Supplementary Material
SIUNet: Sparsity Invariant U-Net for Edge-Aware Depth Completion

Avinash Nittur Ramesh Fabio Giovanneschi María A. González-Huici
Fraunhofer FHR
Wachtberg, Germany
avinash.ramesh@fhr.fraunhofer.de

1. Summary

We proposed a simple yet effective unguided depth completion approach that only takes sparse depth images as input and produces dense depth images, and is robust even towards extremely sparse inputs. During training, learning depth boundaries is first explicitly enforced through auxiliary learning on synthetic SYNTHIA dataset with dense depth and depth contour images (generated using our method) as supervision, followed by fine-tuning on real-world KITTI and NYUv2 datasets with only depth images as supervision.

Here we reiterate the key points of our approach in detail for more clarity:

- We have proposed an unguided depth completion approach although guided approaches provide superior performance. The primary reason is that, for guided approaches obtaining rectified and curated data of different sensors' measurements from a customized end-user system is a tedious and non-trivial task. Because measurement noise: outliers due to occlusion as a result of small displacement in viewpoint of sensors, motion artifacts in dynamic scenarios due to different sensor acquisition time, etc. have to be handled as a preprocessing step.
- Our approach relies solely only on LiDAR sensor measurements without the need for target domain priors (RGB or semantic images) during training. Hence it greatly simplifies the target domain sensor setup.
- We proposed a sparsity invariant U-Net architecture and show the robustness of our method for extremely sparse LiDAR measurements.
- To enforce edge-awareness we propose a novel technique of generating depth contour images and using them as auxiliary targets during training on source domain. Depth contour images facilitate learning structural information in lieu of semantic information (*e.g.* in RGB or semantic images), making the network edge-aware and generic towards unseen classes in the datasets, as shown by generalizing on indoor and outdoor datasets captured from different sensor setups, and containing unseen classes.
- The speciality is that there are no branch-outs in our network, resulting in end-to-end feature sharing. The advantage of this is that for the same number of network parameters our network shares all the parameters and utilizes them towards depth completion in contrast to the networks with branch-outs where only some parameters are utilized for the primary task of depth completion, whereas the other parameters are dedicated for obtaining auxiliary output which can be seen in the works [25, 34, 50].
- The reason for no branch-outs in our network is because both primary and auxiliary tasks, i.e, depth reconstruction and depth contour regression, in our case are in the same domain, i.e., depth.

2. Network architecture

SConv x-y layer takes depth feature maps and binary mask from previous layer and transforms them by performing a set of operations, *i.e.*, point-wise multiplication, convolution, max-pooling, normalization and non-linear activation (ReLU). The blocks indicated with borders are trainable and others are just mathematical operations. **Ones x** is a convolution layer where the convolution kernel contains ones of dimension (**x**, **x**). **Max Pooling x** performs max pooling operation on patches of dimension (**x**, **x**). The output of **Sconv x-y** consists of **y** depth features and one binary mask. Binary mask does not exist for the first layer, so it is created from the input sparse depth image.

3. Directory structure and execution details

This supplementary material is provided in a zip file. We have provided all the training scripts, validation, and data

```
SIUNet
├── code ..... contains training, validation, and data augmentation scripts
├── model ..... contains our model (L1) which achieved best performance
├── dataset ..... datasets have to be copied inside this folder as follows
│   ├── kitti
│   │   ├── train
│   │   └── val_selection_cropped ..... this is KITTI selected validation set
│   ├── synthia
│   │   └── train
│   └── nyu
│       ├── nyudepthv2 ..... this is the original NYU dataset with .h5 files
│       │   ├── train
│       │   └── val
│       └── converted ..... this contains .npy files converted from .h5 files
│           ├── train
│           └── test
└── Readme.pdf
```

Figure 1: Directory structure of supplementary materials for code execution and validation

