Fast Generation of High Fidelity RGB-D Images by Deep-Learning with Adaptive Convolution

Chuhua Xian[*1], Dongjiu Zhang[*1], Chengkai Dai[2], and Charlie C. L. Wang[3], *Senior Member, IEEE*

Abstract—Using the raw data from consumer-level RGB-D cameras as input, we propose a deep-learning based approach to efficiently generate RGB-D images with completed information in high resolution. To process the input images in low resolution with missing regions, new operators for adaptive convolution are introduced in our deep-learning network that consists of three cascaded modules – the completion module, the refinement module and the super-resolution module. The completion module is based on an architecture of encoder-decoder, where the features of input raw RGB-D will be automatically extracted by the encoding layers of a deep neural-network. The decoding layers are applied to reconstruct the completed depth map, which is followed by a refinement module to sharpen the boundary of different regions. For the super-resolution module, we generate RGB-D images in high resolution by multiple layers for feature extraction and a layer for up-sampling. Benefited from the adaptive convolution operators proposed in this paper, our results outperform the existing deep-learning based approaches for RGB-D image complete and super-resolution. As an end-to-end approach, high fidelity RGB-D images can be generated efficiently at the rate of 22 frames per second.

Note to Practitioner: Abstract—With the development of consumer-level RGB-D cameras, industries have started to employ these low-cost sensors in many robotic and automation applications. However, images generated by consumer-level RGB-D cameras are generally in low resolution. Moreover, the depth images often have incomplete regions when the surface of an object is transparent, highly reflective or beyond the distance of sensing. With the help of our method, engineers are able to 'repair' the images captured by consumer-level RGB-D cameras in high efficiency. As the typical deep-learning networks are employed in this approach, the proposed approach fits well with the GPU-based hardware architecture of deep-learning computation – therefore it potentially can be integrated into the hardware of RGB-D cameras.

Index Terms—Adaptive Convolution, Deep-Learning, Image Completion, Super Resolution, RGB-D Cameras

I. INTRODUCTION

With the development of 3D sensing technology, consumer-level RGB-D cameras (such as Microsoft Kinect, Intel Re-

*Joint first authors.

[1]C. Xian, and D. Zhang are with the Guangdong Provincial Key Lab of Computational Intelligence and Cyberspace Information, School of Computer Science and Engineering, South China University of Technology, Guangzhou, Guangzhou 510006, China. (Email: chhxian@scut.edu.cn, 201721041222@mail.scut.edu.cn)

[2]C. Dai is a PhD student supervised by C.C.L. Wang at the Faculty of Industrial Design Engineering, Delft University of Technology, The Netherlands. (Email: c.dai@tudelft.nl)

[3]Corresponding author. C.C.L. Wang is currently with the Department of Mechanical and Automation Engineering, The Chinese University of Hong Kong, Hong Kong. (Email: cwang@mae.cuhk.edu.hk)
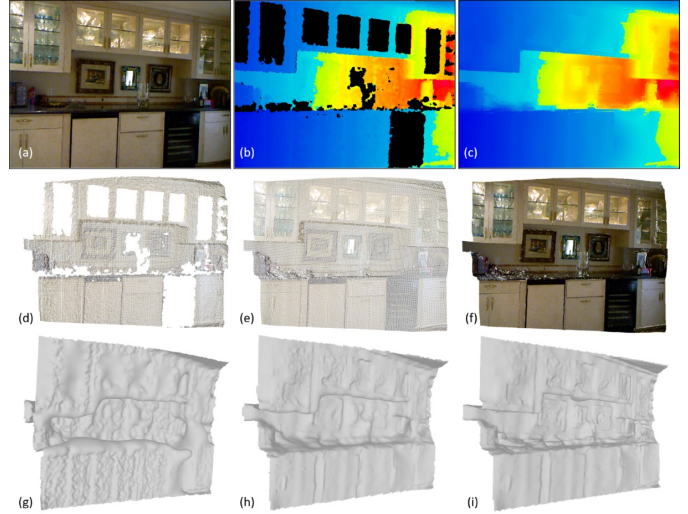
Fig. 1. An example of RGB-D image with missing regions on transparent windows and highly reflective surfaces – see (a) and (b) for the RGB image and the depth image respectively. We propose a deep-learning based method to generate high fidelity RGB-D result – see (c) for the repaired and up-sampled depth image. (d-f) show the point clouds obtained from the raw RGB-D image (d), after completion (e) and after up-sampling (f). (g-i) give the 3D surfaces reconstructed from (d-f) by the widely used Poisson surface reconstruction [1]. It can be observed that the geometry reconstructed from raw data is very poor although the Poisson reconstruction can fill holes at the missing regions. The result of surface reconstruction can be significantly improved after applying the completion (h) and the up-sampling modules (i).

alSense, and Google Tango) are available for daily usage in a low price. All these cameras can capture images with color and depth information (i.e., RGB-D images) with up to 30 fps. The capability to capture RGB-D images in real-time has motivated many applications in robotics and automation, such as 3D scene reconstruction, path planning, logistic packaging, augmented reality, customized product design and fabrication.

A. Problems

Due to the limitations of hardware, these cameras enable a fast acquisition of RGB-D images but in a limited resolution. More seriously, as the depth information is usually obtained by either structured-light or time-of-flight, depth images can have missing regions in large area when the surface of an object is transparent, highly reflective or beyond the distance of sensing [2]. These problems of raw data (i.e., low-resolution and missing regions) can significantly reduce the reliability of downstream applications. For example the indoor environment shown in Fig.1, the depth image captured by a RGB-D camera

has many missing regions (see Fig.1(b)). Directly applying the raw data of RGB-D in motion planning of robots will lead to collision at those missing regions that have transparent glass windows or highly reflective mirrors. Using the state-of-the-art 3D reconstruction method (e.g., the Poisson reconstruction [1]) cannot generate a structure-preserved result as it only mathematically fill the missing regions by an implicit surface interpolating the boundary – see the result of 3D surface reconstructed from the raw data as shown in Fig.1(g). The problems of using RGB-D raw data in a robotic application can also be found in Section IV-D.

To tackle this problem, we propose a method to generate high fidelity RGB-D images from raw data by 1) completing the missing regions and 2) up-sampling the image into a higher resolution, while preserving the structural features presented in the input. In order to use it in real-time applications, a *highly efficient* method is demanded. Moreover, it will be optimistic if the algorithm for computation is *well structured* so that it can be easily implemented on an embedded hardware platform. We employ an end-to-end strategy based on deep-learning to develop a method satisfying these requirements.

B. Related Work

The related prior work mainly focus on image completion, depth estimation, super-resolution and point cloud repair. A comprehensive survey of these areas has been beyond the scope of this paper. Here we only review the most relevant approaches.

1) Image Completion: Many prior works have been proposed for repairing damaged images, including diffusion-based image synthesis (ref. [4], [5], [6]) and patch-based approaches (ref. [7], [8], [9], [10], [11]). The diffusion-based approaches only work on narrow regions so that is also called *inpainting*. The patch-based methods can repair missing regions with large area by progressively reconstructing the structural features, which is not effective for filling regions with complicated features. With the developments of deep learning in recent decades, convolution neural networks (CNNs) have been used for image inpainting [12], [13], [14], which again are limited to very small and thin regions. Along this thread of research, methods have been proposed for repair RGB images with the architectures of auto-encoder [15] and also Generative Adversarial Networks (GAN) [16], [17], [18]. However, the problem of RGB-D image repair is different – we have a complete RGB image but depth map with large missing regions. It is worthy to investigate an effective and efficient method to complete the missing regions in depth map with the help of a complete RGB image. Recently, Zhang and Funkhouser [2] propose a deep-learning based method to complete the depth map of RGB-D image. Their method contains two steps where the surface normals are first estimated in the missing region and then the surface shape is reconstructed by solving a global linear system – i.e., the surface-from-gradient problem. This method depends on the result of the normal estimation seriously, which is also a challenging problem [19]. When the normals are not well-estimated, it will induce results with low quality (see the comparisons given in Section IV). Besides, the computing time

of surface-from-gradient heavily depends on the size of an input image which makes this method less efficient for real-time applications. In a recent work published in [20], designed a *Visual Structure Reconstruction* (VSR) layer to progressively reconstruct the the visual structure and visual feature in the image holes. However, they mainly aim at reconstructing the damaged RGB images.

2) Depth Estimation: Depth estimation from a RGB image is probably helpful for filling the depth information in missing regions. Early methods are mainly based on the hand-tuned models (ref. [21], [22]) or the inversion of rendering techniques (ref. [23], [24]). With the recent development of machine learning, more and more researchers start to propose end-to-end solutions for predicting the depth map directly from RGB images. Eigen et al. [25], [19] first treated the depth estimation as a regression problem and utilized a multi-scale convolution network to train the predictor. Liu et al. [26] combined the deep networks and Markov random fields to learn the depth from single monocular images. Laina et al. [27] proposed a ResNet-based convolution network to predict the depth from a RGB image. However, we cannot directly apply these techniques for the completion task of a depth map as they did not consider the depth information in known regions. Therefore, the estimated depths for the indoor scene in general could be incompatible to the known ones. The approaches of Lee et al. [28] and Imran et al. [29] in literature have considered the problem of completing depth images from sparse sampling. The method presented in [28] learned an extrapolation-like mapping to generate a 'complete' depth image from sparse samples. Similarly, for outdoor scenarios, the approach presented in [29] conducted a new representation as depth coefficients to generate complete depth images from sparse Lidar data. Neither of these approaches can generate geometric details / sharp features in the missing regions, which is very important for the reconstruction of indoor scenes.

