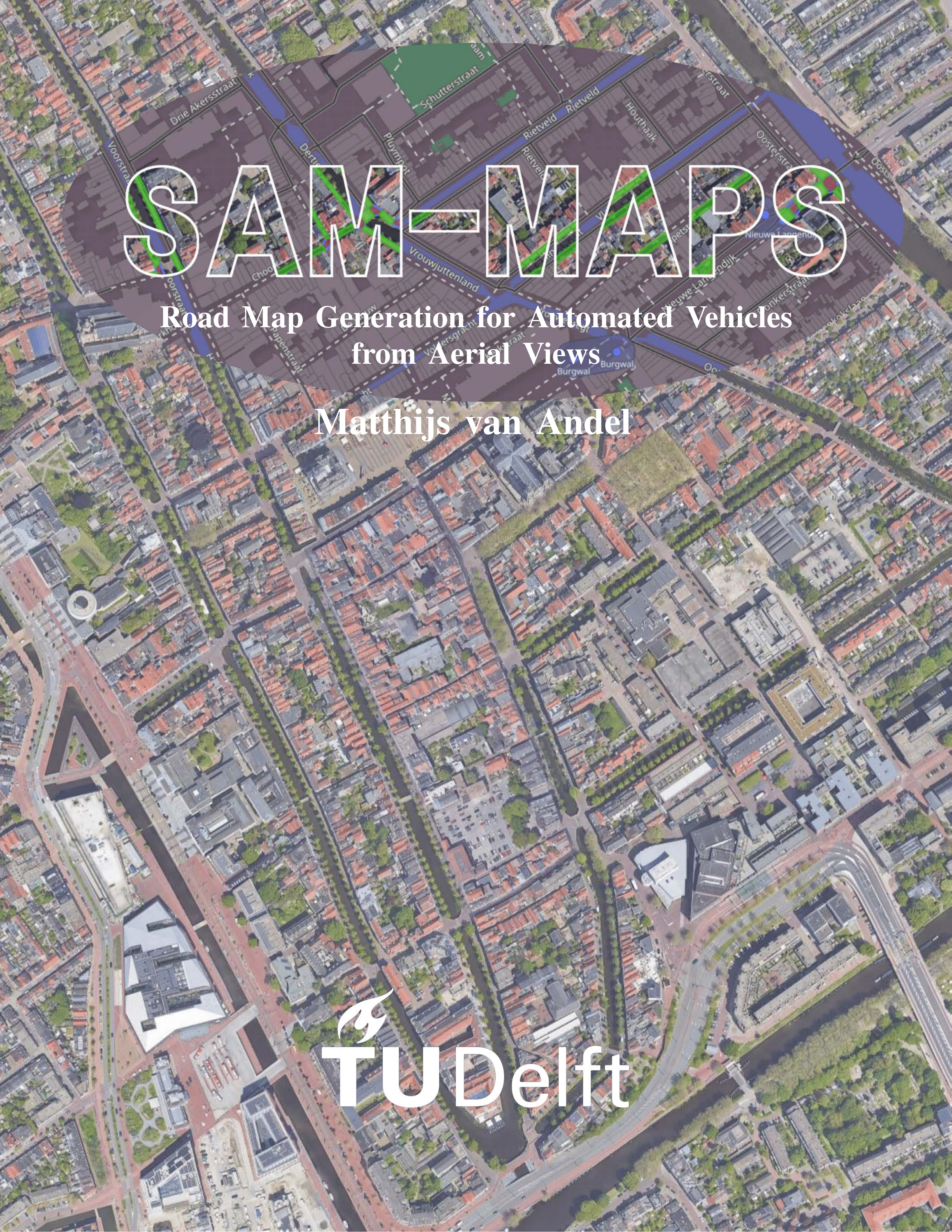


SAM-MAPS

Road Map Generation for Automated Vehicles from Aerial Views

Matthijs van Andel

Road Map Generation for Automated Vehicles from Aerial Views

by

Matthijs van Andel

to obtain the degree of *Master of Science*
at the *Delft University of Technology*,
to be defended publicly on *March 27, 2025 at 11:00*.

Student number:	4952529	
Thesis committee:	Prof. Dr. Dariu M. Gavrila	TU Delft, Academic supervisor
	Dr. Julian F.P. Kooij	TU Delft
	Hidde J-H. Boekema	TU Delft, Local supervisor

Acknowledgement

First, I would really like to thank Prof. Dr. Dariu M. Gavrila for supervising my thesis. His guidance throughout the process, constructive feedback, and support during the submission of the final paper were very valuable. His insights helped shape both this research and the paper that followed.

Second, I am very grateful to H.J. Boekema for co-writing the paper and for his continuous support during my thesis. His advice and feedback helped me to stay on track and improve my work. Collaborating with him was both enjoyable and insightful.

Finally, I would like to thank Dr. Julian F.P. Kooij for joining the committee that evaluates the quality of my research.

M.P. van Andel
Delft, March 11, 2025

SAM-Maps: Road Map Generation for Automated Vehicles in European Urban Areas

Matthijs P. van Andel* Hidde J-H. Boekema* Dariu M. Gavrila

***Abstract*—Automated Vehicles (AVs) rely on up-to-date map information to inform trajectory prediction and planning modules, but these maps are expensive to obtain and update as they are usually annotated by humans. We propose SAM-Maps, a method for automatically generating road maps from aerial images of urban areas that takes advantage of the power of foundation models, requiring no human annotation or additional training to map unseen areas. This method extracts a coarse road graph from the images and then estimates the geometry of the roads from this graph.**

We evaluate our model on the challenging road layouts of the recent View-of-Delft Prediction dataset [1] by comparing the maps generated using our model to the human-annotated maps, achieving a recall of 43.4% with our automatic method and a recall of 75.6% with some human corrections in our method. We also evaluate a trajectory prediction model on our maps to test whether they are sufficiently accurate for downstream tasks. The performance of this model using the map from our automatic method is 37.9% better on the minADE6 metric than not using map data as input. To the best of our knowledge, this is the first method that extracts both the drivable area and road connections of European urban areas from aerial images. The code will be publicly released for research purposes.

***Index Terms*—HD-Maps, Trajectory Prediction, Foundation Models**

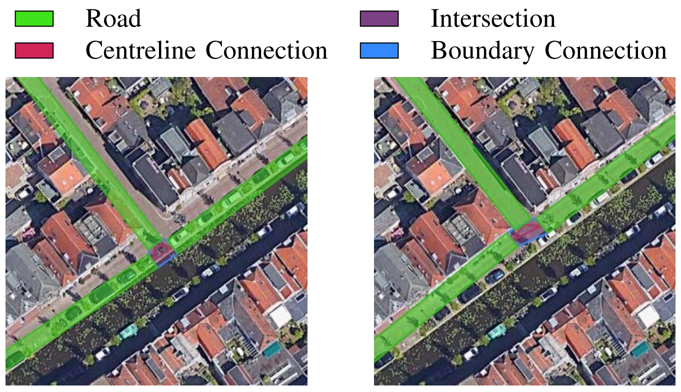
I. Introduction

It is essential for their widespread adoption that Automated Vehicles (AVs) can navigate urban areas without compromising the safety of surrounding road users. An understanding of the behaviour of surrounding agents is critical to this goal. This is especially important in European urban areas, where AVs frequently interact with Vulnerable Road Users (VRUs) such as pedestrians and cyclists. Trajectory prediction models help an AV achieve this understanding by estimating the future positions and/or intent of the agents around the vehicle. These models generally predict future trajectories from the observed trajectories of agents but can also use additional information about the agents or environment.

Current state-of-the-art trajectory prediction models rely heavily on road map data. Road maps contain information about the drivable area and connections of roads, making them a prior for where agents can go in the built environment. Recent trajectory prediction datasets, such as nuScenes [2], Argoverse 2 (AV2), Waymo Motion [3], and View-of-Delft Prediction (VoD-P) [1], provide high-definition (HD) road maps annotated by humans that additionally contain lane information. However, using human annotators is costly and can delay much-needed map updates when the static environment changes.

The authors are with the Intelligent Vehicles Group, TU Delft, 2628 CD Delft, Netherlands.

*Equal contribution



(a) OSM Map (b) SAM-Maps

Fig. 1: Intersection mapped with OSM [4] and SAM-Maps. Accurate OSM annotations are not available everywhere. Our method does not require human annotation and can extract the geometry and topology of urban roads from aerial images.

An alternative map source is OpenStreetMap [4] (OSM), which contains map annotations that can be used for AV tasks such as trajectory prediction [5]. These maps are primarily annotated by a community of volunteers, reducing annotation cost. However, this also makes them vulnerable to mistakes (see Figure 1) and even vandalism (see Figure 3). They are hence unsuited for safety-critical applications such as AVs.

Due to the high manual annotation cost of reliable maps, there is an active research community working on automatic road map generation from sensor data [6]. However, existing approaches either require data from expensive ground-based recording vehicles [7]–[13], do not estimate both the road geometry and topology [14]–[23], or may not generalise well to urban areas [24]–[26]. We propose a method that addresses these shortcomings. To the best of our knowledge, this is the first method that extracts both the drivable area and connections of roads in European urban areas from aerial images.

Our contributions are as follows:

1) We propose a method that uses RGB aerial images and

foundation models to generate AV-suitable road maps (i.e. drivable area and road connections) of unseen urban areas without additional training or human annotation. We show that these maps significantly improve trajectory predictions (compared to not using a map).
2) We investigate the impact of different map information on the coverage of roads in urban areas and on the performance of the state-of-the-art Wayformer [27] trajectory prediction model.
3) We release open-source software to aid geospatial graph and mapping operations[1].

II. Related Work

There is a large body of literature on generating road maps for Automated Vehicles (AVs) using the on-board sensors of a vehicle [7]–[13]. We focus on map generation methods that use aerial images in this section, as these methods do not suffer from the same restrictive mapping costs.

A. Road Graph Extraction

Many methods segment aerial images to extract the topology of the road network as a road graph (RG). [24] uses the DeepLabv3+ [28] to segment road areas and markings from aerial images. Similarly, AerialLaneNet [25] and [26] extract lane-level road maps by segmenting lane markings to find lane centrelines. However, these methods may not work for urban environments due to their dependence on lane markings, which are not always visible (see Figure 5 for an example). SAM-Road [14] instead estimates the centrelines directly, using the Segment Anything [29] model (SAM) to create a road graph from aerial images. This method only estimates the road topology and not its geometry, which is important in planning the trajectory of an AV. The DeepRoadMapper [15] and OrientationRefine [16] methods have the same shortcoming. Note also that these methods are trained on domain-specific aerial images and may not generalise to unseen areas.

Some methods instead opt for an iterative graph-growing approach to road topology estimation. These methods start from a known road point in the image and employ a search algorithm with a decision function to find connected roads, adding nodes and vertices to the graph when new roads are found [22], [23]. These methods do not estimate the road geometry either, however.

B. Road Segmentation

Another essential component of road maps is the geometry of the roads. These can be segmented directly, or can be implicitly estimated by detecting road boundaries. The Topo-Boundary benchmark [17] contains $25,295$ large-scale RGB aerial images with 8 different labels for mapping tasks, such as road boundary and orientation detection. This benchmark paper proposes Enhanced-iCurb, a boundary detection method that has improved the training stability and convergence of iCurb [18], an imitation-learning-based approach for line-shaped object detection. Other road boundary detection methods include [19], a segmentation-based method that requires overhead LiDAR and camera data, csBoundary [20], which extracts boundary keypoints and adjacencies from aerial images to create a boundary graph, and Sat2Graph [21], which combine the advantages of segmentation-based and graph-based methods in a novel encoding scheme. Despite their applicability in road map generation and downstream AV tasks, boundary detection methods do not fully specify the road map, i.e. road geometry and topology. samgeo [30] can be used to segment regions in aerial images through user input, but cannot be used to create a road map fully automatically.

[1]The code will be publicly released for research purposes on paper acceptance.

III. Proposed Method

Our method consists of three modules: **Road Graph Extraction (RGE)**, **Road Segmentation (RS)** and **Road Connection (RC)**. The **RGE module** extracts a coarse representation of road areas and connections that the **RS module** estimates the precise geometry of. The **RC module** reconnects the segmented roads and creates intersections. We describe these modules in detail below. An overview of our method is shown in Figure 2.

A. Road Graph Extraction (RGE)

A road graph (RG) is a global representation of the road network, consisting of **edges**, which represent road segments, and **nodes**, which denote connection points between the segments. We use two approaches for extracting the road graph: using OSM [4] or using SAM-Road [14].

