REAL-TIME VOLUMETRIC 3D CAPTURE OF ROOM-SIZED SCENES FOR TELEPRESENCE

Andrew Maimone[*] *Henry Fuchs*[†]

Department of Computer Science, University of North Carolina at Chapel Hill

Figure 1. Left: Color coded contribution of the five Kinect sensors used in test system. Two center images: Two views of a live capture session. All surfaces displayed are captured dynamically. Right: Raytracing of 3D Volume Segmentation. Pink indicates background, blue indicates foreground, cyan indicates scene motion, and black represents unknown (no data).

ABSTRACT

This paper describes a 3D acquisition system capable of simultaneously capturing an entire room-sized volume with an array of commodity depth cameras and rendering it from a novel viewpoint in real-time. The system extends an existing large-scale capture system with a volumetric measurement integration and rendering engine that has been adapted for use with a fixed array of cameras. New techniques are explored to improve image quality, decrease noise, and reduce the number of required depth cameras while preserving support for a fully dynamic scene.

Index Terms — teleconferencing, sensor fusion, filtering

1. INTRODUCTION

A long-standing goal of telepresence has been to join remote spaces through a shared virtual wall, allowing remote collaborators to see each other's environments as extensions of their own. A key component of this vision is the real-time acquisition of a large-scale dynamic scene in 3D so that it can be rendered from the unique perspective of the viewer.

The recent arrival of the Microsoft Kinect, an inexpensive and high performance depth and color image sensor, has accelerated progress in this area – but limitations remain. The Kinect-based FreeCam [1] system demonstrated high quality real-time 3D acquisition but capture was limited to users segmented from the background. KinectFusion [2] presented very high quality room sized 3D scanning, but instantaneous update was limited to parts of the scene within the field of view of a single camera. High quality offline scanning has also been demonstrated for large scale scenes [3] and humans [4]. Past work by the authors [5, 6] presented large-scale 3D acquisition of all objects in large volume simultaneously, but image quality was lacking. In this paper, we present a 3D acquisition system that offers fully dynamic and simultaneous capture of all objects in a room-sized scene, but offers significantly improved image quality.

[*]e-mail: maimone@cs.unc.edu

[†]email: fuchs@cs.unc.edu

2. BACKGROUND AND CONTRIBUTIONS

Previous work by the authors demonstrated cubicle-sized [6] and room-sized [5] 3D capture by filtering, color matching, and forming a triangle mesh from the data of a fixed array of Kinects and merging the result in image space. Although this approach offered high performance, the lack of a geometric data merger between units resulted in mediocre image quality, and the absence of any temporal coherence (beyond a simple temporal threshold in [6]) resulted in severe temporal noise. These works also did not retain data between frames and therefore required additional cameras to provide a view of temporarily occluded background objects.

As a result, we look to KinectFusion [2], which solved these problems by fusing and holding depth measurements in a voxel grid as a weighted average over time. Although KinectFusion was designed for a single moving camera, we note that at first order the approach is straightforward to adapt to a fixed array of cameras: the measurements from each fixed camera can be fused into a voxel grid at each frame using precalibrated positions.

However, we observe that when using the KinectFusion algorithm in this unintended way (see center image of Figure 4), the results are poor – the fusion of a few measurements of each surface point from fixed cameras does not significantly reduce depth noise as does the hundreds of measurements acquired from a moving camera over time. Additionally, KinectFusion's method for instantaneous response to change in a dynamic scene (segmenting and placing geometry in a more responsive overlaid volume) is not applicable to a large capture volume with fixed cameras as the measurement uncertainty typically exceeds the movement of a near-motionless standing participant – a scenario commonly found in a telepresence setting. We also note the need for blending and color-matching the textures over the camera array.

