Depth-Aware 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: Panoptic Segmentation, RGB Depth Fusion, Dice Loss

Abstract

Panoptic segmentation unifies semantic and instance segmentation and thus delivers a semantic class label and, for so-called *thing* classes, also an instance label per pixel. The differentiation of distinct objects of the same class with a similar appearance is particularly challenging and frequently causes such objects to be incorrectly assigned to a single instance. In the present work, we demonstrate that information on the 3D geometry of the observed scene can be used to mitigate this issue: We present a novel CNN-based method for panoptic segmentation which processes RGB images and depth maps given as input in separate network branches and fuses the resulting feature maps in a late fusion manner. Moreover, we propose a new depth-aware dice loss term which penalises the assignment of pixels to the same *thing* instance based on the difference between their associated distances to the camera. Experiments carried out on the Cityscapes dataset show that the proposed method reduces the number of objects that are erroneously merged into one *thing* instance and outperforms the method used as basis by $+2.2\%$ in terms of panoptic quality.

1. Introduction

Panoptic segmentation combines the tasks of semantic segmentation and instance segmentation (Kirillov et al., 2019b). For a set of *thing* classes, e.g. *car*, it delivers information about individual instances, e.g. in the form of bounding boxes with class labels and binary masks indicating the pixels corresponding to the instance. Image regions not belonging to *thing* instances (*background* in instance segmentation) are assigned to one of the so-called *stuff* classes in a similar way as in semantic segmentation. For these classes (e.g., *wall*), no information about instances is determined.

This task is usually solved using neural networks. Early approaches merged the results of separate methods for instance and semantic segmentation in post-processing (Kirillov et al., 2019b). Recent approaches apply unified strategies that allow for end-to-end training. Li et al. (2021) achieve this goal by predicting a binary mask for every *stuff* class and a binary mask and a class label for every instance of every *thing* class. This avoids the need for bounding box proposals and allows the network to learn the two sub-tasks jointly in an end-to-end manner. Existing work usually only relies on RGB images as input. Fig. 1 shows an example for a binary instance mask predicted from such an image by (Li et al., 2021). In this example, two *car* instances having a similar appearance are actually merged into one. One way to overcome such problems is to integrate additional information. In this work, we utilise *depth* from stereo images as an additional input, thus proposing a method for depth-aware panoptic segmentation.

RGB and depth data have been used for semantic and instance segmentation for some time, but there are only few works on panoptic segmentation exploiting both modalities. Narita et al. (2019) still apply RGB images for panoptic segmentation, using depth only to estimate camera poses and to generate a 3D map. Seichter et al. (2022) process depth and colour in two separate branches of a deep neural network before fusing the outputs to use them for instance segmentation and semantic segmentation. Thus, this method still involves two separate networks. In this work, we try to overcome some of the problems of existing methods, making the following scientific contributions:

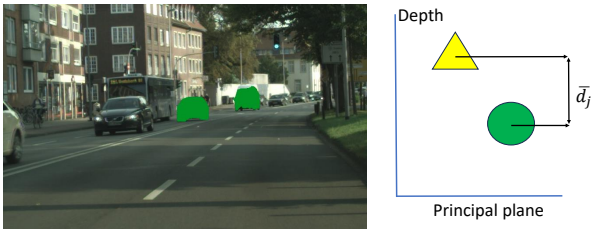

Figure 1. *Left:* A binary instance mask predicted by (Li et al., 2021) which erroneously merges two *car* instances, superimposed to the input. *Right:* We exploit the depth difference $\bar{d}_j$ between pixels corresponding to different instances (the *triangle* and the *circle*) in training to mitigate the problem.

- We propose a method for the joint use of colour and depth for panoptic segmentation that can be trained end-to-end.
- In this context, we investigate two different techniques for the fusion of the colour and depth branches of the network.
- We propose a new depth-aware loss term in order to mitigate problems such as the one indicated in Fig. 1, exploiting the depth differences of separate *thing* instances.
- We show the improvements achieved by the additional information and the new loss function in experiments using a publicly available benchmark dataset.

Our method is based on (Li et al., 2021), which we extend by an additional depth branch, fusing the resultant features with those obtained from the colour branch of the network, and by a new depth-aware loss function.

2. Related Work

Panoptic segmentation methods can be divided into top-down (box-based), bottom-up (box-free) and unified approaches, the

* Corresponding author

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-X-2-2024-153-2024 | © Author(s) 2024. CC BY 4.0 License.

latter being also referred to as single path approaches. Top-down and bottom-up approaches treat semantic and instance segmentation separately before merging their results to obtain the panoptic segmentation results. Top-down methods follow a two-stage design and estimate bounding boxes for *thing* instances first, before a pixel-wise mask and a semantic label is predicted per instance, e.g. (Kirillov et al., 2019a; Xiong et al., 2019; Li et al., 2019). The semantic segmentation of the background (i.e. all pixels not corresponding to a *thing* instance) is commonly carried out separately. As a consequence, the performance highly depends on the quality of the estimated bounding boxes. Consistency between the semantic segmentation masks of overlapping bounding boxes and between instances and the background is not guaranteed, which requires to resolve conflicts in a heuristic post-processing step. Bottom-up approaches address this limitation by estimating semantic and instance segmentation masks without relying on previously estimated bounding boxes. For instance, Cheng et al. (2020) apply semantic segmentation differentiating both *stuff* and *thing* classes. The instance masks are derived from the outputs of two additional network branches: a centre point for each object instance and an offset to the corresponding centre point for each pixel being located on such an instance. All network branches can be trained end-to-end, but some rather complex post-processing is required to derive the instance masks and class labels from the original output.

