Supplementary Material for Vanishing-Point-Guided Video Semantic Segmentation of Driving Scenes

1. Vanishing Point Detection

We adopt a classical solution for vanishing point (VP) detection: Hough-transform [3] on Canny-edge [1] filtered images. The VP detection process is summarized in Algorithm 1. Given a gray-scale image x with height H and width W, we do the following to estimate the VP:

Pre-process. To make edge detection more robust, morphological transform with opening (erosion followed by dilation) [4] is first used to denoise input images. Canny edge filter is then implemented to get the edge map. As the "sky" area constitutes the top part of images, we only do Canny edge filtering for the bottom $2/3$ part of images. Next, Hough-transform is applied to the edges, achieving a set containing the lines detected. For each line ℓ in the set, we denote its slope $(\Delta H/\Delta W)$ as $\text{slope}(\ell)$.

Line selection. Once having the Hough-lines [3], we decide which lines are to retain or discard. We design a criterion based on the likelihood of the lines that could be near the VP. On the one hand, as VPs are normally located around the center of the image, we delete the lines that are more than $d_{\max} = 160$ pixels away from the image center. Besides, we find that most horizontal and vertical lines (*e.g.*, trees, wires) do not contribute to the VP detection. As a consequence, we delete any line ℓ from the Hough-line set that has $\text{slope}(\ell) \notin S$, where $S = (-5, -0.2) \cup (0.2, 5)$ is a pre-defined slope acceptance interval.

Cell vote. After removing undesired lines, we compute the line intersections between line pairs. Upon obtaining N_{line} lines after line selection, we would get $N_{\text{line}}(N_{\text{line}} - 1)/2$ intersections, notated as R. If the number of lines is too large, we randomly sample 100 lines among them. Next, we define several cells inside the image, where each cell is a rectangular box, and count the Hough-line intersections in it. In practice, we only parse cells in the lower part of the image, as the "sky" area takes the upper part of images. Finally, we choose the cell that includes the most intersections and return its center as the estimated VP position.

After obtaining the VP, it is still a problem to pass the VP position to the model. As cropping operations are used in the pre-processing pipeline, we crop the VP proximity map along with the input frame. The cropped input frame and proximity map are concatenated as our new input.

Algorithm 1 VP Detection

Require: gray-scale image $x \in [0, 255]^{H \times W}$, max central-to-line distance $d_{\max}$, slope acceptance interval S, square cells inside x centered at $c_i = (H_i, W_i), i = 1, ..., N_{\text{cell}}$ with size $L = \lfloor H/4 \rfloor$

$x \leftarrow \text{morphology\_opening}(x, \text{kernel} = \mathbf{I}_{5\times5})$
$x \leftarrow \text{Canny\_edge}(x, 50, 150, \text{apertureSize} = 3)$
$\text{lines} \leftarrow \text{Hough\_lines}(x, \rho = 1, \theta = \frac{\pi}{180}, \text{thres} = 200)$
$c \leftarrow (W/2, H/2)$
for $\ell \in \text{lines}$ **do**
  if $d(c, \ell) > d_{\max}$ or $\text{slope}(\ell) \notin S$ **then**
    Delete ℓ from lines
  end if
end for
$R = \text{find\_intersections}(\text{lines})$
for $i = 1, ..., N_{\text{cell}}$ **do**
  $n_i \leftarrow$ number hits of R inside cell i
end for
$i_{\text{opt}} = \arg\max_i n_i$
return $c_{i_{\text{opt}}}$

The Hough-transform-based VP detection proves fast and robust in automated driving scenarios. However, challenges arise when dealing with images featuring messy or unclear edges. For instance, cross street scenes may exhibit multiple VPs, while crowded pedestrian areas can introduce noisy lines, affecting VP detection accuracy. To address these issues, we will combine the hand-crafted VP detection method with deep learning to strike a better balance between accuracy and inference speed in future works.

2. Additional Ablation Studies

Effect of VP proximity embeddings. The linear VP proximity embedding is a VP-centered pseudo-depth map, where the depth of pixel (x, y) is $1 - \Delta D$, $\Delta D \propto \max\{\frac{|y-\hat{y}_j^p|}{H}, \frac{|x-\hat{x}_j^p|}{W}\}$ and $(\hat{x}_j^p, \hat{y}_j^p)$ is the VP pixel coordinate. Similarly, we introduce another two types of VP proximity maps: power and Euclidean decreasing (see Fig. 2).

