PFG (2020) 88:257–269
https://doi.org/10.1007/s41064-020-00114-z

ORIGINAL ARTICLE

Conditional Adversarial Networks for Multimodal Photo-Realistic Point Cloud Rendering

Torben Peters[1] · **Claus Brenner**[1]

Received: 4 September 2019 / Accepted: 17 April 2020 / Published online: 7 July 2020

Abstract
We investigate whether conditional generative adversarial networks (C-GANs) are suitable for point cloud rendering. For this purpose, we created a dataset containing approximately 150,000 renderings of point cloud–image pairs. The dataset was recorded using our mobile mapping system, with capture dates that spread across 1 year. Our model learns how to predict realistically looking images from just point cloud data. We show that we can use this approach to colourize point clouds without the usage of any camera images. Additionally, we show that by parameterizing the recording date, we are even able to predict realistically looking views for different seasons, from identical input point clouds.

Keywords Deep learning · GAN · Point cloud

Zusammenfassung
Nutzung von Conditional Generative Adversarial Networks für das multimodale photorealistische Rendering von Punktwolken. Wir untersuchen, ob Conditional Generative Adversarial Networks (C-GANs) für das Rendering von Punktwolken geeignet sind. Zu diesem Zweck haben wir einen Datensatz erstellt, der etwa 150.000 Bildpaare enthält, jedes bestehend aus einem Rendering einer Punktwolke und dem dazugehörigen Kamerabild. Der Datensatz wurde mit unserem Mobile Mapping System aufgezeichnet, wobei die Messkampagnen über ein Jahr verteilt durchgeführt wurden. Unser Modell lernt, ausschließlich auf Basis von Punktwolkendaten realistisch aussehende Bilder vorherzusagen. Wir zeigen, dass wir mit diesem Ansatz Punktwolken ohne die Verwendung von Kamerabildern kolorieren können. Darüber hinaus zeigen wir, dass wir durch die Parametrierung des Aufnahmedatums in der Lage sind, aus identischen Eingabepunktwolken realistisch aussehende Ansichten für verschiedene Jahreszeiten vorherzusagen.

1 Introduction

Laser scanned (LiDAR) point clouds are difficult to handle when it comes to photo-realistic rendering. Firstly, if images have been recorded together with the LiDAR points, a camera calibration is needed to colourize each scanned point. However, this does not guarantee that each 3D point is captured by a camera viewpoint. Secondly, since point clouds are sparse, it is difficult both to fill empty areas and to exclude occluded points, e.g. behind walls and buildings. To create the impression of a continuous surface, splats can be rendered instead of points. A splat is defined as an elliptical surface with a size determined by the local point density. Lastly, the colourized point cloud does not contain any information about the sky.

Our approach tries to circumvent the whole process of model-based point cloud rendering, by learning, in an end-to-end fashion, how a possible representation of the point cloud could look like in reality. Our key contributions in this work are:

- Creating a large dataset of point cloud–image pairs.
- Predicting photo-realistic views from point clouds which contain only (LiDAR) reflectance and distance information.
- Colourizing point clouds without the usage of cameras.

✉ Torben Peters
peters@ikg.uni-hannover.de

Claus Brenner
brenner@ikg.uni-hannover.de

[1] Institut für Kartographie und Geoinformatik, Leibniz Universität Hannover, Appelstraße 9a, 30167 Hannover, Germany

- Extending a C-GAN to parameterize different points in time, e.g. seasons and months, to predict multimodal images.
- Showing that the input is easily editable by a human to change the appearance of objects in the predicted images.

The paper is organized as follows: after a review of the state of the art of C-GANs in computer vision, the method developed in this paper is presented, as well as the datasets used and the preprocessing of the data, involving a parallelization framework. The results are evaluated both qualitatively and quantitatively using different metrics and approaches. A summary and outlook conclude the paper.

2 Related Work

Generative Adversarial Networks (GANs) were invented by Goodfellow et al. (2014). In their original form, they are deep generative models which are based on a game theoretic scenario. They consist of a generator and a discriminator network. The generator g directly produces samples $g(z) = x$. The discriminator tries to distinguish between samples drawn from the dataset and samples that are drawn from g. Instead of judging the generators performance by using a pixelwise metric, as it has been done, e.g. in variational autoencoders (Kingma and Welling 2013), the GAN loss provides a more sophisticated feature-based loss function. The generator loss is expressed by the probability of the discriminator network $d(x)$ if a sample is genuine or generated by g. Therefore, $d(x)$ is the probability that x is drawn from the training dataset. Both networks are trained jointly: the discriminator is trained by maximizing the probability of assigning the correct label to the training example and the example drawn from g, while the generator is trained by minimizing $1 - d(g(z))$.

Conditional GANs attracted a lot of attention in recent years. Most notable are the *pix2pix* network by Isola et al. (2017) and the improved version *pix2pixhd* (Wang et al. 2017), which is able to predict high-resolution image-to-image mappings. Like traditional GANs, these networks are using an adversarial loss which is learned by a discriminator network. In contrast to L_1-loss, the adversarial loss leads to less blurry images by learning to distinguish between real and generated images (Isola et al. 2017; Wang et al. 2017). In addition to image-to-image translation, there exist a wide range of conditional GANs, such as unpaired image-to-image translation (Zhu et al. 2017a) and text-to-image conversion (Dash et al. 2017; Zhang et al. 2017; Reed et al. 2016a, b).

