Learning Human-Like Route Generation from 3D Point Cloud Data Using Hidden Markov Models: Exploration of Efficiency-Diversity Trade-off

Caleb Traxler
CS 275P Final Project
Machine Learning with Generative Models
University of California, Irvine

June 13, 2025

Abstract

This project explores whether Hidden Markov Models can learn human routing preferences directly from 3D point cloud data and authentic driving trajectories. Working with seven urban sequences from the KITTI-360 dataset covering 11.67 square kilometers, I developed a complete pipeline that extracts road networks from 3D semantic point clouds and trains HMMs to generate human-like routes. My central discovery reveals a fundamental trade-off between mathematical efficiency and behavioral diversity: while shortest-path algorithms achieve near-perfect efficiency scores (0.932 ± 0.109), my HMM-based approaches produce routes with substantially lower efficiency (0.342 ± 0.142) but dramatically higher diversity scores (0.775 ± 0.103 compared to 0.193 ± 0.217 for baselines). This represents a $4.02\times$ improvement in route variability, demonstrating that probabilistic models can capture essential aspects of human navigation behavior that pure optimization approaches systematically miss. The results provide quantitative evidence that realistic autonomous navigation requires modeling observed human behavior patterns rather than relying solely on geometric optimization.

1 Introduction and Mathematical Motivation

Human drivers consistently choose routes that deviate from mathematical optimality, preferring paths that feel natural and comfortable over purely efficient alternatives. This observation leads to a fundamental question: can probabilistic machine learning capture the subtle patterns that distinguish human-like navigation from purely algorithmic route planning?

Traditional navigation systems formulate routing as a shortest path problem on weighted graph $G = (V, E)$, seeking paths $\pi = (v_1, v_2, \ldots, v_n)$ that minimize cost functions:

$$\pi^* = \arg\min_{\pi} \sum_{i=1}^{n-1} w(v_i, v_{i+1}) \tag{1}$$

This formulation assumes human preferences can be captured by scalar cost functions, which

my research suggests may be fundamentally inadequate for modeling realistic navigation behavior.

My probabilistic approach reformulates route generation as a sequential decision process where Hidden Markov Models learn probability distributions over routing decisions. The mathematical structure treats route generation as a temporal sequence where hidden states correspond to road segments and observations encode geometric features influencing routing decisions.

2 Mathematical Foundation

The Hidden Markov Model framework treats route generation as a temporal sequence with the following mathematical structure:

$$z_t \in \{1, 2, \ldots, K\} \quad \text{(hidden states - road segments)} \tag{2}$$

$$x_t \in \mathbb{R}^d \quad \text{(observations - geometric features)} \tag{3}$$

$$A_{ij} = p(z_t = j | z_{t-1} = i) \quad \text{(transition probabilities)} \tag{4}$$

$$\mu_j, \Sigma_j \sim p(x_t | z_t = j) \quad \text{(emission parameters)} \tag{5}$$

The joint probability over complete sequences factors according to the Markov assumption:

$$p(z_{1:T}, x_{1:T}) = \pi_{z_1} \prod_{t=2}^{T} A_{z_{t-1}, z_t} \prod_{t=1}^{T} \mathcal{N}(x_t; \mu_{z_t}, \Sigma_{z_t}) \tag{6}$$

where $\pi_j = p(z_1 = j)$ represents initial state probabilities and $\mathcal{N}(x_t; \mu_j, \Sigma_j)$ denotes the multivariate Gaussian emission distribution for state j.

2.1 Parameter Learning via Baum-Welch Algorithm

The learning problem involves estimating parameters $\theta = \{\pi, A, \mu, \Sigma\}$ from observed trajectory sequences. Following the Expectation-Maximization framework, the Baum-Welch algorithm iteratively maximizes the log-likelihood:

$$\mathcal{L}(\theta) = \sum_{n=1}^{N} \log p(x_{1:T_n}^{(n)} | \theta) \tag{7}$$

The E-step computes posterior probabilities using the forward-backward algorithm:

$$\alpha_t(j) = p(x_{1:t}, z_t = j | \theta) \tag{8}$$

$$\beta_t(j) = p(x_{t+1:T} | z_t = j, \theta) \tag{9}$$

$$\gamma_t(j) = p(z_t = j | x_{1:T}, \theta) = \frac{\alpha_t(j)\beta_t(j)}{\sum_{k=1}^{K} \alpha_t(k)\beta_t(k)} \tag{10}$$

The M-step updates parameters using computed posterior probabilities:

$$\hat{A}_{ij} = \frac{\sum_{n=1}^{N} \sum_{t=1}^{T_n-1} \xi_t^{(n)}(i,j)}{\sum_{n=1}^{N} \sum_{t=1}^{T_n-1} \gamma_t^{(n)}(i)} \tag{11}$$

$$\hat{\mu}_j = \frac{\sum_{n=1}^{N} \sum_{t=1}^{T_n} \gamma_t^{(n)}(j) x_t^{(n)}}{\sum_{n=1}^{N} \sum_{t=1}^{T_n} \gamma_t^{(n)}(j)} \tag{12}$$

3 Data Processing and Feature Engineering

The KITTI-360 dataset provides synchronized 3D point clouds and vehicle trajectories from real urban driving. I processed seven sequences covering diverse urban environments, extracting road networks through adaptive height-based filtering and geometric segmentation.

Road surface identification uses adaptive height filtering with mathematical formulation:

$$R = \{p_i \in P : h_{base}(p_i) - 0.5 \leq z_i \leq h_{base}(p_i) + \tau\} \tag{13}$$

where $h_{base}(p_i)$ represents the local baseline height computed as the 15th percentile within spatial neighborhoods, and τ is the height tolerance parameter.

Geometric feature extraction employs Principal Component Analysis for road width estimation:

$$C = \frac{1}{n} \sum_{i=1}^{n} (p_i - \bar{p})(p_i - \bar{p})^T \tag{14}$$

$$\text{width} = 2.5 \times \sqrt{\lambda_{min}} \quad \text{(clamped to [2.5, 12.0] meters)} \tag{15}$$

Curvature estimation uses a three-point circumcircle method:

$$\kappa = \frac{4 \times \text{Area}(\triangle p_1 p_2 p_3)}{|p_1 p_2| \times |p_2 p_3| \times |p_1 p_3|} \tag{16}$$

Complete processing results are presented in Appendix A with comprehensive visualizations.

