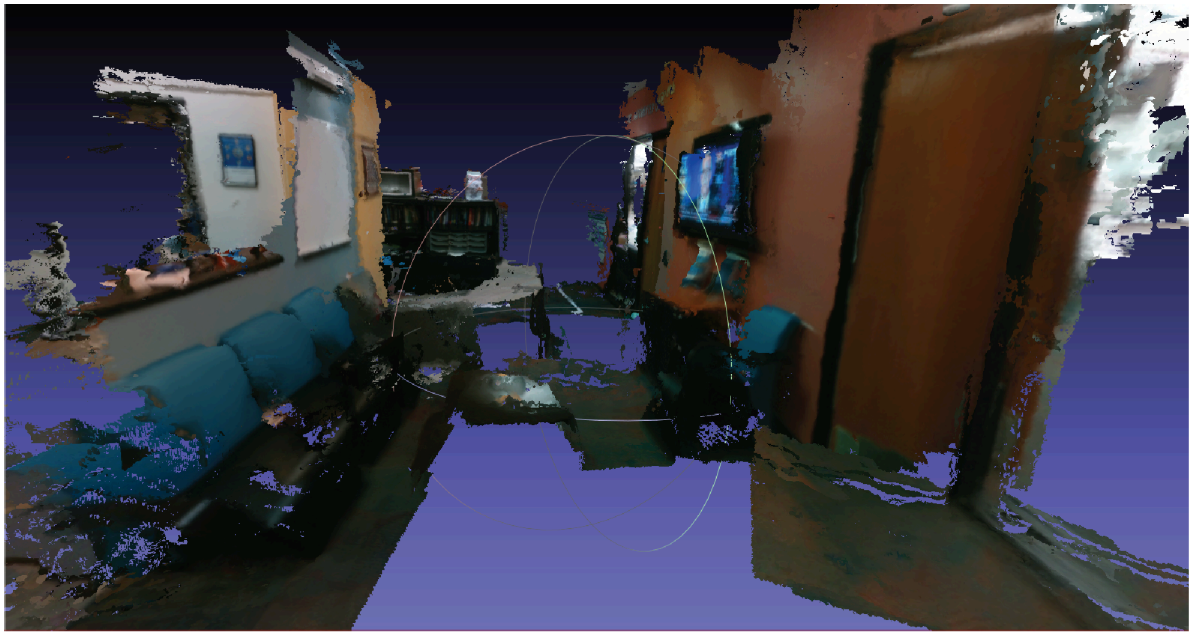
OpenARK Real-Time Dense 3D Reconstruction

Adam Chang *adamchang2000@berkeley.edu*
Kevin Ubilla *omar.ubilla@berkeley.edu*



Real-Time Reconstruction of FHL Vive Center for Enhanced Reality @ UC Berkeley with Realsense D435i

***Abstract* - 3D reconstruction has in recent years become a large topic of interest, especially in the fields of virtual and augmented reality. For augmented reality applications, the transition from traditional offline reconstruction methods to real-time reconstruction is crucial. Our 3D reconstruction module is one feature of many contained within OpenARK, an open-source augmented reality SDK aiming to provide AR functionality to developers. With only a single RGB-D camera, we are able to capture high resolution, large-scale scenes and visualize them in real-time. More importantly, our solution runs entirely on the CPU, an important property enabling OpenARK to be as hardware agnostic as possible.**

1. Introduction

With the growing availability of inexpensive depth cameras, more developers are beginning to be able to enter the augmented reality (AR) domain. Open Augmented Reality Kit (OpenARK) aims to empower these developers with a powerful, open-source SDK which contains the fundamental building blocks for an AR application. Alongside hand tracking and avatar tracking, a real-time reconstruction system opens a world of interesting possibilities. For instance, a real-time reconstruction system would allow applications to merge real world geometry with a generated model to provide all sorts of interaction to the user. Another important application of real-time 3D reconstruction is to provide the user an indication of the completeness of their reconstruction. Especially for large scenes, it is difficult to collect data for every face of each object. With a real-time visualization of the reconstruction, the user is now able to see which areas they may need to record data for. Our system enables real-time reconstruction alongside simultaneous recording of the pose-estimated RGB and depth data for slower but high resolution offline reconstruction.