- Linear (Fig. 2b): $\Delta D \propto \max\{\frac{\Delta y}{H}, \frac{\Delta x}{W}\}$
  In linear decreasing, the depth value of (x, y) is linearly

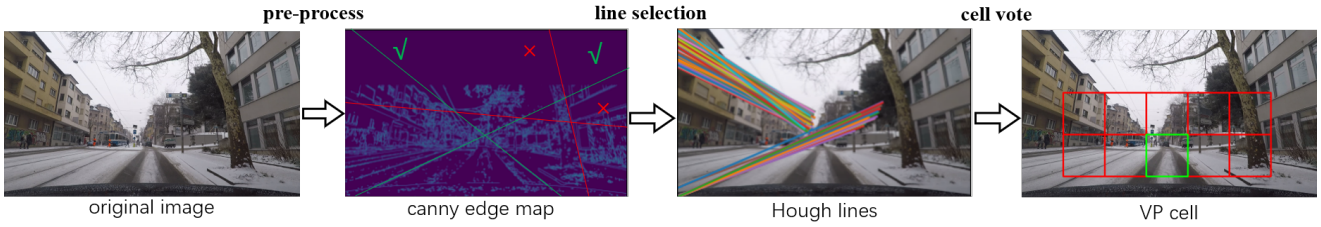
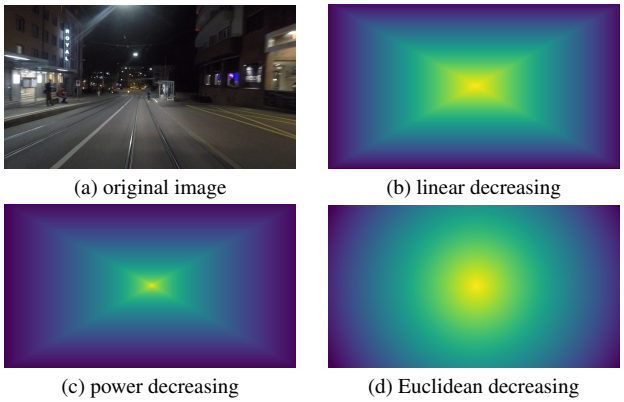


Figure 1. The VP detection pipeline. We first pre-process the input frame with morphology opening transform [4] and Canny edge filtering [1]. Hough-transform [3] is then applied and lines that do not contribute to VP detection are discarded. Finally, cell vote is implemented to count the intersections in each cell to determine the final VP position.



(a) original image (b) linear decreasing (c) power decreasing (d) Euclidean decreasing

Figure 2. Different types of VP proximity map embeddings. (a) represents the input frame, (b) is the VP proximity map with linear decreasing. Compared with linear decreasing, the depth value in our power decreasing map (c) drops much faster around the VP. (d) denotes the proximity map with Euclidean decreasing, where the image aspect ratio $\frac{H}{W}$ is not considered.

decreased according to its distance to the VP. It is fast to compute, and is our default option.

- Power (Fig. 2c): $\Delta D^2 \propto \max\{\frac{|\Delta y|}{H}, \frac{|\Delta x|}{W}\}$
  In power decreasing, the square of the depth value is linearly decreased. It is more concentrated, but the depth drops faster around the VP.
- Euclidean (Fig. 2d): $\Delta D \propto \sqrt{(\frac{|\Delta y|}{H})^2 + (\frac{|\Delta x|}{H})^2}$
  In Euclidean decreasing, the depth value is linearly decreased according to the Euclidean distance. It is circular and isotropic, but ignores the image aspect ratio $\frac{H}{W}$.

We study the impact of above-mentioned VP proximity embeddings in Tab. 1. Notably, VPSeg with linear VP proximity embedding achieves the highest mIoU and mIA-IoU for ACDC [5] and Cityscapes [2]. The experiments with power and Euclidean embeddings perform slightly worse. The possible reason is that the Euclidean decreasing does not consider the image aspect ratio $\frac{H}{W}$. And the depth value of power decreasing drops too fast around the VP.

