Contents lists available at ScienceDirect

Image and Vision Computing

journal homepage: www.elsevier.com/locate/imavis

MVPCC-Net: Multi-View Based Point Cloud Completion Network for MLS data☆

Yahya Ibrahim [a,b,*], Csaba Benedek [a,b]

[a] Institute for Computer Science and Control (SZTAKI), Kende u. 13-17, Budapest 1111, Hungary
[b] Faculty of Information Technology and Bionics, Péter Pázmány Catholic University, Práter u. 50/A, Budapest 1083, Hungary

ARTICLE INFO

Article history:
Received 17 January 2023
Received in revised form 25 March 2023
Accepted 5 April 2023
Available online 11 April 2023

Keywords:
Multi-view
3D point cloud completion
Deep learning
Mobile laser scanning

ABSTRACT

In this paper, we introduce a novel multi view-based method for completing high-resolution 3D point clouds of partial object shapes obtained by mobile laser scanning (MLS) platforms. Our approach estimates both the geometry and color cues of the missing or incomplete object segments, by projecting the 3D input point cloud by multiple virtual cameras, and performing 2D inpainting in the image domains of the different views. In contrast to existing state-of-the-art methods, our method can generate point clouds consisting of a variable number of points, depending on the detailedness of the input measurement, which property highly facilitates the efficient processing of MLS data with inhomogeneous point density. For training and quantitative evaluation of the proposed method, we provide a new point cloud dataset that consists of both synthetic point clouds of four different street objects with accurate ground truth, and real MLS measurements of partially or fully scanned vehicles. The quantitative and qualitative experiments on the provided dataset demonstrate that our method surpasses state-of-the-art approaches in reconstructing the local fine geometric structures as well as in estimating the overall shape and color pattern of the objects.

1. Introduction

As a result of the rapid advancement of 3D data acquisition technology and the decreasing prices of 3D sensors, point clouds have become widely available formats for representing the three-dimensional environment in various robotics, surveillance, and autonomous driving applications. On the other hand, point clouds collected by real scanners frequently provide only partial shapes of the scanned items due to low sensor resolution, occlusions, and the limited number of viewpoints used during scanning. Therefore point cloud completion, i.e. the estimation of an object's full shape from point sets which only partially describe its geometry, is a fundamental key challenge in numerous computer vision and robotic tasks, such as virtual reality (VR)/ augmented reality (AR) applications [1], object tracking and simultaneous localization and mapping (SLAM) [2,3].

Mobile laser scanning (MLS) platforms equipped with time synchronized Lidar sensors and navigation units can produce very dense and feature-rich point clouds for urban areas, as shown in Fig. 1(a). However, due to the fact that the scanning vehicle can only move on the road, the point cloud models collected for many field objects have incomplete shapes. For example, the sidewalk sides of the parking vehicles are typically missing from the scanned scene models (Fig. 1). Exploiting that available semantic point cloud segmentation methods [4,5] can efficiently separate regions of different object classes in an MLS scene, we address in this paper the task of filling the missing regions of selected MLS object shapes to create a more realistic representation of the real environment. For example, from vehicle regions segmented from the raw MLS point cloud (Fig. 1(b)), we aim to derive completed vehicle point cloud models (Fig. 1(c)).

Existing point cloud completion methods apply various approaches to address the underlying unstructured nature of the point clouds, such as voxelization [6], intermediary 3D grids [7], or directly processing the point cloud [8] with the PointNet encoder [5]. These techniques have achieved remarkable success in terms of estimating complete geometric models of various object shapes. However, the obtained 3D point cloud models are often only roughly detailed, because the above methods are restricted to provide outputs with a fixed constant number of points, regardless of the size, shape complexity, or resolution of the

☆ A demonstration video with further results is available from the following url, which will also include links to the source code and to the new benchmark database: https://github.com/sztaki-geocomp/Multi-view-3d-completion.
* Corresponding author at: Institute for Computer Science and Control (SZTAKI), Kende u. 13, Budapest 1111, Hungary.
E-mail address: ibrahim.yahya@sztaki.hu (Y. Ibrahim).

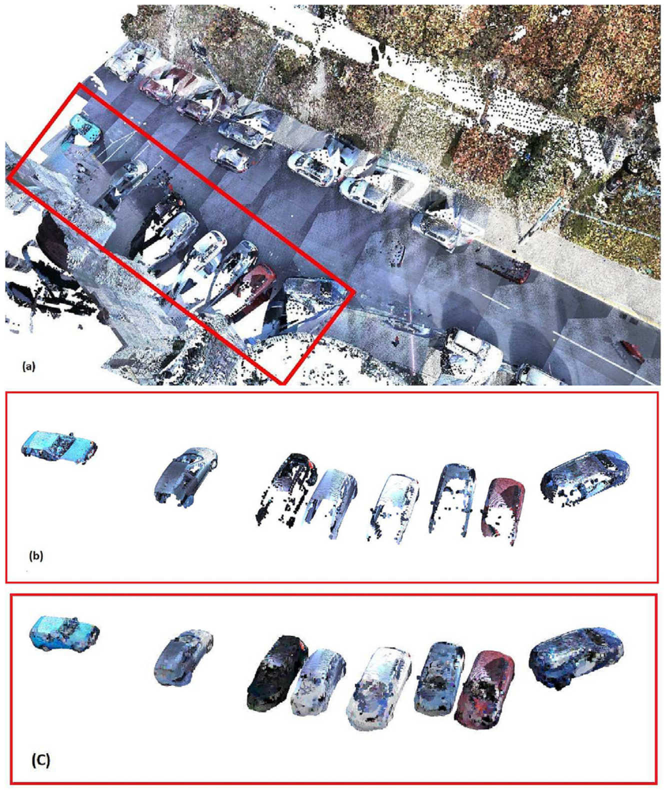

Fig. 1. MLS data and our results on vehicle shape completion: (a) MLS data from a Budapest street showing numerous incomplete car shapes, (b) Partial car shapes obtained by semantic point cloud segmentation [4] from the MLS scene, (c) Our results for completing the car shapes.

input point cloud measurement segment corresponding to a given object.

In order to apply the aforementioned techniques [6–8] to MLS data, the input point cloud should be spatially downsampled, yielding as output simplified object shape models with significantly lower point density and less geometric detailedness compared to the genuine Lidar measurements. Moreover, a typically featured test scenario of these methods focuses on generating realistic object shapes from only a few (e.g. less than 15) measurement points, which task has a large degree of freedom in terms of the possible acceptable solutions: these use-cases are more connected to shape generation than shape completion.

In this paper, we introduce a point cloud completion algorithm that can handle the unstructured nature, and maintain the high resolution of the MLS measurement data. To address the above challenges, we propose the *Multi-View Based Point Cloud Completion Network* (MVPCC-Net), a network designed to generate dense and detailed 3D object models from partial point cloud measurements. The algorithm encodes the input 3D point cloud by a set of sparsely filled multi-channel images representing both geometry and color information available from the sensor data. This approach allows us to use 2D Convolutional Neural Networks (CNNs) for filling in the missing structural and color information in the 2D image domain. Thereafter, the inpainted multi-view grid maps are fused to produce dense 3D point clouds representing the completed object shapes.

The preliminary version of the proposed technique was presented in [9]. In this article, we provide an extended model which can deal with different types of synthetic and real MLS point cloud objects, mainly related to urban street scenarios. We also provide extensive ablation experiments demonstrating the efficiency of various elements of our proposed model, including the choice of the optimal number of views used to encode the 3D objects, and our applied view-fusion strategy to synthesize the 3D point cloud output. In addition, we present here a novel point cloud dataset which consists in part of synthetic data, and in part of real colored MLS point clouds of partially or fully scanned vehicles, captured by a car-mounted mobile laser scanning system in Budapest, Hungary. Since by processing real MLS street measurements, we cannot rely on accurate ground truth models, our method was trained on the synthetic part of the dataset derived from ShapeNet [10] for four street object classes. The ShapeNet-based point clouds are used as well for quantitative comparison of our technique versus various state-of-the-art methods. On the other hand, we also demonstrate that with using an additional network component for orientation adjustment of the input point clouds, the MVPCC-Net trained on purely synthetic data can be directly applied for completing real MLS object samples without the need of any additional fine-tuning step.

The contributions of this paper can be summarized as follows:

1. A novel multi-view image set-based deep neural network is proposed for completing 3D point clouds of objects. Our approach directly uses 2D CNNs for geometry and color inpaiting, by exploiting a new data structure, which represents the 3D incomplete point cloud measurements by multiple 6-channel images ($3 + 3$ channels for geometry and color, respectively) generated from various perspectives. As output, colored 3D point cloud models of the completed object shapes are provided, allowing a variable number of points, and varying point density for efficient representation of the objects.
2. Extensive quantitative and qualitative tests on synthetic and real MLS datasets demonstrate that our method outperforms the current point cloud-based object completion networks for selected street object classes.
3. Extensive ablation experiments are conducted for evaluating the main system components and hyperparameter settings, including the number of views by multi-view mapping, and the view fusion process.

2. Related work

Numerous deep learning approaches have been developed for 3D point cloud processing. In volumetric approaches, point clouds are *voxelized* using a 3D grid which is taken as input of a three-dimensional convolutional neural network. Multi-view techniques project 3D point clouds to several planes from various perspectives and extract view-wise information. The PointNet method [5] and its extensions [11] directly process the point clouds using a symmetric function, which allows the network to tolerate uncertainty in the order of points, and accurately captures both global and local properties of a point cloud.

In this section we provide a methodological review on state-of-the-art algorithms used for 3D shape completion and on multi-view approaches for point cloud processing:

3D shape completion methods can be categorized into three main groups: Geometry-based approaches [12] have effectively been utilized to repair small holes on point clouds, using geometric restrictions such as local surface or volumetric smoothness. Template-based approaches [13] deform or reconstruct 3D point clouds that correspond to the most similar templates detected in a 3D shape database.

Learning-based approaches have been widely adopted by 3D point cloud completion techniques due to the availability of synthetic public datasets like ShapeNet. PointNet is utilized as an encoder in multiple state-of-the-art techniques [8,14–16] with various types of decoders: FoldingNet's decoder [16] warps a predefined 2D grid so that it fits the input point cloud, by using two successive three-layer perceptrons. The Point Completion Network (PCN) [8] employs two-stage decoders that combine the advantages of fully-connected and folding-based decoders. Extending the PCN network, the Vehicle Points Completion-Net (VPC-Net) [15] combines the partial inputs with the PCN's decoder outputs to construct more homogeneous point clouds with finergrained information. TopNet [14] includes a decoder that generates point clouds in a hierarchical structure, where each point operates as a branch of a tree. SoftPoolNet [17] provides a two-stage, multi-resolution architecture for completing point clouds by substituting max pooling with softmax function in order to retain local information. The 3D-Encoder-Predictor Network (3D-EPN) uses directly a volumetric representation of 3D point clouds [6]. However, converting point clouds to 3D volumes yields data quantization, which step can remove many fine-grained details. For this reason, in the Gridding Residual Network (GRNet) [7] 3D grids are proposed to regularize unstructured point clouds while keeping their context and structure detailed.

