Self-taught Recovery of Depth Data

Pan Yang, Haoran Zhao, Lin Qi, Guoqiang Zhong

Ocean University of China, Qingdao, China
E-mail: qilin@ouc.edu.cn Tel/Fax: +86-532-66781729

Abstract

Depth data captured by Kinect provides inexpensive geometric information to higher level computer vision tasks such as object detection and recognition. However, there are missing values in the depth map at object boundaries and those beyond the working distance of Kinect due to the limitations of the hardware employed. In this paper, we proposed a self-taught regression method to recover the missing depth data. First a rough estimation of the scene depth was made based on the color image from Kinect. We then trained a random forest using the estimated depth and the intensity from the neighborhood of each pixel that the depth can be captured by Kinect. The random forest was used to predict missing depth data in a self-taught manner that the pixels with largest number of valid neighborhood were predicted first and then added to the training set for the next round prediction. This repeats until all missing data was recovered. The experiment results show that our method outperforms existing approaches to depth recovery.

I. Introduction

Active depth sensors are becoming a popular alternative to passive methods in 3D scene reconstruction due to the accuracy and efficiency. Microsoft Kinect is one of the devices that can capture both depth and color images, where the depth sensor utilizes projection of fixed pattern of infrared light. However, using Kinect to capture scene depth often encounters the problem of missing values due to the limited working distance and the failure of receiving reflected infrared light at object boundaries or infrared-absorbing surfaces. Therefore, complementing the raw depth image becomes a necessary step before further procedure.

Existing methods on depth recovery usually employ image inpainting techniques or modifications of Gaussian blur. These methods achieve plausible results not only on smooth regions, but also on the object edges; the latter often contain more important scene properties.

The color image captured by Kinect contains various useful information, such as color, texture, shading and so on, which can be used to estimate scene depth [1]. However, the depth derived from a single still image is often unreliable and inaccurate, compared with those measured using active depth sensors such as Kinect. Nevertheless, the scene depth estimated from the single color image can heuristically help recovering the missing values in the depth map captured by Kinect.

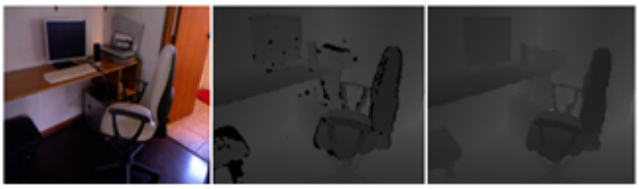

Fig. 1. The left and middle images are an RGB-D pair from Kinect, the right image is the recovery result using our method.

This paper proposes a method for depth recovery that employs neighborhood information and scene depth from still color images. First a rough estimation of the scene depth was made based on the color image using the method from literature. We then trained a random forest using the combined information of the estimated depth, the grayscale intensity of the color image and the depth captured by Kinect. The random forest was used to predict missing depth data in a self-taught manner that the pixels with largest number of valid neighborhood were predicted first and then added to the training set for the next round prediction. The rest of this paper describes the details of our method. Section II reviews related work on depth data recovery. Section III describes the algorithm. The experiment is described in Section IV. We conclude our work in Section V.

II. Related Work

Researchers have proposed methods to recover missing values in the depth image. Yang et al used an adaptive color-guided autoregressive (AR) model for depth recovery [2]. Sergey et al proposed a method of filtering using motional and color information of objects in the video [3]. Park et al proposed an application framework to perform high quality upsampling on depth maps captured from a low-resolution and noisy 3D time-of-flight camera [4]. Andrew et al devised a fast modified two-pass median filter with

dynamic window scales, which is effective in filling small holes, but can not deal with large missing areas [5]. Lai et al filled missing depth values by recursively applying a median filter in the construction of the RGB-D object dataset, but blurring occurs for large occlusions [6]. An adaptive method that responds to the object edges and its directions was proposed in [7]. In [8], a method adaptive to occlusions was proposed. The quality of recovered depth image used by these methods above are not satisfactory around depth discontinuities.

