600.446 CIS II: Final Report
Project 6 Automated RGBD to C-arm Calibration

Dan Adler dadler6@jhu.edu
Tiffany Chung tchung12@jhu.edu

May 6, 2016

1 Background and Project Overview

Orthopaedic surgeries involve surgeons placing tools and fixtures in difficult positions inside the patient. X-ray imaging is used to guide and confirm such placement, but hundreds of scans may be required for a single procedure, lengthening procedure time and exposing the physician and patient to high doses of radiation. However, they are necessary in guiding and confirming the surgeon's placement of tools and fixtures.

The Camera Augmented Mobile C-arm (CAMC) project creates a mixed-reality visualization. A more recent implementation augments the commonly used Cone Beam Computed Tomography (CBCT) with Red-Green-Blue-Depth (RGBD) camera information by overlaying the respective 3D images. Improved live visualization of the patient in orthopaedic surgery increases patient safety, especially without the use of external markers, would greatly reduce the number of X-rays and thus radiation dosage, also improving the overall surgical workflow.

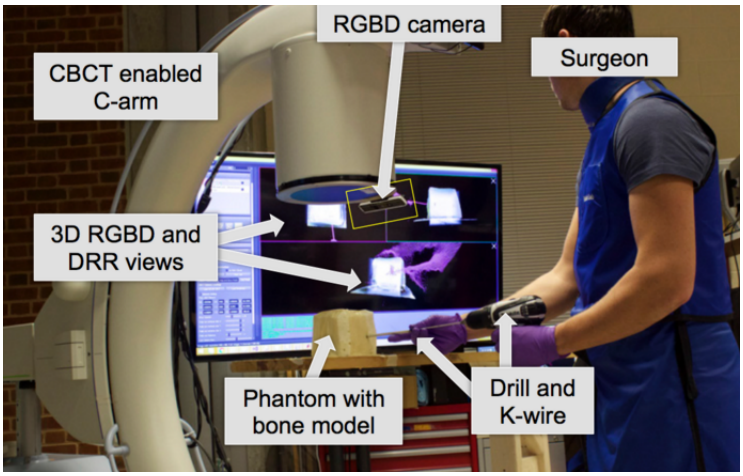


Figure 1: System setup.

2 Problem

A manual calibration algorithm between the CBCT and RGBD is necessary for the mixed-reality visualization because the two scanners do not have the same optical centers and physical properties such as attenuation coefficient and depth information [4]. However, this manual calibration is time and expertise intensive. The first step of the algorithm creates point clouds from raw data by manually clicking through ImFusion SDK's graphical interface. The second step requires visually segmenting the desired surfaces of the point clouds. For example, the CBCT detects both the inside and outside surfaces of the cylinders, while the RGBD only detects the outer surfaces. Additionally, each differ in the amount of noise detected. The third step requires manually refining an alignment between two clouds with visual confirmation of convergence.

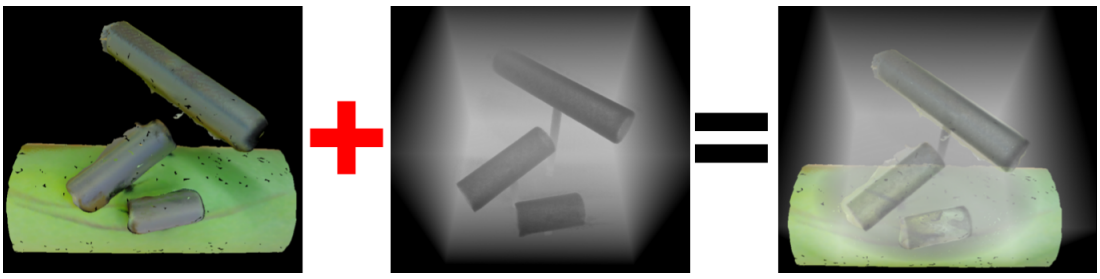


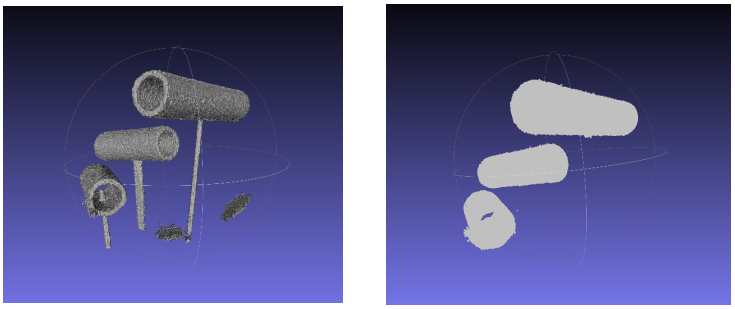
Figure 2: RGBD + CBCT = Overlayed visualization.

