CGI2018 manuscript No.
(will be inserted by the editor)

Point Based Rendering Enhancement via Deep Learning

Giang Bui · Truc Le · Brittany Morago · Ye Duan

December 16, 2018

Abstract Current state-of-the-art point rendering techniques such as splat rendering generally require very high resolution point clouds in order to create high quality photo realistic renderings. These can be very time consuming to acquire and oftentimes also require high-end expensive scanners. This paper proposes a novel deep learning based approach that can generate high resolution photo realistic point renderings from low resolution point clouds. More specifically, we propose to use co-registered high quality photographs as the ground truth data to train the deep neural network for point based rendering. The proposed method can generate high quality point rendering images very efficiently and can be used for interactive navigation of large scale 3D scenes as well as image-based localization. Extensive quantitative evaluations on both synthetic and real datasets show that the proposed method outperforms state-of-the-art methods.

1 Introduction

Large scale 3D scene modeling and visualization are fundamental building blocks for active research fields such as Virtual Reality, Augmented Reality, Autonomous Driving, etc. Despite decades of research in this area, how to model the real world efficiently and accurately remains a very challenging task.

The advancement of 3D depth sensors, such as LIDAR (Light Detection And Ranging) scanners, has provided an effective alternative to the traditional CAD-based and image-based approaches for 3D modeling. When rendering 3D reconstructions directly from an acquired point cloud, the discrete resolution leads to either holes in the reconstruction, or, with more advanced hole-filling point-based rendering methods, leads to low-frequency appearance, as no detail information is available between the point samples. One existing way to circumvent this problem is texture splatting, but that technique requires access to high-resolution camera images taken alongside the (coarser) 3D scan.

In this paper, we propose an alternative approach: the adaptation of a CNN (Convolutional Neural Network) borrowed from existing work on image super-resolution, to take a coarse point-based rendering and up-sample it to a plausible high-resolution image. We demonstrate that the proposed method can generate high quality point rendering images very efficiently and can be used for interactive navigation of large scale 3D scenes as well as image based localization with significantly reduced overhead.

The rest of the paper is structured as follows. Section 2 discusses the related work. Section 3 describes the proposed method. The experimental results are presented in Section 4 followed by our conclusion in Section 5.

2 Related Works

Point-based splat rendering is a technique for rendering a smooth surface with approximated linear piece-wise splats. In order to cover the gaps between the points, a circular disc is assigned to each sample point with a normal vector n_i and radius r_i which are computed based on local geometry. The surface splats serve as linear approximation to a smooth surface if constructed properly. Zwicker et al. [52] proposed a signal-theorectically motivated approach to reconstruct smooth and correctly band-limitted images from splats. Botsch et al. introduced a multi-pass rendering approach which averages colors and normals of overlapping splats [2]. However, those approaches may generate blurred images due to gradient suppression. Sibbing et al. [38] pro-

Giang Bui, Truc Le, Ye Duan
University of Missouri, Columbia

Brittany Morago
University of North Carolina Wilmington

posed two post-processing methods to further improve the results of splat rendering. *Intensity Completion* preserves the intensities of the non-gap pixels, and automatically finds color transitions between the non-gap and gap pixels while *Gradient Completion* requires additional gradient information from intermediate images to reconstruct image more faithfully. Moreover, when additional texture images are available along with projection matrices, they also proposed *Texture Splat Rendering* which improves the rendered image by propagating as much information as possible from a set of texture images. That is to say, instead of using a single color for a whole splat, they used a projection matrix to project the fragments of the splat into a texture image and obtain the color information. However, obtaining the projection matrices of texture images requires non-trivial 2D-3D registration, and storing all the texture images together with a large point cloud requires a lot of memory. Similar to *Intensity Completion* and *Gradient Completion* of [38], we also propose to conduct post-processing image enhancement based on the 3-pass splat rendering results of [2]. Compared with *Intensity Completion* and *Gradient Completion* methods, our results are much better. Compared with *Texture Splat Rendering*, our method does not need to store extra texture images while still generating better image quality. Fig. 1 shows a side-by-side comparisons between the proposed method and the photograph, 3-pass rendering [2], *Intensity Completion*, *Gradient Completion*, *Texture Splat Rendering*. More examples are shown in Fig. 4 and 5.

Image Super Resolution (SR) is a set of methods to reconstruct a high-resolution image from either a single or multiple images. The image super-resolution methods can be classified into 3 categories: interpolation-based method, reconstruction-based method, and example-based method. According to [48], example-based methods show superior performance by modeling the mapping from low-resolution (LR) to high-resolution (HR) image patches. This map then is applied to a new LR image to obtain the most likely HR output [6,4,51]. These methods either exploit internal similarities of the same image [5,10,11,15] or learn mapping functions from external training image pairs [51,47,6,17, 21,36,42,43,49]. Glasner et al. [11] proposed a method to combine example-based SR constraints and classical SR constraints in a unified framework which allows for inter-patch search. The parent of the search results is copied to an appropriate location in the high-resolution image. Freedman et al. [10] proposed a method to adopt a local search by using the multi-step coarse-to-fine algorithm. Since the extracted patches from multiple scale images may not always be sufficiently expressive to cover the textural appearance variations, Huang et al. [15] extended the self-similarity based SR method by allowing geometric variations in patch searching scheme. The sparse-coding-based methods [51,47] are the representative external example-based SR methods. Yang et al. [51] proposed a super-resolution algorithm that learns a pair of over-complete dictionaries with the assumption that both the input and output patches have the same mixing coefficients of their corresponding dictionary.

Besides traditional super-resolution methods [6,4,50,51, 47,10,11], convolutional neural networks (CNNs) have recently demonstrated remarkable performance in the image SR field thanks to their ability to exploit the contextual information through their receptive field to recover missing high frequency components. Inspired by traditional sparse coding based SR [51], Dong et al. [8] proposed a shallow $SRCNN$ consisting of three layers which correspond to patch representation, non-linear mapping, and reconstruction. Kim et al. [20] proposed a deeply recursive CNN that utilizes a very large image context compared to previous SR methods with a single recursive layer. Kim et al. [19] proposed a residual network structure with 16 layers by concatenating 64 3×3 kernels. For these approaches, the SR methods work directly on the high-resolution (HR) space by first applying a simple interpolation method (e.g. bicubic interpolation) to up-sample the LR image to the desired size, then feeding it through a deep neural network to obtain a visually satisfying result. While most of the deep learning based super image super resolution (SISR) methods work directly on the HR space, other methods cope with the LR space and only go back to the HR space at the very last several layers. Bishop et al. [37] presented the first CNN in which extracted feature maps are performed in LR space and an efficient sub-pixel convolution is used to upscale the final LR feature maps to the HR image. Improving over the $SRCNN$, Dong et al. [9] introduced a fast version of SRCNN, named $FSRCNN$, which can reach up to 43 fps with a generic GPU. Ledig et al. [24] proposed SRResNet by employing the ResNet architecture from He at al. [14] that successfully solves time and memory issues with good performance. By removing the batch normalization layers from SRResNet, Lim et al. [27] showed superior performance over the state-of-the-art methods and won the NTIRE2017 Super-Resolution Challenge [41].

