****

Dan Adler (daadler0309@gmail.com)
Tiffany Chung (tiphchung@gmail.com)
600.446 CIS2: CamC Calibration Project

Computer Aided Medical Procedures (CAMP) Lab

Automated Calibratrion of CBCT to RGBD
Bunny Model Proof of Concept

Created: 5/5/2016
Last Updated: 5/5/2016

****

INTRODUCTION

This README is for the calibration of the bunny model, i.e. an improved calibration technique using a new phantom. The basic idea behind the model is to map both the CBCT and RGBD to a known-phantom. The phantom is a model of the Stanford bunny, and five holes were drilled into the bunny where CBCT markers were placed. The CBCT only picks up the markers, and we extracted the location of each marker on the template. These locations are given by cylinders. Thus, the CBCT to RGBD transforamtion is found by:

1. Computing the CBCT to template transformation: p_template = F1 * p_CBCT
2. Computer the RGBD to template transformation: p_template = F2 * p_RGBD
3. Inverting the RGBD to template transformation to compute:
   p_RGBD = (F2)^(-1) * F1 * p_CBCT

Unfortunately, this code was done to improve a proof of concept. Thus it is split up into multiple files, many of which are utilized in our automated algorithm, with a few extra parts. The proof of concept was shown though, and I'll detail how to actually find the transformations below.

METHODS

To compute the transformation, one can follow the following steps, each of which will be detailed below.

1. Compute the centroids of each marker in the template and each marker found by the CBCT
2. Segment the RGBD data to grab just the bunny section
3. Use the PCL Template Alignment Algorithm to map the CBCT centroids to the marker

centroids, and refine with ICP
4. Use the PCL Template Alignment Algorithm to map the RGBD point cloud to the full bunny template
5. Invert the RGBD transformation

Note: these steps will only work if you have the below dependencies. Also, not all steps will be detailed specifically, only those with created code. All transformations from .obj to .pcd or .ply to pcd are not going to be listed, but the file formats can be changed using pcl terminal commnds:

pcl_obj2pcd input.obj output.pcd
pcl_ply2pcd input.ply output.pcd

We put these binaries in the bin folder just because we're nice :)

Again, this was a proof of concept, but could be easily introduced into the expected deliverable GUI if they are not already there. I also assumed that we are given the template in a .ply file or .obj file, as well as the cylinder points, and the RGBD image is already compiled into a .ply file. Tiffany did write code to be able to compile the RGBD image, but it was not used in this proof of concept.

DEPENDENCIES
---

1. Point Cloud Library (PCL): http://pointclouds.org/
2. Insight Segmentation and Registration Toolkit (ITK): http://www.itk.org/
3. The Visualization Toolkit (VTK): http://www.vtk.org/
4. Qt: http://www.qt.io/
5. Boost: http://www.boost.org/
6. Eigen: http://eigen.tuxfamily.org/index.php?title=Main_Page

BUILDING

Unfortunately, the code is not listed in the prettiest format since this was utilized as a maximum deliverable, and only a proof of concet was shown. Thus, for each code that needs to be compiled, there exists a cmake file for that code. Each cmake file is labled with the following:

CMakeLists_srcname.txt

Thus, to compile, one can simply delete the suffix "_srcname.txt" and change it to ".txt". In the main folder for "full_matching", one can type:

cmake CMakeLists.txt

and then "make" to compile the specific program specified by srcname which is the name of the file in the src folder. The program executable will be made in the "bin" directory. In summary

1. Go to MAIN directory.
2. Change "_srcname.txt" to simply ".txt"
3. Type cmake CMakeLists.txt
4. Type make

ALGORITHM

Find the Centroids of a point cloud:

CMake File: CMakeLists_find_centroids.txt
src: find_centroids.cpp
bin: PCLFindCetroids
command: ./PCLFindCentroids input.pcd output.pcd tolerance lower upper

Takes a point cloud and clusters based upon a given tolerance (i.e. distance between clusters), a lower bound number of points per cluster, and an upper bound number of points per cluster. The centroid of each cluster is the computer, and the centroids are outputted in the output.pcd file.

