ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume XI-2-2026
XXV ISPRS Congress 2026 “From Imagery to Understanding”, Commission II, 4–11 July 2026, Toronto, Canada

Geometry-aided Video Panoptic Segmentation

Tuan Nguyen, Max Mehltretter, Franz Rottensteiner

Institute of Photogrammetry and GeoInformation, Leibniz University Hannover, Germany
(tuan.nguyen, mehltretter, rottensteiner)@ipi.uni-hannover.de

Keywords: Video Panoptic Segmentation, Optical Flow, Bounding Box Estimation, Kernel-based Panoptic Segmentation

Abstract

Video panoptic segmentation (VPS) unifies panoptic segmentation and object tracking by assigning each pixel a semantic class label, or for *thing* classes, an instance identifier that is consistent across frames. Addressing this task, we propose a novel online VPS method for processing stereoscopic image sequences, which is based on depth-aware kernel-based panoptic segmentation. Specifically, we introduce a geometrical constraint based on predicted bounding boxes into the segmentation of thing instances to overcome the fundamental limitation of kernel-based panoptic segmentation that only appearance information is considered in this step; this regularly leads to panoptic segmentation results in which distinct instances are erroneously merged into one mask. To link detected instances across frames, we propose to extend the commonly employed appearance-based association with a motion-related constraint based on optical flow; this resolves ambiguities in case of instances of similar appearance and, thus, reduces the number of incorrect associations. We experimentally evaluate our method on the publicly available Cityscapes-VPS dataset and compare our results to those of several related methods from the literature. The results demonstrate that our method improves the panoptic quality for a single frame and enhances the instance association across frames, leading to an overall improvement of 3.5% in Video Panoptic Quality on *thing* classes compared to the employed baseline.

1. Introduction

Panoptic segmentation (PS) delivers rich scene representations, predicting class and instance labels at pixel-level for instances of objects belonging to *thing* classes (e.g., *car*, *pedestrian*) while still providing class labels for *stuff* classes (e.g., *tree*, *sky*) (Kirillov et al., 2019b). Video panoptic segmentation (VPS) (Kim et al., 2020) extends PS by linking the results of consecutive frames in a video. The *stuff* classes might benefit from the additional temporal context, but most importantly, instances of *thing* classes are linked across frames, providing consistent identity labels for the same instance, so that the movement of individual instances can be tracked.

Most VPS approaches (Li et al., 2022; Shin et al., 2024; Kim et al., 2022) are built on top of kernel-based PS methods (Zhang et al., 2021; Yu et al., 2022; Wang et al., 2021): a deep neural network predicts a so-called kernel (also referred to as embedding or object query) per *stuff* class and per *thing* instance and a feature map encoding the input image. Each of these kernels is convolved with this feature map before applying a sigmoid activation to yield a binary map per kernel; this binary map indicates the area in the input image associated with a specific *stuff* class or *thing* instance. Contradictions across these maps are typically solved in a post-processing step, ensuring that each pixel in the input image belongs to exactly one *stuff* class or *thing* instance. However, the concept of kernel-based PS has a fundamental limitation, as illustrated in Figure 1: The binary masks of instances of the same *thing* class having a similar appearance are regularly overlapping, as mainly the appearance of instances is considered when convolving the corresponding kernel with the feature map, but not their geometrical characteristics. During post-processing this often leads to erroneously merged instance masks. Based on the outlined concept of simply linking the predicted instance masks across frames, such errors have a direct negative impact on the VPS quality, too. While Nguyen et al. (2024) propose a depth-aware differentiation between instances, their approach could only mitigate the problem. We follow a different direction and predict a bounding box alongside each instance-related kernel. This bounding box is used to limit the area in which the associated kernel is convolved with the feature map encoding the input image, effectively introducing a learned geometrical constraint regarding the extent of object instances.

Figure 1. Illustration of the effect of our bounding box-based constraint on the segmentation of *thing* instances. *Left:* A mask predicted by (Nguyen et al., 2024), in which two car instances of similar appearance are erroneously merged. *Right:* Our method resolves this problem by using geometric information. The red dots refer to the predicted centres of the segmented *car* instances.

To link *thing* instances across frames, most recent VPS methods rely solely on appearance information (Li et al., 2022; Zhou et al., 2024; Shin et al., 2024); often, the cosine similarity between the kernels of two instances detected in consecutive frames is computed to decide whether these instances show the same object. However, as regularly shown in the context of multi-object tracking, the combination of both, appearance and geometry-related information, typically works best in terms of minimizing ambiguities when linking object instances over time. Kim et al. (2020) use motion information in image space, i.e., an optical flow map, to transform the feature map extracted from a previous frame to the coordinate system of the current frame,

This contribution has been peer-reviewed. The double-blind peer-review was conducted on the basis of the full paper.
https://doi.org/10.5194/isprs-annals-XI-2-2026-321-2026

before fusing the feature maps of both frames. Thus, the prediction of masks for *stuff* classes and *thing* instances can make use of temporal relations between spatial information in consecutive frames. However, the geometric information is only used implicitly, but not for actually linking instances over time. To overcome this limitation, we extend this approach and further use the optical flow map to transform the instance masks predicted for the previous frame to the coordinate system of the current frame. By computing the Intersection over Union between the masks of the previous frame thus transformed and those from the current frame, we obtain a geometric cue which we use together with appearance information for linking object instances across frames.

In summary, this paper addresses the task of VPS from stereoscopic image sequences. Focusing on the outlined limitations of current kernel-based PS methods and on integrating geometric information into the association of *thing* instances over time, we aim to improve both, panoptic and video panoptic segmentation quality. The main contributions of this work are:

- We present a new VPS method to process stereoscopic image sequences in an online manner.
- We extend kernel-based PS by introducing a geometrical constraint: we estimate a bounding box per *thing* instance and limit the spatial extent of such an instance to the pixels enclosed by this bounding box.
- We adapt the association step of VPS by considering motion information, in combination with appearance information, when linking *thing* instances across frames.

2. Related Work

Panoptic Segmentation: As outlined before, most VPS approaches build on top of a PS method, achieving their goals by extending PS by an association step to link *thing* instance across frames. There are different approaches for PS, e.g., top-down (Xiong et al., 2019; Li et al., 2019), bottom-up (Cheng et al., 2020) and kernel-based PS techniques. (Li et al., 2021; Zhang et al., 2021). The latter are commonly the preferred choice as a basis for VPS; this is due to the unified representation for *thing* and *stuff* classes and the reduced need for post-processing. However, by design, kernel-based PS methods tend to merge the segmentation masks of distinct instances of similar appearance: When computing the segmentation masks by convolving an instance-related kernel with the feature map encoding the input image and when deciding whether two instances actually refer to the same object and their segmentation masks should thus be merged during post-processing, only the appearance of these instances is considered, but neither their positions in image space nor their extents. Nguyen et al. (2024) attempt to overcome this limitation by introducing depth information obtained via stereoscopic image matching. Specifically, the depth is used in two ways: as additional input alongside the colour image and in a depth-aware Dice loss function, which penalizes cases in which distinct *thing* instances at different distances from the image plane are merged. While this approach could mitigate the described problem, it did not solve it; in particular, distinct instances of similar appearance and distance to the image plane remain problematic. In the present work, we follow a different approach and propose to explicitly introduce geometric information into the computation of the segmentation masks in kernel-based PS. Specifically, we limit the area in which a kernel is convolved with the feature map based on a predicted bounding box.