3) Super-resolution: Image super-resolution refers to construct a high resolution image from a low resolution one, which has been studied for many years. Existing approaches for this technology can be classified into classical methods, example-based and learning-based methods.

- The classical methods treat the super-resolution problem as the inverse process of a succession of linear transformations from the high-resolved image to a low-resolved image (e.g., [30], [31]). However, these classical approaches often only produce small factor in resolution increase.
- The example-based super-resolution methods are usually based on the similar features between low resolution and high resolution images to select exemplar patches from the input low resolution image for synthesizing the final result (ref. [32], [33]). To accelerate the computation, Freedman and Fattal [34] use limited spatial neighborhoods to generate the self-similar patches. Self-similarity is also used to recover the blur kernel in [35] and to obtain a noise-free image in high resolution from a noisy image in low resolution [36].
- The learning-based methods use machine learning algorithms to learn the mapping from low resolution to
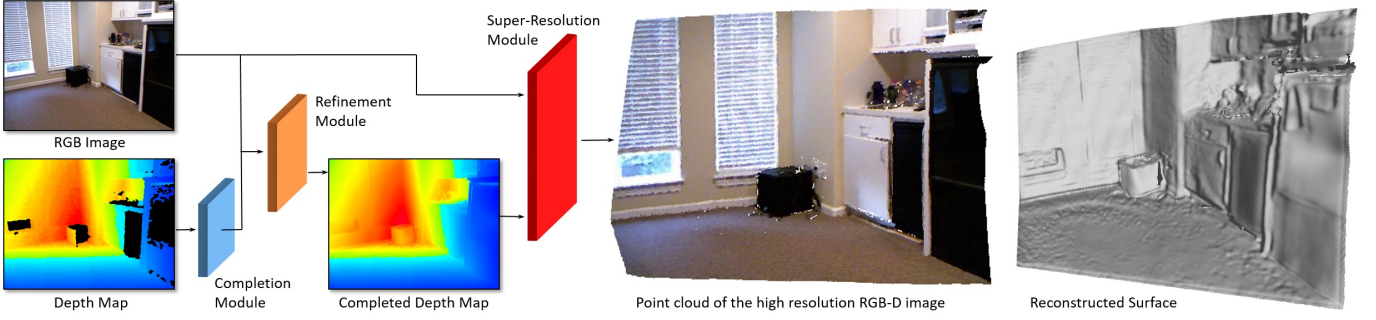
Fig. 2. Overview architecture of our deep-learning based method for generating high fidelity RGB-D images. The network consists of three modules: 1) the completion module to generate depth information in those missing regions, 2) the refinement module to sharpen the boundary of regions in depth image and 3) the super-resolution module to up-sample a RGB-D image while preserving the structural features. 3D surface with high quality can be successfully reconstructed from the resultant RGB-D image [3].

high resolution from a pre-collected database of different resolution image pairs, including manifold learning [37], sparse coding methods [38][39], kernel regression [40]. Many convolution neural networks (CNNs) based methods are developed in recent years (ref. [41], [42], [43]).

In this paper, we develop a learning-based method to generate high fidelity RGB-D images from the raw data captured by consumer-level cameras, which should satisfy the two requirements on high efficiency and well-structured implementation.

4) Point Cloud Repair: In the community of geometric modeling, the problem of point cloud repair has been studied for many years. The strategy of patch-based completion was employed in [44] to repair the missing regions, which is a very time-consuming approach. Iterative consolidation methods have also been developed to push points to have a more regular distribution [45] and also moving into the missing regions [46]. As a result, 3D surfaces can be better reconstructed from the consolidated point cloud. However, these methods based on iteration in general are too slow to be used in real-time applications. Recently, researchers have attempted to apply the end-to-end learning methods to repair point cloud. Cherif et al. [47] tried to enhance the resolution of point clouds using the local similarity at a small scale on the surface. Yu et al. [48] present a data-driven up-sampling architecture which can learn multi-level features per point and expand the point set in feature space. Their work mainly focused on those irregularly sampled general point cloud. In this paper, we propose a simple and well-structured method to process the RGB-D images that can be completed in a very efficient way and has the potential to be integrated into the embedded hardware system.

C. Our Method

A deep-learning based approach is developed in this paper to generate high fidelity RGB-D images from the raw data captured by consumer-level RGB-D cameras, which has low resolution and missing regions with large areas. The proposed architecture of our approach is shown in Fig.2. It mainly consists of three components: the completion module, the refinement module, and the super-resolution module. In the first module, we employ an encoder-decoder network for

completing the missing regions in depth image by using the raw RGB-D image as input. Since the input depth image contains regions with missing data, directly using the standard convolution layers will induce bad results due to the invalid depth values of those missing areas. To address this problem, we introduce an adaptive filter map to conduct the operation of convolution, where the adaptive filter map can filter out the invalid depth values of the missing areas. The features are automatically extracted by the encoding layers, and can be well reconstructed in the completed depth image with the help of decoding layers. After that, a refinement module is used for sharpening the completed image by using the cues obtained from the RGB input. Finally, the super-resolution module is applied to generate an RGB-D image with higher resolution by multiple layers of convolution (akin to the DenseNet [49]) for feature extraction and a layer for up-sampling. To preserve structural features, adaptive convolution operators are employed in the feature extraction layers of this super-resolution module. As a result, high fidelity RGB-D images can be generated efficiently – i.e., images with doubled resolution can be obtained at the rate of 22 frames per second in our experiments.

The major contribution of our work is as follows.

- We introduce two new convolution operators: one for reduction that is adaptive to the missing regions and the other that is adaptive to the discontinuity between regions with significant depth-difference.
- We develop a highly efficient end-to-end deep-learning network with source code accessible [50] to generate high fidelity RGB-D images from the raw data captured by consumer-level RGB-D cameras.
- We construct a publicly accessible training dataset with paired RGB-D images for depth map completion [50].

The proposed technique has been tested on a few publicly available data sets, where the results generated by our approach outperform the existing deep-learning based approaches for both RGB-D image complete and super-resolution.

The rest of this paper is organized as follows. We first propose the adaptive convolution operators in Section II, and then the modules of our deep-learning network are discussed in

detail in Section III. The experimental results are presented in Section IV. Finally, we conclude our paper in Section V.

II. ADAPTIVE CONVOLUTION OPERATION

Two adaptive convolution operators are introduced in this section – a region-adaptive operator for reduction and a depth-adaptive operator for resolution elevation. Formulation of these two operators and the analysis of their functions are presented below.

Given $x_{i,j}$ as the feature value at a pixel (i,j), $N(i,j)$ being the neighbours of (i,j) defined in its convolution mask (e.g., 3×3 kernel given in Fig.3) and b as the corresponding bias, the convolution operation at (i,j) is defined as

$$x'_{i,j} = b + \sum_{(k,l) \in N(i,j)} w_{k,l} x_{k,l}, \qquad (1)$$

where $w_{k,l}$ and b are coefficients of a convolution to be learned through the training process. The convolution operation always follows by an activation function, and the Leacky ReLU function as

$$f(x) = \begin{cases} x & (x > 0) \\ \lambda x & (x \le 0) \end{cases} \qquad (2)$$

with $\lambda = 0.1$ is used in our framework. Note that, multiple convolution operators with different coefficients could be applied to generate multiple feature-channels for an image (or feature maps).

We introduce the concept of adaptive filter map to derive an adaptive convolution operator. For each input (or feature) image $\mathcal{I}$, an adaptive filter map can be defined on every pixel $(i,j) \in \mathcal{I}$ as $\{m_{i,j}\}$. Then, the convolution operator is modified to be adaptive as

$$x'_{i,j} = b + \frac{1}{M} \sum_{(k,l) \in N(i,j)} m_{k,l} w_{k,l} x_{k,l} \qquad (3)$$

$$M = \epsilon + \sum_{(k,l) \in N(i,j)} m_{k,l} \qquad (4)$$

with $\epsilon = 10^{-5}$ for avoiding the singularity caused by the *zero* value for all $m_{k,l}$. Different adaptive filter maps will be defined for different operators.

Different from the depth-aware CNN presented in [51], we introduce a map of adaptive filter to replace the depth similarity matrix used in the convolution step. With the help of this improvement, our framework can better handle the RGB-D image with missing regions which actually happens quite often in practice.

