Efficient RGB-D Semantic Segmentation for Indoor Scene Analysis

Daniel Seichter, Mona Köhler, Benjamin Lewandowski, Tim Wengefeld and Horst-Michael Gross

Abstract—Analyzing scenes thoroughly is crucial for mobile robots acting in different environments. Semantic segmentation can enhance various subsequent tasks, such as (semantically assisted) person perception, (semantic) free space detection, (semantic) mapping, and (semantic) navigation. In this paper, we propose an efficient and robust RGB-D segmentation approach that can be optimized to a high degree using NVIDIA TensorRT and, thus, is well suited as a common initial processing step in a complex system for scene analysis on mobile robots. We show that RGB-D segmentation is superior to processing RGB images solely and that it can still be performed in real time if the network architecture is carefully designed. We evaluate our proposed Efficient Scene Analysis Network (ESANet) on the common indoor datasets NYUv2 and SUNRGB-D and show that it reaches state-of-the-art performance when considering both segmentation performance and runtime. Furthermore, our evaluation on the outdoor dataset Cityscapes shows that our approach is suitable for other applications as well. Finally, instead of presenting benchmark results only, we show qualitative results in one of our indoor application scenarios.

I. INTRODUCTION

Semantic scene perception and understanding is essential for mobile robots acting in various environments. In our research projects, covering public environments from supermarkets [1], [2] to hospitals [3], [4] and domestic applications [5], [6], our robots need to perform several tasks in parallel such as obstacle avoidance, semantic mapping, navigation to semantic entities, and person perception. Most of the tasks require to be handled in real time given limited computing and battery capabilities. Hence, an efficient and shared initial processing step can facilitate subsequent tasks. Semantic segmentation is well suited for such an initial step, as it provides precise pixel-wise information that is well suited for numerous subsequent tasks.

In this paper, we propose an efficient and robust encoder-decoder-based semantic segmentation approach that can be embedded in complex systems for semantic scene analysis as shown in Fig. 1. The segmentation output enriches the robot’s visual perception and facilitates further processing steps by providing individual semantic masks. For our person perception [7], computations can be restricted to image regions segmented as person, instead of processing the entire image. Furthermore, the floor class indicates free space that can serve as a common preprocessing step for subsequent tasks such as (semantically aided) person perception, (semantic) free space detection, (semantic) mapping, and (semantic) navigation. Incorporating depth images can alleviate this effect by providing complementary geometric information, as shown in [9], [10], [11]. In contrast to processing RGB images solely, this design decision comes with some additional computational cost. However, in this paper, we show that two carefully designed shallow encoder branches (one for RGB and one for depth data) can achieve better segmentation performance while still enabling faster inference (application of network) than a single deep encoder branch for RGB images solely. Moreover, our Efficient Scene Analysis Network (ESANet) enables much faster inference than most other RGB-D segmentation methods, while performing on par or even better as shown by our experiments.

We evaluate our ESANet on the common indoor datasets NYUv2 [12] and SUNRGB-D [13] and further present qualitative results in our indoor application. Instead of only focusing on mean intersection over union (mIoU) as evaluation metric for benchmarking, we also strive for a fast inference on embedded hardware. However, rather than reporting the inference time on high-end GPUs, we measure inference time on our robot’s NVIDIA Jetson AGX Xavier. Furthermore, we designed our network architecture such that it can be executed as a single optimized graph using NVIDIA
TensorRT. Moreover, by evaluating on the outdoor dataset Cityscapes [14], we show that our approach can also be applied to other areas of application.

The main contributions of this paper are:

- an efficient RGB-D segmentation approach, which can serve as initial processing step to facilitate subsequent scene analysis tasks and is characterized by:
  - a carefully designed architecture that can be optimized to a high degree using NVIDIA TensorRT and, thus, enables fast inference
  - an efficient ResNet-based encoder that utilizes a modified basic block that is computationally less expensive while achieving higher accuracy
  - a decoder that utilizes a novel learned upsampling
- a detailed ablation study to the fundamental parts of our approach and their impact on segmentation performance and inference time
- qualitative results in a complex system for robotic scene analysis proving the applicability and robustness.

Our code as well as the trained networks are publicly available at: https://github.com/TUI-NICR/ESANet

II. RELATED WORK

Common network architectures for semantic segmentation follow an encoder-decoder design. The encoder extracts semantically rich features from the input and performs downsampling to reduce computational effort. The decoder restores the input resolution and, finally, assigns a semantic class to each input pixel.