Recently, there has been a lot of interest in Generative Adversarial Networks (GANs) [12] which can learn to produce samples that imitate the dataset according to the discriminator network. Mathieu et al. [31] supplemented a squared error loss with both GAN and image gradient-based similarity to improve image sharpness of video prediction. The ability to produce high-quality images of GANs is also demonstrated in the works of Denton et al. [7] and Radford et al. [34]. In the work of Johnson et al. [18] and Ledig et al. [24], the authors proposed a conceptual loss function based on the VGG network [39] obtained results that are more convincing than the ones obtained with traditional low level mean square error (MSE). In this paper, we employ the GAN net-

work for point based rendering enhancement, which will be explained in more details in the following section.

3 Proposed Method

Fig. 2 shows the overview of our proposed method. The algorithm takes a splat image as input and feeds it through a neural network to obtain a high quality image. As shown in Fig. 1, the deep learning result is almost indistinguishable from the photograph taken by cameras. Beside of filling holes, it can generate more natural results while *splat rendering* has blurring effect, *intensity completion* and *gradient completion* can not fill the big holes due to the lack of constrains.

3.1 Input data

There are several ways to obtain a color point cloud of a 3D reconstructed scene. A traditional approach is to use SfM (Structure from Motion) [3,13,40] to generate a point cloud from a collection of densely sampled images. The generated point cloud can be used for many tasks such as image localization [16,26], classification and segmentation [33,35,45, 29]. Another common approach is to use LIDAR laser scanners that produce highly accurate coordinate measurements. The RGB-color information can either come from imagery collected at the time of the LIDAR survey. By placing a laser scanner at several sample locations, a large-scale 3D color point cloud can be obtained with relative ease. In this paper, we focus on LIDAR point cloud.

3.2 Splat rendering

Point-based splat rendering has been proven to be a flexible and efficient technique for rendering 3D objects due to its simplicity. The key idea is to approximate a surface by piece-wise planar ellipses, or splats, in the object space and render them to the image space. This technique is widely used in the literature and often needs an associated normal, a color, and a radius r_i for each point p_i to render the points as small discs. That information can be obtained from neighbors around points. There are two common approaches to obtain this information, radius search and k-nearest neighbors search (KNN). Both can be efficiently retrieved using a k-d tree. However, radius search is not suitable for the case of LIDAR data due to its non-uniform distribution. The points tend to be very dense on surfaces close to the scanning location and become sparse on ones further away. In contrast to radius search, KNN is a naturally adaptive method better suited for this situation. Once the neighborhood information is obtained, the point cloud's normals can be approximated by Principal Component Analysis (PCA) which selects an eigenvector corresponding to the smallest eigenvalue of the covariance matrix [23].

Next, we estimate a radius for each point. Since the point normal is assigned to that of the plane it belongs to, we still have to determine the radius r so that we can render a gap-free surface. We do not want it to be too big as this will likely require many overlapping regions be rendered which is computationally inefficient. We propose a heuristic approach which is similar to [38]. First, for each sample point p_i with corresponding normal n_i, we determine the k nearest neighbors using KNN and define R_i as the furthest distance from a neighbor to the sample point. Therefore, the sphere at p_i with radius R_i covers the whole k nearest neighbors. In the second step, we form a circle at p_i with radius R_i, normal n_i and project all neighbors whose normals are consistent (the angle deviation is less than 5°) to that of the sample point onto it. Lastly, we divide the circle of projected points into 12 sectors and for each non-empty sector, we only keep the closest projection point. The radius assigned to each point is guaranteed that any point is covered by at least one of splats of other points. Since the point distribution is quite uniform, scaling that radius with smaller factor can reduce the effect of blurring and still guarantee the gap between any point is covered. For that reason, we set the splat radii r_i to be the scale factor of 0.7 of the maximum distance of those points. Although computing the splat radius depends on how many of the nearest neighbors are considered, we have found that it is not sensitive to parameter k. Empirically we set $k = 30$ for all models in the dataset.

Since each splat is a solid color (determined from its center point), sharp edges may exist between splats. Following [38], we implement a 3-pass rendering technique to blend the color and normal between neighboring splats. In the first pass called *visibility splatting*, the splats are rendered without color information in order to fill the depth buffer. In the second *blending* pass, after the object is slightly shifted towards the viewer by ϵ (0.05), a simple depth test is used to remove all the splats that are far behind the visible surfaces. For each virtual image pixel, the second pass sums up the colors and normals of the splats that lie in the proximity of the visible surface and perspectively project onto the respective pixel. Finally, the *normalization* pass normalizes the sum-up colors and normals by dividing them by the accumulated weights. Although the 3-pass rendering technique can give visually pleasant rendering images, it smooths out the gradient information and hence blurs the image due to the use of blending techniques. In what follows, we will describe the use of a deep-learning based approach as a post-processing step to alleviate these unwanted effects.

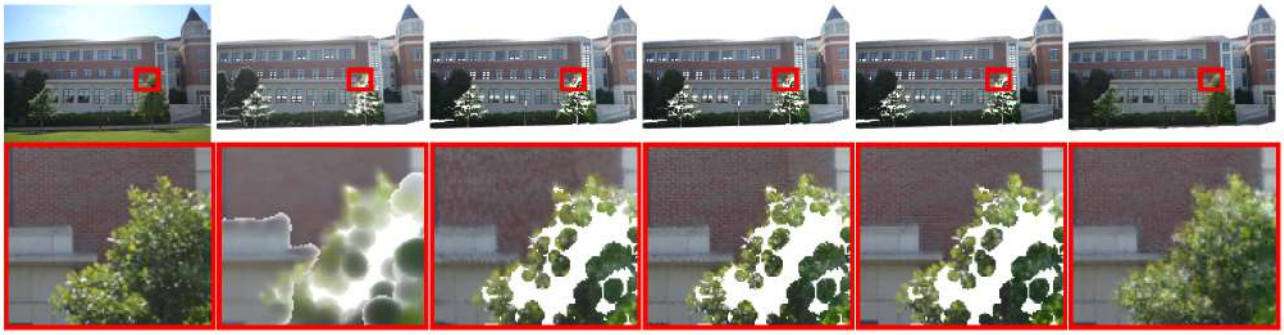

Fig. 1: From left to right, different rendering techniques: (1) Photograph. (2) Splat rendering. (3) Intensity completion. (4) Gradient completion. (5) Texture splat rendering. (6) (Ours) Deep-Learning based rendering with splat rendering of (2) as input.

