Multimed Tools Appl (2017) 76:4357–4380
DOI 10.1007/s11042-016-3523-y

Depth completion for kinect v2 sensor

Wanbin Song[1] · **Anh Vu Le**[1] · **Seokmin Yun**[1] · **Seung-Won Jung**[2] · **Chee Sun Won**[1]

Received: 13 October 2015 / Revised: 9 March 2016 / Accepted: 7 April 2016 /
Published online: 13 April 2016
© Springer Science+Business Media New York 2016

Abstract Kinect v2 adopts a time-of-flight (ToF) depth sensing mechanism, which causes different type of depth artifacts comparing to the original Kinect v1. The goal of this paper is to propose a depth completion method, which is designed especially for the Kinect v2 depth artifacts. Observing the specific types of depth errors in the Kinect v2 such as thin hole-lines along the object boundaries and the new type of holes in the image corners, in this paper, we exploit the position information of the color edges extracted from the Kinect v2 sensor to guide the accurate hole-filling around the object boundaries. Since our approach requires a precise registration between color and depth images, we also introduce the transformation matrix which yields point-to-point correspondence with a pixel-accuracy. Experimental results demonstrate the effectiveness of the proposed depth image completion algorithm for the Kinect v2 in terms of completion accuracy and execution time.

Keywords Kinect v2 · Hole-filling · Depth completion · Depth and color fusion

✉ Chee Sun Won
cswon@dongguk.edu

Wanbin Song
wbsong@dongguk.edu

Anh Vu Le
levuanh.hut@gmail.com

Seokmin Yun
smyun@dongguk.edu

Seung-Won Jung
swjung83@dongguk.edu

[1] Department of Electronics and Electrical Engineering, Dongguk University-Seoul, 26 Pildongro1-gil, Jung-gu, Seoul 100-715, South Korea

[2] Department of Multimedia Engineering, Dongguk University-Seoul, 26 Pildongro1-gil, Jung-gu, Seoul 100-715, South Korea

1 Introduction

The updated version of Kinect, Kinect v2, has been released from Microsoft. The comparative specifications between Kinect v1 and Kinect v2 are summarized in Table 1 [6, 10]. It can be seen that spatial resolution and field of view (FoV) of both color and depth images have been significantly improved and the usability of the Kinect has been enhanced owing to the extension of working range and the increase of the number of trackable skeletons. These advances in the Kinect v2 are enabled by changing the depth sensing mechanism from the structured light coding to the time of flight (ToF) technique.

However, the resolution improvements alone may not be sufficient to open new applications with the Kinect sensor. To fully exploit the captured images with improved resolutions we need a technique to fuse the color and depth images with point-by-point correspondences. Unfortunately, the registration accuracy of the Kinect v2 by the vendor provided function is not sufficient for delicate depth recognition. The goal of this paper is to provide a complete depth image by the fusion and the registration of the color and depth information from Kinect v2.

The structured light coding method of the Kinect v1 causes large holes near the object boundaries and serious interference errors when multiple Kinects are used. In comparison with the Kinect v1, different depth artifacts are observed in the Kinect v2 due to the ToF approach of depth measurement [2]. For example, as shown in Fig. 1, random blobs and thick holes near the object boundaries in the Kinect v1 are reduced in the Kinect v2. Specifically, the large holes due to occlusions in Fig. 1a are reduced and there are thin hole-lines near the object boundaries as in Fig. 1b. On the other hand, as shown in the corners of Fig. 1b, a new type of holes appears in the Kinect v2. As the FOV increases in the Kinect v2, the expanded coverage of the square depth image also includes outer pixels whose intensity values of returned infrared light are weak. Such pixels yield unreliable depth values and are simply deemed as hole pixels in practical ToF depth sensors. Note that these holes in the image corners of Fig. 1b are unseen regions in the Kinect v1 of Fig. 1a.

Acknowledging the existence of the thin hole-lines along the object boundaries, our approach to fill the holes in the depth image of the Kinect v2 is to exploit the edge information in the color image. Edges in color images carry key features in the image and have been used for various applications including indexing and retrieval [20]. Also, the edges constitute the object boundary and we use the color edge information for the hole-filling of depth images. Our approach of exploiting the actual edge location and direction information to fill the holes in the depth image is based on Le et al. [12], where the joint bilateral filters were employed to match the object boundaries between the color image and depth map. This can be viewed as

Table 1 Technical specifications for Kinect v1 and Kinect v2

	Kinect v1	Kinect v2
Depth measurement	Structured light coding	ToF (Time of Flight)
Resolution (Color)	640 × 480 (pixel)	1920 × 1080 (pixel)
Resolution (Depth)	320 × 240 (pixel)	512 × 424 (pixel)
FOV (Depth)	57 × 43°	70 × 60°
Maximum skeletal tracking	2	6
Working range	~4.5 m	~4.5 m

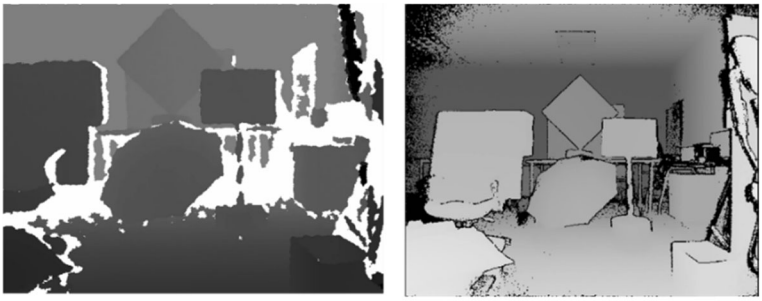

Fig. 1 Depth image comparison: **a** Depth image of Kinect v1 (*holes in white*), **b** Depth image of Kinect v2 (*holes in black*)

the camera fusion of color and depth sensors [7]. However, the method of Le et al. [12] was tuned to the depth images of the Kinect v1 and it may not be suitable for the depth images obtained from the Kinect v2. Note that, due to different methods of depth measurements, the depth images of Kinect v1 and Kinect v2 have different types of errors and holes. As shown in Fig. 1 the depth errors caused by the difference of the two sensing mechanisms look quite different. This demands a different approach to complete the depth errors for Kinect v2, which is also a motivation of our work and the fundamental difference between our work for Kinect v2 and [12] for the Kinect v1. Specifically, in [12], the depth image of the Kinect v1 is classified into four regions before the hole-filling by the combinations of edge/non-edge and hole/non-hole. However, considering the specific depth errors for the Kinect v2, we consider only two types of holes in this paper, namely edge holes (i.e., holes in the object boundary) and non-edge holes (i.e., holes in the interior pixels). Also, we need to change the window size of the filters to fit the depth artifacts of the Kinect v2. Finally, due to the thin holes along the object boundaries a precise registration between the color and the depth cameras and a point-by-point alignment method with a geometric transformation matrix is also needed in our approach.

In Kinect v1, a pixel with a negative-value is considered as a hole with no returned signal (i.e., non-available depth measurement) and a pixel with a positive value is classified as a non-hole region. The non-hole regions in the Kinect v1 are usually noisy. Also, as shown in Fig. 1, the holes along the object boundaries are thick and wide. So, in Le et al. [12], the depth image in the Kinect v1 is classified into four regions before the hole-filling by the combinations of edge/non-edge and hole/non-hole. Then, an appropriate filter is applied for each classified depth region for the hole-filling. However, in the Kinect v2, the measured depths are usually less noisy than the Kinect v1 and there may be no need to apply noise removal filters for the non-hole regions. Therefore, in this paper, we consider only two types of holes, namely edge holes (i.e., holes in the object boundary) and non-edge holes (i.e., holes in the interior regions) instead of the four classifications in Le et al. [12]. We apply an appropriate filter for each of the two classes. More specifically, if the detected hole belongs to the non-edge region, then the partial directional bilateral filter (PDJBF) is adopted. Otherwise, if the hole is in the edge region, then the directional joint bilateral filter (DJBF) is used. Also, as shown in Fig. 1, the holes of the Kinect v1 along the object boundaries take larger areas than those in the Kinect v2. This will be considered for the determination of the window size of the filters. As a prerequisite for our color-edge based approach, the color and

the depth image captured from the Kinect v2 should be precisely registered and we provide a geometric transformation matrix for the registration.

Similar to our approach, inspired by the Joint Bilateral Filter (JBF), the filters in [4, 12, 22] are proposed to fill holes in raw depth maps, where they have the same support regions for the filter kernels. However, there is a major difference among them in assigning the filter coefficients for the support regions. In [12], the coefficients in the support region are determined by the weights of the directional Gaussian filter. That is, the higher values are assigned to the coefficients which stay along the direction of directional Gaussian kernel. On the other hand, in [4], the coefficients are assigned according to the reliability of locations in the support region of filter kernel. Note that the reliable map is based on the properties of Kinect v1 depth map. Besides, in [22], color-guided auto-regressive model and optimal parameters of JBF based on object shapes in color images are used to get the optimal coefficients. We note that none of these three filters are optimized for the depth map captured by the Kinect v2, whereas our goal in this paper is to tune the directional Gaussian filter for the specific depth map of the Kinect v2.

This paper is organized as follows. Preprocessing of our work is described in Section 2 and hole-filling algorithms are presented in Section 3. We provide experimental results in Section 4. Then, this paper concludes in Section 5.

