Efficient Multi-task Progressive Learning for Semantic Segmentation and Disparity Estimation

Hanz Cuevas-Velasquez[a], Alejandro Galán-Cuenca[b], Robert B. Fisher[c], Antonio Javier Gallego[b,*]

[a]*Max Planck Institute for Intelligent Systems, 72076, Max Planck Ring 4, Tuebingen, Germany*
[b]*University Institute for Computing Research, E-03690, San Vicente del Raspeig, Alicante, Spain*
[c]*The University of Edinburgh, School of Informatics, University of Edinburgh, Edinburgh, EH8 9AB, UK*

Abstract

Scene understanding is an important area in robotics and autonomous driving. To accomplish these tasks, the 3D structures in the scene have to be inferred to know what the objects and their locations are. To this end, semantic segmentation and disparity estimation networks are typically used, but running them individually is inefficient since they require high-performance resources. A possible solution is to learn both tasks together using a multi-task approach. Some current methods address this problem by learning semantic segmentation and monocular depth together. However, monocular depth estimation from single images is an ill-posed problem. A better solution is to estimate the disparity between two stereo images and take advantage of this additional information to improve the segmentation. This work proposes an efficient multi-task method that jointly learns disparity and semantic segmentation. Employing a Siamese backbone architecture for multi-scale feature extraction, the method integrates specialized branches for disparity estimation and coarse and refined segmentations, leveraging progressive task-specific feature sharing and attention mechanisms to enhance accuracy for solving both tasks concurrently. The proposal achieves state-of-the-art results for joint segmentation and disparity estimation on three distinct datasets: Cityscapes, TrimBot2020 Garden, and S-ROSeS, using only 1/3 of the parameters of previous approaches.

Keywords: Computer Vision, Stereo Vision, Semantic Segmentation, Joint learning, 3D modelling, Multi-task, Disparity estimation

[*] Corresponding author.

Email addresses: hanz.cuevas@tuebingen.mpg.de (Hanz Cuevas-Velasquez), a.galan@ua.es (Alejandro Galán-Cuenca), rbf@inf.ed.ac.uk (Robert B. Fisher), jgallego@dlsi.ua.es (Antonio Javier Gallego)

1. Introduction

Disparity estimation and semantic segmentation are fundamental problems in computer vision. The goal of disparity estimation is to find the pixel correspondences from a rectified pair of stereo images. On the other hand, semantic segmentation assigns class labels to each pixel in the image. Both tasks, individually, have been intensively investigated [1, 2]. Currently, solutions based on Deep Learning and, more specifically, on Convolutional Neural Networks (CNN), are the predominant approaches for both tasks due to their good performance [3, 4].

Both disparity estimation and semantic segmentation are heavily used in scene understanding, autonomous driving, and robotics [5, 6]. However, running state-of-the-art methods from both tasks independently [7, 8] essentially doubles the computational load, which may be prohibitive for some applications [9, 10]. Therefore, a method that combines both tasks into a single model can potentially reduce the computational burden, and thus allow it to be embedded in portable devices or executed in real-time. This type of approach is called multi-task learning [11], an area that is almost unexplored in the field of joint segmentation and disparity estimation [12, 13], unlike segmentation and monocular depth estimation [13, 14, 15].

In the literature, many methods for disparity estimation and segmentation can be found [2]. However, compared to the extensive literature that addresses these tasks separately (which will be reviewed in the next section), there are few works about solving them simultaneously using data-driven techniques [13, 12, 15]. This is mainly due to two reasons: segmenting an image and finding the disparity map are complex tasks *per se* that require many training examples to learn meaningful representations. This problem is worsened by the scarcity of datasets composed of stereo pairs that have both depth and semantic ground truths, with which to train multi-task proposals.

This paper presents a multi-task approach for disparity estimation and semantic segmentation that takes advantage of the information extracted from the different tasks and combines them progressively at the feature level using self-attention in order to improve the final results and make the training process more efficient. As will be seen in the Experiments section (Sec. 5), learning both tasks simultaneously allows using additional information from the stereo pair to improve the segmentation and, at the same time, using the segmentation to improve the calculation of disparity.

The proposed method uses a Siamese architecture [16] of backbones to extract feature

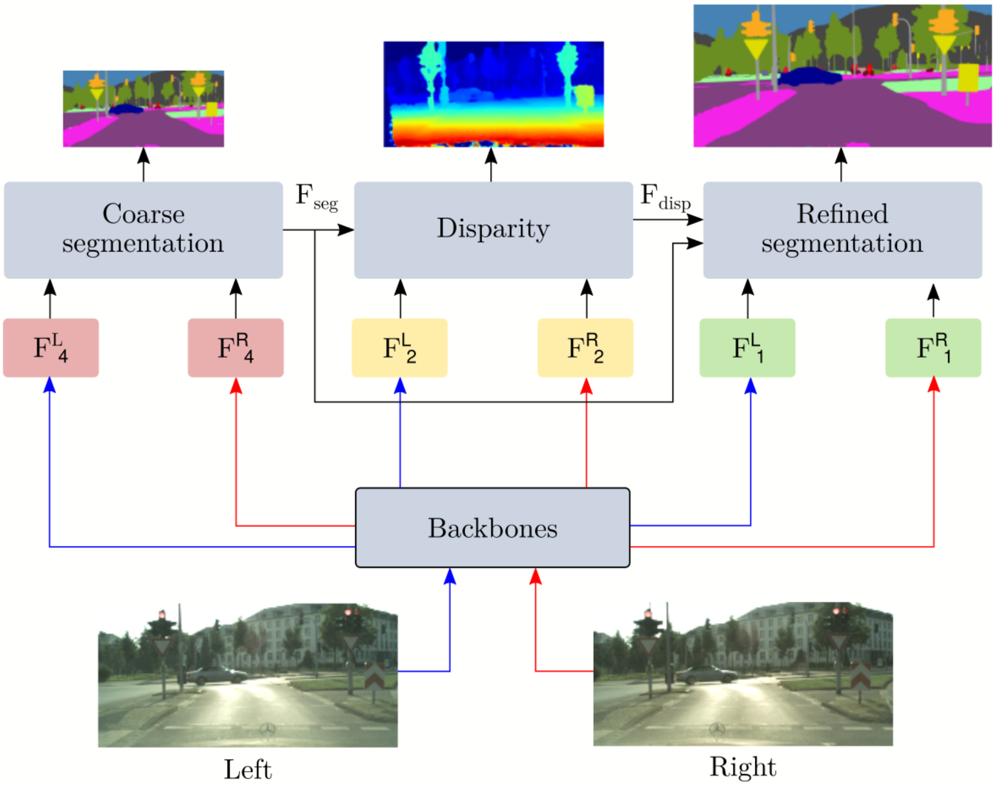


Figure 1: Overview of the proposed architecture. Stereo pair features are extracted from a Siamese backbone structure and used to feed three specialized branches for a multi-scale and multi-task prediction. The first branch generates a coarse segmentation using features from the deepest backbone layer, enriched by high-level image features to enhance context. The second branch computes the disparity map by correlating features from an earlier backbone layer, incorporating coarse segmentation features (F_{seg}) and Spatial Pyramid Pooling for broader feature correlation. The third branch refines the segmentation outcome by leveraging both disparity and segmentation features (F_{seg} and F_{disp}), employing attention mechanisms to focus processing on relevant features.

information at different scales from the left and right input images (see Figure 1). From these features, three specialized branches are connected to carry out the calculation of disparity and segmentation. The first branch extracts a coarse segmentation using low-level features from the backbone, optimized by a 1×1 convolution to reduce parameter depth and hourglass blocks [17] for enhanced context. The second branch calculates the disparity map through feature correlation between stereo images, task-specific features from the coarse segmentation, and the use of Spatial Pyramid Pooling [18] for comprehensive context. The third branch obtains the refined final segmentation by combining high-level features from the backbone and the output of the two previous branches. This last branch employs a self-attention mechanism [19] to focus on the most relevant features from the coarse segmentation and the disparity map. As will be shown, this progressive multi-scale learning allows the segmentation and the disparity estimation to refine each other.

In summary, the main contributions of this paper are:

- An end-to-end novel architecture that successfully learns the semantic segmentation and disparity map together from an input stereo pair. To achieve this, it extracts task-specific features that are shared along with the two tasks progressively.
- State-of-the-art results for multi-task joint learning of disparity and segmentation, and with competitive results against other methods that solve each task individually.
- A network capable of solving both tasks simultaneously using less than 1/3 of the parameters that previous works use to solve only one of these tasks. This reduced number of parameters is useful for systems that have limited resources like robots and autonomous vehicles.

The rest of the paper is structured as follows: Section 2 reviews the related literature to contextualize the work; Section 3 presents the proposed methodology; Section 4 details the experimental setup considered; Section 5 shows the results obtained with the three datasets considered and makes a comparison with the state of the art; Section 6 analyzes in detail the results obtained, and, finally, Section 7 concludes the work by summarising the main insights obtained and proposing future research lines to address.

2. Related work

As previously discussed, many methods for disparity estimation and image segmentation can be found in the literature [1, 2]. The general approach—in both cases—is the use of an encoder-decoder CNN architecture, where the encoder part extracts feature information by reducing (encoding) the dimension of the input image, and the decoder part uses these intermediate encoded representations to predict the disparity map or the semantic segmentation.

For disparity estimation, two main approaches are currently followed in the state of the art. The first one treats disparity estimation as a regression problem, using solely 2D convolutions [20]. The second approach obtains a cost volume by extracting 3D features and discretizing the disparity [21, 22]. A complete review on this topic can be found in Zhou *et al.* [23].

For semantic segmentation, the standard approach is to use a pre-trained network as a backbone (encoder) and add specialized layers (decoder) to obtain the segmentation [24].

Long *et al.* [25] was one of the first successful works using this approach. Later, Deeplab networks introduced *Atrous* convolutions and greatly improved the results with different datasets like Cityscapes [26, 27, 28]. Currently, the state of the art usually employs Deeplab or similar architectures for the encoder part and follows two possible approaches to perform segmentation in the decoder part. One of these is the use of multi-scale images as input in order to optimize the network so it can extract information at different scales, which improves the generality and precision of the results [8, 29, 30]. The other common solution is the use of attention layers, which have been popularized by large language and text-to-image models [31, 32], to capture more context with the relation between the pixels and the classes of the objects to be segmented. This information is usually concatenated to the network or used to weight certain layers [33].

Compared to the number of proposals that address these topics separately, there are only a few works focused on their simultaneous solution [12, 13, 15]. However, as previously argued, for certain applications it can be prohibitive to run two separate methods of this kind in parallel, and assuming unlimited resources is not a realistic or practical approach. Therefore, in these cases, a joint solution that optimizes resources may be of great interest.

One of the main reasons why these tasks are usually approached individually is because of their complexity, which makes it very difficult to overcome the state of the art by following a joint approach. Also, training an end-to-end architecture of this kind requires many ground-truth samples to learn meaningful and generalizable representations. This last requirement worsens the problem since there are very few stereo pair datasets that have both ground truths (i.e., both semantic segmentation and disparity values). Among the few existing datasets with these characteristics, we can find Cityscapes [34], KITTI [35], TrimBot2020 Garden [36], and S-ROSeS [9].

This scarcity of data means that most of the multi-task proposals for this purpose are based on monocular depth estimation [13, 14, 15, 37], which is an ill-posed problem from the geometry perspective [38]. A better solution is to calculate the disparity between two stereo images and take advantage of this information to improve the segmentation [12]. The approaches based on these two paradigms are described in detail below.

Kendall *et al.* [13] proposed a multi-task likelihood to improve the joint learning of monocular disparity, semantic segmentation, and instance segmentation. They used Deeplab v3 [28] as the backbone and three separate branches as decoders, one for each task. Zhang *et al.* [14] also combines feature information from a backbone to estimate monocular depth and segmentation. In this case, they performed a sequence of task-level interactions that evolve along a coarse-to-fine scale space so that the required features can be progressively reconstructed. The approach by Nekrasov *et al.* [15] follows a similar structure for the prediction of monocular depth and segmentation. They introduced a teacher-student approach to be able to learn both tasks when one of them does not have ground truth. Xu *et al.* [37] proposed PAD-Net, a multi-task guided prediction-and-distillation network that leverages a series of intermediate auxiliary tasks, ranging from low to high-level, whose predictions are then distilled to enhance the performance of the final tasks, demonstrating effectiveness across challenging datasets. Yang *et al.* [12] is the only approach that exploits the information of a stereo pair. It first uses a segmentation network as a backbone to extract features from the left and right images of the stereo input and then takes advantage of this information to calculate the disparity and also to refine the segmentation.

All of these approaches demonstrate that combining features from different domains can help to improve similar tasks. However, they do not fully exploit the additional information that a task-specific decoder can learn to improve the other tasks (and vice-versa, that is, combine the information obtained from those other tasks to improve the task solved by the decoder). For example, Yang *et al.* [12] first trains one task and then freezes part of the network and builds on these learned features to solve the other task. According to Baxter [39], learning multiple tasks simultaneously within an environment of related tasks can potentially give much better generalization than learning a single task. This is because models can take advantage of commonalities and differences between tasks.

Our proposal pursues this motivation to learn the segmentation and disparity jointly. Similar to Zhang *et al.* [14], our method uses a refined progressive learning approach rather than concatenating the disparity and segmentation features only once. While they focus on multi-scale refinement for both tasks at the same time, which yields good predictions but is computationally expensive, our method predicts one task per scale. Another main difference with these methods is that our approach employs stereo pairs to obtain the disparity map instead of using a single image, thus avoiding the ill-posed problem and giving more context to the network which can help not only with the calculation of depth but also with segmentation.