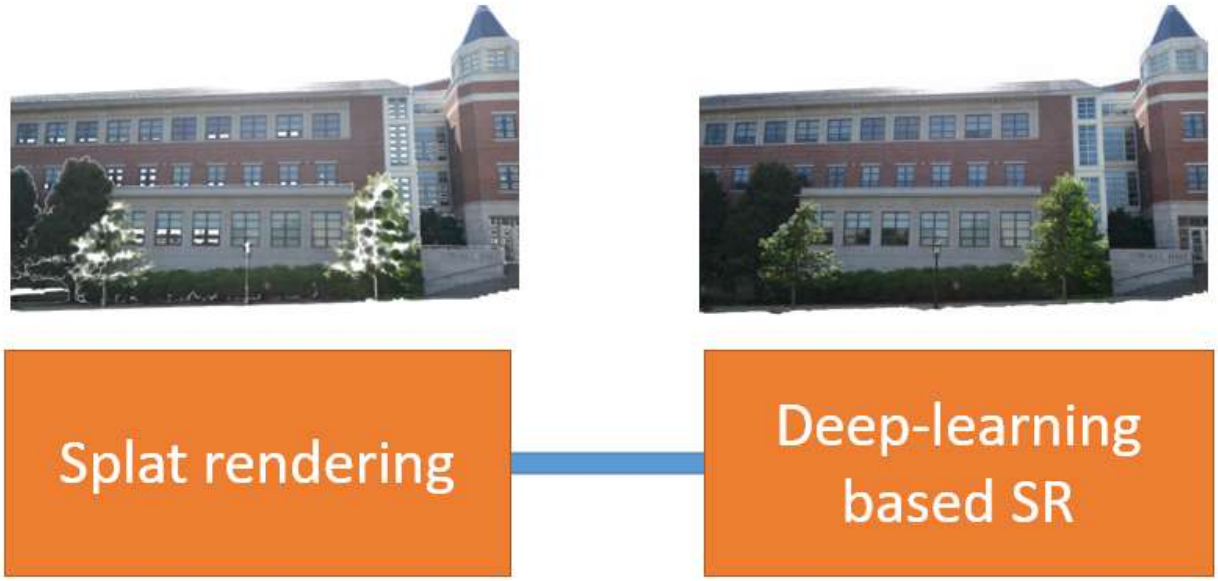

Fig. 2: Proposed method pipeline.

3.3 Deep-Learning based Super-Resolution with Generative Adversarial Network

Following Goodfellow et al. [12], in the context of genarative adversarial networks, we solve the adversarial min-max problem:

$$\min_{\theta_G}\max_{\theta_D} \quad E_I^{HR}\sim p_{train}(I^{HR})\big[logD_{\theta_D}(I^{HR})\big] \quad + \\ E_I^{LR}\sim p_G(I^{LR})\big[log(1-D_{\theta_D}(G_{\theta_G}(I^{LR}))\big] \tag{1}$$

where G_{θ_G} and D_{θ_D} are generator and discriminator networks which are parameterized by θ_G and θ_D respectively. In our problem, the generator network G is the network to predict a HR image given a LR image. The general idea behind this formulation is that it allows one to train a generative model G with the goal of fooling a differentiable discriminator D that is trained to distinguish super-resolved images from real images. That is to say, with this approach our generator can learn to predict outputs that are highly similar to real images and thus difficult to classify by D.

3.3.1 Generator network structure

Our network, named deep network for super-resolution with deeply supervised nets (*SRDSN*), is illustrated in Fig. 3 (top row). The network takes a splat rendering of the LR image as input and progressively predicts the intermediate HR image. We supervise all the intermediate results to alleviate the effect of vanishing/exploding gradients. Our network contains two parts: (i) cascade network and (ii) deeply supervised network.

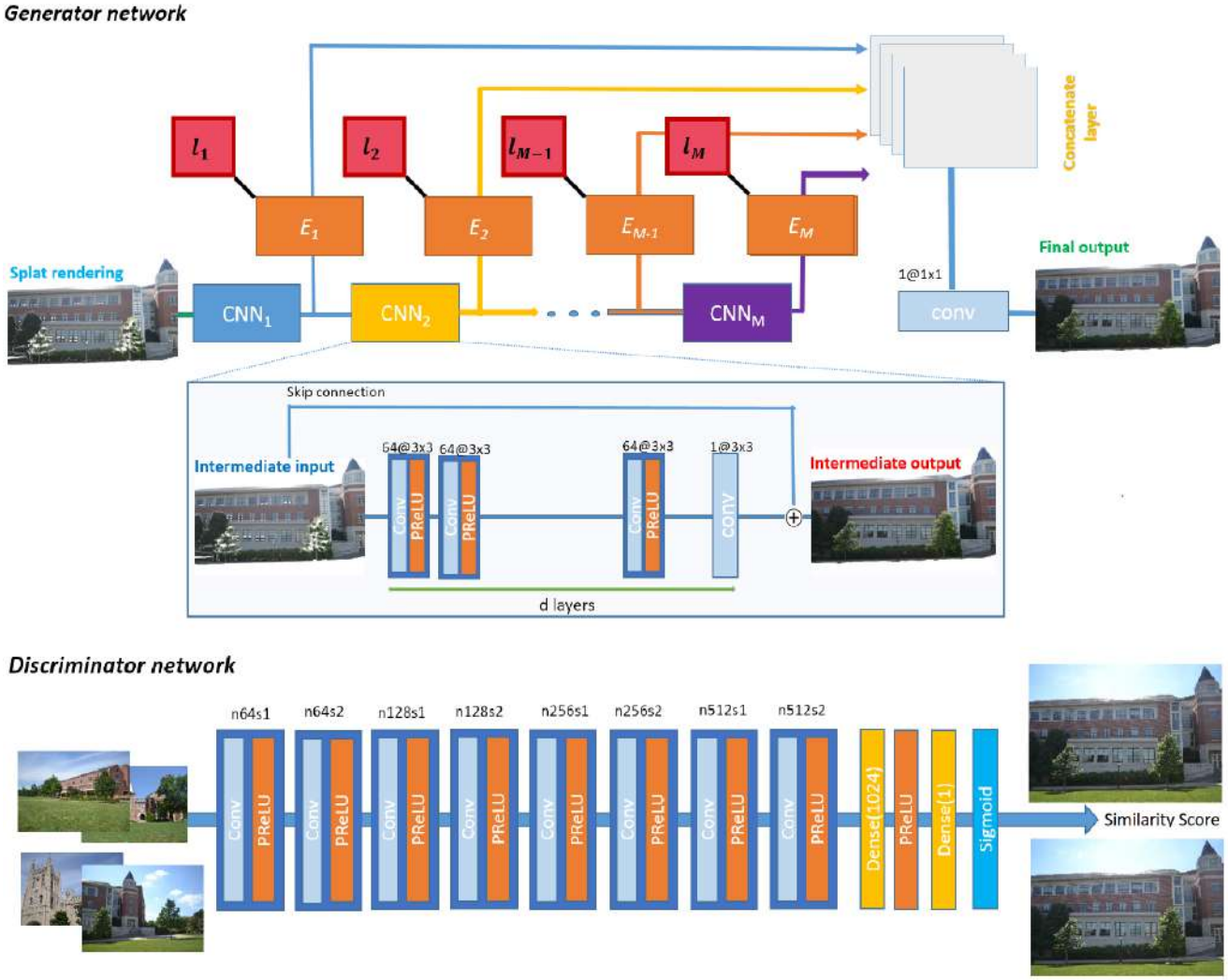

Fig. 3: SRDSN-GAN. *(Top)* Generator Network Structure. The network consists of multiple blocks with deep supervision, every intermediate result is penalized by a Euclidean loss function l. The layout of each network block has a corresponding number of feature maps followed by kernel size indicated for each convolutional (e.g. 64@3x3 stands for 64 3x3 kernels). Our final network consists of four blocks ($M = 4$) and 20 layers ($d = 20$) within each block. *(Bottom)* Discriminator Network Structure with corresponding number of feature maps (n) and stride (s). The network takes a set of natural images to measure the similarity of the generator output image and ground truth image. For the SRDSN-GAN network, the training loss (Eq. 5) is computed by both the Generator Network and the Discriminator Network.

