TEMPORAL FILTERING FOR DEPTH MAPS GENERATED BY KINECT DEPTH CAMERA

Sergey Matyunin, Dmitriy Vatolin, Yury Berdnikov

Moscow State University
Graphics & Media Lab
smatyunin@graphics.cs.msu.ru, dmitriy@graphics.cs.msu.ru
yberdnikov@graphics.cs.msu.ru

Maxim Smirnov

YUVsoft Corp.

ms@yuvsoft.com

ABSTRACT

We propose a method of filtering depth maps provided by Kinect depth camera. Filter uses output of the conventional Kinect camera along with the depth sensor to improve the temporal stability of the depth map and fill occlusion areas. To filter input depth map, the algorithm uses the information about motion and color of objects from the video. The proposed method can be applied as a preprocessing stage before using Kinect output data.

Index Terms— Digital filters, Image processing, Image enhancement

(a) (b)

Fig. 1. The data captured by Kinect. a) Frame from the conventional camera. b) Frame from the depth sensor. Black areas on the depth image are marked as occlusions by Kinect.

1. INTRODUCTION

A depth image-based rendering is a very widespread technique. Depth maps are often used for 3D image creation and representation because of convenience for transmission and virtual view synthesis. 3D video is already quite widespread and requires depth maps creation for its production. Microsoft Kinect is one of the devices that allows to capture the depth map for video. Kinect is able to capture simultaneously a conventional video and a depth image. Its depth sensor is based on a projection of fixed pattern of infrared light [1]. An offset infrared camera receives the projected IR light and the built-in controller estimates the depth using the distortion of the pattern. The produced depth map can be used for 3D scene reconstruction, robotic vision and for interaction between user and computer [2].

Nevertheless, using the depth images captured by the depth sensors is often difficult owing to their inherent problems: optical noise, lost depth information on the shiny surfaces and occlusion areas, and flickering artifacts (See Fig. 1). Such a depth map is inapplicable to 3D image creation because of the temporal instability and errors. A specific preprocessing is required to increase the temporal and spatial stability of the results. This paper proposes such a method of the depth maps processing. Color and motion information from the RGB camera of the Kinect is used to increase the quality of the depth map.

2. RELATED WORK

Depth map errors often leads to noticeable artifacts in 3D video and significantly decrease the resultant quality. Errors often occur in the occlusion areas. Several modifications of Gaussian blur were developed for occlusion areas processing. In [3], the authors proposed asymmetric filter, which have larger length in the vertical than in the horizontal direction. They also propose changing the size of the symmetric smoothing filter depending on the local values in the depth maps.

Depth errors are often visible at the edges of the objects. An adaptive method that responds to the object edges and its directions was proposed in [4]. In [5], a method adaptive to occlusions was proposed.

The above-listed methods only use a portion of the color information from every frame of the source video (for example, only information about object edges).

In [6], the authors proposed a method of minimizing depth flickering for depth images captured by time-of-flight sensors. This method improves depth map stability only for stationary objects because it considers only the presence of the motion rather than the length of the motion vectors.

978-1-61284-162-5/11/$26.00 © 2011 IEEE

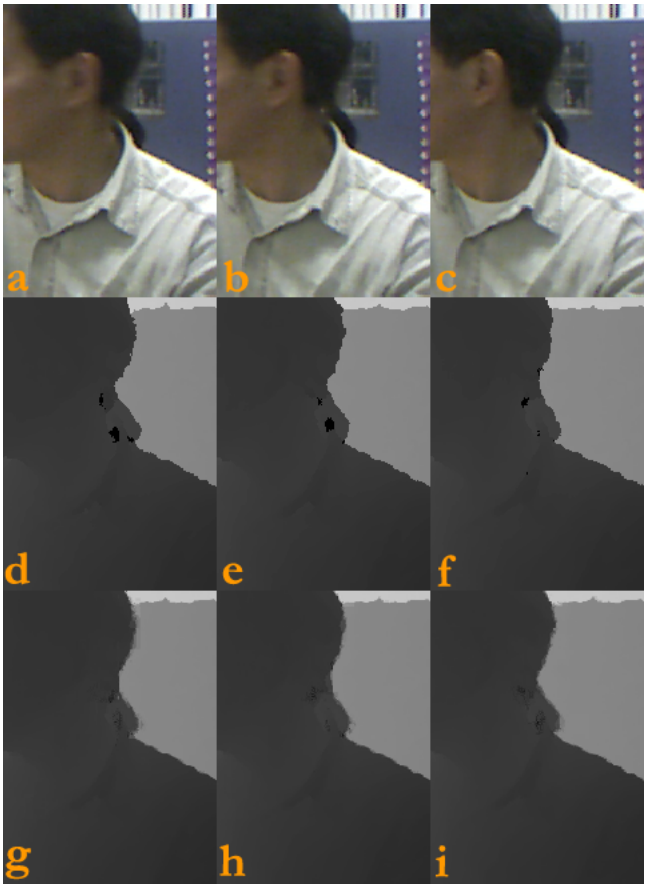

Fig. 2. Segments of three consecutive frames for the test sequence. (a)-(c) Original sequence from the RGB camera. (d)-(f) Original depth map from the depth camera. Black areas are occlusion. (g)-(i) Processed depth map. Occlusions were filled.

