well suited for use as labels for training the HEAT model, as most of the extracted building graphs are missing graph features that are needed for them to be used as training label. After reducing the data set down to 1635 samples, there are still examples of minor misalignment, which makes evaluating the performance of both the graph inference and 3D reconstruction challenging.

Another issue is the fact that the resulting data set is small and, as discussed in section 5.1, excludes more complex structures. This thesis has therefore not been able to examine whether HEAT can be adapted to infer more complex roof structures. The size of the SFKB database was a large motivation for the formulation of the research questions in this thesis, and the hope was that it would enable the construction of a larger dataset than what has been used in earlier experiments. Instead, the resulting dataset is smaller than what was used in J. Chen et al. (2021), which contained 2001 samples.

6.2 Performance of the graph inference

As shown in Table 1, the fine-tuned model performs better than the model using the pre-trained weights. A Few examples of successful [11] inferences using the trained model are shown in figure 23. These examples show that HEAT can infer the roof structures of buildings with various properties, although the variety is limited by the selection process for the training set described in section 5.1. Successful inferences are made in the presence of oblique viewing angles (2, 3, 4, 5, 6), sharp shadows (4, 8), potentially disturbing interior roof elements (1, 2, 3, 6), occluded corners (7) and variation in the overall structure. Note however the edge precision and recall of sub-figure 23-2; this inference should score 1.0 in both metrics, but scores lower due to a weakness of the label data. The label data in question is shown in figure 22, and illustrates a case where the HEAT inference is more accurate than the label data.

Figure 22: *Example of poorly aligned SFKB label data.*

There are several cases of such misalignment in the data set, and these examples likely limit the learning ability of the model. Corner predictions are shown in figure 25. There are several corner predictions that are not matched by an edge and are filtered out in the final prediction. Inference on the same images by the model using the weights from the HEAT paper without further training is shown in figure 24 for comparison. None of these inferences are fully successful.

Figure 26 shows a selection of unsuccessful inferences. These examples demonstrate some common examples of failure. Sub-figures 1, 5, and 6 show HEAT inference of non-roof structural elements such as wall edges and verandas, sub-figures 4 and 7 show interference by shadows, sub-figure 8 shows occlusion of edge elements and sub-figures 3 and 5 show interference by structural elements on the roof. The biggest outliers here are Figures 2,5 and 6. Examples 2 and 5 infer many closely aligned edges, which seems to be caused by smaller features, such as air conditioners and vents etc., in the roof's interior. Sub-figure 6 does not exhibit any of the attributes mentioned here, and there is no apparent reason why the inference should fail, as there are several examples of similar structures

[11] The term "successful" refers to inferences that are complete and accurate enough to be used for 3D reconstruction

