Volumetric Reconstruction of 3D Scene from 2D Images

Ashwin Ravishankar
George Mason University
aravisha@gmu.edu

Bansari Desai
George Mason University
bdesai2@gmu.edu

Abstract

A sophisticated workflow model is demonstrated in this project to perform volumetric reconstruction of a scene from short sub-sequences of input RGB-D image sequence. The steps involved in the line process are: Fragmentation, Registration, Refining and Integration. Open 3D library provides a python package to implement this pipeline. Experimental reconstruction was performed on several datasets including Active Vision Dataset, Redwood Dataset, Lidar Dataset.

1. Introduction

High-quality reconstruction of complete indoor scenes is known to be a particularly challenging problem. The availability of low-cost depth sensors in consumer markets enables us to capture reliable depth maps and provides opportunity to develop robust reconstruction systems. While we can easily create 3D models of real-world objects, we cannot achieve same level of quality and reliability with scene reconstruction, since a large scene must be reconstructed from captured images along a certain complex trajectory that only expose small part of environment. Also, even though these images are captured in close range along the path, there is some drift in odometry, and hence need to match and register these views globally.

In this work, we aim to reconstruct complete indoor scenes for Active Vision dataset [2], Redwood dataset [1], Lidar dataset [6] and more. Active Vision dataset is captured by moving camera along a certain path and taking pictures in 360 degrees, which is inconsistent with the requirement of the proposed pipeline, yet, an attempt is made. Existing approach to create volumetric scene for this dataset is to build structure from motion and then decompose those input images into set of image clusters of manageable size. View camera image and orientation of one of the scenes from the dataset is shown in Figure 1. Then a dense point cloud scene is built using PMVS software. While other datasets sample images from some positions or along a few paths through environment, but with Active Vision there is added advantage that we can simulate robot motion in our environment. In our approach of building volumetric scene for Active Vision dataset, we want to explore by calculating odometry drift and use ICP registration and global registration to register the views.

1.1. Purpose

By reconstructing a 3D scene for Active Vision dataset, the advantages are two-fold: ability to achieve higher success in achieving scene reconstruction through local and global registration techniques and with the added advantage of Active Vision dataset, it can also help us in identifying features/object not just in particular image/fragment, but also features that are spread through multiple images or fragments. This use can be extended to other datasets as well.

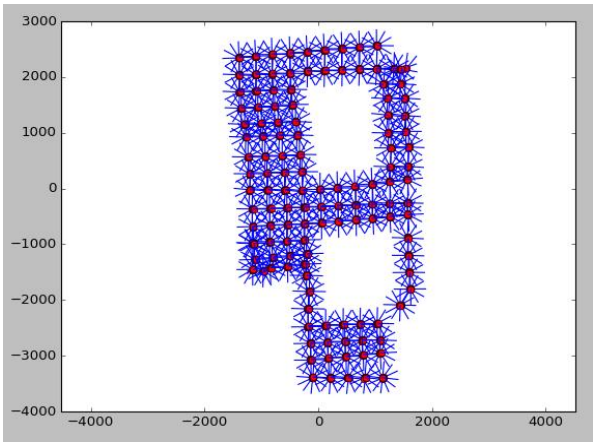

Figure 1: Visualize camera positions and directions of Active Vision Dataset [9]

2. Approach

This process is modelled into a 4-step pipeline – image fragmentation, fragment registration, registration refinement, fragment integration.

2.1. Fragment Construction

To extract more reliable information, we partition our input dataset (RGBD images) into k-frame segment (k=100) as individual range images, are noisy and incomplete. Then build a pose graph for each fragment. As

there is an odometry drift involved, if we try to localize fragment using the transformations, we will get mis-aligned fragments. For this reason, if the images are not neighbors, we first estimate the pose by using OpenCV ORB library to match sparse features over wide baseline images, and then perform 5-point RANSAC to estimate rough alignment, otherwise, we will use identity matrix as initialization parameter in `compute_rgbd_odometry` function provided by Open3D [4][5] to compute transformation from source image to target image. We use this function, as it also implements constraints for geometry in addition to photo consistency.

```
[success, transformation, info] =
compute_rgbd_odometry(
source_rgbd_image, target_rgbd_image,
camera_intrinsic, odo_init,
RGBDOdometryJacobianFromHybridTerm(),
option)
```

