Utrecht University

Department of Information and Computing Sciences
Study Program Game and Media Technology

Geometric Late-Fusion of GPS and Satellite-based Road Network Reconstructions

---

16 September 2025

AUTHOR:	Bsc. Jeroen van Tubergen	Utrecht University
FIRST SUPERVISOR:	Dr. Frank Staals	Utrecht University
SECOND EXAMINATOR:	Dr. Maarten Löffler	Utrecht University

Abstract

Accurate and up-to-date road information is essential for navigation systems, but the immense scale of global road networks and continuous infrastructure changes make manual mapping impractical. Automated map reconstruction offers a promising solution by leveraging widely available data sources, with GPS trajectory data and satellite imagery being the most accessible and commonly used modalities. While recent multi-modal techniques have emerged that fuse these data sources using neural networks, limited work exists on algorithmic approaches that combine separately reconstructed maps.

This thesis investigates a late-fusion approach that combines two independently reconstructed road maps—one derived from GPS trajectory data and one from satellite images—through geometric algorithms rather than neural network fusion. We propose a similarity function based on the Fréchet distance that measures how well individual road segments relate to entire road networks, enabling detection of local disagreements between reconstructed maps. We develop a merging strategy that resolves these disagreements through systematic edge injection and removal operations, aiming to leverage the complementary strengths of both reconstruction methods while preserving geometric accuracy.

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science.

Outline

1. Introduction

Accurate road information is essential for navigation systems, urban planning, and infrastructure management. The immense scale of global road networks makes automated map reconstruction a promising solution for addressing mapping challenges at scale. Map reconstruction infers road network information from available data, with GPS trajectory data and satellite imagery being the most widely used sources due to their extensive availability.

GPS-based methods like Roadster [1] excel at capturing road connectivity from vehicle movement data, but produce geometrically imprecise road networks with sparse coverage in low-traffic areas. In contrast, satellite-based methods like Sat2Graph [2] provide high-quality geometric detail and complete spatial coverage, but struggle with connectivity accuracy, particularly in visually obstructed areas or complex intersections where they frequently misinterpret connectivity patterns. These complementary strengths suggest that multi-modal approaches combining both data sources could achieve reconstruction quality beyond what either method accomplishes alone.

Despite these clear complementary advantages, the literature reveals a significant research gap between well-developed unimodal reconstruction methods and limited multi-modal approaches. Recent advances in artificial intelligence have enabled sophisticated data fusion techniques, yet key questions remain about how to effectively combine different reconstruction approaches while preserving their individual strengths.

This research focuses on developing a geometric algorithm to resolve road continuation conflicts—disagreements between maps about whether roads connect—as a key step toward improving multi-modal map reconstruction quality. By addressing these connectivity disagreements while preserving the geometric accuracy of satellite-based methods, this approach aims to harness the complementary strengths of unimodal map reconstructions. To our knowledge no previous work has specifically addressed resolving road connectivity conflicts between reconstructed maps through geometric algorithms.

1.1. Research Questions

This research investigates whether a map fusion algorithm can improve road network reconstruction by resolving road continuation conflicts while preserving geometric accuracy by developing a late-fusion methodology that operates on fully reconstructed unimodal maps, enabling a geometric algorithm to make informed decisions about road connectivity based on complete road network information.

The research addresses whether map fusion can improve reconstruction quality by resolving road continuation conflicts between GPS and satellite-based maps. This investigation examines whether algorithmic combination of complementary map reconstructions can overcome the limitations of individual maps.

1.2. Contributions

This thesis makes four primary contributions to multi-modal map reconstruction, advancing both theoretical understanding and practical methods for combining GPS and satellite-based road network reconstructions.

Road Continuation Property - Introduction of road continuation, an edge property describing whether a road segment is a dead end or connects with other roads on both ends. This is the core property the late-fusion algorithm seeks to resolve when disagreements exist between two unimodal map reconstructions.

Edge Coverage Mechanism - Introduction of edge coverage, a map matching technique that uses distance thresholds based on the Fréchet distance of curves against graphs, applied in the new context of detecting disagreements in road connectivity. This enables algorithmic identification of road continuation conflicts and provides the core decision-making component for all subsequent fusion operations.

Late-Fusion Map Reconstruction Strategy - The design and implementation of a three-step fusion algorithm that combines GPS and satellite-based reconstructed maps through edge injection, deletion, and reconnection. The late-fusion approach begins with one unimodal map as the base map and replaces discontinuating segments with continuating edges from the other unimodal map to resolve road continuation disagreements while minimizing changes to local topology..

Enhanced Evaluation Methodology - The introduction of TOPO* and APLS* variants that extend the standard map similarity metrics TOPO and APLS to handle comparisons between graphs with significantly different edge densities. These metric adaptations ignore maximally penalized samples when either graph lacks nearby road geometry, enabling meaningful similarity assessment when comparing sparse and dense networks.

1.3. Document Outline

Section 2 describes recent developments in unimodal GPS and satellite-based reconstruction methods, map similarity measures, and existing multi-modal approaches. Section 3 establishes the mathematical foundations by defining curves, graphs, and paths necessary for describing the proposed map reconstruction method. Section 4 explains the three primary contributions: the edge coverage mechanism, the late-fusion reconstruction strategy, and the enhanced evaluation methodology. Section 5 details the experimental infrastructure, data collection methodology, and implementation of all components including spatial indexing, coordinate systems, reconstruction methods, and the complete evaluation pipeline. Section 6 presents experimental validation across two datasets, parameter analysis, and both quantitative and qualitative assessment of fusion effectiveness. Section 7 analyzes the unexpected experimental findings, examines fundamental measurement limitations, and identifies factors that impact the ability to draw definitive conclusions about the research questions. Section 8 concludes on the research outcomes and the contributions. Section 9 suggests directions for future work to advance this multi-modal map reconstruction research.

2. Related Work

I begin by outlining the literature on similarity measures for maps and curves, since evaluating map reconstruction algorithm performance fundamentally depends on map similarity metrics. Additionally, curve similarity serves as a core geometric component in many reconstruction methods, making it essential to understand the existing approaches.

Map reconstruction research can be organized into two main categories based on data modality: GPS traces and satellite images. The literature in these areas has developed largely independently, so I examine each category separately to properly understand their distinct approaches and contributions. After covering these unimodal methods, I discuss the smaller but emerging field of multi-modal map reconstruction, which combines multiple data sources to improve reconstruction quality.

Computational complexity receives secondary attention in map reconstruction literature. Many studies omit computation times entirely from their evaluation results, likely because the field currently prioritizes accuracy over efficiency. Since map reconstruction typically needs to be performed only once for a given area, processing speed has limited practical impact. However, methods incorporating computationally efficient components in their inference pipeline often achieve better results, mainly because they can perform significantly more computation within practical constraints. While this computational capacity can be relevant to the method's effectiveness, it doesn't affect the overall evaluation of the inference pipeline's performance.

2.1. Curve distance functions

Curve similarity functions take two curves as input (see Section 3.1) and return a distance value.

$$d : (C, C) \to \mathbb{R}$$

A lower distance value indicates that curves are more similar, with completely identical curves having a distance value of zero.

Note that curve positioning is just as relevant as curve *shape* similarity when comparing road curvature. For this reason, curves are not pre-aligned by rotation or translation before comparison.

I use the general formula for Fréchet distance described by Alt and Godau [3], which Urhausen [4] extended to the weak Fréchet distance. The general formula to find similarity between two curves P and Q is:

$$d(P, Q) = \inf_{\sigma,\tau} \max_{t\in[0,1]} d(P(\sigma(t)), Q(\tau(t)))$$

Here, σ and τ are functions that remap values from the unit interval: $\sigma, \tau : [0, 1] \to [0, 1]$. Both functions are surjective, meaning they provide a complete traversal of their respective curves. The specific constraints applied to these functions determine which distance variant is being computed.

2.1.1. Fréchet distance

The **Fréchet distance** [3] d_F requires that τ be continuous and monotonic, which means both curves must start at the beginning ($\tau(0) = 0$) and end together ($\tau(1) = 1$). This constraint captures similarity in both curve shape and curve flow better than the Hausdorff distance (see Section 2.1.2).

Since the Fréchet distance requires curve P to be traversed monotonically and continuously from start to end, we can simplify the general formula by removing the σ mapping function:

$$d_F(P, Q) = \inf_{\tau} \max_{t\in[0,1]} d(P(t), Q(\tau(t)))$$

The **weak Fréchet distance** variant maintains continuity while allowing curves to move back and forth during traversal. However, the constraint that both curves must be traversed completely from start to end remains.

The σ function becomes particularly relevant when considering the **k-Fréchet distance**, which allows up to k discontinuous jumps within one of the curves. This creates a spectrum of distance measures where Urhausen [4] notes that the k-Fréchet distance lies between the Hausdorff distance and the standard Fréchet distance: Higher values of k make the measure behave more like the Hausdorff distance. At the extremes, a k-Fréchet distance with $k = \infty$ becomes identical to the Hausdorff distance, while $k = 0$ reduces to the standard Fréchet distance.

2.1.2. Hausdorff distance

The **Hausdorff distance** is originally defined for sets of points P and Q:

$$d_H(P, Q) = \max\left(\overrightarrow{d_H}(P, Q), \overrightarrow{d_H}(Q, P)\right)$$

$$d_H(P, Q) = \max\left(\sup_{p \in P} \inf_{q \in Q} d(p, q), \sup_{q \in Q} \inf_{p \in P} d(p, q)\right)$$

In terms of the general framework, the Hausdorff distance places no constraints on the τ function, allowing any position on curve Q to be selected at each moment during traversal of P. This means the Hausdorff distance is determined by the point on P whose nearest point on Q is furthest away.

2.1.3. Discrete curve similarity functions

Discrete alternatives exist for every curve similarity function mentioned above. These discrete variants consider only the actual sampled points of both curves (like the Discrete Fréchet distance by Eiter and Mannila [5]) and do *not* use line segment interpolation between points.

2.2. Map Similarity functions

Map similarity functions provide a standardized way to compare two road network graphs. These functions take both graphs as input and output a value between 0 (completely dissimilar) and 1 (identical). Mathematically, this can be expressed as:

$$C : (G, G) \to [0, 1]$$

Throughout this analysis, I use the notation where the first argument represents the target (reference) graph T and the second argument represents the source (inferred) graph S.

Different map similarity measures focus on distinct aspects when comparing road network graphs. I use two measures that are commonly used in the research domain. TOPO [6] captures local topological similarity by examining how road curvature overlaps within a specified radius around sample points. In contrast, APLS [7] takes a broader perspective by comparing shortest path distances between randomly sampled position pairs across the entire network to capture global topological similarity.

TOPO.

TOPO evaluates map similarity by computing the harmonic mean of precision and recall across multiple sample points. The evaluation process begins by selecting a position p on the target graph and identifying the nearest corresponding point p' on the source graph. When the distance between p and p' exceeds the threshold ψ, TOPO treats the sample as a complete mismatch with value zero, which significantly reduces the overall recall score.

For cases where points are sufficiently close, TOPO explores the local topology by walking outward from both p and p' within radius ε in all directions. This process traverses all reachable edge curvature and collects positions by sampling every δ meters. This creates point sets P and P' for the target and source graphs respectively. The algorithm considers two points as matching when they fall within distance λ of each other.

TOPO uses a holes-and-marbles analogy to conceptualize the matching process. Holes represent points P from the target graph, while marbles represent points P' from the source graph. A marble "falls into" a hole when the distance between them is within the λ threshold. This approach identifies two types of mismatches: spurious marbles occur when points from P' don't fall into any hole, while missing marbles happen when holes in P remain empty.

The spurious and empty components are calculated as:

$$\text{spurious} = \frac{\#(P' \setminus P)}{\#P'} \qquad \text{empty} = \frac{\#(P \setminus P')}{\#P}$$

Precision measures the percentage of marbles that successfully found a hole, while recall measures the percentage of holes that received a marble. TOPO combines these metrics using the F_1-score, which represents the harmonic mean of precision and recall:

$$F_1 = 2 \cdot \left(\frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \right) = 2 \cdot \left(\frac{(1 - \text{spurious})(1 - \text{missing})}{(1 - \text{spurious}) + (1 - \text{missing})} \right)$$

APLS.

APLS takes a different approach by focusing on path lengths rather than local topological features. Like TOPO, APLS accumulates sample scores, but it computes the final similarity score by averaging all individual sample values rather than using the harmonic mean on precision and recall metrics. Each APLS sample selects two positions a, b on the target graph T and identifies their nearest corresponding positions a', b' on the source graph S.

The algorithm computes the shortest path from a to b in T (curve c) and from a' to b' in S (curve c'). When either endpoint pair exceeds the distance threshold ψ, or when no path c' exists in S despite the existence of path c in T, the sample receives the maximum penalty with a value of zero. For valid path pairs, the sample value reflects path length similarity using the formula: $1 - \min\left\{1, \left|\frac{\|c\| - \|c'\|}{\|c\|}\right|\right\}$.

APLS ensures full coverage by considering all possible node combinations as potential start and end points. Since the road networks in this thesis are undirected graphs, I only need to evaluate half of the possible combinations because opposite directions yield identical path length results.

The main difference between these similarity measures lies in what aspects of network topology they capture. TOPO provides insight into local topological similarity because each sample examines road curvature within a constrained radius around sample seed positions. In contrast, APLS measures global topological similarity since individual samples can traverse large portions of the network when computing shortest paths between distant points.

Both similarity measures exhibit asymmetric behavior, meaning that $\text{APLS}(G, H) \neq \text{APLS}(H, G)$ and $\text{TOPO}(G, H) \neq \text{TOPO}(H, G)$ in general cases. I can demonstrate this asymmetric property through a concrete example:

Consider a scenario where graph H represents a submap containing exactly half of graph G, such as the left half region of a larger network. When H serves as the target (first argument), both similarity measures yield nearly maximum scores ($\approx$1) because any position sampled from the smaller graph finds identical topology in the larger graph G. However, when G serves as the target, approximately 50% of the sampled positions fall outside H's coverage area and get rejected as mismatches. This asymmetry results in TOPO similarity around 0.5 and APLS similarity around 0.25, since APLS can have both start and end points falling outside H's covered region.

2.3. Map Matching

Map matching is a field of research concerned with identifying the path in a road network graph that best corresponds to a given GPS trajectory. These algorithms analyze the geometry and connectivity of potential routes to determine which path through the map most closely aligns with the observed GPS trace.

Various categorization approaches exist for map-matching algorithms [8], [9]. Research in this area is largely driven by the fundamental trade-off between computational speed and matching accuracy, alongside the overarching goal of improving overall accuracy. Real-time applications like navigation systems require algorithms with low computational overhead to operate efficiently without excessive battery drain. In contrast, when analyzing trajectory data for research purposes, the focus shifts toward maximizing accuracy even when this comes at the cost of increased computation time. More recently, AI-based solutions have been proposed [10], [11] which manage to improve accuracy significantly while maintaining performance and reducing computation cost respectively.

Map matching differs fundamentally from our goal of resolving road continuation conflicts (see Section 3.2.2) in its underlying objective. While map matching seeks the best matching path for a given trace, our method requires the algorithm to determine whether the actual underlying path exists in the reconstructed map in the first place.

However we can use map matching to find incorrect road absence. E.g. CrowdAtlas [12]

2.4. GPS-Based Map Reconstruction

A GPS-based map reconstruction method takes as input a collection of n curves $\boldsymbol{C} = \{C_1, ..., C_n\}$ (defined in Section 3.1) and provides a graph G (defined in Section 3.2) as output.

$$M_{\text{GPS}} : \boldsymbol{C} \to G$$

We abstract a GPS trajectory as a curve (defined at Section 3.1), where every point in this curve has a timestamp annotation. Some methods use this additional temporal information to derive speed or acceleration. Note that we represent the set of curves as a set rather than a tensor, because every curve can differ significantly in the number of points it consists of.

2.4.1. GPS data modality

The GPS data modality consists of a collection of GPS traces. A GPS trace is a sequence of GPS samples, where each sample contains a position with a timestamp. A dataset consists of many traces collected by different devices, and each device can have its own timing interval. If timing intervals differ, the resulting dataset is inhomogeneous (and homogeneous otherwise).

By default, a GPS device reconstructs its position from time signals it receives from satellites. The resulting position of the sample can deviate from the actual position, and this deviation is called **sampling noise**.

Any deviation in signal reception is unknown, so unfortunately a GPS device in general cannot infer the accuracy of a sample. GPS sampling noise is generally approximated with an **GPS error model** such as the Gaussian distribution [13], but this does not necessarily hold in practice, making the noise unpredictable. Depending on the context, the rough assumption that GPS errors are contained within some radius deviation often suffices to account for most of the noise that can occur.

The challenge is to take GPS traces in a region of interest and derive the curvature of the roads that have been travelled by cars with GPS devices sampling while driving. This combination of unknown sampling errors and variable sampling intervals means that map reconstruction algorithms must examine a collection of GPS samples (or a collection of GPS traces) to estimate the noise of individual samples and predict the road geometry between samples.

2.4.2. GPS-Based Methods

I divide GPS-based methods into three main categories: density-based, cluster-based, and machine-learning based. All GPS-based methods rely on geometric algorithms to extract features from traces and merge subnetworks together. Even ML-based approaches depend on geometric algorithms to generate input data for their neural networks.

Density-based methods [14–17] first derive a density map using kernel density estimation (KDE) of the trajectories, then process this density information to extract a road network. The density map construction involves projecting trajectories onto a plane, counting trajectories at each pixel, and determining relative trajectory density by convolving with a Gaussian kernel. Both [14] and [15] implement this using a 2D histogram as their KDE mechanism.

Edelkamp and Schrödl [14] pioneered several key processing steps for deriving roadmaps from density maps. Their most significant contribution involves taking contours on a pre-processed KDE and applying the Voronoi graph algorithm against these contours to extract middle lines representing road curvature. They use post-processing shaving to handle minor artifacts from the Voronoi graph algorithm.

[15] takes a different approach, using dilation and erosion to iteratively thicken and thin the pre-processed KDE lines so that roads connect better while remaining thin. To derive the actual road network, they introduce the combustion technique: iteratively walking forward on unexplored grid cells of the bitmap to find all edges.

[16] advances the field by applying skeletonization at different density levels to create gray-scale rather than binary images. This approach better captures roads with significant differences in traversal frequency.

[17] represents a modern take on density-based methods by converting trajectory data into a 12-band image that feeds into a CNN to infer a road mask, then converts that into a road map through vectorization. These 12 channels extract various trajectory properties including traversal count, speed, and acceleration. This method can potentially extend to early-fusion multi-modal solutions by creating a 15-band image that includes satellite imagery, an approach widely used in road masking (section Section 2.6.1).

Cluster-based methods [18–22] create clusters from trajectories and use these clusters to derive road networks.

Brakatsoulas et al. [18] established the foundation by initiating clusters at trajectory points and iteratively settling them to final positions by moving toward nearby trajectories with similar directions. This process continues until all trajectory points are near a cluster, after which these clusters are connected into a road map.

[20] introduces clustering based on turn types, identifying eight turn combinations of incoming and outgoing directions (south, east, north, west orientations). These turns undergo bottom-up agglomerative hierarchical clustering by type within a 50-meter radius, then connect based on whether clusters share trajectories.

[21] focuses on spatial linear clusters—trajectories near each other moving in approximately the same direction. The algorithm first computes anchor positions where sufficiently nearby trajectory points move in the same direction. It then constructs spatial linear clusters by starting at unused anchor positions and iteratively extending clusters with nearby anchor positions moving in similar directions. Road segments emerge by taking moving averages of anchor positions within each spatial linear cluster.

[22] improves upon [21] by taking a more sophisticated approach to subtrajectory cluster construction. Rather than simply looking for trajectory points moving in the same direction, it identifies clusters of subtrajectories below a Fréchet distance threshold. We will examine this method in detail at Section 2.4.3.

Recent papers [23, 24] act directly on traces using customized algorithms that belong to neither density- nor cluster-based categories, but instead represent iterative graph construction approaches.

RoadRunner [23] focuses on constructing accurate maps in dense urban areas and complex intersections. The method builds graphs iteratively by selecting trajectory points at unexplored vertices, then moving in the average direction determined by trajectories that traversed the same road using a way path filter. This filter places circles along trajectories of interest and selects trajectories intersecting all circles. From filtered trajectory sets, it computes prominent outgoing directions and places new vertices toward those directions. To determine whether two way path filtered sets merge at road intersections, the method checks for similarity in traversal distribution and density continuation between trajectory sets.

[24] focuses on reconstructing cycling roads to improve last-mile delivery for food services. The method links nearest GPS points individually, weighting each edge by distance squared, then converts this into a network by connecting points using shortest-path algorithms that promote dense regions due to their lower-weight links.

2.4.3. Roadster

To our knowledge, the state-of-the-art GPS-based map reconstruction method is developed by Buchin et al [1]. We will use it as our base method of choice in our multi-modal map reconstruction implementation (see Section 4.2) and therefore consider it worthwhile to explain the method in more detail.

Method outline.

Roadster hypothesizes that road segments can be identified as representations of subtrajectory clusters in the GPS traces. The method performs reconstruction in two steps: first computing bundles (subtrajectory clusters) [25] and then merging their representations into a map. This algorithm contains improvements both in reconstruction speed and quality compared to their initial release of this method in 2017 [22].

Bundle.

Finding bundles (subtrajectory clusters) is a problem defined and solved by Buchin et al in 2008 [25]. The concept of a bundle is a collection of subcurves where the curve distance (between any pair in the collection) is below a threshold of ε. Their paper approaches various variants of the subtrajectory clustering problem, but we focus on bundling that computes the continuous Fréchet distance and considers a collection of curves (trajectories) *without* reoccurrences, so a bundle can contain at most one subcurve of any given curve. The improved version uses a force-based method which results in better quality [1].

A **bundle** is a relevant cluster of subtrajectories [22]. Being **relevant** means the subtrajectory cluster is constrained to meet properties of being maximal, stable, and large (all defined by the paper [22]).

- A cluster is **large** if it contains k (by default $k = 3$) or more curves.
- A cluster is **stable** if the current value of ε has to change (increase) "significantly" for the cluster size to increase. Significance is expressed by lifespan or relative lifespan: **Lifespan** describes the maximal minus the minimal ε at which a cluster exists, and dividing this number by the minimal value results in the relative lifespan. So for any cluster, its size starts at a value of ε and ends at ε', and its relative lifespan is $\frac{\varepsilon'-\varepsilon}{\varepsilon}$. The default relative lifespan value for a cluster to be considered stable is 1 or larger.
- A cluster is **maximal** if the ε for the current cluster size is maximal, so given a cluster size of k being maximal, for all $\varepsilon' > \varepsilon$ we will end up with a cluster size larger than k. These parameter constraints are found by the algorithm itself; the algorithm only requires a single parameter ε to work with, which acts as an initial distance threshold.

From a cluster of subtrajectories, a **representative** is derived to reduce the set (the bundle) to a single polygonal curve by taking the center line of the subtrajectory cluster.

Intuition.

The road network is derived by connecting bundle representatives (curves), which implies each edge in the network meets the three bundle properties (in order to be relevant). This suggests an expectation of robustness in the inference as follows:

- The edge is large, which prevents constructing an edge for outliers.
- The edge is stable, which suggests we aren't missing out on additional trajectories to derive a bundle from.
- The edge is maximal, which suggests we have taken the largest possible common subcurve along the trajectories involved.

Results.

The method manages to properly extract the underlying roadmap (Figure 1) under various circumstances, such as coping with local changes in the number of trajectories and road width (see Figure 2, Figure 3). However, sometimes it experiences confusion and has scrambling (Figure 4) or misalignment (Figure 5), usually caused by GPS noise, difficult road layout, or differences in start/end positions of trajectories (Figure 6).

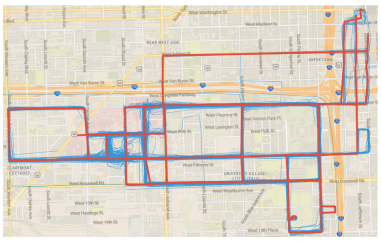


Figure 1: Inferred map by Roadster [1] from the Chicago dataset [6].

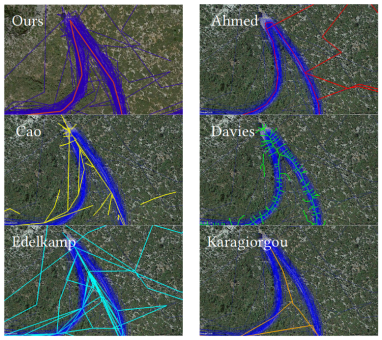


Figure 2: Roadster correctly reconstructs a sharp turn under noisy GPS data [1], [26].

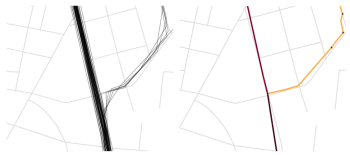


Figure 3: Correct representation of a T-crossing even though the roads have different numbers of trajectories. [1]

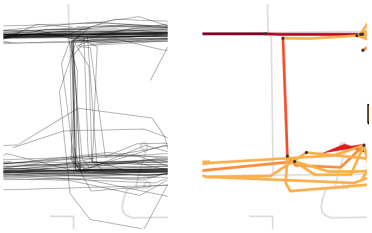


Figure 4: Edge scrambling due to noise at a T-crossroad [22]

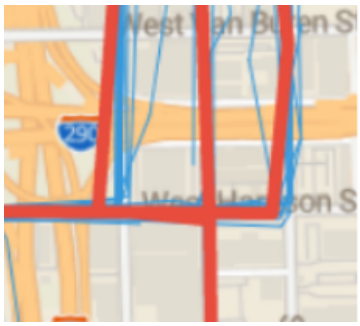


Figure 5: Minor misalignments at some crossings [1]

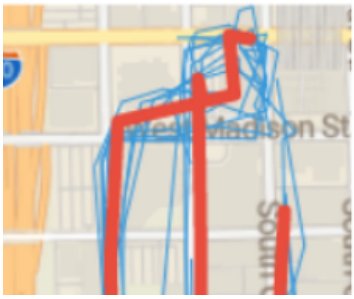


Figure 6: Scrambling where trajectories start/end [1]

2.5. Satellite-based Map Reconstruction

A satellite-based map reconstruction method takes an image as input and provides a map as output. We limit our analysis to three-band RGB images, which can have arbitrary width and height dimensions. Mathematically, we can describe the satellite-based map reconstruction function as:

$$M_{\text{SAT}} : \boldsymbol{I} \rightarrow G$$

with image $\boldsymbol{I} \in \mathbb{R}^{w \times h \times 3}$ a tensor with width w and height h, and G a graph as defined in Section 3.2.

2.5.1. Satellite image data modality

Satellite images can achieve extremely high quality. Quality is expressed through **GSD** (Ground Sampling Distance): the number of meters per pixel in the image. Modern satellites can capture additional light bands outside the visible spectrum, which enables more detailed terrain information capture, but we limit our data to standard three-channel RGB images.

Road infrastructure detection with satellite images can suffer from visual obstructions such as trees, tunnels, and stacked roads. Often the obstruction occurs only at small road segments, so methods can solve this by inferring road continuity through the obstructed areas. However, if methods are too optimistic, this results in constructing road segments that do not exist. Therefore, models must balance how eagerly they infer road continuity at visually obstructed locations. Methods have their own approaches to this trade-off and attempt to make informed decisions based on available visual cues. In Table 1 we see results where both connections are missed and false connections are inferred.

Specific environments, such as building placement or tree coverage, can create unique conditions that give methods the illusion that roads exist (or don't exist) contrary to ground truth. This can be countered by improving the visual models that reconstruction methods use, leveraging more information from visual models, and training on more diverse data to increase exposure to less frequently occurring environments. In Table 1, the train track was incorrectly interpreted as a road, which suggests that training data including such examples could help the model detect these deviating patterns.

Local details are inherent to satellite images, enabling methods to properly distinguish between connection types (crossings versus roundabouts), capture detailed road curvature, and identify the number of lanes in roads. In Table 1, road curvature is often captured with great detail.

2.5.2. Satellite-Based Methods

Image-based methods generally use neural networks to extract features from images, which are then processed into road network reconstructions.

Literature categorizes methods as segmentation-based and graph-based approaches. Segmentation-based methods try to extract the map from a segmentation (road mask), while graph-based methods instead act on inferred graph data to extract the map data.

I find it useful to instead describe these as one-shot inference methods and iterative construction methods. One-shot methods extract the complete map from a single pass through the satellite image, while iterative methods construct the road network step-by-step by moving along inferred graph data and reconsidering windowed portions of the satellite image.

One-shot methods [27], [28], [2], [29], and [30] process each image separately to vectorize road networks, then merge subnetworks into a single road network.

Iterative methods [31], [32], [33], and [34] iteratively extend the road network constructed at each step. This provides models with additional information (the road network under construction) to make more informed decisions on further extensions.

Note that iterative map reconstruction methods differ fundamentally from recurrent neural networks (RNNs). Iterative methods act on inference results from previous steps rather than iterating on hidden internal states in latent space like RNNs. Therefore, iterative methods hold no internal state and can restart their process at any step from an arbitrary position on a partially reconstructed map. With basic state information (such as unprocessed vertices), iterative methods track what points remain to be evaluated. In other words, while an RNN relies on iterating on its internal state, an iterative method does not hold any internal state and restarts its process every step with an updated partial map it has constructed up to that point (excluding some data like what points are left to be processed).

Progression in one-shot methods.
DeepRoadMapper [27] serves as the foundational method, using a CNN to extract road masks and applying post-processing tools such as thinning and reconnection methods to fill missing road segments.

SMBM [28] introduces joint learning of road orientation and segmentation using a single encoder, then trains a second CNN to refine connectivity in the derived road segmentation using the inferred road orientation.

Sat2Graph [2] introduces a graph tensor encoding to capture graph information in image-like format, with vertex and edge probabilities encoded at every pixel and a CNN trained to link satellite images with graph tensor representations.

PaRK-Detect [29] predicts road probabilities, directional connectivity to eight neighboring patches, and intersection positions in small patches, then algorithmically converts this information into graphs with post-processing optimizations.

SAMRoad [30] connects three neural networks: SAM [35] encodes satellite images into latent space, a CNN converts this to per-pixel road and intersection probabilities, and a transformer predicts connectivity among inferred points based on spatial layout and image context. Remarkably, SAMRoad not only outperforms competitors with significant margins (5.8% higher precision than RNGDet++) but requires no manual post-processing steps to patch prediction deficits.

Progression in iterative methods.
RoadTracer [31] picks small satellite image regions centered on inferred road points and uses CNNs to decide walking directions for road discovery, typically in 12-meter steps, tracking visited points to discover connected road graphs.

VecRoad [32] improves this approach by providing additional CNN information (road curvature and intersection points), reducing curvature misalignment accumulation and enabling dynamic step sizes.

RNGDet [33] uses similar information to guide transformer attention mechanisms that predict sets of adjacent vertices, while RNGDet++ [34] improves results by incorporating multiscale feature backbone information and adding instance segmentation heads.

Methods improve significantly over time, but most performance gains correlate with computer vision and neural network advances rather than specific map reconstruction algorithms. For instance, RNGDet [33] adds transformers [36] and outperforms Sat2Graph's convolutional approach [37]. RNGDet++ [34] improves RNGDet through multiscale backbone features and instance segmentation heads, both neural network architectural improvements. The state-of-the-art SAMRoad [30] performs even better, with authors attributing success to effective interfacing with the Segment-Anything model [35].

2.5.3. Sat2Graph

We use Sat2Graph [2] as our base method for multi-modal map reconstruction implementation (see Section 4.2), making detailed explanation worthwhile.