A. Region-adaptive Operation

A convolution operation adaptive to the missing region is developed for the completion network. Specifically, the adaptive filter map is initially generated by the depth map of raw data. $m_{i,j} = 0$ or 1 is assigned for a pixel (i,j) with *invalid* and *valid* depth respectively. After applying a round of convolution, the adaptive filter map $\{m^*_{i,j}\}$ for the newly generated feature map can be updated by the following rules:

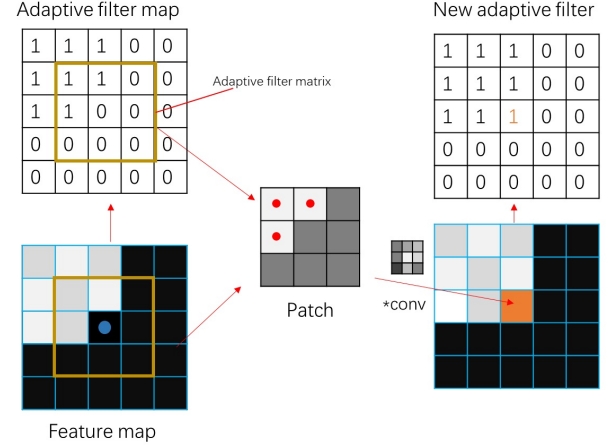- $m^*_{i,j} = 1$ if any pixel $(k,l) \in N(i,j)$ has $m_{k,l} = 1$;

Fig. 3. An illustration of the adaptive convolution operator with 3×3 kernel for completing the missing regions. Black blocks represent the pixels with invalid depth value. The adaptive filter map is first generated by the input depth image, and will be updated after every round of convolution.
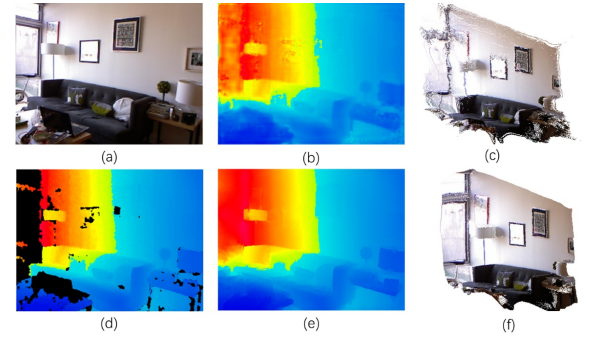

Fig. 4. Comparison with the conventional convolution operation: (a) the RGB image, (b) the depth image completed by using conventional convolution operation, (c) the point cloud of (a) and (b), (d) the raw input depth image, (e) the depth image completed by using adaptive convolution operation, and (f) the point cloud of (a) and (e).

- $m^*_{i,j} = 0$ when $m_{k,l} = 0$ for all $(k,l) \in N(i,j)$.

Repeatedly applying this region-adaptive convolution, the missing regions in an input depth map can be progressively filled. An illustration for the adaptive convolution operator is given in Fig.3.

Deep neural-networks can learn semantic priors in an end-to-end way. These networks employ the conventional convolution operations on images, generating new values for pixels in the feature map. However, all these approaches suffer from their dependency on the initial values, which may include those missing regions containing invalid pixels. The adaptive filter map proposed above is integrated into each convolution layer. The adaptive filter map can be updated directly from the resultant feature map after applying the convolution operation. We actually do not need to store the filter map in each convolution layer, which can save a lot of memory cost.

Figure 4 shows the comparison of our proposed method and the conventional convolution. Fig.4(b) and (c) show the depth image and the corresponding point cloud repaired by conventional convolution. The model of point cloud is seriously distorted – especially at the boundary of the missing areas.
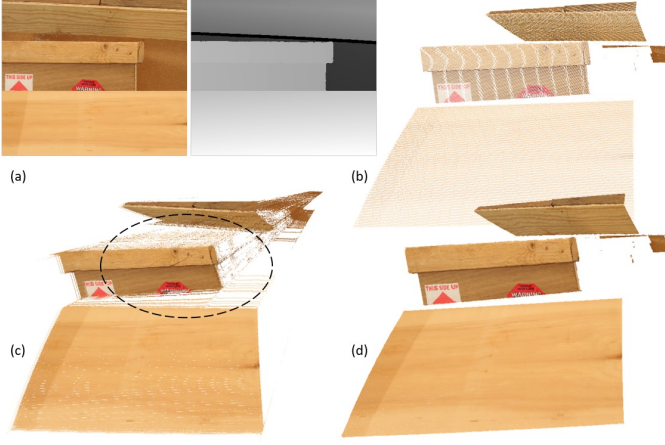
Fig. 5. An example of RGB-D image in super-resolution computed by the conventional vs. the adaptive convolution operators: (a) the input RGB-D image, (b) the RGB-D image with low resolution can be considered as a set of scattered points, (c) the high resolution result obtained from conventional convolution has 'stretched' points near the boundary of different regions (circled by the dash line), and (d) the result of adaptive convolution shows high resolution points with clean boundaries.

Fig. 6. The adaptive filter map for computing the super-resolution of depth image – a local region of the depth image in raw input with size 3×3 is taken as an example. The 3×3 adaptive filter matrix is computed from the raw input by Eq.5, and it is integrated into the convolution computation to make it adaptive to the depth difference.

TABLE I
IMPLEMENTATION OF LAYERS FOR COMPLETION NETWORK

Layer Name	Filter Size	Filters / Channels	Stride / Up-Factor	Output Size
AConv1	7	16 / 16	2 / -	$W/2 \times H/2$
AConv2	5	32 / 32	2 / -	$W/4 \times H/4$
AConv3	3	64 / 64	2 / -	$W/8 \times H/8$
AConv4	3	128 / 128	2 / -	$W/16 \times H/16$
AConv5	3	128 / 128	2 / -	$W/32 \times H/32$
Up-sample	3	- / 128	- / 2	$W/16 \times H/16$
Concat(Aconv4)	3	- / 128 + 128	- / -	-
Aconv6	3	128 / 128	1 / -	$W/16 \times H/16$
Up-sample	3	- / 128	- / 2	$W/8 \times H/8$
Concat(Aconv3)	3	- / 128 + 64	- / -	-
Aconv7	3	64 / 64	1 / -	$W/8 \times H/8$
Up-sample	3	- / 64	- / 2	$W/4 \times H/4$
Concat(Aconv2)	3	- / 64 + 32	- / -	-
Aconv8	3	64 / 64	1 / -	$W/4 \times H/4$
Up-sample	3	- / 32	- / 2	$W/2 \times H/2$
Concat(Aconv1)	3	- / 32 + 16	- / -	-
Aconv9	3	16 / 16	1 / -	$W/2 \times H/2$
Up-sample	3	- / 16	- / 2	$W/2 \times H/2$
Concat(input)	3	- / 16 + 1	- / -	-
Aconv10	3	1 / 1	1 / -	$W \times H$

This is mainly caused by the invalid values in these missing regions, which are incompatible with their neighbors having valid depth-values. However, this sort of incompatibility has been well resolved by the adaptability of our method. The result can be found in Fig.4(e) and (f).

B. Depth-adaptive Operation

The adaptability of convolution operations required in the super-resolution network is different from the network for completion. All pixels of an image (and feature map) are valid. As shown in Fig.2, the input of the super-resolution network is an already completed depth image and the RGB image. However, there is discontinuity of the depth values near the boundary of regions having similar depth-values. Applying conventional convolution operators to these regions of discontinuity will generate blurred artifacts near the boundary of these regions (see Fig.5(c) for an example), where the differences of the depth values around object's boundary should be preserved in the high-resolution depth image. A new adaptive filter map by incorporating the discontinuity of depth-values needs to be developed.

The depth-adaptive filter map is generated by the difference of depths between neighboring pixels. Specifically, when computing the convolution with a kernel centered at the pixel (i, j) with the set of neighbors denoted by $N(i, j)$, the value of the filter map at a pixel $(k, l) \in N(i, j)$ is evaluated by

$$m_{k,l} = \begin{cases} 1 & (G(|D_{k,l} - D_{i,j}|) < \tau), \\ 0 & (G(|D_{k,l} - D_{i,j}|) \geq \tau). \end{cases} \quad (5)$$

$G(\cdot)$ is the Gaussian kernel function

$$G(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}} \quad (6)$$

with the variance $\sigma = 0.0028$. $D_{i,j}$ represents a depth value at the pixel (i, j), which is normalized by mapping the minimal
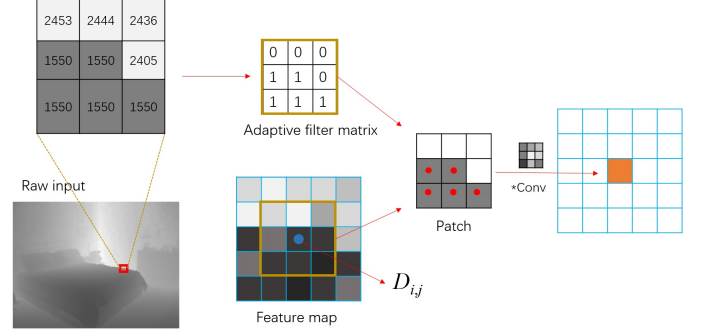
and the maximal depth-values of an input depth-map into the interval $[0, 1]$. τ is a threshold with $\tau = 1.0$ being used in all our tests.

Figure 6 illustrates the meaning of the above equation for adaptive filter map. As shown in Fig. 6, we get a binary matrix by using Eq.(5) to compute the adaptive filter matrix corresponding to the small region in the red rectangular region. This adaptive filter matrix reflects the boundary of objects captured by the input depth image. The filter is integrated into the convolution computation to make it adaptive. Figure 5(c) and (d) show the comparison of results obtained from the conventional convolution vs. our adaptive convolution. It can easily find that the boundaries of three objects are well preserved by using our method.

