OptimusLine: Consistent Road Line Detection Through Time

Paolo Cudrano*[1] Simone Mentasti*[1] Riccardo Erminio Filippo Cortelazzo*[2] Matteo Matteucci[1]

***Abstract*— In the field of autonomous vehicles, the detection of road line markings is a crucial yet versatile component. It provides real-time guidance for navigation and low-level vehicle control, while it also enables the generation of lane-level HD maps. These maps require high precision to provide low-level details to all future map users. At the same time, control-oriented detection pipelines require increased inference frequency and high robustness to be deployed on a safety-critical system. With this work, we present OptimusLine, a versatile line detection pipeline tackling with ease both scenarios. Built around a frame-by-frame transformer-based neural model operating in image segmentation, we show that OptimusLine achieves state-of-the-art performance and analyze its computational impact. To provide robustness to perturbations when deployed on an actual vehicle, OptimusLine introduces a scheme exploiting temporal links between consecutive frames. Enforcing temporal consistency on each new line prediction, OptimusLine can generate more robust line descriptions and produce an estimate of its prediction uncertainty.**

I. Introduction

The detection of road markings is highly significant for automated vehicles. The road lines convey real-time information crucial to the vehicle's navigation, as well as to its self-localization. Knowledge of the road line placement also allows the vehicle to predict the behavior of other road users, and is fundamental when constructing HD maps, which are commonly used by autonomous vehicles [1].

Being essential to multiple tasks, the requirements behind a line detection pipeline can be various and diverse. When mapping, the retrieved line information must be highly precise to respond to the demands of an HD map. When adopted to navigate the road or to control the vehicle [2], [3], instead, the detection must not only be performed at high frequency, but it must also provide guarantees on its robustness to perturbations and disturbances, as its output is fed into safety-critical control components.

Over the years, many approaches have been proposed to tackle the task of line detection [4]. Initial efforts primarily focused on traditional computer vision algorithms [5], [6], relying on geometric techniques and low-level pattern recognition. With the advent of deep learning, data-driven models took over the field thanks to their higher detection performance. Several works, such as [7], [8], adopted convolutional neural networks (CNNs) for their superior pattern recognition capabilities. These networks were typically trained in an image segmentation regime, and would thus predict the presence or absence of line markings in each image pixel.

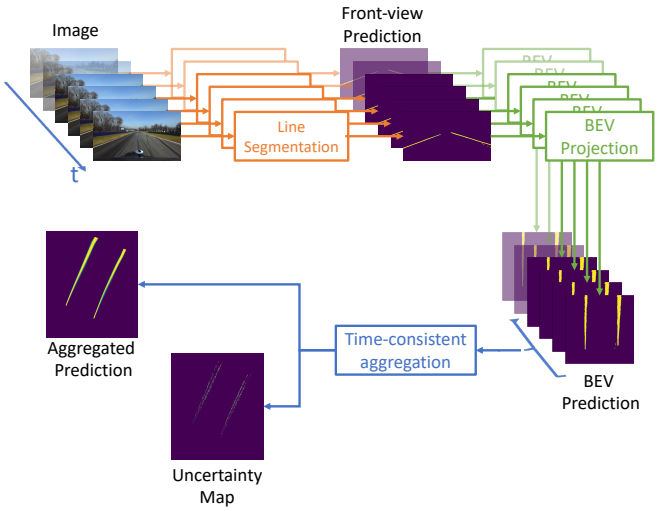


Fig. 1. Overview of OptimusLine. Every camera frame acquired by the vehicle's front camera is segmented independently through a SegFormer network. The resulting predictions are then projected into the Bird's Eye View (BEV) plane, which is isometric to the road surface. The last n BEV predictions are aligned according to the vehicle's corresponding trajectory, after which a per-pixel aggregation is performed, to obtain an enhanced prediction map with enforced temporal consistency. Additionally, a per-pixel uncertainty estimate can be computed at no extra cost. As the aggregation is efficiently performed incrementally at every new frame arrival, OptimusLine can provide enhanced BEV prediction and uncertainty map in an online fashion, as required by downstream mapping or control driving modules.

Recently, the deep learning community has seen a surge in the use of a more performant architecture, the transformer [9], which exploits the attention mechanism to better identify visual global patterns. To exploit this advancement, new line detection models have been recently developed [10], [11]. Early works in this direction take inspiration from the attention mechanism but rely on highly customized architectures to enforce inductive biases specific line detection [12]. Interestingly, such works moved from producing a full description of the lines, via image segmentation, to outputting only a compressed representation. In particular, some works only predict the coordinates of a few points belonging to the lines [13], while others provide a parametric line representation, such as a Bezier curve [8].

Although these representational choices might seem efficient, they highly constrain the capabilities of the network, limiting its generality and making it more prone to error. Indeed, parametric models rely on the choice of a fixed representation, which is typically hard to make a priori. Lines

*These authors contributed equally.

[1,2]Department of Electronics Information and Bioengineering, Politecnico di Milano, p.zza Leonardo da Vinci 32, Milan, Italy, name.surname@polimi.it[1], riccardoerminio.cortelazzo@mail.polimi.it[2]

This paper is supported by "Sustainable Mobility Center (Centro Nazionale per la Mobilità Sostenibile – CNMS)" project funded by the European Union NextGenerationEU program within the PNRR, Mission 4 Component 2 Investment 1.4.

are often represented as splines and, to limit the number of parameters, assumptions are made on the regularity of the curves and the parallelism among line markings [10]. These assumptions might break down when modeling complex urban scenarios or in the presence of lane splits and merges. Analogously, identifying only a limited number of line points poses the problem of undersampling, potentially missing important details of the line shape. Thus, to provide a finer and more suitable representation [3] to safety-critical control systems, a model fitting is still required, and the computational cost is only shifted to later AV stages.

