Computer Integrated Surgery II Augmented Reality Assisted Craniofacial Surgery Project Final Report

Group 13: Yihao Liu, Nikhil Dave
Mentors: Dr. Peter Kazanzides, Ehsan Azimi, Dr. Cecil Qiu, Dr. Sashank Reddy

May 9, 2020

1 Abstract

Orbital blowout fractures occur due to significant blunt trauma to the eye and can result in numerous problems for a patient. Treatment involves an orbital floor reconstruction surgery to repair the orbital aperture. This procedure involves the arduous task of implanting an orbital floor implant within the eye socket. To improve patient outcomes and reduce the operative time of this procedure, an augmented reality navigation system is proposed. The system involves a robust registration process utilizing a fiducial initialized iterative closest point algorithm, calibration of an orbital implant, and visualization through an Unity application.

2 Background

This section introduces the background to the orbital floor reconstruction surgery in terms of the cause of the damage and the reconstruction procedure.

2.1 Orbital Blowout Fracture

Orbital blowout fracture is deformity caused by significant blunt trauma to the orbital aperture, more commonly known as the eye socket. If a patient experiences blunt trauma from an object that is of larger size then the orbital aperture, then fractures can occur on the orbital floor and medial wall of the eye-socket, the anatomical consequence of which are included below [6].

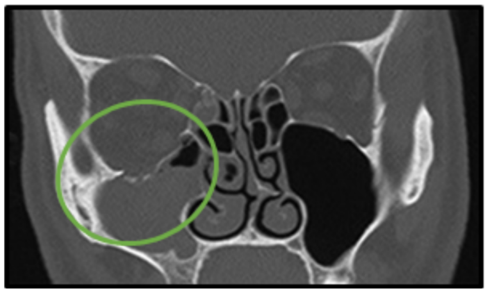


Figure 1: CT Slice of patient with orbital floor fracture and clearly visible herniated tissue into the maxillary sinus, circled [6]

Orbital blowout fractures are common and account for about 40 percent of all facial fractures. This is primarily due to thinness of the orbital floor. Once the orbital has fractured, the bone will be displaced downward into the maxillary sinus [4]. As a result, the contents of the orbital aperture will herniate into the newly opened space. This can cause a variety of problems for the patient and can affect their ability to have proper ocular alignment which can result in difficulty of sight. This is known in the medical community as tropia. Tropia in patients with orbital blowout fracture tend to have constant upward or downward gaze [4].

2.2 Orbital Floor Reconstruction

To return orbital tissue from the maxillary sinus and into the eye socket as well as realign the eye, reconstruction of the orbit floor is necessary. The procedure involves the placement of an orbital implant that restores structure to the eye socket, preventing any orbital tissue from returning to the maxillary sinus and improving any previous tropia [4]. An image of an orbital implant is shown below, to illustrate the implant location with respect to other anatomical landmarks.

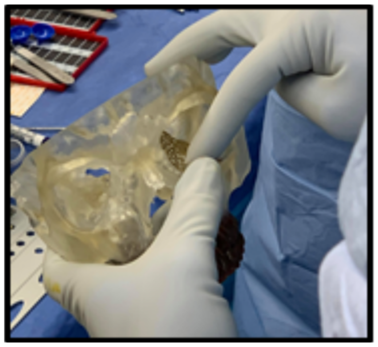


Figure 2: Image of implant being placed in the correct orientation and location by a clinician.

The implant is originally not patient specific and shaping the implant to a relevant size for a particular patient is necessary for restoring stability in the orbital aperture. In order to ensure proper fit in the eye socket, the surgeon may shape the implant multiple times before settling on a shape that is appropriate for the patient's specific anatomy. If the implant is not flush with the orbital wall, it may cantilever during fixation and cause unnecessary damage to orbital tissue.

Orbital floor reconstruction surgery is a long and arduous process, requiring significant attention from the surgeon and manipulation of delicate and complex structures in a tight, compact space. Due to the nature of the orbital floor bone and manner in which it may fracture, shattered bone fragments may be present scattered in the maxillary sinus and in other regions within the operative field.

2.3 Problem

The orbital aperture is a tightly confined space due to the fact that its primary purpose is to provide structure to the ocular system. As a result, a fracture in the orbital aperture is difficult access since it is within the compact space, behind the eye. Additionally, any incision made in order to access the tissue underneath is must be small, giving surgeons limited visibility into the orbital aperture. As a result, it is difficult for surgeons to develop context and orientation of the anatomy once they have dissected along the orbital wall. This sense of orientation is necessary, as placement of the orbital implant plate requires precise shaping of the implant and identification of the posterior edge of fracture. Correct implant placement involves placing the implant's distal portion on this location the fracture. This portion of bone is known as the posterior ledge. The implant must rest on this bony structure into order to remain securely in place. Since the posterior ledge is towards the backside of the orbital aperture, it can take surgeons multiple attempts before they feel confident that the implant is resting on the posterior wall as they struggle to see its location through the relatively small incision and following dissection. Additionally, finding the posterior ledge can take up a significant portion of operating time.

2.4 Need

This relative operative blindness is an indication of a clear need for improved surgical navigation and visualization techniques specific for orbital floor reconstruction. An augmented reality navigation system for orbital floor reconstruction surgeries is proposed to resolve the problem of low visibility of the distal orbital wall during the procedure, so that misplacement may be avoided. The introduction of a head mounted display to provide navigation to surgeons in the orbital floor implant process would reduce operation time and increase surgeon confidence in secure implant placement.

