High-Performance Graphics (2021)
N. Binder and T. Ritschel (Editors)

Multi-Resolution Shared Representative Filtering for Real-Time Depth Completion

Yu-Ting Wu[1] Tzu-Mao Li[2] I-Chao Shen[3] Hong-Shiang Lin[4] Yung-Yu Chuang[1]

[1] National Taiwan University, Taiwan
[2] Massachusetts Institution of Technology CSAIL, United States
[3] The University of Tokyo, Japan
[4] FIH Mobile Limited, Taiwan

Abstract

We present shared representative filtering *for real-time high-resolution depth completion with RGB-D sensors. Conventional filtering-based methods face a dilemma when the missing regions of the depth map are large. When the filter window is small, the filter fails to include enough samples. On the other hand, when the window is large, the method could oversmooth depth boundaries due to the error introduced by the extra samples. Our method adapts the filter kernels to the shape of the missing regions to collect a sufficient number of samples while avoiding oversmoothing. We collect depth samples by searching for a small set of similar pixels, which we call the* representatives*, using an efficient line search algorithm. We then combine the representatives using a joint bilateral weight. Experiments show that our method can filter a high-resolution depth map within a few milliseconds while outperforming previous filtering-based methods on both real-world and synthetic data in terms of both efficiency and accuracy, especially when dealing with large missing regions in depth maps.*

CCS Concepts

• ***Computing methodologies*** → ***Computational photography; Image processing;***

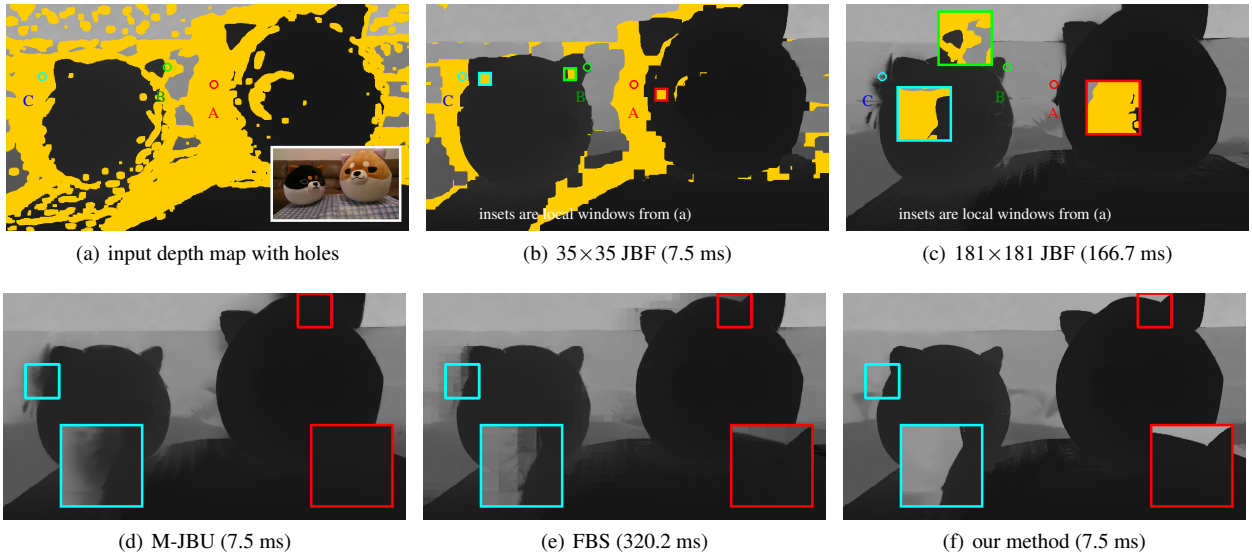
1. Introduction

Modern RGB-D sensors generate high-resolution videos in real-time, enabling applications such as human-computer interaction, environment modeling, autonomous driving, and augmented reality. However, the depth maps generated by these cameras usually contain *holes*: pixels with missing or invalid depth values, due to stereo matching error, range limitation, transparency, reflections, or misalignment between depth and color frames. Figure 1(a) shows a depth map with large holes captured by a modern RGB-D camera, Intel RealSense D435. 44.3% of pixels have missing depth. It is essential to fill the holes efficiently and correctly, as many real-time applications demand complete depth data.

Despite a long history of research, correctly recovering a large number of missing pixels from high-resolution images for real-time applications remains challenging. For applications such as augmented reality, the time budget for each frame is only around 16 milliseconds to achieve at least 60 frames per second, and depth completion is only one step among many. Most previous methods focused on high-quality, offline depth completion, including methods based on global optimization [HCKLH13, PKT*14, SHZ*16], partial differential equations [MFL*12], iterative depth propagation [GLZL13, PP17], and deep-learning-based methods [ZF18]. On the other hand, joint bilateral filtering and upsampling [ED04, PSA*04, KCLU07] are widely adopted in real-time applications for their efficiency thanks to the high parallelism. Methods based on joint bilateral filtering estimate the depth value of an invalid pixel by taking the weighted average of its nearby valid samples within a fixed local window. Unfortunately, while they are effective for filling small holes, their accuracy degenerates quickly as the missing regions become large.

The main challenge of depth completion for large invalid regions is the requirement to search over a large area for valid and reliable samples for filtering. Figure 1 demonstrates several issues of a conventional joint bilateral filter. When the filter size is small, an invalid pixel can remain invalid if no valid samples locate in its local filter window (point A in Figure 1(b)). The depth estimation can also be incorrect if all valid samples in the local filter window are on a different 3D surface (point B in Figure 1(b)). Enlarging the filter size does not solve the problems, since it often leads to oversmoothing and artifacts due to having too many samples from different 3D surfaces (point C in Figure 1(c)). Methods that accelerate joint bilateral filtering for large filter windows (e.g., [RSD*12, BP16]) suffer from the same oversmoothing problem.

We propose a new filtering-based method called *shared representative filtering* that is more accurate than the joint bilateral filtering methods, while achieving real-time performance. Our key idea



(a) input depth map with holes (b) 35×35 JBF (7.5 ms) (c) 181×181 JBF (166.7 ms)

(d) M-JBU (7.5 ms) (e) FBS (320.2 ms) (f) our method (7.5 ms)

Figure 1: *Joint bilateral filtering (JBF) [ED04, PSA*04, KCLU07] is popular for real-time RGB-D depth completion, but it is ineffective when the missing regions are large. (a) Modern RGB-D sensors often produce large regions of invalid depth values (yellow pixels). (b) JBF with a relatively small filter window cannot fill in all the holes. All samples in the kernel could be invalid (point A), or incorrect (for point B, all samples in the kernel are from the foreground while point B itself is in the background). (c) Enlarging the filter size suffers from artifacts due to having too many samples from different 3D surfaces (point C) and is inefficient. Our method (f) can better adapt to hole shapes and preserve depth discontinuities than previous methods, including a method targeting real-time applications: (d) multi-resolution joint bilateral upsampling (M-JBU) [RSD*12] and a method based on global optimization: (e) fast bilateral solver (FBS) [BP16].*

is to use a filter kernel that adapts to the hole shape for each pixel, so that we can collect a sufficient number of valid depth samples without oversmoothing. We develop an efficient algorithm based on line search to sample a small set of most similar depth pixels, which we called the *representatives*. The sampling is amortized over pixels, so that neighboring pixels collectively search for the representatives, reducing the number of required line search steps. We then weight the collected representatives by a joint bilateral weight for computing the filtered depth. We further extend our method to work in a coarse-to-fine fashion to improve the receptive fields and reduce texture copy artifacts.