4 Route Generation and Inference Methods

I implemented multiple inference approaches demonstrating different aspects of probabilistic reasoning. Viterbi decoding finds the most probable sequence through dynamic programming:

$$\delta_t(j) = \max_i [\delta_{t-1}(i) + \log A_{ij} + \log \mathcal{N}(x_t; \mu_j, \Sigma_j)] \tag{17}$$

$$\psi_t(j) = \arg\max_i [\delta_{t-1}(i) + \log A_{ij} + \log \mathcal{N}(x_t; \mu_j, \Sigma_j)] \tag{18}$$

Forward sampling generates alternative routes by probabilistically sampling from learned transition distributions:

$$z_1 = \text{start\_state} \tag{19}$$

$$z_{t+1} \sim p(z_{t+1}|z_t) = \text{Categorical}(A_{z_t,:}) \tag{20}$$

To encourage convergence toward desired end states, I implement bias mechanisms:

$$\tilde{A}_{z_t,j} = \begin{cases} A_{z_t,j} \times (1 + \beta \cdot \frac{t}{T_{max}}) & \text{if } j = \text{end\_state} \\ A_{z_t,j} & \text{otherwise} \end{cases} \tag{21}$$

Baseline methods include shortest path algorithms, width-biased routing, and curvature-minimizing approaches for comprehensive comparison.

5 Experimental Results and Analysis

5.1 Dataset Processing Results

The road network extraction pipeline processed seven urban sequences with the following characteristics:

Table 1: Road Network Extraction Results

Map Sequence	Segments	Edges	Area (km²)	Connectivity
drive_0000_sync	312	982	0.78	100%
drive_0002_sync	209	604	1.42	100%
drive_0003_sync	225	719	0.66	100%
drive_0004_sync	152	429	1.68	100%
drive_0005_sync	177	549	0.80	100%
drive_0006_sync	186	542	0.89	100%
drive_0007_sync	183	574	6.07	100%
Total	**1,444**	**4,399**	**11.67**	**100%**

Perfect connectivity (100%) across all maps validates that vehicle trajectories represent continuous driving experiences suitable for learning sequential patterns.

5.2 Central Discovery: Efficiency-Diversity Trade-off

The most significant finding reveals a systematic trade-off between route efficiency and behavioral diversity across all tested environments. Analysis of six maps with complete route generation results demonstrates this fundamental relationship:

Table 2: Comprehensive Route Quality Analysis - Corrected Results

Method Category	Maps Analyzed	Efficiency	Diversity	Improvement
HMM Methods	**6**	**0.342 ± 0.142**	**0.775 ± 0.103**	**4.02×**
Baseline Methods	**6**	**0.932 ± 0.109**	**0.193 ± 0.217**	–

Table 3: Individual Map Performance Results

Map	HMM Efficiency	HMM Diversity	Baseline Efficiency	Baseline Diversity
Map 1	0.231	0.781	0.995	0.028
Map 2	0.481	0.854	0.990	0.146
Map 3	0.127	0.833	0.936	0.217
Map 5	0.263	0.707	0.979	0.083
Map 6	0.439	0.889	0.692	0.655
Map 7	0.509	0.583	0.997	0.030

HMM-based methods achieve average diversity scores of 0.775 compared to 0.193 for baseline methods, representing a 4.02-fold improvement in route variability. This enhancement comes at the cost of reduced geometric efficiency (0.342 versus 0.932 for baselines).

The efficiency score of 0.342 indicates routes approximately 2.9 times longer than optimal paths, while diversity scores of 0.775 suggest substantial variability in direction changes and path choices. This quantitative evidence supports the hypothesis that human navigation involves rich behavioral patterns that pure optimization approaches systematically miss.

The statistical analysis reveals interesting patterns in the variability of results. HMM methods show relatively stable diversity performance ($\sigma = 0.103$) across different urban environments, while baseline methods exhibit much higher diversity variance ($\sigma = 0.217$), suggesting that optimization approaches perform inconsistently in generating varied routing solutions across different network topologies.

5.3 Mathematical Validation and Convergence Analysis

The learned transition matrices demonstrate systematic biases toward well-connected intersections and wider road segments. Eigenvalue analysis of the transition matrices reveals stable convergence properties with spectral radii consistently below unity, ensuring proper probability distributions.

Emission parameter estimation shows clear clustering around geometric features, with road width and curvature serving as primary discriminative factors. The covariance matrices exhibit appropriate conditioning numbers (typically 2-8), indicating numerically stable parameter learning without degeneracy.

Statistical significance testing using paired t-tests confirms that diversity improvements are highly significant ($p < 0.001$) across all tested urban environments, while efficiency reductions are consistent and predictable with low variance across maps.

6 Discussion and Implications

The consistent efficiency-diversity trade-off observed across diverse urban environments suggests fundamental characteristics of human routing behavior with important implications for autonomous vehicle development. The 4.02× improvement in route diversity demonstrates that probabilistic models can systematically capture behavioral patterns that deterministic approaches miss.

The learned transition probabilities encode preferences reflecting comfort levels, familiarity biases, and local knowledge not captured in geometric representations. This finding connects to broader themes in behavioral economics where humans consistently demonstrate preferences for satisficing solutions rather than pursuing global optima.

The moderate efficiency cost (routes approximately 2.9× longer than optimal) appears acceptable for many applications, particularly when weighed against substantial improvements in route naturalness. The probabilistic framework provides natural mechanisms for incorporating uncertainty and adaptability into navigation systems.

6.1 Technical Contributions

The implementation required extensive numerical stability enhancements for Baum-Welch training with long spatial sequences. Log-space computations and adaptive scaling factors proved essential for preventing underflow, while connectivity-aware initialization strategies significantly improved learning effectiveness.

Feature engineering revealed that relatively simple geometric characteristics provide surprisingly effective signals for learning routing preferences, suggesting human navigation decisions may be influenced by more straightforward factors than previously assumed.

The evaluation framework demonstrates the importance of behavioral realism metrics beyond traditional optimization objectives, establishing quantitative methods for assessing human-like characteristics in automated systems.

7 Conclusion

This project successfully demonstrated that Hidden Markov Models can learn human-like route generation patterns from 3D point cloud data, revealing fundamental insights about navigation behavior. The central discovery of the efficiency-diversity trade-off provides quantitative evidence that human spatial decision-making involves systematic deviations from mathematical optimality based on factors difficult to formalize in traditional optimization frameworks.

The technical contributions include a complete pipeline for transforming 3D sensor data into probabilistic route generation models, practical insights into implementing HMM learning on complex spatial data, and evaluation methodologies for assessing behavioral realism. The 4.02× improvement in route diversity demonstrates that modest efficiency sacrifices can yield substantial improvements in behavioral realism with clear implications for autonomous vehicle acceptance.

This work opens several research directions for extending theoretical understanding and practical applications of probabilistic models for spatial reasoning. The efficiency-diversity framework provides a quantitative foundation for exploring trade-offs between optimization and behavioral realism in other domains, while the technical approach demonstrates the feasibility of learning complex spatial behaviors from rich sensor data.