Saxena et al proposed a method that infers detailed 3-d structure from a single image, which is both quantitatively accurate and visually pleasing [9]. Supervised learning was used to learn the relationships between the image features and the location/orientation of the planes, and between various parts of the image. Their method can estimate 3-d locations and orientations of small planar regions in the image. Figure 5 shows the estimation of the scene depth using this method. Liu et al considered the problem of depth estimation from a single monocular image using deep convolutional neural networks (CNN) [1]. We chose Saxena et al's method as an initial depth estimation due to the efficiency.

Previous work used continuity of scene object in the color image to recover missing depth. We further considered the scene depth implied in the color image, and we also used the spatial information. We used machine learning techniques (a self-taught random forest) to regress missing depth from color image and the derived scene depth. The recovered depth map keeps clearly boundaries, and is visually reasonable at the regions that the actual depth is beyond sensor's working distance.

III. Method

We employs a multi-round, high-priority-first strategy to recover the missing values in the depth image. The color image and the depth image captured by Kinect are denoted by the symbols I_k and D_k respectively. The missing values are pixels with 0 intensity in D_k, as shown in Equation 1.

$$\hat{D}_k = \{D_k(i,j) | D_k(i,j) = 0\} \quad (1)$$

We first employed the Make3D algorithm introduced in [9] to estimate a rough scene depth D_M from the color image I_c. Then we trained a random forest using the neighborhood information of I_c, D_M and D_k. The random forest was used to predict missing depth data in a self-taught manner that the pixels with largest number of valid neighborhood were predicted first and then added to the training set for the next round prediction. This repeats until all missing data was filled up. Figure 2 shows the architecture of our system and the following is the details.

A. Regression Forest

Random forests can be used for regression analysis and are in fact called regression forest. They are an ensemble of different regression trees and are used for nonlinear multiple regression. Each leaf contains a distribution for

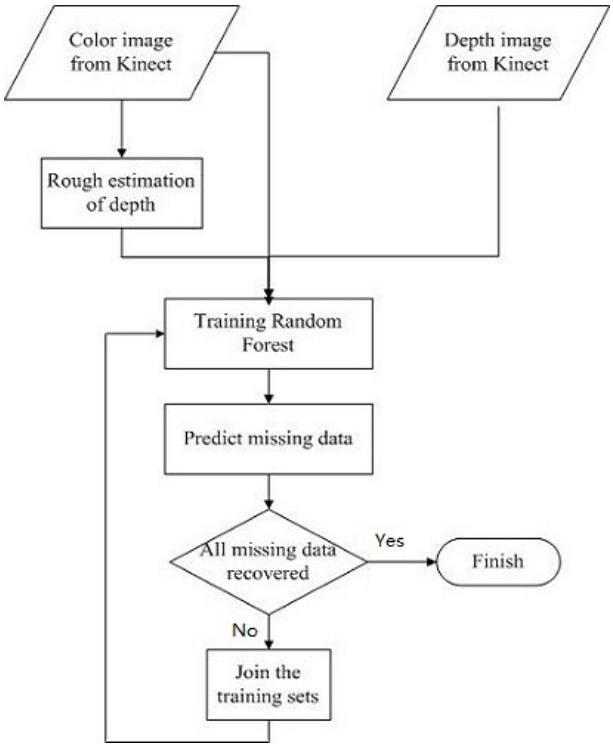

Fig. 2. Illustration of the self-taught depth recovery framework

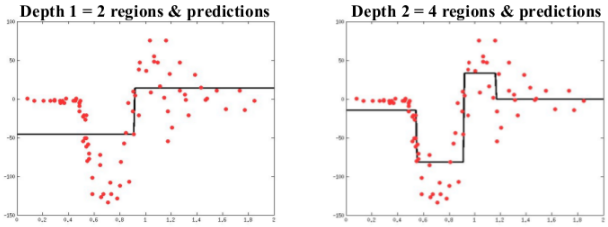

Fig. 3. Results using different depth of random forests

the continuous variables. Other than classification tree, in a regression tree, since the target variable is a real valued number, we fit a regression model to the target variable using each of the independent variables. Then for each independent variable, the data is split at several split points, We use standard sum of squared error (SSE) at each split point between the predicted value and the actual values. The variable resulting in minimum SSE is selected for the node. Then this process is recursively continued till the entire data is covered, Figure 3 and Figure 4 show the middle result and the final result of a regression tree.