Multimodal image-to-image translation defines the process of mapping one-to-many images, by modeling distributions of possible outcomes with an additional latent space vector or matrix, e.g. as used by *BicycleGAN* (Zhu et al. 2017b). Instead of mapping a whole image to many outcomes, Wang et al. (2017) mapped object classes in images to many different versions of the same class. This could be for example different road surfaces like stone or asphalt, or car types. They did this by encoding instance classes with a feature encoder network similar to an autoencoder. These feature maps were additionally fed into the input of the generator and trained end to end. By changing the feature vector in the input, they were able to manipulate specific objects in one image. However, this implies that they need instance labels in addition to image pairs to train the network. Other notable contributions for multimodal image-to-image translations like *PixelNN* (Bansal et al. 2017) or *MAD-GAN* (Ghosh et al. 2018) try to predict a discrete set of different outcomes. By enforcing them to be different, they try to create one-to-many mappings. This can be used, e.g. for image manipulations as shown by Park et al. (2019). Finally, there are approaches like *MUNIT* (Huang et al. 2018) which is able to solve this problem in an unsupervised manner.

Depending on the application, point clouds can be incorporated into deep neural networks in different ways. They can be represented as unordered 3D point sets (Qi et al. 2017a, b), voxelized (Wu et al. 2016; Maturana and Scherer 2015), or projected into images to use traditional 2D-convolutional networks (Boulch et al. 2017). When rendering point clouds, different problems arise. Firstly, point clouds do not contain any colour, sky or light source information. Secondly, they are sparse, and occlusions are common. Many approaches for handling occlusions have been proposed, for example by using surfels (surface elements), by Pfister et al. (2000), and surface splatting, by Zwicker et al. (2001). They try to deal with the fact that 3D points are not connected and thus do not have any surface information. Furthermore, illumination in computer graphics can be tackled by following the light rays from a virtual source by using ray tracing, path tracing or scanline rendering (Whitted 1979; Kajiya 1986; Bouknight 1970). All these approaches are applicable to point clouds, but they are hard to realize due to the sparsity of non-meshed 3D points. Additionally, those approaches introduce high computational complexity. By learning to directly map from 3D points to images, we try to circumvent the whole process of model-based rendering.

Similar GAN-based approaches have been proposed by Atienza (2019) and Milz et al. (2019). The first work uses a combination of point cloud data and a background image patch to render a 2D scene. They condition the general look of the scene by the background patch and the content of the scene with a point cloud of a 3D object. The second model is trained on synthetic objects. They infer the raw point cloud

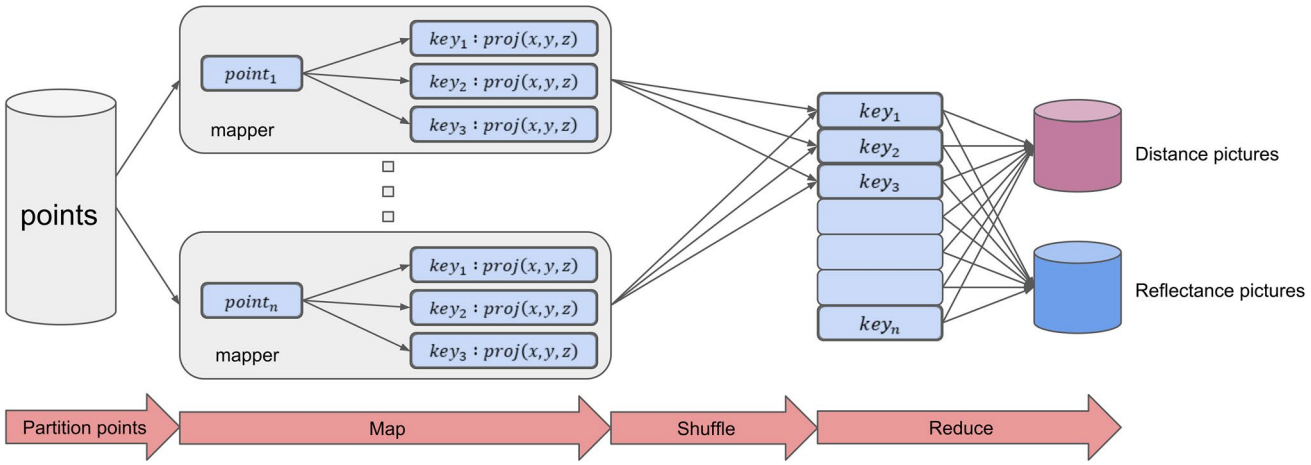

Fig. 1 Our MapReduce approach for rendering large point clouds

of a single object and target viewing angles and generate a rendered image which is compared to a target mesh rendering. In both cases, the object classes are known and the point clouds are annotated.

However, our work differs from these works because we do not need object classes or annotations, since we exploit the geometric correspondence between camera images and 3D LiDAR point cloud. Additionally, we condition the general appearance of the scene by parameterizing the image date to encode seasonal information.

3 Data Acquisition and Pre-processing

3.1 Mobile Mapping Dataset

The datasets used were produced with the mobile mapping system Riegl VMX-250. This system captures a maximum of 600,000 3D points per second and has four cameras, which were set to take images at a rate of 1 Hz each. The points are acquired with a LiDAR accuracy of 1 cm, with absolute accuracies being typically in the range of 10–20 cm.

To prepare a training dataset, we used mobile mapping data which we captured during 14 measurement campaigns, over the duration of 1 year, in Hannover, Germany. For an independent test dataset, we used another point cloud which we recorded during a campaign in the city of Karlsruhe, about 500 km away from Hannover, during winter (in February).

To illustrate the extent of the required processing, the subset of the data from Hannover we were using in this work contains 15 billion (15,017,586,980) 3D points and 123,047 images. Each image capture is given in terms of position (in UTM coordinates) and orientation (roll, pitch, yaw angles). Additionally, the intrinsic parameters of each camera are known due to a pre-calibration.