Our approach also takes inspiration from the single-task network RefineNet [40], where the output is refined using different feature scales. Similar to this method, our method separates the backbone into blocks. Each block outputs features at different scales which are further processed by specialized convolutional layers. RefineNet focuses only on segmenting an image, therefore, they use these convolutional layers to learn residual information at different scales that is subsequently added. In our case, each scale outputs either a disparity map or a segmentation, so each convolutional layer provides feature information for a specific task. To share this information with other scales (organized in branches in our implementation) and guarantee feature propagation, these features are concatenated rather than being used like a residual connection, similar to a DenseNet block in [41].

One last differential aspect to highlight is that the entire architecture is trained to solve both tasks at the same time, following an end-to-end fashion and propagating the information from the obtained disparity to calculate the segmentation (and vice-versa). It allows the architecture to combine the specific information calculated for each task, searching for commonalities and differences, as proposed by Baxter [39], in order to collaboratively improve the final result.

3. Proposed method

This section describes the proposed architecture and elaborates on how to use feature information from one task to improve other tasks using a coarse-to-fine structure with a binocular stereo input.

3.1. Network architecture

Before describing the network in detail, a brief overview of its main parts is given, as well as the general details of the different layers used. The proposed network consists of a Siamese architecture of backbones and a decoder with three specialized branches, as shown in Figure 1. The first branch obtains a coarse segmentation and outputs task-specific features (F_{seg}) that are used as input for subsequent branches. The second branch outputs the disparity of the stereo pair and its own task-specific features (F_{disp}). The third branch uses the previous features (F_{seg} and F_{disp}) to calculate the refined segmentation. As is common in stereo vision, disparity is calculated with respect to the left image and, for consistency, also segmentation. This is why these high-level features are extracted only from the left image, although information from the right image is also reinforced in early stages of the architecture using the F_1^R, F_2^R, and F_4^R features of the backbone.

Unless otherwise stated, each convolution layer used in the network has a kernel size of 3×3 and is followed by a ReLU activation function [42] and a batch normalization operation [43]. Following [22, 44], after concatenating any 2 or more features, they are processed with a series of top-down/bottom-up convolutions known as *hourglass* (encoder-decoder) to learn more context information. Each hourglass block is composed of 3 convolutions and 3 deconvolutions with residual connections [17].

It should also be noted that in order to obtain a prediction with almost the same dimension as the input without this implying an increase in the necessary resources, convolutions with a large field of view are used to extract high-level features that are concatenated with each of the branches before generating the prediction.

The following sections describe the different parts of the proposed architecture in detail. Appendix A provides an in-depth illustration of the complete architectural framework, showcasing the detailed composition of its layers. Furthermore, it includes tables that precisely specify the characteristics of each component.

3.2. Backbone feature extraction

As previously indicated, the network initially uses a Siamese structure [16] for feature extraction. To determine this architecture as well as the backbone to be used, a series of preliminary experiments were carried out (including networks such as DenseNet-121 [41], EfficientNet-B2 [45], MobileNet v3 [46], ResNet-101, and ResNet-50 [47]; some of these results will be analyzed in the experimentation section), eventually selecting DenseNet-121 to be used as the backbone.

Therefore, two instances of DenseNet-121 [41] with shared weights are used to obtain low and high-level features from the left and right input images. The shared weights approach reduces the number of parameters needed by the network [48] and facilitates obtaining consistent features for both inputs.

The backbones are divided into 5 blocks (see Figure 2) to extract features at different scales. Each block reduces the dimension of its input in half. The dimensions of blocks 0, 1, 2, 3, and 4 respectively are 1/2, 1/4, 1/8, 1/16, and 1/32, with reference to the size of the input image. The features extracted from these blocks will be denoted as F_i^L and F_i^R in the rest of the paper, where i is the block and L and R indicate if the feature belongs to the left or right image. Specifically, the features $F_1^{L/R}$, $F_2^{L/R}$ and $F_4^{L/R}$ (highlighted in Figure 2) are the ones that will be used in the specialized branches described in the next sections.

3.3. Coarse segmentation branch

The first branch (see Figure 3) extracts a coarse segmentation that is used as an auxiliary loss in order to optimize the quality of the final segmentation and also to aid in the disparity calculation [49].

The input of this branch is the concatenated feature maps F_4^L and F_4^R, from Block 4 of the backbone. These features undergo an upscaling by a factor of two, followed by a convolution with a 1×1 kernel to reduce the depth of this concatenation, and thus the

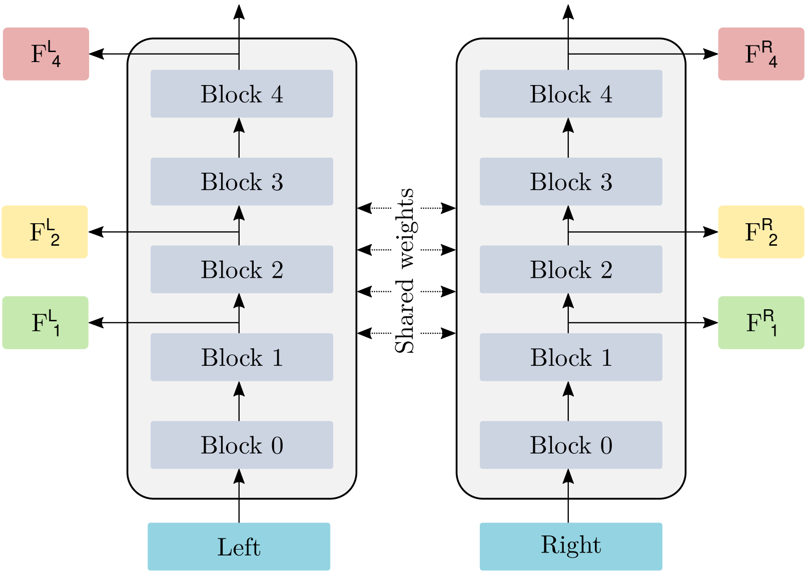


Figure 2: Shared weights backbones structure used to process the input stereo pair. See more details in Appendix A.

number of parameters needed. This information is then processed through an hourglass block in order to learn more context information, whose intermediate output is propagated to the next layer as well as used as a task-specific feature (F_{seg}).

Block 4 returns features with 1/32 of the original image size, causing a loss of detailed information. Following [50], to reduce this problem, high-level features are obtained by processing the left image with a 5×5 convolution (i.e., a larger receptive field). This output is concatenated with the rest of the features in the branch (which are rescaled to match the size of the high-level features) and convolved one more time through the last hourglass block (see Figure 3) to output the final segmentation. The Softmax activation function is used at the pixel level to calculate the probability of each class.

The segmentation branch extracts task-specific features (denoted as F_{seg} in Figure 3), which are used to compute both the disparity and the final refined segmentation. In this way, the network explicitly passes the features learned by this branch for a progressive learning and refinement.

3.4. Disparity branch

The second branch calculates the disparity map from the stereo images (see Figure 4). The inputs of this branch are the features F_2^L and F_2^R obtained from Block 2 of the corresponding left and right backbones. The correlation between these two input features is computed using a correlation layer [20], which takes a block of F_2^L and convolves it in a

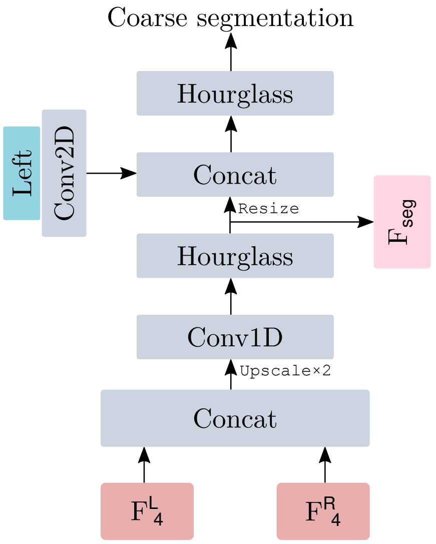


Figure 3: Coarse segmentation branch. It receives as input F_4^L and F_4^R from Block 4 of the backbone. Before predicting the coarse segmentation, the left image is convolved and concatenated to add low-level information. See more details in Appendix A.

neighborhood window around F_2^R. However, it does not capture correlations between distant features because the window size is limited. For this reason, Spatial Pyramid Pooling (SPP) [18] is used on the two feature planes, before extracting their correlations, to obtain multi-scale information, and thus, allow to find distant correlations. The depth of the correlation block is then reduced using a 1×1 convolution and concatenated with the features F_{seg} computed by the coarse segmentation branch (which have the same size thanks to the upscaling applied in the first branch). In this way, the features learned by the segmentation are included to add more information for the disparity estimation.

As before, and following [22, 44], an hourglass block is used after this concatenation to extract more context information. Then, high-level features are added to the network by processing the left input image with a 5×5 convolution, concatenated with the rest of the features in the branch (which are rescaled to match the size), and processed through another hourglass block to calculate the final disparity. Similar to the previous decoder, this branch also outputs an intermediate task-specific feature (F_{disp}), which is used by the refined segmentation branch.

3.5. Refined segmentation branch

The third decoder computes the refined segmentation (see Figure 5). In this branch, Spatial Pyramid Pooling is first used on the backbone features F_1^L and F_1^R. The resulting features are concatenated and processed by an hourglass block. From this result, two sub-

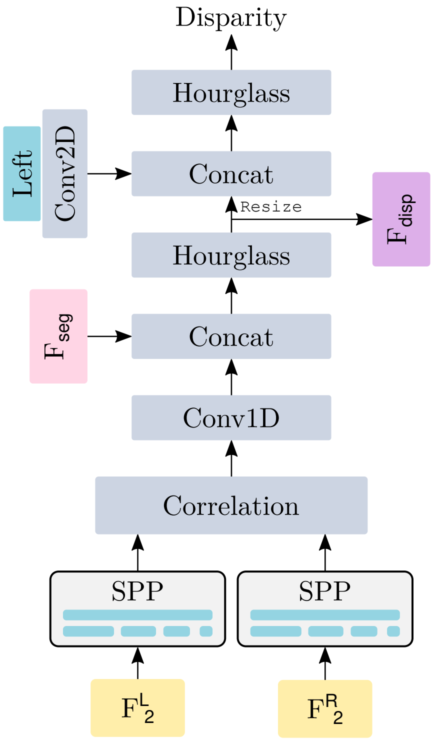


Figure 4: Scheme of the disparity branch. It extracts multi-scale information from backbone blocks F_2^L and F_2^R using Spatial Pyramid Pooling (SPP). This information is processed further and concatenated with the features from the coarse segmentation. See more details in Appendix A.

branches are created: one part is combined with F_{seg} and the other with F_{disp} in order to add task-specific and high-level information from previous branches (note that these features are resized to match the dimensions of the branch's features). A 1×1 convolution with sigmoid activation is used on each part to compute an attention map. This operation focuses on selecting important features from each task (an example can be seen in Figure 6). Each of these subbranches is then multiplied by the features of the previous hourglass block and concatenated with high-level features obtained by processing the left input image with a 5×5 convolution (note that scaling is also applied to adjust the dimensions). Finally, an encoder-decoder block with an output dimension equal to the number of labels and followed by a Softmax activation function is used to obtain the refined segmentation.

3.6. Model optimization

Rather than training each branch separately or a part of the network, freezing the weights and then adjusting the rest (as in the work of Yang *et al.* [12]), the optimization of the complete model is end-to-end.

Each branch optimizes an objective function, either to reduce the segmentation error or the disparity error. For the loss of the two segmentation branches ($\mathcal{L}seg_{cr}$ and $\mathcal{L}seg_{ref}$), the

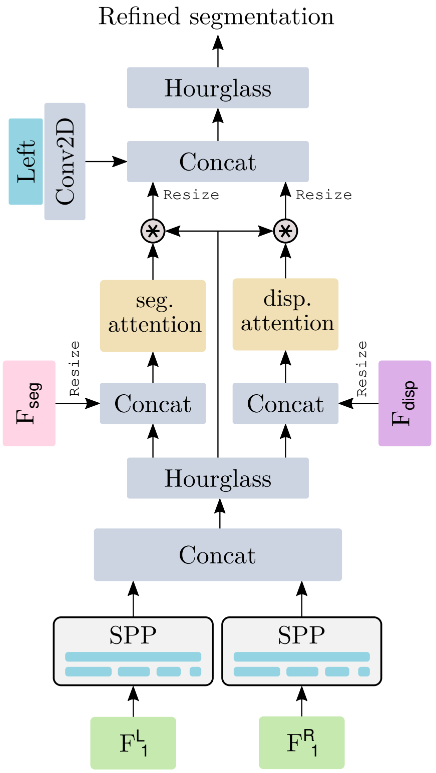


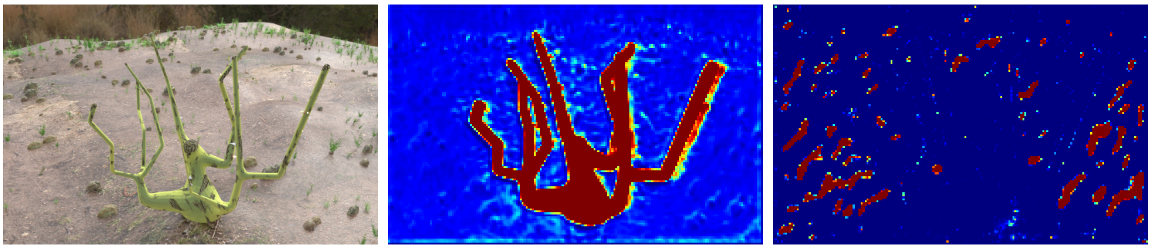
Figure 5: Refined segmentation branch. It concatenates the multi-scale feature information from F_1^L and F_1^R. The features from the previous coarse segmentation branch (F_{seg}) and from the disparity branch (F_{disp}) are also added to the network using an attention layer. See more details in Appendix A.