Unified approaches do not apply separate networks or network branches for semantic and instance segmentation, but solve the panoptic segmentation task directly, e.g. by simultaneously predicting binary masks for *stuff* classes and *thing* instances. Following this strategy, Li et al. (2021) propose to learn the estimation of two types of intermediate feature maps: maps that describe individual *thing* instances and *stuff* classes and maps that encode the input image. Maps of the first type are used to extract filter kernels for convolutions that are applied to the second type of maps. The result of these convolutions is a set of binary masks (one per *thing* instance and one per *stuff* class). A limitation of (Li et al., 2021) is the incorrect assignment of pixels showing distinct objects of similar appearance to a single instance mask (c.f. Fig. 1). de Geus and Dubbelman (2023) claim that this problem is related to the training procedure which only uses image crops, as only a small number of (partially visible) objects is seen by the network at once. They propose an additional loss term that enforces the two kinds of feature maps described above to be different for each image crop, assuming that different crops show different objects. Zhang et al. (2021) propose a method similar to (Li et al., 2021), estimating and using the previously mentioned two types of features maps in the same way. Building on (Kirillov et al., 2019a), the authors focus on the discriminative ability of the feature maps used as filter kernels by following a clustering-based approach, which encourages features from the same class to be similar and features from different classes to be distinct. Wang et al. (2021) propose an attention-based architecture with a 2D pixel-based and a 1D global memory path. The former is used to estimate a binary segmentation mask per instance, the latter provides a semantic class label per mask. The two paths are densely connected with so-called dual-path transformer blocks, which allow to interchange information between the two paths. To ensure consistency across the individual segmentation masks, i.e., that each pixel of an image belongs to exactly one mask, the softmax function is applied per pixel to the set of predicted segmentation masks. Yu et al. (2022) incorporate the concept of conventional clustering approaches into a mask transformer architecture to identify pixels that belong to the same object instance in an early stage of the neural network. The assignment of pixels to clusters as well as the update of cluster centres and per pixel feature descriptors are realised as attention layers and are computed iteratively. Unified approaches could achieve a significant improvement in panoptic segmentation, also reducing the need for post-processing to obtain a consistent result. However, the majority of methods rely on a single RGB image, being thus limited to 2D information on the observed scene.

To improve the results even further, information about the 3D geometry of the observed scene can used as an additional input. Narita et al. (2019) take a sequence of RGB images and corresponding depth maps as input to estimate a panoptic segmentation in 3D in form of a volumetric map. Panoptic segmentation is first carried out in 2D per frame of the sequence, using one RGB image only. The depth information is used to estimate the exterior orientation parameters of the corresponding RGB image and to combine the frame-based 2D panoptic segmentation masks into a volumetric 3D representation for the whole sequence. Wu et al. (2021) present a method for incremental 3D scene graph estimation from RGB and depth data that also delivers a panoptic segmentation of the observed 3D surfaces as a by-product. A graph neural network is trained to build a graph in which clusters of pixels that belong to the same object or object part correspond to the nodes, while the edges represent geometric relations between the nodes. A panoptic segmentation is obtained by combining nodes corresponding to the same object based on the edge information. In both of these methods, depth is not used to support the panoptic segmentation itself, but only to fuse independently estimated 2D panoptic segmentation masks and to lift these masks from 2D to 3D. In contrast, Seichter et al. (2022) use both, an RGB image and a depth map, to perform a 2D panoptic segmentation using an encoder-decoder architecture. First, colour and depth are processed in two separate encoder branches. The extracted feature maps are fused at different scales, making this an example for a late fusion approach. The decoder consists of two separate branches as well, one for estimating a semantic and one for an instance segmentation. Thus, Seichter et al. (2022) follow a bottom-up strategy, which suffers from the limitations discussed earlier. While depth information is used for panoptic segmentation, the relation between depth and the segmentation masks to be estimated is learned in a purely data-driven way, i.e., no constraints on the segmentation are explicitly introduced based on the geometry in training.

In summary, (Li et al., 2021) and (Seichter et al., 2022) can be considered the most similar works to the one presented in this paper. We use (Li et al., 2021) as the basis for our work, but extend it by incorporating an additional branch for processing depth information and by the loss function used in training, for which we propose a new depth-aware term. The way in which we integrate depth is inspired by Seichter et al. (2022), but our overall architecture is different. Furthermore, depth is not just used as as additional input, but also in the loss function to explicitly constraint the assignment of pixels to instance masks.

3. Background: Panoptic FCN

To make this paper self-contained, we start with a brief summary of Panoptic FCN (Li et al., 2021). It uses RGB colour images $X^c \in R^{3\times H\times W}$ as input, where H and W represent the image height and width, respectively. The goal is to assign every pixel of a picture either to one of K^{st} *stuff* classes or to an

instance of one of N^{th} *thing* classes. Every image is presented to a Feature Pyramid Network (FPN) with Resnet50 backbone (Lin et al., 2017a) to extract features at different scales, resulting in a set of feature maps P_p, $p \in \{2, \ldots, 7\}$, with spatial extents $H_p \times W_p = H/(2^p) \times W/(2^p)$. The outputs of the FPN are processed further in two separate branches: the *Feature Encoder* and the *Kernel Generator*, cf. Fig. 2 (Li et al., 2021).

In the Feature Encoder, the feature maps P_2 to P_5 are first processed by the semantic FPN module of Kirillov et al. (2019a) and then by three sequential convolution layers. The result of the last layer is a feature map of dimension $C_e \times H/4 \times W/4$ encoding the image content in a way appropriate for the task.