Once a pose graph is created, we perform multiway registration. Each graph node represents an RGBD image and its pose, which transforms the geometry to the global fragment space. We use RGBD integration is used to reconstruct a colored fragment from each RGBD sequence. As reference coordinates are unavailable, first image in the fragment segment is taken as reference in the global space and other images are built on top of it. After stitching the range images, we create a surface mesh for each segment. These fragments formed have noise removed significantly from data by taking key frames (5 for our datasets) and thus yield more reliable normal information without suffering from significant odometry drift. Images in Figure 2 show a sample of the fragments created for redwood dataset.

2.2. Fragment Registration

We have to align the fragments in a global space once all fragments are created for the scene. But before performing fragment registration, we down sample our fragment point cloud to make it sparser and regularly distributed, and thus reduce noise to achieve higher reliability. For this, we perform voxel down sampling [10] by down sampling the fragment point cloud, estimate normals, then compute a FPFH (Fast Point Feature Histograms) [11] feature for each point.

If sufficient overlap between fragments are identified, that is if the fragments are neighbors, we perform local registration using algorithm like Iterative Closest Point provided by Open3D library to compute registration. Otherwise, if they are not neighbors, we use RANSAC – Random Sampling Consensus, to perform global registration that uses edge length to prune false matches early and thus perform pairwise registration.

After the poses have been estimated, we build pose graph for multiway registration of all fragments in global scene space which is analogous to what we did for images when creating fragments.

2.3. Refine Fragment Registration

As an extension of the fragment registration step, we perform a refining step to get tighter alignment between all fragments before we perform multiway registration, again. Some of the fragment pairs which are registered as neighbors, may not be so in reality. To deal with such false positives, a pruning algorithm is introduced here. This optimization detects false positives found in pairwise registration, by means of local refinement and creating a dense surface registration objective by a line process over the loop closure constraints. Rigid configuration of the scene and constraint validation is estimated by least-square algorithm. This process helps tighten the loop closure and stitches the fragments together, tighter. We refine odometry and loop closure transformations using ICP algorithm provided by Open 3D library, post which, pose graph estimations of the nodes and edges are recorded for the volumetric integration of the global mesh frame.

2.4. Integrate Fragments

The final step in the pipeline is to integrate all the fragments together into single TSDF volume to build the scene and extract a mesh. The pose graph for all the fragments transformation that build the scene, as recorded at the refinement stage, are read. All the images that contribute to that fragment are integrated into the volume by applying transformation as observed in the corresponding fragment's pose graph. The first image is considered to be the global reference frame on which the model is built. The generated scene is then stored as a ply file and can be viewed using MeshLab or be visualized as a point cloud using the Open 3D's visualization module.

3. Experimental Results

With our above pipeline model, we reconstructed two active vision datasets (Home_003 and Home_014).

We also wanted to experiment creating our own dataset through depth camera using iPhone 8 plus, but the portrait images captured from camera would not store the depth data. The AVFoundation library provided by iOS enables capture and recording of video that had depth information, but extraction of images and depth data from the video was not easily available and was thus inconclusive. Hence, we explored for other RGB-D datasets that to experiment above registration and reconstruction approach mentioned in our work.

The camera intrinsic and extrinsic parameters is pivotal to build point cloud fragments. The camera intrinsic parameters were available for datasets like Redwood and Sun3D and Lidar captured via PrimeSense camera, while for other datasets, we tried to estimate the intrinsic matrix.