Cross-entropy and the Lovasz loss are combined [51] to optimize the intersection-over-union between the prediction and the classes labeled in the ground truth of the left input image. For the disparity branch ($\mathcal{L}disp$) the L1 loss is used between the disparity prediction and the labeling of the left image. The final loss consists of the sum of the losses of the 3 branches: $\mathcal{L} = \mathcal{L}seg_{cr} + \mathcal{L}disp + \mathcal{L}seg_{ref}$.

The gradients of each branch loss are propagated across the whole network, for which the standard back-propagation algorithm can be used [52]. Simultaneously optimizing all three losses for the two target tasks allows the network to take advantage of the additional information provided for the other task. For example, knowing the depth can help to differentiate between two objects and, in the same way, knowing the segmentation and class of an object can help to find their correspondences for the calculation of the disparity.

4. Experimental setup

This section describes the datasets used for the experiments as well as the details of the model implementation and the training process. The implementation code is publicly



(a) Left input image. (b) Coarse segmentation attention. (c) Disparity attention.

Figure 6: Example output for an attention map. The intensity of the attention is represented by a color gradient, with red indicating the areas with the highest attention. The map obtained for the coarse segmentation (b) focuses on the inner part of the element to be segmented, where the network is more confident. On the other hand, the attention for the disparity (c) marks textured areas of the background on which the network relies to calculate disparity.

available at: https://github.com/cuevhv/PMT_learning_for_semantic_segmentation_and_disparity.

All experiments were carried out using the Python programming language (v. 3.7) with the public library PyTorch (v. 1.9). The machine used consists of an Intel(R) Core(TM) i7-8700 CPU @ 3.20GHz with 16 GB RAM and an NVIDIA GeForce RTX 2070 with 6GB GDDR6 Graphics Processing Unit (GPU).

4.1. Datasets

For the training and evaluation of the proposed method, three datasets were used: Cityscapes [34], TrimBot Garden [36], and S-ROSeS [9]. These datasets are described in detail below. Table 1 includes a summary of their characteristics and some random image examples can be seen in Figure 7.

Table 1: Summary of the characteristics of the datasets considered, including the type of images, the number of samples, their resolution, the number of segmented classes, and the type of disparity ground-truth (GT) provided.

	Type	**# samples**	**Resolution (px.)**	**Segmented Classes**	**Type of disparity GT**
Cityscapes	Real	5,000	$2,048 \times 1,024$	19	SGM stereo sparse map
TrimBot2020 Garden	Synthetic	12,500	640×480	9	Synthetic dense 3D mesh
S-ROSeS	Synthetic	5,760	720×480	2	Synthetic dense disparity map

Cityscapes [34]: It is an urban scene understanding dataset for pixel-wise semantic segmentation that contains 19 classes plus 1 undetermined class, which—as in the original

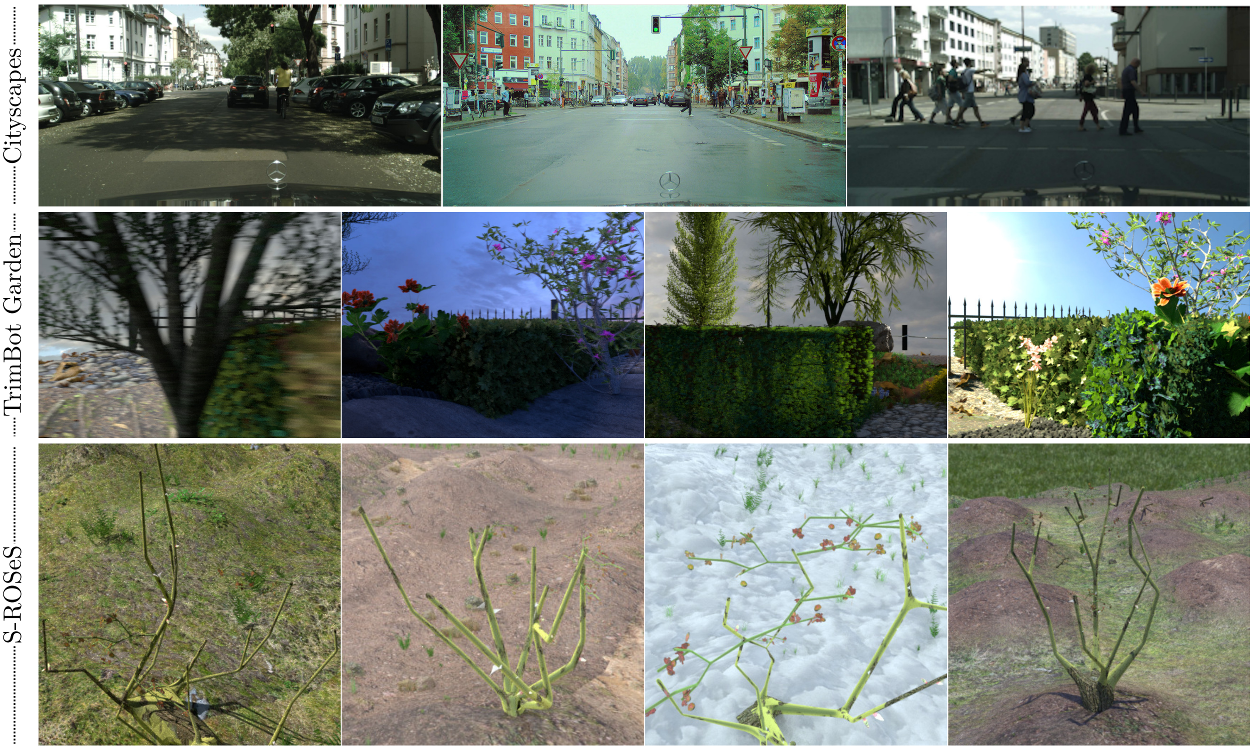


Figure 7: Random examples of the three datasets considered: Cityscapes, TrimBot2020 Garden, and S-ROSeS.

work—is not considered for evaluation. The 19 classes are grouped into 7 categories: flat (including classes such as road and sidewalk), human, vehicles, constructions, objects (with poles, traffic signs, traffic lights, etc.), nature (vegetation and terrain), sky, and void. The disparity ground truth is a sparse map obtained by SGM stereo. This corpus contains 5,000 high-resolution $2,048 \times 1,024$ px images, from which 2,975 (60%) were used for training, 500 (10%) for validation, and 1,525 (30%) for testing (these partitions are those proposed by the authors and have been maintained for experimentation with all the methods considered to make a fair comparison).

TrimBot2020 Garden [36]**:** It consists of images from a synthetic garden under 5 weather conditions. The dataset contains 12,500 stereo pairs with a spatial resolution of 640×480 px, from which 10,000 (80%) were used for training and 2,500 for testing (20%). In this case, 10% of the training set was used for validation, and, as before, the same partitions were kept for all compared methods. The ground truth has 9 pixel-wise segmented classes (grass, ground, pavement, hedge, topiary, rose, obstacle, tree, and background) and their corresponding disparity maps.

S-ROSeS (*Synthetic dataset of Roses for Object Segmentation and Skeletonization*) [9]: It comprises 5,760 synthetic stereo images of rose bushes, divided into 80

% for training and 20 % for testing. Each stereo pair corresponds to a different view of 36 models of synthetic rose bushes. These plants were generated by imitating the morphology and appearance of real rose bushes with a mean height of 0.6 ± 0.2 m. Four backgrounds of exterior environments were created, which vary in background, light conditions, and the position of the sun. Both the left and right input images, as well as the ground truths with the segmentation and the disparity map, were obtained with a resolution of 720×480 px. The depth map was calculated from the simulation of a stereo camera with parallel stereoscopic configuration, sensor size of 32 mm, focal length of 0.032 m, and baseline of 0.03 m.

For the assessment of each dataset, we employed the metrics proposed in the original papers to ensure that our results are directly comparable using the same benchmarks as others in the field. This approach allows for a consistent reference point across different studies. These metrics will be detailed in the results section as they are employed.

4.2. Implementation details

The backbones (DenseNet-121 [41]) were first initialized using the pre-trained weights obtained with the ILSVRC dataset [53, 54]. A fine-tuning process was then applied to the entire network using the specific corpus to be evaluated.

The training lasted a maximum of 70K iterations with a mini-batch size of 16 samples and *early stopping* when the loss did not decrease during 15 epochs. The training of the network parameters was made using Stochastic Gradient Descent (SGD) [52] with Adam [55] as optimizer (learning rate $1e{-3}$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 1e{-8}$).

Data augmentation was used to artificially increase the size of the training set by randomly applying different types of transformations to the original training samples [53, 56]. Specifically, the transformations applied include changes in brightness, contrast, and saturation, the addition of gaussian blur (with a standard deviation in the range $[0.25, 1.15]$), random cropping (512×512 px for Cityscapes and 480×480 px for TrimBot2020 and S-ROSeS), zooming in and out (in the range $[0.7, 1.5]$ with respect to the original image size), and class uniform sampling to handle the class imbalance, as proposed by Choi *et al.* [57]. These augmentations are detailed in Table 2. In each iteration of the training process, and for every image being processed, a set of transformations is randomly selected from the permitted range and applied to the image. This ensures that the data augmentation process introduces a diverse and varied set of modifications, enhancing the robustness and generalizability of the model.

Note that when optimizing two simultaneous tasks, the transformations have to be ap-

Table 2: Transformations considered during the data augmentation process. The dimensions for random cropping vary by dataset†: 512 × 512 for Cityscapes, and 480 × 480 for both TrimBot2020 and S-ROSeS. Additionally, class uniform sampling is tailored to balance the representation of each class within the dataset, without a predefined parameterization range.

Transformation	**Range**
Brightness	[0.5, 1.5]
Contrast	[0.8, 1.2]
Saturation	[0.5, 1.5]
Gaussian blur	[0.25, 1.15]
Random cropping	$\{512 \times 512, 480 \times 480\}^{\dagger}$
Zooming in and out	[0.7, 1.5]
Class uniform sampling	–

plied to both ground truths. In this sense, the operations considered do not alter the segmentation or disparity labeling, with the exception of the zoom, which does modify the disparity. For this reason, and to the best of our knowledge, it is not usually used when training disparity networks (as opposed to segmentation networks, where it is always used). In this work, we propose the use of this augmentation criterion by resorting to simple geometry: The disparity values were re-scaled based on the zooming scale factor, with the new disparity being $disp_{new} = scale * disp_{old}$.

5. Results

This section presents the results obtained with the proposed method following the experimental setup described in Section 4. The first part is devoted to the optimization of the different hyper-parameters of the scheme. After that, the individual results obtained for the three datasets are analyzed and compared with those of the state of the art.

5.1. Hyper-parameter tuning

For simplicity, the initial analysis is presented with Cityscapes as it is the most challenging dataset of those selected, although these experiments were also carried out with the other two with similar results. Table 3 shows a summary of the most relevant results obtained regarding the input and batch size, the data augmentation techniques considered, and the different topologies for the network's backbone. Results are reported in terms of mean class-wise Intersection over Union (mIoU) for the evaluation of semantic segmentation and D-1 error for disparity estimation. The latter calculates the percentage of pixel-wise disparity errors below a threshold, which—as in the related literature—is set to 3 pixels of difference between the ground truth and the estimated disparity.

Table 3: Comparison of the results obtained using different hyperparameters when using the Cityscapes validation set. Coarse segmentation (*Seg cr.*) and refined segmentation (*Seg ref.*) are evaluated considering the IoU figure of merit, and the disparity using the D1-error. The best result per metric is highlighted in bold.

	Input size (px)	Backbone	Batch size	Random cropping	Zoom in/out	Brightness variation	Class sampling	Seg cr. IoU	Seg ref. IoU	Disp. D-1 error
batch & input size	256x256	DenseNet-121	4	✓				63.2	66.0	0.353
	256x256	DenseNet-121	8	✓				64.7	68.3	0.345
	256x256	DenseNet-121	16	✓				65.2	69.6	0.311
	512x512	DenseNet-121	8	✓				67.3	71.3	0.185
	512x512	DenseNet-121	16	✓				70.1	73.2	0.082
	512x512	DenseNet-121	32	✓				68.3	71.1	0.092
data aug.	512x512	DenseNet-121	16	✓	✓			71.7	74.6	0.064
	512x512	DenseNet-121	16	✓	✓	✓		73.0	76.1	0.056
	512x512	DenseNet-121	16	✓	✓	✓	✓	**74.6**	**77.6**	**0.040**
backbones	512x512	EfficientNet-B2	16	✓	✓	✓	✓	72.2	74.7	0.090
	512x512	MobileNet v3	16	✓	✓	✓	✓	69.1	72.0	0.124
	512x512	ResNet-101	16	✓	✓	✓	✓	60.4	64.0	0.453
	512x512	ResNet-50	16	✓	✓	✓	✓	66.6	70.0	0.173
	512x512	ResNet-18	16	✓	✓	✓	✓	62.1	66.7	0.326

Similar to other works that use backbones with batch normalization layers [2, 26], our experiments show that small training batches lead to unstable batch-norm statistics and to poor learning performance. Results show a progressive improvement in the disparity and segmentation when the batch size is increased up to 16, and gets worse after that.