III. DETAILS OF THE NETWORKS

This section presents the detail of three networks employed in our framework, which includes 1) the encoder/decoder network for completion, 2) the refinement network integrating
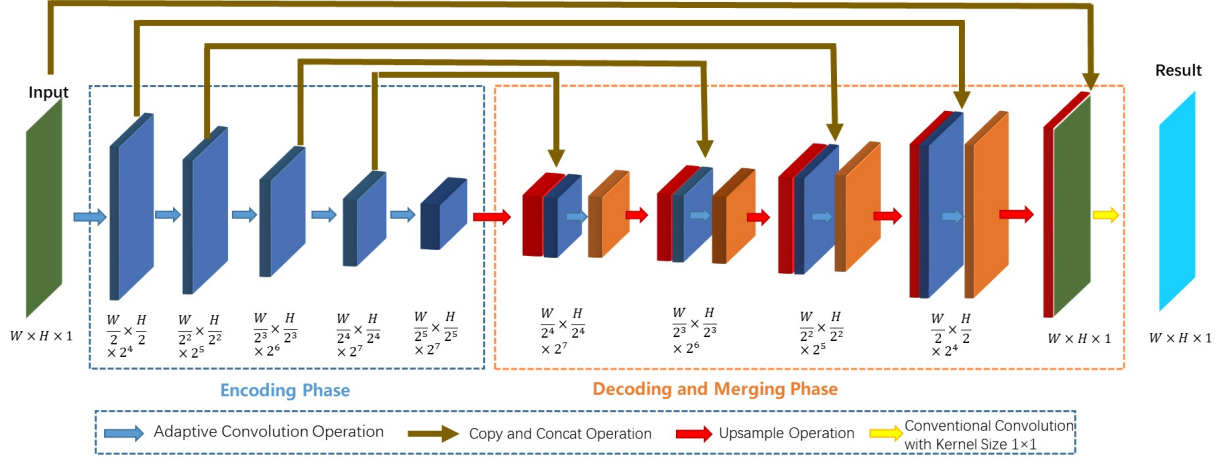
Fig. 7. The completion network consists of two phases: the encoding phase and the decoding and merging phase. Details of every layers are given in Table I.

RGB and depth information and 3) the dense convolution network for elevating the resolution of RGB-D image.

A. Completion Network

The architecture of our completion network is as shown in Fig.7, which consists of two phases: the encoding phase and the decoding phase. This architecture is similar to the UNet [52]. Differently, the adaptive convolution operator proposed in Section II (i.e., Eq.(3)) is employed in our architecture to replace the conventional convolution in each layer.

The encoding module consists of 5 adaptive convolution layers. In our implementation, the stride of the convolution kernel is chosen as 2. As a result, the size of each layer is half of the previous layer while generating double number of channels. A batch normalization layer [53] and the Leacky ReLU activation are also applied after each adaptive convolution layer in the encoding phase. Since the adaptive convolution operations will filter out the invalid values of the input depth image, the depth features in the missing area can be successfully reconstructed in different scales. After applying n layers of adaptive convolution, all the missing regions with size less than $2^{n+1} \times 2^{n+1}$ will be completed successfully.

The decoding phase is used to up-sample the discriminative feature maps and generate the completed depth-map progressively. The architecture of the decoding layers is symmetric to that of the encoding layers. As illustrated in Fig.7, each feature layer of the encoding module is concatenated to a corresponding up-sampling layer of the decoding part. Note that, all the feature layers 'duplicated' from the encoding part have their corresponding adaptive filter maps, which will be applied in the adaptive convolution in the decoding part. In our implementation, the ratio of up-sampling is chosen as $2\times$ and the numbers of the channels keep the same. At last, the input depth map is concatenated to the last up-sampling layer together with its corresponding adaptive filter map to obtain the finally completed depth map with the size $W \times H$.

Loss Function: To successfully reconstruct depth values in the missing regions, influence of both the valid and the invalid

pixels should be considered. Suppose the size of an input depth image is $W \times H$ and its corresponding adaptive filter map is $\{m_{x,y}\}$, the loss of valid and invalid regions are defined as follows.

$$L_{valid} = \frac{\sum_{x=1}^{W} \sum_{y=1}^{H} (m_{x,y}(D_{x,y}^{GT} - D_{x,y}^{OPT}))^2}{\sum_{x=1}^{W} \sum_{y=1}^{H} m_{x,y}} \quad (7)$$

$$L_{invalid} = \frac{\sum_{x=1}^{W} \sum_{y=1}^{H} ((1 - m_{x,y})(D_{x,y}^{GT} - D_{x,y}^{OPT}))^2}{\sum_{x=1}^{W} \sum_{y=1}^{H} (1 - m_{x,y})} \quad (8)$$

where D^{GT} and D^{OPT} are the ground-truth and the output depths respectively. The total loss function for depth completion, L_{CP}, is defined as

$$L_{CP} = w_\alpha L_{valid} + w_\beta L_{invalid}. \quad (9)$$

Here $w_\alpha, w_\beta > 0$ controls the balance between these two terms of loss function. $w_\alpha = 1.0$ and $w_\alpha = 6.0$ employed in all our experiments can generate good results.

B. Refinement Network

Output of the completion network is a depth image repaired from the raw input. However, the boundaries of repaired depth image is blurred (see Fig.8(b) for an example). The complete network only considers the depth map; however, the information of the color image can provide valuable guidance for the boundary of completed regions. To incorporate this cue of information, we design a network to refine the boundaries of objects. As shown in Fig. 9, the input of our refinement network contains both the original RGB image and the completed depth image.

Our refinement process mainly consists of four steps, which are in fact a network-based implementation of bilateral filtering as described below:

1) Extract patches of each pixel from the RGB image and the depth image respectively by using the window size as 9×9 pixels.
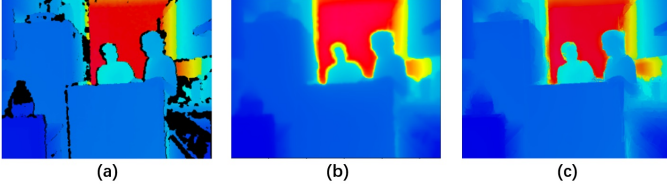
Fig. 8. The input depth image with missing regions (a) can be repaired by the completion network (b), and then further fine-tuned by the refinement network (c).
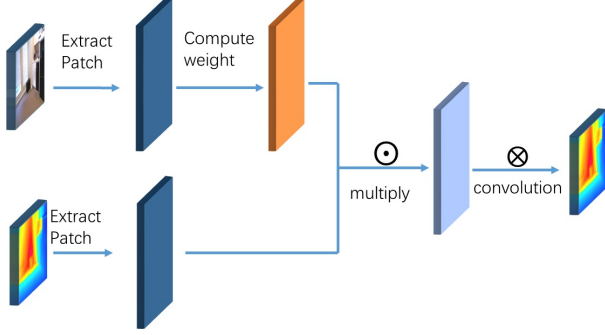

Fig. 9. The architecture of the refinement network.

2) For each patch of the corresponding pixel $\mathbf{p}$ of RGB image, we compute the weight $W_{\mathbf{p}}$ as following:

$$W_{\mathbf{p}} = \sum_{\mathbf{q} \in N(\mathbf{p})} G_s(\|\mathbf{p} - \mathbf{q}\|) G_r(\|I_{\mathbf{p}} - I_{\mathbf{q}}\|) \quad (10)$$

where $G_s(\cdot)$ and $G_r(\cdot)$ are two Gaussian kernel functions with deviations $\sigma_s = 7.0$ and $\sigma_r = 5.0$ respectively, $N(\mathbf{p})$ is the set of neighbors of pixel $\mathbf{p}$, and $I(\cdot)$ is the input RGB image.

3) Multiply the weight patch and the patches of depth image to obtain the new depth value at $\mathbf{p}$ as:

$$D_{\mathbf{p}} = \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q} \in N(\mathbf{p})} G_s(\|\mathbf{p} - \mathbf{q}\|) G_r(\|I_{\mathbf{p}} - I_{\mathbf{q}}\|) D_{\mathbf{q}}, \quad (11)$$

where the feature of RGB information is integrated into the depth map.

4) Conduct a fixed convolution layer with stride $s = 1$ and all the weights as 1 with the bias 0.

There is only one convolution layer. As a result, the computation of refinement network is very efficient.