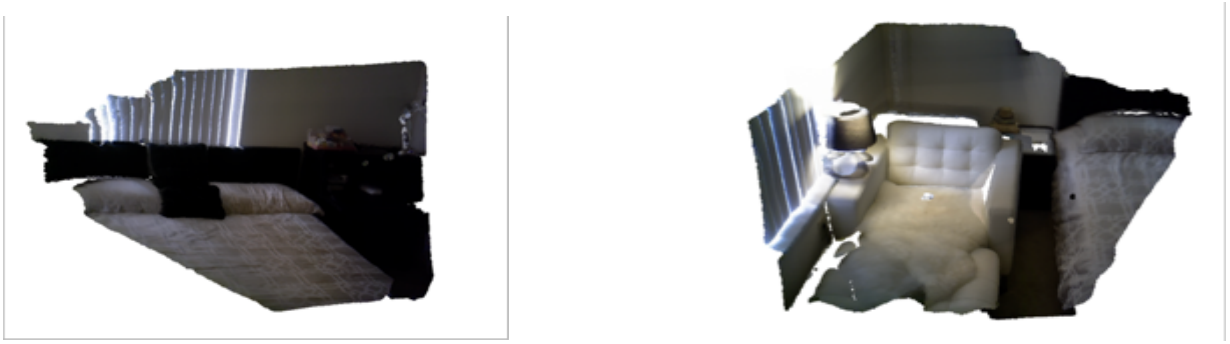

Figure 2a: Fragments created for Redwood dataset.

Figure 2b: Fragments created for Active Vision dataset.

As a result, we can see for Active Vision Datasets (captured via Kinect), the order is distorted, but some objects can be easily recognized.

Figure 3, below, shows scene reconstruction in point cloud for different datasets as visualized from open3D's visualization library.

4. Related Work

Much effort has gone into building devices that capture the visual world to provide digital formats. Trends in state-of-the-art devices like Microsoft Kinect, PrimeSense [8] etc., cameras help create visual datasets by capturing the real world to produce RGB images with the registered corresponding depth images that can be later processed to extract information.

The KinectFusion system provides a real time dense reconstruction model with consumer depth camera demonstrated a pipeline that includes range image integration [3], visual odometry. But this model does not detect loop closures and are restrictive to simple walk-along trajectories or compact office spaces, thus building a relatively disconnected model.

Several reconstruction systems developed over time handle loop closures for RGB-D images [7], but with an underlying assumption that different images that visualize the same scene are significantly similar to one another. Though this approach utilizes visual features like SIFT or SURF keypoints for dense registration, the assumption made over compensates for loss in the real time performance, losing loop closures that are not a part of matching images.

The solution implemented in this project is a line process. This approach supports robust outlier rejection by means of implementing least-squares algorithm. The one step optimization aligns the visualized scene and identifies outliers / false positive registration and prunes it. The library employed – Open3D was originally designed to formulate dense reconstruction from range video, which has been used to build a 3D model from short subsequent images of the scene. Open 3D internally calls several modules of OpenCV that provides a platform for computer vision functionalities.

5. Summary

We have revisited the scene reconstruction from 2D RGB-D images. Using the presented approach, we built scene reconstruction from 2D images with much high fidelity. To sum up, the key idea was to use geometric registration along with global registration for non-neighboring fragments and perform optimization to prune false positives. Although this approach did not take into