B. Rough estimation of the scene depth

The algorithm proposed in [9] can create detailed 3-d models which are both quantitatively accurate and visually pleasing. Other than assuming that the environment is made of a number of small planes, the algorithm does not make any explicit assumptions about the structure of the scene. This allows the model to be generalized well, even to scenes

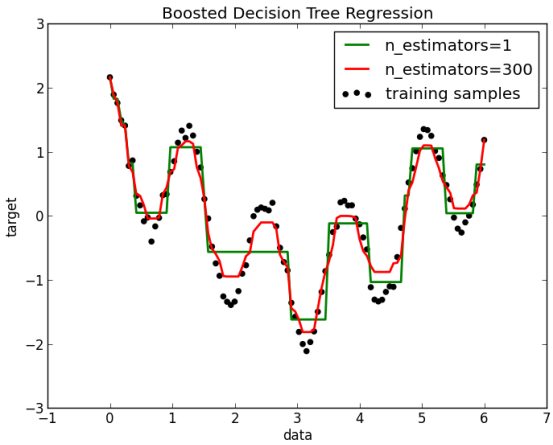

Fig. 4. Result using boosted decision tree for regression problem

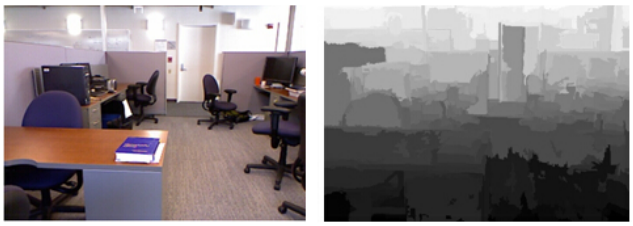

Fig. 5. Predicted scene depth using the method in [9].

with significant non-vertical structures. Thus, we chose this algorithm to generate our rough depth estimation. Figure 5 shows an example of estimated depth using this method. We can see that the resulted scene depth is rough but generally correct, thus we chose this method to generate the rough depth estimation in our approach.

C. The self-taught regression algorithm

The reason why we choose random forest as the regression method is that there are two inherent connections between regression tree and the self-taught algorithm we proposed. For Decision tree learning, there are two principles:

- Sampling with replacement on training example. So, the selection of training example for each decision tree are random, this is tree bagging.
- Sampling with no replacement of features. The selection of a random subset of features is also called feature bagging.

The proposed algorithm use those two ”bagging” concept: inherently:

- training set was chosen by a 'valid' threshold, the pixels above the threshold may be used for several times. This part perform the ”tree bagging”.
- for different iterations, the number of neighborhood was different, that means the number of features for each pixel are different, this part perform the ”feature bagging”

