POINT CLOUD GENERATION AND MATCHING WITH ICP ALGORITHM

Project Proposal by Sahil Arora (32953)

3D reconstruction from 2D images has been an area of interest in Autonomous driving. While various state-of-art algorithms and open source softwares are available [1], this project concerns with one of the fundamental steps of 3D reconstruction i.e. Point could generation. Normally, stereo cameras or RGB-D videos tend to simplify the job by providing pre-set camera positions. But generating a depth map with random images with unknown camera intrinsics is an interesting task and that is the exact scope of this project.

Objective:

Below are the steps to tackle the project (which are further explained with help of sources):

1. Take 6 photographs of a object (to be defined later, probably a mug or a cereal box) from different views. The photos are to be taken with mobile phone digital camera (Samsung J6).

2. Calculate the camera parameters namely Distortion coefficients: Focal length and Optical Center.

3. For a set of pictures, find matching points by using ORB feature descriptor and Brute Force feature matcher.

4. Using the matched points in two images, find the Fundamental Matrix. This now contains information of projection of points from image *j* to image *i*.

5. Triangulate the points in the set of images by calculating the difference vector of *i*-th and *j*-th point in images *i* and *j* respectively. Minimize this vector by finding Jacobian Matrix sand setting it to zero to get depth of that point.

6. Pair 2 images together and generate depth maps using Step 3-5. This depth map can be easily saved as a .ply file that contains the point cloud data.
(a total of 3 point clouds namely A1,A2,A3 will be generated from 6 available images).

7. Merge these 3 point clouds into a single point cloud (named A) using the ICP algorithm with help of Open3D library.

8. To compare our work with available technology, generate a separate point cloud (named B) using Visual SFM. With open-source software CloudCompare[5], compare these 2 point clouds quantitatively and project results.

References/ Sources to be used:

The photographs must be taken from enough angles to understand the scene and 3D-dimensional relationship. Recommendations and guidelines by Carnegie Mellon University, USA can be found at their webpage[2].
OpenCV (an open-source library for image processing) can be downloaded and set up for use by steps given on their website. It provides with an extensive list of tutorials and theory for better understanding. For Steps 2,3 and 4 refer to tutorial webpage[3]. Additionally, a project covered by Mr. Omar Padierna can be found at github repository[4]. The said repository can be referred to export depth map as ply files.

Theory of triangulation is well explained using this github[5], which also covers different methods of optimization. Further, to match point clouds we use Open3D (an open-source library available in C++ and Python, used to deal with 3D data). Open3D can be set up easily in Ubuntu with the given instructions on webpage[6]. It also provides a detailed examples with explanation which can be employed to match point clouds[7]. Work of Dr Cyrill Stachniss (University of Bonn) is helpful in learning details of ICP algorithm[8].
A famous sparse cloud generation software Visual-SFM is considered for benchmarking. Visual SFM can be set up using instructions provided on homepage[9]. Quantitive comparison of point clouds A and B is done using software CloudCompare which can be installed through webpage[10]. Tutorial to compare the two

clouds under consideration is found at webpage[11] which calculates offset, mean and standard deviation of offset. Apart from these references, theory part and conceptual understanding is necessary and can be found at book by Szeliski[12] in Section 4.1, Sec. 6.1 – 6.3 and Section 7.1

[1] Juliano Emir Nunes Masson and Marcelo R. Petry, “Comparison of Algorithms for 3D Reconstruction” in Autonomous Robot Systems and Competitions, 2019 IEEE International Conference on.

[2] “Instruction on Reconstruction Workflow”, https://www.cs.cmu.edu/~reconstruction/basic_workflow.html

[3] “OpenCV-Python Tutorials”, https://docs.opencv.org/3.4/d6/d00/tutorial_py_root.html

[4] “3DReconstruction”, Omar Padierna, https://github.com/OmarPadierna/3DReconstruction

[5] “Stereoscopy”, https://github.com/dmckinnon/stereo

[6] “Open3D”, ‘Compiling from source’, http://www.open3d.org/docs/release/compilation.html

[7] “Open3D”, ‘ICP registration’, http://www.open3d.org/docs/release/tutorial/Basic/icp_registration.html

[8] “Point Cloud Alignment using ICP (Cyrill Stachniss, 2020; updated)”, https://www.youtube.com/watch?v=djnd502836w

[9] “VisualSFM : A Visual Structure from Motion System”, http://ccwu.me/vsfm/

[10] “CloudCompare”, https://www.danielgm.net/cc/

[11] “Distances Computation”, http://www.cloudcompare.org/doc/wiki/index.php?title=Distances_Computation

[12] “Computer Vision: Algorithms and Applications, 1st ed.”, © 2010 Richard Szeliski, Microsoft Research, http://szeliski.org/Book/

■