Cascade network: The network has multiple similar structure blocks. Each block, consisting of d convolutional layers followed by a PReLU (Parametric ReLUs) layer except for the last one, takes an image (1 or 3 channels) to produce an intermediate HR image. This intermediate result is the input to the next block and is supervised by a deeply supervised network.

Deeply supervised network: We supervise all the outputs of the network blocks using a deeply supervised nets (DSN) structure [25]. The DSN can be considered as a network regularization that informs intermediate layers about the final objective, rather than relying on the final layer to back-propagate the information to its predecessors. A similar idea of supervising intermediate layers for a convolutional neural network can be found in [46] and [20].

Generator output: Denote x as the input of low-resolution image, and $\hat{y}_m, m = 1, 2, \ldots, M$, as predicted M intermediate outputs. The output of the generator network is averaged over all the intermediate outputs $\hat{y} = \sum_{i=1}^{M} \alpha_i \hat{y}_m$ with α_i indicates the relative importance of the intermediate output, e.g. setting $\alpha_m = 2 \leq m < M$ makes the model turn into a predictor with only a single output at the top.

3.3.2 Adversarial Network Architecture

For the discriminator network, we modify the network proposed by Ledig [24] to simplify the network architecture. We remove the batch normalization layers from the network as Lim et al. [27] presented in their work. It is better to remove them since batch normalization layers get rid of range flexibility from networks by normalizing the features. We also replace the $LeakyRELU$ layers by $PReLU$ layers to allow the network to adaptively learn the coefficients of negative parts. The network consists of 8 convolution layers with increasing feature depths with factors of 2 and decreasing the feature resolutions each time the feature depth is doubled. The discriminator network (shown in Fig. 3 (bottom row)) is trained to solve for the min-max optimization problem described in Equation 1.

3.3.3 Loss function

Given training pair images $\left\{(x_n, y_n)\right\}_{n=1}^{N}$, our goal is to optimize both the generator network G_{θ_G} and the discriminator network D_{θ_D} simultaneously. Similar to [24], our loss function consists of a mean square error (MSE) loss (Eq. 2), VGG [39] loss (Eq. 3), and adversarial loss components (Eq. 4).

$$l_{MSE}\left(\theta_G\right) = \frac{1}{N}\sum_{n=1}^{N}\left\|G_{\theta_G}(x_n) - y_n\right\|^2 \tag{2}$$

$$l_{VGG/i,j}\left(\theta_G\right) = \frac{1}{N}\sum_{n=1}^{N}\left\|vgg_{i,j}(y_n) - vgg_{i,j}(G_{\theta_G}(x_n))\right\|^2 \tag{3}$$

where $vgg_{i,j}(.)$ is the feature map obtained by the $j-th$ convolution before the $i-th$ max-pooling layer within the VGG network.

$$l_{Gen}\left(\theta_D\right) = \sum_{n=1}^{N} -logD_{\theta_D}\left(G_{\theta_G}\left(x_n\right)\right) \tag{4}$$

The final loss function needs to be minimized is

$$\begin{aligned}\mathcal{L}\left(\theta_G, \theta_D\right) =& \gamma_1 l_{MSE}\left(\theta_G\right) + \gamma_2 l_{VGG/i,j}\left(\theta_G\right) \\ &+ \gamma_3 l_{Gen}\left(\theta_D\right)\end{aligned} \tag{5}$$

where $\gamma_i (i = 1, 2, 3)$ are given weighting parameters.

It is worth mentioning that single l_{MSE} can give a satisfactory results as demonstrated in other deep learning super-resolution approaches [9,8,19,20]. However, it results in lacking details around edge regions. By incorporating with the perceptual loss l_{VGG} and adversarial loss l_{Gen} with an appropriate scale factor, the final loss can give more perceptual and natural results. Empirically, we choose $\gamma_1 = 1$, $\gamma_2 = 1.0e-5$ and $\gamma_3 = 1.0e-3$.

4 Experiments

4.1 Training Dataset

Deep learning techniques usually require many pairs of LR and HR images for training. In our problem, the LR images are 3-pass splat rendering (splat rendering for short) images whereas the HR images are texture mesh images. We synthesize the LIDAR scans on the 3D mesh models where we can control the location, rotation angle and focus of the LIDAR and camera. In this paper, we use 78 mesh models from Google SketchUp. To simulate the LIDAR scan, we first normalize a model so that it can be fit in an unit sphere and set a point on that sphere and in front of the main facet of a building as the virtual LIDAR scanner's position. We create a grid of cells by sub-dividing the space based on the polar and azimuthal angles of the spherical coordinates [1] where rays are cast from the LIDAR's position. We keep the first intersection between the rays and the mesh model as a 3D point along with color.

In addition to using the synthetic dataset, we also evaluate the method on a real LIDAR dataset. We used a Leica laser scanner to capture a large scene along with photographs on a university campus.

Next, we need to generate a set of pairs of LR-HR images which are 3-pass renderings and mesh images respectively. For the synthetic LIDAR data, we use a technique similar to [16] which is used to generate camera locations along with parameters around the 3D model. For each location, we render the point model with splats and keep its rendering as the LR image. We then render the corresponding mesh model and keep its rendering as the corresponding HR image. For the real data, we use the natural images as HR images, and generate splat images with the same camera parameters used in the HR image. In total, we created a dataset consisting of 2000 image pairs of synthetic data and 200 image pairs of real data. Both synthetic data and real data are used for training but we keep two types of data for testing. $Set1$ contains 100 synthetic image pairs and $Set2$ contains 20 real image pairs. All the images have the same size of 768×1024.

4.2 Implementation details

Data samples: To prepare the training data, we sample 400K patches from splat rendering images with a stride of 10 and crop the corresponding HR patches from the ground truth images. For a synthetic dataset, it is trivial to crop a corresponding HR patch of a LR patch since we can control the

[1] Note the sub-dividing in the spherical coordinates (which is used in real LIDAR) is the primary reason for the produced scattered point clouds because there is a higher density in the center region than in the outer region.

camera parameters of virtual views. However, we can not do the same for real datasets due to the numerical error involved when recovering the camera's intrinsic and extrinsic parameters of natural images. To overcome this issue, for each image, we recover the rotation and translation, and then generate the splat and texture splat images. Since the texture splat rendering has high quality, we can do dense-SIFT matching between the texture splat image and the natural image. We extract all the patches at the matched SIFT feature locations with SIFT's orientations and scales in both the splat images and the natural images to form image pairs for training.