For the above reasons, we argue that a segmentation mask remains a more favorable representation, as it can be computed quickly by deep models and has the advantage of retaining the same level of precision as the source data. In this way, it is possible to easily downsample it only when really needed (e.g., in real-time tasks) [3], while it can still be consumed entirely in situations requiring fine-grained details, such as HD mapping [14], [15]. Despite this fact, to the best of our knowledge, none of the state-of-the-art transformer-based models for road line detection performs image segmentation. In Sec. III-A, we explore this research direction, presenting a transformer model for frame-by-frame line segmentation that surpasses state-of-the-art performances while maintaining high-resolution information of all the lines and yet allowing for high-frequency operation.

Before deploying a line detection algorithm on an automated vehicle, its robustness should always be carefully addressed, especially when based on deep learning. In the line detection literature, however, robustness and consistency are seldom analyzed. The robustness of a pipeline is defined as its ability to function reasonably well even when affected by perturbations and disturbances, which are common in the context of autonomous vehicles (sensor noise, high dynamism of the external environment). One way to improve the robustness of a system is by ensuring that its output is consistent when perturbed, and that it remains consistent over time. In this sense, consistency is a crucial component for any autonomous driving pipeline. Thus, studying these two properties becomes necessary, especially as deep learning models are known to produce overconfident predictions and poorly judge their uncertainty [16]. To delve deeper into this issue, in Sec. III-B we present an efficient method to enhance our frame-by-frame pipeline with a time-consistency mechanism. We show that we can exploit time dependencies in the observed road segments to filter noise in each predicted frame, producing more robust line measurements over time.

We name the resulting pipeline, achieving precise and robust road line segmentation, OptimusLine (Fig. 1).

II. Related work

Earlier approaches to road line detection leveraged traditional computer vision algorithms [17], such as Hough Transform [18], Canny edge detection [19], and Laplacian of Gaussian (LoG) [20]. These methods, however, often falter in complex environments, failing to handle occlusions, changes in lighting conditions, and road surface variations. For this reason, deep learning approaches have gradually replaced them. Past works generally adopted a Convolutional Neural Network (CNN), with an encoder-decoder structure. The encoder is responsible for extracting multi-level feature maps from the input image, while the decoder interprets these features to predict information about the line. This information comes in various formats. [21], [22] produce a list of key points belonging to the lines, which can be interpolated at need to produce particular line models. Inspired by works in object detection, 3D-LaneNet [23] predicts lines in an anchor-based format, which also exploits a top-view road representation. Similarly, [24], [25] exploit a CNN to estimate the authenticity of pre-computed line candidates. 3D-SpLineNet [26], instead, directly produces a parametric representation of the lines, in the form of 3D splines, while Feng *et al*. [8] output Bezier splines.

An alternative often adopted is to treat the line detection problem as a segmentation task, predicting an image mask that indicates if a line is present in every image pixel. This approach has the advantage of producing a highly detailed per-pixel prediction of the line locations, while leaving to the downstream components the possibility to model and compress the line representation according to their needs, e.g., fast inference in real-time applications [3] versus high-precision in mapping tasks [14]. To produce a segmentation mask, the decoder upsamples the encoded features, and typically combines them with intermediate representations available to the encoder to recover spatial information, as done in U-Net [27]. Several examples of this approach can be found in the literature. [3] adopts a simplified U-Net architecture to segment the road lines from the image background at high frequency, and later fits them with a line model suitable for vehicle control. In LaneNet [28], authors perform a similar segmentation, but additionally train their pipeline to cluster separate lines, in order to aid their fitting method. Currently, HybridNets [29], YOLOP [7], and YOLOPv2 [30] are the most known models for line segmentation. HybridNets employs multiple parallel decoders to jointly segment road lines and drivable area, showing that this multi-task approach improves the line detection performance. YOLOP and YOLOPv2 are derivatives of YOLO [31] modified for panoptic driving perception. Similarly to HybridNets, they adopt a multi-task configuration, additionally performing drivable area segmentation and road object detection.

Recent advancements in deep learning have seen a shift from CNNs to the transformer model [9], which is now largely adopted also for vision tasks [32], including lane detection. As with CNNs, these models can be trained to output road lines in different formats: parametric models, line points, and segmentation masks. LSTR [10] combines a transformer with a Resnet encoder to directly predict line parameters in the image space. The authors define their parametrization in image coordinates to map to cubic polynomials in the road plane, while they assume parallelism for all lines to further compress their representation. LaneFormer [12], instead, applies an attention mechanism to output a set of point locations for each line marking.

Analogously, LATR [11] defines a specific transformer architecture to perform line detection in 3D world coordinates, producing a set of points belonging to each line. Differently from earlier works, they propose specific lane-aware queries in the transformer architecture, and integrate additional embeddings computed by iteratively estimating the ground plane. A similar point-based line format is produced also by PersFormer [13], which combines 2D and 3D line detection by learning to estimate the image homography between image and bird's eye view (BEV) plane. Despite the advances brought by transformers in the literature, to the best of the authors' knowledge, no work addresses the impact of transformers in road line segmentation. In this work, we bridge this gap. As seen above, producing a full segmentation mask of the image can significantly aid further stages of perception pipelines, both in the context of real-time control or offline mapping, by providing richer per-pixel predictions and thus leading to a more precise description of the road lines. We point out that, in previous works [3], it has been shown how a segmentation mask can be quickly converted into a set of points when needed for fast vehicle control. In this work, we additionally show that precise segmentation masks can also be generated when requiring fast inference.