Impact of the sampling coefficient Δd. We conduct experiments on different Δd in Tab. 2. We found that $\Delta d = 1$ adequately covers fast-moving targets with good performance. MotionVP with $\Delta d = 0$ only samples patches locally, which is unsuitable for high-speed driving scenarios and achieves worse mIoU and mIA-IoU. For $\Delta d > 1$, the performance of VPSeg drops drastically, proving that larger Δd is redundant for our VP-guided motion estimation.

Embeddings	mIoU (A.)↑	mIA-IoU (A.)↑	mIoU (C.)↑
Linear	**77.48**	**41.48**	**82.46**
Power	77.29	41.23	82.29
Euclidean	77.33	41.16	82.35

Table 1. Ablation study of different VP proximity embeddings on ACDC (A.) and Cityscapes (C.) with MiT-B3 [7] backbone.

Δd	mIoU (A.)↑	mIA-IoU (A.)↑	mIoU (C.)↑
0	76.74	40.57	81.83
1	**77.48**	**41.48**	**82.46**
2	77.12	41.01	82.23
3	76.88	40.65	81.79

Table 2. Ablation study of different sampling coefficients Δd on ACDC (A.) and Cityscapes (C.) with MiT-B3 [7] backbone.

Failure cases and improvement. In scenes like roundabouts and intersections we may have multiple vanishing points (VPs), no VP, or VPs at infinity. Fig. 3 depicts failure cases at roundabouts and intersections where unclear/wrong VPs result in minor errors near the detected VP. To address the concerns, we conduct ablation studies using the VP cell only for images for which the confidence of the detected VP surpasses a certain threshold (VP is used only for images in which it is clear) to improve segmentation. Tab. 3 and Fig. 3 show that with an appropriate threshold, we can filter out incorrect VPs images to avoid introducing irrelevant noise, improving performance.

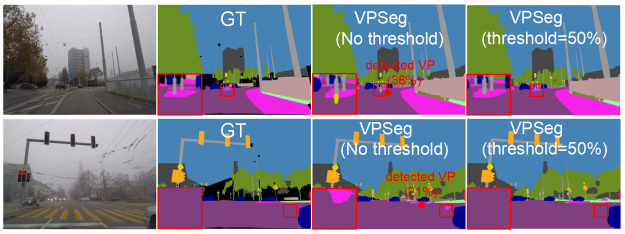


Figure 3. Failure cases and improvement with threshold filtering.

Thresholds	mIoU (ACDC)↑	mIoU (Cityscapes)↑
No threshold	77.48	82.46
40%	77.55(+0.07)	82.50(+0.04)
50%	**77.63(+0.15)**	**82.57(+0.11)**
60%	77.58(+0.10)	82.29(-0.17)

Table 3. Ablation studies of VPSeg on ACDC and Cityscapes with MiT-B3 backbone for different filtering thresholds.

Training Set	mIoU (ACDC)↑	mIoU (Cityscapes)↑
ACDC	77.48 (76.74 w/o MotionVP)	69.17 (67.47 w/o MotionVP)
Cityscapes	62.27 (59.33 w/o MotionVP)	82.46 (81.83 w/o MotionVP)
Both	**78.01** (77.31 w/o MotionVP)	**82.92** (82.35 w/o MotionVP)

Table 4. Ablation studies of VPSeg with MiT-B3 backbone under different cross-dataset settings.

Training Set	Iters (MotionVP)↓	Iters (w/o MotionVP)↓
ACDC	124k	140k
Cityscapes	144k	156k
Both	176k	188k

Table 5. Iterations to converge for VPSeg with MiT-B3 backbone.

Experiments across diverse datasets and training efficiency. Results in Tab. 4 show that MotionVP (VP-guided motion fusion) improves the generalization across ACDC and Cityscapes. Tab. 5 shows that VPSeg with MotionVP converges faster with higher training efficiency.

Re-using VP features with keyframes. In Tab. 6, we compare the results of the experiments with different settings for re-using VP features with keyframes. Specifically, we took one frame as a keyframe for every p frames. Then, we propagated the VP feature of each keyframe to the nearby p adjacent frames. Tab. 6 shows that the cost of the slightly increased FPS is a reduction in mIoU.

p	mIoU (A.)↑	mIoU (C.)↑	FPS↑
No keyframe	**77.48**	**82.46**	3.4
1	77.42	82.41	**3.5**
2	77.35	82.28	**3.5**
3	77.26	82.16	**3.5**

Table 6. Ablation studies of different keyframe intervals p of VPSeg on ACDC (A.) and Cityscapes (C.) with MiT-B3 backbone.