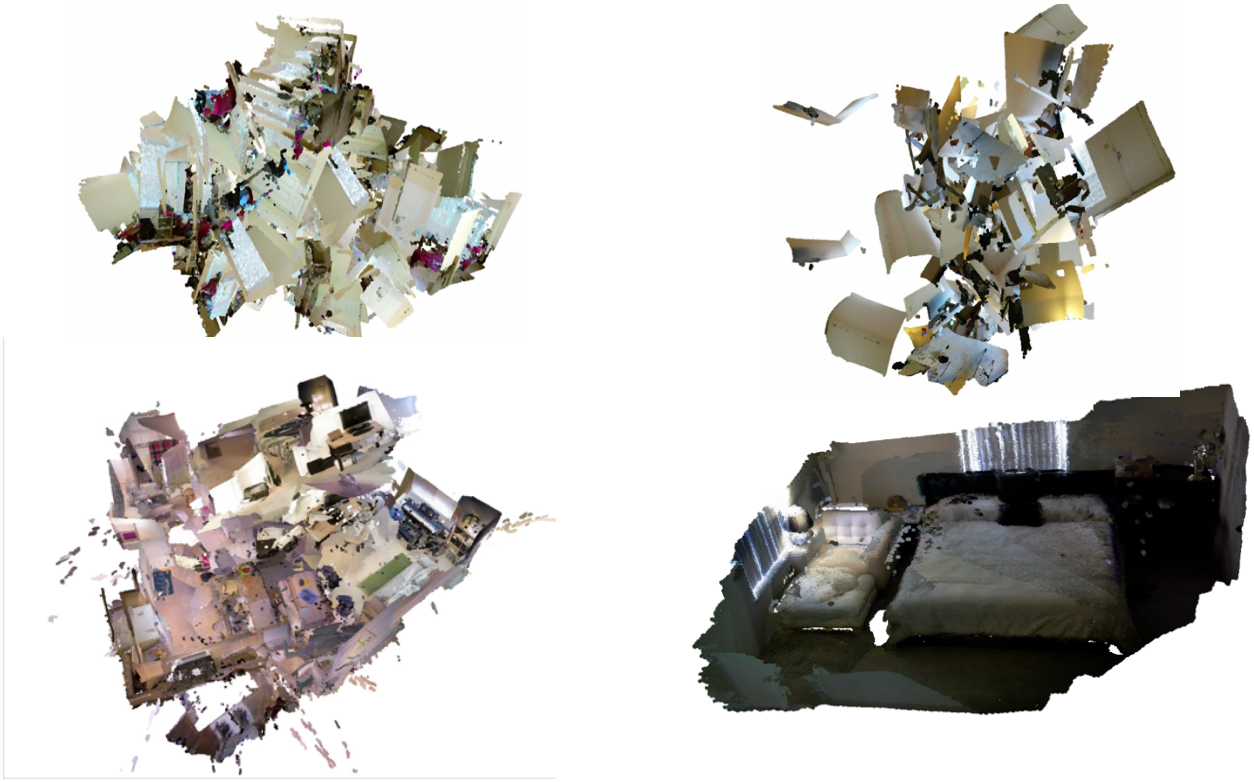

Figure 3: Reconstructed scene from Active Vision dataset (top row – Home_014 left, Home_003 right), Lidar Indoor scene dataset (bottom left) and redwood dataset (bottom right).

account sudden odometry drift that would result in misshaped/missing fragments, slight odometry drift is computed with much higher accuracy.

Due to the orientation with which Active Vision Dataset was captured, large amounts of both, within fragment and inter fragment mis alignments were observed that resulted in a clouded reconstructed scene.

References

[1] Sungjoon Choi and Qian-Yi Zhou and Vladlen Koltun. Robust Reconstruction of Indoor Scenes, 2015, IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

[2] Ammirato, Phil and Poirson, Patrick and Park, Eunbyung and Kosecka, Jana and Berg, Alexander C. A Dataset for Developing and Benchmarking Active Vision, 2017, IEEE International Conference on Robotics and Automation (ICRA).

[3] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon. KinectFusion: Real-time dense surface mapping and tracking, 2011, ISMR

[4] Open 3D library – 0.4.0. http://www.open3d.org

[5] Qian-Yi Zhou and Jaesik Park and Vladlen Koltun. Open3D: A Modern Library for 3D Data Processing, 2018, arXiv:1801.09847

[6] Jaesik Park and Qian-Yi Zhou and Vladlen Koltun. Colored Point Cloud Registration Revisited, 2017, ICCV.

[7] F. Endres, J. Hess, J. Sturm, D. Cremers, and W. Burgard. 3-D mapping with an RGB-D camera. IEEE Transactions on Robotics, 2014.

[8] PrimeSense 3D sensors, camera. http://www.i3du.gr/pdf/primesense.pdf

[9] Active Vision Dataset. http://cs.unc.edu/~ammirato/active_vision_dataset_website/index.html.

[10] Voxel down sampling. http://www.pointclouds.org/documentation/tutorials/voxel_grid.php

[11] Radu Bogdan Rusu and Nico Blodow and Michael Beetz. Fast Point Feature Histograms (FPFH) for 3D Registration, Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), 2009.