2 Color and depth registration

Before we apply the bilateral filters for the hole-filling, color and depth images from Kinect v2 should be registered. The registration can be done by calibrating the color and depth sensors with a transformation matrix. One can use the IR image provided by the Kinect v2 instead of the depth image for the registration. However, the intensity of the IR image cannot be directly used for many reasons including the problems of saturation and intensity value falling off phenomenon [9]. That is, if an object is located very close to the depth sensor or has a high IR reflectance, its intensity value tends to be saturated. Meanwhile, if an object is positioned far from the depth sensor or has a low IR reflectance, its intensity value falls off considerably. In both over- and under-exposed areas, the intensity of the IR image has low confidence, causing hardship for the IR intensity image-based calibration.

In Kinect v2, the color and depth images have different resolutions. Also, they have different FOVs. Specifically, the size of the color image is bigger than the depth image in the Kinect v2, which requires a cropping of the color image before the registration. Microsoft provides a new software development kit (SDK) for the Kinect v2 to enable developers to promote applications [10]. They provide many functions to handle Kinect. Among them there is a registration function named *CoordinateMapper*, mapping the original color image to fit the size of the depth image (see Fig. 2c). However, as shown in the superimposed depth and cropped color image in Fig. 2d, the registration may not be good enough for the hole-filling and we can see some misalignments in the object boundaries of the depth and the color images. Note that, since we heavily rely on the aligned edges on the object boundaries between the color and the depth images for the hole-filling, we need a more accurate registration matrix between the two images before the hole-filling. This motivates us to find a more accurate transformation that converts color image coordinate to depth image coordinate.

The projective matrix that converts the color image coordinate (x,y) into the depth image coordinate (X,Y) is given as follows [17]

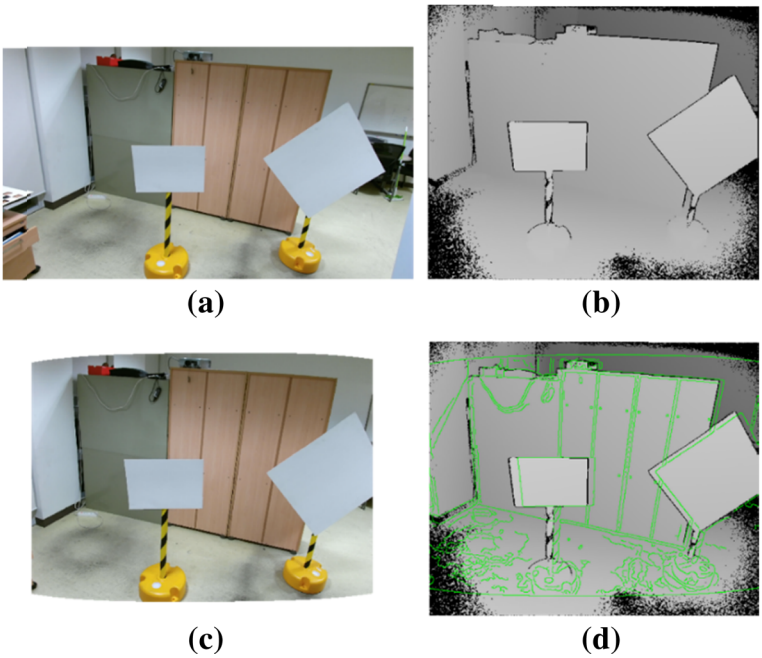

Fig. 2 Color and depth image mapping with the vendor(SDK)-provided function: **a** Original color image (1920 × 1080), **b** Original depth image (512 × 424), **c** Color image mapped to the depth image (512 × 424), **d** Depth image with the superimposed edge map of color image (green lines represent the edge of color image)

$$\begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = \mathrm{T} \begin{bmatrix} \mathrm{x} \\ \mathrm{y} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathrm{a}_1 & \mathrm{a}_2 & \mathrm{a}_3 \\ \mathrm{a}_4 & \mathrm{a}_5 & \mathrm{a}_6 \\ \mathrm{a}_7 & \mathrm{a}_8 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} . \tag{1}$$

In Eq. (1), we have eight unknown parameters and it can be rewritten as the following two equations

$$\mathrm{X} = \mathrm{a}_1\mathrm{x} + \mathrm{a}_2\mathrm{y} + \mathrm{a}_3 - \mathrm{a}_7\mathrm{xX} - \mathrm{a}_8\mathrm{yX} \tag{2}$$

$$\mathrm{Y} = \mathrm{a}_4\mathrm{x} + \mathrm{a}_5\mathrm{y} + \mathrm{a}_6 - \mathrm{a}_7\mathrm{xY} - \mathrm{a}_8\mathrm{yY} . \tag{3}$$

At least four corresponding pairs are needed to solve the above two equations with the eight unknown coefficients. And, if we use more than four pairs of points, the parameters can be found using a least-square method such as the direct linear transform [1]. Quadrilateral calibration boards are employed to find the corresponding pairs between the color and depth images as in Fig. 3a and b. Since the registration can be done off-line, the correspondences can be manually selected. Then, by substituting the coordinates of the corresponding pairs to Eqs. (2) and (3), we can determine the eight unknown coefficients for the transformation matrix. Using our transformation matrix, the color image can be aligned with the depth image. Comparing Figs. 2d with 3c, our transform function yields accurate registration of the color and depth images.

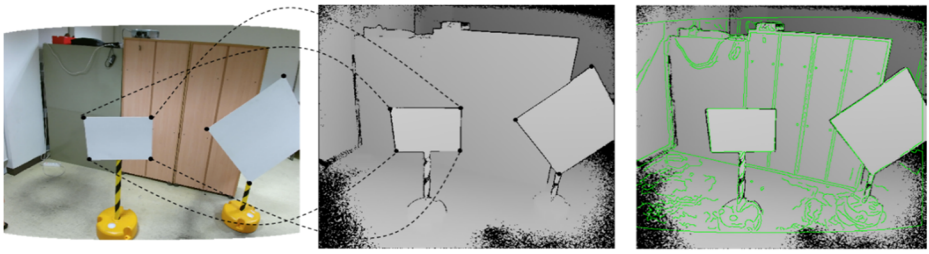

Fig. 3 Registration by our projective transformation (Black dotted line represents correspondence pairs): **a** Color image (512 × 424) aligned by the SDK function, **b** Original depth image (512 × 424), **c** Edge map after registration (green line represents the edge of color image)

3 Edge-based hole-filling method

After the registration between the color and the depth images for the Kinect v2, we are now ready to use the edge information in the corresponding pixel locations of the registered color and depth images. That is, the basic approach of our hole-filling method is to apply designated filters for the holes on the edge and on the non-edge areas selectively. So, it is necessary to classify holes into edge hole or non-edge hole. Thanks to the registration between the color and depth images, we can employ a simple rule to classify the holes into edge or non-edge holes by just checking the co-existence of color and depth edges around hole-pixel. Then, after classifying the hole-pixels as edge and non-edge holes, the non-edge holes are filled by a partial directional joint bilateral filter (PDJBF). The principle of the PDJBF is to use the data only on the region of the object that the hole-pixel actually belongs to for the filling. After filling the non-edge hole-pixels, a directional joint bilateral filter (DJBF) is applied to fill the holes on the edge-pixels. Figure 4 shows the entire block diagram for the proposed method. In this section we will modify the adaptive directional filter proposed by Le et al. [12] for the Kinect v2 depth images.

3.1 Preprocessing for hole-filling

I and D in Fig. 4 denote the registered color and original depth image, respectively. The edges on the object boundaries $\bar{E}_I$ are chosen by the color edges $\bar{E}_I$ and the depth edges E_D.

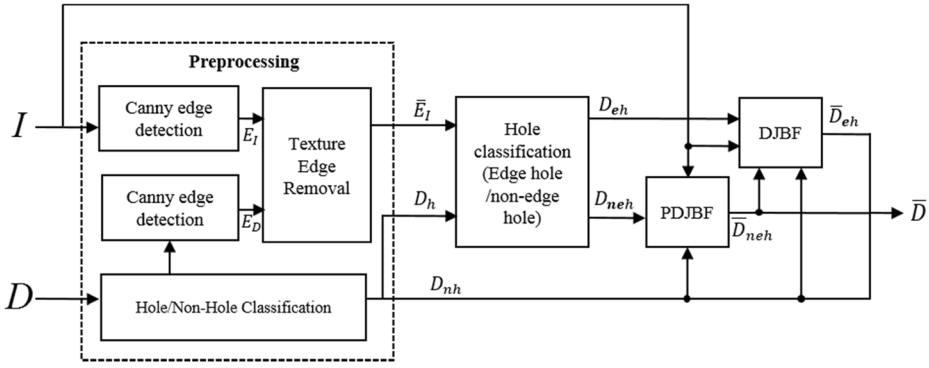

Fig. 4 Block diagram for the proposed hole filling for depth images

Specifically, we have adopted the Canny edge detector [3] to find the edge maps for the color and depth images [5]. Then, we obtained the final edge map by using an AND-operator method, removing the color edge pixels from if there are no depth edge pixels inside their 7x7 neighborhood. We also eliminated the color edge pixels if the number of connected color edge pixels is lower than the threshold T_n. By removing false color edge pixels in the above manner, we obtain the edge map $\bar{E}_I$ which has edge pixels mostly along the object boundaries (see Fig. 5).

3.2 Hole-filling

In Kinect v2, since the hole pixels in depth image have zero value, the input depth image D can be easily classified into non-hole region D_{nh} and hole region D_h. Also, with the edge information $\bar{E}_I$, we can classify the holes into edge holes and non-edge holes. The non-edge holes are filled by the PDJBF and the holes in the edge region are filled by the DJBF [12] (see Fig. 4).