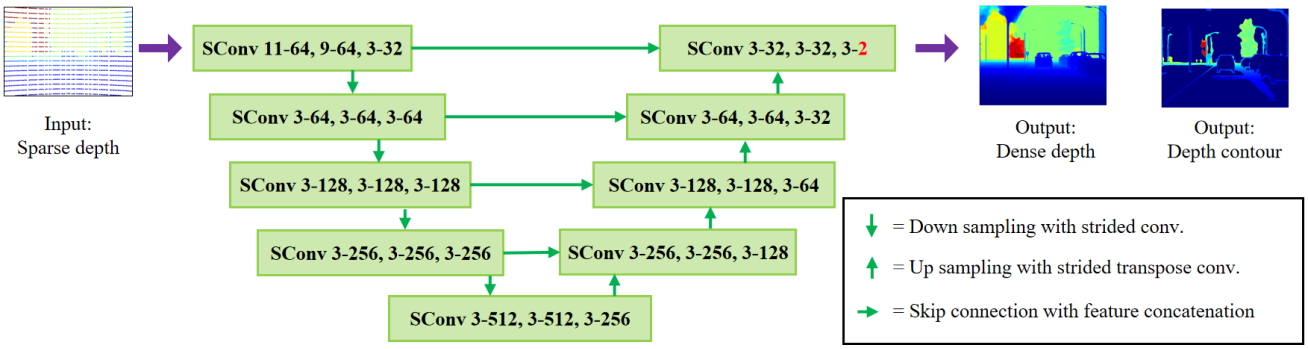


Figure 2: Detailed network architecture of SIUNet. **SConv x-y, ...** : indicates that there are **y** sparse 2D convolution kernels of size (**x**,**x**). A stack of such layers is indicated with ellipsis (**...**). Our model is a fully sparse convolutional network which consists of 27 sparse convolution layers. Our network uses only one input: sparse depth image, and produces two outputs: dense depth and depth contour images

augmentation script. We have provided place holders in the form of directory structure in the format shown in Fig. 1 for the datasets that we have used, and these datasets have to be downloaded for training or validating our scripts. We had employed **random seed generator** during our training process. To ensure reproducibility, the same random seed can be used which is also provided in scripts. All the scripts have been written in Python 3.8, and can be directly executed with *python* command without any command line arguments. Python requirement file has also been provided in the code folder, which can be used with *pip install -r requirements.txt* command to install all python dependencies. For NYUv2 dataset, .h5 files are first converted to .npy files and subsequently used for training and testing our model. The details of epoch, optimizer parameters, training losses, and evaluation metrics are provided in the scripts. For fine-tuning on KITTI dataset, only the last 3 layers were trained, and for NYUv2 dataset, the last 6 layers were trained.

4. Ablation study

Table 1 provides detailed description of different training methods, targets, and loss functions that were used for ablation study. The trained models are large in size so we have provided only one of them in the supplementary materials. We have provided the model which achieves state-of-the-art performance among unguided methods on KITTI depth completion benchmark. With this model, test 7 of ablation study can be carried out on KITTI validation selection dataset. We have provided the scripts for all the test cases of ablation study.

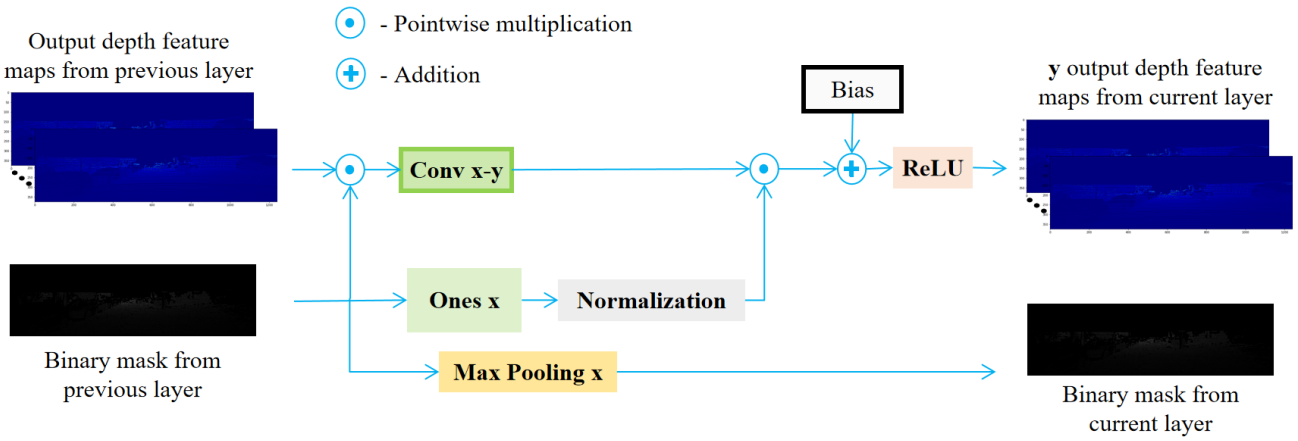


Figure 3: Detailed illustration of **SConv x-y** layer

Table 1: Illustrates detailed description of different training targets, and training methods used for the ablation study. After the training procedure, all the trained models were evaluated on KITTI depth completion validation selected set, with sparse depth as the only input and semi-dense depth image as the target

