

# PIU-Net: Generation of Virtual Panoramic Views from Colored Point Clouds

Michael Georg Adam<sup>a</sup> and Eckeard Steinbach<sup>b</sup>

Chair of Media Technology, Technical University of Munich, Germany

**Keywords:** Virtual View Generation, Rendering, Deep Learning, Digital Twin, Virtual Reality.

**Abstract:** As VR-systems become more and more widespread, the interest in high-quality content increases drastically. One way of generating such data is by digitizing real world environments using SLAM-based mapping devices, which capture both the geometry of the environment (typically as point clouds) and its appearance (typically as a small set of RGB panoramas). However, when creating such digital representations of real-world spaces, artifacts and missing data cannot be fully avoided. Furthermore, free movement is often restricted. In this paper, we introduce a technique, which allows for the generation of high quality panoramic views at any position within a captured real world scenario. Our method consists of two steps. First, we render virtual panoramas from the projected point cloud data. Those views exhibit imperfections in comparison to the real panoramas, which can be corrected in the second step by an inpainting neural network. The network itself is trained using a small set of panoramic images captured during the mapping process. In order to take full advantage of the panoramic information, we use a U-Net-like structure with circular convolutions. Further, a custom perceptual panoramic loss is applied. The resulting virtual panoramas show high quality and spatial consistency. Furthermore, the network learns to correct erroneous point cloud data. We evaluate the proposed approach by generating virtual panoramas along novel trajectories where the panorama positions deviate from the originally selected capturing points and observe that the proposed approach is able to generate smooth and temporally consistent walkthrough sequences.


## 1 INTRODUCTION


With the introduction of consumer-grade head mounted displays (HMDs) such as the HTC Vive or Oculus Rift, Virtual Reality (VR) experiences have become more accessible and affordable. This growing market demands high quality content for gaming and video applications. As programming and rendering virtual scenes can be expensive and time consuming, capturing real world scenarios for later use in VR applications becomes important.

Two main solutions for producing such VR content from real world scenes exist. Both come with their own challenges. One way is by taking multiple high quality pictures from the scene and recording the respective capturing pose. Afterwards, a navigation graph is constructed. The user can then navigate through the graph by selecting the images. A well known example for this is the google streetview service (Anguelov et al., 2010). This approach,

however, leads to discrete movements and a non-immersive user-experience. It can be improved by a denser capturing of pictures, which leads to a higher memory demand. Furthermore, those datasets only support movements in predefined directions and the user is not able to choose his/her position freely. If three-dimensional views are desired, stereoscopic pictures must be taken during the capturing process.

The second way of generating VR content from real world recordings is 3D data, such as colored point clouds. Such data can be produced by a visual SLAM-algorithm, which, for instance, fuses camera and lidar data. The resulting views often lack details, since point clouds still cannot be captured as densely as real photos, even though dense reconstruction algorithms exist (Kerl et al., 2013). Nevertheless, they still have blind spots/holes where not enough data could be captured and are not always applicable due to cost or application specific reasons. The advantage of 3D data is, however, that (stereoscopic) views can be rendered from any point inside the data. This means that the user can move freely in the VR representation and in a smooth way.

<sup>a</sup>  <https://orcid.org/0000-0002-9512-5062>

<sup>b</sup>  <https://orcid.org/0000-0001-8853-2703>

Combined approaches exist as well. For instance the Stanford dataset includes both, point clouds and panoramas (Armeni et al., 2017). Here the user can select between the spatially registered representations. However, no stereoscopic images are included. Again, they can be rendered from the point cloud, but only sparse representations can be achieved.

This paper closes the gap between the two types of representations and inherits their advantages of free movement and high quality visualization. We present a process of producing dense and high resolution monoscopic or stereoscopic panoramic views at any location inside a point cloud, by exploring the photo-inpainting capabilities of neural networks.

## 2 RELATED WORK

Depending on the representation, the problem to be solved is either inpainting point clouds or generating virtual views between two or more images. The latter one can be done either geometry-based or by using transformations learned by neural networks (Flynn et al., 2016). (Sung et al., 2014) for instance predicts the view at the desired position based on its surrounding images and then merges those predictions. Generating training data for those approaches is crucial and often too expensive or not applicable, since certain places for taking photos are not reachable or accessible.