Video Panoptic Segmentation: To proceed from PS, performed separately for each individual frame, to VPS, most methods follow an online approach and aim to link instances detected in the immediately preceding frame to those detected in the current frame, propagating unique identifiers. Kim et al. (2020) extend the top-down PS method UPSNet (Xiong et al., 2019) by using an estimated optical flow map to transform features extracted from the previous frame to the coordinate system of the current frame, before combining them with features extracted from the current frame. The combined features are used to compute a segmentation mask per *stuff* class and *thing* instance and an appearance embedding per instance. This embedding is optimized with a tracking and a contrastive loss, allowing to distinguish between distinct instances and to link instances from both frames showing the same object based on their related appearance embeddings. Video K-net (Li et al., 2022) extends the kernel-based PS method of Zhang et al. (2021) by adapting the attention mechanism (Vaswani et al., 2017). Specifically, the kernels are not predicted in a single step, but are iteratively improved; the resulting intermediate kernels from consecutive frames are fused via cross-kernel attention, ensuring that kernels associated with the same object are similar across frames. This enables the association of instances across frames using the cosine similarity computed on the related final kernels. Zhou et al. (2024) focus on the emergence and disappearance of instances in an image sequence, arguing that such instances are commonly erroneously associated because association is often enforced although there is no corresponding instance in the adjacent frame. The authors state that both cases, the emergence and disappearance of instances, corresponds to the association of foreground and background embeddings, i.e., the association of feature vectors that represent visible instances and others that represent instances that have not been seen so far using some kind of default feature vector, respectively. However, existing methods derive instance-related kernels solely from foreground embeddings, which creates a discrepancy between these two kinds of embeddings, hindering a reliable association. To reduce this discrepancy, Zhou et al. (2024) enable the kernels to retrieve information from both, foreground and background embeddings, through an attention-based scheme (Vaswani et al., 2017). In summary, all methods discussed so far solely consider appearance-related information to link *thing* instances across consecutive frames. However, neglecting the geometric properties regularly leads to ambiguities and incorrect associations if multiple similar object instances are observed.

Aiming to overcome this limitation, Shin et al. (2024) consider both, the appearance and the spatial position in image space of an instance, when linking instances across frames. For this purpose, a cross-attention approach (Yu et al., 2022) based on k-means clustering is used to group pixels belonging to the same instance, computing an appearance and a position-related embedding per instance from the feature vectors and the coordinates corresponding to these pixels, respectively. Based on the pair-wise dissimilarity between embeddings of instances from different frames, the Hungarian algorithm (Kuhn, 1955) is used to obtain a globally optimal association of instances across consecutive frames. However, using the spatial position in image space for association introduces the assumption that instances are subject to only minor movements across frames. This assumption is only valid if the frame rate of the image sequence

is high in relation to the observed motion; in practice, this assumption is regularly violated, because both, the observed *thing* instances and the camera, are moving. In contrast, the present work focuses on the incorporation of motion information using optical flow into the association step and thus explicitly models larger movements of instances across frames in image space.

3. Background: Kernel-based Panoptic Segmentation

We start with a brief review of kernel-based PS (Li et al., 2021) with its extension for considering stereo images as input (Nguyen et al., 2024). The latter forms the basis of our new method for VPS.

The goal of the PS of a single colour image $\mathcal{I}^c \in R^{H\times W\times 3}$ of extent $H \times W$ is to assign each of its pixels either to one of K^{st} *stuff* classes or to one instance of one of the N^{th} *thing* classes. Li et al. (2021) propose a kernel-based approach to achieve this goal. First, a Feature Pyramid Network (FPN) with Resnet50 backbone (Lin et al., 2017) is used as an encoder that extracts a multi-scale feature map from the image. A subsequent network branch, the *Feature Encoder*, converts the output of the FPN into a feature map $\mathcal{F}$ of dimension $H/4 \times W/4 \times C_e$ using a module proposed for that purpose in (Kirillov et al., 2019a); C_e is a hyperparameter.

The hierarchical FPN output also forms the input for another network branch, the *Kernel Generator*. The Kernel Generator predicts one weight vector (called *kernel*) for each of the *stuff* classes and one such kernel per detected instance of any of the *thing* classes. Instance detection is based on detecting the centres of instances similarly to (Zhou et al., 2019). Thus, altogether there will be $K^{st} + K^{th}$ such kernels, each associated with a class label also predicted by the Kernel Generator; here, K^{th} represents the total number of *thing* instances.

Each of these kernels is convolved with the output of the feature encoder. After applying a sigmoid activation to the outputs, upsampling by a factor of 4 and applying a threshold to the resultant score maps, $K^{st} + K^{th}$ binary maps are obtained at the original resolution of the input. The first K^{st} maps identify pixels assigned to the corresponding *stuff* classes, whereas each of the other masks identifies pixels assigned to a specific instance of one of the *thing* classes. Finally, post-processing is applied to remove contradictions between the individual label maps (Kirillov et al., 2019b). The resultant masks along with the class labels represent the final output of PS. Training is based on the minimization of a loss function consisting of two terms: one term is related to the output of the network branch inside the Kernel Generator that predicts the instance centres and the pixels associated with the *stuff* classes (*Position Head*), and the second one is a Dice loss (Milletari et al., 2016) for comparing the predicted masks to a reference.

Nguyen et al. (2024) extend this method, using a depth map $\mathcal{I}^d \in R^{H\times W}$ generated from stereo matching as a second input. For that purpose, a second FPN encoder is trained to process the depth map, and the resultant hierarchical feature map is fused with the one generated from the input colour image. The fused feature map is processed in the same way as in (Li et al., 2021). Nguyen et al. (2024) also propose an additional depth-aware loss designed to mitigate the problem of merged instances, but as shown in the left part of Fig. 1, it did not solve it completely.

4. Methodology

4.1 Overview

Our new stereo-based online VPS method is based on the one described in Section 3. The architecture is shown in Fig. 2. At any time t, we assume a pair of colour images to be available, acquired simultaneously by two synchronized cameras. Dense matching is applied to generate a depth map (we use (Lipson et al., 2021) in our experiments), and in the following steps, the data at time t are represented by the reference colour image $\mathcal{I}^c_t$ and the depth map $\mathcal{I}^d_t$ available in the same coordinate frame. The network will not only process the image and the depth map acquired at time t, but also the corresponding data acquired at the previous time step, $\mathcal{I}^c_{t-1}$ and $\mathcal{I}^d_{t-1}$. The movement of the cameras and the observed objects is considered by an image $\mathcal{O}_t$ representing dense optical flow between $\mathcal{I}^c_{t-1}$ and $\mathcal{I}^c_t$, which serves as an additional input (we use (Teed and Deng, 2020) to generate $\mathcal{O}_t$ in our experiments). Finally, a list of tracked *thing* instances are also required.

The output at time t is similar to the one of standard PS as described in Section 3. On the one hand, it consists of K^{st} binary masks $\mathcal{M}^{st}_{t,j}$, each indicating which pixels are assigned to the *stuff* class j, while on the other hand it consists of the updated list of tracked *thing* instances that will form the input to the network at epoch $t + 1$. Each *thing* instance i is represented by a binary mask $\mathcal{M}^{th}_{t,i}$, a class label c_i, the kernel $\mathcal{K}^{th}_{t,i}$ that was used to generate the mask, and a unique identifier ID_i of that instance. All of the mentioned masks are those obtained after post-processing; they can be used to generate an RGB image in which every RGB-triple encodes both the class labels of all object types and the unique identifiers of the *thing* instances (Kirillov et al., 2019b)[1]. The difference to the representation of the output described in Section 3 is that each instance is associated with its identifier ID_i; this identifier will be identical for the same instance at different time steps and, thus, allows to infer the trajectory of an instance over time. The kernels $\mathcal{K}^{th}_{t,i}$ are required for associating instances over time.