3. Detailed Pipelines

To exploit dynamic and static VP priors, we proposed MotionVP and DenseVP. MotionVP extracts dynamic context and can be divided into four parts: window partition and VP detection, direction assignment, patch sampling, and feature aggregation. DenseVP augments the dynamic context with finer attention around the VP region and consists of three steps: find VP patch index, select VP region, and generate dense features. The augmented dynamic context is sent to the prediction head for the final prediction. The details of MotionVP and DenseVP pipelines are shown in Fig. 4 and Fig. 5, while Tab. 7 explains types, domains, and meanings of the symbols from MotionVP and DenseVP.

References

[1] John Canny. A computational approach to edge detection. *IEEE TPAMI*, PAMI-8(6):679–698, 1986. 1, 2

[2] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *IEEE CVPR*, 2016. 2

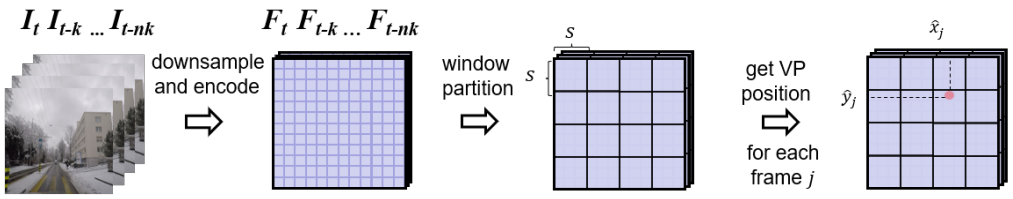
[3] Evelyne Lutton, Henri Maître, and Jaime Lopez Krahe. Contribution to the determination of vanishing points using hough transform. *IEEE TPAMI*, 16(4):430–438, 1994. 1, 2

[4] C. Jeremy Pye and J. A. Bangham. A fast algorithm for morphological erosion and dilation. In *EUSIPCO*, 1996. 1, 2

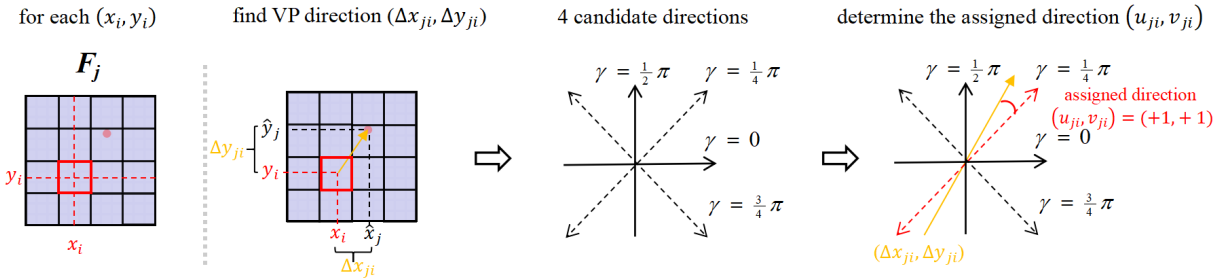
[5] Christos Sakaridis, Dengxin Dai, and Luc Van Gool. Acdc: The adverse conditions dataset with correspondences for semantic driving scene understanding. In *IEEE ICCV*, 2021. 2

[6] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017. 4

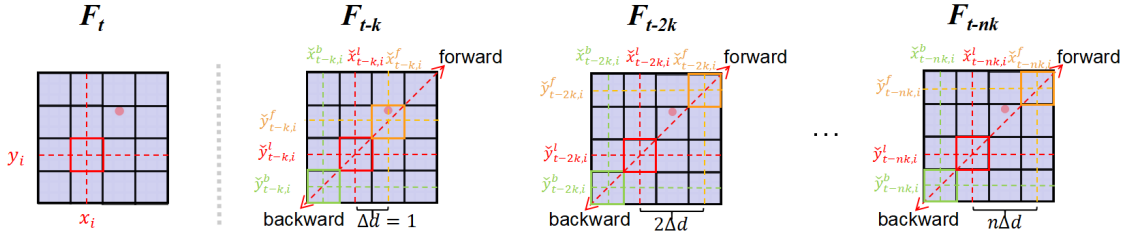
[7] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M Alvarez, and Ping Luo. SegFormer: Simple and efficient design for semantic segmentation with transformers. In *NeurIPS*, 2021. 2



