Texture-assisted Kinect Depth Inpainting

Dan Miao*
Dept. of EEIS, University of Science and Technology of China
Hefei, China
miaodan@mail.ustc.edu.cn

Jingjing Fu, Yan Lu, Shipeng Li
Media Computing Group
Microsoft Research Asia
Beijing, China
{jifu,yanlu,spli}@microsoft.com

Chang Wen Chen
Dept. of CSE, State University of New York at Buffalo
Buffalo, NY, USA
chencw@buffalo.edu

***Abstract*—The emergence of Kinect facilitates the possibility of depth capture in real-time and with low cost by consumers. It also provides powerful tool and inspiration for researchers to engage in new array of technology development. However, the quality of the depth map captured from Kinect is still inadequate for many applications due to holes, noises and artifacts existing within the depth information. In this paper, we present a texture assisted Kinect depth inpainting framework, aiming at obtaining improved depth information. In this framework, the relationship between texture and depth is investigated, and the characteristics of depth are also exploited. More specifically, texture edge information is extracted to assist the depth inpainting. Furthermore, filtering and diffusion are designed for hole-filling and edge alignment. Experiment results demonstrate that the Kinect depth can be appropriately repaired in both smooth and edge region. Comparing with the original depth, the inpainted depth information enhances the quality of advanced processing such as 3D reconstruction.**

I. INTRODUCTION

With the emergence of Kinect, depth capture in real-time and with low cost comes true. With the Kinect depth map, the traditional research topics in many areas can be investigated, such as 3D reconstruction, segmentation, recognition, tracking, etc., and some improvements have been achieved [1] [2]. Different from the traditional depth generated from the stereo video or TOF, the Kinect depth is generated by the measurement of the infrared structured light spackles variation. In more detail, the structured light in a pseudo random pattern is sent from the projector and received by the CMOS sensor after reflection. By comparing with the light pattern which is hardcoded in the memory, the horizontal offset can be measured easily, and the depth value of each point can be calculated in terms of the offset.

However, the quality of the depth captured from Kinect is inadequate due to the holes, noises, artifacts existing in the depth map. All these will further result in distortions and even annoying effects if the depth is employed directly in related applications. To avoid these distortions and artifacts, the captured depth should be pre-processed first. As introduced above, though the principle of Kinect depth generating can be known, it is difficult to obtain the source images for the depth generating, like left view and right view in stereo video. What we can obtain are just the output texture image and depth image. Though these two types of images are captured by two independent sensors, the calibration can be implemented to match the texture with the depth in pixel level [3]. Based on this fact, the texture image can be employed as assistant information in the depth processing.

In a sense, the imperfect depth can be regarded as a kind of damaged image with irregular holes and anamorphic edges. As we know, the normal damaged images are repaired by the image inpainting, which has been studied for years. Many models have been introduced for image inpainting, such as [4] [5], etc. These models cannot directly apply to depth images, since they are designed for texture images. There are also a wide range of stereo correspondence algorithms. Excellent survey can be found in [6]. However, these algorithms are not useful for Kinect depth due to different generating principles.

In this paper, we propose one texture assisted Kinect depth inpainting framework, in which the depth map is inpainted with respect to its local spatial properties in both depth map and texture map. As the result, the invalid depth region can be filled in and the depth boundary is aligned with that of the texture. In our algorithm, the edges are extracted from texture image as the assistant information for depth inpainting. Based on the extracted texture edge, depth image is partitioned into two regions: smooth region and edge region. Diffusion algorithms with different rules are designed for these two regions to conduct the depth inpainting. To the best of our knowledge, this is the first work that studies Kinect depth inpainting. The experiment results show that hole-filling and edge alignment are performed well by the texture assisted depth inpainting. The rendering quality of the processed depth map is substantially improved in comparison with the original one.

The remainder of this paper is organized as follows. In Section II, we introduce the framework of the proposed depth inpainting. Furthermore, the physical characteristic of depth map is discussed in this section. In Section III, the pre-processing of the texture and depth image is introduced. The key techniques of the depth inpainting are described in Section IV. The experimental results are demonstrated in Section V. Finally, we conclude this paper in Section VI.