C. Super-resolution Network

The super-resolution network is utilized to generate high resolution data from the repaired RGB-D data. To achieve this goal, we first extract the features by the deep neural network and then conduct up-sampling in the last few layers to generate the high resolution RGB-D images. The architecture of our super-resolution network is as shown in Fig.10, which consists of two phases – the feature extraction phase and the up-sampling phase. The feature extraction phase is akin to the DenseNet architecture [49] but by differently applying the adaptive convention operation proposed in Section II-B. It contains 5 blocks, and all the sizes of the blocks are the

TABLE II
IMPLEMENTATION OF LAYERS FOR SUPER-RESOLUTION NETWORK

Layer Name	Filter Size	Filters / Channels	Up-Factor
AConv0	3	64 / 64	-
AConv1(Block1)	3	- / 64 + 64	-
AConv2(Block2)	3	- / 128 + 64	-
AConv3(Block3)	3	- / 192 + 64	-
AConv4(Block4)	3	- / 256 + 64	-
AConv5(Block5)	3	- / 320 + 64	-
AConv6	3	64 / 64	-
Concat(AConv0)	-	- / 64 + 64	-
AConv7	3	128 / 128	-
Sub-Pixel-Conv	-	-	4
Conv	1	1 / -	-

same. Each block consists of two adaptive convolution layers and a Leaky ReLU layer. As the network used in our feature extraction module is not very deep, we do not include the normalization layer in each block. With this simplification, the quality of results is not significantly influenced while the training can be conducted more efficiently.

The up-sampling phase mainly consists of three layers: two adaptive convolution layers, and one up-sampling layer. Here the sub-pixel convolution operation [54] is employed for the up-sampling layer, which can generate the high-resolution data by re-assembling the extracted features from the former module. The kernel size of the last convolution layer is 1×1 by using $\tanh(\cdot)$ as the active function.

Loss Function: Suppose the size of an input image is $W \times H$ and r is the ratio of up-sampling, the loss function of our super-resolution network is defined as:

$$L_{SR} = \frac{1}{r^2 WH} \sum_{x=1}^{rW} \sum_{y=1}^{rH} \|(I_{x,y}^{GT}, D_{x,y}^{GT}) - (I_{x,y}^{OPT}, D_{x,y}^{OPT})\|^2$$
$$(12)$$

where $(I_{x,y}^{GT}, D_{x,y}^{GT})$ and $(I_{x,y}^{OPT}, D_{x,y}^{OPT})$ denote the ground truth and the output RGB-D at (x, y) respectively.

D. Training Parameters

We implement the proposed architecture by Python and TensorFlow [55]. The networks are trained for 100 epoch by using Adam [56] algorithm to optimize the loss functions. The batch size is set as 4 for the training of completion network and is set as 8 for the training of super-resolution network. All the learning ratios are set to 10^{-4}. The numbers of the layers in the encoding and decoding parts of the completion network are both 7 as shown in Fig.7, and 5 layers are employed in the super-resolution network as shown in Fig.10.

IV. RESULTS

All the experiments presented in this paper are conducted on a PC equipped with an Intel(R) Core(TM) i7-7700 CPU at 3.6GHz and a NVIDIA GeForce 1080Ti graphic card. In this section, we will first introduce the datasets used for training and testing, and then present the experimental results for completion and super-resolution. Comparisons with other approaches are also provided to demonstrate the advantage of our approach. Lastly, we apply the results generated by
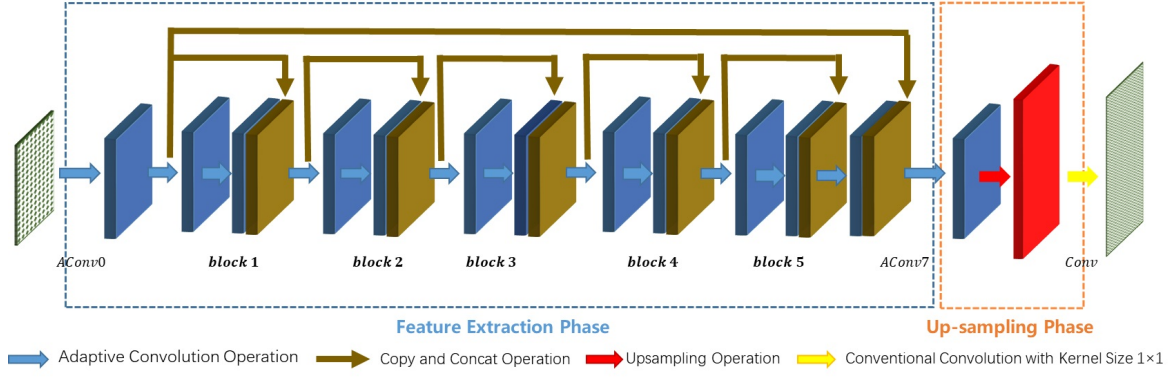
Fig. 10. The architecture of the super-resolution network, which consists a feature extraction phase and an up-sampling phase. Details of every layers have been given in Table II.
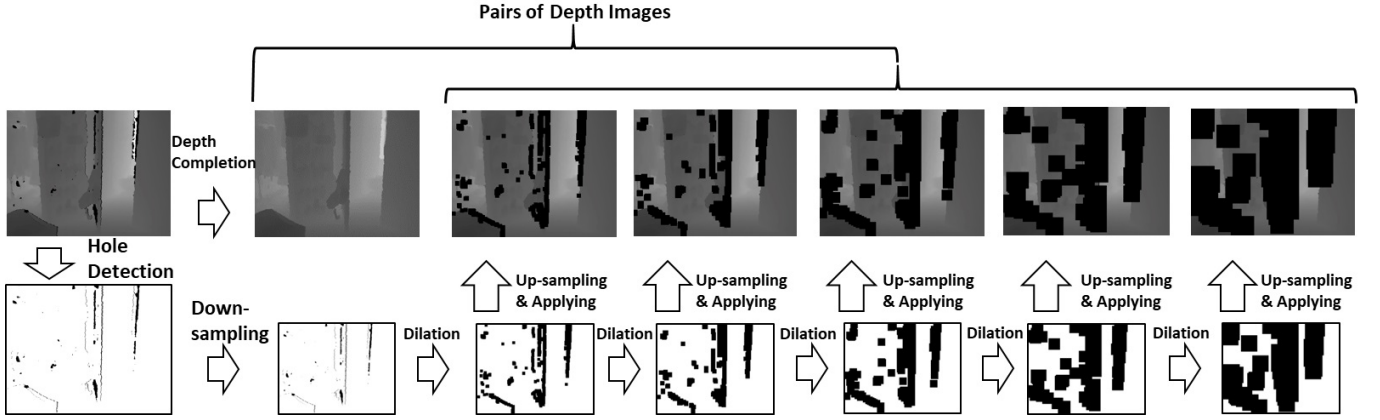

Fig. 11. Details about how to generate training pairs of depth images with large missing area for the training dataset. Starting from an image with small missing area, we first detect the holes and generate a binary image with lower resolution. After that, the morphology operator – dilation is applied to the binary image 4 times to generate the binary images indicating holes in different sizes. Finally, we up-sample the binary images back to the original resolution to generate images with holes. By applying the depth-completion method of [2] to repair the small missing areas in the original image, a few pairs of images can be obtained for our dataset.

our framework in a robotic application that needs fast 3D reconstruction.

A. Dataset

Three publicly available RGB-D datasets, including NYU-v2 [57], RGBD-SCENE-v1 [58] and RGBD-SCENE-v2 [59], are employed in our tests. NYU-v2 dataset is recorded from the Microsoft Kinect, which contains of $1,449$ densely labeled pairs of aligned RGB and depth images and $407,024$ unlabeled frames. The RGBD-SCENE-(v1, v2) datasets are created by aligning a set of video frames using Patch Volumes Mapping [59].

Our networks are trained by a supervised learning method. However, the dataset of RGB-D images paired with 'completed' depth images is not available. For training our completion network, we construct a dataset of paired RGB-D images by ourselves. First of all, $1,302$ images that have relatively small area of missing regions are selected from the NYU-v2. We first extract the missing regions and represent them as black pixels in a down-sampled binary image. Then, the dilation operator is repeatedly applied to this binary image to generate the masks of holes in different sizes. After that, the

binary images are up-sampled back to its original resolution to generate depth-images with large missing areas. The small holes in the original images can be 'repaired' by the method of Zhang and Funkhouser [2]. Each original image can be used to generate a few pairs of depth-images in this way. As a result, among the $1,302$ depth-images of NYU-v2, we randomly select $1,083$ images (around 90%) to generate $22k$ pairs of training images. The remaining 219 depth images (around 10%) are used for testing in our experiment. We have made the training dataset of our completion network publicly accessible together with the source code of our framework [50].

The dataset for training our super-resolution network is generated by using the nearest-neighbor interpolation based sub-sampling to obtain the low-resolution images from RGBD-SCENE-(v1, v2) and NYU-v2. In total, $9,140$ images (from RGBD-SCENE-v1) are down-sampled to generate the pairs of images for training. The other $2,285$ images (from RGBD-SCENE-v2 and NYU-v2) are used as the testing set.