3.2 Preparing the Dataset Using MapReduce

Our C-GAN is based on *pix2pixhd* which means that it needs to be trained in a supervised manner. The C-GAN needs the data in terms of pairs of projected point cloud image and corresponding (real) image. The projected point cloud image contains two channels. The first channel stores the distance between 3D point and camera centre and the second channel the reflectance of the laser ray. The task is therefore to project each of the 15 billion 3D points to 2D pixels in 123,047 images and store the distance and reflection values in the corresponding pixels. To solve this task, we created a massively parallel point cloud renderer, using the MapReduce framework on an Apache Hadoop cluster (Fig. 1).

To apply MapReduce, each mapper has a list of all image orientations. According to the MapReduce principle, it receives a subset (split) of 3D point coordinates and their reflectance values (reflectance is an entity derived from the LiDAR amplitude measurement). To reduce the amount of points emitted by the mapper, we exclude points that are behind the camera or are further away than 300 m. The mapper possibly emits multiple key–value pairs per incoming 3D point, depending on the number of images the point appears in. The key is defined by the image name, identifying a single image take, whereas the value contains the distance, reflectance and the image coordinates of the projected point. Each reducer receives all necessary information, grouped by image (key), and computes two 16-bit grey-value images per key, one containing the distance and the other one containing the reflectance values per point. If more than one point falls into the same 2D pixel, we keep the one with the smallest distance to the camera centre. Depending on the

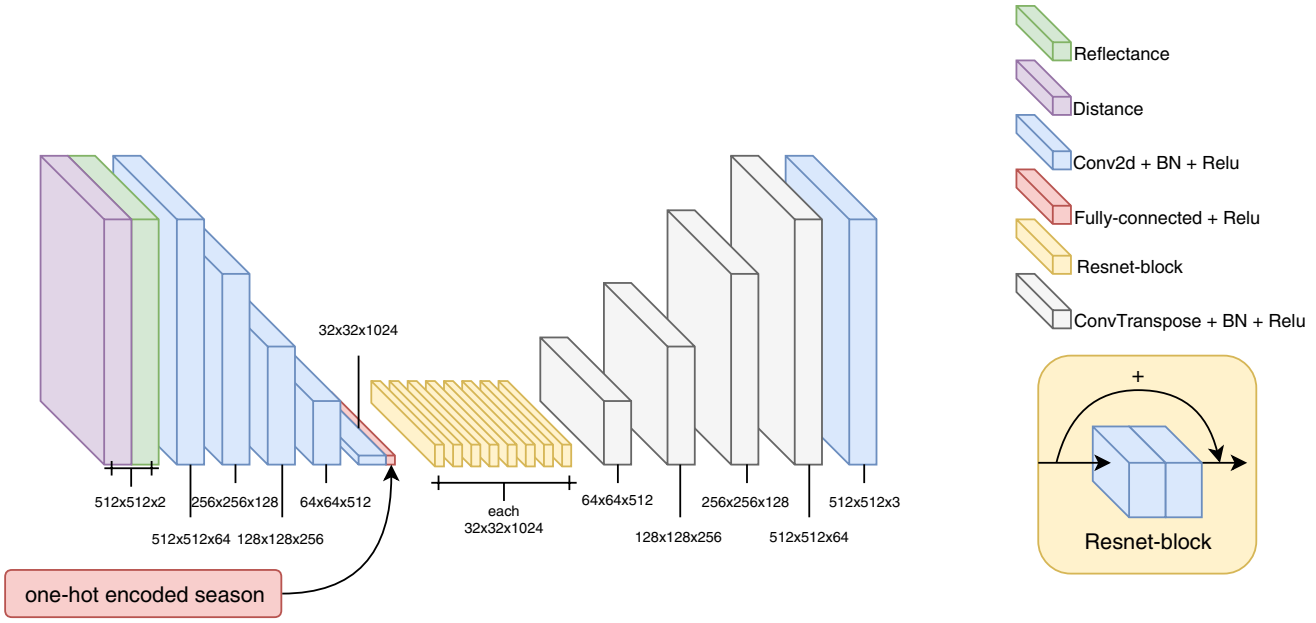

Fig. 2 Our adapted generator network. Note how the capture date is injected by concatenating a fully connected layer (red)

scanning situation, the points appear more or less sparse in the image plane. Additionally, because of the small difference in capture time between camera sensor and LiDAR, moving objects might appear in the camera image but not in the point cloud or vice versa.

4 Learning Using a C-GAN

Our approach is inspired by the *pix2pixhd* network by Wang et al. (2017). The *pix2pixhd* network incorporates instance segmentation information and label maps to enable object manipulation. By encoding the features of one instance, it is able to generate diverse images from the same input.

We modified the generator network as follows. We removed the instance- and label-maps from the network architecture because we do not have any information about the class or instance of each point. The original generator is split into at least two subnetworks, e.g. a global and a local part (Wang et al. 2017). The global section shown in Fig. 2 is based on the architecture of Johnson et al. (2016). Instead of an architecture similar to a U-net with skip connections, this network is based on residual connections that allows each layer to easily learn an identity mapping, which according to Johnson et al. should help to preserve the original image structure. We believe that this architecture is better suited than U-Net for the seasonal encoding, since skip connections could potentially skip the concatenated seasonal information in the bottleneck. The global network forms the core network, which in our case produces an image resolution of 512×512. In *pix2pixhd*, the local enhancer networks are wrapped around the global network and output an image the size of $2\times$ of each image dimension. Due to hardware limitations, we did not use the local enhancer network and reduced the number of multi-scale discriminators to two $\{D_1, D_2\}$. Each discriminator operates on a different image scale, D_1 at the original scale 512×512, and D_2 at 256×256. However, the discriminator networks have exactly the same architecture as defined by Wang et al. (2017). We adopted the $\mathcal{L}_{\text{GAN}}$ part of the loss function as follows:

$$\begin{aligned} &\min_G \max_{D_1,D_2} \sum_{k=1,2} \mathcal{L}_{\text{GAN}}(G, D_k) \\ &\quad = \sum_{k=1,2} E_{(x,y)}[\log D_k(x,y)] \\ &\qquad + E[\log(1 - D_k(x, G(x,s)))]. \end{aligned} \tag{1}$$

The training dataset is given as a set of tuples of corresponding images and dates $\{(x_i, s_i, y_i)\}$, where x_i is the input/reflectance image, y_i is the real image, taken by a camera of our mapping van, and s_i is the date the image was taken.