The input of the Kernel Generator consists of the feature maps P_3 to P_7 generated by the FPN. First, each feature map P_p is processed independently by two heads, each consisting of three sequential convolution layers (Li et al., 2021): the *Kernel Head* and the *Position Head*. The *Kernel Head* is trained to predict the *kernel weight* tensor of dimension $C_e \times H/4 \times W/4$, which contains a weight vector (referred to as *kernel*) for every spatial position of the feature map P_p. The output of the last convolution layer of the *Position Head* consists of (N^{th} + K^{st}) maps of class scores, i.e. one per class (normalised by a sigmoid function). For the K^{st} *stuff* classes, each map contains the probability for every pixel in P_p to belong to the corresponding class; applying a threshold, that map is converted into a binary map indicating the pixels of that class. For the N^{th} *thing* classes, these maps indicate the probability of a pixel to correspond to a centre of a *thing* instance. Instance centres are determined by applying a threshold and local nonmax suppression to these maps (Zhou et al., 2019).

The output of the Position Head is used to define the kernels that are the output of the Kernel Generator. For every *stuff* class, at every scale p, one kernel is obtained by computing the average of the vectors in the kernel weight tensor at the positions assigned to that class in the binary map generated by the Position Head, and the kernels determined at different scales are averaged. For the *thing* classes, the kernel related to an instance at a scale p is extracted from the kernel weight tensor at the position of the instance centre. The resultant *thing* kernels extracted at different scales have to be combined. To do so, kernels related to instances of the same *thing* class are merged by averaging if their cosine similarity is above a threshold. This will lead to K_0^{th} *thing* instances for which the kernel and the class label are known. If K_0^{th} is larger than a pre-defined value K_{max}^{th} (set to 100 in the experiments), the kernels are ordered according to the confidence scores from the heatmap, and the K_{max}^{th} kernels having the highest confidence are preserved, thus $K^{th} = K_{max}^{th}$. Otherwise, K^{th} is set to $K^{th} = K_0^{th}$. The final output of the Kernel Generator consists of the $K^{st} + K^{th}$ kernels of dimension $1 \times C_e$, each associated with a (*stuff* or *thing*) class label.

The output of the Feature Encoder is convolved with each of the kernels, and each of the outputs is normalised by a sigmoid function, yielding $K^{st} + K^{th}$ maps of class scores for pixels to belong to one of the *stuff* classes or to one of the *thing* instances at a reduced resolution ($H/4 \times W/4$). The class label associated with a kernel is also associated with the corresponding mask. These masks are upsampled by bilinear interpolation to obtain scores at the original resolution, and after applying a threshold, $K^{st} + K^{th}$ binary masks are generated that indicate whether a pixel belongs to the corresponding *stuff* class or to the corresponding instance of a *thing* class.

Finally, post-processing is applied to remove contradictions between the predicted binary maps in a way similar to (Kirillov et al., 2019a). Pixels not assigned to any of the classes or instances are considered to be *background*. The resultant $K^{st} + K^{th}$ binary maps of size $H \times W$ along with the corresponding class labels are the final output.

For training, a reference consisting of binary masks for the *stuff* classes and the *thing* instances is required. Training is based on minimising a loss function $\mathcal{L}$ (Li et al., 2021):

$$\mathcal{L} = \lambda_{pos} \cdot \mathcal{L}_{pos} + \lambda_{seg} \cdot \mathcal{L}_{seg}, \quad (1)$$

where λ_{pos} and λ_{seg} are hyperparameters for weighting the two loss terms. The term $\mathcal{L}_{pos}$ is applied to the output of the Position Head of the network. It compares the maps containing class scores determined for every scale to a reference using a focal loss (Lin et al., 2017b). The reference for the *stuff* classes consists of the binary masks downsampled by a factor of 4. In case of the *thing* instances, for every class a binary mask showing the centres of all reference instances of that class is generated first. The reference for the *thing* centres is obtained by blurring this mask. Consequently, this reference is not binary.

The second loss term, $\mathcal{L}_{seg}$, is applied to the sigmoid scores predicted at the resolution $H/4 \times W/4$, i.e. before upsampling. Consequently, the reference maps have to be downsampled by a factor of 4 for training. In training, the kernels are not determined on the basis of the predictions of the Position Head, but they are sampled on the basis of the reference. For the *stuff* classes, one position is randomly sampled inside of the area assigned to that class in the reference at the corresponding scale, and the kernel related to that scale is sampled at that random position. For *thing* instances, the k pixels inside the instance according to the reference mask having the highest confidence in the prediction are used to extract the kernel at every scale (Li et al. (2021) use $k = 7$). In this way, it is known which predicted instance masks correspond to which reference masks. $\mathcal{L}_{seg}$ is modelled as a Dice loss (Milletari et al., 2016), comparing the binary masks for all *stuff* classes and *thing* instances.

4. Depth-aware Panoptic Segmentation

We start the presentation of our method for depth-aware panoptic segmentation with an overview (Section 4.1). Afterwards, we focus on our main modifications compared to the baseline (cf. Section 3): our concept of fusing RGB and depth data is presented in Section 4.2, while the training procedure, introducing our new depth-aware dice loss, is described in Section 4.3.