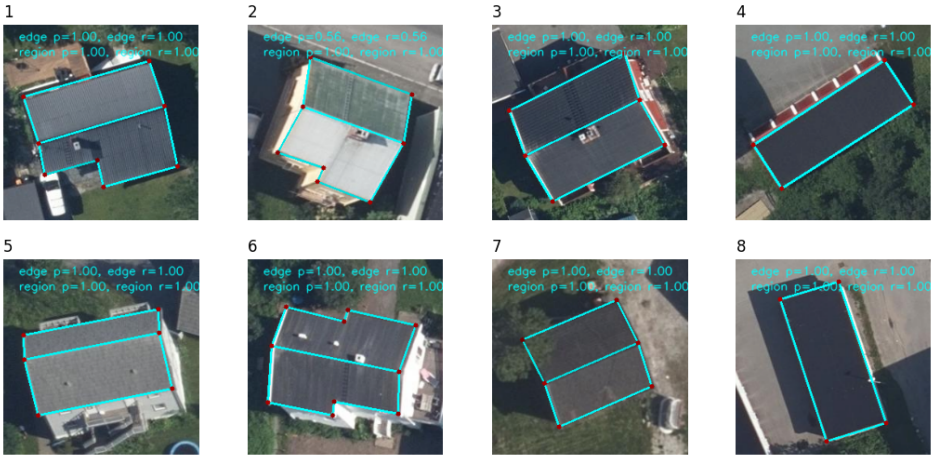


Figure 23: *Successful graph inferences on the test set*

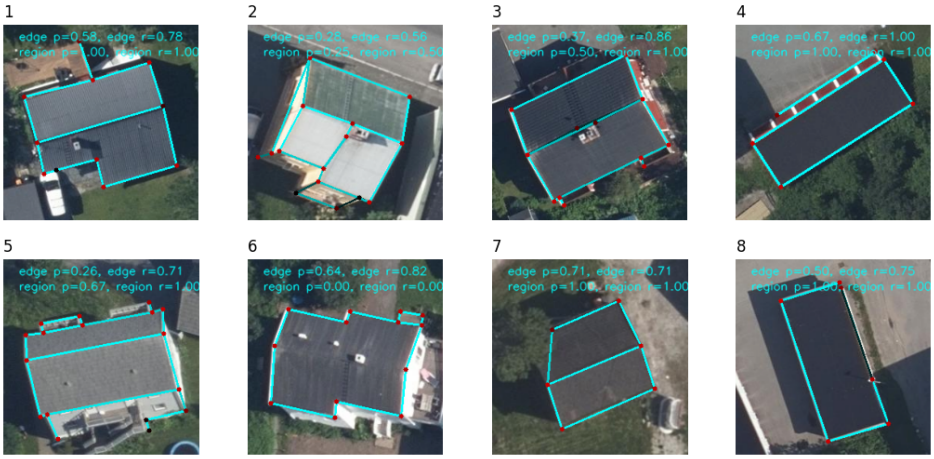


Figure 24: *Inference on the same images using the initial model.*

Figure 25: *Corner detections using the trained model.*

yielding good inferences. Most of the failure cases are a result of shadows from nearby buildings, vegetation, or other structures.

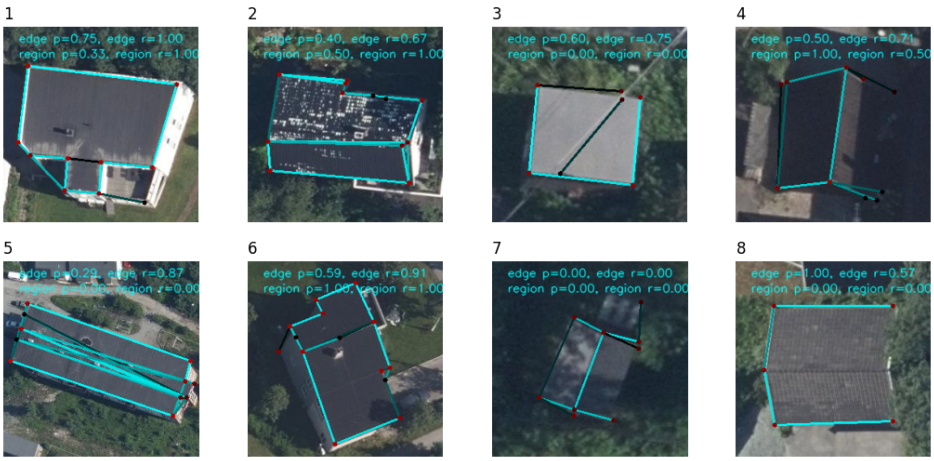


Figure 26: *Unsuccessful graph inferences on the test set*

Finally, a set of semi-successful inferences is shown in figure 27. These are inferences where the correct graph structure is inferred, but there is a small number of extra non-cyclic graph elements present. The reason to highlight these cases is that the non-cyclic extra graph elements can be pruned using graph traversal which will be discussed briefly in section 6.4. This makes them usable for 3D reconstruction after post-processing.

