TrimBot2020 Deliverable D4.4

Missing Data Completion

Principal Author: University of Amsterdam (UvA)
Contributors:
Dissemination: RE

Abstract. The aim of Deliverable 4.4 is to automatically fill gaps due to heavy occlusion and clutter in recorded 3D data. The recovered 3D structure will feed back into the shared representation developed in T3.1.

Deliverable due: Month 38

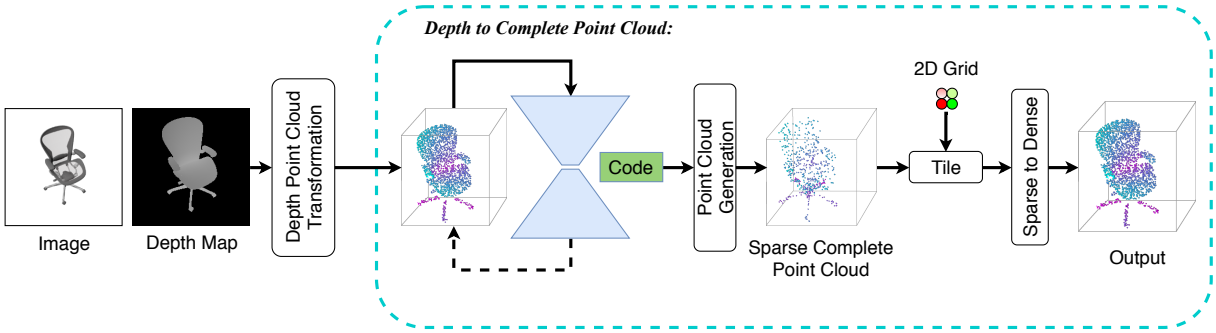

Figure 1: Overview of our framework. Our proposed network receives an image and the corresponding depth map of the input image as input, and calculates the partial point cloud based on the camera geometry. Then, a point cloud auto-encoder is applied to the predicted partial point cloud to generate the representative *code vector*. The full point cloud is computed from the code vector in a sparse-to-dense fashion. Finally, the 3D-2D refinement module enforces the alignment between the generated full 3D point cloud and the depth map.

1 Overview

We approach the problem of missing data completion with a novel framework that generates 3D point cloud of an object from a single-view image.

Previous methods aim to represent the estimated 3D shape as a voxelized 3D occupancy grid. However, those methods are computationally expensive, yielding considerable amount of overhead during both training and inference. To overcome those limitations, recent methods focus on deep CNNs to process and predict 3D point clouds. Nevertheless, they do not impose any constrain for the inference procedure that makes them heavily depend on the quality of the training data and the effectiveness of the learning process to generalize.

First, given an image of an object and the depth map, it computes the point cloud of the visible part of the object. We refer to this (single-view) point cloud as the partial point cloud. The computation of the partial point cloud is based on the camera model geometry. In this way, we explicitly impose the camera model as a geometrical constraint in our transformation to regulate the 2D-3D domain transfer. The point cloud completion module infers the full point cloud using the partial point cloud as input. An encoder-decoder network is used to convert the partial point cloud to the full point cloud [7]. The encoder is an auto-encoder that takes the predicted partial point cloud as input and learns to reproduce it. We use a low-dimensional representation, i.e. *code vector*, as the representative feature vector of the point cloud. The decoder takes this feature vector to produce the full point cloud. Finally, the 3D-2D refinement process enforces the alignment between the generated full point cloud and the depth map. The refinement module imposes a 2D projection criterion on the generated point cloud together with the 3D supervision on the depth.

2 Method

In this section, we give the overview of the each component.

2.1 Partial Point Cloud Generation

First of all, the framework takes a 2D image of an object and the related depth map. Then, the (visible) point cloud is calculated based on the camera model. The aim is to regulate the 2D-3D domain transfer and to constrain the structure of the learned manifold. In this way, we use the provided depth map to compute the partial point cloud. Thus, during inference, geometrical constraints are explicitly incorporated by means of depth estimation and the camera model.