1) Using OSM: OSM contains features relevant for making road graphs, specifically road centrelines, connectivity, and type (e.g., highway, cycle lane). Our OSM-based RGE module queries this information for the area to be mapped and formulates it as a road graph. The width is also annotated for some roads, enabling direct road segmentation by creating polygons from the centreline and width data.

2) Using SAM-Road: SAM-Road [14] generates road graphs from aerial images, eliminating the need for manual annotation. This method estimates the centrelines of the roads (as edges) and the connections between them (as nodes) but does not estimate the road width.

There may be mistakes in the road graph from either approach, such as missing roads or connections, or an incorrectly positioned road; all errors that can be easily fixed by a human annotator. We have therefore written a QGIS [31] plugin that makes it easy to move, add, and remove nodes and edges without breaking the (road) graph. Figure 5 illustrates one such adjustment.

B. Road Segmentation (RS)

After extracting the coarse road graph, our method estimates the geometry of the roads in the graph. The RS module does this by the following steps (explained below): **normalisation**, **coarse segmentation**, and **geometry refinement**. Figure 4 illustrates these steps.

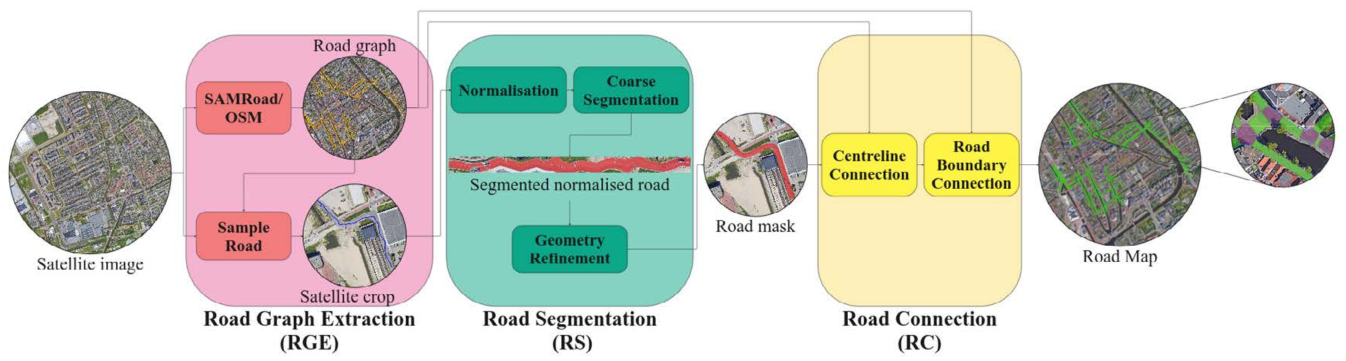


Fig. 2: Overview of our method. The **Road Graph Extraction (RGE)** module extracts a coarse representation of the roads and their connections. The **Road Segmentation (RS)** module estimates the precise geometry of the roads from this representation. The **Road Connection (RC)** module reconnects the segmented roads and creates intersections between them.

Fig. 3: OSM maps can contain mistakes or even vandalism, such as this fictitious town drawn in farmland. Image taken from https://wiki.openstreetmap.org/wiki/Vandalism.

1) Normalisation: The first step normalises the format of the extracted road. Roads are straightened through piecewise **warping** using the nodes generated by the RGE module. The resulting image is then cropped to contain only the road surface from a initial estimate of the road width w_{init} with a safety margin S_{width} to ensure the entire road is captured in the image. This step removes much of the variation in the image, making segmentation of the road easier.

2) Coarse Segmentation: The road surface is then segmented from the warped image. We use Grounding DINO [32], which detects objects based on a text prompt, to get **bounding box proposals** $\{\mathbf{b}_i\}_{i=1}^{N}$ around the road within the warped image. Only the bounding boxes that exceed the threshold probabilities for text-prompt matching $\tau_{\text{gd-text}}$ and bounding box confidence $\tau_{\text{gd-box}}$ are selected. These are input to SAM [29] to generate a **segmentation mask** M_i^{road} for each proposal box $\mathbf{b}_i$.

Since the normalised image should represent a horizontal road, we fit a rectangular mask M^{rect} to the segmented mask in the **mask selection** step. The rectangular mask covers the entire length of the road, but its width is optimised to maximise the Intersection over Union (IoU) with the segmented mask. We select the segmented mask that achieves the highest IoU with its (optimised) rectangular mask. Masks are additionally required to adhere to constraints to be considered: 1) masks must meet a minimum width requirement w_{min}, and 2) masks may not overlap excessively with tree, building, and water masks (generated using Grounded SAM [33]) as this indicates that they are a poor fit or represent the wrong class. This leads to the optimisation problem formulated in Equation (2).

$$\text{IoU}(A, B) = \frac{A \cap B}{A \cup B} \tag{1}$$

$$\begin{aligned} &\arg\max_{i} \; \max_{a,b} \; \text{IoU}(M^{\text{rect}}(a,b) \cap M_i^{\text{road}}) \\ \text{subject to:} \quad & |a-b| > w_{\text{min}}, \\ & \text{IoU}(M_i^{\text{road}}, M^{\text{tree}}) < c_{\text{tree}}, \\ & \text{IoU}(M_i^{\text{road}}, M^{\text{water}}) < c_{\text{water}}, \\ & \text{IoU}(M_i^{\text{road}}, M^{\text{building}}) < c_{\text{building}}. \end{aligned} \tag{2}$$

Here M_i^{road} represents a proposed segmented mask and M^{rect} is a rectangular mask with variable vertical bounds a and b. Finally, M^{tree}, M^{water} and M^{building} are the masks of the trees, water and buildings, respectively, and c_{tree}, c_{water} and c_{building} are constants.

3) Geometry Refinement: The selected segmentation mask can be refined to fill in gaps in the mask and better estimate the width. First, the mask is warped back to the original coordinate system. A heatmap is then created from the mask using the SciPy [34] distance_transform_edt function, which is then max pooled to obtain the most likely centreline points. New centreline proposals are then created: cubic univariate splines with different smoothness parameters S and a line fit through the obtained points. Each centreline proposal is then used to warp the mask again, and then the mask selection step from the **RS** module is repeated for each centreline to select the best one; the road width is taken as $|a-b|$, e.g. the width of the best fitting rectangular box. This results in a complete road geometry i.e. road centreline and width.

C. Road Connection (RC)

To complete the road map, the segmented roads need to be connected to each other. The **RGE** module provides the connections between the roads but represents each connection

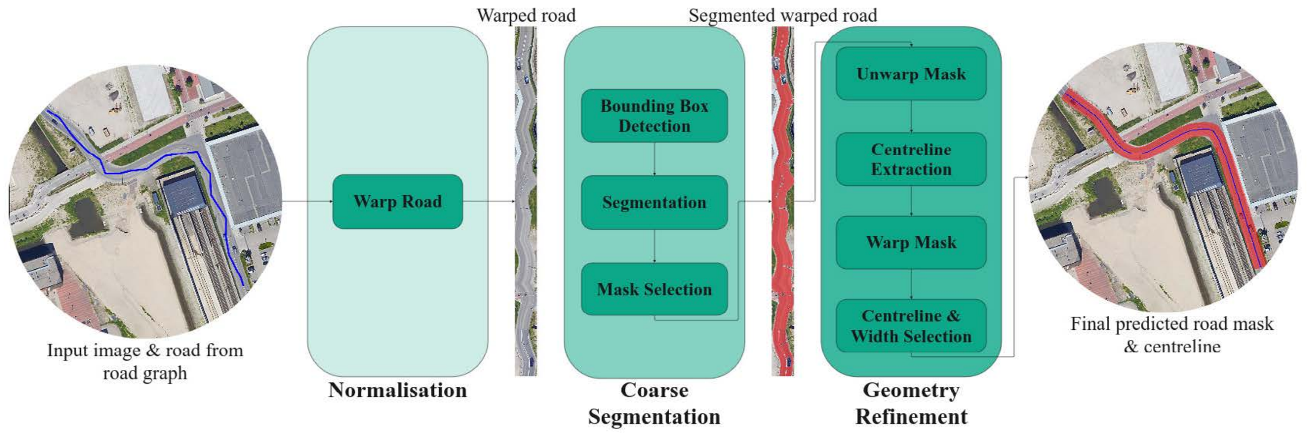


Fig. 4: Overview of the **Road Segmentation (RS)** module.

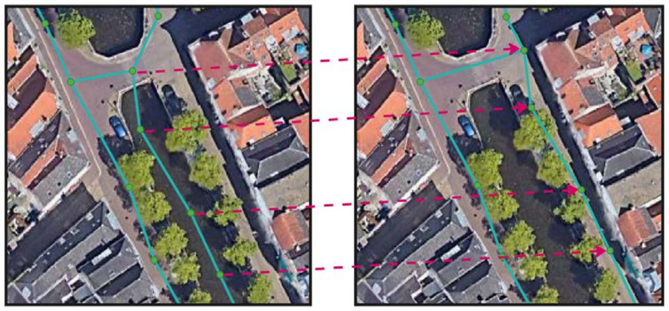


Fig. 5: Manual graph adjustment using our QGIS [31] plugin. Some of the nodes (green) are wrongly placed in a canal, but can be easily moved with one operation by a human annotator. The edges (blue) are automatically adjusted by the plugin.

as a node. However, the centrelines of the segmented roads and intersections between them may not align precisely with these nodes. To address this, the ends of the segmented roads from the **RS** module are trimmed, and a 'connection' line between the centrelines of the (trimmed) connected roads is added. An intersection polygon is additionally created by forming a convex hull around the endpoints of the (trimmed) connected roads to denote the drivable area at the intersection. Examples of intersection polygons are shown in purple in Figure 2. The generated connections ensure smooth and accurate transitions between roads, completing the road map.

IV. Experiments

We evaluate our map generation method on the accuracy of the maps compared to human-made annotations, as well as their usefulness in a downstream task. For these experiments, we select $w_{\text{init}} = 4.5\text{m}$, $w_{\text{min}} = 4\text{m}$, $S_{\text{width}} = 3$, $c_{\text{tree}} = c_{\text{water}} = c_{\text{building}} = 0.05$ and $\tau_{\text{gd-text}} = \tau_{\text{gd-box}} = 0.25$. For fitting splines, we use the SciPy [34] UniVariateSpline implementation with smoothness parameters $S = \{0.1, 0.2, 0.5, 1.0, 2.0, 3.5, 5.0, 10.0\}L$, where L is the length of the segmentation mask. These parameters were empirically determined.

A. *Road Map Coverage*

To assess the fidelity of our generated road maps to the annotated maps provided with recent datasets, we evaluate the recall of the generated road map of the generated roads with respect to the annotated roads of the View-of-Delft Prediction (VoD-P) [1] dataset. This dataset provides challenging road layouts of the city of Delft. We also apply our method on Bratislava, another European city, to show its generalisability. We use (RGB) GeoTIFF images, sourced through the samgeo [30] interface, with a resolution of 8 cm/pixel, as input to our method.

We compare various mapping methods:

- *OSM Map*: a map generated from OSM data. Since OSM data may be available in the area to be mapped, we assess the relative quality of our approach to maps created from this data. We do this by estimating the drivable area from the centreline of annotated roads (that are labelled as drivable roads for vehicles) in combination with the annotated width. If the width is not annotated for a road, we set it to a default width (w_{init}).
- *OSM-RG + A-RS*: our SAM-Maps RS and RC modules with the OSM road graph as input.
- *M-RG + A-RS*: our SAM-Maps RS and RC modules with a road graph drawn manually using the QGIS plugin.
- *SAM-Maps+*: our full SAM-Maps method, with the auto-generated road graph adjusted manually using the QGIS plugin to fix mistakes.
- *SAM-Maps*: our full SAM-Maps method without any human intervention.

An overview of these approaches is shown in Table I.

1) Quantitative results: For the quantitive evaluation we use the recall metric, defined as

$$\text{Recall} = \frac{M \cap M_{\text{GT}}}{M_{\text{GT}}} \tag{3}$$

where M represents the generated mask of the road and M_{GT} represents the ground truth mask derived from the annotations from the VoD-P dataset. Note that the GT map annotations

TABLE I: Method nomenclature.

