Cross-View Yaw Estimation in Location Uncertainty with Line-Aligning Yaw Scoring

Taeho Kang[1], Nairan Zhang*[2], Yelin Kim[3], Yujiao Shi[4], and Youngki Lee[1]

[1] Seoul National University, South Korea
[2] Meta, USA
[3] Amazon, USA
[4] ShanghaiTech University, China
taeho.kang@hcs.snu.ac.kr, nairanzhang@meta.com, kimyelin@amazon.com, shiyj2@shanghaitech.edu.cn, youngkilee@snu.ac.kr

Abstract. Accurate yaw estimation is a bottleneck in cross-view localization between ground view and Bird's Eye View (BEV). Existing methods couple yaw with translation and rely on height or projection assumptions that degrade under large yaw ambiguity. We disentangle yaw from location accuracy and introduce LAYS, a radially invariant line-consensus voting method. By exploiting the radial invariance of our formulation, we achieve sub-degree yaw precision via 3D voting over all candidate poses, while eliminating the need for accurate location. Our key observation is that a ground-image column matched to BEV pixels induces the same yaw across all camera positions along the radial direction of the pixels. LAYS matches BEV pixels to ground columns using feature similarity and accumulates the induced yaw votes into discrete 3D bins, where correct correspondences along the radial line concentrate into a sharp peak for the correct yaw. Experiments on Mapillary, Ford, KITTI, and VIGOR show significant gains under unknown yaw, particularly for normal FoV with unknown yaw (+28∼45%p), and using LAYS as a yaw prior improves downstream 3-DoF localization.

Keywords: cross-view localization · aerial view · 3D vision

1 Introduction

Accurate global localization is essential for autonomous navigation, robotics, and AR/MR systems. In these applications, orientation errors are particularly harmful: even small angular deviations can lead to severe navigation drift or visual misalignment. Among the three rotation components, yaw is uniquely challenging. Unlike roll and pitch, which can be inferred from gravity-aligned cues, yaw lacks a direct geometric reference in ground images. As a result, yaw becomes a persistent bottleneck. Without accurate yaw, global orientation alignment degenerates into an unstable joint optimization of translation and rotation.

* This work was done while working at Amazon.

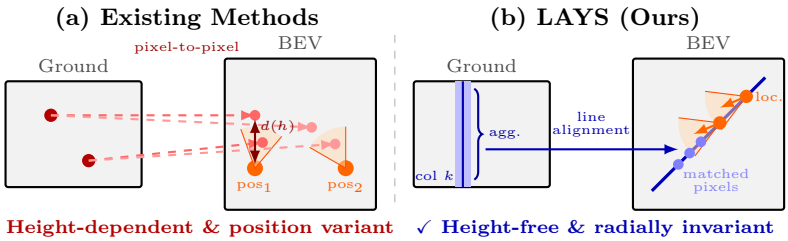


Fig. 1: (a) Existing methods rely on pixel-to-pixel correspondences. Their BEV projections shift based on the assumed camera pose entangling location and yaw, and require a distance estimate $d(h)$ dependent on ground height h. **(b)** LAYS aggregates ground pixels vertically into a column and aligns it to a BEV radial direction. This height-free, column-to-line correspondence makes estimated yaw invariant for any candidate position along the radial line, successfully decoupling yaw from location.

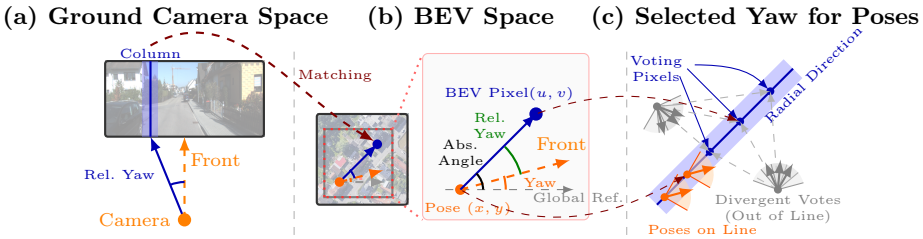


Fig. 2: Yaw Voting Mechanism. (a) A ground column defines a specific relative yaw. **(b)** For any candidate pose, the vector toward its matched BEV pixel defines an absolute angle. The resulting true yaw is calculated by subtracting the relative yaw from this absolute angle. **(c)** A matched BEV pixel casts yaw votes for candidate poses. Poses geometrically aligned on the correct radial direction consistently vote for the exact same true yaw (orange arrows), while poses out of line compute divergent, inconsistent yaws across different pixel matches (gray arrows).

Problem Setup. We consider the following scenario: given a ground-level image and a Bird's Eye View (BEV) image, the 2D camera location has noise (e.g., on the order of $\pm 20\,\text{m}$), while the yaw angle is unknown (up to $\pm 180°$). Our objective is to estimate yaw. Unlike full 3-DoF cross-view localization [28], which jointly estimates 2D location and yaw, our method decouples yaw robustness from location accuracy through Proposition 1's radial invariance: enabling sub-degree yaw precision without accurate camera position.

Although cross-view localization has advanced significantly, existing methods primarily focus on estimating translation, while yaw is assumed to be approximately known or perturbed only slightly (e.g., within $\pm 10°$). Moreover, many approaches tightly couple translation and rotation, relying on point-to-point correspondences with height-dependent projections [11,23,31] (Fig. 1-a) that break down on slopes or in urban clutter.

We argue that yaw estimation should instead be isolated as an independent subproblem. If the camera position were perfectly known, a single-point correspondence would determine the yaw. However, under position uncertainty, a

stronger geometric primitive is required to determine the yaw. To this end, we introduce **LAYS(Line-Aligning Yaw Scoring)**, which formulates cross-view yaw estimation as a *line alignment problem* rather than point-level matching.

LAYS achieves this by vertically aggregating ground image pixels into column features. This inherently removes any dependence on ground height or camera elevation, gracefully bypassing the ground height assumptions of prior work. By matching one such ground column to a set of BEV pixels, we define a radial line (Fig. 1-b). This column-to-line alignment is *radially invariant*: all candidate camera positions along this ray share the exact same absolute direction to the BEV pixel. This yields a consistent yaw estimate regardless of location uncertainty along that line (formalized in Proposition 1).

This geometric invariance motivates a robust voting strategy (Fig. 2): for each candidate position, the match score between a BEV pixel and a ground column votes for a yaw bin determined by their absolute-to-relative angle geometry. True matches accumulate a strong consensus for the correct yaw, while incorrect matches disperse. Unlike traditional voting paradigms that accumulate votes directly for geometric elements, our approach votes for yaw, computed from the geometric relationship between matched features, at all 2D positions. We construct a 3D voting tensor (yaw, 2D position) over all candidate poses, where Proposition 1 ensures correct matches concentrate at the true yaw along the radial line, enabling yaw recovery without pinpointing position.

Decoupling of yaw estimation from 3-DoF localization has two implications. Importantly, **yaw robustness is decoupled from location accuracy**: Proposition 1 guarantees that location uncertainty is handled through radial voting consensus, enabling sub-degree yaw precision without knowing position. Second, precise yaw estimation serves as a reusable module that enhances downstream 3-DoF pipelines beyond what joint optimization alone achieves. Experiments on Mapillary Geo-Localization [21], Ford Multi-AV [2], KITTI [7], and VIGOR [43] confirm the effectiveness of LAYS, consistently outperforming state-of-the-art baselines by large margins.

Contributions. This paper makes the following contributions:

- We **formulate** LAYS, achieving sub-degree yaw precision under $\pm$20m location uncertainty and $\pm180^\circ$ yaw ambiguity, without flat ground or camera height assumptions.
- We **introduce** a line alignment framework integrating column-wise feature extraction, ground-BEV matching, and pairwise yaw voting, decoupling yaw from location through geometric consensus.
- We **demonstrate** state-of-the-art improvements on four datasets, establishing yaw as a tractable subproblem in global pose estimation that further boosts downstream localization.

2 Related Works

Vision-based Global Positioning System In early days [3], methods localized the image coarsely with scale [9,34]. Advanced localization uses feature databases [41]

(e.g., Google Map's Visual Positioning System [19]) or city-scale Structure from Motion [1, 12] with SLAM [17]. These provide accurate localization but require costly, large-scale feature databases. Alternatively, publicly available 2D maps are utilized [21,35], with multi-modal learning to construct a semantic map [22]. 2D maps provide rich semantics, such as roads and buildings, but they lack detailed visual cues for localization.

Visual Camera Rotation Estimation Many rotation-focused pose estimation methods are developed for applications such as Unmanned Aerial Vehicle operations [15], where orientation is critical. Rotation is often part of 6-DoF pose estimation, as location and rotation are typically inseparable in localization. Some focus on frame-to-frame rotation using omnidirectional cameras [5] or handheld cameras in crowded scenes [4]. For 2D rotation, geometry-aware methods predict the horizon line from a single image [39], and roll and pitch estimation using camera parameters has been proposed [10]. Unlike yaw, local features can estimate roll and pitch by using surrounding structural cues to align images with the direction of gravity. In a cross-view setup, rotation estimation is primarily in 2D by using gravity-aligned images.

Ground and Aerial Cross-View Localization Cross-view localization matches a ground-level query to geo-referenced aerial data by bridging the viewpoint gap. Early methods retrieved locations from sparse candidates [8, 13, 25, 28, 29, 42]. Orientation estimation was introduced by applying a polar transform to the satellite image centered at a known position [27], and by matching non-aligned images [43]. Evolved methods estimate precise positions [38] and yaw in BEV by matching deep features [23], often assuming a perspective transform at a specific ground height [6, 30–32]. Techniques include horizontally splitting images [11], rolling descriptors [37], refining homography between BEV and projected ground images [30, 33], and iterative pose refinement [31, 32]. OrienterNet [21] projects ground features onto a 2D map and scores each 3-DoF pose via cross-correlation, matching *what is at a projected location*; this relies on a projection model for ground-to-BEV mapping. Recent method [36] maps the features of the ground view to a 3D grid and uses Procrustes analysis to estimate the pose. A line of works [23, 24, 26] estimate yaw from joint regression or 3-DoF optimization, focusing on slight yaw noise in driving scenes. We propose a yaw estimate that matches *the direction a feature lies in* rather than the projected location, without requiring known position or ground height for ground-to-BEV mapping.

