Project Documentation

3D Reconstruction of Infants' Cranial Shape using Mobile Devices

EN 601.656 Computer Integrated Surgery II

David Shi
Biomedical Engineering,
Computer Engineering
dshi9@jhu.edu

Tara Tang
Biomedical Engineering
ttang14@jhu.edu

Table of Contents

Algorithmic Steps

This algorithm has three main components:

1. Reading the data
2. Preprocessing the data
3. Generating the 3D reconstruction and mesh.

1. Reading Data

Inputs:

- `YML_dir`: name of directory containing all raw depth files (`.yml`) from Structure Sensor

Outputs:

- Thresholded depth files (`.txt`), all saved in directory, `YML_dir`

Algorithm Steps:

1. Read the directory from the name, given as string, `YML_dir`.
2. Read the camera intrinsic parameters and re-scale them according to ratio between depth image size and color image size.
3. If this `.yml` file has not been processed yet, it will contain OpenCV-specific information in the first two rows. Remove these rows if they are there.
4. Read each row of the `.yml` file and convert to an array, `depthMapRaw`.
5. Convert `depthMapRaw` to a numeric array and reshape it according to the depth map size.
6. Apply Gaussian blurring to the depth information.
7. Write blurred depths as 3D spatial points into a `.txt` file, where each row contains the `x`, `y`, depth, using the camera intrinsic parameters.
    a. If the depth is greater than the given threshold, do not include this point in the output `.txt` file.
8. Save the final `.txt` file into the `YML_dir`.

2. Preprocessing Data

Inputs:

- `TXT_dir`: name of directory containing all thresholded depth files (`.txt`, output from reading data)

Outputs:

- `pcds`: list of Open3D point cloud objects, each point cloud corresponding to one image.

Algorithm Steps:

1. Read the directory from the name, given as string, `TXT_dir`.
2. Use Open3D to read each text file line by line and generate a 3D point cloud.
    a. If a threshold has been provided, remove all points in the point cloud with a depth farther than the threshold.
3. Append the point cloud to the list, `pcds`.
4. For every point cloud in `pcds`, estimate the normals using Open3D.

3. Generating 3D Reconstruction

Inputs:

- `pcds`: list of point clouds corresponding to each original image, with normals.

Outputs:

- `pcd_combined_down`: single point cloud combining all other point clouds registered to one coordinate frame, downsampled.
- `mesh`: final triangle mesh generated from `pcd_combined_down`.

Algorithmic Steps:

1. For the point cloud in `pcds` with index `i`: set the `i+1` point cloud as the source and the `i` point cloud as the target.
2. Use Open3D to perform a coarse Iterative Closest Point (ICP) registration starting from identity to find the point-to-plane transformation of the source to the target.
    a. Optional: visualize the coarse registration if `visualize_coarse` is true.
3. Initialize a fine ICP registration using the result from the coarse ICP. The fine ICP has a smaller maximum correspondence distance for more fine tuning of the transformation matrix.
4. If the fitness of the fine ICP is below 65%, repeat steps 3 and 4 with the source point cloud as `i+2` instead.
    a. If the fitness of step 5 is still below 65%, disregard step 5 and continue with the output transformation from step 4.
    b. If the fitness of step 5 is above 65%, delete the `i+1` point cloud from pcds. Continue with the output transformation from step 5.
5. Store the output transformation in `ls_transforms`, a list that contains the rigid transformations between consecutive point clouds.
6. Repeat steps 1-5 for every point cloud in `pcds`.
7. Initialize a point cloud, `pcd_combined`, as the very first point cloud (index `0`) in `pcds`.

8. For the point cloud in `pcds` with index `i≠0`: compose all transformation matrices between *`i`*, *`i-1`*, *`i-2`*, …, *`0`*.
9. Apply this composition transformation to the point cloud and add the point cloud to `pcd_combined`.
10. Repeat steps 7-8 for every point cloud in `pcds`.
11. With the final `pcd_combined`, downsample using Open3D to desired voxel size to get `pcd_combined_down`, which is saved as a `.pcd` file in the current directory.
12. Use Open3D to generate a mesh from `pcd_combined_down` using the Poisson surface construction method.