4.1 Overview

Our method is based on Panoptic FCN as presented in Section 3, expanding it so that it can use a depth map as an additional input. The architecture is shown in Fig. 2, which also highlights our new contributions by red edging. The input consists of a colour (RGB) image $X^c \in R^{3 \times H \times W}$ and a corresponding depth map $X^d \in R^{1 \times H \times W}$ of the same size and given in the same coordinate frame. In principle, any method can be used to generate the depth map; we used stereo matching in our experiments. We decided to use a late fusion approach in which the colour and depth images are processed in separate encoder branches before being fused in order to generate the feature map that serves as the input to the Feature Encoder and to the Kernel Generator. Details about this fusion approach are presented

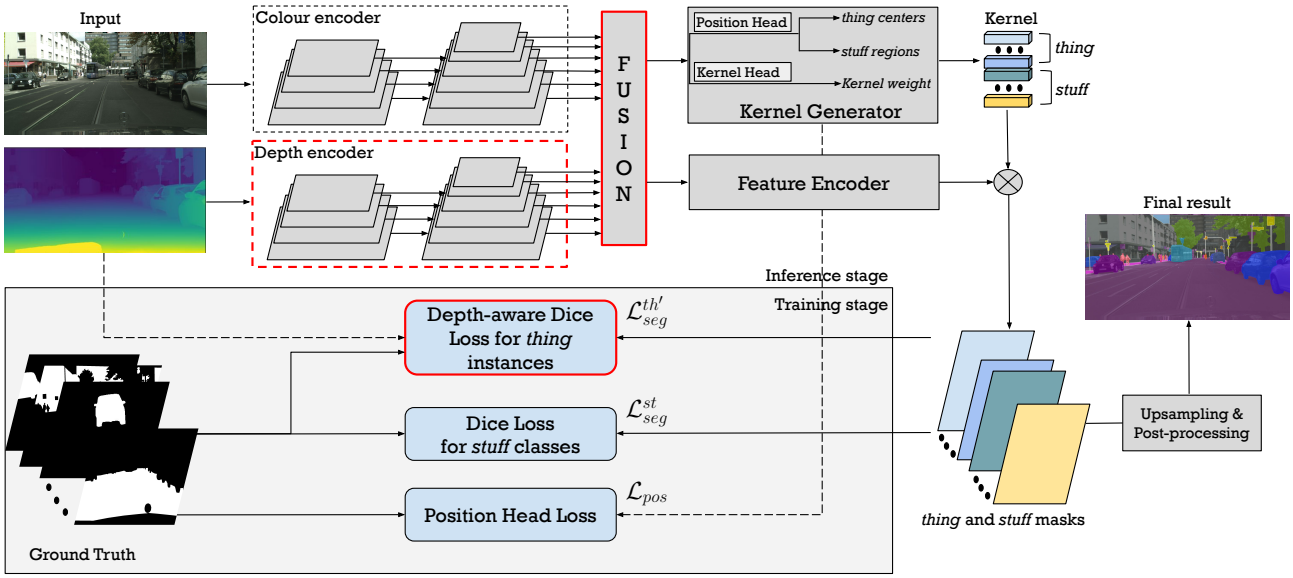

Figure 2. Our proposed method. The blocks with a red edging are our proposed modules. The remaining ones are also used in Panoptic FCN, but there the output of the colour encoder is directly processed by the feature encoder and kernel generator blocks (Li et al., 2021). Our method additionally uses an encoder for the depth map and a fusion module; the subsequent blocks process the results of colour and depth fusion. $\otimes$ indicates a convolution. In training, we use a new depth-aware Dice loss for the *thing* instances.

in Section 4.2. For the Feature Encoder and the Kernel Generator we use the architecture described in Section 3. The output of our method also consists of K^{st} binary maps identifying all pixels of the *stuff* classes and K^{th} binary maps identifying all pixels corresponding to one of the instances of the *thing* classes, in the latter case along with the class labels.

Training is also based on minimising the loss function having two components according to equ. 1. However, in order to alleviate problems such as those indicated in Fig. 1, we propose a new depth-aware Dice loss that is applied to the *thing* instances in our model for the loss term $\mathcal{L}_{seg}$. The training procedure and this new loss function are explained in Section 4.3.

4.2 Colour and Depth Fusion

Seichter et al. (2021) process the colour and depth images in separate encoder branches with a similar architecture before fusing the resultant features. We follow this *late fusion approach*, extending the Panoptic FCN architecture by a depth branch in the encoder. We do so because in preliminary experiments this variant outperformed an early fusion approach in which the depth map was just concatenated to the RGB image presented to the FPN backbone as a fourth input band. The depth branch has the same architecture as the colour branch, except that the input only consists of a single band. Thus, the two encoder branches deliver two multi-scale outputs, P_p^c and P_p^d for colour and depth, respectively, with $p \in \{2, \ldots, 7\}$ and dimensions as described for the colour backbone in Section 3.

The fusion block in Fig. 2 combines the colour and depth feature maps at corresponding scales to obtain fused feature maps P_p^f. There are several ways in which the fusion can be carried out. Our default option is *mean* fusion:

$$P_p^f = (P_p^c + P_p^d)/2 \;\; \forall\, p \in \{2, \ldots, 7\}. \tag{2}$$

In this case, the fused feature map at scale level p is determined as the arithmetic mean of the corresponding colour and depth feature maps. In our experiments, we compare this default method to fusion based on *concatenation*:

$$P_p^f = conv\left(concatenate(P_p^c, P_p^d)\right) \;\; \forall\, p \in \{2, \ldots, 7\} \tag{3}$$

Here, two colour and depth feature maps at scale level p are concatenated first. After that, a point-wise (1×1) convolution $(conv)$ is applied to reduce the number of features to C_e, i.e. the number of features of each of the input maps (cf. Section 3).