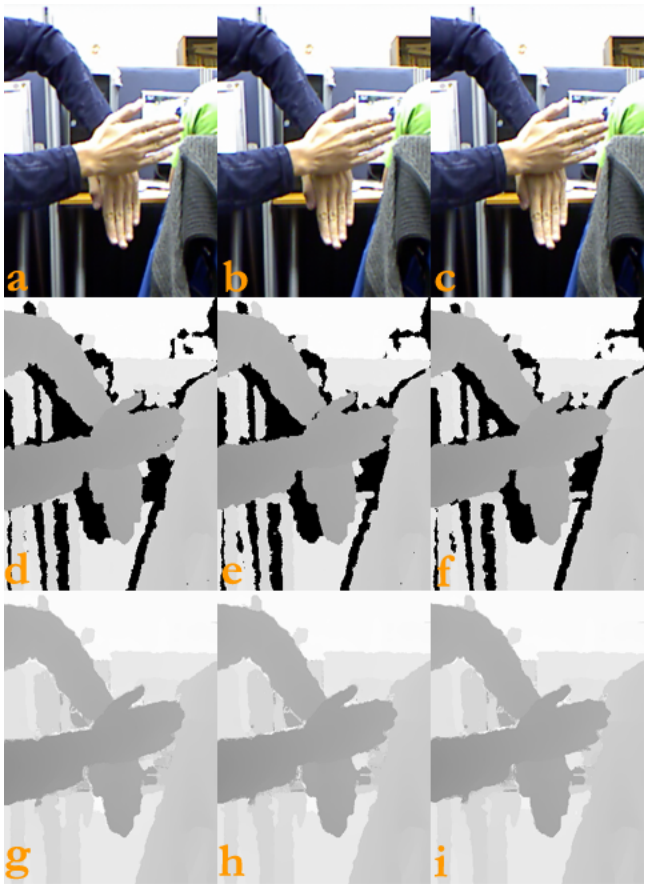

Fig. 3. Segments of three consecutive frames for the test sequence. (a)-(c) Original sequence from the RGB camera. (d)-(f) Original depth map from the depth camera. Black areas are occlusion. (g)-(i) Processed depth map. After filtering depth map become more temporally stable.

Depth maps quality can be improved using stereo RGB cameras instead of the single camera. A method of data fusion from the depth and the stereo cameras was proposed in [7]. A maximum a posterior Markov Random Fields approach was used.

Approaches based on the energy minimization problem and graph cut [8, 9] produce good results, but owing to computational complexity, they require a long time to process the entire video.

The proposed approach uses the information about motion and color of the objects from several consecutive frames to refine input depth map. The algorithm takes information about object motion into account using motion compensation.

3. PROPOSED METHOD

The proposed algorithm uses frames of the conventional video from the RGB camera and the corresponding depth maps provided by the depth sensor. We denote $I_i(x, y)$ as the intensity (or color) of pixel (x, y) in frame i. $I_i(x, y)$ is either a three-vector in case of a color image or a scalar for a grayscale. n denotes the current frame number, and $D_i(x, y)$ represents the depth for the ith frame in position (x, y). $B_i(x, y)$ denotes the map of occlusions for the ith frame:

$$B_i(x,y) = \begin{cases} 1, \text{ if the pixel } (x,y) \in \text{occlusion area,} \\ 0, \text{ otherwise.} \end{cases}$$

The proposed method consists of five steps:

1. Motion estimation between the current frame I_n and neighboring frames $I_{n-m}, \ldots, I_{n-1}, I_{n+1}, \ldots, I_{n+m}$, where $m > 0$ is a parameter. The result of this stage is a field of motion vectors $MV_i(x, y) = (u_i(x, y), v_i(x, y))$. We define $MV_n(x, y) \equiv 0$.

2. Computation of the confidence metric $C_i(x, y)$ for the resultant motion vectors $MV_i(x, y)$. Here, $C_i(x, y) \in$

$[0, 1]$. $C_i(x, y)$ quantifies the estimation quality for motion vector $MV_i(x, y)$. The metric is similar to that described in [10].