The recent PoinTr [18] method and its extension called AdaPoinTr [19] consider the point cloud completion issue as a set-to-set translation problem, so they transform the point clouds into a sequence of point proxies, then they use a Transformer encoder-decoder architecture for point cloud completion. A topology-aware method called LAKeNet [20] fills in the missing parts of the 3D point cloud's structure by using three steps: aligned keypoint localization, surface skeleton generation, and shape refinement. SeedFormer [21] introduces a novel Upsample Transformer with a new shape representation (Patch Seeds) to preserve both regional information and global structures. Snowflake Point Deconvolution (SPD) is an approach developed by SnowflakeNet [22] to complete the shape of the point cloud, where child points are generated progressively from selected parent points.

However, all of the above methods use Chamfer Distance (CD) as training loss, thus that they minimize the mean of local point-to-point distances between the predicted and the ground truth point clouds, which process does not guarantee the effective characterization of shape similarity [7].

Multi-view based approaches have recently shown their efficiency in several tasks, including classification [23,24] and segmentation [25–27], moreover, reconstructing 3D shapes from a single image or a series of images is an active research area with different applications in robotics, and in virtual/augmented reality. Existing approaches adopt various output representations. A voxel-based output is provided by [28], where a 2D convolutional neural network encodes 2D images into a latent representation, which is subsequently decoded into 3D object shapes by a 3D convolutional neural network. The Point Set Generation Network [29] derives a point-based output, so that a set of unordered points is directly extracted from a single image. Lin et al. [30] present pseudo-rendered depth images as output and construct dense 3D objects by re-projecting them.

Image and point cloud fusion-based techniques have also been proposed for 3D shape completion recently. The View-guided Point Cloud completion method (ViPC) [31] relies on auxiliary RGB image data for point cloud completion, assuming that the input image contains structural information for the missing shape part. Similarly, the Cross-modal Shape-transfer model (CSDN) [32] combines the image and point cloud information in expressing a full shape.

3. MVPCC-Net

The main goal of the proposed 3D point cloud completion approach is to transform a point cloud segment representing only a portion of an object into a point cloud describing the entire object shape. For uniform treatment, we initially ensure that the incomplete input point cloud segments are scaled and reshaped to fit inside a 3D unit bounding box with point coordinates in the range of [-0.5, 0.5].

As shown in Fig. 2, our method implements three sequential steps: (a) Calculation of a multi-view image-based representation of the incomplete point cloud measurement, (b) Shape and color completion in the 2D image domains using an inpainting network, and (c) Re-projection of the inpainted multi-view images to obtain a completed 3D point cloud model of the object of interest. The three main steps are introduced in the following subsections in detail.

3.1. Multi-view 3D representation

Let us denote by $\mathcal{P}_{\text{in}}$ the input point cloud of N_{in} points, representing a single, partially scanned scene object:

$$\mathcal{P}_{\text{in}} = \{p_n\}_{n=1}^{N_{\text{in}}},$$

where each point p_n is associated to a 6D descriptor comprising three location coordinates (XYZ) in the point cloud's local Descartes coordinate system, and three color coordinates (RGB).

In the first step, the point cloud $\mathcal{P}_{\text{in}}$ is mapped to different 2D grids from a variety of perspectives: multi-view 2D images are captured by a set of V virtual cameras located at predetermined positions around the object's 3D bounding box. Henceforward, $v \in \{1, \dots, V\}$ refers to the index of a selected view. Assuming that the position and orientation of each virtual camera are known, the view-transformation for each camera v can be described by a R^v rotation matrix and a t^v translation matrix, which can be analytically calculated. Thereafter, the input point cloud $\mathcal{P}_{\text{in}}$ can be transformed to the reference coordinate system of camera v as follows:

$$q_n^v = R^v \cdot p_n + t^v, \quad n = 1, \dots, N_{\text{in}}; \quad v = 1, \dots, V \tag{1}$$

where q_n^v denotes vth the camera coordinates corresponding to point $p_n \in \mathcal{P}_{\text{in}}$. Next, each point is projected onto the camera plane using the virtual cameras' projection matrix K, which is determined by the intrinsic camera parameters:

$$(x_n^v, y_n^v) = K \cdot q_n^v \quad n = 1, \dots, N_{\text{in}}; \quad v = 1, \dots, V \tag{2}$$

The intrinsic settings of the cameras are adjusted to ensure that the objects of the training dataset are entirely contained within the considered image windows, while the coverage rate of the projected regions in the images is maximized. The result of Eq. (2) is a 2D pixel position (x_n^v, y_n^v) on the image plane of camera v. To keep the genuine 3D geometric (XYZ) and 3D color (RGB) information from in the original point cloud $\mathcal{P}_{\text{in}}$, we store the point projections in a six-channel image I_{in}^v associated with each view v. More specifically, if point p_n is projected to pixel (x_n^v, y_n^v) in view v, the values of the different image channels at the given pixel of I_{in}^v are equal to the concerning (XYZRGB) coordinates of p_n.

Note that by projecting 3D point clouds to 2D images, multiple points might be projected to the same pixel of a given image lattice. We handle this issue by sorting these points by their distances from the given camera, and we only retain the closest points, keeping only the exposed portion of the object from the camera perspective. The presence of multiple cameras from different directions will ensure that the object points visible from any viewpoint will be represented by one or multiple projected images.

On the other hand, following the above procedure, the different I_{in}^v camera images will be only sparsely filled, containing several pixels without any point projections: by these pixels we set zero values for all channels.

In summary, as shown in Fig. 2(a), this step derives a collection of multichannel images, that can be directly processed by 2D CNN architectures (see Section 3.2). Each camera records a six-channel image, comprising geometry (XYZ channels) and color (RGB channels) information, however the geometry is directly stored in the point cloud's original Descartes coordinated system. As a result, a point projected to multiple virtual camera planes will have the same geometrical coordinates in each view image. We demonstrate later in Section 3.3 that this property is highly beneficial during the 3D point cloud re-projection phase of the process.

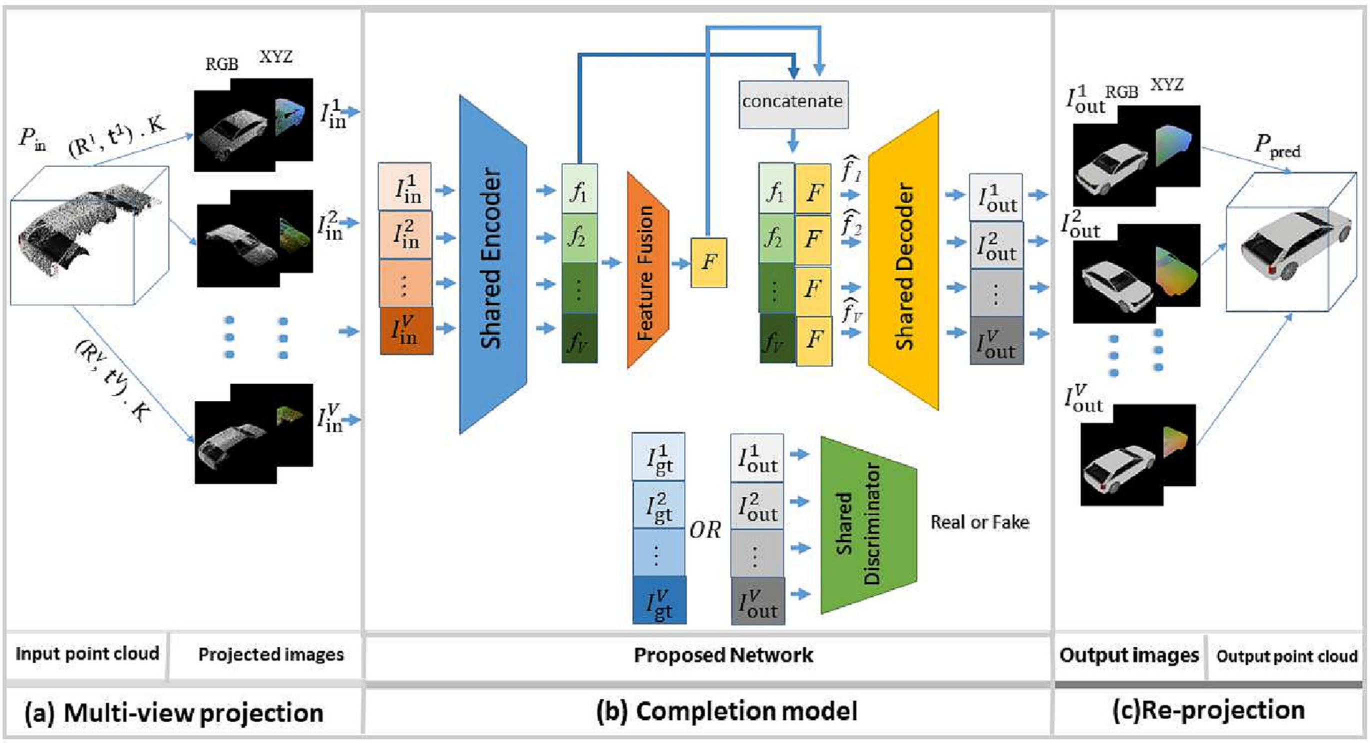

Fig. 2. Dataflow of our algorithm: (a) Multi-view projection: incomplete point cloud measurement is represented as multi-view images, each view is recorded as a six-channel image with RGB color information and XYZ geometry information; (b) Completion model: it completes the shape and color information in the 2D image domain; (c) Re-projection: the inpainted multi-view images are reprojected into the 3D space to generate a completed 3D point cloud model of the object.

3.2. Completion model

The second main step of the proposed approach (see Fig. 2(b)) performs structure and color inpainting of the missing object regions in the domain of the six-channel I_{in}^v images generated from different viewpoints in the previous phase. The resulting inpainted images will later guide the generation of the final 3D point cloud in the last step (Section 3.3).

Our method uses a Generative Adversarial Network (GAN) [33] architecture, that consists of the *Generator* (G) and *Discriminator* (D) networks.

Our *Generator* implements three subsequent steps:

(i) First, we use a *shared encoder* for all views, which applies two-stage downsampling followed by using eight residual blocks to separately encode each of the six-channel images I_{in}^v associated with view v to its own view-level latent feature cube f_v of size $64 \times 64 \times 256$, for $v \in \{1, \ldots, V\}$.

(ii) The intermediate *feature fusion* phase employs (3×3) convolutions, followed by a ReLU activation function, to fuse all the view-level features denoted by $[f_1, f_2, \ldots, f_V]$ into a single global feature cube F of the same size as each f_v. The global feature is expected to facilitate the transmission of shared characteristics between distinct viewpoints.