To create diverse outputs and explicitly control the season of our prediction, we tried to feed the date by concatenating a one-hot vector to the input image, which, however, did not lead to the intended effect, as it was ignored. Similar to the findings of Isola et al. (2017), we also observed that noise fed additionally and directly into the generator was completely ignored by the network and did not create any diverse results at all. Therefore, as shown in Fig. 2, the fully connected layer was instead concatenated to the bottleneck after convolving the input which resulted in the desired effect. Since we have conducted 14 mapping

Fig. 3 Input image (reflectance, left), synthesized image (middle) and real image (right)

Fig. 4 Summer (middle) and winter (right) representation of the same input point cloud (left)

campaigns throughout the year, the fully connected layer encodes the campaign respectively date with an one-hot encoded input vector of 14 and an output of 1024. We reshape the output to a size of $32 \times 32 \times 1$ and concatenate it along the last axis to the image feature maps as shown in Fig. 2.

The following layers of the generator network are identical to the *pix2pixhd* network. The one-hot encoding for each capture date, s_i, is defined as follows:

$$f(s_i) = \begin{cases} 1, & \text{if } s_i = \text{date} \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

We additionally tried to smooth the one-hot vector by adding $\mathcal{N}(\mu, \sigma^2)$ to $f(s_i)$ in order to get a continuous transition between each season. However, this resulted only in slight changes between each prediction and the parameterized seasons stayed discrete.

5 Experiments and Results

We trained the networks for 20 epochs with a batch size of one. After 20 epochs, we observed that the generator starts ignoring the season information and overfits to the actual image.

Figure 3 shows an example of a predicted image from the train set. Remember that this is computed using only the reflectance and distance information from the point cloud. Note that the predicted building is coloured in a typical colour (white walls and red roof), while in reality the building has quite different colours (red walls and dark roof). We therefore believe that the colour information is mostly derived from the spatial information and not from the (LiDAR) reflectance of the points themselves.

Figure 4 shows that by shifting the value in the one-hot encoded season vector, we are able to predict different seasons for the same LiDAR input. In this case, we used a point cloud from the train set that was recorded in Hannover, Germany, in March and predicted an image for June and December. It can be observed that the predicted scene is greener in summer, and also the colours are brighter, whereas the colours in winter are paler. Also, the light goes through the tree crown because it has no leaves in winter. In addition, light snow covers the streets, with less snow in the road centres, where the cars have blown it to the sides. Also typical traces of tyres are captured in the winter scene.

However, we think that some features will stay encoded in the point cloud itself. For example, the amount of leaves which are captured by the LiDAR could be the reason for a relatively sparse tree crown in the summer image. It is also worth mentioning that there are a large number of occluded points in the left pane of Fig. 4. From the middle and right pane of Fig. 4, it can be seen that the generator has learnt to hide occluded points.

We also created a video[1] which shows the differences between summer and winter.

5.1 Evaluation

Since GANs are not trained on the basis of a traditional loss function, the evaluation of the results is extremely difficult. The overview made by Borji shows that there is currently no consensus on how to evaluate a GAN (Borji 2019). Like Wang et al. and Isola et al., we decided to evaluate the C-GAN based on the interpretability of a pre-trained network (Wang et al. 2017; Isola et al. 2017). The idea is that a network trained on real data can interpret the generated samples well if they have a high degree of realism. For this purpose, we used Deeplabv3+, a state of the art network created by Chen et al., which is used for semantic segmentation in images (Chen et al. 2018). Depending on the situation, we pre-trained Deeplabv3+ on Cityscapes or PASCAL VOC 2012 (Cordts et al. 2016; Everingham et al. 2010). Both datasets contain similar street scenes as the dataset we created. In Cityscapes, different object classes are evaluated that would be encountered by a typical road user, for example, cars, streets, buildings, pedestrians, trees or street signs. The PASCAL VOC 2012 data on the other hand separates between the class background and foreground objects like cars.

Fig. 5 Karlsruhe in winter (left) and the corresponding synthesized summer images (right)

Fig. 6 Examples for summer (middle) and winter (right) representation of the same input point cloud coloured by reflectance (left) in Hannover (row 1 and 2) and Karlsruhe (row 3 and 4)

[1] Video url: youtu.be/33fBXfaYA7E.

Fig. 7 Synthesized images with cars

Fig. 8 Cars from Fig. 7 successfully classified in fake images (bright grey)

5.1.1 Qualitative Evaluation

We investigated whether our approach generalizes well by testing the C-GAN on a different city. The point cloud captured in the city Karlsruhe was projected to 2D images as described before. Because the data were recorded in winter, we decided to map the point cloud to summer to show the capability of synthesizing a different season. In Fig. 5, we show pairs of images taken in Karlsruhe and the corresponding synthesized images from the point cloud. We would like to point out that these images and 3D points were never part of the training dataset. Also here, the general impression of a summer scene is given: it mainly relates to the green scenery and the warmer colours used. In Fig. 6, we additionally show pairs of synthetic images for the seasons winter and summer, taken in Hannover and Karlsruhe. The example should show that the performance of the generator is similar for both seasons in the training and test set.

