3D TEXTURE SUPER RESOLUTION VIA THE RENDERING LOSS

Rohit Ranade *Yangwen Liang* *Shuangquan Wang* *Dongwoon Bai* *Jungwon Lee*

SOC R&D, Samsung Semiconductor, Inc.

ABSTRACT

Deep learning-based methods have made significant impact and demonstrated superior performance for the classical image and video super-resolution (SR) tasks. Yet, deep learning-based approaches to super-resolve the appearance of 3D objects are still sparse. Due to the nature of rendering 3D models, 2D SR methods applied directly to 3D object texture may not be a good approach. In this paper, we propose a rendering loss derived from the rendering of a 3D model and demonstrate its application to the SR task in the context of 3D texturing. Unlike other literature on the 3D appearance SR, no geometry information of the 3D model is required during network inference. Experimental results demonstrate that incorporating the rendering loss during network training outperforms existing state-of-the-art methods for 3D appearance SR. Furthermore, we provide a new 3D dataset consisting of 97 complete 3D models for further research in this field.

Index Terms— deep learning, super resolution, texture atlas, computational imaging, 3D

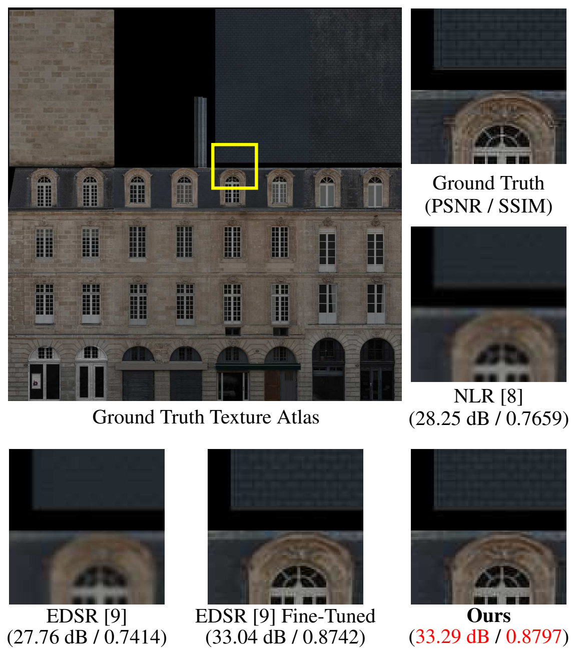


Fig. 1: $\times 2$ texture atlas super-resolution of our method compared with existing methods.

1. INTRODUCTION

Image super-resolution (SR) is a classical yet very important class of techniques in computer vision and image processing. It refers to the process of recovering high-resolution (HR) images from low-resolution (LR) images. In particular, to increase the realism of a reconstructed 3D object, SR techniques can be applied to its texture map to retrieve a detailed appearance of the 3D object. This high quality 3D content enjoys a wide range of real-world applications, such as cross reality (XR), video game, movie production, digital culture heritage preservation and so on.

In general, image SR problem is very challenging and inherently ill-posed. Image SR has been developed for more than three decades and has evolved from traditional model-based SR algorithms, cf. e.g. [1], to modern deep learning-based methods, cf. e.g. [2]. Traditional model-based SR techniques include but not limited to statistical methods [3], patch-based methods [4], and sparse representation methods [5], etc. In the deep learning SR domain, a variety of Convolutional Neural Networks (CNN)-based and Generative Adversarial Networks (GAN)-based methods with different architectures [6] and different loss functions [7] have been applied to tackle the image SR problem.