To increase the availability of the filled data for the hole-filling of un-filled holes at the edge regions, the non-edge holes are filled first. Then, the filled non-edge holes will be used to fill the edge holes. The filling method of the non-edge holes is based on the PDJBF. The PDJBF is basically a weighted sum of a set of partial depth data within a window of $(2w+1)\times(2w+1)$ centered at a non-edge hole $p=(p_x,p_y)$. Here, the size of the window is adaptively determined for better hole-filling performance. It is determined by comparing the minimum distance d from non-edge hole p to nearest edge with pre-fixed maximum window size w_{max}, as follows:

$$w = \begin{cases} w_{max}, & if\ w_{max} \le d \\ d, & if\ w_{max} > d \end{cases} \tag{4}$$

There are two weighting functions, namely the directional Gaussian filter (DGF) function f_{ds}^{d} and the weighting function from the color difference f_r. The DGF is a rotated Gaussian

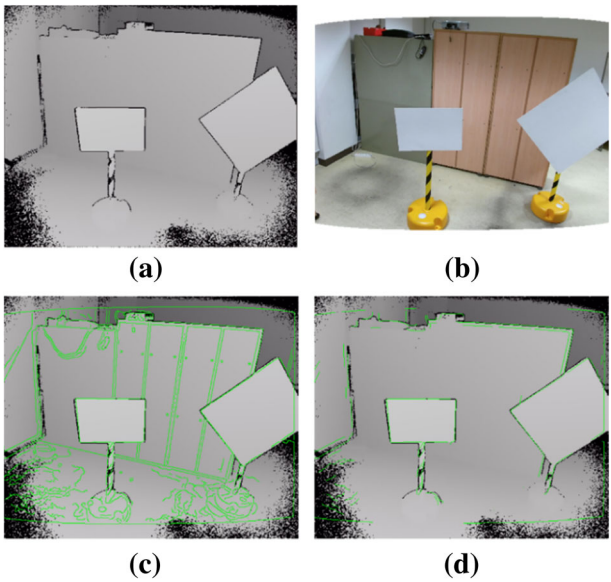

Fig. 5 Original depth, registered color image by and overlay of depth and edge maps (Green lines represent the color edge map): **a** Original depth, **b** Registered color, **c** Overlaid image by D and E_I, **d** Overlaid image by D and $\bar{E}_I$

function by the edge direction θ. Then, the filled depth non-edge hole $\overline{D}^p_{neh}$ at a hole-pixel p is obtained as follows

$$\overline{D}^p_{neh} = \sum_{q\in\Omega^p} D^q_{nh} f^d_{ds}\left(q_x - p_x, q_y - p_y\right) f^c_r(I^p - I^q) \tag{5}$$

where D^q_{nh} is the non-hole depth value at q and Ω^p is a subset of pixels in the window $(2w+1)\times(2w+1)$. Because the pixels in the subset Ω^p are partially used according to the edge direction of the nearest edge pixel, we need to acquire the edge direction θ (see red dotted line in Fig. 6). Specifically, for each hole-pixel to be filled, the nearest edge pixel is found and its edge direction θ is calculated by the spatial gradients g_x and g_y of the nearest edge-pixel in x and y directions, respectively, (i.e., $\theta = \tan^{-1}(g_x/g_y)$). Then, the subset Ω^p as one of the four sub-regions in the window, namely left, right, top, or bottom part of the window can be determined according to the edge direction θ. Here, the selected sub-region of the window is supposed to be the object area where the hole-pixel p actually belongs to. We refer the readers for more details about Ω^p to Le et al. [12].

The weighting function f^c_r measures the color difference between neighboring pixels as

$$f^c_r(I^p - I^q) = e^{-\frac{1}{2}\left(\frac{I^p - I^q}{\sigma_r}\right)^2} \tag{6}$$

and the weighting function f^d_{ds} represents the rotated spatial Gaussian filter kernel as follows

$$\begin{aligned} f^d_{ds}\left(q_x - p_x, q_y - p_y\right) &= e^{-\frac{1}{2}\left(\frac{x^2_\theta}{\sigma^2_x} + \frac{y^2_\theta}{\sigma^2_y}\right)} \\ x_\theta &= (q_x - p_x)\cos\theta - \left(q_y - p_y\right)\sin\theta \\ y_\theta &= (q_x - p_x)\sin\theta + \left(q_y - p_y\right)\cos\theta \end{aligned} \tag{7}$$

In Eq. (7) the DGF is rotated according to the edge direction θ, and its horizontal and vertical standard deviations σ_x and σ_y are controlled separately.

After filling the non-edge holes, the DJBF is applied to fill the remaining edge holes. The DJBF is also a weighted sum of the depth data within the window. The only difference

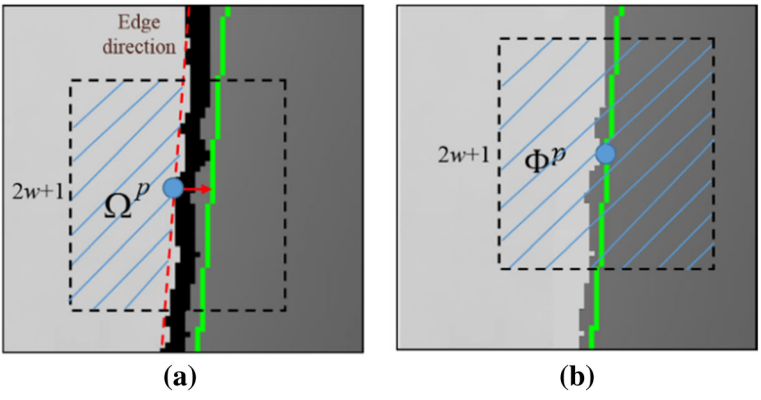

Fig. 6 Difference between the PDJBF and the DJBF (Green lines represents nearest edges of the hole (blue dot): **a** Support region of PDJBF (Since the nearest edge pixel is at the right-hand side of p, the support region (*the blue lines*) is on the left-half side of the window), **b** Support region of DJBF

between the PDJBF and the DJBF is the support region of the pixel in the window, and Fig. 6 describes it in detail.

As you can see in Fig. 6a, in the PDJBF, we use the partial support region of the pixel (blue dashed area) for hole-filling. On the other hands, the entire pixels in the window Φ^p of $(2w+1)\times(2w+1)$ are used in the DJBF as in Fig. 6b. As we filled non-edge holes first, the majority of pixels in Φ^p are already filled except for unknown values. It is fixed with pre-fixed value w_{max}, centered at the edge hole p and the interpolated data $\overline{D}^p_{eh}$ at an edge hole p can be obtained as follows

$$\overline{D}^p_{eh} = \sum_{q\in\Phi^p} \left(D^q_{nh} \cup \overline{D}^q_{neh}\right) f^d_{ds}\left(q_x - p_x, q_y - p_y\right) f^c_r\left(I^p - I^q\right) \tag{8}$$

where $D^q_{nh} \cup \overline{D}^q_{neh}$ selects either D^q_{nh} or $\overline{D}^q_{neh}$ depending on the type of the available depth data at q. Then, D_{nh}, $\overline{D}_{neh}$, and $\overline{D}_{eh}$ constitute the resulting depth image, we have the final hole-filled depth image $\overline{D}$.

4 Experimental results

The overall block diagram of our hole-filling method for the Kinect v2 is shown in Fig. 7. First, the color and depth images generated by the Kinect v2 are initially aligned using the *CoordinateMapper* function in the Microsoft SDK. Then, the depth and color images are more finely registered by our transformation matrix. Finally, the holes in the depth image are filled by exploiting the edge information in the registered color image. The performance of our hole-filling method is demonstrated in this section. All test images in the experiments are captured by our Kinect v2 sensor.

4.1 Color and depth image registration

Executing the registration between the color and the depth image for the Kinect v2 as explained in Section 2, we obtain the projective transformation matrix T in Eq. (1) as follows

$$\mathrm{T} = \begin{bmatrix} 0.9964 & -0.0033 & -8.0255 \\ -0.0197 & 0.9879 & 1.5348 \\ -0.00001 & -0.00002 & 1 \end{bmatrix}. \tag{9}$$

This is our transformation matrix for the registration in Fig. 7. Figure 8 shows a visual comparison between the *CoordinateMapper* function in the SDK and our transformation matrix. As one can see in Fig. 8c and d our registration yields much more accurate results than the *CoordinateMapper* transformation function. The transformation matrix in Eq. (9)