2.5 Current Surgical Workflow

Current surgical process is made up of three core phases. The surgical process begins with a dissection along the orbital bone in order to access the fracture area and the orbital aperture. The eye is pulled upward in order to get clear access to the orbital floor or medial wall. Once access is established, the fracture cavity is examined, and herniated tissue is slowly returned to its proper location. This process allows the fracture cavity to be exposed. The end of the first phase is characterized by the clearing of the fracture cavity and the removal of pieces of fractured bone.

The second phase of the surgery involves the stabilization of the orbital aperture by the introduction of an orbital plate. This phase of the surgery is an arduous process, taking up a significant portion of operation time. First, the orbital implant is shaped to the patient's specific anatomy by the surgeon's intuition of the operative field from the completion of phase 1. Once the surgeons feel as if the implant is in the correct shape, they will attempt implantation. It is very rare for the implant placement to be perfect on the first try. The surgeon will take multiple attempts to shape the implant correctly and then place the implant in its proper place. This phase poses the most challenges to the surgeon. Implant placement is difficult due to the nature of the dissection and tight, compact space. Additionally, the presence of delicate, complex anatomy in the area adds additional complexity to an already complicated process. The use of a navigation for this portion of the surgery would decrease operation time used for searching for the posterior ledge and increase surgeon confidence in implant placement.

Phase 3 of the operation begins after the surgeon is convinced with their placement of the orbital implant and its shape. Afterward, the surgeon will return the eye to its proper location and test eye mobility. Due to the invasive nature of the procedure in the eye socket, it is important to ensure that all ocular muscles are in their correct location and that the eye is able to move properly. Once the eye mobility is checked, the dissection and incision are sutured and closed up.

3 Deliverables

The deliverables for this project fall into three main categories of achievement; Registration, Calibration and Visualization.

Point/Surface Registration Method for Orbital Socket The following are descriptions of camera-based tracker accuracy.

- *Minimum*: Target Registration Error of less then 4 mm.
- *Expected*: Target Registration Error of less then 3 mm.
- *Maximum*: Target Registration Error of less then 2 mm.

Calibration of Implant with Respect to Tracked Hemostat

- *Minimum*: Pivot calibration of the distal edge of the implant. Only modeling distal edge.
- *Expected*: Use calibrated pointer to model the implant distal edge. Only modeling distal edge.
- *Maximum*: Use calibrated pointer to model the entire implant.

Visualization of Tracked Implant with Respect to CT

- *Minimum*: Visualization on 3D-Slicer via OpenIGTLink.
- *Expected*: Visualization in augmented reality system like HoloLens or Unity.
- *Maximum*: Comparison between 3D-Slicer implementation and HoloLens implementation.

4 Results

Compared against the deliverables, the results of our project is presented in this section.

4.1 Registration

Two anatomic features are selected to calculate the target registration error: temporal bone tip and the intersection of the zygomatic bone and superaorbital margin. The CT model (converted to left handedness) coordinates for these two anatomic features can be shown in Figure 4 and 3. The TRE for fiducial-based registration on these two anatomic features are calculated to be 1.6366 mm and 1.0700 mm, respectively.

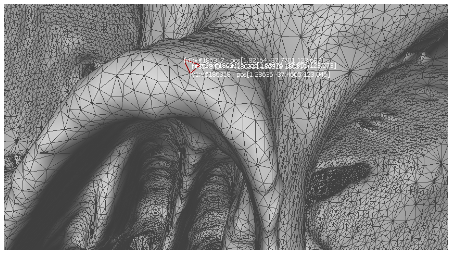


Figure 3: The coordinate of the temporal bone tip on the left-handed CT model

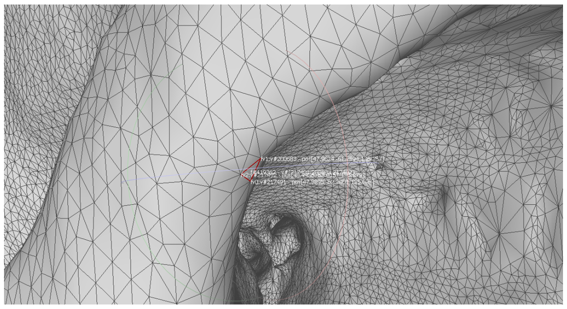
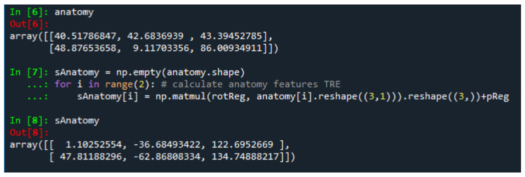


Figure 4: The coordinate of the intersection of the zygomatic bone and superaorbital margin on the left-handed CT model