2.1 Manual Calibration Steps

1. CBCT and RGBD Data Pre-processing – Done with ImFusion SDK
   KinectFusion Plugin (KinFu) creates the surface reconstruction for the CBCT and RGBD raw data.
   The RGBD data is exported as a `.xyz` file.
   The CBCT data is normalized for the desired intensity and exported as a `.ply` file.

2. Mesh Pre-processing – Done with MeshLab
   From the CBCT and RGBD raw data, extract the outer surfaces of the cylinders only.
   Both are exported as point clouds in `.ply` files.

3. Point Cloud Matching – Done with PCL
   Convert the point clouds to PCL `.pcd` format.
   Fast Point Feature Histogram (FPFH) based-algorithm for initial registration, and Iterative Closest Point (ICP) for refinement.



(a) Before processing in MeshLab. (b) After processing in MeshLab.

Figure 3: Processing in MeshLab.

3 Methods

3.1 Automated Calibration Steps

This project aims to develop a simple, fast and automated calibration process with minimal dependencies and expertise required. In short, the objective is a "one-click" solution that a user with little expertise can use to quickly calibrate and create the visualization. The only third-party libraries used are the Point Cloud Library (PCL), Insight Segmentation and Registration Toolkit (ITK), and Visualization Toolkit (VTK), all of which are open source and freely available.

1. CBCT Data Pre-processing – Done with ITK, PCL
   ITK thresholds the phantoms using user-defined intensity data and saves to a `.obj` file.
   PCL converts the ITK output to a `.pcd` file.

2. RGBD Data Pre-processing – Done with VTK, PCL
   VTK and PCL iteratively registers subsequent 2D RGBD point clouds to reconstruct a 3D point cloud.
   PCL saves this result as a `.pcd` file.

3. Point Cloud Initial Alignment using FPFH – Done with PCL
   The initial alignment result is written to a `.txt` file.

4. Point Cloud Refinement using ICP – Done with PCL
   The final alignment result is written to a `.txt` file, and the resulting overlayed point clouds are displayed with the PCL viewer.

3.2 Four Modules

3.2.1 CBCT Pre-processing

To convert the CBCT DICOM data into a point cloud usable by PCL, we first converted DICOM files into a `.raw` file and a MetaImage `.mhd` file in the `results` folder named `mod1_cbct.mhd`. The position in `mod1_cbct.mhd` is centered at (0, 0, 0). Next the intensities are thresholded using a normal distribution to `mod1_cbct_normalized.mhd.` Voxel intensities are then thresholded such that only voxels with an intensity between a user-inputted lower and upper bound standard deviations from the mean are kept. These voxels are then masked to a binary image (0 or 1) saved in `mod1_cbct_threshold.mhd`. The voxels are converted to a binary 3D mesh and saved as `mod1_cbct.obj` using `ITK::BinaryMask3DMeshSource` to generate a region-based mesh. The algorithm utilized first decides which points to combine into a 3D voxel, and then decides which voxel points to fit together by slowly merging the voxels. ITK based this on Lorenson's Marching Cubes algorithm. Finally, PCL is used to convert the `.obj` file to a `.pcd` file.

3.2.2 RGBD Pre-processing

To convert the RGBD `.png` files into a point cloud usable by PCL, each `.png` file must be processed. For each `.png` file, the depth value from each pixel of the image is converted to a (x, y, z) 3D point location in a point cloud. Secondly, each point cloud is down-sampled and null points (NaNs) are removed. To combine them, the consecutive point clouds are matched to compute the transformation from one point cloud to the next. Each successive point cloud is aligned and combined into one point cloud and saved to a `.pcd` file.