In the next step, we did a qualitative evaluation of how good dynamic objects (cars) are recognized by a pretrained neural network. We would like to note that we cannot calculate any metric for dynamic objects in synthesized images because they do not intersect with the corresponding real images. We show a few examples where DeepLabv3+ was able to successfully predict cars in the generated fake images. We have chosen a network that was trained on the PASCAL VOC 2012 dataset which separates between background class and object classes (e.g. car). We had the impression that this network achieved better results for cars than the ones trained on Cityscapes. We used the images in Fig. 7, which include many cars at different positions. To visualize the results, we merged the images by laying the prediction over the corresponding fake image (Fig. 8).

In Fig. 8, the bright regions are the predicted cars and the dark colour shows the class background. The semantic maps were not altered, which means that the network only predicted cars and background classes in these scenes. As can be seen, the results are looking convincing, which means that it could be possible to correctly label cars in the generated fake images.

5.1.2 Input Manipulation

In contrast to *pix2pix* which uses RGB, our input data contain distance and reflectance values. We observed that these are distinct features which can be manipulated to change the output of the C-GAN in a meaningful way. In the following section, we altered the normalized reflectance channel x_{ref} by using a threshold:

$$x_{\text{ref}} = \begin{cases} 0.5, & \text{if } x_{\text{ref}} > 0 \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

As seen in Fig. 9, the objects in the predicted images remain intact and only differ in the textures. It appears that the reflectance encodes properties such as lane markings, objects colours, road sign and seasonal information. For example, the C-GAN predicted no road markings in the altered images. Additionally, the trees looked fuller and greener, and the colour of the buildings did change. In the

Fig. 9 Images with orignal (left) and thresholded reflectance (right)

next step, we show in Fig. 10 that changing the reflectance manually (using a painting program) results in different outcomes related to its spatial context. We painted some bright lines on the road and on the tree crowns, the rest of the image was given a reflectance of 0.5. Firstly, a higher reflectance on the road is translated to lane markings by the C-GAN. Secondly, the same reflectance value on trees results in more and darker leafs (upper right corner). This makes sense, because the laser ray uses near infrared, which is reflected better by leaves with high amounts of chlorophyll. The same applies to lane markings which are made using retro-reflective paint, to be visible for traffic participants. We think that the context awareness of the C-GAN makes it easier for humans to manipulate the input. Finally, we can imagine that the manipulation could be combined with point class labels to selectively change the appearance of specific objects by changing their reflectance.

Table 1 IoU between real and synthesized image

Classes	IoU
Road	0.845
Sidewalk	0.289
Building	0.561
Wall	0.186
Fence	0.218
Pole	0.177
Traffic light	0.013
Traffic sign	0.150
Vegetation	0.806
Terrain	0.330
Sky	0.813
Average IoU	0.399

5.1.3 Quantitative Evaluation

For a quantitative evaluation of our approach, we created pairs of semantic segmented images from a real and a corresponding fake image. For the semantic segmentation, we used DeepLabv3+ which was trained on the Cityscapes dataset. The degree of correspondence between the two representations is evaluated by comparing the resulting classifications. The performance of the model was calculated by measuring the intersection over union (IoU). The IoU has a range between 0 (worst) and 1 (best). It is calculated as follows:

$$\mathrm{IoU} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP} + \mathrm{FN}}, \tag{4}$$

Fig. 10 Left to right: reflectance altered manually, original output and altered output

Table 2 FID and SSIM scores computed for every campaign. The closer FID is to zero and SSIM to one, the better it is

Campaign	0	1	2	3	4	5	6
FID ↓	13.1	15.5	12.9	12.4	11.9	14.3	14.8
MS-SSIM ↑	0.54	0.47	0.54	0.59	0.56	0.5	0.55
Campaign	7	8	9	10	11	12	13
FID ↓	15.4	13.3	12.7	12.7	14.0	31.0	34.2
MS-SSIM ↑	0.51	0.55	0.54	0.54	0.49	0.43	0.43
Campaign	All						
FID ↓	9.6						
MS-SSIM ↑	0.51						

where TP, FP, and FN are the true positive, false positive, and false negative pixel counts. However, due to the time difference between capturing the point cloud and the camera image, dynamic objects may appear at different positions. Therefore, we think it makes only sense to measure the IoU for classes of objects which are non-moving. As can be seen in Table 1, classes have a large IoU if the corresponding objects have large extents, such as road, building, vegetation and sky. The mean IoU is small for classes corresponding to small objects, such as traffic sign and traffic light, pole, fence and wall.

We think that one reason for the relatively low IoU comes from the fact that we have only limited hardware capacity and cannot train the full *pix2pixhd* model. Our model was trained on a Nvidia Titan X which has 12 GB of memory. As stated by Wang et al. (2017), they needed 24 GB of memory to train their model, which achieved a mean IoU of 0.6389.

We also calculated the Fréchet Inception Distance (FID) and the multi-scale structural similarity (MS-SSIM) between the synthesized and target images. The basic idea of FID is that the distributions of the extracted features by a pre-trained network between the generated and real images should be similar if the generator performs well. This method was introduced by Heusel et al. to measure the performance of GANs in images (Heusel et al. 2017). In practice, these features are extracted by the penultimate layer of Inception-v3. The synthesized $X_g = \mathcal{N}(\mu_g, \Sigma_g)$ and target image $X_t = \mathcal{N}(\mu_t, \Sigma_t)$ distributions are modeled as multi-dimensional Gaussians parameterized by their mean

Fig. 11 Randomly sampled pairs of predictions (left) and ground truth images (right) for every campaign. The pairs are sorted by campaign number (campaign 0 is top left and 13 bottom right)