(iii) In the third step, we take each view-level latent feature f_v, and concatenate it to the global feature F calculated in the previous phase, obtaining a $\hat{f}_v = [f_v, F]$ extended feature cube of size of size $64 \times 64 \times 512$ for each view $v \in \{1, \ldots, V\}$. Next, the *shared decoder* processes the different view's $\hat{f}_v$ features sequentially, so that each $\hat{f}_v$ is upsampled to the size of the original I_{in}^v image using dilated convolutions, with a dilation factor of two. The output of the decoder is an inpainted I_{out}^v image, with the same size and six-channel format (i.e. XYZRGB channels) as the encoder input I_{in}^v. Let us observe, that although the decoder generates the different I_{out}^v images separately for the different views, the decoder's input features contain information from all views via the F global feature component of $\hat{f}_v$, thus cross-view fusion is implicitly implemented at this step.

Next, the six-channel I_{out}^v images predicted by the *Generator* are presented to the *Shared Discriminator*, whose task is to decide whether they are real or fake. The discriminator architecture is based on the 70×70 PatchGAN [34], which determines whether overlapping image patches of size 70×70 are real or not.

For model training, we follow a supervised approach using Ground Truth (GT) images projected from complete object models, as detailed later in Section 4.1. In the training phase, we attempt to ensure that for each view v the six-channel image $I_{out}^v = G(I_{in}^v)$predicted by the *Generator*, becomes as similar to the GT image I_{gt}^v as feasible. The network is trained using a combined loss function consisting of six subterms: smooth $L1$ loss, adversarial loss, perceptual loss [35], style loss [36], binary cross entropy loss and Total Variation loss [37] as detailed in the following.

The *smooth L*1 *loss* term ℓ_{L1} is used to keep the distance low between the algorithm's six-channel image output and the corresponding (GT) target image. Smooth $L1$ loss combines the benefits of $L1$-loss and $L2$-loss by providing stable gradients when the distance is high, while it reduces the oscillations when the distance is small [38].

The adversarial loss ℓ_{adv} is also applied to all channels of the generated I_{out}^v image. It is presented as a zero-sum competition between the generator and discriminator networks so that the generator attempts to minimize the value defined by Eq. (3), while the discriminator attempts to increase it:

$$\ell_{\text{adv}} = \mathbb{E}_{\{I^v_{\text{gt}}\}}\left[\log\left(D\left(I^v_{\text{gt}}\right)\right)\right] + +\mathbb{E}_{\{I^v_{\text{out}}\}}\left[\log\left(1 - D\left(G\left(I^v_{\text{in}}\right)\right)\right)\right]. \tag{3}$$

The *perceptual loss* ℓ_{perc} and *style loss* ℓ_{sty} terms are calculated exclusively for the RGB color image channels to make them perceptually and stylistically more similar to the GT's coloring. The *binary cross entropy*ℓ_{Lcro} measures the difference between two binary masks defined by non-zero pixels in the output image I^v_{out} and the ground truth I^v_{gt}, respectively. Finally, the *Total Variation* (TV) loss ℓ_{TV} - which promotes spatially smooth output images - is utilized to smooth the geometric output channels.

The *Generator*'s combined loss function ℓ_G is derived from the above defined six subterms, with using $\lambda_1, \ldots, \lambda_6$ regularization parameters discussed in Section 4.2:

$$\ell_G = \lambda_1 \ell_{L_1} + \lambda_2 \ell_{\text{adv}} + \lambda_3 \ell_{\text{perc}} + +\lambda_4 \ell_{\text{sty}} + \lambda_5 \ell_{\text{Lcro}} + \lambda_6 \ell_{TV} \tag{4}$$

3.3. Re-projection

As a result of the previous step, a set of inpainted six-channel images are available, which represent the projections of the object shape from various viewpoints. Each non-zero pixel in each view image encodes a 3D point in the object's coordinate system, where the geometry channels determine the given point's normalized XYZ Descartes coordinates, and the color channels define the associated RGB value. Consequently, the completed point cloud of the object can be derived in a straightforward way by re-projecting all points stored in the *V* different views one after another to the same 3D space (see Fig. 2(c)). Let us observe that this process admits generating output point clouds that consist of a variable number of points. More specifically, the number of points added to the incomplete input point cloud is determined by the total number of inpainted pixels in the view images.

During this stage of the algorithm, two post-processing steps are applied for enhancing the quality of the generated point cloud. First, since we observed that re-projecting the boundary pixels of the objects from the different image views results in noisy points surrounding the 3D shape, we slightly erode the foreground regions of the inpainted images before executing the re-projection. Second, we also employ a statistical outlier filter [39] to eliminate further outliers from the final point cloud.

4. Experiments

4.1. Data generation

While supervised deep learning-based algorithms require extensive training data, collecting proper 3D point cloud measurements for our method from real-world environments – including both incomplete and complete or manually completed object shapes – is highly challenging and resource intensive. To circumvent these limitations, we trained and quantitatively evaluated our model using synthetic data, which consists of pairs of partial and complete point cloud models of various 3D object shapes, so that the incomplete point clouds can be used as input of our method, while the complete shapes of the same objects as ground truth. In addition, we extensively tested our trained shape completion network on real-world (incomplete) Mobile Laser Scanning (MLS) measurements. We provide access to both the synthetic and real data used in our tests in a novel public dataset for the scientific community.

4.1.1. Synthetic data

Recent works [8,14,15] utilized ShapeNet [10], a large-scale 3D synthetic dataset to construct the training data, deriving point clouds from meshes available in ShapeNet by sampling a predefined number of points. However, point cloud models generated in this way often cannot be considered as efficient references for real MLS data, since, for example, they may also contain various internal object structures which are occluded during an outdoor scanning process.

To address the above issue, we used the approach described by [40] to generate the (Ground Truth) 3D point clouds of objects from their complete mesh models. First we render a model based on projections of the mesh to discrete 2D lattices from distinct viewpoints, which are re-projected in the next step to the object's coordinate system. We set the resolution of the lattice of projection so that for each object we produce a dense, colored point cloud that accurately depicts even fine visible surfaces. Henceforward we refer to this point cloud as *fine GT*, where the different object models may consist of a variable number of points depending on the detailedness of the shape's mesh model.

Next we generate incomplete point cloud models for the objects of interest, which can be used as input of our algorithm during training and also in the test phase for quantitative evaluation. Here the previously created point clouds models of the complete object shapes are projected by a *subset* of the above defined virtual cameras, and the views of the selected cameras are re-projected in the 3D space. In this way we can synthesize partial object point cloud samples, that represent only points visible from the selected virtual cameras.

As processing urban MLS data is the primary goal of our approach, we selected four object categories from ShapeNet, that are relevant for street scenarios: car, bus, motorcycle, and train. Since a MLS system captures a scene from the top of a scanning vehicle, it often cannot capture the bottom part of street objects. Therefore by simulated MLS point cloud generation, we did not place upward looking virtual cameras to the ground plane, we applied instead several side-view and downward facing cameras around the object.

Our synthetic dataset contains in total 4918 distinct models, of which 4580 objects are used to train our model and 338 ones are used for evaluation. Twenty partial samples were generated for each complete object shape using twenty distinct perspectives, yielding a training set of 91,600 objects and a test set of 6760 samples. Some partial object samples of the new dataset are shown in the first column of Fig. 3, while the last column of the same figure demonstrates the corresponding complete point clouds used as ground truth.

Technically, our deep neural network is trained using a set of 2D six-channel images, which are derived from the partial and complete point cloud samples by projections in advance. To prepare the data, in the preprocessing phase, we generate twenty images from twenty distinct perspectives, while in the training phase, we follow a preliminary fixed view selection strategy - presented in Section 6.1 - which ensures that the selected views evenly surround all sides of the object.

4.1.2. MLS data

Apart from synthetic training and test data generation, we also created a real-world test set that consists of (mostly partial) vehicle point clouds extracted from measurements of a Riegl VMX-450 MLS scanner. The raw MLS test data was provided by the City Council's Road Management Department (Budapest Közút Zrt.) in Budapest, Hungary. For ensuring accurately segmented vehicles in the new test set, we utilized a user friendly 3D point cloud annotator tool described in [4]. In the preprocessing phase, each object sample was scaled and transformed to fit within a 3D box with coordinates between −0.5 and 0.5.

Our real MLS data collection consists of 424 object samples in total. On one hand, 370 point clouds represent partial vehicle shapes, where the scans of the complete objects are not available in the MLS data, thus they can only be used for qualitative analysis of the proposed technique (see Fig. 5). On the other hand, 54 samples depict almost entire vehicle shapes (see Fig. 4 (g)), which can be also used as ground truth similarly to the synthetic models presented in Section 4.1.1. Here we generated four partial point cloud samples from each complete MLS vehicle shape generating overall 216 samples, each one was created by reprojecting an image created from a single virtual camera position, which was located in the front, behind, to the right, or to the left of the selected object of interest (see Fig. 4 (a)). Note that while the

