# Eye-in-Hand Registration for the REMS Robot

Zach Sabin and Joseph Min

Mentors: Russell Taylor, Yunus Sevimli

May 6, 2016

**Abstract**

We created a software package that registers images taken from an RGB-Depth (RGBD) camera mounted on the REMS robot to a mesh model generated from a pre-operative Computerized Tomography (CT) scan.

## 1 Introduction

Minimally invasive surgeries are almost always preferable to invasive ones. They allow patients to recover more quickly and have fewer risks of complications after surgery. However, particularly around the head and the throat, these surgeries can be extremely difficult. There are many critical structures, and instruments often need to be removed and reinserted. The tissue in these regions is sensitive, and it can impair a surgeon's vision when they look through an endoscope [1]. Using a robot to aid in the surgery can reduce hand tremor, increase precision, and help the surgeon navigate through the complex anatomy of the head and the throat.

The REMS robot has been developed in order to achieve those goals. The robot is designed to aid surgeons with ear, nose, and throat surgeries via cooperative control with the surgeon.

The goal of our project was to help the surgeon align the robot to the patient in a certain orientation. This is useful because it allows the robot to be positioned in the optimal fashion for a certain surgery. In order to achieve this we mounted an RGBD camera to the robot's tool holder. Using this camera we could then register the patient's position relative to the camera to a pre-operative CT scan. A surgeon could then position the robot such that the transformation is of a certain value that corresponds to the ideal starting position.

A secondary goal of our project was to be able to perform a calibration such that we can determine the transformation between the wrist of the robot and the camera. This type of calibration is known as a "Eye in Hand Calibration". This calibration also allows for the system to determine the pose of the patient relative to the base of the robot, which is important in order to optimally position the robot.

# 2 Technical Approach

## 2.1 Camera

The camera that we mounted onto the REMS robot was an Intel RealSense camera. This camera computes a depth value for each pixel in its sensor. Using these depth values one can then compute a point cloud. This point cloud is extremely high resolution as the camera is a 640x480 resolution, providing around 300,000 points in the point cloud.

## 2.2 Point Cloud Processing

This level of detail for the point cloud caused our registration algorithm to be too slow. Additionally the camera is fairly noisy, and such a fine level of detail is therefore extremely noisy. In order to reduce the size of the point cloud we downsample the point clouds as they come in from the camera.

We used a K-dimensional Tree (KD Tree) with a fixed leaf size to downsample this cloud. Using the point cloud, we built a KD tree where every leaf node's bounding box had a fixed size (e.g. 1 mm). Once the KD tree was populated the centroid of every leaf node was placed into the new "downsampled" cloud. This method is advantageous for two reasons. The first is that sections of lower density are not sampled away since each leaf node is included in the sample cloud, regardless of whether it has 50 points or 1 point within it. The second is that it filters clouds to comparable resolutions regardless of their original density since it is based on a fixed measurement rather than a proportion of points.

An additional issue we dealt with the point clouds is that they are taken while the patient (or phantom) is placed on a table. This results in the table being included in the point cloud anywhere the camera is not viewing the patient. In order to remove the table we used the open source Point Cloud Library's plane segmentation method.

As the base algorithm for plane segmentation we selected the RANSAC algorithm. RANSAC is a general algorithm for estimating the parameters of a model within data by sampling pieces of the data. In this case we were using it to estimate the parameters of the plane within the overall point cloud. Briefly, RANSAC works by randomly selecting a certain part of the data. It then attempts to fit the model parameters (in this case the coefficients of a plane) to this data. Once it has done this it computes what percentage of the overall data fits the model parameters that it has just computed. It does this a certain number of times and then returns the best parameters found, or no parameters if it could not successfully fit a model to a high enough percentage of points.

## 2.3 Registration

Once we have a processed point cloud we then attempt to compute the transformation between the cloud and the reference mesh (from the pre-operative CT scan). To get this registration, we use the standard ICP algorithm. We will not discuss ICP in great detail, but it is an algorithm that iteratively approximates the transformation between two objects and (hopefully) improves on each guess.

### 2.4 Calibration

Being able to calibrate the camera such that we know the pose of the camera relative to the wrist of the robot makes the registration much more useful. In order to do this we set up a system of equations resembling $AX = XB$ where A are the differences between poses of the robot and B are the differences between poses of the calibration object relative to the camera, and then we can solve for X.

The first step to do this is to set up the $AX = XB$ equation. This is done by taking data from the camera at a sequence of poses. At each of these poses we also record the pose of the robot (stored as a position and a quaternion).

The system of equations is then

$$F_{co,k}F_{co,0}^{-1}F_w = F_w F_{bw,k}F_{bw,0}^{-1}$$

Where $F_{co}$ is the pose of the calibration object relative to the camera and $F_{bw}$ is the pose of the robot wrist relative to the robot base.

We then find the best solution for this system of equations. The method for this is out of the scope of this paper, but it essentially constructs a minimization problem using the quaternions of the poses, then solves the minimization problem in order to find the best rotation for $F_w$. This can be seen in more depth in [2].