3.2.3 Matching

To compute and perform the transformation between `mod1_output.pcd`, the CBCT point cloud, and `mod2_output.pcd`, the RGBD point cloud, a modified version of the PCL Template Alignment algorithm is used. The clouds are pre-processed by removing distant points and down-sampling. Then PCL's FPFH-based registration is used to compute the transformation, which is saved to `mod3_result.txt` [7]. The transformation is performed to combine the point clouds and saved in `mod3_output.pcd`.

3.2.4 Refinement

The initial transformation computed in the previous Matching module is loaded, or else the identity matrix is used. The `PCL::IterativeClosestPoint` algorithm is then called and the source is set as the CBCT, the target is set as the RGBD camera, and the initial transformation matrix is placed in as a "guess." The three ending criteria for ICP are max iterations, convergence on Euclidean distances between points, and a "low" Euclidean distance. Only one of these criteria needs to be satisfied for the algorithm to terminate. The Euclidean distance algorithm, called a fitness score in PCL, goes through the CBCT target points after performing a registration and finds the corresponding nearest neighbor in the RGBD point cloud (since the point clouds are of unequal size). Based on some predefined max range, if the point exists in the RGBD KD-Tree using `PCL::nearestKSearch`, the squared neighbor distance is added to the fitness score, and a number of points `nr` is incremented to keep track of how many points have been found in the tree.

3.3 Graphical User Interface (GUI)

The GUI combines all 4 modules. The user selects the file path for the CBCT DICOM and RGBD `.png` files. Then the user selects the "Transformation" button and the transformation is computed. There are options to adjust thresholding and convergence criteria. The results can be viewed in the `results` folder. A live visualization of the final overlay accompanies the GUI.

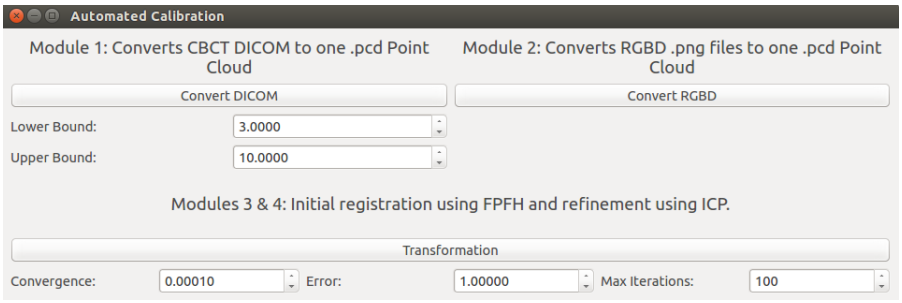


Figure 4: Simplistic GUI.

4 Validation

4.1 CBCT Point Cloud Creation

The raw CBCT scanner data was successfully processed using the ITK normalization filter. The phantom was extracted from noise. The phantom was found to exist between 3-10 standard deviations from the mean intensity (Figure 5).