In preliminary experiments, similarly to Seichter et al. (2021), we also tested fusion based on Squeeze-and-Excitation blocks (Hu et al., 2018). However, while requiring more parameters, it did not give better results than *mean* and *concatenation* fusion, so that it is not considered in this paper.

4.3 Training and Depth-aware Dice Loss

As in the baseline, the loss minimised in training consists of two terms (cf. Section 3, equ. 1). The component $\mathcal{L}_{pos}$ used to constrain the output of the Position Head is identical to the one used in (Li et al., 2021). However, we modify the term $\mathcal{L}_{seg}$, i.e. the loss applied to the output of the panoptic segmentation. Li et al. (2021) use a loss based on the Dice function (Milletari et al., 2016) which measures the level of agreement of two binary images Pr and Gt of equal size:

$$Dice(Pr, Gt) = \frac{2 \cdot \sum_{j=1}^{N} p_j \cdot g_j}{\sum_{j=1}^{N} p_j^2 + \sum_{j}^{N} g_j^2}, \tag{4}$$

where $p_j \in \{0, 1\}$ is the grey value of the j^{th} pixel in the predicted mask Pr, $g_j \in \{0, 1\}$ is the corresponding grey value in the ground truth mask Gt, and N is the number of pixels in the masks. As the Dice function according to equ. 4 measures similarity, the Dice loss is based on $1 - Dice(Pr, Gt)$.

However, Panoptic FCN trained using the Dice loss for $\mathcal{L}_{seg}$ occasionally delivers instance masks that contain two spatially

separated *thing* objects of the same type if the latter have a similar appearance (e.g. Fig. 1). To address this problem, we introduce a new term into the loss $\mathcal{L}_{seg}$ which utilises depth information to penalise the assignment of a pixel to a *thing* instance if the absolute difference between its depth value and the average depth of the instance according to its extents in the ground truth is large. In this way, the network can learn that pixels within one instance of a *thing* class have similar depth values. In the original Dice Loss, a false positive (FP) pixel p_j in the prediction mask (indicated by $g_j = 0$ and $p_j = 1$) will decrease the output of the Dice function (equ. 4), because that pixel will increase the denominator by 1 while not increasing the numerator. Thus, a FP pixel will increase the loss. Our idea is to increase the loss even further for FP pixels that are at a depth different from the one of the instance. This can be achieved by a loss based on a new depth-aware Dice function *DDice* defined as:

$$DDice(Pr, Gt, d) = \frac{2 \cdot \sum_{j=1}^{N} p_j \cdot g_j}{\sum_{j=1}^{N} \left[p_j \cdot \left(1 + \omega \cdot \bar{d}_j\right)\right]^2 + \sum_j^N g_j^2}, \tag{5}$$

where Pr and Gt are a predicted and a ground truth binary map for a specific *thing* instance, p_j as well as g_j are the corresponding grey values at pixel j, N is the number of pixels in a map, and d is a depth map having the same size as Pr and Gt. The desired depth-awareness is achieved by the factor $\left(1 + \omega \cdot \bar{d}_j\right)$ in the denominator. Here, ω is a hyperparameter modulating the impact of the depth on the loss and $\bar{d}_j$ is based on the difference of the depth d_j of a FP pixel j from the mean depth d_g of the pixels assigned to the instance corresponding to Gt:

$$\bar{d}_j = \left| \frac{d_j - d_g}{\max\left(d_g, d_{max} - d_g\right)} \right| \cdot p_j \cdot (1 - g_j), \tag{6}$$

with

$$d_g = \frac{1}{\sum_{j=1}^{N} g_j} \cdot \sum_{j=1}^{N} g_j \cdot d_j.$$

In equ. 6, d_{max} denotes a hyperparameter corresponding to the maximum possible depth value. As the product $p_j \cdot (1 - g_j)$ is 0 except for FP pixels, the depth-dependent term $\omega \cdot \bar{d}_j$ only decreases the output (and, thus, increases the loss) for FP pixels. Note that for $\omega = 0$, our depth-aware Dice function is equivalent to the Dice function in equ 4.

In the training stage, after estimating K^{st} masks for *stuff* classes and K^{th} *thing* instance masks, the loss term $\mathcal{L}_{seg}$ is evaluated and used to update the network parameters. For the *stuff* classes, we use the standard Dice loss based on equ. 4 to define a term $\mathcal{L}_{seg}^{st}$, whereas for *thing* instances, a loss $\mathcal{L}_{seg}^{th'}$ based on our depth-aware dice loss (equ. 5) is applied. We obtain the following formulation for the loss $\mathcal{L}_{seg}$ in equ. 1:

$$\begin{aligned} \mathcal{L}_{seg} &= \mathcal{L}_{seg}^{st} + \mathcal{L}_{seg}^{th'} = \\ &= \frac{1}{K^{st}} \cdot \sum_{k_{st}=1}^{K^{st}} \left[1 - Dice\left(Pr_{k_{st}}, Gt_{k_{st}}\right)\right] \\ &+ \frac{1}{K^{th}} \cdot \sum_{k_{th}=1}^{K^{th}} \left[1 - DDice\left(Pr_{k_{th}}, Gt_{k_{th}}, X^d\right)\right]. \end{aligned} \tag{7}$$

This is different from Li et al. (2021), who also use the $Dice$ function to model the loss component for the *thing* instances. Equ. 7 gives the loss term $\mathcal{L}_{seg}$ for a single training image with corresponding depth map X^d. $Pr_{k_{st}}$ and $Gt_{k_{st}}$ are the predicted and ground truth maps for the *stuff* class k_{st} for that training image. Similarly, $Pr_{k_{th}}$ and $Gt_{k_{th}}$ are the predicted and ground truth maps for the k_{th} *thing* instance. Note that, due to the way in which the instance centres are initiated at training time (cf. Section 3), for each predicted map it is known to which reference instance it corresponds. There is no need for matching instance predictions to ground truth instance maps to establish which predicted instance map is considered to correspond to the ground truth instance k_{th}. The actual loss used in training is a sum over all images of a minibatch.