The results also show that larger input sizes provide better overall performance, which may be due to the following two reasons. First, the input size is obtained by randomly cropping the original image. This means that the bigger the extracted chunk, the more context the network sees, which translates into learning better relationships between objects. The second reason is related to the reduction in dimension due to the backbone topology. In the case of ResNet and DenseNet, the backbones are divided into blocks, and each block reduces by half its input size. This means that the feature dimensions of blocks 1 to 5 are 1/2 to 1/32 of the input image size. Therefore, if we use an input size of 256 × 256 px, the output feature from block 5 will be only 8 × 8.

Regarding data augmentation, it is observed how the different techniques progressively improve the segmentation result by up to 4.5 % for the IoU and reduce the disparity error by a total of 0.042. Specifically, the different augmentation processes provide a similar improvement for segmentation. However, for disparity (see Table 3), zooming seems to help the most (reducing the error by 0.018), then class uniform sampling (with a decrease of 0.016), and finally brightness changes (lowering the error by 0.008).

The proposal was also evaluated considering six different backbones: EfficientNet-B2, MobileNet v3, ResNet-101, ResNet-50, ResNet-18, and DenseNet-121. As can be seen, the latter is the one that reports the best results both for segmentation and for calculating the disparity. For this reason, to carry out the rest of the experiments, we will use this backbone together with the best hyper-parameters previously determined: a batch size of 16, an input size of 512 × 512 px, and the use all the types of data augmentation proposed.

5.2. Cityscapes

Once the best configuration of hyper-parameters has been determined, we proceed to evaluate the results obtained with the considered datasets, starting with Cityscapes.

Table 4 compares the proposal with the state of the art in the Cityscapes ranking. For comparison, the highest-scored methods for segmentation were considered, including Panoptic-DeepLab [30], RCNet [3], PSPNet [49], as well as the multi-task approach with the highest score for both tasks [13], which is called MTU. As our approach is based on DenseNet, the work with the highest score that uses this backbone is also included, which is Ladder-style DenseNet [58]. In addition, given that the best-ranked methods are based on Deeplab v3, three versions of this architecture using different backbones (Xception-71, ResNet-101, and ResNet-50) were also included in the comparison. Please note that the column referring to input size denotes the size used during training, where crops of this size are extracted from the original training images. For evaluation, images are processed at their original size without cropping or scaling, ensuring a fair comparison.

When analyzing the results of this table, it is important to keep in mind the column

Table 4: Comparison of the results obtained with the Cityscapes dataset in terms of class and category segmentation IoU, and disparity error. Categories group the 19 classes into 7 groups (see Section 4.1). Results are included for both the test and validation partitions, the latter being the most used in the literature. The best result per metric is marked in bold type and the second-best result is underlined.

Method	Backbone	Train input size (px)	# Params.	Seg. mIoU Class		Seg. mIoU Category	Disp. RMSE
				Val.	Test	Test	Test
Multi-task methods							
Ours	DenseNet-121	512 × 512	18.0M	77.6	76.1	91.0	**3.28**
MTU [13]	Deeplab v3	512 × 512	64.2M	78.1	78.5	89.9	5.88
Semantic segmentation methods							
Panoptic [30]	Deeplab v3	2049 × 1025	46.7M	**81.5**	**84.5**	**92.9**	N/A
RCNet [3]	ResNet-50	512 × 1024	14.1M	70.9	–	–	N/A
PSPNet [49]	ResNet-101	2048 × 1024	54.7M	73.9	–	–	N/A
Ladder-style [58]	DenseNet-121	2048 × 1024	**8.2M**	62.3	–	–	N/A
Ladder-style [58]	DenseNet-169	2048 × 1024	15.6M	75.7	74.3	89.7	N/A
Deeplab v3 [27]	Xception-71	513 × 513	46.7M	79.5	82.1	–	N/A
Deeplab v3 [57]	ResNet-101	768 × 768	64.2M	79.2	–	–	N/A
Deeplab v3 [57]	ResNet-50	768 × 768	45.1M	77.8	–	–	N/A

with the number of parameters, as it represents an efficient solution that addresses both tasks simultaneously. Most of the proposed approaches that yield better results are based on Deeplab v3, which ranges from 41M to 64.2M parameters, depending on the implementation [57]. In our case, the DenseNet-121 backbone [41] has only 8M parameters, and with the addition of specialized decoders for disparity and semantic segmentation, it reaches 18M parameters. This is between 2.6 to 3.6 times fewer than the top-performing state-of-the-art solutions (i.e., Panoptic-DeepLab and MTU).

If we compare the proposal with the best single-task solution (Panoptic), we can observe that it is only 3.9 points lower on the class-level validation set and 1.9 points lower on the category-level test set. As mentioned before, we must highlight that it is a much more efficient network (with 2.6 times fewer parameters) and that it also addresses two tasks simultaneously. When compared to other more efficient single-task solutions, such as Ladder-style or RCNet, with a similar number of parameters, our proposal achieves notably better results.

If we compare it with the multi-task alternative that obtains the best result in the

Cityscapes ranking—i.e., MTU [13]—our network demonstrates a reduced disparity error and superior category-wise IoU, while being only 2.4% less effective in class-wise IoU on the test set and 0.5% on the validation set. Moreover, it is noteworthy that this level of performance is achieved with fewer than a third of the parameters used by MTU. To determine the source of performance differences—whether due to the backbone, training process, or loss function—we conducted two additional experiments (refer to Table 5): 1) We applied their loss function within our architecture (see the second row in the table) and 2) We replaced the MTU backbone with our method's DenseNet-121, while retaining its decoder and uncertainty loss (see the third row). The outcomes indicate that our approach remarkably surpasses MTU in both segmentation and disparity metrics under identical evaluation and training conditions. The integration of its loss function negatively impacts the performance of our model, and the use of our backbone in MTU leads to a significant deterioration in their results. Hence, the superior class-wise IoU of MTU presented in Table 4 should be directly attributed to its backbone, which has a considerably higher parameter count.

Table 5: Comparison of our proposal and MTU [13] on the Cityscapes validation set when using the same backbone (DenseNet-121) and changing the decoder and the loss function. The changes applied are underlined. The original results of our approach and MTU are included for reference (1st and 4th rows).

Method (decoder)	Backbone	Loss	Seg.	Disp.
			mIoU class	D-1 error
Ours	DenseNet-121	Ours	77.6	0.040
Ours	DenseNet-121	MTU	72.2	0.067
MTU	DenseNet-121	MTU	59.6	0.347
MTU	Deeplab v3	MTU	78.1	0.102

For a qualitative analysis of the results, Figure 8 shows four examples (columns) of the segmentation and disparity estimation obtained with our network. The first two rows show the input images, followed by the disparity and segmentation results compared with the GT. In these images, it can be seen that the errors made occur mainly at edges, isolated pixels, or in the lower area that corresponds to the hood of the car, for which the sensor does not seem to provide adequate information.

5.3. TrimBot2020 Garden

In this section, the proposed method is evaluated with the TrimBot2020 Garden dataset and compared with the best-performing state-of-the-art methods for semantic segmentation

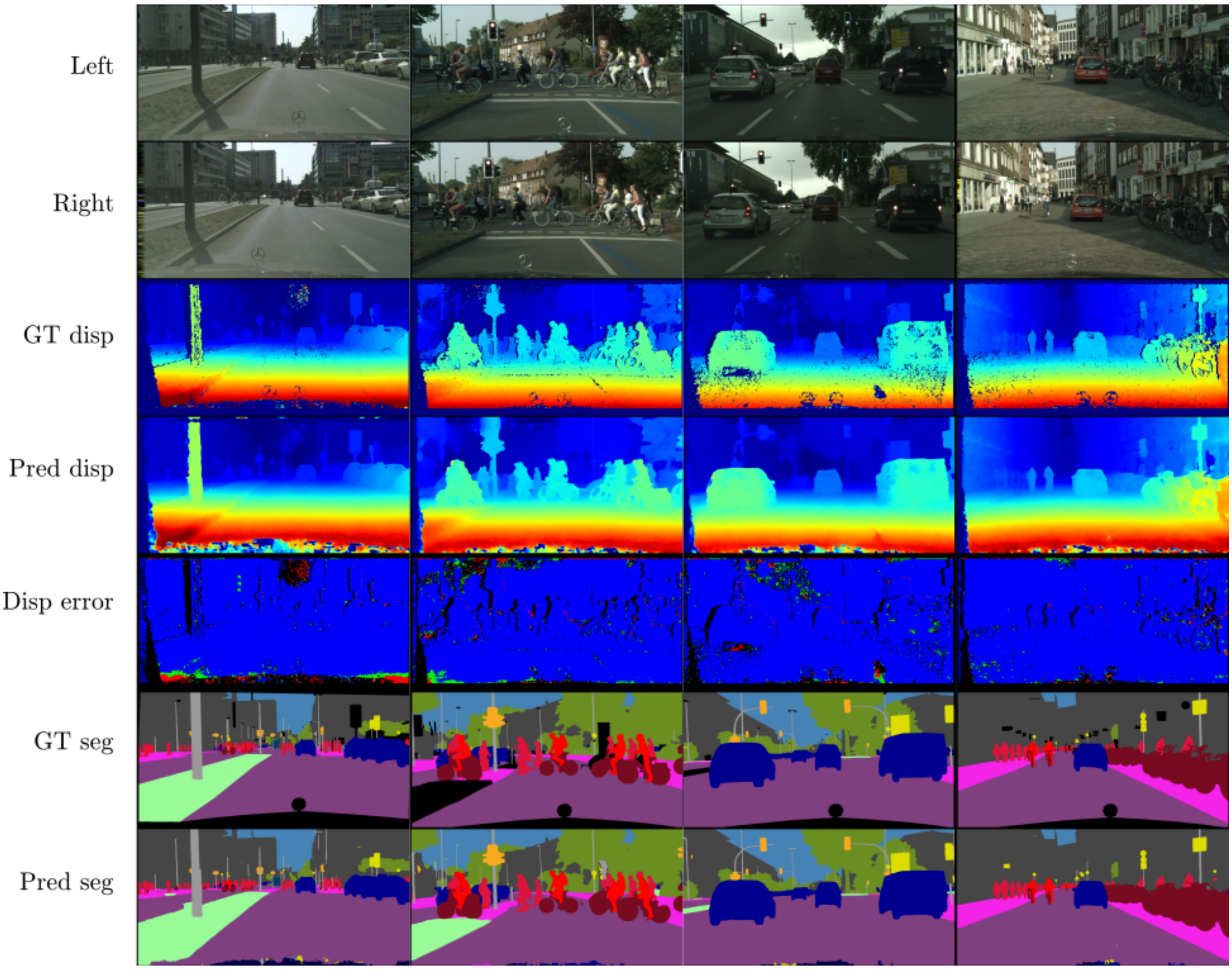


Figure 8: Four examples of results with Cityscapes. The input stereo pair is shown in the first two rows. The ground truth and predicted disparities are shown in rows 3 and 4. The error of the predicted disparity (row 5) encodes the error e in 3 colors: blue ($e < 3$), green ($e < 6$), and red ($e \geq 6$). Rows 6 and 7 show the ground truth and the prediction obtained for the segmentation task. The black color in the disparity and segmentation images means that the ground truth has no disparity or class assigned to that pixel, respectively.

and depth estimation: DTIS [59], HAB [36], and LAPSI360 [60]. DTIS uses a network similar to FuseNet [61], which has as input an RGB image and a depth map, and outputs both the semantic segmentation and a refined version of the depth map. HAB uses ELAS stereo [62] to produce a dense point cloud and DeepLab v3 [28] to obtain the segmentation. The resulting point cloud is denoised with class-specific filters based on the 3D geometry. LAPSI360 focuses on the 3D reconstruction; for this it generates the geometric mesh of the garden using the information from the 10 cameras provided in the dataset.

Following the original HAB paper [36], pixel-wise accuracy was used to evaluate the semantic segmentation. Using the TrimBot2020 Garden dataset allows evaluation of the quality of the 3D reconstruction instead of finding the error of the 3D depth or disparity. To evaluate the goodness of these reconstructions, 2 metrics were used: completeness and 3D accuracy. Completeness is calculated by considering a predicted point as correct if it differs from the ground truth by $\leq 0.05m$, similar to the D-1 error. The 3D accuracy is the distance d in meters, such that 90% of the reconstruction is within d meters of the ground truth mesh. In our case, disparity predictions were transformed into depth maps and back-projected into the 3D space using the camera parameters (provided in the original paper).

Table 6: TrimBot2020 dataset comparison. The segmentation accuracy (Seg acc.) and completeness (comp.) are in percentages. The 3D reconstruction accuracy (Acc. (m)) reports the error in meters, therefore, the lower the value the better. The best result per metric is highlighted in bold.

Method	# Params.	Seg. acc.	3D reconstruction	
			Acc. (m)	Comp.
Ours	**18.0M**	**91.9**	**0.061**	**77.8**
DTIS [59]	207.8M	**91.9**	0.122	66.2
HAB [36]	64.2M†	79.0	0.069	74.0
LAPSI [60]	N/A	–	0.164	23.9

† The number of parameters refers only to the segmentation algorithm, an iterative classical vision algorithm is used for the disparity.

Table 6 shows the results of this comparison. As can be seen, our method obtains a segmentation accuracy similar to DTIS—using 11.5 times fewer parameters—and outperforms HAB. Regarding the 3D reconstruction, our network achieves the best score in both metrics considered. It is 3.8% more complete than the best result (i.e., HAB), and the reconstruction accuracy is almost $0.01m$ better. It is important to highlight that, unlike HAB, our method obtains both results at the same time without applying any heavy point cloud filtering or using 3D features.