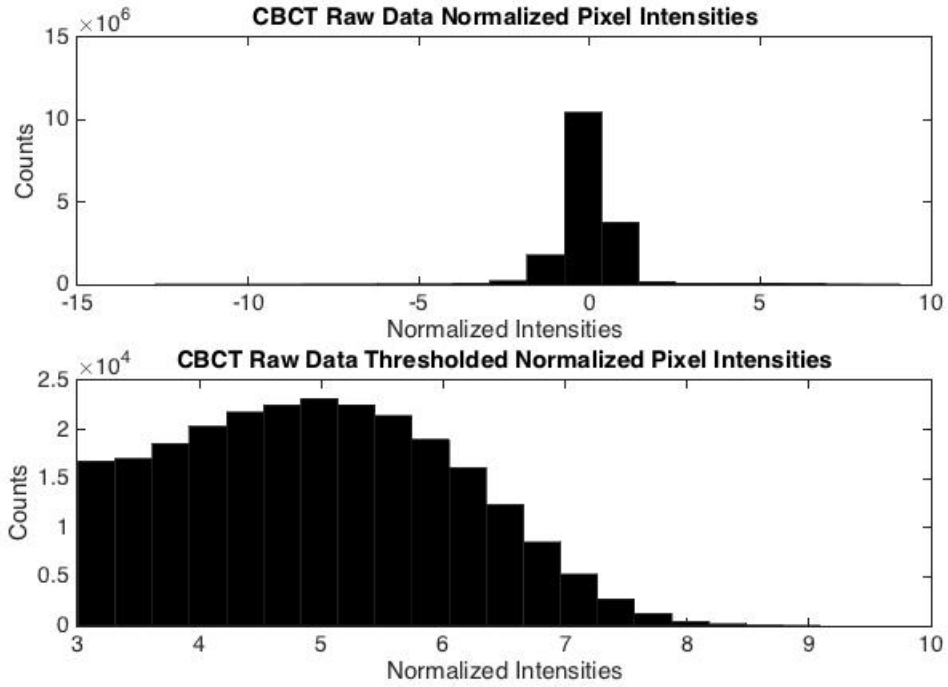


Figure 5: Normalized versus thresholded pixel intensities for CBCT.

4.2 RGBD Point Cloud Creation

The RGBD point clouds were successfully created from the 2D `.png` images. Visually, the RGBD point cloud was recreated, and further testing is being done to validate it.

4.3 Initial Registration and Refinement

After finding an initial matching and refinement using ICP, the matching error between the transformed CBCT points and RGBD points was found. Since there were more points in the CBCT scan, each transformed point was matched to its unique nearest neighbor in the RGBD space. The distribution of errors from the manual calibration and automated calibration were graphed and compared (Figure 6). The mean squared error (MSE) was also compared, where:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(p_{(i,RGBD)} - p_{(i,CBCT)})^2$$

n is the number of points in the RGBD cloud, and $p_{(i,RGBD)}$, $p_{(i,CBCT)}$ are unique nearest neighbor points. Results showed the manual MSE = 12.5563 and the automated MSE = 12.5545.

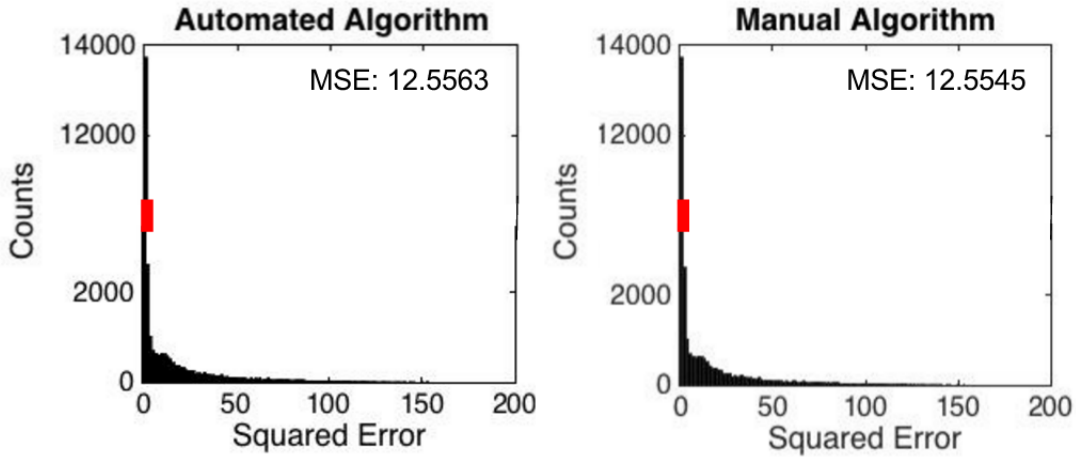


Figure 6: Automated versus manual calibration results.
Manual MSE = 12.5563 and the automated MSE = 12.5545

5 Significance

This project was able to create an automated process for an existing manual calibration algorithm between a CBCT and RGBD volume. The manual calibration algorithm has already been detailed in prior research and received a reasonable registration accuracy of 2.58 mm. Our registration appeared to have similar accuracy, as found by graphing the distribution of squared errors across the nearest-neighbor point cloud.