At time t, the network as presented in Fig. 2 first processes the image and the depth map of each epoch (t and $t-1$) using a shared encoder. The encoder is the one of Nguyen et al. (2024) and fuses the output of two branches (cf. Section 3). The resultant hierarchical feature maps P_t and P_{t-1} are combined by a temporal fusion module adapted from (Kim et al., 2020), taking into account the optical flow image $\mathcal{O}_t$. In this way, the PS at time t can benefit from the information available at the previous timestep. This fusion block generates a joint hierarchical feature map $P_{t,f}$ having the same structure as the two input feature maps; it is described in Section 4.2.

The fused feature map $P_{t,f}$ serves as the input to the Feature Encoder and the Kernel Generator. We use the Feature Encoder of (Li et al., 2021), but propose to extend the Kernel Generator of (Li et al., 2021) by the *BBox Estimation Head*, an additional branch for predicting a bounding box $BB_{t,i}$ for every *thing* instance i in order to overcome the problem of merged instances (cf. Fig. 1). These bounding boxes are used in the generation of the instance masks, restricting the convolution of each instance kernel to the area indicated by the corresponding bounding box. The masks are post-processed as described in Li et al. (2021)

[1] https://github.com/cocodataset/panopticapi/tree/master/panopticapi (accessed 15/11/2025).

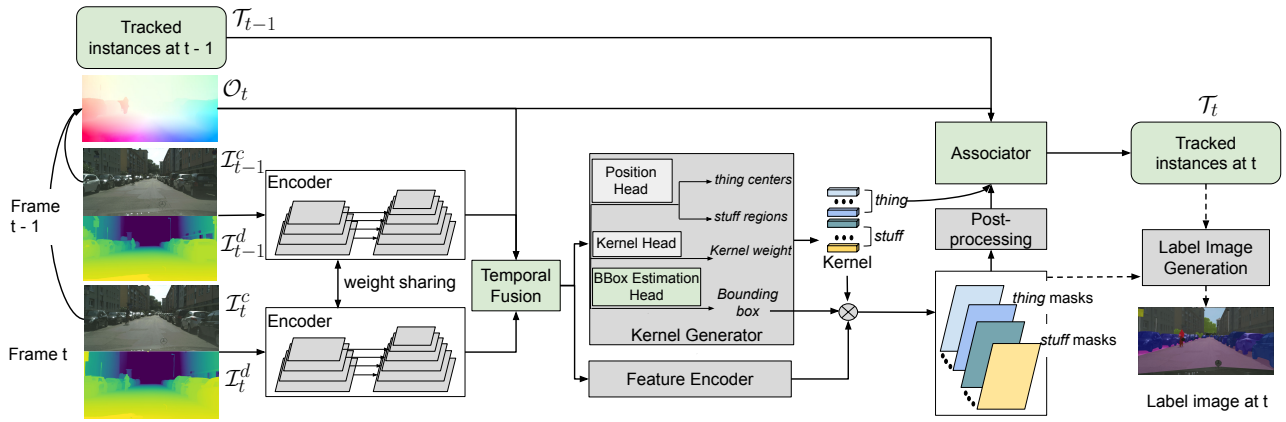

Figure 2. Given a colour image and a depth map for the current and the previous frame, an optical flow map and a list of instances tracked up to the previous frame, our method performs a PS of the current frame and updates the list of tracked instances by linking instances from the list of tracked instances to those detected in the current frame. The grey boxes represent the method we use as baseline (Nguyen et al., 2024). The green boxes present our extensions to this baseline, focusing on incorporating temporal information into the encoder (*Temporal Fusion*), the computation of the segmentation masks (*BBox Estimation Head*) and the association of *thing* instances across frames (*Associator*). *Label Image Generation* represents an optional processing step (thus the dashed arrows), and is used to visualize the result and for evaluation purposes.

(cf. Section 3). The Kernel Generator and mask prediction are presented in Section 4.3.

The last step of VPS is to link the instances detected at time t to those in the list of tracked instances, thus defining the identifiers ID_i of these instances. This is done by an *Associator*, which also considers the optical flow image $\mathcal{O}_t$. The Associator also updates the list of tracked instances, so that it can serve as input for the next epoch $t+1$. It is described in Section 4.4.

At time $t = 0$, the image and the depth map of the previous time step are not available. In this case, $\mathcal{I}^c_0$ and $\mathcal{I}^d_0$ are used to represent the previous timestep also, i.e., they are processed twice; the optical flow image is initialized by zeroes. The list of tracked objects will be empty.

The training procedure follows the one described in (Nguyen et al., 2024) as outlined in Section 3, but requires an additional loss term to train the BBox Estimation Head. The training procedure is described in Section 4.5.

4.2 Temporal Fusion

As outlined in Section 4.1, the temporal fusion module combines the hierarchical feature maps P_{t-1} and P_t generated by the encoder based on the images and depth maps of two timesteps and generates a joint hierarchical feature map $P_{t,f}$, considering the optical flow image $\mathcal{O}_t$. Each hierarchical input feature map contains feature maps at different scales $k \in [2 \dots 7]$, each with a resolution smaller than the input by a factor of $1/2^k$; these scale-specific maps are denoted by p^k_{t-1}, p^k_t and $p^k_{t,f}$ for the two input timesteps and the output, respectively.

The temporal fusion module is inspired by (Kim et al., 2020). First, all scale-specific feature maps of both time steps are upsampled to the resolution corresponding to $k = 2$ using nearest neighbour interpolation, and one average feature map is determined per timestep by averaging the corresponding upsampled maps. This results in two feature maps $\bar{p}_{t-1,o}$, $\bar{p}_t$ that combine information from all scales. To enable a pixel-wise exchange of information between the two timesteps, the optical flow $\mathcal{O}_t$

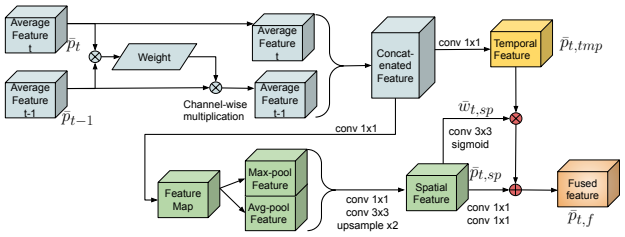

Figure 3. The temporal fusion module inspired by (Kim et al., 2020). Green blocks are related to spatial information, the yellow block is related to temporal information.

is used to transform the feature map $\bar{p}_{t-1,o}$ into the coordinates system of $\bar{p}_t$, yielding a co-aligned feature map $\bar{p}_{t-1}$. The architecture of the block used to fuse the two aligned feature maps $\bar{p}_{t-1}$ and $\bar{p}_t$ is shown in Fig. 3.