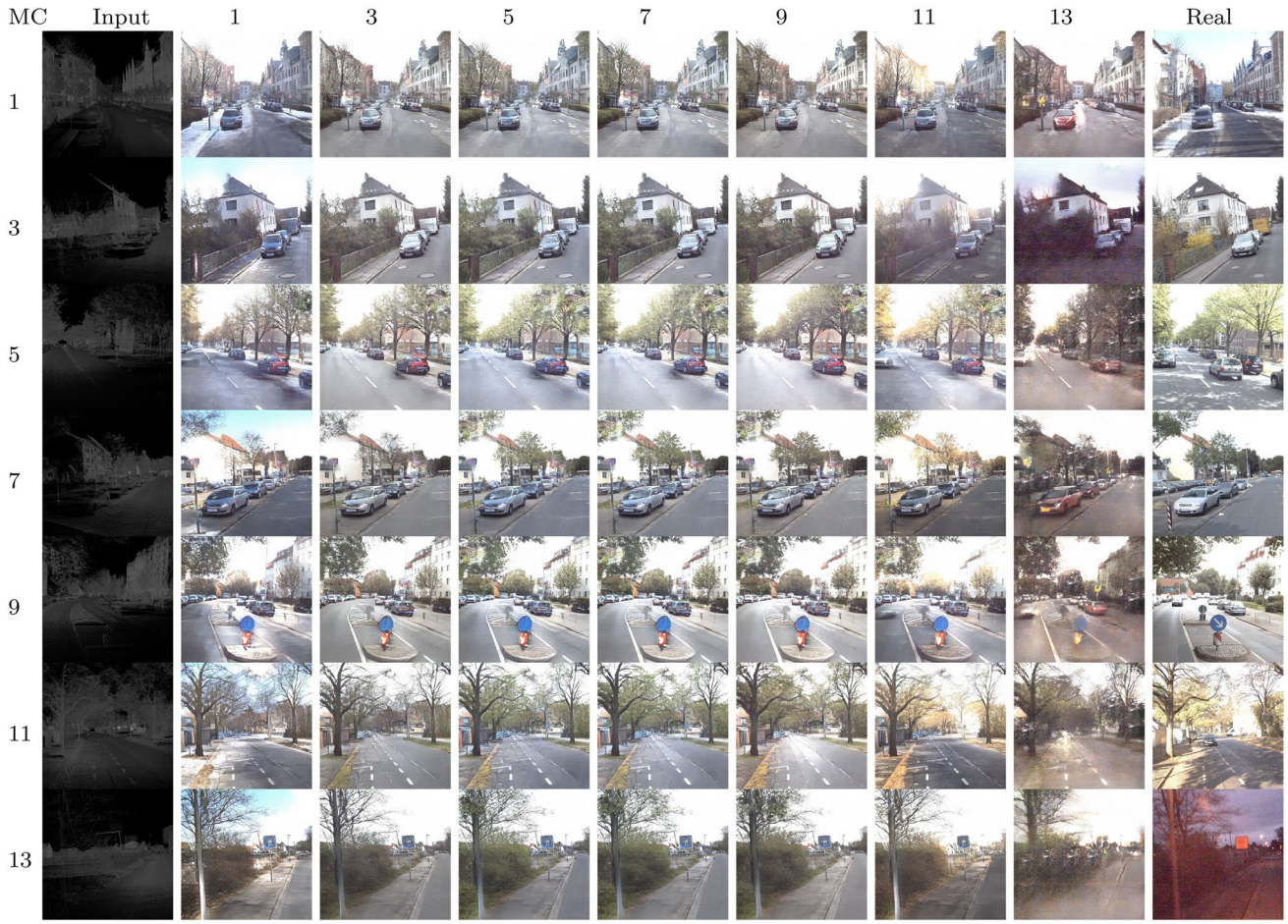

Fig. 12 Different representations for the same input point clouds. Each row shows an example from one measurement campaign (MC). The columns show the input (left), the different predicted seasons (1–13) and the corresponding real image (right)

μ and covariance Σ. The FID distance can be calculated by the following equation:

$$\begin{aligned} \text{FID} = &||\mu_t - \mu_g||^2 \\ &+ Tr(\Sigma_t + \Sigma_g - 2(\Sigma_t \Sigma_g)^{1/2}). \end{aligned} \tag{5}$$

MS-SSIM introduced by Wang et al. (2003, 2004), on the other hand, measures the distance by comparing the luminance, contrast, and structure of the images at different scales. The score is then calculated by the weighted product of all three terms.

We calculated firstly the total distances on the entire dataset and secondly the distance per measurement campaign to see if the generator would capture the seasonal characteristics. The images are generated according to the date they are captured. As a rough guide, Atienza gives an FID value of 31.5 and an SSID value of 0.64 at best (Atienza 2019). However, it should be noted that the scores are difficult to compare, as both procedures were trained on different datasets.

Table 2 shows that the results of FID and MS-SSIM are similar for all campaigns except 12 and 13. It shows that the generator has problems capturing their characteristics, resulting in lower scores. One reason for this could be that these campaigns were started in daylight and ended at night. As a result, many bright but also dark images have been captured which also contain image noise.

To visualize the results in the Table 2, we show a pair of predicted and real images for each campaign in Fig. 11. To create these images, we used the correct date for the predictions. In addition, in Fig. 12 we show a different representation for the same input point cloud for every other campaign. Both figures are intended to show that the generator is generally able to capture the specific characteristics of a season or mapping campaign, including campaigns 12 and 13.

Fig. 13 Point clouds colourized by using our C-GAN

5.2 Colourizing Point Clouds

To colourize point clouds, the conventional approach is to map 3D points into the camera coordinate system, using exterior and interior orientation. As described above, this process has to tackle several difficulties. We therefore want to show that we are able to circumvent this process by removing the camera and using our synthesized images for point cloud colouration. The general idea is that the colours of the synthesized images can be easily projected back into the point cloud. The colourization is assumed to be good if the colours match the form and structure of the 3D objects. Figure 13 shows the result of mapping the generated images back to the point cloud. It can be seen that the buildings and cars show convincing textures. This is especially true for the red lights of the cars, and also for the homogeneous darker colour of the first floor of the buildings on the left side of the road. These images demonstrate that the generated RGB information generally fits the structure of the point cloud.