The automated algorithm was able to achieve similar results utilizing the previous manual calibration phantom when each part was utilized individually. One notable difference between this algorithm and the previous algorithm was that the automation did not include the pre-processing extraction of the point clouds, and assumed that the images picked up point clouds that could be mapped to each other without any further processing. This proves to have issues with the original phantom, and the CBCT scanner picks up the outside and inside surfaces of the cylinders. The RGBD camera picks up just the outside surface of the cylinders. A new phantom was proposed in order to be able to automate the algorithm without having to manually extract relevant parts to be able to align each point cloud, and is described in our future work.

In addition, the project began to explore alternatives to the existing primary FPFH matching algorithm. This algorithm is slow because a nearest-neighbor search is used to compare similar features within two point clouds, and becomes computationally expensive with large point clouds. Our group discussed simple alternatives to this initial matching, like principal component analysis (PCA). This discussion is ongoing.

One concept to note is that this calibration is dependent on a stable environment. Since it is based upon an initial calibration that creates a mixed-reality visualization, changes in the positioning of the cameras or patient between may require re-calibrations. Continuing research is being done to ensure the effectiveness of this work in the operating room. Generally, a need for re-calibration is obvious because the misalignment is visually noticeable. In conclusion, this work does provide an automated approach to an already effective calibration algorithm and meets the objectives of faster time and minimal expertise required. It enhances patient safety by offering surgeons real-time image-guided surgical navigation.

6 Current and Future Work

Our automated algorithm did not invoke a pre-processing step to segment the point clouds of the RGBD and CBCT scanner after the point clouds were assembled from raw data. The reasoning behind deleting this step was that it depended upon ImFusion SDK, a commercial software, and required segmentation expertise that could not be easily automated. This segmentation could not be automated on the phantom that was utilized in the manual calibration algorithm, as both the CBCT and RGBD picked up different objects, and the CBCT was able to pick up the hollowness of the cylinders. Thus, it became necessary to develop a phantom that would not require this segmentation step such that the algorithm could have easier automation.

Currently, our group is working on a new phantom design that is based upon a computer-aided design (CAD) template. The phantom template is the Stanford Bunny, a well-known structure [8]. The plastic utilized to print the bunny is not picked up by the CBCT scanner, but is picked up by the RGBD camera. Thus, five holes were drilled in irregular but strategic and known places inside of the phantom, and metal spherical CBCT markers were placed inside each of the drilled holes. The coordinates of the holes were documented, and a CBCT and RGBD scan was taken. The CBCT scan only picked up the markers within the new phantom, and these markers were successfully extracted using our CBCT pre-processing algorithm. The RGBD scan acquired an acceptable mesh of the bunny, but will be improved with a larger model.