3 Method

3.1 Overview

LAYS framework is illustrated in Fig. 3. We follow standard cross-view localization input. LAYS takes as input an undistorted ground-level image $\mathrm{I}_{grd} \in \mathbb{R}^{3\times H_g\times W_g}$, nearly gravity-aligned along the y axis, and a BEV image $\mathrm{I}_{bev} \in \mathbb{R}^{3\times H_b\times W_b}$, a top-down view that spatially covers the surrounding area. The

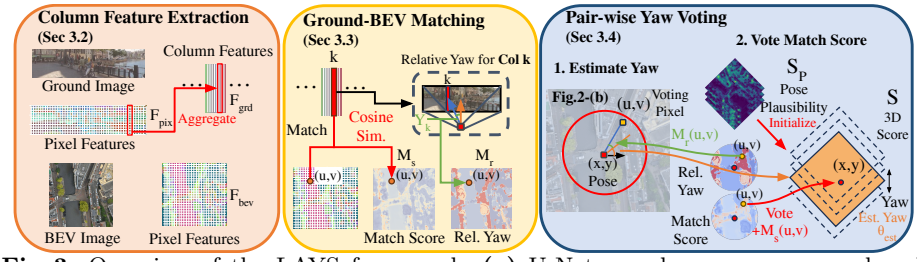


Fig. 3: Overview of the LAYS framework. **(a)** U-Net encoders process ground and BEV images. Ground pixel features are aggregated column-wise to produce column features $\mathbf{F}_{\text{grd}}$. **(b)** Each BEV pixel is matched to the most similar ground column, producing a match score $\mathbf{M_s}$ and relative yaw $\mathbf{M_r}$. **(c)** For each pair of 2D pose and BEV pixel, match scores are voted into yaw bins based on the geometric relationship between absolute and relative angles. (as in Fig. 2-b) The output yaw is the yaw from the highest scoring bin θ from $S(\theta, (x, y))$.

camera intrinsics are given. Importantly, **our formulation does not assume ground-truth location; the 2D pose** (x, y) **remains unknown and is exhaustively tested** over all candidate positions within the noise range. The goal is to estimate yaw, formulated to output a 2D position-conditioned discrete yaw score bins $S(\theta, (x, y))$, a 3D array $S \in \mathbb{R}^{|\Theta|\times|\mathcal{X}|\times|\mathcal{Y}|}$, where (x, y) represents the discrete 2D position in BEV coordinates, and θ denotes the yaw. The yaw from the highest scoring 3D bin is the estimated yaw.

Our relative yaw-based formulation is visualized in Fig. 2. Each ground column defines a relative yaw from the camera's front (Fig. 2-a). For any candidate 2D pose, the yaw is estimated by subtracting the column's relative yaw from the absolute angle to the matched BEV pixel (Fig. 2-b). A single match provides only a point-wise vote; radial lines emerge from consensus. Distinctive structures (roads, trees, building edges) span multiple co-radial BEV pixels sharing the same matched column, and accumulate votes for the same yaw along the radial line regardless of the camera's position along that direction (Fig. 2-c).

Proposition 1 (Position-Invariant Yaw from Line Correspondence). *Let a BEV pixel* (u, v) *match to the ground column* k *with relative yaw* $\mathbf{Y}(k)$*. For any two candidate camera pose* (x_1, y_1) *and* (x_2, y_2) *that are collinear with* (u, v)*, the estimated yaw* $\theta_{\text{est}} = \arctan\left(\frac{v-y}{u-x}\right) - \mathbf{Y}(k)$ *is identical. (Proof in appendix.)*

This is fundamental to our approach: it **decouples yaw estimation from camera location**. Without knowing the true 2D position, we vote over all candidate poses; within line, Proposition 1 ensures that correct matches vote the same yaw regardless of candidate position (as long as it lies on the corresponding radial line). This enables precise yaw estimation despite location uncertainty. The ground height assumption is removed by the column-wise formulation itself (Sec. 3.2): relative yaw depends only on horizontal image position, not on object distance or ground elevation.

Based on these properties, three key components enable precise yaw estimation: **Column Feature Extraction** (Sec. 3.2) extracts column-wise features

from the ground image that correspond to unique relative yaw. **Ground-BEV Matching** (Sec. 3.3) computes feature similarity between ground column and BEV pixel and selects a representative matching ground column for each BEV pixel. **Pair-wise Yaw Voting** (Sec. 3.4) votes match scores for the estimated yaw for each pair of BEV pixel match and 2D candidate pose to find the most probable yaw. More details are in the appendix.

3.2 Column Feature Extraction

We extract features from the BEV and ground images for cross-view matching. The ground pixel features are aggregated column-wise, capturing features with unique relative yaw. This relative yaw is later used to estimate yaw from a pair of BEV pixels and 2D pose. Relative yaw is not impacted by ground height or distance, relaxing the ground height assumption.

Pixel Feature Extraction We employ two U-Net [20] with a VGG-16 [14] encoder to extract C-dimensional features for two images following prior works [26, 31]. The BEV feature map is denoted as $\mathbf{F}_{\text{bev}} \in \mathbb{R}^{C \times H_b \times W_b}$. The ground feature map is extracted at the pixel level $\mathbf{F}_{\text{pix}} \in \mathbb{R}^{C \times H_g \times W_g}$. We add a 4th channel to the ground image before encoding, the heading embedding [32], representing the relative yaw of each pixel.

Column-wise Aggregation We aggregate features column-wise to obtain a relative yaw-aligned feature. For each column k, we weigh and sum the pixel features to extract important features. A fully connected layer N and a softmax compute the pixel feature weights. Given the feature of pixel (k, j) as $\mathbf{F}_{\text{pix}}(k, j)$:

$$\mathbf{F}_{\text{grd}}(k) = \sum_{j=1}^{H_g} \frac{\exp\left(N(\mathbf{F}_{\text{pix}}(k,j))\right)}{\sum_{j'=1}^{H_g} \exp\left(N(\mathbf{F}_{\text{pix}}(k,j'))\right)} \mathbf{F}_{\text{pix}}(k,j) \tag{1}$$

This results in a column-wise feature $\mathbf{F}_{\text{grd}} \in \mathbb{R}^{C \times W_g}$.

Feature Confidence Estimation We compute confidence scores for both BEV and ground features to focus on distinct and matchable features. MLP attached to U-Net computes confidence from $\mathbf{F}_{\text{bev}}$ and $\mathbf{F}_{\text{pix}}$, producing $\mathbf{C}_{\text{bev}} \in \mathbb{R}^{H_b \times W_b}$ and $\mathbf{C}_{\text{pix}} \in \mathbb{R}^{H_g \times W_g}$. For each column, the highest pixel confidence is selected as $\mathbf{C}_{\text{grd}} \in \mathbb{R}^{W_g}$. These confidence scores are used in the matching process to improve robustness by assigning greater weight to more confident features.

3.3 Ground-BEV Matching

We compute cosine similarity scores for each pair of a ground column and a BEV pixel. One representative ground column is selected per BEV pixel to ensure an effective match. This process does not use projection. Instead, we retain a relative yaw from the ground column, assigning a match score to each BEV pixel. This is used to vote yaw in Sec. 3.4 for each BEV-pixel match and 2D pose.

Match Score Computation Match score is absolute cosine similarity between the BEV feature at pixel (u, v) and each ground column feature. Given BEV feature $\mathbf{F}_{\text{bev}}(u, v)$ and k-th ground column feature $\mathbf{F}_{\text{grd}}(k)$, the pairwise match score is:

$$\mathbf{P}((u, v), k) = |\mathbf{F}_{\text{bev}}(u, v) \cdot \mathbf{F}_{\text{grd}}(k)| \cdot \mathbf{C}_{\text{grd}}(k) \tag{2}$$

where $|\cdot|$ denotes the absolute value of the dot product, which yields a scalar since features are C-dimensional unit vectors after channel-wise normalization. Outer $\cdot$ denotes the scalar multiplication.

Probabilistic Feature Selection Column selection differs between training and inference. During training, we use probabilistic selection for the model to learn from diverse column-pixel correspondences, preventing dominant features from overshadowing valid ground-truth matches. In inference, we switch to deterministic (hard) selection to pick the single best match for efficiency.

For each BEV pixel (u, v), we compute a selection distribution over all columns based on their match scores, then sample according to the phase:

$$\text{Selection weights: } w_k = \frac{\mathbf{P}((u, v), k)}{\sum_{k'=1}^{W_g} \mathbf{P}((u, v), k')} \tag{3}$$

where the match scores $\mathbf{P}((u, v), k)$ are normalized into a probability distribution across the W_g columns. The chosen column index $\mathbf{c}(u, v)$ is then:

$$\mathbf{c}(u, v) = \begin{cases} \text{sample from } \{1, \ldots, W_g\} \text{ with probabilities } w_k, & \text{if Training,} \\ \arg\max_k \mathbf{P}((u, v), k), & \text{if Inference} \end{cases} \tag{4}$$

Training-time randomization mitigates overfitting to dominant features and prevents failure cases in which no feature votes for the ground-truth yaw. Without the absolute value, the model collapses to predicting a negative match score for anti-correlated features in non-matching pairs, which is an easier target. Taking the absolute value closes this shortcut while preserving matching quality, at the cost of discarding feature sign. The final BEV pixel match score is computed as:

$$\mathbf{M_s}(u, v) = \mathbf{C}_{\text{bev}}(u, v) \cdot \mathbf{P}((u, v), \mathbf{c}(u, v)) \tag{5}$$

where $\mathbf{C}_{\text{bev}}(u, v)$ and $\mathbf{P}((u, v), \mathbf{c}(u, v))$ are scalar confidence and match scores for the pixel (positive by construction), $\cdot$ is a scalar multiplication. No absolute value is required here, since both operands are already nonnegative.

Relative Yaw Mapping Each ground column k corresponds to a unique relative yaw $\mathbf{Y}(k)$, computed from the 3D ray direction in the camera's coordinate frame. The selected column's relative yaw is assign it to the relative yaw map $\mathbf{M_r}$:

$$\mathbf{M_r}(u, v) = \mathbf{Y}(\mathbf{c}(u, v)). \tag{6}$$

Each BEV pixel (u, v) has an associated match score $\mathbf{M_s}(u, v)$ and a relative yaw $\mathbf{M_r}(u, v)$. These values are utilized in the Yaw Voting to estimate yaw.