Figure 9 shows some results of our network for different views of the TrimBot2020 Garden. The proposed approach, captures details like thin branches not only in the segmentation but also in the disparity. In addition, and for a better qualitative analysis, Figure 10 shows the 3D reconstruction obtained for the entire garden. Here you can see how most errors are made on the edges, corners, or in areas where many fine branches appear. In the latter case, it is mainly due to the error accumulated by combining multiple views in which the completeness of the reconstruction of the branches is not exact.

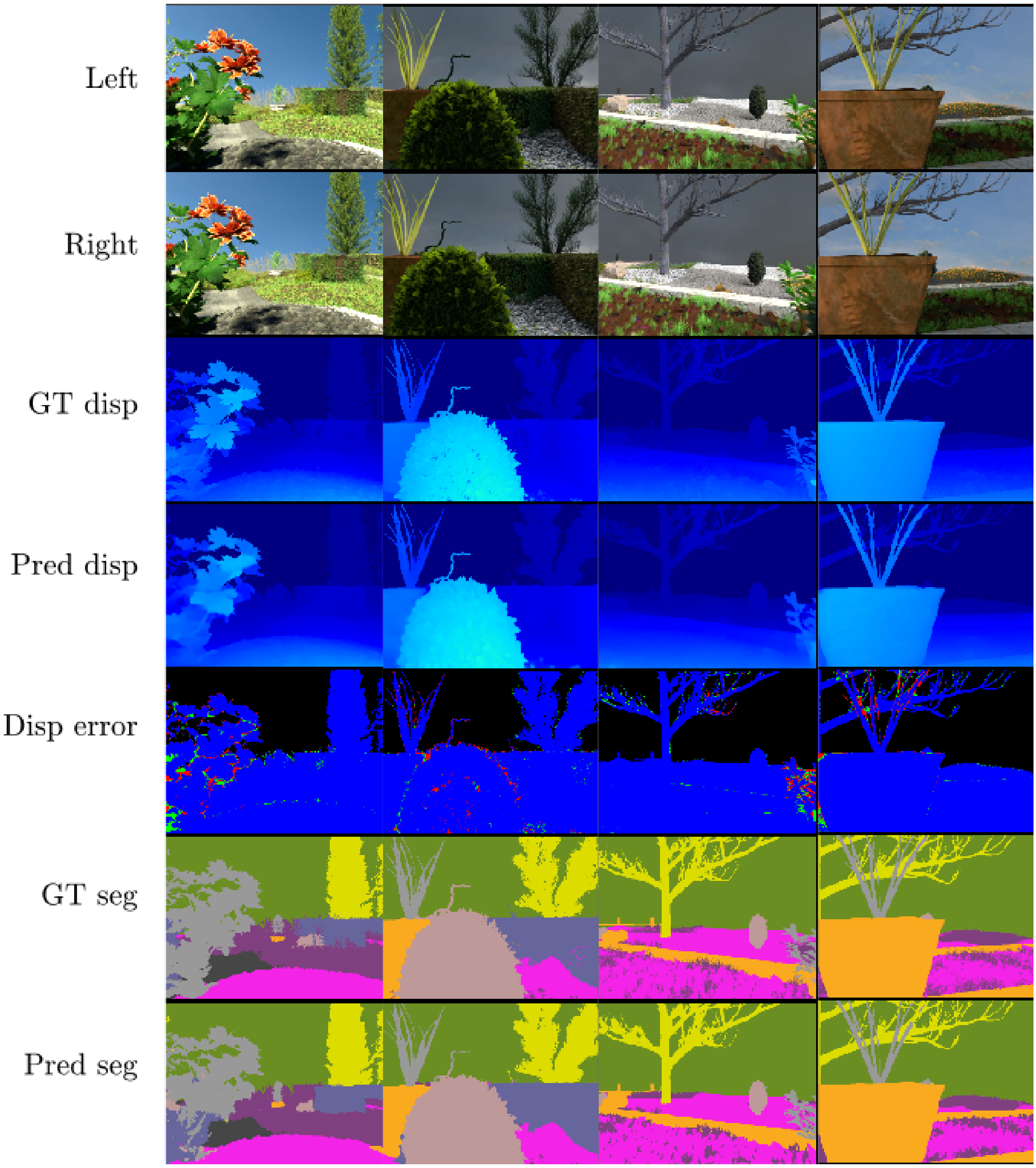


Figure 9: Four examples of the results obtained for the TrimBot2020 Garden. The input stereo pair is included in the first two rows. The ground truth and predicted disparities are shown in rows 3 and 4, respectively. Row 5 shows the error e of the predicted disparity using 3 colors: blue ($e < 3$), green ($e < 6$), and red ($e \geq 6$). Rows 6 and 7 show the ground truth and the prediction obtained for the segmentation task.

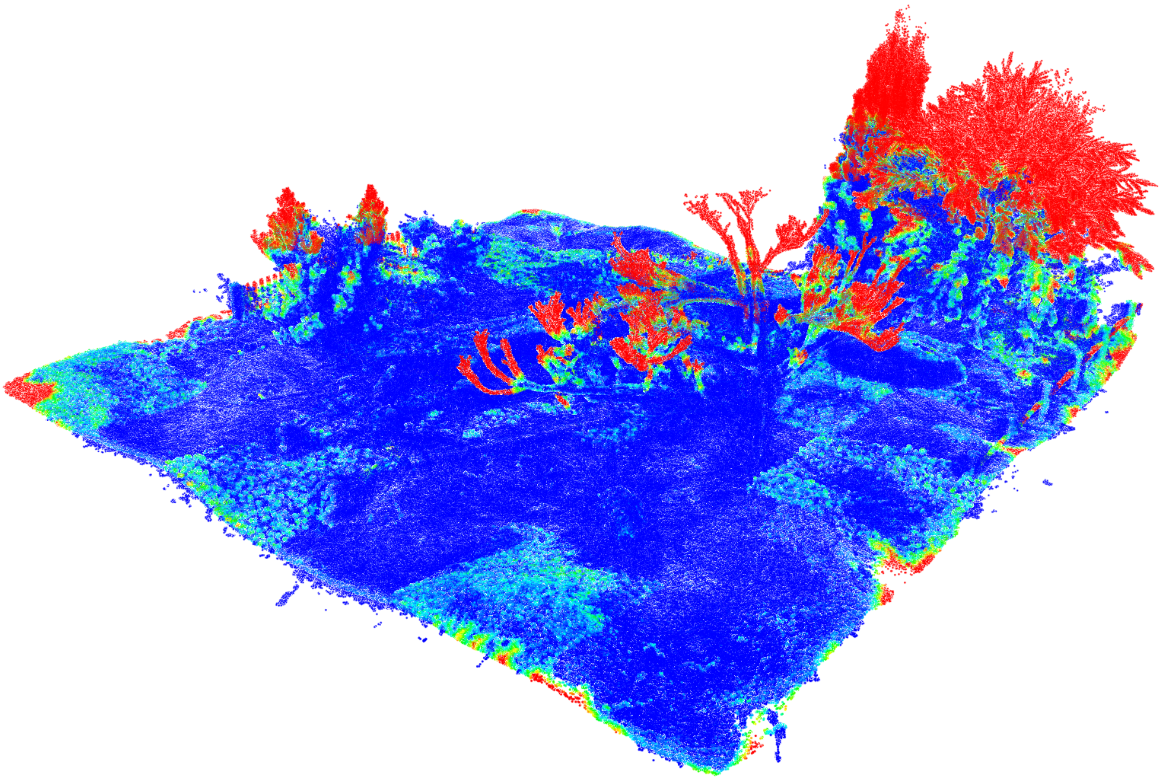


Figure 10: 3D reconstruction obtained for the entire TrimBot2020 Garden dataset. Cold colors indicate well-reconstructed segments. Hot colors indicate missing parts (completeness).

5.4. S-ROSeS dataset

This section focuses on the evaluation of the proposed method using the third dataset—i.e., the S-ROSeS dataset—and on the comparison with different methods from the state of the art for this task: U-Net [63], SegNet [64], and DeepLab v3 [28], as well-known algorithms for segmentation, and the methods FCSN and FCSN-Combined [9], which also compute the disparity. These last two methods are based on a custom architecture for segmentation and on the Block Matching algorithm [65] for disparity. The FCSN-Combined variant performs an iterative process that combines the segmentation and the disparity obtained to improve each other.

The results of this comparison are shown in Table 7. As in the original paper [9], we resort to Precision, Recall, F1, and IoU to assess segmentation, and to Root Mean Squared Error (RMSE) and Squared Relative difference (Sq-Rel) to evaluate disparity. As observed in Table 7, the proposed method obtains the best results in all the metrics considered, remarkably surpassing the rest of the proposals, especially in Precision. The only exception is the number of parameters. However, it is important to note that in the case of U-Net (as well as for SegNet and DeepLab), only the segmentation is calculated, and in FCSN the number of parameters refers only to the segmentation architecture, since it requires the use of another iterative and slow method for calculating the disparity.

Table 7: Comparison of the results obtained for the calculation of the disparity and segmentation with the S-ROSeS dataset. The best results per metric is highlighted in bold.

		Segmentation				Disparity	
Method	# Params.	Precision	Recall	F1	mIoU	RMSE	Sq-Rel
Multi-task methods							
Ours	18.0M	**98.49**	**93.28**	**95.80**	**97.52**	**0.05**	**0.0134**
FCSN [9]	2.9M†	90.47	93.18	91.70	94.16	0.5648	0.0917
FCSN-Comb. [9]	2.9M†	92.86	92.59	92.96	95.07	0.5082	0.0630
Semantic segmentation methods							
U-Net [63]	**1.9M**	91.08	88.50	89.77	90.21	N/A	N/A
SegNet [64]	5.5M	88.84	77.07	81.81	83.60	N/A	N/A
DeepLab v3 [28]	11.8M	76.94	84.63	80.59	79.64	N/A	N/A

† The number of parameters refers only to the segmentation algorithm.

An iterative classical vision algorithm is used for the disparity.

As before, for a qualitative analysis of the results, Figure 11 shows four examples (columns) of the segmentation and disparity estimation obtained with our network. The first two rows show the input images, followed by the disparity and segmentation results compared with the GT. Regarding the disparity, it is observed that errors are only made at the edges of the fine branches. This is better understood if we also analyze the segmentation result, for which both coarse and refined segmentations are shown. The auxiliary loss detects the main parts of the bush, which are later refined to obtain the final segmentation where only a few mistakes are made at the edges of the finer branches. The reported improvement, especially in the detection of thin branches, is very important for the original task of this dataset, since it will allow a better reconstruction of the rose bush and, in this way, avoid possible errors in the pruning process.

6. Discussion

This section presents a more in-depth analysis of the selected architecture and the results obtained. First, we evaluate the improvement provided by the use of progressive feature

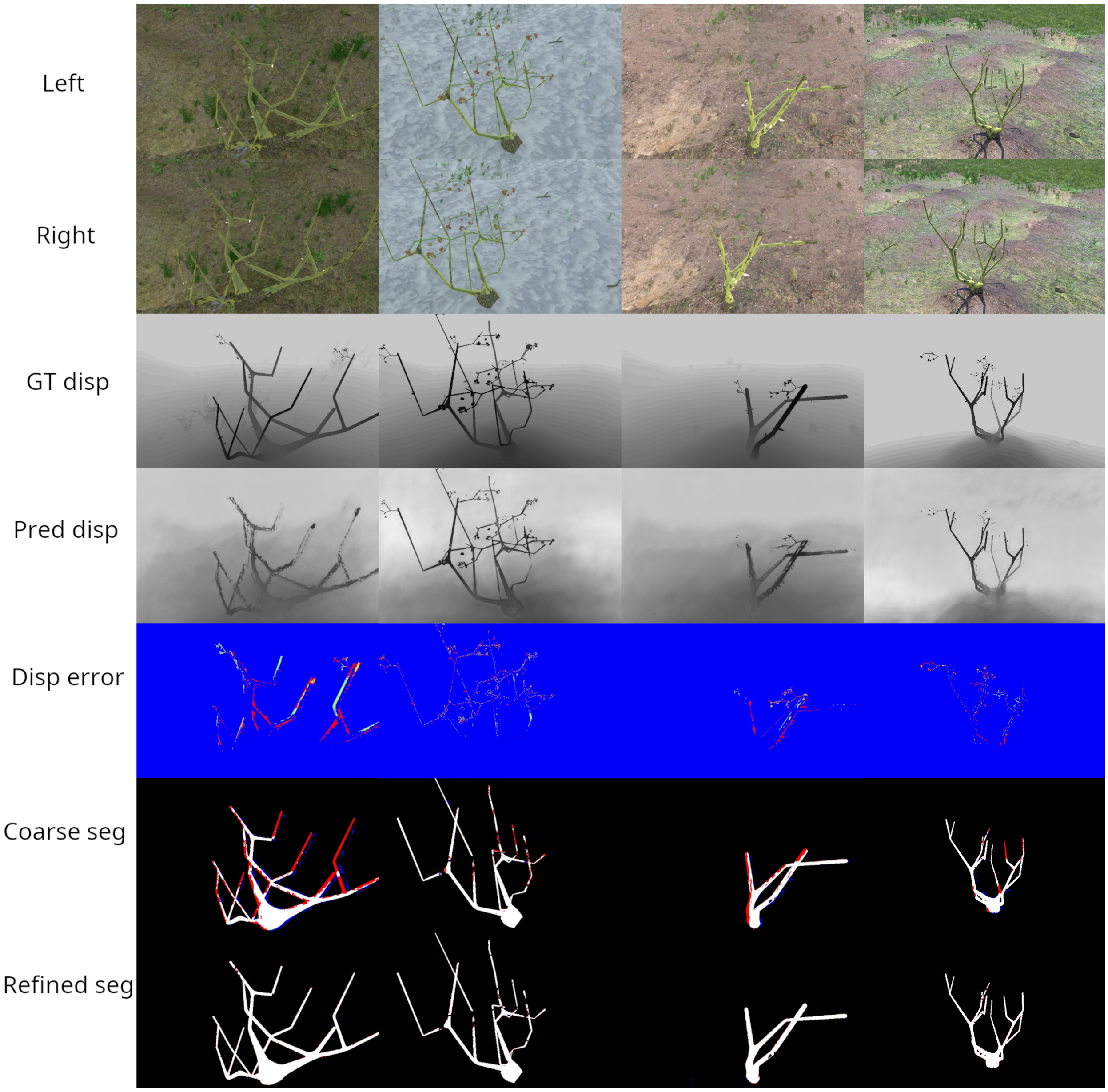


Figure 11: Four examples of the results obtained for S-RoSES. The input stereo pair is included in the first two rows. The GT and predicted disparities are shown in rows 3 and 4. Row 5 shows the error e of the predicted disparity using 3 colors: blue ($e < 3$), green ($e < 6$), and red ($e \geq 6$). Rows 6 and 7 respectively show the coarse and the refined segmentation. In this case, the differences with the GT have been marked with two colors: blue for false positives (predicted branch pixels that do not appear in the GT) and red for false negatives (branch pixels in the GT that have not been detected).

learning with auxiliary loss functions. We also study the relationship between the efficiency and precision of the method and compare it with the other state-of-the-art methods.