6 Conclusion and Outlook

In this work, we have shown that it is possible to predict realistically looking images, using only point cloud data, once they are trained with pairs of point clouds and corresponding images. By parameterizing the different capture dates of the images and point clouds, we were able to map the same point cloud to different seasons. We have shown that the C-GAN was able to encode seasonal information, like snow in winter or green trees in summer. Furthermore, the generator was able to hide occluded points and also fill gaps in the point cloud appropriately. Furthermore, we have shown that the generated images fit nicely to the 3D points by mapping the generated RGB pixel back to the 3D points. A quantitative evaluation showed the similarity between the original and the synthesized images.

Additionally, we were able to show that our network generalizes well by testing it on a different city. We are convinced that this process gives us the ability of mapping a specific city style in a specific season to a completely different city, only by providing point clouds.

Lastly, we were able to show that the input can be edited by a human to change the appearance of objects in the predicted images. We did this by altering the reflectance with a painting program. The C-GAN translated the reflectance values according to the objects and context and changed their appearance appropriately. We think that this is a property that makes it easier for humans to manipulate the output.

Provided that our point cloud is labelled, it is imaginable that this framework allows us to project high precision labels to the generated images to create or enrich datasets for semantic segmentation. Since our approach allows to define arbitrary view positions and angles, this would enable us to generate an arbitrary amount of training examples.

To improve the seasonal conditioning in the future, we want to implement an ACGAN-like structure which has been introduced by Odena et al. (2017). By forcing the discriminator to classify the season, the generator should be less likely to overfit to the capture date of the input point cloud and could better generalize to different seasons. This should also allow a U-Net-like architecture which could further improve the quality of the predictions.

Acknowledgements Open Access funding provided by Projekt DEAL. This work was funded by the German Research Foundation (DFG) as a part of the Research Training Group GRK2159, ‘Integrity and collaboration in dynamic sensor networks’ (i.c.sens).

References

Atienza R (2019) A conditional generative adversarial network for rendering point clouds. In: IEEE conference on computer vision and pattern recognition workshops, CVPR workshops 2019, Long Beach, CA, USA. Computer Vision Foundation/ IEEE, pp 10–17

Bansal A, Sheikh Y, Ramanan D (2017) Pixelnn: example-based image synthesis. arXiv preprint arXiv:1708.05349

Borji A (2019) Pros and cons of gan evaluation measures. Comput Vis Image Underst 179:41–65

Bouknight WJ (1970) A procedure for generation of three-dimensional half-toned computer graphics presentations. Commun ACM 13(9):527–536

Boulch A, Guerry J, Le Saux B, Audebert N (2018) SnapNet: 3D point cloud semantic labeling with 2D deep segmentation networks. Comput Graph 71:189–198. https://doi.org/10.1016/j.cag.2017.11.010

Chen L-C, Zhu Y, Papandreou G, Schroff F, Adam H (2018) Encoder-decoder with atrous separable convolution for semantic image segmentation. In: Ferrari V, Hebert M, Sminchisescu C, Weiss Y (eds) Computer vision – ECCV 2018. Springer International Publishing, Cham, pp 833–851

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. In: 2016 IEEE conference on computer vision and pattern recognition (CVPR). Presented at the 2016 IEEE conference on computer vision and pattern recognition (CVPR), pp 3213–3223. https://doi.org/10.1109/CVPR.2016.350

Dash A, Gamboa JCB, Ahmed S, Liwicki M, Afzal MZ (2017) Tac-gan-text conditioned auxiliary classifier generative adversarial network. arXiv preprint arXiv:1703.06412

Everingham M, Van Gool L, Williams CKI, Winn J, Zisserman A (2010) The pascal visual object classes (voc) challenge. Int J Comput Vis 88(2):303–338

Ghosh A, Kulharia V, Namboodiri VP, Torr PH, Dokania PK (2018) Multi-agent diverse generative adversarial networks. In: 2018 IEEE/CVF conference on computer vision and pattern recognition. Presented at the 2018 IEEE/CVF conference on computer vision and pattern recognition, pp 8513–8521. https://doi.org/10.1109/CVPR.2018.00888

Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y (2014) Generative adversarial nets. In: Proceedings of the 27th international conference on neural information processing systems, vol 2, NIPS'14. MIT Press, Cambridge, MA, USA, pp 2672–2680

Milz S, Simon M, Fischer K, Pöpperl M, Gross H-M (2019) Points2Pix: 3D point-cloud to image translation using conditional GANs. In: Fink GA, Frintrop S, Jiang X (eds) Pattern recognition. Springer International Publishing, Cham, pp 387–400

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. In: Proceedings of the 31st international conference on neural information processing systems, NIPS'17. Curran Associates Inc., Red Hook, NY, USA, pp 6629–6640

Huang X, Liu MY, Belongie S, Kautz J (2018) Multimodal unsupervised image-to-image translation. In: Ferrari V, Hebert M, Sminchisescu C, Weiss Y (eds) Computer vision – ECCV 2018. Springer International Publishing, Cham, pp 179–196

Isola P, Zhu JY, Zhou T, Efros AA (2017) Image-to-Image translation with conditional adversarial networks. In: 2017 IEEE conference on computer vision and pattern recognition (CVPR), pp 5967–5976