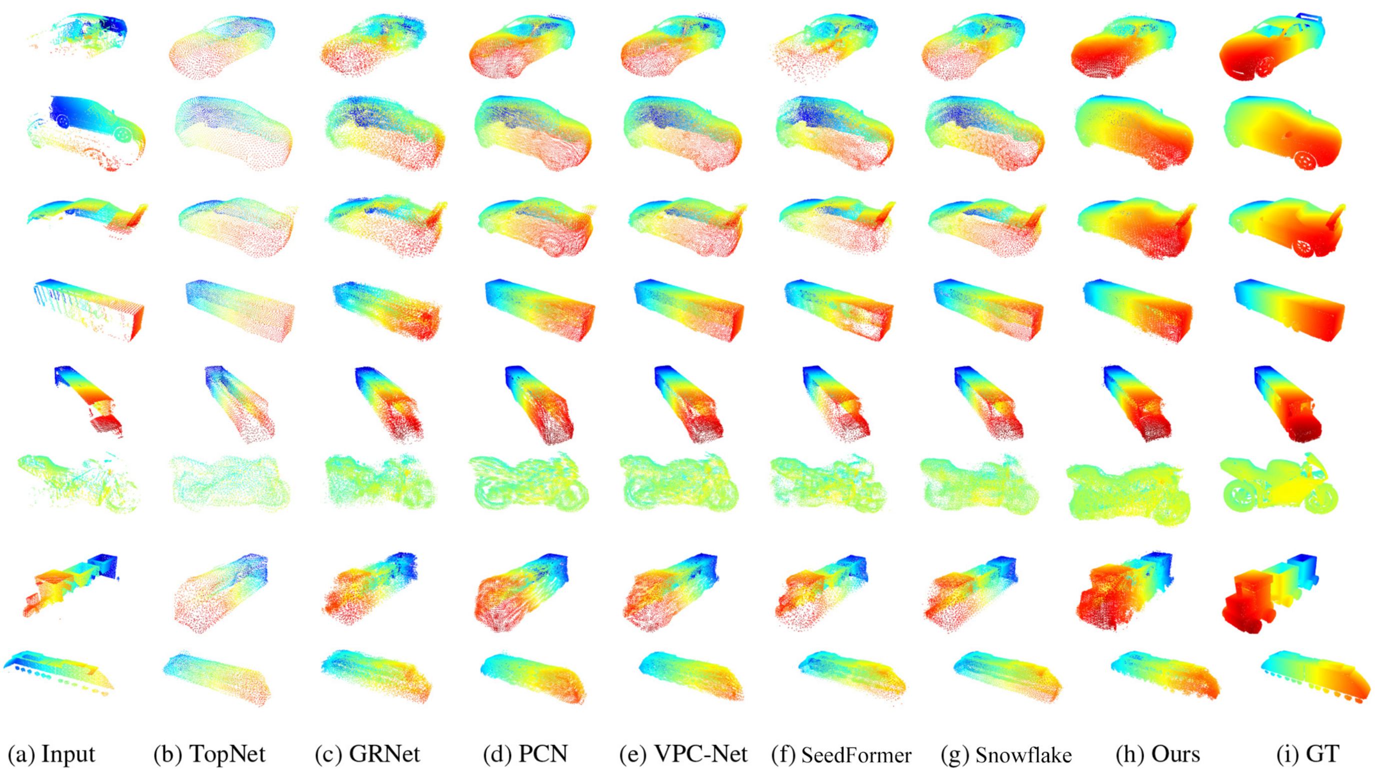

Fig. 3. Qualitative results on the synthetic dataset, where we present the input partial point cloud, the results of the references methods, PCN, TopNet, GRNet, VPC-Net, SeedFormer, SnowflakeNet, Our results, and the GT stands for the complete 3D object.

synthetic dataset has a quite homogeneous point cloud characteristic, we can observe notable point density variations within the set of real MLS samples. In our dataset, an input MLS point cloud representing an incomplete car shape consists of – in average – 31 K points, and their size range varies between 2 K and 130 K points, while the complete car point clouds contain 35 K points in average, spanning the range of 15 K–72 K points.

4.2. Model training setup and strategy

PyTorch is used to implement the proposed completion network, which is trained on 256×256 images. As an optimization algorithm, we employ the Adam optimizer [41] with the settings $\beta_1 = 0$ and $\beta_2 = 0.9$. The default batch size is 4, but when the number of selected views is greater than five, the batch size is reduced to 1, so that the computation can be completed on the GPU, and the weights are only updated in every four iterations. The model is trained in three sessions in which the learning rate parameter, that determines the step size at each iteration while moving toward a minimum of the loss function, is gradually decreased: in each section, we train the model until converge, while using learning rates 10^{-4}, 10^{-5}, and 10^{-6} respectively. In the loss function, the TV loss term is only used in the last two sections.

For our experiments, the following regularization hyperparameters of the loss function are used: $\lambda_1 = 50, \lambda_2 = 0.1, \lambda_3 = 250, \lambda_4 = 0.1, \lambda_5 = 0.1, \lambda_6 = 0.1/S$, where S refers to the image size (here $S = 256 \times 256$). The first three parameters were chosen based on [42], and the last three parameters were chosen to balance the impact of each loss. In the re-projection stage, based on ablation experiments detailed in Section 6, we use the erosion operator with 3×3 rectangular kernels as structural elements, and the statistical outlier removal filter with the standard deviation parameter of 5 calculated among 20 neighbors around each object point.

4.3. Evaluation methodology

Since the output of the proposed method is a colored point cloud, we separately evaluate the quality of object geometry prediction and the realistic nature of RGB color estimation.

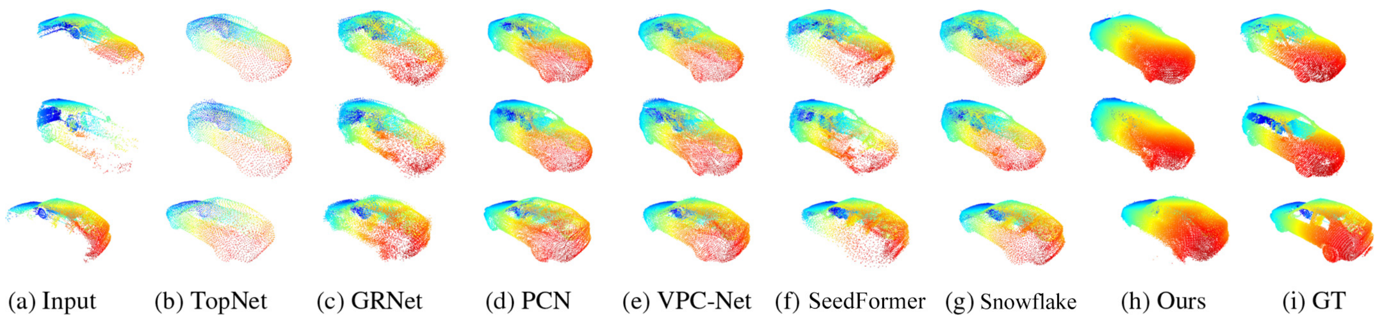

Fig. 4. Qualitative results on Real-world dataset, where we present the input partial point cloud, and the results of the references methods, PCN, TopNet, VPC-Net, SeedFormer, SnowflakeNet, Our results, and the GT stands for the complete 3D object.

We perform the *geometric assessment* of the object shapes against various state-of-the-art 3D point cloud completion approaches in the point cloud's local 3D coordinate system, so that we compare each predicted point cloud ($\mathcal{P}_{\text{pred}}$) to the corresponding ground-truth point cloud ($\mathcal{P}_{\text{gt}}$). For quantitative analysis, we rely on the Chamfer Distance (CD) and the F1-score, which are two frequently used measures for comparing the similarity of two sets of points [7,17]. The Chamfer Distance is calculated by searching for the closest point pairs between the predicted point cloud $\mathcal{P}_{\text{pred}}$ and the corresponding ground truth $\mathcal{P}_{\text{gt}}$ in two directions, as described by Eq. (5).

$$\begin{aligned} D_{\text{CD}}(\mathcal{P}_{\text{pred}}, \mathcal{P}_{\text{gt}}) = & \frac{1}{2N_{\text{pred}}} \sum_{p \in \mathcal{P}_{\text{pred}}} \min_{q \in \mathcal{P}_{\text{gt}}} \|p-q\|_2^2 \\ & + \frac{1}{2N_{\text{gt}}} \sum_{q \in \mathcal{P}_{\text{gt}}} \min_{p \in \mathcal{P}_{\text{pred}}} \|p-q\|_2^2, \end{aligned} \tag{5}$$

where N_{pred} and N_{gt} denote the number of points in $\mathcal{P}_{\text{pred}}$ and in $\mathcal{P}_{\text{gt}}$, respectively; and $\|p-q\|_2^2$ stands for the Euclidean distance between the locations of points p and q.

For considering an alternative geometric accuracy measure of point cloud completion, we also calculate the F1-score, which considers as a match any pair of points whose distance is less than a given distance threshold τ. The F1-score (F1) as a function of τ is computed as follows:

$$\text{F1}(\tau) = \frac{2 \cdot \text{Pr}_\tau \cdot \text{Rc}_\tau}{\text{Pr}_\tau + \text{Rc}_\tau}, \tag{6}$$

where Pr_τ and Rc_τ stand for Precision and Recall, for a given threshold τ:

$$\text{Pr}_\tau = \frac{1}{N_{\text{pred}}} \sum_{p \in \mathcal{P}_{\text{pred}}} \left[\min_{q \in \mathcal{P}_{\text{gt}}} \|p-q\| < \tau \right] \tag{7}$$

$$\text{Rc}_\tau = \frac{1}{N_{\text{gt}}} \sum_{q \in \mathcal{P}_{\text{gt}}} \left[\min_{p \in \mathcal{P}_{\text{pred}}} \|p-q\| < \tau \right], \tag{8}$$

where based on [7] we adopted the threshold $\tau = 0,01$, commonly used for normalized point coordinates.

In contrast to 3D geometry analysis, we can mainly rely on qualitative tests for evaluating the *color prediction* of the proposed method, which can be performed either in the 3D point cloud space (Section 5.2), or in the 2D image domain of the individual views (Section 6). Since the considered reference methods only deal with geometry completion, they cannot be involved here in the comparative tests.

Generally, the assessment of RGB image inpainting is regarded as a highly subjective process, where we cannot find any straightforward numerical metric for evaluation of the results [43,44]. However, there are a number of standard evaluation metrics used in literature which we also adopt here, including the Peak Signal-to-Noise Ratio (PSNR), the Structural Similarity Index (SSIM) [45], and the Relative $L1$ error [44]. We will calculate the later measures in the ablation experiments (-Section 6), where we analyze their dependency on various settings of the proposed model.

5. Results and discussions

We have trained and evaluated the proposed technique using our new dataset introduced in Section 4.1, which consists of both synthetic and real MLS object point cloud samples. In this section, we present a detailed quantitative and qualitative performance analysis, and comparison versus various state-of-the-art reference methods.

5.1. Comparative evaluation on synthetic data

In the first part of the evaluation process, we use the synthetic dataset presented in Section 4.1.1 for testing our method, and for comparing it to four recent state-of-the-art 3D point cloud completion algorithms: TopNet [14], GRNet [7], PCN [8], VPC-Net [15], SeedFormer [21] and SnowflakeNet [22].

For providing a fair comparison, a number of careful considerations should be taken. First, while our proposed model is able to process and generate object point clouds that consist of a variable number of points, the above mentioned reference methods are limited to produce point cloud outputs with a fixed size of 16,384 points. For this reason, apart from using the high density *fine GT* described in Section 4.1.1 for training our proposed MVPCC-Net, we also generated a downsampled version of each object's ground truth point cloud using the Farthest Point Sampling technique [46]. The downsampled point clouds - referred henceforward as *coarse GT* samples – consist of exactly 16,384 points, thus they can be used for training the reference approaches.

Second, since the reference methods exclusively deal with geometry information, in these experiments we also limited our model's training and evaluation only considering the XYZ channels. For this reason, during this comparison, we ignored the RGB channels, and we used the perceptual loss and the style loss terms exceptionally for the XYZ channels (with a learning rate of 10^{-4}).

The qualities of the completed object shapes are characterized by the geometric evaluation parameters defined in Section 4.3. Comparative results are provided in Tables 1 and 2, using as quality measures the mean Chamfer Distance and the mean F1-score over the test set, respectively. Here we used the proposed MVPCC-Net model with four views ($V = 4$), which proved to be the most efficient settings in our ablation experiments, as detailed later in Section 6.1. To demonstrate that the relative performances of the considered techniques are fairly stable regardless of how detailed reference point clouds are used as ground truth, we calculated the geometric evaluation metrics for both the *fine GT* and *coarse GT* samples, which results are shown side by side in Tables 1 and 2. Given that the denser *fine GT* samples have more points than the *coarse GT* objects, it is evident that the CD error rates associated with the *fine GT* are generally lower (and the F1-scores are higher) than the results connected to the *coarse GT* regarding each method.