For the template points the following parameters are recommended:
tolerance = 2.5
lower = 10
upper = 200

For the CBCT point cloud the following parameters are recommended:
tolerance = 2.5
lower = 10
upper = 500

Specific Inputs:
1. inptut.pcd is an input point cloud
2. output.pcd is the output point cloud of centroids.
3. tolerance is the minimum distance between two clusters
4. LB is the lowest number of points per cluster
5. UB is the upper bound number of points per cluster

Specific Outputs:
1. output.pcd is the point cloud of cluster centroids

Segment the RGBD:

CMake File: CMakeLists_rgbd_crop.txt

src: rgbd_crop.cpp
bin: PCLRGBDCrop
command: ./PCLRGBDCrop input.pcd output.pcd

Takes in an input point cloud and segments it based upon hard coded parameters. These hard coded parameters essentially cut planes in the point cloud and segment towards chosen points in the plane. The parameters chosen to segment just the RGBD bunny were:
-100 < x
-50 < y < +5
-inf < z < +inf (all z)

Specific Inputs:
1. input.pcd is the RGBD image to be segmented
2. output.pcd is the name of the file to be outputted with the segmented data

Specific Outputs:
1. output.pcd is the segmented data

Match the CBCT to Template:

CMake File: CMakeLists_template_alignment.txt
src: template_alignment.cpp
bin: PCLTemplateMatching
command: ./PCLTemplateMatching PointCloud1.pcd PointCloud2.pcd Output.pcd Output.txt

Uses a Fast Point Feature Histogram Algorithm (FPFH) to align features from the PointCloud1.pcd to the features in the PointCloud2.pcd file. It then computes a rigid transformation (translation vector and rotation matrix) between corresponding points nad outputs this to output.txt. The output.pcd file is the transformed PointCloud1.pcd

Specific Inputs:
1. PointCloud1.pcd is the starting point cloud
2. PointCloud2.pcd is the target point cloud
3. Output.pcd is the filename to the ending point cloud
4. Output.txt is the filename to the resulting transformation

Specific Outputs:
1. Output.pcd is PointCloud1 transformed into PointCloud2 space
2. Output.txt is the best transformation found

For the CBCT centroids matching to the template centroids, timing information was:

real 0m0.150s
user 0m0.132s
sys 0m0.016s

Refine the CBCT to Template Transformation:

CMake File: CMakeLists_icp_alignment.txt
src: icp_alignment.cpp
bin: PCLICPAlignment
command: ./PCLICPAlignment template.pcd target.pcd init_reg.txt output.txt err_criteria curr_criteria iter_criteria

Uses Iterated Closest Point (ICP) to refine a given transformation between two point clouds. Convergence is based upon
three error criteria describing the summed euclidean distance between point clouds, the difference between subsequent
error iterations, and the maximum number of iterations that are allowed.

Specific Inputs:
1. template.pcd is the thing that is being registered
2. target.pcd is the thing the template is being registered to
3. init_reg.txt is the initial registration from the FPFH algorithm
4. output.txt is the name of the file to be outputted after ICP
5. err_criteria is the lower bound for the sum of squared errors that would make the algorithm terminate
6. curr_criteria is the lower bound difference between subsequent errors that would make the algorithm terminate
7. iter_criteria is the maximum number of iterations ICP can run

Specific Outputs:
1. output.txt is the resulting transformation after ICP

Match the RGBD to Template:

(SAME AS ABOVE for Match the CBCT to Template)

CMake File: CMakeLists_template_alignment.txt
src: template_alignment.cpp
bin: PCLTemplateMatching
command: ./PCLTemplateMatching PointCloud1.pcd PointCloud2.pcd Output.pcd Output.txt

Uses a Fast Point Feature Histogram Algorithm (FPFH) to align features from the PointCloud1.pcd to the features in the PointCloud2.pcd file. It then computes a rigid transformation (translation vector and rotation matrix) between corresponding points nad outputs this to output.txt. The output.pcd file is the transformed PointCloud1.pcd

Specific Inputs:
1. PointCloud1.pcd is the starting point cloud
2. PointCloud2.pcd is the target point cloud
3. Output.pcd is the filename to the ending point cloud
4. Output.txt is the filename to the resulting transformation

Specific Outputs:
1. Output.pcd is PointCloud1 transformed into PointCloud2 space
2. Output.txt is the best transformation found

For the RGBD bunny matching to the template, timing information was:
real 0m37.373s
user 0m32.884s
sys 0m1.772s

which was very fast!