Category	Name	RGE	RS
Manual	OSM Map	OSM	OSM + Heuristic
Semi-Automatic	OSM-RG + A-RS	OSM	Auto-RS
	M-RG + A-RS	Manual (Man.)	Auto-RS
	SAM-Maps+	SAM-Road + Man.	Auto-RS
Automatic	SAM-Maps	SAM-Road	Auto-RS

TABLE II: Map segmentation results and annotation time of various map sources on the VoD-P [1] dataset.

Name	Annotation Time (h) $\downarrow$	Recall (%) $\uparrow$
Annotated Map	~ 80	**100**
OSM Map	Many	60.5
OSM-RG + A-RS	Many	61.7
M-RG + A-RS	~ 2	**75.6**
SAM-Maps+	~ 0.5	75.0
SAM-Maps	0	**43.4**

only span roads that were relevant to the scenarios recorded in VoD-P. To avoid penalising the additional roads generated by SAM-Maps, we evaluate using the recall for the areas that were annotated instead of the Intersection over Union (IoU).

The recall metric penalises both missing roads and inaccuracies in road placement, but does not penalise excessively wide roads. We provide qualitative examples in Figures 1 and 6 to illustrate that this property of the metric is not exploited by SAM-Maps.

Table II presents the performance of the methods described above on the recall metric.

The highest recall is achieved with M-RG + A-RS. This method requires manual annotation of the road graph, which takes an annotator approximately 2 hours for the roads in the VoD-P dataset. Alternatively, using SAM-Maps+ leads to a small drop in performance but needs only about 30 minutes of manual annotation. Both of these methods significantly outperform the methods that use OSM. This is primarily due to roads having incorrect labels in OSM, resulting in roads missing from the map. The OSM-RG + A-RS map shows only marginal improvements over the OSM Map for the same reason.

Finally, SAM-Maps performs worst on the recall metric. This is mainly due to missing and misaligned (see Figure 5 for an example) roads in the graph generated by SAM-Road. This shows that some level of human annotation is still important for optimal performance.

2) Failure Cases: There are failure cases where SAM-Maps generates a mask that does not fit the masks of the annotated maps. Figure 6 shows examples of some of these cases.

The first example shows a road for which the geometry is incorrectly estimated. The mask for the rightmost road suggests that it turns just before the intersection instead of continuing straight towards it. This could affect downstream tasks such as trajectory prediction and planning by biasing

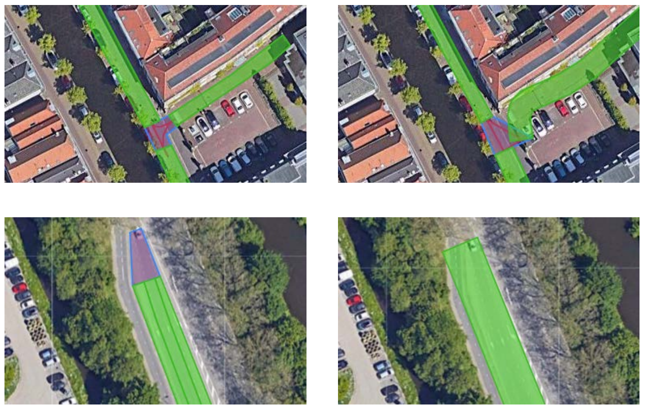


Fig. 6: Qualitative examples of failure cases of SAM-Maps, showing the annotated (left) and generated (right) maps.

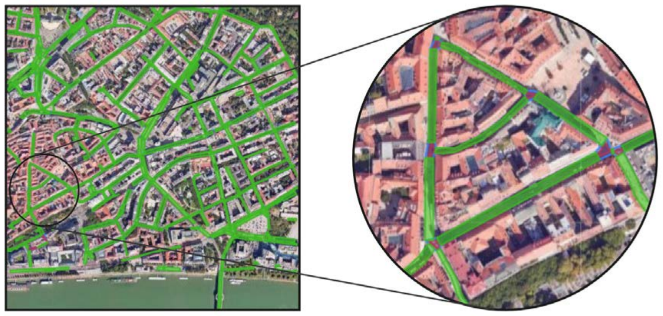


Fig. 7: Output of SAM-Maps on the city of Bratislava. No additional training or annotation was needed to create this map.

them towards following the non-existent turn. This example shows that SAM-Maps could benefit from better handling of intersections using the incoming and outgoing lanes in the road graph, e.g. smoothing the transitions between nodes.

The second example exposes another limitation of the model: its inability to predict individual lanes. In this case, the number of lanes decreases from three to two, but SAM-Maps fails to capture this change. It instead estimates a best fit for the road that combines the width of the three lanes, extending into the narrowed section. Evaluating the individual lanes could therefore improve the performance.

3) Generalisability: We applied SAM-Maps to aerial images of Bratislava without any additional training or human intervention. There are no annotations available for this data, making a quantitative analysis difficult, but the qualitative result shown in Figure 7 gives an indication of the generalisability of the method. Our approach is able to map large areas of the city correctly without any annotation effort.

B. Ablation Study

We conduct an ablation study to determine the value of the key components in our method. We use the SAM-Maps+ method (with corrected road graph) because it achieves the best results out of the variations that use all components of our method. The results presented in Table III highlight the importance of each component of our method.

TABLE III: Ablation study results using SAM-Maps+.

Mask Segmentation	Normalisation	Bbox Proposal	Geometry Refinement	Recall (%) ↑
☐	☐	☐	☐	55.3
☑	☐	☐	☐	37.7
☑	☑	☐	☑	59.3
☑	☑	☑	☐	**75.9**
☑	☑	☑	☑	75.0

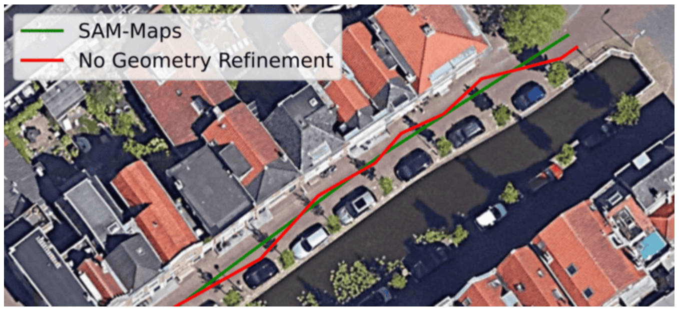


Fig. 8: Example of the effect of geometry refinement.

The modules in SAM-Maps can be further broken down into key operations, including **mask segmentation**, **normalisation**, **bounding box proposal**, **mask selection**, and **geometry refinement**. We systematically ablate these components to assess their impact. Since mask selection is inherently tied to the normalisation and bounding box proposal steps, we ablate these steps together.

1) Mask Segmentation: When the segmentation step is ablated, the method cannot determine road width. The edges generated by the RGE module are then taken as centrelines, and a fixed width w_{init} is assigned to all roads. This method, therefore, depends on the quality of the graph alignment and fails to account for road width variations. As shown in the top row of Table III, this results in a performance drop of 19.7%, underscoring the importance of segmentation in handling diverse road geometries.

2) Normalisation: The normalisation step reduces background noise and standardises the roads in the images by warping them. Without warping, the segmentation masks often capture irrelevant objects, such as buildings or vegetation, leading to noisy or incorrect results. This causes the recall of the method to halve, emphasising the need for a normalisation step.

3) Bounding Box Proposal: Although the normalisation step standardises the image of the road, uncertainty in the road width and road graph alignment remains. Without the bounding box proposals, the entire normalised image is input to SAM, which often segments irrelevant objects (e.g. trees or cars) as part of the road surface. This again leads to a significant drop in performance.

4) Geometry Refinement: The geometry refinement module extracts and selects new road centrelines and widths from the selected segmentation mask. This step has little impact on recall performance; SAM-Maps+ even performs slightly better on the recall metric without this module. This can be explained by how the width estimation algorithm works: it is based on the best fit of a rectangular box on the warped mask of the road. If the nodes in the road graph are inaccurate, the centreline and segmented mask may snake across the road. This, in turn, leads to wider rectangular fits, leading to over-estimation of the road width. Hence the method can perform similarly (or even better) on the recall metric without geometry refinement, but the resulting road map can adversely affect downstream tasks such as trajectory prediction. An example of a centreline that can cause this is shown in Figure 8. This figure also illustrates the smoothed road centreline that the geometry refinement produces.

TABLE IV: Road boundary detection results on the VoD-P [1] dataset.

Method Name	Recall (%) ↑		
	$\tau = 2$	$\tau = 5$	$\tau = 10$
OrientationRefine [16]	0.328	0.832	1.79
Enhanced-iCurb [17]	1.99	5.23	10.4
SAM-Maps	**6.29**	**16.2**	**28.9**
SAM-Maps+	**12.3**	**35.1**	**66.7**

C. Topological Road Boundary Detection

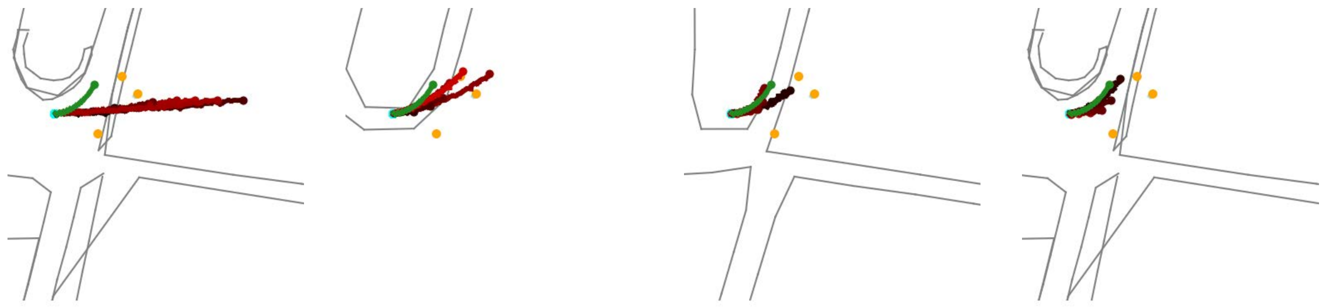
We can easily extract road boundaries from the map produced by SAM-Maps, enabling comparison with road boundary detection methods from the literature. We choose the OrientationRefine [16] and Enhanced-iCurb [17] methods as baselines to compare to because they are the top-performing methods on the Topo-boundary [17] benchmark that additionally have a public implementation. These methods need RGB+NIR images with a resolution of approximately 15 cm/pixel, so we source openly-available RGB and NIR images of Delft from PDOK[2] and upsample these from 25 cm/pixel to 15 cm/pixel. Note that we do not train any of the underlying models for the city Delft since they should be able to generalise to new areas. Following [17], we evaluate the per-pixel Recall of the boundaries where the predicted boundary pixels may lie within a threshold τ of the ground truth boundary pixels. In Table Table IV, we report results for $\tau = \{2, 5, 10\}$ pixels.

The performance of the baselines is significantly worse than that of SAM-Maps and SAM-Maps+, with the best baseline, Enhanced-iCurb, scoring only 10.4% on the recall metric with $\tau = 10$ against a score of 66.7% for SAM-Maps+. This shows that the baselines cannot generalise well to a European urban area such as Delft.

D. Trajectory Prediction

We further test our maps by using them as input data in the trajectory prediction task. The goal in this task is to estimate the future poses $\{\mathbf{x}_{1:T_f}\}_A$ of a set of agents A from their observed past states $\{\mathbf{x}_{-T_h:0}\}_A$ and map information $\mathcal{M}$.

[2]https://www.pdok.nl



(a) No Map (b) OSM Map (c) SAM-Maps (d) Annotated Map

Fig. 9: Qualitative trajectory prediction results on the VoD-P dataset with the Wayformer model and different map inputs. Predictions are shown in red, ground truth future in green, and other agents in orange. Annotated map shown for the 'No Map' setup for clarity. The model using our SAM-Maps map correctly predicts the turn, but the model without map does not.

TABLE V: Trajectory prediction results of Wayformer [27] on VoD-P [1] with various map information.

Map Information	**Automatic**	**minADE6 (m) ↓**	**MissRate6 ↓**
SAM-Maps+	☐	1.80	0.38
OSM Map	☐	1.70	0.29
Annotated Map	☐	**1.23**	**0.28**
No Map	☑	2.98	0.55
SAM-Maps	☑	**1.85**	**0.40**

We select the Wayformer [27] trajectory prediction model for our experiments as it is a recent high-performing model on the Waymo Motion [3] benchmark and has an architecture that is easily modified to work without map data. We use the UniTraj[3] [35] open-source implementation of this model.