Generating 3D representations with geometric methods has already been shown in (Moezzi et al., 1997). Advancements are still proposed, for instance by the authors of (Hedman et al., 2017), who are able to generate photo-realistic point clouds. Rendering virtual views in this case collapses to a simple point cloud rendering task. However, they have in common that their 3D representation contains areas where data is missing, even though this data can be found in the corresponding images. Closing those blind spots turned out to be difficult and only classical approaches, which fill the missing information by interpolation of the existing data, were developed (Sahay and Rajagopalan, 2015; Fu et al., 2018).

A combined classical and learning-based approach is (Lai et al., 2019). They try to predict dense depth maps from stereo images with neural networks, in order to then use the depth-map for denser three-dimensional reconstruction.

An approach similar to ours (Aliev et al., 2019) describes the point cloud by neural network generated features, which are later decoded into dense views. Contrary to (Aliev et al., 2019), we do not use descriptors but render views directly from the

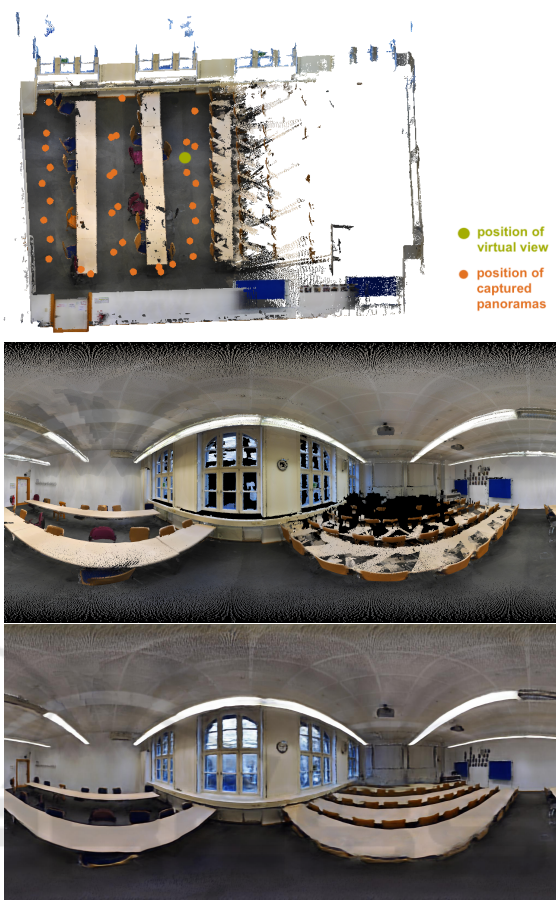


Figure 1: Point cloud showing a classroom and the position of the captured photos (top). Comparison of the rendered (middle) and the inpainted view (bottom) at a position not included in the training data. The wall and the windows as well as the back of the classroom are completed with semantically and geometrically correct information.

point cloud, which guarantees consistency of rendered views throughout the scene. We evaluate this by rendering walkthrough video sequences. Since we use real photos we can also fill in bigger holes with complex structure, for which the corresponding information is missing in the point cloud data. As no decoder has to be learned, only few training samples are sufficient.

The paper (Bui et al., 2018) introduces the same underlying approach as our method. However, we show in this paper, that one can achieve similar results without the use of generative adversarial networks, but rather only a CNN. Because of the reduced complexity, we were able to train high-resolution networks. Further, we specialized the network to produce panoramic images, rather than one view. Since the authors do not directly train on the real images, their network is not capable of filling missing point cloud data. A very recent related work (Dai et al.,

2020) tries to improve this approach by directly rendering views from the volume data.