In this work, we integrate our exisiting multi-Kinect capture calibration and filtering framework [6] with the KinectFusion measurement integration and rendering engine, while enhancing the latter to support dynamic scenes with fixed camera arrays. Our specific contributions are as follows:

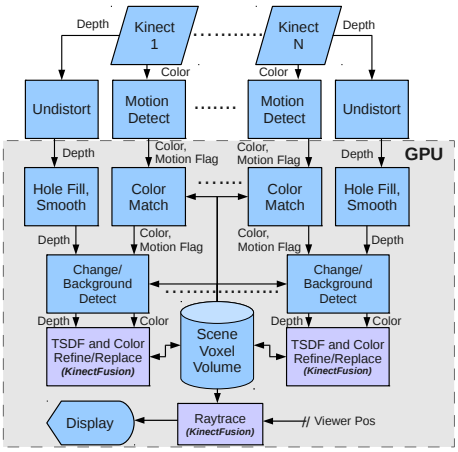

Figure 2. Data processing, integration, and rendering pipeline.

1. Adaptation of KinectFusion to fixed camera arrays
2. A technique for supporting dynamic scenes with a single volume by observing both color and depth changes
3. A technique for learning a volumetric scene background model that enables data retention and noise suppression
4. A technique for color matching cameras in a voxel volume
5. A hybrid voxel coloring and texture projection technique to provide color for novel view renderings

3. SYSTEM OVERVIEW

Hardware Configuration and Layout For the system presented in this paper, five Kinect cameras were used to capture a volume of approximately $4.25m \times 2.5m \times 2.5m$. The contribution of each Kinect is illustrated in the leftmost image of Figure 1. The Kinects were connected to a PC with an Intel Core i7-960 CPU and an NVIDIA GeForce GTX 580 GPU for real-time scene processing and novel view rendering. In this work we focus primarily on the acquisition component of a 3D telepresence system; see our previous work for information regarding integration with a head-tracked autostereo display [6] or large display wall [5].

Software Overview The OpenGL shader language was used for depth map preprocessing and the excellent CUDA-based KinFu module provided in the PCL library[1] was used as the base KinectFusion implementation. The OpenCV library was used for camera calibration and 2D image operations. The overall data processing, integration and rendering pipeline is illustrated in Figure 2 and individual components are described in Section 4.

4. IMPLEMENTATION

4.1. Calibration and Preprocessing

Calibration The Kinects were calibrated and corrected for radial distortion and depth biases using the method described in our previous system [6], except that extrinsic calibration between cameras was performed simultaneously (rather than pairwise to a master camera) and bundle adjustment was performed using the Simple Sparse Bundle Adjustment (SSBA)[2] software. These alterations allowed for more accurate extrinsic calibration necessary to accurately fuse geometry between cameras.

[1] http://pointclouds.org/

[2] http://www.inf.ethz.ch/personal/chzach/opensource.html

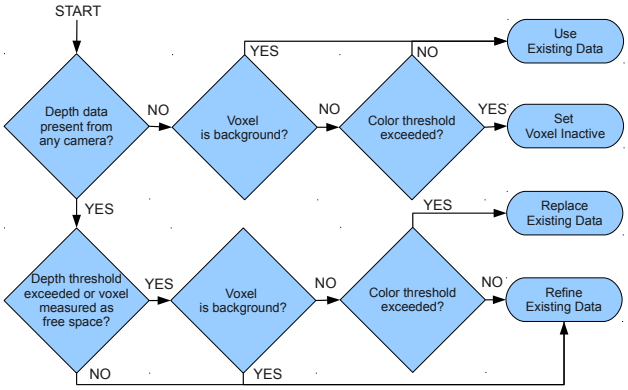

Figure 3. Decision process for voxel change detection and data retention.

Depth Preprocessing As noted in Section 2, integrating the measurements of a few fixed Kinect sensors does not provide sufficient geometric noise reduction, even if accumulation is performed over a long period of time. Additionally, holes are present in the depth data due to multi-Kinect interference. Therefore, we apply the GPU-based hole filling and smoothing operators described in our previous work [6], while adding color distance information to the bilateral filter to improve edge quality.

Color Preprocessing The scene change detection algorithm described in Section 4.2 relies in part on color change information. To detect color change we first find the absolute difference between the current and last color frame from each Kinect and apply a morphological erosion to remove isolated color noise. A morphological dilation is then applied to expand the suspected motion region to fill in gaps that occur within textureless regions. A threshold is then applied to remove values beneath the noise floor, and the result is stored as a per-pixel binary change map in the color image's alpha channel.