Test	Training method	Dataset	Details	Training loss
1	Conventional	KITTI	Input: Sparse depth Target: Sparse depth	MAE
2	Transfer learning (TL)	Step 1: SYNTHIA	Input: Sparsified depth Target: Dense depth	MAE
		Step 2: KITTI	Input: Sparse depth Target: Sparse depth	
3	Auxiliary learning (AL) + Zero-shot learning (ZSL)	SYNTHIA	Input: Sparsified depth Target: Dense depth, Depth contour	MAE
4	Conventional	KITTI	Input: Sparse depth Target: Semi-dense depth	MAE
5	Transfer learning (TL)	Step 1: SYNTHIA	Input: Sparsified depth Target: Dense depth	MAE
		Step 2: KITTI	Input: Sparse depth Target: Semi-dense depth	
6	Auxiliary learning (AL) + Transfer learning (TL): **Ours L2**	Step 1: SYNTHIA	Input: Sparsified depth Target: Dense depth, Depth contour	MAE
		Step 2: KITTI	Input: Sparse depth Target: Semi-dense depth	RMSE
7	Auxiliary learning (AL) + Transfer learning (TL): **Ours L1**	Step 1: SYNTHIA	Input: Sparsified depth Target: Dense depth, Depth contour	MAE
		Step 2: KITTI	Input: Sparse depth Target: Semi-dense depth	MAE

5. Sparsity invariance and generalization

Figures 4 to 7 show illustration of sparsity invariance of SIUNet evaluated on SYNTHIA dataset, KITTI and NYUv2 dataset. For the sake of uniformity in defining the induced sparsity levels, we have taken the ratio of pixels in a raw Velodyne input depth image of KITTI and pixels in its corresponding output dense depth image as a reference. It can be seen that in-spite of training SIUNet on outdoor SYNTHIA dataset (source domain) in the auxiliary step, it generalizes well for indoor NYUv2 dataset (target domain).