Our method better preserves depth discontinuities than previous methods (Fig. 1(d) - 1(f)). Our pipeline takes fewer than ten milliseconds for HD resolution (1280 × 720) depth maps on a modern consumer GPU. We achieve 0.92 to 6.02 dB gain of PSNR on a challenging set of simulated RGB-D data, and an average improvement of 1.18 dB on the Middlebury dataset [SHK*14]. Our method also generates more visually accurate results on real-world data.

2. Related Work

We review previous depth completion algorithms for RGB-D images by categories.

Filtering-based approaches. These methods estimate an invalid pixel's depth value by weighted averaging the depth values of its nearby depth samples in a regular window. Joint bilateral filtering based methods [PSA*04, KCLU07] determine sample weights based on color and spatial differences. They are suitable for real-time processing for their high parallelism. However, joint bilateral filtering becomes ineffective when the depth holes are large. The required large kernel size not only considerably increases computational time, but also overblurs the depth discontinuities because of including more samples from different 3D surfaces.

Several fast approximations of bilateral filters have been proposed by filtering in a coarse-to-fine fashion [RSD*12], filtering in a higher-dimensional space where the filtering kernel is separable (e.g., [CPD07, AGDL09, GO12, BP16]), or making the filters hardware-friendly [MAB*17]. Some methods accelerate the filtering with sparse sampling [BCCS12, QHW*13, CYWW14]. He et al. [HST13] proposed a fast alternative edge-preserving filter. However, when the missing regions are large, these methods face the same dilemma as conventional filtering-based methods for being difficult to determine the support of the samples.

Several variants have been proposed to improve joint bilateral filtering for hole filling. Some methods reduce oversmoothing by masking the kernels using edge information [CLL12, LJW14, CCZ*15]. Unfortunately, the per-pixel mask generation is computationally too expensive for real-time applications. Iterative methods gradually close the invalid regions from the boundaries (e.g., [Tel04, GLZL13, HHC14, PP17]). These methods could better preserve depth edges; however, they are less efficient because of the lack of sufficient parallelism.

Optimization-based methods. Several methods explicitly optimize objective functions by specifying smoothness pri-

ors [HCKLH13, PKT*14, SHZ*16, XZC17]. These methods achieve excellent results. However, high computation overhead accompanied by optimization makes them less suitable for real-time processing. Compared to these methods, our approach is more GPU-friendly and better tailored for real-time applications.

Recently some SLAM-based approaches [VKB*18, HK18] also propagate sparse depth data from monocular images in an edge-aware manner by minimizing cost functions. These methods additionally use temporal information for retrieving depth information and achieving temporal smoothness.

Diffusion-based methods. Anisotropic diffusion is a popular tool for image and depth inpainting [BSCB00, MFL*12, AAB17]. These methods formulate inpainting as solving partial differential equations, and can incorporate edge information by controlling the diffusion. However, their iterative nature makes them less efficient.

Deep learning methods. Zhang and Funkhouser [ZF18] and Huang et al. [HWLH19] trained deep neural networks for depth completion. These methods are capable of filling extremely large holes in the depth map. However, they take hundreds of milliseconds to seconds to process a low-resolution depth image. Several data-driven methods focused on completing sparse LiDAR data [JdCW*18, MCK19, XZS*19, CYLU19, QCZ*19]. These methods typically take about 70 to 100 milliseconds to process a 1216×352 LiDAR image [USS*17]. Furthermore, data-driven methods require collecting massive training data, and acquiring accurate reference depth maps for arbitrary images is not trivial.

3. Shared Representative Filtering

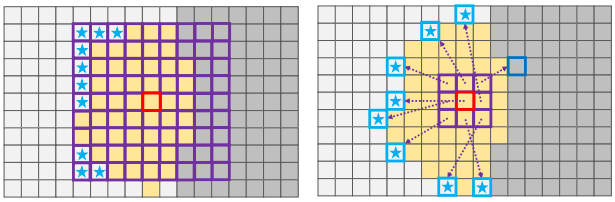
Our method builds on filtering-based approaches. Given an incomplete depth map D and a complete color image C, for an invalid pixel i in the depth map, we want to estimate its depth using nearby valid pixels:

$$D_i = \frac{\sum_{j \in N_i} w_{ij} D_j}{\sum_{j \in N_i} w_{ij}}, \tag{1}$$

where N_i is some neighborhood around pixel i, D_j is the depth of the pixel j, and w_{ij} is the weight which depends on the similarity between pixels i and j by considering the spatial and color information. The pixel j needs to have a valid depth value for correctly gathering depth information.

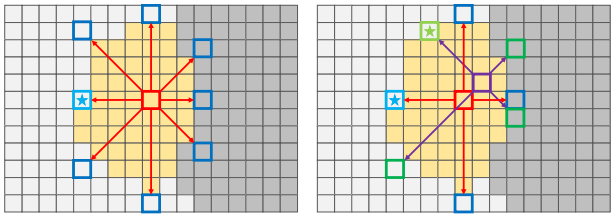
Previous filtering-based methods typically use a square window for N_i (e.g., [KCLU07, RSD*12]). The square window is both inefficient and inaccurate when the invalid regions are large. Instead, our method chooses a neighborhood that adapts to the shapes of the invalid regions. The adaptation allows us to avoid including too many samples from different 3D surfaces while being efficient.

Figure 2 illustrates the difference between our method and conventional filtering-based approaches. Given a pixel i, we collect a set of valid and reliable depth samples to form the neighborhood N_i. Our method works by finding a set of *representative* pixels j for each pixel, such that each j has a valid depth value and is most similar to i in terms of spatial and color distance (Section 3.1). We then combine the representatives in the neighborhood by taking a weighted average (Section 3.2). Finally, to further speed up the method and increase the receptive field size, we propose a multi-resolution extension to our algorithm (Section 4).