However, SR techniques for 2D images may not be directly applicable to 3D object appearance SR. This is not only due to the process of 3D reconstruction where only sequences of unaligned image captures may be available for reconstruction, but also due to the nature of rendering 3D models. During the rendering step of the 3D model, a standard renderer (such as an OpenGL-based renderer [10]) samples texture from the texture atlas and projects the 3D model onto the image plane at the viewing camera location to create a 2D image. The texture unwrapping process during rendering is independent of the actual size of the texture atlas, so that at a later time, a higher resolution or manipulated image may be re-used without the need to perform the texturing step again. This flexibility allows us to incorporate a conventional 2D image processing task such as SR into a 3D reconstruction pipeline - all the previous steps in the pipeline (i.e. point cloud registration, surface reconstruction and texture mapping) may still be performed with the input LR images while the generated object file (containing all the geometric information of the reconstructed 3D model) is re-used with the super-resolved texture atlas to create the final 3D model. In general, we can use any conventional 2D image processing algorithms in this context.

In this paper, we retrieve HR texture atlases for 3D objects via deep learning-based SR. Our main contributions are as follows: 1) We introduce the rendering loss for 3D texture manipulations/enhancement and incorporate it into a deep learning-based SR architecture to tackle 3D appearance SR; 2) Our 3D appearance SR does not require any 3D geometric information during inference although it is required during training; 3) We demonstrate that including our rendering loss during training is able to achieve the state-of-the-art 3D appearance SR performance and is able to outperform 2D SR methods applied directly to 3D object texture atlases; 4) A new 3D dataset consisting of 97 complete 3D models are available for the community for future researches. Qualitative results of our method as compared to other existing methods is presented in Figure 1 for texture atlas.

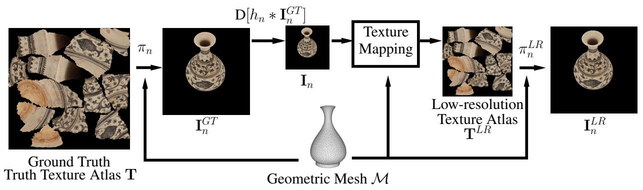
The rest of the paper is organized as follows: We summarize previous works for SR in the 3D scenario in Section 2, we introduce the rendering loss and describe how to incorporate it for training a SR network in Section 3, while experiments and results are given in Section 4. Conclusions are drawn in Section 5.

2. PREVIOUS WORKS

3D model rendering has already been applied for a variety of 3D tasks and more recently, been integrated into deep learning frameworks via differentiable rendering [11]. Differentiable rendering is a novel field which allows the gradients of 3D objects to be calculated and propagated through images. The applications of utilizing differentiable rendering in deep learning include 3D object recognition [12], human face deblurring and/or super-resolution [13], and mesh reconstruction [14] and so on.

Yet, deep learning-based approaches to super-resolve the appearance of 3D objects are very sparse. In [8], Li et al. propose a network based on well-known EDSR [9] network for texture atlases SR. Their main contribution is incorporating the normal maps along with the texture atlases as inputs. The authors of [8] claim that adding the normal maps helps in enhancing the texture and allows them to obtain better PSNR as compared to just EDSR. The proposed two networks, NLR and NHR, differ in terms of the resolution of the normal maps concatenated into the network layers (LR and HR respectively). The authors provide this comparison to back their claim that normal maps do indeed help in performance enhancement. However, during the typical inference step for a 3D object texture atlas, HR normal maps may not be available.

Unlike [8], our 3D appearance SR does not require the



$\mathrm{D}[h_n * \mathbf{I}_n^{GT}]$
π_n
Texture Mapping
π_n^{LR}
$\mathbf{I}_n^{GT}$
$\mathbf{I}_n$
Low-resolution Texture Atlas $\mathbf{T}^{LR}$
$\mathbf{I}_n^{LR}$
Ground Truth Truth Texture Atlas $\mathbf{T}$
Geometric Mesh $\mathcal{M}$

Fig. 2: The forward texture formation model.

3D object normal maps or any other 3D geometric information during inference. Instead, we propose an elegant training scheme to incorporate the 3D object geometric information during training. To the best of our knowledge, we are the first to extend the usage of differentiable rendering to the 3D texture atlas SR domain.