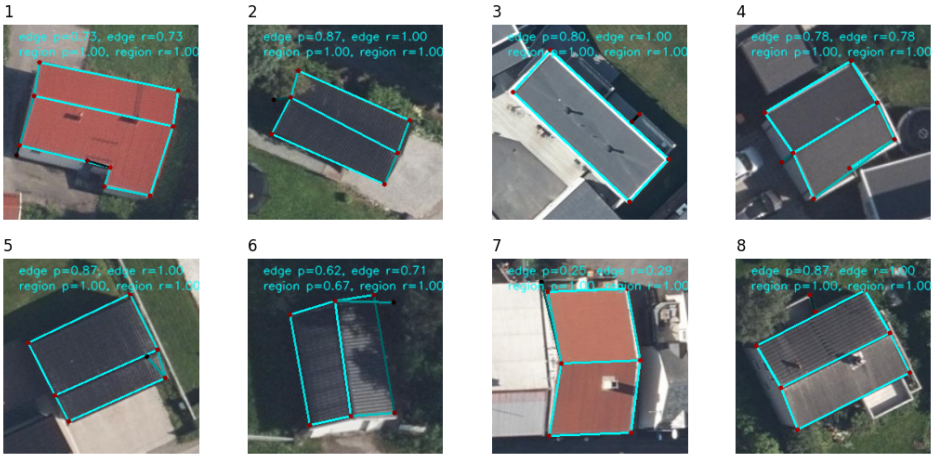


Figure 27: *Semi-successful graph inferences on the test set*

Looking at the recall and precision metrics supplied shown in the figures above, it is clear that these metrics do not necessarily describe the usefulness of each inference with 3D reconstruction in mind. The test set inferences were therefore analyzed manually to classify them into the three groups exemplified by figures 23, 26, and 27. There were 107 (43.7 %) successful inferences, 91 (37.1 %) failures, and 47 (19.2 %) semi-successful inferences. This equates to a total of 62.9 % usable inferences.

The results show that the HEAT model can adapt to a new data set by using fine-tuning, even in the case of inconsistent label data. The data set used in this experiment differs markedly from the data set used in J. Chen et al. (2021) both in that that the depicting buildings are from different architectural traditions (the J. Chen et al. (2021) data is from central Europe) and that the input

images are not ortho-rectified.It therefore contains many images taken at oblique angles that depict more of the non-roof structure of the building. The HEAT model is shown to be able to infer only the roof structure and ignore other structural elements in the building in most cases. Comparing panels 2, 3, 4, and 5 in figures 23 and 24, the model is not able to make this differentiation as often before fine-tuning.

Several types of features reduce the quality of the inferences such as shadows and occlusions. There are, however, examples of successful inferences in these cases as well, and further training may make the model more consistent. Indeed, it should be noted that the model was fine-tuned only for less than 200 epochs on a very small data set and the validation accuracy was still improving, although slowly, at the time of stopping.

6.3 3D reconstruction

Figure 21 suggests that there is a strong correlation between the resampled HEAT inference error and the 3D reconstruction error, and the correlation coefficient between the two is 0.83 (table 3). The sample size is however too small to make any meaningful quantitative assessments of the impact of factors such as the size of the building and number of views used in the reconstruction. The unreliability of the label data also makes it difficult to evaluate the process.

The biggest drawback of the reconstruction method is the need for perfect graph correspondence between inferences in different views. Only 43.7 % of the inferred graphs on the test set would be suitable for use in 3D reconstruction without any further processing, and this assumes that there is at least one more successfully inferred graph in the other views.

There are a few examples of multi-tiered roof structures in the data set (figure 28). These roofs contain sections that are partially or completely disconnected from the immediately neighboring roof structure. As shown in the second row of figure 28, these are not correctly represented in the SFKB data. Additionally, any separating space will not necessarily be shown in every view, and from some vantages, the roof section will appear connected. Clearly, the SFKB dataset is not suitable to train a model to correctly infer such structures. It is not certain that the model would be able to learn to infer such hidden structures even given correct label data. The data set used to perform fine-tuning in this thesis does not contain very many such buildings, as those with more complex roof structures were filtered out. However, such features are very common in buildings and a rule for how these features should be represented should be defined if the model is to be trained on an expanded data set at a later time.

Figure 28: *Examples of buildings with multi-tiered roof structures. The graphs in the second row are transformed SFKB graphs.*

6.4 Avenues for improvement

The results demonstrate that the proposed approach could be viable for use in automatic building registration, but needs improvement. This section suggests some potential improvements.