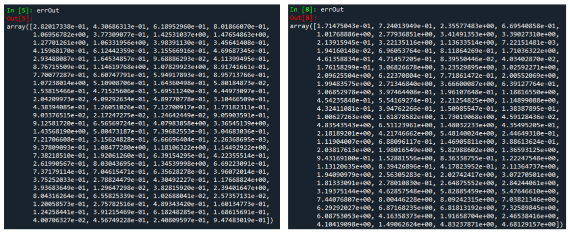

```
In [6]: anatomy
Out[6]:
array([[40.51786847, 42.6836939 , 43.39452785],
       [48.87653658,  9.11703356, 86.00934911]])

In [7]: sAnatomy = np.empty(anatomy.shape)
   ...: for i in range(2): # calculate anatomy features TRE
   ...:     sAnatomy[i] = np.matmul(rotReg, anatomy[i].reshape((3,1))).reshape((3,))+pReg

In [8]: sAnatomy
Out[8]:
array([[  1.10252554, -36.68493422, 122.6952669 ],
       [ 47.81188296, -62.86808334, 134.74888217]])
```

Figure 5: The target coordinate of the two anatomic features

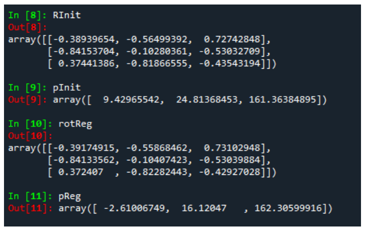
The average registration residual is calculated to be 0.63 mm and 2.33 mm for fiducial-based registration and fiducial-initialized ICP registration, respectively.


```
In [5]: errOut
Out[5]:
array([2.82017338e-01, 4.30686313e-01, 6.18952960e-01, 8.01866070e-01,
       1.06956782e+00, 3.77309077e-01, 1.42531037e+00, 1.47654863e+00,
       1.27701261e+00, 1.06331956e+00, 3.98391130e-01, 3.45641408e-01,
       4.15968170e-01, 6.12442359e-01, 3.15566916e-01, 4.69687345e-01,
       2.93488087e-01, 1.64534857e-01, 9.68886293e-02, 4.11399495e-01,
       8.76715509e-01, 1.14619768e+00, 1.07829923e+00, 8.91741661e-01,
       7.70077287e-01, 6.60747791e-01, 5.94917893e-01, 8.95713766e-01,
       1.07238014e+00, 5.10908706e-01, 1.64360498e-01, 5.80184873e-02,
       1.53815466e-01, 4.71525606e-01, 5.69511240e-01, 4.44973097e-01,
       2.04209973e-02, 4.09292634e-01, 4.89770778e-01, 3.10466509e-01,
       4.38394085e-01, 1.26051026e-01, 7.12700917e-01, 1.73182311e-01,
       9.03376515e-02, 2.17247275e-02, 1.24642449e-02, 9.05903591e-01,
       9.12581720e-01, 6.56569724e-01, 4.07983858e+00, 3.36545139e+00,
       1.43568190e+00, 5.80473187e-01, 7.39682553e-01, 3.04683036e-01,
       7.21706608e-01, 3.15624828e-01, 6.66696404e-01, 2.26368695e-03,
       9.37809093e-01, 1.08477280e+00, 1.18106322e+00, 1.14492922e+00,
       7.38218510e-01, 1.92061260e-01, 6.39154295e-01, 4.22355514e-01,
       2.61990567e-01, 8.03043695e-01, 1.34539998e+00, 6.69223091e-01,
       7.37179114e-01, 7.04615471e-01, 6.35628278e-01, 3.96072014e-01,
       3.75252033e-01, 2.78824479e-01, 4.30492227e-01, 1.17668824e+00,
       3.93683649e-01, 1.29647298e-02, 3.82815920e-01, 2.39401647e+00,
       8.04316264e-01, 6.55825339e-01, 1.02688041e-02, 2.57357131e-02,
       1.20058573e-01, 2.75782516e-01, 4.89343420e-01, 1.60134773e-01,
       1.24258441e-01, 3.91215469e-01, 6.18248285e-01, 1.68615691e-01,
       4.00706327e-02, 4.56749228e-01, 2.40809597e-01, 9.47483019e-01])
```

```
In [8]: errOut
Out[8]:
array([1.71475043e-01, 7.24013949e-01, 2.35577483e+00, 6.69540858e-01,
       1.01768886e+00, 2.77936851e+00, 3.41491353e+00, 3.39027310e+00,
       2.13915945e-01, 3.22135116e+00, 1.13633514e+00, 7.22151481e-03,
       1.94160148e-02, 6.96053764e-01, 8.11864269e-01, 1.71036322e+00,
       4.61358834e-01, 4.71457205e-01, 8.39550446e-02, 4.03402870e-02,
       1.76158299e-01, 3.06826678e+00, 3.23529895e+00, 3.02592271e+00,
       2.09625504e+00, 6.22370804e-01, 7.71861472e-01, 2.00552069e+00,
       1.99483575e+00, 2.71346840e+00, 3.66600087e+00, 6.39127764e-01,
       3.06852978e+00, 3.97464408e-01, 1.96107648e-01, 1.18816550e+00,
       4.54235848e-01, 5.54169274e-01, 2.21254825e+00, 1.14899088e+00,
       4.32411081e-01, 3.94762266e-01, 1.50985547e-01, 1.38387895e-01,
       1.00627263e+00, 1.61878582e+00, 1.73019068e+00, 4.59128436e-02,
       4.83543543e+00, 6.51123961e+00, 1.48032233e+00, 4.35495205e-01,
       2.18189201e+00, 4.21746662e+00, 5.48140024e+00, 2.44649310e-01,
       1.11904007e+00, 6.88096117e-01, 1.46905811e+00, 3.88613624e-01,
       2.03817613e+00, 1.98016549e+00, 5.82988602e+00, 1.36593125e+00,
       9.43169100e-01, 1.52881556e+00, 8.36338755e-01, 1.22247548e+00,
       1.13120635e+00, 8.39426896e-01, 4.17823952e-01, 2.11364737e+00,
       1.94090979e+00, 2.56305283e-01, 2.02742417e+00, 3.07270501e+00,
       1.81333091e+00, 2.78010830e-01, 2.64875552e+00, 2.84244061e+00,
       3.19375144e+00, 4.62857548e+00, 5.82885459e+00, 5.47646610e+00,
       7.44076807e+00, 8.00446228e+00, 8.09242315e+00, 7.03821346e+00,
       6.29292027e+00, 6.87168235e+00, 6.81813192e+00, 7.32589845e+00,
       6.08753053e+00, 4.16358373e+00, 1.91658704e+00, 2.46538416e+00,
       4.10419098e+00, 1.49062624e+00, 4.83237871e+00, 4.68129157e+00])
```