Although the precision of the retrieved line markings is of primary importance, we argue that it is not the only factor to consider when deploying a perception system on a vehicle. As vehicles perform safety-critical operations, it is important to enforce a certain degree of robustness in each component. In this regard, assuring temporal consistency across consecutive frames provides a valuable source of contextual information that can significantly enhance the quality of the overall pipeline. Given the temporal continuity of video streams in typical driving scenarios, it is commonly assumed that road lines detected in consecutive frames exhibit high similarity. Under this assumption, various post-processing techniques have emerged, typically relying on filtering methods that involve a parametrization of the line [3]. A common approach is employing optical or scene flow, which estimates the pixel-wise motion between consecutive frames. By integrating these motion fields into the line detection process, the model can enhance the road lines' spatial accuracy and infer their dynamics, offering a more reliable basis for temporal interpolation and prediction [33]. Other techniques involve tracking algorithms, such as Kalman filters [34] or particle filters [35], which can maintain the identity and continuity of road lines across frames. These algorithms, however, can only be applied to a parametrized formulation of the line. Any parametrization, however, is significantly compressed with respect to a full segmentation mask, and thus implies a loss of information. In this work, we study whether it is possible to enforce a similar temporal consistency without a loss of low-level detail.

III. OptimusLine: transformer-based road line detection with time consistency

We present OptimusLine (Fig. 1), our pipeline for time-consistent road line segmentation. OptimusLine includes two components: a frame-by-frame deep segmentation model based on the transformer architecture, and a temporal aggregation module enforcing time-consistency between consecutive frames for improved precision and robustness.

A. Frame-by-frame segmentation

To perform an image-level segmentation of the road lines, we exploit a transformer-based architecture. Transformers [9] are deep learning models that have recently shown significant success across a range of natural language processing tasks [36]. Initially designed for sequence-to-sequence tasks, transformers have also proven effective when applied in computer vision, with adaptations encompassing image classification [32] and segmentation [37]. The transformer architecture used in vision (ViT) [32] finds its strength in the attention mechanism, which allows the model to capture global dependencies between different parts of its input. After dividing an image into patches and projecting them in an embedding space, the transformer processes all patch embeddings through multiple layers, each composed of two sub-modules: multi-head attention, which handles the exchange of information among patches, and a feed-forward MLP, which computes a non-linear function on the information aggregated by each token.

Training larger transformer models typically requires large data samples, thus the community often relies on models pretrained on preliminary tasks. Pretraining is a well-established technique and has been shown effective in the community [38]. In recent years, several adaptations of the classical ViT have been proposed to perform segmentation [37], [39]. Among these, SegFormer [40], relying on a hierarchical architecture and a simple decoder, has shown state-of-the-art performance. In this work, we finetune a pretrained SegFormer model for line marking segmentation. SegFormer originally comes in 5 variants (B0–B5), each with an increasing number of parameters, and thus higher accuracy at the cost of higher computational cost.

We consider variants B0 and B4, each suitable for a different scenario. B0, with only 3.7M parameters, is extremely lightweight and allows for the best computational efficiency in online scenarios, such as when controlling the vehicle. B4, instead, having about 17x more parameters (64.1M), could be challenged by real-time constraints but produces higher performance and is thus suitable for mapping tasks. We initialize our model with weights pretrained on the Cityscapes dataset [41]. This dataset contains urban street scenes annotated with classes related to the autonomous vehicle context, such as drivable road areas, static road elements, vehicles, and buildings. It does not contain annotations of the road markings, but the affinity between pretraining and downstream task typically leads to high transfer, as during pretraining the model had the chance to acquire features pertinent to the road environment.

To finetune this model towards segmenting road markings, we exploit an adaptation of the BDD100k line detection dataset we customly obtained. BDD100k [42] is a large-scale driving video dataset with pixel-level annotations specifically

collected for road features detection, such as road lines and drivable area. It contains 100k samples, each taken from a different video sequence recorded by different users at different times and locations, leading to a very diverse set of samples and thus encouraging model generalization. We adapt this dataset into a specialized subset designed only for line marking, and fixing a known unsolved issue in the original dataset. In particular, the original BDD100k does not provide annotations as pixel-level segmentation masks, but only provides an approximated polyline model of the edge between each road line and the asphalt (i.e, a polyline at the immediate left and right of each line marking, and not through the center of the line). At times, moreover, one of the line edges results missing, further complicating its usage. This representation hinders any task involving the detection of the line, as we typically require systems to identify its center and not its boundaries. In segmentation, additionally, we require image masks that cover the entire line width, which is hard to identify when one of the boundaries is not present. We address these shortcomings of BDD100k by processing the polyline representations and providing a multi-class segmentation mask for each originally annotated sample. We carefully preserve the line type categorization (solid vs dashed, white vs yellow) to allow for more detailed segmentation tasks. We also standardize the width of all line markings, automatically identifying the center of each line and enlarging it to a predefined, customizable width. This newly generated dataset enables us to train the SegFormer model to detect and segment line markings accurately, while we make our processing technique available to enable further research with our adapted dataset.[1]

We finetune using the Focal-Tversky loss function [43], which combines Focal loss [44] and Tversky [45] loss. The Focal loss helps to address class imbalance issues by assigning higher weights to difficult examples, while the Tversky component encourages the model to produce more precise segmentations by penalizing both false positives and false negatives. Let $\mathbf{P}$ be the predicted segmentation mask and $\mathbf{G}$ be the ground truth mask. The Tversky index is then defined as:

$$\text{Tversky}(\mathbf{P},\mathbf{G}) = \frac{\sum_i \mathbf{P}_i\mathbf{G}_i}{\sum_i \mathbf{P}_i\mathbf{G}_i + \alpha\sum_i \mathbf{P}_i(1-\mathbf{G}_i) + \beta\sum_i (1-\mathbf{P}_i)\mathbf{G}_i}, \quad (1)$$

where $\mathbf{P}_i$ and $\mathbf{G}_i$ are the i-th pixels of the predicted and ground truth masks, respectively. The parameters α and β control the balance between false positives and false negatives. The Focal loss, instead, assigns higher weights to difficult examples through a focusing parameter γ, which helps address class imbalance. The resulting Focal-Tversky loss is defined as:

$$\text{Focal-Tversky}(\mathbf{P},\mathbf{G}) = (1 - \text{Tversky}(\mathbf{P},\mathbf{G}))^{\gamma}. \quad (2)$$

A larger value of γ increases the penalty for mis-classifying difficult examples. During training, the overall loss function is computed as the average Focal-Tversky loss over all pixels in the image:

$$\text{Loss} = -\frac{1}{N}\sum_i \text{Focal-Tversky}(\mathbf{P}_i,\mathbf{G}_i), \quad (3)$$

where N represents the total number of image pixels. By utilizing the Focal-Tversky loss during finetuning, we guide our model to focus on challenging examples and produce more accurate and balanced line marking segmentations.