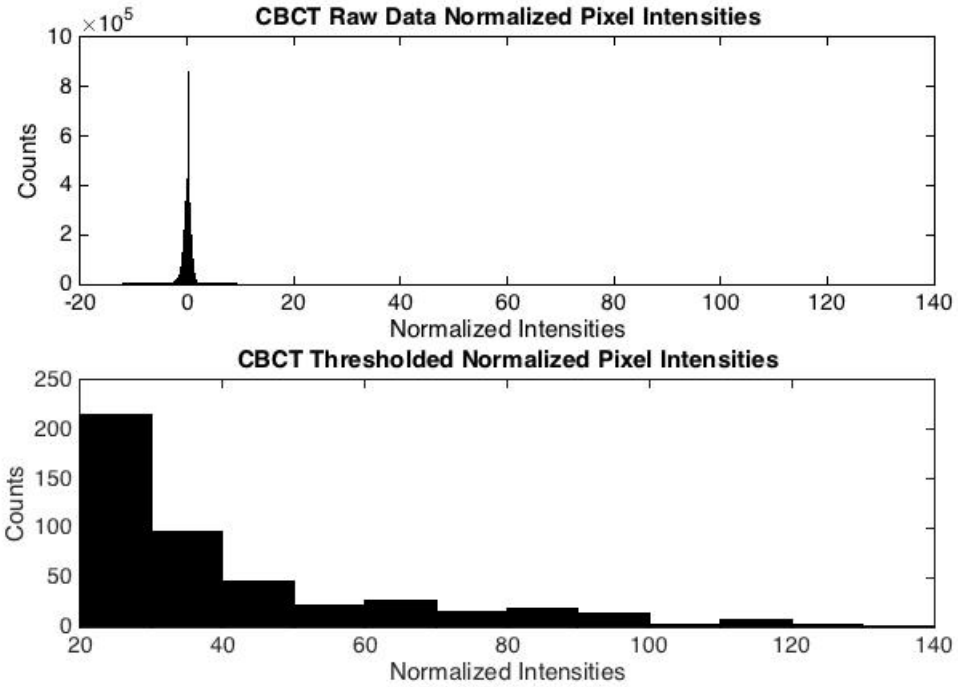
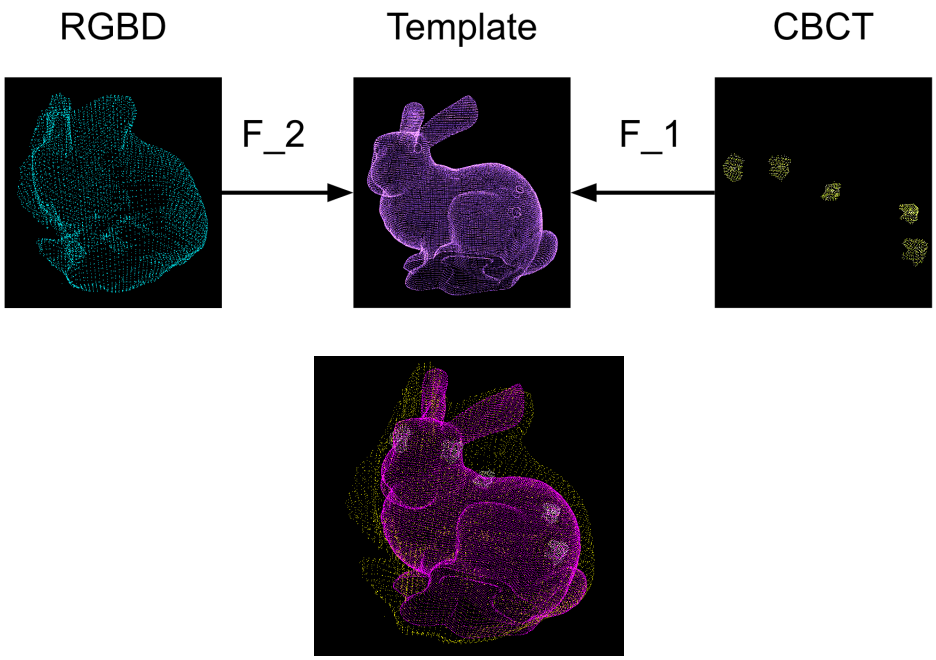


Figure 7: Raw versus thresholded pixel intensities for CBCT.

We then developed a proof-of-concept model to show that it would be possible to transform the CBCT and RGBD to template space, and then utilize each transformation to map the CBCT to RGBD space. This, in addition to a segmentation step that masks the RGBD, is much less intensive than the previous manual process. The new process could be directly applied to our automated code in the future. These results were not refined, and are still being improved. The figure below shows the simple reasoning behind this new model, and the first step of our work to create this transformation. This transformation is being refined and a quantitative analysis will be completed in the future.



$$p_{Template} = F_1 \cdot p_{CBCT}$$
$$p_{Template} = F_2 \cdot p_{RGBD}$$
$$p_{RGBD} = F_2^{-1} F_1 \cdot p_{CBCT}$$

Figure 8: Calibration of RGBD and CBCT to Stanford Bunny with overlay.

The proposed algorithm will be as follows:

1. Create point clouds of raw RGBD and CBCT data.
2. Segment RGBD using a predefined box around the phantom.
3. Compute CBCT to template transformation and RGBD to template transformation.
4. Invert the RGBD transformation and combine with CBCT transformation to create final result.

7 Management Summary

The project was split up based upon technical strengths, but both partners consistently worked together to confirm that milestones were reached correctly and in a timely manner. The team worked to document the code they wrote so that it is readable to new users and updated the wiki as needed. They also scheduled meetings with their mentors, and they attended weekly lab meetings to give updates, discuss current progress, and get feedback for the project.