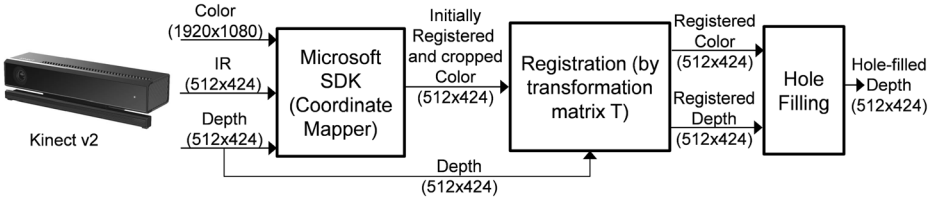

Fig. 7 Overview of the proposed hole-filling method

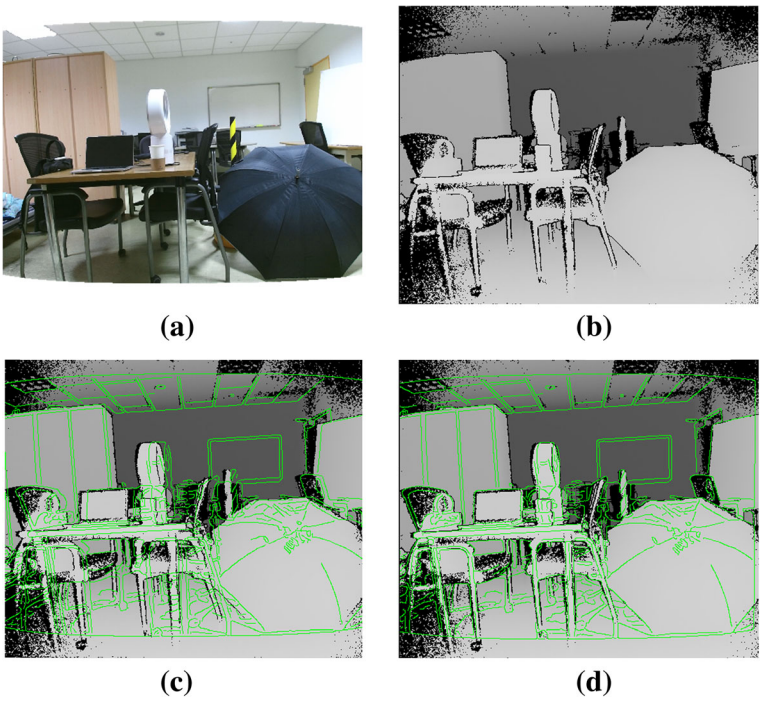

Fig. 8 Color and depth registration result (Green lines represents the edge map of color image): **a** Cropped color, **b** Original depth, **c** Registration by the *CoordinateMapper* function, **d** Registration by the proposed transformation matrix

corresponds to a projective transformation matrix. We observed that the registrations by the projective and affine transformations sometimes yield noticeable differences (see Fig. 9) and the projective transformation yields better registration results.

For the quantitative evaluation of the registration accuracy, as shown in Fig. 10, we measured the Euclidean distance of 20 corresponding points for 5 pairs of calibrated RGB and depth images. The comparison in terms of the average pixel-error distance between the SDK-provided registration and the proposed one is shown in Table 2, where the pixel error of

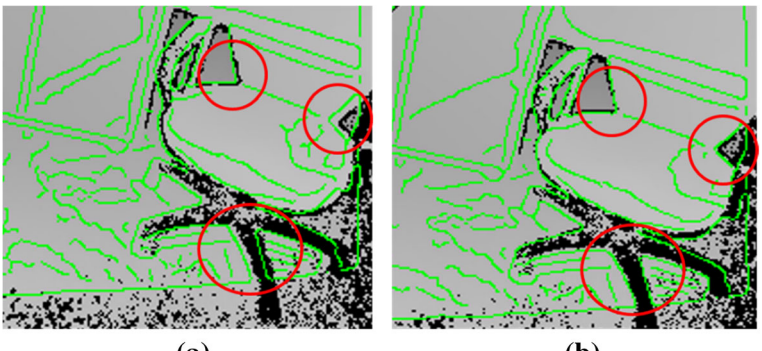

Fig. 9 Registration by: **a** Affine transformation, **b** Proposed projective transformation

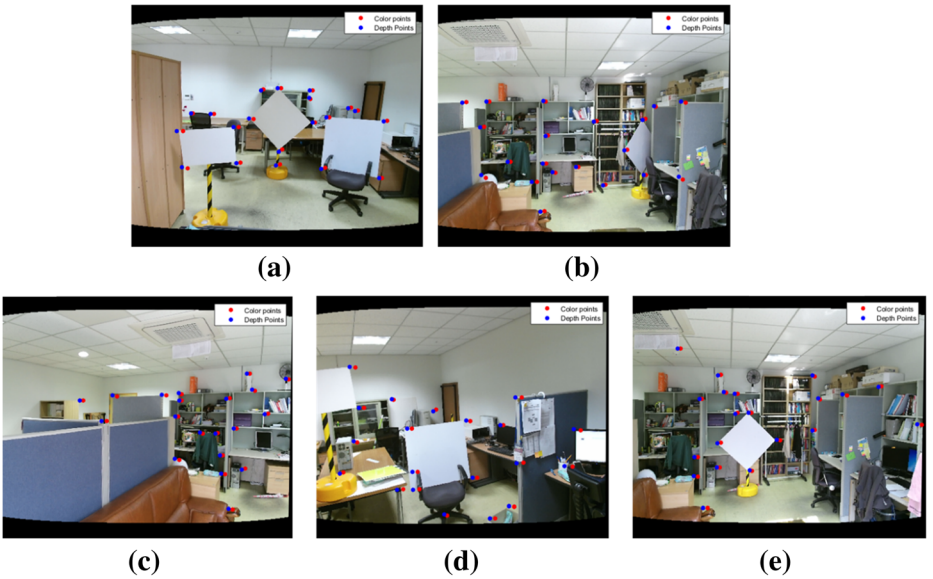

Fig. 10 Corresponding points for: **a** Image 1, **b** Image 2, **c** Image 3, **d** Image 4, **e** Image 5

the proposed method is less than 2 pixels and is much smaller than that of the SDK-provided method.

4.2 Hole-filling results

Since the weighting function in Eq. (6) is originated from Tomasi et al. [19], we used the same parameter value $\sigma_r = 0.1$ for our experiments. Also, parameter values of $\sigma_x = 3$, and $\sigma_y = 1$ in Eq. (7) recommended by Horng et al. [8] were used. As mentioned in Section 1, the hole-areas

Table 2 Comparison of registration accuracy (mean/standard deviation of pixel errors by Euclidean distance)

	Method	X-direction		Y-direction		Point-by-Point	
		Mean	Std	Mean	Std	Mean	Std
Image 1	SDK Function	5.9660	2.1842	1.1632	1.0178	6.1791	2.1204
	Proposed	1.3750	1.2234	0.6751	0.7122	1.6942	1.2043
Image 2	SDK Function	5.5877	2.0457	1.0502	0.9189	5.7951	1.9886
	Proposed	1.7715	0.6486	0.6725	0.5884	1.9815	0.6800
Image 3	SDK Function	5.8311	2.1348	0.9482	0.8297	5.9544	2.0433
	Proposed	1.2370	0.4529	0.8622	0.7544	1.6888	0.5795
Image 4	SDK Function	5.2346	1.9164	1.1819	1.0342	5.3817	1.8468
	Proposed	2.3509	0.8607	0.9376	0.8204	2.5373	0.8707
Image 5	SDK Function	5.1863	1.8987	0.8673	0.7589	5.2962	1.8174
	Proposed	1.5800	0.5785	0.7664	0.6706	1.8793	0.6449
Average	SDK Function	5.5611	2.0360	1.0422	0.9119	5.7213	1.9633
	Proposed	1.6628	0.7528	0.78276	0.7092	1.95622	0.7959

of the Kinect v2 along the object boundaries are usually thinner and narrower than those in the Kinect v1. Considering this, we do not need a large enough window for the support of the filtering, so we fixed the maximum window size as 5x5 (i.e., $w_{max} = 5$), which was 11x11 in Le et al. [12]. We compared the hole-filling method in Le et al. [12] and Yang et al. [22] with our method explained in Section 3. Figures 11, 12, 13 and 14 show the results. For all our experiments, the proposed hole-filling method and the method of Yang et al. [22] yield more accurate results than the hole-filling method for the Kinect v1 [12]. Both the method of Yang et al. [22] and the proposed hole-filling method recover accurate depths particularly at the object boundaries. However, as shown in Figs. 13 and 14, the method of Yang et al. [22] often yields errors at frontal areas. Also, the proposed method is computationally much more efficient than the method of Yang et al. [22]. Table 3 shows that the proposed method is almost 120 times faster than the method of Yang et al. [22] and about 6 times faster than the hole-filing method of Le et al. [12].

Experiments were conducted with Kinect v2's RGB and depth images. There are hole-filling methods for Kinect v1 [11, 13, 14, 21], among them, the proposed method was compared with the adaptive trimmed median filtering method and the colorization method [13]. The adaptive trimmed median filter is a modified median filter, where an adaptive mask is used to select only valid pixels to be used in filtering. The colorization method relies exclusively on neighboring pixels in the depth map to fill the hole. The results of the adaptive trimmed median filter are shown in Figs. 15c, 16c, 17c and 18c, where big holes are still remained. The colorization method in Figs. 15d, 16d, 17d and 18d and the proposed method in Figs. 15e, 16e, 17e and 18e yield comparable results.