[1] https://github.com/AIRLab-POLIMI/BDD100k-SegmentationMasks

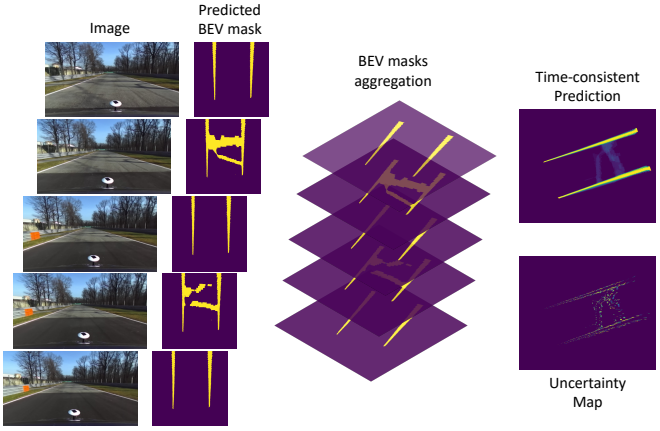
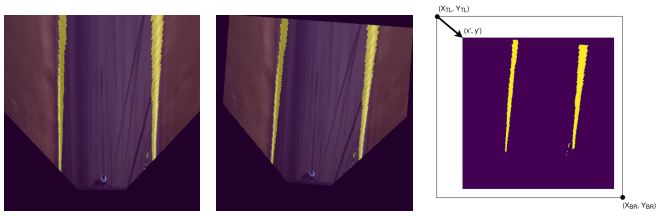


Fig. 2. Illustration of our time-consistency aggregation and uncertainty estimation. Even if spurious detections are collected on single frames, our robust aggregation discards them, providing an additional feedback on the road regions where it is most uncertain.

B. Time consistency and uncertainty estimation

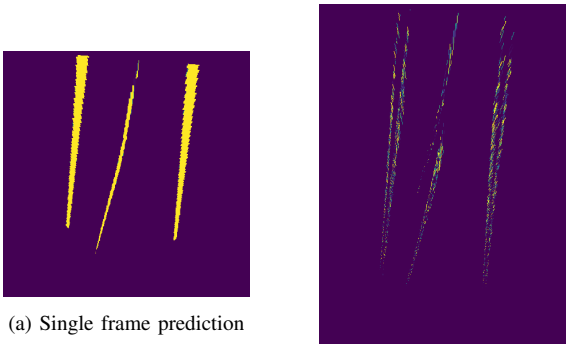
Self-driving vehicles use cameras to capture high-speed road images typically at 30 frames per second. These individual frames are then separately processed in real-time with models aiming to identify each road line, as discussed in Sec. II. However, at high frame rates, the same part of the road can be captured multiple times before the vehicle actually moves past it. A frame-by-frame approach can thus result in a loss of valuable information, as large portions of the scenes are discarded and recomputed repeatedly.

To exploit this information, instead, we aggregate the latest segmentation masks, in order to improve simultaneously the precision and temporal consistency of our predictions. An overview of our aggregation pipeline can be found in Fig. 2. To prevent perspective-based distortions, we perform the aggregation in the Bird's-Eye View (BEV) plane, which is isomorphic to the world ground plane and can be obtained through a simple calibration-based homography [46]. The use of the BEV plane is largely employed in the literature, especially when fitting a model to perform trajectory control [47], and is thus readily available. The prime benefit of aggregating the segmentation masks in the BEV-plane lies in the lack of nonlinear distortions due to the vehicle motion, which are instead present in the image-plane. This simplifies the aggregating algorithm, a crucial benefit for real-time performances. Upon acquisition of a frame f_t, we generate a segmentation mask m_t using our finetuned model, as detailed in Sec. III-A. Each segmentation mask m_t is subsequently projected into Bird's-Eye View (BEV) to produce a BEV mask b_t. This mask depicts the road



(a) b_t (b) r_t (c) Aligned r_t

Fig. 3. BEV mask alignment. (a) The segmentation mask m_t is projected onto the BEV plane to obtain (b_t). (b) b_t is rotated to match the global reference frame (r_t). (c) r_t is aligned to match the previous mask for aggregation.



(a) Single frame prediction (b) Uncertainty map

Fig. 4. The entropy over the cumulated predictions can be used as uncertainty measure of the model's predictions. In this example, the entropy assumes higher values on a spurious line in the center of the road, signaling that such line is a misdetection. Similarly, we register high uncertainty along the edges of the two line markings, as those pixels are changing most often. In the center of the road lines, instead, we see very low uncertainty, indicating that the network is confident about where the lines are centered.

surrounding the vehicle on a flattened plane, wherein each pixel equates to a small square of $0.05\,\mathrm{m}$ in the real world. We employ a resolution of $0.0025\,\mathrm{m}^2/\mathrm{pixel}$, which balances acceptable positional accuracy and computational power demands. With the vehicle in motion, the BEV masks obtained through several time steps represent slightly shifting ground regions. Crucially, at any given time t, the road ahead of the vehicle has been captured by the n preceding frames $t-n, ..., t-1$. Therefore, each associated BEV mask $b_{t-n}, ..., b_{t-1}$ contains some information about the same road segment. Assuming standard camera frame rates and vehicle speeds resulting in minor positional shifts between the n consecutive vehicular poses, we can postulate that the observed region is locally flat, thus all BEV masks computed lie on the same plane. We can then align and aggregate the latest $b_{t-n}, ..., b_t$ BEV masks given knowledge of the vehicle motion. The motion estimate is provided by a Real Time Kinematic GPS (RTK-GPS) sensor, typically present in modern autonomous vehicles. Vehicular odometry could also be used as a replacement when the GPS is not available. This process is done iteratively in real-time, maintaining the last $b_{t-n}, ..., b_t$ BEV masks in a fixed-capacity queue, together with their real-world spatial dimensions.