### 3 THE DATA STRUCTURE

Our virtual panorama generation process assumes the availability of a colored point cloud representing a scene. Further, a finite set of high-quality pictures needs to be taken inside the scene. The corresponding capturing positions of the images have to be recorded as well. Such data can be produced by advanced SLAM algorithms such as (Mur-Artal and Tardós, 2017). In this paper, only indoor environments are tested. Those are considered more difficult since they are typically more complex. Also more occlusion occurs, due to shorter distances. In addition to that, we use equirectangular images as they capture all possible views at a certain position. Such data can be found, for instance, in the Stanford dataset (Armeni et al., 2017). After a training process, only the point cloud data is needed when handling data of the same domain.

### 4 THE ALGORITHM

Our method of producing panoramic images at any position inside the captured scene consists of two steps. In the first step, we render virtual views from the point cloud. In the second step, the resulting images are enhanced by a trained neural network. The point cloud rendering, neural network architecture and its training process are described in the following subsections.

#### 4.1 Panoramic Projection

Instead of moving the ego view to the desired position, the point cloud is translated and rotated accordingly. This causes the viewpoint to be equal to the system origin. The  $x, y, z$ -coordinates of the individual points are then transformed into spherical coordinates:

$$\begin{bmatrix} r \\ \phi \\ \theta \end{bmatrix} = \begin{bmatrix} \sqrt{x^2 + y^2 + z^2} \\ \arctan2(y, x) \\ \arccos \frac{z}{\sqrt{x^2 + y^2 + z^2}} \end{bmatrix} \quad (1)$$

Now  $\phi$  and  $\theta$  can be seen as the new  $x$  and  $y$  coordinates of a projected image plane and can be quantized. The number of quantization steps corresponds to the desired image resolution.

```

translate & rotate ( points );
 $\phi, \theta, r$  = spherical coordinates ( points );
quantize (  $\phi, \theta$ , desired_image_resolution );
sort ( points, by_r );
for  $p$  in points do
| image (  $\phi(p) \pm \epsilon, \theta(p) \pm \epsilon$  ) = p.color_value;
end

```

Figure 2: Pseudo-Code of the panorama rendering.

Successively, the points are sorted in descending order by the  $r$  coordinate, equivalent to the depth information of the rendered view. In an iteration over this ordered list, the empty image gets filled with the color-value of the point at its projected position. Also the adjacent pixels inside a  $\epsilon$ -region of the projected point are colored. If only the pixel at the projected point position is colored, a sparse image is rendered. By also considering the neighboring pixels, a denser result can be achieved, without losing information of close-by point cloud points. This corresponds to a simplified surfel-strategy (Pfister et al., 2000) and optimizes the training-process as the network needs to fill in less black space, but rather learns the high-frequency components of the image.

At the end, the origin of the image coordinate system is shifted to the upper left corner. The Pseudo-code of this rendering method is given in Fig. 2. The middle part of Fig. 1 shows an example of a rendered panorama. The rendered image has holes and black areas, where no corresponding information is found in the point cloud.

#### 4.2 PIU-Net

Next, the generated panoramas are used to train a neural network. This network should learn how to fill the missing data, hence Panorama Inpainting U-Net (PIU-Net). U-Net is a convolutional feedforward network originally developed for image segmentation. We adapt its structure, as it provides direct connections between input and output layers (Ronneberger et al., 2015). This makes sense, since most of the pixels in the panorama are already colored and thus can be passed through. The network only has to learn, which of the pixels can be kept and which have to be modified. Furthermore, U-Net allows for a small training dataset to achieve reasonable results. This is important, since the amount of captured real photos, which are used for training, is finite.

As the network has to deal with panoramic information, where the end of the image corresponds to the beginning and vice-versa, normal convolution is not ideal. Instead of applying a specific panorama

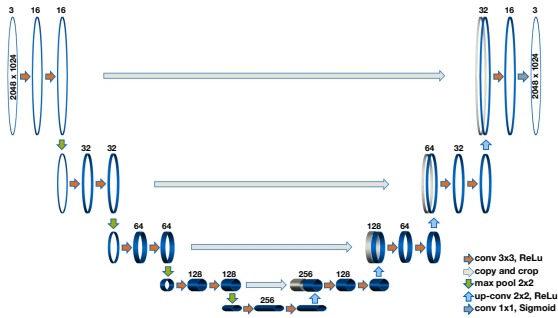


Figure 3: Structure of the Panorama Inpainting U-Net, based on (Ronneberger et al., 2015).