Sat2Graph uses a custom intermediate representation called a **graph tensor** that aims for isomorphism with road maps. This tensor can be interpreted as a two-dimensional "image" with features at every pixel describing vertex probabilities and related edge connections (probabilities for up to five edges including directivity). Having one-to-one correspondence between image pixels and graph features allows neural networks to learn pixel-feature relationships conveniently. Since graph tensors are isomorphic with maps, they can be directly converted into road networks, though additional post-processing steps are necessary to handle inference noise and errors.

A neural network is trained against ground truth road networks to learn relationships between images and graph tensors, establishing mappings between pixels and graph tensor vectors. New image data can then be processed through the neural network to infer graph tensors, which are subsequently transformed into road networks. Processing improvements include choosing local vertex maxima, using custom distance measures for vertex connectivity decisions, and enforcing minimum and maximum inter-vertex distance constraints.

Table 1: Ground truth (left) and inferred road network (right) on RNGDet++ [34].

2.6. Multi-Source Map Reconstruction

Building on GPS-based and satellite-based approaches, multi-source map reconstruction tackles the challenge of reconstructing maps using *both* GPS trajectories and satellite images as input. Using the definitions from GPS-based (Section 2.4) and satellite-based (Section 2.5) reconstruction, we can describe this task mathematically as:

$$M_{\text{MUL}} : (\boldsymbol{C}, \boldsymbol{I}) \rightarrow G$$

where we have a collection of n curves $\boldsymbol{C} = \{C_1, ..., C_n\}$, an image $\boldsymbol{I} \in \mathbb{R}^{w \times h \times 3}$ as a tensor with width w and height h, and G a graph as defined in Section 3.2.

Multi-modal map reconstruction literature includes [38–43]. Most papers focus on constructing road masks (see Section 2.6.1) from multi-modal data rather than directly generating road networks. Some papers process these masks further into actual road maps, such as DeepDualMapper [39] and the deep-fusion stage of DelvMap, but they rely on standard techniques like shaving [14] or combustion [15]. This approach doesn't provide new methodological insights for graph reconstruction beyond existing unimodal techniques.

Multi-source reconstruction can be approached through three main fusion strategies.

- Early fusion (LCGD [38]) combines satellite images and GPS trajectory data as input to the reconstruction method. The typical early fusion approach uses a vision network that processes a satellite image with additional dimensions to capture road trajectory data as extra image channels.
- Deep fusion (DeepDualMapper [39], CMMPNet [41], DICN [42], DelvMap [43]) begins by processing both data modalities separately, then fuses intermediate data representations or embeddings during inference. The neural network layers applied on both modalities exchange information at various stages, allowing each modality to inform the other throughout the reconstruction process.
- Late fusion (DuARE [40], DelvMap [43]) applies computational geometry to combine two graphs generated independently from each data modality.

I focus on late-fusion methods in more detail since this represents the multi-modal reconstruction category that this thesis addresses.

DuARE.

DuARE [40] applies late-fusion for map reconstruction through a three-step procedure: direct fusion, cross-check fusion, and topology reconstruction. The direct fusion step extracts edges that occur in both graphs, then adds satellite edges showing similarity with GPS traces. The cross-check fusion step includes satellite edges that agree with at least two GPS traces matched by longest common subsequence. These edges didn't qualify in the first step because GPS map reconstruction failed to infer them due to insufficient GPS traces in the region. Finally, the topology reconstruction step reduces duplication by placing "subnodes" at segment intersections, clustering those within 5 meters, and deriving a representative from each cluster to maintain a single node. This approach provides certainty when both graphs agree on an edge's presence, while enabling recovery of satellite edges on roads where GPS data was sparse.

DelvMap.

DelvMap [43] employs a hybrid approach that combines deep learning with late-fusion components. The method first applies deep learning to derive an initial map, then extends this map with late-fusion techniques in areas where only GPS reconstruction inferred edges. Since the deep fusion step merely extracts a road mask through standard post-processing techniques, this aspect holds limited relevance to my thesis focus.

The late-fusion component demonstrates how multiple data modalities can be combined through late fusion, making it particularly relevant to my thesis. This component extends the satellite-based graph by incorporating missing cycling roads that appear only in GPS-based reconstruction. The algorithm targets subregions that satellite-based reconstruction overlooked, typically off-road areas between buildings where cycling paths exist but lack visual clarity in satellite imagery.

This approach stems from their dataset's characteristics, where areas containing buildings often have food delivery services cycling through them, implying existing cycling infrastructure. Their deep fusion algorithm purposefully avoids inferring roads at building pixels by generating a building mask that prevents road inference at such locations. This design choice creates a "guaranteed" disagreement between road network reconstructions at building areas.

To recover these missing cycling roads, DelvMap implements a map completion algorithm that operates in several steps. First, it identifies GPS graph vertices that are sufficiently distant from existing satellite graph edge geometry, indicating potential missing roads. Starting from such a disagreement vertex, the algorithm iteratively collects adjacent vertices from the GPS-based graph that are also sufficiently remote from satellite-based graph edge geometry. This collection process continues until the algorithm encounters a vertex sufficiently close to existing satellite graph geometry. At this connection point, it integrates the collected path segment into the satellite-based map, thereby extending it with the missing GPS-inferred road. This process continues until all remote GPS graph vertices have sufficiently nearby edge geometry in the fused map.

Limited Evaluation.

Despite the variety of fusion approaches, evaluation methodologies appear to present challenges across the field. Most early fusion and deep fusion algorithms extract road masks without extracting road networks for comprehensive evaluation. Late fusion methods also seem to have limited comparison scope in their evaluations. For instance, DuARE compares only to Sat2Graph [2] using what appears to be an uncommon and unspecific map similarity evaluation technique. They use ontology relaxed precision and recall [44] as a map similarity

metric, which requires modifications to support embedded graph comparisons, but I cannot find documentation of how DuARE implements this. Based on the available information, it seems they might compare recall and precision on edges using a proximity-based matching criterion where an edge matches if it has *any* nearby edge curvature from the opposing graph. This could potentially represent a weaker similarity measure compared to TOPO or APLS, though without full implementation details it's difficult to be certain. Similarly, DelvMap appears to apply only TOPO for evaluation.

This apparent lack of comprehensive evaluation makes it challenging to assess the true benefits of multi-modal approaches over state-of-the-art unimodal methods.

2.6.1. Road Masking

This research field focuses on extracting road masks from satellite images. **Road masking** is a binary classification task that annotates every pixel of a satellite image based on whether the pixel is part of a road. Given an RGB image $\boldsymbol{I} \in \mathbb{R}^{w \times h \times 3}$ with width w and height h, we obtain a binary image $\boldsymbol{J} \in ([0,1])^{w \times h}$:

$$M : \boldsymbol{I} \to \boldsymbol{J}$$

While road mask extraction can serve as a relevant substep in map reconstruction (typically vectorized into a graph through processes like thinning and combustion), these papers lack proper evaluation methods in their experiments to demonstrate reconstruction impact. Instead, they focus on pixelwise similarity, which makes sense for evaluating road masking performance but doesn't indicate whether the road mask will result in a good road map. Furthermore, road masks lack information needed to infer overpasses (and directionality, though that's outside this thesis scope).

3. Definitions

This section provides definitions for curves (Section 3.1), graphs (Section 3.2) and paths (Section 3.3). These definitions are necessary to describe the proposed map fusion algorithm (Section 4.2) and its core component edge coverage (Section 4.1).

I took inspiration from the paper by Gudmundsson, Seybold and Wong [45] for definitions on curves and paths.

3.1. Curves

In this thesis all curves are polygonal curves: piecewise linear curves constructed from a sequence of points. There are two inconsistencies to resolve: curves can differ in length and the lengths of linear line segments can differ. By introducing a normalization function, any position on the curve can be accessed in an interval of 0 to 1 irrespective of the total curve length with uniform speed (magnitude of derivative at non-singularities) everywhere along the curve.

A **curve** C is defined as a continuous function, mapping all real values of the unit interval onto two-dimensional space:

$$C : [0,1] \to \mathbb{R}^2$$

The function space is limited to curves constructed from polygonal curves Q composed with a normalization mapping ψ, both described below.

A **polygonal curve** Q_P is a piecewise-linear function for a sequence of points P. Given $P = \{\boldsymbol{p}_i \in \mathbb{R}^2\}_{i=2}^{n}$ consisting of n points with $n \geq 2$, we have

$$Q_P : [1,n] \to \mathbb{R}^2$$

satisfying $Q_P(i) = \boldsymbol{p}_i$. The **linear interpolation function** for the piecewise linear components satisfies

$$Q_P(i+\mu) = (1-\mu)\boldsymbol{p}_i + \mu \boldsymbol{p}_{i+1}$$

for integers $i \in [1, n-1]$ and reals $\mu \in [0, 1]$.

A polygonal **subcurve** is defined as the construction of a polygonal curve $Q_{P'} = Q_P([a, b])$ from a point sequence $P' = \big(Q_P(a), P_{i_1}, ..., P_{i_m}, Q_P(b)\big)$ derived from P with any real interval $[a, b]$ with $1 \leq a < b \leq n$ where $i_1, ..., i_m$ are all integers within the interval $[a, b]$. Note $Q \equiv Q([0, 1])$.

The **length** of the polygonal curve Q_P is defined as the accumulation of interpoint distances and uses the Euclidean norm:

$$\|Q_P\| = \sum_{i=1}^{n} \|\boldsymbol{p}_i - \boldsymbol{p}_{i-1}\|$$

For convenience and consistency in the input range irrespective of the number of points in the polygonal curve, I apply a **normalization mapping** ψ to map the unit interval to the input range of the polygonal curve function. Given a sequence of points P consisting of n points, the mapping function ψ_P is a strictly increasing and bijective function

$$\psi_P : [0, 1] \to [1, n]$$

such that a polygonal curve can be converted into a curve by composition of ψ_P and Q_P

$$C_P = Q_P \circ \psi_P : [0, 1] \to \mathbb{R}^2$$

The result is a unique mapping of ψ for any curve $C = Q \circ \psi$ to satisfy the following constraint:

$$\forall a, b \in [0, 1], a < b : \quad \|C([a, b])\| \equiv (b - a)\|C\|$$

This mapping provides uniform speed: take for instance half the unit interval $\left[x, x + \frac{1}{2}\right]$ at arbitrary starting position $x \in \left[0, \frac{1}{2}\right]$, then the resulting subcurve length of $C\big(\left[x, x + \frac{1}{2}\right]\big)$ is half the curve length irrespective of where the interval starts. In other words, the mapping ψ ensures that the speed/derivative at every point of all piecewise linear segments (thus at any non-singular point) is constant.

Every mention of a curve C implies this curve to be constructed from a sequence of points P and to thereby have a unique definition of its polygonal curve Q_P and a unique normalization mapping ψ_P.

Error margin.

The discrete technique is sensitive to point placement and can produce greatly varying distance values even when the resulting curvature (drawn with straight line segments) is identical. For instance, two nearly identical curves could show significant distance differences due to different point placement, which contradicts expectations.

While the deviation depends on the specific case, the maximum potential error can be computed, called the **discrete error margin**. Given a Fréchet distance between two polygonal curves C_P and C_Q of $D_F(C_P, C_Q) = d$ and an error threshold λ, the discrete Fréchet distance is guaranteed to satisfy:

$$D_F(C_P, C_Q) - \lambda \quad \leq \quad \widetilde{D_F}(C_P, C_Q) \quad \leq \quad D_F(C_P, C_Q) + \lambda$$

3.2. Graphs

The graph data structure serves as the abstraction to capture the road network, following most literature on this topic. However, there are options for what properties such graph should have. Any usage of a graph in this thesis refers to an embedded, (potentially) disconnected, simplified (see Section 3.2.1), undirected multi-graph, and this is used interchangeably with **(road) map**, **(road) network**, or just **graph**.

A **graph** $G = (V, E)$ is a pair of **nodes** $V = \{v_i \mid 1 \leq i \leq n\}$ and **edges** $E = \{e_j \mid 1 \leq j \leq m\}$. The set of nodes and edges specific to a graph G can be mentioned explicitly with V_G and E_G.

Nodes and edges have attributes associated with them. These properties are denoted with a bold capitalized subscript to prevent confusion on identifiers (e.g. the i-th edge e_i and its geometry e_{i_C}). Every node v has a position $v_{\boldsymbol{P}} \in \mathbb{R}^2$ and has a degree $v_{\boldsymbol{D}}$ that describes the number of edges connected to that node. Every edge

e has a start node $e_{\boldsymbol{A}} \in V$, end node $e_{\boldsymbol{B}} \in V$, a polygonal curve $e_{\boldsymbol{C}}$ starting at $e_{\boldsymbol{A}}$ and ending at $e_{\boldsymbol{B}}$, and a continuation boolean value (see Section 3.2.2).

Below I go into more detail on each property.

Graphs are **undirected** where the order of nodes of an edge is irrelevant and an edge can be traversed in both directions, because directionality reconstruction falls outside the scope of this thesis. Additionally graphs are allowed to be **disconnected** (so it can consist of multiple subgraphs which have no connection with one another), because this can be the result of a map reconstruction.

Every graph is **embedded** in the two-dimensional plane, so every node has a two-dimensional position in Euclidean space. In practice, road maps are embedded with world coordinates on a sphere and not in Euclidean space, however these coordinates can be transformed into Mercator coordinates which approximate Euclidean space.

3.2.1. Simplified and vectorized

Two graph variants are used: simplified and vectorized, as mentioned by OSMNx [46]. In a **vectorized** graph all edges are direct line segments between two nodes. In a **simplified** graph all edges have a geometry attribute and nodes are only present at road intersections and dead-ends. The geometry of an edge e consisting of n vertices has the point sequence $e_{\boldsymbol{A}}, p_2, ..., p_{n-1}, e_{\boldsymbol{B}}$. Note that it always consists of at least two vertices (namely $e_{\boldsymbol{A}}, e_{\boldsymbol{B}}$).

A vectorized graph is necessary when functions expect edges to be a straight line segment between two nodes. A simplified graph is convenient when traversing a graph, because traversal takes fewer jumps with the edge geometry abstracted away. Additionally a simplified edge better represents a road in the road map.

3.2.2. Edge continuation

An important concept for the proposed map fusion algorithm (Section 4.2.1) and its underlying motivation is road continuation. I use this terminology deliberately to distinguish it from edge connectivity or the mathematical concept of continuity. Edge connectivity can occur indirectly through detours, which fails to capture whether a specific road segment actually continues beyond its endpoints. Similarly, while every edge exhibits geometric continuity along its line structure, applying this mathematical concept to describe relationships between separate edges creates ambiguity.

Given a simplified undirected graph $G = (V, E)$, an edge $e \in E$ has continuation if both of its endpoints connect to other edges. Formally, for an edge e with endpoints v_A and v_B:

$$\text{continuation}(e) = \deg(v_A) > 1 \wedge \deg(v_B) > 1$$

This definition applies specifically to the simplified graph structure.

3.2.3. Map Density

Map density is defined as kilometers of edge per square kilometer of surface area. Sparsity therefore refers to relatively low map density, where fewer road segments are captured per unit area. In our experiments, this sparsity is most clearly observed in the Chicago GPS map reconstruction (see Table 19), where the GPS-derived map appears notably sparse compared to both the Chicago SAT and Chicago OSM maps.

3.3. Paths

A path is a walk along edges of a graph. I define a **path** s (consisting of n steps) for a graph G as a sequence of node-edge pairs $s = (v_1, e_1), ..., (v_n, e_n)$ starting at node v_1 and ending at node v_{n+1} that is connected by v_n with e_n (thus $v_{n+1} \equiv e_{n_{\boldsymbol{B}}}$).

A path has the requirement that it provides a connected sequence of node-edge pairs, so the resulting path geometry (described below) is continuous. Mathematically this means that for all $i \in [1, n]$ holds $e_{i_{\boldsymbol{B}}} \equiv v_{i+1}$.

The **path geometry** (a polygonal curve) $s_{\boldsymbol{C}}$ of the path s is obtained by concatenating the edge geometries of the edges in the path. Mathematically notated as a path $s = (v_1, e_1), ..., (v_n, e_n)$ with its geometry $s_{\boldsymbol{C}} = \text{concatenate}\left(e_{1_{\boldsymbol{C}}}, ..., e_{n_{\boldsymbol{C}}}\right)$. The order of traversal is taken into account implicitly and the concatenate function

drops the end vertex of most edge geometries (except the last) to prevent duplicated vertices in the resulting curve.

The path description of a simplified graph requires a sequence of node-edge pairs to prevent ambiguity. Edges are needed in the description to prevent ambiguity of what edges are involved in the node sequence of a multi-graph, and nodes are needed to prevent ambiguity in the traversal direction.

4. Method

I propose a late-fusion multi-modal map reconstruction method in Section 4.2 that fuses together two reconstructed maps from GPS traces and satellite images separately. The fusion process works by replacing edges of the SAT map (reconstructed from satellite images) with edges of the GPS map (reconstructed from GPS trajectories).

To determine which edges to add, I apply a mechanism I call edge coverage (described in Section 4.1), which computes whether the geometry of an edge is covered by a map. Essentially, edge coverage works by map matching the edge geometry against the map and deciding coverage based on a distance threshold.

During evaluation (see Section 6.7.1), I discovered that existing map similarity metrics fail to provide meaningful insights when either map is sparse (see Section 3.2.3). I have adapted TOPO and APLS to ignore samples if either map lacks road geometry near the opposing graph (defined in Section 4.3). This better captures similarity by removing the statistical significance of maximally penalizing samples.

4.1. Curve Coverage

Curve coverage is a mechanism I define to check for agreement of edge continuation (Section 3.2.2) between two graphs. It operates by map matching a curve against a graph with a maximal accepted distance value.

Curve coverage computes whether there exists any path s in the graph G that has geometry with a Fréchet distance to a curve C below a given distance value ε.

$$f_{\text{coverage}} : (G, C, \varepsilon) \to (s, [a, b]) : d_F(s_{\boldsymbol{C}}[a, b], C) < \varepsilon$$

This checks for the existence of and returns a minimal path s in G alongside a minimal subcurve $s_{\boldsymbol{C}}[a, b] \subset [0, 1]$ such that the Fréchet distance between $s_{\boldsymbol{C}}[a, b]$ and C is smaller than ε.

A minimal path implies that the length of the path geometry is minimal. Minimalization of the subcurve takes precedence over minimalization of the path.

Examples of coverage and no coverage are given in Figure 7 and Figure 8.

Edge coverage is curve coverage applied to its geometry. My implementation is described in Section 5.3.2. It is used in the map fusion algorithm (Section 4.2.1).

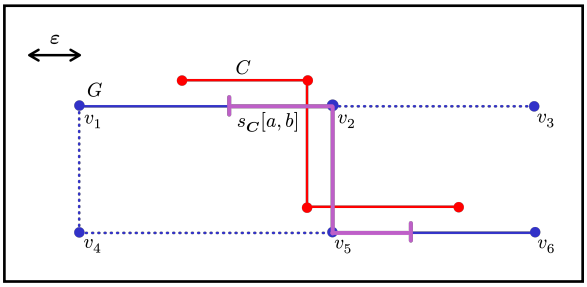


Figure 7: Curve C (red line) is covered by graph G (dotted blue lines) by the minimal path $s = (v_1, v_2, v_5, v_6)$ (blue line) on its subcurve $s_C[a, b]$ (purple line), because the Fréchet distance is below the distance threshold ε.

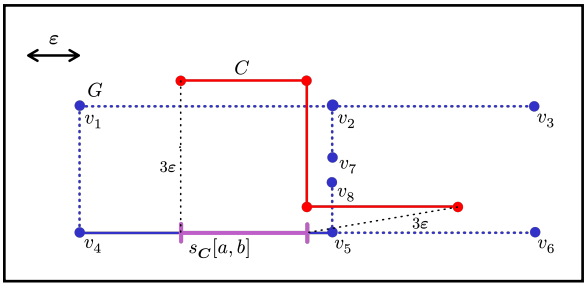


Figure 8: Curve C (red line) is *not* covered by graph G (dotted blue lines), because the path $s = (v_4, v_5)$ (blue line) with subcurve $s_C[a, b]$ (in purple) with lowest Fréchet distance to C exceeds the distance threshold ε.

4.1.1. Design choices

I allow taking a subcurve of the path geometry because when considering map data, some simplified edges can be very long compared to the curve seeking coverage. By allowing a subcurve of the path geometry, the definition can find a path that covers such a curve. This effect occurs in Figure 7 at edge v_1, v_2.

As a similarity measure, I consider it necessary to use the Fréchet distance rather than the Hausdorff distance to check for agreement in road continuation. For instance, in Figure 8, if the Hausdorff distance was used instead, then the curve would be covered by the path v_1, v_2, v_7 combined with the path v_8, v_5, v_6 as we can jump from v_7 to v_8. An additional problem with the Hausdorff distance is that it does not result in a single path related to this solution.

4.2. Map Reconstruction

I propose a late-fusion multi-modal map reconstruction method. The complete reconstruction process first generates a GPS and SAT map separately, then applies map fusion to merge both graphs into a single unified graph. The contribution focuses on the map fusion process, while both the GPS and SAT maps are generated using existing research methods.

4.2.1. Map Fusion

The procedure fuses two unimodal map reconstructions through three sequential steps. One unimodal map serves as the **base map**, while the other acts as the **patch map** that enhances the base map. By default, the SAT map is chosen as the base map and the GPS map as the patch map. The three map fusion steps are performed sequentially in batch processing (each step is executed exactly once):

1. **Inject**: Inject GPS edges into the SAT graph that lack coverage by the SAT graph. First, uncovered GPS edges are identified, resulting in GPS subgraph(s) for injection. Every node in this subgraph is checked for degree reduction compared to the original GPS graph, and each such node connects to the SAT graph using the connect function (Section 4.2.3.5).
2. **Delete**: Remove SAT edges that have coverage from the injected GPS edges of step 1.
3. **Reconnect**: Identify nodes that experienced degree reduction in step 2 and connect these to the injected GPS subgraph from step 1.

Each step of the map fusion procedure receives detailed explanation below. Steps are visualized in Table 2 and the algorithm appears in pseudo-code at Table 3.

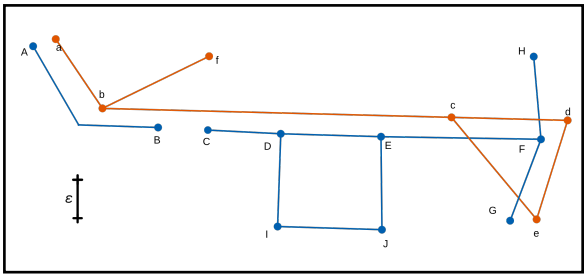


Figure 9: Overview of graphs (SAT graph is blue, GPS graph is red). The SAT graph has a discontinued edge at *B* and *C*, in contrast to the GPS edge *b*-*c* which is continuating.

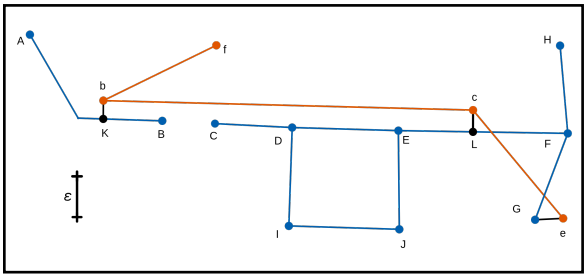


Figure 10: Step 1: Inject simplified GPS edges. *a*-*b*, *c*-*d*, *d*-*e* are filtered out because they have coverage. *f* does not reconnect because it did not experience degree reduction. *e* reconnects to a nearby node, while *b* and *c* reconnect to nearby edge geometry.

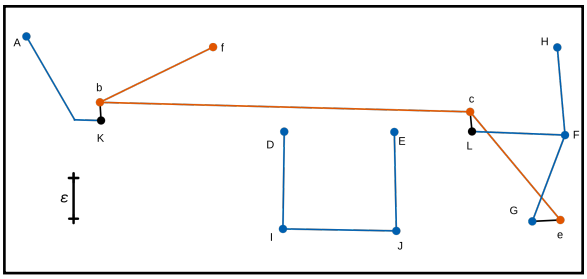


Figure 11: Step 2: Remove duplicated SAT edges. SAT edges *K*-*B*, *C*-*D*, *D*-*E*, *E*-*L* have coverage from the GPS edge *b*-*c* and therefore get deleted.

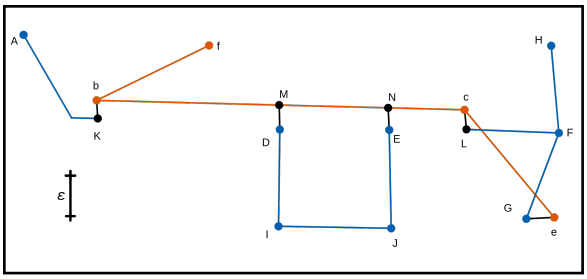


Figure 12: Step 3: Reconnect SAT edges to injected GPS edges. SAT edges *D*-*I* and *E*-*J* had dangling edge endpoints (at *D* and *E*) and reconnect to a nearby GPS edge. In this case both *D* and *E* connect to GPS edge *b*-*c*.

Algorithm 1: Map Fusion

procedure MAP FUSION(G_{SAT}, G_{GPS}, ε)

▷ Step 1: Inject uncovered GPS edges.
$G_{\text{diff}} \leftarrow G_{\text{GPS}} - \textbf{graph\_coverage}(G_{\text{SAT}}, G_{\text{GPS}}, \varepsilon)$
$V_{\text{conn}} \leftarrow \textbf{degree\_reduced}(G_{\text{GPS}}, G_{\text{diff}})$
$G_1 \leftarrow G_{\text{SAT}} \cup G_{\text{diff}}$
$G_1 \leftarrow \textbf{connect}(G_1, V_{\text{conn}}, \textit{exclude} = G_{\text{diff}})$

▷ Step 2: Delete duplicated edges of G_1.
$G_{\text{cov}} \leftarrow \textbf{graph\_coverage}(G_{\text{SAT}}, G_1, \varepsilon)$
$G_2 \leftarrow G_1 - G_{\text{cov}}$

▷ Step 3: Reconnect dangling SAT edge endpoints.
$V_{\text{rec}} \leftarrow \textbf{degree\_reduced}(G_{\text{SAT}}, G_2)$
$G_3 \leftarrow \textbf{connect}(G_2, V_{\text{rec}}, \textit{exclude} = G_{\text{SAT}})$

return G_3
end

Table 3: Pseudo-code for the map fusion algorithm. Any changes to graphs implicitly perform graph simplification afterwards to maintain a simplified graph (see Figure 12 as an example requiring simplification).

1. Edge Injection.

This step identifies GPS edges lacking SAT coverage and injects them into the SAT graph.

Starting with both simplified graphs G_{SAT} and G_{GPS}, the process first identifies uncovered GPS edges and groups them into connected subgraphs. Next, it determines which GPS nodes in these subgraphs should connect to the SAT graph by filtering for nodes with reduced degree, as dead-ends in the GPS graph remain unconnected

to avoid artifacts. Finally, the filtered GPS subgraph endpoints connect to the nearest SAT geometry, ensuring connections target SAT elements rather than the GPS subgraph itself.

The injection process follows these substeps:
1. Identify GPS subgraphs to inject into SAT
2. Find nodes from step 1 that have reduced degree (Section 4.2.3.4)
3. Locate connection points to the SAT graph for each qualifying node (Section 4.2.3.5)
4. Execute all scheduled connections in batch

2. Edge Removal.

After edge injection, the graph may contain duplicated edges where discontinuating SAT edges lie adjacent to continuating injected GPS edges. This step removes these redundant SAT edges by checking each original SAT edge for coverage against the injected GPS subgraphs.

Simply deleting all covered SAT edges would remove unrelated edges (see Figure 13). Instead, the method excludes SAT edges covered by single-node GPS paths, which preserves short edges at crossroads and endpoints (Figure 14). This strategy targets discontinuating SAT edges while preserving legitimate short segments, unlike alternatives that might ignore important discontinuating edges (Figure 15).

These strategies are compared in Table 4 to explain the methodological choice.

The removal process involves:
1. Identify SAT edges covered by GPS paths with subcurve intervals $[a, b]$ where $a < b$
2. Delete the selected SAT edges from the graph

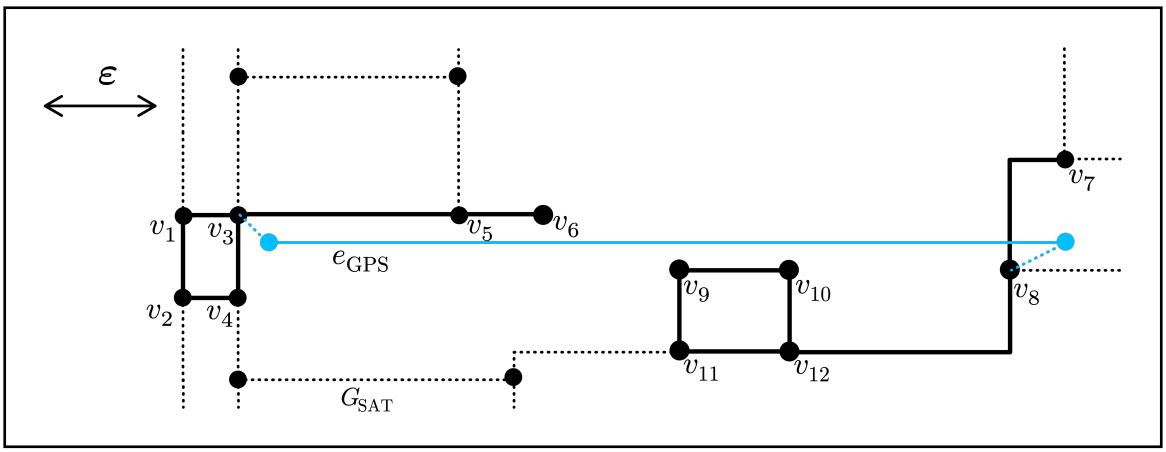


Figure 13: Strategy to delete SAT edges covered by GPS edge geometry.

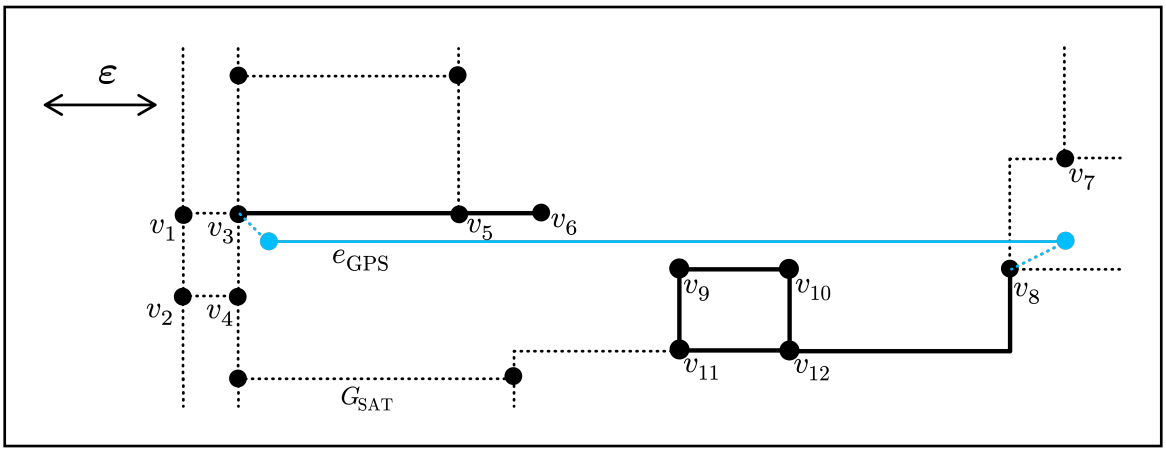


Figure 14: Strategy to delete SAT edges covered by GPS edge geometry excluding SAT edges covered by a GPS edge endpoint.