II. FRAMEWORK OF OUR DEPTH INPAINTING SCHEME

On account of Kinect's capturing principle, the Kinect depth possesses some special characteristics: First, the depth value is valid within a range which is between 800mm to 4000mm for Microsoft Kinect SDK. If out of this range, the depth value is invalid with zero value. Second, even if the depth value

*This work was done while the author was with Microsoft Research Asia as a research intern.

lies in the valid range, it is possible that Kinect fails to derive the depth values due to lighting influence and imaging constraints. The invalid depth regions exhibit as irregular holes in the depth map, and the anamorphic edges are fattened or shrunk compared to the ground truth. Third, even in the smooth region there is fluctuation in depth value, and the jump size is nonlinear to the depth value.

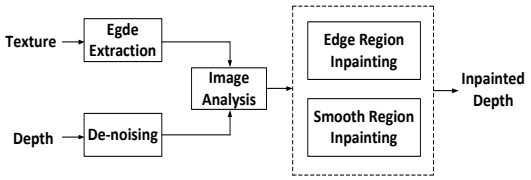

Fig. 1 The framework of our proposed depth inpainting scheme.

In terms of the characteristics of the Kinect depth, we propose a depth inpainiting scheme to repair the depth map. Fig. 1 illustrates the architecture of the proposed depth inpainting scheme. In our framework, the edges are first extracted from the original texture image which assist the inpainting module to process the depth image. The denoising module is employed as the preprocessing to suppress the fluctuation effects. Based on the edges extracted from the texture, the depth image is partitioned into two regions: smooth region and edge region. The partial differential equations (PDEs) method is employed in depth inpainting, and the different rules of PDEs are designed in these two regions to jointly conduct the edge alignment and hole-filling.

III. TEXURE AND DEPTH IMAGE PRE-PROCESSING

In this section, we will discuss the pre-processing of the texture and depth image in our framework.

A. Depth Image De-noising

As aforementioned, to avoid the fluctuation impact, the input depth image is first smoothed by the filter. There are many mature tools available to perform the de-noising. In our work, the bilateral filter [7] is adopted as the edge-preserving denoising filter. Bilateral filter provides a weighted average of nearby pixels as the filtered result, with two kernels defining the weight: a domain filter kernel and a range filter kernel. Domain filter kernel is used to describe the geometric closeness between two pixels, while range filter kernel is used to measure the photometric similarity. For a pixel p, the filtered depth value is:

$$D'_p = \frac{1}{K_p}\sum_{q\in\Omega_p} D_q\, f(p,q)g(D_p,D_q) \tag{1}$$

where f is the spatial filter kernel with the pixel locations p, q as inputs; g is the range filter kernel with corresponding pixel values D_p , D_q as inputs. Ω_p is the pixel sets used in calculating the filtered result and K_p is a normalization factor which is equal to the sum of the $f(*)\cdot g(*)$ filter weights. Bilateral filter can preserve edges since pixels belongs to different regions are usually more different from each other on pixel value and thus leads to a small value for range filter kernel.

B. Edge Extraction

As discussed in Section II, texture edge information plays an important role in the proposed inpainting scheme. Since there is no need to extract complete and continuous edges to represent image's topological properties, which is required in segmentation or object restoration, Canny operator is adopted in our algorithm to extract texture edges with one-pixel width. Other edge extraction operators might be used, though satisfactory results were already obtained with this very simple selection.

IV. DEPTH INPAINTING

Generally speaking, the ideal depth map should be continuous in most parts of the map besides the discontinuity between two objects. Based on this characteristic, in our work the smooth region is treated differently from the edge region. For smooth region, high-order partial differential equations (PDEs) model is employed in our scheme, since this model can predict smooth region well as shown by impressive results. For edge region, a revised PDEs method is designed for jointly considering hole-filling and edge alignment.

Before the description of two region inpainting schemes, we will introduce the theory of the Laplace equation which is adopted as the inpainting method as one of the PDEs methods. The continuous form of the Laplace equation is as follow:

$$\frac{\partial I}{\partial t} = \Delta I \tag{2}$$

where $\Delta = (\partial^2)/(\partial x^2) + (\partial^2)/(\partial y^2)$ is known as the Laplacian operator. As shown in [8], in continuous case the solver is essentially convolving the initial state with a Gaussian kernel with variance t

$$I(x,y,t) = I(x,y,0) \circledast G(x,y,t) \tag{3}$$

This relation reveals that the Laplace equation is capable in generating smooth regions. Specifically, the discrete form of (2) is

$$I(x,y,t+1) = I(x,y,t) + c(x,y,t)\cdot\Delta I \tag{4}$$

Until convergence, i.e., $\|I(t+1) - I(t)\|$ less than a threshold, the state $I(t+1)$ is regarded as the solution. Note that $c(x,y,t)$ as the step size can vary in iterations but should only ensure the convergence.