3. RENDERING LOSS

3.1. The Forward 3D Texture Formation Model

We first describe the forward 3D texture formation model [15, 16, 17] as shown in Figure 2. Let $\mathbf{T}$ be the ground truth texture atlas and $\mathcal{M}$ be the geometric mesh. For a camera location n, with known camera intrinsics and extrinsics, the operator π_n renders the geometric mesh with texture on to the image plane to obtain the image $\mathbf{I}_n^{GT} = \pi_n(\mathcal{M}, \mathbf{T})$. Assuming a simple degradation process of lens blur and downsampling, the degraded images are formed as $\mathbf{I}_n = \mathrm{D}[h_n * \mathbf{I}_n^{GT}]$, where h_n is a Gaussian kernel, $*$ is a convolution operator and $\mathrm{D}[\cdot]$ is a downsampling operator. The LR images $\mathbf{I}_n$ are then used to texture the geometric mesh using any existing methods [18, 19, 20] to produce the LR texture atlas $\mathbf{T}^{LR}$. The geometric model ($\mathcal{M}$) can now be rendered with the degraded texture atlas at any camera location n to obtain the LR images $\mathbf{I}_n^{LR} = \pi_n^{LR}(\mathcal{M}, \mathbf{T}^{LR})$.

3.2. Rendering Loss

We can now define the rendering loss as

$$\mathcal{L}_{rendering} = \rho_R(\mathbf{I}_n^{LR}, \mathbf{I}_n^{GT}) \,, \tag{1}$$

where $\rho_R(\cdot)$ can be $\|\cdot\|_p$ (such as L1-norm or L2-norm) or SSIM or any differentiable objective metric comparing two images. Similarly, for N rendered images, the average rendering loss is

$$\mathcal{L}_{rendering} = \frac{1}{N} \sum_{n=0}^{N-1} \rho_R(\mathbf{I}_n^{LR}, \mathbf{I}_n^{GT}) \,. \tag{2}$$

Next, we describe how we incorporate the rendering loss in a typical 2D supervised deep learning setting for the SR task. In a typical 2D deep learning-based image SR system, a convolutional neural network (CNN) $f(\boldsymbol{\theta}, \cdot)$ characterized by parameters $\boldsymbol{\theta}$ is presented a LR image $\mathbf{T}^{LR}$. The network

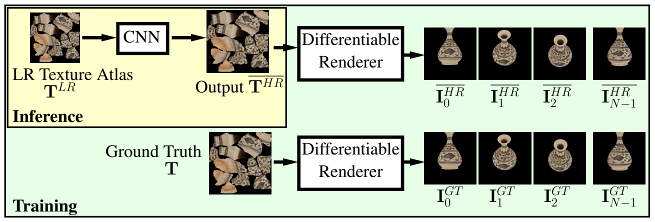

CNN
LR Texture Atlas $\mathbf{T}^{LR}$
Output $\overline{\mathbf{T}^{HR}}$
Inference
Differentiable Renderer
$\overline{\mathbf{I}_0^{HR}}$ $\overline{\mathbf{I}_1^{HR}}$ $\overline{\mathbf{I}_2^{HR}}$ $\overline{\mathbf{I}_{N-1}^{HR}}$
Ground Truth $\mathbf{T}$
Differentiable Renderer
$\mathbf{I}_0^{GT}$ $\mathbf{I}_1^{GT}$ $\mathbf{I}_2^{GT}$ $\mathbf{I}_{N-1}^{GT}$
Training

Fig. 3: Incorporating the rendering loss into a 2D supervised deep learning framework.

output $\overline{\mathbf{T}^{HR}} = f(\boldsymbol{\theta}, \mathbf{T}^{LR})$ is then compared with the corresponding HR or ground truth image $\mathbf{T}$. The loss is then back-propagated through the network and the parameters are updated according to the particular optimization scheme. For this case, the loss function is given as

$$\mathcal{L}_{atlas} = \rho_A(\overline{\mathbf{T}^{HR}}, \mathbf{T}) \ , \tag{3}$$

where $\rho_A(\cdot)$ can also be $\|\cdot\|_p$ (such as L1-norm or L2-norm) or SSIM or any differentiable objective metric comparing two images.