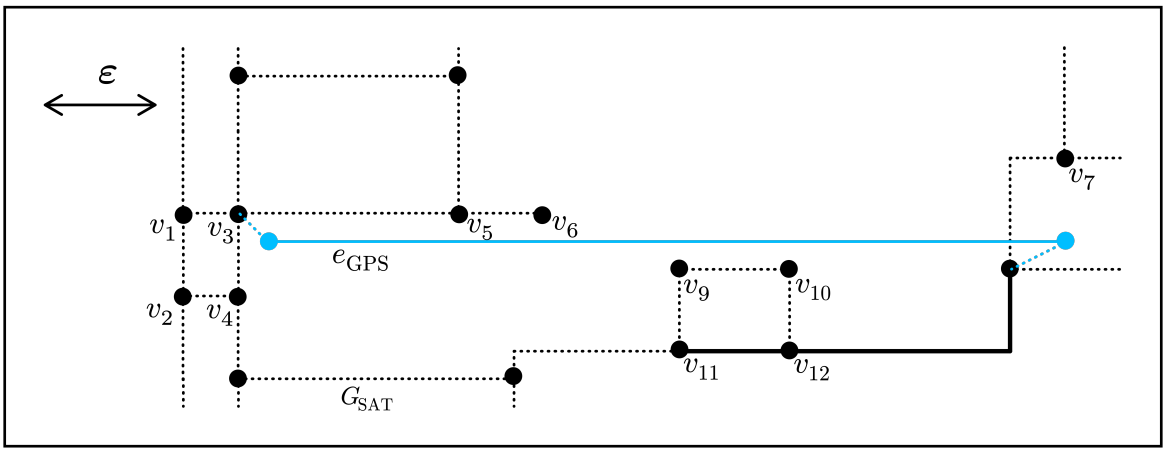


Figure 15: Strategy to delete SAT edges covered by GPS edge geometry excluding SAT edges covered by a single point on GPS edge geometry.

Table 4: Different strategies for selecting SAT edges in the edge deletion step of the map fusion algorithm. The solid black edges are selected for deletion, the blue line is an injected GPS edge, and the dotted blue lines represent related connection edges.

3. Edge Reconnection.

After removing duplicated edges, some SAT nodes may become dangling when their connecting edges were deleted. This step reconnects these isolated SAT endpoints to the injected GPS edges.

The method identifies dangling nodes by finding SAT nodes with reduced degree compared to the original graph. The connect function (Section 4.2.3.5) then links these endpoints to the nearest points on injected GPS subgraphs.

The reconnection process includes:
1. Collect SAT nodes with reduced degree compared to the original graph
2. Find the nearest edge points on injected GPS subgraphs for each node
3. Execute the connection actions

4.2.2. Motivation

Similar to DuARE [40], I hypothesize that unimodal reconstruction methods have complementary strengths based on their data modalities. SAT methods excel at geometric accuracy but may struggle with connectivity, while GPS methods provide reliable connectivity information despite potential geometric limitations.

The late-fusion procedure reflects this hypothesis by using SAT as the base for high-quality geometry, then replacing discontinuating segments with GPS connectivity information. I test these assumptions in the experiments section (Section 6).

4.2.3. Subfunctions

Graph union.

The union combines the vertices and the edges of both graphs. In practice, this function is more elaborate (e.g., handling duplicated node identifiers and duplicated edge identifiers), but for the union at line 6 in the map fusion pseudo-code (Table 3), all node identifiers in G_{diff} and G_{SAT} are guaranteed to be unique. Since edge identifiers rely on the node identifiers of their endpoints, the edge identifiers are also guaranteed to be unique.

Graph exclusion.

The exclusion returns the first graph with all edges removed that also occur in the second graph, and additionally removes any dangling nodes. Similar to the graph union, this operation uses node and edge identifiers.

graph_coverage(G_T, G_S, ε).

The coverage function computes a subgraph consisting of all edges (and their related node endpoints) of the source graph G_S that have their edge geometry covered within a threshold of ε by the target graph G_T. The core component of this function is computing the edge coverage, which is explained in detail at Section 4.1.

degree_reduced(G_1, G_2).

This function collects the node identifiers that exist in both the graph before G_1 and the graph after G_2 and experienced a degree reduction.

connect($G, V,$ exclude $= H,$ difference $= x$).

The connect function connects a set of nodes V to the graph G, with optional parameters for excluding a subgraph H from connection consideration and a distance difference value x. The distance difference x determines how much closer the nearest edge geometry must be compared to the nearest node before choosing to connect to the edge instead. Pseudo-code for the algorithm is shown in Table 5 and a visualization of connecting to nearest edge geometry is shown in Figure 16.

The purpose of this function is to maintain road continuation at multiple stages in the map fusion algorithm. It is called with $V \subset V_G$, and to ensure appropriate behavior, the exclusion parameter always includes the set V itself alongside the edges that v_n (the node under consideration for connection) is already connected to.

The nodes of V should connect only to existing geometry of G, not to any geometry introduced by the connect function itself. In the pseudo-code, I achieve this by first computing how every vertex of V should connect, then performing the connection actions in batch. This batch approach is important because sequential connections would produce different outcomes - at every iteration, newly introduced nodes could become connection candi-

dates for subsequent iterations. For sequential processing, the exclusion set H would need updating to exclude connection edges (such as v_n, e_c in Figure 16) introduced at each iteration.

The behavior of the connect function is visualized in Figure 16 and Figure 17 for the node v_n. In that scenario, the nearest edge geometry is more than x meters closer to v_n compared to the nearest node v_c. A new node v_x is inserted at the closest point on the edge geometry, two new edges e_a and e_b are introduced to connect with v_x, the original edge $e_c \in G$ is removed to prevent duplicating edge geometry, and v_n connects to v_x through a new edge.

Note: Graph simplification is performed as a separate step after the connect function to keep the connection logic straightforward. In the example of Figure 17, the node v_n could potentially be removed during simplification, but this doesn't happen in general since v_n could have had a degree higher than 1.

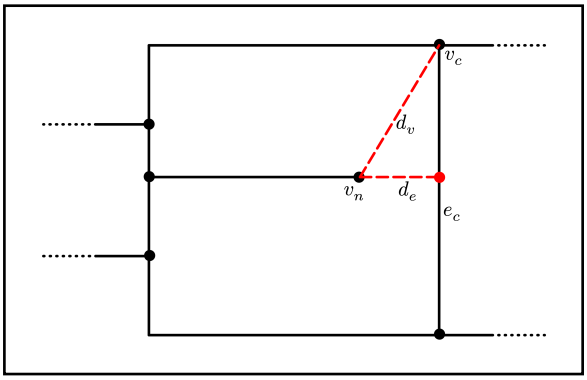


Figure 16: The distance d_e to nearest edge e_c and the distance d_v to the nearest node v_c. Since nearest edge distance d_e is more than x meters closer than d_v, the algorithm chooses to connect to edge e_c.

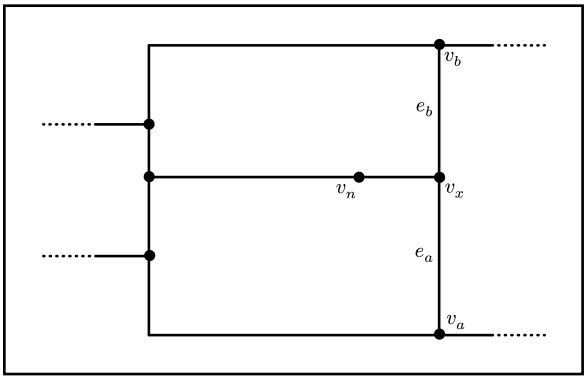


Figure 17: The node v_x is inserted, the edge e_n is cut into two separate edges e_a and e_b, and v_n connects to v_x through a new edge. Note that graph simplification (such as potential removal of v_n) is not part of the connect function logic itself, but is performed as a separate function afterwards.

Algorithm 2: Connect

procedure CONNECT(G_{base},V_{connect}, exclude $=H$,$x = 10$)

▷ Initialize sets for batch processing.
$V_i \leftarrow \emptyset$ ▷ Vertices to batch insert.
$E_i \leftarrow \emptyset$ ▷ Edges to batch insert.
$E_d \leftarrow \emptyset$ ▷ Edges to batch delete.
$E_{\text{adj}} \leftarrow$ **adjacent_edges**(V) ▷ Edges adjacent to the vertices to connect.
$H \leftarrow H \cup (V, E_{\text{adj}}$ ▷ Subgraph to ignore to connect to.

▷ Iterate every node in V_{connect}.
for $v \in V_{\text{connect}}$ **do**
  $v_c \leftarrow$ **nearest_node**$(G_{\text{base}} - H, v)$
  $e_c, p_x \leftarrow$ **nearest_edge**$(G_{\text{base}} - H, v)$
  $d_v, d_e \leftarrow \|v_{c_P} - v_P\|, \|p_x - v_P\|$
  if $d_v < d_e - x$ **then**
    $E_i \leftarrow E_i \cup (v_n, v_c)$ ▷ Schedule edge insertion
  else
    $E_d \leftarrow E_d \cup e_c$ ▷ Schedule edge deletion
    $V_i \leftarrow V_i \cup v_x$ ▷ Schedule vertex insertion
    $E_i \leftarrow E_i \cup \left(e_{c_A}, e_{c_B}\right)$ ▷ Schedule edge insertion
  end
end

▷ Apply changes.
$V_{\text{base}} \leftarrow V_{\text{base}} \cup V_i$
$E_{\text{base}} \leftarrow (E_{\text{base}} \cup E_i) - E_d$

▷ Simplify graph.
$G_{\text{base}} \leftarrow$ **simplify_graph**(G_{base})

return G_{base}
end

Table 5: Pseudo-code for the connect algorithm. The graph simplification logic details are omitted from this algorithm for clarity.

4.3. Evaluation

The similarity metrics TOPO and APLS require modifications to properly evaluate maps with significantly different map densities (see Section 3.2.3), as they show unexpected behavior in relation to their parameters.

I introduce two key modifications to address these limitations. First, in Section 4.3.1, I develop TOPO* and APLS* variants that filter out maximally penalized samples due to missing sample seeds. This modification makes the metrics less sensitive to density differences between maps by focusing evaluation on regions where both maps have road coverage.

Second, I examine specific TOPO parameter choices that impact evaluation quality. I analyze the TOPO sampling interval in Section 4.3.2 to minimize error margin effects, and explore the sensitivity of the TOPO hole size parameter in Section 4.3.3.

4.3.1. Map similarity under different map densities

During early experimentation, I found that standard TOPO and APLS metrics (see Section 2.2) obscure meaningful similarity information when comparing graphs with significantly different map densities. These metrics maximally penalize samples when the source graph lacks nearby edge geometry at sample positions from the target graph. While this penalty appropriately captures differences in map density, it dominates the accumulated similarity scores when maps differ significantly in density, obscuring the impact of road geometry similarity in regions where both graphs do have roads. I developed an approach to make the metrics density-agnostic, providing more insightful comparisons between maps that differ significantly in density.

A **sample seed** is the initial position chosen for evaluation. TOPO uses one seed location per sample, while APLS uses two seed positions to find the shortest path between them.

This problem becomes evident when comparing the Chicago GPS map with the Chicago OSM ground truth map (Table 19). The GPS map contains significantly less road geometry, causing maximal penalties to dominate the TOPO and APLS similarity scores (see Table 12, second row, first column). The resulting scores provide limited insight into how well the GPS map represents roads in areas where it does have coverage.

I therefore introduce TOPO* and APLS* variants that filter out samples contributing maximal penalties. This filtering allows better evaluation of geometry similarity specifically in regions where both maps have road coverage, regardless of overall density differences.

Standard TOPO assigns maximal penalty when the source map lacks edge geometry near the sample position in the target graph. **TOPO*** addresses this by continuing to select new sample positions in the target map until finding one with nearby edge geometry in the source map. This ensures all samples contribute meaningful similarity information rather than just maximal penalties.

Standard APLS assigns maximal penalty when the source map lacks edge geometry near the start or end positions, or when no shortest path exists despite the target graph having one. **APLS*** continues selecting start and end positions in the target map until both have nearby edge geometry in the source map. However, APLS* maintains the full penalty for missing shortest paths, since I expect sparse graphs to reconstruct at least the most relevant roads within regions where they have presence.

4.3.2. TOPO sampling interval to achieve negligible error margin

TOPO can suffer from error margin issues (see Section 3.1.0.1) when the sampling interval is too large, making expectations on actual geometric distances render incorrect. Figure 18 demonstrates how overly large sampling intervals cause TOPO holes and marbles to misalign, producing recall and precision scores that deviate from expectations for the given geometric similarity. Poor sampling alignment can make geometrically similar roads appear dissimilar.

To properly estimate road geometry capture within 5m distance, I set the sampling interval to 5m with a hole size of 5.5m in all experiments. This ensures recall and precision scores reflect actual geometric similarity rather than sampling misalignment artifacts. The slightly larger hole size (5.5m vs 5m interval) provides a small buffer to account for discrete sampling effects.

The required sampling interval relates directly to hole size, so I adapt the interval appropriately to the chosen hole size. Smaller sampling intervals improve recall and precision accuracy by reducing misalignment, though this comes with increased computational costs for similarity analysis.

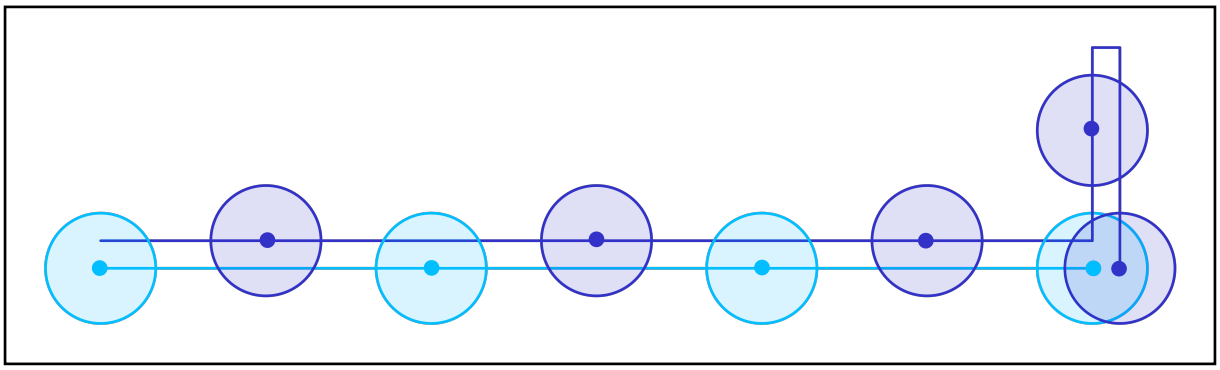


Figure 18: Two graph edges depicted of target graph (aqua) and source graph (navy blue). Positions are sampled with a 5m hole size and 30m distance between samples. Artifact at start of source graph edge causes a desync in marble placement further down the edge geometry. This sample has a recall of 25% and a precision of 20% with TOPO.

4.3.3. TOPO hole size

The hole size parameter relates closely to sampling interval and presents similar challenges where slightly distant curves can cause significant precision and recall drops. However, this represents the expected behavior of the similarity metric and reflects the inherent nature of this sampling technique, which applies strict distance

thresholds. These thresholds can cause large drops in recall or precision when road segments fall just outside the specified distance. I examine the sensitivity to this parameter in experiment Section 6.6 to understand how hole size choices affect evaluation outcomes.

5. Experimental Setup

This section presents the experiment setup that includes data retrieval and the implementation of the steps involved in the map reconstruction method. The full late-fusion map reconstruction pipeline is as follows:

1. Data retrieval: GPS trajectory data, Satellite images, ground truth road maps.
2. Unimodal map reconstruction: GPS-based map reconstruction, SAT-based map reconstruction.
3. Map fusion: Edge coverage, graph manipulation.

Section 5.1 provides information on the implementation of general parts in the code (the R*-tree and coordinate systems). Section 5.2 explains what data is collected for the experiments and how. Section 5.3 goes into detail of the implementation of the modules involved in the map fusion algorithm (includes running the base methods).

5.1. General Components

This section provides details on the R*-tree data structure and multiple spatial reference systems. I use these components in the data preparation and map fusion algorithm.

5.1.1. R*-tree

I use the R*-tree in an optimization of computing edge coverage to filter out a subgraph that is nearby the curve (Section 5.3.2.2) and in finding the nearest node and edge to a position in the connect function (Section 5.3.7.1).

It is used both for range searches and nearest neighbor searches on the nodes and the edges of a graph. For nodes and edges a separate R*-tree is constructed. In case of nodes the bounding box is a point at the node position which does not have a surface (the R*-tree implementation supports this, otherwise a zero-surface bounding box can be approximated with an arbitrary small positive non-zero value). In case of edges the bounding box of an edge is constructed on its geometry.

For the implementation of the R*-tree structure [47] I use a Python wrapper [48] of the libspatial project [49] that provides a C++ implementation of the R*-tree.

Info on R*-tree.

The R*-tree behaves similarly to the the R-tree [50], it constructs a tree on bounding boxes and provides logic to query and delete entries from the tree by recursively checking of children nodes on intersection (of their bounding box) with the input bounding box.

The core difference is the construction of the tree. On the one hand does the R*-tree provide a more elaborate heuristic for deciding what node in the tree to insert an element to (by taking the change in perimeter and area covered into consideration). On the other hand does it try to re-insert 30% of the least “belonging” elements of a leaf node in case it would otherwise have to perform a split at this node.

Neither the R-tree the R*-tree can guarantee any worst-case query performance, because the query properties and the element properties are too unspecific (e.g. we could query the entire domain and the elements can have arbitrary positioning and sizes), but their tests show that for queries on real data the R*-tree does improve significantly in comparison to the R-tree (and a few others).

The insertion logic of R*-tree is quadratic, while R-tree can insert in linear time (yet the authors of R*-tree observe that the exponential insertion algorithm of R-tree performs significantly better and ignore the linear variant in their comparison).

5.1.2. Spatial Reference Systems

Multiple spatial reference systems are used in the data retrieval and map operations. The three systems are the World Geodetic System (WSG84), Universal Transverse Mercator projection (UTM) and the Web Mercator projection (WGS84):

- GPS trajectory data is stored in UTM projection and this projection is used for graph-related computations (because it approximates the Cartesian coordinate system).

- Satellite image requests use the web mercator projection.
- Map data is retrieved with using the World Geodetic System.

I default to use WSG84 as the reference coordinate system and try to maintain data as much as possible in this coordinate system for consistency. Furthermore do I indicate
- latlon for latitude-longitude, implying the WSG84 system.
- pixel coordinates for a latitude-longitude position in the Web Mercator system.

WSG84.

In WSG84 the positions on the earth are described by longitudal λ and latitudal φ rotations. These rotations occur about two axii orthogonal to each other intersecting at the at the center of the earth. To obtain a position, first perform a longitudal rotation, then a latitudal rotation.

The longitudal rotation is made around the axis that goes through the poles, pointing towards the north pole. The latitudal axis initially points towards London and is rotated by the longitudal rotation. Both rotations are made counter-clockwise, a negative rotation value means to rotate clockwise.

UTM and Web Mercator.

Both UTM and the Web Mercator are Mercator projections, but they differ in configuration. A Mercator projection maps the surface of the earth onto the surface of a cilinder. The projection can be made to the outside of the cilinder (secant form) or on the inside of the cilinder (tangent form). The Web Mercator projects on a cilinder to the equator tangentially, while UTM projects on a cilinder placed transverse to the equator.

The Web Mercator projection suffers from the scaling distortion as we move further away from the equator. The UTM projection tries to minimize the scaling distortion by subdividing the world surface by 60 cones each 6 degrees, resulting in 60 zones that together cover the entire earth surface.

The benefit of the Web Mercator projection is to have the entire world localizable on a single plane, which shows its value on obtaining an image of the entire world. While the UTM has the benefit of having a plane with minimal scale distortion, making it effective at maintaining correct distances between points, which is beneficial to compute distances between different points on the world surface.

UTM and latlon conversion.

The zone letter and zone number are necessary for a UTM coordinate to obtain the latlon, while the latlon contains all informatoin to derive the zone letter, zone number, and x,y coordinate from.

I perform conversion between latlon and UTM-WGS84 with the utm python library [51]. This code is self-contained, including to derive the zone number (longitudal sector) and zone letter (latitudal sector) from a latlon.

Web Mercator and latlon conversion.

I require the Web Mercator system for image retrieval with Google Image API, so any peculiarities involved specific to that interface are explained in this section as well.

The Google Image API projects the entire world surface on a single image (256x256 image at zoom level 0). By increasing the zoom level by one, the resolution doubles. Sat2Graph can only infer images effectively at a GSD of 0.5 which is around a zoom level of 16 or 17 (see Section 5.2.2.3). This means a total image size of $8.4 * 10^6$ or $16.8 * 10^6$ pixels in width and height ($256 * 2^{\text{zoom}}$).

With the transformation we can link a pixel coordinate of this image (using the Web Mercator projection) to a latitude-longitude (the WSG projection). At time of coding I had difficulty to obtain a working (and understandable) function of the transformation, so I implemented one myself. The Web Mercator projection manages to map the curved surface of the earth onto a flat surface while preserving relative angle differences, so two lines that are orthogonal/parallel on the sphere appear orthogonal/parellel on the flat surface This is accomplished by using the Gudermannian function. There are multiple ways of computing the Gudermannian function, I use the following:

$$\text{gd}(\tau) = 2 * \arctan(\exp(\tau)) - \frac{\pi}{2}$$

With the inverse

$$\mathrm{gd}^{-1}(\rho) = \log\left(\frac{1}{\cos(\rho)} + \tan(\rho)\right)$$

I notate conversion from degrees to radians as $(\varphi)^\star = \frac{\varphi\pi}{180}$ and from radians to degrees with $(\varphi)^\circ = \frac{\varphi 180}{\pi}$.

The Web Mercator projection has its domain normalized to the range $[0, 1]$ while the related WSG coordinates are within range of latitude $[-85, 85]$ and longitude $[-180, 180]$. The x-axis corresponds with WSG at zero longitude. The y-axis corresponds with the the maximal reachable the maximal reachable WSG coordinate, which is $\mathrm{gd}(\pi)^\circ \approx 85.05$ degrees. Including this remapping to the transformation logic results in:

$$\text{to\_latlon}(y, x) = (\mathrm{gd}(2\pi * (0.5 - y))^\circ, (2\pi * (x - 0.5))^\circ)$$

The latlon to webmercator is the combined inverse of the normalization and the Gudermannian function:

$$\text{to\_webmercator}(\varphi, \lambda) = \left(0.5 - \mathrm{gd}^{-1}\left(\frac{\varphi^\star}{2\pi}\right), 0.5 + \frac{\lambda^\star}{2\pi}\right)$$

In case of pixel coordinates for image retrieval (see Section 5.2.2) we adjust the normalized range of $[0, 1]$ to the number of pixels $[0, 256 * 2^z]$ with $z \in \mathbb{N}$ the zoom level.

5.2. Data

5.2.1. Trajectory data

The GPS dataset used for experimentation is provided by [52]. This dataset provides GPS trajectory data at three cities Athens, Berlin and Chicago.

The GPS locations are stored as UTM coordinates without further information, so I use the latitude-longitude of the city to obtain the zone letter and zone number.

The GPS data of Athens is stored in the Hellenic Geodetic Reference System 1987. I think with [53] it can be translated into to World Geodetic System (WGS84), however I discovered the deviating coordinate system later along the development process and at that moment I made the choice to leave out the Athens dataset from the experiments.

5.2.2. Satellite image retrieval

To obtain related satellite images I built a tool to extract an image of arbitrary size on the area of interest by leveraging the Google Image API.

A visual overview of the steps is shown in Table 7 and the related details are described in this section.

Google Image API.

This paragraph is how the Google API functioned at time of writing. Images are requested by the parameters latitude, longitude, zoom, resolution and scale.

- The latitude and longitude provide the middle point of the resulting image.
- The resolution of the image retrieved. Given a resolution of r the image is $r \times r$. Google API supports a maximal resolution value of 640.

City	Num Trips	Avg Samples/Trip	Avg Distance/ Trip (km)	Total Samples	Total Distance (km)	Avg Sample Interval (s)
Athens	120	603.7	111.9	72,439	13,432.4	60.3
Berlin	27,189	7.1	1.5	192,223	40,603.2	41.7
Chicago	889	133.1	3.2	118,360	2,869.2	3.6

Table 6: Information on the GPS data at the three cities. Note: The datasets provides a small and a large dataset on Athens, I pick the large variant.

- The zoom mentions the total size of the world image to query a subimage from. At the default zoom of 1 the world is a 256x256 image; the image size is described by the formula $256 * 2^z$.
- The scale parameter provides the ability to query an image of increased resolution. The API supports the values 1 (by default) and 2. The scale s updates the resolution to r^{s-1} and the zoom level to $z + (s - 1)$, so by increasing the scale by 1 the zoom level increases by 1 and the resolution is doubled. With a scale of 2 the same pixel coordinates are retrieved, but at double resolution. I apply this feature, because it reduces the number of API calls.

Computing the Ground Density Sampling.

It is necessary to have an image with a GSD similar to that requested by Sat2Graph, otherwise its inferrence quality drops unacceptably low. The Web Mercator projection suffers from scaling distortion so this has to be addressed to be properly informed about the GSD at the place of interest. There is no scale distortion at the equator, but the scale distortion increases as we move further away from the equator. So computing the GSD relies both on the zoom level applied and the latitude of the requested position. The three cities are at different latitudes, so I decided to write code for it to compute the GSD for a place.

The scale factor by latitude s_1 is computed as follows:

$$s_\varphi = \frac{1}{\cos(\varphi)}$$

Additionally do we have the zoom level z. In case there is a a scale factor s per Google API, I apply it to the zoom level beforehand ($z = z + s$) to simplify the formula:

$$\mathrm{GSD}(\varphi, z) = \frac{W}{256 * 2^z * s_\varphi} = \frac{W * \cos(\varphi)}{256 * 2^z}$$

latitude φ in radians, zoom z, scale s, earth circumference $W \approx 2\pi * 6378137$ meter.

Derive the zoom level for a given GSD and latitude.

Sat2Graph inference network is trained on satellite images with a GSD of 0.5 meters. Leveraging the output images of Google API we cannot obtain this GSD exactly. I pick a GSD which closely matches that by approximately ± 0.25 meters, preferring the higher GSD (thus lower quality image), because this significantly reduces the inferrence computation load: doubling the resolution means quadrupling memory space usage. At some moment I considered to sample Google Images at a much lower GSD and then resample this image to match exactly the GSD necessary, but the Sat2Graph inferrence results were good already.

Padding the Region Of Interest.

The region of interest (ROI) is decided by the availability of GPS data at the three cities. Sat2Graph has reduced inferrence quality at its borders, because it cannot accumulate GTE results from multiple (sliding) windows, so I extended the ROI by a few meters (100 pixel coordinates) for the satellite image retrieval so Sat2Graph can capture a roadmap of the entire ROI properly. Furthermore is the ROI further extended in such that the resulting width and height both are a multiple of the stride applied in the GTE inferrence step (88 pixels, see Section 5.3.6.4)

Recovering WSG coordinates of pixels in the image.

The final step in the map reconstruction of Sat2Graph is to convert the inferred graph to WSG coordinates (see). For this it is important to know what the pixel coordinates are of every pixel in the image. I annotate the input image (PNG) with attribute information of the upper-left pixel in the image with its pixel coordinate and the zoom level of the image. With this information (the upper-left pixel coordinate with the zoom level) Sat2Graph has the necessary information to derive the latitude-longitude of every pixel in the input image and thereby the coordinates of the graph nodes it inferred.

Floating point inaccuracy in coordinate transformation.

My implementation of the transformation of a pixel coordinate (in Web Mercator system) to latlon (in WSG system) is sufficiently imprecise to potentially provide a be slightly off transformation result: Transforming from a pixel coordinate to latlon and then back can result in a deviating pixel coordinate (resulting in a pixel coordinate adjacent to the input coordinate). I think this happens due to the conversion math I apply (see Section 5.1.2.4) in

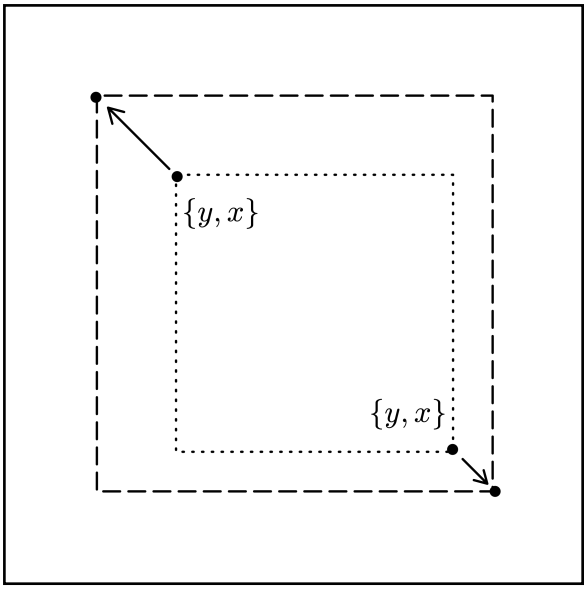


Figure 19: Convert the coordinates of the bounding box on the ROI from WSG to Web Mercator pixel coordinates. Extend region by padding and make it a multiple of stride.

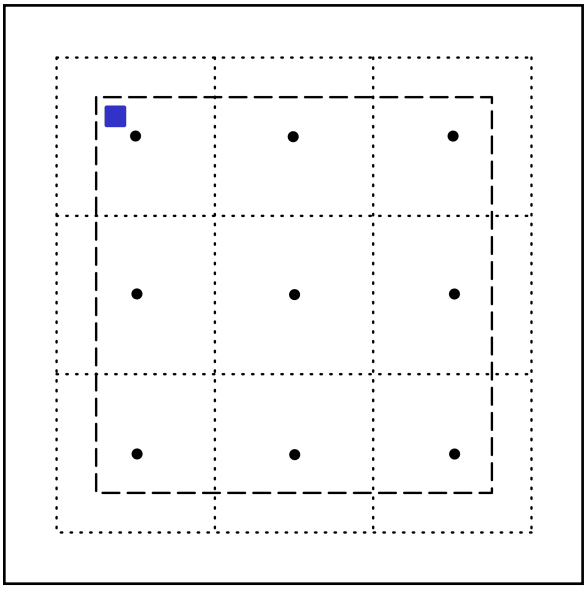


Figure 20: Download tiles to cover the region. Requests take middle position of tile. Afterwards glue image tiles together, cut off excessive pixels, store upper-left pixel coordinate (in blue) alongside the image.

Table 7: Visual overview on the satellite image retrieval.

combination with the bit size of the coordinates (I do not enforce double precision (64 bit) floating points, so it probably uses 32 bit floating points). I solved this issue by updating the resulting latlon (by values of 0.000001) until the conversion back (from latlon to pixel coordinate) results in the input pixel coordinate. Clearly this solution is not robust and only works effectively at this scale, for instance if the pixel coordinates are made at 4 zoom levels higher the step size of 0.000001 would be too large to resolve floating point errors in relation to neighboring pixel coordinates.

Bulk image retrieval.

I use my own tile size of 596x596 pixels (640x640 pixels minus a margin of 22 pixels) to index the world image into an indexed grid, and collect all tiles necessary to cover the ROI. The image takes the provided latlon as the center point of the image, so I provide the middle position of every tile that has to be retrieved. With the scale property the images retrieved ar 1280x1280 and I double the margin (to 44 pixels on both sides), so retrieving images at 1196x1196.

Image statistics.

The surface of Athens,the bounding box on the road trajectory data, is a hundred times larger than Berlin and Chicago. The satellite image data related to this surface is too large to process on my computer. In combination with the issue with the UTM to WSG conversion (see Section 5.2.1), is the reason to exclude Athens from the experiments.