With this queue we can produce a single aggregated BEV mask B of the entire ground plane spanned during the past n time steps. To do so, the single BEV masks must be first aligned. Upon acquiring each BEV mask b_t (Fig. 3a), we apply a rotation to align the vehicle heading with ϕ our Est-Nord-Up (ENU) global world reference frame (Fig. 3b). We then determine the spatial location and dimension of this rotated BEV r_t in the world frame, exploiting knowledge of the vehicle's RTK-GPS location (x, y) (Fig. 3c). We can simply relate the vehicle center of mass to the BEV reference frame using the BEV image resolution, which in our case is fixed at $0.05\,\mathrm{m/pixel}$. Given all rotated masks $r_{t-n}, \ldots, r_t$, we can infer the dimensions of the aggregated mask B and allocate it efficiently. Finally, we can aggregate each rotated mask computing its location in B.

Once the different BEVs are effectively overlapped, it is possible to aggregate their prediction to obtain a final road line representation more consistent in time. We consider two aggregation strategies:

Prediction Average (PA): average of the output predictions of the network, i.e., average of the binary values that classify each pixel as a line or background.

Logit Average (LA): average of the logits returned by the network before the sigmoid layer. These are real values and have not yet been compressed to $[0, 1]$ by the sigmoid function; thus, they also capture more information about the network's confidence in its prediction.

After the aggregation, the resulting mask is thresholded to obtain a binary mask of the road line locations.

The output of this pipeline has the same shape and size as the single-frame predictions obtained with any SoTA segmentation model. However, our mask combines information sensed at subsequent time steps, thus enforcing an additional degree of time consistency in its predictions. Thanks to the way it is computed, it also does not introduce significant delays to downstream tasks, and can thus substitute frame-by-frame road line detectors in any line detection pipeline.

Since we obtain several aligned predictions for each pixel over time, we can further exploit them to compute an uncertainty measure for each pixel. We consider the output of the last sigmoid layer of the network as probability estimates, and use them to compute a per-pixel entropy:

$$H(x, y) = -\sum_i (p_i \log_2(p_i) + (1 - p_i) \log_2(1 - p_i)), \quad (4)$$

where p_i is the predicted probability for pixel (x, y) being a line marking at frame $i \in t - N, \ldots, t$. This measure indicates how inconsistent our network has been in estimating the presence of a line from different samples. H can thus be interpreted as an uncertainty map: higher entropy values indicate higher uncertainty in pixel x, y, while lower entropy values correspond to higher confidence. Fig. 4 shows an example of an uncertainty map and discusses its possible uses. By incorporating this uncertainty map in the downstream pipeline, it is possible to make more informed decisions and improve the robustness and reliability of the entire system, with negligible additional costs.

IV. Experimental validation

We validate the capabilities of OptimusLine in two steps. First, we evaluate with known benchmarks its performance

TABLE I

SINGLE FRAME SEGMENTATION PERFORMANCE

Model	Line IoU (%) ↑
Yolop [7]	49.70
YolopV2 [30]	53.24
HybridNets [29]	53.82
OptimusLine-B0 (ours)	53.63
OptimusLine-B4 (ours)	56.87

TABLE II

DISTANCE ERROR AND COVERAGE PERCENTAGE ON DIFFERENT SECTIONS OF THE MONZA ENI CIRCUIT

	Chicanes	Lesmo	Ascari	Parabolica
Single Frame	0.377 m 83.0 %	0.420 m 83.4 %	0.476 m 78.0 %	0.690 m 56.0 %
PA Aggregation	0.357 m 82.2 %	0.398 m 84.3 %	0.418 m 80.9 %	0.630 m 56.0 %
LA Aggregation	0.336 m 78.8 %	0.365 m 79.4 %	0.384 m 76.4 %	0.560 m 50.9 %

on the task of frame-by-frame line segmentation on front-view images (Sec. III-A). This allows us to compare our results with most of the literature, which only performs this type of validation. Then, we quantify the performance improvement introduced by our time consistency aggregation (Sec. III-B). To do so, we use streams of data instead of single frames, and we compare predictions directly in a ground reference frame, where there is no perspective distortion. Finally, in Section IV-C we assess the efficiency of the entire pipeline, required for deploying it on an autonomous vehicle.

A. Frame-by-frame segmentation

We evaluate the performance of our model on the BDD100k test set [42] and compare our performance against state-of-the-art models [7], [29], [30] in terms of Intersection over Union (IoU, also known as Jaccard index), which for binary segmentation is defined as:

$$IoU = \frac{Area\ of\ overlap}{Area\ of\ union} = \frac{TP}{TP + FP + FN}, \quad (5)$$

with TP, FP, and FN being respectively the true positives, false positives, and false negatives pixels, treating each pixel as a binary classification task. As shown in Tab. I, our model surpasses all current state-of-the-art models in its B4 version. Our B0 model is instead on par with concurrent methods, but its lightweight nature makes it more suitable for deployment on embedded systems. A detailed evaluation of the time efficiency of the pipelines is presented in Section IV-C.