Since there are no ground-truth depth images for the Kinect v2 available, it is difficult to evaluate the hole-filling methods quantitatively. However, using the widely-used Middlebury ground-truth depth images [18] we can synthesize Kinect v2-like hole pixels. From the ToF depth sensing mechanism and empirical distributions of Kinect v2 depth images' hole pixels,

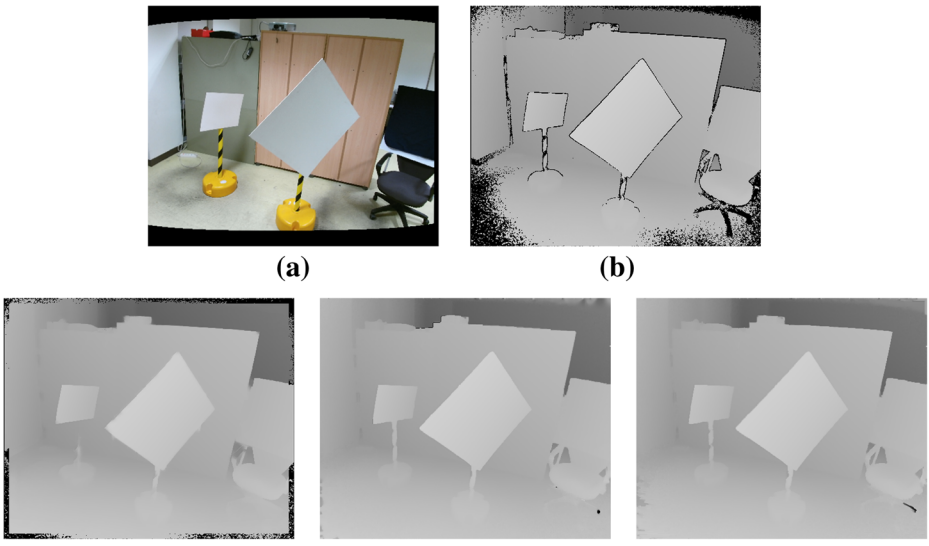

Fig. 11 Comparison of hole-filling results: **a** Cropped color, **b** Original depth, **c** Hole-filling method of Le et al. [12], **d** Hole-filling method of Yang et al. [22], **e** Proposed method

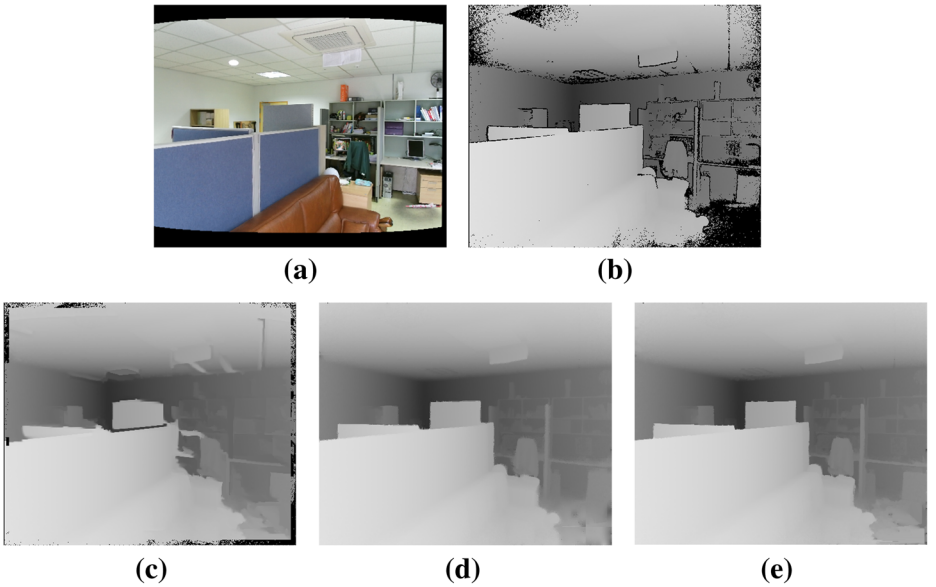

Fig. 12 Comparison of hole-filling results: **a** Cropped color, **b** Original depth, **c** Hole-filling method of Le et al. [12], **d** Hole-filling method of Yang et al. [22], **e** Proposed method

our observation is that hole pixels frequently occur when the received IR signal's intensity is weak. This happens when a depth pixel corresponds to an object with a low IR reflectivity or is located around object boundaries. To simulate this, from the Middlebury color image [18], the intensity channel **I** is deemed as the IR channel due to the unavailability of IR signals in the Middlebury database. Also, detecting the object boundaries by applying the Canny edge

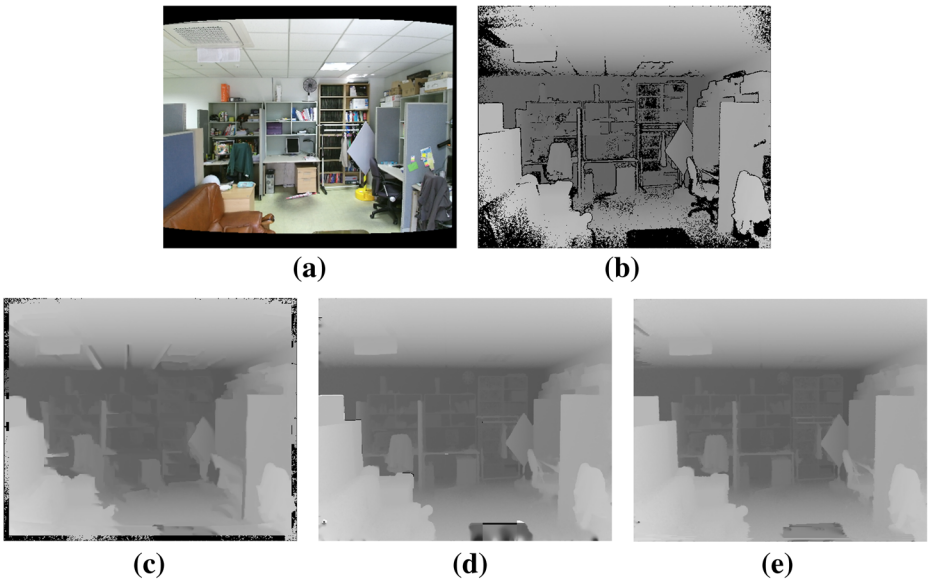

Fig. 13 Comparison of hole-filling results: **a** Cropped color, **b** Original depth, **c** Hole-filling method of Le et al. [12], **d** Hole-filling method of Yang et al. [22], **e** Proposed method

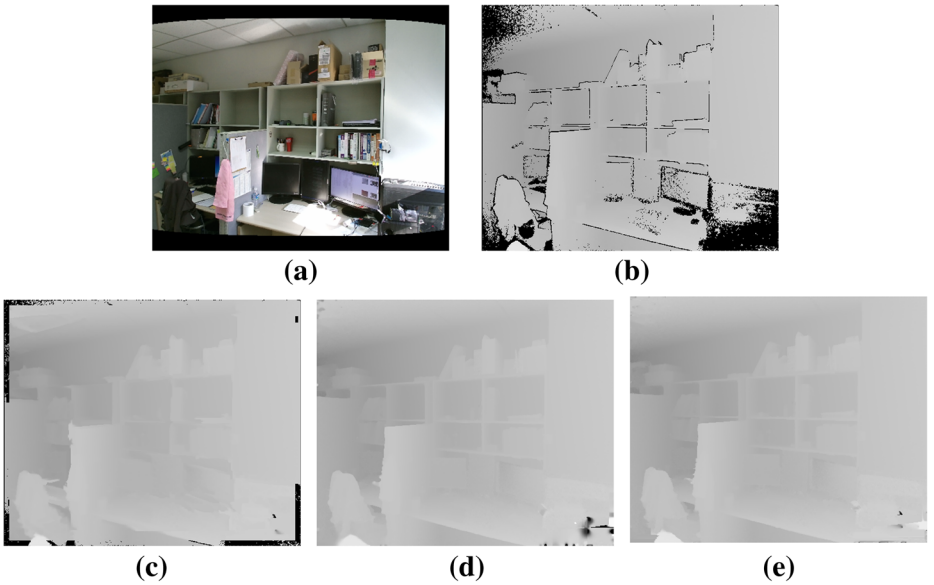

Fig. 14 Comparison of hole-filling results: **a** Cropped color, **b** Original depth, **c** Hole-filling method of Le et al. [12], **d** Hole-filling method of Yang et al. [22], **e** Proposed method

detector to the ground-truth depth image, we can construct the distance map **D**, where each pixel value represents the distance to the nearest object boundary pixel. Then, we recast a hole generation problem as a binary labeling problem (1: hole, 0: non-hole). That is, given **I** and **D**, the binary hole map **H*** is found by maximizing the following probability:

$$\mathbf{H}^* = \arg\max_{\mathrm{H}} p\left(\mathbf{H}\middle|\mathbf{I},\mathbf{D}\right) = \arg\max_{\mathrm{H}} p\left(\mathbf{I}\middle|\mathbf{H}\right)p\left(\mathbf{D}\middle|\mathbf{H}\right)p(\mathbf{H}). \tag{10}$$

Equivalently, we can find **H*** by

$$\mathbf{H}^* = \arg\min_{\mathrm{H}}\left\{-\ln\left(p\left(\mathbf{I}\middle|\mathbf{H}\right)\right)-\ln\left(p\left(\mathbf{D}\middle|\mathbf{H}\right)\right)-\ln(p(\mathbf{H}))\right\}. \tag{11}$$

The first and second conditional probabilities in Eq. (11) are modeled as exponential distributions as

$$\begin{aligned} p\left(i\middle|h\right) &= K_1\exp(-\alpha\cdot i) \\ p\left(d\middle|h\right) &= K_2\exp(-\beta\cdot d). \end{aligned} \tag{12}$$

Table 3 Computing time comparison of hole-filling methods (unit: second)

	Le et al. [12]	Yang et al. [22]	Proposed method
Image1	14.5882	296.2135	2.3171
Image2	13.7871	301.1478	2.6135
Image3	18.9081	272.3111	2.9322
Image4	15.7064	273.5581	2.3635
Average	15.7474	285.8076	2.5566