Our proposed framework leverages the ability of a differentiable renderer to propagate gradients backward from rendered images to the different aspects of model information, allowing the renderings to be incorporated into deep learning pipelines [11]. A standard renderer takes the 3D model information, e.g. geometry, textures, cameras and lights, as inputs and outputs a projected image on the image plane of the camera locations. The differentiable renderer allows us to relate the 2D loss from the pixels of the rendered image back to the properties of the 3D shape such as the positions of mesh vertices. The proposed framework is shown in Figure 3. Here, we first pass the LR texture atlas $\mathbf{T}^{LR}$ through the CNN $f(\boldsymbol{\theta}, \cdot)$ to obtain the network output $\overline{\mathbf{T}^{HR}}$. The ground truth atlas $\mathbf{T}$ and $\overline{\mathbf{T}^{HR}}$ are then rendered using a differentiable renderer to render images $\mathbf{I}_n^{GT}$ and $\overline{\mathbf{I}_n^{HR}}$ at N different camera locations, respectively. The camera locations are pre-determined using a particular trajectory. $\mathcal{L}_{rendering}$ is calculated between the rendered pairs and back-propagated through the network via the differentiable renderer. The network parameters are then updated according to the optimization scheme. Note the renderings are only required for training. During inference, the network still functions as a regular CNN which is provided a 2D texture atlas as an input to reconstruct a super-resolved image as output.

We can also combine the losses from Equations 2 and 3 as

$$\mathcal{L} = \lambda \cdot \mathcal{L}_{atlas} + (1 - \lambda) \cdot \mathcal{L}_{rendering} \ , \tag{4}$$

where λ is a hyper-parameter for different losses weights. In Section 4, we observe network models trained by the combination losses outperforms the current state-of-the-art models.

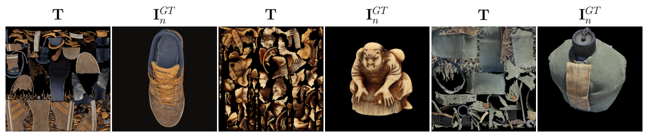


Fig. 4: An overview of our dataset.

4. EXPERIMENTS

4.1. Dataset

Existing datasets either had just the texture atlases, geometric meshes or HR images/videos or image-depth pairs [8, 21, 22]. We were unable to find complete large-scale 3D dataset consisting of ground truth geometric and texture information.

Thus, we created a 3D dataset from publicly and freely available 3D models from Sketchfab [23]. The models were selected to provide a mixture of real world and simulated texture along with irregular object shapes. Our dataset consists of 97 3D models with texture atlases of resolutions with width (W) and height (H), $W = H = [1024, 2048, 4096]$ and geometric models ranging from 100 to 70000 triangular faces. An overview of our dataset is shown in Figure 4. The texture atlas is shown on the left side while a rendered image from a particular camera location is shown on the right, for a few different models. For comparison purposes, we also incorporated 3DASR dataset from [8] in our experiments.

4.2. Experiment Setup

For the CNN backbone, we re-use the EDSR architecture [9] as is while the differentiable renderer from [11] is used to incorporate the rendering loss into the framework. For the camera locations, we adopted Fibonacci lattice. For any given number of points, Fibonacci lattice is a mathematical optimal packing on a sphere, where the area represented by each point is almost identical [24]. Since the 3D models have different shapes, searching the optimal camera locations for best performance does not seem feasible. Therefore, the Fibonacci lattice is a reasonable choice for our network training. In detail, the Cartesian coordinates for K camera locations are $(x_k, y_k, z_k) = (r \sin\theta_k \cos\phi_k, r \sin\theta_k \sin\phi_k, r \cos\theta_k)$ where r is the radius of sphere, $k \in \{1, 2, ..., K\}$, the elevation angle $\theta_k = \arccos(2(k-1)/(K-1) - 1)$, and the azimuth angle $\phi_k = \pi(3 - \sqrt{5})k$. The elevation and azimuth angles are controlled by the Fibonacci lattice while the radius parameter r is set heuristically. Other distribution of camera locations on the sphere can also be adopted, cf. e.g. [25]. The camera is facing towards the center of the 3D model.