The Dice loss and our depth-aware Dice loss are visualised in Fig. 3, where the circle represents the true positive (TP) pixels of an instance and the triangle corresponds to FPs. Our new loss function penalises FP pixels having a large depth difference from the mean of the TPs. The larger the difference in depth between the FP segments and the ground truth, the larger the penalty that is added for this segment.

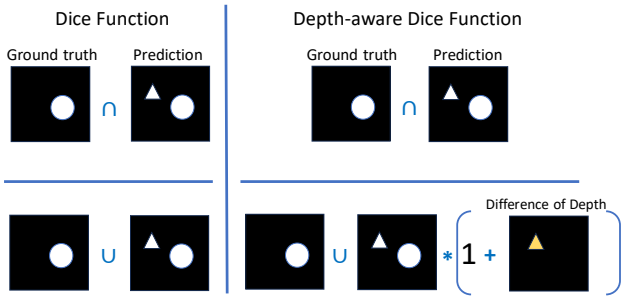

Figure 3. Visualisation of the Dice function (left) and the new depth-aware variant (right). In the latter case, the consideration of depth will increase the penalty for FPs with large depth differences compared to the TPs.

5. Experiments

We start the presentation of our experiments by introducing the experimental setup in Section 5.1. The results achieved by our method are described in Section 5.2, while Section 5.3 presents two ablation studies.

5.1 Experimental Setup

5.1.1 Dataset: We perform our experiments on the Cityscapes dataset (Cordts et al., 2016). It consists of 5k stereo image pairs showing various street scenes in Germany. The size of all images is 1024×2048 pixels, and panoptic labels are provided for the left image of every stereo pair, considering $K^{st} = 11$ *stuff* and $N^{th} = 8$ *thing* classes. The dataset is split into training, validation and test sets. We used the training set consisting of 2975 images for training our method. As the reference is unavailable for the test set, we followed the experimental protocol of our baseline methods (Lipson et al., 2021; de Geus and Dubbelman, 2023) and used the validation set, consisting of 500 image pairs, for testing.

The Cityscapes dataset also provides disparity maps for every stereo pair, computed with a variant of Semi-global Matching (SGM) (Cordts et al., 2016; Hirschmuller, 2007), from which we derived depth maps. However, we found the SGM-based depth maps to contain a considerable number of incorrect depth estimates and relatively large regions without any meaningful depth values. Thus, the relevance of these depth maps for the

classification was expected to be small, which was confirmed in preliminary experiments. We decided to generate better depth maps, using RAFT-stereo (Lipson et al., 2021). The resulting depth maps, containing depth values in $[m]$, were post-processed by filtering out unreasonably small or large depth values, setting depth values smaller than $d_{min} = 1\,m$ and larger than $d_{max} = 500\,m$ to 0.

5.1.2 Experimental Protocol: Training is based on minimising the loss according to equs. 1 and 7. Similarly to Li et al. (2021), we use Stochastic Gradient Descent with a weight decay of 10^{-4} and a momentum of 0.9 for that purpose. We also follow the baseline method by applying data augmentation, using minibatches consisting of patches of 512×1024 pixels. These patches are randomly cropped from the input after scaling the images and depth maps by a random factor $f \in [0.5, 2]$ and applying a random horizontal flip. The depth values are also scaled to maintain the ratio between the extents in the image plane and the depth. The input patches are normalised by subtracting the channel-wise means μ and dividing the differences by the channel-wise standard deviations σ, these values being computed from all images and depth maps in the training set, respectively. Note that pixels marked as not being in the depth range $[d_{min}, d_{max}]$ are not considered to determine μ and σ. A minibatch that is processed in one training iteration consists of 12 such patches, and 180k such iterations are carried out. The learning rate is initially set to 0.02 and reduced by a factor of 0.9 after every 1000^{th} iteration. The parameters of the colour and depth encoders are both initialised by values obtained from pretraining on ImageNet (Deng et al., 2009). The two hyperparameters introduced in equ. 1 are set to $\lambda_{pos} = 1.0$, and $\lambda_{seg} = 3.0$, and we use $\omega = 3.0$ for the weight associated with the influence of depth equ. 5. These values were determined in preliminary experiments. We apply mean fusion to combine RGB and depth features (cf. Section 4.2) and use a feature dimension of C_e = 256 for the resultant feature maps (cf. Section 3); this is larger than C_e = 64, the value used in (Li et al., 2021). In the following, we refer to the variant of our methodology trained and parameterised as described in this section as *Ours*. All experiments are carried out on a Nvidia A100 GPU with 40 GB memory.

5.1.3 Evaluation Protocol: We follow the evaluation scheme of Kirillov et al. (2019b), using the *panoptic quality* (PQ) as a quality measure:

$$PQ = \frac{\sum_{(Pr,Gt)\in TP} IoU\,(Pr, Gt)}{|TP| + \frac{1}{2}\cdot|FP| + \frac{1}{2}\cdot|FN|}\,, \tag{8}$$

where Pr and Gt are a predicted and a ground truth mask found to correspond to each other and IoU denotes the Intersection over Union of these masks. TP indicates the set of true positive masks, i.e. the set of masks Pr for which a ground truth mask with an $IoU > 50\%$ could be found. Similarly, FP and FN denote the set of false positive masks (e.g. predicted *thing* instances without a match in the ground truth) and false negative masks (e.g. ground truth *thing* instances without correspondence in the predictions). In addition to PQ we also report the panoptic quality obtained only for *thing* (PQ^{th}) and *stuff* classes (PQ^{st}). Details about the way in which PQ, PQ^{th} and PQ^{st} are determined can be found in (Kirillov et al., 2019b).