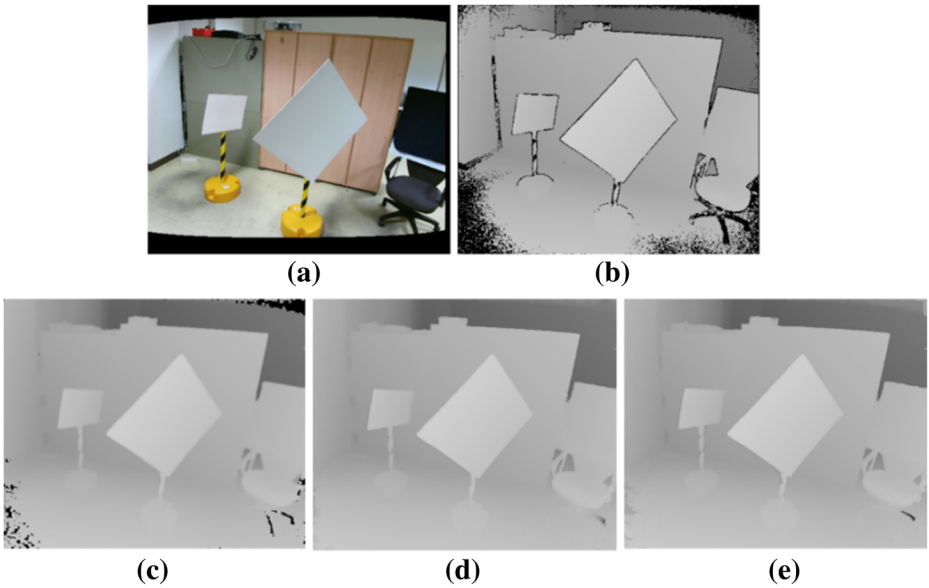

Fig. 15 Comparison of depth completion: **a** Cropped color, **b** Original depth, **c** Adaptive trimmed median filter, **d** Colorization method, **e** Proposed method

where i, d, and h represent an intensity value, distance to the nearest depth boundary pixel, and hole label for each pixel, respectively. α is determined to make the bottom p percent intensity value i_p satisfy $\exp(-\alpha i_p) = 0.5$. Similarly, β is set to have

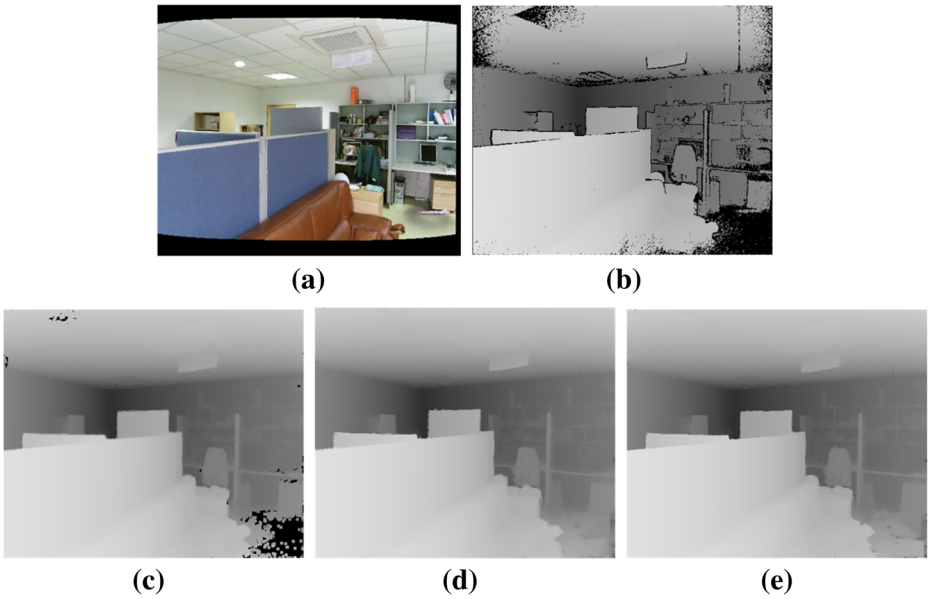

Fig. 16 Comparison of depth completion: **a** Cropped color, **b** Original depth, **c** Adaptive trimmed median filter, **d** Colorization method, **e** Proposed method

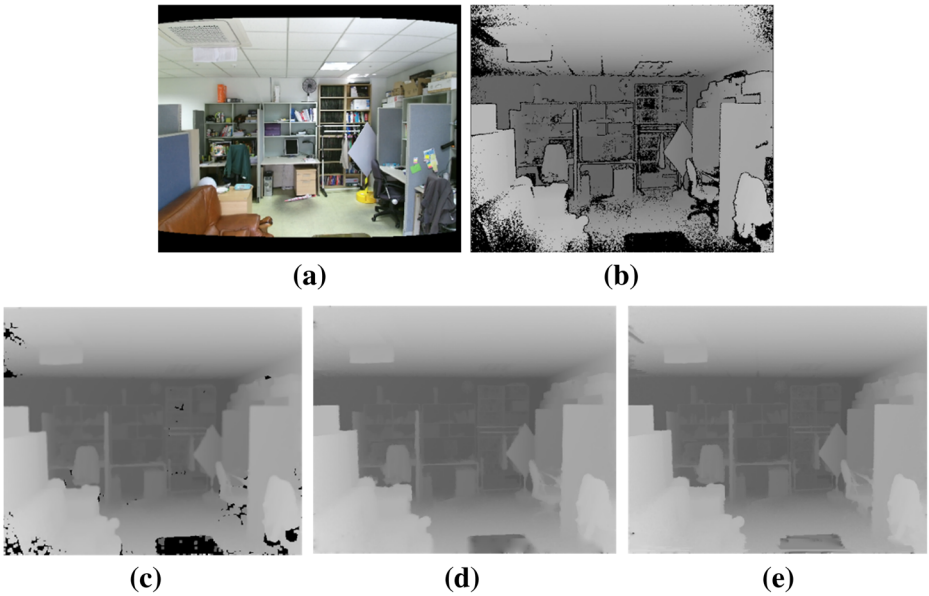

Fig. 17 Comparison of depth completion: **a** Cropped color, **b** Original depth, **c** Adaptive trimmed median filter, **d** Colorization method, **e** Proposed method

$\exp(-\beta d_p) = 0.5$, where d_p represents the bottom p percent value of $\mathbf{D}$. K_1 and K_2 are normalization constants. Eqs. (11) and (12) say that the probability of a pixel being a hole pixel exponentially decreases as the intensity value increases and the distance to

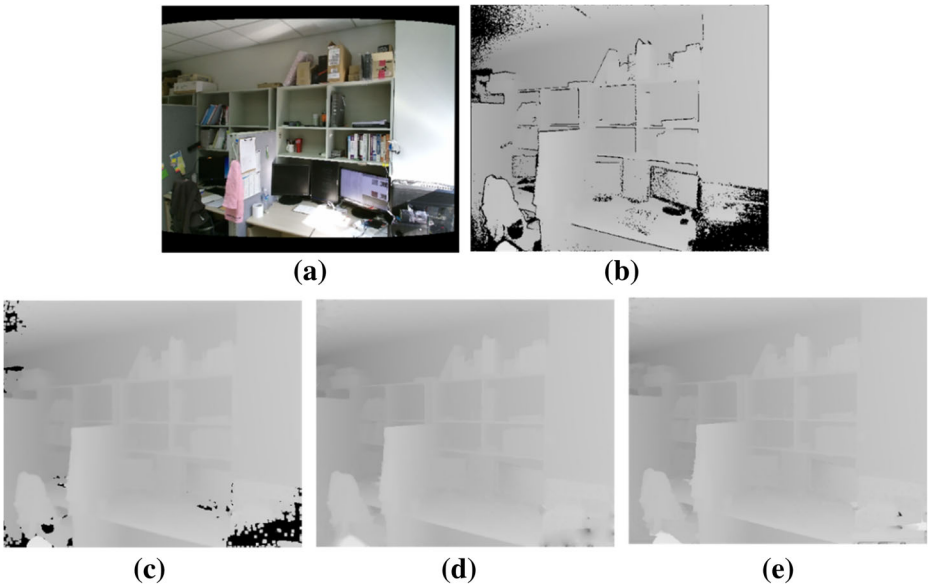

Fig. 18 Comparison of depth completion: **a** Cropped color, **b** Original depth, **c** Adaptive trimmed median filter, **d** Colorization method, **e** Proposed method