Our real-time 3D reconstruction system builds off of previous years of work in OpenARK. The major

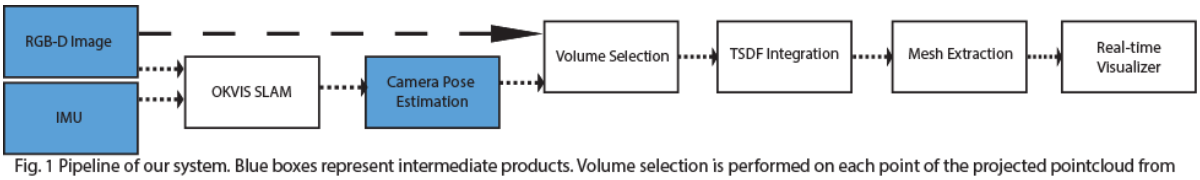


Fig. 1 Pipeline of our system. Blue boxes represent intermediate products. Volume selection is performed on each point of the projected pointcloud from a pose-estimated image.

improvements we have made are 1) Increase in scalability 2) A more robust SLAM system 3) A more effective visualization of the current mesh model 4) No longer requiring GPU acceleration.

2. Related Work

The KinectFusion project began the venture into real-time reconstruction with consumer-grade depth cameras[3]. Using the Kinect camera, the system performed real-time camera tracking and mapping and pioneered the use of the truncated signed distance function (TSDF) introduced by Curless and Levoy as a dense voxel space representation[1]. The signed distance function is a space representation that provides efficient and error-correcting integration of multiple depth images into a single reconstruction volume. It involves assigning values to voxel coordinates based on whether it is inside or outside a volume. This can be calculated through a combination of many different depth images observing the same area. The system offered real-time (30Hz) performance but suffered from several severe limitations. First, it required GPU parallelization to reach real-time performance. Second, it was tightly bound by memory limitations, only working well with scenes $\leq 7m^3$ in volume.

Other works in the following years have aimed to address the limitations of KinectFusion, specifically the memory limitation which needed to be solved in order to enable large-scale real-time reconstruction. Improvements have been found from changes in several different directions. Zeng et al. created an octree-based SDF system which relied on a hierarchical octree that greatly diminished the amount of SDF which needed to be stored for a typical scene reconstruction [6]. However, their implementation relied heavily on graphics hardware parallelization, which was not ideal for the goals of OpenARK.

Whelan et al. demonstrated the technique of using a moving window to allow reconstruction of a large scene by only addressing a small portion of the scene at a time [5]. By streaming portions of the reconstruction in and out of the current window, it could still reconstruct scenes with a dense voxel grid at high resolution. Our implementation of a segmented TSDF is very similar to Whelan's work, but simplified to allow faster real-time reconstruction performance.

3. Pipeline

Fig. 1 depicts a general overview of our reconstruction system pipeline. The sensor we use for all of our data recording and testing is the Intel Realsense D435i. It includes a global shutter RGB camera, stereo depth obtained from two IR cameras, active IR projection, and an inertial measurement unit (IMU). The RGB and depth frame are both 640x480 in resolution and the depth frame is aligned to the RGB frame upon retrieval from the camera. Both the RGB and depth frame outputs are 30Hz. Fig. 2 shows the different inputs retrieved from the camera. The depth image is filtered by a close and far threshold before entering the system to eliminate noisy measurements (Intel Realsense D435i reports a $\leq$ 2% error at 2m distance).

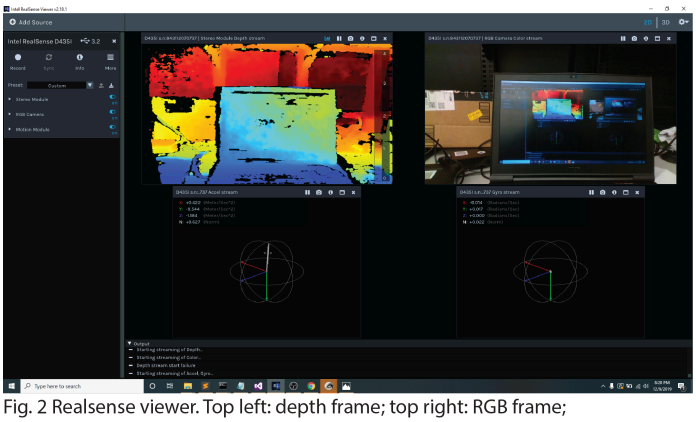


Fig. 2 Realsense viewer. Top left: depth frame; top right: RGB frame; bottom left: acceleration; bottom right: gyro