City	Zoom Level	GSD	Surface Area (km^2)	Surface Width (km)	Surface Height (km)	Image Width (pixels)	Image Height (pixels)	Total Pixels (10^6)	Image size (MB
Athens	18	0.468	14981.0	146.7	102.1	314,688	215,688	67874.4	-
Berlin	17	0.726	33.1	5.2	6.3	7,392	8,536	63.1	138.3
Chicago	18	0.444	9.2	3.8	2.4	8,624	5,368	46.3	78.9

Table 8: Information on the region of interest and the satellite image size per city. Note: By increasing the zoom level by 1, the GSD halves and the image width/height doubles (making total pixel count four times as large).

5.2.3. Road map data retrieval

OpenStreetMaps allows to download its map data. I interface with their API using OSMnx (a python code library). The data can be retrieved conveniently by providing a bounding box on WSG (latitude-longitude) coordinates to retrieve all infrastructure informatoin of the requested region. The following properties for retrieval I apply:

- OSMnx provides filters on various type of street networks ("all", "all_public", "bike", "drive", "drive_service", "walk"). After assessing the various types I noticed that "drive" variant provides the road infrastructure of cars that reflects the map reconstruction task.
- I limit to publically accessible roads (default choice).
- I consolidate intersections on the default value of 10 meters. Nearby nodes/curvature used to describe adjacent roads are consolidated to a single road, the algorithm is described more in-depth in the paper related to OSMnx [54].
- I drop the directionality on the edges by converting the graph into an undirected graph.

5.3. Implementation

In this section I go into detail on the implementation of the various functions involved on getting the unimodal map reconstruction methods running on the input data and applying the map fusion algorithm:

- Curve similarity Section 5.3.1
- Curve/Edge coverage Section 5.3.2
- Map similarity Section 5.3.3
- GPS-based method Roadster Section 5.3.5
- Satellite image retrieval Section 5.2.2
- SAT-based method Sat2Graph Section 5.3.6
- Map fusion procedure Section 5.3.7

5.3.1. Curve similarity (Fréchet distance)

I have implemented a portion of the Fréchet distance algorithm by Alt & Godau to check for two polygonal curves whether their (continuous) Fréchet distance is below a given threshold ε. Note it does not compute the Fréchet distance, it only checks the Fréchet distance is below a given distance threshold. The part of the paper [3] that optimizes finding the minimal ε (for which a monotonically increasing path in the Free Space Diagram exists) is left out. I have implemented it from scratch as an exercise and I could not find a library that matched my needs. The result is a Rust library with bindings to Python.

Continuous variant.

The implementation is based on the algorithm [3]: it constructs a Free Space Diagram (FSD) between the straight line segments of both curves for the provided threshold distance ε and then looks for a monotonic increasing curve from the lower-left corner to the upper-right corner of the FSD (to traverse from start vertex to end vertex of both curves). In case directionality is irrelevant, the algorithm simply recomputes from scratch with one of the curves reversed. This can be optimized, but the algorithm runs fast enough not to care about that. The remaining of this section provides more information on the algorithm of [3].

Free-Space Diagram.

The **Free-Space Diagram** (FSD) is a two-dimensional field constructed for two curves and a distance threshold. The horizontal axis aligns with traversal along one curve, the vertical axis with traversal along the other curve. Any position in this two-dimensional domain is related to a point on both curves and is classified as either **valid** (within the distance threshold) or **invalid** (above the distance threshold). In our case we deal with polygonal curves only which allows to subdivide the FSD domain by the vertices of the curves, resulting in consecutive subregions called **FSD cells** (bound by two vertices of both polygonal curves).

Use of the Free-Space Diagram for computing the Fréchet distance.

The solution approach is to compute the FSD and then to seek a monotonic path in the FSD that traverses from the beginning to the end of the domain that passes through valid points only. This is done by computing the **Reachable Space Diagram** (RSD), the valid space reachable by a monotonically increasing path. After computing the RSD it is only necessary to check the upper-right point (the end point of both curves) is valid. To

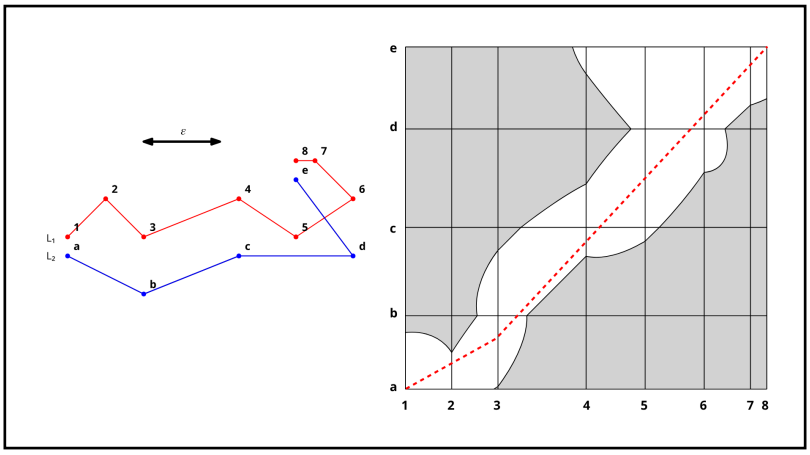


Table 9: Free-Space Diagram layout for the lines L_1 (with points $1, ..., 8$) and L_2 (with points $a, ..., e$). The free space region in the FSD is rendered with a white background. The free space can be traversed with a monotonic path, an example is drawn with the red dashed line.

compute RSD start at the bottom left (the starting point of both curves) and iteratively discover the reachability of adjacent FSD cells. An important observation is that the valid region within a FSD cell is always convex, this simplifies the problem to only having to compute FSD cell boundaries and seek a path along the boundaries, because two arbitrary valid points on a boundary can be reached by a straight valid line.

Discrete variant.

Beforehand I implemented the threshold check on the discrete variant of the Fréchet distance. However, I noticed that there would be an error margin (see Section 2.1.3 for the definition) and this caused the logic of the threshold check to become sufficiently complicated (injecting vertices and smarter comparison along vertices), that instead I decided to switch to implementing the continuous variant.

5.3.2. Curve and edge coverage

Curve coverage checks for the existence of a graph path that matches with the curve. The curve matches with a path if the Fréchet distance to the path geometry is below the threshold value. I implement curve coverage by implementing the map matching algorithm of Alt et al [55] (sections 2.2 t/m 2.4) without the parametric search (section 2.6). I leave out the parametric search, because the parametric search is used to compute the minimal threshold value, while curve coverage only has to check map matching occurs at a specific threshold value.

Given a polygonal curve consisting of p line segment and a *vectorized* graph consisting of q edges, in section 2.5 they describe how the algorithm has a time complexity of $O(pq \log(pq))$ and a space complexity of $O(pq)$. My implementation achieves the same time and space complexity by following the description of their algorithm.

I further optimized the curve coverage function by filtering out a subgraph that has edges with geometry nearby the curve, described in Section 5.3.2.2. With the implementation of Alt et al in combination with the subgraph prefiltering the function sufficed for fast computation in my use-case, I therefore did not find it necessary to implement the algorithmic optimization by Maheshwari et al [56].

Below I go into more detail on how the map matching algorithm of Alt et al [55] (without the parametric search of section 2.6) works.

Free Surface Diagram.

Construct a Free-Space Diagram for the curve against every vectorized graph edge. This results in a collection of Free-Space Diagrams and every diagram being a single column of FSD cells. The goal is to glue Free-Space Diagrams together so we obtain a Free-Space Surface that can be traversed completely through the valid domain.

Unlike computing the Fréchet distance against a curve, the solution of iteratively constructing a Reachable Space Diagram does not work against a graph, because a point in the Free-Space Surface can be reached from multiple directions. To find a solution instead this Free-Space Surface is explored with a depth-first search. To find a route efficiently the algorithms applies a partial maxima stack to figure out the relevant cell boundary range intervals

and storing boundary positions as shortcut pointers to know what points in the FSS to revisit for alternative possibilities if the current path does not find a solution.

Optimization by prefiltering the subgraph on the region of interest.

The algorithm by Alt et al [55] precomputes all Free Space Diagram columns, namely by comparing the curve against every vectorized edge of the graph. In my usage this precomputation step is the most computational expensive part in the process, because computing an FSD column is (relatively) expensive, and there are many FSD columns to precompute because the graphs are quite large. However, in practice often just a small subset of graph edges are sufficiently nearby to the curve to be involved in computing curve coverage. I use an R*-tree that can perform cheap bounding box checks (in comparison to computing FSD columns) to filter out a subgraph with these nearby edges which saves precomputing a lot of FSD columns.

I populate an R*-tree with the graph edge bounding boxes (see Section 5.1.1) and the range query is performed with the bounding box of the curve padded by the threshold distance. The subgraph is extracted from the resulting set of edges of the range query. This subgraph has all edges from the graph removed that are not included in the set, and as well removes all dangling nodes as a result of the edge removal. This subgraph is then used to compute the curve coverage with.

The reason I can apply this optimization is because I do not need to compute the Fréchet distance, but compute curve coverage which only checks the Fréchet distance is below a certain threshold. This means that any edge bounding box not captured by the range query cannot be involved in a path that covers the curve. This is also why the optimization guarantees correctness: If an edge is involved in the minimal path that covers a curve for a threshold distance ε, then the edge is guaranteed to be included in the range query result (of the curve bounding box padded by ε), and thus in the subgraph used in the map matching logic.

5.3.3. Map Similarity Metrics TOPO* and APLS*

I leverage the TOPO implementation of the Sat2Graph soure code [57] and extend it to support TOPO*. The implementation performs a sample with random selection from all vectorized target graph nodes. The only change needed for TOPO* is to filter out vertices that have a nearby vertex in the source graph.

Sat2Graph source code additionally provides code for APLS [58], but I found it difficult to grasp it and tweak it (especially the absence of the sample categorization which grealy simplified differentiating APLS from APLS*) so I decided to implement APLS from scratch myself and implement APLS* on top of that. The core of the APLS algorithm consists of the following steps:

1. First precompute **control points**: a related source graph node (or vertex) for every node of the target graph. If no nearby source graph node exists, instead seek nearby edge curvature in the source graph and inject a vertex at that position and relate to that.
2. Then pick from the target graph two nodes at random until a pair is found for which a path between exists (in the target graph). Then categorize that sample to one of the three categories (missing control point at start or end node, missing path between control nodes, or existing path between control nodes).
3. Accumulate the requested number of samples and compute the average.

The changes to APLS* are to leave out the first category of samples (those of which either a start or end node is missing). The APLS* does attempt to accumulate the requested amount of samples rather than just dropping the samples of the first category with the potential to have a significantly reduced resulting sample count.

5.3.4. Change in APLS sampling strategy

I applied a change to APLS for practical reasons. Both the Berlin and Chicago map ROI is quite large with a lot of edges. Injecting nodes to Berlin results in +-3000 nodes, and computing the shortest paths takes a ton of data (on average 500MB with path length stored as 64-bit floating point numbers). Precomputing was no longer a solution, because loading in the data to RAM already took long.

The authors of APLS solved this bottleneck by making a preselection of nodes of the graph and then computing all paths between these nodes. I instead solved it by keeping the original graph (so not injecting nodes to ensure maximal 50m length simplified edges) and adding some logic to be able sampling at arbitrary position on the target graph. This (a) results in being able to sample the entire graph and (b) shortest path computation is faster alongside smaller data storage. The downside is that constructing samples takes significantly more time: e.g.

from 500 predefined points one can easily generated 100.000+ samples near instantly while my technique takes a few minutes to achieve that.

5.3.5. GPS-based method: Roadster

Use of the code repository [59] developed by authors of the Roadster paper [22] including the optimizations [1].

The code is run on the three cities contained in the dataset Section 5.2.1 in UTM coordinates, because the algorithm relies on computations by distance (if it was converted to WSG then distance is inhomogeneous). After inferrence the reconstructed map vertex coordinates are transformed to WSG.

5.3.6. SAT-based method: Sat2Graph

I choose Sat2Graph as the satellite-based unimodal method for reconstruction of the SAT graph, because I consider this method simple to understand and practical to use. Simple because it uses a single CNN in its inferrence pipeline and the intermediate data representation is straightforward. Practical because I consider it doable to debug or change things since the neural network is a single step in the inferrence pipeline. Choosing Sat2Graph turned out to be a good choice to tweak it for inferring larger images (of arbitrary size) without data bottleneck (RAM overflooding), described at Section 5.3.6.2.

Sat2Graph is surpassed by TD-road [60], RNGDet++ [34] as well by SAMRoad [30], but Sat2Graph is still considered a competitive method, and these competing methods rely on multiple neural networks in their pipeline which I think makes it more difficult to rebuild.

How Sat2Graph works.

Sat2Graph takes an satellite image as input and reconstructs the road map visible in the image. The RGB image is converted into Graph Tensor Encodings (GTE), a special multi-dimensional data structure that captures the probability that a vertex and an edge (in various directions) exist at a given pixel, and derives the graph from this data representation.

The neural network to infer a GTE from an image can take on a 352x352 image, so to convert the entire image this CNN uses a sliding window to infer GTEs on the entire image. By applying a step size in the sliding window of 88 pixels (one fourth of the GTE inferrence window width/height) every pixel is evaluated multiple times resulting in a more stable/robust solution. Afterwards all parital GTE are accumulated and the pixels are normalized by the total weight applied to them to obtain a single GTE.

To derive the (vectorized) nodes of the graph, local maxima in vertex probability in this GTE are searched and their centerpoint are chosen to be marked as a graph node. These pixels (appointed as graph nodes) their edge probabilities are read (along their respective edge directionality) to figure out if and in what direction edges are moving from the node. An extrapolation trick is used by checking whether a nearby vertex exists in the edge direction to better predict the edges present at the node.

The resulting graph has no coordinates (but uses relative “pixel coordinates” in relation to the upper-right corner of the input image). I annotate the input image with metadata containing the zoom level and the Web Mercator pixel coordinate of the upper-left image pixel (explained at Section 5.2.2) to obtain the WSG coordinates of all nodes in the graph.

Processing arbitrary image sizes.

The original implementation had hardcoded the image input size it supports. I want the method to accept arbitrary image width and height, especially since the satellite image of the regions of interest Berlin and Chicago are significantly larger.

The changes necessary to make it dynamic was quite minimal, but I faced a problem with the RAM of my PC getting overflooded by having to keep all partial inferred GTE tiles alongside the CNN parameter weights in memory. I solved this by processing partial GTEs one by one and writing them to disk, resulting in constant RAM usage irrespective of the number of tiles to process.. After all partial GTEs are computed the CNN is taken out of memory, all partial GTEs are read from memory, and the complete GTE is constructed. To fix the RAM problem I purposefully decided *not* to infer subgraphs and fuse them afterwards, because this would change the logic (and behavior) of Sat2Graph, and it was not necessary because all partial GTEs could fit in memory.

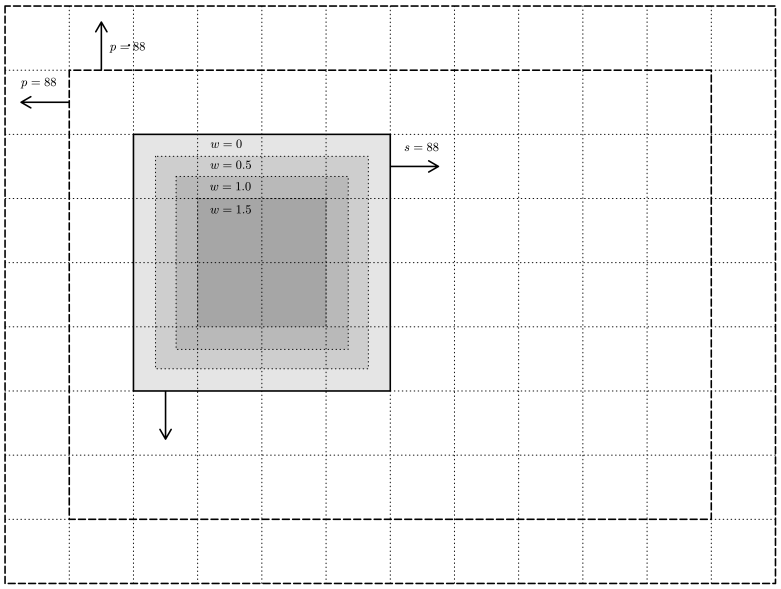


Figure 21: Depicting an input image of arbitrary width and height (the inner rectangle with dashed line). The input image is enfored to a multiple of stride s of 88 pixels, and note that $s = 88$ is a quarter of the partial GTE width/height. Partial GTE tiles of 352x352 pixels (the square with the solid line) are inferred as sliding windows with steps equal to the stride s (the thin dotted lines in the image). The outer rectangle with dashed line is the image padded by $p = 88$ pixels to obtain better inferrence results at the border of the image.

Note: It is not necessary to store the masks applied to every partial GTE, because these are all the same. When normalizing we can use the same weight mask for every partial GTE.

The post-processing of the GTE already acted on arbitrary GTE sizes, so this logic can remain unchanged.

Getting started with using Sat2Graph.

To get the code up and running was a challenge.

I failed to resolve dependencies on the original codebase, because the code was written for python version 2. The authors do provide a docker image that did run, but it has its logic interwoven with a web server and functionality to support arbitrary input images was missing (see Section 5.3.6.2).

Since the code in the docker ran, I had a reference point of working code. Together with code in the github repo I managed to make the necessary changes to get it running on python version 3.

They provide the pre-trained CNN models they mention in their readme in the docker image. Unfortunately, most of these models could not be load by tensorflow, but luckily the most prominent model globalv2 did load. In comparison to global v1 that got trained at half resolution (1m^2 per pixel), this model (with 0.5m^2 per pixel) as well predicts road type (car or sidewalk).

I ran the inferrence on the CPU which works just fine, it takes around ten minutes to infer Berlin and another ten minutes to infer Chicago.

Some info on data sizes:

- Every partial GTE has a size of 15MB.
- Berlin has to infer 1794 partials GTEs and is in total ≈ 26.67 GB.
- Chicago has 1288 partial GTEs and is in total ≈ 18.43 GB.

Inferring partial GTE with global V2 model.

The Sat2Graph models infers a GTE at a size of 352x352, and the GlobalV2 model takes as input the images at a double resolution (of 704x704) to obtain higher accuracy. The other models are trained on a GSD of 1m, this model is trained on a GSD on 0.5m.

I changes the step size to be the same as the stride, setting them both at 88 pixels (a quarter of the inferrence window size) which becomes 176 pixels with the globalV2 model because of the scaling.

A partial GTE inferrence is more accurate at the center part of its visual perception and to reflect that in the inferrence the weight w of the GTE window gradually decreases at the outer s border (see Figure 21).

- $w = 0$ at pixels $\left[0, \frac{s}{3}\right]$ and $\left[352 - \frac{s}{3}, 352\right]$
- $w = 0.5$ at pixels $\left[\frac{s}{3}, 2\frac{s}{3}\right]$ and $\left[352 - \frac{2}{3}, 352 - \frac{s}{3}\right]$
- $w = 1$ at pixels $\left[2\frac{s}{3}, s\right]$ and $\left[352 - s, 352 - 2\frac{s}{3}\right]$
- $w = 1.5$ at pixels $[s, 352 - s]$

Note these values are doubled in case of globalV2 because the context window width and height is twice as large.

GlobalV2 classifies road segments into three categories: freeway roads, traffic roads and service roads (e.g., parking roads and foot paths) [61]. I decided to ignore it and consider all these categories to be part of the road network.

Leaving out misclassified edges.

The graph tensor encoding uses 31 channels:

- one for the vertexness (first channel)
- 24 for edges (6 directions with edgeness, direction)

Difficult not documented how globalv2 is working. 4 channels per edge direction instead of 3 as mentioned in the paper:

- First is edgeness
- Second is probably the edge probability complement, used for binary classification loss during training.
- Third is x and fourth is y to construct edge direction

Another two channels are used to describe segmentation (is road or is not road. Last three channels are used for classifying road type ("freeway roads, traffic roads and service roads (e.g., parking roads and foot paths)"))

5.3.7. Map Fusion

Pseudo-code on the algorithm is mentioned in the method Table 3, here I go into more detail how I implemented the functions involved. Both graphs G_{GPS} and G_{SAT} are first simplified and node positions are transformed into UTM coordinates.

Connect.

I use an R*-tree for finding the nearest node and (another R*-tree) for finding the nearest edge. Both R*-trees are populated with bounding boxes capturing the node position and edge curvature respectively; the nodes are captured by bounding boxes with no surface (the R*-tree implementation supports this, otherwise a zero-surface bounding box can be constructed with an arbitrary small positive non-zero value). Since the connect function is called often on the same graph data, the R*-tree construction only has to be done once. The insertion in the implementation is done iteratively, which means the R*-trees are updated by inserted nodes and inserted and deleted edges, as well to exclude the connection edge after every iteration.

The nearest entry in the node R*-tree is searched for and this is v_n. The nearest entry in the edge R*-tree is searched for and this provides a nearby edge e'_n. It is however the bounding box of e'_n, so the actual edge curvature can be further away from the connection node in comparison to another edge. So I compute the distance to the nearest edge curvature from that edge and use this as a query distance to collect a set of nearby edges that potentially have curvature more close to v_n. Then the nearest edge curvature among these edges is searched for and this results in e_n.

Graph vectorization.

Some functionality requires a vectorized graph, such as

- The transformation of coordinate system by changing the position of every node position, including the vertices along the edge curvature. (It can be done by updating the graph edge properties, but at the time of coding I kept it simple by only allowing such conversion to be performed on a vectorized graph.)
- Computing edge coverage against a graph, that requires the underlying edge curvature which is made explicit by vectorization

Vectorization works by replacing all simplified edges which have curvature of three or more points, otherwise the edge is a straight line segment and already in the appropriate format. A key component in the vectorization is to

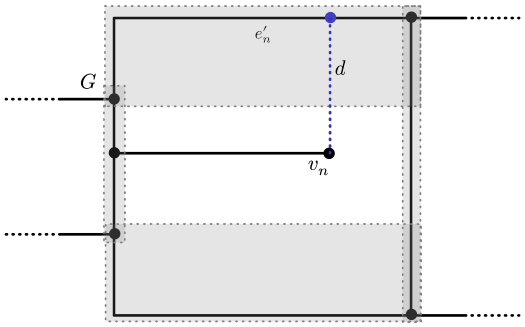


Figure 22: Perform an R*-tree nearest neighbor search at v_n to find a nearest edge e'_n by its bounding box and compute the distance d to this edge e'_n.

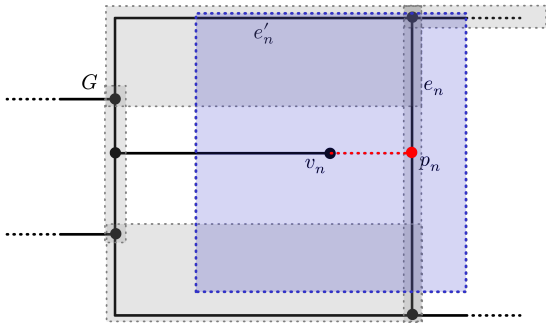


Figure 23: Perform an R*-tree range query of position v_n with a bounding box with width and height of $2d$, to find the nearest edge e_n (at point p_n).

Table 10: Finding the nearest edge e_n of a graph G to a point v_n. The edge bounding boxes are the transparent gray rectangles, padded for visualization.

store at every vectorized edge segment the originating simplified edge. This allows to reconstruct the simplified graph path related to a result found in edge coverage.

Graph coverage.

The target graph is vectorized and is preprocessed into a format necessary for the edge coverage function. The source graph is simplified and then iteratively for every simplified edge the edge curvature is checked against the target graph for coverage. The OSMnx library provides functionality to filter out the subgraph by the edges that have coverage. Implementation information on computing edge coverage is provided at Section 5.3.2.

6. Experiments and Results

In this section I research whether the map fusion algorithm is effective using the experimental setup described in the previous chapter Section 5.

I start by outlining hypotheses and related research questions in Section 6.1. Based on these hypotheses, I evaluate a first experiment at Section 6.2 to check whether map reconstruction performance aligns with my expectations.

The baseline results revealed patterns that differed from my expectations, prompting a more thorough investigation into the underlying factors driving these performance outcomes. This deeper analysis examines two main questions that emerged: how does the map fusion threshold parameter actually affect map fusion performance Section 6.3, and what role do the properties of individual unimodal maps play in overall performance Section 6.4.

To ensure confidence in my conclusions, I also examine the evaluation metrics themselves by analyzing TOPO and APLS sample distributions Section 6.5 and testing how TOPO hole size parameters influence the performance measurements Section 6.6.

Throughout this research, I used images and plots of the maps to understand certain quantitative behaviors. I include the qualitative analysis last in Section 6.7, it includes images showing all the maps involved, as well as detailed zoomed views of relevant map fusion behavior and actions. It is worthwhile to look first at the qualitative analysis to make sense of some quantitative results. it definitely helps motivate, reason through, and provide evidence for certain quantitative behaviors.

Note: The map fusion threshold implies the edge coverage threshold.

6.1. Hypotheses and Research Questions

The goal of the map fusion algorithm is to improve map reconstruction by increasing similarity to the ground truth, as measured primarily by the TOPO and APLS metrics. These metrics serve as the primary evaluation criteria, while the TOPO* and APLS* metrics are employed as analytical tools for graph examination. It's important to note that a map can improve in one metric while declining in another; for example, an increase in TOPO may coincide with a decrease in APLS.

This research is guided by the general question of whether map fusion can improve uni-modal map reconstructions, to what extent, and through which mechanisms. Since this question is overly broad, I focus specifically on resolving conflicts in road continuation Section 3.2.2. I expect to measure road geometry agreement with the similarity measures, and expecting to measure road continuation with edge coverage. This leads to the following research questions:

Main Research Question: *Does map fusion improve map similarity by resolving conflicts in road continuation between two uni-modal map reconstructions?*

The design of the map fusion algorithm is motivated by two key assumptions:
1. GPS graphs are more accurate at capturing road continuation.
2. SAT graphs are more accurate at capturing road geometry.

Based on these assumptions, I formulate the main hypothesis:

Main Hypothesis: *Applying the map fusion algorithm to a base map with relatively better road geometry but relatively poorer road continuation will increase map similarity scores.*

Implied Hypothesis (derived from the above): *Map fusing GPS onto SAT will produce a map that outperforms both uni-modal methods.*

To verify these assumptions and hypotheses, I address the following research questions:
- Are GPS graphs more accurate at capturing road continuation?
- Are SAT graphs more accurate at capturing road geometry?

- Does map fusion improve a map if the base map has relatively better road geometry and worse road continuation?
- Does applying map fusion of GPS onto SAT improve map reconstruction?
- Does map fusion guarantee that the resulting graph performs at least as well as the better of the two base maps?
- How can we measure (dis)agreement of road continuation between two maps?
- How can we measure (dis)agreement of road geometry between two maps?

Although the last research question may appear trivial, experimental results indicate it is not.

6.2. Map Reconstruction Performance

6.2.1. Motivation of experiment

The first experiment evaluates my hypotheses by comparing the similarity performance of unimodal and fusion maps. Map fusion uses SAT as the default base map, replacing discontinuous SAT edges with continuous GPS edges. I also examine an alternative variant using GPS as the base map to determine whether base map selection affects the results.

6.2.2. Description of experiment

Tables Table 11 and Table 12 present the performance of the unimodal map reconstruction methods and their fusion variants on the Berlin and Chicago datasets respectively.

The maps are abbreviated as follows:
- Graph $\mathbf{OSM}$: The ground truth map retrieved from OpenStreetMaps.
- Graph $\mathbf{SAT}$: The unimodal map reconstructed with Sat2Graph on satellite images.
- Graph $\mathbf{GPS}$: The unimodal map reconstructed with Roadster on GPS traces.

The fusion variants with SAT as base map include:
- Graph $\mathbf{I}_{\mathrm{SAT}}$: Base map $\mathbf{SAT}$ with the first fusion step applied, injecting simplified GPS edges not covered by the SAT graph.
- Graph $\mathbf{ID}_{\mathrm{SAT}}$: Extends $\mathbf{I}_{\mathrm{SAT}}$ by deleting simplified SAT edges covered by injected GPS edges.
- Graph $\mathbf{IDR}_{\mathrm{SAT}}$: Extends $\mathbf{ID}_{\mathrm{SAT}}$ by reconnecting SAT edges to injected GPS edges that lost connections during deletion.

Conversely, graphs $\mathbf{I}_{\mathrm{GPS}}$, $\mathbf{ID}_{\mathrm{GPS}}$, and $\mathbf{IDR}_{\mathrm{GPS}}$ follow the same progression but use GPS as the base map and SAT as the patch map.

TOPO and TOPO* each provide three columns in Table 11 and Table 12 showing recall (**Rec**), precision (**Prec**) and the F_1 (**TOPO**) score. The definitions of TOPO and APLS are described in Section 2.2, while TOPO* and APLS* are defined in Section 4.3. I use the asymmetric variants of TOPO and APLS, taking sample seeds only from the ground truth so that reverse sampling does not impact or obfuscate the similarity score.

Table 13 lists the TOPO and APLS sampling properties used in the experiment. I use 10000 samples for TOPO and APLS to ensure the accumulation sufficiently approximates the expected value. The prime variants use 2000 samples since they have more concentrated distributions (see Figure 32).

I use default APLS properties but slightly adjust the TOPO parameters by increasing the hole size to 5.5m and reducing the interval placement to 5m. This adjustment minimizes the error margin (Section 3.1.0.1) so that recall and precision properly reflect the amount of geometry within 5m range of each other.

6.2.3. Expected experiment outcomes

To support my hypotheses, I expect the following outcomes:
1. Hypothesis 1: $\mathbf{IDR}_{\mathrm{SAT}}$ will achieve the highest score, with $\mathbf{IDR}_{\mathrm{GPS}}$ also ranking among the top performers.
2. Hypothesis 2: The SAT-based variants ($\mathbf{I}_{\mathrm{SAT}}$, $\mathbf{ID}_{\mathrm{SAT}}$, $\mathbf{IDR}_{\mathrm{SAT}}$) will outperform their GPS-based counterparts ($\mathbf{I}_{\mathrm{GPS}}$, $\mathbf{ID}_{\mathrm{GPS}}$, $\mathbf{IDR}_{\mathrm{GPS}}$) respectively.

Both $\mathbf{IDR}$ variants should benefit from improved edge continuation without processing artifacts from earlier steps, such as duplicated edges from injection and dangling endpoints from deletion. These final fusion maps

should combine the complementary strengths of both approaches: the superior road continuation of **GPS** and the superior road geometry of **SAT**.

This expectation stems from the complementary nature of the data sources. It is expected that SAT generally provides better road geometry while GPS offers better road continuation, so applying GPS patches onto a SAT base should prove more effective than the reverse approach.

		Rec	Prec	TOPO	APLS	Rec*	Prec*	TOPO*	APLS*
Berlin	**SAT**	0.307	0.593	0.405	0.544	0.679	0.591	0.632	0.888
	GPS	0.494	0.849	0.625	0.703	0.816	0.845	0.830	0.952
	$\mathbf{I}_{\mathrm{SAT}}$	0.395	0.582	0.471	0.651	0.729	0.581	0.647	0.932
	$\mathbf{ID}_{\mathrm{SAT}}$	0.380	0.616	0.470	0.620	0.705	0.614	0.657	0.896
	$\mathbf{IDR}_{\mathrm{SAT}}$	0.380	0.606	0.467	0.627	0.716	0.604	0.655	0.908
	$\mathbf{I}_{\mathrm{GPS}}$	0.397	0.582	0.472	0.796	0.739	0.580	0.650	0.956
	$\mathbf{ID}_{\mathrm{GPS}}$	0.386	0.606	0.472	0.759	0.707	0.608	0.654	0.946
	$\mathbf{IDR}_{\mathrm{GPS}}$	0.390	0.609	0.475	0.769	0.719	0.608	0.659	0.949

Table 11: Map similarity scores of Berlin maps in comparison to the ground truth. The fusion maps are constructed with a threshold of 30m.