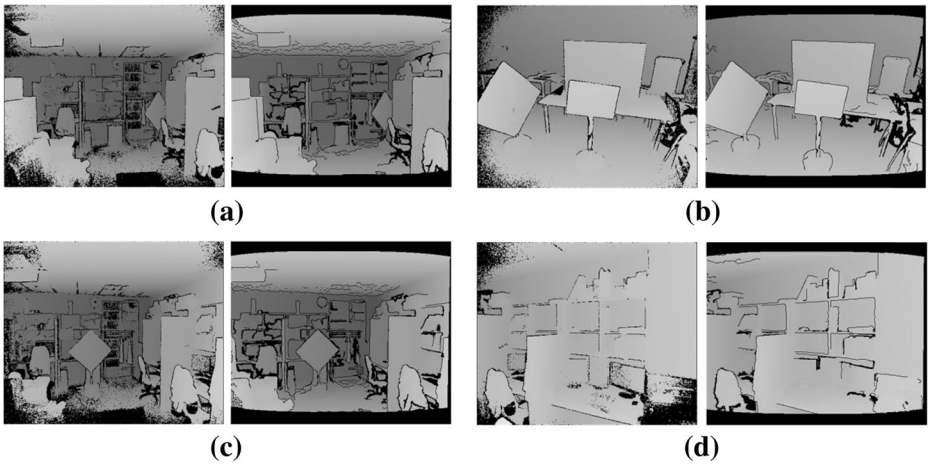

Fig. 19 Comparison between the original Kinect v2 depth images (before the depth-completion) (*left*) and the depth-completed image with the simulated-holes (*right*)

the depth boundary increases. Finally, a prior probability of **H** in Eq. (11) is modeled as

$$p(\mathbf{H}) = K_3 \sum_{(x,y)\in N_4} \delta_{h_x \neq h_y} \exp\left(-\frac{(d_x - d_y)^2}{2\sigma_d^2} - \frac{(i_x - i_y)^2}{2\sigma_i^2} \right). \tag{13}$$

where N_4 represents the four-nearest-neighborhood, $\delta_{h_x \neq h_y} = 1$ when the neighboring pixels are differently labeled and $\delta_{h_x \neq h_y} = 0$ otherwise. K_3 is the normalizing constant. d_x and i_x are the depth value and intensity value at the pixel x, respectively. According to Eq. (13), we enforce the similar labeling of neighboring pixels especially when their intensity and/or depth values are

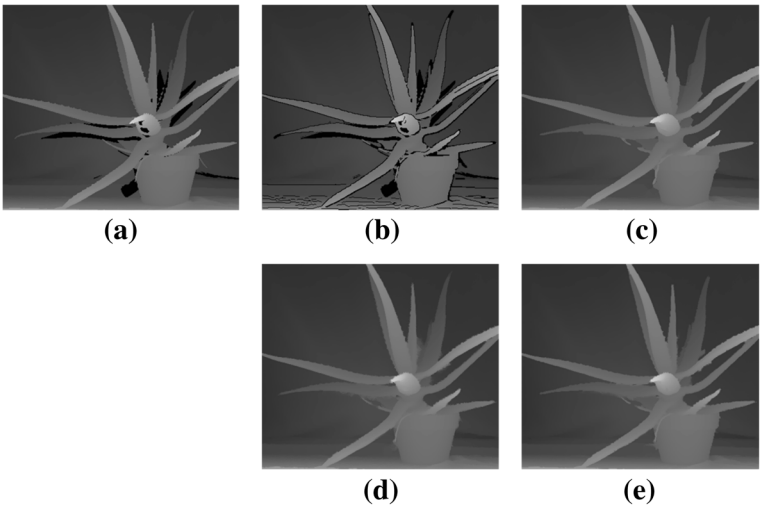

Fig. 20 Aloe image: **a** Ground truth, **b** Synthesized hole image, **c** Adaptive trimmed median filter, **d** Colorization method, **e** Our proposed method

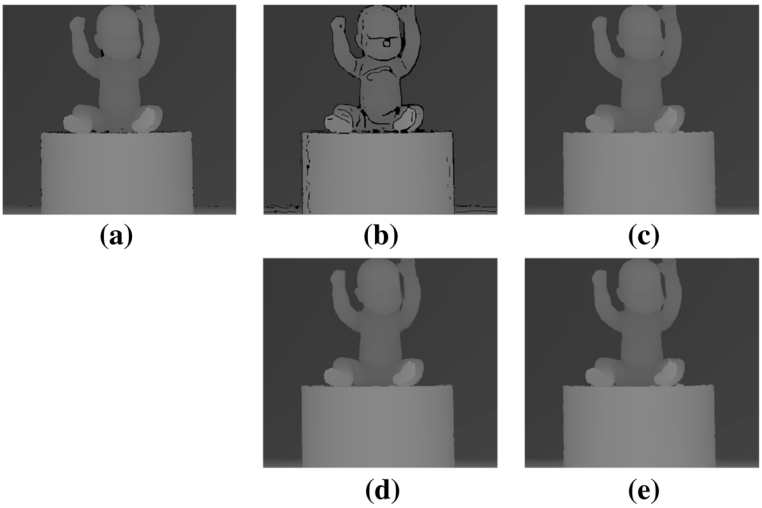

Fig. 21 Baby1 image: **a** Ground truth, **b** Synthesized hole image, **c** Adaptive trimmed median filter, **d** Colorization method, **e** Our proposed method

similar. The final label map is found using the Graph-cut optimization [16]. The parameters p, σ_d, and σ_i are fine tuned to make our method closely approximate real Kinect v2 depth holes and we set $p = 4$, $\sigma_d = 2$, and $\sigma_i = 0.5$. The effectiveness of the proposed hole simulation method can be verified by generating the holes with the depth-completed images and comparing the result with the original Kinect v2 depth image. We used the same parameter settings of $\sigma_d = 2$, and $\sigma_i = 0.5$ but varied p values from 17 to 22 to exclude the influence of black pixels in the color images caused by the FoV (Field of View) difference. Figure 19 shows that our scheme imitates natural Kinect-v2 holes quite well.

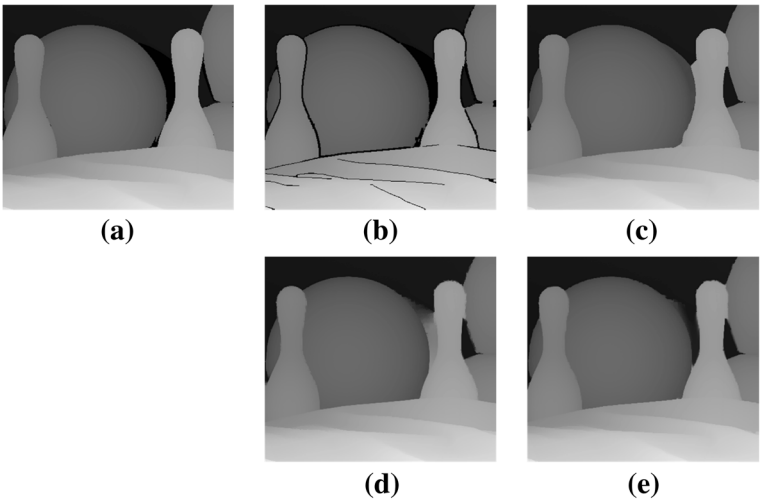

Fig. 22 Bowling1 image: **a** Ground truth, **b** Synthesized hole image, **c** Adaptive trimmed median filter, **d** Colorization method, **e** Our proposed method

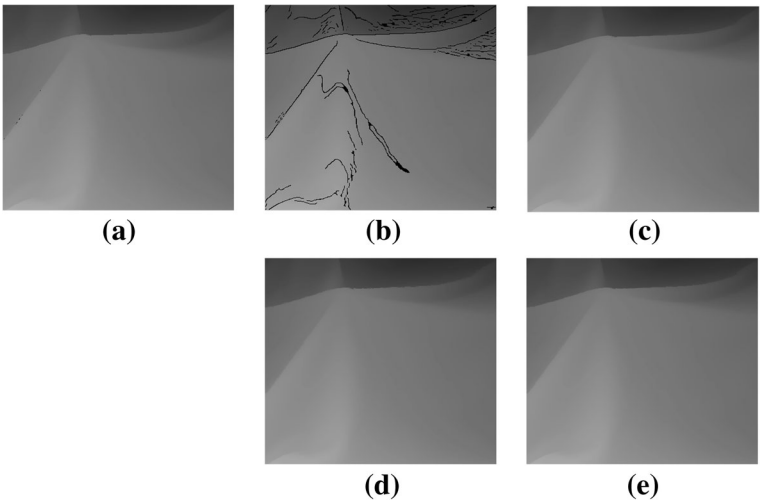

Fig. 23 Cloth1 image: **a** Ground truth, **b** Synthesized hole image, **c** Adaptive trimmed median filter, **d** Colorization method, **e** Our proposed method

The above hole-simulation algorithm is applied to the Middlebury ground-truth depth images [18] to generate Kinect v2-like hole images. These synthesized-hole images and their corresponding ground-truth images are used to compare the hole-filling performance (see Figs. 20, 21, 22 and 23). The PSNR results are shown in Table 4.

We also compare the proposed algorithm with the existing ones in terms of the computational complexity. As shown in Table 5, the proposed method runs faster than the adaptive trimmed median filter and colorization method. The comparative experiments are conducted by a computer with Intel Xeon CPU 3.60Ghz, 16GB RAM with NVIDIA Quadro 600 Graphic card and the implementation is done with MATLAB R2015b.

To demonstrate the effectiveness of our method we used the Features from Accelerated Segment Test (FAST) [15] algorithm to find feature points in the depth images. Figure 24 shows the results from FAST feature extraction and the top 200 strongest points are marked in colors. As in Fig. 24a–c, the detected features of the original Kinect v2 depth image are mostly on the meaningless hole edges. On the other hand, as shown in Fig. 24d–f, detected features on our depth completed images are marked on true corners and edges points.