Computer Program Structure

`constants.py`

This file contains the provided camera intrinsic parameters for the Structure Sensor.

`preprocess.py`

```
preprocess()
    └── visualizeDepthMap()
    └── visualizeColorImage()
    └── convertYMLdir()
        └── getOutputFileName()
        └── getDepthMap()
        └── gaussianBlurDepthMap()
        └── writeYAMLtoTXT()
            └── calculate3DpointFromDepth()
    └── readtxt2PointCloud()
```

`preprocess.py` contains functions to convert raw depth maps to depth-thresholded `.txt` files. It also contains methods for applying Gaussian blur to the depth map, and converts the text files into Open3D point clouds.

`convertDepthMaps.py`

```
convertDepthMaps()
    └── preprocess.writeYAMLtoTXT()
```

This file uses the `writeYAMLtoTXT()` function in `preprocess.py` to convert the raw depth maps (`.yml`) to thresholded text files (`.txt`). This function takes quite a while to run, but it only needs to be run once for a set of raw images. Once these images are converted, the rest of the pipeline only uses the text files.

pointCloud.py

```
pointCloud()
    └── consecutive_registration()
        └── pairwise_registration()
            └── draw_registration_result()
```

`pointCloud.py` contains functions to find the rigid transformations between consecutive point clouds via a coarse and fine Iterative Closest Point (ICP) method. It thresholds the fitness of the fine ICP result to remove individual badly registered point clouds. It also composes all of the consecutive transformations in order to generate the final combined point cloud, which is saved to the current directory.

util.py

This file contains basic Open3D methods for outlier-removal and voxel down-sampling, which are used in several other files.

readInput.py

```
readInput()
    └── preprocess.readtxt2PointCloud()
    └── pointCloud.consecutive_registration()
    └── util.draw()
```

`readInput.py` first reads each depth map's `.txt` files into Open3D PointCloud objects and then runs the consecutive registration algorithm on the list of PointClouds. It returns the combined PointCloud after registration and displays it.

reconstruct.py

```
reconstruct()
    └── util.remove_outliers()
    └── util.downsample_voxels()
        └── poissonSurfaceMethod()
    OR
        └── alphaShape()
```

`reconstruct.py` takes in the `.txt` or `.pcd` file and the mesh generation algorithm option and turns the point cloud into a mesh. It preprocesses the files by first removing outliers and downsamples the point cloud to a specific voxel size according to the methods in `util.py`. The two mesh generation algorithms are alpha shapes and surface Poisson reconstruction, both implemented by Open3D. The generated mesh can be displayed and can be saved as a `.ply` file.

visualize.py

```
visualize.py
     └── util.remove_outliers()
     └── util.downsample_voxels()
     └── util.draw()
```

`visualize.py` takes in the filename of the `.txt` or `.pcd` file and displays it. It preprocesses the files by first removing outliers and down-samples the point cloud to a specific voxel size according to the methods in `util.py`. It then displays the point cloud after these preprocessing steps.

eval.py

This file takes in an output point cloud (`.pcd`) and a corresponding ground-truth mesh. The user must manually find basic transformation matrices (e.g. rotations, translations) to line up the two point clouds as accurately as possible. Then, the program composes these transformations and performs a fine ICP registration on them. With these registered point clouds, it calculates and displays the average surface distance between the two point clouds.

Citations

Qian-Yi Zhou, Jaesik Park, Vladlen Koltun. (2018). Open3D: A Modern Library for 3D Data Processing. arXiv:1801.09847.

Packages

```
python=3.7.4.final.0
numpy=1.19.1
open3d=0.12.0
matplotlib=3.1.1
pyyaml=5.1.2
scipy=1.6.1
opencv<=3.4.2.16
```