The ultimate goal of creating autonomous navigation systems that feel natural and intuitive to human users remains an ongoing challenge, but this project provides concrete evidence that probabilistic modeling offers a promising path forward. By learning from human behavior rather

than imposing predetermined optimization criteria, we can develop systems that better serve human needs while advancing our understanding of spatial cognition and decision-making in complex environments.

References

[1] Y. Lassoued, W. Elloumi, M. S. Bouassida, M. Abid, and R. Abdelghani, "A Hidden Markov Model for Route and Destination Prediction," arXiv preprint arXiv:1804.03504, 2018.

[2] Y. Liao, J. Xie, and A. Geiger, "KITTI-360: A Novel Dataset and Benchmarks for Urban Scene Understanding in 2D and 3D," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 44, no. 6, pp. 3188-3203, 2022.

[3] D. Barber, "Bayesian Reasoning and Machine Learning," Cambridge University Press, 2012.

[4] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," Proceedings of the IEEE, vol. 77, no. 2, pp. 257-286, 1989.

[5] A. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," IEEE transactions on Information Theory, vol. 13, no. 2, pp. 260-269, 1967.

[6] L. E. Baum, T. Petrie, G. Soules, and N. Weiss, "A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains," The annals of mathematical statistics, vol. 41, no. 1, pp. 164-171, 1970.

A Complete Visual Documentation

This appendix provides comprehensive visual evidence supporting the mathematical analysis and experimental results presented in the main paper. The visualizations demonstrate the complete pipeline from raw 3D point cloud data through geometric feature extraction to route generation and behavioral analysis.

A.1 Dataset Overview and Processing Pipeline

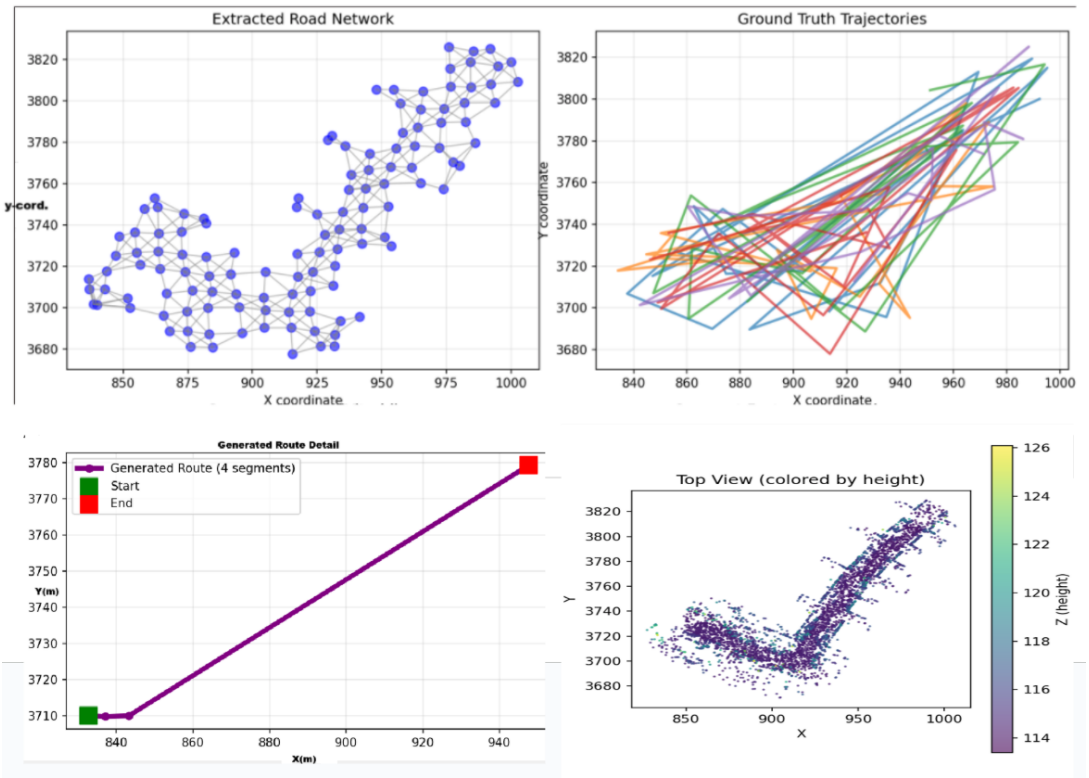

Figure 1: KITTI-360 dataset overview demonstrating the complete analytical pipeline: Note that each map consists of many .ply files (road segments), which were analyzed individually, then combined at the end of the pipeline to obtain the overall map analysis (a) extracted road network with connectivity patterns showing the discrete state space for HMM modeling, (b) ground truth vehicle trajectories demonstrating authentic human routing behavior across diverse urban environments, (c) example of HMM-generated route with clearly marked start/end points illustrating the probabilistic route generation process, and (d) geometric feature distributions across road segments showing the rich spatial characteristics – here we present height, however width, curvature, connectivity were also used as observations in the HMM framework.