We evaluate the model on the VoD-P dataset to test the generated maps for this dataset. We compare the performance of the model using a map from our SAM-Maps approach to its semi-automatic variant SAM-Maps+ and the OSM map to assess their quality. Since VoD-P has a relatively small number of scenarios compared to other public datasets, we pre-train the model on the nuScenes dataset for 150 epochs and fine-tune the best model (on the validation data) on the VoD-P dataset for 300 epochs. We convert both datasets to the ScenarioNet [36] format to homogenise the data, and make training and evaluation scenarios with a short history of 0.5s and a long future (prediction) horizon of 6s to encourage the model to use the map data, when available, in its predictions.

Table V shows the results on the minimum average displacement (minADE) and miss rate (MissRate) metrics for 6 predictions. The performance of the model without map information is poor on both metrics, with all map-based models outperforming it by a significant margin. None of the automatically-generated maps leads to the same performance as the annotated map. Notably, the model performs 38.2% worse on the minADE6 metric with the (annotated) OSM map than with the annotated VoD-P map. The drop in performance when using the SAM-Maps+ and SAM-Maps maps instead of the annotated OSM map is less than 10% on minADE6 and around 0.10 points on MissRate6. SAM-Maps+ only slightly outperforms SAM-Maps, at the cost of some human annotation effort.

[3]https://github.com/vita-epfl/UniTraj

Figure 9 shows qualitative results for some of the results in Table V. The prediction model is unable to infer that the vehicle will turn without map input in the example shown, whereas it correctly estimates the turn with the SAM-Maps map. This map also has better coverage than the OSM map, which does not contain some of the roads. These results confirm the usefulness of the maps of the SAM-Maps method for the trajectory prediction task.

V. Conclusion

We presented a method for generating road maps containing the drivable area and road connections of unseen urban areas from aerial images without needing human annotation, significantly cutting annotation cost and time. These maps can, however, be easily edited by humans to fix errors made in the automatic pipeline through software that we developed. We evaluated the maps created by this method by comparing them to human-annotated maps of the European city of Delft, and found that the auto-generated map has a recall of 43.4% with the annotated map, rising to 75.6% using our semi-auto-generated map. We further tested these maps for the AV task of trajectory prediction on the urban View-of-Delft Prediction [1] dataset, and found that using our auto-generated map led to a performance improvement of 37.9% (over 1.0m) on the minADE6 metric compared to using no map as input to the state-of-the-art Wayformer model [27]. Future work includes segmenting the lanes of a road individually and improving the estimation of the geometry of intersections.

Acknowledgement

This work is part of the research programme Efficient Deep Learning (EDL) with project number P16-25, which is funded by the Dutch Research Council (NWO).

References

[1] H. J. Boekema, B. K. Martens, J. F. Kooij, and D. M. Gavrila, “Multi-class trajectory prediction in urban traffic using the View-of-Delft Prediction dataset,” *IEEE Rob. and Aut. Lett.*, 2024.

[2] H. Caesar *et al.*, “nuScenes: A multimodal dataset for autonomous driving,” in *Proc. of the IEEE/CVF Conf. on Comp. Vis. and Patt. Rec.*, 2020, pp. 11 621–11 631.

[3] S. Ettinger *et al.*, “Large scale interactive motion forecasting for autonomous driving: The Waymo Open Motion dataset,” in *Proc. of the IEEE/CVF Intl. Conf. on Comp. Vis.*, 2021, pp. 9710–9719.

[4] OpenStreetMap contributors, “Planet dump retrieved from https://planet.osm.org ,” https://www.openstreetmap.org, 2017.

[5] J.-Y. Liao, P. Doshi, Z. Zhang, D. Paz, and H. Christensen, “OSM vs HD maps: Map representations for trajectory prediction,” in *2024 IEEE/RSJ Intl. Conf. on Int. Robots and Systems (IROS)*, 2024, pp. 9990–9996.

[6] Z. Bao, S. Hossain, H. Lang, and X. Lin, “High-definition map generation technologies for autonomous driving,” *arXiv preprint arXiv:2206.05400*, 2022.

[7] Q. Li, Y. Wang, Y. Wang, and H. Zhao, “HDMapNet: An online hd map construction and evaluation framework,” in *2022 Intl. Conf. on Rob. and Aut. (ICRA)*. IEEE, 2022, pp. 4628–4634.

[8] B. Liao *et al.*, “MapTR: Structured modeling and learning for online vectorized HD map construction,” *arXiv preprint arXiv:2208.14437*, 2022.

[9] Y. Liu, T. Yuan, Y. Wang, Y. Wang, and H. Zhao, “VectorMapNet: End-to-end vectorized HD map learning,” in *Intl. Conf. on Machine Learning*. PMLR, 2023, pp. 22 352–22 369.

[10] W. Ding, L. Qiao, X. Qiu, and C. Zhang, “Pivotnet: Vectorized pivot learning for end-to-end hd map construction,” in *Proc. of the IEEE/CVF Intl. Conf. on Comp. Vis.*, 2023, pp. 3672–3682.

[11] Y. Cai, W. Dong, Z. Liu, H. Wang, and L. Chen, “HoMap: End-to-end vectorized hd map construction with high-order modeling,” *IEEE Trans. on Int. Vehicles*, 2024.

[12] K. Tang *et al.*, “THMA: Tencent Hd map AI system for creating HD map annotations,” in *Proc. of the AAAI Conf. on Artificial Int.*, vol. 37, no. 13, 2023, pp. 15 585–15 593.

[13] G. Máttyus, S. Wang, S. Fidler, and R. Urtasun, “HD maps: Fine-grained road segmentation by parsing ground and aerial images,” in *Proc. of the IEEE Conf. on Comp. Vis. and Patt. Rec.*, 2016, pp. 3611–3619.

[14] C. Hetang, H. Xue, C. Le, T. Yue, W. Wang, and Y. He, “Segment Anything model for road network graph extraction,” in *Proc. of the IEEE/CVF Conf. on Comp. Vis. and Patt. Rec.*, 2024, pp. 2556–2566.

[15] G. Máttyus, W. Luo, and R. Urtasun, “DeepRoadMapper: Extracting road topology from aerial images,” in *Proc. of the IEEE Intl. Conf. on Comp. Vis.*, 2017, pp. 3438–3446.

[16] A. Batra, S. Singh, G. Pang, S. Basu, C. Jawahar, and M. Paluri, “Improved road connectivity by joint learning of orientation and segmentation,” in *Proc. of the IEEE/CVF Conf. on Comp. Vis. and Patt. Rec.*, 2019, pp. 10 385–10 393.

[17] Z. Xu, Y. Sun, and M. Liu, “Topo-boundary: A benchmark dataset on topological road-boundary detection using aerial images for autonomous driving,” *IEEE Rob. and Aut. Lett.*, vol. 6, no. 4, pp. 7248–7255, 2021.

[18] ——, “icurb: Imitation learning-based detection of road curbs using aerial images for autonomous driving,” *IEEE Rob. and Aut. Lett.*, vol. 6, no. 2, pp. 1097–1104, 2021.

[19] J. Liang, N. Homayounfar, W.-C. Ma, S. Wang, and R. Urtasun, “Convolutional recurrent network for road boundary extraction,” in *Proc. of the IEEE/CVF Conf. on Comp. Vis. and Patt. Rec.*, 2019, pp. 9512–9521.

[20] Z. Xu *et al.*, “csBoundary: City-scale road-boundary detection in aerial images for high-definition maps,” *IEEE Rob. and Aut. Lett.*, vol. 7, no. 2, pp. 5063–5070, 2022.

[21] S. He *et al.*, “Sat2Graph: Road graph extraction through graph-tensor encoding,” in *Comp. Vis.–ECCV 2020: 16th European Conf., Glasgow, UK, August 23–28, 2020, Proc., Part XXIV 16*. Springer, 2020, pp. 51–67.

[22] F. Bastani *et al.*, “RoadTracer: Automatic extraction of road networks from aerial images,” in *Proc. of the IEEE Conf. on Comp. Vis. and Patt. Rec.*, 2018, pp. 4720–4728.

[23] Y.-Q. Tan, S.-H. Gao, X.-Y. Li, M.-M. Cheng, and B. Ren, “VecRoad: Point-based iterative graph exploration for road graphs extraction,” in *Proc. of the IEEE/CVF Conf. on Comp. Vis. and Patt. Rec.*, 2020, pp. 8910–8918.

[24] G.-W. Chen and H.-Y. Lai, “Extracting high definition map information from aerial images,” in *Workshop Proc. of the 51st Intl. Conf. on Parallel Processing*, 2022, pp. 1–5.

[25] J. Yao, X. Pan, T. Wu, and X. Zhang, “Building lane-level maps from aerial images,” in *ICASSP 2024-2024 IEEE Intl. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 2024, pp. 3890–3894.

[26] S. He and H. Balakrishnan, “Lane-level street map extraction from aerial imagery,” in *Proc. of the IEEE/CVF Winter Conf. on Applications of Comp. Vis.*, 2022, pp. 2080–2089.

[27] N. Nayakanti, R. Al-Rfou, A. Zhou, K. Goel, K. S. Refaat, and B. Sapp, “Wayformer: Motion forecasting via simple & efficient attention networks,” in *2023 IEEE Intl. Conf. on Rob. and Aut. (ICRA)*. IEEE, 2023, pp. 2980–2987.

[28] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, “DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs,” *IEEE Trans. on Patt. analysis and machine Int.*, vol. 40, no. 4, pp. 834–848, 2017.

[29] A. Kirillov *et al.*, “Segment Anything,” in *Proc. of the IEEE/CVF Intl. Conf. on Comp. Vis.*, 2023, pp. 4015–4026.

[30] Q. Wu and L. P. Osco, “samgeo: A python package for segmenting geospatial data with the Segment Anything model (SAM),” *Journal of Open Source Software*, vol. 8, no. 89, p. 5663, 2023.

[31] QGIS Development Team, *QGIS Geographic Inf. System*, QGIS Association, 2025. [Online]. Available: https://www.qgis.org

[32] S. Liu *et al.*, “Grounding Dino: Marrying dino with grounded pre-training for open-set object detection,” in *European Conf. on Comp. Vis.* Springer, 2025, pp. 38–55.

[33] T. Ren *et al.*, “Grounded SAM: Assembling open-world models for diverse visual tasks,” *arXiv preprint arXiv:2401.14159*, 2024.

[34] P. Virtanen *et al.*, “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python,” *Nature Methods*, vol. 17, pp. 261–272, 2020.

[35] L. Feng, M. Bahari, K. M. B. Amor, É. Zablocki, M. Cord, and A. Alahi, “Unitraj: A unified framework for scalable vehicle trajectory prediction,” in *European Conf. on Comp. Vis.* Springer, 2025, pp. 106–123.

[36] Q. Li *et al.*, “Scenarionet: Open-source platform for large-scale traffic scenario simulation and modeling,” *Advances in Neural Inf. Processing Systems*, vol. 36, 2024.

[37] Z. Xu, Y. Liu, Y. Sun, M. Liu, and L. Wang, “Rngdet++: Road network graph detection by transformer with instance segmentation and multi-scale features enhancement,” *IEEE Robotics and Automation Letters*, vol. 8, no. 5, pp. 2991–2998, 2023.

[38] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, “Feature pyramid networks for object detection,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2117–2125.

[39] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, “Masked autoencoders are scalable vision learners,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 16 000–16 009.

[40] A. Dosovitskiy, “An image is worth 16x16 words: Transformers for image recognition at scale,” *arXiv preprint arXiv:2010.11929*, 2020.

[41] A. Radford *et al.*, “Learning transferable visual models from natural language supervision,” in *International conference on machine learning*. PmLR, 2021, pp. 8748–8763.

[42] A. Vaswani *et al.*, “Attention is all you need,” *Advances in neural information processing systems*, vol. 30, 2017.

[43] H. Zhang *et al.*, “Dino: Detr with improved denoising anchor boxes for end-to-end object detection,” *arXiv preprint arXiv:2203.03605*, 2022.

[44] Z. Liu *et al.*, “Swin transformer: Hierarchical vision transformer using shifted windows,” in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 10 012–10 022.

[45] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” in *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, 2019, pp. 4171–4186.

[46] T. Ren *et al.*, “Grounded sam: Assembling open-world models for diverse visual tasks,” *arXiv preprint arXiv:2401.14159*, 2024.