		Rec	Prec	TOPO	APLS	Rec*	Prec*	TOPO*	APLS*
Chicago	**SAT**	0.569	0.289	0.383	0.718	0.846	0.294	0.437	0.868
	GPS	0.002	0.606	0.005	0.010	0.110	0.609	0.187	0.630
	$\mathbf{I}_{\mathrm{SAT}}$	0.574	0.287	0.383	0.721	0.858	0.286	0.429	0.869
	$\mathbf{ID}_{\mathrm{SAT}}$	0.501	0.281	0.360	0.596	0.751	0.283	0.411	0.776
	$\mathbf{IDR}_{\mathrm{SAT}}$	0.543	0.278	0.368	0.692	0.810	0.282	0.418	0.885
	$\mathbf{I}_{\mathrm{GPS}}$	0.582	0.290	0.387	0.646	0.859	0.290	0.434	0.895
	$\mathbf{ID}_{\mathrm{GPS}}$	0.510	0.280	0.362	0.623	0.762	0.278	0.407	0.875
	$\mathbf{IDR}_{\mathrm{GPS}}$	0.535	0.279	0.366	0.624	0.810	0.276	0.412	0.874

Table 12: Map similarity scores of Chicago maps in comparison to the ground truth. The fusion maps are constructed with a threshold of 30m.

Metric	Sample Seed Distance (m)	Sample Radius (m)	Sample Interval (m)	Number of Samples	Number of Prime Samples
TOPO	5.5	150	5	10000	2000
APLS	5	-	-	10000	2000

Table 13: The TOPO and APLS sampling properties used in the first experiment.

6.2.4. Describing experiment outcomes

The experimental results contradict all predicted outcomes.

Neither Berlin nor Chicago shows $\mathbf{IDR}_{\text{SAT}}$ as the best performer. Instead, $\mathbf{I}_{\text{SAT}}$ and $\mathbf{I}_{\text{GPS}}$ emerge as the top fusion variants across both datasets and all metrics. Base map selection shows different patterns for each city: Chicago performs better with SAT as the base, while Berlin shows worse performance for SAT-based fusion compared to GPS-based variants.

Map fusion fails to match or exceed the best unimodal performance in either city. However, Berlin $\mathbf{I}_{\text{GPS}}$ and Chicago $\mathbf{I}_{\text{SAT}}$ do outperform the unimodal maps on APLS and APLS* metrics. The reconnection step generally improves reconstruction quality, with Berlin $\mathbf{IDR}_{\text{SAT}}$ as a notable exception. The deletion step creates stronger negative effects that reconnection cannot fully compensate for, causing most **IDR** variants to underperform **I** variants except for Berlin $\mathbf{IDR}_{\text{GPS}}$.

A clear pattern emerges regarding patch map quality: applying all fusion steps tends to improve reconstruction when the patch map performs relatively better than the base map. Conversely, fusion reduces overall quality when the patch map performs worse than the base. For Berlin, base map choice minimally affects TOPO scores, though GPS as a base yields higher APLS scores despite GPS's extremely low performance on both metrics. The fusion actions visualization in Table 20 shows that almost the entire SAT map was incorporated into $\mathbf{IDR}_{\text{GPS}}$, explaining their similar performance levels.

The relationship between TOPO and TOPO* scores appears primarily through their recall components, with Rec* consistently equal to or higher than Rec. Failed sample seeds in TOPO often contain numerous empty holes without spurious marbles, heavily penalizing recall while leaving precision unaffected. This explains why precision values remain similar between TOPO and TOPO* while TOPO* recall is significantly higher.

Chicago GPS follows this recall pattern, though its Rec* remains low at 0.110. This suggests many empty holes occur even in samples containing marbles at seed locations, possibly due to deviating road geometry. However, examination of the Chicago GPS map in Table 19 reveals that much road infrastructure is missing. The reconstruction covers only main roads and excludes side alleys, explaining the low TOPO* recall.

APLS* scores show general similarity within each city, with Chicago GPS as the notable exception. This similarity likely occurs because shortest paths are computed on undirected and unweighted graphs. Even when major infrastructure is missing, alternative alleys can still provide connections with minimal penalty. Chicago GPS stands out because its sparse reconstruction lacks such alternative routes, as demonstrated in Table 19. It's worth noting that APLS scores alone do not reveal whether reconstructed path lengths are shorter or longer than ground truth paths.

6.3. Threshold Parameter Impact on Map Fusion Performance

This experiment examines how the map fusion threshold parameter affects reconstruction performance across values ranging from 1m to 50m. The primary question is whether a specific threshold range exists where **IDR** graphs achieve optimal performance. Results from the baseline experiment (Section 6.2) showed that **I** outperforms **IDR** at a 30-meter threshold, which suggests a potential performance crossover point may exist at different threshold values. I expect the optimal threshold to fall within the 20-30 meter range, as this distance should allow the fusion algorithm to effectively identify similar roads with proper edge coverage and resolve road continuation issues.

Figure 24 and Figure 25 show the map fusion performance on map similarity metrics at different threshold values. Figure 26 shows information on graph complexity (number of nodes, edges, and average edge length) of the fused maps. Figure 27 shows the actions (number of edges injected, deleted, and reconnected) performed by the map fusion algorithm.

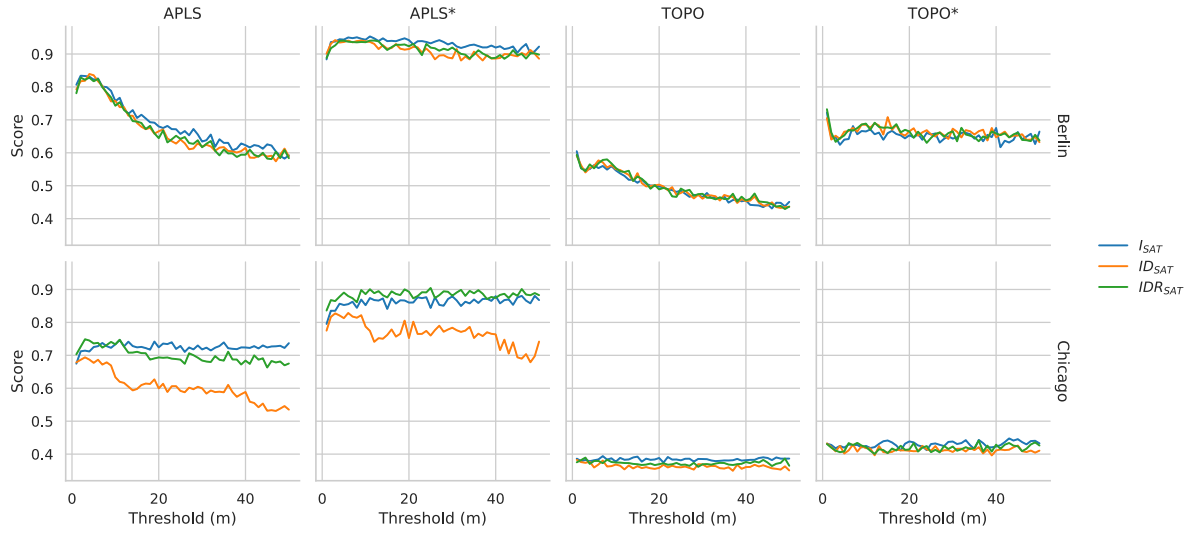


Figure 24: $\mathbf{IDR}_{\mathrm{SAT}}$ fusion performance at different thresholds (2000 normal, 400 prime samples).

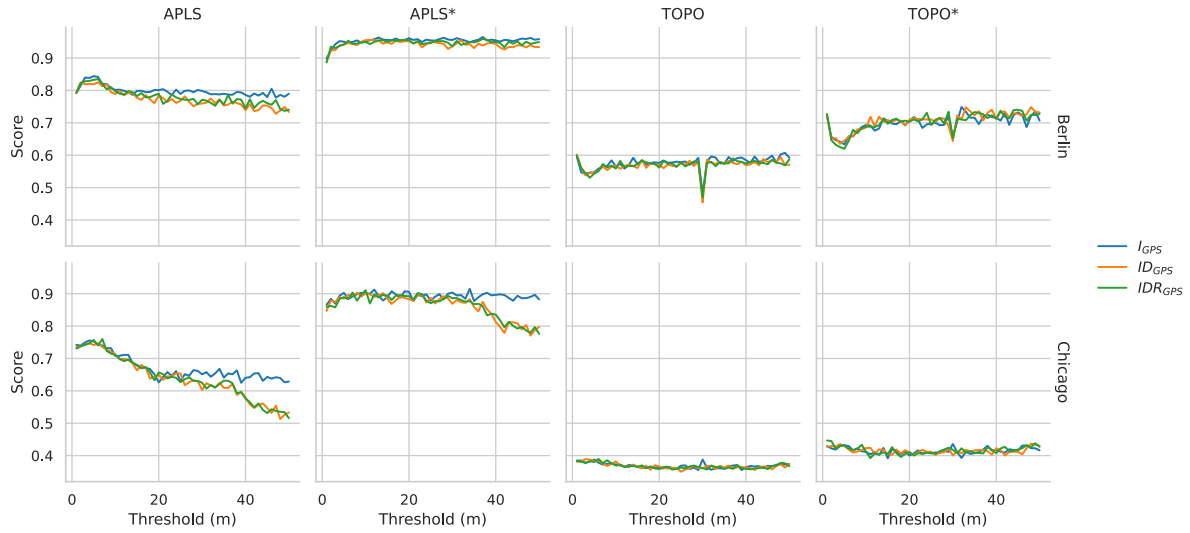


Figure 25: $\mathbf{IDR}_{\mathrm{GPS}}$ fusion performance at different thresholds (2000 normal, 400 prime samples).

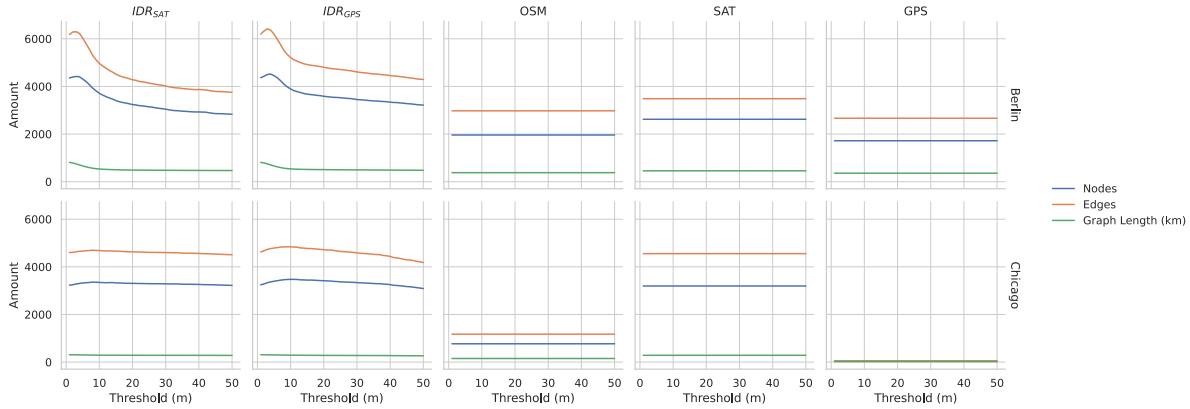


Figure 26: Graph complexity of the map fusion maps $\mathbf{IDR}_{\mathrm{SAT}}$ and $\mathbf{IDR}_{\mathrm{GPS}}$ at different thresholds.

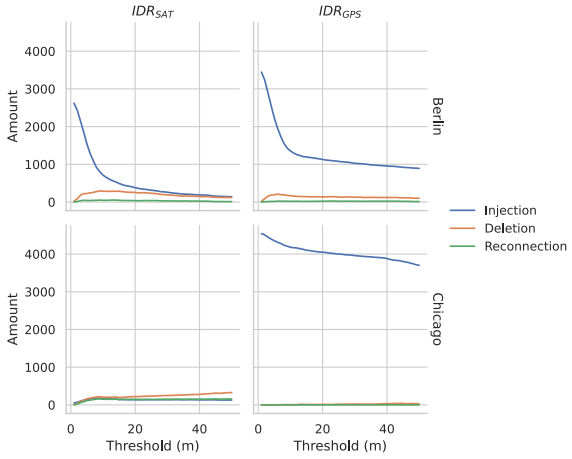


Figure 27: Actions performed in the map fusion process at different thresholds.

6.3.1. General observations in relation to hypotheses

Contrary to my expectations, the map similarity values generally remain stable across different threshold values. I consider the peaks and drops at very low thresholds to be artifacts and exclude them from this analysis, addressing these artifacts in the following section Section 6.3.5. Ignoring these artifacts, threshold changes have only minor performance impacts. The APLS and TOPO scores of the Berlin SAT-based fusion maps decrease linearly as the threshold increases. For GPS-based fusion maps, the APLS decreases linearly with increasing thresholds in both Berlin and Chicago, while the TOPO score increases with increasing thresholds for the Berlin map.

6.3.2. Better performing unimodal map leads fusion results

The APLS and TOPO scores decrease linearly with increasing threshold values in all SAT-based fusion maps of Berlin. This decline occurs because the number of injected GPS edges in $\mathbf{IDR}_{\mathrm{SAT}}$ decreases as the threshold rises, as shown in Figure 27.

As the threshold increases, the fusion process replaces a smaller portion of the SAT map with GPS edges, meaning the algorithm retains more edges from the lower-performing SAT map. Since Experiment 1 demonstrated that Berlin GPS performs better than Berlin SAT (Section 6.2), this reduction in GPS edge injection decreases overall performance. The results indicate that $\mathbf{ID}_{\mathrm{SAT}}$ and $\mathbf{IDR}_{\mathrm{SAT}}$ follow this same pattern, suggesting that the most effective strategy involves maximally incorporating GPS edges into the fused map.

When GPS serves as the base map for Berlin, all three fusion variants perform optimally at the highest threshold values, which aligns with the behavior observed in $\mathbf{I}_{\mathrm{GPS}}$. The results indicate that optimal performance occurs when minimal SAT edge injection takes place, as demonstrated in Figure 27.

In contrast, the Chicago GPS-based reconstruction exhibits inverse behavior, with performance deteriorating as threshold values increase. Since the SAT map outperforms the GPS map in Chicago, the same underlying pattern emerges: as the threshold rises and fewer high-quality SAT edges get incorporated into the fused map, overall performance declines.

6.3.3. Minimal performance impact of the combined deletion and reconnection steps

The performance relationship between fusion methods reveals that $\mathbf{IDR}_{\mathrm{SAT}}$ generally maintains comparable results to $\mathbf{I}_{\mathrm{SAT}}$. This means that the combined deletion and reconnection steps generally do not benefit performance compared to the straightforward approach of incorporating edges from the best performing unimodal map.

There is one notable exception with the APLS and APLS* scores for Chicago $\mathbf{ID}_{\mathrm{SAT}}$, which shows significantly lower performance compared to both $\mathbf{I}_{\mathrm{SAT}}$ and $\mathbf{IDR}_{\mathrm{SAT}}$. This demonstrates the need to reconnect after deletion to maintain performance, and in this case, the reconnection step successfully recovers the performance loss from deletion.

6.3.4. Fluctuations in map similarity values

The map similarity values show fluctuations at all threshold values. These fluctuations occur either due to stochastic noise or because of localized effects where a significant number of edges simultaneously pass the edge coverage threshold at specific values. Because I took a substantial number of samples, the stochastic noise should be small, so the fluctuations can be attributed to these localized edge coverage effects.

However, the large jump of Berlin GPS-based TOPO around 30m is too substantial to be explained by stochastic noise alone. Similar jumps are not observed at this threshold in other plots presented here, and no corresponding peculiarities appear in the edge coverage plots in text:experiments:edge-coverage. I suspect the plot generation did not sample adequately at this value due to a bug in my experiment code, particularly since the anomaly occurs suspiciously close to 30m, which was the sampling interval for many experiments.

6.3.5. Unexplainable peak in graph complexity at a threshold of 4m

The relationship between threshold and edge injection should follow a predictable pattern. As the threshold increases, fewer patch edges should be injected because more patch edges become covered by the base map, while simultaneously more base edges should be deleted as they become covered by the injected patch edges. This inverse relationship means that the total number of graph edges can only decrease as the threshold increases.

However, this expected behavior makes the peak graph complexity around 4 meters for both $\mathbf{IDR}_{\mathrm{SAT}}$ and $\mathbf{IDR}_{\mathrm{GPS}}$ in Berlin unexplainable to me. What makes this even more contradictory is that the total graph edge length decreases around this threshold value in Berlin, which coincides with both the steep drop in injected edges and the peak in edge deletion.

6.3.6. Edge injection behavior

The map fusion actions illustrated in Figure 27 reveal that edge injections decrease significantly at thresholds below 10 meters. This pattern aligns with the edge coverage analysis from Table 15, which demonstrates that a substantial proportion of edges fall within the 10-meter edge coverage range.

Furthermore, the edge coverage data shows that injection rates stabilize around 30 meters for Berlin and 20 meters for Chicago, indicating that most proximate edges have been processed at these threshold values. These represent practical upper bounds for meaningful fusion thresholds.

6.4. Detailed Analysis of Unimodal Map Properties and Performance

This experiment examines the properties and similarity characteristics of unimodal maps compared to the ground truth. Understanding these individual map behaviors is crucial for interpreting what happens during fusion, since the base map properties directly influence fusion outcomes. The full maps can be viewed in Table 17 for a visual impression.

The analysis is organized as follows: Table 14 provides basic map statistics, Table 15 presents edge coverage distributions, and Table 16 shows map distance distributions.

6.4.1. Basic Properties of Unimodal and Ground Truth Maps

SAT reconstructs too many nodes and edges, resulting in total graph length that exceeds the ground truth. In Berlin, SAT shows reasonable performance, but Chicago SAT predicts twice as many edges with half the average edge length compared to the ground truth. This overestimation likely stems from inference of service roads (like parking areas) not included in OSM, as well as non-existing alleys.

Berlin GPS aligns well with Berlin OSM across all properties, matching node count, edge count, and total graph length. Chicago GPS shows a different pattern with few nodes connected by long edges. This pattern results from a limited number of inferred roads, which reduces intersections (lower node count) and creates longer segments due to fewer breaking points. The sparse road count stems from limited GPS samples available for Chicago (see Table 6).

		Nodes	Average Node Degree	Edges	Total Length (m)	Average Edge Length
Berlin	**OSM**	1,961	3.04	2,976	379,065	127.37
	SAT	2,621	2.66	3,485	455,754	130.78
	GPS	1,719	3.10	2,664	357,631	134.25
Chicago	**OSM**	769	3.05	1,173	150,999	128.73
	SAT	3,196	2.85	4,553	285,710	62.75
	GPS	33	3.09	51	23,530	461.36

Table 14: Network statistics for Berlin and Chicago across different data sources (OSM: OpenStreetMap, SAT: Satellite, GPS: GPS traces).

6.4.2. Edge Coverage Distribution Analysis

Edge coverage is the core mechanic of map fusion, so understanding its distribution provides insights into fusion performance. Table 15 visualizes these distributions using a log10 x-axis scale, which clearly shows that most edge coverage occurs at low distance thresholds. The horizontal dashed red line indicates total edge count in each map.

Most maps have edges covered under 100 meters, with the notable exception of maps using Chicago GPS as target. Edges with coverage above 100 meters indicate source roads in regions where the target graph lacks nearby roads.

The Chicago analysis reveals an interesting asymmetry: SAT-to-OSM coverage shows 30% of edges with thresholds above 50m, while OSM-to-SAT shows only 5%. Initially this seemed surprising since Chicago SAT and OSM appear similar (see Table 19), aside from the service roads noted earlier. However, the diagram reveals that around 2000 SAT edges fall within 30-100m coverage thresholds. These edges likely represent SAT road predictions in regions where OSM has no corresponding infrastructure.

6.4.3. Graph Distance Analysis for Metric Comparison

To understand performance differences between APLS/APLS* and TOPO/TOPO* metrics, I measure spatial distances between graph structures at random positions. This distance distribution should help predict sampling success probability and directly impact TOPO* performance by revealing deviation patterns in road geometry.

The distance distributions show a consistent pattern: clear KDE peaks within 0-10 meters followed by sharp declines, indicating good spatial proximity across road networks. Chicago shows two notable deviations from this pattern. First, Chicago SAT and OSM comparisons against Chicago GPS show extended distribution tails due to GPS sparseness limiting nearby reference points. Second, Chicago SAT versus OSM reveals distributional anomalies from SAT's road predictions in regions lacking OSM infrastructure, confirming patterns identified in the edge coverage analysis.

6.5. TOPO and APLS Sample Distribution Analysis

The baseline map performance experiment Section 6.2 provides overall performance metrics but lacks detailed information about the underlying APLS and TOPO sample characteristics, which could be valuable for comprehensive performance assessment.

This information becomes especially important when drawing conclusive remarks about performance differences between maps. Understanding whether superior performance stems from fewer poor-quality samples or the presence of exceptionally good samples provides crucial context for interpreting the baseline results. The resulting distribution analysis should provide insights that complement the baseline experimental findings.

For the sample distribution plots, I focus on TOPO* and APLS* metrics specifically, since the number of successful initial sample seeds can be derived from the graph distance distribution analysis presented in Section 6.4.3.

The sample distribution plot is presented in Figure 32.

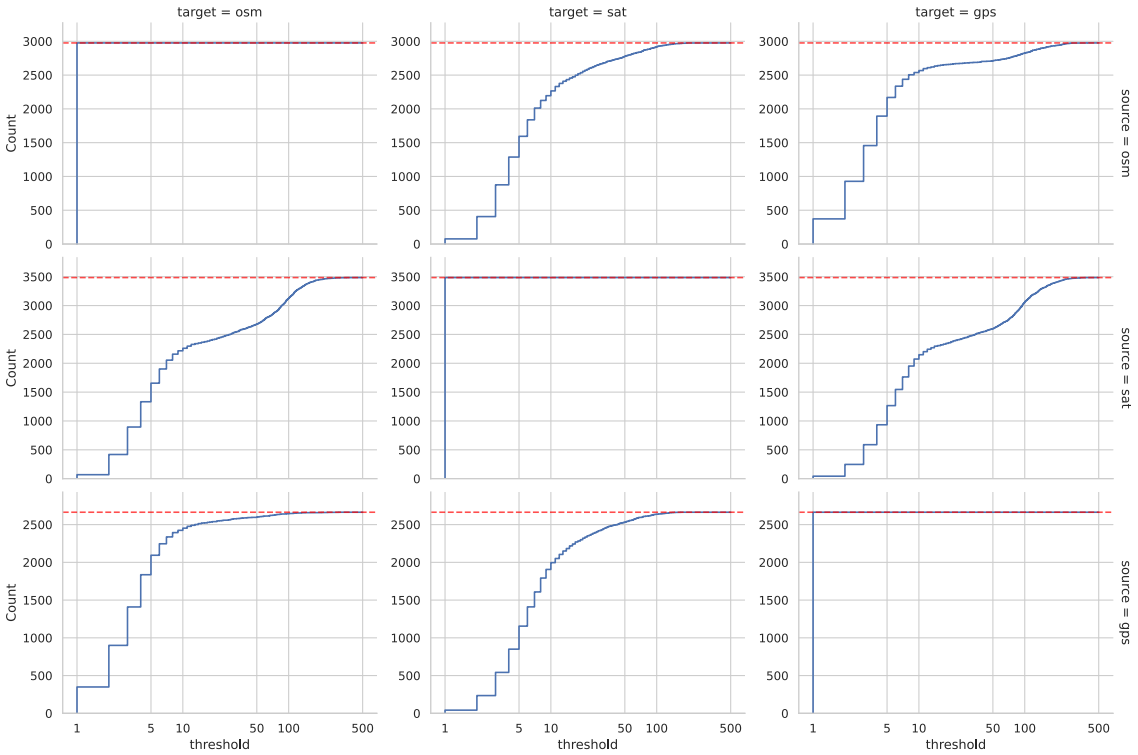


Figure 28: Empirical cumulative distribution matrix of edge coverage on Berlin.

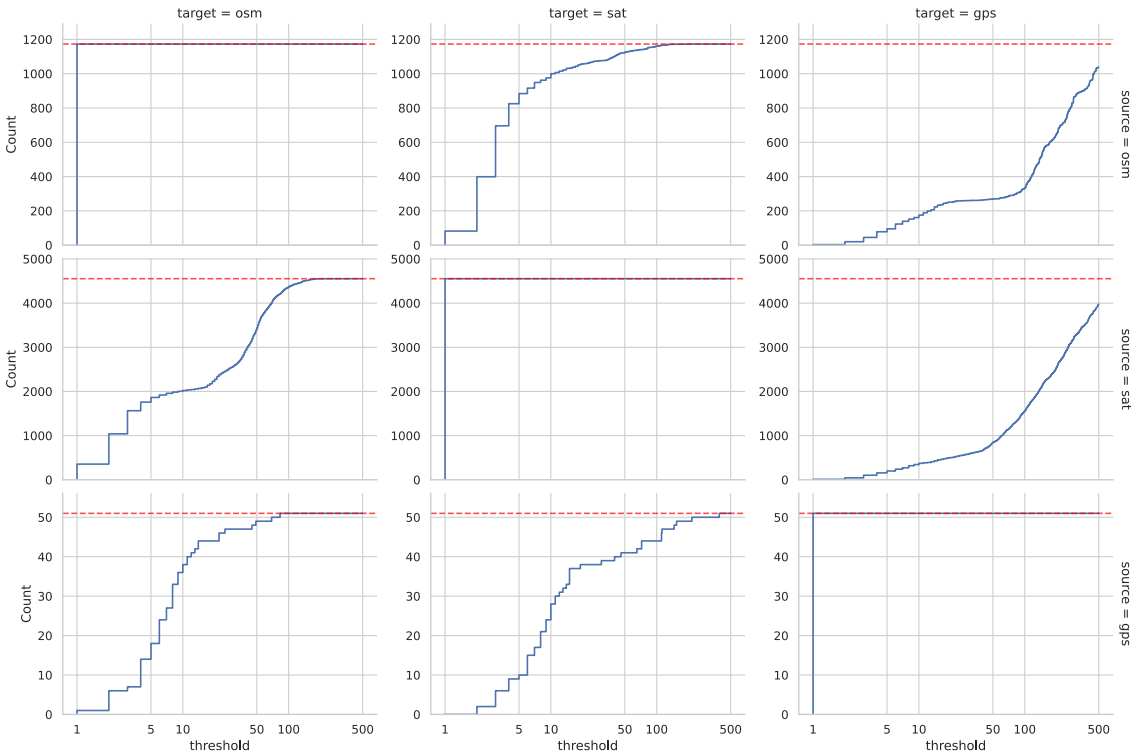


Figure 29: Empirical cumulative distribution matrix of edge coverage on Chicago.

Table 15: Empirical cumulative distribution matrix of edge coverage on base maps (GPS, SAT, and OSM). The red dashed line represents total edges in the source graph. The x-axis uses log10 scale covering $[0, 500]$ meters. Edge coverage is measured from the source graph (rows).

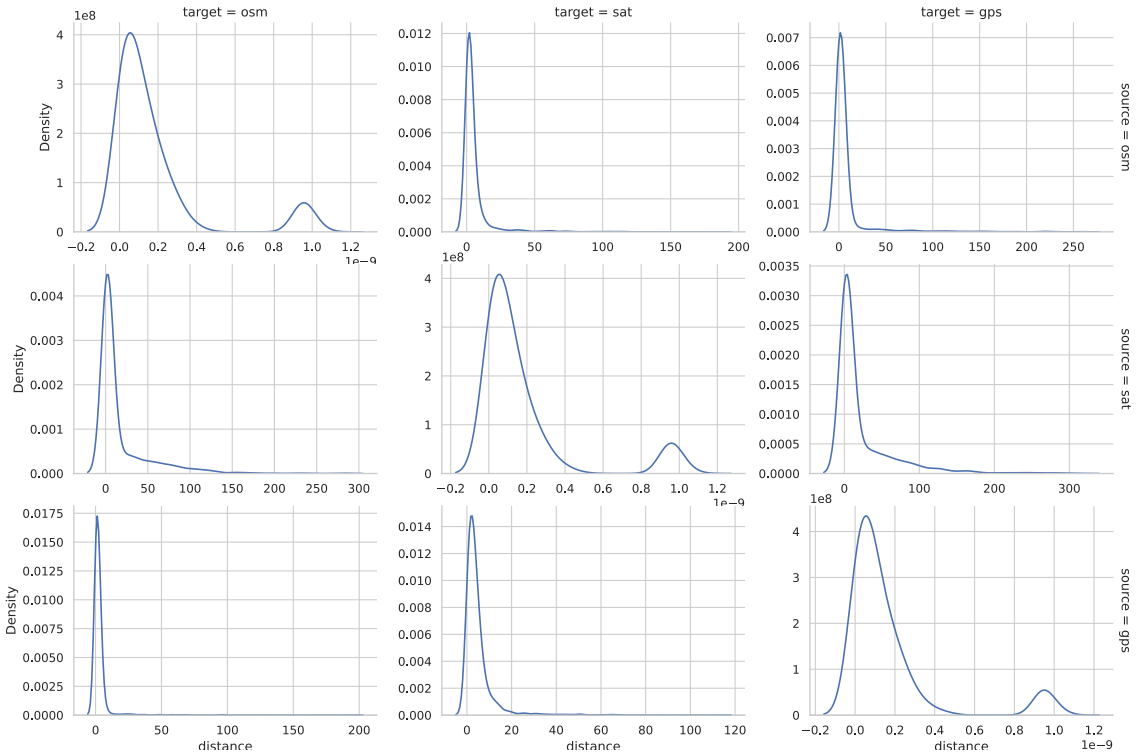


Figure 30: Graph sample distance matrix on Berlin.

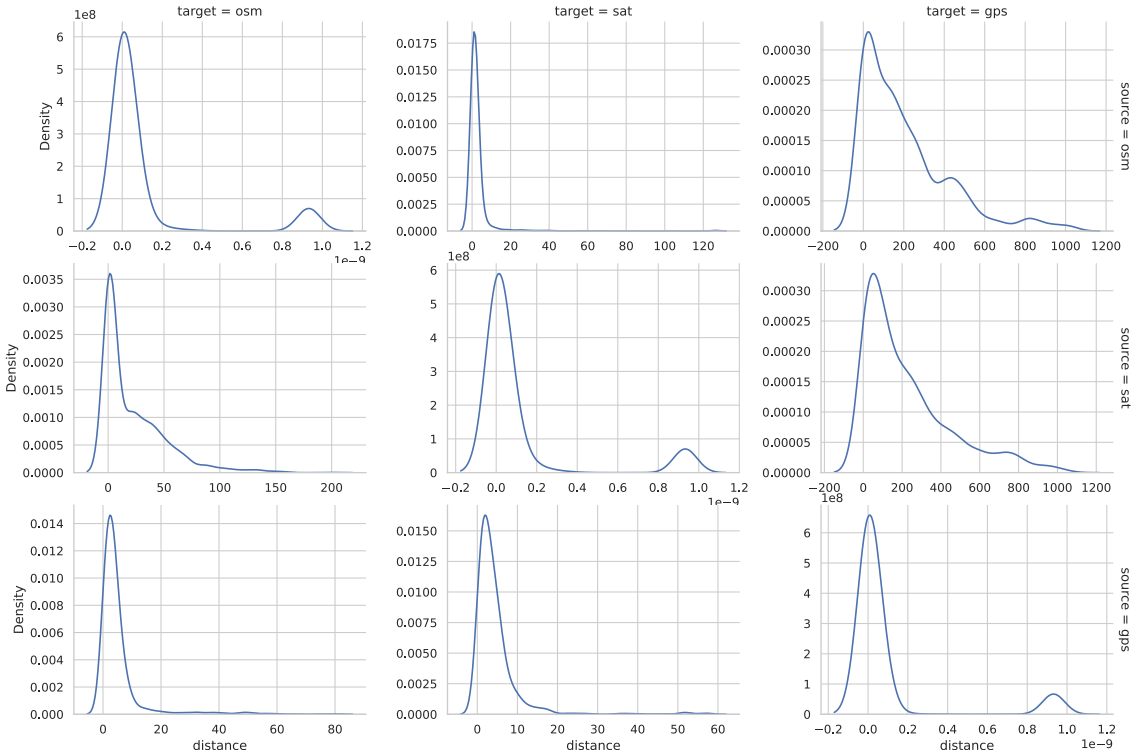


Figure 31: Graph sample distance matrix on Chicago.