The superiority of our method's *coarse GT*-related results demonstrates its efficacy in recreating the global structure of the objects under study (i.e, size and shape of their main components). On the other hand, our MVPCC-Net approach is also superior at reconstructing local features and fine geometric structures appearing only in denser point clouds: This observation is supported by our efficient *fine GT*-related numerical rates, and also by comparative qualitative results shown in Fig. 3, which displays for various sample objects the input partial point clouds, the results of all considered techniques, and the *fine GT* as a reference.

5.2. Comparative evaluation on real MLS data

In the second phase of the experiments, we evaluate the performance of the proposed approach and the reference techniques on real MLS point cloud samples, presented in Section 4.1.2. Due to the lack of sufficient number of (complete) training samples among the available MLS object point clouds, we use here the MVPCC-Net with weight parameters trained on synthetic data in the previous test phase.

However, by replacing synthetic point cloud inputs with real MLS measurements we have to deal with a practical issue: While in synthetic datasets standardized objects alignments can be ensured (e.g. the forward direction of vehicles is equal to one of the axes in their local Descartes coordinate system), object fragments extracted from real MLS data may have arbitrary orientations. For this reason, we proposed an additional network component for re-aligning the input MLS point cloud segments, enabling the direct application of our model for real measurements without the need for any additional fine-tuning, although the model had been trained on purely synthetic data. For this purpose, we adopted a spatial transformer network (STN) [47], which provides as output a 3×3 rotation matrix, that can be used to rotate an input point cloud sample around its vertical axis, so that it is

Table 1
Evaluation of our algorithm's geometric accuracy compared to the state-of-the-art algorithms on the synthetic dataset using **Chamfer Distance** ($\times 10^{-3}$)↓. By each object category, the first column refers to the comparison with the coarse GT (16384 points), while the second column represents the comparison results to the fine GT, the best results are highlighted in bold.

Method	Buses 1440 samples		Cars 4680 samples		Motorcycles 180 samples		Trains 480 samples		Overall 6780 samples	
	Coarse GT	Fine GT	Coarse GT	Fine GT	Coarse GT	Fine GT	Coarse GT	Fine GT	Coarse GT	Fine GT
TopNet [14]	7.254	6.514	10.724	8.189	19.362	18.976	10.264	9.919	10.184	8.242
GRNeT [7]	7.776	7.153	8.596	7.807	9.077	8.675	7.149	7.544	8.332	7.672
PCN [8]	5.870	5.130	7.549	6.107	13.326	12.869	7.257	6.823	7.325	6.129
VPC-Net [15]	**4.688**	**3.965**	6.666	5.604	9.945	9.531	5.935	5.512	6.281	5.353
SeedFormer [21]	6.592	5.922	7.609	6.685	8.524	8.074	6.737	6.243	7.355	6.528
SnowflakeNet [22]	4.94	4.202	6.101	5.274	**8.232**	**7.883**	**5.516**	**5.102**	5.869	5.103
Ours	5.113	4.333	**5.813**	**4.702**	8.448	7.928	5.853	5.309	**5.737**	**4.752**

transformed into a canonical orientation defined by ground truth samples in the training phase. The STN network segment was trained independently of the other components of the MVPCC-Net with synthetic vehicle shape models (described in Section 4.1.1). The input of this training phase consists of partial object point clouds rotated with randomly chosen angles around their vertical axes, and its reference outputs are the same point cloud segments with standard orientations, facing in the direction of the x-axes of their local coordinate system. The aim of this training phase is to learn the transform providing an appropriate rotation matrix for new object samples with arbitrary initial orientations.

After orientation adjustment, we tested the proposed MVPCC-Net and the reference techniques on real MLS object samples, so that neither our model nor other models were fine-tuned for the MLS measurements. As mentioned in Section 4.1.2, we have 54 completely scanned vehicle models in our MLS dataset which can be used for quantitative evaluation in a similar manner to the synthetic data, while the remaining 370 MLS objects enable us to perform a qualitative study on a widely diverse set of real vehicle point cloud measurements. Numerical and qualitative evaluation results are shown in Table 3 and in Fig. 4 respectively, which confirm that proposed MVPCC-Net outperforms the reference methods, and it is capable of efficiently processing MLS measurements. As demonstrated in Table 3, our model is clearly better than the other techniques regarding the Chamfer Distance (2nd column of the table), however, the F-scores of VPC-Net and MVPCC-Net are nearly identical with the standard distance threshold settings $\tau = 0.01$ (see the 3rd column). For this reason, we also calculated the F-score values with a more strict threshold selection $\tau = 0.005$, which choice yielded already a clear advantage of the proposed method (4th column). We can also conclude based on Fig. 4 that our method is more capable of producing dense and finely detailed point clouds which property is highly advantageous by processing dense MLS data.

Although in Figs. 3 and 4 we only visualized the geometry of the generated object point clouds, the MVPCC-Net method can also estimate RGB color value for each point as described in Section 3. For selected MLS objects, the input–output pairs of the proposed model are shown as colored point clouds in Fig. 5. These qualitative results confirm, that in many different situations, both the vehicle's global shape and its color schema can be predicted in a realistic manner, and the generated object segments fit well with the captured partial MLS measurements both in geometry and in color. Exploiting that vehicles have in general symmetric shapes, many components such as tail lamps or door textures are efficiently transformed from one side to the other one. On the other hand, the proposed method is also successful in predicting completely missing frontal or back regions, where symmetry-based shape completion cannot be performed. As a limitation, some erroneously textured areas may appear in different vehicle regions, which phenomenon is in part a consequence of texture errors in the raw MLS measurements.

5.3. Completion results for asymmetric objects

While the objects investigated in the previous sections have symmetric geometry, the proposed method is not limited to dealing with such object shapes. Since asymmetrical objects are usually considered more challenging for shape completion methods, we also investigated how our MVPCC-Net is able to complete partial point clouds of sofa objects. The sofa dataset was generated from ShapeNet in a similar manner as described in Section 4.1.1 for vehicles. For this experiment, we used 2930 training objects and 43 test samples for validation.

The results of shape completion for three sample sofa objects are displayed in Fig. 6, where each row shows the same input point cloud from two different perspectives, alongside the predicted object shapes. The results confirm at a proof-of-concept level that our model can also generate realistic results for asymmetrically shaped items.

5.4. Computational time

In this section, we present an experimental study about the execution time of each individual step of the proposed algorithm. Our experiments were performed on a personal computer (PC) with AMD Ryzen 9 5900X 12-Core Processor, 32-GB RAM and a NVIDIA GeForce RTX 3060 Ti GPU. We run the proposed MVPCC-Net model with four views ($V = 4$) on 216 real MLS samples described in Section 4.1.2. The

Table 2
Evaluation of our algorithm's geometric accuracy compared to the state-of-the-art algorithms on the synthetic testing dataset, **F1-score** (%)↑, The first number is a comparison with the coarse GT (16384 points), while the second number represents a comparison with the fine GT, the best results are highlighted in bold.

Method	Buses 1440 samples		Cars 4680 samples		Motorcycles 180 samples		Trains 480 samples		Overall 6780 samples	
	Coarse GT	Fine GT	Coarse GT	Fine GT	Coarse GT	Fine GT	Coarse GT	Fine GT	Coarse GT	Fine GT
TopNet [14]	82.06	82.85	69.56	73.31	24.85	25.95	64.70	64.91	70.68	73.48
GRNeT [7]	76.73	78.42	72.25	75.37	69.20	70.75	77.68	78.70	73.50	76.13
PCN [8]	89.74	90.96	83.79	86.99	49.35	51.17	80.49	81.29	83.91	86.47
VPC-Net [15]	**95.50**	**96.22**	87.63	89.57	66.89	68.23	87.56	88.12	88.75	90.31
SeedFormer [21]	85.62	86.80	81.03	84.28	75.22	76.86	84.71	85.59	82.11	84.71
SnowflakeNet [22]	94.01	94.83	88.34	90.23	74.85	75.89	**89.79**	**90.12**	89.28	90.77
Ours	92.64	93.70	**89.61**	**91.89**	**76.46**	**77.58**	88.22	88.99	**89.81**	**91.68**

Table 3
Evaluation of our algorithm's geometric accuracy compared to the state-of-the-art algorithms on the real MLS object samples, using **Chamfer Distance (CD,**$\times 10^{-3}$**)**↓, and **F1-score** (%)↑.

Method	↓**CD** ($\times 10^{-3}$)	↑**F1-score** (%) $\tau = 0.01$	↑**F1-score** (%) $\tau = 0.005$
TopNet [14]	12.725	52.87	14.10
GRNet [7]	10.185	67.73	31.47
PCN [8]	10.410	67.25	31.72
VPC-Net [15]	7.563	**80.77**	51.62
SeedFormer [21]	8.248	77.61	36.11
SnowflakeNet [22]	8.487	79,23	46.28
Ours	**6.962**	80.75	**58.28**

completion model part was executed on the GPU, while the remaining steps have been implemented for CPU. Table 4 presents the measured average execution time per object in milliseconds (ms), for the consecutive steps of our proposed shape prediction workflow. As the results show, with our method the mean total computing time for a sample object was around 55 ms. As for the reference techniques, we tested two methods [21,22] on the same PC configuration as our model, and their execution time varied between 28–48 ms for the different MLS object samples. Note that as shown in Table 4, almost half the processing efforts in our model correspond to the statistical outlier filter [39], which step can be significantly accelerated further by using a GPU-based implementation [48]. The remaining considered reference methods were tested on slightly different hardware platforms, nevertheless the experienced running times were largely similar to our model. We should also emphasize, that in the targeted MLS data processing application, real-time operation is usually not a strict requirement, thus we regard the speed of our algorithm adequate for applicability.

6. Ablation studies

The proposed MVPCC-Net model consists of various components and it contains a number of hyperparameters which influence its performance. To support our decisions regarding model design, and provide more information about parameter settings, we present ablation studies in this section, wherein each experiment we train our model with the car shape training dataset until the convergence (using a learning rate of 10^{-4}), and we validate the model performance on the corresponding test set.

6.1. Optimal settings of the number of views

In this section, we examine further the performance of the proposed method on colored incomplete point cloud inputs, and study the impact of changing the number of views with respect to the geometry and coloring of the completed point cloud output.