B. Results of Completion

We test the performance of our completion network by the images from NYU-v2 dataset that are not included in
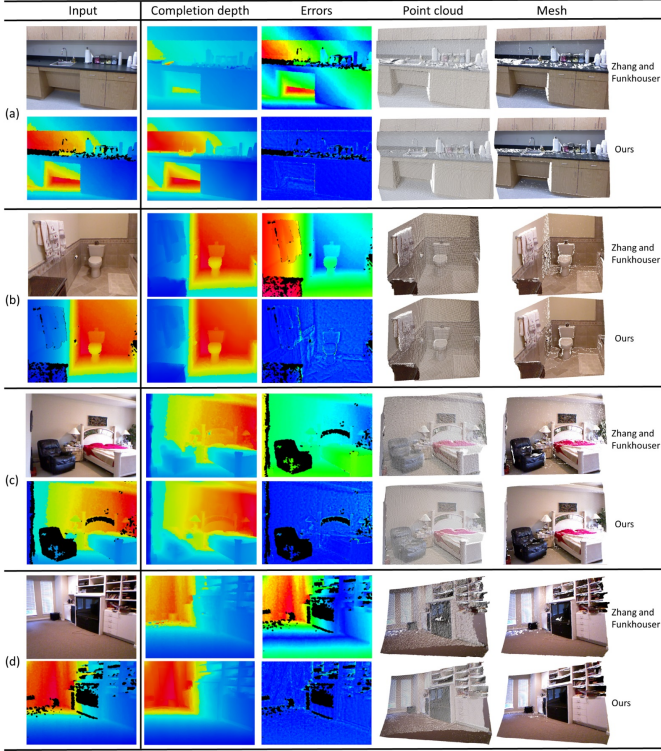
Fig. 14. The distribution of errors generated on the test set for depth-completion – ours vs. the method of Zhang and Funkhouser [2].

TABLE III
COMPARISON OF AVERAGE ERROR AND COMPUTING TIME: OURS VS. THE APPROACH OF ZHANG AND FUNKHOUSER [2].

	Average Error†		Comp. Time (sec.)	
	[2]	Ours	[2]	Ours
Fig. 12(a)	0.089	0.009	13.40	0.044
Fig. 12(b)	0.072	0.009	13.39	0.044
Fig. 12(c)	0.087	0.021	14.01	0.047
Fig. 12(d)	0.039	0.009	13.28	0.043
Fig. 13(a)	0.018	0.003	13.44	0.044
Fig. 13(b)	0.042	0.017	14.31	0.047

† To conduct a fair comparison, the errors are only evaluated on the pixels with known values in the input image.

Fig. 12. Examples to demonstrate the performance of our complete network. From left to right, the input raw RGB-D images, the completed depth maps, the error-maps, the point-cloud rendering and the mesh rendering of completion results. The results generated by our method have been compared with the state-of-the-art completion results of Zhang and Funkhouser [2]. Note that, the error maps only provide the evaluation on pixels with depth-values known in the input RGB-D images. All the examples are from the NYU-v2 dataset [57].
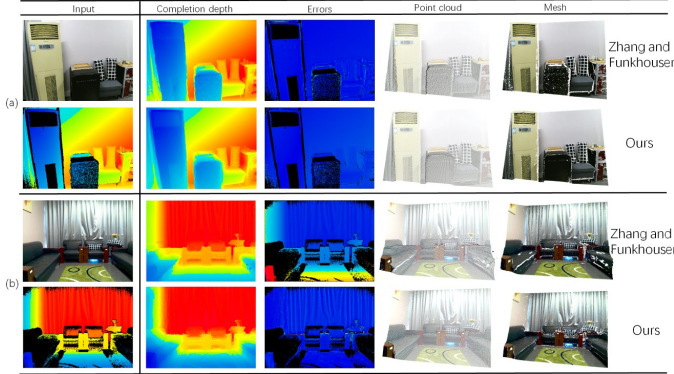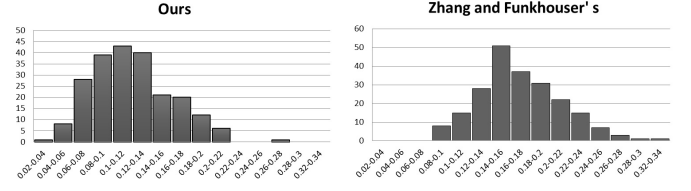
Fig. 13. Results of two RGB-D images captured by ourselves using Kinect v2. The results of our completion network are also compared with the approach of Zhang and Funkhouser [2].

our training dataset. The results of completion are shown in Fig.12. Moreover, the function of our complete network has been tested on RGB-D images captured by ourselves using Kinect v2 – the results are given in Fig.13.

Quantitative Evaluation: To quantitatively evaluate the error of the reconstructed depth images, we compare the predicted depths values with the original input. All depth values are first normalized into the interval $[0, 1]$. Then the errors are calculated by the absolute difference between the ground-truth

and the prediction. The color-maps indicating these errors have shown in the third column of Figs.12 and 13. To conduct a fair comparison, the errors are only evaluated on the pixels with known values in the input image – i.e., black colors are leaved for those missing regions. Table III show the comparisons on the average error and also the computing time of examples given in Figs. 12 and 13. Besides of these shown examples, we also also conduct experiment on the whole test set with 219 depth-images. The statistics are given in Table IV and Fig.14. It is easy to find that our method can generate predictions more accurate than that obtained by [2]. Besides, different from the approach in [2] that needs a post-processing step for surface-from-gradient reconstruction, our end-to-end framework is very efficient and can generate repaired RGB-D image at the rate of 22 frames per second. This is very important for industrial applications in automation.

C. Results of Super-resolution

To verify the performance of our super-resolution module, we have tested our network on all images in two datasets – NYU-v2 [57] and RGBD-SCENE-v2 [59]. In all these tests, we downsample them into images with lower resolution by a ratio of 4 and compare the reconstructed results with the

TABLE IV
STATISTIC AND COMPARISON ON OUR TEST DATASET†

	Average Err.	Max Err.	Min Err.	Avg. Time‡ / Frame
[2]	0.170	0.329	0.085	13.45 sec.
Ours	0.119	0.261	0.037	0.045 sec.

† The experiments are conducted on the test dataset with 219 images obtained from the NUY-v2 as discussed in Section IV-A.
‡ In average, our method can achieve the rate of 22 frames per second.

TABLE V
COMPARISONS OF SUPER-RESOLUTION RESULTS

NYU-v2 Dataset [57]					
	Bicubic	VDSR	SRGAN	RDN+	Ours
PSNR (Avg.)	41.68	48.85	38.28	39.42	**73.22**
PSNR (Min.)	32.91	33.02	7.065	2.797	**86.09**
RMSE (Avg.)	0.1351	0.0973	0.1845	0.2096	**0.0112**
RMSE (Max.)	2.8656	2.8270	6.6030	5.8610	**0.1478**
RGBD-SCENCES-v2 Dataset [59]					
	Bicubic	VDSR	SRGAN	RDN+	Ours
PSNR (Avg.)	43.29	56.55	50.16	67.66	**101.6**
PSNR (Min.)	34.18	50.44	19.64	48.15	**86.01**
RMSE (Avg.)	0.1079	0.3127	0.1507	0.1019	**0.0696**
RMSE (Max.)	0.1300	0.5785	0.2265	0.1529	**0.1068**

TABLE VI
PERFORMANCE STUDY OF OUR SUPER-RESOLUTION MODULES ON THE TEST DATASET[†] WHEN USING DIFFERENT DOWN-SAMPLING METHODS

Down-sampling by nearest-neighbor interpolation			
	SRGAN	RDN+	Ours
PSNR (Avg.)	59.264	64.834	**98.527**
RMSE (Avg.)	0.311	0.284	**0.068**
Down-sampling by bilinear interpolation			
	SRGAN	RDN+	Ours
PSNR (Avg.)	46.632	64.530	**73.070**
RMSE (Avg.)	0.358	0.288	**0.200**
Down-sampling by bicubic interpolation			
	SRGAN	RDN+	Ours
PSNR (Avg.)	55.711	64.211	**72.365**
RMSE (Avg.)	0.417	0.293	**0.250**

[†] The experiments are conducted on the test dataset with $2,285$ images obtained from RGBD-SCENE-v2 and NYU-v2 as discussed at the end of Section IV-A.
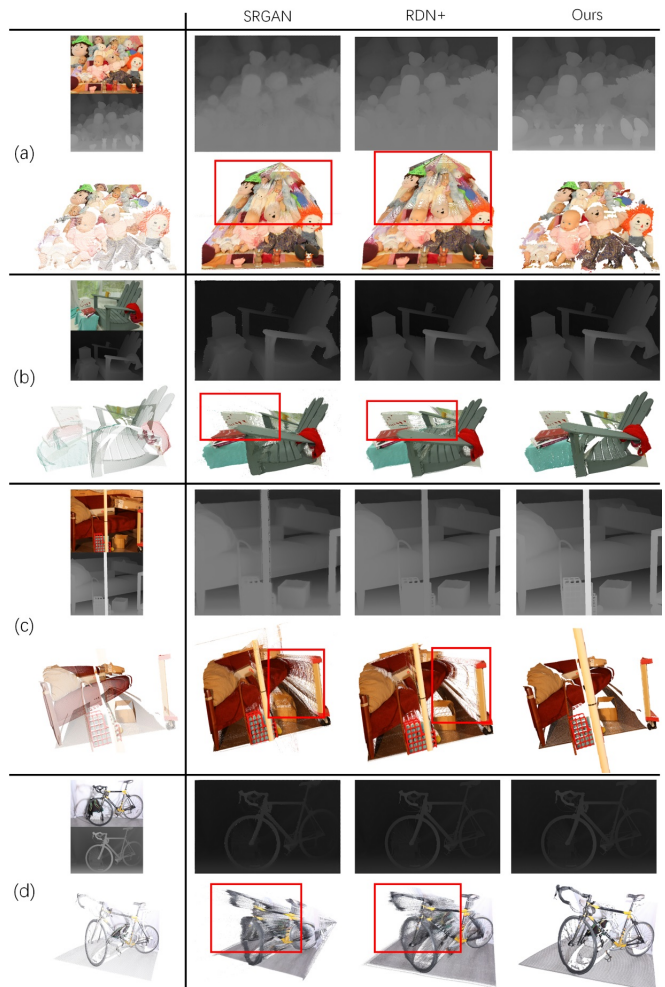

Fig. 15. Results of super-resolution computation on four examples from the Middlebury Stereo Databases [62]. The first column shows the RGB image, the depth map and the corresponding point cloud model. The second and the third columns give the results of SRGAN [43] and RDN+ [61] respectively. The last column shows the results of our method. The regions in red rectangle show the incorrect interpolation at the boundaries generated by other methods.