First, a weight matrix w is determined that has the same spatial dimensions as $\bar{p}_t$. The element $w(x, y)$ at position x, y is determined according to

$$w(x, y) = \sigma\left(\bar{p}_{t-1}(x, y) \cdot \bar{p}_t(x, y)\right), \tag{1}$$

where $\bar{p}_{t-1}(x, y) \cdot \bar{p}_t(x, y)$ represents the inner product of the two feature vectors at position (x, y) in $\bar{p}_{t-1}$ and $\bar{p}_t$, respectively, and $\sigma(\cdot)$ denotes the logistic sigmoid function. The weight encodes the relevance of the information from time $t-1$ for time t, which is large if the corresponding feature vectors are similar. Every feature vector $\bar{p}_{t-1}(x, y)$ in $\bar{p}_{t-1}$ is multiplied with the corresponding weight $w(x, y)$, and the resultant weighted feature map is concatenated with $\bar{p}_t$. On the one hand, a 1×1 convolution is applied to reduce the feature dimension of the concatenated tensor to the one of $\bar{p}_t$; the result $\bar{p}_{t,tmp}$ is considered to encode temporal context. On the other hand, the concatenated features are processed by another 1×1 convolution with the same purpose. Then, 3×3 max pooling and 3×3 average pooling are applied, both with stride 2, and the results are concatenated. After reducing the feature dimension by another 1×1 convolution, a 3×3 convolution is applied, and the

resultant feature map is upsampled by a factor of 2, yielding a result $\bar{p}_{t,sp}$ with the same dimensions as $\bar{p}_t$ that is considered to encode local spatial context.

Next, the two vectors $\bar{p}_{t,tmp}$ and $\bar{p}_{t,sp}$ are fused. For that purpose, a 3×3 convolution is applied to $\bar{p}_{t,sp}$, and the sigmoid function is applied to every element of the resultant tensor. The output $\bar{w}_{t,sp}$ is supposed to contain weights for $\bar{p}_{t,tmp}$. Also, two 1×1 convolutional layers are applied to $\bar{p}_{t,sp}$, and the result is added to the element-wise product $\bar{w}_{t,sp} \circ \bar{p}_{t,tmp}$, yielding a fused tensor $\bar{p}_{t,f}$ that is supposed to embed both spatial and temporal context across all scales. Finally, $\bar{p}_{t,f}$ is downsampled by a factor of 2 multiple times, resulting in multiple feature maps $\bar{p}^k_{t,f}$ with $k \in [2 \dots 7]$. These feature maps are added to the corresponding input feature maps p^k_t associated with time t (Kim et al., 2020). As a result, we obtain a hierarchical feature map $P_{f,t}$ that represents the final output of the temporal fusion module and serves as input to the subsequent network branches.

4.3 Kernel Generator and Mask Prediction

In Section 3, the Kernel Generator and the generation of the output masks proposed by (Li et al., 2021) were outlined. This workflow resulted in the problem indicated in Fig. 1: similar instances of a class are frequently merged. The depth-aware training procedure of Nguyen et al. (2024) could only mitigate this problem. An inspection of intermediate results of the original method revealed that the problem was not instance detection; in the example in Fig. 1, both instances were originally detected and masks for both instances were determined. However, both masks did not coincide well with the instance borders, and in the ad-hoc post-processing step used in (Li et al., 2021), the mask of the instance for which the prediction of the position head has a higher confidence always superseded the one of the second instance, so that ultimately, the two instances were merged. Thus, the problem was not due to kernel generation, but due to the way in which the masks were generated and in which the masks were post-processed to remove contradictions. This section focuses on the modifications we propose to overcome this problem, which only affect the way in which *thing* instances are dealt with; the remaining aspects of kernel generation and the definition of the binary masks are identical to those presented in (Li et al., 2021).

Thing instances are generated on the basis of the output of the Position Head in the Kernel Generator (Fig. 2): for every *thing* class, it will predict one heatmap per scale of the feature map $P_{f,t}$ that is the input of the Kernel Generator. Hypotheses for centres of *thing* instances of that class are determined as the positions of relative maxima of such a heatmap with a value above a certain threshold. For each such hypothesis, the kernel is sampled from the feature map that is the output of the Kernel Head in the Kernel Generator. Hypotheses generated from heatmaps at different scales are merged based on their spatial position; this differs from (Li et al., 2021) and (Nguyen et al., 2024), where only the similarity of kernels (thus, only appearance) is considered. Weak hypotheses are suppressed if there are too many such hypotheses. Each remaining hypothesis corresponds to a detected *thing* instance, and the corresponding instance mask is generated based on a convolution of the output of the Feature Encoder with the resultant kernel $\mathcal{K}^{th}_{t,i}$.

As the kernels are vectors extracted from a feature map that is generated from the input by convolutional and other layers, they mainly encode the *appearance* of *thing* instances. Consequently, the kernels extracted at the centres of instances of similar appearance at the same scale will be very similar, and the results of the convolutions will also be similar, which leads to the problem described above. In particular, there is no *geometrical* constraint that could, e.g., identify unreasonably large instance masks. Consequently, we propose to consider geometrical information to avoid the generation of unreasonably large instances as indicated by the foreground pixels of the generated instance masks. This is achieved by adding an additional network branch to the Kernel Generator, the *BBox Estimation Head*. For every spatial position and every scale of the input feature map $P_{f,t}$, this head predicts the most probable bounding box of an object if that position were the centre of a *thing* instance. Here, any bounding box is represented by a quadruple (x_0, y_0, w, h), where (x_0, y_0) represent the coordinates of the upper left corner and (w, h) denote the width and height of the box, respectively. The structure of the Bbox Estimation Head follows the one of the Kernel Head. In particular, each scale-specific feature map $p^k_{f,t}$ in the hierarchical input feature representation $P_{f,t}$ is processed by three sequential (3×3) convolutional layers, resulting in new feature maps $b^k_{f,t}$ of dimension C_b (set to the same value as the dimension of the kernels). Then, a fully connected layer is applied to the resultant feature vector at every spatial position, which regresses the four target parameters. The weights of all of these layers are shared between different scales. Whereas for most spatial positions, the regressed bounding box parameters are irrelevant, the Bbox Estimation Head is trained to predict meaningful bounding boxes at spatial positions corresponding to instance centres (cf. Section 4.5). Consequently, when kernels are generated for *thing* instances in the way described above, for every such instance, the centre and scale at which it is detected are known; the corresponding bounding box can be easily sampled from the output of the Bbox Estimation Head at the same position and at the same scale. Similarly to (Tian et al., 2019) and (Ge et al., 2021), this approach is anchor-free and does not require any region proposal network.

As mentioned earlier, kernels extracted at different scales are merged based on the spatial position of the instance they represent. Specifically, we perform a non-maximum suppression on overlapping bounding boxes (we use a threshold on the IoU of 0.5), keeping the bounding boxes with the highest confidence for the corresponding centre prediction. Analogously, we only keep the kernels corresponding to the remaining bounding boxes and discard all others. As a result, for every kernel $\mathcal{K}^{th}_{t,i}$, and thus, for every instance i at time t, a corresponding bounding box $BB_{t,i}$ is available.

The mask generation process for a *thing* instance i is also based on a convolution of the output of the Feature Extractor with $\mathcal{K}^{th}_{t,i}$. However, the mask is initialized by zeroes, and the convolution is only applied to the window identified by $BB_{t,i}$, thus constraining the potential extents of the instance geometrically in a way that was learned from training data. This approach is not specific for VPS, but can be applied to any PS method relying on similar principles; in our experiments, it will thus also be evaluated for mono-temporal PS.

