DEPTH MAP ENHANCEMENT ON RGB-D VIDEO CAPTURED BY KINECT V2

Ke-Yu Lin and Hsueh-Ming Hang

Electrics Inst., ECE College, National Chiao Tung University, Taiwan
Email: mvp9sit@yahoo.com.tw, hmhang@nctu.edu.tw

ABSTRACT

The quality of depth map is one key factor contributing to the quality of 3D video and virtual reality (VR) rendering. In this study, we use RGB-D camera (Microsoft Kinect for Windows v2) to capture the color sequences and depth sequences as our system inputs. The captured depth map contains various noises and artifacts in addition to the occlusion regions. We use the color sequences in both spatial domain and time domain to improve the quality of the depth map. Our main contributions are alignment between color and depth images and reducing artifacts in the reflection regions. Several techniques are adopted, modified, and re-designed such as moving object compensation, unreliable depth pixel detection, and locally adaptive depth pixel refinement algorithm. The experimental results show that the quality of the depth map is significantly improved.

Index Terms — Depth map, depth refinement, camera synchronization, backward warping, disocclusion filling, Kinect v2

1. INTRODUCTION

Owning to the growing popularity of 3D video and virtual reality (VR), there is growing demand for high-quality depth map for 3D-video rendering. The current consumer depth cameras produce depth maps with various kinds of noises and artifacts. Therefore, a number of depth map enhancement algorithms have been developed to improve the depth map quality. Most of them combine the information from the color camera or another depth camera to reduce the defects of the target depth map. For instance, in [1], it combines the disparity maps derived from two color images with ToF camera SR4000 to produce high resolution depth map. Other similar works are [2] and [3].

In this paper, we propose a depth map refinement algorithm to improve the quality of captured depth map sequence. We use the Microsoft Kinect v2 to capture color and depth sequences. We up-sample the depth map and correct the wrong depth pixel, and also fill up the holes with the correct depth values. We align the depth map with its associated high resolution color image. The overall flow diagram of our proposed scheme is shown in Figure 1. To begin with, we align the raw depth map to the color image with proper motion and shift compensation. After that, we adopt the method in [4] to detect the occlusion regions, and fill them up with the correct background. Then, we identify the unreliable depth pixels and the reflection regions and fill in the segmented depth values in their neighborhood. At the end, we use a simple bilateral filter to smooth out the depth map random noises if necessary. We repeat the same work flow for every frame in an RGB-D sequence to construct the final depth map sequence. We will elaborate on the key components in our algorithm in the following sections.

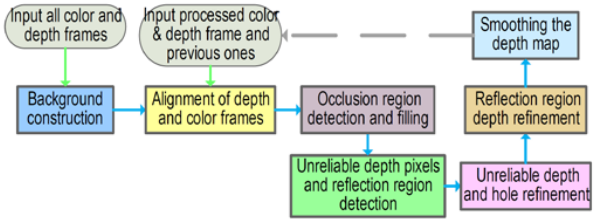


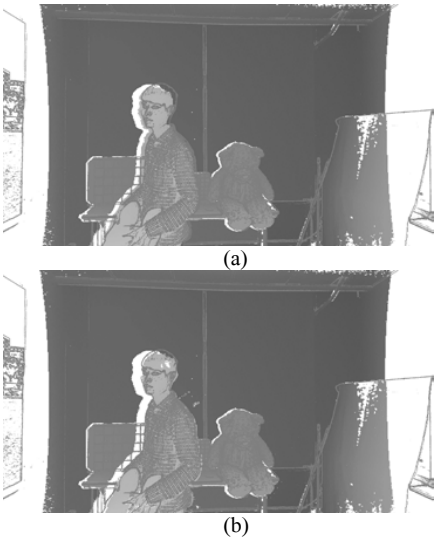
Figure 1. Overall work flow of proposed method.

2. ALIGNMENT BETWEEN COLOR IMAGE AND DEPTH MAP

Because the color camera and the depth camera in Kinect v2 are not synchronized, they capture the scenes at slightly different time instances as shown in Figure 2(a). To align the depth map with the corresponding color image, we start from the last refined depth frame, which has matched to its associated (last) color image. We shift it according to the motion vectors [5] derived from the color images to produce a stitched depth map. We assume the stitched depth map is matched to the color image in the spatial domain.

