ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume V-2-2022
XXIV ISPRS Congress (2022 edition), 6–11 June 2022, Nice, France

SELF-SUPERVISED ADVERSARIAL SHAPE COMPLETION

Torben Peters[a,*], Konrad Schindler[a] , Claus Brenner[b]

[a] Photogrammetry and Remote Sensing, ETH Zurich, 8093 Zürich, Switzerland,
(torben.peters,konrad.schindler)@geod.baug.ethz.ch
[b] Institute of Cartography and Geoinformatics, Leibniz University, 30167 Hannover, Germany,
claus.brenner@ikg.uni-hannover.de

Commission II,WG II/3

KEY WORDS: 3D Point Clouds, Adversarial Learning, Unsupervised Learning, Shape Completion

ABSTRACT:

The goal of this paper is 3D shape completion: given an incomplete instance of a known category, hallucinate a complete version of it that is geometrically plausible. We develop an adversarial framework that makes it possible to learn shape completion in a self-supervised fashion, only from incomplete examples. This is enabled by a discriminator network that rejects incomplete shapes, via a loss function that separately assesses local sub-regions of the generated example and accepts only regions with sufficiently high point count. This inductive bias against empty regions forces the generator to output complete shapes. We demonstrate the effectiveness of this approach on synthetic data from ShapeNet and ModelNet, and on a real mobile mapping dataset with nearly 9'000 incomplete cars. Moreover, we apply it to the KITTI autonomous driving dataset without retraining, to highlight its ability to generalise to different data characteristics.

1. INTRODUCTION

Incomplete 3D shape information is the norm, rather than the exception, when objects are observed in the wild: since visual sensors such as laser scanners and cameras are line-of-sight instruments, large parts of most objects remain unobserved due to occlusions, especially self-occlusions. For applications like localization or scan matching, occlusions can lead to poor results. It thus makes sense to try and predict the missing portion, using knowledge about the shape distribution of the objects of interest. Reconstructing complete 3D shapes from incomplete ones boils down to constructing a generative model of the target category, from which we can sample (complete) instances. Shape completion is achieved by conditioning the sampling on the available, incomplete observations. The question we ask in this paper is: *can one learn such a generative model in an self-supervised fashion, only from incomplete instances?* To do so, we design an adversarial training scheme with a built-in preference for complete shapes: a generator network is trained to synthesise object instances from incomplete inputs; the generator is guided by a discriminator network that accepts only instances which on the one hand fit the observed portions of the input, and on the other hand are complete.

Our main *contribution* is an adversarial architecture for 3D shape completion, in the spirit of a conditional GAN, with a novel loss function that makes it possible to train the model without any complete example instances. In our case, the geometry of an object is represented by a set of points or voxels on its surface. The underlying idea is that if we split the bounding box of the object into smaller sub-regions, then at least *some* of them will correspond to observed object parts and thus have high point density. If we sort the sub-regions by point density and divide them into a positive set of sufficiently dense ones and a negative set of overly sparse ones, we get access to a supervision signal. This allows us to train the discriminator such that it rejects both samples with unlikely shapes and samples with too few points on *any* of their parts. Taken together, these two criteria express a preference for shapes that are plausible (according to the consensus of many partial shapes observed during training) and complete. Additionally, we describe a pipeline to extract thousands of (incomplete) instances of cars from mobile mapping data and align them. This enables us to collect the training data for the shape completion network without human interaction. Due to the scanning geometry of the Mobile Mapping System (MMS), the cars in the scanned point cloud are typically 40-80% occluded. Furthermore, even the visible surfaces may be only sparsely sampled, depending on the type of scanner used.

We evaluate our approach first on a synthetic dataset by creating occluded samples for which the complete shapes are known. Then, we show that our approach works on a mobile mapping dataset with nearly 9'000 instances of cars. Additionally, we also run the method on KITTI data without retraining it, to show that it generalises to very different, much sparser scan data.

2. RELATED WORK

Generative adversarial networks (GANs) are a class of implicit generative models based on a game-theoretic scenario (Goodfellow et al., 2014). A GAN consists of a generator network that produces samples $x = G(z)$, and a discriminator network $y = D(x)$ that classifies samples as being real data or fakes produced by the generator. The two networks are trained jointly, so that the discriminator acts as a loss function for the generator, forcing it to produce samples that are indistinguishable from real data. Of particular interest for our work are conditional GANs (cGANs) (Mirza and Osindero, 2014), where the generator output is conditioned on some observation q, such that the samples $x = G(z, q)$ are not only realistic, but also compatible with q. Conditional GANs have become a popular tool to generate and analyse images, e.g. (Isola et al., 2017,

* Corresponding author

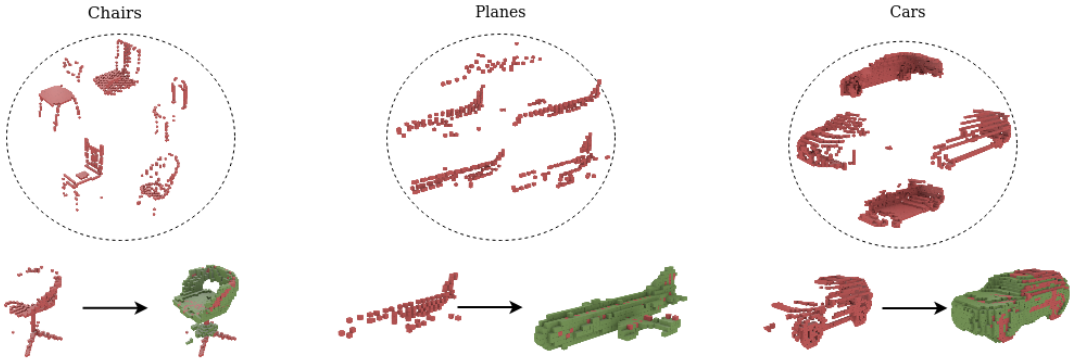


Figure 1. Our GAN learns to complete shapes from sparse observations (red voxels are observations, green voxels are predictions).