4.4 Association

In this final step of VPS, the list $\mathcal{D}_t$ of instances extracted at the current timestep t is to be matched with the list of instances $\mathcal{T}_{t-1}$ from the previous timestep $t-1$ to identify which of the

new instances are already available in the original list, thus allowing to track instances over time. Let us assume that the list of tracked instances $\mathcal{T}_{t-1}$ consists of K^{th}_{t-1} instances, each instance n being represented by its mask $\mathcal{M}^{th}_{t-1,n}$, kernel $\mathcal{K}^{th}_{t-1,n}$, class label c_n, and identifier ID_n:

$$\mathcal{T}_{t-1} = \{(\mathcal{M}^{th}_{t-1,n}, \mathcal{K}^{th}_{t-1,n}, c_n, ID_n) \mid n \in [1 \dots K^{th}_{t-1}]\}. \quad (2)$$

Similarly, the list $\mathcal{D}_t$ of instances detected at time t contains K^{th}_t detected instances:

$$\mathcal{D}_t = \{(\mathcal{M}^{th}_{t,m}, \mathcal{K}^{th}_{t,m}, c_m) \mid m \in [1 \dots K^{th}_t]\}. \quad (3)$$

Thus, for an instance m in $\mathcal{D}_t$, no unique instance identifier ID_m is defined yet. The goal of the association blocks is to identify matches between instances from both lists, in order to be able to transfer the identifiers from instances in $\mathcal{T}_{t-1}$ to matching instances in $\mathcal{D}_t$. This process has to consider that some instances from either set may have no correspondence in the other one, because instances may move in or out of view between epochs $t-1$ and t. In order to achieve that goal, we first have to determine a similarity score $s_{n,m}$ between all pairs consisting of a tracked instance n and a detected instance m. As usual in tracking approaches, it is useful to consider both appearance and geometry in this context. Similarity of appearance can be evaluated by the similarity of kernels for reasons outlined in Section 4.3. As in our case, the association is to be performed in image space, we abstain from using a physical model of the object movement in order to obtain a geometrical cue. Rather than that, and different from (Kim et al., 2020), we use the optical flow $\mathcal{O}_t$ to represent the movement in image space. This can be achieved by warping the mask $\mathcal{M}^{th}_{t-1,n}$ of instance n in $\mathcal{T}_{t-1}$ to the coordinate frame of the input at time t, resulting in a warped map $\bar{\mathcal{M}}^{th}_{t-1,n}$ for that instance. Note that if, according to the optical flow, the instance is outside the image acquired at time t, the mask is not warped. The similarity score is defined as follows:

$$s_{n,m} = sim\left(\mathcal{K}^{th}_{t-1,n}, \mathcal{K}^{th}_{t,m}\right) + IoU\left(\bar{\mathcal{M}}^{th}_{t-1,n}, \mathcal{M}^{th}_{t,m}\right), \quad (4)$$

where $sim(\cdot)$ returns the cosine similarity between the two kernels associated with the two instances and $IoU(\cdot)$ represents the intersection over union between the foreground areas identified by the two masks. Thus, the first term evaluates the similarity of appearance, whereas the second one considers the geometrical aspects. The warped mask can be seen as the prediction of the mask of an instance identified at epoch $t-1$ for time t under the motion encoded by the optical flow; if two instances represent the same object, there should be a large overlap between the foreground areas indicated by the two masks.

Matching consists of linking pairs of instances from the two lists such that the sum of the similarity scores becomes a maximum under the constraint that every instance may only be assigned to one match. We use a variant of the Jonker-Volgenant algorithm (Crouse, 2016) for that purpose, which builds on the Hungarian algorithm (Kuhn, 1955), and allows the two list to be of different lengths. As it requires matching costs rather than similarity scores, we determine a matching cost matrix $\mathcal{C} \in R^{K^{th}_{t-1} \times K^{th}_t}$ with elements $c_{n,m} = -s_{n,m}$ and then find the solution with the minimum matching costs. This algorithm will establish a match for every element in the shorter of the two lists. Thus, the list of matches may also contain wrong associations between tracked instances that are actually no longer visible and new instances. To solve this issue, we discard established matches with similarity scores below a threshold (we use a value of 0.9 in our experiments). After that, we update the list of tracked objects, yielding the final list $\mathcal{T}_t$ that is one of the outputs of our method. Instances in $\mathcal{T}_{t-1}$ for which no match could be established are marked as being no longer visible. We record the time step at which an instance is marked like this for the first time; if an instance remains invisible for more than N_e timesteps, it is discarded, otherwise the instance is still included in $\mathcal{T}_t$. Instances n in $\mathcal{T}_{t-1}$ for which a match with an instance m in $\mathcal{D}_t$ could be established are also included in $\mathcal{T}_t$. In this case, the mask, kernel and the class label are taken from the instance m in $\mathcal{D}_t$, while the instance ID will be ID_n from $\mathcal{T}_{t-1}$, thus identifying the track to which this instance corresponds. Finally, instances in $\mathcal{D}_t$ for which no match could be found in $\mathcal{T}_{t-1}$ will be added to $\mathcal{T}_t$ as new instances. In this case, the next available new identifier is assigned to ID_m of that instance.

The final output of VPS consists of the stuff masks and the updated list of tracked instances $\mathcal{T}_t$. For evaluation purposes, a label image that encodes the class label and the instance ID in a unique RGB value is also generated.

4.5 Training

In the training, we minimize a loss function $\mathcal{L}$ consisting of four components:

$$\mathcal{L} = \lambda_{pos} \cdot \mathcal{L}_{pos} + \lambda_{seg} \cdot \left(\mathcal{L}^{st}_{seg} + \mathcal{L}^{th'}_{seg}\right) + \lambda_{bb} \cdot \mathcal{L}_{bb}\,, \quad (5)$$

where the Position Head loss $\mathcal{L}_{pos}$ and the segmentation loss for *stuff* classes $\mathcal{L}^{st}_{seg}$ were already used in (Li et al., 2021), while $\mathcal{L}^{th'}_{seg}$ represents the depth-aware dice loss for *thing* instances proposed in (Nguyen et al., 2024). The reader is referred to the cited literature for further details. The fourth loss term, $\mathcal{L}_{bb}$, measures the agreement of the predicted bounding box with the reference bounding box based on the IoU score and is taken from (Ge et al., 2021). The reference bounding box of an instance can be derived from its reference mask. The weights λ_{pos}, λ_{seg} and λ_{bb} are hyperparameters. In general, we follow the training procedure of (Nguyen et al., 2024). To integrate the bounding box loss, that loss is evaluated in all available scales at N_b positions sampled randomly in the vicinity of the reference centre of every instance (we use N_b = 7).

5. Experimental Setup

5.1 Dataset

We use the Cityscapes-VPS dataset (Kim et al., 2020) for the experimental validation of our method. This dataset was created from 500 videos of the validation set of the Cityscapes dataset (Cordts et al., 2016). Each video consists of 30 frames. For each timestep, stereo images are available. Kim et al. (2020) expanded the annotation for every 5^{th} frame, yielding 6 annotated frames per video. The dataset is divided into 400 videos with 2400 images for training, 50 videos with 300 images for validation, and 50 videos with 300 images for testing. As the labels of the test set are not publicly available, similarly to other authors, we use the validation set for testing. Thus, we will refer to the validation set of Cityscapes-VPS as our test set. We defined 50 videos of the training set to constitute our validation set; this implies that our training set consisted of 350 videos (2100 images) of the original training data.