Appendix

The appendix is organised as follows:

- Appendix A presents an extended overview of the related work, by analysing the nearest neighbours of SAM-Maps.
- Appendix B analyses the three models that are adopted by SAM-Maps: SAM [29], SAM-Road [14] and Grounding DINO [32].
- Appendix C provides a detailed description of the road segmentation (**RS**) pipeline of SAM-Maps and illustrates an example of how a road is processed by the **RS** pipeline.
- Appendix D expands on the experimental results presented in Section IV.
- Appendix E discusses the results, highlighting limitations and potential directions for future work.

A. Related Work - Nearest Neighbours

SAM-Maps is a road map generation method that extracts the drivable area and road connections from aerial imagery. Various approaches exist for generating different types of map representations. One common category involves HD map generation. Methods that generate HD maps rely currently on vehicle-mounted sensor suites, including sensors such as LiDAR and onboard cameras. These maps provide centimeter-level accuracy and include fine details like lane markings, curbs, and traffic signs. [7]–[13] However, utilising the vehicle-mounted sensor suite suffers from poor scalability due to the high cost of gathering the data and the limited geographic coverage.

In contrast, aerial imagery-based methods offer a more scalable alternative. This forms therefore the focus of this work. Some approaches focus on extracting the road graph [14]–[16], [21], [22], [37], capturing connectivity like SAM-Maps, but they do not explicitly define the drivable area. On the other hand, road boundary detection methods delineate road edges, making it possible to infer the drivable area and, to some extent, road connectivity. [17], [20] We compare SAM-Maps to road boundary detection methods from the topo-boundary benchmark [17], considering them as the "nearest neighbours".

In Section IV, SAM-Maps is evaluated against Orientation-Refine [16] and Enhanced-iCurb [17] on the VoD-P dataset [1]. This section provides a detailed analysis of these two nearest neighbours.

1) Enhanced-iCurb: Enhanced-iCurb is an advanced road boundary detection method proposed by Xu et al. in their work [17]. In this research, the authors established a benchmark dataset specifically for road boundary detection methods and reimplemented several road graph extraction techniques tailored for this task. Additionally, they built upon their original road boundary detection method, iCurb [18], by introducing enhancements referred to as Enhanced-iCurb.

Like its predecessor, Enhanced-iCurb is an imitation learning framework for road boundary detection. The method follows a structured approach consisting of three main components: a **feature extraction backbone**, a **segmentation module** generating initial vertex candidates, and an **agent network** that iteratively predicts and refines graph structures. The architectural layout of iCurb is illustrated in Figure 10. Below, iCurb will be discussed as well as the adaptation made to iCurb in the Enhanced-iCurb framework.

a) Feature Extraction Backbone: iCurb processes aerial images using a Feature Pyramid Network (FPN) [38] as its feature extraction backbone. An FPN captures multi-scale features essential for road boundary detection. The backbone has a large receptive field to capture detailed local spatial information, which in this case refers to the long and thin road curbs. The FPN generates a feature map, which is then utilised by the next step.

b) Segmentation and Initial Vertex Candidates: iCurb utilises two segmentation heads to locate the initial vertex candidates. The first outputs the probability map of the binary segmentation for the road curbs. This mask is then skeletonised and filtered to obtain initial predictions of the end vertices of each remained skeleton segment. To correct these predictions, the second segmentation head predicts a heatmap of the initial vertices. These new initial vertex candidates can update the proposed vertices from the first segmentation head.

c) Agent Network: iCurb operates with an agent that has two prediction heads: one for predicting 2D coordinates and another for determining stop actions. The agent's objective is to connect the initial vertices and construct a road boundary graph.

The first prediction head selects the next vertex by considering the nodes it has connected to so far, along with a cropped region of the feature map extracted by the backbone network. Meanwhile, the second prediction head controls the growth of the connected curbs. When the agent reaches the endpoint of an edge, the stop action is triggered, prompting it to move on to another initial vertex if available. Additionally, if the agent spends too long expanding a specific curb instance, the stop action is also activated.

This iterative process continues until all predicted vertices have been processed, ultimately forming the final graph that represents the road boundaries in the aerial image.

d) Enhancements in Enhanced-iCurb: In iCurb, the agent learns to grow road boundaries by iteratively predicting the next vertex based on its current position and the extracted feature map. During training, the supervision signal is generated dynamically by selecting the closest ground-truth boundary point to the predicted vertex. However, this approach has limitations. It can lead to inconsistencies, as the selected supervision points may vary unpredictably, sometimes resulting in unstable edge lengths and requiring careful tuning to achieve good performance.

Enhanced-iCurb addresses this by introducing an orientation-based supervision method. Instead of relying solely on proximity, it considers the direction of the road boundary to determine the next ground-truth vertex. By selecting a supervision point that aligns with the expected direction of the boundary, the training process becomes more stable, reducing randomness in predictions and improving the overall quality of the generated road graph.

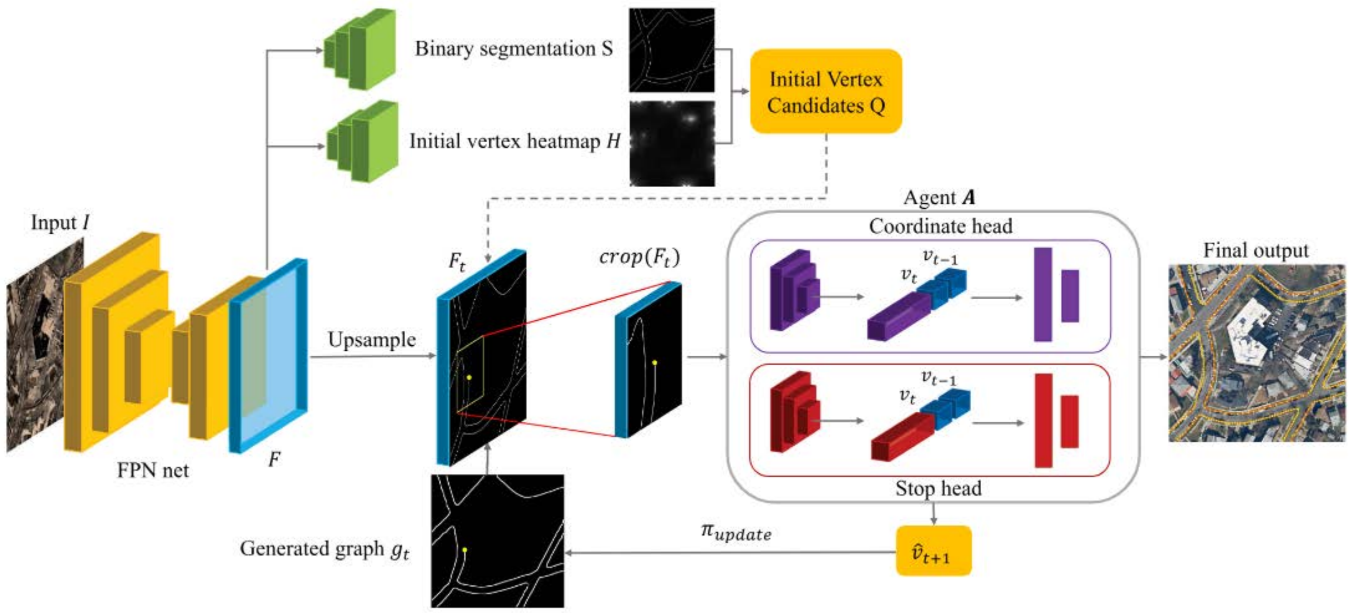


Fig. 10: Overview of iCurb [18]. The model employs an FPN as the feature extraction backbone, followed by two segmentation networks trained to generate an initial set of vertex candidates. Finally, an agent network is responsible for connecting the vertices.

2) OrientationRefine: OrientationRefine [16] was originally designed as a road graph extraction method. Later, [17] adapted and retrained the model for the task of detecting the road boundaries. This section first describes the original approach and subsequently outlines the adaptations required for boundary detection.

To extract the road graph, OrientationRefine consists of three main components: **orientation learning**, **connectivity refinement**, and the **stacked multi-branch module**. The architecture of OrientationRefine is illustrated in Figure 11.

a) Orientation Learning: Instead of merely classifying each pixel as road or background, OrientationRefine predicts a unit vector field representing the local road direction at each pixel. This process encourages topological correctness by learning to infer the connectivity between neighbouring road pixels.

b) Connectivity Refinement: Despite the improved connectivity from orientation learning, road predictions may still suffer from gaps and false connections, particularly in occluded or complex regions such as junctions and underpasses. To address this, OrientationRefine employs a connectivity refinement module, which is pre-trained on corrupted road masks to learn how to restore missing road segments and suppress false detections. During training, artificially corrupted ground-truth masks are provided as input, allowing the model to learn to reconstruct plausible road structures. At inference time, the connectivity refinement module refines the predicted road network iteratively, further improving topological accuracy.

c) Stacked Multi-Branch Module: The stacked multi-branch module is a CNN designed to jointly learn road segmentation and orientation. It consists of three main components: a shared encoder, iterative fusion with multi-branch (the segmentation and orientation branch), and prediction branches for orientation and segmentation.

The shared encoder extracts feature representations from the input image, which are then processed by multiple stacked multi-branch modules. Each stack refines coarse predictions through repeated recalibration, allowing information to flow between the road orientation and segmentation task. This stacking serves three purposes: capturing a larger spatial context with an increased receptive field, progressively refining feature maps, and improving connectivity by iteratively enhancing segmentation and orientation predictions.

By enabling cross-task information exchange, where orientation predictions refine segmentation and vice versa, the stacked multi-branch module improves topological correctness and reduces fragmentation in road network extraction.

d) Road Graph Extraction to Boundary Detection: Xu et al. [17] adapted OrientationRefine for road boundary detection with minimal modifications, as both tasks involve line-shaped object detection. By training it on the Topo-boundary dataset [17], OrientationRefine can be applied as a road boundary detection method.

B. Related Work - Building Blocks of the Proposed Method

SAM-Maps adopts three existing methods from the literature and combines these to generate the road maps that provide the drivable area and road connections using aerial imagery. These three ”building blocks” are: Segment Anything (SAM) [29], SAM-Road [14] and Grounding DINO [32]. These building blocks will be analysed in depth in this section.

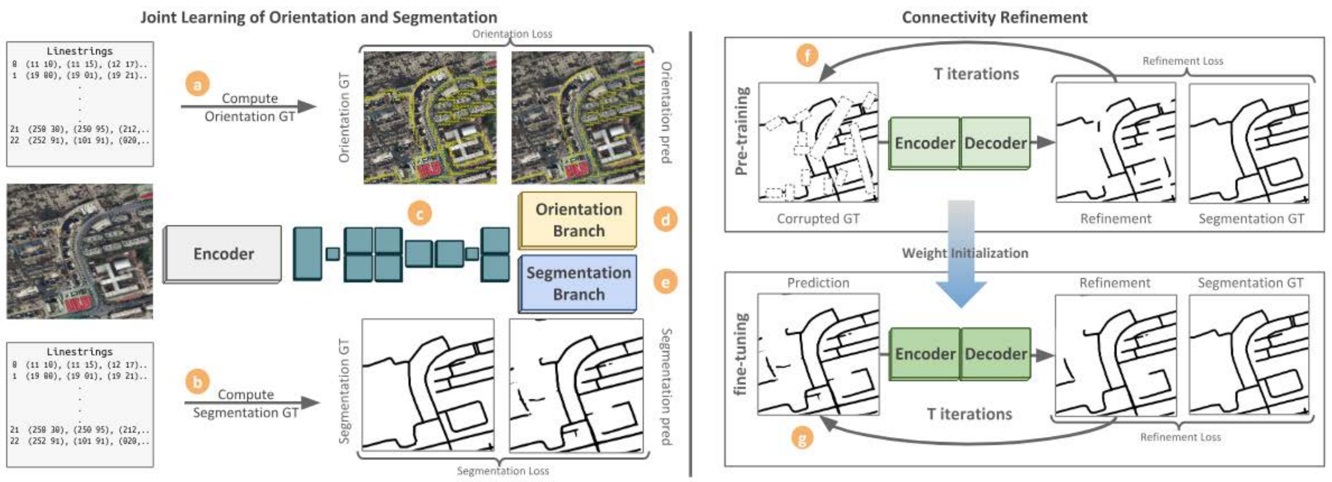


Fig. 11: Overview of OrientationRefine [16]. The model consists of orientation learning to enforce connectivity, connectivity refinement to iteratively enhance topology, and a stacked multi-branch module for joint learning of segmentation and orientation.