3. Motion compensation for the depth map and source frames. Here, D_i^{MC} denotes the motion-compensated depth maps, and I_i^{MC} denotes the motion-compensated source frames. Both the I_i^{MC} and D_i^{MC} images are computed using motion vectors MV_i, which are estimated from the source video sequence:

$$I_i^{MC}(x, y) = I_i(x + u_i(x, y), y + v_i(x, y)),$$
$$D_i^{MC}(x, y) = D_i(x + u_i(x, y), y + v_i(x, y)).$$

4. Depth map filtering using the computed D_i^{MC}, C_i and I_i^{MC} values.

5. Occlusion areas are filled using filtered neighbor pixels.

3.1. Depth Filtering

We use $2m + 1$ consecutive frames for filtering. To eliminate sharp discontinuities in the time domain we apply temporal median filtering at the first step of processing:

$$D_n^{med}(x, y) = \underset{\substack{i=n-m,\dots,n+m;\\ C_i(x,y)>Th_C;\\ |I^{MC}(x,y)-I(x,y)|<Th_{SAD};\\ B_i(x,y)=0}}{median} D_i^{MC}(x, y).$$

The median is calculated over the pixels from the current and neighboring depth maps which have sufficiently small interframe difference $|I^{MC}(x, y)-I(x, y)|$, well estimated motion vectors (confidence measure of vector is high), and do not belong to occlusion areas. Thresholds Th_C and Th_{SAD} depend on the noise level of the source video. The thresholds have an influence on the resultant quality of processing and must be selected carefully. In the current version of the filter we use the same thresholds for all frame areas. We intend to make it adaptive to the local contrast level in the source video frame.

The next processing step is temporal smoothing.

$$D_n^{smooth}(x, y) = \frac{1}{k(x, y)} \cdot \sum_{t=n-m}^{n+m} \sum_{(x',y')\in\sigma(x,y)\setminus B_n} \omega(t, x, y, x', y') D_t^{input}(x', y'),$$
$$k(x, y) = \sum_{t=n-m}^{n+m} \sum_{(x',y')\in\sigma(x,y)\setminus B_n} \omega(t, x, y, x', y'). \quad (1)$$

$\omega(t, x, y, x', y')$ is a weight function, and D_t^{input} is the input depth for this step.

We tested two configurations for D_t^{input}:

1. $D_n^{input} = D_n^{med}$, $D_{n\pm p}^{input} = D_{n\pm p}$,

2. $D_n^{input} = D_n^{med}$, $D_{n-p}^{input} = D_{n-p}^{smooth}$, $D_{n+p}^{input} = D_{n+p}$,

where $p \in [1, m]$. The latter approach yields a smooth resulting depth map, but it is less accurate for small details. $\sigma(x, y)$ denotes spatial neighborhood of pixel (x, y). Size of $\sigma(x, y)$ must be chosen as a tradeoff between computation speed and processing quality.

Weighting function ω is given by

$$\omega(t, x, y, x', y') = f(t, x', y') \cdot C_t(x, y) \cdot g(|x - x'|, |y - y'|),$$

where function $f(t, x', y')$ denotes the dependence on interframe difference $|I_t^{MC}(x', y') - I_n(x', y')|$; $C_t(x, y)$ is the confidence of motion compensation of pixel (x, y) on frame t; g denotes the dependence of the weight function on the spatial distance between (x, y) and (x', y'). In the simplest case g is identically constant. To achieve better quality we tested other types of dependencies between the spatial distance and weight: linear, polynomial, exponential. Function f is given by the formula

$$f(x) = max\left(0, min\left(1, \sum_{i=0}^{3} \mu_i \cdot \left(\frac{x}{\nu}\right)^i\right)\right),$$

where μ_i and ν are parameters of the algorithm.

Thus, the algorithm averages the depth in the neighborhood of each pixel using the information about interframe difference for the source video, the confidence metric for motion vectors, spatial proximity, and occlusion areas.

3.2. Occlusion areas filling

Kinect depth camera produces the depth map with large occlusion areas. It is necessary to fill these areas to obtain a depth map suitable for further using. We use depth values of neighbor pixels with similar color to estimate depth of occlusion pixel. Only pixels with frequent depth values in neighborhood are taken into account. Let (x_o, y_o) be an occlusion pixel ($B_n(x_o, y_o) = 1$). Then the resultant depth for occlusion areas is filled in the following way:

$$D_n^{result}(x_o, y_o) = \underset{(x,y)\in\sigma_o(x_o,y_o)}{median} \Big\{ D_n^{smooth}(x, y) | \sum_{(x',y')\in\sigma_o(x_o,y_o)} \delta(D_n^{smooth}(x, y), D_n^{smooth}(x', y'))) > Th_{occl}, |I_n(x_o, y_o) - I_n(x, y))| < Th_{SAD}, B_n(x, y) = 0 \Big\},$$

where $\sigma_0(x_o, y_o)$ denotes the neighborhood of pixel (x_o, y_o), the symbol δ is Kronecker's delta. Threshold Th_{occl} and size of $\sigma_0(x_o, y_o)$ must be determined adaptively to keep the number of relevant pixels (x,y) large enough for robust median filtering results.