Table 4 PSNR comparisons on the synthesized Kinect v2-like hole images

Image	Adaptive trimmed median filter	Colorization method	Proposed method
Aloe	33.17	36.00	36.09
Baby1	38.76	40.61	40.10
Bowling1	35.19	35.54	36.31
Cloth1	49.95	50.69	51.87
Average	39.27	40.71	41.09

Table 5 Computing-time comparisons (unit: second)

	Adaptive trimmed median filter	Colorization method	Proposed method
Aloe	1.2618	6.0707	1.6571
Baby3	2.9211	6.2613	1.4819
Bowling1	2.0842	5.7625	1.2129
Cloth4	1.1942	5.9926	1.1892
Lampshade1	2.6610	5.6379	1.3304
Midd1	5.4237	6.2837	2.0217
Monopoly	1.3999	6.5718	1.2303
Plastic	1.0469	5.7512	1.0006
Wood1	2.8990	6.3263	1.4248
Average	2.3213	6.0731	1.3943

5 Conclusions

A point-by-point alignment method for the fusion of the color and depth images captured by Kinect v2 is the main contribution of this paper. The basic approach of the point-by-point registration is to find a sophisticated transformation to finely align the cropped color and depth images by the SDK function of the Kinect v2. Our transformation matrix yields the point-to-point correspondence with a pixel-accuracy, which introduces new applications for Kinect v2. Also, as a result of the fine registration, we can exploit the pixel-wise edge information in the color image for the hole-filling of the corresponding hole-pixel in the depth image. That is, our hole-filling method is to exploit the position information of the color edges extracted from the Kinect v2 sensor to guide the accurate hole-filling around the object boundaries. Experimental results show that the proposed hole-filling method recovers more accurate depths particularly at the object boundaries than the existing method.

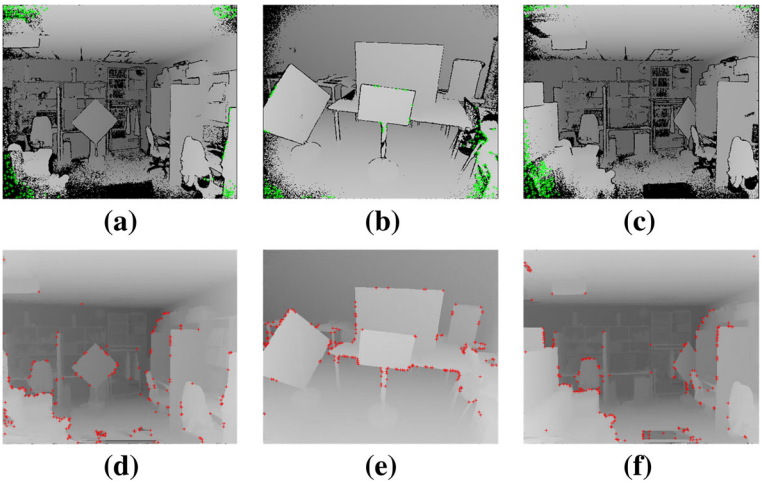

Fig. 24 Feature extraction comparison: (**a**), (**b**), (**c**) Feature detection on the original depth images, (**d**), (**e**), (**f**) Feature detection after the depth completion

Acknowledgments This work was supported by the MSIP (Ministry of Science, ICT and Future Planning), Korea, under ITRC support program (IITP-2016-H8501-16-1014) supervised by IITP and by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2015R1D1A1A01057269). C.S.Won was supported by the research program of Dongguk University, 2016.

References

1. Abdel-Aziz YI (1971) Direct linear transformation from comparator coordinates in close-range photogrammetry. In: ASP Symposium on Close-Range Photogrammetry in Illinois
2. Breuer T, Bodensteiner C, Arens M (2014) Low-cost commodity depth sensor comparison and accuracy analysis. In: SPIE Security+ Defence International Society for Optics and Photonics, pp 92500G–92500G
3. Canny J (1986) A computational approach to edge detection. IEEE Trans Pattern Anal Mach Intell 6:679–698
4. Chen C, Cai J, Zheng J, Cham T-J, Shi G (2013) A color-guided, region-adaptive and depth-selective unified framework for kinect depth recovery. In: Multimedia Signal Processing (MMSP), 2013 I.E. 15th International Workshop on. IEEE, pp 7–12
5. Chen L, Lin H, Li S (2012) Depth image enhancement for Kinect using region growing and bilateral filter. In: Pattern Recognition (ICPR), 2012, 21st IEEE International Conference, pp 3070–3073
6. El-laithy RA, Huang J, Yeh M (2012) Study on the use of microsoft kinect for robotics applications. In: Proceedings of Position Location and Navigation Symposium (PLANS), pp 1280–1288
7. Ho YS (2013) Challenging technical issues of 3D video processing. J Converg 4(1):1–6
8. Horng Y-R, Tseng Y-C, Chang T-S (2010) Stereoscopic images generation with directional Gaussian filter. In: Proceedings of 2010 I.E. International Symposium on Circuits and Systems (ISCAS), pp 2650–2653
9. Jung S-W, Choi O (2013) Color image enhancement using depth and intensity measurements of a time-of-flight depth camera. Opt Eng 52(10):103104–103104
10. Kinect 1 vs. Kinect 2, a quick side-by-side reference. Available online: http://channel9.msdn.com/coding4fun/kinect/Kinect-1-vs-Kinect-2-a-side-by-side-reference. Accessed 11 Oct 2015
11. Lai K, Bo L, Ren X, Fox D (2011) A large-scale hierarchical multi-view rgb-d object dataset. In: Robotics and Automation (ICRA), 2011 I.E. International Conference on, pp 1817–1824
12. Le AV, Jung S-W, Won CS (2014) Directional joint bilateral filter for depth images. Sensors 14(7):11362–11378
13. Levin A, Lischinski D, Weiss Y (2004) Colorization using optimization. ACM Trans Graph 23(3):689–694
14. Qi F, Han J, Wang P, Shi G, Li F (2013) Structure guided fusion for depth map inpainting. Pattern Recogn Lett 34(1):70–76
15. Rosten E, Drummond T (2006) Machine learning for high-speed corner detection. In: Computer vision–ECCV 2006. Springer, Berlin Heidelberg, pp 430–443
16. Rother C, Kolmogorov V, Blake A (2004) Grabcut: interactive foreground extraction using iterated graph cuts. ACM Trans Graph 23(3):309–314
17. Rothwell CA, Forsyth DA, Zisserman A, Mundy JL (1993) Extracting projective structure from single perspective views of 3D point sets. In: Proceedings of Fourth International Conference on Computer Vision, pp 573–582
18. Scharstein D, Pal C (2007) Learning conditional random fields for stereo. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), pp 1–8
19. Tomasi C, Manduchi R (1998) Bilateral filtering for gray and color images. In: Sixth International Conference on Computer Vision, 1998, pp 839–846
20. Vipparthi SK, Nagar SK (2014) Color directional local quinary patterns for content based indexing and retrieval. Hum Centric Comput Inf Sci 4(1):1–13
21. Wasza J, Bauer S, Hornegger J (2011) Real-time preprocessing for dense 3-D range imaging on the GPU: defect interpolation, bilateral temporal averaging and guided filtering. In: Computer Vision Workshops (ICCV Workshops), 2011 I.E. International Conference on, pp 1221–1227
22. Yang J, Ye X, Li K, Hou C, Wang Y (2014) Color-guided depth recovery from RGB-D data using an adaptive auto-regressive model. IEEE Trans Image Process 23(8):3443–3458

Wanbin Song received B.S. and M.S. degrees in the Division of Electronics and Electrical Engineering from Dongguk University, Seoul, in 2013 and 2015. He is currently an Engineer at The MathWorks Korea, LLC. His current research interests include multi-sensor registering, 3D Point clouds processing and depth image improvement.

Anh Vu Le received his BS in electronic and telecommunication engineering of Ha Noi University of Technology in 2007 and his MS and Ph.D degrees in Electronic and Electrical engineering from the Dongguk University in 2012 and 2015, respectively. He is currently a postdoc fellow at Korea Institute of Science and Technology. His current research interests include video compression, image interpolation, image filtering, 3D video processing, and robotic vision.

Seokmin Yun received the B.S. degree in electronics engineering from Dongguk University, Seoul, in 2012. He is currently working toward the PhD degree at Division of Electronics and Electrical Engineering, Dongguk University. His research interests include depth image hole filling and human tracking, 2D-3D calibration, and robot vision.

Seung-Won Jung received the B.S. and Ph.D. degrees in electrical engineering from Korea University, Seoul, Korea, in 2005 and 2011, respectively. He was a Research Professor with the Research Institute of Information and Communication Technology, Korea University, from 2011 to 2012. He was a Research Scientist with the Samsung Advanced Institute of Technology, Yongin-si, Korea, from 2012 to 2014. He is currently an Assistant Professor at the Department of Multimedia Engineering, Dongguk University, Seoul, Korea. He has published over 40 peer-reviewed articles in international journals. His current research interests include image enhancement, image restoration, video compression, and computer vision.

Chee Sun Won received the B.S. degree in electronics engineering from Korea University, Seoul, in 1982, and the M.S. and Ph.D. degrees in electrical and computer engineering from the University of Massachusetts, Amherst, in 1986 and 1990, respectively. From 1989 to 1992, he was a Senior Engineer with GoldStar Co., Ltd. (LG Electronics), Seoul, Korea. In 1992, he joined Dongguk University, Seoul, Korea, where he is currently a Professor in the Division of Electronics and Electrical Engineering. He was a Visiting Professor at Stanford University, Stanford, CA, and at McMaster University, Hamilton, ON, Canada. His research interests include MRF image modeling, image segmentation, robot vision, image resizing, stereoscopic 3D video signal processing, and image watermarking.