Now turn to realization, the Laplacian can be estimated by

$$\mathcal{L}I(x,y) = \sum_{(i,j)\in\mu_8(x,y)} \kappa(i,j)(I(i,j) - I(x,y)) \tag{5}$$

where $\mu_8(x,y)$ involves eight neighboring pixels of *(x, y)* and $\kappa(i,j)$ is an indicator function that is evaluated to one for available pixels and zero for unavailable ones. By setting the step size to

$$c(x,y,t) = c(x,y) = \left(\sum_{(i,j)\in\mu_8(x,y)} \kappa(i,j)\right)^{-1} \tag{6}$$

we get the final evolution

$$I(x,y,t+1) = c(x,y)\sum_{(i,j)\in\mu_8(x,y)} \kappa(i,j)I(i,j,t) \tag{7}$$

Texture image usually possesses an edge map similar to that of the corresponding depth but not exactly equivalent. If the object physical features are preserved in the texture map, it is possible that the depth and the texture share the same edges after calibration. However, the smooth surface could be covered by complicated texture, and the depth discontinuity might be neglected by low contrast in texture.

Based on analysis, we categorized their edge relation to four cases. Case 1: Both texture edge and depth edge existing; Case 2: Texture edge existing, depth edge no existing; Case 3: Neither of them existing; Case 4: Texture edge no existing,

depth edge existing. Since depth map is noisy and instable in comparison with the texture, we partition the depth map based on available texture edges. For the first two cases, the region is regarded as edge region no matter whether depth edges exist or not. Since our edge region inpainting is one more severe scheme than PDEs, it works well for the first case and obviously also well for the second. The designed smooth region inpainting for last two cases will be discussed in the next subsection. In realization, the inpainting scheme is conducted in block level. Based on whether the block contains texture edge or not, the blocks are classified into two categories: edge region block and smooth region block. Next we will describe the inpainting schemes respectively.

A. Smooth Region Inpainting

For the smooth region block, the key problem is whether the information in the neighbors is enough and credible for inpainting. As discussed in section II, the holes exist due to several reasons.

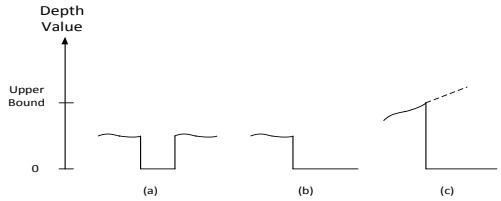

Fig. 2 The conditions of the holes in smooth region

For small size hole such as shown in Fig. 2(a), since there exist enough available pixels in neighbors, the hole can be filled by Laplaces equation similar to the inpainting in texture image. For the large size hole, if the pixel value of the neighbor is near the upper bound of depth range, the hole might be generated due to out of depth range as shown in Fig. 2(c). Since the Laplace equation can describe the trend of signal in smooth region, we can predict the pixel value by expending the depth region from the available side. If the value of the neighbor is within the valid range as shown in Fig. 2(b), the light speckle of the hole region is missed due to depth discontinuity or specular reflection. That means this region might not be smooth in depth ground truth though there is no texture edge extracted as mentioned above in Case 4. Therefore this region will not be inpainted since the information from the neighboring pixels is not enough to predict the real depth value.

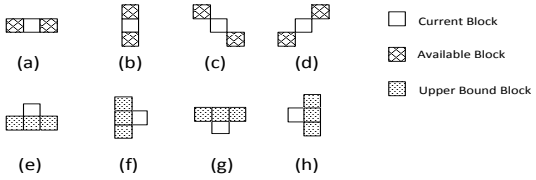

Fig. 3. The Judgement of the information sufficiency

Turn to realization, in order to judge if the neighboring information is enough for inpainting, we measure the available pixels number in the block. If the ratio is beyond a threshold (e.g. 20%) the block is set as *available block* otherwise it is set as un*available block*. For the available block, the average of pixel value in the block is calculated. If the average is near the upper bound of depth range, the block is named as *upper bound block*. Based on the above discussion, we investigate the eight neighboring blocks, and the following cases shown in Fig. 3 will be regarded as the information is enough in the neighbors. We need to point out that we investigate this problem in block level that is efficient to obtain the quasi-global information and to avoid wrong propagation in pixel level.