1) Segment Anything Model: Segment Anything Model (SAM) [29] is a foundation model developed by Meta AI in 2023, designed to perform high-quality image segmentation based on user-provided prompts. **Foundation models** are large-scale machine learning models trained on vast amounts of data, enabling them to generalise to tasks and data distributions beyond those seen during training. These models leverage self-supervised or weakly supervised learning to build robust feature representations, which makes them highly adaptable to new scenarios.

Segment Anything follows three main steps consisting of an **image encoder**, **prompt encoder**, and **mask decoder**, enabling segmentation of arbitrary images. The architecture of Segment Anything is illustrated in Figure 12.

a) Image Encoder: The image encoder is a Masked Autoencoder (MAE) [39] pre-trained Vision Transformer (ViT) [40], which is a transformer-based architecture designed for image recognition. This pre-trained ViT is minimally adapted to process high-resolution inputs, encoding the image into a high-dimensional embedding space before any prompt is provided. The model is trained on the Segment Anything dataset, which contains 11 million images and over a billion segmentation masks. This allows the model to generalise effectively across diverse objects and scenes.

b) Prompt Encoder: The prompt encoder processes the user-provided input, which can be:

- **Sparse prompts**: Points, bounding boxes, or text descriptions.
- **Dense prompts**: Masks.

The points and bounding boxes are encoded using positional encodings combined with learned embeddings for each prompt type, while text descriptions are processed through the CLIP text encoder [41]. The resulting embeddings are then fed into the mask decoder.

For dense prompts, the prompt undergoes convolution, and the resulting output is then summed element-wise with the image embedding before being fed into the mask decoder.

c) Mask Decoder: The mask decoder receives the image embedding, prompt embedding, and an output token, which acts as a query to guide the generation of the final segmentation mask. It fuses these inputs to produce high-quality segmentations using an adapted transformer decoder block [42] followed by a dynamic mask prediction head. The mask decoder ensures that the output masks align accurately with the intended objects or regions defined by the input prompts. However, due to potential ambiguity from overlapping valid masks, the model outputs three different masks, which has proven sufficient for handling most common cases.

Overall, the Segment Anything Model is designed for zero-shot segmentation, meaning it can segment objects without requiring fine-tuning on new datasets. This makes it particularly effective for general-purpose segmentation tasks across various domains, including applications for autonomous vehicles.

2) SAM-Road: SAM-Road is a specialised adaptation of the Segment Anything Model (SAM) [29] for extracting large-scale, vectorised road network graphs from satellite imagery. Its primary objective is to predict road graph geometry, formulated as a dense semantic segmentation task, leveraging SAM's inherent strengths. SAM-Road follows three main steps: **image encoder**, **geometry decoder**, and **topology decoder**. Figure 14 illustrates the architecture of SAM-Road.

a) Image Encoder: The image encoder of SAM-Road is adapted from the pre-trained Segment Anything Model (SAM) and utilises the ViT-B (Vision Transformer Base) [40] architecture. This encoder processes high-resolution satellite images to produce a feature map. This feature map is then passed to the geometry decoder.

b) Geometry Decoder: The geometry decoder begins with the mask decoder, which is a convolutional neural network designed to produce two probability maps. These maps indicate the existence probabilities for intersection points and roads.

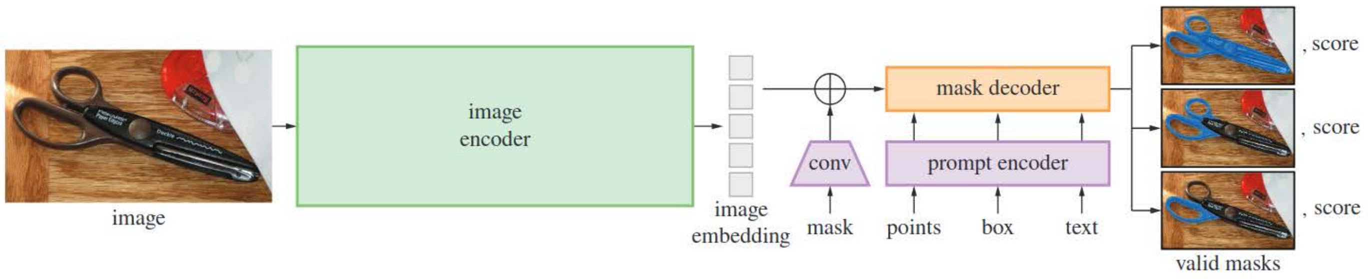


Fig. 12: Overview of Segment Anything Model (SAM) [29]. An image encoder outputs an image embedding that can then be efficiently queried by a variety of input prompts to produce object masks. For ambiguous prompts corresponding to more than one object, SAM can output multiple valid masks and associated confidence scores.

Next, sparse graph vertices are extracted from the dense mask output, while maintaining geometric accuracy. This is achieved through non-maximum suppression (NMS), where pixels with probabilities below a certain threshold are discarded. The remaining pixels are processed in descending order of their probability and any pixels within a defined radius of the currently selected pixel are removed. During this process, SAM-Road places a higher priority on intersection points with higher probability values, as accurately preserving intersection locations is crucial for the integrity of the graph geometry.

c) Topology Decoder: The topology decoder constructs the predicted graph by connecting vertices into their correct structure, ensuring the road network is accurately represented. It uses a transformer-based graph neural network to predict the presence of edges between vertices. This process involves assessing small local subgraphs around each vertex and determining whether the source vertex should connect to any nearby target vertices based on their spatial relationships and the image context. The decoder examines the layout of vertices in the graph, modeling their connections as immediate neighbors, similar to how a breadth-first search operates on the road network.

Through this approach, the decoder predicts the probability of edge existence by considering the relative spatial arrangement of vertices and their context within the image. The final output of the topology decoder is a set of probabilities indicating whether each potential edge between vertices should be present. This effectively completes the road graph.

3) Grounding DINO: Grounding DINO is a framework designed for open-set object detection. This means it can handle novel objects, it was not presented to by the training data. This is a critical capability that allows a system to detect arbitrary objects specified by text inputs.

Grounding DINO is based on the Detection Transformer (DETR)-like model DINO [43], an end-to-end Transformer-based object detector. DETR models combine the power of transformers, with traditional object detection techniques.

Grounding DINO follows a dual-encoder-single-decoder architecture, consisting of the following six main stages: **backbone (feature extraction)**, **text encoding**, **feature enhancement**, **language-guided query selection**, **cross-modality decoding** and **output generation**.

The architecture of Grounding DINO is shown in Figure 13.

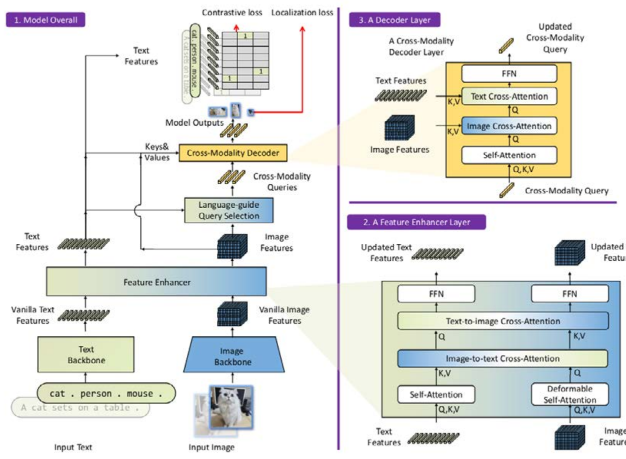


Fig. 13: Overview of Grounding DINO [33]. Block 1 shows the overall framework, block 2 shows a feature enhancer layer, and block 3 shows a decoder layer.

a) Backbone (Feature Extraction): Grounding DINO begins with an image backbone, such as the Swin Transformer [44], to extract multi-scale image features. This produces initial vanilla image features.

b) Text Encoding: Simultaneously, the input text is processed by a text encoder, such as BERT [45], which converts descriptive phrases or labels into embeddings that capture semantic meanings aligned with the visual features.

c) Feature Enhancement: The feature enhancement layer facilitates cross-modality feature fusion, effectively integrating extracted visual and text features. It employs self-attention mechanisms to capture relationships among visual features, emphasising relevant regions. Similarly, a self-attention layer processes text features, capturing word relationships to enhance contextual understanding. Following this, cross-attention layers are applied, incorporating both text-to-image and image-to-text cross-attention modules. These mod-

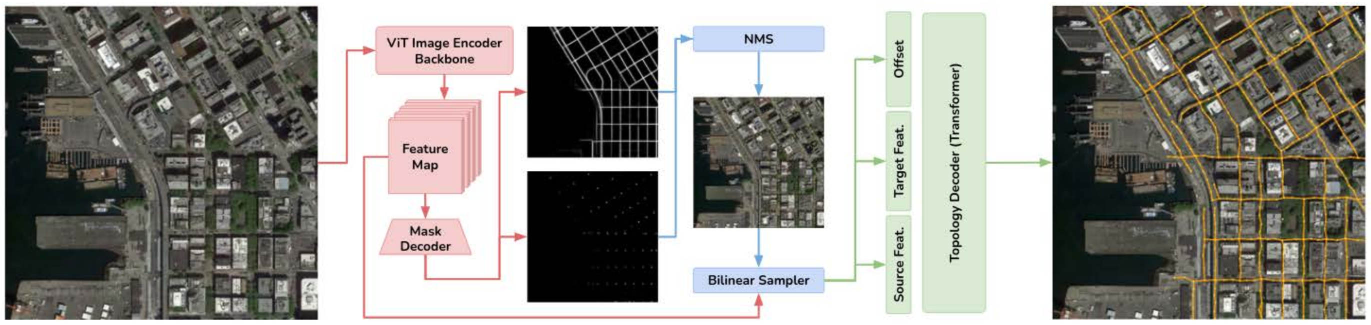


Fig. 14: Overview of SAM-Road [14]. It consists of an image encoder taken from the pre-trained SAM [29], a geometry decoder, and a topology decoder. It directly predicts vectorised graph vertices and edges from an input RGB satellite image.

ules play a crucial role in aligning features across modalities for improved fusion.

d) Language-Guided Query Selection: The language-guided query selection module is designed to identify features in the image that are most relevant to the input text, which will subsequently be used as decoder queries. To achieve this, the module computes the similarity between the image features and the text features. The top indices, based on this similarity score, are selected to represent the most relevant features for decoding.

The query selection process follows the approach used in DINO [43] and employs a mixed query selection technique. Mixed query selection incorporates learned content queries, which are trainable parameters, along with a positional component, namely dynamic anchor boxes that provide positional details within the image. This combination helps ensure that the queries utilised in the decoder are both contextually relevant and spatially informed.

e) Cross-Modality Decoding: The Cross-Modality Decoder in the Grounding DINO architecture is essential for integrating image and text features, enhancing the model's ability to correlate visual content with corresponding textual descriptions.

Structurally, it includes several key layers. Self-attention layers allow each query to consider its own context within the same modality, ensuring the model captures relationships among features. The image cross-attention layer enables queries to selectively attend to relevant image parts, while the text cross-attention layer injects semantic context from the text features.

Each decoder layer also has a Feed Forward Network (FFN) that enhances the output representations produced by the attention layers.

f) Output Generation: The final output consists of multiple pairs of object bounding boxes and their corresponding noun phrases. This means that for a given image-text input, Grounding DINO can accurately identify and locate multiple objects, correlating each detected object with specific descriptions from the input text.

Grounding DINO can also be combined with SAM. The predicted bounding boxes can be utilised by SAM to extract meaningful segmentation masks. This combination, introduced by IDEA as Grounded SAM [46], enables semantic segmentation by integrating both methods, without requiring any finetuning of SAM. Figure 15 illustrates how SAM and Grounding DINO work together for this task.

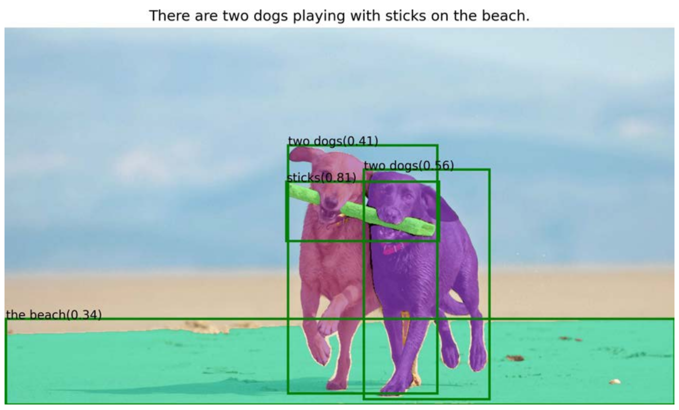


Fig. 15: Semantic segmentation using Grounded SAM [46]. Green bounding boxes are predicted by Grounding-DINO [32] based on the text labels retrieved from the given text prompt, while Segment Anything [29] the corresponding colored segmentation masks.