Fig. 16. RGB-D images with high resolution are generated by our network with input down-sampled by using different strategies. All results are acceptable although the one from nearest-neighbor interpolation is the best.

original image (as ground truth). The statistical results of our verification tests are given in Table V. We compare our method with Bicubic interpolation, VDSR [60], SRGAN [43] and RDN+ [61]. The average value and the 'worst' value for both the *peak signal-to-noise ratio* (PSNR) and the *root-mean-square error* (RMSE) are reported, and the best performance is marked in bold in the table. It is easy to find that our approach outperforms all other approaches on these two datasets. This is mainly benefited by employing the newly developed adaptive convolution operations in our computation. As a result, the boundaries of objects can be well preserved in the reconstructed images.

Four examples from Middlebury database 2005 [62] are shown in Fig.15 and compared with the results generated by SRGAN [43] and RDN+ [61]. As can be found, our method can preserve the boundaries of the object quite well while the other two methods induce incorrect interpolations among objects with different depths. The 3D reconstruction results from repaired RGB-D images in super-resolution can also be found in Fig.1 and 2.

It is also a very interesting study to test the performance of our super-resolution module on images generated by different down-sampling methods. The testing set obtained from RGBD-SCENE-v2 and NYU-v2 are conducted in this experiment by using three different down-sampling methods, including 1) the nearest-neighbor interpolation (i.e., the method we used to generate the training samples), 2) the bilinear interpolation and 3) the bicubic interpolation, to generate images with low resolution as input. The results are compared with SRGAN [43] and RDN+ [61] – see the statistic given in Table VI. As the training of our super-resolution network is conducted on the training dataset with images down-sampled by the nearest-neighbor interpolation, the best performance of our network is observed on the input obtained by using
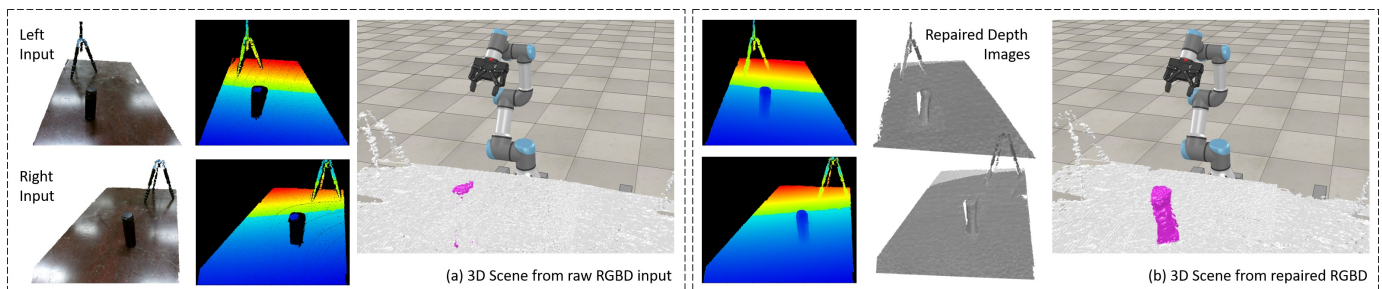
Fig. 17. A robotic application of using rapidly repaired RGB-D images for grasping, where the setup has two RGB-D cameras located at the left and right sides of the working table. When working on a water-bottle with black metallic surface, the captured depth images have the regions of the bottle's surface missed. As a result, the 3D scene reconstructed from raw RGB-D images has the bottle missed (a). After applying the completion technique proposed in this paper, the water-bottle can be well reconstructed in the 3D scene (b).

the same down-sampling strategy. This is because that the pattern of inverse down-sampling 'mapping' according to a particular down-sampling strategy (i.e., the nearest-neighbor interpolation in our case) has actually been learned from the training dataset. When applying this network to images generated by other down-sampling strategies, worse results are observed although they are still acceptable (see Fig.16). Moreover, benefited from the adaptive convolution operations introduced in this paper, our approach still outperforms other two methods.

D. Robotic Application

We have applied the technique developed in this paper in a robotic grasping scenario. For objects to be grasped, 3D reconstruction is usually conducted to analyze their shapes to realize a better grasping strategy (ref. [63], [64]). As shown in Fig.17, the grasping task is to be completed with the help of two RGB-D cameras located at two sides of a working table. When using this setup to capture the shape of a water-bottle with black metallic surface, regions are missed on the surface. To resolve this problem, we apply the method presented in this paper to obtain completed RGB-D images to reconstruct a more complete 3D object for the water-bottle. The results and comparisons are given in Fig.17. A very important property of our approach is its efficiency, which is very important for robotic and automation applications.

V. Conclusions

In this paper, we proposed a deep-learning method for efficiently generating repaired RGB-D images in high resolution with the help of newly proposed adaptive convolution operations. Our deep-learning network consists of three cascaded modules. First, the completion module is developed based on the architecture of encoder-decoder. Then, the boundary of different regions is sharpened by a refinement module, which is a convolution-based implementation of bilateral filtering. Thirdly, we conduct multiple layers for feature extraction and a layer for up-sampling to generate RGB-D images in high resolution as the super-resolution module. The adaptive operations are newly developed in our framework to generate results outperforming conventional deep-learning networks. Numerous experiments on public datasets have demonstrated

the effectiveness and efficiency of our approach (i.e., with 22 frames per second in our experiments), which is very important for robotic and automation applications such as grasping, packaging and planning. To further enhance the generality of our method in computing the super-resolution of the depth map, we plan to train the network by using paired training images obtained from different down-sampling algorithms in our future work.

References

[1] M. Kazhdan and H. Hoppe, "Screened poisson surface reconstruction," *ACM Transations on Graphics*, vol. 32, no. 3, pp. 29:1–29:13, July 2013.

[2] Y. Zhang and T. Funkhouser, "Deep depth completion of a single rgb-d image," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 175–185.

[3] S. Liu, C. C. Wang, G. Brunnett, and J. Wang, "A closed-form formulation of hrbf-based surface reconstruction by approximate solution," *Computer-Aided Design*, vol. 78, pp. 147–157, 2016.

[4] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester, "Image inpainting," in *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*. ACM Press/Addison-Wesley Publishing Co., 2000, pp. 417–424.

[5] C. Ballester, M. Bertalmio, V. Caselles, G. Sapiro, and J. Verdera, "Filling-in by joint interpolation of vector fields and gray levels," *IEEE Transactions on Image Processing*, vol. 10, no. 8, pp. 1200–1211, 2001.

[6] A. Levin, A. Zomet, and Y. Weiss, "Learning how to inpaint from global image statistics," in *Proceedings Ninth IEEE International Conference on Computer Vision*, 2003, p. 305.

[7] A. A. Efros and W. T. Freeman, "Image quilting for texture synthesis and transfer," in *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*. ACM, 2001, pp. 341–346.

[8] V. Kwatra, I. Essa, A. Bobick, and N. Kwatra, "Texture optimization for example-based synthesis," in *ACM Transactions on Graphics*, vol. 24, no. 3. ACM, 2005, pp. 795–802.

[9] C. Barnes, E. Shechtman, D. B. Goldman, and A. Finkelstein, "The generalized patchmatch correspondence algorithm," in *Proceedings of the European Conference on Computer Vision*. Springer, 2010, pp. 29–43.

[10] T. Kwok, H. Sheung, and C. C. L. Wang, "Fast query for exemplar-based image completion," *IEEE Transactions on Image Processing*, vol. 19, no. 12, pp. 3106–3115, Dec 2010.

[11] J.-B. Huang, S. B. Kang, N. Ahuja, and J. Kopf, "Image completion using planar structure guidance," *ACM Transactions on Graphics*, vol. 33, no. 4, p. 129, 2014.

[12] J. Xie, L. Xu, and E. Chen, "Image denoising and inpainting with deep neural networks," in *Advances in Neural Information Processing Systems*, 2012, pp. 341–349.

[13] R. Köhler, C. Schuler, B. Schölkopf, and S. Harmeling, "Mask-specific inpainting with deep neural networks," in *German Conference on Pattern Recognition*. Springer, 2014, pp. 523–534.

[14] J. S. Ren, L. Xu, Q. Yan, and W. Sun, "Shepard convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2015, pp. 901–909.