loss like in (Lai et al., 2019) or input-padding, we replace all convolution layers with circular convolutions, which were first introduced in the context of CNNs in (Schubert et al., 2019). This has the advantage that the flow of information is not cut at the borders of the panoramas and the circular information is used throughout all layers of the network. This leads to rotation invariance, which is discussed later. Corresponding to the original implementation, ReLu-activation-functions and He-normal weight-initialization is used. As a last step, the input and output is fixed to have three channels, since we use the network to inpaint but not to segment and classify. The resulting structure is shown in Fig. 3.

### 4.3 Perceptual Panorama Loss

During the experiments, we noticed that a pure mean-squared-error (MSE) loss already leads to convincing results. However, introducing a more advanced loss for training can further improve the quality of the inpainted panoramas. In order to apply a perceptual loss (Johnson et al., 2016), we use the last feature vector  $f$  of VGG16-Net (Simonyan and Zisserman, 2014), which was pretrained on the ImageNet dataset (Deng et al., 2009). This helps the network to create even more realistic looking images.

Further, we introduce a panoramic loss, which considers the different patch size a pixel covers on the sphere.

$$A = \int_{\phi_i}^{\phi_{i+1}} \int_{\theta_j}^{\theta_{j+1}} r^2 \sin\theta d\theta d\phi \quad (2)$$

According to the formula, the area of a pixel only varies along the vertical axis in a (co)-sinusoidal manner. This problem has to be accounted for, because errors along the equator correspond to bigger errors in the projection of the real world than errors at the poles. Hence, the loss term should be weighted accordingly. Since we shift the origin of the image to the upper left corner, the cosine becomes a sine. The

total loss is then defined as

$$Loss(x, y) = \sin\left(\frac{y+0.5}{Y_{res}}\right) \left[ \text{MSE}(o_{pr}(x, y), o_{gt}(x, y)) + w \sum_f \text{MSE}(\text{VGG16}(o_{pr}), \text{VGG16}(o_{gt})) \right] \quad (3)$$

Where  $o_{pr}$  and  $o_{gt}$  correspond to the current output of the network and the ground truth.  $w$  is a weighting factor between the two losses. This also accounts for other constants resulting from integrating Equation 2. The 0.5 is added inside the cosine, so that each pixel center is used.  $Y_{res}$  is defined as the amount of pixels in the image along the  $y$ -axis.

### 4.4 Training Process

In order to train the neural network, first panoramas are rendered from the point cloud at the same positions where the real panoramas were captured by high-quality cameras. The rendered panoramas are then used as the network input. The camera-produced panoramas are utilized as the trainable output. The expected result is that the network learns to transform sparse virtual panoramas into photo-quality views. For the loss-function the previously defined loss with a weighting factor  $w = 0.1$  is used. As more training data is needed, we make use of data augmentation techniques such as vertical flips and shifts. The resulting data is split into training and validation data. In a first step, we want to produce realistic views anywhere in the scene, thus we test the network only by judging the quality of photos produced at different positions than the training data without comparison to specific test data. In a second step, we then also compare real world test-photos to their virtual counterparts.

## 5 EXPERIMENT

In this paper an image resolution of  $2048 \times 1024$  is used. Higher resolutions could be achieved with the same architecture. Therefore, we reduce the number of channels used in the original U-Net implementation by a factor of four. We keep the stride of three in the circular convolutions. Figure 3 visualizes this. The tested datasets were recorded with the M3 mapping trolley from NavVis (NavVis GmbH, 2019). One set was captured in a classroom and the other in a hallway. For the first training run, the class room was used. It took around one day on an NVIDIA Titan xp to converge and resulted in a MSE-loss of below 0.008. For all following tests we provide video sequences in the supplementary material which are more illustrative.