5.2 Results and Discussion

Tab. 1 shows the quality metrics achieved by our method (*Ours*) on the Cityscapes validation set, and Fig. 4 shows some qualitative examples. The table also presents the results for two baseline methods, (Li et al., 2021) and (de Geus and Dubbelman, 2023). We chose (Li et al., 2021) for comparison because our method is an extension of that method, so that the comparison will highlight the impact of our modifications. We trained that baseline using the protocol described in Section 5.1.2, i.e. using $C_e = 256$. Fig. 4 also shows some qualitative results produced by this baseline method. The second baseline (de Geus and Dubbelman, 2023) was chosen because it tries to solve the same problem of (Li et al., 2021) as our method, but using a different strategy (and also not using depth). In this case, the quality indices are those published in (de Geus and Dubbelman, 2023), which are based on the same definition of training and test images as ours.

Method	PQ	PQ^{th}	PQ^{st}
(Li et al., 2021)	60.4	53.6	65.4
(de Geus and Dubbelman, 2023)	60.8	54.7	65.3
Ours	**62.6**	**56.2**	**67.3**

Table 1. Panoptic Quality for all classes (PQ) and for *thing* (PQ^{th}) and *stuff* (PQ^{st}) classes achieved by our method and two baselines. All values are given in [%].

Note that the values for (Li et al., 2021) are better than those published in the original paper, probably due to the use of another value for the feature dimension C_e (the minibatch size and number of training iterations we used were also different). Compared to (Li et al., 2021), the results of (de Geus and Dubbelman, 2023) are slightly better for PQ^{th} and PQ, but slightly lower for PQ^{st}. Our method outperforms both methods in all quality indices, i.e. both for *thing* and *stuff* classes. Compared to (Li et al., 2021), the improvement for *thing* classes is more pronounced (+2.6%) than the one for *stuff* classes (+1.9%), yielding a total improvement in PQ of +2.2%. Compared to (de Geus and Dubbelman, 2023), where the problem of merged instances is tackled as well, the improvement for *thing* classes is still +1.5%. In total, the gain in PQ of our method is +1.8%. We believe that these numbers confirm the hypothesis made in the beginning, namely that the consideration of depth supports the differentiation of *thing* instances of similar appearance and, thus, improves the quality of panoptic segmentation. This positive effect can also be seen in the areas highlighted by red boxes in Fig. 4. Whereas (Li et al., 2021) tends to assign pixels located on visually similar but distinct *thing* instances at different depth levels to the same instance mask, our method mitigates this effect and is able to differentiate such instances.

However, there are also some remaining problems. In our depth-aware Dice Loss function, the difference in the distances between camera and objects is used to identify distinct *thing* instances. As a result, instances looking similar and occurring at similar distances remain problematic, as shown in Fig. 5. In this case, the depth information does not lead to a further penalisation of FP instance pixels in the loss function compared to the plain dice loss, leading to problems that are similar to those of the baseline (Li et al., 2021). We aim to address this problem in future work, e.g., by including a penalty based on the 3D distance between distinct instances in the loss function instead of only relying on the difference in depth.

5.3 Ablation Studies

5.3.1 Influence of the weight ω: In this section, we investigate the influence of the weight ω associated with the influence of depth information in our loss function (cf. Sec. 4.3) on the

Figure 4. Qualitative examples of results achieved on the Cityscapes data. Top: results of the Panoptic FCN baseline (Li et al., 2021), bottom: results of our method. Different colours are used to identify *stuff* class or *thing* instance to which a pixel is assigned, and the resultant label maps are superimposed to the RGB input images. The red boxes highlight examples in which the baseline erroneously merged two *thing* instances, whereas our method separated them correctly.

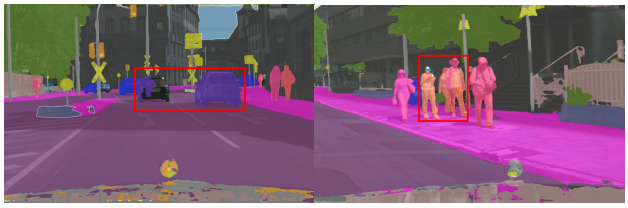

Figure 5. Failure cases of our method: the red boxes indicate instances erroneously merged by our method. The merged instances occur at a similar depth.

performance of our method. For this purpose, we trained our method several times in the way described in Section 5.1.2, using different values for ω, namely $0, 1, 3, 5$ and 10. Note that $\omega = 3$ is the setting analysed in the previous section. In the setting with $\omega = 0$, the original Dice loss is used for training, i.e., depth is used as an additional input, but training is based on the loss used in (Li et al., 2021). The results are shown in Tab. 2. In general, the differences in PQ are in the order of 1%. The influence of ω is larger for *thing* classes than it is for *stuff*. Using $\omega = 3$ achieves the best results with respect all compared quality metrics. It is particularly interesting to compare this result to the one achieved when the original Dice loss is used ($\omega = 0$). The quality indices in Tab. 2 show that using the extension of the Dice Loss leads to an improvement of PQ independently from the value of ω used, and in case of $\omega = 3$, the improvement is +1.5%. Interestingly, the PQ values for both, the *thing* and *stuff* classes are positively affected, even though the depth-aware Dice loss is only applied to *thing* instances in training; probably a smaller number of FP instances leads to a lower error rate for *stuff* pixels, which would affect the PQ^{st} metric via the IoU values in equ. 8. Nevertheless, the improvement in PQ^{th} (+2.7%) is larger than the one in PQ^{st} (+1.2%), probably for that very reason. On the other hand, the PQ metric achieved using $\omega = 0$ in Tab. 2 is still slightly better than the one reported for both baselines in Tab. 1, which is largely due to an improvement of the segmentation quality for *stuff* classes, as indicated by the PQ^{st} values. We can conclude that just using depth as an additional input improves the results slightly, mainly for *stuff* classes; introducing the depth-aware Dice loss in training further improves the results, in this case with a larger impact on the *thing* classes.