Dai et al., 2017b, Wang et al., 2018).

Shape completion has been a long-standing topic in computer vision and graphics (Pauly et al., 2005, Dai et al., 2017a). Classical approaches were often based on deforming a shape template to fit the data (Kraevoy and Sheffer, 2005) or fitting a generic deformable surface model (Zhao et al., 2001). For regular, man-made objects, many authors have also used symmetry to fill in missing parts by copying, extruding or mirroring existing ones (Mitra et al., 2006, Podolak et al., 2006, Pauly et al., 2008, Mitra et al., 2013, Sipiran et al., 2014, Sung et al., 2015). Shape completion is related to *inpainting*, where the goal is to predict missing pixel intensities, rather than geometric surfaces. Inpainting algorithms are often derived from signal processing considerations, but conceptually they also either fit a "geometric" prior of the intensity landscape, e.g., a piece-wise constant model (Getreuer, 2012); or they paste in content that is either copied from the image itself (Criminisi et al., 2004) or synthesised from a collection of training images (Yang et al., 2017).

More recently, *learning-based* approaches are favoured for shape completion. Most of them operate in a fully supervised setting, assuming that paired training data with incomplete and complete versions of the same shapes is available (Yuan et al., 2018, Firman et al., 2016, Han et al., 2017, Rezende et al., 2016, Riegler et al., 2017). Supervised learning has also been combined with symmetry, for the task of face inpainting (Zhang et al., 2018). While such strong supervision simplifies the task from a machine learning point of view, a big effort is needed to assemble a large enough training set. More importantly, the system will be fitted to the specific sensor characteristics and environment of the training data, and may be hard to adapt to other scenarios.

A few works relax the assumption and investigates *weakly supervised* settings where only a small part of the data is labelled (Chen et al., 2019b). Another possibility is transfer learning with large volumes of synthetic training data and a smaller amount of real data. E.g., (Stutz and Geiger, 2018) train a variational auto-encoder (Kingma and Welling, 2013) for shape completion with synthetic data, then fine-tune only the encoder with real data. An issue with synthetic training data is that it is not easy to simulate realistic scanning conditions, with, e.g., reflections, boundary effects, etc.

A recent trend is to exploit unpaired input and output shapes (Lu and Dubbelman, 2019, Chen et al., 2019a, Chen et al., 2020, Wu et al., 2020). Notably is the work of (Zhang et al., 2021), who use a GAN pre-trained on complete shapes and search the latent representation for a complete shape that best reconstructs a given incomplete input. Other notable unsupervised approaches are (Yang et al., 2018, Sharma et al., 2016), who created auto-encoders for 3D data.

3. METHOD

The basis of our method is a simple, statistical "big data" argument: if we collect a sufficiently large set of *incomplete* scans of an object class, then they will nevertheless capture the underlying shape distribution of *complete* objects, because the dataset will contain all the necessary local shape features, and the pairwise overlaps to align and assemble the "puzzle pieces".

Moreover, we also make the fairly weak assumption that we can regard a local chunk of the object shape as "complete" if it has a sufficiently high point density, respectively as "incomplete" if the density is too low. If we explicitly divide the (extended) bounding box into sub-regions of appropriate size, we can thus find sufficiently many complete and incomplete examples for every sub-region in the data. By accumulating that information during training, it should therefore be possible to learn a shape distribution for the object class that is complete, i.e., has a sufficiently high point density everywhere.

3.1 Sub-region based GAN model

Let us for a moment take the perspective of a vanilla GAN that should learn to synthesise complete 3D shapes, represented as binary voxel grids in which voxels on the object surface have label *surface* and all other voxels have label *background* (note that, despite using the voxel grid format, this is a surface representation, not a volumetric one). The generator takes as input an incomplete shape and shall learn to output a complete one. To train that model we would, however, require samples from the target distribution as supervision, i.e., complete shape that define the positive "real" class of the discriminator. In the self-supervised setting, we need a mechanism to teach the discriminator the difference between complete and incomplete samples *locally*, since positive examples of completeness are only available for local sub-regions of the volume.

Our solution is to tile the volume into a fixed set of $n = n_x \times n_y \times n_z$ sub-volumes which we call "blocks", see Fig. 2. The size of those blocks must be matched to the object type and the data characteristics, such that in some of them the surface is completely observed, i.e., populated with observations of reasonable density. Every input sample is first encoded into a PointNet-style (Qi et al., 2017b) feature vector and fed to the

This contribution has been peer-reviewed. The double-blind peer-review was conducted on the basis of the full paper.
https://doi.org/10.5194/isprs-annals-V-2-2022-143-2022 | © Author(s) 2022. CC BY 4.0 License.