3.4 Pair-wise Yaw Voting

Yaw estimation is ambiguous without a known 2D pose (x, y). To resolve this ambiguity, we construct a 3D score bin $S(\theta, (x, y))$ over the full pose search space. We iterate over all candidate 2D poses. For each candidate (x, y), we compute the conditional yaw that aligns each matched BEV pixel (u, v) with the radial line of the ground column, and cast its match score $M_s(u, v)$ into the corresponding yaw bin $S(\theta_{\text{est}}, (x, y))$. This process isolates yaw through a relative yaw-based formulation (Fig. 2-(b)), where correct BEV pixel matches sharing the same ground column agree on the same yaw for all (x, y) candidates along that radial direction (Fig. 2-(c)). This radial consensus removes the need for height-assuming projection and enables sub-degree yaw estimation under location ambiguity.

Yaw Estimation The absolute angle θ_{abs} in BEV coordinate for each candidate pose (x, y) relative to a matched BEV pixel (u, v) is computed by:

$$\theta_{\text{abs}}((x, y), (u, v)) = \arctan\left(\frac{v - y}{u - x}\right) \tag{7}$$

The estimated yaw is then computed by subtracting the matched ground-view relative yaw $\mathbf{M_r}(u, v)$ from the absolute angle, as visualized in Fig. 2:

$$\theta_{\text{est}}((x, y), (u, v)) = \theta_{\text{abs}}((x, y), (u, v)) - \mathbf{M_r}(u, v) \tag{8}$$

Crucially, when the 2D position (x, y) lies on a radial line from pixel (u, v), the absolute angle θ_{abs} remains constant: all positions on the same radial line from the camera toward (u, v) share the same angular direction, so $\arctan\left(\frac{v-y}{u-x}\right)$ is invariant. Therefore, $\theta_{\text{est}} = \theta_{\text{abs}} - Y(k)$ is also constant along the entire line. This radial invariance is the key property that enables robust yaw estimation under location uncertainty (Proposition 1): correct matches vote for the same yaw regardless of where the camera actually sits on that radial direction.

Yaw Voting To handle location uncertainty, we construct a 3D score tensor that votes over all candidate 2D positions and all yaw angles: $\mathbf{S} \in \mathbb{R}^{|\Theta| \times |\mathcal{X}| \times |\mathcal{Y}|}$, where each bin $(\theta, (x, y))$ accumulates votes. The radial invariance from Proposition 1 directly enables this 3D approach. When a BEV-to-column match is evaluated at candidate position (x_1, y_1), it votes for yaw θ_1; the same match evaluated at nearby position (x_2, y_2) on the same radial line votes for the identical yaw θ_1. Thus, correct matches accumulate votes across multiple position bins along the radial line. From multiple BEV matching consensus along a radial line, the true yaw forms a strong peak in the 3D tensor, while incorrect yaws receive scattered votes across disconnected position-angle combinations.

For each candidate pose (x, y), we compute the yaw that each BEV pixel vote contributes to (Equation 7) and accumulate match score into the corresponding

bin. The match score $\mathbf{M_s}(u, v)$ is accumulated to the corresponding yaw bin of θ for each candidate pose (x, y), if (u, v) is within distance R from (x, y):

$$\mathbf{S_M}(\theta, (x, y)) = \sum_{u,v} \mathbf{M_s}(u, v) \cdot \mathbb{I}\Big(\theta = [\theta_{\text{est}}((x, y), (u, v))] \wedge d((x, y), (u, v)) \leq R\Big), \tag{9}$$

where $\mathbb{I}$ is an indicator function ensuring voting only for matching yaw bins, $d((x, y), (u, v)) = \sqrt{(x-u)^2 + (y-v)^2}$. We set R to the largest radius such that a circle of radius R centered at any candidate pose fits entirely within the BEV image; this prevents boundary poses from receiving fewer votes. (details in the appendix) Although the discrete bin assignment is non-differentiable, the match scores $\mathbf{M_s}$ accumulated in each bin are fully differentiable, allowing gradients to flow back to the feature extractors and confidence estimators.

Pose Plausibility Different poses have different likelihoods due to the presence of roads and structures. We add a pose plausibility score $\mathbf{S_P}(\theta, (x, y))$ derived from BEV features using an MLP that provides a hint for estimation, inspired by prior work [26]

$$\mathbf{S}(\theta, (x, y)) = \mathbf{S_P}(\theta, (x, y)) + \mathbf{S_M}(\theta, (x, y)) \tag{10}$$

Multi Level Score We use a 3-level estimate, with different resolutions [26]. One lower level has half the resolution, with a halved number of angular bins. Each level uses a per-pixel 2D pose, and the highest level uses a 1-degree angle bin. Lower-resolution scores are interpolated and added to higher-resolution scores. The highest-resolution is used for the final estimate.

Loss Function Applying softmax over the final score tensor $\mathbf{S}$ for each resolution, we maximize the log probability of the ground-truth pose bin. For ground-truth pose $(x_{\text{gt}}, y_{\text{gt}}, \theta_{\text{gt}})$:

$$\mathcal{L} = -\log \frac{\exp\big(\mathbf{S}(x_{\text{gt}}, y_{\text{gt}}, \theta_{\text{gt}})\big)}{\sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} \sum_{\theta \in \Theta} \exp\big(\mathbf{S}(x, y, \theta)\big)} \tag{11}$$

4 Experiments

4.1 Experiment Setup and Metrics

To evaluate the method under representative real-world conditions in which GPS and sensor data can be noisy, we perturb the ground-truth poses in both position and orientation, following the protocols of prior work [26,37]. Yet, under location uncertainty, we focus on assessing the angular accuracy for yaw.

The angular accuracy metric, commonly used for prior works [26, 32, 37], is the ratio of the correct yaw estimate, where the estimation is correct if the yaw angle error is less than θ degrees relative to the ground truth. Consistent with previous studies [31, 32], we use thresholds of $\theta = 1°$, $2°$, and $4°$.

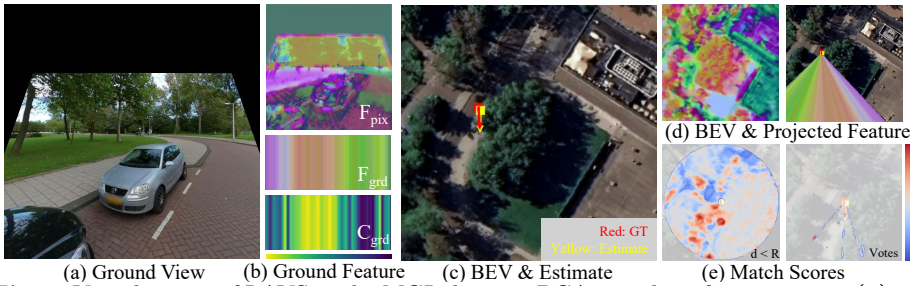


Fig. 4: Visualization of LAYS in the MGL dataset. PCA visualizes feature space. **(a)** is the ground view. **(b)** are the ground pixel, column features, and confidence: $\mathbf{F}_{\text{pix}}$, $\mathbf{F}_{\text{grd}}$, and $\mathbf{C}_{\text{grd}}$, with a colormap for confidence. **(c)** is the BEV image with the estimation result. **(d)** are the BEV and the projected ground feature for the estimate pose (for visualization, not used by our method). **(e)** are match scores M_s within an R distance of the estimated pose and ones that vote for the estimated yaw, with a color map for the voted score scale. Confident votes are concentrated on a line. **More examples from all datasets are in the appendix.**

4.2 Baselines

We compare LAYS against four state-of-the-art baselines with official code: CCVPE [37], BoostAcc [26], G2S [24], and FG2 [36]. CCVPE utilizes rolling descriptors with orientation-map-based pose estimation. BoostAcc and G2S utilize an optimizer and Spatial Transformer Networks to estimate 3-DoF poses, taking only yaw to compute correlations for 2-DoF poses. G2S uses self-supervised learning with only BEV for yaw estimation. FG2 estimates a 3-DoF pose jointly by weighted matching of the 3D grid-mapped ground view and the BEV, using Procrustes analysis.

All baselines and our method use identical training data, splits, and evaluation protocol. The baselines originally report only a small yaw error (e.g., $\pm 10°$), except for VIGOR, where an omnidirectional camera enables estimation of the unknown yaw. Official codes are used to retrain the baselines for unknown yaw setups, with dataloaders adapted accordingly. Some baselines already have the KITTI dataloader; we only changed the rotation error range. For the VIGOR dataset, we use their public weights for evaluation, unless noted as retrained due to unavailability. Despite using official implementation, most methods struggled with a common limited-FoV camera with unknown yaw. We clarify details in the appendix on how each method-dataset pair is adapted.

4.3 Dataset

MGL (Mapillary Geo-Localization) Dataset [21][1] is a crowd-sourced dataset with ground images from vehicles, handheld cameras, and bicycles. Many views are not front-aligned with the road direction, introducing a common challenge for

[1] All images are publicly available under a CC-BY-SA license via the Mapillary API [21].

AR/MR applications where orientation alignment is crucial. SfM-refined poses are provided with gravity-aligned images. We augment the Amsterdam split with Google Maps, which provides ground view at a constant resolution consistent with cross-view localization datasets (27,159 train / 9,103 test, 512×512). Further details in the appendix.

KITTI Dataset [7] is a widely used cross-view localization benchmark with one training set and two test sets (same-area and cross-area). Following prior work [26], images are resized to match the Ford dataset. Up to ±20m location noise and ±180° yaw noise are injected.

Ford Multi-AV Dataset [2] evaluates driving scenes with satellite imagery from Google Map [26]. We use log 1 (highway), where small angular errors are critical. We follow prior works [26], using their splits, ground images are resized to 256×1024, and satellite images are cropped to 512×512. Consistent with prior work [31], up to ±20m location noise and ±45° yaw noise is injected.

VIGOR Dataset [43] contains ground panorama images of four cities. We follow standard fine-grained protocols [11, 37] using positive samples and locational noise (center half width and height of BEV). Two splits are used: same-area (all cities for train/test) and cross-area (two cities each). We use the unknown orientation setup with ±180° yaw uncertainty.

4.4 Results

Table 1 is the evaluation result for the **MGL** dataset, where urban variability introduces height ambiguity that projection-based methods cannot handle. LAYS achieves 72.10% sub-degree accuracy under ±45° noise, doubling G2S (32.67%). Unknown yaw is particularly challenging, as methods are typically evaluated with slight angular noise unless given a 360° panoramic image (e.g., VIGOR). Existing methods largely fail, while our method achieves 34.81% sub-degree accuracy versus 6.55% for CCVPE.

Table 2 shows results on the **KITTI** dataset with ±180° yaw noise. Most methods perform well under small noise (±10°) but break down when the yaw is unknown. Our method achieves 51.02% and 48.46% sub-degree accuracy on same- and cross-area splits, while CCVPE achieves 8.96% and 3.14%. Performance remains consistent across areas, indicating robustness to unseen regions. Finer-grained 1–5° threshold accuracies for MGL and KITTI are reported in the supplementary material.