After solving for the rotation matrix of $F_w$ we then solve for the translational section using a simple least squares method. Again this technique can be seen in [2]

We found the poses of the calibration object relative to the camera by using a large, solid angle bracket as calibration object. By taking RGBD images of this object and then segmenting the two planes of the object and the plane of the table we can compute a centroid and a set of axes for each image. This allows us to generate a pose.

## 3 Results

### 3.1 Point Cloud Processing

The point cloud processing has been largely successful at downsampling and the removal of background planes. The downsampling can easily and quickly return a point cloud sampled to any degree of simplification while keeping the overall integrity of the point cloud. Shown below is a 206,201 point point cloud first downsampled with a leaf size of 15 millimeters to a point cloud with 24,117 points. The overall integrity is easily observed. The plane behind the desired cloud is then segmented and removed using a tolerance of 20 millimeters, leaving 5,634 points to use for registration. The final product still contains some residual points from the background plane outside the tolerance level and some noise above the point cloud.

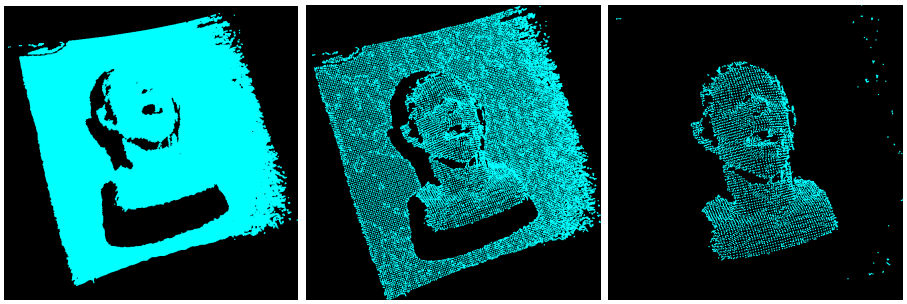

Figure 1: A point cloud gets processed

### 3.2 Registration

The ICP registration algorithm returned an average error of 15 millimeters from each point in the point cloud to their respective surface triangles on the mesh. One such example of these results can be seen in the figure to the right where ICP was unable to tilt the point cloud (in blue) backwards to correctly align the point cloud to the mesh (in red).

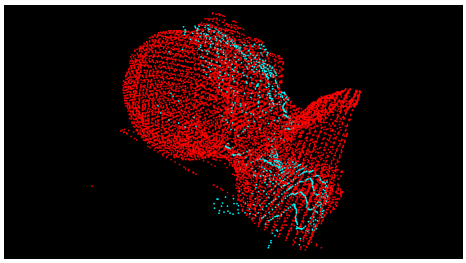

Figure 2: A point cloud registered to a mesh

The performance of the ICP algorithm is highly variable. Often it completely flips the point cloud relative to the mesh, registering the shoulders of the cloud to the head of the mesh. Of note though is that it successfully places the point cloud onto or within the mesh.

### 3.3 Calibration

We were able to rigorously test our $AX = XB$ given various test data, and we verified its correctness; however, we were unable to successfully integrate calibration into our product at this time. As it is currently, we use a perpendicularly angled calibration object and attempt to use plane segmentation to find three perpendicular planes that define the object's coordinate system. Given the definition of this coordinate system with multiple frames of the calibration object, we should be able to find a frame transformation from the coordinate system of the camera to that of the calibration object. However, we have thus far been unable to output the correct frame transformation.

## 4 Discussion

Moving forward, things we would like to improve on include outlier rejection during point cloud processing, improving the registration algorithm, and implementing our calibration step.

### 4.1 Point Cloud Processing

A large portion of error also came from the camera itself. Its depth imaging is actually very sensitive and often produced large amounts of noise due to thing such as small smudges on the lens or (possibly) other sources of infrared light. We would like our algorithms to be able to be robust to the point that registration and calibration would both work even with noisy data sets such as those produced by a overly sensitive camera. Methods to deal with this would possibly include large outlier rejection during pre-processing of the point cloud, and more fine tuned down sampling of point clouds and meshes.

### 4.2 Registration

We suspect that the high error returned by the ICP registration algorithm is largely due to the fact that ICP has trouble dealing with incomplete point clouds (the camera cannot return what it can't see, such as the backside of the patient's head), and that it often gets stuck at local minimums. The incomplete point cloud means that the point cloud has a higher possibility of fitting to incorrect portions of the mesh, increasing the probability for misalignment. However, if the misalignment has a locally minimum error, ICP is likely to label it as the correct registration. Another possible source of error comes from the scaling of the camera. The camera depth is scaled in micrometers and using the depth one can compute the "real world" x and y coordinates using the depth as the z. However the accuracy of these depth units is questionable as we often saw point clouds that appeared flattened or elongated.