4.2. Volumetric Integration and Rendering

As noted previously, our measurement integration and rendering engine is based on KinectFusion [2], a fast GPU-optimized version of the Truncated Signed Distance Function (TSDF) volumetic integration method. We extend this method to fixed multi-camera configurations responsive to dynamic scenes by applying the modifications listed in this section.

Background Model As noted in Section 2, KinectFusion provides a method for instantaneous response to changes in a dynamic scene – segmenting moving objects by depth into a more responsive foreground volume that is overlaid onto the slowly refined background volume. However, this approach is not practical for our fixed camera arrangement as there is too much sensor noise to detect subtle motions, such as that of a standing person. Instead, we opt for a single volume augmented with a foreground-background indicator implemented as a per-voxel temporal counter. At each frame, the counter is incremented if the voxel is not measured as free space by any of the Kinects and is otherwise reset to zero. When the counter reaches a maximum value, the voxel is considered background. The rightmost image of Figure 1 shows a segmentation of a participant standing in a 3D scene.

Note that this background model is 3D and therefore allows multiple layers of background to exist simultaneously. For example, a rear wall learned as background may be subsequently occluded by a lamp, which is then also learned as background.

When the lamp is removed, the occluded area of the wall will still be considered background without a relearning period as it was never measured as free space. As shown in the rightmost image of Figure 1, a background object (the coffee table) may also occlude a foreground object (the participant).

Scene Change Detection To represent a dynamic scene, it is necessary to determine if the discrepancy between newly acquired data and previously accumulated data represents a scene change or can be attributed to noise. As a starting point, we determine if the difference between a new depth measurement and the corresponding value in the accumulated volume exceeds the expected noise of a Kinect sensor at the measurement distance. However, this test often results in false positives in areas with excessive noise (such as object edges), areas where multi-Kinect interference is more pronounced, and areas where there is above average calibration error. To reduce these false positives, we eliminate regions corresponding to background and those where no color change was reported using the method described in Section 4.1. This decision process is performed per-voxel.

Data Retention With each frame, new measurements may be unavailable for parts of the scene that were observed in the past – the result of occlusion, sensor noise, or multi-sensor interference. To present a complete scene using fewer cameras and to reduce temporal noise, it is desirable to retain data; however, one must be careful not to hold transient measurements in a dynamic scene.

We make the assumption that background voxels (those that have consistently not represented free space) and those that do not represent a color change beyond the noise threshold of the color camera do not represent a material scene change and are retained when no new sensor measurements are available. Note that the data stored in unretained voxels is not erased, but rather flagged as inactive so that data may be later refined if future measurements are close to those previously stored.

TSDF Integration We modify the standard update rules of KinectFusion [2] as follows:

1. The Kinect data is aligned with the voxel grid using a fixed precalibrated position, rather than a tracked position.
2. Measurements weights and the size of the truncated SDF region behind the surface is set on a per-Kinect basis using an estimate of depth error at the given surface distance.
3. The criteria illustrated in Figure 3 is used to decide whether the TSDF should be updated as a weighted average or replaced with the most recent value.

Raytracing and Color We utilize the raytracing approach of KinectFusion to render the volume from a novel view, except that we skip voxels that do not meet the previously described data retention criteria. To color the scene according to the RGB images acquired from the Kinect cameras, we evaluated two options – projective texture mapping and voxel coloring.

In projective texture mapping, the current color image acquired by the Kinect is projected into the scene during the raytracing step. This approach allows the full resolution of the color camera to be used and requires no extra memory, but does not provide temporal noise reduction or allow color data to be retained over time. In our multi-Kinect implemention, we also note the need to project surface voxels onto the image plane of each color camera, resolve occlusions, and perform blending.

In voxel coloring, an option in the utilized PCL KinectFusion implementation, colors are stored as a weighted average in each voxel. This approach allows a reduction of color noise and data to be retained between frames, but color data is stored at volume resolution – which is typically much lower than the available color resolution. In our implementation, we also note the need to discard accumulated color data when scene changes are detected.

As a compromise, we propose a third hybrid option: the use of projective texture mapping with a fallback to voxel coloring for data that is to be retained across frames. We found that this approach made projective texturing practical for use in our multi-camera setup when rendering from novel views.