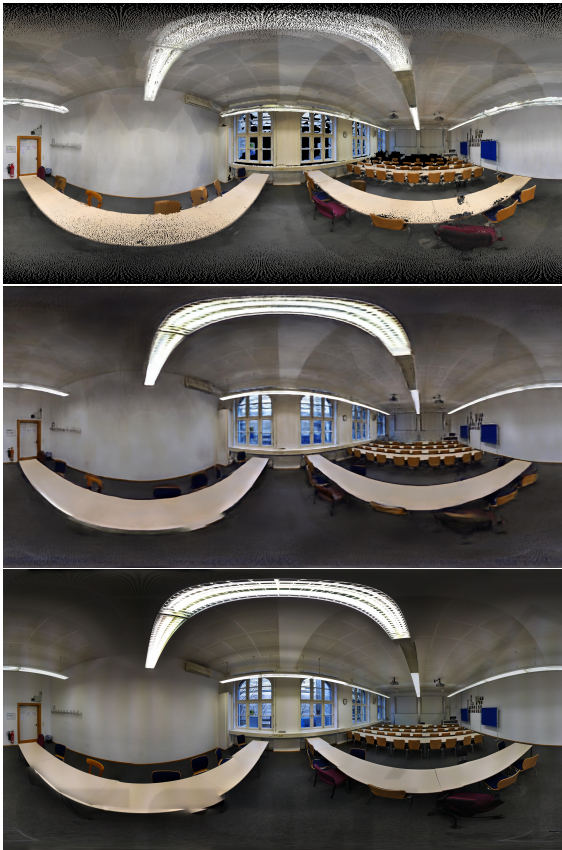


Figure 4: Comparison of captured photo (bottom), the rendered (top) and the inpainted view (middle). Only the correct type of chair is revised with blue pads. The panorama stitching artifacts (e.g. discontinuity at the table) occurring in the captured photo, are also learned by the network.

### 5.1 Fill-in

In Fig. 1 and all following examples it can be seen that the holes in the rendered panoramic images can be filled geometrically as well as semantically in a meaningful manner. For instance, we did not scan the end of the classroom, which had the effect that data was missing in the point cloud representation. The network, however, could learn this information from the photos, which contain the relevant information. Furthermore the windows are filled now. Such transparent and reflective areas are hard to capture, hence the data is usually missing in the point cloud.

### 5.2 Correction

The network not only learns to inpaint missing data, but also corrects image areas as can be seen in Fig 4. In the point cloud representation of the chairs, blue pads are either missing or are displayed in brown. However, in reality the pads are blue as can be seen



Figure 5: Three generated views at the same position, but with different orientation.

in the real photos. Furthermore the network learns to differentiate between the chair type, since the blue pad is only added to the correct type of chair.

### 5.3 Orientation Independency

We next show that the resulting pictures in Fig. 5 do not depend on the orientation. In order to test this, we generate several images at the same position each with a different horizontal angle. Differences between the frames of a rotation cannot be recognized by a human. The computed MSE between the rotated images, when rotated back, stays below 0.006. The result can be reasoned in the architecture since we use circular convolution and shift as a data augmentation technique. This confirms and also visualizes the results of (Schubert et al., 2019).

### 5.4 Spatial Consistency

Figures 1, 4–6 and 9 show the same classroom scene at different positions. One can see by inspection that they all show the same underlying data. This shows that the network is aware of its position and the geom-



Figure 6: Generated view outside of the captured area.

erty of the environment. Otherwise the generated images would not fit together. This spatial consistency can especially be seen in the video sequences, where all frames are generated individually. The frames only change slightly due to the ego motion, but not due to some misrepresentation of the environment. This geometric consistency can further be explained by the fact that the basic structure of each view is rendered from the point cloud. The point cloud itself always stays the same and forms the underlying structure for each rendered view. Thus, the rendered views can be seen as a template on which the network builds to produce artificially completed views.

Shadow artifacts in the training photos, which are not spatially consistent across the room, such as the vertical stripes seen on the walls, can be reduced by the network.