Table 6 is the **Ford Multi-AV** highway dataset result. LAYS achieves 67.05% sub-degree accuracy, improving 25.69% over the next best method. Following prior work, it is trained and tested without accounting for pitch and roll, yet stays robust to sensory noise.

Table 3 shows results on the **VIGOR** dataset with omnidirectional ground images. The panoramic view provides more information, so prior methods remain competitive. At the $< 4°$ threshold in the same area, FG2's two-stage variant slightly outperforms ours, as our voting concentrates scores in narrow angular peaks optimized for sub-degree precision. At the stricter $< 1°$ and $< 2°$ thresholds, LAYS leads by 13.38% and 12.10%, and at all cross-area thresholds.

Table 1: Comparison of cross-view yaw estimation accuracy on MGL Dataset with ±20m location noise under ±45° and ±180° yaw noise.

Method	±45°			±180°		
	$< 1°$	$< 2°$	$< 4°$	$< 1°$	$< 2°$	$< 4°$
CCVPE [37]	21.36	40.78	68.15	6.55	12.55	23.79
BoostAcc [26]	7.40	14.19	27.54	0.58	1.09	2.13
G2S [24]	32.67	57.49	82.50	0.59	1.27	1.86
FG2 [36]	18.59	34.97	59.20	3.13	6.28	12.44
Ours	**72.10**	**90.16**	**95.63**	**34.81**	**52.42**	**61.74**

Table 2: Comparison of cross-view yaw estimation accuracy on KITTI dataset with ±20m location noise and unknown yaw. *: CCVPE reports accuracy at $< 1°$, $< 3°$, and $< 5°$ thresholds [37]; $\leq$ entries denote that the published value is from a looser threshold ($< 3°$ for $< 2°$, $< 5°$ for $< 4°$), so true accuracy is equal or lower.

Method	Test 1 (Same Area)			Test 2 (Cross Area)		
	$< 1°$	$< 2°$	$< 4°$	$< 1°$	$< 2°$	$< 4°$
CCVPE* [37]	8.96	$\leq$26.48	$\leq$42.75	3.14	$\leq$9.24	$\leq$14.56
BoostAcc [26]	0.53	0.90	1.96	0.56	1.05	2.09
G2S [24]	0.58	1.17	2.17	0.54	0.82	1.64
FG2 [36]	1.62	3.15	6.47	1.62	2.96	6.13
Ours	**52.98**	**65.07**	**67.72**	**49.59**	**58.92**	**60.25**

Table 3: Comparison of cross-view yaw estimation accuracy on VIGOR Dataset (omnidirectional camera), ±0.25 BEV width and height location noise, and unknown yaw. *: BoostAcc did not evaluate on VIGOR. **: G2S evaluated only the translation estimator on VIGOR. †: FG2 provides VIGOR result with a two-stage approach.

Method	Same Area			Cross Area		
	$< 1°$	$< 2°$	$< 4°$	$< 1°$	$< 2°$	$< 4°$
CCVPE [37]	8.31	16.69	32.40	8.96	17.66	34.34
BoostAcc [26]*	0.50	1.10	2.16	0.64	1.20	2.30
G2S [24]**	2.33	4.60	8.92	3.98	7.72	14.45
FG2 [36]	19.05	35.94	58.91	9.88	19.37	35.57
FG2 [36]†	20.78	38.17	**62.11**	12.39	23.48	41.40
Ours	**34.16**	**50.27**	59.32	**23.67**	**37.09**	**47.15**

4.5 Analysis

Matching Process and Location Fig. 4 and appendix examples across all datasets illustrate the matching process. High-scoring matches arise along short co-linear pixels and vote the same yaw for all locations along the radial line (as in Fig. 2-c). The voting is agnostic to the physical nature of matched structures; appendix visualizations confirm correspondences among road segments, tree branches, and building facades across diverse scenes. Overall, our formulation distributes scores along a radial line rather than concentrating them at a single pixel. This is by design: the method learns the most confident line

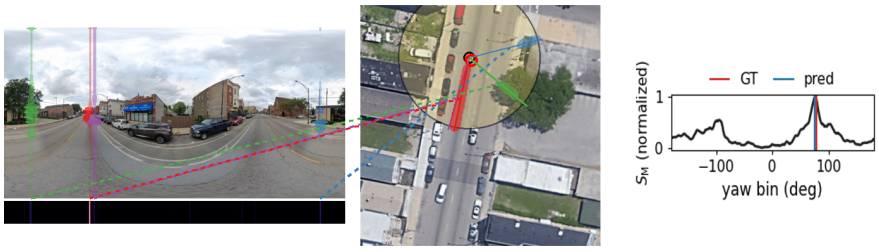


Fig. 5: Matching visualization for one VIGOR cross-area sample. *Ground view (top):* the four columns that contribute most to the predicted yaw, with bar length proportional to the per-pixel saliency $|\nabla \cdot I|$ of the matching score w.r.t. the input pixels. *BEV (left):* thin colored lines mark each column's view direction from the predicted pose; circles on every pixel show the score each column votes into the predicted yaw bin, with circle area proportional to match score. Yellow and red markers denote the predicted and the GT pose. *Match score curve (right):* S_M over the full 360° yaw range is plotted.

correspondences for yaw. Our method can instead support 3-DoF localization methods as shown in Sec. 4.5, reducing their search space to 2-DoF.

Fig. 5 traces a single VIGOR cross-area prediction through the matching pipeline. Different columns focus on different cues: two lock onto road segments and produce the largest match scores, one onto a tree, and one onto a grass patch in front of a building. The matching-only S_M curve peaks at the correct yaw, with a smaller peak near the opposite (180°) direction.

Ablation Study Table 7 ablates key components. In **Point-to-Point Match**, we replace radially invariant scoring with projection-based 3-DoF scoring that maps ground pixels to BEV for each pose candidate; yielding decent results yet significantly behind radially invariant scoring. In **Polar-Transform**, we adapt OrienterNet [21]'s polar-transform-based projective 3-DoF pose scoring to satellite view input, and their flexible mapping overfits compared to the naive 3-DoF scoring, demonstrating our formulation's advantage over existing 3-DoF scoring approaches. We analyze Column Feature Extraction (Sec. 3.2). In **Keypoint-based**, we use a keypoint feature extractor [32], which leads to poor performance because keypoints often miss important features. **Column MLP** uses MLP to aggregate pixel features per column. The weighted-sum formulation yields better accuracy. Finally, we show the effectiveness of the hybrid BEV-ground matching. In **Greedy Selection**, we match each BEV pixel to the ground feature with the highest correlation score during training, where dominant features overshadow others and limit information diversity. **Test-time Weighted** uses probabilistic matching during the test-time. While yielding decent results, the highest-scoring match proved more effective during inference.

Table 4: FG2 localization error (m) on cross-area VIGOR with different first-stage yaw initializers.

First-stage yaw	Mean↓
FG2	10.02
+ Two-stage	5.95
+ **Ours init.**	5.43
+ **Ours init.** (2-DoF)	**5.10**
+ GT Yaw (oracle)	2.41

Table 5: FG2 localization error (m) on MGL and KITTI with different first-stage yaw initializers.

First-stage yaw	Mean↓	Median↓
MGL (±45°)		
FG2	10.53	8.90
+ Two-stage	10.54	8.95
+ **Ours init.**	**6.42**	**3.99**
KITTI (±180°)		
FG2	14.32	14.40
+ Two-stage	14.44	12.26
+ **Ours init.**	**2.52**	**1.48**

Accurate Yaw's Impact on Localization Method We show that isolating yaw from the full 3-DoF problem is effective even when the localization method does not break down. Tables 4 and 5 report the localization error of FG2 [36] when its first-stage yaw is replaced by different estimators. + **Two-stage** initializes FG2's second-stage localizer with FG2's own first-stage yaw; + **Ours init.** substitutes LAYS's yaw, and the VIGOR **(2-DoF)** variant additionally fixes our yaw and solves only 2D location via Procrustes analysis. On VIGOR (Table 4), where FG2's yaw is already functional, our 2-DoF variant still improves localization by 14% over FG2's two-stage (5.95 → 5.10), approaching the ground-truth-yaw oracle (2.41). The benefit grows as the base yaw becomes unreliable (Table 5): even under mild noise on MGL the substitution more than halves the median error (8.95 m → 3.99 m), and under unknown yaw on KITTI, where the two-stage FG2 stays near random, our yaw makes precise localization possible (12.26 m → 1.48 m median). Isolating yaw is therefore most critical in the hardest cases.

Robustness Against Pitch and Roll Noise Our column-wise formulation is inherently insensitive to pitch errors, as features are aggregated per column rather than per pixel row. Roll is handled at the feature-encoding level, since the roll-tilted horizon is clear in the input image. Table 8 confirms our method stays robust under ±10° pitch and roll noise on MGL, with minimal degradation.

Computational Cost LAYS runs at 0.16 s per query for VIGOR [43] Dataset with most columns on an NVIDIA RTX A6000, competitive with FG2 [36]'s 0.26 s. The yaw-voting stage dominates but stays comparable to feature extraction (appendix has a breakdown), with room for kernel-level optimization.

Failure Cases and Limitations Our method has two main limitations. First, it cannot provide precise standalone localization, as the voting formulation distributes scores along radial lines rather than at a single location; we address

Table 6: Comparison of cross-view yaw estimation accuracy on Ford Dataset Log1 (Highway) with ±20m location noise and ±45° yaw noise.

Method	<1°	<2°	<4°
CCVPE [37]	41.36	62.74	75.11
BoostAcc [26]	24.38	43.24	67.86
G2S [24]	16.81	47.71	68.43
FG2 [36]	16.75	41.84	57.13
Ours	**67.05**	**85.14**	**91.76**

Table 7: Impact of components in LAYS for yaw estimation with 180° yaw ambiguity in MGL dataset.

Alternative Method	<1°	<2°	<4°
Point-to-Point Match	13.19	23.67	38.88
Polar-Transform [21]	7.59	14.53	23.74
Keypoint-based	20.12	28.75	33.10
Column MLP	31.73	48.23	58.13
Greedy Selection	28.69	41.70	47.62
Test-time Weighted	30.77	44.72	51.33
Ours	**34.81**	**52.42**	**61.74**

Table 8: Impact of pitch and roll noise on cross-view yaw estimation accuracy on MGL Dataset with 20m location noise and unknown yaw.