A. RGB-D Semantic Segmentation

Depth images provide complementary geometric information to RGB images and, thus, improve segmentation [9], [10], [11]. However, incorporating depth information into RGB segmentation architectures is challenging as depth introduces deviating statistics and characteristics from another modality.

In [15], depth information is used to project the RGB images into a 3D space. However, processing the resulting 3D data leads to significantly higher computational complexity. [16], [17], [18], [19], [20] design specifically tailored convolutions, taking into account depth information. Nevertheless, these modified convolutions often lack optimized implementations and, thus, are slow and not applicable for real-time segmentation on embedded hardware.

The majority of approaches for RGB-D segmentation [9], [21], [10], [11], [22], [23], [24], [25] simply use two branches, one for RGB and one for depth data and fuse the feature representations later in the network. This way, each branch can focus on extracting modality-specific features, such as color and texture from RGB images and geometric, illumination-independent features from depth images. Fusing these modality-specific features leads to stronger feature representations. Instead of fusing only low-level or high-level features, [9] shows that the segmentation performance increases if the features are fused at multiple stages. Typically, the features are fused once at each resolution stage with the last fusion at the end of both encoders. Using only one decoder for the combined features reduces the computational effort. FuseNet [9] and RedNet [10] fuse the depth features into the RGB encoder, which follows the intuition that the semantically richer RGB features can be further enhanced using complementary depth information. SA-Gate [22] combines RGB and depth features and fuses the recalibrated features back into both encoders. In order to make the two encoders independent of each other, ACNet [11] uses an additional, virtual, third encoder that obtains modality-specific features from the two encoders and processes the combined features. Instead of fusing in the encoder, the modality-specific features can also be used to refine the features in the common decoder via skip connections as in RDFNet [23], SSMA [24] and MMAF-Net [25].

However, none of the aforementioned methods focus on efficient RGB-D segmentation for embedded hardware. Using deep encoders such as ResNets with 50, 101 or even 152 layers results in high inference times and, therefore, makes them inappropriate for deploying to mobile robots.

B. Efficient Semantic Segmentation

In contrast to RGB-D segmentation, recent RGB approaches [26], [27], [28], [29], [30], [31], [32] also address reducing computational complexity to enable real-time segmentation. Most efficient segmentation approaches propose specifically tailored network architectures, which reduce both the number of operations and parameters to enable faster inference while still retaining good segmentation performance. Approaches, such as ERFNet [26], LEDNet [27], or DABNet [28] introduce efficient encoder blocks by replacing expensive $3 \times 3$ convolutions with more light-weight variants such as factorized, grouped, or depth-wise separable convolutions. Nevertheless, although requiring more operations and memory, SwiftNet [30] and BiSeNet [31] are still faster than many other methods, while achieving higher segmentation performance by simply using a pretrained ResNet18 as encoder. This can be deduced to utilizing early and high downsampling in the encoder, a light-weight decoder, and using standard $3 \times 3$ convolutions, which are currently implemented more efficiently than grouped or depth-wise convolutions and have large representational power.

Following SwiftNet and BiSeNet, our approach also uses a ResNet-based encoder. However, in order to further reduce inference time, we exchange the basic block in all ResNet layers with a more efficient block, based on factorized convolutions.

III. EFFICIENT RGB-D SEGMENTATION

The architecture of our Efficient Scene Analysis Network (ESANet) for RGB-D semantic segmentation is depicted in Fig. 2 (top). It is inspired by the RGB segmentation approach SwiftNet [30], i.e. a shallow encoder with a pretrained ResNet18 backbone and large downsampling, a context module similar to the one in PSPNet [33], a shallow decoder with skip connections from the encoder, and a final upsampling by a factor of 4. However, SwiftNet does not
neatly fits into the RGB encoder at several stages using an attention mechanism. The decoder is comprised of multiple modules, each is upsampling the resulting feature maps by a factor of 2 and is refining the features using convolutions as well as by incorporating encoder features. Finally, the decoder maps the features to the classes and rescales the class mapping to the input resolution.

Our entire network features simple components implemented in PyTorch [54]. We do not use complex structures or specifically tailored operations as these are often incompatible for converting to ONNX [35] or NVIDIA TensorRT and, thus, result in slower inference time.

In the following, we explain each part of our network design in detail as well as its motivation. Fig. 2 (bottom) depicts the exact structure of our network modules.