In the following steps, we shift the moving part in the raw depth map to the corresponding location in the stitched depth map. Firstly, we use the Scale Invariant Feature Transform (SIFT) [6] to find the feature points of the two depth maps. Then, we match the feature points of the two depth maps. Using the matched feature pairs, we calculate a homography matrix transforming one depth map to another. However, the two depth maps are not exactly the same scene because they are captured at two time instances. As a result, using the homography matrix to transform does not give good result. Instead, we use the shifting vectors to map the raw depth pixels to the stitched map. Because each matched feature pair can provide a shifting vector candidate, we need to identify the one that is most reliable shifting vectors. After careful evaluation on the most reliable shifting vectors, we shift the moving

object in the raw depth map by the chosen shifting vector as shown in Figure 2(b), which is an aligned depth map.



(a) (b)

Figure 2. The gray gradient of a color image overlaps on its raw depth map of the 3rd frame in Person and Bear seq. (a) The original depth map. (b) The aligned depth map. [White regions are missing depth pixels due to occlusion and defects.]

3. UNRELIABLE DEPTH PIXEL AND REFLECTION REGION DETECTION

After the alignment, the occlusion region filling is done using the approach in [4]. Next, we process the unreliable depth pixels. We first identify them. Often, depth pixel error happens near edges (object boundaries) due to depth map upsampling and the spatial transformation in the alignment process. Also, there are flicker noises appearing in a depth map. To label these depth pixels as unreliable depth pixels, we use the information from the color images. There are some similar works, which also make use of the color image to refine the depth map such as segmentation based refinement [7]. However, sometimes an inaccurate segmentation would lead to undesirable result in a region. Hence, we do the pixel-by-pixel check to increase accuracy.

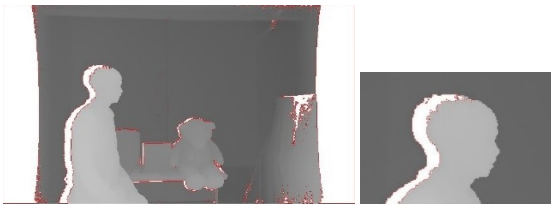
Firstly, we partition a depth map into blocks and process each block separately. We divide the depth pixel in each block into 2 or 3 groups according to the histogram of depth values in the block. This is a locally controlled segmentation. Then, we calculate the average hue and gray values of each depth group. We also calculate the difference of hue and gray values for each depth pixel against the average ones and determine which pixel should be labeled as unreliable depth pixel. The result is shown in Figure 3, in which the red-color points are unreliable depth pixels.

Figure 3. Depth map with labeled unreliable depth pixels.

The difficult part is the reflection region, where the missing depth pixels are due to the reflection surface (which interfere with the infrared sensor). These pixels often have legitimate depth values and are clustered to form a sizable region. Thus, we cannot simply use the conventional noise-reduction method to process them. An example is shown in Figure 4, the red window in hair is a reflection region. The depth values in the hair are wrongly assigned by using the background depth. Because the dominant depth value may be wrong in the reflection region, we cannot use the previous unreliable depth pixel detection method to detect the reflection region. Hence, we detect the reflection region and refine it using a specifically designed procedure.

Figure 4. Display of reflection region in depth map.

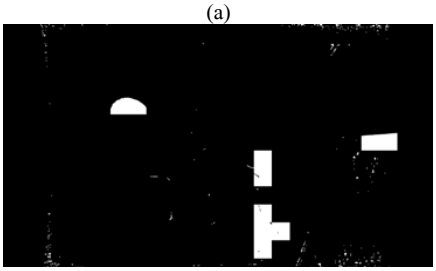
Our assumption is that the large hole of depth map is due to two main causes, one is occluded region, and the other is reflection region. We observe that the hole caused by occluded region is often strongly connected (no gap), while the hole caused by reflection region is often sparse (gaps between pixels). Because sometimes the Kinect receiver does not detect the correct infrared light and produce the correct depth in the reflection region, it results in sparse depth holes. The hair in Figure 5(a) is an example of reflection region. This part is enlarged in Figure 5(b). Thus, our first step is to detect these sparse depth holes as shown in Figure 6(a). Then, we grow from these seeds to cover the entire reflection region based on the color similarity, and finally label the reflection region as shown in Figure 6(b).



(a) (b)

Figure 5. The full depth map is shown in (a) and the enlarged part of head is shown in (b).

(a)



(b)

Figure 6. Sparse depth holes in the depth map (a). Final label (detection) of reflection region in the depth map (b).