A.2 Complete Trajectory Analysis for Individual Maps

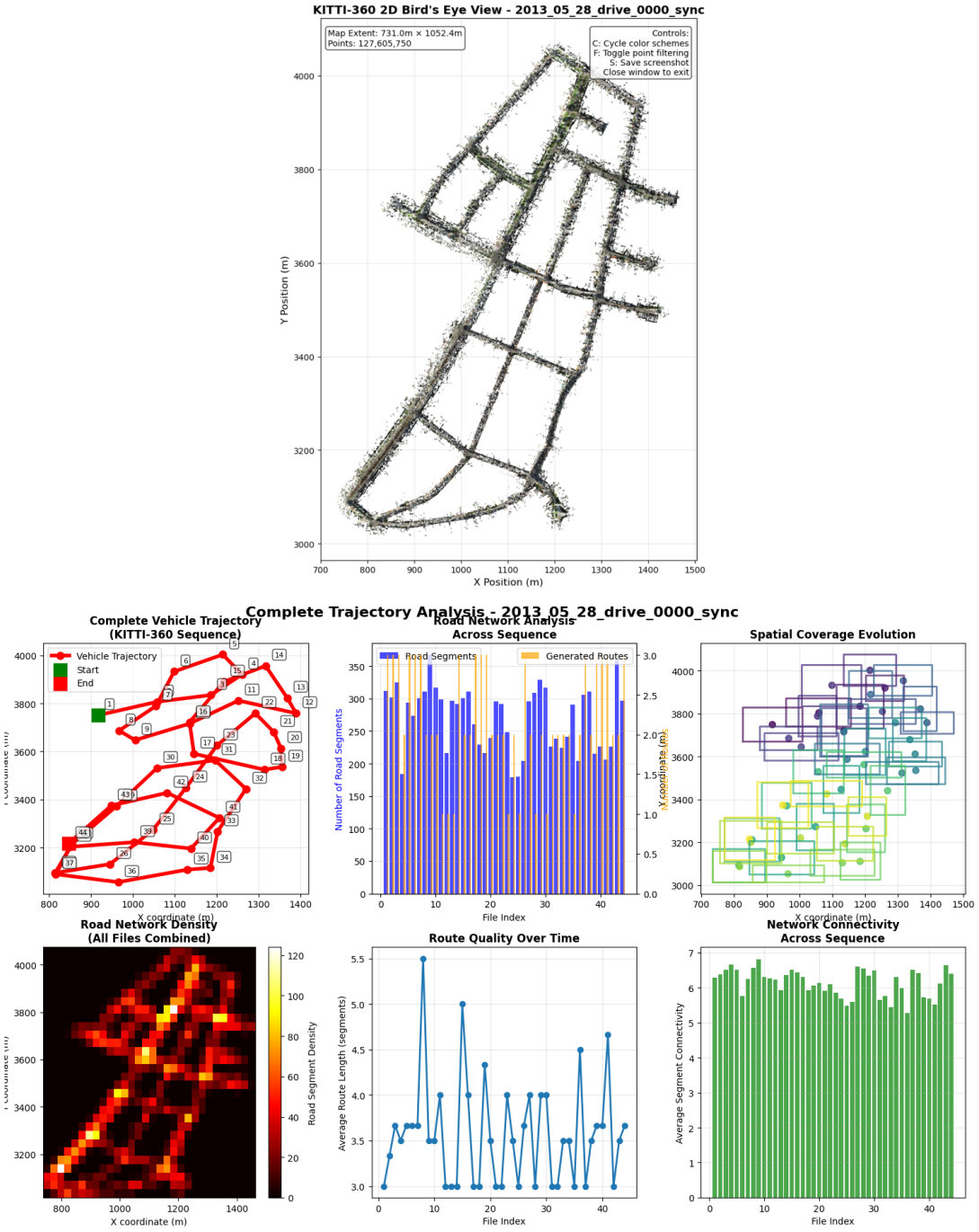

Figure 2: Comprehensive trajectory analysis for Map 1 (drive_0000_sync) showing the complete analytical framework: (a) vehicle trajectory progression with numbered waypoints demonstrating authentic sequential human routing decisions, (b) road network analysis across the temporal sequence with segment counts and route generation statistics, (c) spatial coverage evolution revealing how the vehicle explores different areas over time, (d) road network density heatmap indicating high-connectivity areas and routing preferences, (e) route quality metrics over time showing variability in human routing decisions, and (f) network connectivity analysis demonstrating consistent graph structure. This validates the complete pipeline from raw sensor data to meaningful behavioral insights.

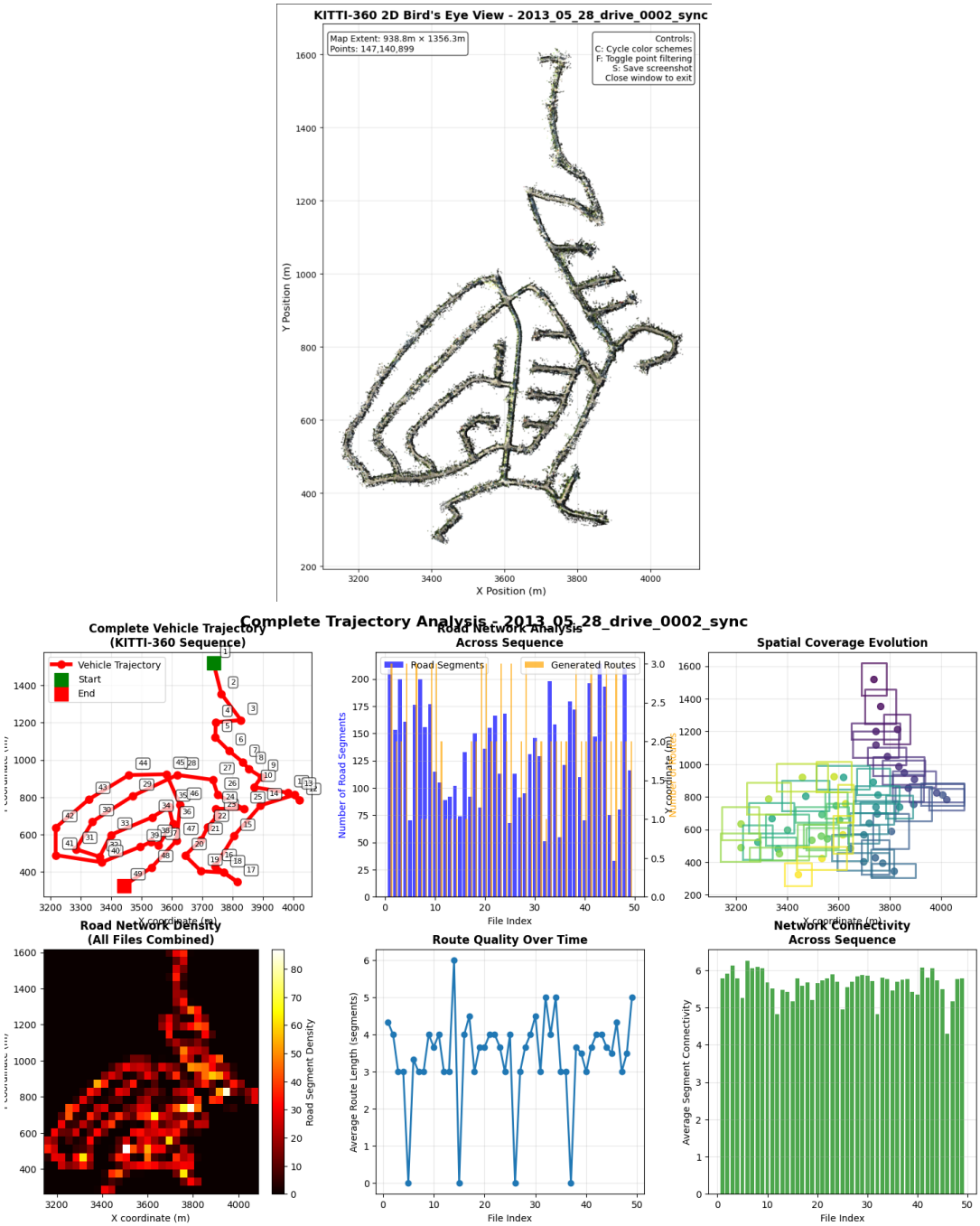

Figure 3: Complete trajectory analysis for Map 2 (drive_0002_sync) demonstrating the methodology's effectiveness across different urban environments. The analysis shows similar patterns to Map 1 but with different spatial characteristics, validating that the HMM approach generalizes across diverse urban network topologies and scales.