The IMU and image frames are then fed into the SLAM (Simultaneous Localisation and Mapping) system. The SLAM system we currently use is built on OKVIS, open keyframe-based visual-inertial SLAM, a SLAM system which demonstrates much more robust

real-time camera tracking by combining visual-based tracking with IMU data [4]. Loop closure is one important aspect of the SLAM system. It refers to the technique of recognizing previously observed features and adjusting the camera's trajectory to realign the path of the camera, completing the loop, hence the name loop closure. Although our SLAM system is capable of keyframe-based loop-closure detection and map realignment, we currently disable it as it causes misalignments in the real-time reconstruction. Further work needs to be done to allow loop-closure realignments to deform the current model accordingly.

The camera pose estimation is then used to project the depth frame into a point cloud which is then iterated through to integrate each measurement point into the correct TSDF volume. This process is where our algorithm finds the most improvement over the original KinectFusion real-time reconstruction implementation. Instead of using a uniform dense voxel grid, an array holding the TSDF values at each 3D position in space, we have many smaller overlapping TSDF volumes which are queried by a hashing function. This process is run entirely on the CPU and enables much higher memory efficiency over a standard dense voxel grid. This implementation is powered by Open3D's integration library [7]. Fig. 3 depicts the process of segmenting and mapping each TSDF volume to a portion of reconstruction space.

The triangle mesh is extracted from the TSDF volume(s) using an implementation of the marching cubes algorithm, a well-studied and well-known surface reconstruction algorithm involving generating surface triangles using the vertex values of each voxel in the TSDF representation [2].

Finally, the triangle mesh is shown to the user in real-time via OpenGL. The virtual camera follows the trajectory of the real camera, therefore giving the user only a visualization of the immediate reconstructed scene, the immediate area being the most relevant to the user. Fig. 4 shows the GUI setup of our reconstruction module, with a visualization of the mesh model on the left and the current camera frame on the right. We also enable the user to disable frame integration, allowing the user to navigate through their mesh model without worrying about integrating new data. By continuing to track the camera's trajectory, the user can move through the virtual space with ease.

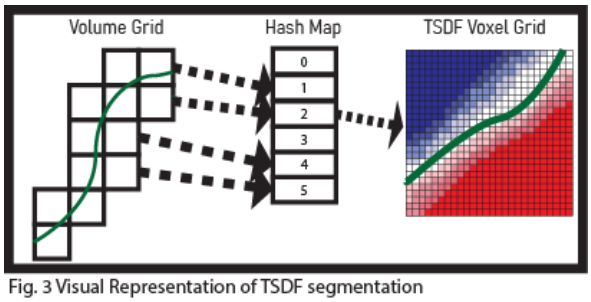


Fig. 3 Visual Representation of TSDF segmentation

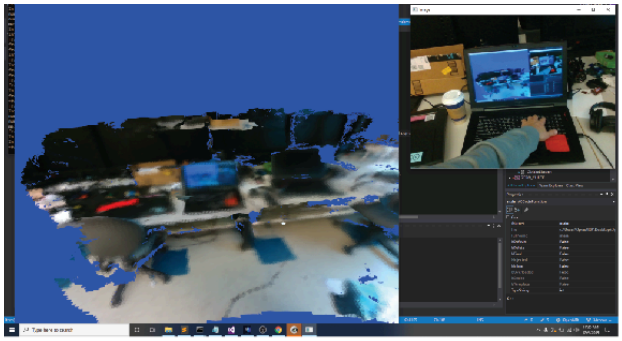


Fig. 4 Left: Real-time visualization of reconstructed model; Right: Current camera frame