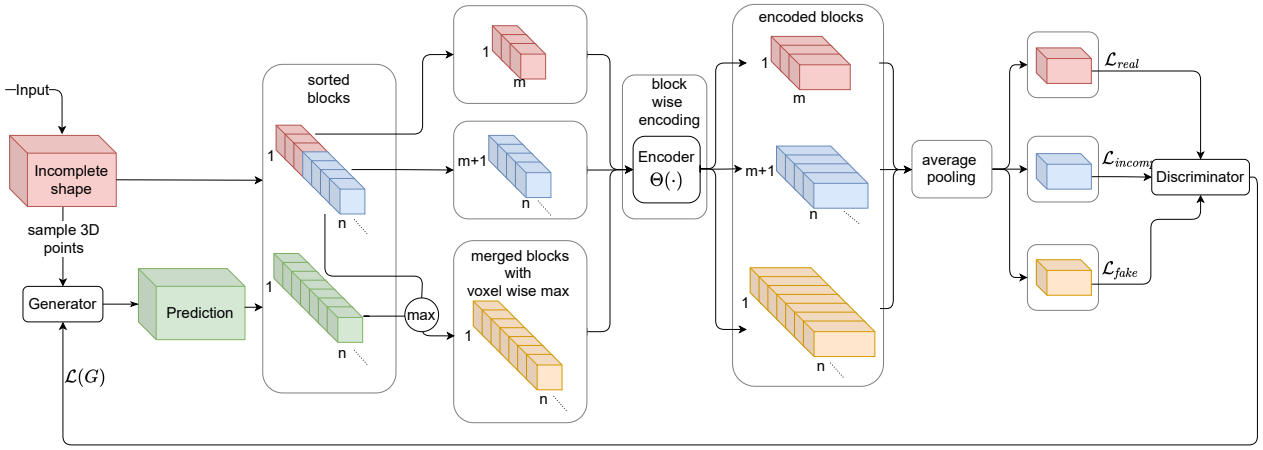


Figure 2. Methodology: Red/blue and green color indicate real and generated data, respectively. Orange blocks are merged voxel-wise between real and synthesized, indicated by the max operation. The Encoder $\Theta(\cdot)$ is shown in Figure 4. It encodes each block into a vector which are then average pooled to a fixed size representation and passed to the discriminator.

generator – this is done globally for the whole volume. The encoding step is important to ensure that the generator cannot copy the input and learns to re-synthesise the shape. After the generation step, both the input sample and the generator output are tiled into blocks, and the blocks are *sorted* by decreasing number *surface* voxels in the real, incomplete input. Among the blocks of the real input, only a fixed number m of most densely populated blocks are encoded into a global representation by a series of 3D convolutions, average pooled into a single feature vector, and passed to the discriminator as "real" examples, whereas the remaining, less densely populated blocks are encoded and pooled in the same way and passed on as "fake" examples. The hyper-parameter m is set conservatively, in our implementation to 25% of all blocks. Alternatively, one could set a threshold on the density – this would give the user better control over the output density, but needs to be tuned more carefully to the training data.

With the supervision provided so far, the discriminator can learn to ensure a sufficiently high point density, but it is neither biased against implausible distributions with appropriate point count, nor conditioned on the input. To ensure the generated shape fits the input in those regions where the latter has been observed, we proceed as follows: In each block we determine the set union between the *surface* voxels of the original input and those of the generator output. With 1 representing the surface label and 0 the background, this corresponds to an element-wise max-operation on the voxel labels. The blocks thus generated are again encoded and pooled, then passed to the discriminator as further "fake" examples. To understand the effect of this supervision signal, consider first a block that has a too low point density in the generator output: that case, with a "fake" label, reinforces the desired bias towards complete shapes. More interesting is the case where the generated block has sufficiently many points. There are three cases: (a) the point count is correct, but the distribution is implausible – again the "fake" label is appropriate and steers the model towards plausible point distributions; (b) the distribution is plausible in itself, but not aligned with the input – the "fake" label is again correct and implements the conditioning on the input; (c) the block matches the input shape and has the right density – if this case becomes frequent the model has been successfully trained: the generator has learned to generate complete shapes that are well-aligned with the input scan.

3.2 Loss function

We derive our loss from the Least Squares GAN-loss (LSGAN), which for our case proved to be more stable and yielded better results than the original GAN-loss. Note, though, that our block scheme can be implemented with any other GAN loss function, too. The original LSGAN objective reads

$$\begin{aligned} \min_G \mathcal{L}(G) &= \frac{1}{2}\mathbb{E}[(D(G(z)-1)^2] \\ \min_D \mathcal{L}(D) &= \frac{1}{2}\mathbb{E}[(D(x)-1)^2] + \frac{1}{2}\mathbb{E}[(D(G(z)))^2] \,. \end{aligned} \tag{1}$$

In our scheme, the incomplete shape x is subdivided into blocks $q_j, j \in \{1 \dots n\}$. Moreover, we denote the ensemble of blocks, sorted in descending order of point count, by an asterisk superscript, so the sorted input becomes q^* and the sorted generator output $G^*(x)$. The discriminator is divided into three functions: The operation of encoding a block into a 1D vector with a series of 3D convolutions is denoted as $\Theta(\cdot)$. That same encoding, applied to multiple blocks and followed by average pooling into a single 1D vector, is written $\Xi(\cdot)$. Finally, the function $D(\cdot)$ denotes the last layer of the discriminator, which receives the pooled 1D vector and predicts whether the input is "real" or "fake".

The **discriminator loss** has three parts, as described in the previous section. They correspond to the m most complete input blocks, the $(n-m)$ remaining, "incomplete" input blocks, and the full set of merged input/output blocks:

$$\mathcal{L}(D) = \mathcal{L}_{real}(D) + \alpha\mathcal{L}_{incomp}(D) + \mathcal{L}_{fake}(D) \tag{2}$$

where α controls the weight for rejecting incomplete blocks. Since our target distribution should only contain complete shapes, the first term is simply the LSGAN loss for the "real" class,

$$\mathcal{L}_{real} = \left(D(\Xi_{j=1}^{m}(q_j^*)) - 1\right)^2 \tag{3}$$

To reject incomplete shape parts, we assign the low-density blocks to the "fake" class, although they were not produced by the generator:

$$\mathcal{L}_{incomp} = \left(D(\Xi_{j=m+1}^{n}(q_j^*))\right)^2 \tag{4}$$

This contribution has been peer-reviewed. The double-blind peer-review was conducted on the basis of the full paper.
https://doi.org/10.5194/isprs-annals-V-2-2022-143-2022 | © Author(s) 2022. CC BY 4.0 License.