Figure 6: The registration residual. Left: fiducial-based; Right: fiducial-initialized ICP

Although the average residual of ICP registration was low, the TRE was tested to be high. We conclude that it did not converge for the points we collected specifically. By comparing the derived transformations of fiducial-based registration and ICP, we found that rotation matrix did not differ too much but the position offset changed by a lot (around 10 mm in both x axis and y axis, 1 mm in z aixs) as shown in Figure 7. We conclude that the ICP made the fiducial-based registration worse. However, the fiducial-based registration works well in our case.


```
In [8]: RInit
Out[8]:
array([[-0.38939654, -0.56499392,  0.72742848],
       [-0.84153704, -0.10280361, -0.53032709],
       [ 0.37441386, -0.81866555, -0.43543194]])

In [9]: pInit
Out[9]: array([  9.42965542,  24.81368453, 161.36384895])

In [10]: rotReg
Out[10]:
array([[-0.39174915, -0.55868462,  0.73102948],
       [-0.84133562, -0.10407423, -0.53039884],
       [ 0.372407  , -0.82282443, -0.42927028]])

In [11]: pReg
Out[11]: array([ -2.61006749,  16.12047   , 162.30599916])
```

Figure 7: The derived transformations of both fiducial-based and ICP

4.2 Calibration (Digitization of the Implant)

The implant used to validate the calibration process of the system was an orbital floor implant sample from the manufacturer, DePuy Synthes. To complete the calibration process, points were captured along the sides and on the edges of the grids within the implant. These points were captured using the tracked pointer tool of the Polaris system. Prior to this step, the implant was securely clamped by the hemostat, as shown below.

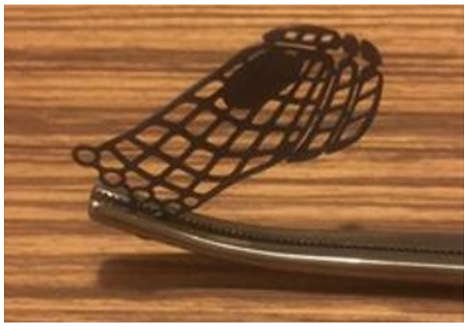


Figure 8: Depuy Synthes Orbital Floor Implant Clamped by Hemostat

After points were captured, the data was then parsed by the calibration code and digitized. Using a number of python packages including *pyvista [8], pymeshfix, and vtk [7]*, a surface mesh of triangles was created from the digitized points. It was observed that the greater number of points collected resulted in a more accurate surface mesh. The accuracy of the implant model is a key portion of making the system functional. Though point collection can take 1 to 2 minutes, it is paramount that a large number of points be collected to ensure that the model is as accurate as possible.

Once a surface mesh is made, a thickness factor is added to the mesh. First, the surface mesh is extruded using the *vtk* package. It is important to note that this is thickness must be measured from the implant directly in order to maintain the best accuracy. Once this thickness is added, *pymeshfix* is used to rid the new mesh of any potential non-manifold faces. This step is crucial in ensuring that the mesh not have any faces that cannot exist in the real world. This includes faces within faces and other triangles in the mesh that are intersecting each other. After the mesh has been rid of these deformities, it is then tetrahedralized to give it a 3D shape. This step is crucial to ensure that the implant is visible in the visualization portion of the system. A plain surface mesh will not be visible in the visualization due to the fact that has no thickness.

4.3 Visualization

The visualization is done by Unity webcam application. It can also be compiled to be a HoloLens application so the user can use the system in head-mounted display. The skull model overlay is successfully made while the implant motion is not stable, although the static pose and position was tested to be correct. This can be solved by a better moving method in Unity frame update. The skull model overlay can be seen in Figure 10.