C. Method - Detailed Road Segmentation (RS) Module

In this section, the Road Segmentation pipeline (**RS**) will be explained in more detail. To do so, we take an example output for a road from the road graph extraction module (**RGE**) and pass it through the **RS** pipeline, leading to a final mask for the road displayed in Figure 24.

1) Normalisation: The normalisation of the road representation is achieved through warping, with the aim of straightening the road. This is done using a piecewise affine warping approach. Affine warping requires three non-collinear points, but the only available input is the centreline extracted from the graph. In the piecewise warping process, each edge of the centreline is evaluated individually. Since this provides only two points, the road width must be estimated. This estimation is based on the common width of a single-lane road (w_{init}). To perform the warping, four points are derived by offsetting the centreline segment in both directions using the estimated width. These four points are then mapped onto a straight line while preserving their relative proportions. This process is repeated for each edge, and the individually warped sections are concatenated to reconstruct a fully straightened road. A visualisation of this process is shown in Figure 16.

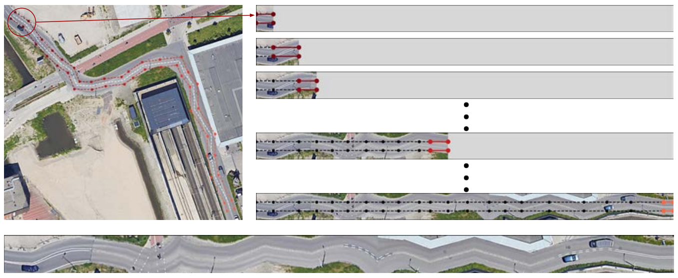


Fig. 16: Visualisation of the piecewise affine warping process. In the left figure (zoom in for details), the white line represents an edge obtained by the Road Graph Extraction (**RGE**) module. The red dots indicate nodes offset by $(w_{\text{init}}/2)$ on either side of this edge. The right images show the progressive warping results at each step.

One important observation is that the resulting warp extends beyond the initially selected points. This occurs because w_{init} may not accurately reflect the actual road width. For instance, if the road has two lanes, it will be wider than the estimated single-lane width. Additionally, the extracted centreline may not perfectly align with the true road centre, introducing an offset. To account for these uncertainties, a safety margin S_{width} is applied when cropping the warped images, ensuring the entire road is captured. The final normalised road is displayed in Figure 17.

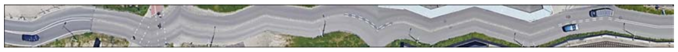


Fig. 17: Example of a normalised road obtained through the piecewise affine warping process.

The normalised road is not perfectly straight. This is expected, as the assumption is that the centreline derived from the road graph extraction module is not entirely accurate and requires adjustment. However, this warping process removes a significant amount of background noise and produces a shape that is easier to analyse in the segmentation step (this is further addressed in Appendix C2). This forms the primary purpose of this normalisation.

For long roads, this warping process becomes computationally expensive. The number of nodes increases with road length, but more critically, the computational cost of the warping operation scales linearly with the amount of pixels. To mitigate this, only a smaller, relevant region of the image is selected for each warping step, reducing the overall computational load.

2) Coarse Segmentation: The coarse segmentation of the road is guided by Grounding DINO [32] and SAM [29]. First, Grounding DINO generates bounding boxes based on a descriptive prompt. For this task, the prompt is set to ”*individual horizontal straight road*”, as it best describes the format obtained during the normalisation step. Grounding DINO generates multiple bounding box proposals, selecting only those that exceed the threshold probabilities for text-prompt matching $\tau_{\text{gd-text}}$ and bounding box confidence $\tau_{\text{gd-box}}$. If no bounding box satisfies these requirements, the bounding box is set to the bounds of the warped image.

The next step involves segmentation using SAM, which can segment based on either annotated points or bounding boxes. Automatically selecting annotated points is challenging, as the chosen points may include trees, vehicles, or other objects unrelated to the road. This could lead to inaccurate segmentation, since there is no guarantee that the selected points are actually on the road. Bounding boxes, in contrast, are less prone to this issue. If the majority of the box aligns with the desired segmentation mask, the approach remains effective.

The advantage of normalising roads into a straight format now becomes evident. Besides eliminating much of the noisy background, this process ensures that the road can be captured within a single bounding box, significantly simplifying the segmentation task. In Figure 18, the bounding box detection and the segmentation are illustrated.

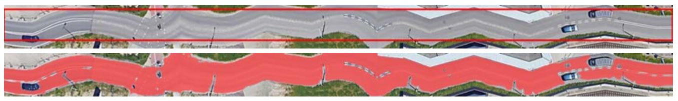


Fig. 18: Bounding box detection (top) and segmentation (bottom) of the normalised road. The red box in the top figure represents the bounding box predicted by Grounding DINO [32], while the red mask in the bottom figure denotes the corresponding segmentation mask obtained using SAM [29].

To select the optimal segmentation mask, each mask must be evaluated using Equation (2). In this example, only one proposed bounding box has a probability exceeding the predefined thresholds, meaning that only a single mask can be assessed. Figure 19 presents the evaluation of the best-fitting rectangular box for the segmentation mask, as formulated in Equation (2).

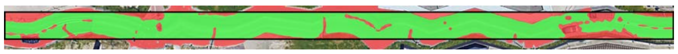


Fig. 19: The best-fitting rectangular box for the segmentation mask. The black box represents the optimal fit, the green area indicates the intersection between the box and the segmentation mask, while the red area denotes the exclusive union. Here, the IoU of the rectangular box with respect to the segmentation mask is **0.67**.

Due to the zigzagging nature of the proposed road, a relatively large non-overlapping area is present, resulting in a lower IoU. Additionally, it is crucial to consider the constraints: the IoU must be evaluated with respect to the masks of trees, water, and buildings, ensuring that it remains below c_{tree}, c_{water}, and c_{building}, respectively. Moreover, the width of the black box in Figure 19 must be at least w_{min}, representing the minimum feasible road width.

The mask in this example satisfies all constraints. However, if no proposed mask meets these criteria, the original centreline and w_{init} are used to generate the road mask, as no reliable data is available to refine the initial prediction.

3) Geometry Refinement: To reduce the zigzagging of the proposed road, geometry refinement is required. The objective is to fit a line through the segmentation mask that accurately represents the centreline of the road mask. This is achieved through six steps:

1) Transform the normalised segmentation mask into the world frame.
2) Generate a heatmap.
3) Perform max pooling on the heatmap.
4) Fit linear lines and univariate splines through the spatially pooled heat map.
5) Assess the centrelines by normalising the mask based on the newly proposed centreline using Equation (2).
6) Extract the road width from the optimisation step.

Transforming the segmentation mask to the world frame is essential, as the ground-truth centreline in the warped frame may exhibit sharp corners due to inaccuracies in the centreline proposed by the **RGE** module. This complexity makes it challenging to fit a spline accurately. Figure 20 illustrates the segmentation mask in the world frame.

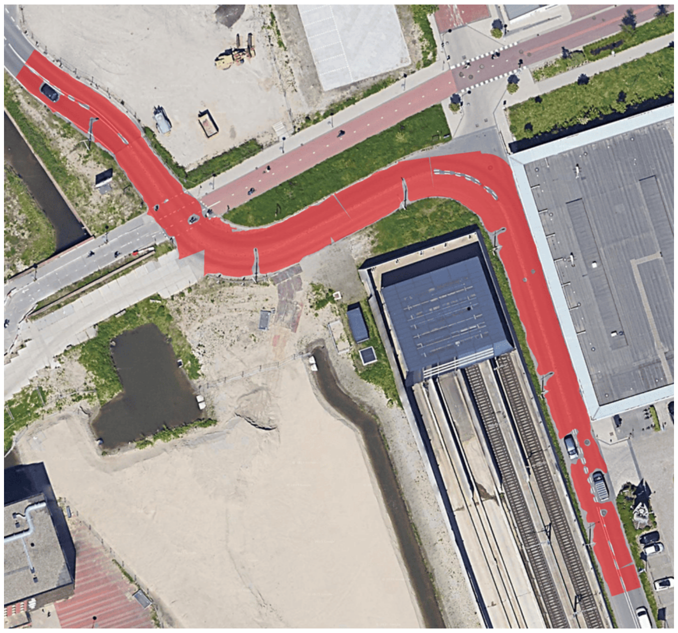


Fig. 20: Segmentation mask overlaid on the road in the world frame.

Through this mask, a centreline is fitted guided by the initially proposed road centreline. To achieve this, a heatmap is generated based on the distance of each pixel to a point outside the segmentation mask, using SciPy [34] distance_transform_edt function (as mentioned in Section III-B3). This assigns higher values to pixels more likely to match the road centre, which informs the spline fitting step.

However, the masks provided can contain numerous gaps caused by vehicles, lane markings, lamp posts, or tree occlusions. To address this, two heatmap versions are generated: one directly from the segmentation mask and another which undergoes a post-processing step to fill holes in the mask and inpaint trees if necessary. Figure 21 illustrates these heatmaps.

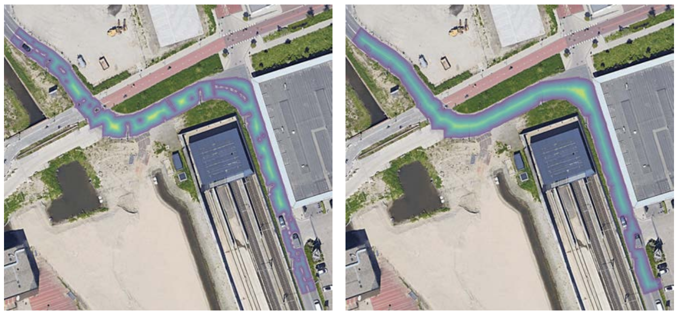

(a) Original heatmap (b) Processed heatmap

Fig. 21: Comparison of the original and processed heatmap, generated from the segmentation mask.

These heatmaps assign weights to each pixel in the mask, which can be utilised in the spline fitting process. Before fitting a line to the heat maps, a max pooling procedure is applied to extract the most likely centreline points.

To identify the most suitable centreline, a linear fit and multiple cubic univariate splines with different smoothness factors S are fitted to the pooled heat maps and evaluated using Equation (2). Figure 22 shows two examples of the proposed splines.

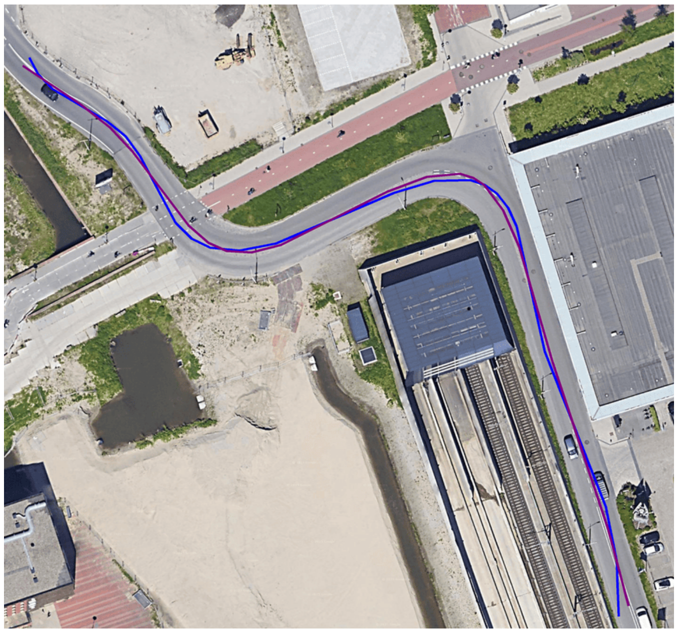
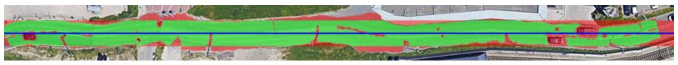


Fig. 22: Comparison of two different spline proposals.

Using these fitted lines, the segmentation mask is normalised back to a straight road and then re-evaluated by calculating the IoU between the best-fitting rectangular box and the warped segmentation mask, as illustrated in Figure 23.