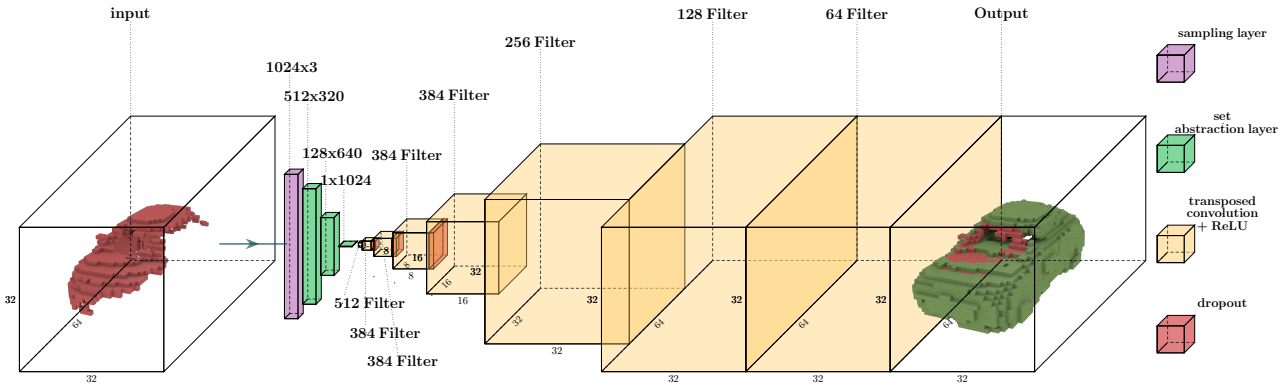


Figure 3. The generator network receives incomplete shapes (red) as input and completes them (green).

Note, completely empty blocks are removed before the discriminator, because nothing can be inferred from them. The third loss term is responsible for conditioning on the input x. Since, other than classical cGANs such as pix2pix (Isola et al., 2017), we do not have access to (complete) samples of the target distribution, we pull the generator output towards the incomplete input shape with the trick of merging them with the set union, see 3.1. By using 1 as surface label and 0 as background label, the set union can be written as element-wise max-operation and we obtain

$$\mathcal{L}_{fake}(D) = \left(D(\Xi_{j=1}^{n}(\max(G_j^*(x), q_j^*)))\right)^2 \tag{5}$$

As usual, the **generator loss** serves to penalise the generation of samples that are exposed as fake by the discriminator:

$$\mathcal{L}_{GAN}(G) = \left(D(\Xi(\max(G_k^*(x), q_k^*)) - 1)\right)^2 \tag{6}$$

Similar to pix2pix, the generator is also guided by a least squares loss so that it favors surface points near the input points.

$$\mathcal{L}_{\ell 2}(G) = \frac{1}{m}\sum_{j=1}^{m} ||G_k^*(x) - q_k^*||_2 \tag{7}$$

Note, that term is only applied to the m most complete blocks, so that the generator is not forced to produce regions that are incomplete or have low point density. Finally, we add a feature matching term that pulls the encodings $\Theta(\cdot)$ of generated blocks towards those of real ones:

$$\mathcal{L}_{feat}(G) = \frac{1}{m}\sum_{j=1}^{m} ||\Theta(q_k^*) - \Theta(G_k^*(x))||_2 \tag{8}$$

The generator is optimized by minimizing the weighted sum

$$\mathcal{L}(G) = \beta\mathcal{L}_{GAN}(G) + \gamma\mathcal{L}_{\ell 2}(G + \delta\mathcal{L}_{feat}(G) \tag{9}$$

3.3 Choice of architecture

After defining the loss function we still have to chose suitable architectures for the generator and the discriminator. An obvious idea for the **generator** would be a straight-forward encoder-decoder structure with 3D convolutions and 3D transposed convolutions, as in 3D-GAN (Wu et al., 2016). However, we observed difficulties with such a design. On the one hand, skip connections can lead to bad local minima where the generator just copies the input. On the other hand 3D convolutions do not work all that well with sparse, binary voxel data. We therefore prefer to use PointNet++ layers (Qi et al., 2017b) to encode the input. To do so, we start with a sampling layer that randomly picks a fixed number of 3D surface points, in our implementation 1024. These are passed through a sequence of PointNet++ set abstraction layers to obtain a feature vector of size 1×1024. Each set abstraction layer consists of three steps: (1) The sampling step, which uses iterative farthest-point sampling to select a set of points that define the centroids of local regions; (2) The grouping step that constructs local sets by selecting neighboring points around the centroids with a spherical query; (3) The PointNet step, which uses a mini-PointNet on each group to learn local patterns and encode them into feature vectors. The generator uses a total of three set abstraction steps, marked in green in Fig. 3. The implementation and hyper-parameters are the same as proposed by (Qi et al., 2017a). The output of the final set abstraction layer is transformed from 1×1024 into a $1 \times 1 \times 1 \times 1024$ tensor and passed through a series of 3D transposed convolutions to obtain the prediction. The transposed convolutions are arranged in a way that the output size of the generator matches the size of the input voxel grid. The output of the last layer is passed through a sigmoid to give an occupancy score between 0 and 1 for each voxel. To make the training more stable, batch-normalisation (Ioffe and Szegedy, 2015) and spectral normalisation (Miyato et al., 2018) are applied at every layer. The activation functions for all layers except the last one are rectified linear units (ReLUs).

The **discriminator** network consists of two parts, Θ and D. The encoder Θ is a traditional sequence of 3D convolutions

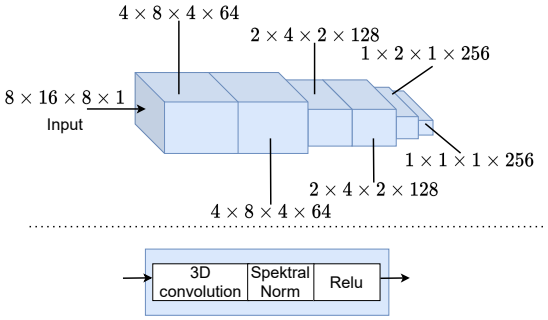


Figure 4. The network $\Theta(\cdot)$ encodes a block of size $8 \times 16 \times 8$ into a feature vector of size $1 \times 1 \times 1 \times 256$.