During training, we start with pre-trained EDSR models and then fine-tune the models for our dataset for different values of the hyper-parameters, i.e. λ, N and camera locations. For all sets of hyper-parameters, fine-tuning on our dataset is performed with the loss set to $\mathcal{L}$ and $\rho_{atlas} = \rho_{rendering} = \|\cdot\|_1$ i.e. L1-norm, for a 100 additional epochs and with the

Table 1: Objective metrics for different scale factors and hyper-parameters for rendered images with our dataset. FS and RS denote Fixed Sampling and Random Sampling, respectively. Fine-tuned EDSR is equivalent to $(1.0, -, -)$.

Method (λ, N, Sampling)	×2		×4	
	PSNR	SSIM	PSNR	SSIM
Bi-cubic	29.93	0.9021	25.87	0.8535
$(0.0, 1$, FS)	33.64	0.9427	28.29	0.8993
$(0.0, 1$, RS)	33.70	0.9398	28.92	0.9050
$(0.0, 4$, FS)	33.76	0.9442	28.80	0.9039
$(0.0, 4$, RS)	33.76	0.9445	29.21	0.9051
$(0.0, 40$, FS)	33.77	0.9440	28.87	0.9039
$(0.0, 40$, RS)	34.03	0.9443	29.35	0.9053
$(0.50, 40$, RS)	34.65	0.9465	29.64	0.9056
$(0.75, 40$, RS)	**34.95**	**0.9474**	**29.96**	**0.9058**
$(1.0, -, -)$	34.44	0.9449	29.70	0.9052

learning rate set to 0.00005. For camera locations in the Fibonacci lattice trajectory, we first sample $K = 64$ number of camera locations. Next, for each N we either fix the camera locations to the first $N \in \{1, 2, ..., K\}$ or randomly sample N locations from K for each model, each epoch. The size of the rendered images is fixed to 640 × 640 for both $\mathbf{I}_n^{GT}$ and $\mathbf{I}_n^{LR}$. To perform objective comparisons, we compute PSNR and SSIM metrics on the texture atlases from the test set.

4.3. Results

In the first set of experiments, we check whether propagating the rendering loss back through the network to update parameter weights actually works and whether it improves PSNR as well as subjective image quality. For this, we set $\lambda = 0.0$ (i.e. $\mathcal{L} = \mathcal{L}_{rendering}$). To check the impact of the number of rendered images during training, we vary $N = [1, 4, 40]$. In Table 1, the rows corresponding to $\lambda = 0.0$ show the average objective metrics calculated over the rendered image for this case. As N increases, both PSNR and SSIM improve but eventually plateaus. In addition, random sampling of camera locations provides better performance than fixed sampling.

In the next set of experiments, we vary the value of $\lambda = [0.0, 0.5, 0.75, 1.0]$ while fixing $N = 40$. Note that for $\lambda = 1.0$, the combined loss equation reduces to $\mathcal{L} = \mathcal{L}_{atlas}$ i.e. the conventional 2D image loss. This would be equivalent to fine-tuning the pre-trained EDSR models on our dataset with just the 2D image loss. The average objective metrics over the test set for the different cases above, calculated over the rendered images are given in Table 1. Here, we observe that the model corresponding to $\lambda = 0.75, N = 40$, Random Sampling provides the best performance. It demonstrates that introducing rendering information into the training process does indeed help in improving performance.