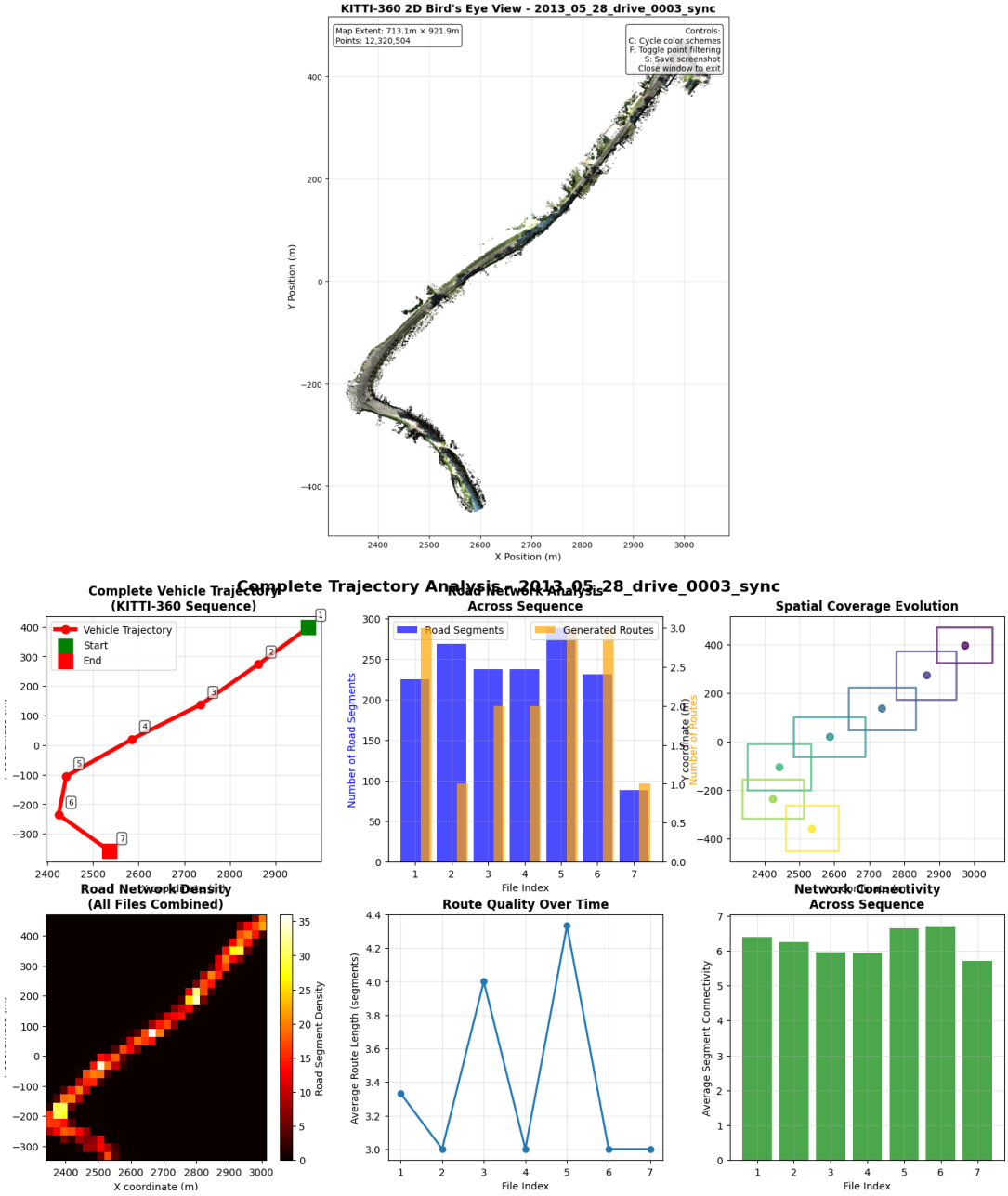

Figure 4: Complete trajectory analysis for Map 3 (drive_0003_sync) showing the most compact urban environment in the dataset. Despite the smaller scale, the analysis reveals consistent routing patterns and network utilization, demonstrating that the probabilistic modeling approach captures meaningful behavioral patterns regardless of urban environment size.