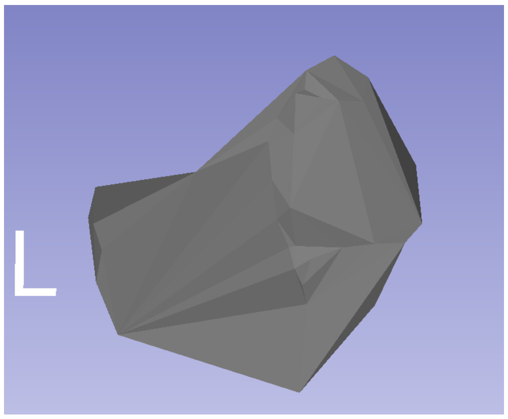


Figure 9: Final 3D Accurate Implant Model

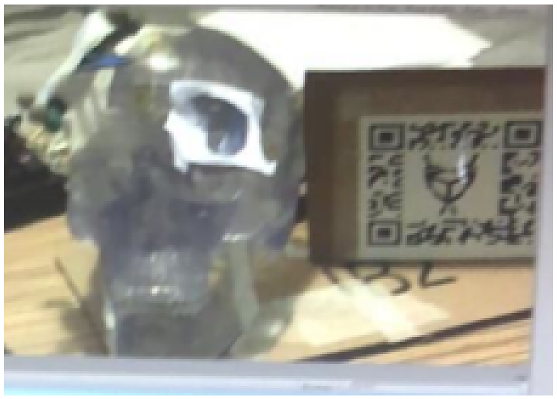


Figure 10: The skull model overlay and the static implant

5 Methods

The experiment is set up as the Figure 11 shows. The system has a Northern Digital Inc. Polaris optical tracking system which provides high accuracy in tracking of other components. There are three tracked tools: one attached to the physical skull model, another attached to the hemostat that holds the implant, and a tracked pointer. The detailed procedure and theory are discussed in the following sections.

5.1 Transformation Map

The transformation map can be viewed in the Figure 13. The three unknown transformation relations are the transformation from the Unity/HMD space to the physical skull space, the transformation from the tracked AR marker to the physical skull space, and the transformation from CT space to the physical skull space. The Python server is implemented to solve these transformations. Iterative closest point algorithm can be used to get the transformation from CT to skull. And the AR marker can be tracked by Vuforia virtual reality engine that is integrated in Unity. To get the transformation from AR marker to the skull, we simply used a point cloud registration by getting the four corners of the AR marker in skull space to the desired corresponding points in virtual space.

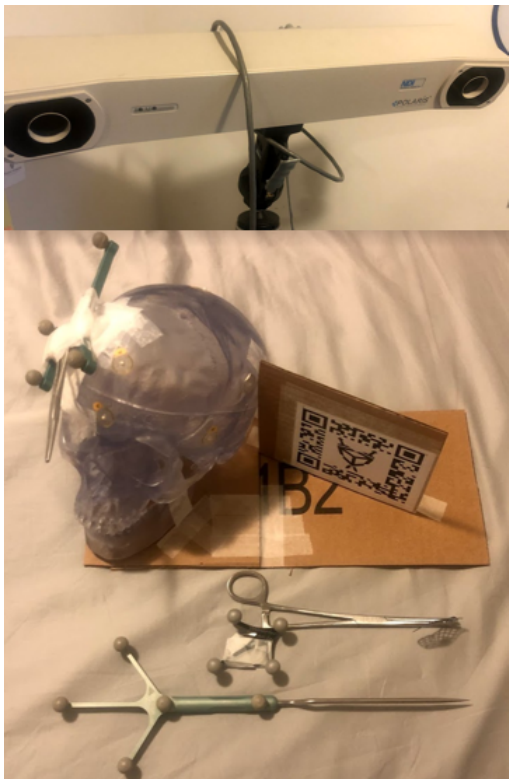


Figure 11: Hardware and experiment setup

5.2 System Components and Main Procedure

The system consists of a server in Python and a client in Unity. The communication is done by UDP protocol. The Figure 12 depicts the composition of the system.

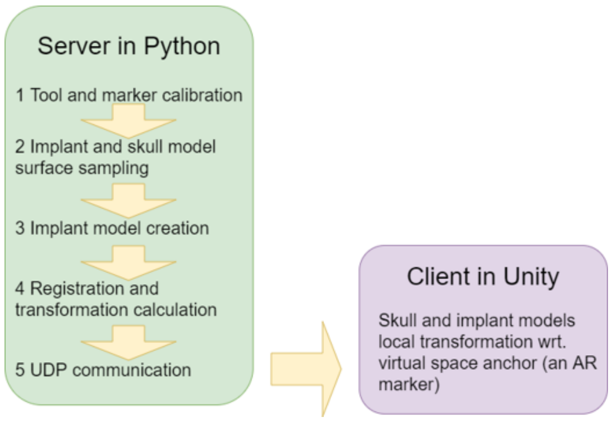

Server in Python
1 Tool and marker calibration
2 Implant and skull model surface sampling
3 Implant model creation
4 Registration and transformation calculation
5 UDP communication
Client in Unity
Skull and implant models local transformation wrt. virtual space anchor (an AR marker)

Figure 12: System components diagram

The user procedure includes five steps that was showed in the Figure 12. The calibration method is pivot calibration. Then, the model sampling method allows the user to specify frequency and the number of the points the user needs to take. The sampling method also allows the user to specify which tracked tool to be the base coordinate system. After that, the implant model is created by Delaunay triangulation. In order to give the implant surface a thickness, we can use the Uniform Mesh Resampling in Meshlab, or using the *pymeshfix* library available in Python. The registration and transformation calculation is then carried out based upon the transformation map that was discussed in Section 5.1 Transformation Map. Among these steps, handedness handling should be made. There are two different methods to handle handedness which will be discussed in Section 5.5.1 Left Handedness. Now the transformation (the pose of the skull and implant model) be can send to the Unity client for visualization.