Fig. 7 displays the results per view as images for a sample vehicle object, using a total of $V = 10$ views in the proposed model. Each row displays the input, our model's output, and the ground truth for the corresponding view. ten views, including the inputs, our model's outputs, and the GTs of the color and geometry images side-by-side. As shown in the first three columns of Fig. 7, the results depict realistic

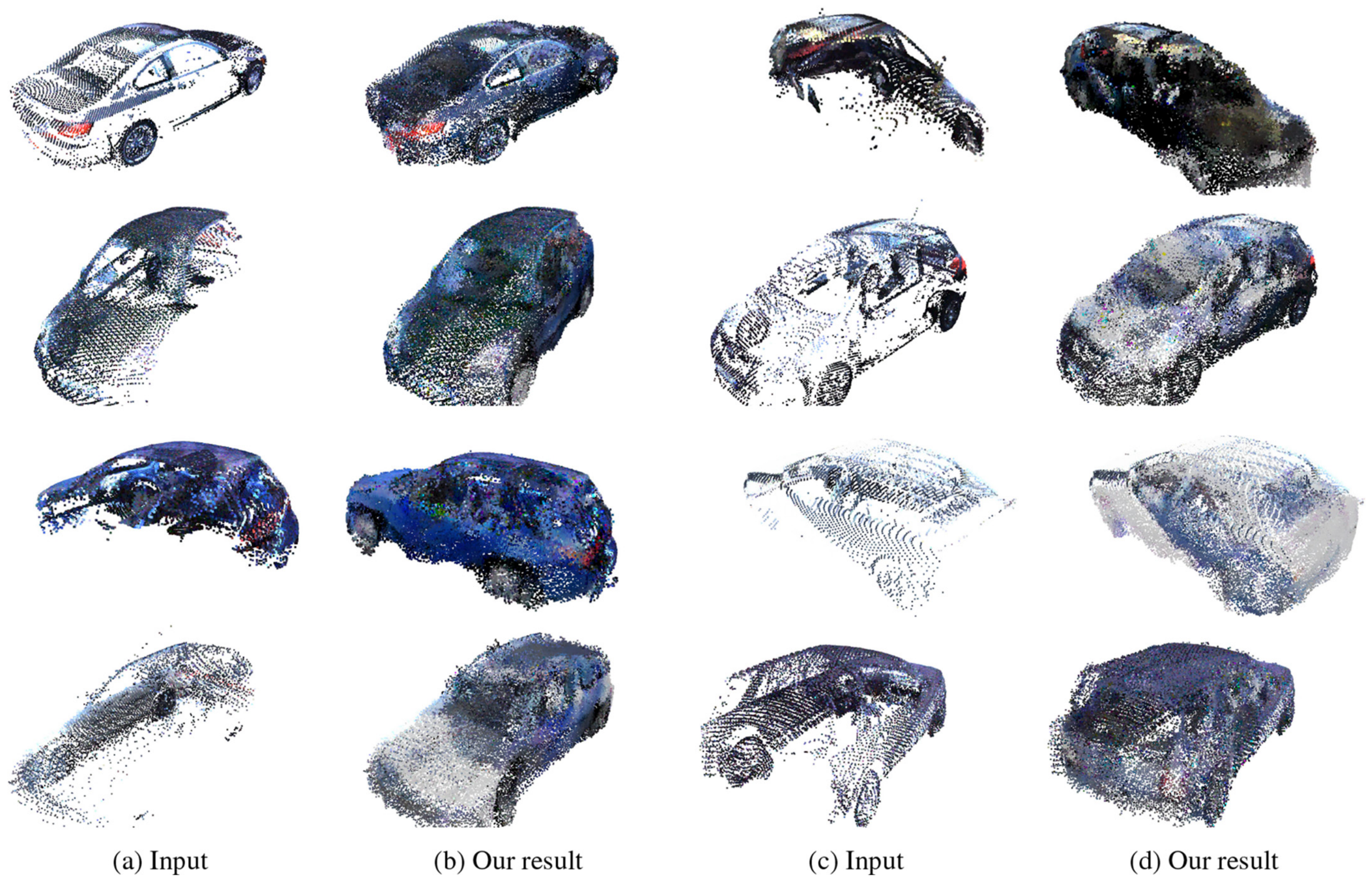

Fig. 5. Results of the proposed method with MLS point clouds acquired using a Riegl VMX Mobile Laser Scanner.

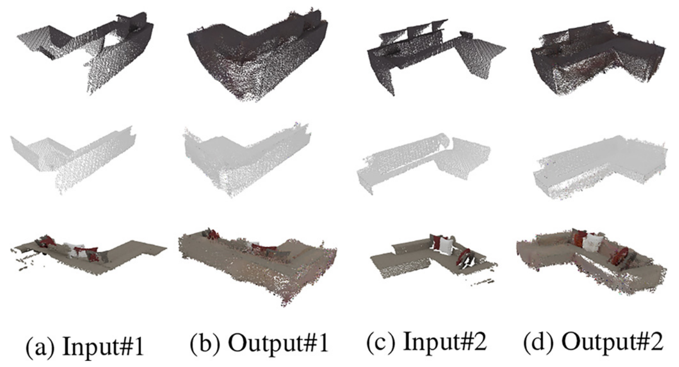

Fig. 6. Results of the proposed approach on the Shapenet dataset's asymmetrical sofa objects. For the same item, each row shows the input and our method's output from two viewing angles.

color predictions, with color characteristics matching on both sides of the vehicle (see rows 9 and 10), while realistic tail lamps can be observed on the vehicle's rear end (see in rows 1, 2, and 5), and the texturing style of the wheels also looks real (see rows 1–4). The color images of the last three columns of Fig. 7 visualize the shape geometry, so that the normalized XYZ coordinate values stored in the geometry channels are displayed as *pseudo geometry images*. These results demonstrate that the model is able to accurately predict the shape of the car, and even details such as predicted wheels and mirrors are visible in the output.

In the next phase of the analysis, we trained and tested the proposed model with increasing the number of views one by one from three to ten. The views are chosen in the order indicated by the consecutive rows in Fig. 7, for example, the four-view configuration uses the images shown in the first four rows.

Table 5 presents quantitative results of our model with different numbers of views for the whole car shape test dataset, using various metrics for geometry and color evaluation defined in Section 4.3. These experiments confirm that we can obtain the most accurate 3D geometry prediction by using in total four different viewpoints (following the settings of the first four views in Fig. 7). While additional views yield a higher point density output point cloud, they also cause additional noisy points surrounding the object shape, reducing the accuracy of shape estimation. This phenomenon is shown in Table 5, where the Chamfer Distance increases (and the F-score decreases) in cases of more than four viewpoints.

We can observe similar tendencies regarding color estimation based on Table 5 and Fig. 8: Using more than four views yields weaker performance rates, and we can also see noisier texture by six or eight views via visual verification (Fig. 8(d), (e)). Note that although according to Table 5 the best color-based evaluation rates are obtained by three views, such a configuration still generates incomplete object shapes (Fig. 8(b)), leading to larger geometric error values (see CD and F1-score rates in Table 5).

6.2. View fusion strategies

In this section, we compare the view fusion strategy introduced in Section 3.2 to a straightforward early fusion technique used as baseline model, where the I_{in}^{v} images of all v views are concatenated into a single

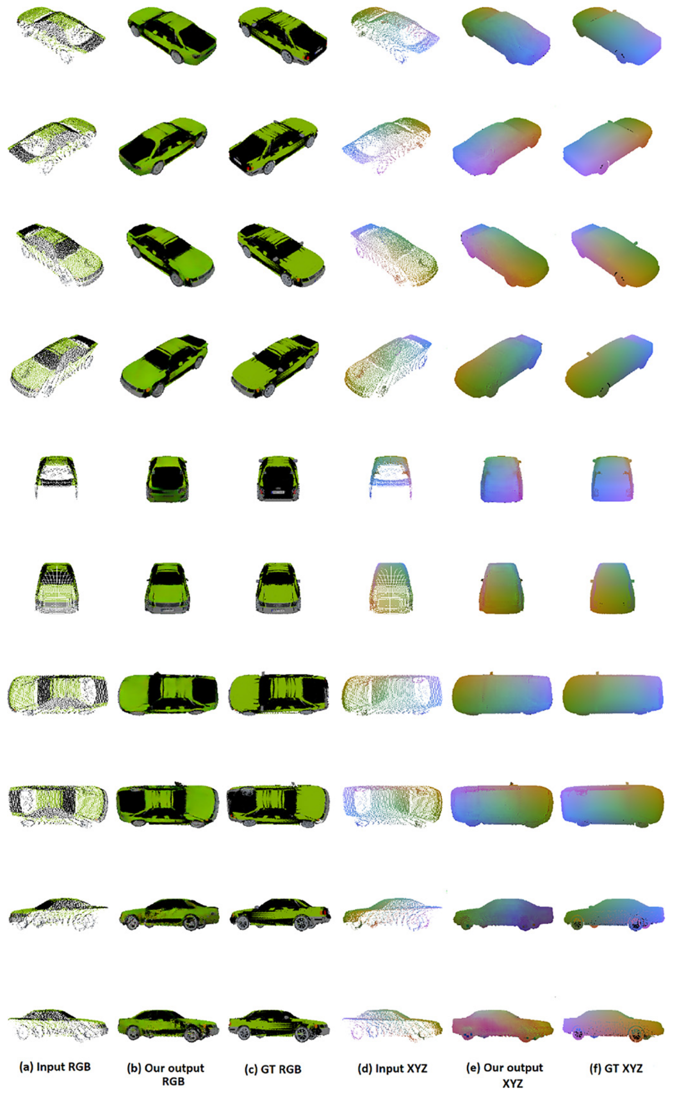

Fig. 7. Result of the XYZRGB channels from ten views for a partial input point cloud. For each row: (a) RGB input, (b) RGB output, (c) RGB GT, (d) XYZ input, (e) XYZ output, (f) XYZ GT.

input data cube for the encoder, while the decoder generates all output views in one step. On the contrary, our method implements a late fusion approach, which generates first a separate feature representation f_v for each view v by a shared encoder, thereafter a *Feature Fusion* network component generates a global feature F from the view-level features which is used by a shared decoder to produce the inpainted images per view in a sequential process. The block diagrams of the above defined early and late fusion approaches are shown in Fig. 9.

Next, we conducted experiments for comparing the efficiency of the early fusion to the late fusion strategies in the MVPCC-Net model. As input, we considered on the one hand measurements with color and geometry channels (XYZRGB), and on the other hand, pure shape data containing geometry channels only (XYZ). The obtained results

Table 4
Execution Time for each step of our algorithm in milliseconds (ms). The time is measured on an NVIDIA GeForce RTX 3060 Ti GPU with batch size of 1.

NO. views	Projection	Completion model			Re-projection	Filter	Overall
		Encoder	Fusion	Decoder			
4 views	6.311	10.493	0.132	4.514	7.933	25.621	55.004

Table 5
Effects of view aggregation. Results on a test set of synthetic data (car shape), **PSNR**↑, **SSIM**↑, **MAE**↓ on color channels, **Chamfer Distance** (10^{-3})↓, **F1-score** (%)↑ on geometric accuracy.