Pitch, Roll Noise	<1°	<2°	<4°
±0°, ±0°	34.81	52.42	61.74
±10°, ±0°	34.36	50.26	58.61
±10°, ±10°	33.15	48.98	58.60

this by using LAYS as a yaw prior for downstream 3-DoF methods (Sec. 4.5). Second, performance degrades in scenes with highly uniform texture (e.g., open fields), rare in practice. The appendix provides a detailed failure analysis.

5 Conclusion

We introduced LAYS, a sub-degree yaw estimator based on geometric line-alignment consensus between the ground view and the BEV. Our key insight is that a single radial correspondence is sufficient to determine yaw, even when the camera position is unknown: pairwise voting across all candidate positions makes the formulation radially invariant, eliminating the ground-height or known-location assumptions that prior methods require. Experiments on Mapillary, Ford, KITTI, and VIGOR show consistent state-of-the-art gains across diverse scenarios, from urban imagery to highway driving. By isolating yaw from location, LAYS shrinks the localization search space and improves downstream 3-DoF accuracy, opening directions for integrating yaw priors into broader localization pipelines for navigation and AR/MR applications.

Acknowledgments This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No. RS-2024-00463802, No. RS-2022-NR070595).

References

1. Agarwal, S., Snavely, N., Simon, I., Seitz, S.M., Szeliski, R.: Building rome in a day. In: 2009 IEEE 12th International Conference on Computer Vision. pp. 72–79 (2009). https://doi.org/10.1109/ICCV.2009.5459148
2. Agarwal, S., Vora, A., Pandey, G., Williams, W., Kourous, H., McBride, J.: Ford multi-av seasonal dataset (2020)
3. Brejcha, J., Čadík, M.: State-of-the-art in visual geo-localization. Pattern Anal. Appl. **20**(3), 613–637 (Aug 2017). https://doi.org/10.1007/s10044-017-0611-1, https://doi.org/10.1007/s10044-017-0611-1
4. Delattre, F., Dirnfeld, D., Nguyen, P., Scarano, S., Jones, M.J., Miraldo, P., Learned-Miller, E.: Robust frame-to-frame camera rotation estimation in crowded scenes. In: IEEE/CVF International Conference on Computer Vision (ICCV). pp. 9752–9762 (10 2023)
5. Du, Y., Mateo, C., Tahri, O.: A multilayer perceptron-based spherical visual compass using global features. Sensors **24**(7) (2024). https://doi.org/10.3390/s24072246, https://www.mdpi.com/1424-8220/24/7/2246
6. Fervers, F., Bullinger, S., Bodensteiner, C., Arens, M., Stiefelhagen, R.: Uncertainty-aware vision-based metric cross-view geolocalization. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 21621–21631 (June 2023)
7. Geiger, A., Lenz, P., Stiller, C., Urtasun, R.: Vision meets robotics: The kitti dataset. International Journal of Robotics Research (IJRR) (2013)
8. Guo, Y., Choi, M., Li, K., Boussaid, F., Bennamoun, M.: Soft exemplar highlighting for cross-view image-based geo-localization. IEEE Transactions on Image Processing **31**, 2094–2105 (2022). https://doi.org/10.1109/TIP.2022.3152046
9. Hays, J., Efros, A.A.: im2gps: estimating geographic information from a single image. In: Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR) (2008)
10. Jin, L., Zhang, J., Hold-Geoffroy, Y., Wang, O., Matzen, K., Sticha, M., Fouhey, D.F.: Perspective fields for single image camera calibration. In: CVPR (2023)
11. Lentsch, T., Xia, Z., Caesar, H., Kooij, J.F.P.: Slicematch: Geometry-guided aggregation for cross-view pose estimation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 17225–17234 (June 2023)
12. Li, Y., Snavely, N., Huttenlocher, D., Fua, P.: Worldwide pose estimation using 3D point clouds. In: European Conf. on Computer Vision (2012)
13. Liu, L., Li, H.: Lending orientation to neural networks for cross-view geolocalization. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 5617–5626 (2019)
14. Liu, S., Deng, W.: Very deep convolutional neural network based image classification using small training sample size. In: 2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR). pp. 730–734 (2015). https://doi.org/10.1109/ACPR.2015.7486599
15. Liu, Y., Tao, J., Kong, D., Zhang, Y., Li, P.: A visual compass based on point and line features for uav high-altitude orientation estimation. Remote Sensing **14**(6), 1430 (2022)
16. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. In: International Conference on Learning Representations (2019), https://openreview.net/forum?id=Bkg6RiCqY7

17. Middelberg, S., Sattler, T., Untzelmann, O., Kobbelt, L.: Scalable 6-dof localization on mobile devices. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) Computer Vision – ECCV 2014. pp. 268–283. Springer International Publishing, Cham (2014)
18. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: Pytorch: An imperative style, high-performance deep learning library. Advances in Neural Information Processing Systems **32** (2019)
19. Rajpurohit, A., Kumar, P., Singh, D., Kumar, R.: A review on visual positioning system. SSRN Electronic Journal (01 2024). https://doi.org/10.2139/ssrn.4485458
20. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. CoRR **abs/1505.04597** (2015), http://dblp.uni-trier.de/db/journals/corr/corr1505.html#RonnebergerFB15
21. Sarlin, P.E., DeTone, D., Yang, T.Y., Avetisyan, A., Straub, J., Malisiewicz, T., Bulo, S.R., Newcombe, R., Kontschieder, P., Balntas, V.: OrienterNet: Visual Localization in 2D Public Maps with Neural Matching. In: CVPR (2023)
22. Sarlin, P.E., Trulls, E., Pollefeys, M., Hosang, J., Lynen, S.: SNAP: Self-Supervised Neural Maps for Visual Positioning and Semantic Understanding. In: NeurIPS (2023)
23. Shi, Y., Li, H.: Beyond cross-view image retrieval: Highly accurate vehicle localization using satellite image. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2022)
24. Shi, Y., Li, H., Perincherry, A., Vora, A.: Weakly-supervised camera localization by ground-to-satellite image registration. In: Computer Vision – ECCV 2024: 18th European Conference, Milan, Italy, September 29–October 4, 2024, Proceedings, Part IX. p. 39–57. Springer-Verlag, Berlin, Heidelberg (2024). https://doi.org/10.1007/978-3-031-72673-6_3, https://doi.org/10.1007/978-3-031-72673-6_3
25. Shi, Y., Liu, L., Yu, X., Li, H.: Spatial-aware feature aggregation for image based cross-view geo-localization. In: Wallach, H., Larochelle, H., Beygelzimer, A., d'Alch'e-Buc, F., Fox, E., Garnett, R. (eds.) Advances in Neural Information Processing Systems 32, pp. 10090–10100. Curran Associates, Inc. (2019), http://papers.nips.cc/paper/9199-spatial-aware-feature-aggregation-for-image-based-cross-view-geo-localization.pdf
26. Shi, Y., Wu, F., Perincherry, A., Vora, A., Li, H.: Boosting 3-dof ground-to-satellite camera localization accuracy via geometry-guided cross-view transformer. In: 2023 IEEE/CVF International Conference on Computer Vision (ICCV). pp. 21459–21469 (2023). https://doi.org/10.1109/ICCV51070.2023.01967
27. Shi, Y., Yu, X., Campbell, D., Li, H.: Where am i looking at? joint location and orientation estimation by cross-view matching. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 4063–4071 (2020). https://doi.org/10.1109/CVPR42600.2020.00412
28. Shi, Y., Yu, X., Liu, L., Zhang, T., Li, H.: Optimal feature transport for cross-view image geo-localization. In: arXiv preprint arXiv:1907.05021 (2019)
29. Shi, Y., Yu, X., Wang, S., Li, H.: Cvlnet: Cross-view feature correspondence learning for video-based camera localization. In: Proceedings of the Asian Conference on Computer Vision (ACCV). pp. 652–669 (December 2022)
30. Song, Z., xianghui, z., Lu, J., Shi, Y.: Learning dense flow field for highly-accurate cross-view camera localization. In: Oh, A., Naumann, T., Glober-

son, A., Saenko, K., Hardt, M., Levine, S. (eds.) Advances in Neural Information Processing Systems. vol. 36, pp. 70612–70625. Curran Associates, Inc. (2023), https://proceedings.neurips.cc/paper_files/paper/2023/file/df5f94d6ac6e13d830d70536cde9f0d2-Paper-Conference.pdf
31. Wang, S., Nguyen, C., Liu, J., Zhang, Y., Muthu, S., Maken, F.A., Zhang, K., Li, H.: View from above: Orthogonal-view aware cross-view localization. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 14843–14852 (June 2024)
32. Wang, S., Zhang, Y., Perincherry, A., Vora, A., Li, H.: View consistent purification for accurate cross-view localization. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 8197–8206 (2023)
33. Wang, X., Xu, R., Cui, Z., Wan, Z., Zhang, Y.: Fine-grained cross-view geolocalization using a correlation-aware homography estimator. In: Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., Levine, S. (eds.) Advances in Neural Information Processing Systems. vol. 36, pp. 5301–5319. Curran Associates, Inc. (2023), https://proceedings.neurips.cc/paper_files/paper/2023/file/112d8e0c7563de6e3408b49a09b4d8a3-Paper-Conference.pdf
34. Weyand, T., Kostrikov, I., Philbin, J.: Planet - photo geolocation with convolutional neural networks. ArXiv **abs/1602.05314** (2016), https://api.semanticscholar.org/CorpusID:171846
35. Wu, H., Zhang, Z., Lin, S., Mu, X., Zhao, Q., Yang, M., Qin, T.: Maplocnet: Coarse-to-fine feature registration for visual re-localization in navigation maps. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2024)
36. Xia, Z., Alahi, A.: Fg^2: Fine-grained cross-view localization by fine-grained feature matching. In: Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR). pp. 6362–6372 (June 2025)
37. Xia, Z., Booij, O., Kooij, J.F.P.: Convolutional cross-view pose estimation. IEEE Transactions on Pattern Analysis and Machine Intelligence **46**(5), 3813–3831 (2024). https://doi.org/10.1109/TPAMI.2023.3346924
38. Xia, Z., Booij, O., Manfredi, M., Kooij, J.F.: Visual cross-view metric localization with dense uncertainty estimates. In: European Conference on Computer Vision. pp. 90–106. Springer (2022)
39. Xian, W., Li, Z., Fisher, M., Eisenmann, J., Shechtman, E., Snavely, N.: Uprightnet: Geometry-aware camera orientation estimation from single images. 2019 IEEE/CVF International Conference on Computer Vision (ICCV) pp. 9973–9982 (2019), https://api.semanticscholar.org/CorpusID:201107189
40. Xu, B., Wang, N., Chen, T., Li, M.: Empirical evaluation of rectified activations in convolutional network (2015), https://arxiv.org/abs/1505.00853
41. Yang, A., Beheshti, M., Hudson, T.E., Vedanthan, R., Riewpaiboon, W., Mongkolwat, P., Feng, C., Rizzo, J.R.: Unav: An infrastructure-independent vision-based navigation system for people with blindness and low vision. Sensors **22**(22) (2022). https://doi.org/10.3390/s22228894, https://www.mdpi.com/1424-8220/22/22/8894
42. Zhu, S., Shah, M., Chen, C.: Transgeo: Transformer is all you need for cross-view image geo-localization. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1162–1171 (2022)
43. Zhu, S., Yang, T., Chen, C.: Vigor: Cross-view image geo-localization beyond one-to-one retrieval. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 3640–3649 (2021)