A. Encoder

For both the RGB and the depth encoder, we use a ResNet [36] as backbone. For efficiency reasons, we do not replace strided convolutions by dilated convolutions as in PSPNet [33] or DeepLabv3 [87]. Thus, the resulting feature maps at the end of the encoder are 32 times smaller than the input image. For a trade-off between speed and accuracy, we use ResNet54 but also show results for ResNet18 and ResNet50. We replace the basic block in each layer of ResNet18 and ResNet34 with a spatially factorized version. More precisely, the $3 \times 3$ convolutions are replaced by a $3 \times 1$ and a $1 \times 3$ convolution with a ReLU in-between. The so-called Non-Bottleneck-1D-Block (NBt1D) is depicted in Fig. 2 (violet) and was initially proposed in ERFNet [26] for another network architecture. In our experiments, we show that this modified ResNet as backbone simultaneously reduces inference time and increases segmentation performance.

B. RGB-D Fusion

At each of the five resolution stages in the encoders (see Fig. 2), depth features are fused into the RGB encoder. The features from both modalities are first reweighted with a Squeeze and Excitation (SE) module [38] and then summed element-wisely, as shown in Fig. 2 (green). Using this channel attention mechanism, the model can learn which features of which modality to focus on and which to suppress, depending on the given input. In our experiments, we show that this fusion mechanism notably improves segmentation.

C. Context Module

As noted in [33], the actual receptive field of ResNet is much smaller than the theoretical one induced by spatial operations. Therefore, we additionally incorporate context information by aggregating features at different scales using several branches in a context module similar to the Pyramid Pooling Module in PSPNet [33] (see Fig. 2 (orange)). Since NVIDIA TensorRT only supports pooling with fixed sizes, we carefully designed the context module such that the pooling sizes are always a factor of the input resolution and no adaptive pooling is required. Note that, depending on the image resolution of the respective dataset, the number of factors and, thus, the branches $b$ and pooling sizes $p_b \times p_b$ differ. Our experiments show that this additional context module improves segmentation.
D. Decoder

As shown in Fig 2, our decoder is comprised of three decoder modules (depicted in red in Fig 2). Our decoder module extends the one of SwiftNet [30], which is comprised of a 3×3 convolution with a fixed number of 128 channels and a subsequent bilinear upsampling. However, our experiments show that for indoor RGB-D segmentation a more complex decoder is required. Therefore, we use 512 channels in the first decoder module and decrease the number of channels used in the decoder, and add them to the decoder feature maps. Incorporating these skip connections results in more detailed semantic segmentations.

Similar to SwiftNet, we only process feature maps in the decoder until they are four times smaller than the input images and use a 3×3 convolution to map the features to the classes of the respective dataset. Two final learned upsampling modules restore the resolution of the input image.

Instead of calculating the training loss only at the final output scale, we add supervision before each upsampling. At each scale, a 1×1 convolution computes a segmentation of a smaller scale, which is supervised by the down-scaled ground truth segmentation.

IV. EXPERIMENTS

We evaluated our approach on two commonly used RGB-D indoor datasets, namely SUNRGB-D [13] and NYUv2 [12] and present an ablation study to essential parts of our network. Furthermore, to demonstrate that our approach is suitable for other areas of application as well, we also show results on the Cityscapes [14] dataset, the most widely used outdoor dataset for semantic segmentation. Finally, instead of reporting benchmark results only, we present qualitative results when using our approach in an indoor application.

A. Implementation Details & Datasets

We trained our networks using PyTorch [34] for 500 epochs with batches of size 8. For optimization, we used both SGD with momentum of 0.9 and Adam [39] with learning rates of {0.00125, 0.0025, 0.005, 0.01, 0.02, 0.04} and {0.0001, 0.0004}, respectively, and a small weight decay of 0.0001. We adapted the learning rate using PyTorch’s one-cycle learning rate scheduler. To further increase the number of training samples, we augmented the images using random scaling, cropping and flipping. For RGB images, we also applied slight color jittering in HSV space.

The best models were chosen based on the mean intersection over union (mIoU). We used bilinear upsampling to rescale the resulting class mapping to the size of the ground truth segmentation before computing the argmax for the final segmentation mask.

NYUv2 & SUNRGB-D: NYUv2 contains 1,449 indoor RGB-D images, of which 795 are used for training and 654 for testing. We used the common 40-class label setting. SUNRGB-D has 37 classes and consists of 10,335 indoor RGB-D images, including all images of NYUv2. There are 5,285 training and 5,050 testing images. Our ablation study is based on NYUv2 as it is smaller and, thus, leads to faster trainings. However, according to [40], training on a subset is sufficient for a reliable model selection. For both datasets, we used a network input resolution of 640×480 and applied median frequency class balancing [41]. As the input to the context module has a resolution of 20×15 due to the downsampling of 32, we used b = 2 branches, one with global average pooling and one with a pooling size of 3×4.