For our registration algorithm, we would like to implement a way to automatically find the necessary scaling factor between the point cloud and the mesh. Possible ways to do this include using multiple guesses when running ICP (effectively running ICP multiple times on differently scaled clouds) and choosing the scale factor that minimizes the error, and finding the distance between two known points on both the cloud and the mesh to calculate a scale factor. The prior would be significantly slower but would require no additional information, whereas the latter would be more efficient but would require more pre-processing at the very least. Another improvement on the registration algorithm would be to develop a method to have a better initial guess (a starting frame transformation) to reduce computation time and final error.

### 4.3 Calibration

For the calibration, there are several things to improve on, the main concern being able to output a valid frame transformation given a series of frames of a calibration object, and integrating this code with an automated way of getting each calibration image.

## 5 Management Summary

### 5.1 Who Did What

Joe got the point cloud from the Real Sense camera on Unix, used PCL to downsample point cloud and remove the plane, and worked on creating the

poses of the calibration object from point clouds.

Zach used Seth Billing's code, wrote the Hand Eye Registration and Calibration classes, created the software package and class structure, wrote the Interface class, wrote the interface for the Intel camera on Window, and set up the environment for the software package.

## 5.2 Planned vs. Accomplished

Our original plan was to get registration working, implement calibration, and potentially add some kind of visualization or guidance for the surgeon so that they could properly align the robot. We failed to accomplish these goals. We created a package that has the capability to do all of these things. However, our registration is not successful as it frequently fails to properly align the point cloud with the mesh. We did not fully implement calibration, as the processing step where we compute the poses from the point clouds of the calibration image is not functionally implemented.

## 5.3 Future Work

The first thing we would like to do is experiment with different registration algorithms. ICP, Generalized ICP, Feature Based Registrations, and IMLP are all possible options for this algorithm. We would also like to successfully implement calibration. The remaining step is to successfully compute the poses from the point clouds, which is currently not working. The third piece of work would be to improve the user interface. Implementing a visualization of how to position the robot would be fairly straightforward once registration and calibration are functioning.

## 5.4 What We Learned

### 5.4.1 Software Engineering

Our number one takeaway from this project was that setting up an environment and getting dependencies can be a massive undertaking. Neither of us were experienced setting up complicated C++ (or other languages) development environments that depended on multiple SDKs and libraries and this took up a much greater proportion of time and effort than it should have and prevented us from successfully achieving our deliverables as we would have wanted to. However we have learned a great deal about CMake and, in general, how to set up complex C++ build systems with lots of external dependencies. It's unfortunate that it took us such a great deal of time and effort that could have been spent actually working on the problem.

### 5.4.2 Technical

In terms of technical knowledge we also learned a great deal. We learned more about different registration algorithms such as ICP, IMLP, and G-ICP and their strengths and weaknesses. We also learned a great deal about Eye in Hand calibration and the math behind it, since we implemented the algorithm (although could not successfully integrate the algorithm with the camera's point cloud

images. We also learned about point cloud processing and modeling by downsampling the clouds from the camera and applying the RANSAC algorithm to remove the plane of the table from the cloud.

## Works Cited


1. "Robotic Ear Nose and Throat Microsurgery System (REMS)." Research.rems – CIIS. Johns Hopkins University

2. Taylor, Russell. "Calibration." Computer Integrated Surgery 1. Johns Hopkins University. http://www.cs.jhu.edu/cista/445/Lectures/Calibration.pdf

3. S. Billings, A. Kapoor, M. Keil, B. J. Wood, and E. Boctor, "A hybrid surface/image-based approach to facilitate ultrasound/CT registration", in SPIE Medical Imaging 2011: Ultrasonic Imaging, Tomography, and Therapy, Lake Buena Vista, Florida, Feb 13, 2011. pp. 79680V-1 to 79680V-12.

4. S. Billings, E. Boctor, and R. H. Taylor, "Iterative Most-Likely Point Registration (IMLP): A Robust Algorithm for Computing Optimal Shape Alignment", PLOS ONE, vol. 10- 3, pp. (e0117688) 1-45, 2015. http://journals.plos.org/plosone/article? id=10.1371/journal.pone.0117688 doi: 10.1371/journal.pone.0117688

5. S. Billings and R. Taylor, "Generalized Iterative Most-Likely Oriented Point (G-IMLOP) Registration", Int. J. Computer Assisted Radiology and Surgery, vol. 8- 10, pp. 1213-1226, 2015. DOI 10.1007/s11548-015-1221-2

6. K. C. Olds, P. Chalasani, P. Pacheco-Lopez, I. Iordachita, L. M. Akst, and R. H. Taylor, "Preliminary Evaluation of a New Microsurgical Robotic System for Head and Neck Surgery", in IEEE Int. Conf on Intelligent Robots and Systems (IROS), Chicago, Sept 14-18, 2014. pp. 1276-1281.

7. K. Olds, Robotic Assistant Systems for Otolaryngology-Head and Neck Surgery, PhD thesis in Biomedical Engineering, Johns Hopkins University, Baltimore, March 2015.