B. Edge Region Inpainting

The depth edge fattening or shrunk effects commonly exist along the boundary of the foreground object. Therefore, in the process of the inpainting, besides hole-filling the edge alignment should be also performed. An typical edge region block with edge fattening effect is illsustated in Fig. 4. The solid line within the block indicates the texture edge while the dash-dot line is the depth edge.

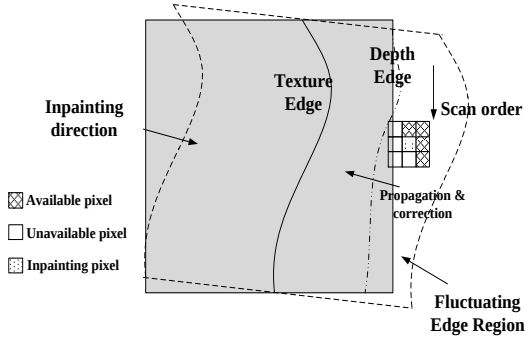

Fig. 4. Fluctuating Edge Region for edge region inpainting

To address these problems, we introduce *fluctuating edge region*, which locates beside the texture edges with adaptive size for inplainting. As shown in Fig. 4, the fluctuating edge region is restricted by two pairs of parallel dashed lines. This region is generated as follow: for each texture edge in the edge region block, the edge direction $\boldsymbol{k}$ is figured out based on its slope. The fluctuating edge region is extended along the direction $\boldsymbol{k}^{\perp}$, perpendicular to the direction $\boldsymbol{k}$. The progress of the expansion will stop if another texture edge is met or the width is beyond a threshold. The pixels in this region are treated as "hole" before inpainting. The inpainting is performed in the direction $\boldsymbol{k}^{\perp}$ from outer pixel to interior pixel and is stopped when meeting the texture edge. The pixel updating process can be depicted by following formula:

$$I(x,y,t+1) = \chi(x,y,t+1)\cdot I(x,y,t) + (1-\chi(x,y,t+1))\cdot R(x,y,t+1) \quad (8)$$

where $R(x,y,t+1)$ is the pixel value calculated based on PDEs as discribed in Eq. (7), and

$$\chi(x,y,t+1) = \begin{cases} 0 & if\,|I(x,y,t) - R(x,y,t+1)| > \alpha \\ 1 & otherwise \end{cases} \quad (9)$$

where α is the maximum tolerable difference. The final updated pixel value is the weighted sum of original value $I(x,y,t)$ and inpainting value $R(x,y,t+1)$. Here the weight coefficient is set as one indicator function $\chi(x,y,t+1)$. That means the pixel value is modified to the inpainting one if the difference is beyond the threshold, otherwise it is kept as original value. In this way, the pixel value will propagate from smooth region and stop beside the texture edge. Finally, the depth edge will match with the texture one.

Fig. 5. The original texture image.

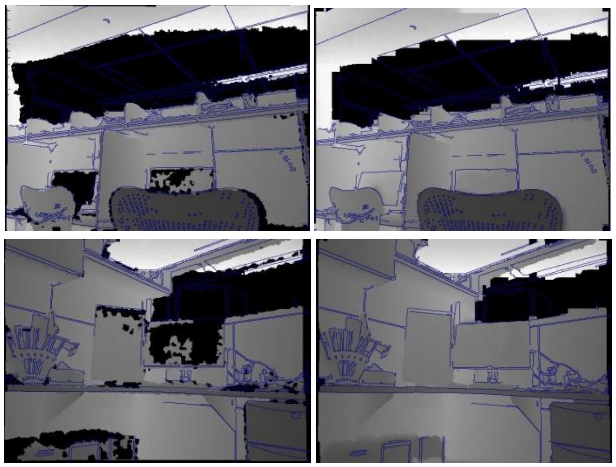

Fig. 6 Comparison of the depth map. Left: depth image without inpainting. Right: depth image after inpainting.

V. EXPERIMENT RESULTS

We have carried out experiments to evaluate the effectiveness of the proposed depth inpainting scheme. We capture the texture image and depth image using the XBOX 360 Kinect. The resolutions of texture and depth are set as 640*480. Since there is no ground truth for depth, we adopt the subjective comparison which is selected as rendering results in our paper to compare the depth map quality. The 3D mesh is rendered using texture and depth image and shown by the MeshLab software [9].