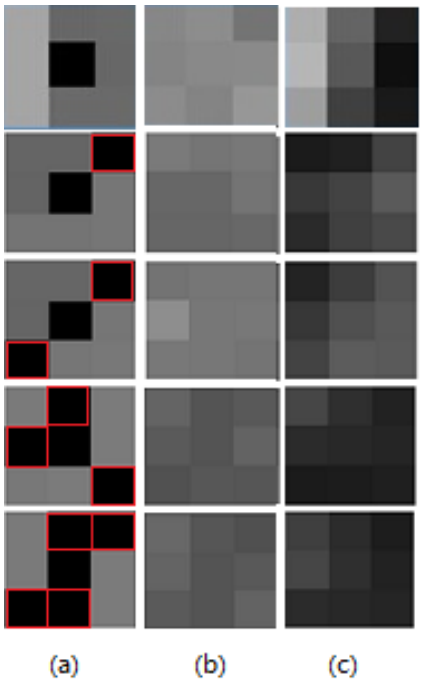

Fig. 6. Examples of 8-neighborhood in D_k, I_k and D_M. (a) Missing values (the black square in the center) in the depth image captured by Kinect. The squares with red bounding box are pixels that the depth data are also missing. Therefore, the numbers of 'valid' neighborhood are 8, 7, 6, 5, 4 for each row respectively. (b) Grayscale intensity of the corresponding pixels in the color image captured by Kinect. (c) Estimated depth of the corresponding pixels from the color image using the method in [9].

The pixels in the image always correspond to different objects in the 3D scene, and this also happens to the missing values in the depth image captured by Kinect. This fact implies that neighboring pixels may share similar properties, such as color and depth. Inspired by this fact, we used the neighborhood information to predict the missing values.

In this paper, we considered the 8-neighborhood of each pixel:

$$\mathcal{N}_{i,j} = \{(x,y)|i-1 \leqslant x \leqslant i+1, j-1 \leqslant y \leqslant j+1, (x,y) \neq (i,j)\} \quad (2)$$

where $\mathcal{N}_{i,j}$ denotes the neighborhood of pixel at image location (i,j).

We labeled each pixel in the depth image D_k in terms of the number of 'valid' neighboring pixels. 'Valid' means that the depth from Kinect is not missing.

$$\mathcal{N^*}_{i,j} = \{(x,y)|(x,y) \in \mathcal{N}_{i,j}, D_k(x,y) \notin \hat{D}_k\} \quad (3)$$

Figure 6 shows missing pixels with different number of 'valid' neighbourhoods.

To form the training set, we picked up those pixels that their own and all their 8-neighborhood are valid:

$$\Omega_{train} = \{\mathcal{N^*}_{i,j}|D_k(i,j) \notin \hat{D}_k, |\mathcal{N^*}_{i,j}| = 8\} \quad (4)$$

For each pixel in Ω_{train}, the gray intensity of the color image I_k and the roughly estimated depth D_M from its 8-neighbourhood were catenated to form the input vector and

the Kinect depth was the output. The training data were fed to a random forest as the regression model.

The trained random forest was not used to predict all missing values in one shot. As we use the neighborhood information and the number of 'valid' neighboring depth varies across missing pixels. The missing pixels with most 'valid' neighbors should be predicted first. Therefore in each round of prediction, we sort the missing pixels in a descending order according to the number of valid neighboring pixels ($|\mathcal{N}^*{}_{i,j}|$). The missing values that the number of their valid neighborhood is equal or greater than 4 were predicted ($|\mathcal{N}^*{}_{i,j}| > 4$). After recovering these pixels, we updated the training set by adding new data that meet the criterion (Equation 4). The random forest was re-trained thereafter and used for the next round prediction. The procedure is repeated until all missing values are recovered. The proposed algorithm resembles the region growing method in image segmentation.

The whole algorithm is described in Algorithm 1.

Algorithm 1 Self-taught complement algorithm.

Require:

Color image captured by Kinect, I_c;
Depth image captured by Kinect, D_k;

Ensure:

1: Rough estimation of scene depth D_M based on I_c;
2: Choose training set Ω_{train};
3: Train Random Forest ;
4: Predict missing data where $|\mathcal{N}^*{}_{i,j}| > 4$;
5: Update training set Ω_{train};
6: GOTO 3;
7: **return** $\hat{D}_k$;

IV. Experiments and Results

In the experiment, we used RGB-D pairs from the Berkeley 3-D Object Dataset (B3DO) to test our method [10]. The dataset consists of color and depth image pairs, gathered in real domestic and office environments. The depth data is missing on most glass surfaces and infrared-absorbing surfaces. The dataset provides recovered depth images by using average filtering or applying a global descriptor to a part of the depth image. Figure 8 shows some examples of the recovered depth maps using their method, from which we can see that the boundaries and some part of the objects are lost.