Finally, we compare the performance of our best trained model (i.e. $\lambda = 0.75, N = 40$, Random Sampling) against the pre-trained models NLR from [8] on their 3DASR dataset. The average values over the dataset for the different methods above are given in Table 2. We observed significant performance gains of our network with rendering loss over NLR [8], which is considered the state-of-the-art network for the 3D appearance SR. Subjective comparisons for some of the atlases in the set are shown in Figure 5. As we can observe, our proposed network successfully reconstruct the detailed textures and edges in the ground truth images and exhibit much better SR outputs compared with the previous works.

Table 2: Objective metrics comparing different models for texture atlases on 3DASR dataset [8].

Method	×2		×4	
	PSNR	SSIM	PSNR	SSIM
Bi-cubic	24.72	0.8241	19.46	0.6797
Pre-trained EDSR [9]	21.65	0.7643	15.99	0.6008
NLR [8]	26.54	0.8694	23.78	0.7991
Ours	**28.43**	**0.9057**	**25.76**	**0.8503**

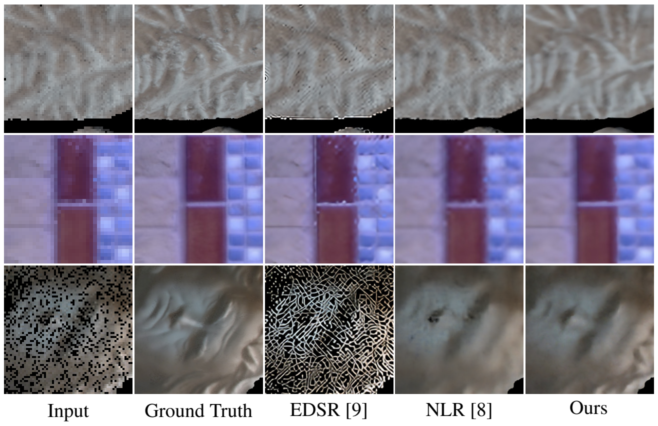


Fig. 5: Subjective comparison for model outputs (×4) with an input from 3DASR dataset [8].

5. CONCLUSION

In this paper, we have introduced the rendering loss and demonstrated its application for texture manipulation in a deep learning framework in the 3D scenario. For the SR task, we have shown that models trained on the rendering loss are able to perform almost as well as compared to a conventional loss. We also show that our model outperforms the current state-of-the-art in the 3D appearance SR. The rendering loss can be applied for any texture manipulation task in the 3D setting. Future work on this topic includes extending the application of this loss to many other tasks, for example filling missing faces in 3D models.

6. REFERENCES

[1] Kamal Nasrollahi and Thomas B. Moeslund , "Super-resolution: A comprehensive survey," *Machine Vision and Applications*, vol. 25, 2014.

[2] Saeed Anwar, Salman Khan, and Nick Barnes, "A deep journey into super-resolution: A survey," *ACM Comput. Surv.*, vol. 53, no. 3, May 2020.

[3] Z. Xiong, X. Sun, and F. Wu, "Robust web image/video super-resolution," *IEEE Trans. Image Process.*, vol. 19, no. 8, pp. 2017–2028, 2010.

[4] Daniel Glasner, Shai Bagon, and Michal Irani, "Super-resolution from a single image," in *Int. Conf. Comput. Vis.* IEEE, 2009, pp. 349–356.

[5] J. Yang, J. Wright, T. S. Huang, and Y. Ma, "Image super-resolution via sparse representation," *IEEE Trans. Image Process.*, vol. 19, no. 11, pp. 2861–2873, 2010.

[6] Ben Niu, Weilei Wen, Wenqi Ren, Xiangde Zhang, Lianping Yang, Shuzhen Wang, Kaihao Zhang, Xiaochun Cao, and Haifeng Shen, "Single image super-resolution via a holistic attention network," in *Eur. Conf. Comput. Vis.* 2020, pp. 191–207, Springer International Publishing.