(a) regular filtering (b) our filtering

Figure 2: *Our method adapts to the geometry of the invalid regions. Yellow pixels are pixels with invalid depths, while white and gray pixels represent pixels from two different 3D surfaces. (a) Conventional filtering-based methods require a large filter window to include sufficient samples for estimating the missing depth value. However, most samples within the filter are either invalid (49/81) or located on a different 3D surface (22/81). (b) Our method finds each neighboring pixel a valid and reliable representative depth, adapting the filter to the shape of the hole. Due to higher-quality samples, we could efficiently predict the depth values of invalid pixels with a small local filter.*

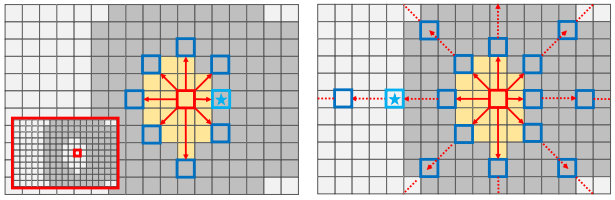


(a) per-pixel searching (b) joint searching

Figure 3: *We sample lines to search for the* representative *pixels. (a) At each pixel with an invalid depth value (marked yellow), we uniformly sample a few directions for searching the best representative based on a similarity measurement. In this case, the neighbor labeled with a light blue asterisk is selected. (b) We stratify the sampling directions of adjacent pixels to search for different directions.*

3.1. Searching for representatives

To find a representative for each pixel with invalid depth without exhaustively searching a large neighborhood, we sparsely sample the nearby valid pixels. Figure 3 demonstrates our approach for searching valid samples around an invalid pixel. Starting at a pixel i, we uniformly sample K directions. Along each direction, we walk along the line to gather valid depth samples. Importantly, we do not stop at the first valid depth sample we meet. Instead, we continue the search for an extra number of steps. The extra steps, which we call the *non-local samples*, allow us to take samples that are occluded by a valid but dissimilar pixel into account (Figure 4(a) illustrates the idea, and Figure 5 compares our reconstruction with and without the non-local samples). After all valid depth samples



(a) w/o non-local samples (b) w/ non-local samples

Figure 4: ***Non-local sampling.*** *(a) If we stop walking along the line after reaching the first valid sample, the method can not correctly handle the case where foreground objects (gray) surround a background hole (yellow). In this case, all valid samples found along sampling directions belong to foreground objects. We show the ground truth at the bottom left side for reference. (b) By continuing along with the sampling directions for an extra number of steps, our method can locate a better representative (labeled with a light blue asterisk).*

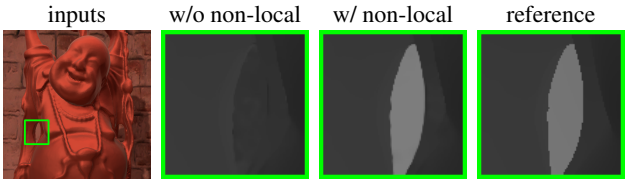


Figure 5: *We show an example of the ablation study for the non-local sampling described in Figure 4. With the non-local sampling scheme, our method can better estimate the depth values, evident as shown in the holes of the buddha.*

along the sampling directions are collected, we select a representative among them based on their similarities to pixel i.

We measure the similarity of representatives using the following cost E_{ij}, defined with spatial and color differences as in joint bilateral filtering [PSA*04, KCLU07] and non-local means [BCM05]:

$$E_{ij} = \alpha_s \|i-j\|^2 + \alpha_c \left\|C_i - C_j\right\|^2 + \alpha_p \left\|P_i - P_j\right\|^2, \qquad (2)$$

where $\|i-j\|^2$ is the spatial distance between pixels i and j; C_i and C_j are the color vectors at pixel i and j; P_i and P_j are vectors consisting of all color pixels in a 3×3 patch centered at pixel i and j; and α_s, α_c, and α_p are weights to balance these three terms. We select the sample with the smallest cost as the representative.

One of our core ideas that reduce the number of required sampling directions for each pixel is to stratify the search direction to take account for different sets of samples per pixel (Figure 3(b)). At each pixel, we choose a different initial sampling direction so that the adjacent pixels could discover different representatives. We then combine the representatives during the weighted average (Section 3.2). In our implementation, we use a hash function and set the initial angle to $(3*(y\%3)+x\%9)*(360/K)/9$ degree, where % is a modulo operation. Other hash functions can be used.

Figure 6 shows the relationship between the number of sampling

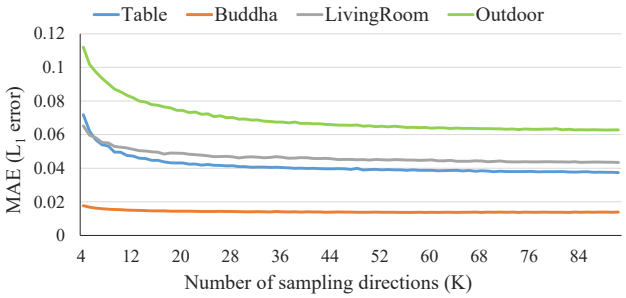


Figure 6: *We show the relationship between the number of sampling directions and the accuracy of the recovered depth map, measured by the mean absolute error (in meter). Using more sampling directions could give more accurate results, at the cost of higher computational cost. We typically use* $8-24$ *sampling directions per pixel.*

directions and the accuracy of the recovered depth map for four synthetic scenes. As shown in the figure, the mean absolute error decreases as the number of sampling directions grows consistently. It also quickly converges within a few samples (about $8-24$ sampling directions per pixel, depending on the scene complexity).

3.2. Reconstructing depth

The depth value D_i of an invalid pixel i is estimated by weighted averaging the depth values of the representatives in its local neighborhood. The weighting scheme is similar to the cost E_{ij} for determining the representative:

$$D_i = \frac{\sum_{j\in\Omega} w_s(i,j)w_c(i,R_j)w_p(i,R_j)D_{R_j}}{\sum_{j\in\Omega} w_s(i,j)w_c(i,R_j)w_p(i,R_j)}, \qquad (3)$$

where Ω denotes the small neighborhood around i defined by a local regular window. R_j is the representative of pixel j in Ω (for a pixel with valid depth, the representative is itself; otherwise, the representative is found by using the procedure described earlier), and D_{R_j} is its depth value. The weighting functions w_s, w_c and w_p measure the spatial, color and structural distances between the target point and the filter sample. We define the weighting functions as Gaussian falloffs as in conventional joint bilateral filtering:

$$\begin{aligned} w_s(i,j) &= \exp(-\|i-j\|^2/2\sigma_s^2), \\ w_c(i,j) &= \exp(-\left\|C_i-C_j\right\|^2/2\sigma_c^2), \\ w_p(i,j) &= \exp(-\left\|P_i-P_j\right\|^2/2\sigma_p^2). \end{aligned} \qquad (4)$$

We use the spatial distance between i and j instead of the one between i and R_j, since the representatives are non-local by nature, and the distance between i and R_j has been considered during the representative search. We usually set the variances of color and patch σ_c^2, σ_p^2 to the inverse of the similarity weights α_c, α_p, while tuning the spatial variance σ_s^2 differently, since for the representative searching we want to search for the samples far away.