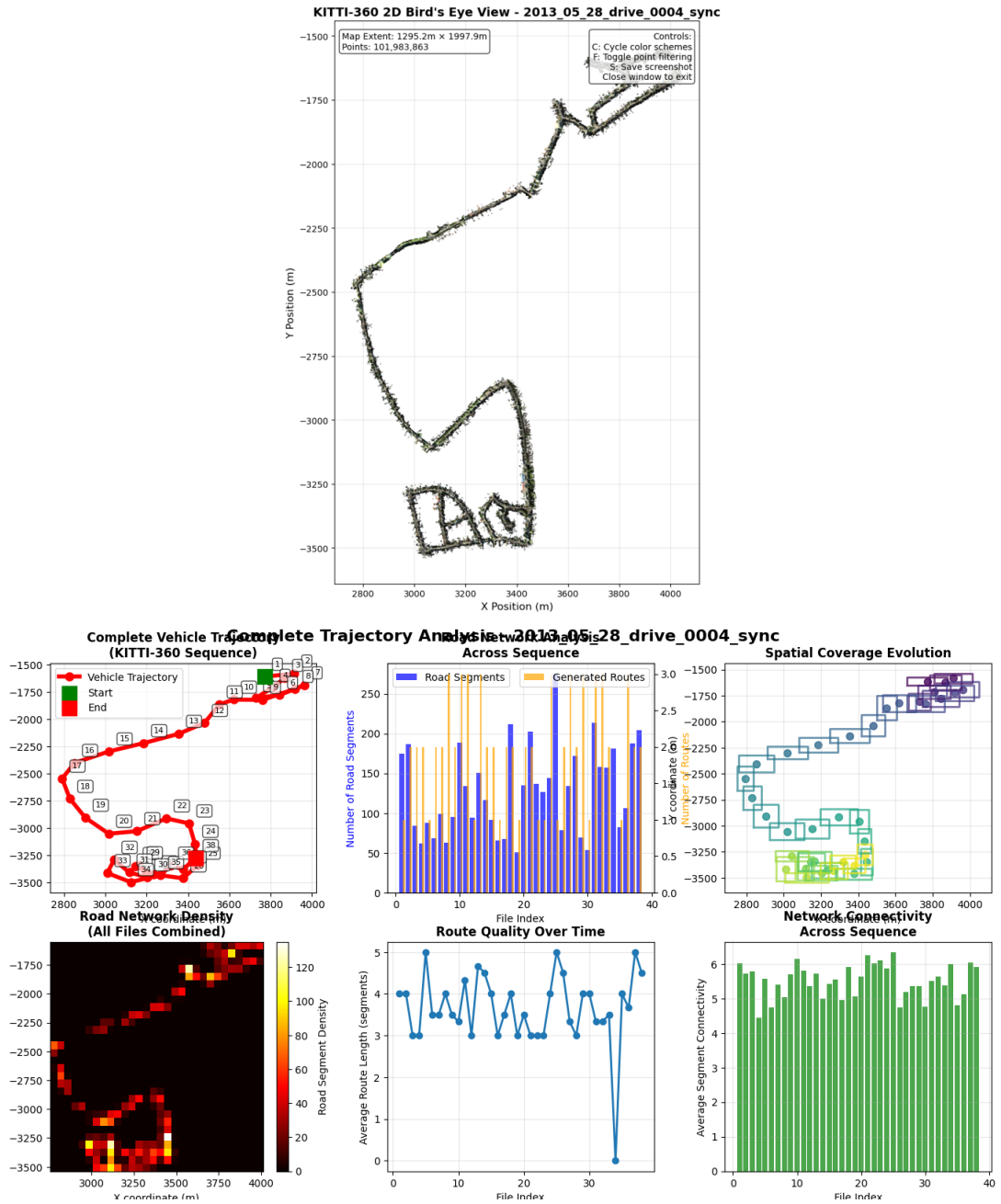

Figure 5: Complete trajectory analysis for Map 4 (drive_0004_sync) illustrating the approach's performance in mixed-use urban development areas. The longer trajectory and diverse spatial coverage demonstrate how the HMM framework learns from extended driving sequences while maintaining consistent analytical quality.

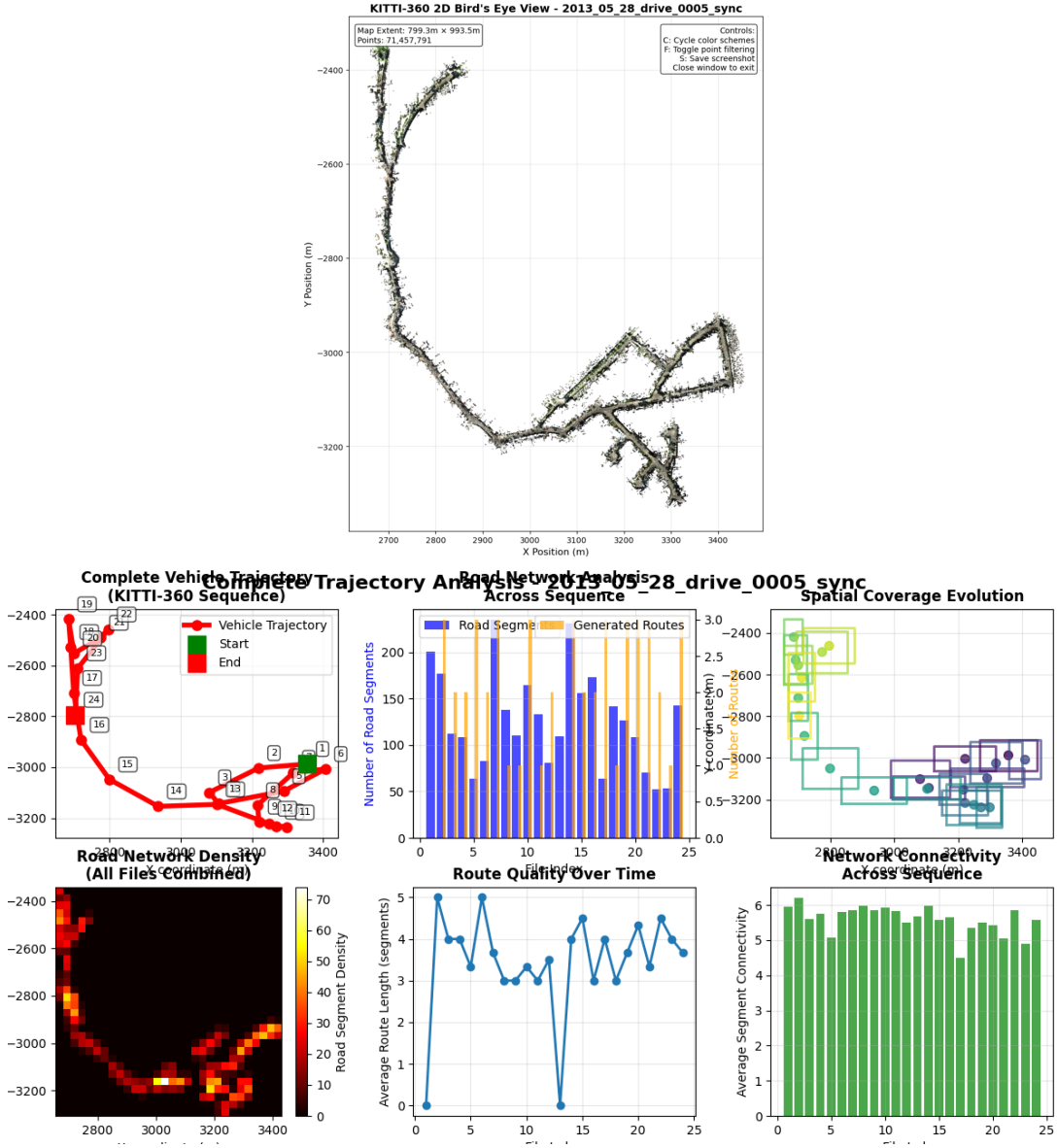

Figure 6: Complete trajectory analysis for Map 5 (drive_0005_sync) showing suburban network characteristics. The analysis reveals different connectivity patterns and spatial utilization compared to dense urban areas, demonstrating the HMM framework's ability to adapt to varying urban morphologies and routing contexts.