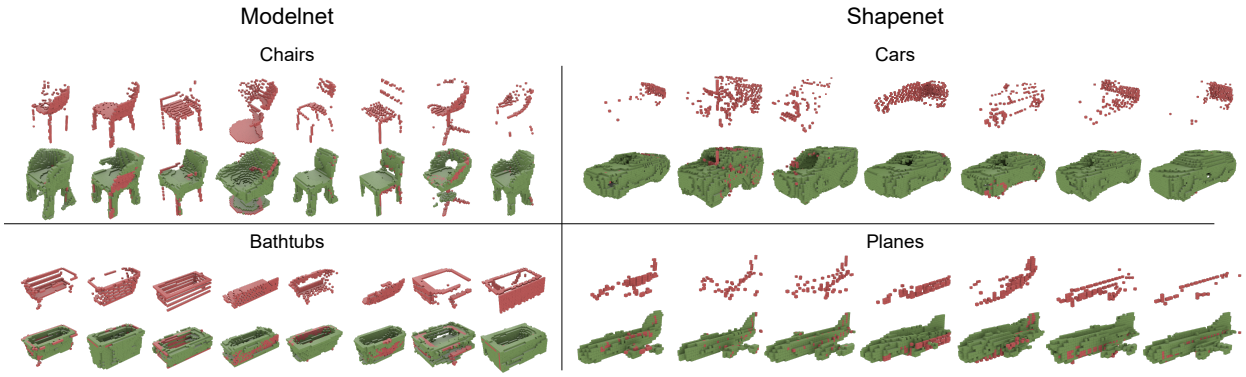


Figure 5. Predicted shapes for ModelNet chairs (top left), bathtubs (bottom left), ShapeNet cars (top right), and ShapeNet airplanes (bottom right). All networks were trained on very sparse samples. Red voxels denote the input and green voxels the predictions.

down to a vector of size 256, again with spectral normalization in each layer, see Fig. 4.

A set of encoded blocks is pooled and passed to the actual discriminator D, which classifies the blocks as "real" or "fake". For D, we use two fully connected layers, the first with 64 output features and ReLU activation, the second with an output of one and identity activation (no non-linearity). In the fully connected layers there is no normalisation. The same, shared encoder weights Θ are used fort all blocks. The encoder and the fully connected layers are trained jointly as part of the discriminator update. The complete architecture, as shown in Fig. 2, was implemented in Tensorflow and trained end-to-end.

3.4 Training

Adding noise to the voxel data turned out to decrease the quality of the output, so we refrain from it. The generator is trained with 50% drop-out, see Fig. 3, the discriminator without drop-out. Training is run with batch size 3 for 5 epochs. As optimiser, we use Adam with separate learning rates of 0.0001 for the discriminator and 0.001 generator, a strategy sometimes referred to as TTUR, (Heusel et al., 2017). During training, we chose the same number of blocks in all experiments, $n_x \times n_y \times n_z = 4 \times 4 \times 4$. For a voxel grid of size $32 \times 64 \times 32$, this gives 64 blocks of size $8 \times 16 \times 8$, of which 25% are considered "real", i.e., $m = 16$. Setting these parameters is not critical. Empirically, the training is stable across a range of different block sizes and leads to similar results.

3.5 Data preparation

Synthetic datasets. We first test our model on two synthetic datasets, ShapeNet (Chang et al., 2015) and ModelNet (Wu et al., 2015). From ShapeNet we used the categories *plane* and *car*, from ModelNet *chair* and *bathtub*. To create occluded samples, we used the procedure described by (Stutz and Geiger, 2018) with their recommended parameters, except that we do not employ scaling for data augmentation. Meshes are rotated randomly to sample different viewing angles and projected to a 2D image grid to obtain points, which are then back-projected into the 3D voxel grid. By adjusting the virtual camera parameters we can control the sampling density. We measure that density by the mIoU in voxels between the rendered data and the complete shape (hull only, without interior voxels). For *planes* these densities are $\approx$0.49 mIoU (high) and $\approx$0.12 mIoU (low), for *cars* they are $\approx$0.09 mIoU (high) and $\approx$0.024 mIoU (low). For Modelnet we only sample lower resolution samples with densities $\approx$0.19 mIoU for *bathtub* and $\approx$0.21 mIoU for *chair*.

Some existing works, in particular (Stutz and Geiger, 2018), predict filled shapes. To make the results comparable, we post-processed all our *car* predictions by flood-filling them with a graph cut algorithm, where the graph source and sink nodes are the barycentre and the boundary of the voxel volume, and an template car serves as prior to determine the edge weights.

Real data. To show that our model works also on real-world 3D data, we created a database of cars from scans captured with a Riegl VMX-250 mobile mapping system (MMS). Since we did not have car annotations for that data, we generated them in the following manner: we perform semantic segmentation of the image sequences acquired together with the scans, with Deeplabv3+ (Chen et al., 2018) trained on the Cityscapes dataset (Cordts et al., 2016), without any fine-tuning. We then projected the labels from all images onto the 3D point cloud and performed majority voting to obtain a label per point. Next, we removed all points not assigned to the car class, estimated point normals, and segmented the points into individual car instances by region-growing in the oriented point cloud. Since this simple segmentation is, as expected, rather noisy, we filtered out all instances that did not fit into a bounding box of $2 \times 5 \times 3$ m. In this way we obtained a set of 8941 fairly clean cars. Due to the low viewpoint from the road, they are heavily occluded, with at least one side entirely missing in most cases. To approximately align the incomplete scans, we fitted each of them to a generic car template (prototype car) with ICP.