Gastal and Oliveira adopted a similar idea of stratified line search for image matting [GO10]. Compared to their work, our method possesses several features that are specifically designed for depth

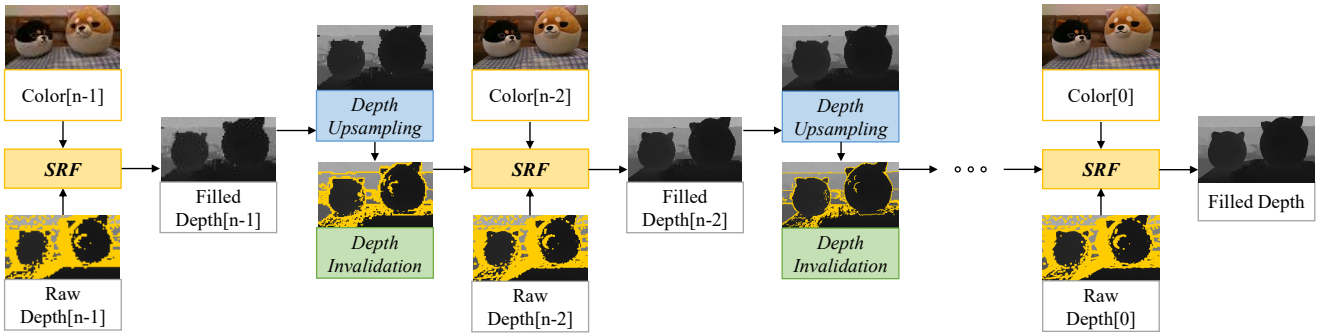


Figure 7: *We illustrate the flow of our multi-resolution joint representative filtering. Given input color and depth images, our method first builds n-level image pyramids for the color and depth images:* 0 *is the finest level, and* $n-1$ *is the coarsest one. Then we apply our shared representative filtering (SRF) from coarse to fine. After filling the holes at a coarse level, we upsample the low-resolution result to a finer level. We remove unreliable depth values with large gradients at object boundaries during the depth invalidation stages. The depth values of those unreliable pixels are then estimated using the proposed SRF method. The process continues until we reach the finest level.*

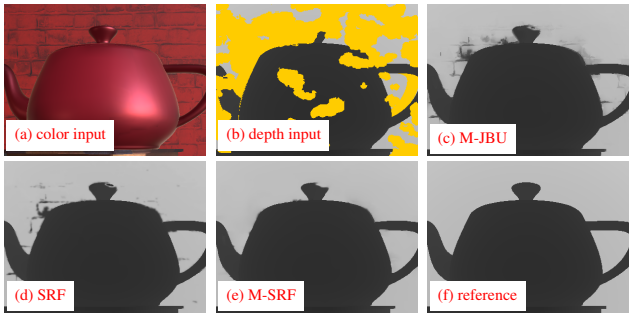


Figure 8: *(a) and (b) show a case of missing regions with similar foreground and background color. Our multi-resolution approach (e) can reduce the texture copy artifacts compared to the single resolution one (d). It is worth noting that although M-JBU [RSD*12] also runs in a multi-resolution manner, it still suffers from texture copy artifacts due to the insufficient background data inside the filter window, as shown in (c).*

completion. First, unlike their search process which stops at the first valid sample, our method continues the search with extra non-local samples. This way, we can successfully recover the depth values of a background hole surrounded by foreground objects. Second, as introduced in the next section, our multi-resolution approach can significantly reduce texture copy artifacts that are common in color-guided depth inpainting. Finally, our bilateral-filtering-based approach is especially suitable for processing depth maps because they are usually smooth except for object boundaries.

4. Multi-Resolution Shared Representative Filtering

For high-resolution inputs, the cost of searching a per-pixel representative increases because each direction requires more steps to find the first valid depth sample. To reduce the searching cost, we employ a multi-resolution scheme with a push/pull approach [GGSC96], inspired by previous studies [RSD*12].

Figure 7 illustrates the flowchart of our multi-resolution algorithm. We build an n-level image pyramid where the coarser level has half of the spatial resolution to its successive finer level. We generate the low-resolution images by rendering the input color and depth images into smaller render targets. We first use our shared representative filtering introduced in Section 3 for filling the holes in the coarsest-level depth map, based on the guidance of the coarsest-level color image. After we finish processing the coarsest level, for each remaining level l, we upsample the result from its previous coarser level $(l+1)$ to the resolution at level l. The upsampling process could produce inaccurate depth values at object boundaries. Therefore, we apply a depth invalidation operation after upsampling for removing the unreliable pixels. We compute depth gradients with a 3×3 Sobel approximation and remove the depth pixels whose gradients are larger than a threshold (0.15 in our implementation). Next, we fill the holes in the l-level depth map by our shared representative filtering based on the l-level color image. The process continues until we reach the finest-level.

Our multi-resolution approach not only improves the efficiency, but also produces better results than the single-resolution version in most cases. Filling the holes in a coarse-to-fine manner allows us to generate smoother results and greatly reduce texture copy artifacts. Figure 8 shows such an example. However, we observed that more levels do not always lead to better results. When the spatial resolution is too small, the image features in color images could get lost or blurred, making them unable to provide accurate guidance for depth completion. The number of levels is determined according to the image resolution and content. For the test cases in this paper, our experiments show that using 3 or 4 levels offers a good compromise between efficiency and quality.

5. Results and Discussions

We implemented our proposed algorithm on top of the GPU shaders in the Unity game engine. All results were generated on a machine with an Intel Core i5-7400K at 3.0 GHz, 16-GB of RAM, and an NVIDIA GeForce GTX 2080 Ti graphics card.

5.1. Test cases

We assess the performance and accuracy of our method, using both real-world and synthetic RGB-D data, including:

Simulated RGB-D data. We create 3D scenes and render them with a pair of virtual color and depth cameras in Unity. We simulate the depth holes using heuristics based on distance or material properties. Pixels farther away from the camera or with more shiny material have a higher probability of becoming invalid. We render referenced depth maps for numerical validation. The supplementary material describes the generation process. All images have a resolution of 1280×720.

Middlebury 2014 dataset. We adapt the high-resolution (around 3000×2000) real-world RGB-D data reference from Scharstein et al. [SHK*14]. These images contain a very small number of invalid depth pixels and therefore are suitable for numerical error evaluation. We simulate the depth holes using 2D Perlin noise [Per85]. For each depth map, we generate two masks based on Perlin noise and report the average error of the two.

Real-world data. We capture several pairs of color and depth images with an Intel RealSense D435 RGB-D camera to evaluate our method on data with real invalid depth regions. We observe that the captured depth maps contain noise. Therefore we preprocess them using a small bilateral filter [TM98] on pixels with valid depth values before the depth completion. We also remove the depth values of pixels with large gradients because the depth data along object boundaries are inaccurate. All images have a resolution of 1280×720.

Our test cases contain large holes caused by occlusions, specular reflection, and extensive depth ranges. Some of them have similar colors of the foreground and background objects and can confuse joint bilateral filters. Some test cases contain complex occlusion patterns or thin geometry.