We tested our method on this dataset. The results are shown in Figure 8. We can see that our method has two advantages. 1. Object details, especially the boundaries, are well preserved. This is because we used spatial information of both color image and depth image. 2. The regions beyond the Kinect's working distance are better recovered. There are often big patches of missing values in these regions(see the last row in Figure 7). The multi-round strategy and the use of rough estimation of scene depth from the color image provide such better result. Figure 9 shows a rendered mesh derived from the depth image recovered using our method.

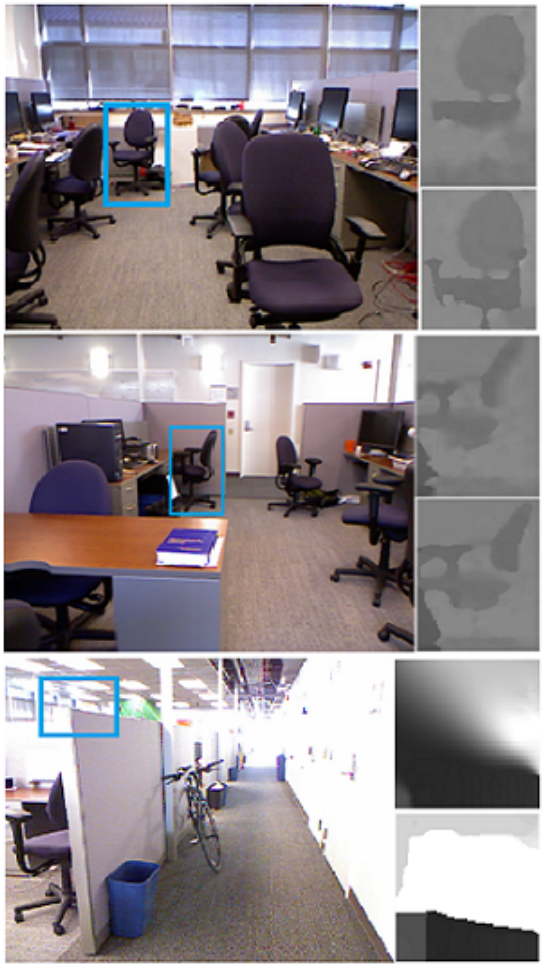

Fig. 7. Zoom-in regions of the recovered depth maps show the advantage of our method. In each example, the top right is result of B3DO method and the bottom right is the result of our method. We can see that the details of objects are better recovered and the sharp edges are also reserved.

V. Conclusion

In this paper, we propose a self-taught regression method to recover missing depth data captured by Kinect. We first estimated a rough scene depth based on the color image from Kinect. A random forest was trained using the estimated depth and the grayscale intensity from the neighborhood of each pixel that the depth can be captured by Kinect. The random forest was then used to predict missing depth in a self-taught manner that the pixels with largest number of valid neighborhood were predicted first and then added to the training set for the next round prediction.

We tested our method on the Berkeley 3-D Object Dataset. The experiment results show that our method can well recover missing depth data meanwhile keeping object boundaries and can predict reasonable depth at regions beyond the sensor working distance, which outperforms existing approaches to depth recovery.

Fig. 8. Recovered depth. (a) Color image from Kinect. (b)Depth image from Kinect. (c) Recovered depth maps by B3DO. (d) Our method.