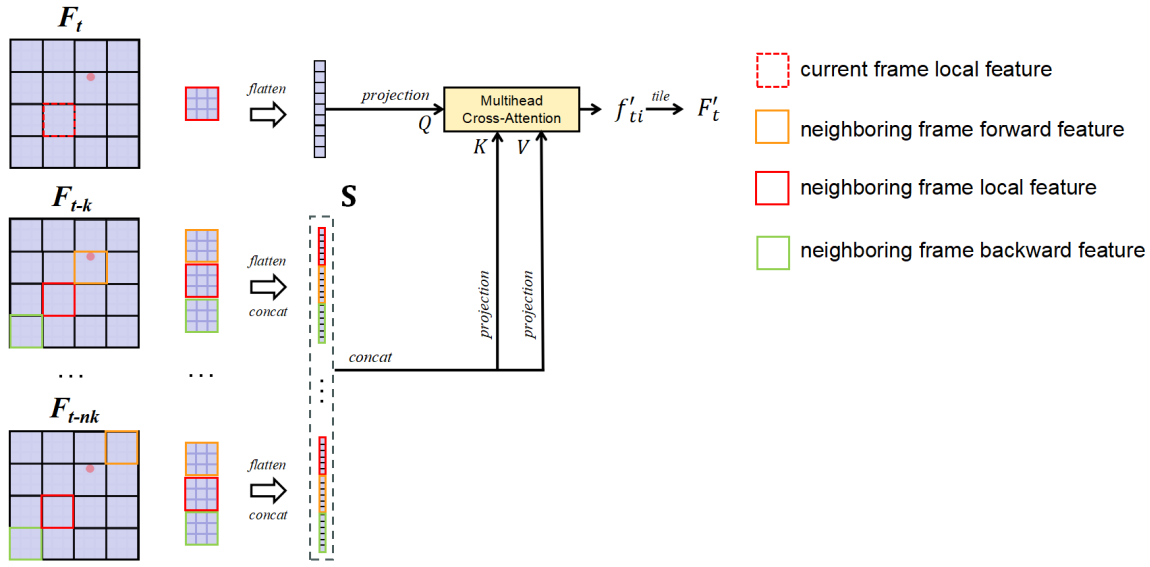
(a) **Window partition and VP detection:** given $n+1$ input video frames, we first extract feature maps with pre-trained transformer encoder. The feature maps are then subdivided into feature blocks of size $s \times s$ (indexed by i).



(b) **Direction assignment:** we determine the assigned direction for each patch (x_i, y_i). The assigned direction (u_{ji}, v_{ji}) is the closest candidate direction to vector $(\Delta x_{ji}, \Delta y_{ji})$, which points from the patch center to the VP.



(c) **Patch sampling:** after obtaining the assigned direction, we sample adjacent patches both forward and backward along the assigned direction. As the frame interval increases, the sampling distance also increases with the sampling coefficient Δd. But the sampled patches should not exceed the boundaries of the feature map.



(d) **Feature aggregation:** we generate dynamic context F'_t with cross-attention [6] operations. Specifically, for each patch (x_i, y_i), the patch features of the current frame serve as queries, while the sampled features in neighboring frames serve as keys and values. After cross-attention, we achieve patch-level dynamic features f'_t, which are then simply tiled together to reconstruct the complete frame-level dynamic context F'_t.

Figure 4. Detailed MotionVP pipeline.

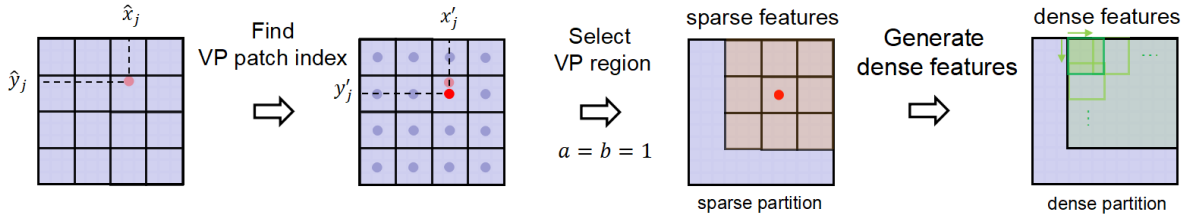


Figure 5. Detailed DenseVP pipeline. **Find VP patch index:** we find the closest patch to the VP as our VP patch. **Select VP region:** we select a rectangular region around the VP patch as our VP region $\mathbf{A}$. **Generate dense features:** the overlapping dense partition strategy is applied in the VP region, obtaining dense features $f_{\mathbf{A}}$.