B. Time consistency and uncertainty estimation

To evaluate the impact of our temporal consistency, we must rely on a dataset including continuous frame sequences with labeled line positions in a global reference frame. Most known datasets for line detection, however, do not satisfy these requirements. Therefore, we resort to the publicly available dataset presented in [3], which can offer this crucial information. This dataset comprises road images taken from

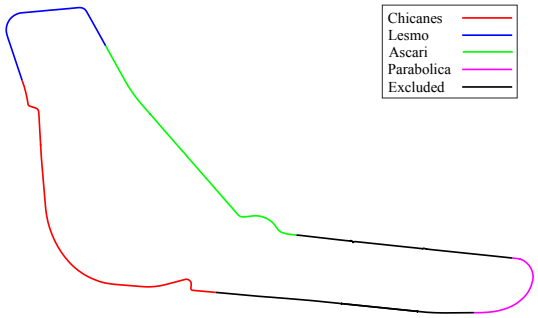
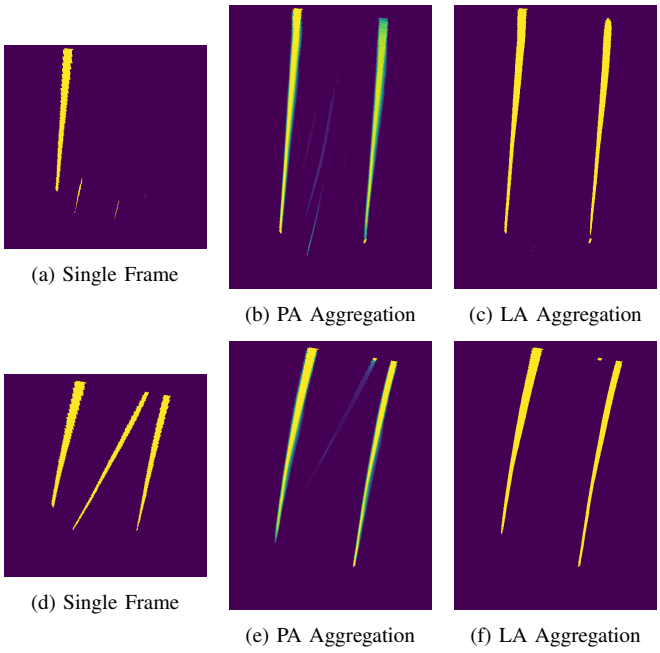


Fig. 5. Map of the Monza ENI circuit employed for the experimental validation, with different testing sections highlighted.

an instrumented vehicle, the corresponding RTK-GPS position of the vehicle, and the ground truth GPS annotations of each road line in the world frame. The recordings are collected on two Italian race tracks. For the validation of our pipeline, we employed the recordings acquired in the Monza ENI circuit (Fig. 5), analyzing our performance through different sections of the racetrack. Following [3], we compute the distance between each pixel predicted as a line by our pipeline and the actual line position (*distance error*, Dist). At the same time, we assess the proportion of ground truth points covered by at least one prediction (*coverage*, Cov). With these two metrics, we can evaluate whether our aggregation method provides predictions of a larger section of the track more robustly, while also improving its accuracy by filtering out noisy detection. It is important to notice that, even in the case of optimal prediction, the distance error Dist is always greater than zero. This happens as each line has a non-zero width (approximately $0.4\,\mathrm{m}$), while we compute the distance between each predicted pixel and the line center.

In Tab. II, we analyze Dist and Cov across four relevant sections of the Monza ENI Circuit, each presenting different curvature profiles and thus emulating different urban scenarios. We report the performance for single frame segmentation against the aggregation methods presented in Sec. III-B. We consider a window of $n = 30$ preceding frames, and report each metric as a temporal average across all frames in the section. The presented data underlines the added value of enforcing temporal consistency through our aggregation. LA Aggregation consistently ensures a significant reduction of the distance error, albeit not improving coverage. This approach is well suited for scenarios where our predictions are fed into a robust line-fitting algorithm, capable of interpolating between minor gaps in the data while benefitting from its more precise nature. Conversely, PA Aggregation leads to enhancements in coverage, accompained by slight distance error improvements. This renders this approach particularly advantageous when coverage is more impactful, as when mapping a large portion of road without requiring multiple trasversals. Fig. 6 provides a qualitative comparison of the different approaches in two challenging scenarios.

We point out that the improvements produced by our aggregation module are particularly valuable when the system encounters specific adverse conditions, such as challenging



(a) Single Frame (b) PA Aggregation (c) LA Aggregation

(d) Single Frame (e) PA Aggregation (f) LA Aggregation

Fig. 6. Examples and comparison of the aggregation algorithms. The first column shows the single-frame predictions without time consistency.

TABLE III
INFERENCE FREQUENCY OF OPTIMUSLINE ON A CONSUMER LAPTOP

Model	Single Frame	PA Aggregation	LA Aggregation
OptimusLine-B0	20.20 Hz	14.60 Hz	9.80 Hz
OptimusLine-B4	5.80 Hz	4.96 Hz	3.79 Hz

lighting conditions or occlusions. For this reason, their impact is limited when evaluated over long sequences, as most frames exhibit conditions manageable by the single frame-by-frame component. For instance, in Fig. 6a we realize the detection of the right line is challenging for the network, while the efficacy of our postprocessing is compellingly evident. Both PA and LA Aggregation robustly retrieve the undetected right line, integrating small amounts of information over time. Notably, LA maintains a conservative stance, highlighting only a slim line center, whereas PA tends to provide a more spread-out detection, especially at higher distances from the vehicle, yet covering the line more completely. Conversely, in Fig. 6d a spurious line is detected by the frame-by-frame model due to anomalies in the road pavement, but this detection is robustly discarded by our aggregations. PA significantly downweights this line, which can easily be removed through thresholding. Even more interestingly, this false positive line never even appears when using LA, showing how this method leads to more robust detections, at the expense of slightly lower coverage.