4. DEPTH REFINEMENT ON THE UNRELIABLE DEPTH PIXELS AND HOLES

After we label all the pixels to be refined, we apply the proposed pixel refinement algorithm to them. The basic assumption is that the depth value of its neighborhood is mostly correct. Thus, we use the neighborhood to alter the depth value of target pixel based on the information of color, distance, and depth pixel reliability.

First, we set an adaptive window centered at an unreliable depth pixel or hole which is the target we like to modify. Then, we pick up the neighborhood pixels inside the window as candidates, and use them to calculate and estimate the correct depth value of the target depth pixel. Next, we calculate the weights of the depth candidates and then compute the weighted average. The weight is location dependent and is defined by the following formula,

$$\text{weight\_location} = \mathbf{e}^{-\frac{x\_dis^2+y\_dis^2}{100}}, \quad (1)$$

where x_dis is the distance in the x-axis between the depth candidate and the target pixel, and similarly, y_dis is the distance in the y-axis. And the weight in color similarity is defined below,

$$\text{weight\_color} = \mathbf{e}^{-\left(\frac{diff\_gray^2}{50}+\frac{diff\_hue^2}{50}+\frac{diff\_sat^2}{400}\right)}, \quad (2)$$

where diff_gray is the difference in gray intensity between the depth candidate and the target depth pixel, diff_hue is the difference in hue, and diff_sat is the difference in saturation. The weight of an unreliable depth pixel is defined as follows.

If a depth pixel∈unreliable, weight_reliable = 0.1, else = 1. (3)

As for the depth candidate in a reflection region, its weight is assigned as follows.

If a depth pixel∈reflection, weight_reflection = 0.3, else = 1. (4)

Finally, we combine all the weights to form the final weight of a depth candidate as follows.

Weight = weight_location×weight_color×weight_reliable×weight_reflection. (5)

The parameters in the above equations are selected experimentally and they are fixed constant values for all test sequences. The experimental results show that the weighting parameter is quite robust. After finishing all the weights calculation, we remove the depth candidates of weight lower than 0.7 times the average weight. Next, we calculate the average depth value using all the remaining depth candidates together with their weights and also calculate the depth variance of these candidates. We check whether the variance is smaller than a specified threshold. If so, we jump to the final decision step; otherwise, we come back to remove the unlikely candidates so that the depth variance of the candidates is lower than the threshold. In the final step, we replace the depth value of the target depth pixels by the majority of the surviving depth candidates if the number of surviving candidates is larger than a specific threshold. The above process can be described by the pseudo codes shown in Figure 7. In Figure 7, D is the depth value of depth candidate, W_C is the weight of depth candidate, N_c is the number of remaining inlier depth candidate, and σ^2 is variance threshold, set to 200 based on experiments; the threshold value is not very sensitive.

```
1: preocedure RefineDepth(D,WC)
2:     Z = ∑C∈inlier Wc
3:     Wavg = Z / NC
4:     Davg = 1/Z ∑C∈inlier Wc × Dc
5:     Var = 1/Nc ∑C∈inlier(Dc − Davg)^2
6:     while Var > σ^2 and NC is changing do
7:         inlier = {C| (C∈inlier) and (WC > 0.7*Wavg)}
8:         Z = ∑C∈inlier Wc
9:         Wavg = Z / NC
10:        Davg = 1/Z ∑C∈inlier Wc × Dc
11:        Var = 1/Nc ∑C∈inlier(Dc − Davg)^2
12:    end while
13:    return Maj(Dc)
14:end procedure
```

Figure 7. Simplified algorithm of depth pixel refinement.

5. DEPTH REFINEMENT ON THE REFLECTION REGION

As discussed earlier, we need to refine the depth pixels in the reflection region using a special procedure. This process is shown in the flowchart of Figure 8. We first compare the gray intensity between the current pixel and the background pixel to check whether it belongs to the background. If it is not, we compare the gray intensity between the current pixel and the previous frame pixel to check whether it is a static pixel. If not, we simply use the process described

in Figure 7 to refine it. But, we make some modifications in the weights of depth candidates.

Our weight_location and weight_color follow the same definitions in the last section. The weight_reliable is defined by the following formula.

If a depth pixel∈unreliable, weight_reliable =0.8, else = 1. (6)

After refining the unreliable depth pixels in the last section, the unreliable depth pixels should become more reliable. Therefore, we increase the weight of unreliable depth candidates in (6).