(a) IoU: **0.73**

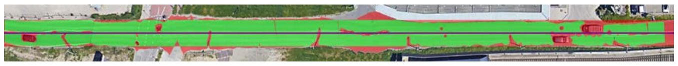

(b) IoU: **0.79**

Fig. 23: A comparison of IoU values for the best-fitting rectangular box relative to the segmentation mask, warped using the two different spline proposals shown in Figure 22.

The line that achieves the highest IoU for the best-fitting rectangular box relative to the segmentation mask is selected. In this example, the geometry refinement has increased the IoU from **0.67** to **0.79**.

The road width is defined as $|a - b|$ from Equation (2), i.e. the width of the best fitting rectangular box. This results in a complete road geometry, consisting of both the road centreline and width, finalising the **RS** module. Figure 24 overlays the final road mask on the aerial image.

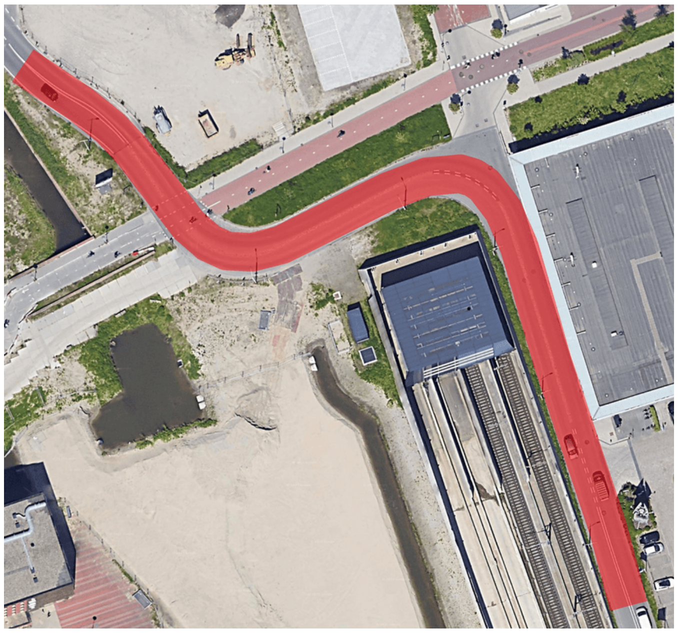
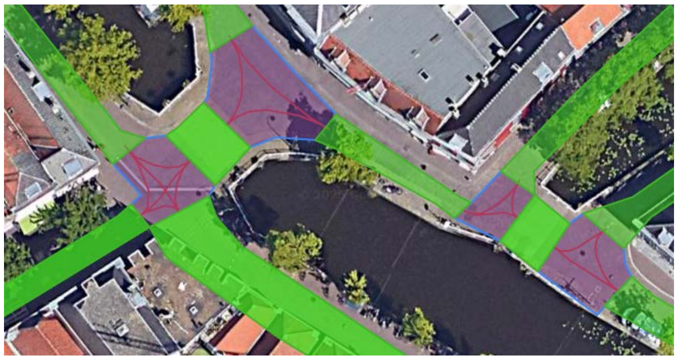


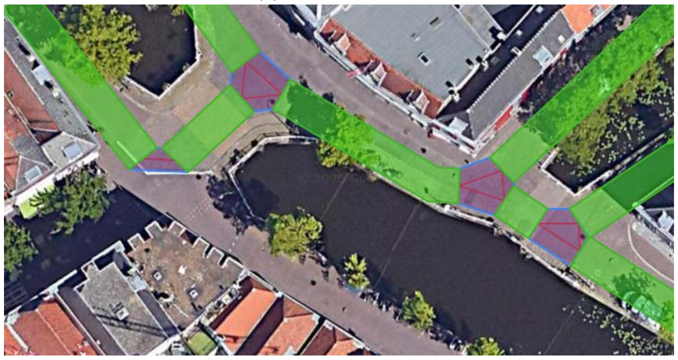
Fig. 24: The final refined road segmentation mask.

D. Experimental Results

The road map generation itself has been evaluated in two different ways. First, the coverage of the road map was evaluated by computing the recall of the drivable area in Section IV-A. Second, the performance of SAM-Maps as a topological road boundary detection method was evaluated in Section IV-C by comparing it to the road boundary detection methods: OrientationRefine [16] and Enhanced-iCurb [17] on the VoD-P dataset [1]. In this section some extra qualitative results of these two experiments will be illustrated and discussed.

1) Road Map Coverage: Figure 25 presents a comparison of various roads and intersections from the VoD-P dataset, generated using SAM-Maps+, alongside the road map generated from OSM and the ground truth.


(a) Ground Truth


(b) OSM Map

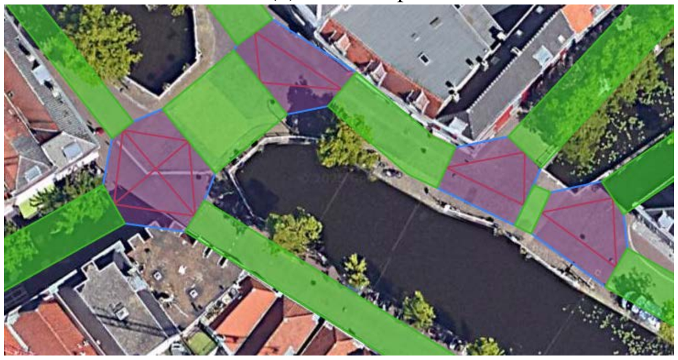
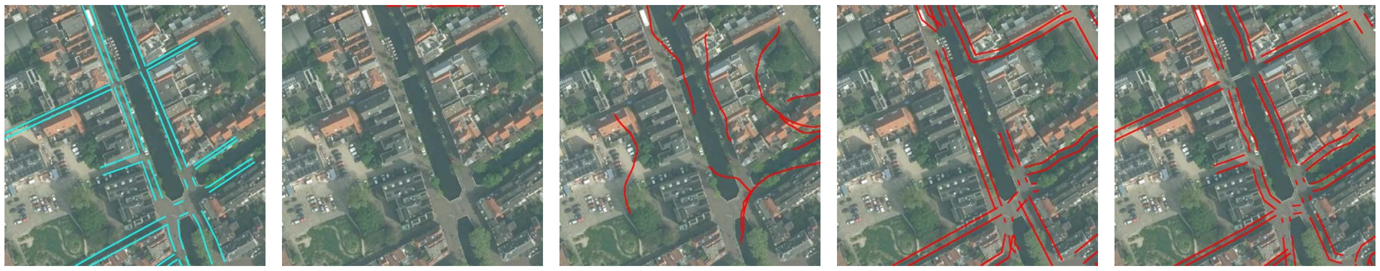

(c) SAM-Maps+

Fig. 25: Evaluation of various roads and intersections from the VoD-P [1] dataset, comparing the maps generated by SAM-Maps+ and OSM against the ground truth.

This example highlights some shortcomings of OSM. First of all, road width is often unannotated, leading to underestimations of the width in some cases. Second, two roads are missing in the OSM map in Figure 25 due to inconsistency in the labeling of the roads. SAM-Maps+ offers better coverage in this example by capturing all roads and providing a more



(a) Ground Truth (b) Orientation Refine [16] (c) Enhanced-iCurb [17] (d) SAM-Maps (e) SAM-Maps+

Fig. 26: Qualitative boundary detection results, evaluated on the VoD-P [1] dataset.

accurate width estimation at times. While some roads are predicted slightly too narrow due to occlusions or too wide due to sidewalks being included in the drivable area, overall, SAM-Maps+ predicts the drivable area with high accuracy.

2) Topological Road Boundary Detection: In addition to the quantitative evaluation of the topological road boundary detection methods presented in Section IV-C, a qualitative evaluation has been conducted. Figure 26 displays an aerial view from the VoD-P dataset evaluated with OrientationRefine [16], Enhanced-iCurb [17], SAM-Maps, and SAM-Maps+.

OrientationRefine has difficulty distinguishing road boundaries, often predicting them along the image edges. Enhanced-iCurb detects some road-like features but produces a significant number of false positives and false negatives. SAM-Maps provides a relatively accurate representation, although it still misses some roads and incorrectly classifies certain areas, like canals, as roads. SAM-Maps+ addresses these shortcomings, offering the most accurate road boundary predictions, closely matching the ground truth.

E. Discussion

We evaluated the road map generation performance of SAM-Maps by comparing its output to manually annotated OpenStreetMap (OSM) data using the VoD-P dataset [1]. In addition, we performed a qualitative evaluation on a different European city (Bratislava) to assess the model's generalisability. We further evaluated its effectiveness on a road boundary detection task, benchmarking it against state-of-the-art methods. Finally, we analysed the impact of SAM-Maps on motion prediction performance using the Wayformer model [27].

In terms of road annotation recall, SAM-Maps does not outperform OSM. However, with minimal manual adjustment to the road graph (SAM-Maps+), the generated maps achieve improved recall, demonstrating that small refinements can significantly enhance map quality. The **RGE** module SAM-Road struggles to distinguish roads from canals, leading to some missing roads. In contrast, OSM annotations are also subject to inconsistencies, where certain roads are mislabeled.

As discussed in Section IV-A3, SAM-Maps demonstrates promising generalisability. Without any fine-tuning, the model successfully generated road maps for previously unseen areas in Bratislava. The qualitative analysis of these results indicates that SAM-Maps is capable of generalising to different European urban environments.

On the VoD-P dataset [1], SAM-Maps outperforms traditional road boundary detection methods, including OrientationRefine [16] and Enhanced i-Curb [17]. The performance of these baselines appears to be influenced by their dependence on the quality of the aerial imagery and the similarity between the test data and their training distributions.

For motion prediction, the inclusion of maps generated with SAM-Maps in Wayformer [27] leads to a substantial improvement over the no-map baseline, confirming the advantages of incorporating structured map information. The performance is also comparable to that achieved using OSM, suggesting that, even though SAM-Maps does not always reach OSM's annotation accuracy, it still provides a strong prior for trajectory forecasting.

1) Limitations and Future Work: Despite these promising results, there is still room for improvement.

Firstly, SAM-Maps has a limited range of road widths it can account for. To reduce background noise that does not belong to the road mask, the safety factor S_{width} was introduced. However, this means that the theoretical maximum width it can detect is $S_{\text{width}} \times w_{\text{init}}$. In practice, this value is even slightly lower, as the initial estimate of the centreline from the **RGE** module is not the exact centreline of the road. To make SAM-Maps more adaptable to a wider range of scenarios, such as wider roads, we could make the **RS** pipeline iterative. At the end of each iteration, a new centreline and width is obtained. Using these updated values, we can perform another warp with the new width instead of w_{init}, followed by segmentation and the rest of the pipeline. This process could be repeated until convergence. This adaptation would allow for a smaller safety margin S_{width}, reducing background noise even further, and it would enable the evaluation of roads of any width.

Another limitation of SAM-Maps is that it models roads as single-width entities, restricting its ability to fully represent complex road structures. This is particularly relevant in cases where a lane is added to a road. One potential solution is to combine SAM-Maps with an explicit road boundary detection method, which could refine road delineation and improve overall performance.

Additionally, there are steps that could be taken which may not directly enhance the recall of the drivable area but could focus on refining connections or integrating other contextual information into the generated road maps. This could be beneficial for downstream tasks such as motion prediction.

One of these next steps is lane extraction. Individual lane information is crucial for high-precision motion prediction. Extending the model to differentiate lanes could provide richer contextual information, leading to improved trajectory forecasting.

Another crucial aspect of the generated road map for the motion prediction task is the accuracy of the intersection mapping. Currently, intersections pose a challenge, as errors often arise at the endpoints of predicted road segments. For example, SAM-Maps may predict a turn before reaching an intersection, resulting in inaccuracies in road connectivity. To tackle this, an additional step could be implemented to specifically evaluate intersections, allowing for a more precise understanding of the geometry of the converging roads.

Moreover, SAM-Maps could benefit from a post-processing step that analyses incoming and outgoing lanes at intersections to establish feasible connections between them. This step could also focus on improving the smoothness of road transitions, replacing abrupt, linear connections with more natural curves. Smoother transitions may not only enhance the realism of the generated road network but also improve motion prediction performance by providing a more accurate representation of drivable paths.

By addressing all these challenges, SAM-Maps could potentially bridge the remaining performance gap in motion prediction with OSM maps while maintaining its advantage of being unaffected by annotator errors or vandalism.