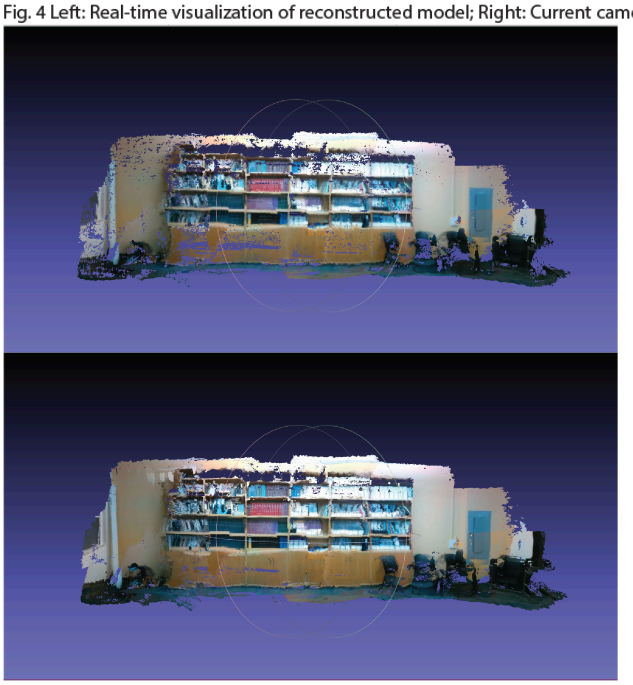


Fig 5. Top: online reconstruction of Trust conference room bookshelf at 2.5 cm resolution; Bottom: offline reconstruction of Trust conference room bookshelf at 1cm resolution.

The entire system runs in real-time, with the camera output at 30Hz, the saving of data for offline reconstruction and the integration of frames at 10Hz, and the extraction of a new triangle mesh at 1Hz. Although the mesh model only updates once a second, we find that with virtual camera tracking occurring with each frame, the user experience is not negatively affected by a slightly delayed mesh update.

4. Results

Our real-time reconstruction system is capable of reconstruction fairly large scenes ($200m^2$) at a resolution of 2cm voxels. Fig. 5 and 6 depict two scene reconstructions, each with an online version and and offline version. Because our system records the RGB and depth frames as well as the pose estimate output of the SLAM system, we are able to perform offline reconstruction at higher resolutions after data capture has been completed. In both reconstructions, the offline reconstruction has a 1cm voxel resolution.

Due to the current limitations of our system, we do not have loop closure detection enabled on the SLAM system. As a result, large scene reconstruction is affected by accumulated SLAM drift due to noise and other minor error. Fig. 7 depicts an example of such SLAM drift.

After following a path around the circumference of the room and returning to the starting location, the trajectory has misaligned by several degrees. With some further work, we will be able to include loop closure into our SLAM algorithm and real-time reconstruction system, decreasing the occurrence of SLAM drifts.

It should be noted that the performance of the reconstruction system relies heavily on the technique of the user, at least with the Intel Realsense D435i. Without moving slowly and deliberately, the user can cause SLAM drifts as well as noisy depth measurements. The optimal technique requires the user to move slowly and only observe portions of the scene at a reasonable distance. Looking across a large area may cause a multitude of noisy measurements. This seems to be primarily a limitation of the camera module we are using.

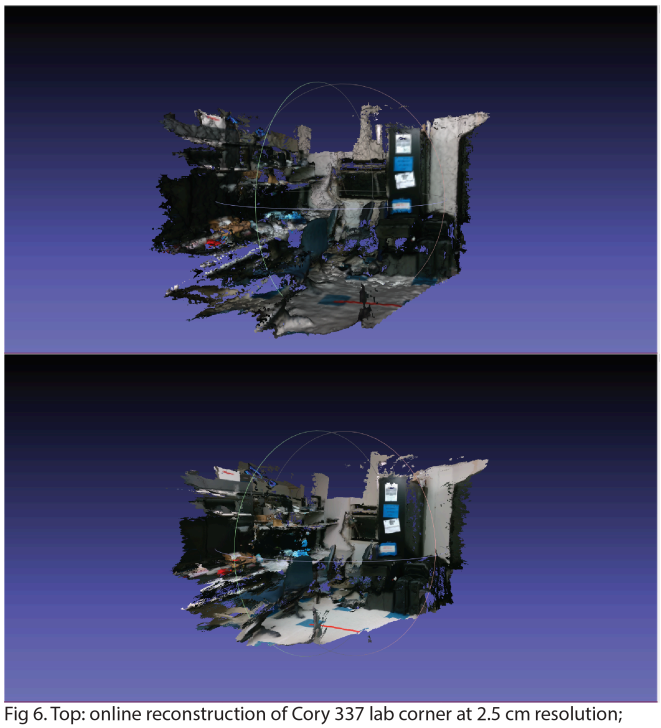


Fig 6. Top: online reconstruction of Cory 337 lab corner at 2.5 cm resolution; Bottom: offline reconstruction of Cory 337 lab corner at 0.8cm resolution.