5.2. Comparisons

We compare with the standard single-pass joint bilateral filtering/upsampling (JBF) [ED04, PSA*04, KCLU07], the multi-resolution version (M-JBU) [RSD*12], the fast bilateral solver (FBS) [BP16] and the single-pass and multi-resolution versions of our shared representative filtering method (SRF and M-SRF) under different time constraints. For JBF, we implemented Kopf et al.'s method [KCLU07]. For M-JBU, Richardt et al.'s method [RSD*12] consists of both depth inpainting and filtering algorithms. We only implemented their multi-resolution geometry fill-in algorithm since we focus on depth completion. We implement all the above methods in Unity's GPU shaders, except for the fast bilateral solver. For FBS, we tried the Python implementations provided by the authors and the C++ implementation in OpenCV, both running on CPUs. The author's version produces better results while the OpenCV code is 2×-4× faster. We used the author's version in this paper.

Parameters. To meet a given time budget, for JBF and M-JBU, we tune the filter window radius (2× the spatial standard deviation). For our SRF and M-SRF, we tune the number of sampling lines K.

For joint-bilateral filtering (JBF and M-JBU) and our shared representative filtering (SRF and M-SRF), we choose the color standard deviation that produces the lowest L_1 and L_2 errors for the simulated RGB-D cases and the Middlebury cases, and the value that produces the most visually pleasing results for the real-world data per-scene. In our test cases, we choose the best color standard deviation in the range of $[0.01, 0.20]$. For multi-resolution methods (M-JBU and M-SRF), we additionally optimize the number of image pyramid levels. We choose to use the level that produces the lowest error in the range of $[2, 4]$ for each scene. We provide the exact parameters used by all methods in the supplement.

We use the same setting for the rest parameters of our SRF and M-SRF. We set the spatial standard deviation for the representative searching to 0.12 times the image width of the coarsest level. We only employ non-local sampling at the coarsest level. The distance between adjacent non-local samples along a sampling line is set to 0.05 times the image width of the coarsest level. The patch standard deviation is fixed to 0.1. For performance consideration, we do not involve the patch difference term at the finest level. The spatial standard deviation for the depth reconstruction is fixed to 1.5, resulting in a 7×7 window.

For FBS, we find the results of depth completion are very sensitive to the filter parameters. For simulated RGB-D cases and the Middlebury cases, we employ the Bayesian optimization [Nog] for exploring the combination of parameters which optimizes L_1 and L_2 errors. For real-world data, we tune the parameters by hands for the most visually pleasing results. We set the extent of spatial support to $[3, 32]$; the extent of luminance and chroma supports are chosen in the range of $[2.55, 38.25]$; the smoothness multiplier is selected from $[1, 256]$, and the maximum iteration of the conjugate gradient solver to 25. We obtain the best parameters by running 200 iterations of Bayesian optimization. The original fast bilateral solver paper [BP16] describes a post-processing step using another edge-aware filter [GO11]. We do not include the post-processing as it requires additional processing cost and parameter tuning.

5.2.1. Comparisons of simulated RGB-D data

As we target real-time applications, the depth completion must finish within a short given time. We allocate fixed time budgets for filtering-based methods: 15 milliseconds for the single-pass ones (JBF and SRF), and 7.5 milliseconds for the multi-resolution ones (M-JBU and M-SRF). We do not allocate fixed time budgets for FBS since it is difficult to control its execution time. In our experiments, FBS takes about 0.3 - 0.5 seconds to construct the bilateral grid and solve the optimization, using the unoptimized Python code. The results of the three methods are shown in Figure 9.

For the simulated RGB-D data (Figure 9), we evaluate the results numerically by calculating their mean absolute error (L_1 error) and Peak Signal-to-Noise Ratio (PSNR) to the reference images. Our method significantly outperforms other methods, both visually and numerically. JBF fails to complete all the missing regions within the 15-millisecond time budget. Both JBF and M-JBU oversmooth the object boundaries (see insets), because they include too many foreground samples for a background pixel, or vice versa. Unfortunately, these depth discontinuities are precisely the places where occlusion and misalignment can happen. Similarly, FBS overblurs depth discontinuities and produces significant errors in regions of complex occlusion, as shown in the insets of *Buddha* and *Outdoor*.

Figure 9: *Equal-time comparisons for four synthetic RGB-D scenes: Table, Buddha, LivingRoom, and Outdoor. For each scene, the first column shows the input depth map (top) and the color image (bottom). In the depth maps, yellow pixels represent missing depth values. The second column shows results of single-pass joint bilateral upsampling [KCLU07] (JBF) and our method (SRF), while the third column shows multi-resolution versions (M-JBU [RSD*12] and M-SRF). We allocate 15 ms for the single-pass methods and 7.5 ms for the multi-resolution ones. The Python version fast bilateral solver (FBS) [BP16] takes about 0.3 to 0.5 second, running on CPU. Its results are still listed for quality comparison. The peak signal-to-noise ratio (PSNR) is denoted for each depth map. In the given time budget, JBF fails to complete all missing regions (therefore, we do not measure its error), while M-JBU and FBS oversmooth depth boundaries. Our method demonstrates significant improvements both numerically and visually, and shows* 0.92 *to* 6.02 *dB of PSNR gain over previous methods.*

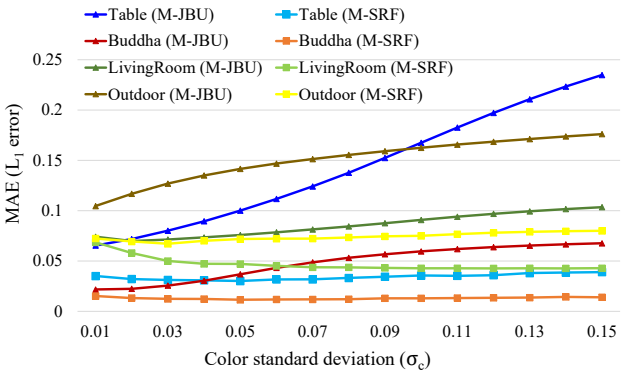


Figure 10: ***Ablation study on the relationship between color standard deviation and accuracy.*** *We label the error plots of M-JBU and M-SRF with triangles and squares, respectively. The error plots for the same scene are shown with the similar color. Our method (M-SRF) consistently produces smaller errors than M-JBU, regardless of the chosen color standard deviation for each scene.*

Our method significantly improves the quality at the boundaries, thanks to the representatives searching. The multi-resolution reconstruction further improves efficiency and produces better results. For regions with intricate occlusion patterns such as the inset of *Buddha*, our non-local samples help to find better representatives.

Trade-offs between execution time, filter size, and accuracy. We further analyze the trade-offs between the time budgets and accuracy using the simulated RGB-D data. Figure 11 shows the graphs for MAE. Single-pass JBF can not fill all holes within 30 milliseconds, so we do not include it in the figure. For conventional filters (JBF and M-JBU), the only way to increase samples is to enlarge the filter size. The error, however, does not necessarily decrease with the increased filter size. This is because a larger filter could also include samples located on other surfaces and introduce bias. For our methods, the error reduces as the number of lines increases in the beginning, and is robust to a larger number of samples, thanks to the representative searching. For all test cases, our methods produce much lower errors comparing to others.