1. More and better training data The main problem with the approach presented in this thesis is the lack of good-quality training data. To make consistently good predictions, the HEAT model must be exposed to a large amount of accurately annotated data constructed. The SFKB data is too inconsistent to be used to train a model that can make inferences on more complex and varied roof structures, and the extensive pruning that had to be made resulted in a too-small and uniform data set. A hand-annotated data set could be constructed to not only be significantly larger and more accurate, but also to include more varied and complex structures. This would result in more robust inferences. Manually labeling data is a work-intensive process, but the SFKB data can be used as a starting point to make the process more efficient. It would also make an opportunity to clearly define a rule set for how the training data should represent multi-tiered roof structures as was discussed at the end of section 6.3. Additionally, the HEAT model can accept 512×512 images instead of the 264×264 images that were used in this study. This requires more memory and processing power on the hardware but could lead to better results. However,the building images in the data set used in this study are on average 263.8×263.8; very close to the current resampled size, so on this reduced data set it may not be necessary. The average size of the building images before pruning was 365.3×365.3, so if using a more varied data set including more larger buildings, a model accepting 512×512 images should be tested.

2. Disincentivize inference of non-cyclic graph elements Most of the failed predictions by HEAT include inferring non-cyclic graph elements, and it is a very prominent source of error. Any roof graph representation should be cyclic, so any non-cyclic element of an inferred graph will always constitute an unsuccessful prediction. The following is a list of three possible strategies to achieve this:

1. Train the model on better data for longer: If the model is exposed to a larger data set that categorically excludes non-cyclic elements and is trained for longer it may learn to avoid making non-cyclic inferences naturally.

2. Modify the cost function: A term can be added to the cost function that scales with e.g. the number of non-cyclic nodes. This would have to involve detecting the non-cyclic nodes by a graph traversal method such as a depth-first search (Even, 2011).

3. Prune the graph after inference: This would be the simplest strategy to implement, and would involve simply removing non-cyclic graph elements from the inferences. Removal of the non-cyclic elements can be done by iteratively removing any node and its connected edge whenever a node has only one connected edge until there are no such nodes remaining. However, this adds a heuristic step to the inference pipeline, and if similar results can be achieved by a pure learning based approach, that would be preferable.

3. Graph matching for more robust 3D reconstruction The success of the 3D reconstruction seems to be very dependent on the inference error in the image plane, and the DLT (3.5.2) should be a sufficient method for 3D reconstruction if the prediction errors are sufficiently low. Indeed, the reconstruction error of the DLT would be zero using only two views if the graph inferences are perfect (Hartley & Zisserman, 2004). The biggest challenge is therefore to ensure that each building has enough matching graph representations in the different views to allow for reconstruction. In the current implementation, the graphs are assumed to have identical structure, and the matching is not performed at all if this is not the case. This would once again be resolved by better predictions, but can also be helped by a method for correctly matching graphs that do not have identical structures and extracting subgraphs that match the correct structure from inferences with too many inferred elements. This is not an easy problem to solve, however, and would necessitate further research.

It would involve either being able to identify which of the subgraphs are correctly inferred and solving some version of the subgraph isomorphism problem (Eppstein, 1999), or implementing a deep learning approach such as the one suggested in Zanfir and Sminchisescu (2018).

7 Conclusions

The goal of this study was to automatically extract three-dimensional, geolocated building roof structures from non-orthorectified aerial photos, with a view to automate time-intensive tasks that are currently handled manually. A deep learning model (*HEAT;* J. Chen et al., 2021) built on a transformer-based neural architecture was used to automatically extract planar graph representations roof structure elements from the aerial photos, and a multi-view triangulation method (Direct Linear Transformation, *DLT*) was used to transform the planar graphs to 3D UTM coordinates without the use of a digital surface model (DSM). This two-stage framework involves making simultaneous graph inferences with HEAT for the same building in up to five overlapping aerial photographs from different viewpoints, followed by applying the DLT to the set of graph representations in different image coordinate systems to produce a representation of the roof structure in 3D UTM coordinates. Current, manual, methods of extracting building geometry are time and labour intensive, and this work explores the viability of automating the tasks based on state-of-the-art deep learning tools.