Table 16: Graph sample distance matrix comparing unimodal maps and ground truth, with kernel density estimation using 2000 samples per visualization. (Note: diagonal entries have their x-axis scaled by $1e^{-9}$, so all samples measure essentially 0m distance.)

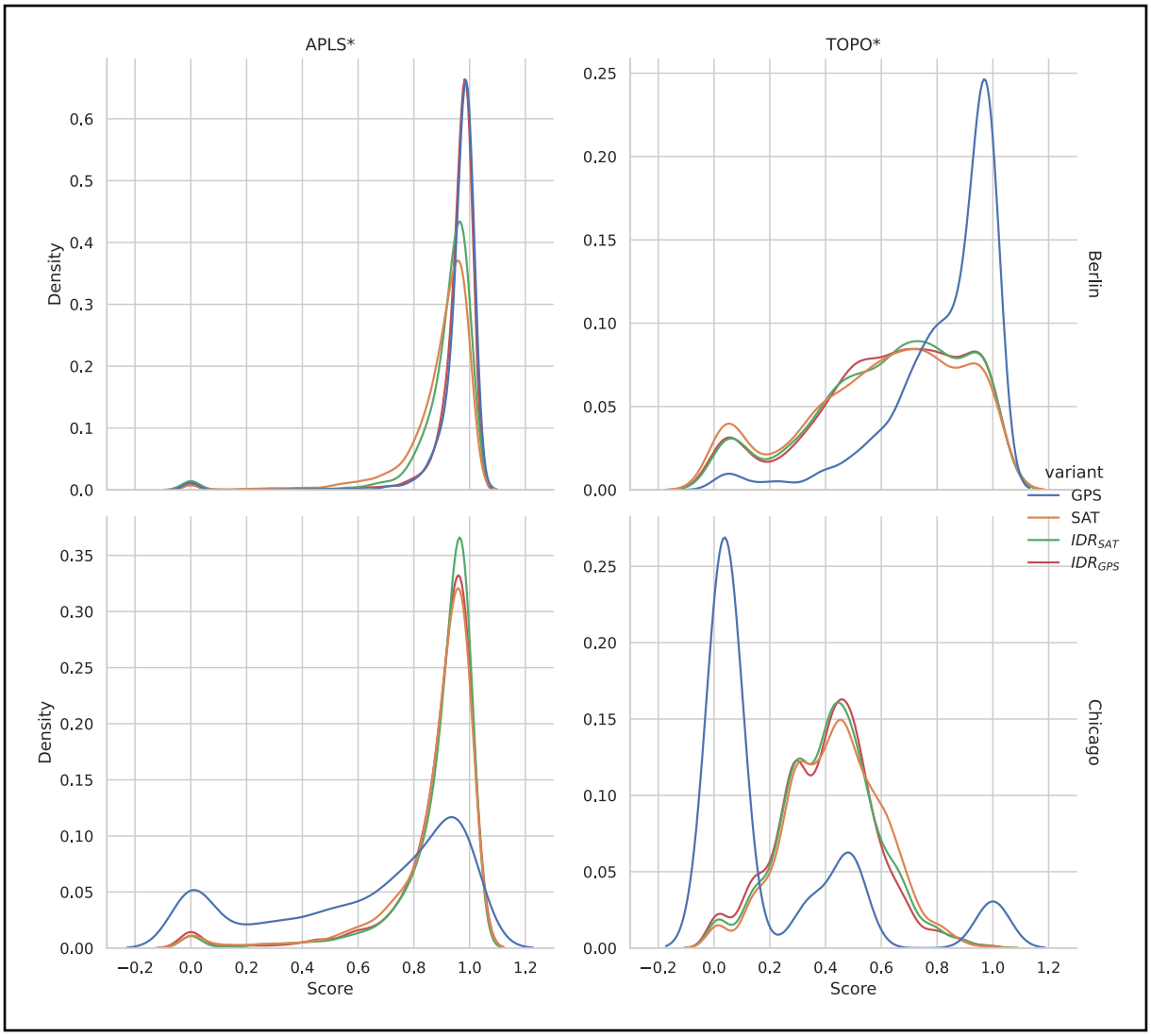


Figure 32: Sample score distributions for TOPO* and APLS* metrics. Kernel density estimations are computed using 2000 samples.

6.5.1. Distribution peak at zero

All distributions exhibit a small peak or dent near zero, and I think the underlying cause is the same for both metrics: a sample taken at a position where either the source or target graph has a disconnected subgraph, but not both.

For APLS, these values can only occur when the predicted and ground truth networks disagree on path existence between start and end nodes. This pattern indicates the presence of small disconnected graphs in the network, affecting only a limited number of sample seeds at these positions. Since these cases show agreement in road existence (otherwise the initial sample seed would not have succeeded), the disagreement stems specifically from road continuation patterns.

The TOPO metric shows a similar pattern with the same underlying cause. Disconnected subgraphs create asymmetric sampling conditions where the map with the connected graph allows hole or marble placement throughout the entire sampling radius, while the disconnected counterpart restricts placement to the minor region of the disconnected graph. This asymmetry results in significantly reduced recall and precision values.

The Chicago GPS maps present a distinctly different TOPO distribution pattern compared to other datasets. The observed low precision and recall scores result from sparse road density within the sampling radius, which fails to provide sufficient network structure for meaningful topological comparison. This sparsity creates two notable characteristics: a less flat curve between 0 and 1 where shortest path differences span the entire interval, and an increased peak around 0. Additionally, the TOPO* distribution exhibits three distinct peaks at approximately 0,

0.5, and 1. The peaks at 0.5 and 1 likely originate from two specific geographic locations, though I didn't have sufficient time to fully visualize and confirm this hypothesis.

6.5.2. Fusion map sampling distributions follow those of SAT

The sampling distributions reveal an interesting pattern where both $\mathbf{IDR}_{\mathrm{SAT}}$ and $\mathbf{IDR}_{\mathrm{GPS}}$ variants demonstrate stronger alignment with the **SAT** distribution than with the **GPS** distribution. Intuitively I would expect each fusion variant to follow its respective base map's distribution, however the data reveals a different pattern entirely. $\mathbf{IDR}_{\mathrm{GPS}}$ aligns more closely with the SAT distribution patterns despite GPS being its base map.

This counterintuitive behavior cannot be explained by SAT consistently outperforming GPS, since SAT does not actually perform better across all evaluation metrics for Berlin and Chicago. The most plausible explanation I can think of is the difference in edge density between GPS and SAT maps. Since SAT has higher edge density, all fusion maps end up with relatively more SAT edges, which then dominate the sampling distribution characteristics. I do not, however, test this hypothesis in my experiments.

6.6. Impact of TOPO Hole Size Parameters on Map Performance

This experiment examines whether relative map performance is sensitive to TOPO hole size. TOPO recall and precision should properly capture the amount of road presence in reconstructed maps, but two factors can cause unexpectedly low scores. We already discussed the error margin issue in Section 4.3.2 as one cause. The other cause occurs when nearby roads are positioned slightly out of range, causing many failed samples even though the maps look visually similar. This positioning issue suggests that relative map performance might be quite sensitive to the TOPO hole size parameter.

I hypothesize that a fused map performs relatively better at TOPO when its threshold is similar to the chosen TOPO hole size, compared to fused maps constructed with significantly different thresholds. To test this, I compute the TOPO score using different hole sizes against three fused maps constructed with thresholds of 10m, 30m, and 50m. I expect each map to achieve its peak performance when evaluated with a TOPO hole size corresponding to its construction threshold.

The results are presented in Figure 33.

The hypothesis that fusion threshold correlates with optimal TOPO hole size does not hold based on these results. Rather than showing the expected threshold-specific performance peaks, relative performance between maps remained fairly consistent across different hole sizes once the hole size exceeded 5m. This suggests that the positioning sensitivity I expected from slightly misaligned roads does not significantly impact the TOPO score, meaning small geometric variations have less influence on the evaluation metric than I anticipated.

The accuracy of the map fusion performance experiment is somewhat questionable regarding its TOPO and TOPO* scores, because the samples are taken at 5.5 meters. This threshold sits just slightly above the distance at which TOPO scores become unstable. However, I consider this limitation not particularly relevant since all collected experiment data already provides sufficient insight into the performance characteristics. There remains no doubt about how well the different fusion methods perform, and these experiments do not attempt to provide any form of official benchmark.

This 5m threshold appears to correlate with the point where most of the graph sampling distance density falls within the measurement range. This finding might provide guidance for future research: when the graph sampling distance density concentrates at a higher threshold, it may indicate a need to increase the TOPO hole size as well to prevent unstable metric results.

6.7. Qualitative Analysis of Map Fusion Results

This section presents a visual analysis of the base maps and generated fusion maps to examine their structural differences and fusion outcomes.

Table 17 displays the complete maps for both Berlin and Chicago datasets: the ground truth map, the unimodal reconstructed maps, and the final fusion results (both $\mathbf{IDR}_{\mathrm{SAT}}$ and $\mathbf{IDR}_{\mathrm{GPS}}$) using a threshold of 30m.

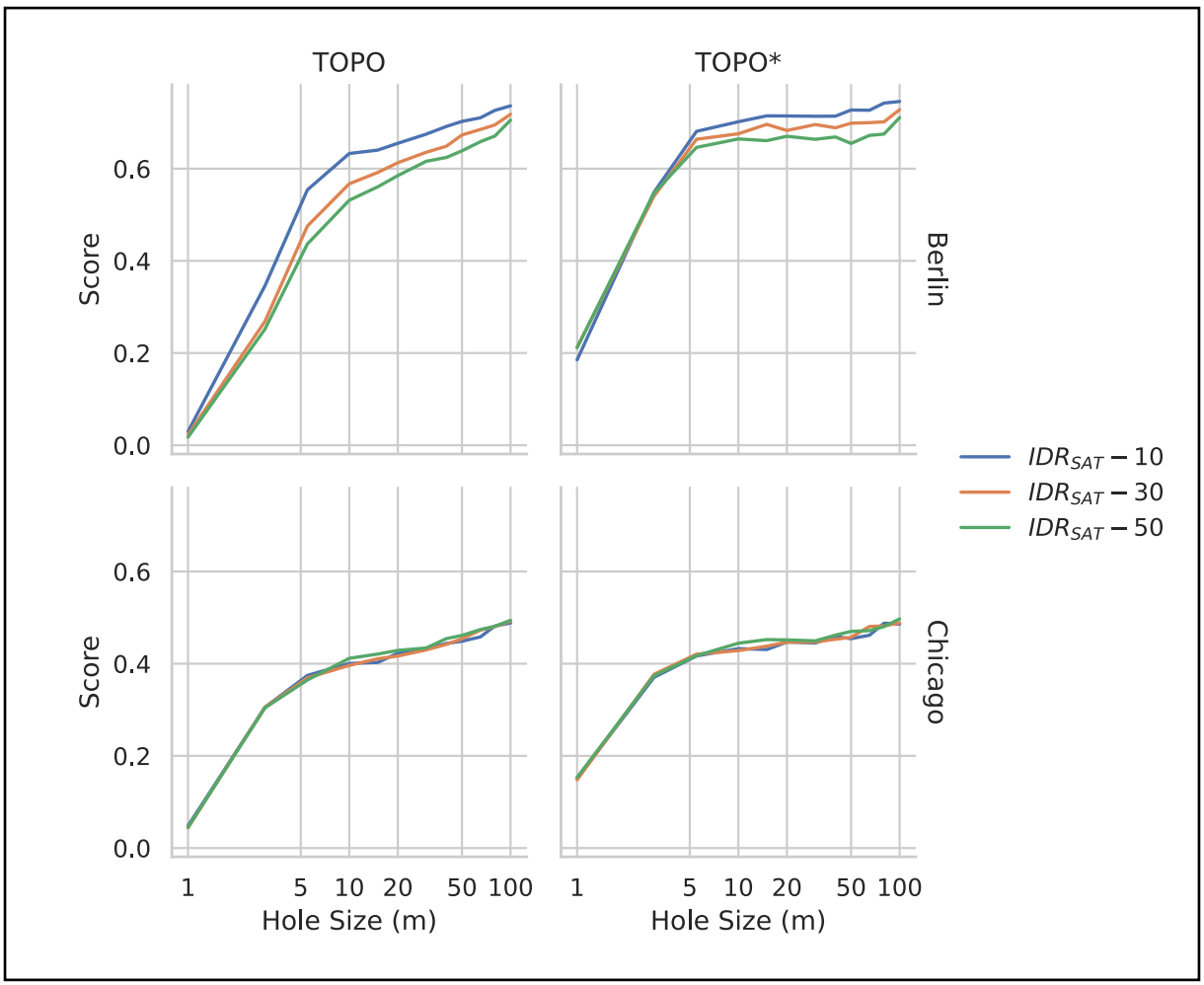


Figure 33: TOPO and TOPO* map similarity scores computed across different hole sizes (ranging from 0m to 50m) for three $\mathbf{IDR}_{\mathrm{SAT}}$ fusion maps generated using different distance thresholds (10m, 30m, and 50m). Each analysis uses 2000 samples per TOPO data point and 1000 samples per TOPO* data point.

Table 20 and Table 21 illustrate the specific actions performed during the map fusion process across the entire fusion maps at thresholds of 1m, 30m, and 50m. This shows how the map fusion algorithm behaves differently as the threshold increases.

Figure 44, Figure 48, Figure 48, Figure 52 provide detailed views of selected areas where the practical outcomes of the three map fusion actions are most clearly visible. These zoomed-in views allow for closer examination of how injection, deletion, and reconnection operations affect the final map structure in both Berlin and Chicago datasets. The detailed views use a threshold of 30m, as this threshold effectively demonstrates the general behavior of the map fusion process.

6.7.1. Visual Comparison of Complete Maps

The full map comparisons in Table 17 reveal that $\mathbf{IDR}_{\mathrm{GPS}}$ and $\mathbf{IDR}_{\mathrm{SAT}}$ produce visually similar results for both Berlin and Chicago datasets. The fusion action visualizations in Table 20 and Table 21 help explain this similarity by showing that almost any remote edge gets injected during the first step of the algorithm. This makes sense because these patch map edges lack coverage from the base map. Consequently, regardless of whether GPS or SAT serves as the base map, both resulting graphs end up containing edges in the same general regions.

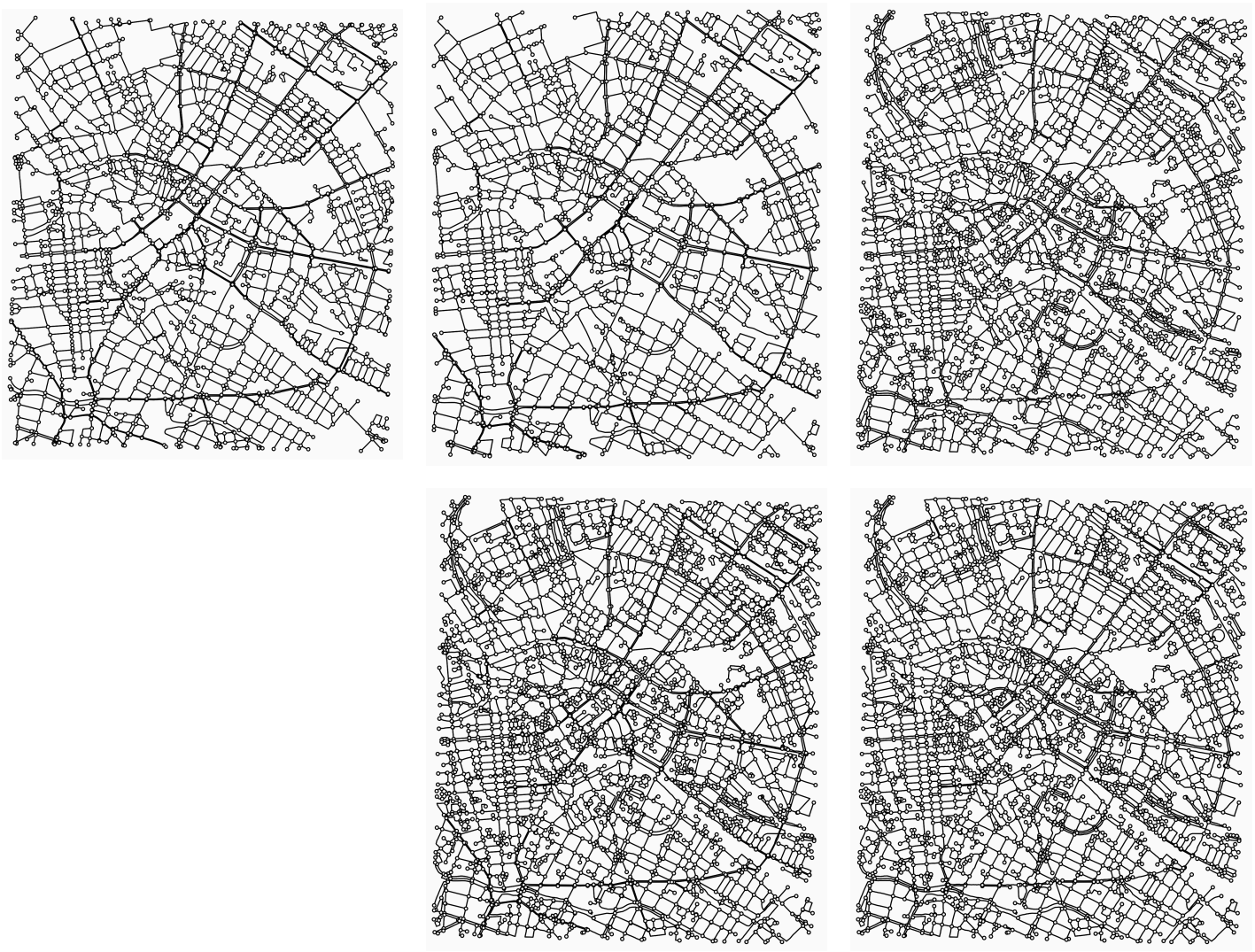
6.7.2. Fusion Process Behavior Across Different Thresholds

As the threshold increases, fewer patch map components are injected into the base map, while more base map components are identified as covered and subsequently deleted.

At the default threshold of 30m, the fusion process demonstrates distinct regional behaviors. In Chicago, $\mathbf{IDR}_{\mathrm{GPS}}$ shows effective connectivity maintenance where numerous purple reconnection nodes successfully

Berlin

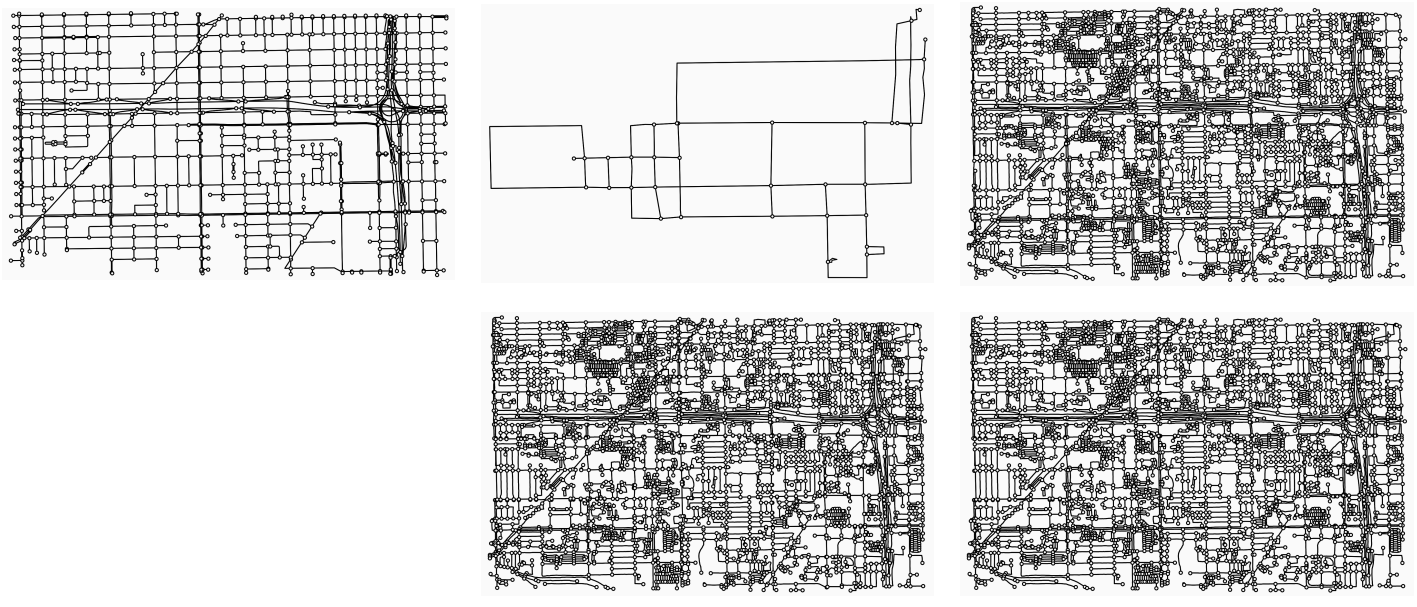
OSM | GPS | SAT



IDR_{GPS} | IDR_{SAT}

Chicago

OSM | GPS | SAT



IDR_{GPS} | IDR_{SAT}

Table 17: Map visualization for Berlin and Chicago showing ground truth, unimodal reconstructions, and final fusion results ($\mathbf{IDR}_{SAT}$ and $\mathbf{IDR}_{GPS}$) with 30m distance threshold applied.

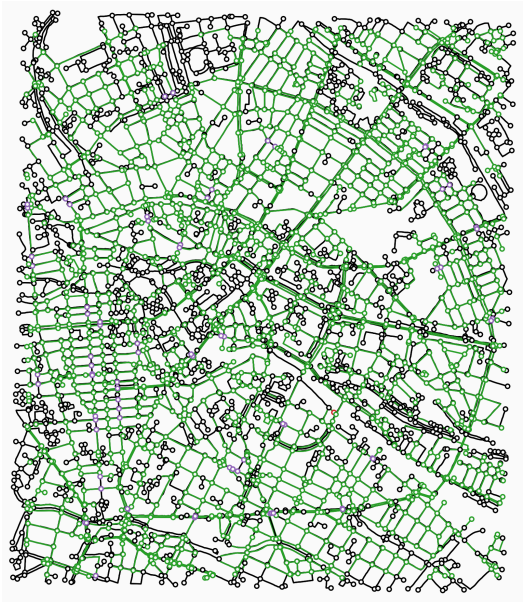
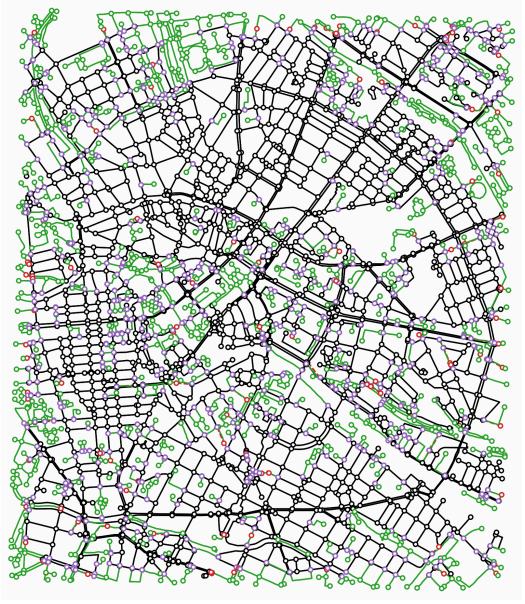
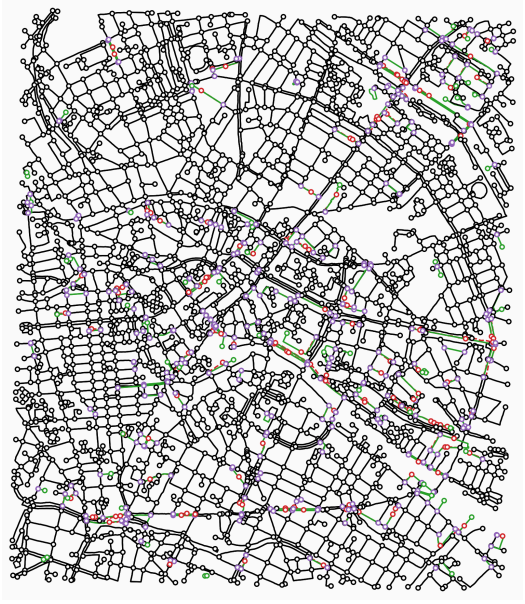
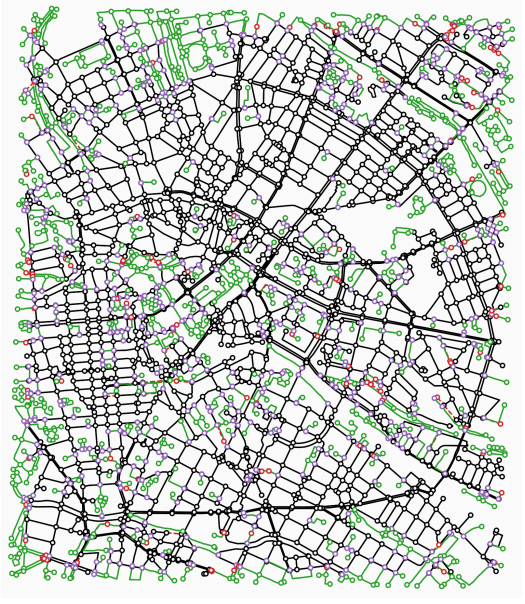
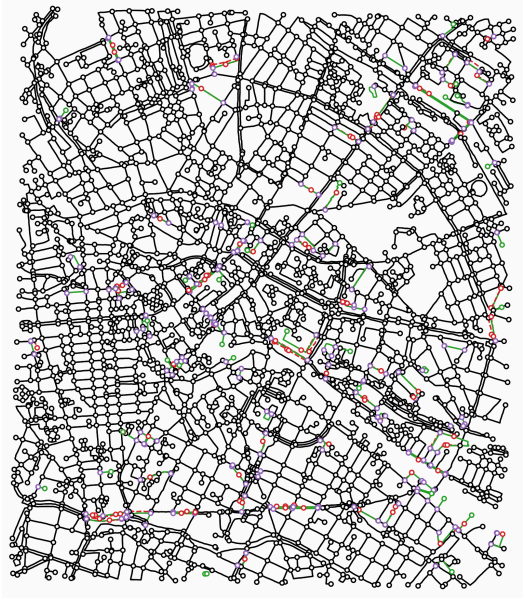
	IDR_{GPS}	IDR_{SAT}
1m		
30m		
50m		

Table 20: Fused maps of Berlin at different map fusion thresholds (1m, 30m, 50m) showing fusion actions through colored nodes and edges: Black (unchanged base map), green (injected patch edges), red dashed (deleted base edges), and purple dotted (connection and reconnection edges).

Table 21: Fused maps of Chicago at different map fusion thresholds (1m, 30m, 50m) showing fusion actions through colored nodes and edges: Black (unchanged base map), green (injected patch edges), red dashed (deleted base edges), and purple dotted (connection and reconnection edges).

link the injected SAT graph components to the original GPS edges. Berlin's $\mathbf{IDR}_{\mathrm{GPS}}$ reveals multiple subregions containing injected SAT subgraphs that fill gaps missing from the original GPS map. Across all four fusion map variants, deletion operations occur only in localized areas, indicating that opportunities for road continuation improvements remain relatively sparse.

At an extremely low threshold of 1m, the algorithm injects the entire patch map while deleting no base map edges, resulting in no reconnection operations since no base map edges are removed. Conversely, at a high threshold of 50m, significantly fewer patch map edges qualify for injection since more are considered covered by the base map. This higher threshold does trigger base map edge deletions, as more base map edges are covered by the injected patch edges.

Figure 35: A discontinuating SAT edge is replaced by a continuating GPS edge during fusion.

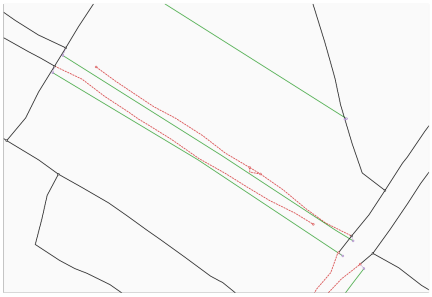


Figure 36: Two parallel discontinuating SAT edges are replaced by two continuating GPS edges.

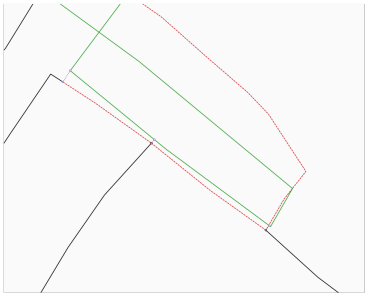


Figure 37: GPS edge injection adds two nodes to the graph while three SAT edges are deleted.

Figure 38: A single GPS edge covers two parallel partially discontinuating SAT edges.

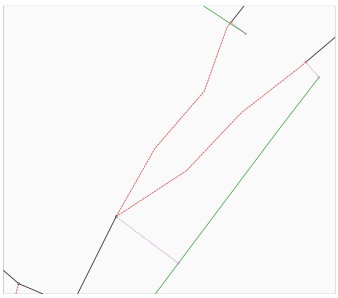


Figure 39: A single injected GPS edge covers two parallel continuating SAT edges.

Figure 40: GPS edges are injected for side alleys not covered by the SAT graph.

Figure 41: Complex GPS infrastructure replaces a single discontinuating SAT edge.

Figure 42: Two GPS edges are injected, but the connection at the corner of the deleted SAT edge is lost.

Figure 43: Both discontinuating and continuating SAT edges are replaced during fusion.

Figure 34: Detailed view of Berlin $\mathbf{IDR}_{\mathrm{SAT}}$ map fusion steps with specific examples of injection, deletion, and reconnection actions.

6.7.3. Zoomed-in Map Fusion Behavior

Detailed examination of the results in Figure 44, Figure 48, Figure 48, Figure 52 shows mixed performance from the algorithm. The process works well in straightforward cases but encounters difficulties with complex road geometries and parallel structures.

Figure 45: Multiple SAT edges are injected, including segments that appear incorrectly discontinuating.

Figure 46: Injection of an alley inferred by the SAT graph, including its reconstruction artifacts.

Figure 47: An injected SAT edge from the circular road connects to a GPS edge that is subsequently deleted.

Figure 44: Detailed view of Berlin $\mathbf{IDR}_{\text{GPS}}$ map fusion steps with specific examples of injection, deletion, and reconnection actions.

In successful cases, the algorithm demonstrates its intended functionality by effectively resolving mapping inconsistencies through combining complementary information from both input maps. The three-step process replaces discontinuous edges with continuous ones while preserving map topology.