We also vary the number of levels in the experiments. Our single-pass method is significantly faster than the single-pass JBF. It fills all holes within $\sim$ 10 milliseconds and produces much lower error than JBF and M-JBU. For both JBF and our method, using a multi-resolution approach dramatically reduces the execution time for the same receptive field size. However, M-JBU suffers from additional error because samples from different 3D surfaces are included for filtering in coarser levels. Our M-SRF preserves the advantage of the single-pass one, but significantly improves performance.

Relationship between color standard deviation and accuracy. Previous experiments optimize color standard deviation (σ_c) for each scene. Figure 10 studies how the MAE changes with σ_c for M-JBU and M-SRF. We fixed all other parameters except for σ_c. As shown in the figure, the best σ_c is scene dependent. Scenes with complex layout and occlusion (*Table* and *Outdoor*) favor smaller σ_c for better preserving depth discontinuities, while scenes with simple geometry (*LivingRoom*) favor relatively larger σ_c for producing smoother images. Our method consistently produces smaller errors than M-JBU regardless of the chosen parameters for each scene, meaning it is less sensitive to the color standard deviation.

	MAE ↓	PSNR ↑
M-JBU [RSD*12]	0.0429	32.14
FBS [BP16]	0.0404	33.95
M-SRF (our)	**0.0323**	**35.13**

Table 1: ***Quantitative evaluation on Middlebury data [SHK*14].*** *We allocate 20 milliseconds for M-JBU and M-SRF. The unoptimized FBS takes about 1.2 to 2.0 seconds. Compared to M-JBU and FBS, our method reduces error by 0.753× and 0.800×, and achieves 2.99 dB and 1.18 dB gains of PSNR.*

5.2.2. Comparisons of Middlebury 2014 data

Table 1 shows the comparisons of average MAE and PSNR over 23 cases in Middlebury data. Because of high resolutions (around 3000 × 2000) of the images in the Middlebury 2014 dataset [SHK*14], we allocate 20 milliseconds for M-JBU and our M-SRF. FBS takes about 1.2 - 2.0 seconds for constructing the bilateral grids and solving the optimization. The average MAE of our method is 0.753× and 0.800× smaller than M-JBU and FBS, respectively. In terms of PSNR, the average gains over M-JBU and FBS are 2.99 dB and 1.18 dB, respectively. Figure 12 shows the visual results for the *Adirondack* and *Umbrella* scenes. In both cases, the foreground objects (book and umbrella) have a similar color to the background wall. As shown in the figures, our method can better preserve the depth discontinuities between the foreground and the background objects than previous methods. For comparisons of other scenes, please refer to the supplementary materials.

5.2.3. Comparisons of real-world data

The depth completion results of real-world data captured by Intel RealSense D435 are shown in Figure 1 and Figure 13. We can only compare the results visually for real data because we do not have ground-truth depth maps. For *Puppies* (Figure 1) and *Work*, our method outperforms previous methods concerning boundaries preservation. The improvement is particularly evident at image boundaries where fewer samples are within the local window (see the insets). In *Man*, previous methods fail to recover the background depth between the man's head and hand. Our non-local samples can retrieve the correct background representatives. *Office* is a challenging case because the lighting on the wall causes significant brightness variations, and the depth values at the wall boundaries are inaccurate. JBF, M-JBU, FBS, and our SRF all incorrectly propagate the door's depth values to the wall. Our M-SRF correctly reconstructs most depth values.

5.3. Limitations

While our method significantly improves over joint bilateral filtering, it inherits some of its limitations. Our method can still over-smooth depth maps when color and patch similarity does not lead to depth similarity (e.g., in *Table*), or inaccurate depth estimation at the boundaries (e.g., in *Man*). Significant local depth variation

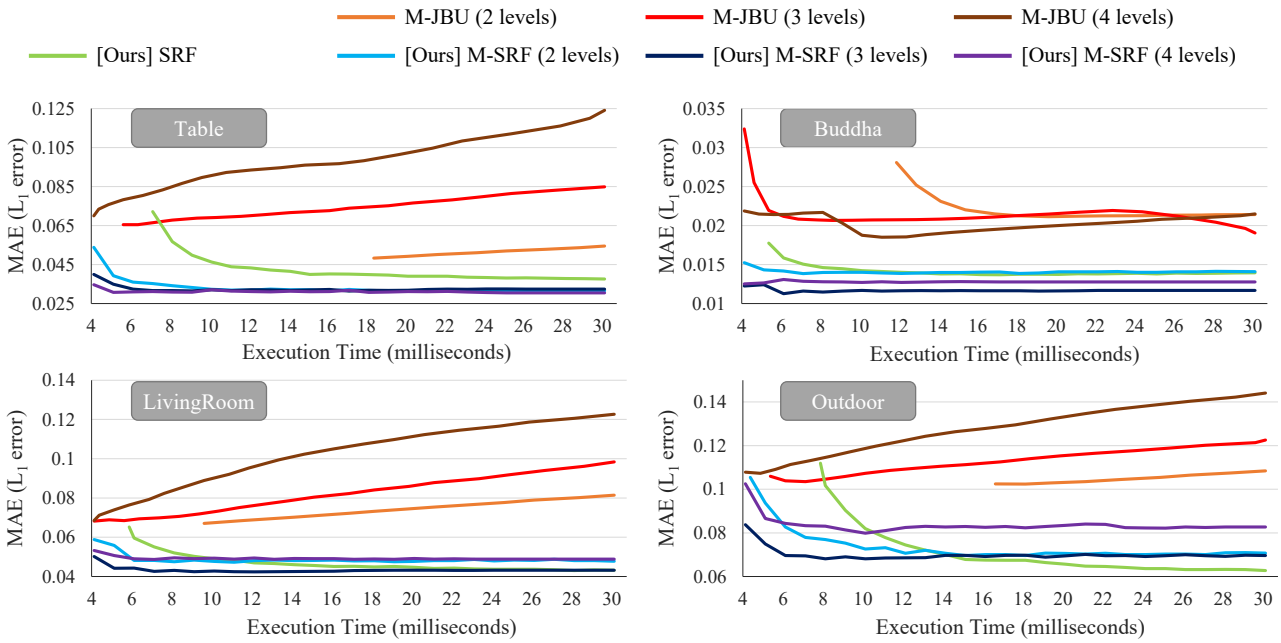


Figure 11: ***Comparisons of the L_1 error on the synthetic data under different filter sizes and number of samples.*** *Our method consistently achieves the lowest error, regardless of the time budgets and the number of levels. Using larger kernels in M-JBU does not necessarily result in smaller errors due to the extra samples' bias. In contrast, the errors of our method reduce with more lines robustly because of the representative searching. We do not include single-pass JBF since it could not fill all holes within the* 30*-millisecond time budget.*