The HEAT model is trained using 256 $\times$ 256 resampled image cut outs of individual buildings as input and corresponding graph representations of the building roof structure in the image coordinate system as labels. The first objective of this study was therefore to find a projection to transform 3D UTM building representations to 2D representations in the image coordinate systems for use as label data, and make cutouts around the building coordinates in the aerial images for use as input data. The image cutouts were resampled to 256 $\times$ 256, and the graph representations were resized correspondingly. The projection was successfully implemented using the camera matrix (section 3.5.1). However, the data set had to be reduced from 21376 to 1635 due to the SFKB graph representations missing many structural roof elements.

Using the resampled image cutouts as input data and the projected and resized graphs as label data, the HEAT model was trained over 192 epochs. The training resulted in a mean F1-score of 0.77 for the corner, edge and region inferences up from a mean F1-score of 0.55 using the initial weights. Visual evaluation of the inferences revealed that 43.7 % of the inferred graphs could be used for 3D reconstruction, and a further 19.2 % could be used if pruning non-cyclic graph elements.

Of the fully successful inferred graphs, ten were chosen for 3D reconstruction. Cutouts of these buildings were extracted from the remaining overlapping aerial photos, and HEAT was applied to infer a roof structure graph in every view. For each buildings, HEAT was able to correctly infer the roof structure in 2 to 5 of the views. DLT was applied to construct 3D representations of the roof structures in UTM coordinates. The DLT reconstruction process was successful, but the inference and resampling errors were carried through, and the mean average error of the reconstructed corners was 0.43 m compared to the SFKB data.

The study has shown that the use of HEAT for building roof structure inference in non-orthorectified aerial images is viable if the buildings are relatively simple – even with a small data set and limited training time. However, the 3D reconstruction method requires very accurate and uniform graph inferences, and further research is required to develop either better inferences or post-processing techniques to allow for a higher success rate. If the methods are to be used for more complex structures, a data set must be made by hand.

The introduction of this paper outlined four research objectives. The following is a summary of the findings related to these objectives:

1. **Find and implement a method for transforming SFKB building graphs from 3D UTM to the image coordinate systems of the aerial images:** This objective was accomplished by constructing a camera matrix using the metadata provided in the imaging project file, and applying it to the SFKB 3D UTM coordinates.

2. **Process the transformed SFKB graphs and aerial images to be used as labels and input data respectively for the HEAT model:** The transformed SFKB building graphs in image coordinates were resized, and corresponding resampled image cutouts were generated, both conforming to the formats expected by the HEAT model.

3. **Train the HEAT model on the prepared data:** The HEAT model was trained on a training set consisting of 1308 image - planar graph pairs, and a achieved an F1-score of 0.77 on the test set. 43.7% of the inferences on the test set were good enough to be used for 3D reconstruction.

4. **Apply triangulation to the features extracted by the trained HEAT model in several different views of the same building to transform them back to UTM coordinates:** Ten buildings with successful inferences in the test set were reconstructed in 3D UTM coordinates with a mean average error of 0.43 m compared to the SFKB data.

Building geometry extraction is currently done by hand, and automation has the potential to improve efficiency and accuracy. The results were promising, and automating the task is viable, but the methodology requires further development.

8 Appendix

Code for the project can be found at https://github.com/sfo065/building_detection