The depth estimation can be achieved by DispNet, FPGA SGM stereo or GT point cloud projection. Then, the partial point cloud is computed using a camera model. For a perspective camera model, the correspondence between a 3D point (X, Y, Z) and its projected pixel location (u, v) on an image plane is given by:

$$Z[u, v, 1]^T = \mathbf{K}(\mathbf{R}[X, Y, Z]^T + \mathbf{t}) \ , \tag{1}$$

where $\mathbf{K}$ is the camera intrinsic matrix. $\mathbf{R}$ and $\mathbf{t}$ denote the rotation matrix and the translation vector.

2.2 Point Cloud Completion

The point cloud completion module consists of two parts: an extraction and a generation stage. The aim of the extraction stage is to concisely represent the geometric information of the partial point cloud by a code vector $\mathbf{v}$. A point cloud auto-encoder is proposed to compute the (lower-dimensional) code vector. The encoder part is based on graph max-pooling [5] and DenseNet [2]. The output of the encoder is passed to a feature-wise global max-pooling component to produce a k-dimensional vector, which is the basis of our latent space. The decoder transforms the latent vector using 3 fully connected layers to produce the same input. We use a $k = 1024$ representation, i.e. *code vector*, as the input for the generation of the full point cloud.

In the generation stage, the network architecture is similar to the decoder of PCN [7]. The code vector is taken as input. It produces a sparse output point cloud by a fully-connected decoder [1]. Then, a detailed output point cloud is obtained by a folding-based decoder [6]. The fully-connected decoder predicts a sparse set of points representing the global geometry of an object. The folding-based decoder approximates a smooth surface representing the local geometry of a shape. We generate $n = 256$ sparse point clouds by the fully-connected decoder, which are later used as key point sets. Eventually, a $N = 1024$ complete point cloud is generated as output of the network.

2.3 3D-2D Refinement

The aim of the 3D-2D refinement is to reduce these estimation error due to misalignments and artifacts. To that end, the generated point cloud is used as a 3D self-supervision component. A point-wise 3D Euclidean distance is used between the partial point cloud and the full point cloud. To constrain the generated point cloud using the 2D projection supervision, we penalize points in the projected image which are outside the silhouette.

3 Results

3.1 Synthetic Natural Environment Dataset

We consider the Natural Environment Dataset (NED) [4] created for the Trimbot2020 project. In contrast to man-made objects, the NED dataset consists of (3D) synthetic scene-centric images from outdoor (natural) environments like gardens and parks. Three categories are selected: hedges, rocks and topiaries. We train and test PSGN [1], GAL [3] and our method on these images, see Figure 2. Both GAL and our method generate accurate 3D point clouds while PSGN fails for some parts of the 3D shapes. Table 1 shows the quantitative results for this dataset. Our proposed method outperforms the other methods to recover the point clouds of the three categories of NED. Additional results and other details can be found in [8][1].

	CD			EMD			IoU		
	PSGN	GAL	Ours	PSGN	GAL	Ours	PSGN	GAL	Ours
hedges	6.445	3.809	**3.298**	11.564	5.836	**5.489**	0.526	**0.704**	0.697
rocks	4.592	2.775	**2.481**	6.973	4.179	**3.927**	0.383	**0.596**	**0.596**
topiaries	3.702	2.285	**2.065**	7.355	3.763	**3.393**	0.435	0.633	**0.648**
mean	4.913	2.956	**2.615**	8.631	4.593	**4.270**	0.448	0.644	**0.647**

Table 1: Chamfer Distance (CD), Earth Mover's Distance (EMD) and intersection over union (IoU) metrics on the NED dataset. Both CD and EMD numbers are scaled by a factor of 100.