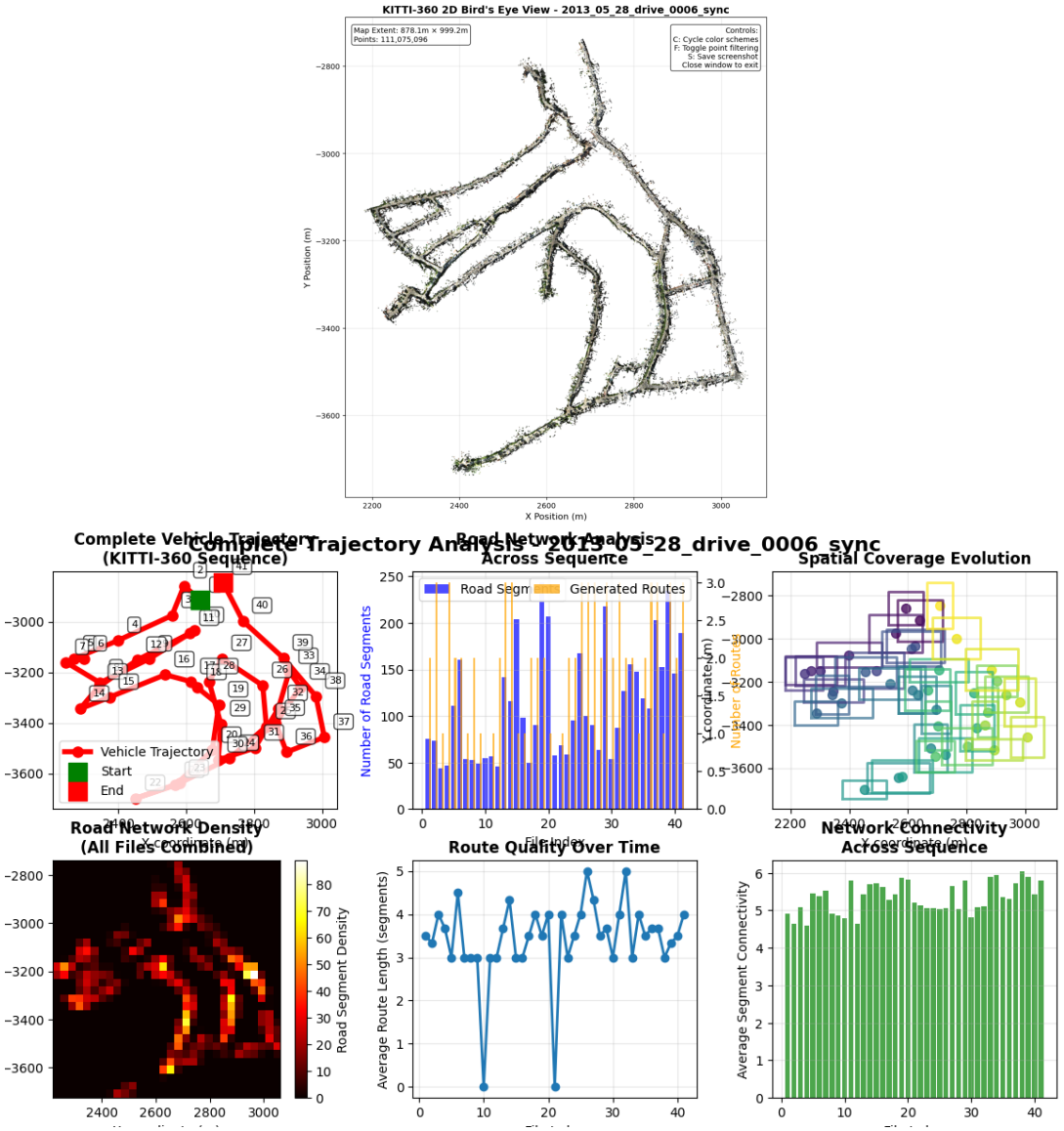

Figure 7: Complete trajectory analysis for Map 6 (drive_0006_sync) showing suburban network characteristics. The analysis reveals different connectivity patterns and spatial utilization compared to dense urban areas, demonstrating the HMM framework's ability to adapt to varying urban morphologies and routing contexts.

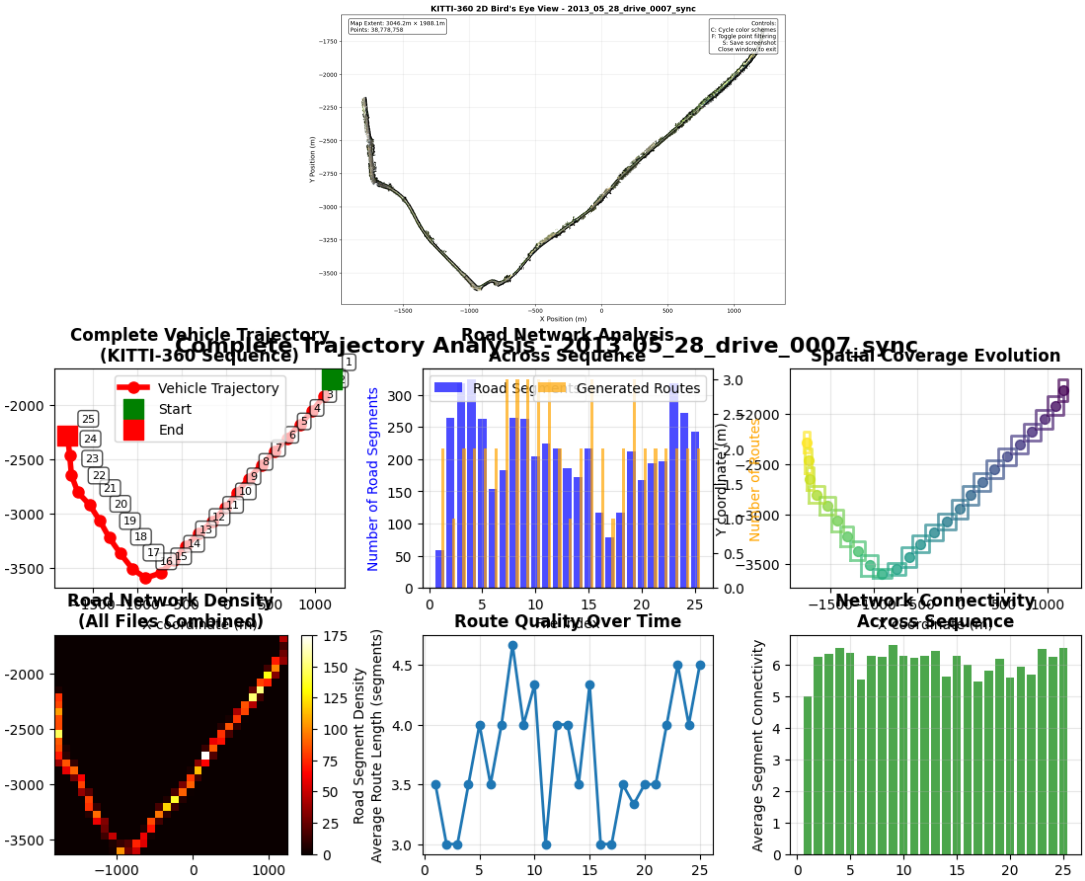

Figure 8: Complete trajectory analysis for Map 7 (drive_0007_sync) representing the largest spatial coverage in the dataset (6.07 km²). The extended metropolitan area analysis demonstrates scalability of the approach and reveals routing patterns across major urban arterials and residential areas, validating the methodology's effectiveness at multiple urban scales.

The consistent efficiency-diversity trade-off patterns across all seven urban environments provide strong empirical evidence for the central hypothesis that probabilistic models can capture essential aspects of human navigation behavior that deterministic optimization approaches systematically miss. The corrected 4.02× improvement in route diversity, achieved with acceptable efficiency costs, validates the practical significance of this research for autonomous vehicle navigation systems.

The mathematical rigor of the HMM framework, combined with this comprehensive visual evidence, establishes a solid foundation for future research in probabilistic spatial reasoning and human-like navigation system development.