Table 7. Table of symbols, their types, domains, and meanings.

Symbol	Type	Size (length)	Domain	Meaning
$\mathcal{I}$	set	$n+1$	-	a set of input frames
$\mathcal{T}$	set	$n+1$	-	a set of timestamps
$\mathcal{F}$	set	$n+1$	-	a set of feature maps
$\mathcal{D}$	set	$n+1$	-	a set of patch indexes
$\mathcal{A}$	set	$(2a+1)(2b+1)$	-	a set of sparse patch indexes of the VP region
$\mathcal{V}$	set	4	-	a set of vector representations of candidate directions
$\mathcal{S}$	set	$3n$	-	a set of sampled features in n neighboring frames
c	scalar	-	$\mathbb{N}$	number of feature channels
h, w	scalar	-	$\mathbb{N}$	spatial height/width of the feature map
H, W	scalar	-	$\mathbb{N}$	spatial height/width of the input frame
k	scalar	-	$\mathbb{N}$	frame sampling interval
K	scalar	-	$\mathbb{N}$	number of semantic classes
s	scalar	-	$\mathbb{N}$	size of the feature block
m	scalar	-	$\mathbb{N}$	number of dense patches in the VP region
Δd	scalar	-	$\mathbb{N}$	sampling coefficient
$(\hat{x}_j, \hat{y}_j)$	coordinate	2×1	$\mathbb{R}$	patch-level VP position in frame j
$(\hat{x}^p_j, \hat{y}^p_j)$	coordinate	2×1	$\mathbb{N}$	pixel-level VP position in frame j
(x_i, y_i)	coordinate	2×1	$\mathbb{N}$	index of the i-th patch
$(\check{x}^f_{ji}, \check{y}^f_{ji})$	coordinate	2×1	$\mathbb{N}$	forward sampled patch index for the i-th patch in frame j
$(\check{x}^b_{ji}, \check{y}^b_{ji})$	coordinate	2×1	$\mathbb{N}$	backward sampled patch index for the i-th patch in frame j
$(\check{x}^l_{ji}, \check{y}^l_{ji})$	coordinate	2×1	$\mathbb{N}$	locally sampled patch index for the i-th patch in frame j
(x'_j, y'_j)	coordinate	2×1	$\mathbb{N}$	VP patch index in frame j
I_t	matrix	$H \times W$	$\mathbb{R}$	frame in time t
F_t	matrix	$c \times h \times w$	$\mathbb{R}$	feature map for frame t
f_{ti}	matrix	$c \times s^2$	$\mathbb{R}$	patch-level feature for the i-th patch in frame t
F_{tl}, F_{th}	matrix	$c \times h \times w$	$\mathbb{R}$	low/high-resolution feature map of I_t
$\check{f}_{ji}$	matrix	$c \times 3s^2$	$\mathbb{R}$	sampled features for the i-th patch in frame j
F'_t	matrix	$c \times h \times w$	$\mathbb{R}$	frame-level dynamic features in frame t
f'_{ti}	matrix	$c \times s^2$	$\mathbb{R}$	patch-level dynamic features for the i-th patch in frame t
$f_{\mathcal{A}}$	matrix	$c \times ms^2$	$\mathbb{R}$	dense features of VP region
F''_t	matrix	$c \times h \times w$	$\mathbb{R}$	augmented dynamic context in frame t
E	matrix	$h \times w$	$\mathbb{R}$	VP proximity map
Q, Q_c	matrix	$c \times K$	$\mathbb{R}$	learnable/contextualized queries in CMA
F_m	matrix	$c \times K$	$\mathbb{R}$	the merged context
G_{nz}	matrix	$H \times W$	$\{0,1\}$	ground truth of the z-th class in the n-th image
P_{nz}	matrix	$H \times W$	$\{0,1\}$	prediction of the z-th class in the n-th image
M_n	matrix	$H \times W$	$\{0,1\}$	invalid mask of the n-th image