Data augmentation: Inspired by [44], we augment the training data in two ways: (i) Rotation: rotate images by $90^o, 180^o$ and 270^o; (ii) Flipping: flip images horizontally or vertically. These steps together lead to an augmented training set that is a factor of 8 times larger than the original data set.

Training details: We implement our network using the deep learning framework TensorFlow [1]. For simplicity, we represent our proposed *SRDSN* network as *SRDSN(d, M)* where d is the number of convolutional layers in a block and M is the number of blocks in the model. We first train the *SRDSN* with *MSE loss*. Empirically, we choose the following hyper-parameters: batch size (64), patch size (48), convolutional filters and bias randomly initialized as described in [9]. We adopt the adjustable gradient clipping [19] to boost the convergence rate while suppressing exploding gradients. Specifically, the gradients are clipped to $[-\frac{\theta}{\gamma}; \frac{\theta}{\gamma}]$, where γ is the current learning rate and $\theta = 0.01$ is the gradient clipping parameter. Furthermore, we use the Adam optimizer [22] with an initial learning rate of 10^{-4}. Next, we train *SRDSN-GAN*. We increase the patch size to 244 and reduce the batch size to 8 due to the GPU memory limitation. All the other parameters are kept the same as the *SRDSN*. Training takes four days on a PC using a Nvidia TitanX GPU.

It is worth mentioning that in training, we use image patches (e.g. 256x256) for both generator and discriminator networks. In testing, we feed image of arbitrary size to the generator only. Thus, our method can work on any size of input image.

4.3 Comparison with state-of-the-art rendering techniques

For quantitative comparison, we use the *Peak signal-to-noise ratio* (PSNR) and *The Structural SIMilarity* (SSIM) indices. As mentioned in Section 4.1, it is trivial to compute those indices for the synthetic testing set for comparison. For real data sets, we follow the same methodology of generating real training data sets. To compute $PSNR$ and $SSIM$ of a rendered image with its corresponding natural image, we perform Dense-SIFT feature matching. We extract the patches on both of the images at matched locations and compute their $PSNR$ and $SSIM$ respectively. The $PSNR$ and $SSIM$ of the two images are the average of all $PSNR$ and $SSIM$ at patch level. Table 1 shows $PSNR$ and $SSIM$ indices on the designed testing sets. For synthetic testing set $Set1$, texture splat rendering obtains an average *PSNR* of 33.18 and an average *SSIM* of 0.96, $SRDSN$ 33.35 and 0.97 whereas *SRDSN-GAN* 33,57 and 0.97. For the real testing set $Set2$, although the texture splat image can render quite well at high texture regions, it performs worse than $SRDSN - GAN$ in terms of $PSNR$ and $SSIM$ (26.24 and 0.86) because it cannot fill the holes in the data. Both the $SRDSN$ and $SRDSN - GAN$ can fill holes in the image since they learn missing information from the training data. The average $PSNR$ and $SSIM$ of $SRDSN - GAN$ on $Set2$ are 27.32 and 0.89 whereas ones of $SRDSN$ are 25.44 and 0.84. Some visual examples of the two testing sets are shown in Fig. 4 and Fig. 5.

4.4 Interactive 3D Scene Navigation

The proposed method can be used to support interactive navigation of large scale 3D scenes represented by colored point clouds. Since the generator network (*SRDSN*) is quite deep (up to 80 layers), it cannot render every frame in real-time. We have designed a novel rendering pipeline to avoid this issue. Imagine that we are navigating a very large point cloud such as an urban scene. Instead of doing super-resolution for every frame, we just do super-resolution on several synthetic views which cover the whole scene. Those images are selected visually during the interactive navigation. The high-resolution versions of those synthetic views are kept as texture images along with the Model, View, and Projection matrices (known as MVP). Using those matrices and texture images, we can perform texture splat rendering which can be done in real time using ordinary computers. We repeat this process every time users enter a new scene. Single frame super-resolution is very fast with modern GPU's as shown in Table 2. Moreover, this process can be done off-line which is transparent to users. Thus, the cost of rendering is reduced to the cost of texture splat rendering. Table 3 shows more details of rendering times for different sizes of point clouds. An example of real time 3D scene navigation can be seen on the video https://www.youtube.com/watch?v=94I4IV0i_mc.

4.5 Image matching with synthetic views

In this section, we demonstrate the ability to extract SIFT features [30] for matching from generated synthetic images. Since we have all the mesh images of all the SketchUp models, we choose 5 mesh images for each model to match with the synthetic images. In total, we have 390 mesh images for