6.1. *Effect of task-specific features*

A series of experiments were conducted to test the effectiveness of sharing task-specific features among the branches of the proposed network. Recall that it concatenates feature information from the coarse segmentation branch (F_{seg}) with the disparity, and then uses the segmentation and disparity features (F_{seg} and F_{disp}) to refine the final segmentation (see Figure 1).

To test if this progressive sharing of feature information helps the network to improve the calculation of disparity and/or semantic segmentation, three variations of the proposed method (generated by removing the different connections between the task branches) were trained and tested on the validation set of Cityscapes[1]: 1) All the connections between the task-specific features and branches were removed, 2) The connection between the features from the coarse segmentation (F_{seg}) and the disparity decoder was kept, but the connections of the disparity and the coarse segmentation features (F_{seg} and F_{disp}) with the refined segmentation branch were removed, and 3) The connection F_{seg} from the disparity branch was removed and the connections F_{seg} and F_{disp} with the refined segmentation branch were kept. Table 8 shows the results of this experiment, where the fourth row corresponds to the original model with all the connections.

Table 8: Task-specific feature sharing effect. The first two columns show if the features F_{disp} and F_{seg} were concatenated to the disparity or refined segmentation branches. Seg cr. and Seg ref. IoU show the results for coarse and refined segmentation. The best result per metric is highlighted in bold.

F_{seg} ↓ Br. disp	$F_{seg} \wedge F_{disp}$ ↓ Br. seg ref.	Seg cr. IoU	Seg ref. IoU	Disp D-1 error
✗	✗	73.3	64.8	0.064
✓	✗	73.3	64.7	0.051
✗	✓	72.5	75.7	0.060
✓	✓	**74.6**	**77.6**	**0.040**

The experiments show that the connection between the coarse segmentation features F_{seg} and the disparity branch helps to reduce the disparity error from 0.064 to 0.051 (results from the first and second row of the table). The lowest disparity error, 0.040, is obtained when

[1] We show the results for this dataset as it is the most challenging of the three, although similar experiments were performed with the others with similar results.

F_{seg} and F_{disp} are also shared with the refined segmentation branch (fourth row). This result points out that, even though this last connection does not affect the disparity prediction at inference time, the progressive connection between task-specific features does help in the training process to learn better descriptors.

These experiments also demonstrate that the refined segmentation is improved when it receives the task-specific features F_{seg} and F_{disp}. The improvement ranges from 64.8% to 75.7% when the disparity branch does not receive F_{seg}, and to 77.6% when it does. This also shows that if the disparity branch receives information from the coarse segmentation task, it can learn more meaningful features that can be passed to the refined segmentation branch.

Finally, the experiments show that the refined segmentation performs worse than the coarse segmentation (first two rows of Table 8) when it does not receive the features F_{seg} and F_{disp}. This is probably because it can only use the features at the beginning of the backbone (low-level features), which are not enough for such a complex task as semantic segmentation.

6.2. Effectiveness vs. efficiency

Another important aspect to discuss is the objectives of high efficiency (measured in the number of parameters) and precision of the proposed method, because they are, quite often, opposite objectives, since attempting to improve one of them usually implies a deterioration in the other. From this point of view, this task can be seen as a Multi-objective Optimization Problem (MOP) in which two functions are optimized simultaneously.

The means commonly employed to evaluate this type of problem is the use of the concept of non-dominance: One solution is said to dominate another if, and only if, it is better or equal in each objective function and, at least, strictly better in one of them. The Pareto frontier stands for the set of all non-dominated elements and represents the different optimal solutions to the MOP. The strategies within this set can be considered the best without any particular priority or order among them.

Figure 12 compares the semantic segmentation precision and the efficiency of the algorithms evaluated assuming a MOP scenario. In addition, the Pareto frontier is marked with the non-dominated results, in which four combinations are found to be non-dominated for the Cityscapes dataset (Our proposal, Ladder D121, Ladder D169, and Panoptic), two for the Garden dataset (Our proposal and DTIS), and three for S-ROSeS (Our proposal, U-Net, and FCSN-Comb.). Therefore, for all datasets, our solution is the multi-task approach (marked with triangles in the graph) with the best combination of efficiency and precision.

The rest of the non-dominated results are optimized for a single task (Ladder, Panoptic, and U-Net), or they are multi-task solutions but much less efficient (DTIS and FCSN-Comb.). In the case of FCSN-Comb., it is important to remember the considerable slowdown produced by the combination of segmentation and disparity in an iterative algorithm. 560

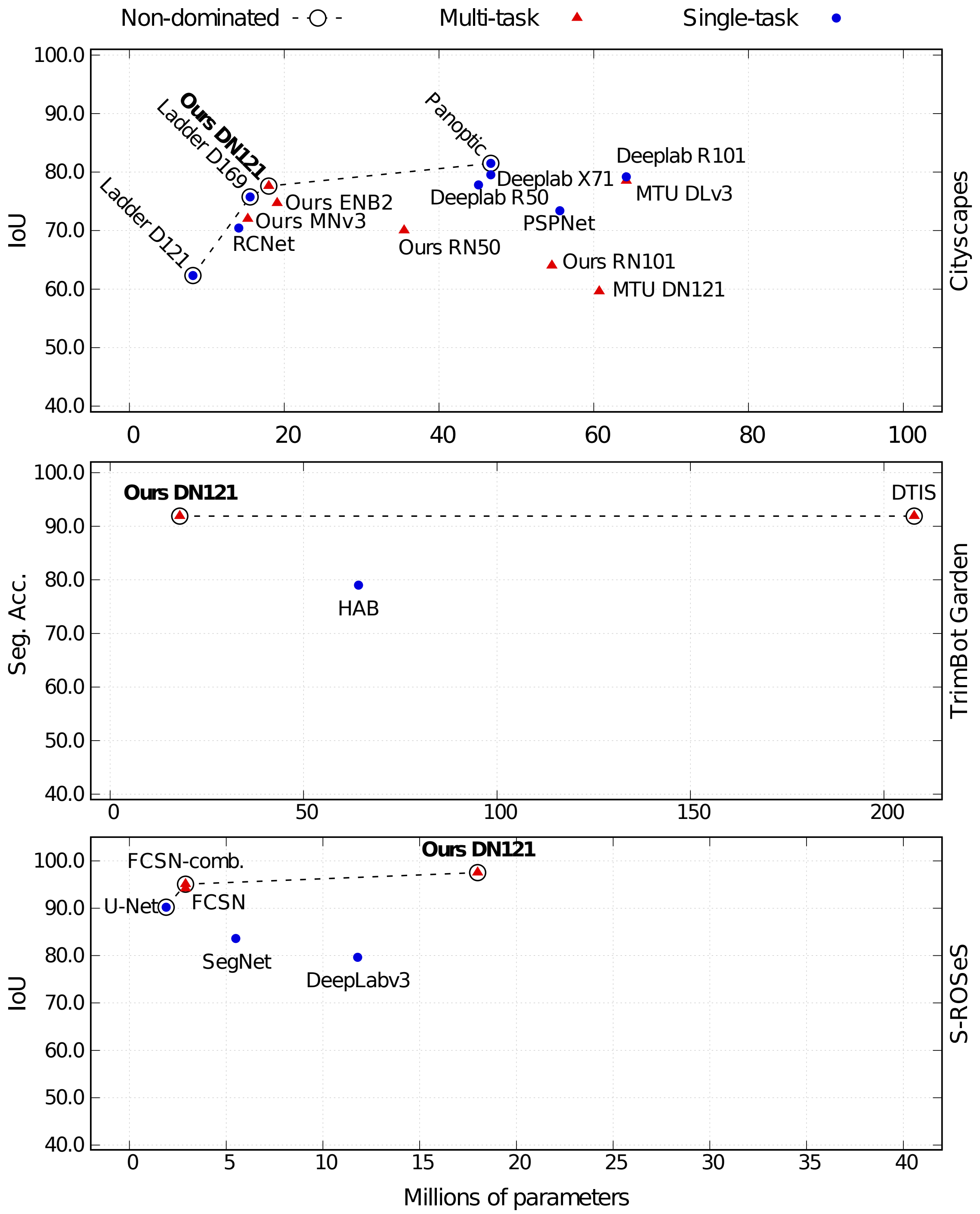


Figure 12: Analysis of efficiency (measured in the number of parameters) and effectiveness (considering IoU for the Cityscapes and S-ROSeS datasets and Pixel Accuracy for TrimBot2020 Garden) as a Multi-objective Optimization Problem (MOP). Non-dominated elements and multi-task approaches are highlighted.

6.3. Computational complexity

To assess the efficiency of the compared architectures, we have focused on the number of parameters, disregarding execution time due to its dependence on factors not related to the method itself, such as the programming language, code optimization, external libraries, or hardware configuration, among others. Therefore, we consider using the number of parameters more fair, as it is directly related to the time neural architectures will take to perform a forward pass.

However, in this section, we analyze the execution time to give an intuitive idea of the efficiency of the proposal according to the available resources. For this, an experiment was carried out on three different machines: one with very high resources (equipped with a DGX A100 GPU with 40Gb of RAM), one more modest (RTX 4090 with 24Gb of RAM), and one with low resources (GTX 1650 Super with 4Gb of RAM). Table 9 shows the average training times per epoch and inference times. This was calculated using the S-ROSeS dataset, averaging over 100 training epochs and 100 forward passes for inference. These times were obtained for a non-optimized implementation in Python, using the PyTorch library. Nonetheless, acceptable times of about 2 FPS are observed for the low-resource architecture, which could be substantially improved by optimizing the code.

Comparing these results with some state-of-the-art methods, we can confirm the argument regarding the efficiency relationship with the number of parameters. For example, the compared architecture with the most similar number of parameters, Ladder-169, which has 15.6M (vs. 18M in our case), has an average inference time of 0.56 sec. in the low-resource scenario. Heavier architectures like DTIS or FCSN-Comb. report much slower inference times, 4.23 and 5.67 sec., respectively. Meanwhile, more efficient architectures, such as SegNet or U-Net, manage to execute in average times of 0.1 sec. However, in return, they achieve worse results and only predict segmentation. Thus, it should be added the execution time for disparity calculation, a more complex and demanding task than segmentation.

6.4. Limitations

Having discussed the strengths of the proposal, this final section addresses its limitations, which are mainly related to the training data, its type, and the types of errors, as detailed below.

Although the method is proposed as efficient in terms of parameters and computational resources, it still requires a significant volume of annotated data with information on both disparity and semantic segmentation for training, which can be challenging in domains where

Table 9: Comparison of execution time (in seconds) for the proposed model across three computer architectures with varying resource levels, presenting the mean time plus standard deviation for both training over 100 epochs and inference on 100 images.

GPU Model	Epoch Training Time (s)	Inference Time (s)
DGX A100	133.74 ± 13.81	0.15 ± 0.01
RTX 4090	386.32 ± 48.73	0.49 ± 0.06
GTX 1650 S	601.46 ± 37.27	0.60 ± 0.02

data annotation is costly or difficult to obtain. As a solution to this limitation, future work could explore the integration of Self-Supervised Learning [66] to improve weight initialization and reduce the amount of annotated data required. Alternatively, the integration of Domain Adaptation techniques [67] could be considered to adapt models learned in one domain to new application domains. This latter technique would also allow the method to generalize to new domains or types of scenes, which is especially relevant in robotics and autonomous driving applications where environmental conditions can vary significantly.

Another potential limitation of the proposal is its dependency on a very specialized type of data for disparity estimation, such as stereoscopic images. This dependency could limit its applicability in situations where only monocular images are available or where capturing stereoscopic images is not feasible. In these cases, adapting the architecture to implement depth estimation techniques from a single image, which can leverage monocular depth cues such as the relative size of objects, perspective, and shadows, to infer depth without the need for a stereo pair, could be studied [15]. In this case, most of the architecture already operates with only the left image, and it would only be necessary to integrate these types of techniques replacing the depth calculation when using the right image.