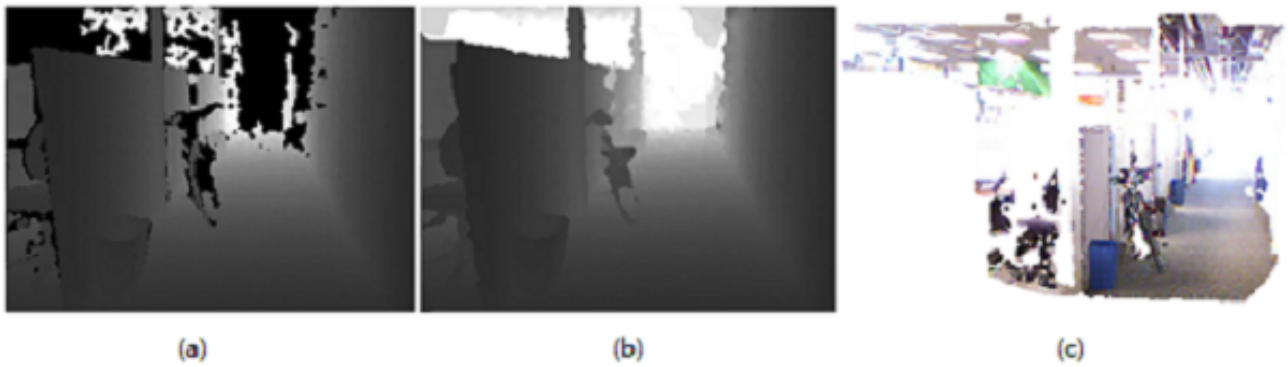

Fig. 9. Depth recovery result. (a) Raw Kinect depth image. (b) Recovered depth image. (c) Rendered 3D scene using (b) as 3D point cloud.

Acknowledgments

This work was supported by China Postdoctoral Science Foundation (No.2014M551963), National Natural Science Foundation of China (NSFC) (No. 61271405, 61401413, 61501417), The PH.D. Program Foundation Of Ministry Of Education of China (No.20120132110018), International Science & Technology Cooperation Program of China (ISTC) (NO. 2014DFA10410) and Shandong Province Science and Technology Research (No.2012GHY11524).

References

[1] F. Liu, C. Shen, and G. Lin, "Deep convolutional neural fields for depth estimation from a single image," *arXiv preprint arXiv:1411.6387*, 2014.

[2] J. Yang, X. Ye, K. Li, C. Hou, and Y. Wang, "Adaptive color-guided and model-assisted depth recovery from rgb-d data," 2014.

[3] S. Matyunin, D. Vatolin, Y. Berdnikov, and M. Smirnov, "Temporal filtering for depth maps generated by kinect depth camera," in *3DTV Conference: The True Vision-Capture, Transmission and Display of 3D Video (3DTV-CON), 2011*, pp. 1–4, IEEE, 2011.

[4] J. Park, H. Kim, Y.-W. Tai, M. S. Brown, and I. Kweon, "High quality depth map upsampling for 3d-tof cameras," in *Computer Vision (ICCV), 2011 IEEE International Conference on*, pp. 1623–1630, IEEE, 2011.

[5] A. Maimone and H. Fuchs, "Encumbrance-free telepresence system with real-time 3d capture and display using commodity depth cameras," in *Mixed and Augmented Reality (ISMAR), 2011 10th IEEE International Symposium on*, pp. 137–146, IEEE, 2011.

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

[7] W.-Y. Chen, Y.-L. Chang, S.-F. Lin, L.-F. Ding, and L.-G. Chen, "Efficient depth image based rendering with edge dependent depth filter and interpolation," in *Multimedia and Expo, 2005. ICME 2005. IEEE International Conference on*, pp. 1314–1317, IEEE, 2005.

[8] S.-B. Lee and Y.-S. Ho, "Discontinuity-adaptive depth map filtering for 3d view generation," in *Proceedings of the 2nd International Conference on Immersive Telecommunications*, p. 8, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2009.

[9] A. Saxena, M. Sun, and A. Y. Ng, "Make3d: Learning 3d scene structure from a single still image," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 31, no. 5, pp. 824–840, 2009.

[10] A. Janoch, S. Karayev, Y. Jia, J. T. Barron, M. Fritz, K. Saenko, and T. Darrell, "A category-level 3d object dataset: Putting the kinect to work," in *Consumer Depth Cameras for Computer Vision*, pp. 141–165, Springer, 2013.