C. Time performance

To ensure that our system can be deployed on an actual vehicle, we finally assess the time performance of the entire OptimusLine pipeline. Table III shows the inference frequency of the different pipeline components. These tests were conducted on a consumer-grade laptop equipped with a modest GPU (Nvidia GTX 970) and a 3rd generation Intel-i7 processor. Consequently, we can consider these results as a lower bound, as potentially significant improvements can be achieved using specialized automotive hardware and GPU parallelization in the postprocessing phase. Specifically, OptimusLine-B0 performs frame-by-frame segmentation at 20 Hz, whereas with the larger variant OptimusLine-B4 the segmentation is done at 5 Hz on the Nvidia GTX 970. Enforcing time-consistency through aggregation introduces a minor decline in performance, attributable to the increased computational demand and especially noticeable when computing the logits average. Nevertheless, it is noteworthy that even on a consumer-grade device, the achieved inference frequency is sufficient for autonomous driving tasks, particularly when our smaller model is employed. We thus suggest the use of OptimusLine-B0 in safety-critical scenarios requiring high-frequency responses, leaving OptimusLine-B4 for more precise, low-frequency HD mapping tasks.

V. CONCLUSION

In this work, we presented OptimusLine, a time-consistent transformer-based line detection pipeline. OptimusLine is composed of two modules. A finetuned transformer performs line detection through image segmentation in a frame-by-frame manner. To train this model, we adapt a known large dataset for the task and share this construction for future research. Each frame-by-frame prediction is then processed by an aggregation module, where they are cumulated over a temporal window. Aligning all recent predictions in a single world reference frame, we can combine the information acquired over time into a refined prediction map, while also computing uncertainty estimates over each output pixel. This process effectively minimizes misdetections and reduces false positives, allowing for more robust deployment in safety-critical scenarios of the entire line detection pipeline.

ACKNOWLEDGMENT

The authors would like to thank Dr. Gianluca Bardaro (Politecnico di Milano) for his valuable insights.

REFERENCES

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

[2] J. Hu, S. Xiong, J. Zha, and C. Fu, "Lane detection and trajectory tracking control of autonomous vehicle based on model predictive control," *International journal of automotive technology*, vol. 21, pp. 285–295, 2020.

[3] P. Cudrano, S. Mentasti, M. Matteucci, M. Bersani, S. Arrigoni, and F. Cheli, "Advances in centerline estimation for autonomous lateral control," in *2020 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2020, pp. 1415–1422.

[4] S. P. Narote, P. N. Bhujbal, A. S. Narote, and D. M. Dhane, "A review of recent advances in lane detection and departure warning system," *Pattern Recognition*, vol. 73, pp. 216–234, 2018.

[5] Y. Wang, E. K. Teoh, and D. Shen, "Lane detection and tracking using b-snake," *Image and Vision computing*, vol. 22, pp. 269–280, 2004.

[6] J. H. Yoo, S.-W. Lee, S.-K. Park, and D. H. Kim, "A robust lane detection method based on vanishing point estimation using the relevance of line segments," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 12, pp. 3254–3266, 2017.

[7] D. Wu, M.-W. Liao, W.-T. Zhang, X.-G. Wang, X. Bai, W.-Q. Cheng, and W.-Y. Liu, "Yolop: You only look once for panoptic driving perception," *Machine Intelligence Research*, pp. 1–13, 2022.

[8] Z. Feng, S. Guo, X. Tan, K. Xu, M. Wang, and L. Ma, "Rethinking efficient lane detection via curve modeling," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 17 062–17 070.

[9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[10] R. Liu, Z. Yuan, T. Liu, and Z. Xiong, "End-to-end lane shape prediction with transformers," in *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, 2021, pp. 3694–3702.

[11] Y. Luo, C. Zheng, X. Yan, T. Kun, C. Zheng, S. Cui, and Z. Li, "Latr: 3d lane detection from monocular images with transformer," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 7941–7952.

[12] J. Han, X. Deng, X. Cai, Z. Yang, H. Xu, C. Xu, and X. Liang, "Laneformer: Object-aware row-column transformers for lane detection," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 1, 2022, pp. 799–807.

[13] L. Chen, C. Sima, Y. Li, Z. Zheng, J. Xu, X. Geng, H. Li, C. He, J. Shi, Y. Qiao, *et al.*, "Persformer: 3d lane detection via perspective transformer and the openlane benchmark," in *European Conference on Computer Vision*. Springer, 2022, pp. 550–567.

[14] M. Bellusci, P. Cudrano, S. Mentasti, R. E. F. Cortelazzo, and M. Matteucci, "Semantic interpretation of raw survey vehicle sensory data for lane-level hd map generation," *Robotics and Autonomous Systems*, vol. 172, p. 104513, 2024.

[15] P. Cudrano, B. Gallazzi, M. Frosi, S. Mentasti, and M. Matteucci, "Clothoid-based lane-level high-definition maps: Unifying sensing and control models," *IEEE Vehicular Technology Magazine*, vol. 17, no. 4, pp. 47–56, 2022.

[16] M. Mundt, Y. Hong, I. Pliushch, and V. Ramesh, "A wholistic view of continual learning with deep neural networks: Forgotten lessons and the bridge to active and open world learning," *Neural Networks*, vol. 160, pp. 306–336, 2023.

[17] J. M. Collado, C. Hilario, A. de la Escalera, and J. M. Armingol, "Adaptative road lanes detection and classification," in *Advanced Concepts for Intelligent Vision Systems: 8th International Conference, ACIVS 2006, Antwerp, Belgium, September 18-21, 2006. Proceedings 8*. Springer, 2006, pp. 1151–1162.