Moreover, we define the weight_background, which is associated with the background depth candidate below.

If a depth pixel∈background, weight_background=0.01, else=1. (7)

A depth candidate is background or not is checked by its depth value. A pixel is examined whether it is a background pixel before applying this final refinement step. If it belongs to the background depth, we correct its depth value directly and will not go to the final refinement step. At the final step, we cannot use the background neighborhood to refine a non-background pixel. Hence, we reduce the weight of the background depth candidate to 0.01 in (7). Similar to eq.(5), we multiply all the weights associated with a candidate pixel within a window centered at the target pixel. A procedure similar to that in Figure 7 is applied to refine the depth pixels in the reflection region.

After we refine the depth pixels in the reflection region, we use a simple bilateral filter to smooth the depth map. Because most of our operations is pixel-based, we reduce the pixel outliers by using a smoothing filter.

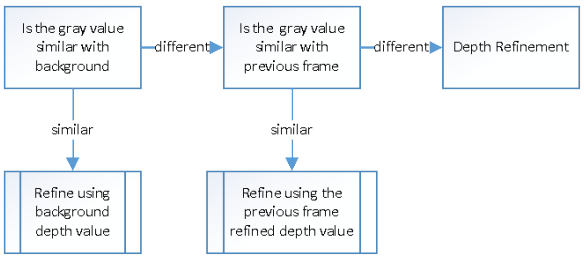


Figure 8. Overall process in reflection region refinement.

6. EXPERIMENTAL RESULTS

The Microsoft Kinect v2 is used to capture images with the depth maps. We use Kinect v2 SDK to warp and interpolate the depth map to match the color images in resolution and to compensate the base line distance between color camera and depth camera. We took four video sequences in our experiments, that is, Person and Bear, Person, Bear Rotation, and Bear Slide. We compare our results with the methods proposed in [8] and [9].

In [8], they also use the color image and depth map from Kinect v2 to improve the depth map. But, they mainly handle the depth holes and do not correct the wrong depth pixels. First, they input all the color frames and depth frames to construct the color and depth background. To achieve a precise depth map, they use the segmentation of color image to assist the refinement of the background depth map. Then, they detect the foreground of each frame by the color background. After that, they fill up the depth holes in the background region with the produced depth background. Finally, the remaining depth holes in foreground are filled by the dominant depth value of a neighborhood region inside the foreground.

In [9], they also use the color image to enhance the depth map. But, their process is restricted to one picture rather than a video sequence. Their algorithm is based on the heat diffusion framework. The pixels with known depth values are treated as the heat sources to diffuse. The diffusion conductivity is designed based on the color image. It will form a linear anisotropic diffusion problem. They case it into walk model and solve the sparse linear system. They need manually to tag out the region needed to refine so that the depth point will be filled up with a known depth value from the heat source. Because it needs manually to tag the region to be refined on every depth map, and we only process a few frame using this algorithm, thus we only compare the results on some selected frames rather than a video sequence.

We clip our depth map to 1460×1080 in comparison. Figure 9 shows the results of the 3rd frame in the Person and Bear sequence. We can see that the method in [9] cannot fill up all the holes and some refined depth pixels are incorrect (Figure 9(d)). From Figure 9(f), we can see that the method in [8] can enhance most parts of a depth map correctly. However, it still suffers from the non-alignment problem, and therefore the depth map does not match well with the color image as shown in Figure 9(g). Also, it does not handle the reflection region well. Our proposed method can refine nearly all the pixels correctly as shown in Figure 9(h) and (i). The difference can be observed more clearly on Figure 10.

Figures 11 and 12 show the experimental results for different dataset. Some processed depth video sequences can be found in [10]. Limited by space, we do not include the details of all the steps and parameters used in our scheme.

7. CONCLUSIONS

The depth map generated by Kinect v2 is of low resolution and has artifacts such as occlusion holes and wrong depth pixels. We propose a depth map refinement algorithm using the information of high-resolution color video captured by Kinect v2 to enhance the resolution and quality of depth map. Different from the previous approaches, we fix the temporal misalignment problem using the motion information in a video sequence and also, we use the feature matching technique to reduce the minor shift. We also construct a background to build the static information to assist the refining process. Another contribution of our algorithm is handling the reflection region. It consists of two steps: reflection region detection and its refinement procedure. At the end, we show the visual results of our

algorithm. Our scheme can produce a good quality depth map, consistent with its associated color image.