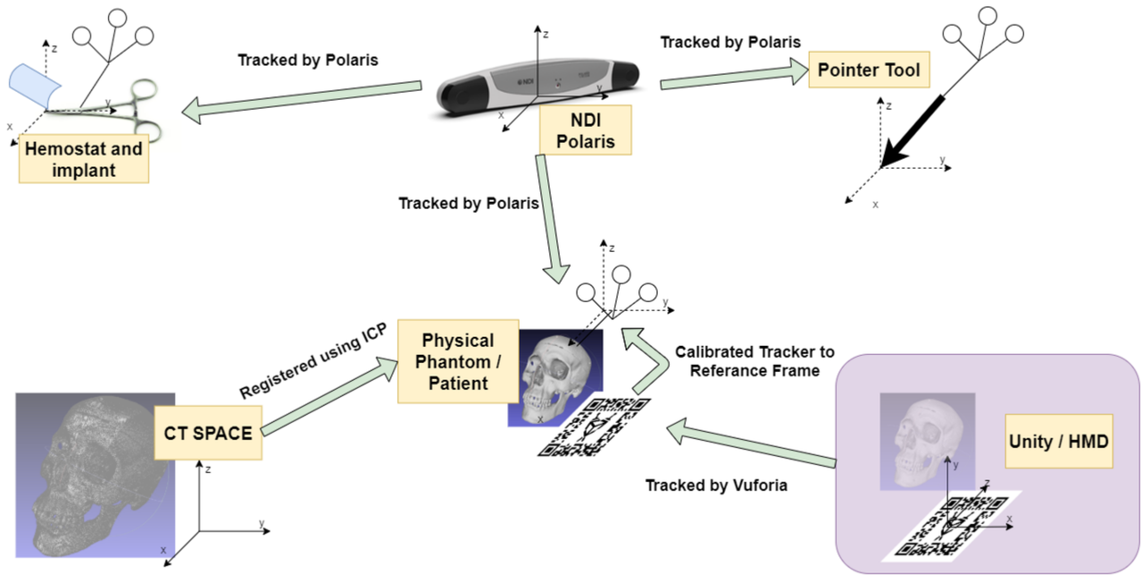


Figure 13: The transformation map used in this project

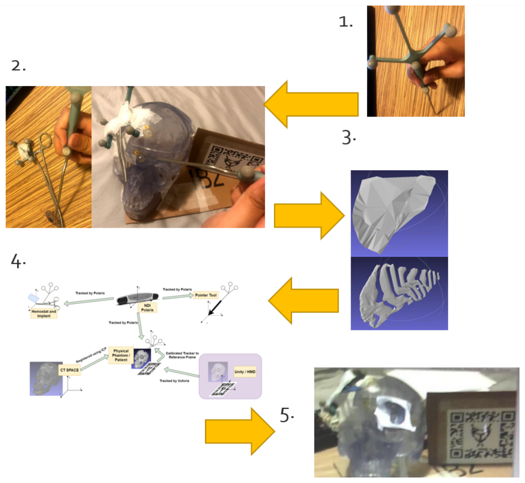


Figure 14: The main procedure in the system server

5.3 Polaris Interface

We used the older version of Polaris optical tracker by Northern Digital Inc.

In this project, the raw Polaris data collection hardware interface used is the *scikit surgerynditracker* library developed by Wellcome EPSRC Centre for Interventional and Surgical Sciences at University College London. It is a light-weighted package that is easy to customize. There are two known issues when working with this package. One is that on Windows development environment, it only properly installs under Python 3.6. However, Python 3.7 works for both Linux and MacOS development environment. The other problem is that an *UnicodeDecodeError* is reported when using 3 tracked bodies and the system uses “BX Transforms”, which can be fixed by changing source code in *nditracker.py* to “TX Transforms”. The same problem also happens when using 1 tracked body and the system uses ”TX Transforms”. Details can be seen in the github issue link: [5].

To customize the interface, a *polarisUtilityScript.py* is created. This script is located in the *serverPython/util* folder of the project repository. To improve code reusability, the point collection method is made to be able to collect point directly with respect to another tracked tool. This largely decreases the transformation calculation procedure. Details can be seen in the *polarisUtilityScript.py* script.

5.4 Iterative Closest Point Registration

The iterative closest point (ICP) algorithm is used for finding the transformation between the CT model and the physical model. In this project, we converted the Matlab code from the programming assignment 4 of the Computer Integrated Surgery I to Python code. The original Matlab version Github repository is here: [3]. The Python code is integrated in the *serverPython* folder of this project.

The search method of ICP is octree search. The ICP convergence largely depends on the coverage of the sampled points. However, more points will result in a very long processing time, which would be a drawback of the clinical use. Moreover, some points may not practically accessible, because only the eye socket area is exposed.

In our implementation, the ICP is initialized by a fiducial-based registration. In the practical use, these fiducials can be some anatomical features. Then, the registration is refined by ICP.

5.5 Noteworthy Solutions

There were some challenges posed during the implementation of the project. Most challenges have been handled but the visualization still has issues that are not solved.

5.5.1 Left Handedness

Unity uses a convention of left handedness, while all the other coordinate systems (Polaris, CD models etc.) are in right handedness. This problem is solved by two different methods, and there are two different branches in the Github repository. The "master" branch handles the handedness by "Calculate, convert, send", while the "lefthanded" branch handles the handedness by "Convert, calculate, send". More specifically, the "Calculate, convert, send" method means to calculate the transformations under assumption that all the coordinate systems are in right handedness, then only the desired results are converted to lefthandedness and sent to Unity client. On the other hand, the "Convert, calculate, send" methods corresponds to converting all the coordinate system to lefthandedness initially and then conduct procedures needed. In "Convert, calculate, send" method, the skull model used in ICP is flipped along x-axis to match the Unity. In the Polaris collection, all the x, y, and z are flipped to have a left handed coordinate while not changing the rotation matrix.