[7] Adrian Bulat and Georgios Tzimiropoulos, "Super-fan: Integrated facial landmark localization and super-resolution of real-world low resolution faces in arbitrary poses with gans," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2018, pp. 109–117.

[8] Yawei Li, Vagia Tsiminaki, Radu Timofte, Marc Pollefeys, and Luc Van Gool, "3D appearance super-resolution with deep learning," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2019, pp. 9671–9680.

[9] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee, "Enhanced deep residual networks for single image super-resolution," in *IEEE Conf. Comput. Vis. Pattern Recog. Worksh.*, 2017, pp. 136–144.

[10] Dave Shreiner, Graham Sellers, John M. Kessenich, and Bill M. Licea-Kane, *OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 4.3*, Addison-Wesley Professional, 8th edition, 2013.

[11] Shichen Liu, Tianye Li, Weikai Chen, and Hao Li, "Soft rasterizer: A differentiable renderer for image-based 3d reasoning," *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2019.

[12] Ze Yang and Liwei Wang, "Learning relationships for multi-view 3d object recognition," in *Int. Conf. Comput. Vis.*, 2019, pp. 7505–7514.

[13] Xiaobin Hu, Wenqi Ren, John LaMaster, Xiaochun Cao, Xiaoming Li, Zechao Li, Bjoern Menze, and Wei Liu, "Face super-resolution guided by 3d facial priors," in *Eur. Conf. Comput. Vis.* Springer, 2020, pp. 763–780.

[14] Angjoo Kanazawa, Shubham Tulsiani, Alexei A Efros, and Jitendra Malik, "Learning category-specific mesh reconstruction from image collections," in *Eur. Conf. Comput. Vis.*, 2018, pp. 371–386.

[15] B. Goldluecke and D. Cremers, "Superresolution texture maps for multiview reconstruction," in *Int. Conf. Comput. Vis.*, 2009, pp. 1677–1684.

[16] Bastian Goldlücke, M. Aubry, K. Kolev, and Daniel Cremers, "A super-resolution framework for high-accuracy multiview reconstruction," *Int. J. Comp. Vision*, vol. 106, pp. 172–191, 2014.

[17] V. Tsiminaki, J. Franco, and E. Boyer, "High resolution 3d shape texture from multiple videos," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2014, pp. 1502–1509.

[18] Victor Lempitsky and Denis Ivanov, "Seamless mosaicing of image-based texture maps," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2007, pp. 1–6.

[19] Zhaolin Chen, Jun Zhou, Yisong Chen, and Guoping Wang, "3d texture mapping in multi-view reconstruction," in *International Symposium on Visual Computing*. Springer, 2012, pp. 359–371.

[20] Michael Waechter, Nils Moehrle, and Michael Goesele, "Let there be color! — Large-scale texturing of 3D reconstructions," in *Eur. Conf. Comput. Vis.* 2014, Springer.

[21] Andreas Ley, Ronny Hansch, and Olaf Hellwich, "Syb3r: A realistic synthetic benchmark for 3d reconstruction from images," in *Eur. Conf. Comput. Vis.*, 2016, pp. 236–251.

[22] Thomas Schops, Johannes Schonberger, Silvano Galliani, Torsten Sattler, Konrad Schindler, and Marc Pollefeys, "A multi-view stereo benchmark with high-resolution images and multi-camera videos," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2017, pp. 2538–2547.

[23] Sketchfab website, *https://sketchfab.com*, 2020.

[24] A. González, "Measurement of areas on a sphere using fibonacci and latitude–longitude lattices," *Mathematical Geosciences*, vol. 42, pp. 49–64, 2009.

[25] D. P. Hardin, T. Michaels, and E. B. Saff, "A comparison of popular point configurations on $\mathbb{S}^2$," *Dolomites Res. Notes Approx.*, vol. 9, pp. 16–49, 2016.