ω	PQ	PQ^{th}	PQ^{st}
0	61.1	53.5	66.1
1	61.5	53.3	67.5
3	62.6	56.2	67.3
5	62.1	55.0	67.2
10	61.6	54.8	66.6

Table 2. Quality metrics [%] achieved when training our method with different values of the hyperparameter ω in equ. 5.

5.3.2 Comparison of fusion schemes: In this section, we investigate the influence of the fusion scheme used for combining features extracted from the RGB images and the depth maps (cf. Section 4.2). For this purpose, we compare the results achieved using the mean fusion scheme (equ. 2), which were already discussed in Section 5.2, to those achieved when applying fusion based on concatenation. In order to obtain the latter, another model was trained, using the protocol described in Section 5.1.2 but replacing mean fusion by the fusion scheme according to equ. 3. The results are shown in Tab. 3. The quality indices in Tab. 3 indicate that mean fusion is to be preferred: the PQ is better by 1.1% when using mean fusion, and the other indices are also higher for that variant. For *thing* instances (PQ^{th}), the difference is 2.1%.

Fusion	PQ	PQ^{th}	PQ^{st}
mean	62.6	56.2	67.3
concatenation	61.5	54.1	67.0

Table 3. Quality metrics [%] achieved when using different fusion schemes for combining RGB and depth features: *mean* fusion (equ. 2) and fusion by *concatenation* (equ. 3). The values for mean fusion are identical to those in Tab. 1.

6. Conclusion

In this paper, we present a new CNN-based method for panoptic segmentation which combines colour and depth information to overcome problems of existing methods based on RGB images only. Depth is considered in two ways. On the one hand, depth is processed along with RGB images in separate network branches, and the resultant feature maps are combined in a late fusion approach. On the other hand, our method is based on a new depth-aware dice loss term which penalises the assignment of pixels to the same *thing* instance based on the difference

between their associated depth values. Experiments carried out on the Cityscapes dataset show that the proposed method outperforms the baseline method in terms of panoptic quality by $+2.2\%$ in total and by $+2.6\%$ and $+1.9\%$ for *thing* and *stuff* classes, respectively. The improvements in *thing* classes are mainly achieved by a reduction in the number of objects that are erroneously merged into one *thing* instance. Our results confirm that it is beneficial to consider explicit 3D information about the scene in panoptic segmentation.

As we use the difference depth to compute a penalty term in our loss function, the correct segmentation of distinct objects of similar appearance located at the same depth remain a challenge. We want to address this problem in future work by including a penalty term into the loss function that based on the 3D distance between distinct objects. Moreover, we plan to extend the presented method by incorporating temporal information, i.e., by using sequences of images with associated depth maps instead of data acquired at a single point in time.

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.

de Geus, D., Dubbelman, G., 2023. Intra-batch supervision for panoptic segmentation on high-resolution images. *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 3165–3173.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., Fei-Fei, L., 2009. Imagenet: A large-scale hierarchical image database. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 248–255.

Hirschmuller, H., 2007. Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2), 328–341.

Hu, J., Shen, L., Sun, G., 2018. Squeeze-and-excitation networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 7132–7141.

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.

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., 2017a. Feature pyramid networks for object detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2117–2125.

Lin, T.-Y., Goyal, P., Girshick, R., He, K., Dollár, P., 2017b. Focal loss for dense object detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2980–2988.

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 Vision (3DV)*, 565–571.

Narita, G., Seno, T., Ishikawa, T., Kaji, Y., 2019. Panopticfusion: Online volumetric semantic mapping at the level of stuff and things. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 4205–4212.

Seichter, D., Fischedick, S. B., Köhler, M., Groß, H.-M., 2022. Efficient multi-task rgb-d scene analysis for indoor environments. *International Joint Conference on Neural Networks (IJCNN)*, 1–10.

Seichter, D., Köhler, M., Lewandowski, B., Wengefeld, T., Gross, H.-M., 2021. Efficient RGB-D semantic segmentation for indoor scene analysis. *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 13525–13531.

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 Conference on Computer Vision and Pattern Recognition (CVPR)*, 5463–5474.

Wu, S.-C., Wald, J., Tateno, K., Navab, N., Tombari, F., 2021. Scenegraphfusion: Incremental 3d scene graph prediction from rgb-d sequences. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 7515–7525.

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 Conference on Computer Vision and Pattern Recognition (CVPR)*, 8818–8826.

Yu, Q., Wang, H., Kim, D., Qiao, S., Collins, M., Zhu, Y., Adam, H., Yuille, A., Chen, L.-C., 2022. Cmt-deeplab: Clustering mask transformers for panoptic segmentation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2560–2570.

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., Krähenbühl, P., 2019. Objects as points. *arXiv preprint arXiv:1904.07850*.