On the other hand, the method benefits from the use of attention mechanisms and the progressive sharing of task-specific features across its branches. However, this strategy might not be optimal for all object classes or objects at different scales, which could affect the precision of the segmentation and disparity estimation in complex scenes. This limitation has been observed in the qualitative analysis conducted for coarse segmentation in Figure 11, where the main errors appear at the edges of thin elements. However, these errors seem to be resolved when refining the segmentation by leveraging shared information from the different branches and the details of higher-level features. Further research is intended on how to improve this limitation by including other types of techniques, such as a loss function that penalizes errors at the edges, the use of specific data augmentation techniques that generate variations at the edges of objects, or the inclusion of other techniques in the architecture like *Atrous* or dilated convolutions [26].

7. Conclusions

This work presents an efficient end-to-end method based on multi-task learning that successfully learns the semantic segmentation and disparity map together from an input stereo pair. To increase efficiency without sacrificing accuracy, a combination of a series of techniques is proposed, such as the use of Siamese architecture, decoders that perform progressive multi-scale learning and share task-specific features, the inclusion of attention layers to focus on the relevant parts, or Spatial Pyramid Pooling to capture correlations between distant features, among others.

The proposal is evaluated with 3 datasets—Cityscapes, TrimBot2020 Garden, and S-ROSeS—and compared with 21 state-of-the-art methods, outperforming the multi-task learning solutions and obtaining competitive results against the methods that solve each task individually but using only 1/3 of their parameters.

It is experimentally demonstrated that using separate decoders for each task is not sufficient to achieve good performance, and that sharing the knowledge of segmentation and disparity tasks between them, in a progressive fashion, improves the results of both tasks. Since the features of each task are obtained at different resolutions, these task-specific features also provide multi-scale information for coarse-to-fine learning. In addition, sharing the learned features also helps to reduce the complexity of the network and thus to increase its performance. This reduced number of parameters may allow its application on systems that have limited resources, such as low-budget autonomous driving systems or the development of affordable robots for garden care.

As future work, it is intended to extend the proposal to process other types of data, such as point clouds, to directly calculate their segmentation and depth. We also intend to further improve the efficiency of the method by studying more advanced network architectures, using more efficient backbones, and also by optimizing the implementation. Another line of work is exploring proposals to reduce the amount of training data needed, including techniques such as self-supervised learning, few-shot learning, or domain adaptation.

Acknowledgments

This work was supported by the I+D+i project TED2021-132103A-I00 (DOREMI), funded by MCIN/AEI/10.13039/501100011033.

References

[1] E. K. Stathopoulou, F. Remondino, A survey on conventional and learning-based methods for multi-view stereo, The Photogrammetric Record 38 (183) (2023) 374–407. doi:https://doi.org/10.1111/phor.12456.

[2] S. Hao, Y. Zhou, Y. Guo, A brief survey on semantic segmentation with deep learning, Neurocomputing 406 (2020) 302–321. doi:https://doi.org/10.1016/j.neucom.2019.11.118.

[3] J. Jiang, J. Liu, J. Fu, W. Wang, H. Lu, Super-resolution semantic segmentation with relation calibrating network, Pattern Recognition 124 (2022) 108501. doi:https://doi.org/10.1016/j.patcog.2021.108501.

[4] C. Wang, X. Wang, J. Zhang, L. Zhang, X. Bai, X. Ning, J. Zhou, E. Hancock, Uncertainty estimation for stereo matching based on evidential deep learning, Pattern Recognition 124 (2022) 108498. doi:https://doi.org/10.1016/j.patcog.2021.108498.

[5] N. Strisciuglio, R. Tylecek, M. Blaich, N. Petkov, P. Biber, J. Hemming, E. van Henten, T. Sattler, M. Pollefeys, T. Gevers, et al., Trimbot2020: an outdoor robot for automatic gardening, in: ISR 2018; 50th International Symposium on Robotics, VDE, 2018, pp. 1–6. doi:10.3030/688007.

[6] Y. Nie, S. Guo, J. Chang, X. Han, J. Huang, S.-M. Hu, J. J. Zhang, Shallow2deep: Indoor scene modeling by single image understanding, Pattern Recognition 103 (2020) 107271. doi:https://doi.org/10.1016/j.patcog.2020.107271.

[7] Z. Shen, Y. Dai, Z. Rao, MSMD-Net: Deep Stereo Matching with Multi-scale and Multi-dimension Cost Volume, arXiv preprint arXiv:2006.12797 (2020). doi:10.48550/arXiv.2006.12797.

[8] A. Tao, K. Sapra, B. Catanzaro, Hierarchical multi-scale attention for semantic segmentation, arXiv preprint arXiv:2005.10821 (2020).

[9] H. Cuevas-Velasquez, A.-J. Gallego, R. B. Fisher, Segmentation and 3d reconstruction of rose plants from stereoscopic images, Computers and Electronics in Agriculture 171 (2020) 105296. doi:https://doi.org/10.1016/j.compag.2020.105296.

[10] H. Cuevas-Velasquez, A.-J. Gallego, R. Tylecek, J. Hemming, B. van Tuijl, A. Mencarelli, R. B. Fisher, Real-time stereo visual servoing for rose pruning with robotic arm, in: 2020 IEEE International Conference on Robotics and Automation (ICRA), 2020, pp. 7050–7056. doi:10.1109/ICRA40945.2020.9197272.

[11] Y. Zhang, Q. Yang, A survey on multi-task learning, IEEE Transactions on Knowledge and Data Engineering (2021) 1–1doi:10.1109/TKDE.2021.3070203.

[12] G. Yang, H. Zhao, J. Shi, Z. Deng, J. Jia, Segstereo: Exploiting semantic information for disparity estimation, in: Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 636–651. doi:https://doi.org/10.1007/978-3-030-01234-2_39.

[13] A. Kendall, Y. Gal, R. Cipolla, Multi-task learning using uncertainty to weigh losses for scene geometry and semantics, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 7482–7491. doi:https://doi.org/10.1109/cvpr.2018.00781.

[14] Z. Zhang, Z. Cui, C. Xu, Z. Jie, X. Li, J. Yang, Joint task-recursive learning for semantic segmentation and depth estimation, in: Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 235–251. doi:10.1007/978-3-030-01249-6_15.

[15] V. Nekrasov, T. Dharmasiri, A. Spek, T. Drummond, C. Shen, I. Reid, Real-time joint semantic segmentation and depth estimation using asymmetric annotations, in: 2019 International Conference on Robotics and Automation (ICRA), IEEE, 2019, pp. 7101–7107. doi:https://doi.org/10.1109/icra.2019.8794220.

[16] I. Melekhov, J. Kannala, E. Rahtu, Siamese network features for image matching, in: 2016 23rd International Conference on Pattern Recognition (ICPR), 2016, pp. 378–383. doi:10.1109/ICPR.2016.7899663.

[17] A. Newell, K. Yang, J. Deng, Stacked hourglass networks for human pose estimation, in: B. Leibe, J. Matas, N. Sebe, M. Welling (Eds.), Computer Vision – ECCV 2016, Springer International Publishing, Cham, 2016, pp. 483–499.

[18] K. He, X. Zhang, S. Ren, J. Sun, Spatial pyramid pooling in deep convolutional networks for visual recognition, IEEE transactions on pattern analysis and machine intelligence 37 (9) (2015) 1904–1916.

[19] H. Zheng, J. Fu, T. Mei, J. Luo, Learning multi-attention convolutional neural network for fine-grained image recognition, in: Proceedings of the IEEE international conference on computer vision, 2017, pp. 5209–5217. doi:https://doi.org/10.1109/iccv.2017.557.

[20] N. Mayer, E. Ilg, P. Hausser, P. Fischer, D. Cremers, A. Dosovitskiy, T. Brox, A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 4040–4048. doi:https://doi.org/10.1109/cvpr.2016.438.

[21] A. Kendall, H. Martirosyan, S. Dasgupta, P. Henry, R. Kennedy, A. Bachrach, A. Bry, End-to-end learning of geometry and context for deep stereo regression, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 66–75. doi:https://doi.org/10.1109/iccv.2017.17.

[22] J.-R. Chang, Y.-S. Chen, Pyramid stereo matching network, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 5410–5418. doi:https://doi.org/10.1109/cvpr.2018.00567.

[23] K. Zhou, X. Meng, B. Cheng, Review of stereo matching algorithms based on deep learning, Computational Intelligence and Neuroscience 2020 (2020) 12. doi:https://doi.org/10.1155/2020/8562323.

[24] X. Yuan, J. Shi, L. Gu, A review of deep learning methods for semantic segmentation of remote sensing imagery, Expert Systems with Applications 169 (2021) 114417. doi:https://doi.org/10.1016/j.eswa.2020.114417.

[25] J. Long, E. Shelhamer, T. Darrell, Fully convolutional networks for semantic segmentation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 3431–3440. doi:https://doi.org/10.1109/cvpr.2015.7298965.

[26] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, A. L. Yuille, Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully con-

nected crfs, IEEE transactions on pattern analysis and machine intelligence 40 (4) (2017) 834–848. doi:https://doi.org/10.1109/tpami.2017.2699184.

[27] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, H. Adam, Encoder-decoder with atrous separable convolution for semantic image segmentation, in: Proceedings of the European conference on computer vision (ECCV), 2018, pp. 801–818. doi:https://doi.org/10.1007/978-3-030-01234-2_49.

[28] L.-C. Chen, G. Papandreou, F. Schroff, H. Adam, Rethinking atrous convolution for semantic image segmentation, CoRR abs/1706.05587 (2017).
URL http://arxiv.org/abs/1706.05587

[29] L.-C. Chen, Y. Yang, J. Wang, W. Xu, A. L. Yuille, Attention to scale: Scale-aware semantic image segmentation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 3640–3649. doi:https://doi.org/10.1109/cvpr.2016.396.

[30] B. Cheng, M. D. Collins, Y. Zhu, T. Liu, T. S. Huang, H. Adam, L.-C. Chen, Panoptic-deeplab: A simple, strong, and fast baseline for bottom-up panoptic segmentation, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020, pp. 12475–12485. doi:https://doi.org/10.1109/cvpr42600.2020.01249.

[31] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, B. Ommer, High-resolution image synthesis with latent diffusion models, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2022, pp. 10684–10695. doi:https://doi.org/10.1109/cvpr52688.2022.01042.

[32] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, et al., Llama: Open and efficient foundation language models, arXiv preprint arXiv:2302.13971 (2023).
URL https://arxiv.org/abs/2302.13971

[33] Y. Yuan, X. Chen, J. Wang, Object-contextual representations for semantic segmentation, in: A. Vedaldi, H. Bischof, T. Brox, J.-M. Frahm (Eds.), Computer Vision – ECCV 2020, Springer International Publishing, Cham, 2020, pp. 173–190.

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

[35] M. Menze, A. Geiger, Object scene flow for autonomous vehicles, in: Conference on Computer Vision and Pattern Recognition (CVPR), 2015.

[36] R. Tylecek, T. Sattler, H.-A. Le, T. Brox, M. Pollefeys, R. B. Fisher, T. Gevers, The second workshop on 3d reconstruction meets semantics: Challenge results discussion, in: L. Leal-Taixé, S. Roth (Eds.), ECCV 2018 Workshops, Springer International Publishing, Cham, 2019, pp. 631–644.

[37] D. Xu, W. Ouyang, X. Wang, N. Sebe, Pad-net: Multi-tasks guided prediction-and-distillation network for simultaneous depth estimation and scene parsing, in: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, pp. 675–684. doi:10.1109/CVPR.2018.00077.

[38] R. Hartley, A. Zisserman, Multiple view geometry in computer vision, Cambridge university press, 2003.

[39] J. Baxter, A model of inductive bias learning, J. Artif. Int. Res. 12 (1) (2000) 149–198.

[40] G. Lin, A. Milan, C. Shen, I. Reid, Refinenet: Multi-path refinement networks for high-resolution semantic segmentation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 1925–1934.

[41] G. Huang, Z. Liu, L. Van Der Maaten, K. Q. Weinberger, Densely connected convolutional networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 4700–4708.

[42] X. Glorot, A. Bordes, Y. Bengio, Deep Sparse Rectifier Neural Networks, Journal of Machine Learning Research (JMLR) W&CP 15 (2011) 315–323.

[43] S. Ioffe, C. Szegedy, Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, JMLR W&CP 37 (2015).

[44] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, S. Belongie, Feature pyramid networks for object detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 2117–2125.

[45] M. Tan, Q. Le, EfficientNet: Rethinking model scaling for convolutional neural networks, in: K. Chaudhuri, R. Salakhutdinov (Eds.), Proceedings of the 36th International Conference on Machine Learning, Vol. 97 of Proceedings of Machine Learning Research, PMLR, 2019, pp. 6105–6114.

[46] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, Q. V. Le, H. Adam, Searching for mobilenetv3 (2019). arXiv:1905.02244.

[47] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.

[48] G. Koch, R. Zemel, R. Salakhutdinov, Siamese neural networks for one-shot image recognition, in: 32nd International Conference on Machine Learning (ICML), Vol. 37, Lille, France, 2015, p. 0.

[49] H. Zhao, J. Shi, X. Qi, X. Wang, J. Jia, Pyramid scene parsing network, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 2881–2890.

[50] R. Atienza, Fast disparity estimation using dense networks, in: 2018 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2018, pp. 3207–3212.