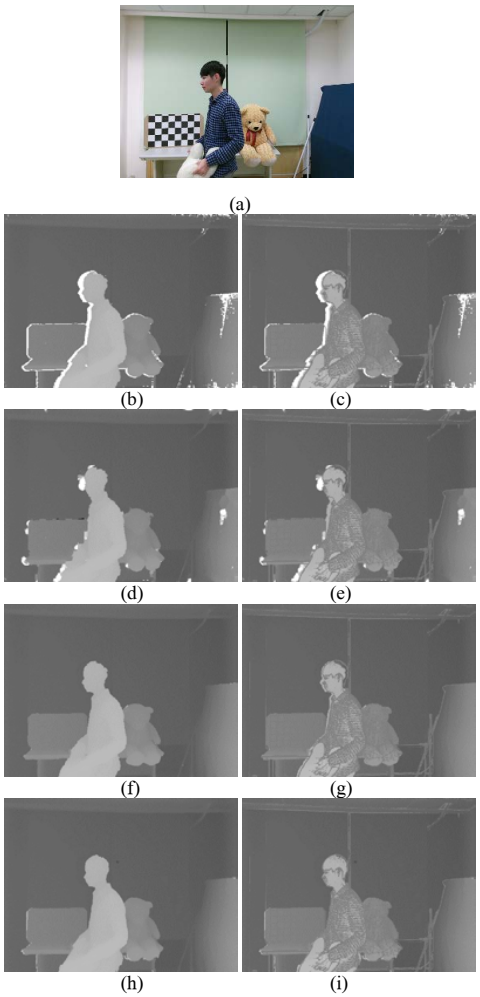


Figure 9. Experimental result of frame 3 of Person and Bear sequence. (a) Color image. (b) Initial depth map. (c) Overlap (b) with the gradient of (a). (d) Processed depth map using [9]. (e) Overlap of (d) with the gradient of (a). (f) Processed depth map from [8]. (g) Overlap (f) with the gradient of (a). (h) Our processed depth map. (i) Overlap (h) with gradient of (a).

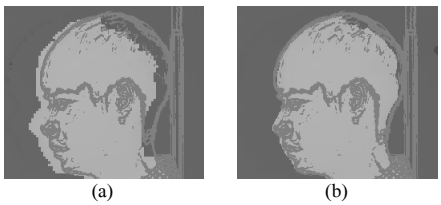


Figure 10. Zoom in of Figure 9. (a) Zoom in view of Figure 9(g). (b) Zoom in view of Figure 9(i).

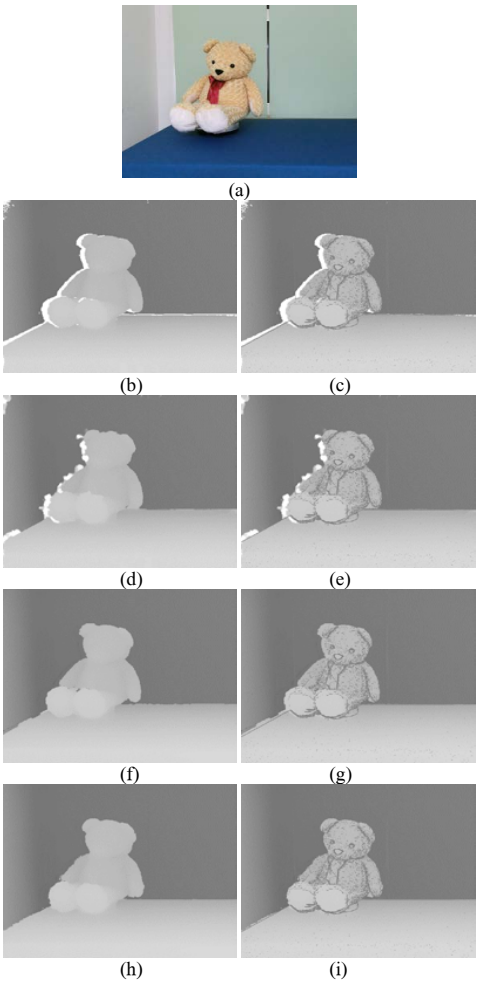


Figure 11. Experimental result of frame 61 of Bear glide sequence. (a) Color image. (b) Initial depth map. (c) Overlap (b) with the gradient of (a). (d) Processed depth map using [9]. (e) Overlap of (d) with the gradient of (a). (f) Processed depth map from [8]. (g) Overlap (f) with the gradient of (a). (h) Our processed depth map. (i) Overlap (h) with gradient of (a).