The Chicago SAT-based results highlight why the reconnection step is essential. Long GPS edges injected into the SAT base map cause many short SAT segments to be deleted in the second step. Without reconnection, these first two steps would create more discontinuous edges than the original map contained, but the third step successfully restores proper connectivity.

When using GPS as the base map, the algorithm sometimes injects entire subgraph regions from the SAT patch map and connects them directly to the base without requiring additional modifications.

The algorithm shows limitations when processing parallel roads, sometimes removing multiple edges that should be preserved in the final result. Similar problems occur when input maps have significantly different edge lengths or geometries, causing the algorithm to misidentify corresponding road segments and make incorrect deletion decisions.

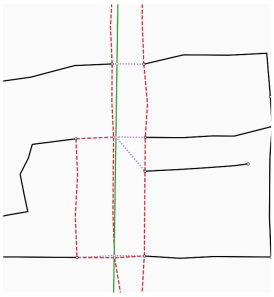


Figure 49: Two sets of parallel SAT edges are replaced by a single GPS edge with reconnection.

Figure 50: Short, fragmented SAT edges are consolidated into one continuous GPS edge.

Figure 51: A single long GPS edge that is somewhat off-target causes arbitrary short SAT edges to be deleted and breaks the topology.

Figure 48: Detailed view of Chicago $\mathbf{IDR}_{\text{SAT}}$ map fusion steps with specific examples of injection, deletion, and reconnection actions.

Figure 53: Injection of an entire neighborhood; the numerous connection nodes indicate that the GPS edge covered multiple SAT edges.

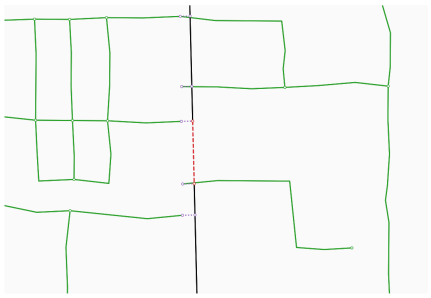


Figure 54: Incorrect deletion of short base edge covered by injected neighborhood.

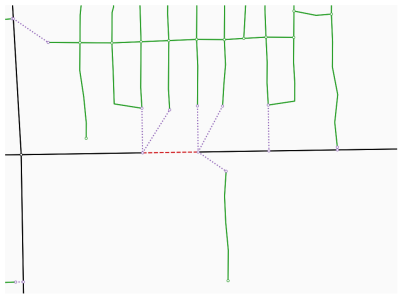


Figure 55: Incorrect deletion of short base edge covered by injected neighborhood.

Figure 52: Detailed view of Chicago $\mathbf{IDR}_{\text{GPS}}$ map fusion steps with specific examples of injection, deletion, and reconnection actions.

This qualitative analysis uncovered an algorithmic flaw illustrated in Figure 47. The algorithm injects a patch edge that connects to a base edge, then deletes that same base edge because it appears covered by the newly injected patch edge. While this issue occurs infrequently, it demonstrates how conflicting road patterns like roundabouts versus straight segments create matching difficulties. These problems are more common when GPS serves as the base map, since SAT-based reconstruction typically captures geometric features more accurately.

Additionally, note the impact of the iterative approach to edge connection is shown in Figure 49 and Figure 55. Instead of properly extending roads in a straight line and restoring actual road topology, some form of crossing is constructed instead.

7. Discussion

This section interprets the experimental findings, uses these insights to answer the research questions, and reflects on the research overall. The experimental results in Section 6 contradict my initial hypotheses about map fusion effectiveness, yet they provide important insights into multi-modal road network reconstruction.

I first summarize the experimental findings in Section 7.1. I then answer the research questions from Section 6.1 in Section 7.2. Finally, I reflect on broader implications and limitations in Section 7.3.

7.1. Key findings from map fusion experiments

7.1.1. Fusion success depends on context

Quantitatively, map fusion does not reliably improve upon the best performing unimodal reconstruction. Instead, fusion effectiveness proves highly context-dependent, varying significantly between regions and input map quality. Berlin favors GPS-based fusion variants while Chicago favors SAT-based approaches, which suggests that fusion success correlates with the relative performance gap between input maps rather than the fusion process itself.

7.1.2. Injection step delivers the best results

Performing only the edge injection step often quantitatively outperforms the complete fusion pipeline of injection, deletion, and reconnection. From a performance perspective, duplicated road infrastructure that includes correct roads proves better than single infrastructure with partially misaligned geometry. The risk of replacing correct base map edges with misaligned patch map edges outweighs the benefits of removing duplicates.

Threshold analysis confirms copying behavior.

The threshold analysis reveals remarkably stable performance across the 1m-50m range with no clear optimal threshold. This stability occurs because fusion map performance correlates with the proportion of edges retained from the better-performing input map. This copying behavior indicates that fusion effectiveness depends on preserving edges from the higher-quality source rather than meaningful road continuation resolution. The gradual performance changes with thresholds reflect gradual edge inclusion and exclusion rather than true improvements.

Sample distributions reveal edge copying dominance.

Fusion distributions consistently follow SAT patterns even when GPS serves as the base map, including cases where GPS outperforms SAT. I think SAT's higher edge density is the primary factor behind this behavior. This pattern indicates that fused maps become dominated by SAT edges, making sample distributions reflect SAT characteristics regardless of performance expectations. The algorithm functions more like naive edge copying than sophisticated integration.

Visual assessment shows similar outcomes across variants.

Visual examination reveals that $\mathbf{IDR}_{\mathrm{GPS}}$ and $\mathbf{IDR}_{\mathrm{SAT}}$ produce remarkably similar results. This similarity occurs because the injection step copies edges at regions with road absence rather than through meaningful fusion improvements.

However, qualitatively the maps are worse off when road infrastructure is duplicated. It does raise the question how effective the similarity measures are at describing this component of a map.

7.1.3. Limited opportunities for road continuation resolution

The consistently low deletion and reconnection counts indicate few locations where road continuation issues can be addressed. This contradicts the fundamental assumption that widespread road continuation conflicts exist between GPS and SAT reconstructions. Action plots show injection counts reaching high levels while deletion and reconnection remain consistently low across all thresholds.

Visual inspection confirms that deletion operations occur only in localized regions, which indicates that opportunities for road continuation improvements are more limited than anticipated.

7.1.4. Core challenge: detecting road continuation disagreement

A critical limitation emerged that undermines the entire approach: the inability to detect and measure road continuation disagreement as distinct from road existence disagreement. This challenge appears throughout the research and represents the core weakness preventing decisive answers to the research questions.

Edge coverage cannot distinguish disagreement types.

The edge coverage function cannot distinguish between road continuation disagreement and road existence disagreement. As the Fréchet distance increases between edges, this could indicate either a detour (continuation disagreement) or complete road absence (existence disagreement). Road topology diversity creates uncertainty about whether edge coverage indicates continuation issues or existence disagreements.

TOPO* and APLS* prove inadequate for measuring road continuation disagreement. These methods only detect disconnected graph components through peaks near zero in their distributions, providing weak detection of road continuation disagreement. This weakness has two sources: unclear identification of problematic edges, and the requirement for complete subgraph disconnection. Most road continuation disagreements will have some shortest path existing rather than complete disconnection, creating very high false negative counts.

Impact on research validity.

Since I cannot effectively detect road continuation disagreement, I cannot test whether map fusion actually improves upon it. This limitation extends to other evaluations: distance sampling density plots and ECDF edge coverage do not provide conclusive evidence about SAT's road geometry alignment, because I cannot distinguish between geometry deviation and completely absent roads.

Without reliable methods to assess road continuation correctness, the research becomes indecisive about performance impacts of resolving road continuation. My hypothesis that GPS-based graphs better capture road continuation than SAT graphs cannot be confirmed due to this measurement gap. Qualitative results show entire patch network subnetworks being copied into fused maps, further complicating true fusion effectiveness assessment.

7.1.5. Algorithm limitations with complex road scenarios

The algorithm shows expected behavior in various practical scenarios, yet it struggles with parallel roads and shows confusion in edge replacement when geometry differs significantly. Road resolution causes changes in both road geometry and topology, creating additional complications for the fusion process.

7.2. Answering research questions

Given the limitations in detecting road continuation disagreement, most research questions cannot be answered definitively. However, the experiments provide clear evidence on the practical effectiveness of the proposed fusion approach.

7.2.1. Does the algorithm improve map similarity by resolving road continuation conflicts?

From a technical implementation standpoint, effectiveness depends entirely on context. When the base map performs worse than the patch map and sufficient high-quality patch map edges get injected, the algorithm can improve performance. However, the fusion logic may inject poor-quality patch map components while removing better-performing base map elements.

The core issue is that performance gains result entirely from incorporating edges from whichever map performs better, rather than from addressing road continuation specifically. Since patch map quality is essentially random from the algorithm's perspective, the method doesn't truly target road continuation problems.

From an academic perspective, this research question remains fundamentally unanswered due to measurement limitations. Answering this question properly requires two developments: a technique that isolates road continuation disagreements while ignoring road existence disagreements, and a fusion algorithm that specifically targets road continuation issues without making broader topology changes.

7.2.2. Do the fusion variants outperform both unimodal methods?

The empirical evidence provides a definitive answer: no guaranteed improvement exists. Fusion performance depends on whether the best-performing components from both unimodal maps make it into the final result, which is controlled by the edge coverage threshold.

Applying only the edge injection step produces the best fusion performance, sometimes outperforming the superior unimodal map, but this remains unpredictable and context-dependent. Even if the method had been more effective at addressing road continuation conflicts specifically, I think road conflict resolution alone would not result in sufficient performance gains to outperform the best-performing unimodal map, because resolving road existence disagreements turned out to be far more impactful.

Without reliable methods to assess road inference quality and distinguish between disagreement types, fusion performance will continue to depend largely on chance rather than systematic improvement.

7.2.3. Are GPS graphs more accurate at capturing road continuation and are SAT graphs more accurate at capturing road geometry?

Both questions remain unanswered due to the absence of reliable techniques for measuring continuation agreement and geometric accuracy for edges that agree on road existence. While the theoretical hypotheses remain reasonable, practice often contradicts expectations, as demonstrated by the Berlin GPS map showing superior road geometry inference compared to the SAT map.

7.3. Broader implications and limitations

7.3.1. Data considerations

The GPS data from Berlin and Chicago originates from 2018, while the SAT and OSM data comes from 2024, creating a six-year temporal misalignment. However, the high similarity scores observed suggest that changes in road infrastructure over this period remained negligible for these urban regions.

The peculiarities of overpredicting SAT graphs and underpredicting GPS graphs actually proved helpful for testing the fusion algorithm under more extreme circumstances. Future research would benefit from access to much more GPS data at diverse locations, including rural regions with different vegetation, building styles, and road infrastructure layouts.

7.3.2. Benchmarking limitations

The multi-modal map reconstruction papers did not open source their code, preventing benchmarking against their methods. This limitation makes it difficult to assess how the fusion approach compares to existing state-of-the-art techniques.

7.3.3. Map similarity limitations

The map similarity metrics turned out to be inadequate for maps with different subregions. Both TOPO and APLS assume complete identical road sets, while TOPO* and APLS* only partially account for this by taking valid initial sample seeds into consideration. This limitation hampers deriving useful insights about successfully reconstructed edges and creates uncertainty about the true quality of reconstructed map segments.

7.3.4. Measuring road continuation (dis)agreement

Capturing road coverage agreement and road geometry agreement turned out to be less trivial than expected, and deserve research questions in themselves. All this time the assumed ways of measuring them were unconsciously hypotheses. The fundamental question of how we can measure agreement of road continuation between two maps remains partially unresolved.

8. Conclusion

This master's thesis investigated multi-modal road network reconstruction by developing a late-fusion algorithm that merges GPS and satellite-based reconstructed maps. Current state-of-the-art deep fusion and early fusion methods make it difficult to understand how individual modalities contribute to reconstruction, which limits our ability to combine their complementary strengths effectively.

My research focused on what I call road continuation — disagreements about whether roads connect, which I consider distinct from conflicts over road existence or geometry. Since GPS methods show superior accuracy at capturing road continuation compared to satellite approaches, I developed a late-fusion algorithm to combine GPS-based road continuation strengths with satellite imagery's superior road geometry. The algorithm targets continuation conflicts through injection, deletion, and reconnection operations, with the goal of improving TOPO and APLS scores compared to ground truth.

The experimental results contradicted my initial hypotheses, as map fusion did not systematically improve reconstruction performance over the best-performing unimodal methods. This unexpected outcome revealed important insights about how fusion actually works in practice. The most significant finding was that fusion performance depends on retaining edges from higher-quality source maps rather than sophisticated integration. Duplicating road infrastructure by copying from both sources proved more effective than reducing graph complexity with the risk of deleting well-inferred roads.

A fundamental measurement limitation prevented me from definitively answering the core research questions about road continuation conflict resolution. I could not reliably separate road continuation disagreements from road existence disagreements, which undermined my ability to evaluate whether the algorithm actually addresses road continuation problems. The proposed prime variants of the similarity metrics (TOPO*, APLS*) proved insufficient for this task, only detecting distribution peaks at zero due to disconnected components. Similarly, Frechet distance cannot distinguish between edge continuation and edge existence disagreement. Without reliable methods to measure road continuation specifically, the main research question remains unanswered.

Despite these measurement challenges, this research made several contributions to the field. I constructed the entire map fusion pipeline from data retrieval through inference with unimodal methods to fusing maps and evaluating results. This included building a multi-modal dataset and implementing the complete processing workflow. I introduced the concept of road continuation as a distinct type of map disagreement and developed an initial technique to measure it by computing Frechet distance between edges and graphs. I also created an algorithm to resolve road continuation conflicts through systematic injection, deletion, and reconnection operations.

On the evaluation side, I introduced APLS* and TOPO*, which ignore samples with failed sample seeds. These modified metrics proved valuable for understanding performance differences when maps have varying coverage areas, enabling more meaningful comparisons between dense and sparse reconstructions. More broadly, this research provides important insights showing that while resolving road continuation conflicts may improve map performance, missing road infrastructure and misaligned road geometry have much larger impacts on overall results. The findings demonstrate that understanding why fusion approaches succeed or fail provides valuable insights for the field, even when the proposed method doesn't achieve systematic improvements.

These results indicate that future multi-modal reconstruction efforts should prioritize addressing road existence and geometry alignment before tackling continuation issues. The measurement challenges identified here highlight a critical gap in current approaches for multi-modal reconstruction. Future research should develop methods that can reliably distinguish between different types of map disagreements to enable more targeted fusion strategies. Only with such measurement capabilities can we properly evaluate whether fusion algorithms actually improve map reconstructions in the ways this research attempted to address.

9. Future Work

The map fusion algorithm showed promising results by outperforming both unimodal maps in some scenarios, providing reason to continue this research direction. However, the experiments revealed fundamental challenges that must be addressed before multimodal map fusion can achieve reliable effectiveness.

Improving Road Continuation Detection and Resolution

The most pressing challenge involves developing better methods to detect and resolve road continuation disagreements while distinguishing them from road existence disagreements. Currently, the fusion algorithm relies solely on edge coverage using the Fréchet distance, which cannot differentiate between missing roads and roads with continuation problems.

I think of two potential solutions to increase accuracy when resolving road continuation conflicts. The first approach incorporates the Hausdorff distance in edge coverage computation to identify situations where a low Hausdorff distance occurs alongside a high Fréchet distance. This indicates that one graph took a detour even though it has some edge geometry along a direct route that the other graph has, suggesting a continuation disagreement rather than missing road segments. The second approach implements a bidirectional edge coverage requirement in the map fusion inject step, ensuring that injected edges cover base edges themselves. This bidirectional requirement guarantees that injection only occurs where base map edges exist, preventing resolution of existence disagreements and creating more selective injection within the existing fusion algorithm.

Beyond improving detection capabilities, the fusion algorithm requires enhanced resolution mechanisms that can make roads discontinuating when necessary. Additionally can the current road continuation resolution process be improved by having smaller changes in geometry. For instance by inserting a single missing edge instead of replacing a collection of edges.

Achieving Baseline Performance Guarantees

The challenge of performing at least as well as the best unimodal map remains unsolved. The fusion algorithm currently lacks knowledge about which map or map regions perform better, making edge selection essentially random. The experiments demonstrated that we cannot blindly assume inferred map properties and qualities, as they prove sensitive to context and available data.

A promising approach involves informing the algorithm about edge quality through certainty measures. While maintaining the late-fusion pipeline and treating unimodal models as black-boxes, we can derive certainty by running inference multiple times with input data subsets. For GPS data, this means using different trace subsets, while for satellite images, this could involve scrambling or cropping input images. Multiple graph outputs can then provide uncertainty estimates for each edge.

Alternatively, existing model outputs already contain some certainty information that could be leveraged. Sat2Graph's Graph Tensor Encoding shows confidence levels for vertices and edges. For Roadster, bundle size and local kernel density estimation can indicate edge reliability, with lowest certainty at 3 samples per bundle.

Developing Better Evaluation Methods and Datasets

Current evaluation metrics proved insufficient for understanding road continuation improvements. TOPO* and APLS* have limited ability to measure road continuation disagreement, and both TOPO and APLS struggle with the undirected, unweighted graphs produced by current reconstruction methods.

Future work needs evaluation metrics that can isolate road continuation issues from road existence problems. The evaluation toolkit also requires significant optimization since current implementations run too slowly for larger maps or comprehensive benchmarking. Multi-processing and more efficient algorithms are needed to make evaluation practical at city scale.

This research also highlighted the need for more extensive GPS datasets. I felt constrained by insufficient public GPS data available for training and inference. More GPS data would extend dataset capabilities and provide more robust insights across different urban contexts.

10. Appendix

This material was written after the thesis was assessed and graded. I included it in the week following grading because I gained insights that seemed worth documenting.

The appendix provides a description and impact analysis of a more selective injection step that addresses road continuation resolution more precisely. This procedure ignores resolving road existence disagreements in the original map fusion algorithm while focusing exclusively on road continuation conflicts.

I developed this approach after observing injection-deletion misalignment in threshold experiments (Figure 27) and fusion action visualizations (Section 6.7.2). By adding the criterion that injected patch edges must cover at least one base edge, they can no longer explain road existence disagreements, leaving only road continuation conflicts to resolve.

To describe the selective injection procedure, I introduce a mathematical operator for edge coverage in Appendix A to simplify describing injection and the selective injection logic. This notation enables a precise description of selective injection in Appendix B. I provide experimental results in Appendix C that perform similar experiments as done in Section 6 to demonstrate the performance of selective injection, and due to its precision in targeting road continuation conflicts, this allows answering the research question about the impact of road continuation on map performance.

A Mathematical notation of edge injection

To explain selective edge injection mathematically, I introduce notation for edge coverage that builds on the definition in Section 4.1. The notation uses the subset symbol with a lower subscript indicating the applied threshold.

A curve e covered by a collection of curves $\boldsymbol{C}$ at threshold x is notated as:

$$e \underset{x}{\in} \boldsymbol{C}$$

This notation extends naturally to edges and graphs. An edge e covered by a collection of edges E is notated as:

$$e \underset{x}{\in} E$$

The subset of edges $E_I \subset E_H$ of a graph H covered by a graph G is notated as:

$$E_I \underset{x}{\subset} G$$

B Selective edge injection

Building on the mathematical notation above, this section describes selective edge injection.

The adjustment involves adding a second criterion to the injection process. The original map fusion algorithm injects patch edges that meet one criterion: they are not covered by the base graph. Selective injection adds the requirement that patch edges must also cover at least one base map edge before injection. The motivation for this technique is that when a patch edge covers at least one base map edge, it addresses road continuation rather than road existence disagreements.

Using the notation from Appendix A, I can formalize both injection approaches. Basic injection injects all patch edges ($I \subset E_P$) not covered by the base map:

$$I = \left\{ e \in E_P \mid e \underset{\varepsilon}{\notin} E_B \right\}$$

where E_P represents edges in patch map P and E_B represents edges in base map B.

Selective injection adds the requirement that injected edges must cover at least one base edge:

$$\begin{aligned} I^* &= \left\{ e \in I \mid \left(E'_B \underset{\varepsilon}{\subseteq} e \right) \neq \emptyset \right\} \\ &= \left\{ e \in E_P \mid e \underset{\varepsilon}{\notin} E_B \text{ and } \left(E'_B \underset{\varepsilon}{\subseteq} e \right) \neq \emptyset \right\} \end{aligned}$$

where $E'_B \subset E_B$ represents the set of base edges covered by the patch edge e under consideration for injection.

This formulation demonstrates how selective injection constrains the original injection set I to include only edges with coverage relationships to base map edges.

The algorithm modification is straightforward, requiring only a single filtering step added to the existing injection process. This step filters patch edges that would otherwise be injected, ensuring injected patch edges cover at least one base edge.

C Experimental Results for Selective Injection

This section presents experimental results similar to Section 6 to evaluate selective injection performance across eight comprehensive analyses. The experimental evaluation covers: (C.1) full map visualizations at three thresholds to demonstrate overall selective injection behavior (Appendix C.1), (C.2) detailed regional analysis revealing partial deletion patterns at different scales (Appendix C.2), (C.3) injection-deletion alignment analysis showing improved balance compared to original fusion (Appendix C.3), (C.4) graph complexity preservation indicating better structural integrity maintenance (Appendix C.4), (C.5) map performance metrics using consistent experimental setup (Appendix C.5), (C.6) road continuation quality assessment of unimodal maps (Appendix C.6), (C.7) correlation analysis between continuation quality improvements and map performance gains (Appendix C.7), and (C.8) synthesis of findings regarding selective injection effectiveness (Appendix C.8).

C.1 Full Map Visualizations

This experiment visually shows the impact of the map fusion algorithm. Table 22 and Table 23 present Berlin and Chicago I*DR results across three thresholds: 1m, 30m, and 50m.

The selective injection approach shows improved precision in resolving road continuation disagreements by reducing the number of injections compared to original fusion. The original unimodal map structure remains clearly recognizable because of the substantially fewer injected edges.

Chicago GPS I*DR shows zero injection need at all thresholds because all GPS edges are already continuing, validating the selective injection logic to accurately target road continuation disagreements.

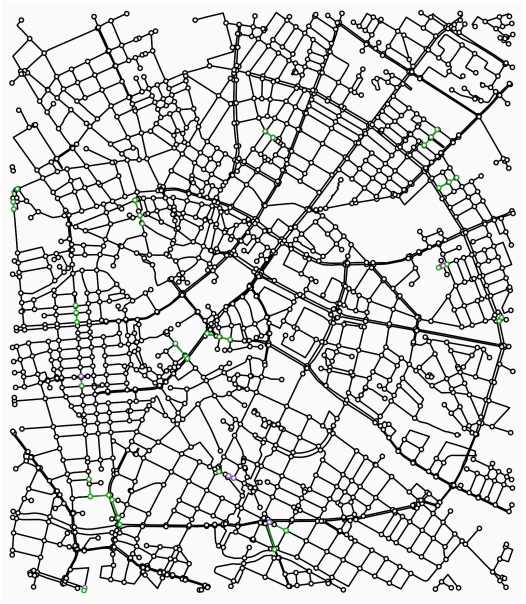
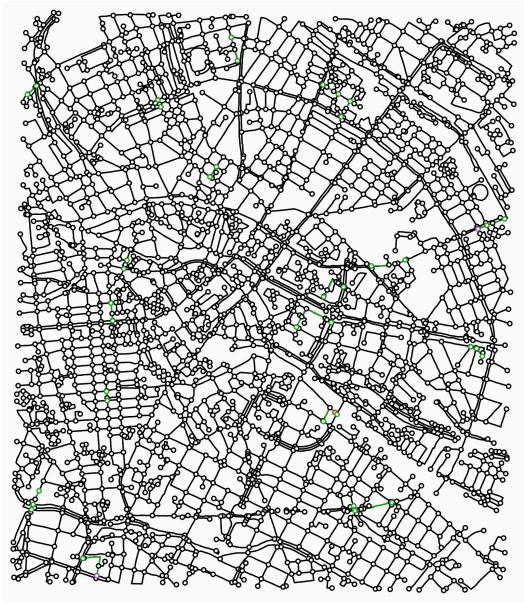
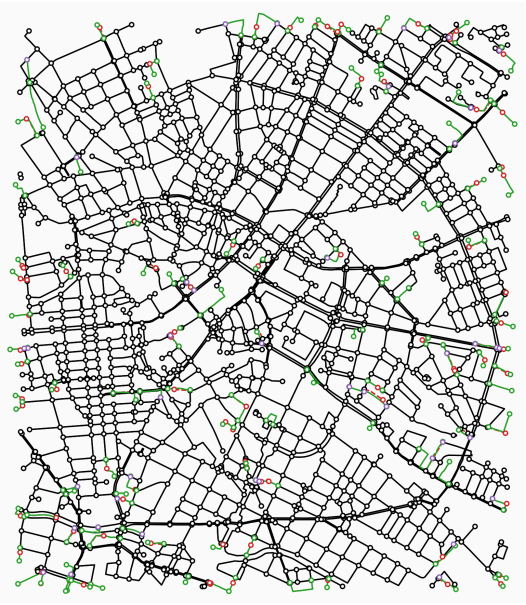
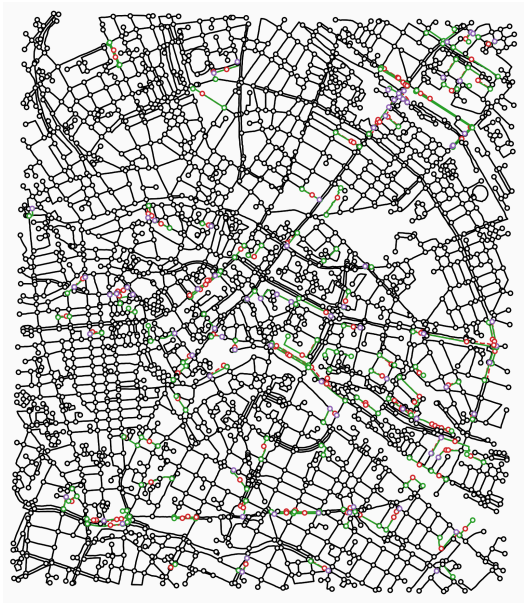
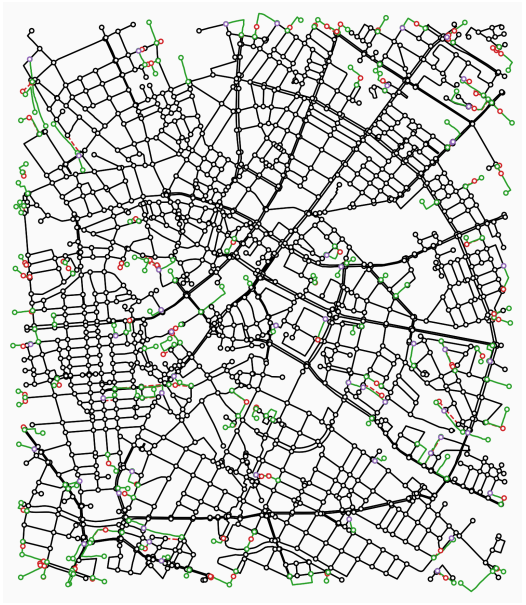
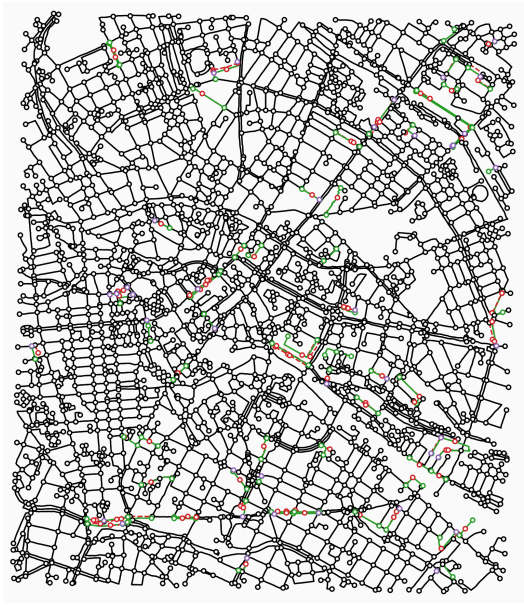
	I*DR$_{\text{GPS}}$	I*DR$_{\text{SAT}}$
1m		
30m		
50m		

Table 22: Berlin fused maps at different map thresholds (1m, 30m, 50m) showing fusion actions through colored nodes and edges: black (unchanged base), green (injected), red dashed (deleted), purple dotted (reconnected).

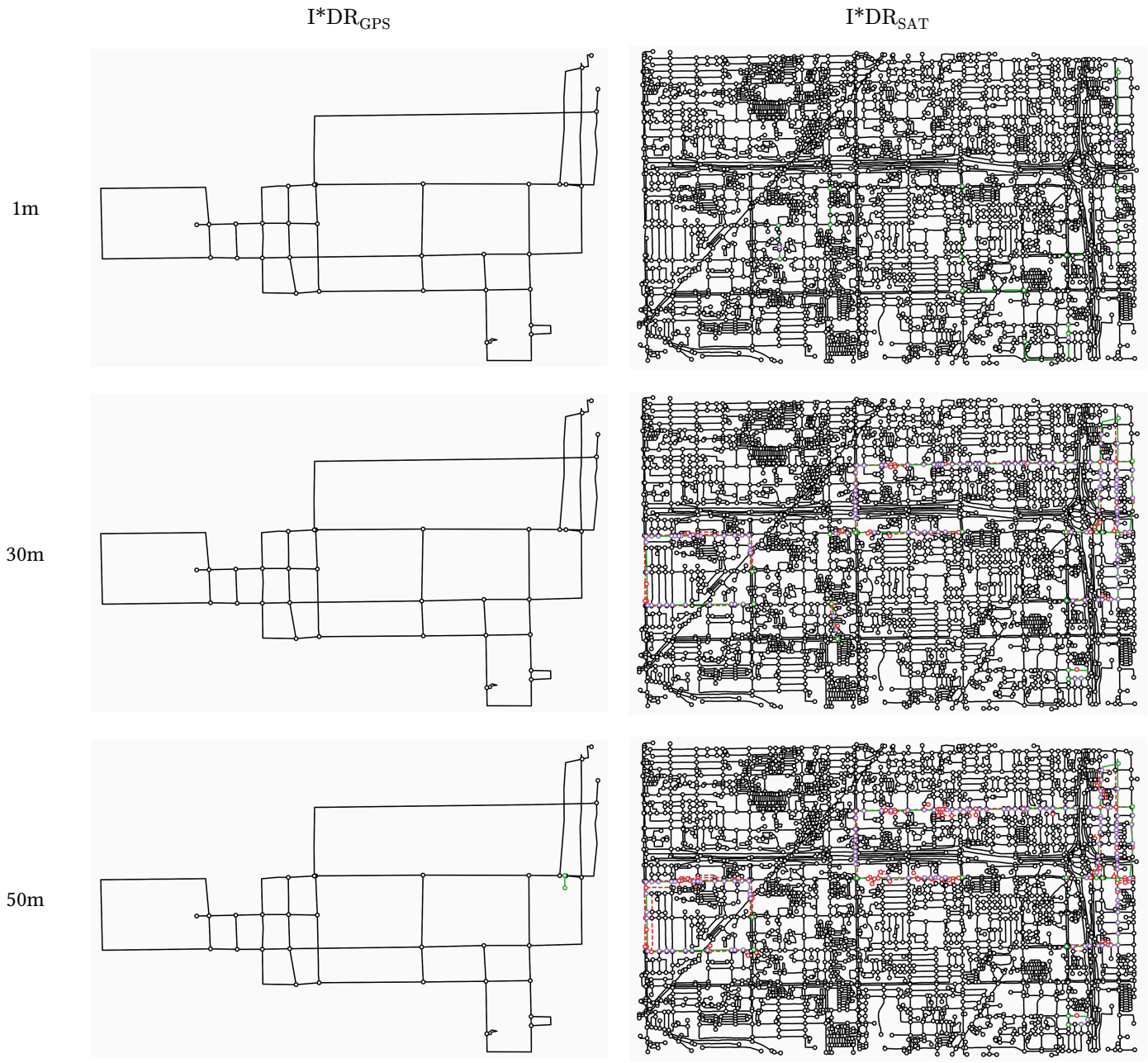


Table 23: Chicago fused maps at thresholds 1m, 30m, 50m with color-coded fusion actions: black (unchanged base), green (injected), red dashed (deleted), purple dotted (reconnected).