Johnson J, Alahi A, Fei-Fei L (2016) Perceptual losses for real-time style transfer and super-resolution. In: Leibe B, Matas J, Sebe N, Welling M (eds) Computer vision – ECCV 2016. Springer International Publishing, Cham, pp 694–711

Kajiya JT (1986) The rendering equation. In: Proceedings of the 13th annual conference on computer graphics and interactive techniques, SIGGRAPH '86. Association for Computing Machinery, New York, NY, USA, pp 143–150. https://doi.org/10.1145/15922.15902

Kingma DP, Welling M (2013) Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114

Maturana D, Scherer S (2015) A 3D convolutional neural network for real-time object recognition. In: 2015 IEEE/RSJ international conference on intelligent robots and systems (IROS). Presented at the 2015 IEEE/RSJ international conference on intelligent robots and systems (IROS), pp 922–928. https://doi.org/10.1109/IROS.2015.7353481

Odena A, Olah C, Shlens J (2017) Conditional image synthesis with auxiliary classifier GANs. In: Precup D, Teh YW (eds) Proceedings of the 34th international conference on machine learning, vol 70, ICML'17. JMLR.org, pp 2642–2651

Park T, Liu MY, Wang TC, Zhu JY (2019) Semantic image synthesis with spatially-adaptive normalization. In: 2019 IEEE/CVF conference on computer vision and pattern recognition (CVPR). Presented at the 2019 IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 2332–2341. https://doi.org/10.1109/CVPR.2019.00244

Pfister H, Zwicker M, Van Baar J, Gross M (2000) Surfels: surface elements as rendering primitives. In: Proceedings of the 27th annual conference on computer graphics and interactive techniques, SIGGRAPH '00. ACM Press/Addison-Wesley Publishing Co., USA, pp 335–342. https://doi.org/10.1145/344779.344936

Qi CR, Yi L, Su H, Guibas LJ (2017) PointNet++: deep hierarchical feature learning on point sets in a metric space. In: Proceedings of the 31st international conference on neural information processing systems, NIPS'17. Curran Associates Inc., Red Hook, NY, USA, pp 5105–5114

Qi CR, Su H, Mo K, Guibas LJ (2017) PointNet: Deep learning on point sets for 3D classification and segmentation. In: 2017 IEEE conference on computer vision and pattern recognition (CVPR). Presented at the 2017 IEEE conference on computer vision and pattern recognition (CVPR), pp 77–85. https://doi.org/10.1109/CVPR.2017.16

Reed SE, Akata Z, Mohan S, Tenka S, Schiele B, Lee H (2016) Learning what and where to draw. In: Proceedings of the 30th international conference on neural information processing systems, NIPS'16. Curran Associates Inc., Red Hook, NY, USA, pp 217–225

Reed S, Akata Z, Yan X, Logeswaran L, Schiele B, Lee H (2016) Generative adversarial text to image synthesis. In: Proceedings

of the 33rd international conference on international conference on machine learning, vol 48, ICML'16. JMLR.org, pp 1060–1069

Wang TC, Liu MY, Zhu JY, Tao A, Kautz J, Catanzaro B (2018) High-resolution image synthesis and semantic manipulation with conditional GANs. In: 2018 IEEE/CVF conference on computer vision and pattern recognition. Presented at the 2018 IEEE/CVF conference on computer vision and pattern recognition, pp 8798–8807. https://doi.org/10.1109/CVPR.2018.00917

Wang Z, Bovik AC, Sheikh HR, Simoncelli EP (2004) Image quality assessment: from error visibility to structural similarity. IEEE Trans Image Process 13:600–612. https://doi.org/10.1109/TIP.2003.819861

Wang Z, Simoncelli EP, Bovik AC (2003) Multiscale structural similarity for image quality assessment. In: The thirty-seventh asilomar conference on signals, systems & computers, 2003. Presented at the thirty-seventh asilomar conference on signals, systems & computers, vol 2. pp 1398–1402. https://doi.org/10.1109/ACSSC.2003.1292216

Whitted T (1979) An improved illumination model for shaded display. In: Proceedings of the 6th annual conference on computer graphics and interactive techniques, SIGGRAPH '79. Association for Computing Machinery, New York, NY, USA, p 14. https://doi.org/10.1145/800249.807419

Wu J, Zhang C, Xue T, Freeman B, Tenenbaum J (2016) Learning a probabilistic latent space of object shapes via 3D generative-adversarial modeling. In: Proceedings of the 30th international conference on neural information processing systems, NIPS'16. Curran Associates Inc., Red Hook, NY, USA, pp 82–90

Zhang H, Xu T, Li H, Zhang S, Huang X, Wang X, Metaxas D (2017) StackGAN: text to photo-realistic image synthesis with stacked generative adversarial networks. In: 2017 IEEE international conference on computer vision (ICCV). Presented at the 2017 IEEE international conference on computer vision (ICCV), pp 5908–5916. https://doi.org/10.1109/ICCV.2017.629

Zhu JY, Park T, Isola P, Efros AA (2017) Unpaired image-to-image translation using cycle-consistent adversarial networks. In: 2017 IEEE international conference on computer vision (ICCV). Presented at the 2017 IEEE international conference on computer vision (ICCV), pp 2242–2251. https://doi.org/10.1109/ICCV.2017.244

Zhu JY, Zhang R, Pathak D, Darrell T, Efros AA, Wang O, Shechtman E (2017) Toward multimodal image-to-image translation. In: Proceedings of the 31st international conference on neural information processing systems, NIPS'17. Curran Associates Inc., Red Hook, NY, USA, pp 465–476

Zwicker M, Pfister H, Van Baar J, Gross M (2001) Surface splatting. In: Proceedings of the 28th annual conference on computer graphics and interactive techniques, SIGGRAPH '01. Association for Computing Machinery, New York, NY, USA, pp 371–378. https://doi.org/10.1145/383259.383300