No. views	**2D Color Images**			**3D Geometry**	
	PSNR↑	**SSIM**↑	**MAE**↓	**CD** $(\times 10^{-3})$↓	**F1-score** (%)↑
3 views	25.03	0.9013	0.0860	6.937	81.31
4 views	24.95	0.8969	0.0892	**6.585**	**83.07**
5 views	22.67	0.7803	0.1768	7.45	80.67
6 views	21.44	0.7063	0.2340	8.28	76.03
7 views	20.11	0.6367	0.2768	11.214	65.08
8 views	19.42	0.5908	0.2968	9.19	73.16
10 views	18.40	0.5342	0.3689	13.697	63.54

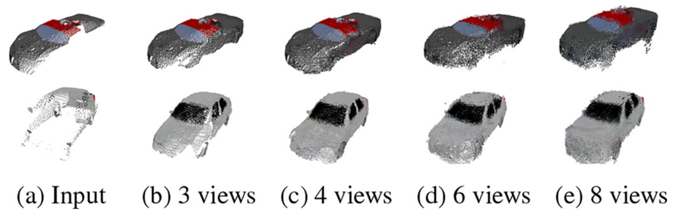

Fig. 8. Effects of view aggregation. Results of the proposed method with different number of views: (a) Input, our model results using three, four, six, and eight views are shown in (b)-(e) respectively.

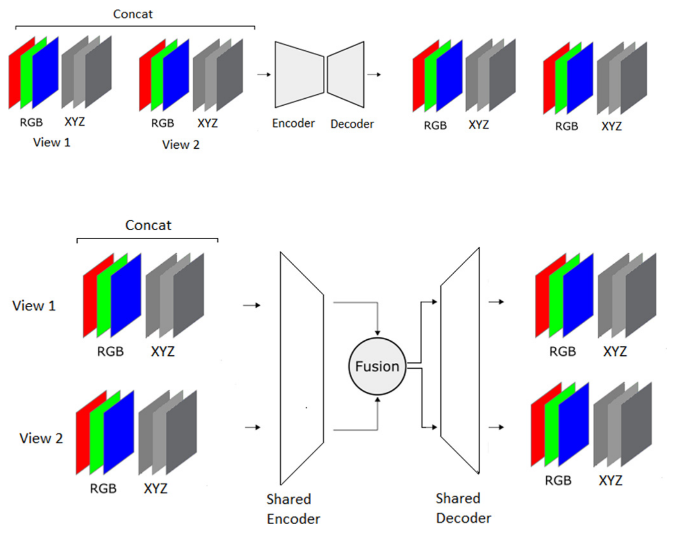

Fig. 9. Types of fusions, first represents the early fusion method, the second represents the late fusion method.

Table 6
Effect of various fusion strategies, Results on a testing set of synthetic data (car shape), **Chamfer Distance** $(\times 10^{-3})$↓, **F1-score** (%)↑ on geometric accuracy.

Fusion method	**CD** $(\times 10^{-3})$↓	**F1-score** (%)↑
Early fusion (XYZ)	5.735	87.33
Early fusion (XYZRGB)	11.792	69.29
Late fusion (XYZ)	**5.315**	**89.04**
Late fusion (XYZRGB)	6.585	83.07

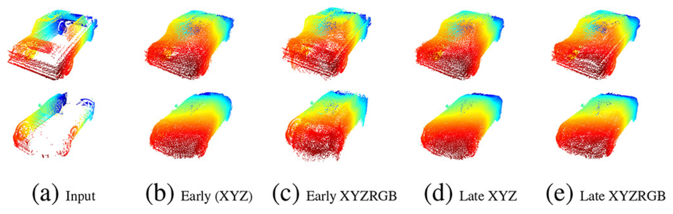

Fig. 10. Effects of Fusion strategies. Results of the proposed method with different fusion strategies: (a) Input, our model results using (b) early fusion method on geometry channels, (c) early fusion method on color and geometry channels, (d) late fusion method on geometry channels, (e) late fusion method on color and geometry channels.

regarding 3D geometric accuracy over the car shape set are presented in Table 6, while Fig. 10 shows the results of the different fusion strategies for two sample objects, where we present the point cloud results without displaying RGB color for enabling better visual comparison of the object geometries. The quantitative and qualitative results confirm that the proposed late fusion approach outperforms the early fusion baseline technique for both types of input data (XYZ and XYZRGB). Regarding the geometric parameters, the models purely considering XYZ channels yield in general better results, which fact indicates that generating colored point clouds comes with some degradation of the shape geometry. On the other hand, adding RGB information to the early fusion approach yields a significantly larger reduction of the geometric accuracy, than by using the late fusion technique, where we can only observe a slight negative effect on the accuracy of the predicted object shape.

6.3. Re-projection filter parameters

This section demonstrates the significance of applying the erosion and outlier filters as post processing steps of the view re-projection phase (Section 3.3), and we also justify here the filters' parameter selection strategy. In the following experiment, we test the proposed MVPCC-Net model with four views, and purely geometric (XYZ) input data, so that we calculate the geometric CD and F1-score values for the output point clouds provided by the re-projection step with various filter parameters.

We start with the analysis of using the erosion operator, where we evaluate the results in four configurations: first without using the erosion step, thereafter with implementing the erosion with square shaped kernels of sizes $1 \times 1, 3 \times 3$, and 5×5, respectively. Based on the results presented in Table 7 and Fig. 11, we can conclude that by applying the erosion operator we can get a smoothed output point cloud, where several noisy points around the investigated object's shape are removed. Since by increasing the kernel size to 5×5, we can often observe the removal of some real object components, we decided to use the 3×3 kernel, which choice also corresponds to the best evaluation rates in Table 7.

Next, we demonstrate the significance of the Statistical Outlier Filter (SOF). Fig. 12(b) shows the results without SOF, while Fig. 12(c)-(e) display the output point clouds using the statistical outlier filter with its

Table 7
Analysis of the effect of using erosion operation as a post-processing step with different kernel sizes.

Kernel size	**CD** $(\times 10^{-3})$↓	**F1-score** (%)↑
Not used	5.583	88.69
(1×1)	5.335	88.89
(3×3)	**5.315**	**89.04**
(5×5)	5.371	88.84

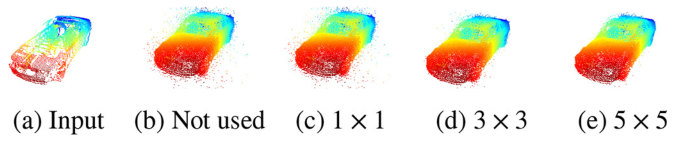

Fig. 11. Analysis of the effect of using the erosion operation in post-processing: (a) input point cloud, (b) result without using the erosion operation, (c)-(e) results using erosion operation with kernel sizes $1 \times 1, 3 \times 3$ and 5×5, respectively.

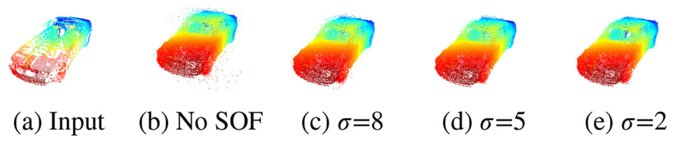

Fig. 12. Analysis of the effect of using Statistical Outlier Filter (SOF) as a post-processing step: (a) input point cloud, (b) results without using SOF, (c)-(e) results using SOF with standard deviation rates $\sigma = 8, 5$ and 2, respectively.

Table 8
Analysis of the effect of using Statistical Outlier Filter (SOF) as a post-processing step with different standard deviation (σ) parameters.

Settings	**CD** ($\times 10^{-3}$)↓	**F1-score** (%)↑
No SOF	7.076	86.86
$\sigma = 8$	**5.311**	88.99
$\sigma = 5$	5.315	89.04
$\sigma = 2$	5.349	**89.05**

standard deviation parameter σ equal to 8, 5, and 2, respectively. We can see that the filter can remove several noisy points around the object boundary, while the number of the filtered points increases by decreasing the standard deviation rate.

Applying the statistical outlier filter as the last step of our algorithm has a significant impact on the geometric accuracy of the final results: according to Table 8, both the CD ad F1-score rates can be improved by around 2%. We can also observe that using SOF with a standard deviation of $\sigma = 8$ decreases the CD by removing a large number of noisy points, whereas selecting a $\sigma = 5$ improves the F1-score while it maintains the CD nearly unchanged compared to the case of $\sigma = 8$. Using a standard deviation of $\sigma = 2$ we experience a minor rise in the F1-score at the expense of a significant decrease in CD. Based on the above experiments, in our final model we applied the statistical outlier filter with a standard deviation of $\sigma = 5$ in order to obtain a reasonable balance between the CD score and the F1-score evaluation rates.

7. Conclusion

In this paper, we proposed a novel method for completing colored 3D point clouds representing various incomplete object shapes. We focus on generating models of street objects based on Mobile Laser Scanning measurements, where a key requirement is to keep the high detailness of the input data. Our method generates first multiple projections of the incomplete input point cloud by virtual cameras positioned around the object of interest. Thereafter the sparsely filled multichannel view images are completed in the 2D domain, and they are reassembled in the 3D space, resulting in dense point clouds of the whole object. It has been demonstrated by quantitative and qualitative experiments both on synthetic and on real-world MLS data that the new method is applicable and it outperforms various state-of-the-art techniques in terms of geometric shape accuracy, realistic RGB coloring, and preserving high resolution.

CRediT authorship contribution statement

Yahya Ibrahim: Conceptualization, Methodology, Software, Validation, Writing-original-draft. **Csaba Benedek:** Writing-review-editing, Supervision.

Data availability

I have shared the link to my data/code. https://github.com/sztaki-geocomp/Multi-view-3d-completion

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

The research work was supported by the European Union within the framework of the National Laboratory for Autonomous Systems (RRF-2.3.1–21-2022–00002) and the Artificial Intelligence National Laboratory (MILAB) programs, and by the TKP2021-NVA-01, and OTKA K-143274 projects of the Hungarian NRDI Office.