5.5.2 Vuforia Tracking

Vuforia is a augmented reality engine that is integrated in Unity. One important feature in Vuforia is the image tracking feature. Vuforia tracks an image and estimate the pose of the image. We utilize this feature to complete our transformation chain. However, sometimes the Vuforia engine takes a flipped pose of a horizontal marker. The effect can be seen in Figure 15. We found Vuforia is more capable to successfully calculate a vertical AR marker pose.

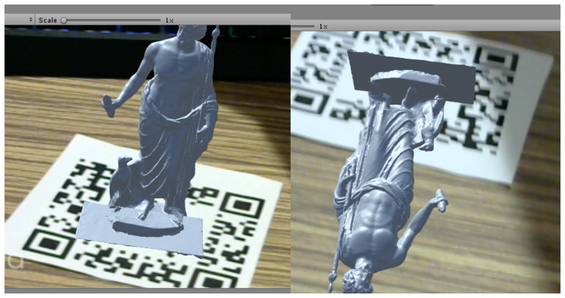


Figure 15: The wrong tracking of Vuforia

5.5.3 Non-Manifold Implant Issue

After parsing the point calibration data collected from the tracker pointer on the implant clamped to the hemostat, a surface mesh could be created of the digitized points using the *pymesh* package however, this model would not appear in the visualization. After inspection, it was determined that thickness would need to be added to the surface mesh. This proved to be difficult as a simple extrusion of the surface mesh would simply yield another surface mesh. In order to tetrahedral the surface, several packages were attempted to be used to accomplish the task, yet all yielded the same error, *Cannot tetrahedralize non-manifold mesh.* After researching the definition of a manifold mesh, it was difficult to determine a remedy to the issue without having to load the surface mesh into a computer-aided design software to remove all non-manifold vertices within the model. However, after some more research, it was determined that it could be done within the code using the *pymeshfix* package. Unfortunately, after this step, the tetrahedralized mesh turned out to not be Representative of the actual geometry. Up till this point thickness was only added in one cardinal direction. After some experimentation, it was discovered that in order to have the most accurate 3D model, thickness must be added in all three cardinal directions.

5.5.4 Visualization Issue

Although the transformation can be accurately calculated, the accuracy is still limited by the Vuforia image tracking. To improve the accuracy of the model overlay, a calibration procedure may need to be done. A calibration work done by Azimi et al [1] can be a solution. This calibration procedure requires a tracked head mounted display.

6 Accomplished vs Planned

6.1 Registration

Fortunately, the **maximum** deliverable declared at the beginning of the project was achieved by fiducial-based registration. Through the use of a robust communication system and interface with the NDI Polaris optical tracking system, we were able to efficiently collect tracking data and implement an effective iterative closest point algorithm. This iterative closest point algorithm, initialized with fiducial registration, provided us with a highly accurate registration procedure, with a target registration error of around 1 mm. This is elaborated in the result section.

6.2 Calibration

Of our set deliverables for the calibration portion of this project, we achieved the **maximum** objective. Through a robust calibration procedure, we were able to fully digitize the implant and create a highly accurate 3D model that can be used in the visualization portion of the system. The methods to develop this system are elaborated in the results section as well.

6.3 Visualization

Through significant effort, the **expected** deliverable for this portion of the project was achieved.

6.3.1 Unity Application

We were able to successfully overlay a registered portion of the skull on the skull model using a developed Unity webcam application. This application is easily transferable and compiled onto the HoloLens head mounted display. More details are provided in the results section.

6.3.2 3D-Slicer Visualization

Despite significant effort, this portion of the project was not fully accomplished. We were able to make critical head way into this portion of the visualization, but were not able to complete it due to an issue with the NDI Polaris interface with 3D-slicer.

We able to effectively visualize the skull along with the digitized implant. Additionally, through the use of the software extension, *OpenIGTLink* in conjunction with the use of an example data server, we were able to move the implant around the visual field.

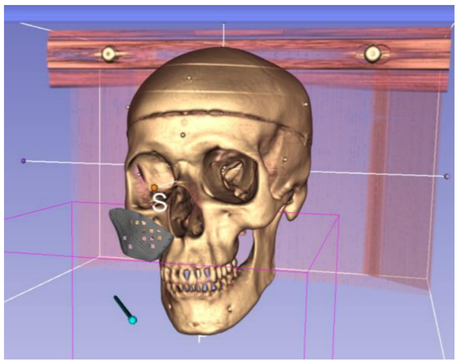
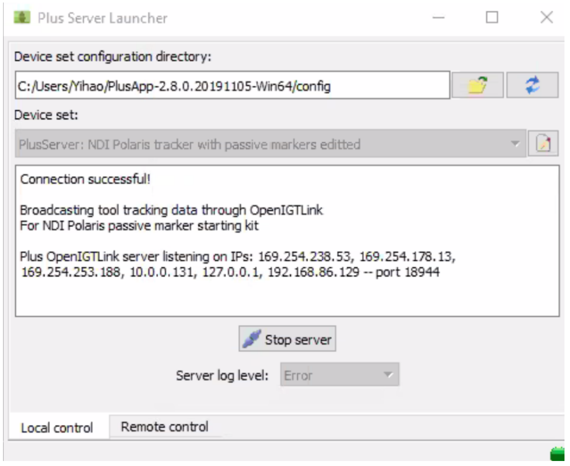


Figure 16: Skull along with tracked implant 3D-Slicer visualization from example data server.