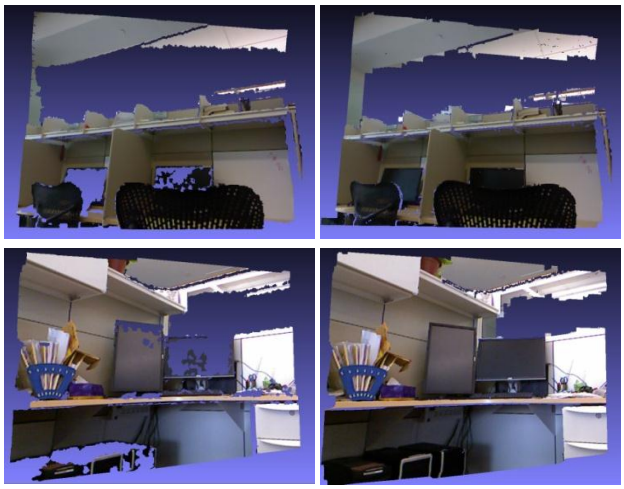

Fig. 7. Comparison of the rendering results. Left: Rendering result using the depth map without inpainting. Right: Rendering result using the depth map with inpainting.

The comparison of the depth image is shown in Fig. 6. The blue curves in these images represent the edge extracted from texture image. Comparing with the pair of the two images, we can see that small holes in the original depth map are filled in the processed one including the monitor screen region in the image. (As shown in Fig. 5) The huge holes of the glass and the roof region are not filled in our scheme since either there exist available blocks only in one side and these blocks are not upper bound one or the depth value is out of the range. Therefore the information is not enough for inpainting. The rendering results comparison is shown in Fig. 7. Note that rendering result using the processed depth map can achieve much better quality than that without depth inpainting subjectively. The detail result is shown in Fig. 8. We can see there are foreground fattening or shrunk effects near depth discontinuities between two objects. These effects in depth discontinuities will lead to poor rendering quality. After rendering, the rendering quality is improved due to satisfactory edge alignment.

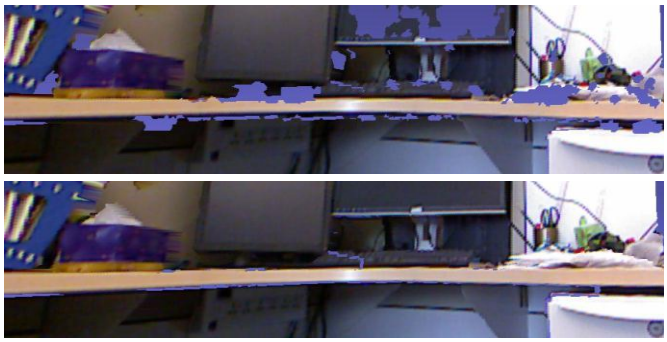

Fig. 8. Comparison of the detail rendering results. Top: the partial mesh rendered from the original depth map. Bottom: the partial mesh rendered from the inpainted depth map

VI. CONCLUSION

In this paper, we have presented a novel Kinect depth inpainting framework. More specifically, we proposed a texture assisted depth inpainting scheme in which the texture edge is extracted as assistant information. Based on the texture edge, smooth region and edge region in depth map will be performed inpainting differently for edge alignment and hole-filling. Experiment results have demonstrated that the processed depth will improve the advanced processing quality.

REFERENCES

[1] T. Weise, S. Bouaziz, H. Li, and M. Pauly, "Realtime performance-based facial animation," in Proc. ACM SIGGRAPH, 2011

[2] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake, "Real-time human pose recognition in parts from single depth images," in Proc. CVPR, 2011, pp. 1297-1304.

[3] http://research.microsoft.com/en-us/um/redmond/projects/kinectsdk/

[4] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester, "Image inpainting," in Proc. ACM SIGGRAPH, 2000, pp. 417-424

[5] T. F. Chan, and J. Shen, "Mathematical models for local nontexture inpaintings," SIAM J Appl. Math., vol. 62, no. 3, pp. 1019-1043, 2002.

[6] D. Scharstein, and R. Szeliski "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," International Journal of Computer Vision, 47(1-3): 7-42, Apr.-Jun. 2002.

[7] C. Tomaso, and R. Manduchi, "Bilateral Filtering for Gray and Color Images," in Proc. ICCV, pp. 839-846.

[8] P. Perona and J. Malik. "Scale-space and edge detection using anisotropic diffusion." *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 12, no. 7, pp. 629-639, Jul. 1990.

[9] http://meshlab.sourceforge.net