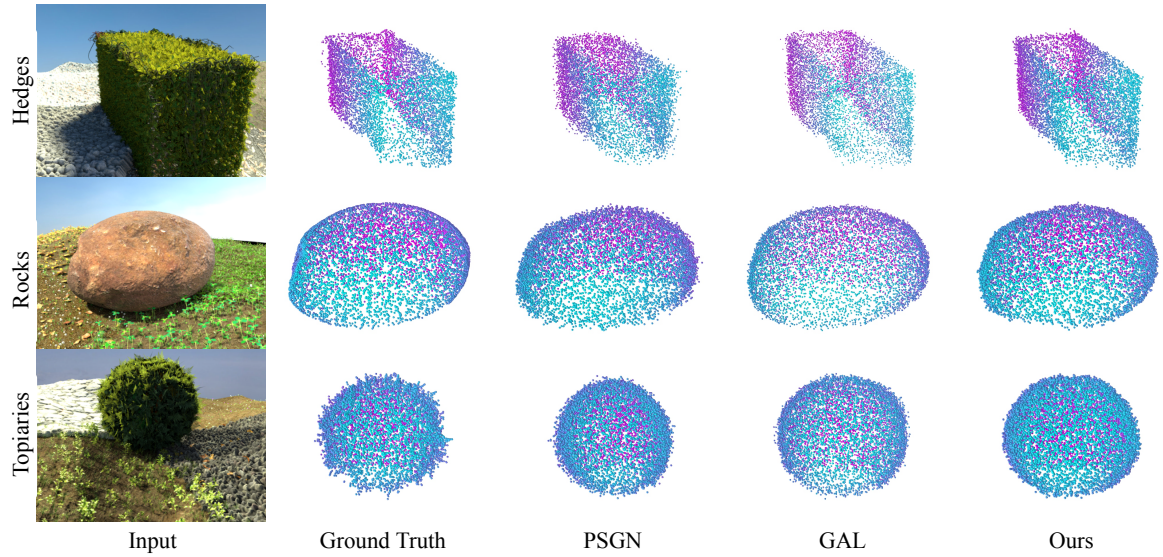

Figure 2: Qualitative results on the object-centric NED dataset.

[1] Under submission to a conference

3.2 Real World Experiments

We also provide test results of our approach on real-world garden images. We use the model trained with the NED dataset directly and run it on real images without any fine-tuning. Figure 3 shows a topiary object and the reconstruction results from four representative viewpoints, where the depth information is provided by the Trimbot FPGA SGM stereo. Figure 4 shows a hedge object and the reconstruction results from four representative viewpoints, where the depth information is provided by GT point cloud projection. Both images are taken from Wageningen test garden. It can be (visually) derived that our model trained on synthetic data (NED) generalizes well to the real-world garden images also with depth maps provided by different modules.

4 Software Package

The algorithm is implemented in Tensorflow 1.0 and is provided within the ROS ecosystem and split into multiple ROS nodes:

- `uva_data_completion`, `uva_data_completion_view`: our pipeline module that receive RGB/grayscale images and the corresponding depth map, and reconstructs a complete point cloud of the object.