Fig 7. One-pass reconstruction of Cory 337 lab. Notice the non-parallel walls of the originally rectangular rooms due to SLAM drift.

5. Discussion

Although there are offline reconstruction algorithms which can produce much higher quality reconstructions, they require heavy amounts of postprocessing. Our reconstruction module aims to address the needs of OpenARK as an augmented reality SDK. It provides a user-friendly real-time visualization of a 3D reconstructed scene and exposes the source code for further development.

Additionally, by running entirely on the CPU, performance is affected and tradeoffs need to be made between scalability, resolution, and speed. We conducted our scene reconstructions on a reasonably powerful machine, and the lower boundary on the processor power needed to run the algorithm in real-time still needs to be established.

Improvements in our reconstruction module can also be found in improving the sensor. We currently run the system on an Intel Realsense D435i, a cost-effective but not necessarily the highest quality consumer-grade depth camera. Our modular code structure allows the developer to incorporate any camera with an RGB output, a depth output, and an IMU.

6. Further Work

Much work can be put into improving the current reconstruction module. Our first priority is enabling loop-closure mesh readjustment. This would greatly improve the quality of our reconstructions, especially on large scenes. One possible implementation of loop-closure enabled reconstruction would require segmenting the mesh model into different segments that could individually be transformed based on loop-closure realignments.

Additional improvements can be made to our GUI. It currently only allows a small window into the reconstruction to be observed at any one time. An additional bird's eye view of the model may serve to provide a more complete status of the reconstructed scene.

Due to the limitations of the consumer-grade cameras' depth accuracy, noisy depth measurements result in inaccurate points distributed throughout the scene. Filtering of the depth image removes a lot of these noisy measurements, but further algorithmic improvements need to be made to address these measurements.

7. Conclusion

Using only a single RGB and depth camera, our system is capable of creating a dense real-time reconstruction of large scenes. It can be used as a general reconstruction system as well as a guideline for the user to know which areas in the scene need to be scanned. By recording the camera feed and SLAM pose estimates, we enable slower and high resolution offline reconstructions with the output of our system.

The OpenARK real-time reconstruction module provides developers with the ability to perform real-time high resolution reconstruction of large scenes and visualize them in real-time in an easy-to-use manner. By running only on the CPU, applications built on the OpenARK SDK are able to run on a range of devices without a dedicated graphics card. This module aims to empower a wide variety of augmented reality software development.

References

[1] CURLESS, B. AND LEVOY, M. A volumetric method for building complex models from range images. In ACM Transactions on Graphics (SIGGRAPH), 1996.

[2] LORENSEN, W., AND CLINE, H. 1987. Marching cubes: A high resolution 3D surface construction algorithm. Computer Graphics 21, 4, 163–169.

[3] NEWCOMBE, R. A., IZADI, S., HILLIGES, O., MOLYNEAUX, D., KIM, D., DAVISON, A. J., KOHLI, P., SHOTTON, J., HODGES, S., AND FITZGIBBON, A. 2011. KinectFusion: Real-time dense surface mapping and tracking. In IEEE International Symposium on Mixed and Augmented Reality (ISMAR).

[4] STEFAN LEUTENEGGER, SIMON LYNEN, MICHAEL BOSSE, ROLAND SIEGWART and PAUL TIMOTHY FURGALE. Keyframe-based visual–inertial odometry using nonlinear optimization. The International Journal of Robotics Research, 2015.

[5] WHELAN, T., JOHANNSSON, H., KAESS, M., LEONARD, J., AND MCDONALD, J. 2013. Robust real-time visual odometry for dense RGB-D mapping. In IEEE International Conference on Robotics and Automation (ICRA).

[6] ZENG, M., ZHAO, F., ZHENG, J., AND LIU, X. 2013. Octreebased fusion for realtime 3D reconstruction. Graphical Models 75, 3.

[7] ZHOU, Q.-Y., AND KOLTUN, V. 2013. Dense scene reconstruction with points of interest. ACM Transactions on Graphics (TOG) 32, 4, 112.