We also note the need to match colors between Kinect cameras as the units utilize automatic color control. As a solution, we adapt our previous color matching approach [6] from triangle based rendering to volume rendering. We maintain a per-Kinect color bias that is computed as the average difference between each voxel's color (which itself is a weighted average between cameras) and the color contribution from each Kinect. For stability, this bias is calculated as a rolling average over time.

5. RESULTS

Rendering Quality Figure 4 shows a comparison between various rendering engines. In the leftmost image (corresponding to our previous rendering engine [6]), note the misalignment of some surfaces (ex: green poster), noisy surfaces (ex: front right floor tiles), missing color data due to occlusion (shadow behind user), and mismatched colors (coffee table). In the center image (a naive application of the core KinectFusion [2] algorithm to a fixed multi-Kinect setup), note the extreme noisiness resulting from a lack of depth prefiltering, color mismatches between cameras, and missing data (the moving arms have disappeared due to the slow weighted average update). In the right image (the rendering engine described in this work), these problems have been resolved, but we note a roughness of edges and an overall geometric quality inferior to those presented in KinectFusion [2] with a moving camera.

The left column of Figure 6 shows a comparison of temporal noise (measured as the difference between two frames of a captured static scene) using our previous system [6] and the proposed system using both projective texture mapping and voxel coloring. We note a significant decrease in geometric noise among object edges. With voxel coloring, we also note a significant decrease in color noise over the entire rendering. Quantitatively, the PSNR between frames increased 4.0 dB to 32.08 dB with projective texture mapping, and increased 13.8 dB to 41.81 dB with voxel coloring.

Figure 5 compares texture quality for two coloring methods. Projective texture mapping at a low volume resolution offers a visible improvement over voxel coloring even at a high volume resolution, although temporal noise is higher.

Support for Dynamic Scenery The center and right images of Figure 6 show a moving person captured with our proposed system, which does not exhibit the long geometric motion trails associated with simple weighted average volumetic integration. However, we have observed small temporal inconsistencies at object edges during fast motion sequences that are the result of unsynchronized sensors. These effects could be mitigated by performing temporal interpolation between depth maps to match a synchronization marker placed in the scene or by applying a small amount of non-rigid registration across sensor data, as in [4]. Color synchronization could be handled by making small texture adjustments using the technique of [7].

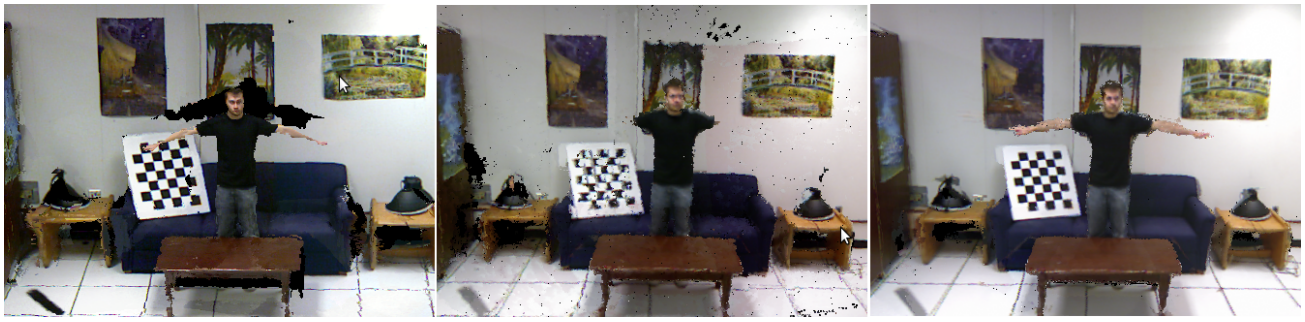

Figure 4. Rendering quality. Left: rendering engine of our previous work [6]. Center: rendering using naive application of KinectFusion [2] to a fixed array of Kinects. Right: rendering engine described in this work (using voxel coloring).

Figure 5. Texture quality. Left to right: 1) voxel coloring at $384 \times 268 \times 268$ and 2) $875 \times 612 \times 612$ volume resolution, 3) projective texture mapping at $384 \times 268 \times 268$ and 4) $875 \times 612 \times 612$ volume resolution.