Cityscapes: This dataset contains 5,000 images with fine-grained annotation for 19 classes. The images have a high resolution of 2048×1024. There are 2,975 images for training, 500 for validation and 1,525 for testing. Cityscapes also provides 20k coarsely annotated images, which we did not use for training. We computed corresponding depth images from the disparity images. Since we set the network input resolution to 1024×512, the input to our context module has a resolution of 32×16, which allows b = 4 branches in the context module, one with global average pooling and the others with pooling sizes of 16×8, 8×4 and 4×2.

For further details and other hyperparameters, we refer to our implementation.

B. Results on NYUv2 & SUNRGB-D

Fig. 5 compares our RGB-D approach on NYUv2 to single-modality baselines for RGB and depth (single encoder) as well as compares different encoder backbones. As expected, neither processing depth data nor RGB data alone reach the segmentation performance of our proposed RGB-D network. Remarkably, the shallow ResNet18-based RGB-D network performs better than the much deeper ResNet50-based RGB network while still being faster. Moreover, replacing the ResNet’s basic block with the Non-Bottleneck-1D (NBT1D) block can further improve both segmentation performance and inference time. Note that ResNet50 incorporates a bottleneck block, which cannot be replaced the
same way. Tab. I lists the results of our RGB-D approach for both indoor datasets. For the larger SUNRGB-D dataset, a similar trend can be observed.

Compared to the state of the art, our carefully designed smaller networks achieve similar results as the often much deeper networks. Besides focusing on segmentation performance alone, we also strive for low inference time on the embedded hardware of our robots. Therefore, we measured the inference time of our ESANet and - whenever possible - also of the other approaches. As shown in Tab. II (last column), our approach enables much faster inference while performing on par or even better. For application, we choose ESANet with ResNet34 backbone and Non-Bottleneck-1D (NBt1D) block as it offers the best trade-off between inference time and performance. The last row in Tab. I further indicates that additional pretraining on synthetic data, such as SceneNet [42], should be preferred to deeper backbones, especially if the target dataset is small.

C. Ablation Study on NYUv2

Fig. 3 shows the ablation study for fundamental parts of our network architecture and justifies our design choices. Furthermore, it indicates the impact of each part when it is necessary to adapt our selected network to deviating real-time requirements.

The purple line demonstrates that a shallow decoder similar to the one in SwiftNet [30] is not as good as more complex decoders. Therefore, we gradually increased the number of additional Non-Bottleneck-1D (NBt1D) blocks in the decoder module. Apparently, a fixed number of three blocks in each decoder module performs better than a different number or a reversed layout of the encoder’s design.

In cyan, different upsampling methods in the decoder are displayed. Although increasing inference time, the learned upsampling improves mIoU by 0.9. Moreover, as shown in Fig. 5 the obtained segmentation contains more fine-grained details compared to using bilinear interpolation. It further prevents gridding artifacts introduced by transposed convolutions as used in RedNet [10] or ACNet [11].

As shown in gray in Fig. 4, a context module, encoder-decoder skip connections, as well as reweighting modality-specific features with Squeeze-and-Excitation before fusion, independently improve segmentation performance. Incorporating all three network parts leads to the best result.

D. Results on Cityscapes

To demonstrate that our approach is applicable to other areas such as outdoor environments as well, in Tab. II we further present an evaluation on the Cityscapes dataset.

We first focus on the smaller resolution of 1024×512 as it is commonly used for efficient segmentation. Moreover, since most approaches rely on RGB as input solely, we start by comparing an RGB-only version of our approach. Efficient approaches with custom architectures such as ERFNet [26], LEDNet [27], and ESPNetv2 [32] are quite fast but also perform notably worse than our ESANet. Compared to the
In this paper, we have presented an efficient RGB-D segmentation approach, called ESANet, which is characterized by two ResNet-based encoders utilizing the Non-Bottleneck-1D block, an attention-based fusion for incorporating depth information, and a decoder utilizing a novel learned upsampling. On the indoor datasets NYUv2 and SUNRGB-D, our carefully designed ESANet performs on par or even better while enabling much faster inference compared to other state-of-the-art methods. Finally, we have also shown that our segmentation approach is well suited for embedding in a complex system for robotic scene analysis.

V. Conclusion

TABLE II: Mean intersection over union of our ESANet on Cityscapes for both common input resolutions compared to state-of-the-art methods. FPS is reported for NVIDIA Jetson AGX Xavier (Jetpack 4.4, TensorRT 7.1, Float16). Legend: \*: trained with additional coarse data. 

![Image](https://www.youtube.com/watch?v=ig6073MPH14)

Fig. 6: Application in our robotic scene analysis system.