## 5.5 Dreaming

In order to investigate the limits of the proposed approach, panoramas are generated at the boundary of the captured scene. One can observe that the further the view exits the scene, the more the neural network is, what we call, dreaming about the correct appearance. An example is shown in Fig. 6. The result gets more blurred and sometimes resembles deep dreaming effects (Mordvintsev et al., 2015). This can be reasoned as the input to the network gets more and more black and less learned data can be applied. The part of the picture, which still shows meaningful data, can still be filled. This can be seen, for instance, on the right, where the blue pin board and the gallery can be reconstructed, although they never have been captured from this frontal angle.

## 5.6 Transfer Learning

As a second step we also run this process on a recorded hallway in order to proof that it is applicable to any data set. This time, however, we did not train from the beginning but used the previously trained weights for initialization. Training now only took less

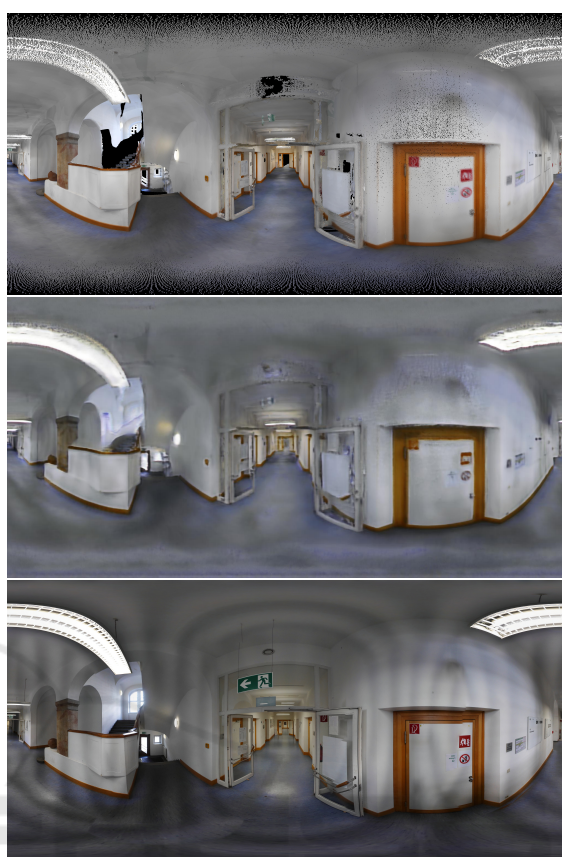


Figure 7: Comparison of the rendered (top) and the painted view (middle) with a captured test-photo (bottom) inside the trained area. The real panoramas show vertical stripes from bad illumination and stitching, which also affects the quality of the predictions.

than half an hour and the results, shown in Fig. 7, are of similar quality as for the classroom scene. For instance, the exit on the left and the stairs can be fully recovered. For this we also provide a comparison to a real world panorama, which was not used for training.

## 5.7 Same Domain

Furthermore, we test this previously refined network on another section of the hallway, which was not part of the training. As the scene is from the same domain (looking similar) the network is still able to fill in the missing information in a meaningful manner. However, better results can be achieved when retraining the network. This can be seen by comparing Fig. 7 and Fig. 8. It is important to note that the network not only fills the missing data by learned patterns, but also considers the sparse data which form the basis of the rendered views. This can be seen by close observation of the posters and floor-plan. Although a bit blurry, they are all filled individually without changing the