References

[1] D. Bolkas, J. Chiampi, J. Chapman, V. Pavill, Creating a virtual reality environment with a fusion of sUAS and TLS point-clouds, Int. J. Image Data Fusion 11 (2020) 1–26, https://doi.org/10.1080/19479832.2020.1716861.
[2] H. Durrant-Whyte, T. Bailey, Simultaneous localization and mapping: part I, IEEE Robot. Autom. Mag. 13 (2) (2006) 99–110, https://doi.org/10.1109/MRA.2006.1638022.
[3] F. Hariz, H. Souifi, R. Leblanc, Y. Bouslimani, M. Ghribi, E. Langin, D. Mccarthy, Direct Georeferencing 3D Points Cloud Map Based on SLAM and Robot Operating System, in: IEEE International Symposium on Robotic and Sensors Environments (ROSE), 2021, pp. 1–6. doi:10.1109/ROSE52750.2021.9611774.
[4] B. Nagy, C. Benedek, 3D CNN-Based Semantic Labeling Approach for Mobile Laser Scanning Data, IEEE Sens. J. 19 (21) (2019) 10034–10045, https://doi.org/10.1109/JSEN.2019.2927269.
[5] R.Q. Charles, H. Su, M. Kaichun, L.J. Guibas, PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 77–85. doi:10.1109/CVPR.2017.16.
[6] A. Dai, C.R. Qi, M. Nießner, Shape Completion using 3D-Encoder-Predictor CNNs and Shape Synthesis, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2017.
[7] H. Xie, H. Yao, S. Zhou, J. Mao, S. Zhang, W. Sun, GRNet: Gridding Residual Network for Dense Point Cloud Completion, European Conference on Computer Vision (ECCV), Springer International Publishing 2020, pp. 365–381.
[8] W. Yuan, T. Khot, D. Held, C. Mertz, M. Hebert, PCN: Point Completion Network, in: International Conference on 3D Vision (3DV), 2018, pp. 728–737.
[9] Y. Ibrahim, B. Nagy, C. Benedek, Multi-view Based 3D Point Cloud Completion Algorithm for Vehicles, in: International Conference on Pattern Recognition (ICPR), 2022, pp. 2121–2127. doi:10.1109/ICPR56361.2022.9956459.
[10] A.X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, F. Yu, ShapeNet: An Information-Rich 3D Model Repository, Tech. Rep. arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015.
[11] C.R. Qi, L. Yi, H. Su, L.J. Guibas, PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space, in: Advances in Neural Information Processing Systems, vol. 30, 2017.
[12] A. Tagliasacchi, M. Olson, H. Zhang, G. Hamarneh, D. Cohen-Or, VASE: Volume-Aware Surface Evolution for Surface Reconstruction from Incomplete Point Clouds, Comput. Graph. Forum (2011), https://doi.org/10.1111/j.1467-8659.2011.02030.x.
[13] J. Rock, T. Gupta, J. Thorsen, J. Gwak, D. Shin, D. Hoiem, Completing 3D object shape from one depth image, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 2484–2493. doi:10.1109/CVPR.2015.7298863.
[14] L.P. Tchapmi, V. Kosaraju, H. Rezatofighi, I. Reid, S. Savarese, TopNet: Structural Point Cloud Decoder, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 383–392. doi:10.1109/CVPR.2019.00047.
[15] Y. Xia, Y. Xu, C. Wang, U. Stilla, VPC-Net: Completion of 3D vehicles from MLS point clouds, ISPRS J. Photogramm. Remote Sens. 174 (2021) 166–181, https://doi.org/10.1016/j.isprsjprs.2021.01.027.
[16] Y. Yang, C. Feng, Y. Shen, D. Tian, FoldingNet: Point Cloud Auto-Encoder via Deep Grid Deformation, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 206–215. doi:10.1109/CVPR.2018.00029.

[17] Y. Wang, D.J. Tan, N. Navab, F. Tombari, SoftPoolNet: Shape Descriptor for Point Cloud Completion and Classification, European Conference on Computer Vision (ECCV), Springer International Publishing 2020, pp. 70–85.

[18] X. Yu, Y. Rao, Z. Wang, Z. Liu, J. Lu, J. Zhou, Pointr: Diverse point cloud completion with geometry-aware transformers, in: IEEE/CVF International Conference on Computer Vision (ICCV), 2021.

[19] X. Yu, Y. Rao, Z. Wang, J. Lu, J. Zhou, Adapointr: Diverse point cloud completion with adaptive geometry-aware transformers (2023). doi:10.48550/ARXIV.2301.04545. URL: https://arxiv.org/abs/2301.04545.

[20] J. Tang, Z. Gong, R. Yi, Y. Xie, L. Ma, Lake-net: Topology-aware point cloud completion by localizing aligned keypoints, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2022, pp. 1726–1735.

[21] H. Zhou, Y. Cao, W. Chu, J. Zhu, T. Lu, Y. Tai, C. Wang, Seedformer: Patch seeds based point cloud completion with upsample transformer, European Conference on Computer Vision (ECCV), Springer-Verlag, Berlin, Heidelberg 2022, pp. 416–432, https://doi.org/10.1007/978-3-031-20062-5_24.

[22] P. Xiang, X. Wen, Y.-S. Liu, Y.-P. Cao, P. Wan, W. Zheng, Z. Han, Snowflake point deconvolution for point cloud completion and generation with skip-transformer, IEEE Trans. Pattern Anal. Mach. Intell. (2022) 1–18, https://doi.org/10.1109/TPAMI.2022.3217161.

[23] H. Su, S. Maji, E. Kalogerakis, E. Learned-Miller, Multi-view Convolutional Neural Networks for 3D Shape Recognition, in: IEEE/CVF International Conference on Computer Vision (ICCV), 2015, pp. 945–953. doi:10.1109/ICCV.2015.114.

[24] J. Zhang, D. Zhou, Y. Zhao, W. Nie, Y. Su, MV-LFN: Multi-view based local information fusion network for 3D shape recognition, Vis. Inform. 5 (3) (2021) 114–119, https://doi.org/10.1016/j.visinf.2021.09.003.

[25] F.J. Lawin, M. Danelljan, P. Tosteberg, G. Bhat, F.S. Khan, M. Felsberg, Deep Projective 3D Semantic Segmentation, Computer Analysis of Images and Patterns, Springer International Publishing 2017, pp. 95–107.

[26] A. Boulch, B.L. Saux, N. Audebert, Unstructured point cloud semantic labeling using deep segmentation networks, in: Eurographics Workshop on 3D Object Retrieval, vol. 2, 2017.

[27] M. Jaritz, J. Gu, H. Su, Multi-View PointNet for 3D Scene Understanding, in: IEEE/CVF International Conference on Computer Vision Workshop (ICCVW), 2019, pp. 3995–4003. doi:10.1109/ICCVW.2019.00494.

[28] C.B. Choy, D. Xu, J. Gwak, K. Chen, S. Savarese, 3D–R2N2: A Unified Approach for Single and Multi-view 3D Object Reconstruction, European Conference on Computer Vision (ECCV), Springer International Publishing 2016, pp. 628–644.

[29] H. Fan, H. Su, L. Guibas, A Point Set Generation Network for 3D Object Reconstruction from a Single Image, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Los Alamitos, CA, USA, 2017, pp. 2463–2471. doi:10.1109/CVPR.2017.264.

[30] C.-H. Lin, C. Kong, S. Lucey, Learning Efficient Point Cloud Generation for Dense 3D Object Reconstruction, in: AAAI Conference on Artificial Intelligence (AAAI), 2018.

[31] X. Zhang, Y. Feng, S. Li, C. Zou, H. Wan, X. Zhao, Y. Guo, Y. Gao, View-Guided Point Cloud Completion, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2021, pp. 15890–15899.

[32] Z. Zhu, L. Nan, H. Xie, H. Chen, M. Wei, J. Wang, J. Qin, CSDN: Cross-modal Shape-transfer Dual-refinement Network for Point Cloud Completion (2022). doi: 10.48550/ARXIV.2208.00751.

[33] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative Adversarial Nets, in: Advances in Neural Information Processing Systems 27, 2014, pp. 2672–2680.

[34] J. Zhu, T. Park, P. Isola, A.A. Efros, Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks, in: IEEE/CVF International Conference on Computer Vision (ICCV), 2017, pp. 2242–2251.

[35] J. Johnson, A. Alahi, L. Fei-Fei, Perceptual losses for real-time style transfer and super-resolution, European Conference on Computer Vision (ECCV), Springer International Publishing, Amsterdam, Netherlands, 2016.

[36] L.A. Gatys, A.S. Ecker, M. Bethge, Image Style Transfer Using Convolutional Neural Networks, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 2414–2423.

[37] K. Vanjigounder, K. Narayanankutty, S. Veni, Performance Comparison of Total Variation based Image Regularization Algorithms, Int. J. Adv. Sci. Eng. Inf. Technol. 6 (4) (2016) 419–425, https://doi.org/10.18517/ijaseit.6.4.850.

[38] Z. Liu, J. Cheng, Q. Wang, L. Xian, Improved Design Based on IoU Loss Functions for Bounding Box Regression, in: IEEE Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), 2022, pp. 452–458. doi:10.1109/IAEAC54830.2022.9929938.

[39] Q.-Y. Zhou, J. Park, V. Koltun, Open3D: A modern library for 3D data processing, arXiv:1801.09847 (2018).

[40] Y. Nie, Y. Lin, X. Han, S. Guo, J. Chang, S. Cui, J. Zhang, Skeleton-bridged Point Completion: From Global Inference to Local Adjustment, in: Advances in Neural Information Processing Systems, vol. 33, 2020, pp. 16119–16130.

[41] D.P. Kingma, J. Ba, Adam: A Method for Stochastic Optimization, in: International Conference on Learning Representations (ICLR), 2014.

[42] K. Nazeri, E. Ng, T. Joseph, F. Qureshi, M. Ebrahimi, EdgeConnect: Generative Image Inpainting with Adversarial Edge Learning, in: IEEE/CVF International Conference on Computer Vision Workshop (ICCVW), 2019, pp. 3265–3274.

[43] Y. Ibrahim, B. Nagy, C. Benedek, A GAN-based Blind Inpainting Method for Masonry Wall Images, in: International Conference on Pattern Recognition (ICPR), 2021, pp. 3178–3185. doi:10.1109/ICPR48806.2021.9413009.

[44] G. Liu, F.A. Reda, K.J. Shih, T.-C. Wang, A. Tao, B. Catanzaro, Image Inpainting for Irregular Holes Using Partial Convolutions, European Conference on Computer Vision (ECCV), Springer International Publishing, 2018.

[45] Zhou Wang, A.C. Bovik, H.R. Sheikh, E.P. Simoncelli, Image quality assessment: from error visibility to structural similarity, IEEE Trans. Image Process. 13 (4) (2004) 600–612.

[46] C. Moenning, N.A. Dodgson, Fast Marching farthest point sampling, in: Eurographics - Posters, Eurographics Association, 2003. doi:10.2312/egp.20031024.

[47] M. Jaderberg, K. Simonyan, A. Zisserman, k. kavukcuoglu, Spatial Transformer Networks, in: Advances in Neural Information Processing Systems, vol. 28, 2015.

[48] N.A. Hadi, S.A. Halim, N. Alias, Statistical Filtering on 3D Cloud Data Points on the CPU-GPU Platform, J. Phys: Conf. Ser. 1770 (1) (2021), 012006, https://doi.org/10.1088/1742-6596/1770/1/012006.