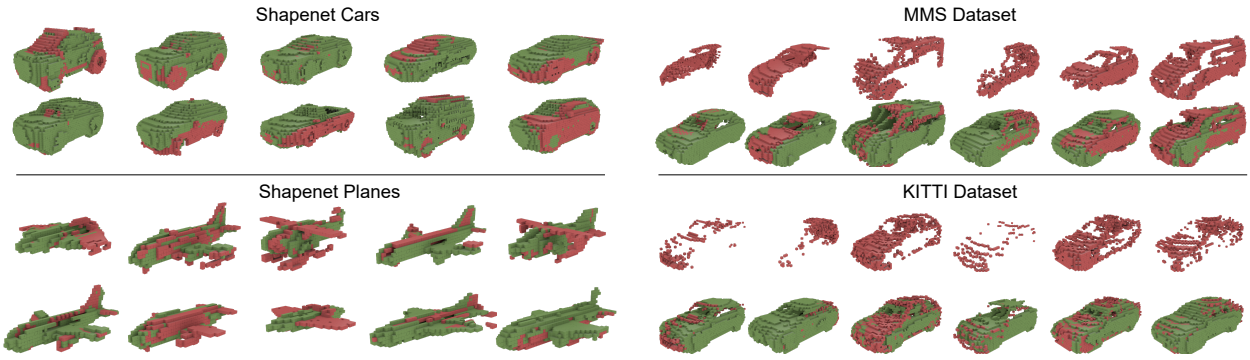
4. EXPERIMENTAL RESULTS

4.1 Qualitative Evaluation

Predictions for the synthetic as well as the real datasets were made with the same GAN architecture, trained per category, respectively dataset. In this section, we show (1) the impact of different scanner resolutions, (2) the performance of our model on real data, and (3) its ability to generalize to new sensors without retraining.

In Figure 5 we show the results for the categories *chair*, *bathtub*, *car* and *plane* with lower scan resolution. As can be seen, our GAN is able to complete shapes even from very sparse data. The figure shows that the bias against sparse sub-regions leads

This contribution has been peer-reviewed. The double-blind peer-review was conducted on the basis of the full paper.
https://doi.org/10.5194/isprs-annals-V-2-2022-143-2022 | © Author(s) 2022. CC BY 4.0 License.



(a) Completion examples for *cars* and *planes*.

(b) GAN trained on MMS (top), applied to KITTI without retraining.

Figure 6. Examples for synthetic data (a) and real data (b)

the model to fill more voxels. Still, overly implausible completions are detected by the discriminator and rejected. This is particularly visible for the ShapeNet planes in the lower row of Fig. 5, where the observations are so sparse that the planes are hardly recognisable.

Figure 6a shows additional results for high-resolution cars and planes from ShapeNet. These results are illustrate the variability of the data set, and the ability of our model to complete various instances with very different shapes.

Figure 6b shows predictions for the KITTI dataset, generated with the GAN trained on our MMS dataset. KITTI features a completely different type of laser scanner (Velodyne), which records a much lower point density. The model generalises very well, one can see that also for KITTI cars the predictions are mostly complete and match the input observations (i.e., the model does more than returning a mean car shape). Failures, where the generator predicts a severely incomplete shape or one that is in significant disagreement with the input, occur mostly on rare, big shapes, like trucks or buses. Indeed, there are fewer failures on KITTI, which does not contain any trucks, than on the MMS data.

4.2 Quantitative Evaluation

We measure the quality of shape completion by calculating, separately per category, the mean intersection over union (mIoU), accuracy (Acc) and completeness (Comp) of our predictions w.r.t. the ground truth shape. Accuracy is defined as the average distance from the predicted shape to the ground truth target, completeness is the average distance in the opposite direction. We compare our results to those obtained by(Stutz and Geiger, 2018) and by (Dai et al., 2017a). Note that both methods are to some degree *supervised* and constitute an upper bound for our self-supervised approach. Note also, they have used a slightly different voxel grid size for the *car* category. For synthetic data, we report accuracy and completeness in multiples of the voxel edge length [vx], for KITTI and MMS we use meters. As a simple baseline for the unsupervised setting, we also used a fixed template as the "completion result". For MMS and KITTI, this "prior shape" is the prototype car that was used to align the scans; for the other classes, we arbitrarily picked the first example in the database.

In Table 1, one can see that our GAN increases the mIoU significantly compared to the fixed prior shape, which confirms that

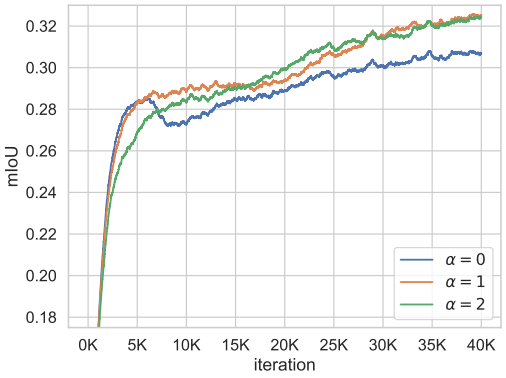


Figure 7. Ablation study showing the influence of the weight α for the loss term $\mathcal{L}_{incomp}$ on mIoU during training.

our generator does condition its predictions on the data. We point out that testing shape completion outputs against ground truth is somewhat problematic, for two reasons. First, there are many plausible completions of a partial shape – if a predicted shape does not match the ground truth, then that does not imply that it does not match another real instance of the target category. Second, we applied a threshold to the soft predictions to decide whether a cell is occupied or not – hence, differences to the ground truth can also be due to aliasing and noise, in situations where the soft posterior is well estimated. For several applications, e.g., localisation or shape comparison, there may not be a need to threshold the probabilistic occupancy grid to hard binary values, rather it is advantageous to work with soft occupancies that are more forgiving.