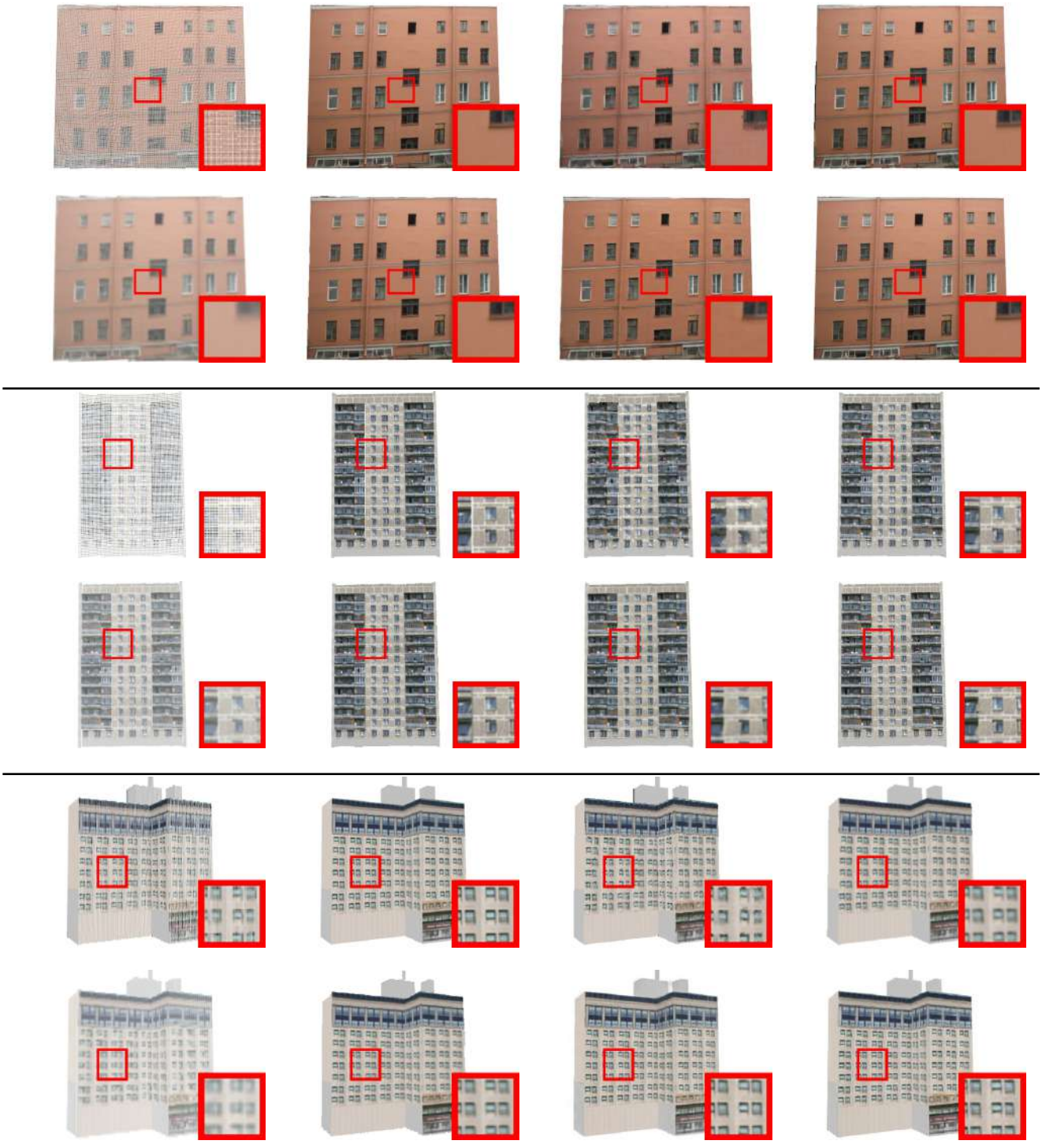

Fig. 4: Comparisons with other methods on synthetic testing set $(Set1)$ for the three examples. From *left* to *right* and *top* to *bottom*: *OpenGL Point Rendering*, *Texture Mesh*, *Intensity Completion*, *Gradient Completion*, *Splat Rendering*, *Texture Splat*, *SRDSN*, and *SRDSN-GAN*.

Table 1: Quantitative comparison of the approaches on the testing sets.

Method	Set1		Set2	
	PSNR	SSIM	PSNR	SSIM
Splat rendering	23.5	0.84	22.98	0.78
Intensity Completion [38]	24.78	0.89	22.72	0.78
Gradient Completion [38]	28.56	0.96	25.12	0.84
Texture splat rendering	33.18	0.96	26.24	0.86
SRDSN	33.35	0.97	25.44	0.84
SRDSN-GAN	33.37	0.97	27.32	0.89

Fig. 5: Comparisons with other methods on real testing set $(Set2)$ for the three examples. From *left* to *right* and *top* to *bottom*: *OpenGL Point Rendering*, *Photograph*, *Intensity Completion*, *Gradient Completion*, *Splat Rendering*, *Texture Splat*, *SRDSN*, and *SRDSN-GAN*.

Table 2: Running time of $SRDSN$ with different image sizes

	480×640	768×1024	1080×1920	2016×3940
Running time(ms)	259	780	1356	4324

Table 3: Rendering time (fps) of interactive 3D scene navigation with different resolutions.

No of points \ Resolution	460×640	768×1024	1080×1920	2016×3940
500K	87	67	64	56
1000K	58	40	38	30
2000K	44	36	31	26
3000K	37	30	28	23
4000K	29	24	21	16

the synthetic data. For the real LiDAR data, we chose 10 images taken using a variety of modern devices: Nikon D90, Nikon D3100, and cell phones to form the real data.

Similar to [38], we measure the average numbers of SIFT features extracted from the images. In order to measure the repetitiveness and the distinctiveness of extracted features, we use the SIFT ratio test: two SIFT features f_1, f_2, one in a mesh image and one in a synthetic image, are considered as a match if their descriptors pass the SIFT ratio test $||des(f_1) - des(f_2)|| < 0.7 * ||des(f_1) - des(f_2')||$ for all features f_2' also extracted in the synthetic image. We match each query photo against all synthetic views rendered from the same scene. For each feature on a synthetic view, we back project to the 3D point cloud. The resulting 2D-3D correspondences are then used to estimate the pose of each query image. The *Avg. Inliers* columns of Fig. 4 describe the number of SIFT features that are used to compute the camera pose position. We notice that our proposed method outperforms other methods on synthetic data sets in number of extracted features, number of inliers, and number of repetitive features whereas its performance on our real data set is a little bit worse than texture splat rendering as shown in Table 4. Remember that for the real data set, we do not have a mesh model, thus we use texture splat images as ground truth images.

4.6 Location recognition using synthetic views.

Following [38], in this section, we also demonstrate that such generated synthetic views can be used to recognize new images taken from viewpoints substantially different from the photos included in the scans. We collect 200 synthetic from 3 synthetic models to form *Set 1* and 50 images taken by the camera within our campus (*Set 2*), together with their locations. For each query image, we extract SIFT features and match against all the synthetic views from the same scan. We assume that the query image has a tagged GPS location so that we can limit it to one scan. Since the correspondence from synthetic views and the point cloud are known, we can establish the 2D-3D correspondences which are then used for pose estimation using the five-point algorithm [32] with RANSAC. Similar to [38], we consider a match to be an inlier to an estimated pose if the reprojection error is below 4 pixels, and regard a novel image as localized if we can find a pose with at least 12 inliers. Those meta parameters are chosen empirically.

Fig. 6 shows the result of the experiment. As we can see, all of the methods perform well on both datasets. *Splat super-resolution* obtains the highest recognition rates which are 98% and 92% on *(*Set 1) and *(*Set 2) respectively. These are 16% and 20% improvements over the base-line method *splat rendering*.

5 Conclusion

In this paper, we have proposed a novel framework for point-based rendering by incorporating the deep learning based super-resolution technique with point rendering methods. The network takes a splat rendering image as input and generates the corresponding high-quality image. Our network which is based on the deeply supervised stacked model with generative adversarial network has demonstrated superior performance in terms of accuracy compared to other methods.

To generate a photo realistic rendering of the real 3D world is a holy grail in computer graphics and remains a challenge even after several decades of research. Our proposed method is a small step in this direction. In the future we would like to build on the idea proposed in this paper, i.e. using realistic high quality natural images as the ground truth data to train the neural network to learn to generate rendering images indistinguishable from the real images. Currently, the proposed method is working in the 2D image space. We would like to extend the proposed network to work directly in 3D and be able to generate high resolution 3D point clouds using low resolution colored point clouds and single and/or multiple high quality images as input. That would be very useful for 3D scene modeling. Another direction we would like to pursue is to investigate shallower networks that can achieve similar results but with a much faster speed so that it can generate the output in real-time.