[15] A. van den Oord, N. Kalchbrenner, L. Espeholt, O. Vinyals, A. Graves, *et al.*, "Conditional image generation with pixelcnn decoders," in *Advances in Neural Information Processing Systems*, 2016, pp. 4790–4798.

[16] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, 2014, pp. 2672–2680.

[17] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros, "Context encoders: Feature learning by inpainting," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2536–2544.

[18] S. Iizuka, E. Simo-Serra, and H. Ishikawa, "Globally and locally consistent image completion," *ACM Transactions on Graphics*, vol. 36, no. 4, p. 107, 2017.

[19] D. Eigen and R. Fergus, "Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2650–2658.

[20] J. Li, F. He, L. Zhang, B. Du, and D. Tao, "Progressive reconstruction of visual structure for image inpainting," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 5962–5971.

[21] A. Saxena, S. H. Chung, and A. Y. Ng, "Learning depth from single monocular images," in *Advances in Neural Information Processing Systems*, 2006, pp. 1161–1168.

[22] A. Saxena, M. Sun, and A. Y. Ng, "Learning 3-d scene structure from a single still image," in *Proceedings of the IEEE International Conference on Computer Vision*. IEEE, 2007, pp. 1–8.

[23] R. Zhang, P.-S. Tsai, J. E. Cryer, and M. Shah, "Shape-from-shading: a survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 8, pp. 690–706, 1999.

[24] S. Suwajanakorn, C. Hernandez, and S. M. Seitz, "Depth from focus with your mobile phone," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3497–3506.

[25] D. Eigen, C. Puhrsch, and R. Fergus, "Depth map prediction from a single image using a multi-scale deep network," in *Advances in neural information processing systems*, 2014, pp. 2366–2374.

[26] F. Liu, C. Shen, G. Lin, and I. D. Reid, "Learning depth from single monocular images using deep convolutional neural fields." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 10, pp. 2024–2039, 2016.

[27] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab, "Deeper depth prediction with fully convolutional residual networks," in *the Fourth International Conference on 3D Vision*. IEEE, 2016, pp. 239–248.

[28] B.-U. Lee, H.-G. Jeon, S. Im, and I. S. Kweon, "Depth completion with deep geometry and context guidance," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 3281–3287.

[29] S. Imran, Y. Long, X. Liu, and D. Morris, "Depth coefficients for depth completion," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2019, pp. 12 438–12 447.

[30] M. Elad and Y. Hel-Or, "A fast super-resolution reconstruction algorithm for pure translational motion and common space-invariant blur," in *The 21st IEEE Convention of the Electrical and Electronic Engineers in Israel*. IEEE, 2000, pp. 402–405.

[31] S. Baker and T. Kanade, "Limits on super-resolution and how to break them," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 9, pp. 1167–1183, 2002.

[32] J.-B. Huang, A. Singh, and N. Ahuja, "Single image super-resolution from transformed self-exemplars," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 5197–5206.

[33] D. Glasner, S. Bagon, and M. Irani, "Super-resolution from a single image," in *IEEE 12th International Conference on Computer Vision*. IEEE, 2009, pp. 349–356.

[34] G. Freedman and R. Fattal, "Image and video upscaling from local self-examples," *ACM Transactions on Graphics*, vol. 30, no. 2, p. 12, 2011.

[35] T. Michaeli and M. Irani, "Nonparametric blind super-resolution," in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 945–952.

[36] A. Singh, F. Porikli, and N. Ahuja, "Super-resolving noisy images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 2846–2853.

[37] H. Chang, D.-Y. Yeung, and Y. Xiong, "Super-resolution through neighbor embedding," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1. IEEE, 2004, pp. I–I.

[38] Y. Jianchao, J. Wright, T. Huang, and Y. Ma, "Image super-resolution as sparse representation of raw image patches," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2008, pp. 1–8.

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

[40] K. I. Kim and Y. Kwon, "Single-image super-resolution using sparse regression and natural image prior," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, no. 6, pp. 1127–1133, 2010.

[41] C. Dong, C. C. Loy, K. He, and X. Tang, "Image super-resolution using deep convolutional networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 2, pp. 295–307, 2016.

[42] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *Proceedings of the European Conference on Computer Vision*. Springer, 2016, pp. 694–711.

[43] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. P. Aitken, A. Tejani, J. Totz, Z. Wang, *et al.*, "Photo-realistic single image super-resolution using a generative adversarial network." in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, no. 3, 2017, p. 4.

[44] A. Sharf, M. Alexa, and D. Cohen-Or, "Context-based surface completion," *ACM Transactions on Graphics*, vol. 23, no. 3, pp. 878–887, 2004.

[45] H. Huang, D. Li, H. Zhang, U. Ascher, and D. Cohen-Or, "Consolidation of unorganized point clouds for surface reconstruction," *ACM Transations on Graphics*, vol. 28, no. 5, pp. 176:1–176:7, 2009.

[46] S. Liu, K.-C. Chan, and C. C. L. Wang, "Iterative consolidation of unorganized point clouds," *IEEE Computer Graphics and Applications*, vol. 32, no. 3, pp. 70–83, May 2012.

[47] A. Hamdi-Cherif, J. Digne, and R. Chaine, "Super-resolution of point set surfaces using local similarities," vol. 37, no. 1, pp. 60–70, 2018.

[48] L. Yu, X. Li, C.-W. Fu, D. Cohen-Or, and P.-A. Heng, "Pu-net: Point cloud upsampling network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2790–2799.

[49] F. Iandola, M. Moskewicz, S. Karayev, R. Girshick, T. Darrell, and K. Keutzer, "Densenet: implementing efficient convnet descriptor pyramids," *arXiv preprint arXiv:1404.1869*, 2014.

[50] https://github.com/chuhuaxian/HF-RGBD.

[51] W. Wang and U. Neumann, "Depth-aware CNN for RGB-D segmentation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 135–150.

[52] O. Ronneberger, P. Fischer, and T. Brox, "U-net: convolutional networks for biomedical image segmentation," in *International Conference on Medical Image Computing and Computer-assisted Intervention*. Springer, 2015, pp. 234–241.

[53] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[54] W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang, "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*, 2016, pp. 1874–1883.

[55] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, *et al.*, "Tensorflow: a system for large-scale machine learning." in *OSDI*, vol. 16, 2016, pp. 265–283.

[56] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[57] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from rgb-d images," in *Proceedings of the European Conference on Computer Vision*. Springer, 2012, pp. 746–760.

[58] K. Lai, L. Bo, X. Ren, and D. Fox, "A large-scale hierarchical multi-view rgb-d object dataset," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2011, pp. 1817–1824.

[59] K. Lai, L. Bo, and D. Fox, "Unsupervised feature learning for 3d scene labeling," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2014, pp. 3050–3057.

[60] J. Kim, J. Kwon Lee, and K. Mu Lee, "Accurate image super-resolution using very deep convolutional networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 1646–1654.

[61] Y. Zhang, Y. Tian, Y. Kong, B. Zhong, and Y. Fu, "Residual dense network for image super-resolution," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2472–2481.

[62] http://vision.middlebury.edu/stereo/data/.

[63] T. Kwok, W. Wan, J. Pan, C. C. L. Wang, J. Yuan, K. Harada, and Y. Chen, "Rope caging and grasping," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, May 2016, pp. 1980–1986.

[64] S. El-Khoury and A. Sahbani, "A new strategy combining empirical and analytical approaches for grasping unknown 3D objects," *Robotics and Autonomous Systems*, vol. 58, no. 5, pp. 497–507, 2010.
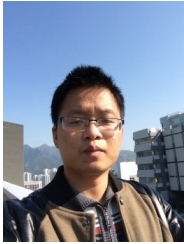
Chuhua Xian is currently an associate professor in the School of Computer Science and Engineering at South China University of Technology. He has a PhD in Computer Science from the State Key Lab of CAD&CG at Zhejiang University in 2012. He was a postdoctoral researcher in CUHK during 11/2013-05/2014 and 09/2015-04/2016. His current research interests include intelligent computer graphics, 3D computer vision, image processing and geometry processing.

ELKHOURY2010497

Dongjiu Zhang is master candidate in the School of Computer Science and Engineering at South China University of Technology. He has a BS in South China Agriculture University. His current research interests include computer graphics, image processing and computer vision.

Chengkai Dai is currently a Ph.D. candidate of the Department of Sustainable Design Engineering at Delft University of Technology. His research area includes robotics, geometry computing and computational design.
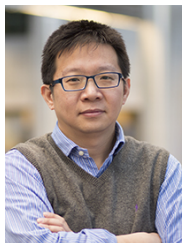
Charlie C. L. Wang is currently a Professor of Mechanical and Automation Engineering and Director of Intelligent Design and Manufacturing Institute at the Chinese University of Hong Kong (CUHK). Before that, he was a tenured Professor and Chair of Advanced Manufacturing at Delft University of Technology (TU Delft), The Netherlands. He received the Ph.D. degree (2002) from Hong Kong University of Science and Technology in mechanical engineering, and is now a Fellow of the American Society of Mechanical Engineers (ASME) and the Hong Kong Institute of Engineers (HKIE). His research areas include geometric computing, intelligent design and advanced manufacturing.