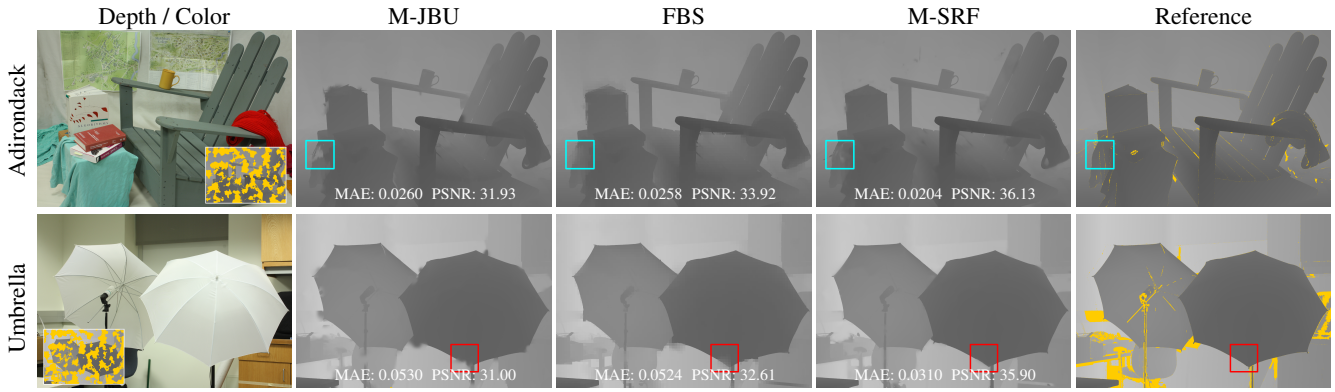


Figure 12: ***Comparisons of the Adirondack and Umbrella scenes in Middlebury 2014 data set [SHK*14].*** *We allocate 20 milliseconds for M-JBU and M-SRF. The unoptimized FBS takes about 1.2 to 2.0 seconds. Our method can better preserve the depth discontinuities and achieve lower MAE and higher PSNR than M-JBU and FBS.*

caused by thin or high-curvature geometry is also challenging for our algorithm, as shown in *Outdoor*. Finally, our depth invalidation process based on pixel gradients could fail at times. Depth values of missing regions could be incorrectly predicted based on inaccurate depth samples that are not invalidated.

6. Conclusion and Future Work

We propose *shared representative filtering*, a real-time depth completion algorithm for filling large missing regions in RGB-D data. We observe that conventional joint bilateral filtering is not effective in dealing with large holes of depth maps with the regular filter kernels. Our representative searching adapts the filter kernels to the geometry of missing regions efficiently by locating reliable samples. Our multi-resolution filter further improves both accuracy and efficiency. Experiments show that our algorithm can accurately predict pixels' depth values in large missing regions while achieving real-time frame rate on HD resolution images.

In the future, we would like to extend our method to the temporal domain for RGB-D videos. We believe that exploiting tem-

Figure 13: ***Equal-time comparisons of real data captured by Intel RealSense D435.*** *Same as Figure 9, we allocate 15 milliseconds for the single-pass methods and 7.5 milliseconds for the multi-resolution ones. We also list the results of single-pass joint bilateral upsampling with large enough kernels for filling all holes and the ones of fast bilateral filter (FBS) for quality comparisons. Our method reconstructs depth maps significantly better than other methods by avoiding to oversmooth the depth boundaries.*

poral coherence could not only reduce per-frame cost, but also improve temporal consistency. We also plan to leverage the success of recent research on scene understanding, such as image segmentation [BKC17] or depth edge detection [HK18], to better preserve the depth boundaries.

References

[AAB17] ATAPOUR-ABARGHOUEI A., BRECKON T.: DepthComp: real-time depth image completion based on prior semantic scene segmentation. In *Proceedings of the British Machine Vision Conference* (2017), pp. 58:1–58:13. 3

[AGDL09] ADAMS A., GELFAND N., DOLSON J., LEVOY M.: Gaussian kd-trees for fast high-dimensional filtering. *ACM Trans. Graph. (Proc. SIGGRAPH) 28*, 3 (2009), 21. 2

[BCCS12] BANTERLE F., CORSINI M., CIGNONI P., SCOPIGNO R.: A low-memory, straightforward and fast bilateral filter through subsampling in spatial domain. *Comput. Graph. Forum 31*, 1 (2012), 19–32. 2

[BCM05] BUADES A., COLL B., MOREL J.-M.: A non-local algorithm for image denoising. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2005), vol. 2, pp. 60–65. 4

[BKC17] BADRINARAYANAN V., KENDALL A., CIPOLLA R.: SegNet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell. 39*, 12 (2017), 2481–2495. 10

[BP16] BARRON J. T., POOLE B.: The fast bilateral solver. In *Europeon Conference on Computer Vision* (2016), pp. 617–632. 1, 2, 6, 7, 8

[BSCB00] BERTALMIO M., SAPIRO G., CASELLES V., BALLESTER C.: Image inpainting. In *SIGGRAPH* (2000), pp. 417—-424. 3

[CCZ*15] CHEN C., CAI J., ZHENG J., CHAM T. J., SHI G.:

Kinect depth recovery using a color-guided, region-adaptive, and depth-selective framework. *ACM Trans. Intell. Syst. Technol. 6*, 2 (2015), 12:1–12:19. 2

[CLL12] CHEN L., LIN H., LI S.: Depth image enhancement for kinect using region growing and bilateral filter. In *International Conference on Pattern Recognition* (2012), pp. 3070–3073. 2

[CPD07] CHEN J., PARIS S., DURAND F.: Real-time edge-aware image processing with the bilateral grid. *ACM Trans. Graph. (Proc. SIGGRAPH) 26*, 3 (2007), 103. 2

[CYLU19] CHEN Y., YANG B., LIANG M., URTASUN R.: Learning joint 2D-3D representations for depth completion. In *International Conference on Computer Vision* (2019), pp. 10022–10031. 3

[CYWW14] CHEN W., YUE H., WANG J., WU X.: An improved edge detection algorithm for depth map inpainting. *Optics and Lasers in Engineering 55* (2014), 69–77. 2

[ED04] EISEMANN E., DURAND F.: Flash photography enhancement via intrinsic relighting. *ACM Trans. Graph. (Proc. SIGGRAPH) 23*, 3 (2004), 673–678. 1, 2, 6

[GGSC96] GORTLER S. J., GRZESZCZUK R., SZELISKI R., COHEN M. F.: The lumigraph. In *SIGGRAPH* (1996), pp. 43–54. 5

[GLZL13] GONG X., LIU J., ZHOU W., LIU J.: Guided depth enhancement via a fast marching method. *Image Vision Comput. 31*, 10 (Oct. 2013), 695–703. 1, 2

[GO10] GASTAL E. S. L., OLIVEIRA M. M.: Shared sampling for real-time alpha matting. *Comput. Graph. Forum (Proc. Eurographics) 29*, 2 (2010), 575–584. 4

[GO11] GASTAL E. S. L., OLIVEIRA M. M.: Domain transform for edge-aware image and video processing. *ACM Trans. Graph. (Proc. SIGGRAPH) 30*, 4 (2011), 69:1–69:12. 6