Figure 29: *Graph inference for the five remaining buildings.*

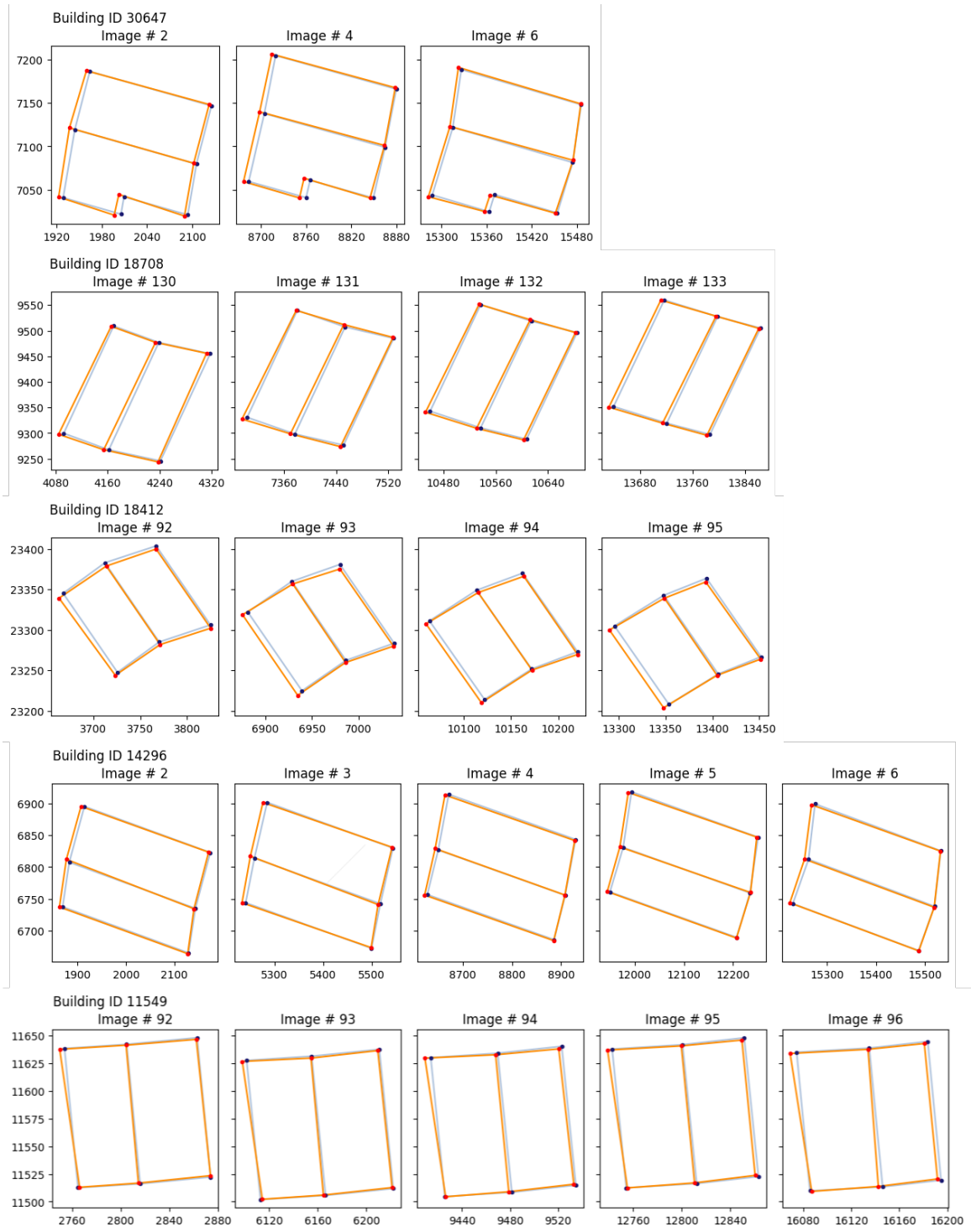


Figure 30: *Resampled graphs for the 5 remaining buildings.*

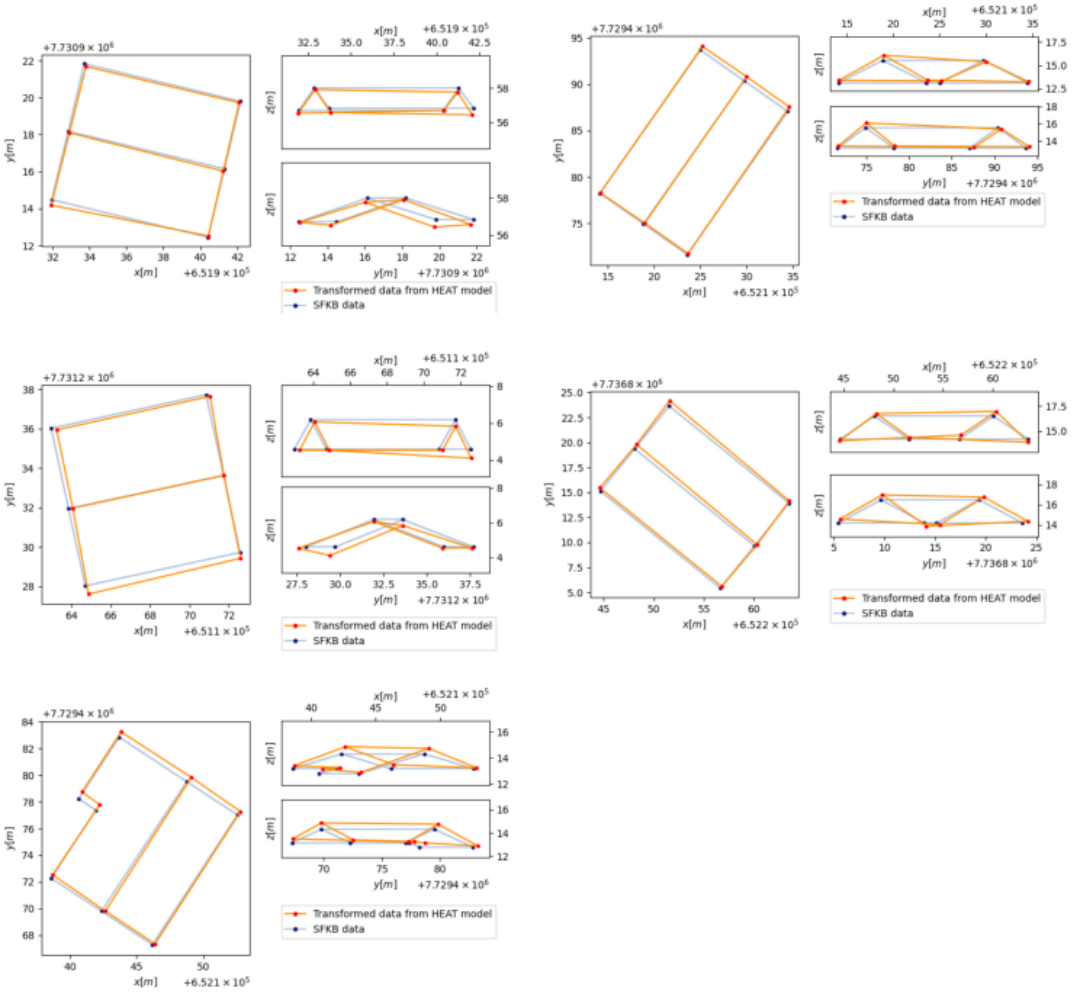


Figure 31: *3D reconstructions of the 5 remaining buildings.*

References

Barthelemy, M. (2017). *Morphogenesis of spatial networks.* Springer International Publishing.

Biljecki, F., Ledoux, H., & Stoter, J. (2016). An improved lod specification for 3d building models. *Computers, Environment and Urban Systems*, *59*, 25-37.

Campoverde, C., Koeva, M., Persello, C., Maslov, K., Jiao, W., & Petrova-Antonova, D. (2024). Automatic building roof plane extraction in urban environments for 3d city modelling using remote sensing data. *Remote Sensing*, *16*(8).

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., & Zagoruyko, S. (2020, 11). End-to-end object detection with transformers. In (p. 213-229).

Chen, J., Qian, Y., & Furukawa, Y. (2021). Heat: Holistic edge attention transformer for structured reconstruction. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 3856-3865.

Chen, J., Wu, D., Song, P., Deng, F., He, Y., & Pang, S. (2020). Multi-view triangulation: Systematic comparison and an improved method. *IEEE Access*, *8*.

Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., & Yuille, A. L. (2016). *Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs.*

Corless, R., & Fillion, N. (2013). *A graduate introduction to numerical methods: From the viewpoint of backward error analysis.* Springer New York.