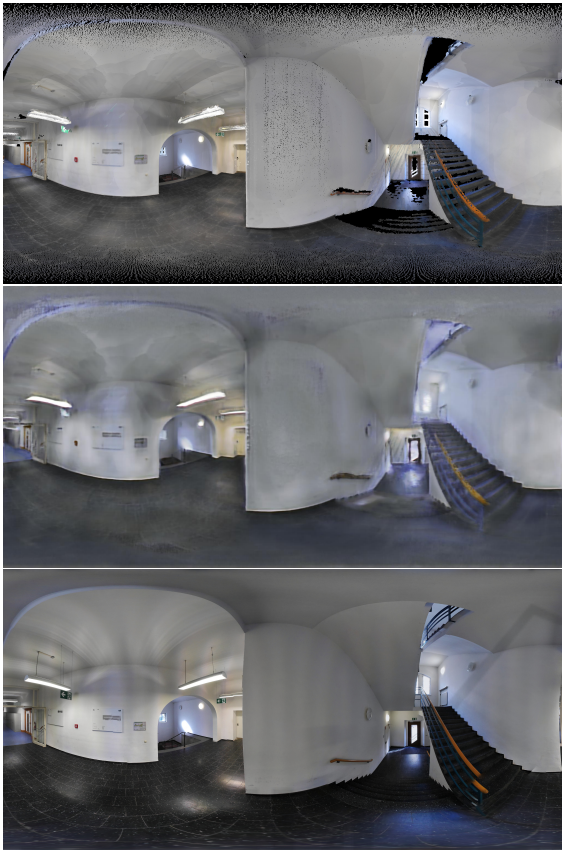


Figure 8: Comparison of the rendered (top) and the in-painted view (middle) with a captured test-photo (bottom) outside the trained area.

underlying pattern. This blurriness again reminds of some dreaming artifacts. For VR related applications this probably has low relevance as long as it is not a part of the user’s focus.

## 5.8 Loss Comparison

All previously generated panoramic views made use of the more advanced perceptual panoramic loss. In order to visualize its effect, a second network was trained with only a channel-wise MSE. A comparison between two generated panoramas can be seen in Fig. 9. When using the more advanced loss, the overall sharpness of the image improves. For instance, the ceiling maintains its texture. Further, the grey chair in the middle of the image does not get blurred out. Also shadow effects are reduced.

## 6 DISCUSSION

We have shown that our method produces virtual panoramic views at any position within the captured

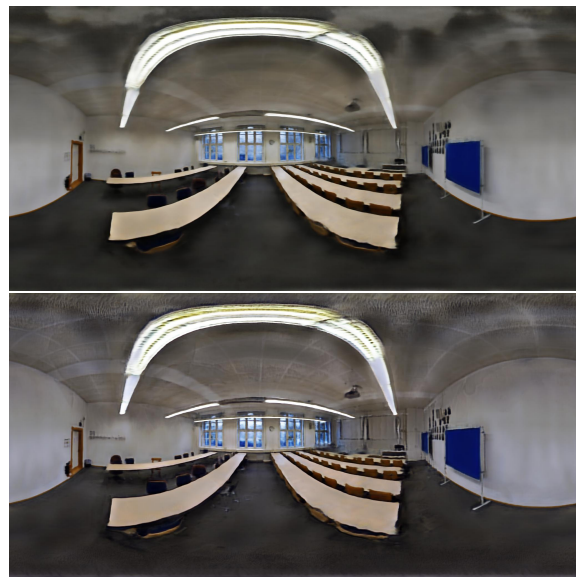


Figure 9: Comparison of a network prediction using MSE (top) and the panoramic perceptual loss (bottom) during training.

scene. The missing data is filled in a meaningful and consistent manner throughout all positions and orientations. This allows us to produce smooth transitions in animations. By introducing a perceptual panoramic loss the image quality can be enhanced. The network is also able to learn how certain things look and to correct wrong data accordingly. We further tested the behavior of the network on completely missing data as well as on data which it has not seen before. Lastly we have shown that transfer learning enables us to change the domain of the environment easily.

The virtual pictures also contain some artifacts. Especially on the walls, shadows are enforced and create some small darker clouded areas. The chairs are rendered darker, too. We attribute this to the weakly illuminated scene, hence the ceiling is affected the most. Some of those illumination artifacts also have their origin in the training data and can be seen in the photos, which show stripes on the wall. The network is able to reduce them, however not completely.

Sometimes the poles of the panoramas (top/bottom) are predicted with inconsistencies. This also comes from the training data as the poles of the real pictures exhibit the most distortion. Furthermore, the training data has some stitching artifacts, where, for instance, the edge of a table has a discontinuity. Some of the predicted images display the same error. We believe some of those problems can be mitigated by introducing a more advanced loss and training method.