Appendix

In this appendix, we provide the following:

- Proof of Proposition 1 (Sec. A)
- Training setup (Sec. B)
- Pseudo-code for Pair-wise Yaw Voting (Sec. C)
- Implementation Details (Sec. D)
- Baseline Evaluation Details (Sec. F)
- Preprocessing for Mapillary Geo-Localization Dataset (Sec. E)
- Visualization of Matching Process (Sec. G)
- Computational Cost Analysis (Sec. H)
- Failure Case Analysis (Sec. I)
- Extended Yaw Analysis (Sec. J)
- Discussion and Future Work (Sec. K)

A Proof of Proposition 1

Proof. If (x_1, y_1), (x_2, y_2), and (u, v) are collinear, then $\frac{v-y_1}{u-x_1} = \frac{v-y_2}{u-x_2}$, so $\arctan\left(\frac{v-y_1}{u-x_1}\right) = \arctan\left(\frac{v-y_2}{u-x_2}\right)$. Since $\mathbf{Y}(k)$ depends only on the matched column, θ_{est} is constant along the radial line through (u, v).

B Training Setup

For the training setup, we based our implementation on the codebase provided by Shi et al. [26]. We use NVIDIA RTX A6000 for training, with batch size 6 and learning rate 10^{-4}, using AdamW [16] optimizer. Loss weight e^{-5} is used to adjust the loss scale. StepLR from PyTorch [18] is used, with a gamma of 0.25 and step size of 4000 iterations.

C Pseudo-code for Pair-wise Yaw Voting

We provide the pseudo-code of the Yaw-aligned Bin Score Scattering method's key component, score accumulation in Alg. 1, for clarification of its unique formulation. We wrote it in a loop form to make it more intuitive. PyTorch functions were used for effective implementation and GPU-accelerated parallelism. Further implementation details are in Sec. D.

The $\mathbf{S}$ array is indexed with coordinates and discrete yaws in the pseudo-code. $\mathcal{X}$ and $\mathcal{Y}$ are all valid x and y 2D camera pose's pixels in BEV space within noise range, and Θ represents valid discrete yaws in degree $[-\theta_{max}, -\theta_{max} + 1, ..., \theta_{max} - 1, \theta_{max}]$, which depends on the maximum yaw noise θ_{max}.

Algorithm 1 Pair-wise Yaw Voting with Distance Constraint

Input: Match yaw array $\mathbf{M_r}$, Match score array $\mathbf{M_s}$, Pose plausibility $\mathbf{S_P} \in \mathbb{R}^{|\Theta|\times(|\mathcal{X}|\times|\mathcal{Y}|)}$
Output: Yaw bins for 2D poses $\mathbf{S} \in \mathbb{R}^{|\Theta|\times(|\mathcal{X}|\times|\mathcal{Y}|)}$
$\mathbf{S} \leftarrow$ Zero-filled array $\in \mathbb{R}^{|\Theta|\times(|\mathcal{X}|\times|\mathcal{Y}|)}$
$\theta_{\max} \leftarrow$ Max yaw noise (degrees)
$R \leftarrow$ Max distance radius
for all each candidate pose $(x, y) \in \mathcal{X} \times \mathcal{Y}$ **do**
  for all each BEV pixel (u, v) in All BEV pixels **do**
    $d \leftarrow \sqrt{(u-x)^2 + (v-y)^2}$ ▷ Calculate distance
    if $d < R$ **then** ▷ Apply distance constraint
      $\theta_{\text{abs}} \leftarrow \arctan\left(\frac{v-y}{u-x}\right)$ ▷ Absolute angle
      $\theta_{\text{rel}} \leftarrow \mathbf{M_r}(u, v)$ ▷ Relative yaw
      $\theta_{\text{est}} \leftarrow \theta_{\text{abs}} - \theta_{\text{rel}}$ ▷ Estimated yaw
      $B_{\text{est}} \leftarrow \lfloor \theta_{\text{est}} + 0.5 \rfloor$ ▷ Yaw bin index
      if $-\theta_{\max} \leq B_{\text{est}} \leq \theta_{\max}$ **then**
        $\mathbf{S}(B_{\text{est}}, (x, y)) \mathrel{+}= \mathbf{M_s}(u, v)$ ▷ Vote match score to yaw
      end if
    end if
  end for
end for
$\mathbf{S} \leftarrow \mathbf{S} + \mathbf{S_P}$ ▷ Pose plausibility adjustment

D Implementation Details

This section provides a detailed formulation of the method not discussed in the main paper for brevity. The variables denoted using the notation of a hat, such as $\hat{x}$ instead of x, are used for implementation.

D.1 Feature Extraction

We utilize the implementation of U-Net [20] built on the VGG-16 [14] encoder backbone, which features separate weights for the ground and the BEV, as provided in the official code of Shi et al. [26], along with an additional heading encoding channel. Instead of the cosine-based representation used by Wang et al. [32], we use linear mapping of the x coordinate from -1 to 1 and concatenate them along the channel axis.

Unlike the original implementation of Shi et al. [26], the features are not normalized per image. Instead, ground and BEV features are normalized along channel dimensions:

$$\hat{\mathbf{F}}_{\text{bev}}(u, v) = \frac{\mathbf{F}_{\text{bev}}(u, v)}{\|\mathbf{F}_{\text{bev}}(u, v)\|_2}, \quad \hat{\mathbf{F}}_{\text{grd}}(c) = \frac{\mathbf{F}_{\text{grd}}(c)}{\|\mathbf{F}_{\text{grd}}(c)\|_2} \tag{12}$$

$\hat{\mathbf{F}}_{\text{bev}}$ and $\hat{\mathbf{F}}_{\text{grd}}$ are used for the cosine-similarity computation explained in the method section.

The confidence for each feature is computed using the original feature before normalization. For the ground confidence estimator, output values are directly from the dense layer, while the BEV confidence estimator values are the sigmoid of the dense layer output. Max normalization is applied to the BEV confidence value, so the maximum confidence is 1.

$$\mathbf{C}_{\text{bev}}^{\max} = \max_{u,v} \mathbf{C}_{\text{bev}}(u, v) \tag{13}$$

$$\hat{\mathbf{C}}_{\text{bev}}(u, v) = \frac{\mathbf{C}_{\text{bev}}(u, v)}{\mathbf{C}_{\text{bev}}^{\max}} \tag{14}$$

D.2 Pair-wise Yaw Voting with Distance Constraint

We only consider possible positions under the assumption of the noise range for the score bins. The score bin is $\mathbb{R}^{|\Theta| \times (|\mathcal{X}| \times |\mathcal{Y}|)}$. $\mathcal{X}$ and $\mathcal{Y}$ are the sets of ranges for the x and y of pixels covering the maximal location noise range:

$$\mathcal{X} = \{[i\Delta x,\, (i+1)\Delta x) \,|\, i \in \{-L, \ldots, L-1\}\} \tag{15}$$

$$\mathcal{Y} = \{[j\Delta y,\, (j+1)\Delta y) \,|\, j \in \{-L, \ldots, L-1\}\} \tag{16}$$

where Δx and Δy correspond to the same meter per pixel value of the BEV feature. Denoting maximal location noise along one direction as $N_{\mathbf{max}}$, L is:

$$L = \left\lceil \frac{N_{\mathbf{max}}}{\Delta x} \right\rceil \tag{17}$$

Θ is the range of θ in degrees, from the ceiling of $-\theta_{\mathbf{max}}$ to $\theta_{\mathbf{max}}$ divided by each resolution's bin's angular interval $\Delta\theta$ explained in the Sec. D.4.

$$\Theta_i = \{((i-1)\Delta\theta, i\Delta\theta)\,, i \in \{-n, \ldots, -1, 0, 1, \ldots, n-1\}\} \tag{18}$$

where

$$n = \left\lceil \frac{\theta_{\max}}{\Delta\theta} \right\rceil \tag{19}$$

We choose a fixed $R = 36.32$, subtracting 20m from the maximal distance from the center 256×0.22m, for pixel length and meters per pixel for the Mapillary Geo-localization dataset [21] and the Ford dataset [2]. For the VIGOR dataset [43], we use a quarter of BEV's length of one side for R, as the meter per pixel varies across different cities.

D.3 Pose Plausibility Score Model

The pose plausibility score $\mathbf{S_P}$ is computed with a model based on Shi et al. [26]'s Uncertainty estimator implementation. BEV features from a separate VGG-16 U-Net are used for input to the model, as sharing features disturbs the training of the main match scores. The intermediate layer dimension is adjusted to fit the output channel and number of angular bins for each resolution. The LeakyReLU [40] replaces the final non-linearity with a negative slope 0.01.

D.4 Multi Resolution Scoring

The final score is computed for feature at each resolution, following common practice [26, 31, 32]. The ground and BEV image features with the same downsampling ratio from the U-Net are used for each resolution. Following Shi et al. [26], we use features downsampled 2, 4, and 8 times for scoring, and the score from the 2 times downsampling features is used for the final estimation. Each resolution is supervised using the log-probability loss described in the paper, with the lower-resolution feature used for coarser estimation.

The score bin is built per pixel for different coarseness levels for each resolution. The angular bin has discrete intervals of 1, 2, and 4 degrees for each resolution, with larger intervals for lower resolutions. The score is repeated along each axis and summed to a higher resolution. The summation is done before applying kernels for the scattering-based match score S_M and the 3D plausibility score S_P.