We have also observed that a small amount of color motion blur may occur in some scenarios, as well as a short geometric transition period when objects move from background to foreground (ex: when moving a table).

Performance For the five Kinect system described in the paper, combined volume data integration (at $384 \times 268 \times 268$ voxel resolution) and novel view rendering (at 640×480 pixel resolution) occurred at an average rate of 8.5 Hz, while re-rendering an unchanged volume from a novel view occurred at a rate of 26.3 Hz.

6. CONCLUSIONS AND FUTURE WORK

We have presented a real-time 3D capture system that offers improved image quality and significantly reduced temporal noise over existing real-time systems capable of capturing entire room-sized scenes simultaneously. We are excited at prospect of future systems offering these capabilities with photorealistic, video-like image quality. As a first step towards this goal, we plan to improve the edge quality of moving objects by striking a better balance between scene accumulation and change response and by resolving temporal inconsistencies between unsynchronized sensors.

7. ACKNOWLEDGMENTS

This work was supported in part by the National Science Foundation (award CNS-0751187) and by the BeingThere Centre, a collaboration of UNC Chapel Hill, ETH Zurich, NTU Singapore, and the Media Development Authority of Singapore.

8. REFERENCES

[1] C. Kuster, T. Popa, C. Zach, C. Gotsman, and M. Gross, "Freecam: A hybrid camera system for interactive free-viewpoint video," in *Proceedings of Vision, Modeling, and Visualization (VMV)*, 2011.

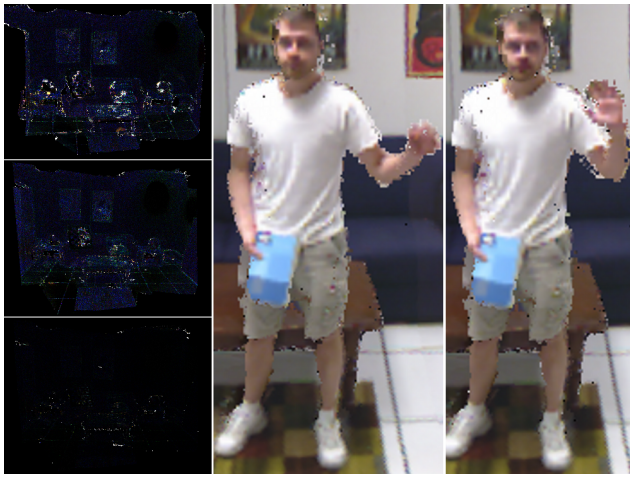

Figure 6. Left column: Noise between frames (amplified $5\times$). Rows top to bottom: 1) rendering of [6] (PNSR = 28.01 dB), 2) rendering of this work with projective texture mapping (PSNR = 32.08), and 3) voxel coloring (PSNR = 41.81 dB). Center and right column: two frames of a capture session showing support for motion.

[2] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, and A. Fitzgibbon, "Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera," in *Proceedings of ACM UIST*, 2011.

[3] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox, "Rgb-d mapping: Using kinect-style depth cameras for dense 3d modeling of indoor environments," *I. J. Robotic Res.*, vol. 31, no. 5, pp. 647–663, April 2012.

[4] J. Tong, J. Zhou, L. Liu, Z. Pan, and H. Yan, "Scanning 3d full human bodies using kinects," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 18, no. 4, april 2012.

[5] A. Maimone and H. Fuchs, "A first look at a telepresence system with room-sized real-time 3d capture and large tracked display," in *Artificial Reality and Telexistence (ICAT), The 21st International Conference on*, nov 2011.

[6] A. Maimone, J. Bidwell, K. Peng, and H. Fuchs, "Enhanced personal autostereoscopic telepresence system using commodity depth cameras," *Computers & Graphics*, vol. 36, no. 7, pp. 791 – 807, 2012.

[7] M. Eisemann, B. De Decker, M. Magnor, P. Bekaert, E. de Aguiar, N. Ahmed, C. Theobalt, and A. Sellent, "Floating textures," *Computer Graphics Forum (Proc. of Eurographics)*, vol. 27, no. 2, pp. 409–418, Apr. 2008.