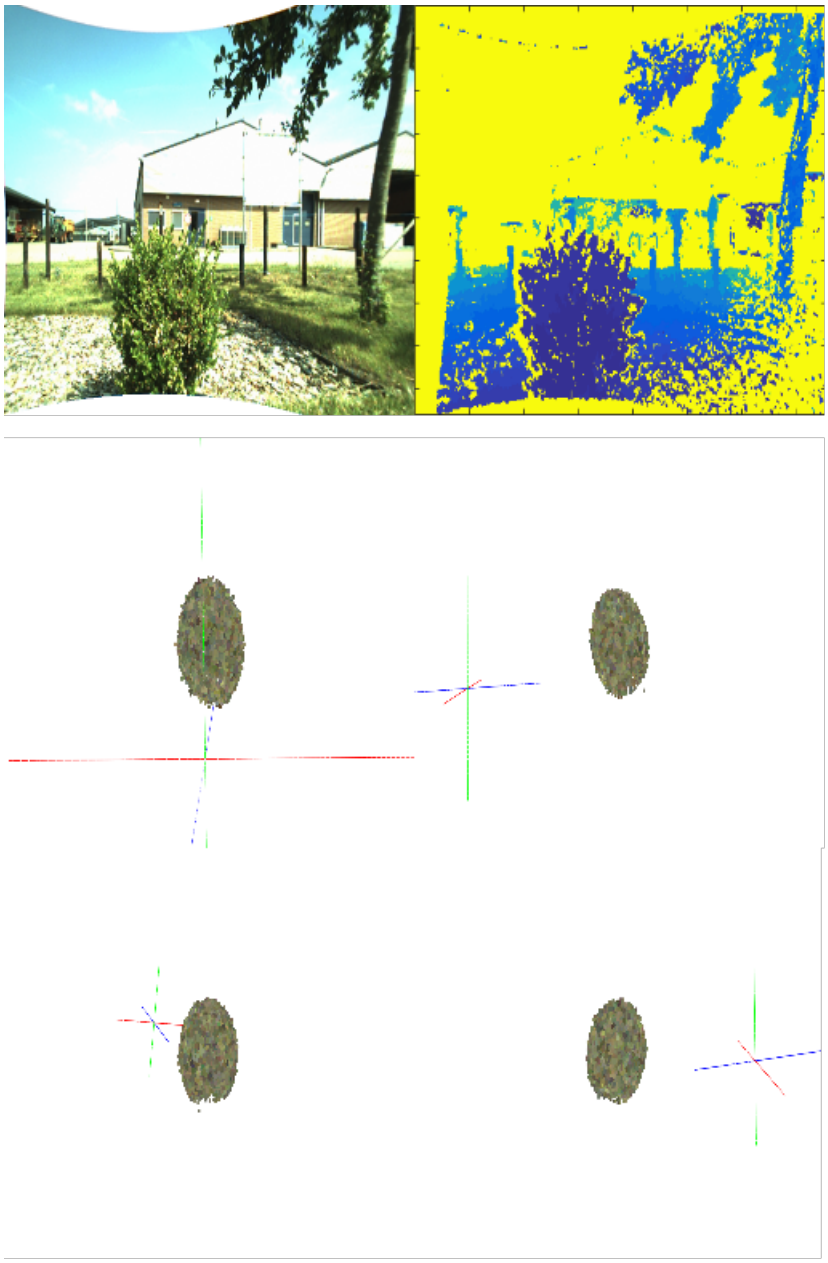

Figure 3: Qualitative results on the Wageningen test garden for a topiary object. First row is input image and the corresponding depth map by the Trimbot FPGA SGM stereo. Next two rows show the reconstruction results from four representative viewpoints.