E Preprocessing for Mapillary Geo-Localization Dataset

The dataset, paired with a 2D map, can be downloaded by following instructions from OrienterNet [21]'s repository. The data contains ground-view images with camera pose information. We use the urban Amsterdam part of the dataset for the cross-view setup. The Google Static Maps API fetches satellite-view images, and we filtered out top-down occluded views. Consistent with previous work [26, 31], satellite views are obtained as 1280×1280 images with a zoom level of 18 and a scale of 2. The satellite view is fetched for each ground view; however, no new image is downloaded if any existing satellite image covers a 512×512 region around the ground view.

OrienterNet [21], which introduced the Mapillary Geo-Localization Dataset, states that they removed ground views with low visibility of the surroundings, such as those facing a wall. We found such images remained, and removed those portions. The Amsterdam dataset is generally at a 512×512 resolution; we filtered out a small number of images with different resolutions to ensure a fair evaluation against the baselines. The train and test sets are split by sequence index to minimize spatial overlap.

F Baseline Evaluation Details

In this section and Table 9, we describe how each baseline uses each dataset and how we adapted the official code to evaluate each method at its best under our setting.

F.1 Per-Dataset Adaptation

The **Ford dataset** has been evaluated less frequently in recent work; only BoostAcc [26] has reported results on it. Since they train and evaluate on the $\pm 10^\circ$

Table 9: Summary of baseline official implementations for Ford, KITTI, and VIGOR datasets. For each implementation status, we did the following: **None**: Dataloader adapted from external codebase. **Metrics on Paper**: Results cited from the original paper. **Small Yaw Range**: Retrained for high-yaw-noise. **Unused in Paper**: The official code's unused train script is fixed to evaluate full 3-DoF pose estimation. **Only Location**: The official code trains and evaluates only the location estimator for the dataset; the rotation estimator training code is adapted from another dataset script.

Method	Ford	KITTI	VIGOR
CCVPE [37]	None	Metrics on Paper	Metrics on Paper
BoostAcc [26]	Small Yaw Range	Small Yaw Range	Unused in Paper
G2S [24]	None	Small Yaw Range	Only Location
FG2 [36]	None	Small Yaw Range	Metrics on Paper

yaw noise, we retrained the method with a $\pm 45°$ yaw range. The BoostAcc dataloader is adapted for both our method and other baselines for training and evaluation.

The **KITTI dataset** is widely used as a benchmark, yet methods commonly evaluate with only $\pm 10°$ yaw error. We used official codes and modified the rotation range during training and testing to evaluate baselines. CCVPE [37] reported an official result with unknown yaw, but the weights were not released. CCVPE reports only $1°/3°/5°$ accuracies, which we place in our $1°/2°/4°$ columns. Since accuracy increases with the threshold, its $2°/4°$ entries are higher, conservative than reality.

For the **VIGOR dataset**, CCVPE [37] and FG2 [36] report unknown yaw estimation accuracy, and released model source code. FG2 released the VIGOR weight, and we ran inference to compute the accuracy for our angle threshold. CCVPE did not release the VIGOR weight, so we use the metrics reported in the paper. BoostAcc [26] did not use the dataset in the paper, but has a dataloader in the released code; we use it to retrain and test the model. G2S [24] provides a dataloader, but they did not use their rotation estimator for unknown yaw setups, stating that panoramic ground-view and BEV differences make it difficult to apply their self-supervised method for yaw estimation.

For the **MGL dataset**, none of the methods support it; thus, we adapted the official dataloader from OrienterNet [21] code to use BEV input, for all baselines and ours.

F.2 Behavior Across Yaw Uncertainty Ranges

LAYS's margin over the baselines in Tables 2 and 3 widens as the yaw candidate range grows. Table 1 shows this on MGL across $\pm 45°$ and $\pm 180°$ noise (a $4\times$ expansion of the candidate range): CCVPE's $< 1°$ accuracy drops from 21.36% to 6.55%, FG2 from 18.59% to 3.13%, BoostAcc from 7.40% to 0.58%, and G2S from 32.67% to 0.59%. Refinement-based methods (G2S, BoostAcc) fall

the most: with a limited-FoV camera, their refinement cannot escape an initial $\sim 180°$ error. LAYS also drops ($72.10 \rightarrow 34.81$), but far less.

G Visualization of Matching Process

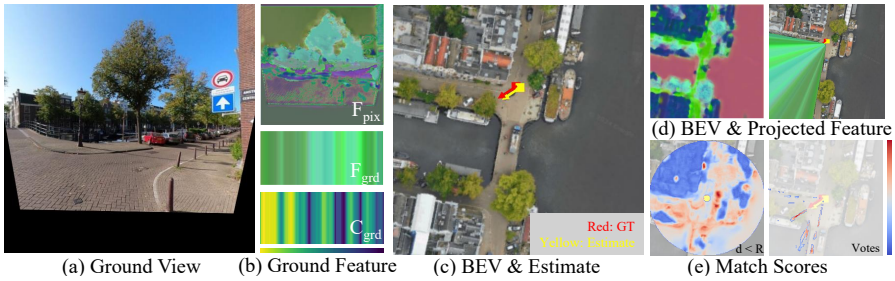


Fig. 6: Visualization of our yaw estimation in the MGL dataset. Our method gives high confidence and a match score for the column-aligned road on the left and the tree in the middle.

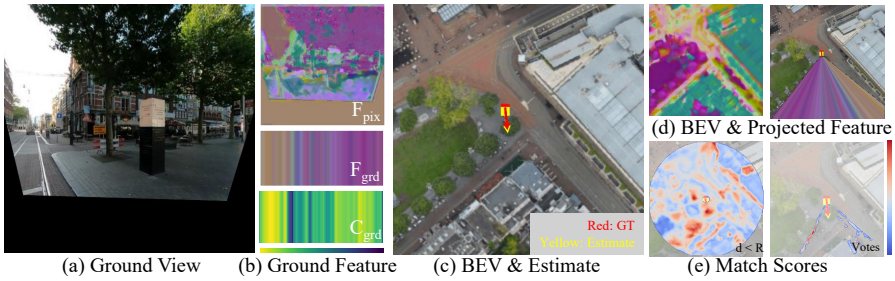


Fig. 7: Visualization of our yaw estimation in the MGL dataset. The estimated pose has the most votes from a tree.

In this section, we demonstrate the matching process and explain how our method accurately estimates yaw with more examples. We visualize the matching process using Principal Component Analysis (PCA). We extract three major components to represent it as RGB channels. For each sample, ground column features and BEV features are sampled and re-scaled for PCA. Since we use an absolute similarity score, feature vectors are normalized by flipping the sign when they have a negative similarity with the mean of all feature vectors.

Experiments in the MGL [21] dataset revealed that our method not only focuses on the road, but also on prominent roadside features. Fig. 6 and Fig. 7

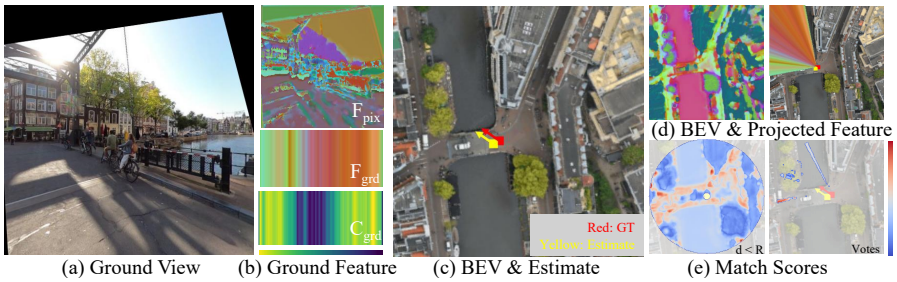


Fig. 8: Visualization of our yaw estimation in the MGL dataset. Our method has high confidence and a match score on the features extracted from the building at the front of the road.

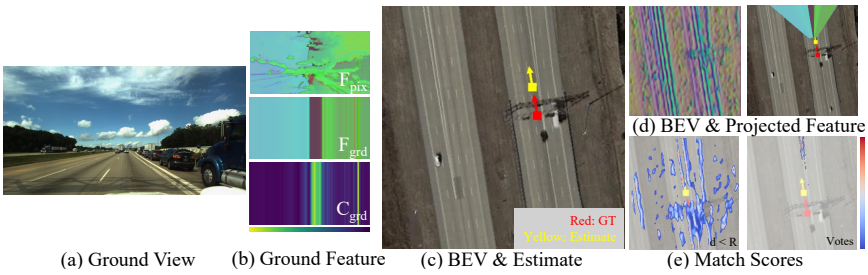


Fig. 9: Visualization of our yaw estimation in the Ford dataset. Our method aligns the road components.

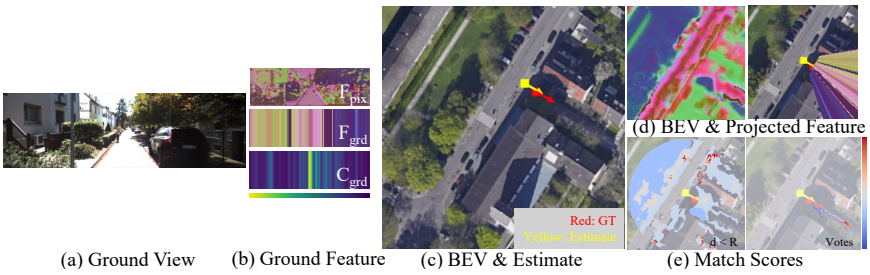


Fig. 10: Visualization of our yaw estimation in the KITTI dataset, same-area (Test 1). Despite the wide-baseline perspective change between the ground view and BEV, our method identifies road and roadside structure correspondences to estimate yaw.

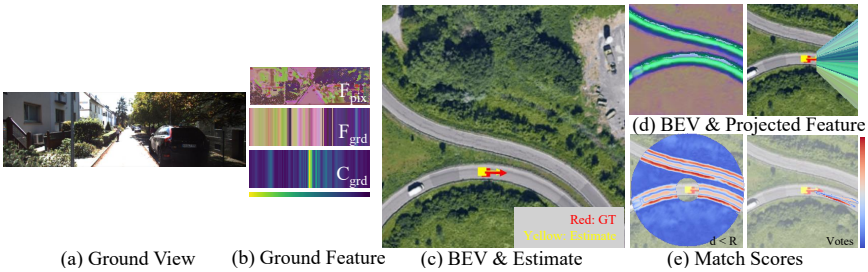


Fig. 11: Visualization of our yaw estimation in the KITTI dataset, cross-area (Test 2). Our method generalizes to unseen geographic areas. The same area is mostly urban; our method generalizes to curved highways in the cross-area data, finding radial correspondences along road boundaries.