Refine the RGBD to Template Transformation:

(SAME AS ABOVE for Refine the CBCT to Template Transformation)

CMake File: CMakeLists_icp_alignment.txt
src: icp_alignment.cpp
bin: PCLICPAlignment
command: ./PCLICPAlignment template.pcd target.pcd init_reg.txt output.txt err_criteria curr_criteria iter_criteria

Uses Iterated Closest Point (ICP) to refine a given transformation between two point clouds. Convergence is based upon
three error criteria describing the summed euclidean distance between point clouds, the difference between subsequent
error iterations, and the maximum number of iterations that are allowed.

Specific Inputs:
1. template.pcd is the thing that is being registered
2. target.pcd is the thing the template is being registered to
3. init_reg.txt is the initial registration from the FPFH algorithm
4. output.txt is the name of the file to be outputted after ICP
5. err_criteria is the lower bound for the sum of squared errors that would make the algorithm terminate
6. curr_criteria is the lower bound difference between subsequent errors that would make the algorithm terminate
7. iter_criteria is the maximum number of iterations ICP can run

Specific Outputs:
1. output.txt is the resulting transformation after ICP

Transform each to template space:

CMake File: CMakeLists_transform2template.txt
src: transform2template.cpp
bin: PCLTransform2Template
command: ./PCLTransform2Template cbct.pcd rgbd.pcd output_cbct.pcd output_rgbd.pcd cbct_transform.txt rgbd_transform.txt

Transforms the CBCT and RGBD to template space based upon the transforms found using ICP. It then outputs the transformed
points clouds.

Specific Inputs:
1. cbct.pcd is the input CBCT point cloud file (the original...not the centroids!)
2. rgbd.pcd is the input RGBD point cloud file (preferably segmented!)
3. output_cbct.pcd will be the transformed CBCT cloud file
4. output_rgbd.pcd will be the transformed RGBD cloud file
5. cbct_transform.txt is the rigid transformation CBCT text file outputted from PCLICPAlignment
6. rgbd_transform.txt is the rigid transformation RGBD text file outputted from PCLICPAlignment

Specific Outputs:
1. output_cbct.pcd is the transformed CBCT point cloud
2. otuput_rgbd.pcd is the transformed RGBD point cloud

Transform CBCT to RGBD space:

CMake File: CMakeLists_transform_cbct.txt
src: transform_cbct.txt
bin: TransformCBCT
command: ./TransformCBCT cbct.pcd output.pcd cbct_transform.txt rgbd_transform.txt

Transforms the CBCT scan to RGBD space. Note that the rgbd_transform.txt is not the same RGBD to template
transformation! I used this when I was experimenting, and I computed the transform from the template to
RGBD space, and that's what should be inputted into this for rgbd_transform.txt. Note you can find this transformation the same way as above utilizing FPFH and ICP, but switching the template and target.
Fun fact, the FPFH algorithm takes a longer time to match template to RGBD than RGBD to template. Not
sure why, but good future questions!

Specific Inputs:
1. cbct.pcd is the CBCT original point cloud (not the centroid, the entire point cloud!)
2. output.pcd is the name of the transformed cloud file
3. cbct_transform.txt is the transformation from CBCT to Template space
4. rgbd_transform.txt is the transformation from Template space to RGBD

Specific Outputs:
1. output.pcd is the transformed CBCT in RGBD space

INPUTTED DATA

All of the original point clouds can be found in the "bunny" folder. Here are the pcd files:

1. BBUNNY is the raw DICOM CBCT data
2. bunny.pcd is the RGBD point cloud
3. bunny_flatfoot.pcd is the full phantom with drilled holes
4. bunny-flatfoot-points-processed.pcd are the cylinders on the template that designate the marker placement

If you want my notes to figure out how all of this happened (i.e. my thought process) you can find them in DanThoughts.txt.

Thanks for a fun semester :)