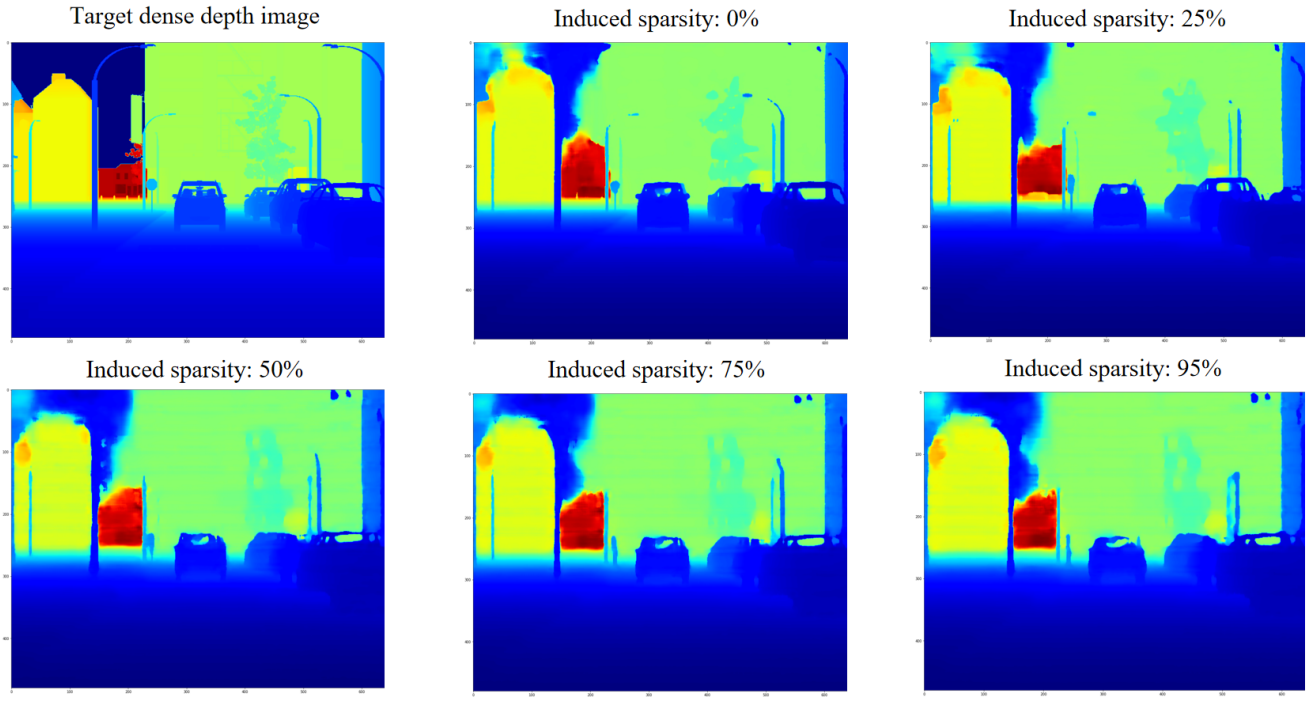


Figure 4: Sample illustration of dense depth outputs for varying sparsity levels of input image (SYNTHIA dataset)

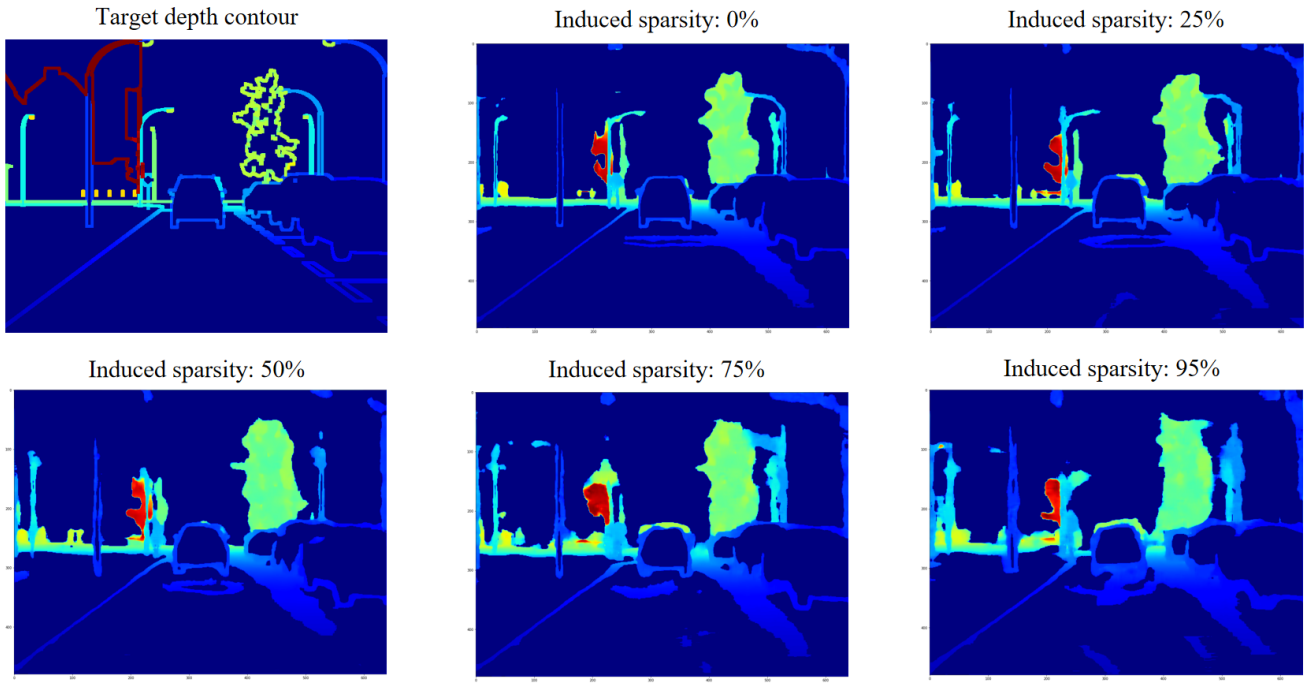


Figure 5: Sample illustration of depth contour outputs for varying sparsity levels of input image (SYNTHIA dataset)