[51] M. Berman, A. Rannen Triki, M. B. Blaschko, The lovász-softmax loss: A tractable surrogate for the optimization of the intersection-over-union measure in neural networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 4413–4421.

[52] L. Bottou, Large-scale machine learning with stochastic gradient descent, in: Proceedings of COMPSTAT'2010, Springer, 2010, pp. 177–186.

[53] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, in: F. Pereira, C. J. C. Burges, L. Bottou, K. Q. Weinberger (Eds.), Advances in Neural Information Processing Systems, Vol. 25, Curran Associates, Inc., 2012.

[54] J. Yosinski, J. Clune, Y. Bengio, H. Lipson, How transferable are features in deep neural networks?, in: Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, K. Q.

Weinberger (Eds.), Advances in Neural Information Processing Systems (NIPS), 2014, pp. 3320–3328.

[55] D. Kingma, J. Ba, Adam: A method for stochastic optimization, International Conference on Learning Representations (ICLR) (12 2015).

[56] K. Chatfield, K. Simonyan, A. Vedaldi, A. Zisserman, Return of the Devil in the Details: Delving Deep into Convolutional Nets, in: British Machine Vision Conference, 2014, pp. 1–11. doi:10.5244/C.28.6.

[57] S. Choi, J. T. Kim, J. Choo, Cars can't fly up in the sky: Improving urban-scene segmentation via height-driven attention networks, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020, pp. 9373–9383.

[58] I. Kreso, S. Segvic, J. Krapac, Ladder-style densenets for semantic segmentation of large natural images, in: Proceedings of the IEEE International Conference on Computer Vision Workshops, 2017, pp. 238–245.

[59] M. Carvalho, M. Ferrera, A. Boulch, J. Moras, B. L. Saux, P. Trouvé-Peloux, Technical Report: Co-learning of geometry and semantics for online 3D mapping, arXiv preprint arXiv:1911.01082 (2019).

[60] G. Ilha, T. Waszak, F. Pereira, A. Susin, Lapsi-360, in: 3DRMS Workshop Challenge, ECCV, 2018.

[61] C. Hazirbas, L. Ma, C. Domokos, D. Cremers, Fusenet: Incorporating depth into semantic segmentation via fusion-based cnn architecture, in: Asian conference on computer vision, Springer, 2016, pp. 213–228.

[62] A. Geiger, M. Roser, R. Urtasun, Efficient large-scale stereo matching, in: Asian conference on computer vision, Springer, 2010, pp. 25–38.

[63] O. Ronneberger, P. Fischer, T. Brox, U-net: Convolutional networks for biomedical image segmentation, in: Medical Image Computing and Computer-Assisted Intervention (MICCAI), 2015.

[64] V. Badrinarayanan, A. Kendall, R. Cipolla, Segnet: A deep convolutional encoder-decoder architecture for image segmentation, IEEE Transactions on Pattern Analysis and Machine Intelligence 39 (12) (2017) 2481–2495. doi:10.1109/TPAMI.2016.2644615.

[65] K. Konolige, Small vision systems: Hardware and implementation, in: Y. Shirai, S. Hirose (Eds.), Robotics Research, Springer London, London, 1998, pp. 203–212.

[66] X. Liu, F. Zhang, Z. Hou, L. Mian, Z. Wang, J. Zhang, J. Tang, Self-supervised learning: Generative or contrastive, IEEE Transactions on Knowledge and Data Engineering 35 (1) (2023) 857–876. doi:10.1109/TKDE.2021.3090866.

[67] A.-J. Gallego, J. Calvo-Zaragoza, R. B. Fisher, Incremental unsupervised domain-adversarial training of neural networks, IEEE Transactions on Neural Networks and Learning Systems 32 (11) (2021) 4864–4878. doi:10.1109/TNNLS.2020.3025954.

Appendix A. Network details

This section complements the network details described in the main body of the paper. Figure A.13 shows a comprehensive overview of the proposed architecture with detailed information of each of its parts. Table A.10 shows the split of the backbone and the dimension of each output feature F_i. Tables A.11, A.12, and A.13 show the layer settings, inputs, and outputs of the coarse segmentation, disparity estimation, and refined segmentation branches, respectively. Each layer has a stride of 1 except for the Spatial Pyramid Pooling (SPP), where the stride is the same size as the pooling window.

Table A.10: Backbone settings. It shows the input image (Left and Right) for each backbone and the features that are extracted. The size of each feature is shown with respect to the input size (height H and width W). The last dimension of the output represents the number of channels.

Input	Layer settings	Output	
		Name	Dim
Left	Densenet121	F_1^L	$\frac{H}{4}$ x $\frac{W}{4}$ x 128
		F_2^L	$\frac{H}{8}$ x $\frac{W}{8}$ x 256
		F_4^L	$\frac{H}{32}$ x $\frac{W}{32}$ x 1024
Right	Densenet121	F_1^R	$\frac{H}{4}$ x $\frac{W}{4}$ x 128
		F_2^R	$\frac{H}{8}$ x $\frac{W}{8}$ x 256
		F_4^R	$\frac{H}{32}$ x $\frac{W}{32}$ x 1024

Table A.11: Coarse segmentation branch modules. For the convolutional layers conv_1d and conv_2d, the layer settings $k \times k, f$ represent the kernel size k and the number of filters f.

Input	Layer settings	Output
Left	conv2D$(5 \times 5, 1)$	L_conv_0
F_4^L F_4^R	concat	cat_0
cat_0	upscale ×2 conv1D$(1 \times 1, 64)$	conv1d_0
conv1d_0	hourglass$(3 \times 3, 32)$	F_{seg}
F_{seg}	resize to L_conv_0 dim	convdec_0
L_conv_0 convdec_0	concat	cat_1
cat_1	conv1D$(1 \times 1, 32)$	conv1d_1
conv1d_1	hourglass$(3 \times 3, 32)$	convdec_1
convdec_1	conv2D$(3 \times 3, n_labels)$	conv2d_0
conv2d_0	resize to Left dim	coarse_seg

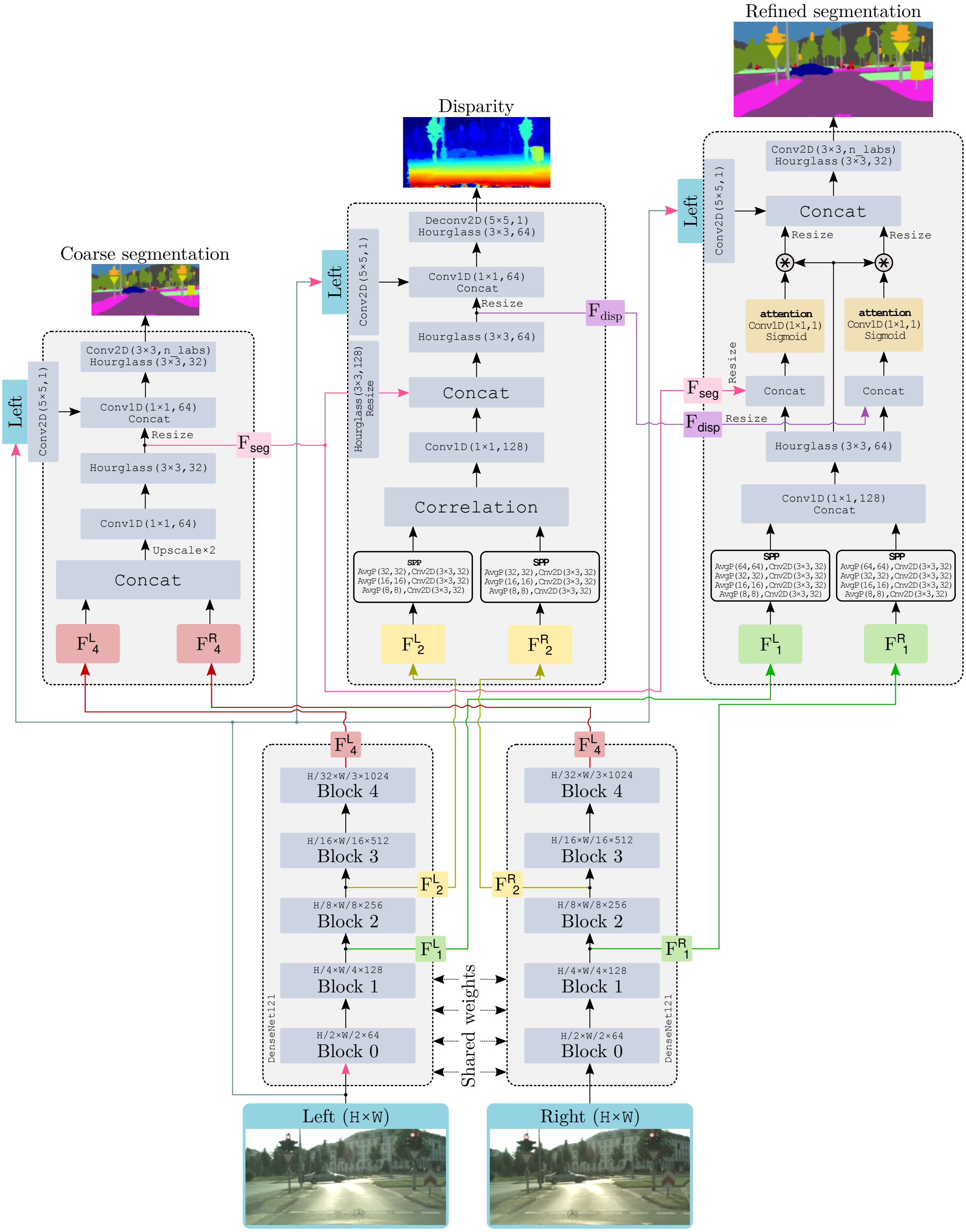


Figure A.13: Comprehensive visual overview of the proposed architecture including the detail of each of its parts. Refer to Tables A.10, A.11, A.12, and A.13 for specific information on the backbones and branches related to coarse segmentation, disparity, and refined segmentation, respectively.

Table A.12: Disparity branch modules. The brackets in the *SPP* module indicate that the input was processed in 4 different ways and the result of each process was concatenated.

Input	Layer settings	Output
Left	conv2D($5 \times 5, 1$)	L_conv_1
F_2^L	$\begin{bmatrix} \text{avgPool}(32, 32), \text{conv2D}(3 \times 3, 32) \\ \text{avgPool}(16, 16), \text{conv2D}(3 \times 3, 32) \\ \text{avgPool}(8, 8), \text{conv2D}(3 \times 3, 32) \\ F_2^L \end{bmatrix}$	SPP_2^L
F_2^R	$\begin{bmatrix} \text{avgPool}(32, 32), \text{conv2D}(3 \times 3, 32) \\ \text{avgPool}(16, 16), \text{conv2D}(3 \times 3, 32) \\ \text{avgPool}(8, 8), \text{conv2D}(3 \times 3, 32) \\ F_2^R \end{bmatrix}$	SPP_2^R
SPP_2^L SPP_2^R	correlation_layer	corr
corr	conv1D($1 \times 1, 128$)	conv1d_2
F_{seg}	hourglass($3 \times 3, 128$) resize to conv1d_2 dim	convdec_2
conv1d_2 convdec_2	concat	cat_3
cat_3	hourglass($3 \times 3, 64$)	F_{disp}
F_{disp}	resize to L_conv_1 dim	convdec_3
L_conv_1 convdec_3	concat	cat_4
cat_4	conv1D($1 \times 1, 64$)	conv1d_3
conv1d_3	hourglass($3 \times 3, 64$)	convdec_4
convdec_4	deconv2D($5 \times 5, 1$)	deconv2d_0
deconv2D_0	resize to Left dim	disp

Table A.13: Refined segmentation branch modules. The brackets in the SPP module follow the same notation as in Table A.12.

Input	Layer settings	Output
Left	conv2D$(5 \times 5, 1)$	L_conv_2
F_1^L	avgPool$(64, 64)$, conv2D$(3 \times 3, 32)$ avgPool$(32, 32)$, conv2D$(3 \times 3, 32)$ avgPool$(16, 16)$, conv2D$(3 \times 3, 32)$ avgPool$(8, 8)$, conv2D$(3 \times 3, 32)$ F_1^L	SPP_1^L
F_1^R	avgPool$(64, 64)$, conv2D$(3 \times 3, 32)$ avgPool$(32, 32)$, conv2D$(3 \times 3, 32)$ avgPool$(16, 16)$, conv2D$(3 \times 3, 32)$ avgPool$(8, 8)$, conv2D$(3 \times 3, 32)$ F_1^R	SPP_1^R
SPP_1^L SPP_1^R	concat	cat_5
cat_5	conv1D$(1 \times 1, 128)$	conv1d_4
conv1d_4	hourglass$(3 \times 3, 64)$	convdec_5
F_{seg}	resize to convdec_5 dim	F_{seg}
F_{disp}	resize to convdec_5 dim	F_{disp}
F_{seg} convdec_5	concat	cat_6
F_{disp} convdec_5	concat	cat_7
cat_6	conv1D$(1 \times 1, 1)$, sigmoid()	seg_att
cat_7	conv1D$(1 \times 1, 1)$, sigmoid()	disp_att
seg_att	seg_att∗convdec_5 resize to L_conv_2 dim	seg_att
disp_att	disp_att∗convdec_5 resize to L_conv_2 dim	disp_att
seg_att disp_att L_conv_2	concat	cat_8
cat_8	hourglass$(3 \times 3, 32)$	convdec_6
convdec_6	conv2D$(3 \times 3, n_labels)$	conv2d_1
conv2d_1	resize to L_conv_2 dim	ref_seg