8. ACKNOWLEDGEMENT

This work was supported in part by the MOST, Taiwan un-der Grant MOST 104-2221-E-009 -069 -MY3 and by the Aim for the Top University Project of National Chiao Tung University, Taiwan.

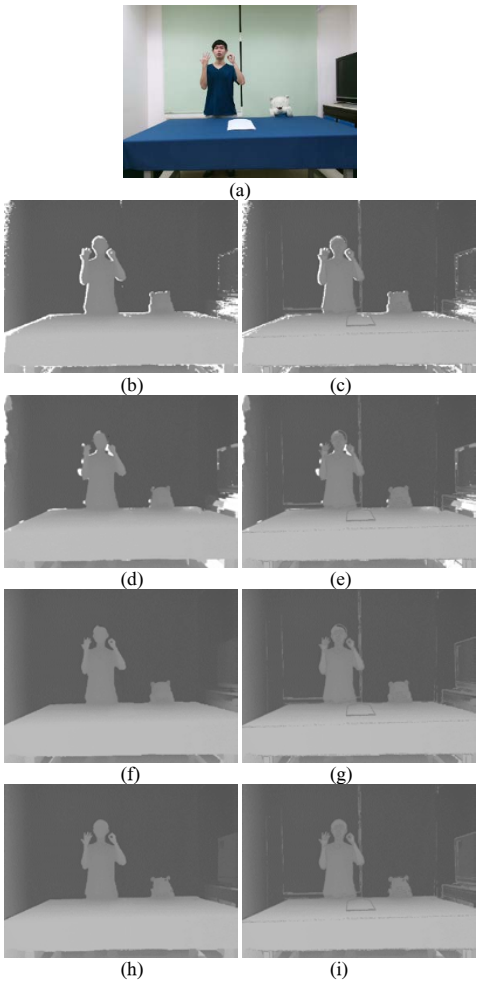


Figure 12. Experimental result of frame 17 of Person sequence. (a) Color image. (b) Initial depth map. (c) Overlap (b) with the gradient of (a). (d) Processed depth map using [9]. (e) Overlap of (d) with the gradient of (a). (f) Processed depth map from [8]. (g) Overlap (f) with the gradient of (a). (h) Our processed depth map. (i) Overlap (h) with gradient of (a).

9. REFERENCES

[1] V. Gandhi, J. Čech and R. Horaud, "High-Resolution Depth Maps Based on Tof-Stereo Fusion." In *IEEE Robotics and Automation* (ICRA), 2012.

[2] D. Ferstl, et.al., "Multi-Modality Depth Map Fusion Using Primal-Dual Optimization." In *IEEE International Conference on Computational Photography* (ICCP), 2013.

[3] Y.-H. Chiu, M.-S. Lee and W.-K. Liao, "Voting-Based Depth Map Refinement and Propagation for 2d to 3d Conversion." In *Asia-Pacific Signal & Information Processing Association Annual Summit and Conferenc*e (APSIPA ASC), 2012.

[4] C.-L. Chien , A. Wang, S.-P. Weng , H.-M. Hang, "A Method of Disocclusion Detection and Filling for the Depth Map Generated by Kinect Sensor," *2015 Conf. on Computer Vision, Graphics, and Image Processing*, Yilan, Taiwan, 2015.

[5] R. Li, B. Zeng and M. L. Liou, "A New Three-Step Search Algorithm for Block Motion Estimation," *IEEE trans. on Circuits and Systems for Video Tech*., pp. 438-442, 1994.

[6] D.G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints." In *Int'l J. Computer Vision*, pp. 91-11, 2004.

[7] S.M. Hong and Y. S. Ho, "Depth Map Refinement Using Superpixel Label Information." In *2016* APSIPA ASC, 2016.

[8] S.-P. Weng, *Depth Map Enhancement based on Its Associated High-Resolution RGB Video*, MS thesis, National Chiao Tung University, July 2015

[9] J. Liu, X. Gong and J. Liu, "Guided Inpainting and Filtering for Kinect Depth Maps." In *IEEE Int'l Conf. Pattern Recognition* (ICPR), 2012.

[10] Experimental results (videos) are shown in http://people.commlab.tw/ Hang's Research Resource/DepthMapEnhancement