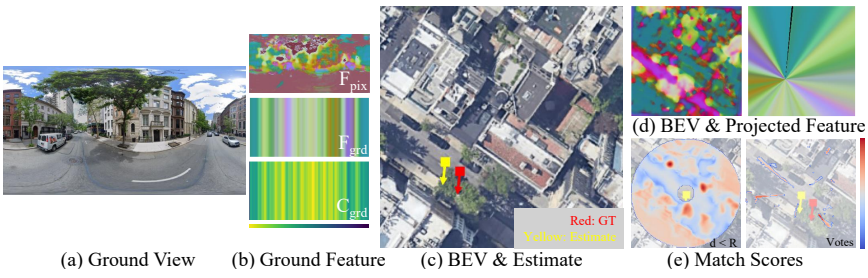


Fig. 12: Visualization of our yaw estimation in the VIGOR dataset, same-area setup. Our method aligns a specific building and road.

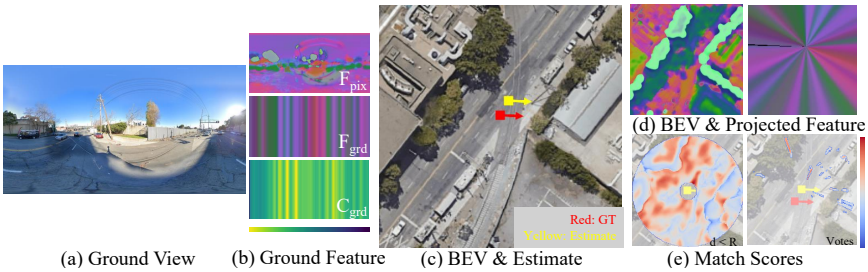


Fig. 13: Visualization of our yaw estimation in the VIGOR dataset, cross-area setup. Our method aligns the road and a tree.

provide examples from the MGL dataset that focus on a tree, colored in dark purple and cyan, respectively. Fig. 8 shows the case where our process focuses on the building in the road front direction, with the column feature visualized as light brown on the left side of the ground image. Generally, our method estimates yaw based on the focused score in one direction.

The Ford [2] and VIGOR [43] dataset results primarily focus on the driveway where the datasets were captured. An example from the Ford dataset is visualized in Fig. 9, with ground-view-related features resized to their original aspect ratio. The features from the left and right sides of the view appear different, due to positional encoding attached to the input image. Match scores are focused on road lanes, and the final estimation primarily comes from the lane in the front direction.

The KITTI [7] dataset results are shown in Fig. 10 (same-area, Test 1) and Fig. 11 (cross-area, Test 2). The cross-area example demonstrates generalization to previously unseen geographic regions. Similar to the Ford dataset, matches are focused on road structure, as KITTI also focuses on ground-level views from a moving car.

Fig. 12 shows the matching in the VIGOR dataset, same-area setup. Our method identifies correspondences such as sidewalk boundary (brown) and driveway (purple). Fig. 13 illustrates the matching process using the VIGOR dataset, specifically the cross-area setup. At the road intersection, our method identifies correct road correspondence and finds yaw, focusing on a road (purple) and a tree (dark green) near the other road.

H Computational Cost Analysis

Table 10 breaks down the per-batch forward pass by stage. The majority of computation comes from voting; however, it remains comparable to the feature extraction process. Note that the voting with absolute and relative yaw indexing is not implemented efficiently using the existing PyTorch function. A better kernel implementation can improve runtime performance.

Table 10: Per-batch forward pass breakdown

Stage	Percentage in Time
Feature extraction	24.2
Pose plausibility network	0.9
Ground-BEV matching	1.7
Yaw voting	73.2

I Failure Case Analysis

We identify three main categories of failure cases:

Severe Temporal Change When the BEV image and ground image are captured at significantly different times, scene content may have changed substantially due to construction, seasonal vegetation changes, or demolition. In such cases, the learned features may fail to find correct correspondences because the visual content no longer matches between views. Datasets include some temporal variation, and our method demonstrates robustness to moderate changes; however, extreme differences (e.g., a demolished building or newly constructed road) can cause failures.

We observed that other failures often accompanied more common temporal changes, such as mismatched weather conditions, day and night, and trees blocking the road. While those typically do not significantly impact estimation, since those shifts can be learned during training, they can impair performance when combined with other sources of confusion, such as the failure cases below.

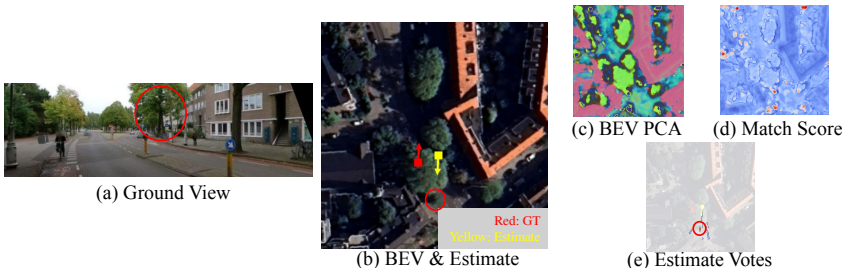
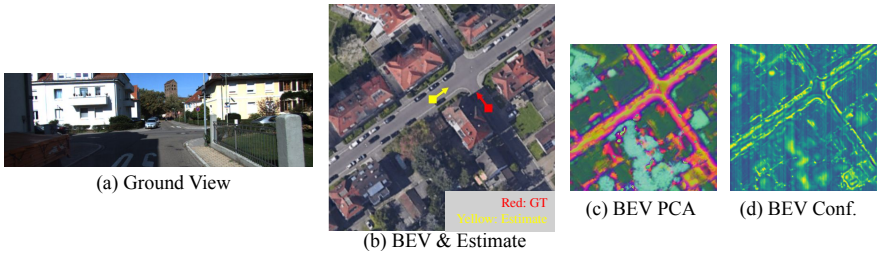


Fig. 14: Failure case examples in the MGL dataset **Severe Occlusion of BEV**: road is completely covered by trees when the BEV image was captured. Severe occlusion makes it difficult to match ground view and BEV, especially in day night time change from ground view to BEV. For the estimated pose, vote mainly comes from a small tree in the BEV, and high match scores (red color) are generally on small trees, which aligns with the tree in the middle of image.

Severe Occlusion of BEV There are some ground-level images taken in places that are occluded from a top-down view, such as a road completely covered by

trees in Fig. 14. Due to a lack of matching visual features, our method focuses on a small tree in the ground view; however, it matches a different tree, and without support from surrounding visuals, this leads to yaw estimation failure.



(a) Ground View (b) BEV & Estimate (c) BEV PCA (d) BEV Conf.

Fig. 15: Failure case examples in the KITTI dataset. **Symmetric scene**: a crossroad produces ambiguous votes between two visually similar directions. Temporal impact, such as shades on the road, reduced confidence (viridis colormap, yellow indicates high confidence, green indicates lower) on the correct road, resulting in similar-looking different road positions selected.

Symmetric Scenes At locations with rotational symmetry, such as crossroads with four visually similar streets, circular plazas, or repetitive building patterns, multiple radial directions may produce equally strong votes, such as Fig. 15, leading to ambiguous yaw estimates. The voting mechanism may split its consensus across multiple yaw angles or select an incorrect symmetric alternative. This failure mode is inherent to any appearance-based matching method operating under high yaw uncertainty.

J Extended Analysis

J.1 Yaw Error Distribution

Table 11 reports the 1/2/3/4/5-degree threshold accuracies together with the median and mean yaw error for MGL and KITTI, where threshold metrics are the standard summary for limited-FoV camera datasets. The 1/2/4° values differ slightly from the main table, recomputed after a later code update that left the method core unchanged. The same pattern holds on VIGOR, where the median yaw error is 1.94° (same-area) and 5.45° (cross-area), far below the mean of 35.7° and 45.5°. The gap stems from a front/back ambiguity: a column-aggregated road correspondence looks nearly identical 180° apart, so a small fraction of predictions flip and inflate the mean. Asymmetric cues such as building facades, or a temporal prior (speed, heading continuity) in navigation, resolve this binary ambiguity, making the median and threshold accuracy the representative indicators.

Table 11: Yaw threshold accuracy (%, ↑) and median/mean yaw error (°, ↓) for LAYS on MGL and KITTI under ±180° yaw uncertainty.

Split	Threshold accuracy (%) ↑					Error (°) ↓	
	$< 1°$	$< 2°$	$< 3°$	$< 4°$	$< 5°$	Median	Mean
MGL	34.81	52.42	58.78	61.74	63.29	1.79	44.26
KITTI (same-area)	51.07	62.47	64.06	64.64	64.75	0.96	46.17
KITTI (cross-area)	48.44	57.62	58.63	58.80	58.87	1.07	54.94

J.2 Matching-only Estimation

Our framework combines S_M with a learned pose-plausibility term S_P. To isolate the yaw signal carried by S_M, we set S_P to zero at every voting level of a trained LAYS model and re-run the forward pass.

On MGL dataset, the matching-only score at the GT location ranks the GT yaw bin in the top 2.5% of all yaw bins (mean rank percentile 0.926, median 0.975). The matching-only sub-degree accuracy at the GT location is 15.10%, $< 2°$ is 28.47%, $< 4°$ is 49.07%, above the strongest baselines in Table 1.

Training LAYS with S_P removed converges but is unstable: early matching scores are near-uniform across many 2D pose bins, so the softmax loss is dominated by correct-match-to-wrong-location votes before features become discriminative, reaching 14.18/25.84/44.25 within $1°/2°/4°$ on MGL ±180°. S_P therefore serves mainly as an early-training stabilizer, while radial voting carries the yaw signal.

K Discussion and Future Work

Our method has limitations in exact location estimation when used standalone, due to the formulation that adds a match score to all pixels. Combining line-aligning scoring with the conventional projection-based method has the potential to yield more accurate yaw and location measurements.

Further improving cross-view localization for outdoor AR and MR applications is a promising direction; however, it poses new challenges due to the significantly greater pose freedom and the rich surrounding environment. Most existing vehicle-based works assume that features are mapped to flat ground and roads in the vehicle dataset. While the MGL dataset provides a more varied environment, the less-structured scenes typical of AR and MR applications present significant opportunities for future exploration.

The 2D map-based localization [21, 35] and the cross-view localization have different advantages. The 2D map provides semantic information about the surroundings but does not provide exact visual cues. The BEV from the cross-view localization setup provides precise visual cues but lacks semantic meaning. The fusion of two modalities would guide a new direction to more challenging AR and MR localization.