Table 4: Feature extraction evaluation

Method	Set1			Set2		
	Avg. SIFT	Avg. Repeatabilites	Avg. Inliners	Avg. SIFT	Avg. Repeatabilites	Avg. Inliners
splat rendering	1224	28.26	10.62	2845	13.67	8.45
Intensity completion	1135	31.36	10.38	2754	12.93	9.34
Gradient completion	1345	43.40	19.41	3255	34.83	14.45
Texture splat	1456	61.91	21.35	3524	43.53	18.53
Splat super-resolution	1554	64.74	23.49	3552	47.91	19.45

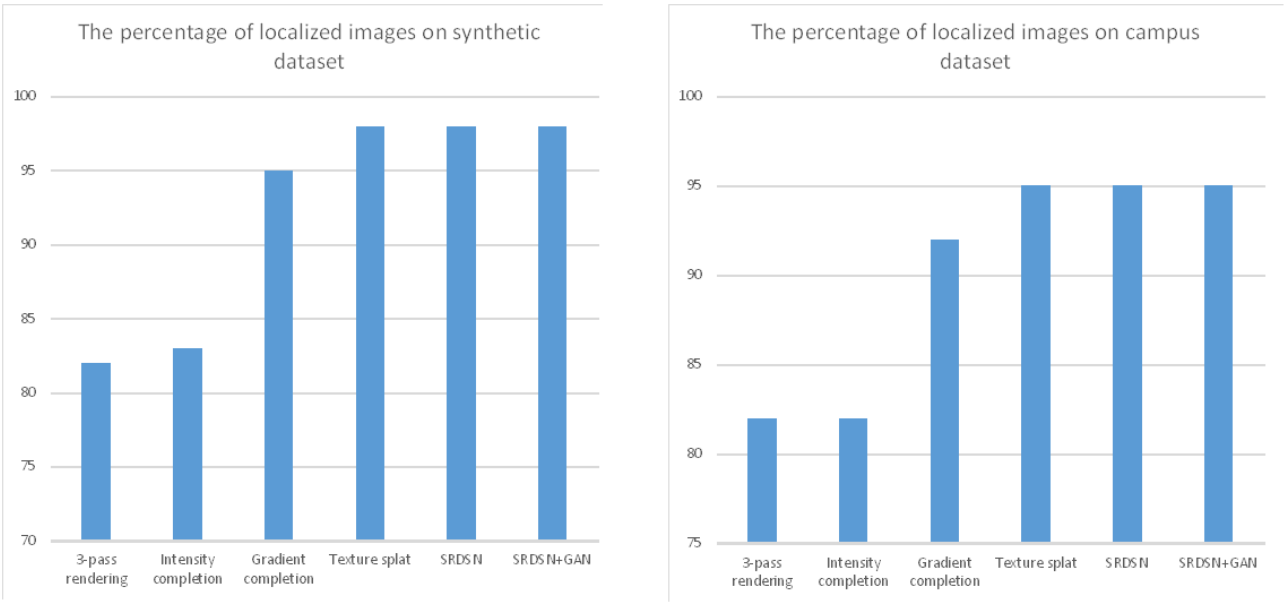

Fig. 6: The percentage of query images that can be localized using synthetic views with different rendering techniques.

Acknowledgment

We would like to thank Sebastian Lipponer for providing open source code [28] of which our splat rendering implementation is mainly based on. We also thank him for all the suggestions during the implementation. We would like to thank Qing Lei and Xu Wang for helping us to generate the video. We also like to thank Roger Kiew, Fan Gao and Chuhang Wang for helping us to generate the training data.

Compliance with Ethical Standards

Conflict of Interest: Giang Bui declares that he has no conflict of interest. Truc Le declares that he has no conflict of interest. Brittany Morago declares that she has no conflict of interest. Ye Duan declares that he has no conflict of interest.

References

1. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., et al.: Tensorflow: Large-scale machine learning on heterogeneous systems, 2015. url h ttp. tensorflow. org/. Software available from tensorflow. org
2. Botsch, M., Hornung, A., Zwicker, M., Kobbelt, L.: High-quality surface splatting on today's gpus. In: Proceedings Eurographics/IEEE VGTC Symposium Point-Based Graphics, 2005., pp. 17–141. IEEE (2005)
3. Brown, M., Lowe, D.G.: Unsupervised 3d object recognition and reconstruction in unordered datasets. In: 3-D Digital Imaging and Modeling, 2005. 3DIM 2005. Fifth International Conference on, pp. 56–63. IEEE (2005)
4. Chang, H., Yeung, D.Y., Xiong, Y.: Super-resolution through neighbor embedding. In: Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on, vol. 1, pp. I–I. IEEE (2004)
5. Cui, Z., Chang, H., Shan, S., Zhong, B., Chen, X.: Deep network cascade for image super-resolution. In: European Conference on Computer Vision, pp. 49–64. Springer (2014)
6. Dai, D., Timofte, R., Van Gool, L.: Jointly optimized regressors for image super-resolution. In: Computer Graphics Forum, vol. 34, pp. 95–104. Wiley Online Library (2015)
7. Denton, E.L., Chintala, S., Fergus, R., et al.: Deep generative image models using a laplacian pyramid of adversarial networks. In: Advances in neural information processing systems, pp. 1486–1494 (2015)
8. Dong, C., Loy, C.C., He, K., Tang, X.: Image super-resolution using deep convolutional networks. IEEE Transactions on Pattern Analysis and Machine Intelligence pp. 295–307 (2015)