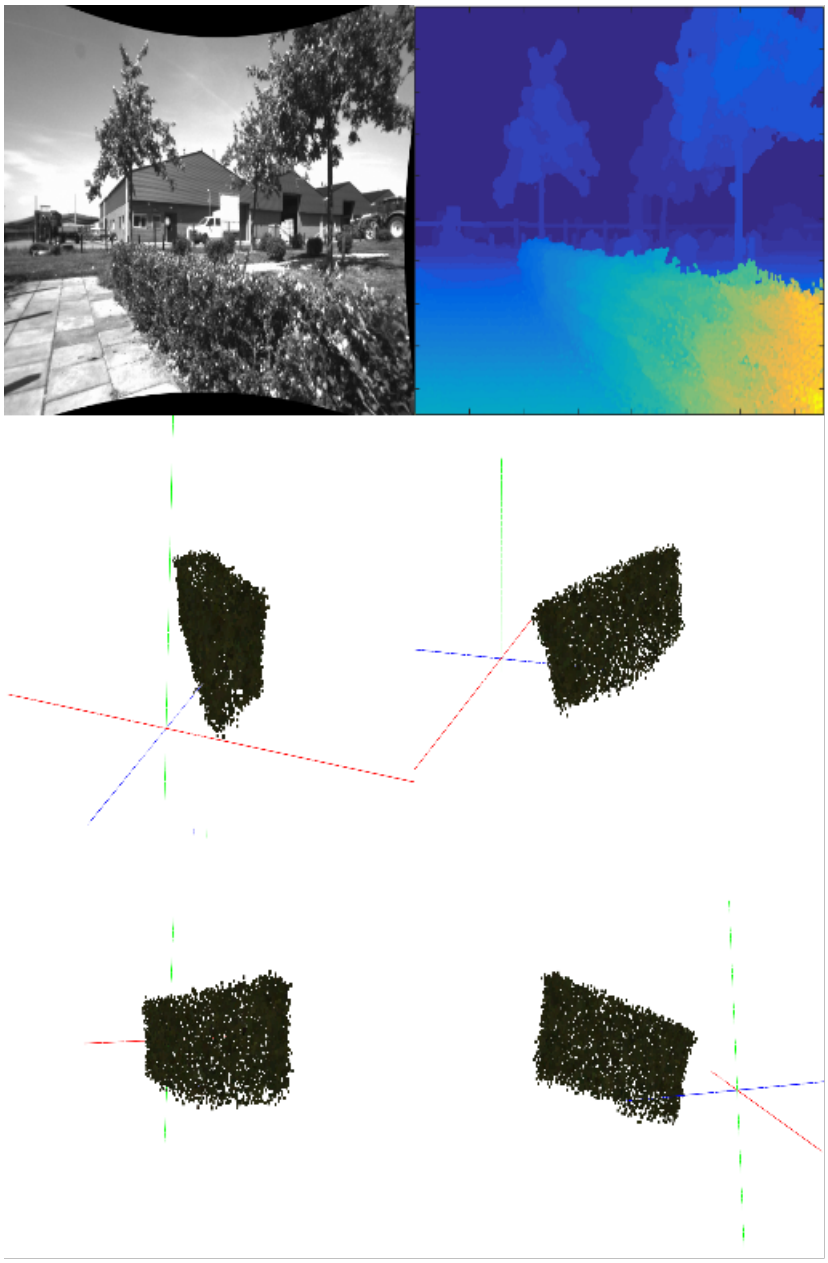

Figure 4: Qualitative results on the Wageningen test garden for a hedge object. First row is input image and the corresponding depth map by GT point cloud projection. Next two rows show the reconstruction results from four representative viewpoints.

References

[1] Haoqiang Fan, Hao Su, and Leonidas J Guibas. A point set generation network for 3d object reconstruction from a single image. In *CVPR*, volume 2, page 6, 2017.

[2] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *CVPR*, volume 1, page 3, 2017.

[3] Li Jiang, Shaoshuai Shi, Xiaojuan Qi, and Jiaya Jia. Gal: Geometric adversarial loss for single-view 3d-object reconstruction. In *European Conference on Computer Vision*, pages 820–834. Springer, Cham, 2018.

[4] Hoang-An Le, Anil S Baslamisli, Thomas Mensink, and Theo Gevers. Three for one and one for three: Flow, segmentation, and surface normals. In *Proceedings of British Machine Vision Conference*.

[5] Yiru Shen, Chen Feng, Yaoqing Yang, and Dong Tian. Mining point cloud local structures by kernel correlation and graph pooling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 4, 2018.

[6] Yaoqing Yang, Chen Feng, Yiru Shen, and Dong Tian. Foldingnet: Point cloud auto-encoder via deep grid deformation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, volume 3, 2018.

[7] Wentao Yuan, Tejas Khot, David Held, Christoph Mertz, and Martial Hebert. Pcn: Point completion network. In *2018 International Conference on 3D Vision (3DV)*, pages 728–737. IEEE, 2018.

[8] W. Zeng, S. Karaoglu, and T. Gevers. Inferring point clouds from single monocular images by depth intermediation. In *arXiv preprint arXiv:1812.01402*.