5.2 Experimental Protocol

In our experiments, we only use the stereo images for which annotations are available, i.e., every fifth frame of the input videos. This corresponds to a frame rate of 6 fps. As pointed out in Section 4.1, we use (Lipson et al., 2021) to generate a depth map from the stereo images and (Teed and Deng, 2020) to compute the optical flow between consecutive reference images in the sequence. The pretrained models are provided by (Lipson et al., 2021) and (Teed and Deng, 2020), respectively. The input for our method consists of the (colour) reference images and the depth maps of two consecutive frames along with the optical flow from the previous to the current frame. For the first frame of every video ($t = 0$), the reference image and the depth map of this frame are used as input twice (for the current and the previous epoch) and the optical flow is set to zero for all pixels, compensating for the fact that no information for the actual previous frame is available. The colour images, depth maps and optical flow images all have a dimension of 1024×2048 pixels.

The hyperparameters of the modules adopted from (Nguyen et al., 2024) are set to those of the original method, and we also follow a similar training protocol. We first pre-train the method for PS of (Nguyen et al., 2024) using the Cityscapes traning set and use these parameter values to initialize the components of our new method that are shared between the two methods; this includes the parameters of the encoder processing the data from timestep $t-1$, because the two encoders share their parameters (cf. Section 4.1). The parameters of the BBox Estimation Head are initialized similarly to those of the Kernel Generator in (Li et al., 2021), i.e., by random sampling from a normal distribution with $(\mu, \sigma) = (0, 0.01)$. The parameters of the temporal fusion block for exchanging information between two consecutive frames are randomly initialized according to (He et al., 2015). We use Stochastic Gradient Descent with a weight decay of 10^{-4} and a momentum of 0.9 to minimize the loss defined in Equation 5, following the protocol described in Section 4.5 for evaluating the loss $\mathcal{L}_{bb}$ for the BBox Estimation Head. The weights of the loss terms are set to $\lambda_{pos} = 1.0$, $\lambda_{seg} = 4.0$ and $\lambda_{bb} = 5.0$. The learning rate is initially set to 0.0005 and reduced by a factor of 0.9 after every 1000^{th} iteration. The input images and depth maps are normalized following (Nguyen et al., 2024). We apply data augmentation by scaling the input with factors randomly drawn from $[0.5, 1.2]$, by random horizontal flips with a probability of 0.5 and by random crops to windows of size 512×1024 pixels. One minibatch consisting of 10 patches of 512×1024 pixels each is fed to our network in each training iteration. While the maximum number of training iterations is set to 60,000, we use early stopping based on the video panoptic quality (cf. Section 5.3) computed on the validation set and keep the set of network parameters achieving the highest value on the validation set for testing. All experiments are carried out on a Nvidia A100 GPU with 40 GB memory.

To put the performance of our method into context, we compare our results to those of three methods from the literature: The PS method we use as our baseline, extended by our association procedure to lift this method to VPS (Nguyen et al., 2024), as well as (Kim et al., 2020) and (Li et al., 2022). To enable a fair comparison, we use variants of the listed methods that all employ the same backbone in the encoder, namely Resnet50 (Lin et al., 2017). Note that the quantitative results reported for (Kim et al., 2020) are taken from (Qiao et al., 2021), to ensure an experimental setup consistent with the one we use in the present paper. However, while we split the training set of the Cityscapes-VPS dataset into a training and a validation part (cf. Sec. 5.1), the methods taken from the literature use the complete training set for optimizing the unknown network parameters, not performing any separate validation.

5.3 Evaluation Protocol

We use the *Video Panoptic Quality (VPQ)* from (Kim et al., 2020) as our main evaluation metric, which is an extension of the *Panoptic Quality (PQ)* (Kirillov et al., 2019b) used for the image-level evaluation. Unlike PQ, *VPQ* is computed on the basis of all frames in a video:

$$VPQ = \frac{\sum_{(Pr,Gt)\in TP} IoU\,(Pr, Gt)}{|TP| + \frac{1}{2}\cdot|FP| + \frac{1}{2}\cdot|FN|}\,, \tag{6}$$

where Pr, Gt are a predicted and a ground truth mask found to correspond to each other and thus correspond to the set of true positives (TP), and $IoU\,(Pr, Gt)$ is the intersection over union between these masks. The true positives, false positives and false negatives are determined for every frame based on (Kirillov et al., 2019b). $|TP|$ is the number of TPs; the fact that a mask is considered to be a TP implies that $IoU > 0.5$. $|FP|$ and $|FN|$ are the numbers of false positive and false negative pairs, respectively. In our experiments, we also present VPQ^{th} and VPQ^{st}, which give the VPQ quality specifically for *thing* and *stuff* classes. More details about VPQ, VPQ^{th} and VPQ^{st} can be found in (Kim et al., 2020). This metric can also be determined as a function of the length T of a video, yielding values VPQ_T. Specifically, we adopt the procedure of Kim et al. (2020) and report such values for $T = 0, 1, 2, 3$, corresponding to videos with $1, 2, 3, 4$ frames, respectively.

6. Results and Discussion

6.1 Results and Comparison to Other Methods

Table 1 shows the VPQ metrics obtained by our method and by the baseline methods introduced in Section 5.2. Our method achieves a $VPQ_{T=0}$ of 64.3% for PS and a $VPQ_{T=3}$ of 51.9% for VPS if four consecutive frames are considered in the evaluation. It outperforms (Nguyen et al., 2024), which it extends, by a margin of 1% in the PS setting ($T = 0$) and by 1.7% for $T = 3$. The improvement is mainly due to an improved performance for *thing* classes, as indicated by VPQ^{th}: in the PS setting ($T = 0$), this metric is improved by 3.3%, while for $T = 3$, the improvement in $VPQ^{th}_{T=3}$ is 4.3%. The improvement for $T = 0$ is due to the prediction of bounding boxes and their consideration when computing binary segmentation masks. This improvement is directly propagated to the VPQ, because only correctly detected instances can be correctly associated in the second step.

Compared to the other methods from the literature, our results are slightly worse than those reported in the cited papers, though one has to take into account that we use a smaller dataset for the actual determination of the unknown network parameters due to considering a subset of them for validation. In the PS setting, the best method is (Li et al., 2022), with a margin of 1.3% in VPQ, mainly due to a better segmentation for *stuff*. However, the margin in VPQ becomes smaller with increasing length of the sequence used for evaluation, and for $T > 1$, our

Method	$T = 0$	$T = 1$	$T = 2$	$T = 3$	avg
(Kim et al., 2020)	65.0/59.0/69.4	57.6/45.1/66.7	54.4/39.2/65.6	52.8/35.8/65.3	57.5/44.8/66.7
(Li et al., 2022) with Resnet50 backbone	65.6/57.4/71.5	57.7/43.4/68.2	54.2/36.5/67.1	52.3/33.1/66.3	57.5/42.6/68.3
(Nguyen et al., 2024) with our association	63.3/54.1/70.0	54.9/38.0/67.2	52.4/34.1/65.7	50.2/30.0/64.8	55.2/39.5/66.9
Ours	64.3/57.4/69.4	56.4/42.6/66.4	53.6/37.7/65.2	51.9/34.3/64.7	56.6/43.0/66.4

Table 1. Evaluation of the results of our method for VPS and comparison to methods from the literature. The reported metrics are VPQ / VPQ^{th} / VPQ^{st}, i.e., the VPQ for all classes, for *thing* classes, and for *stuff* classes, all computed on our test set; all metrics are given in [%]. T refers to the length of the considered sequence, with $T = 0$ being equivalent to PS.

method achieves a better VPQ for the *thing* classes (e.g. +1.2% in $VPQ^{th}_{T=3}$). This indicates that our combined association approach, considering appearance and optical flow information, stabilizes the linking of *thing* instances for longer image sequences, compared to a purely appearance-based association. Compared to (Kim et al., 2020), our VPQ is worse by 0.7% in the PS setting, mainly due to a worse performance for *thing* classes, yielding a value for $VPQ^{th}_{T=0}$ that is 2.3% larger than the one achieved by our method. This difference indicates that the temporal fusion performed in the encoders of both methods, theirs and ours, has a different effect: whereas it leads to a significant improvement in (Kim et al., 2020), our ablation studies show that it even had a slightly negative effect (cf. Section 6.2 and Table 2). In this context it might be important to mention that $T = 0$ only refers to the length of the sequence that is considered when evaluating the VPQ (cf. Eq. 6); for frames $t > 0$, the previous frame is still considered when computing the PS masks, i.e., the temporal fusion block also affects the results for $T = 0$. In the case $T = 3$, the margin between our method and (Kim et al., 2020) in the quality metrics for *thing* classes becomes larger, indicated by a difference in $VPQ^{th}_{T=3}$ of 1.5%. However, in general our method achieves results at a similar level as the state of the art.