C.2 Detailed Regional Analysis

Figure 56 provides detailed examples by zooming in on the reconstructed fused maps.

At the 1m threshold, examination reveals that sometimes only a few of the duplicated base edges get deleted, resulting in a fraction of injected edges undergoing duplicated geometry deletion. This occurs because only some base edges along the entirety of the injected patch edge geometry are covered by the injected patch edge. Other base edges remain beyond the 1m Fréchet distance threshold and therefore stay uncovered and avoid deletion.

The algorithm only checks for *any* covered base edge. I don't think it requires another check to find more duplicated base edges in these cases, because at the 30m threshold road existence disagreement issues become negligible.

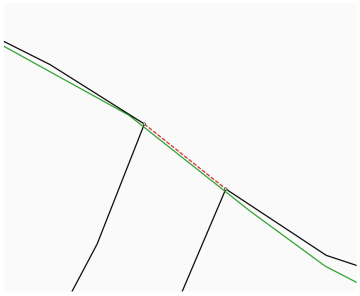


Figure 57: Berlin $\mathbf{I^*DR}_{\text{SAT-1}}$: SAT edge covered by uncovered GPS edge, but adjacent SAT edges remain uncovered, causing partial deletion of duplicated geometry.

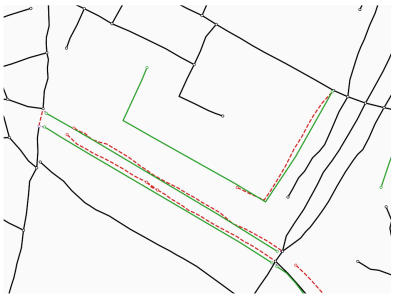


Figure 58: Berlin $\mathbf{I^*DR}_{\text{SAT-30}}$: selective injection maintains good edge continuation resolution.

Figure 59: Berlin $\mathbf{I^*DR}_{\text{SAT-30}}$: selective injection maintains good edge continuation resolution.

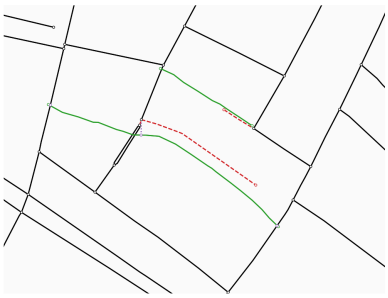


Figure 60: Berlin $\mathbf{I^*DR}_{\text{GPS-30}}$: selective injection maintains good edge continuation resolution.

Figure 61: Chicago $\mathbf{I^*DR}_{\text{SAT-30}}$: identical road continuation conflict resolution as $\mathbf{IDR}_{\text{SAT-30}}$.

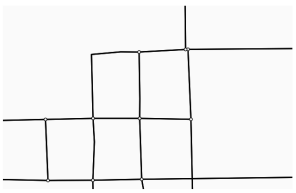


Figure 62: Chicago $\mathbf{I^*DR}_{\text{GPS-30}}$: no fusion activity as expected.

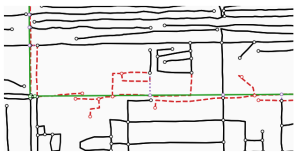


Figure 63: Chicago $\mathbf{I^*DR}_{\text{SAT-50}}$: larger threshold causes more adjacent road deletions.

Figure 56: Detailed views of Berlin and Chicago selective injection at different thresholds.

C.3 Map Fusion Action at different thresholds

Figure 64 shows the map fusion actions at different thresholds. The alignment between injection and deletion counts shows substantial improvement over the original approach (compare with Figure 27) where injections were significantly higher.

The number of edges injected as threshold increases follows a more complex pattern rather than simply decreasing (fewer uncovered patch edges) or increasing (more uncovered patch edges cover base edges). This probably reflects a balance between an increase of patch edges getting covered and patch edges that cover base edges. On one hand, as we increase the threshold more patch edges get covered (and therefore will not meet the first criterion to get injected), yet for those that do still remain uncovered there is an increased chance of covering a base edge themselves.

Note the flat lines at Chicago with GPS as base: no road continuation conflicts to resolve because all GPS edges are continuing.

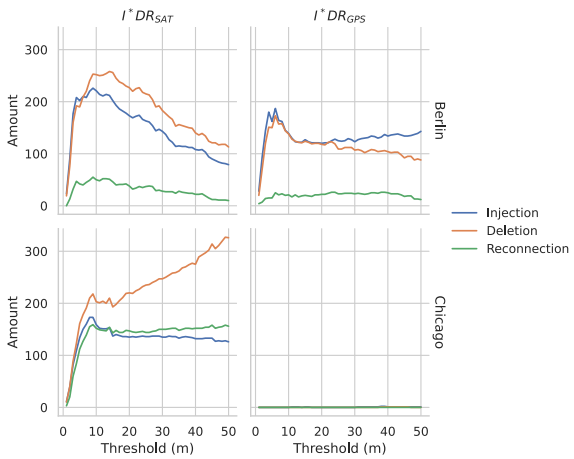


Figure 64: Selective injection map fusion actions at different thresholds.

C.4 Map Fusion Graph Complexity

Figure 65 shows that fused map graph complexity aligns much better with unimodal maps, preserving structural properties during fusion.

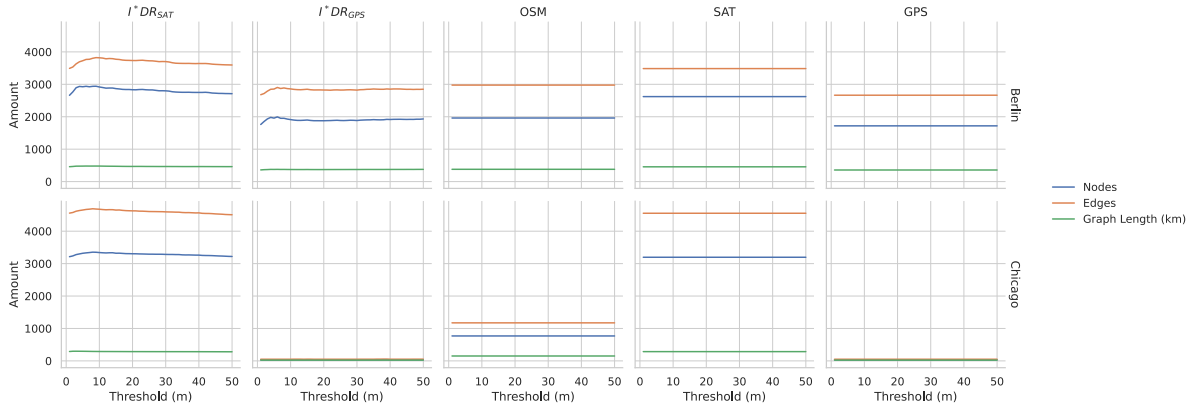


Figure 65: Graph complexity of selective injection at different thresholds.

C.5 Map Performance

Map performance for Berlin and Chicago appears in Table 24 and Table 25 respectively. The experimental setup is the same as Section 6.2 but uses a 10m seed sampling distance to reduce instability as observed in the TOPO sampling hole size experiment (Figure 33) and the graph sampling densities that are sometimes high at a distance of 5m (Table 16).

The results maintain the same pattern as with the map performance results with normal injection: performing only injection proves most effective, with TOPO* as an exception.

Berlin $\mathbf{I*D}_{\mathrm{SAT}}$ shows notable TOPO and TOPO* increases with deletion without reconnection. Given the large sampling count, this likely isn't statistical noise, though I'm uncertain why this happens.

The fused map similarity values align much better with their base unimodal maps, providing stronger evidence that resolving road continuation alone cannot guarantee performance equal to the best unimodal method.

		Rec	Prec	TOPO	APLS	Rec*	Prec*	TOPO*	APLS*
Berlin	**SAT**	0.409	0.604	0.488	0.718	0.679	0.590	0.632	0.879
	GPS	0.602	0.864	0.710	0.784	0.831	0.861	0.846	0.948
	I*$_{SAT}$	0.454	0.597	0.516	0.777	0.695	0.592	0.639	0.920
	I*D$_{SAT}$	0.449	0.618	0.520	0.748	0.686	0.618	0.650	0.886
	I*DR$_{SAT}$	0.434	0.612	0.508	0.750	0.694	0.615	0.652	0.897
	I*$_{GPS}$	0.613	0.823	0.703	0.802	0.829	0.830	0.829	0.948
	I*D$_{GPS}$	0.578	0.857	0.690	0.777	0.809	0.851	0.830	0.928
	I*DR$_{GPS}$	0.582	0.846	0.690	0.790	0.792	0.858	0.824	0.938

Table 24: Berlin map performance with selective injection at 30m threshold, using a map similarity sample seed distance of 10m.

		Rec	Prec	TOPO	APLS	Rec*	Prec*	TOPO*	APLS*
Chicago	**SAT**	0.663	0.293	0.407	0.792	0.858	0.300	0.444	0.861
	GPS	0.003	0.658	0.006	0.024	0.087	0.670	0.155	0.628
	I*$_{SAT}$	0.653	0.296	0.407	0.794	0.850	0.297	0.440	0.858
	I*D$_{SAT}$	0.577	0.290	0.386	0.671	0.770	0.293	0.425	0.757
	I*DR$_{SAT}$	0.624	0.293	0.399	0.784	0.830	0.293	0.433	0.882
	I*$_{GPS}$	0.003	0.679	0.006	0.022	0.081	0.677	0.144	0.633
	I*D$_{GPS}$	0.003	0.676	0.006	0.024	0.084	0.667	0.150	0.655
	I*DR$_{GPS}$	0.003	0.676	0.005	0.023	0.087	0.667	0.154	0.637

Table 25: Chicago map performance with selective injection at 30m threshold, using a map similarity sample seed distance of 10m.

C.6 Road Continuation Quality of Unimodal Maps

Now that road continuation is addressed more specifically, I can test whether SAT or GPS graphs perform better at road continuation. I perform map fusion using each unimodal map as base and patch it with ground truth to measure how often ground truth disagrees with the unimodal map about road continuation. Results appear in Table 26.

Road continuation conflicts are measured by counting edge injections, where each injection indicates that ground truth continuation differs from the unimodal map. I normalize by total edge length since graphs have different edge densities.

When using the unimodal map as base, edge injection counts measure incorrectly discontinuing edges in the unimodal maps. When using ground truth as base, edge injection counts measure incorrectly continuing edges in the unimodal map.

Place	Base	Patch	Injected Edges	Total Edges	Total Length (km)	Inj./Tot. Edges	Inj./Tot. Length
Berlin	**SAT**	**OSM**	143	3907	488.27	0.0366	0.29
Chicago	**SAT**	**OSM**	99	4832	303.02	0.0205	0.33
Berlin	**OSM**	**SAT**	169	3383	412.04	0.0500	0.41
Chicago	**OSM**	**SAT**	50	1297	156.30	0.0386	0.32
Berlin	**GPS**	**OSM**	58	2790	370.66	0.0208	0.16
Chicago	**GPS**	**OSM**	1	54	23.66	0.0185	0.04
Berlin	**OSM**	**GPS**	26	3049	384.10	0.0085	0.07
Chicago	**OSM**	**GPS**	6	1188	152.03	0.0051	0.04

Table 26: Unimodal fusion analysis results against ground truth at a threshold 30m.

Taking these results at face value, GPS demonstrates superior road continuation quality compared to SAT. For Berlin, GPS has 2.1% incorrectly discontinuing edges while SAT has 3.7%, and GPS has 0.9% incorrectly continuing edges while SAT has 5.0%.

For Chicago, both unimodal maps show around 2.0% incorrectly discontinuing edges, but GPS has only 0.5% incorrectly continuing edges compared to SAT's 3.9%.

When examining injections per total edge length, GPS outperforms SAT by significant margins.

These results demonstrate that GPS map reconstruction achieves significantly better road continuation quality compared to SAT.

We can apply the same analysis to examine the road continuation quality of fused maps. Results appear in Table 27.

Place	Base	Patch	Injected Edges	Total Edges	Total Length (km)	Inj./Tot. Edges	Inj./Tot. Length
Berlin	$\mathbf{I^*DR_{SAT}}$	**OSM**	53	3838	475.13	0.0138	0.11
Chicago	$\mathbf{I^*DR_{SAT}}$	**OSM**	97	4869	301.42	0.0199	0.32
Berlin	**OSM**	$\mathbf{I^*DR_{SAT}}$	167	3374	411.59	0.0495	0.41
Chicago	**OSM**	$\mathbf{I^*DR_{SAT}}$	49	1296	156.25	0.0378	0.31
Berlin	$\mathbf{I^*DR_{GPS}}$	**OSM**	41	2913	382.65	0.0141	0.11
Chicago	$\mathbf{I^*DR_{GPS}}$	**OSM**	1	54	23.66	0.0185	0.04
Berlin	**OSM**	$\mathbf{I^*DR_{GPS}}$	94	3206	398.83	0.0293	0.24
Chicago	**OSM**	$\mathbf{I^*DR_{GPS}}$	6	1188	152.03	0.0051	0.04

Table 27: Selective injection (I*DR) fusion analysis results (threshold = 30m).

Comparing the fused maps to their unimodal counterparts shows that the fused maps demonstrate improved road continuation quality, with fewer incorrectly continuing and discontinuing edges relative to their total edge counts.

C.7 Relating Road Continuation Quality to Map Similarity Performance

The most pressing question arising from the ability to better target road continuation conflicts is whether resolving road continuation disagreements actually improves map similarity performance. We can examine this by comparing map performance tables from Appendix C.5 with road continuation improvements in Appendix C.6 to identify positive correlations.

Figure 66 shows a correlation heat map between changes in road continuation quality and changes in map similarity when comparing fused maps to their base unimodal maps.

I define road continuation quality as the percentage of correctly continuing edges relative to the total number of edges in the fused map. Incorrectly continuing edges are identified as the selectively injected edges when we fuse ground truth as patch against the fused map, making all remaining fused edges correct by definition.

Similarly, road discontinuation quality represents the percentage of correctly discontinuing edges, calculated by subtracting incorrectly discontinuing edges from total edges in the fused map. Incorrectly discontinuing edges are identified as injected edges when we apply map fusion with the fused map as patch against ground truth as base.

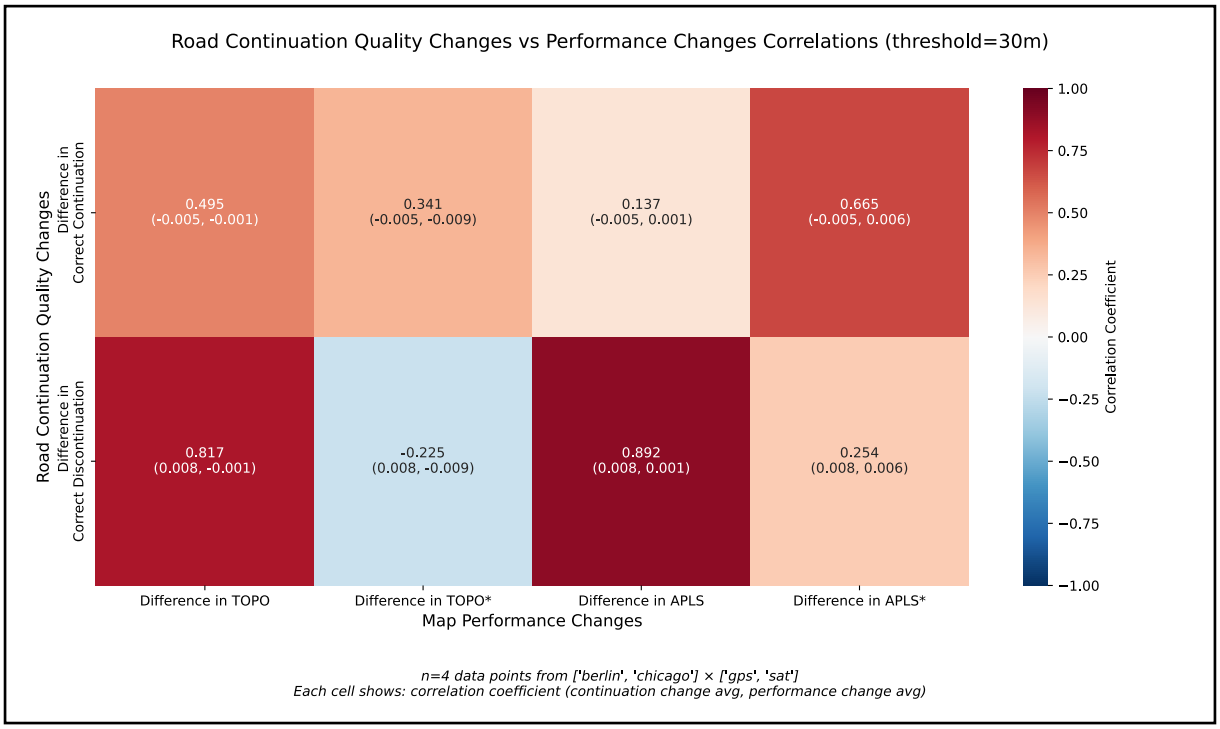


Figure 66: Road continuation quality changes vs. map performance changes along fused map to unimodal base map.

The correlation matrix reveals a positive correlation between changes in road continuation quality and map performance improvements. The only exception occurs with TOPO* regarding road discontinuation, where the fused map shows improved discontinuation quality on average yet the TOPO* score still decreases.

C.8 Concluding Remarks

These results demonstrate selective injection successfully addresses over-injection of road existence disagreements while resolving road continuation disagreement.

Some research question answers remain unchanged: performance stays sensitive to base map choice, which cannot be determined beforehand, so the map fusion algorithm cannot confidently outperform both unimodal maps. However, we do see that resolving road continuation tends to improve map reconstruction by applying the injection step. The same behavior of injection-only approaches achieving optimal performance continues to hold.

We can now answer that GPS maps prove better at road continuation reconstruction compared to the respective SAT maps for this dataset (in combination with the choice of unimodal map reconstruction techniques) per Appendix C.6. And that this dataset provides a positive correlation between improvements in road continuation quality and improvements in map performance, suggesting that road continuation improvements improve road map reconstruction performance.

Bibliography

1 Buchin, K., Buchin, M., Gudmundsson, J., *et al.*: "Improved Map Construction using Subtrajectory Clustering", in "Proceedings of the 4th ACM SIGSPATIAL Workshop on Location-Based Recommendations, Geosocial Networks, and Geoadvertising" SIGSPATIAL '20: 28th International Conference on Advances in Geographic Information Systems, (ACM, 2020), pp. 1–4

2 He, S., Bastani, F., Jagwani, S., *et al.*: "Sat2Graph: Road Graph Extraction Through Graph-Tensor Encoding" (Springer International Publishing, 2020), 12369, pp. 51–67

3 Alt, H., Godau, M.: "Computing the Fréchet Distance between Two Polygonal Curves"*International Journal of Computational Geometry & Applications*, 1995, **5**, (1), pp. 75–91.

4 Urhausen, J.E.: "On Geometric Measures and Their Computation". Utrecht University, 2023

5 Eiter, T., Mannila, H.: "Computing Discrete Fréchet Distance", in (1994)

6 Biagioni, J., Eriksson, J.: "Inferring Road Maps from Global Positioning System Traces: Survey and Comparative Evaluation"*Transportation Research Record: Journal of the Transportation Research Board*, 2012, **2291**, (1), pp. 61–71.

7 Van Etten, A., Lindenbaum, D., Bacastow, T.M.: "SpaceNet: A Remote Sensing Dataset and Challenge Series", http://arxiv.org/abs/1807.01232, accessed January 2024

8 Chao, P., Xu, Y., Hua, W., Zhou, X.: "A Survey on Map-Matching Algorithms", http://arxiv.org/abs/1910.13065, accessed April 2025

9 Jiang, L., Chen, C., Chen, C., Huang, H., Guo, B.: "From driving trajectories to driving paths: a survey on map-matching Algorithms"*CCF Transactions on Pervasive Computing and Interaction*, 2022, **4**, pp. 252–267.

10 Jin, Z., Kim, J., Yeo, H., Choi, S.: "Transformer-based Map Matching Model with Limited Ground-Truth Data using Transfer-Learning Approach"*arXiv preprint*, 2021.

11 Mohammadi, S., Smyth, A.W.: "NLP-enabled Trajectory Map-matching in Urban Road Networks using a Transformer-based Encoder-decoder"*arXiv preprint*, 2024.

12 Wang, Y., Liu, X., Wei, H., Forman, G., Chen, C., Zhu, Y.: "CrowdAtlas: self-updating maps for cloud and personal use", in "Proceeding of the 11th annual international conference on Mobile systems, applications, and services" MobiSys'13: The 11th Annual International Conference on Mobile Systems, Applications, and Services, (ACM, 2013), pp. 27–40

13 He, E., Bai, F., Hay, C., Chen, J., Bhagavatula, V.: "A Map Inference Approach Using Signal Processing from Crowd-sourced GPS Data"*ACM Transactions on Spatial Algorithms and Systems*, 2021, **7**, (2), pp. 1–23.

14 Davies, J., Beresford, A., Hopper, A.: "Scalable, Distributed, Real-Time Map Generation"*IEEE Pervasive Computing*, 2006, **5**, (4), pp. 47–54.

15 Shi, W., Shen, S., Liu, Y.: "Automatic Generation of Road Network Map from Massive GPS Vehicle Trajectories", in "IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC" (2009), pp. 1–6

16 Biagioni, J., Eriksson, J.: "Map inference in the face of noise and disparity", in "Proceedings of the 20th International Conference on Advances in Geographic Information Systems" SIGSPATIAL'12: SIGSPATIAL 2012 International Conference on Advances in Geographic Information Systems, (ACM, 2012), pp. 79–88

17 Eftelioglu, E., Garg, R., Kango, V., Gohil, C., Chowdhury, A.R.: "RING-Net: road inference from GPS trajectories using a deep segmentation network", in "Proceedings of the 10th ACM SIGSPATIAL International Workshop on Analytics for Big Geospatial Data" SIGSPATIAL '22: The 30th International Conference on Advances in Geographic Information Systems, (ACM, 2022), pp. 17–26

18 Edelkamp, S., Schrödl, S.: "Route Planning and Map Inference with Global Positioning Traces", in "Computer Science in Perspective" (2003)

19 Cao, L., Krumm, J.: "From GPS traces to a routable road map", in "Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems" GIS '09: 17th SIGSPATIAL International Conference on Advances in Geographic Information Systems, (ACM, 2009), pp. 3–12

20 Karagiorgou, S., Pfoser, D.: "On vehicle tracking data-based road network generation", in "Proceedings of the 20th International Conference on Advances in Geographic Information Systems"

SIGSPATIAL'12: SIGSPATIAL 2012 International Conference on Advances in Geographic Information Systems, (ACM, 2012), pp. 89–98

21 Li, H., Kulik, L., Ramamohanarao, K.: "Automatic Generation and Validation of Road Maps from GPS Trajectory Data Sets", in "Proceedings of the 25th ACM International on Conference on Information and Knowledge Management" CIKM'16: ACM Conference on Information and Knowledge Management, (ACM, 2016), pp. 1523–1532

22 Buchin, K., Buchin, M., Duran, D., *et al.*: "Clustering Trajectories for Map Construction", in "Proceedings of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems" SIGSPATIAL'17: 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, (ACM, 2017), pp. 1–10

23 He, S., Bastani, F., Abbar, S., *et al.*: "RoadRunner: improving the precision of road network inference from GPS trajectories", in "Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems" SIGSPATIAL '18: 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, (ACM, 2018), pp. 3–12

24 Chen, J., Jin, W., Chen, S., Jiang, H.: "Cycling Map Inference Using Global Positioning System Trajectories: A Case Study in the Online Food Delivery Business"*Transportation Research Record: Journal of the Transportation Research Board*, 2023, **2677**, (2), pp. 1013–1026.

25 Buchin, K., Buchin, M., Gudmundsson, J., Löffler, M., Luo, J.: "Detecting Commuting Patterns by Clustering Subtrajectories" (Springer Berlin Heidelberg, 2008), 5369, pp. 644–655

26 Duran, D., Sacristán, V., Silveira, R.I.: "Map construction algorithms: an evaluation through hiking data", in "Proceedings of the 5th ACM SIGSPATIAL International Workshop on Mobile Geographic Information Systems" (Association for Computing Machinery, 2016), pp. 74–83

27 Máttyus, G., Luo, W., Urtasun, R.: "DeepRoadMapper: Extracting Road Topology from Aerial Images", in "2017 IEEE International Conference on Computer Vision (ICCV)" (2017), pp. 3458–3466

28 Batra, A., Singh, S., Pang, G., Basu, S., Jawahar, C., Paluri, M.: "Improved Road Connectivity by Joint Learning of Orientation and Segmentation", in "2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)" (2019), pp. 10377–10385

29 Xie, S., Zheng, W., Xian, Z., Yang, J., Zhang, C., Wu, M.: "PaRK-Detect: Towards Efficient Multi-Task Satellite Imagery Road Extraction via Patch-Wise Keypoints Detection", http://arxiv.org/abs/2302.13263, accessed February 2024

30 Hetang, C., Xue, H., Le, C., Yue, T., Wang, W., He, Y.: "SAMRoad: Segment Anything Model for Road Network Graph Extraction", http://arxiv.org/abs/2403.16051, accessed October 2024

31 Bastani, F., He, S., Abbar, S., *et al.*: "RoadTracer: Automatic Extraction of Road Networks from Aerial Images", http://arxiv.org/abs/1802.03680, accessed October 2024

32 Tan, Y.-Q., Gao, S.-H., Li, X.-Y., Cheng, M.-M., Ren, B.: "VecRoad: Point-Based Iterative Graph Exploration for Road Graphs Extraction", in "2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)" (2020), pp. 8907–8915

33 Xu, Z., Liu, Y., Gan, L., *et al.*: "RNGDet: Road Network Graph Detection by Transformer in Aerial Images"*IEEE Transactions on Geoscience and Remote Sensing*, 2022, **60**, pp. 1–12.

34 Xu, Z., Liu, Y., Sun, Y., Liu, M., Wang, L.: "RNGDet++: Road Network Graph Detection by Transformer with Instance Segmentation and Multi-scale Features Enhancement", http://arxiv.org/abs/2209.10150, accessed February 2024

35 Kirillov, A., Mintun, E., Ravi, N., *et al.*: "Segment Anything", http://arxiv.org/abs/2304.02643, accessed November 2024

36 Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S.: "End-to-End Object Detection with Transformers", http://arxiv.org/abs/2005.12872, accessed April 2024

37 He, K., Zhang, X., Ren, S., Sun, J.: "Deep Residual Learning for Image Recognition", http://arxiv.org/abs/1512.03385, accessed April 2024

38 Sun, T., Di, Z., Che, P., Liu, C., Wang, Y.: "Leveraging Crowdsourced GPS Data for Road Extraction from Aerial Imagery", http://arxiv.org/abs/1905.01447, accessed April 2024

39 Wu, H., Zhang, H., Zhang, X., Sun, W., Zheng, B., Jiang, Y.: "DeepDualMapper: A Gated Fusion Network for Automatic Map Extraction using Aerial Images and Trajectories", http://arxiv.org/abs/2002.06832, accessed April 2024

40 Yang, J., Ye, X., Wu, B., *et al.*: "DuARE: Automatic Road Extraction with Aerial Images and Trajectory Data at Baidu Maps", in "Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining" KDD '22: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, (ACM, 2022), pp. 4321–4331

41 Liu, L., Yang, Z., Li, G., Wang, K., Chen, T., Lin, L.: "Aerial Images Meet Crowdsourced Trajectories: A New Approach to Robust Road Extraction", http://arxiv.org/abs/2111.15119, accessed April 2024

42 Yuan, H., Wang, S., Bao, Z., Wang, S.: "Automatic Road Extraction with Multi-Source Data Revisited: Completeness, Smoothness and Discrimination" *Proceedings of the VLDB Endowment*, 2023, **16**, (11), pp. 3004–3017.

43 Wang, S., Wang, Z., Ruan, S., *et al.*: "DelvMap: Completing Residential Roads in Maps Based on Couriers Trajectories and Satellite Imagery" *IEEE Transactions on Geoscience and Remote Sensing*, 2024, **62**, pp. 1–14.

44 Ehrig, M., Jérôme, E.: "Relaxed Precision and Recall for Ontology Matching", in Ashpole, B., Ehrig, M., Euzenat, J., Stuckenschmidt, H. (Eds.): "Proceedings of the Workshop on Integrating Ontologies" (2005), p. 8

45 Gudmundsson, J., Seybold, M.P., Wong, S.: "Map matching queries on realistic input graphs under the Fr\'echet distance", http://arxiv.org/abs/2211.02951, accessed December 2023

46 Boeing, G.: "OSMnx: New methods for acquiring, constructing, analyzing, and visualizing complex street networks" *Computers, Environment and Urban Systems*, 2017, **65**, pp. 126–139.

47 Beckmann, N., Kriegel, H.-P., Schneider, R., Seeger, B.: "The R*-tree: An Efficient and Robust Access Method for Points and Rectangles", in "Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data" (ACM, 1990), pp. 322–331

48 Toblerity: "rtree: Python ctypes bindings for libspatialindex", https://github.com/Toblerity/rtree

49 Hadjieleftheriou, M.: "libspatialindex: C++ implementation of R*-tree, an MVR-tree and a TPR-tree", https://github.com/libspatialindex/libspatialindex

50 Guttman, A.: "R-trees: A Dynamic Index Structure for Spatial Searching", in "Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data" (ACM, 1984), pp. 47–57

51 Bieniek, T.: "utm: Bidirectional UTM-WGS84 converter for Python", https://github.com/Turbo87/utm

52 Ahmed, M., Karagiorgou, S., Pfoser, D., Wenk, C.: "A Comparison and Evaluation of Map Construction Algorithms" *GeoInformatica*, 2015, **19**, (3), pp. 601–632.

53 Lourakis, M.: "egsa87: Greek Geodetic Reference System EGSA'87 transformations", https://github.com/mlourakis/egsa87

54 Boeing, G.: "Topological Graph Simplification Solutions to the Street Intersection Miscount Problem" *Transactions in GIS*, 2025, **29**, (3), p. e70037.

55 Alt, H., Efrat, A., Rote, G., Wenk, C.: "Matching planar maps" *Journal of Algorithms*, 2003, **49**, (2), pp. 262–283.

56 Maheshwari, A., Sack, J.-R., Shahbaz, K., Zarrabi-Zadeh, H.: "Improved Algorithms for Partial Curve Matching" *Algorithmica*, 2014, **69**, (3), pp. 641–657.

57 He, S.: "Sat2Graph Topology Metrics", https://github.com/songtaohe/Sat2Graph/tree/master/metrics/topo

58 He, S.: "Sat2Graph Topology Metrics", https://github.com/songtaohe/Sat2Graph/tree/master/metrics/apls

59 Hosseini, E.: "Roadster: Road Network Extraction from Satellite Images", https://github.com/Erfanh1995/Roadster

60 He, Y., Garg, R., Chowdhury, A.R.: "TD-Road: Top-Down Road Network Extraction with Holistic Graph Construction" (Springer Nature Switzerland, 2022), 13669, pp. 562–577

61 He, S.: "Sat2Graph Github Repository", https://github.com/songtaohe/Sat2Graph