Côté, M., & Saeedi, P. (2013). Automatic rooftop extraction in nadir aerial imagery of suburban regions using corners and variational level set evolution. *IEEE Transactions on Geoscience and Remote Sensing*, *51*, 313-328.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). *Bert: Pre-training of deep bidirectional transformers for language understanding.*

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., ... Houlsby, N. (2020). *An image is worth 16x16 words: Transformers for image recognition at scale.*

Eppstein, D. (1999). Subgraph isomorphism in planar graphs and related problems. *Journal of Graph Algorithms and Applications*, *3*(3), 1–27.

Even, S. (2011). *Graph algorithms* (2nd ed.). Cambridge University Press.

Gao, W., Peters, R., & Stoter, J. (2024). *Unsupervised roofline extraction from true orthophotos for lod2 building model reconstruction.*

Hartley, R., & Zisserman, A. (2004). *Multiple view geometry in computer vision* (2nd ed.). Cambridge University Press.

He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). *Mask r-cnn.*

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. , 770-778.

Hirt, C. (2016). Digital terrain models. *Encyclopedia of Geodesy*.

Hu, Y., Wang, Z., Huang, Z., & Liu, Y. (2023). Polybuilding: Polygon transformer for building extraction. *ISPRS Journal of Photogrammetry and Remote Sensing*, *199*, 15-27.

KartAI. (2021). *Rapport - kartai, arbeidsgruppe 2 bygningsidentifikasjon.* Retrieved from https://kartai.no/wp-content/uploads/2022/02/AP2-Rapport-Byggidentifikasjon-KartAI-17.12.2021-1.pdf

Kim, T., & Muller, J.-P. (1999). Development of a graph-based approach for building detection. *Image and Vision Computing*, *17*(1), 3-14.

Langley, R. B. (1998). The utm grid system. *GPS world*, *9*(2), 46–50.

Li, E., Femiani, J. C., Xu, S., Zhang, X., & Wonka, P. (2015). Robust rooftop extraction from visible band images using higher order crf. *IEEE Transactions on Geoscience and Remote Sensing*, *53*, 4483-4495.

Lussange, J., Yu, M., Tarabalka, Y., & Lafarge, F. (2023). *3d detection of roof sections from a single satellite image and application to lod2-building reconstruction.*

Macé, S., Locteau, H., Valveny, E., & Tabbone, S. (2010). A system to detect rooms in architectural floor plan images. In *International workshop on document analysis systems.*

Nauata, N., & Furukawa, Y. (2019). *Vectorizing world buildings: Planar graph reconstruction by primitive detection and relationship inference.*

Qin, X., He, S., Yang, X., Dehghan, M., Qin, Q., & Martin, J. (2018). Accurate outline extraction of individual building from very high-resolution optical images. *IEEE Geoscience and Remote Sensing Letters*, *15*(11), 1775-1779.

Ronneberger, O., Fischer, P., & Brox, T. (2015). *U-net: Convolutional networks for biomedical image segmentation.*

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems* (Vol. 30).

Weixiao Gao, R. P., & Stoter, J. (2023). Unsupervised roofline extraction from true orthophotos for lod2 building model reconstruction. *Lecture Notes in Geoinformation and Cartography (LNGC) series*.

Wolf, P., DeWitt, B., & Wilkinson, B. (2013). *Elements of photogrammetry with application in gis, fourth edition.* McGraw Hill LLC.

Yuan, J. (2016). *Automatic building extraction in aerial scenes using convolutional networks.*

Zanfir, A., & Sminchisescu, C. (2018). Deep learning of graph matching. In *2018 ieee/cvf conference on computer vision and pattern recognition* (p. 2684-2693).

Zhang, F., Nauata, N., & Furukawa, Y. (2019). *Conv-mpn: Convolutional message passing neural network for structured outdoor architecture reconstruction.*

Zhao, W., Persello, C., Lv, X., & Stein, A. (2023). Vectorizing planar roof structure from very high resolution remote sensing images using transformers. *IGARSS 2023 - 2023 IEEE International Geoscience and Remote Sensing Symposium*, 4899-4902.

Zhou, Y., Qi, H., & Ma, Y. (2019, 10). End-to-end wireframe parsing. In (p. 962-971).

Zhu, X., Su, W., Lu, L., Li, B., Wang, X., & Dai, J. (2020). *Deformable detr: Deformable transformers for end-to-end object detection.*