4. RESULTS

The results were obtained using a block matching motion-estimation algorithm based on the algorithm described in [11]. We used macroblocks of size 16×16, 8×8 and 4×4 with adaptive partitioning criteria. Motion estimation is performed with quarter-pixel precision. Both luminance and chroma planes are considered.

The proposed algorithm was implemented in C as a console application. The source video and its depth map is the input data for the algorithm and the filtered depth is output. Our algorithm uses one-pass processing and it can be conveniently implemented in hardware. The algorithm's performance is 1.4 fps on 640×480 video resolution on PC with Intel Celeron 1.8 GHz CPU.

Fig. 2 shows the results of the algorithm. To illustrate temporal features of depth map we give RGB image, original and processed depth map for three consecutive frames from the test sequence. Original depth map (Fig. 2, (d)-(f)) is unstable in the temporal domain an has large occlusion areas (black regions), especially near the objects borders. Proposed method allows to make it more stable, fixes errors and makes depth map smoother (See Fig. 2, (g)-(i)). Filtering results for another test sequence are presented in Fig. 3.

5. FURTHER WORK

In the proposed approach we utilize the information about occlusions areas produced by the Kinect depth camera. We intend to use the detection and the processing of occlusions caused by the motion of the objects. In occlusion areas, motion estimation algorithm produces incorrect motion vectors, thus artifacts occurs. Confidence metric in these areas must be reduced adaptively to improve processing quality. Further improvement of motion estimation stage is critically important. Using a good optical flow algorithm instead of the block-based motion estimation reduces blocking artifacts and improves the resultant quality of the depth map.

6. CONCLUSIONS

We described a method of depth map filtering for Kinect. The proposed algorithm improves the visual quality of the depth maps and the rendered 3D images. Using the proposed method can significantly extend the range of the applications of the Kinect depth map in computer vision.

7. ACKNOWLEDGEMENTS

This research was partially supported by grant number 10-01-00697-a from the Russian Foundation for Basic Research.

8. REFERENCES

[1] "The PrimeSensor Reference Design 1.08", `http://www.primesense.com`.

[2] Andrew Wilson, "Using a depth camera as a touch sensor", in Proc. of *International Conference on Interactive Tabletops and Surfaces*, 2010.

[3] W. J. Tam and L. Zhang, "Non-uniform smoothing of depth maps before image-based rendering", in *Proceedings of Three-Dimensional TV, Video and Display III (ITCOM'04)*, vol. 5599, pp. 173–183, 2004

[4] Wan-Yu Chen, Yu-Lin Chang, Shyh-Feng Lin, Li-Fu Ding, and Liang-Gee Chen, "Efficient depth image based rendering with edge dependent depth filter and interpolation," in Proc. of *IEEE International Conference on Multimedia and Expo*, pp. 1314–1317, 2005.

[5] Sang-Beom Lee and Yo-Sung Ho, "Discontinuity-adaptive depth map filtering for 3d view generation", in Proc. of the *2nd International Conference on Immersive Telecommunications (IMMERSCOM), Brussels, Belgium*, pp. 1–6, 2009

[6] Sung-Yeol Kim, Ji-Ho Cho, Andreas Koschan, and Mongi A. Abidi, "Spatial and temporal enhancement of depth images captured by a time-of-flight depth sensor", in Proc. of the *IEEE International Conference on Pattern Recognition (ICPR)*, pp. 2358–2361, 2010

[7] Jiejie Zhu, Liang Wang, Ruigang Yang, and James. E. Davis, "Fusion of time-of-flight depth and stereo for high accuracy depth maps", in Proc. of the *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008)*, 2008

[8] Guofeng Zhang, Jiaya Jia, Tien-Tsin Wong, and Hujun Bao, "Consistent depth maps recovery from a video sequence", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 6, pp. 974–988, 2009

[9] Guofeng Zhang, Jiaya Jia, Tien-Tsin Wong, and Hujun Bao, "Recovering consistent video depth maps via bundle optimization", in Proc. of the *IEEE Conference on Computer Vision and Pattern Recognition*, 2008

[10] K. Simonyan, S. Grishin, and D . Vatolin, "Confidence measure for block-based motion vector field", in Proc. of *GraphiCon*, pp. 110–113, 2008

[11] Karen Simonyan, Sergey Grishin, Dmitriy Vatolin, and Dmitriy Popov, "Fast video super-resolution via classification", in Proc. of *International Conference on Image Processing*, pp. 349–352, 2008