To show the influence of the adversarial hyper-parameters, we ran ablation studies with ModelNet chairs. Figure 7 shows three models that were trained on the dataset in a deterministic setting, with the exact same training samples in every iteration, such that the three runs differ only in terms of the weight α for the loss $\mathcal{L}_{incomp}$. Figure 7 shows that higher α indeed results in higher mIoU, and ensures that the model improves smoothly and monotonically as the training progresses.

In another study we set β and δ to zero to eliminate the adversarial term $\mathcal{L}_{GAN}(G)$ and the feature matching term $\mathcal{L}_{feat}(G)$ of the generator. We plot mIoU w.r.t. ground truth against training iterations. Figure 8 shows that the adversarial losses significantly improve the completion, and that the per-

This contribution has been peer-reviewed. The double-blind peer-review was conducted on the basis of the full paper.
https://doi.org/10.5194/isprs-annals-V-2-2022-143-2022 | © Author(s) 2022. CC BY 4.0 License.

Table 1. Quantitative Results. Numbers marked with † are taken from (Stutz and Geiger, 2018). Note that they use a slightly different voxel grid for the ShapeNet classes.

			ShapeNet Cars			ShapeNet Planes		
method	complete supervision	resolution	mIoU↑	Acc↓ [vx]	Comp↓ [vx]	mIoU↑	Acc↓ [vx]	Comp↓ [vx]
(Dai et al., 2017a)	100%	32×72×32	0.87†	0.32†	0.56†	–	–	–
(Stutz and Geiger, 2018)	≤7.7%		0.78†	0.54†	0.74†	–	–	–
Dataset	0%	32×64×32	0.09	–	2.9	0.49	–	1.75
Prior Shape			0.64	1.07	0.96	0.36	1.49	1.49
Ours			0.70	0.78	0.59	0.54	0.25	0.65

			ModelNet Bathtubs			ModelNet Chairs		
method	complete supervision	resolution	mIoU↑	Acc↓ [vx]	Comp↓ [vx]	mIoU↑	Acc↓ [vx]	Comp↓ [vx]
(Dai et al., 2017a)	100%	32×32×32	0.59†	–	–	0.61†	0.66†	0.67†
(Stutz and Geiger, 2018)	≤10%		0.50†	–	–	0.41†	1.49†	1.07†
Dataset	0%		0.19	–	2.75	0.21	–	1.85
Prior Shape			0.17	1.06	1.68	0.15	1.70	2.20
Ours			0.34	0.90	0.99	0.33	0.88	1.58

			MMS Dataset			KITTI (trained on MMS)		
method	complete supervision	resolution		Acc↓ [m]	Comp↓ [m]		Acc↓ [m]	Comp↓ [m]
Prototype Car	0%	32×64×32	–	–	0.12	–	–	0.09
Ours			–	–	0.03	–	–	0.08

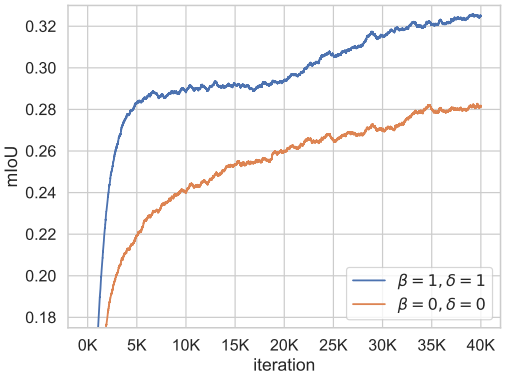


Figure 8. Ablation study showing the influence of the loss terms $\mathcal{L}_{GAN}(G)$ and $\mathcal{L}_{feat}(G)$ during training.

formance gap already appears at an early stage of the training and then persists.

5. CONCLUSION

We have presented a method for self-supervised, adversarial completion of partially observed 3D shapes. We assume that the object class is known and has moderate shape variability, such that it can be roughly aligned inside a bounding box. Two ideas form the core of our method and make self-supervised learning possible: (i) the overall volume is tiled into blocks, such that blocks with well-sampled surfaces can be used as positive samples for the discriminator, whereas blocks with too low point density serve as negative examples; and (ii) the set union between an incomplete input and the predicted output of the generator serves to condition the prediction on the input, by steering the generator towards shapes that coincide with the input, where available.

We have tested our method on ShapeNet, ModelNet and on real scans of cars in road scenes, where >50% of the object surface is in most cases unobserved. We have also shown that one can, with standard computer vision machinery, automatically extract car instances from raw scan data, such that the training set can be assembled without human interaction. Our datasets comprise >9'000 cars extracted from high-density mobile mapping data, and another >9'000 cars extracted from KITTI. An interesting direction for future work is to include attributes of the surface points beyond their location, such as colours or normal vectors, perhaps even certain material properties. It may also be useful to move away from the voxel representation and predict completed shapes in the form of point clouds or continuous, implicit occupancy functions.

REFERENCES

Chang, A. X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., Xiao, J., Yi, L., Yu, F., 2015. ShapeNet: An Information-Rich 3D Model Repository. *arXiv preprint arXiv:1512.03012*.

Chen, L.-C., Zhu, Y., Papandreou, G., Schroff, F., Adam, H., 2018. Encoder-decoder with atrous separable convolution for semantic image segmentation. *ECCV*.

Chen, X., Chen, B., Mitra, N. J., 2019a. Unpaired Point Cloud Completion on Real Scans using Adversarial Training. *arXiv preprint arXiv:1904.00069*.

Chen, X., Chen, B., Mitra, N. J., 2020. Unpaired point cloud completion on real scans using adversarial training. *ICLR*.

Chen, Y.-C., Tan, D. S., Cheng, W.-H., Hua, K.-L., 2019b. 3d object completion via class-conditional generative adversarial network. *International Conference on Multimedia Modeling*.

Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B., 2016. The Cityscapes dataset for semantic urban scene understanding. *CVPR*.