6.2 Ablation Studies

In this section, we present experiments to analyse the effect of the individual main novelties of our method. For this purpose, we evaluate six different variants of our method (V0 - V5, cf. Table 2), leaving out individual components; V5 corresponds to our complete method. Specifically, we investigate the influence of the following components: the *temporal fusion* in the encoder ($\mathcal{F}$), the *bounding box-based* extension of the employed kernel-based PS method ($\mathcal{BB}$), the *appearance-based association* ($\mathcal{TK}$) and the proposed *association based on optical flow* ($\mathcal{TM}$).

Discarding the fusion block (variant V1), shows slightly better result than our complete method, for the single frame case $VPQ_{T=0}$ as well as for the sequence case $VPQ_{T>0}$. While this block improved the panoptic and video panoptic quality significantly in (Kim et al., 2020), it apparently does not show its potential within our architecture. We assume that some auxiliary loss function directly focusing on the optimization of this block may help to overcome this limitation, and we will investigate this direction in future work. In contrast, discarding our bounding box-based extension (variant V2) leads to a clear deterioration of the VPQ, in particular for VPQ^{th}, for all investigated lengths of sequences. This emphasizes the importance of considering geometric information in the segmentation of *thing* instances and is also illustrated in Figure 4. In variant V3, our optical flow-based association is discarded, limiting the linking of *thing* instances across frames to solely using appearance information. Compared to our complete method, V3 shows clearly worse results, in particular, for *thing* classes in the sequence case $VPQ^{th}_{T>0}$, where it comes to an increased number of incorrect associations of *thing* instances, so-called ID switches, across frames (cf. Fig. 4). The results for the single frame case ($T = 0$) are identical, as this case is equivalent to a PS, with association having no impact on the evaluation. For variant V4, we discard the appearance-based association and link *thing* instances solely based on the optical flow-based motion information. This leads to similar effects as observed for V3, but with a smaller difference to our complete method. Comparing the results of V3 and V4 to those of our complete method (V5), it becomes evident that a combined association approach, considering appearance and motion information, works best.

7. Conclusion

In this paper, we present a new online VPS method, focusing on two key aspects: (i) the consideration of geometric information in the computation of segmentation masks for thing instances using predicted bounding boxes and (ii) the consideration of optical flow-based motion information in the association step. Comprehensive experiments on the Cityscapes-VPS dataset have shown that expanding (Nguyen et al., 2024) based on bounding boxes leads to a clear improvement in the PS of *thing* classes by 3.3%. Together with our optical flow-based extension in the association step, an overall improvement of 3.5% in the average VPS for the *thing* classes is achieved, compared to that baseline. These results emphasize the importance of considering spatial information in both steps, the panoptic segmentation and the association. Compared to other methods from the state of the art, our method obtains slightly worse results, though this might be attributed to the fact that we used a smaller dataset for parameter determination, sparing a subset of the available training data for validation.

It has to be noted that the model we used in the encoder to fuse feature information across frames using the optical flow does not yield noticeable improvements. We will further analyze this limitation and explore more effective strategies for integrating information from previous frames in future work. Lastly, we currently use depth and optical flow information separately; in future work, we will combine both into scene flow, allowing us to model the movement of *thing* instance in 3D object space, opening up exciting new research directions.

This contribution has been peer-reviewed. The double-blind peer-review was conducted on the basis of the full paper.
https://doi.org/10.5194/isprs-annals-XI-2-2026-321-2026 | © Author(s) 2026. CC BY 4.0 License.

	$\mathcal{BB}$	$\mathcal{F}$	$\mathcal{TK}$	$\mathcal{TM}$	$T=0$	$T=1$	$T=2$	$T=3$	avg
V0			✓	✓	63.3/54.1/70.0	54.9/38.0/67.2	52.4/34.1/65.7	50.2/30.0/64.8	55.2/39.5/66.9
V1	✓		✓	✓	64.6/57.6/69.7	56.8/43.4/66.6	53.9/38.2/65.4	52.0/34.6/64.6	56.8/43.4/66.6
V2		✓	✓	✓	63.2/54.7/69.4	54.8/38.4/66.7	52.7/35.4/65.3	50.1/31.4/64.5	55.2/40.0/66.5
V3	✓	✓	✓		64.3/57.4/69.4	54.6/37.9/66.4	50.8/31.1/65.2	48.8/27.0/64.7	54.6/38.4/66.4
V4	✓	✓		✓	64.3/57.4/69.4	56.0/41.8/66.4	52.9/36.1/65.2	50.9/31.9/64.7	56.0/41.8/66.4
V5	✓	✓	✓	✓	64.3/57.4/69.4	56.4/42.6/66.4	53.6/37.7/65.2	51.9/34.3/64.7	56.6/43.0/66.4

Table 2. Ablation studies on different variants of our proposed method. The reported metrics are VPQ / VPQ^{th} / VPQ^{st}, i.e., the VPQ for all classes, for *thing* classes, and for *stuff* classes, all computed on our test set; all metrics are given in [%]. T refers to the length of the considered sequence, with $T=0$ being equivalent to PS. The different modules refer to: the *temporal fusion* in the encoder ($\mathcal{F}$), the *bounding box-based* extension of the employed kernel-based PS method ($\mathcal{BB}$), the *appearance-based association* ($\mathcal{TK}$) and the proposed *association based on optical flow* ($\mathcal{TM}$).