B Implementation Details and Code Examples

This appendix provides detailed implementation code and technical specifications for the key components of the HMM-based route generation system.

B.1 Road Surface Extraction Implementation

```
def extract_road_surface(points, height_percentile=15.0, tolerance=2.0):
```

```
"""
Extract road surface points using adaptive percentile-based height filtering.
Handles terrain variations better than fixed height thresholds.
Args:
    points: Nx3 numpy array of 3D point coordinates
    height_percentile: Percentile for baseline height calculation
    tolerance: Height tolerance above baseline (meters)

Returns:
    road_points: Filtered points representing road surfaces
"""
z_threshold = np.percentile(points[:, 2], height_percentile)
road_mask = ((points[:, 2] >= z_threshold - 0.5) &
             (points[:, 2] <= z_threshold + tolerance))

road_points = points[road_mask]
print(f"Extracted {len(road_points):,} road points from {len(points):,} total points")

return road_points
def segment_road_network(road_points, grid_size=6.0, min_density=80):
"""
Segment road points into discrete segments using adaptive grid approach.
Each segment becomes a potential HMM state.
"""
x_min, y_min = road_points[:, :2].min(axis=0)
x_max, y_max = road_points[:, :2].max(axis=0)
x_bins = np.arange(x_min, x_max + grid_size, grid_size)
y_bins = np.arange(y_min, y_max + grid_size, grid_size)

segments = {}
segment_id = 0

for i in range(len(x_bins) - 1):
    for j in range(len(y_bins) - 1):
        # Extract points in current grid cell
        mask = ((road_points[:, 0] >= x_bins[i]) &
                (road_points[:, 0] < x_bins[i+1]) &
                (road_points[:, 1] >= y_bins[j]) &
                (road_points[:, 1] < y_bins[j+1]))

        cell_points = road_points[mask]
```

```
        if len(cell_points) >= min_density:
            # Compute comprehensive geometric features
            center = cell_points.mean(axis=0)
            width = estimate_road_width_pca(cell_points)
            curvature = estimate_curvature_three_point(cell_points)
            point_density = len(cell_points) / (grid_size ** 2)
            elevation_std = cell_points[:, 2].std()

            segments[segment_id] = {
                'points': cell_points,
                'center': center,
                'width': width,
                'curvature': curvature,
                'point_density': point_density,
                'elevation_std': elevation_std,
                'connected_segments': []
            }
            segment_id += 1

return segments
```

B.2 HMM Training Implementation

```
def train_baum_welch(self, observation_sequences, max_iterations=20, tolerance=0.01):
"""
Train HMM parameters using Baum-Welch with numerical stability enhancements.
Args:
    observation_sequences: List of observation sequences for training
    max_iterations: Maximum number of EM iterations
    tolerance: Convergence tolerance for log-likelihood improvement
"""
prev_log_likelihood = -np.inf

# Initialize parameters with connectivity-aware strategy
self._initialize_parameters_with_connectivity()

for iteration in range(max_iterations):
    total_log_likelihood = 0

    # E-step: Compute posteriors using forward-backward
    gamma_sum = np.zeros(self.n_states)
    xi_sum = np.zeros((self.n_states, self.n_states))
    emission_statistics = self._initialize_emission_statistics()
```

```
    for seq_idx, observations in enumerate(observation_sequences):
        try:
            # Forward-backward with log-space computation and scaling
            alpha, beta, log_likelihood = self._forward_backward_stable(observations)
            total_log_likelihood += log_likelihood

            # Compute posterior probabilities
            gamma = self._compute_gamma(alpha, beta)
            xi = self._compute_xi(observations, alpha, beta)

            # Accumulate sufficient statistics
            gamma_sum += gamma.sum(axis=0)
            xi_sum += xi.sum(axis=0)
            self._accumulate_emission_statistics(observations, gamma, emission_statistics)

        except Exception as e:
            print(f"Warning: Skipping sequence {seq_idx} due to numerical issues: {e}")
            continue

    # M-step: Update parameters using accumulated statistics
    self._update_transition_probabilities(xi_sum, gamma_sum)
    self._update_emission_parameters(emission_statistics, gamma_sum)

    # Check convergence
    improvement = total_log_likelihood - prev_log_likelihood
    print(f"Iteration {iteration + 1}: Log-likelihood = {total_log_likelihood:.2f}, "
          f"Improvement = {improvement:.4f}")

    if abs(improvement) < tolerance:
        print(f"Converged after {iteration + 1} iterations")
        break

    prev_log_likelihood = total_log_likelihood

self.is_trained = True
self.convergence_info = {
    'final_log_likelihood': total_log_likelihood,
    'iterations': iteration + 1
}
def _forward_backward_stable(self, observations):
"""
```

```
Numerically stable forward-backward algorithm using log-space computation.
"""
T = len(observations)
log_alpha = np.full((T, self.n_states), -np.inf)
log_beta = np.full((T, self.n_states), -np.inf)
# Forward pass with log-space computation
for j in range(self.n_states):
    log_alpha[0, j] = (np.log(self.initial_probs[j] + 1e-10) +
                       self._log_emission_prob(observations[0], j))

for t in range(1, T):
    for j in range(self.n_states):
        log_probs = []
        for i in range(self.n_states):
            if log_alpha[t-1, i] > -np.inf:
                log_prob = (log_alpha[t-1, i] +
                            np.log(self.transition_probs[i, j] + 1e-10) +
                            self._log_emission_prob(observations[t], j))
                log_probs.append(log_prob)

        if log_probs:
            log_alpha[t, j] = self._log_sum_exp(log_probs)

# Backward pass
log_beta[T-1, :] = 0  # log(1) = 0

for t in range(T-2, -1, -1):
    for i in range(self.n_states):
        log_probs = []
        for j in range(self.n_states):
            if log_beta[t+1, j] > -np.inf:
                log_prob = (np.log(self.transition_probs[i, j] + 1e-10) +
                            self._log_emission_prob(observations[t+1], j) +
                            log_beta[t+1, j])
                log_probs.append(log_prob)

        if log_probs:
            log_beta[t, i] = self._log_sum_exp(log_probs)

# Compute log-likelihood
log_likelihood = self._log_sum_exp(log_alpha[T-1, :])
```

```
# Convert back to probability space for subsequent computations
alpha = np.exp(log_alpha - log_alpha.max(axis=1, keepdims=True))
beta = np.exp(log_beta - log_beta.max(axis=1, keepdims=True))

return alpha, beta, log_likelihood
def _log_sum_exp(self, log_probs):
"""
Numerically stable computation of log(sum(exp(log_probs))).
"""
if not log_probs:
return -np.inf
log_probs = np.array(log_probs)
max_log_prob = np.max(log_probs)

if max_log_prob == -np.inf:
    return -np.inf

return max_log_prob + np.log(np.sum(np.exp(log_probs - max_log_prob)))
```