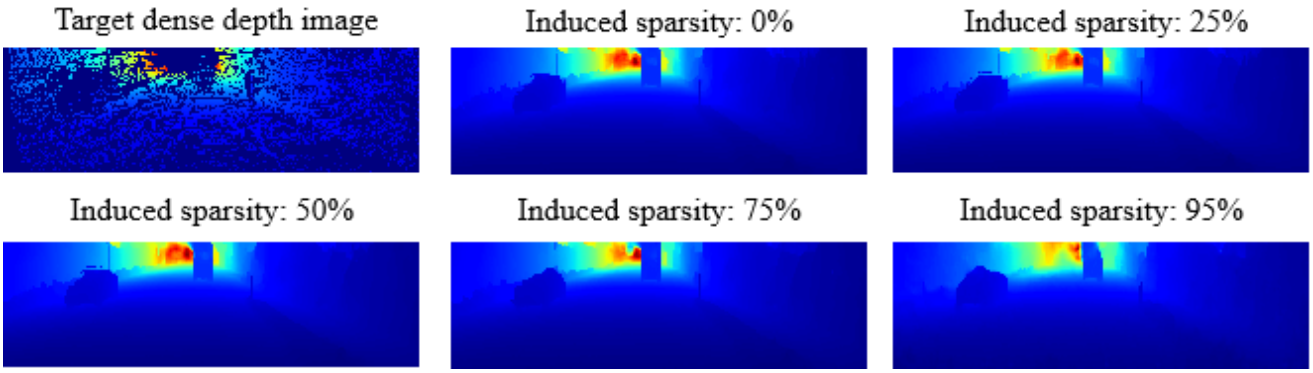


Figure 6: Sample illustration of dense depth outputs for varying sparsity levels of input image (KITTI dataset)