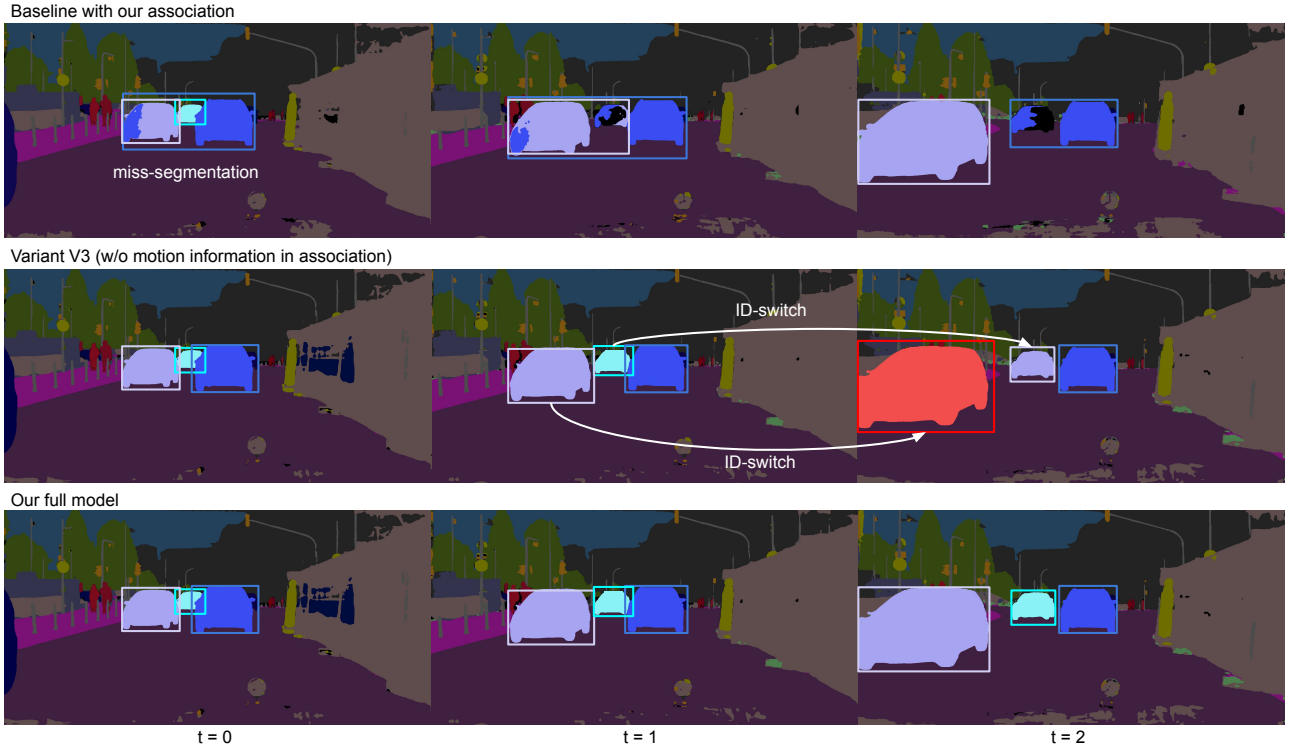

Figure 4. Exemplary results to illustrate the effect of our main contributions, produced by the approach we used as basis for our method (first row, variant V0, cf. Tab. 2), our method without considering our optical flow-based instance association (second row, variant V3) and our complete method (bottom row, variant V5). The instances are supposed to remain a consistent identity across frames, indicated by their colour. t refers to the timestamp of the respective image.

Acknowledgements

This work was supported by the German Research Foundation (DFG) as a part of the Research Training Group i.c.sens [GRK2159]. Computations were carried out on the LUH computer cluster, funded by the Leibniz Universität Hannover, the Lower Saxony Ministry of Science and Culture (MWK), and DFG.

References

Cheng, B., Collins, M. D., Zhu, Y., Liu, T., Huang, T. S., Adam, H., Chen, L.-C., 2020. Panoptic-deeplab: A simple, strong, and fast baseline for bottom-up panoptic segmentation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 12475–12485.

Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B., 2016. The cityscapes dataset for semantic urban scene understanding. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 3213–3223.

Crouse, D. F., 2016. On Implementing 2D Rectangular Assignment Algorithms. *IEEE Transactions on Aerospace and Electronic Systems*, 52(4), 1679–1696.

Ge, Z., Liu, S., Wang, F., Li, Z., Sun, J., 2021. Yolox: Exceeding yolo series in 2021. *arXiv preprint arXiv:2107.08430*.

He, K., Zhang, X., Ren, S.and Sun, J., 2015. Delving Deep into rectifiers: Surpassing human-level performance on ImageNet classification. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 1026–1034.

Kim, D., Woo, S., Lee, J.-Y., Kweon, I. S., 2020. Video panoptic segmentation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 9859–9868.

Kim, D., Xie, J., Wang, H., Qiao, S., Yu, Q., Kim, H.-S., Adam, H., Kweon, I. S., Chen, L.-C., 2022. Tubeformer-deeplab: Video mask transformer. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 13914–13924.

Kirillov, A., Girshick, R., He, K., Dollár, P., 2019a. Panoptic feature pyramid networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 6399–6408.

Kirillov, A., He, K., Girshick, R., Rother, C., Dollár, P., 2019b. Panoptic segmentation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 9404–9413.

Kuhn, H. W., 1955. The Hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2), 83–97.

Li, X., Zhang, W., Pang, J., Chen, K., Cheng, G., Tong, Y., Loy, C. C., 2022. Video k-net: A simple, strong, and unified baseline for video segmentation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 18847–18857.

Li, Y., Chen, X., Zhu, Z., Xie, L., Huang, G., Du, D., Wang, X., 2019. Attention-guided unified network for panoptic segmentation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 7026–7035.

Li, Y., Zhao, H., Qi, X., Wang, L., Li, Z., Sun, J., Jia, J., 2021. Fully convolutional networks for panoptic segmentation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 214–223.

Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S., 2017. Feature pyramid networks for object detection. *CVPR*, 2117–2125.

Lipson, L., Teed, Z., Deng, J., 2021. Raft-stereo: Multilevel recurrent field transforms for stereo matching. *Proceedings of the International Conference on 3D Vision (3DV)*, 218–227.

Milletari, F., Navab, N., Ahmadi, S.-A., 2016. V-net; Fully convolutional neural networks for volumetric medical image segmentation. *Proceedings of the International Conference on 3D Visiion*, 565–571.

Nguyen, T., Mehltretter, M., Rottensteiner, F., 2024. Depth-Aware Panoptic Segmentation. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, X-2-2024, 153–161. https://isprs-annals.copernicus.org/articles/X-2-2024/153/2024/.

Qiao, S., Zhu, Y., Adam, H., Yuille, A., Chen, L.-C., 2021. Vip-deeplab: Learning visual perception with depth-aware video panoptic segmentation. *CVPR*, 3997–4008.

Shin, I., Kim, D., Yu, Q., Xie, J., Kim, H.-S., Green, B., Kweon, I. S., Yoon, K.-J., Chen, L.-C., 2024. Video-kmax: A simple unified approach for online and near-online video panoptic segmentation. *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 229–239.

Teed, Z., Deng, J., 2020. Raft: Recurrent all-pairs field transforms for optical flow. *European Conference on Computer Vision (ECCV)*, Springer, 402–419.

Tian, Z., Shen, C., Chen, H., He, T., 2019. Fcos: Fully convolutional one-stage object detection. *CVPR*, 9627–9636.

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

Wang, H., Zhu, Y., Adam, H., Yuille, A., Chen, L.-C., 2021. Max-deeplab: End-to-end panoptic segmentation with mask transformers. *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 5463–5474.

Xiong, Y., Liao, R., Zhao, H., Hu, R., Bai, M., Yumer, E., Urtasun, R., 2019. Upsnet: A unified panoptic segmentation network. *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 8818–8826.

Yu, Q., Wang, H., Qiao, S., Collins, M., Zhu, Y., Adam, H., Yuille, A., Chen, L.-C., 2022. k-means mask transformer. *European Conference on Computer Vision (ECCV)*, Springer, 288–307.

Zhang, W., Pang, J., Chen, K., Loy, C. C., 2021. K-net: Towards unified image segmentation. *Advances in Neural Information Processing Systems*, 34, 10326–10338.

Zhou, X., Wang, D., Krähbühl, P., 2019. Objects as Points. *arXiv preprint arXiv:1904.07850*.

Zhou, Y., Zhang, T., Ji, S., Yan, S., Li, X., 2024. Improving video segmentation via dynamic anchor queries. *European Conference on Computer Vision (ECCV)*, Springer, 446–463.