[18] Y. Otsuka, S. Muramatsu, H. Takenaga, Y. Kobayashi, and T. Monj, "Multitype lane markers recognition using local edge direction," in *Intelligent Vehicle Symposium, 2002. IEEE*, vol. 2, 2002.

[19] J. Son, H. Yoo, S. Kim, and K. Sohn, "Real-time illumination invariant lane detection for lane departure warning system," *Expert Systems with Applications*, vol. 42, no. 4, pp. 1816–1824, 2015.

[20] H. Kong, S. E. Sarma, and F. Tang, "Generalizing laplacian of gaussian filters for vanishing-point detection," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 1, pp. 408–418, 2012.

[21] W. Van Gansbeke, B. De Brabandere, D. Neven, M. Proesmans, and L. Van Gool, "End-to-end lane detection through differentiable least-squares fitting," in *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, 2019, pp. 0–0.

[22] Z. Chen, Q. Liu, and C. Lian, "Pointlanenet: Efficient end-to-end cnns for accurate real-time lane detection," in *2019 IEEE Intelligent Vehicles Symposium (IV)*, 2019, pp. 2563–2568.

[23] N. Garnett, R. Cohen, T. Pe'er, R. Lahav, and D. Levi, "3d-lanenet: end-to-end 3d multiple lane detection," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019.

[24] J. Li, X. Mei, D. Prokhorov, and D. Tao, "Deep neural network for structural prediction and lane detection in traffic scene," *IEEE transactions on neural networks and learning systems*, vol. 28, no. 3, pp. 690–703, 2016.

[25] B. He, R. Ai, Y. Yan, and X. Lang, "Accurate and robust lane detection based on dual-view convolutional neutral network," in *2016 IEEE intelligent vehicles symposium (IV)*. IEEE, 2016, pp. 1041–1046.

[26] M. Pittner, A. Condurache, and J. Janai, "3d-splinenet: 3d traffic line detection using parametric spline representations," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2023, pp. 602–611.

[27] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*. Springer, 2015, pp. 234–241.

[28] D. Neven, B. D. Brabandere, S. Georgoulis, M. Proesmans, and L. V. Gool, "Towards end-to-end lane detection: an instance segmentation approach," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, 2018.

[29] D. Vu, B. Ngo, and H. Phan, "Hybridnets: End-to-end perception network," *arXiv preprint arXiv:2203.09035*, 2022.

[30] C. Han, Q. Zhao, S. Zhang, Y. Chen, Z. Zhang, and J. Yuan, "Yolopv2: Better, faster, stronger for panoptic driving perception," *arXiv preprint arXiv:2208.11434*, 2022.

[31] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

[32] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," in *International Conference on Learning Representations (ICLR)*, 2021.

[33] G. Xing and Z. Zhu, "Lane and road marker semantic video segmentation using mask cropping and optical flow estimation," *Sensors*, vol. 21, no. 21, p. 7156, 2021.

[34] A. Borkar, M. Hayes, and M. T. Smith, "Robust lane detection and tracking with ransac and kalman filter," in *2009 16th IEEE International Conference on Image Processing (ICIP)*. IEEE, 2009.

[35] M. Nieto, A. Cortés, O. Otaegui, J. Arróspide, and L. Salgado, "Real-time lane tracking using rao-blackwellized particle filter," *Journal of Real-Time Image Processing*, vol. 11, pp. 179–191, 2016.

[36] A. Gillioz, J. Casas, E. Mugellini, and O. Abou Khaled, "Overview of the transformer-based models for nlp tasks," in *2020 15th Conference on Computer Science and Information Systems (FedCSIS)*. IEEE, 2020, pp. 179–183.

[37] R. Strudel, R. Garcia, I. Laptev, and C. Schmid, "Segmenter: Transformer for semantic segmentation," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 7262–7272.

[38] S. Khan, M. Naseer, M. Hayat, S. W. Zamir, F. S. Khan, and M. Shah, "Transformers in vision: A survey," *ACM computing surveys (CSUR)*, vol. 54, no. 10s, pp. 1–41, 2022.

[39] S. Zheng, J. Lu, H. Zhao, X. Zhu, Z. Luo, Y. Wang, Y. Fu, J. Feng, T. Xiang, P. H. Torr, *et al.*, "Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 6881–6890.

[40] E. Xie, W. Wang, Z. Yu, A. Anandkumar, J. M. Alvarez, and P. Luo, "Segformer: Simple and efficient design for semantic segmentation with transformers," *Advances in Neural Information Processing Systems*, vol. 34, pp. 12 077–12 090, 2021.

[41] M. Cordts, M. Omran, S. Ramos, T. Scharwächter, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset," in *CVPR Workshop on the Future of Datasets in Vision*, 2015.

[42] F. Yu, H. Chen, X. Wang, W. Xian, Y. Chen, F. Liu, V. Madhavan, and T. Darrell, "Bdd100k: A diverse driving dataset for heterogeneous multitask learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 2636–2645.

[43] N. Abraham and N. M. Khan, "A novel focal tversky loss function with improved attention u-net for lesion segmentation," in *2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019)*. IEEE, 2019, pp. 683–687.

[44] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2980–2988.

[45] S. S. M. Salehi, D. Erdogmus, and A. Gholipour, "Tversky loss function for image segmentation using 3d fully convolutional deep networks," in *Machine Learning in Medical Imaging: 8th International Workshop, MLMI 2017*. Springer, 2017, pp. 379–387.

[46] J. Du, S. Su, R. Fan, and Q. Chen, "Bird's eye view perception for autonomous driving," in *Autonomous Driving Perception: Fundamentals and Applications*. Springer Nature Singapore, 2023, pp. 323–356.

[47] M. Bersani, S. Mentasti, P. Cudrano, M. Vignati, M. Matteucci, and F. Cheli, "Robust vehicle pose estimation from vision and INS fusion," in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2020.