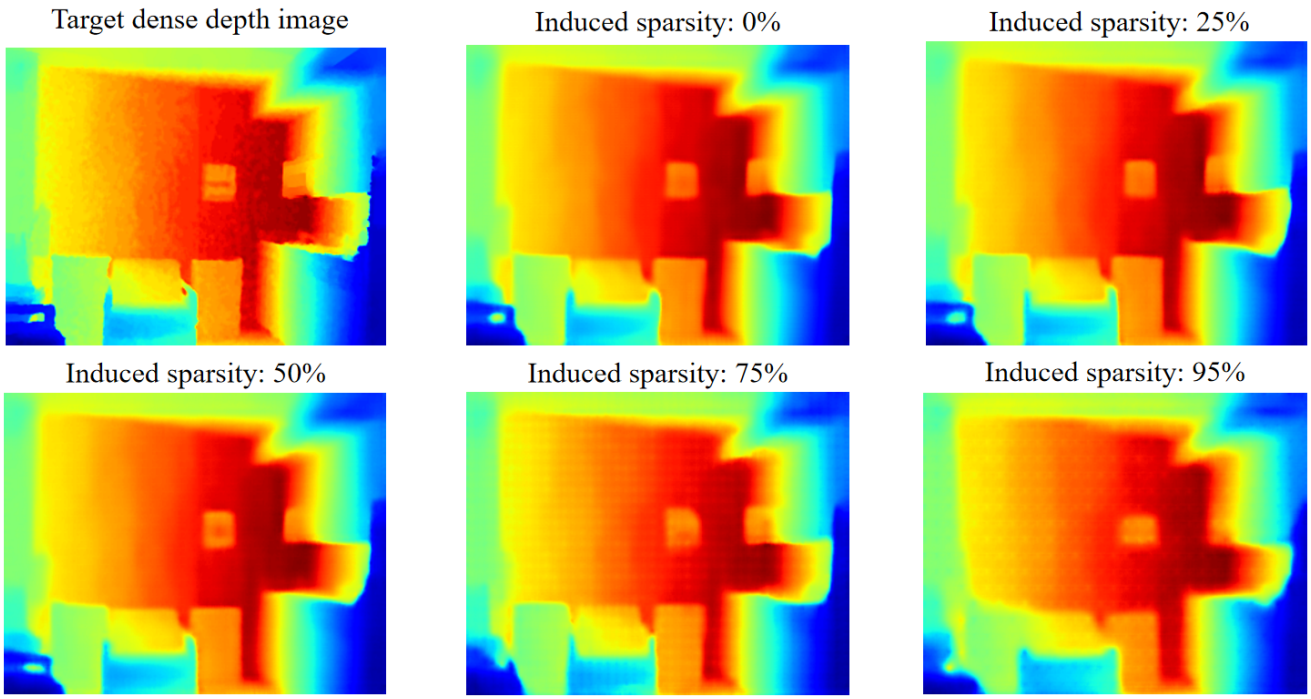
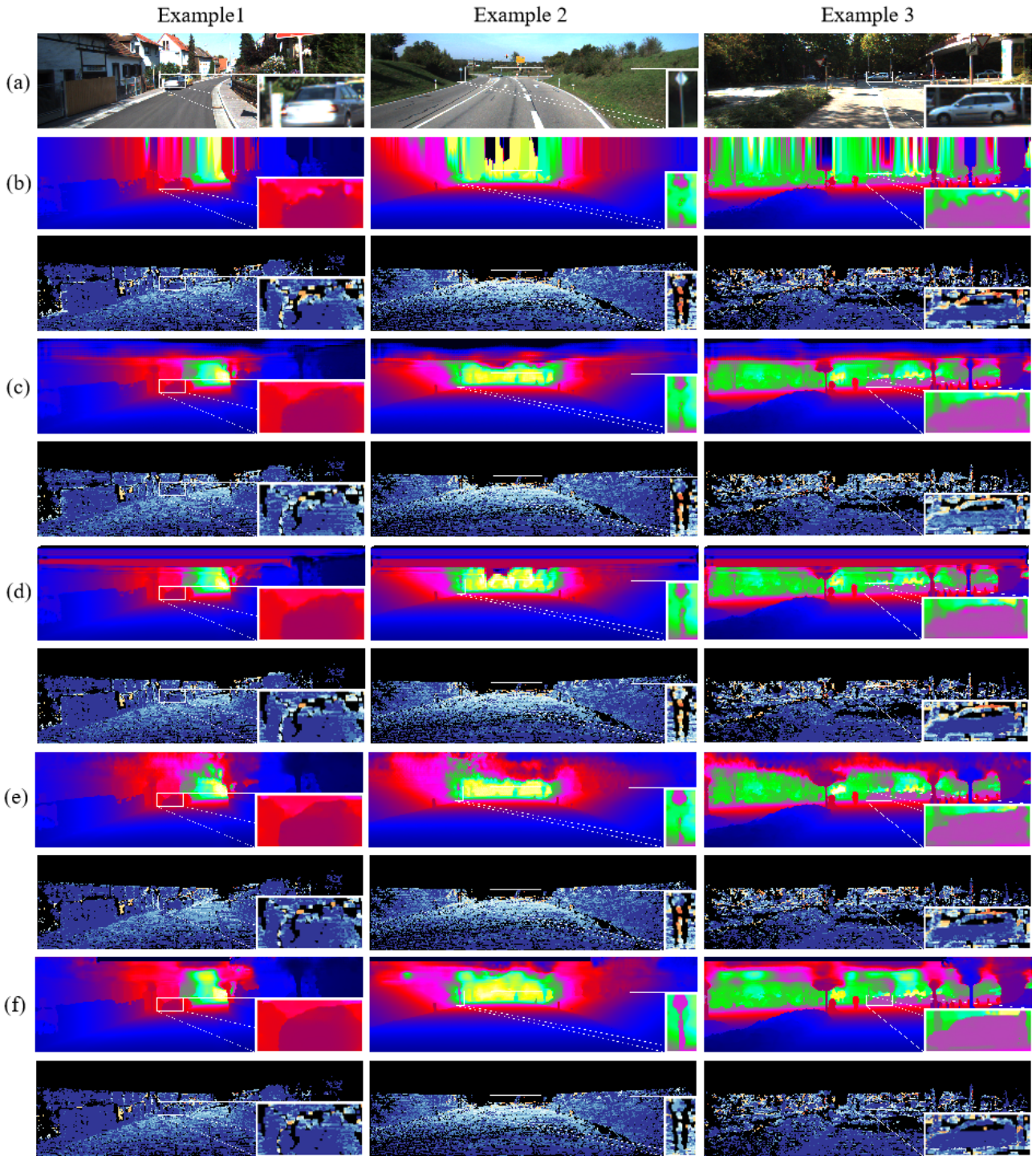


Figure 7: Sample illustration of dense depth outputs for varying sparsity levels of input image (NYUv2 dataset)



(a) RGB images for reference, (b) PSM [54], (c) StoD(d) [26], (d) pNCNN [8], (e) Spade-sD [28], (f) SIUNet (Ours)

Figure 8: Detailed evaluation on the boundary areas: Qualitative comparison of our approach with SoTA unguided depth completion approaches sorted in decreasing order of MAE from top to bottom. The closeup views of our method show sharpness along object boundaries and structural correctness. Depth bleeding can be observed in the reconstructions of other methods. Our network produces small errors along boundaries compared to SoTA unguided method (e). Small errors are displayed in blue and large errors in red. Black regions indicate missing ground truth.