Dan Adler worked on the algorithms for the project, since his personal background is in applied mathematics. He built the methodology for using ITK to extract relevant point clouds in the CBCT images, and studied different techniques to normalize the point clouds and to choose which areas of the point clouds to threshold. In addition, Dan built the termination criteria for the matching portion of the algorithm (FPFH and ICP). Lastly, he formulated the new approach on the maximum deliverable (i.e. mapping to a CAD template), and researched why FPFH might not be the best algorithm for usage.

Tiffany Chung worked on the GUI for the project, and applied her coding background to package the automated algorithm. She did this by concatenating the code in both the point cloud processing and matching. Then, she applied Qt to build a front-end GUI such that users can easily choose which files they would like to match, and walk through the algorithm. Tiffany combined the 2D RGBD point clouds into a 3D point cloud, and studied the clinical applications of this process such that it could be practically implemented in the operating room.

Lessons learned by both Dan and Tiffany include the methodology and culture of a research group, exploration and integration of third party software and libraries (ImFusion SDK, MeshLab, PCL, ITK), and building a large C++ application with Qt GUI and numerous dependencies with CMake.

8 Acknowledgments

We would like to thank Dr. Nassir Navab's Computer Aided Medical Procedures (CAMP) group at Johns Hopkins University, especially Bernhard Fuerst, Sing Chun Lee, and Javad Fotouhi for their mentorship during this project. Special thanks to our guest Jennifer and moral support from Phoebe Yeo.

9 Technical Appendices

The User Manual can be found in READMEs and header comments for each module. Below are the files where each module is implemented, and then used. Continuing work on the new phantom (Stanford Bunny) can be found in the bunny_matching folder.

`module1_cbct.h/.cpp`
`module2_rgbd.h/.cpp`
`module3_matching.h/.cpp`
`module4_registration.h/.cpp`
`autoCalViewer.cpp` – driver file
`autoCalibration.pro` – Qt GUI

References

[1] Paul J. Besl and Neil D. McKay. "A Method for Registration of 3-D Shapes". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (1992), pp. 239–256. ISSN: 0162-8828. DOI: 10.1109/34.121791.

[2] Marius Fischer et al. "Preclinical usability study of multiple augmented reality concepts for K-wire placement". In: *International Journal of Computer Assisted Radiology and Surgery* (Mar. 2016), pp. 1–8. ISSN: 1861-6410. DOI: 10.1007/s11548-016-1363-x.

[3] Risto Kojcev et al. "Dual-robot ultrasound-guided needle placement: closing the planning-imaging-action loop". In: *Int J CARS* (2016). DOI: 10.1007/s11548-016-1408-1.

[4] Sing Chun Lee et al. "Calibration of RGBD camera and cone-beam CT for 3D intra-operative mixed reality visualization". In: *Int J CARS* (2016). DOI: 10.1007/s11548-016-1396-1.

[5] Nassir Navab, Sandro-Michael Heining, and Joerg Traub. "Camera Augmented Mobile C-Arm (CAMC): Calibration, Accuracy Study, and Clinical Applications". In: *IEEE Transactions on Medical Imaging* 29.7 (2010), pp. 1412–1423. DOI: 10.1109/tmi.2009.2021947.

[6] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. "Fast Point Feature Histograms (FPFH) for 3D registration". In: *IEEE International Conference on Robotics and Automation* (2009), pp. 3212–3217. DOI: 10.1109/ROBOT.2009.5152473.

[7] Radu Bogdan Rusu and Steve Cousins. "3D is here: Point Cloud Library (PCL)". In: (May 2011).

[8] Greg Turk and Marc Levoy. "Zippered Polygon Meshes from Range Images". In: *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '94. New York, NY, USA: ACM, 1994, pp. 311–318. ISBN: 0-89791-667-0. DOI: 10.1145/192161.192241. URL: http://doi.acm.org/10.1145/192161.192241.

[9] Eric Wahl, Ulrich Hillenbrand, and Gerd Hirzinger. "Surflet-Pair-Relation Histograms: A Statistical 3D-Shape Representation for Rapid Classification". In: *Fourth International Conference on 3-D Digital Imaging and Modeling* (2003), pp. 474–481. DOI: 10.1109/IM.2003.1240284.