## 7 CONCLUSION

In this paper, we describe a novel two stage process which allows for the generation of photo-realistic virtual views inside colored point clouds. Our approach has the advantage of being independent from the capturing method and allows for free movements inside the scene. High-resolution, realistic-looking panoramas are produced, which can be used for virtual reality applications. Since this can be done at any position and the process is locally and temporally consistent, rendering stereoscopic views is possible as well. However, the network still produces some unwanted shadow artifacts. In future work, an even better suited loss function may be found to reduce those errors. In addition, the rendering step and network should be modified such that it also handles smaller view ports with the same resolution. Further, it would be of interest to evaluate the performance in a consumer-centered study.

## ACKNOWLEDGEMENTS

This work is funded by Germany's Federal Ministry of Education and Research within the project KIMaps (grant ID 01IS20031C).

## REFERENCES

- Aliev, K.-A., Ulyanov, D., and Lempitsky, V. (2019). Neural Point-Based Graphics. *arXiv preprint arXiv:1906.08240*.
- Anguelov, D., Dulong, C., Filip, D., Frueh, C., Lafon, S., Lyon, R., Ogale, A., Vincent, L., and Weaver, J. (2010). Google street view: Capturing the world at street level. *Computer*, 43(6):32–38.
- Armeni, I., Sax, S., Zamir, A. R., and Savarese, S. (2017). Joint 2d-3d-semantic data for indoor scene understanding. *arXiv preprint arXiv:1702.01105*.
- Bui, G., Le, T., Morago, B., and Duan, Y. (2018). Point-based rendering enhancement via deep learning. *The Visual Computer*, 34(6-8):829–841.
- Dai, P., Zhang, Y., Li, Z., Liu, S., and Zeng, B. (2020). Neural point cloud rendering via multi-plane projection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7830–7839.
- Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee.
- Flynn, J., Neulander, I., Philbin, J., and Snavely, N. (2016). Deepstereo: Learning to predict new views from the world's imagery. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5515–5524.
- Fu, Z., Hu, W., and Guo, Z. (2018). Point cloud inpainting on graphs from non-local self-similarity. In *2018 25th IEEE International Conference on Image Processing (ICIP)*, pages 2137–2141. IEEE.
- Hedman, P., Alसान, S., Szeliski, R., and Kopf, J. (2017). Casual 3d photography. *ACM Transactions on Graphics (TOG)*, 36(6):234.
- Johnson, J., Alahi, A., and Fei-Fei, L. (2016). Perceptual Losses for Real-Time Style Transfer and Super-Resolution.
- Kerl, C., Sturm, J., and Cremers, D. (2013). Dense visual SLAM for RGB-D cameras. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2100–2106. IEEE.
- Lai, P. K., Xie, S., Lang, J., and Laquarère, R. (2019). Real-time panoramic depth maps from omni-directional stereo images for 6 dof videos in virtual reality. In *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 405–412. IEEE.
- Moezzi, S., Tai, L.-C., and Gerard, P. (1997). Virtual view generation for 3d digital video. *IEEE multimedia*, 4(1):18–26.
- Mordvintsev, A., Olah, C., and Tyka, M. (2015). Inceptionism: Going deeper into neural networks.
- Mur-Artal, R. and Tardós, J. D. (2017). Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262.
- NavVis GmbH (2019). <https://www.navvis.com>.
- Pfister, H., Zwicker, M., Van Baar, J., and Gross, M. (2000). Surfels: Surface elements as rendering primitives. In *Proceedings of the 27th annual conference on computer graphics and interactive techniques*, pages 335–342. ACM Press/Addison-Wesley Publishing Co.
- Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer.
- Sahay, P. and Rajagopalan, A. N. (2015). Geometric inpainting of 3d structures. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–7.
- Schubert, S., Neubert, P., Pöschmann, J., and Pretzel, P. (2019). Circular Convolutional Neural Networks for Panoramic Images and Laser Data. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 653–660. IEEE.
- Simonyan, K. and Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition.
- Sung, J. W., Jeon, Y. J., Jeon, B. M., and Lim, J. H. (2014). Virtual view image synthesis method and apparatus.