9. Dong, C., Loy, C.C., Tang, X.: Accelerating the super-resolution convolutional neural network. In: European Conference on Computer Vision, pp. 391–407. Springer (2016)
10. Freedman, G., Fattal, R.: Image and video upscaling from local self-examples. ACM Transactions on Graphics (TOG) **30**(2), 12 (2011)
11. Glasner, D., Bagon, S., Irani, M.: Super-resolution from a single image. In: 2009 IEEE 12th International Conference on Computer Vision, pp. 349–356. IEEE (2009)
12. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Advances in neural information processing systems, pp. 2672–2680 (2014)
13. Hartley, R., Zisserman, A.: Multiple view geometry in computer vision. Cambridge university press (2003)
14. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770–778 (2016)
15. Huang, J.B., Singh, A., Ahuja, N.: Single image super-resolution from transformed self-exemplars. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 5197–5206. IEEE (2015)
16. Irschara, A., Zach, C., Frahm, J.M., Bischof, H.: From structure-from-motion point clouds to fast location recognition. In: Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, pp. 2599–2606. IEEE (2009)
17. Jia, K., Wang, X., Tang, X.: Image transformation based on learning dictionaries across image spaces. IEEE transactions on pattern analysis and machine intelligence **35**(2), 367–380 (2013)
18. Johnson, J., Alahi, A., Fei-Fei, L.: Perceptual losses for real-time style transfer and super-resolution. In: European Conference on Computer Vision, pp. 694–711. Springer (2016)
19. Kim, J., Lee, J.K., Lee, K.M.: Accurate image super-resolution using very deep convolutional networks. arXiv preprint arXiv:1511.04587 (2015)
20. Kim, J., Lee, J.K., Lee, K.M.: Deeply-recursive convolutional network for image super-resolution. arXiv preprint arXiv:1511.04491 (2015)
21. Kim, K.I., Kwon, Y.: Single-image super-resolution using sparse regression and natural image prior. IEEE transactions on pattern analysis and machine intelligence **32**(6), 1127–1133 (2010)
22. Kingma, D., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
23. Kobbelt, L., Botsch, M.: A survey of point-based techniques in computer graphics. Computers & Graphics **28**(6), 801–814 (2004)
24. Ledig, C., Theis, L., Huszár, F., Caballero, J., Cunningham, A., Acosta, A., Aitken, A., Tejani, A., Totz, J., Wang, Z., et al.: Photo-realistic single image super-resolution using a generative adversarial network. arXiv preprint arXiv:1609.04802 (2016)
25. Lee, C.Y., Xie, S., Gallagher, P., Zhang, Z., Tu, Z.: Deeply-supervised nets. In: Artificial Intelligence and Statistics, pp. 562–570 (2015)
26. Li, Y., Snavely, N., Huttenlocher, D.P.: Location recognition using prioritized feature matching. In: European conference on computer vision, pp. 791–804. Springer (2010)
27. Lim, B., Son, S., Kim, H., Nah, S., Lee, K.M.: Enhanced deep residual networks for single image super-resolution. In: Computer Vision and Pattern Recognition Workshops (CVPRW), 2017 IEEE Conference on, pp. 1132–1140. IEEE (2017)
28. Lipponer, S.: Surface splatting. `https://github.com/sebastianlipponer/surface_splatting` (2015)
29. Liu, Y., Xiong, Y.: Automatic segmentation of unorganized noisy point clouds based on the gaussian map. Computer-Aided Design **40**(5), 576–594 (2008)
30. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. International journal of computer vision **60**(2), 91–110 (2004)
31. Mathieu, M., Couprie, C., LeCun, Y.: Deep multi-scale video prediction beyond mean square error. arXiv preprint arXiv:1511.05440 (2015)
32. Nistér, D.: An efficient solution to the five-point relative pose problem. IEEE transactions on pattern analysis and machine intelligence **26**(6), 756–770 (2004)
33. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. Proc. Computer Vision and Pattern Recognition (CVPR), IEEE **1**(2), 4 (2017)
34. Radford, A., Metz, L., Chintala, S.: Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434 (2015)
35. Savva, M., Yu, F., Su, H., Aono, M., Chen, B., Cohen-Or, D., Deng, W., Su, H., Bai, S., Bai, X., et al.: Shrec16 track large-scale 3d shape retrieval from shapenet core55. In: Proceedings of the eurographics workshop on 3D object retrieval (2016)
36. Schulter, S., Leistner, C., Bischof, H.: Fast and accurate image upscaling with super-resolution forests. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3791–3799 (2015)
37. Shi, W., Caballero, J., Huszár, F., Totz, J., Aitken, A.P., Bishop, R., Rueckert, D., Wang, Z.: Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1874–1883 (2016)
38. Sibbing, D., Sattler, T., Leibe, B., Kobbelt, L.: Sift-realistic rendering. In: International Conference on 3D Vision, pp. 56–63 (2013)
39. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. CoRR **abs/1409.1556** (2014)
40. Snavely, N., Seitz, S.M., Szeliski, R.: Photo tourism: exploring photo collections in 3d. In: ACM transactions on graphics (TOG), vol. 25, pp. 835–846. ACM (2006)
41. Timofte, R., Agustsson, E., Van Gool, L., Yang, M.H., Zhang, L., Lim, B., Son, S., Kim, H., Nah, S., Lee, K.M., et al.: Ntire 2017 challenge on single image super-resolution: Methods and results. In: Computer Vision and Pattern Recognition Workshops (CVPRW), 2017 IEEE Conference on, pp. 1110–1121. IEEE (2017)
42. Timofte, R., De Smet, V., Van Gool, L.: Anchored neighborhood regression for fast example-based super-resolution. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 1920–1927 (2013)
43. Timofte, R., De Smet, V., Van Gool, L.: A+: Adjusted anchored neighborhood regression for fast super-resolution. In: Asian Conference on Computer Vision, pp. 111–126. Springer (2014)
44. Timofte, R., Rothe, R., Van Gool, L.: Seven ways to improve example-based single image super resolution. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1865–1873 (2016)
45. Vinyals, O., Bengio, S., Kudlur, M.: Order matters: Sequence to sequence for sets. arXiv preprint arXiv:1511.06391 (2015)
46. Xie, S., Tu, Z.: Holistically-nested edge detection. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 1395–1403 (2015)
47. Yang, C.Y., Huang, J.B., Yang, M.H.: Exploiting self-similarities for single frame super-resolution. In: Proceedings of the Asian Conference on Computer Vision, pp. 497–510 (2011)
48. Yang, C.Y., Ma, C., Yang, M.H.: Single-image super-resolution: A benchmark. In: European Conference on Computer Vision, pp. 372–386. Springer (2014)
49. Yang, J., Wang, Z., Lin, Z., Cohen, S., Huang, T.: Coupled dictionary training for image super-resolution. IEEE Transactions on Image Processing **21**(8), 3467–3478 (2012)
50. Yang, J., Wright, J., Huang, T., Ma, Y.: Image super-resolution as sparse representation of raw image patches. In: Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on, pp. 1–8. IEEE (2008)

51. Yang, J., Wright, J., Huang, T.S., Ma, Y.: Image super-resolution via sparse representation. IEEE Transactions on Image Processing **19**(11), 2861–2873 (2010)
52. Zwicker, M., Pfister, H., Van Baar, J., Gross, M.: Surface splatting. In: Proceedings of the 28th annual conference on Computer graphics and interactive techniques, pp. 371–378. ACM (2001)

Giang Bui received the B.S. and M.S. degrees from the Vietnam National University of Hanoi in 2004 and 2007, respectively. He is currently pursuing the Ph.D degree at the University of Missouri, Columbia. He was a Research Assistant with the Computer Graphics and Image Understanding Laboratory under the supervision of Dr. Y. Duan. His research interests include image and video processing, 3-D computer vision, and machine learning.

Truc Le received his B.S. in Computer Science in 2012 from the University of Science, VNU-HCM of Vietnam. He is currently pursuing the Ph.D degree at the University of Missouri, Columbia in the Computer Graphics and Image Understanding Laboratory under the supervision of Dr. Ye Duan. His research interests include Computer Graphics, 3D computer vision, and machine learning.

Brittany Morago received the B.S. degree in digital arts and sciences from the University of Florida in 2010, and the Ph.D. degree in computer science from the University of Missouri, Columbia, in 2016. She is currently an Assistant Professor with the Department of Computer Science, University of North Carolina at Wilmington. Her research interests include computer vision and graphics. She was a recipient of the NSFGRF and GAANN fellowships.

Ye Duan received the B.A. degree in mathematics from Peking University in 1991, and the M.S. degree in mathematics from Utah State University in 1996, and the M.S. and Ph.D. degrees in computer science from the State University of New York, Stony Brook, in 1998 and 2003, respectively. From 2003 to 2009, he was an Assistant Professor of Computer Science with the University of Missouri, Columbia. He is currently an Associate Professor of Computer Science with the University of Missouri, Columbia. His research interests include computer graphics and visualization, biomedical imaging, and computer vision.