Criminisi, A., Pérez, P., Toyama, K., 2004. Exemplar-Based Image Inpainting. *IEEE TIP*, 13(9).

Dai, A., Qi, C. R., Nießner, M., 2017a. Shape completion using 3d-encoder-predictor CNNs and shape synthesis. *CVPR*.

Dai, B., Fidler, S., Urtasun, R., Lin, D., 2017b. Towards diverse and natural image descriptions via a conditional GAN. *ICCV*.

Firman, M., Mac Aodha, O., Julier, S., Brostow, G. J., 2016. Structured prediction of unobserved voxels from a single depth image. *CVPR*.

Getreuer, P., 2012. Total variation inpainting using split Bregman. *Image Processing Online*, 2, 147–157.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y., 2014. Generative adversarial nets. *NeurIPS*.

Han, X., Li, Z., Huang, H., Kalogerakis, E., Yu, Y., 2017. High-resolution shape completion using deep neural networks for global structure and local geometry inference. *ICCV*.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S., 2017. GANs trained by a two time-scale update rule converge to a local Nash equilibrium. *NeurIPS*.

Ioffe, S., Szegedy, C., 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *ICML*.

Isola, P., Zhu, J.-Y., Zhou, T., Efros, A. A., 2017. Image-to-image translation with conditional adversarial networks. *CVPR*.

Kingma, D. P., Welling, M., 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Kraevoy, V., Sheffer, A., 2005. Template-based mesh completion. *Eurographics Symposium on Geometry Processing*.

Lu, C., Dubbelman, G., 2019. Hallucinating beyond observation: learning to complete with partial observation and unpaired prior knowledge. *arXiv preprint arXiv:1907.09786*.

Mirza, M., Osindero, S., 2014. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*.

Mitra, N. J., Guibas, L. J., Pauly, M., 2006. Partial and approximate symmetry detection for 3D geometry. *ACM Transactions on Graphics*, 25(3), 560–568.

Mitra, N. J., Pauly, M., Wand, M., Ceylan, D., 2013. Symmetry in 3d geometry: Extraction and applications. *Computer Graphics Forum*, 32(6), 1–23.

Miyato, T., Kataoka, T., Koyama, M., Yoshida, Y., 2018. Spectral normalization for generative adversarial networks. *ICLR*.

Pauly, M., Mitra, N. J., Giesen, J., Gross, M. H., Guibas, L. J., 2005. Example-based 3d scan completion. *Eurographics Symposium on Geometry Processing*.

Pauly, M., Mitra, N. J., Wallner, J., Pottmann, H., Guibas, L. J., 2008. Discovering structural regularity in 3D geometry. *ACM Transactions on Graphics*, 27(3), 43.

Podolak, J., Shilane, P., Golovinskiy, A., Rusinkiewicz, S., Funkhouser, T., 2006. A planar-reflective symmetry transform for 3D shapes. *ACM Transactions on Graphics*, 25(3), 549–559.

Qi, C. R., Su, H., Mo, K., Guibas, L. J., 2017a. Pointnet: Deep learning on point sets for 3d classification and segmentation. *CVPR*.

Qi, C. R., Yi, L., Su, H., Guibas, L. J., 2017b. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *NeurIPS*.

Rezende, D. J., Eslami, S. A., Mohamed, S., Battaglia, P., Jaderberg, M., Heess, N., 2016. Unsupervised learning of 3d structure from images. *NeurIPS*.

Riegler, G., Ulusoy, A. O., Bischof, H., Geiger, A., 2017. OctnetFusion: Learning depth fusion from data. *3DV*.

Sharma, A., Grau, O., Fritz, M., 2016. VConv-DAE: Deep volumetric shape learning without object labels. *ECCV*.

Sipiran, I., Gregor, R., Schreck, T., 2014. Approximate symmetry detection in partial 3d meshes. *Computer Graphics Forum*, 33(7), 131–140.

Stutz, D., Geiger, A., 2018. Learning 3d shape completion from laser scan data with weak supervision. *CVPR*.

Sung, M., Kim, V. G., Angst, R., Guibas, L., 2015. Data-driven structural priors for shape completion. *ACM Transactions on Graphics*, 34(6), 175.

Wang, T.-C., Liu, M.-Y., Zhu, J.-Y., Tao, A., Kautz, J., Catanzaro, B., 2018. High-resolution image synthesis and semantic manipulation with conditional GANs. *CVPR*.

Wu, J., Zhang, C., Xue, T., Freeman, W. T., Tenenbaum, J. B., 2016. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. *NeurIPS*.

Wu, R., Chen, X., Zhuang, Y., Chen, B., 2020. Multimodal shape completion via conditional generative adversarial networks. *ECCV*.

Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J., 2015. 3d shapenets: A deep representation for volumetric shapes. *CVPR*.

Yang, C., Lu, X., Lin, Z., Shechtman, E., Wang, O., Li, H., 2017. High-resolution image inpainting using multi-scale neural patch synthesis. *CVPR*.

Yang, Y., Feng, C., Shen, Y., Tian, D., 2018. FoldingNet: Point cloud auto-encoder via deep grid deformation. *CVPR*.

Yuan, W., Khot, T., Held, D., Mertz, C., Hebert, M., 2018. PCN: Point completion network. *3DV*.

Zhang, J., Chen, X., Cai, Z., Pan, L., Zhao, H., Yi, S., Yeo, C. K., Dai, B., Loy, C. C., 2021. Unsupervised 3d shape completion through GAN inversion. *CVPR*.

Zhang, J., Zhan, R., Sun, D., Pan, G., 2018. Symmetry-aware face completion with generative adversarial networks. *ACCV*.

Zhao, H.-K., Osher, S., Fedkiw, R., 2001. Fast surface reconstruction using the level set method. *IEEE Workshop on Variational and Level Set Methods in Computer Vision*.