[GO12] GASTAL E. S. L., OLIVEIRA M. M.: Adaptive manifolds for real-time high-dimensional filtering. *ACM Trans. Graph. (Proc. SIGGRAPH) 31*, 4 (2012), 33:1–33:13. 2

[HCKLH13] HERRERA C. D., KANNALA J., LADICKÝ L., HEIKKILÄ J.: Depth map inpainting under a second-order smoothness prior. In *Image Analysis* (2013), pp. 555–566. 1, 3

[HHC14] HUANG Y., HSU T., CHIEN S.: Edge-aware depth completion for point-cloud 3D scene visualization on an RGB-D camera. In *IEEE Visual Communications and Image Processing Conference* (2014), pp. 422–425. 2

[HK18] HOLYNSKI A., KOPF J.: Fast depth densification for occlusion-aware augmented reality. *ACM Trans. Graph. (Proc. SIGGRAPH Asia) 37*, 6 (Dec. 2018). 3, 10

[HST13] HE K., SUN J., TANG X.: Guided image filtering. *IEEE Trans. Pattern Anal. Mach. Intell. 35*, 6 (2013), 1397–1409. 2

[HWLH19] HUANG Y.-K., WU T.-H., LIU Y.-C., HSU W. H.: Indoor depth completion with boundary consistency and self-attention. In *The IEEE International Conference on Computer Vision Workshops* (2019), pp. 1070–1078. 3

[JdCW*18] JARITZ M., DE CHARETTE R., WIRBEL E., PERROTTON X., NASHASHIBI F.: Sparse and dense data with CNNs: Depth completion and semantic segmentation. In *International Conference on 3D Vision* (2018), pp. 52–60. 3

[KCLU07] KOPF J., COHEN M. F., LISCHINSKI D., UYTTENDAELE M.: Joint bilateral upsampling. *ACM Trans. Graph. (Proc. SIGGRAPH) 26*, 3 (2007). 1, 2, 3, 4, 6, 7

[LJW14] LE A. V., JUNG S.-W., WON C. S.: Directional joint bilateral filter for depth images. *Sensors 14*, 7 (2014), 11362–11378. 2

[MAB*17] MAZUMDAR A., ALAGHI A., BARRON J. T., GALLUP D., CEZE L., OSKIN M., SEITZ S. M.: A hardware-friendly bilateral solver for real-time virtual reality video. In *High Performance Graphics* (2017). 2

[MCK19] MA F., CAVALHEIRO G. V., KARAMAN S.: Self-supervised sparse-to-dense: Self-supervised depth completion from LiDAR and monocular camera. In *IEEE International Conference on Robotics and Automation* (2019), IEEE, pp. 3288–3295. 3

[MFL*12] MIAO D., FU J., LU Y., LI S., CHEN C. W.: Texture-assisted Kinect depth inpainting. In *IEEE International Symposium on Circuits and Systems* (2012), pp. 604–607. 1, 3

[Nog] NOGUEIRA F.: Bayesian Optimization: Open source constrained global optimization tool for Python, 2014–. URL: https://github.com/fmfn/BayesianOptimization. 6

[Per85] PERLIN K.: An image synthesizer. *SIGGRAPH Comput. Graph. 19*, 3 (1985), 287–296. 6

[PKT*14] PARK J., KIM H., TAI Y., BROWN M. S., KWEON I. S.: High-quality depth map upsampling and completion for RGB-D cameras. *IEEE Trans. Image Process. 23*, 12 (2014), 5559–5572. 1, 3

[PP17] PRAHARA A., PRANOLO A.: Depth inpainting scheme based on edge guided non local means. In *International Conference on Science in Information Technology* (2017), pp. 706–710. 1, 2

[PSA*04] PETSCHNIGG G., SZELISKI R., AGRAWALA M., COHEN M., HOPPE H., TOYAMA K.: Digital photography with flash and no-flash image pairs. *ACM Trans. Graph. (Proc. SIGGRAPH) 23*, 3 (Aug. 2004), 664–672. 1, 2, 4, 6

[QCZ*19] QIU J., CUI Z., ZHANG Y., ZHANG X., LIU S., ZENG B., POLLEFEYS M.: DeepLiDAR: Deep surface normal guided depth prediction for outdoor scene from sparse lidar data and single color image. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019), pp. 3308–3317. 3

[QHW*13] QI F., HAN J., WANG P., SHI G., LI F.: Structure guided fusion for depth map inpainting. *Pattern Recogn. Lett. 34*, 1 (Jan. 2013), 70–76. 2

[RSD*12] RICHARDT C., STOLL C., DODGSON N. A., SEIDEL H.-P., THEOBALT C.: Coherent spatiotemporal filtering, upsampling and rendering of RGBZ videos. *Comput. Graph. Forum (Proc. Eurographics) 31*, 2 (2012), 247–256. 1, 2, 3, 5, 6, 7, 8

[SHK*14] SCHARSTEIN D., HIRSCHMÜLLER H., KITAJIMA Y., KRATHWOHL G., NEŠIĆ N., WANG X., WESTLING P.: High-resolution stereo datasets with subpixel-accurate ground truth. In *German conference on pattern recognition* (2014), pp. 31–42. 2, 6, 8, 9

[SHZ*16] SONG X., HUANG H., ZHONG F., MA X., QIN X.: Edge-guided depth map enhancement. In *Proceedings of the International Conference on Pattern Recognition* (2016), pp. 2758–2763. 1, 3

[Tel04] TELEA A.: An image inpainting technique based on the fast marching method. *J. Graph. Tools 9*, 1 (2004), 23–34. 2

[TM98] TOMASI C., MANDUCHI R.: Bilateral filtering for gray and color images. In *International Conference on Computer Vision* (1998), pp. 839–846. 6

[USS*17] UHRIG J., SCHNEIDER N., SCHNEIDER L., FRANKE U., BROX T., GEIGER A.: Sparsity invariant CNNs. In *International Conference on 3D Vision* (2017), pp. 11–20. 3

[VKB*18] VALENTIN J., KOWDLE A., BARRON J. T., WADHWA N., DZITSIUK M., SCHOENBERG M. J., VERMA V., CSASZAR A., TURNER E. L., DRYANOVSKI I., AFONSO J., PASCOAL J., TSOTSOS K. N. J., LEUNG M. A., SCHMIDT M., GULERYUZ O. G., KHAMIS S., TANKOVICH V., FANELLO S., IZADI S., RHEMANN C.: Depth from motion for smartphone AR. *ACM Trans. Graph. 37*, 6 (2018). 3

[XZC17] XUE H., ZHANG S., CAI D.: Depth image inpainting: Improving low rank matrix completion with low gradient regularization. *IEEE Trans. Image Process. 26*, 9 (2017), 4311–4320. 3

[XZS*19] XU Y., ZHU X., SHI J., ZHANG G., BAO H., LI H.: Depth completion from sparse LiDAR data with depth-normal constraints. In *International Conference on Computer Vision* (2019), pp. 2811–2820. 3

[ZF18] ZHANG Y., FUNKHOUSER T.: Deep depth completion of a single RGB-D image. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2018), pp. 175–185. 1, 3