However, the major roadblock encountered with this portion of the project was being able to send tracked maker data to 3D-Slicer. In order to accomplish this, we chose to use the *PLUS Toolkit* software interface. *PLUS Toolkit* is a software developed for providing a reliable interface between tracker technology and 3D-Slicer. The software includes the capability to interface with a number of different trackers and send the corresponding tracked information to 3D-Slicer to be visualized. In order to interface with the tracking system through *PLUS Toolkit*, it is necessary to develop a device set configuration file, to indicate to the *PLUS Toolkit* software that you would like to communicate with the tracker to provide you with data regarding your tracked markers of interest[2]. Several different parameters must be accurately defined in the file in order to ensure that interfacing with the tracked system can occur. After trying numerous different device set configuration files, we were able to develop one that was able to effectively communicate with *PLUS Toolkit* without any errors. Though *PLUS Toolkit* was not outputting any error messages, it was still not visualizing the desired tracking data in 3D-Slicer.



Device set configuration directory: C:/Users/Yihao/PlusApp-2.8.0.20191105-Win64/config

Device set: PlusServer: NDI Polaris tracker with passive markers editted

```
Connection successful!

Broadcasting tool tracking data through OpenIGTLink
For NDI Polaris passive marker starting kit

Plus OpenIGTLink server listening on IPs: 169.254.238.53, 169.254.178.13,
169.254.253.188, 10.0.0.131, 127.0.0.1, 192.168.86.129 -- port 18944
```

Server log level: Error

Figure 17: PLUS Toolkit Software indicating successful connection to NDI Polaris

To communicate any tracking data in 3D-Slicer, a software extension known as *OpenIGTLink* [9] must be installed. This extension must then be configured to receive information from outside 3D-Slicer. *OpenIGTLink* is a TCP-based communication system, in which *OpenIGTLink* can act as either a server or client. In this case, we would be using *OpenIGTLink* as a client to receive information from the local server created by *PLUS Toolkit*. *OpenIGTLink* was able to communicate with *PLUS Toolkit* and indicated in 3D-Slicer that it was receiving messages from the *PLUS Toolkit* server, however no data was being actively received by *OpenIGTLink*. We believe that *OpenIGTLink* was not the culprit in this problem, but rather *PLUS Toolkit*. We dug deeper into investigating why the interface between *PLUS Toolkit* and the NDI Polaris was not yielding any meaningful tracker data and found that *PLUS Toolkit* had established a connection with the NDI Polaris but was not receiving any valid tracking data from the system. This was further evidenced by the fact that *PLUS Toolkit* gave us no error messages, debug suggestions, and other notices regarding the connection between the two.

This was the central issue with the 3D-Slicer visualization. After considerable effort in trying to debug this issue, researching and implementing software examples and tutorials, as well as reading all the available documentation available, we reached out to the developer team at *PLUS Toolkit* to diagnose what element of the process we could have missed in establishing the interface between *PLUS Toolkit* and the NDI Polaris.

7 Future Works

7.1 Hololens Application

While achieving the assigned deliverables was an accomplishment for the semester, the primary objective for the project is to develop a functional, highly accurate augmented reality navigation system; ideally visualized through Microsoft's HoloLens. Having created a functional Unity application, it is not unrealistic to say that a functional HoloLens application is possible in the near future. Developing such an application of a head mounted display (HMD) will provide the surgeon's with the most most practical visualization method. While previous studies have shown that these systems struggle with providing accurate enough visualization for surgical procedures, from consultations with surgeons, an HMD based visualization would provide the surgeon with the most benefit in comparison to alternatives. A clear view of operative field in addition to the information provided by the system, would make for an excellent and desirable tool in the operating room with little discomfort for surgeons and lead to better patient outcomes.

7.2 3D Slicer Visualization

Having made significant headway into this part of the project, it would also not be unrealistic to continue to develop this method of visualization. As stated in previous sections, the development of this portion of the project was not to far from completion and could provide a great deal of validation to the future Hololens application by providing a direct comparison to identify the strengths and weaknesses of both systems. By developing the 3D-Slicer visualization further, a baseline standard of visualization can be established. Thus, a more empirical validation of the future HoloLens application's ability to visualize the system information can be achieved.

7.3 User Studies

The goal of any computer-integrated system is to eventually see use. As an engineer, it is very common to get lost in the technicality of a project. While such an intense focus can be beneficial in developing a functional system, other aspects of the overall system can be neglected, particularly usability. In order to ensure that the system is simple and intuitive to use for surgeons, it is imperative that user studies be conducted to empirically evaluate aspects of the system that are not user friendly or actually turn out to prevent the accomplishment of system goals. Testing the system in this manner, is a clear and effective way to ensure that the workflow of the system is sensible and therefore lead to a more functional product. Such user studies need not be done on a patient, but can be tested on an anatomically similar phantom.

7.4 Clinical Use

Once the system has been validated through user and cadaver studies, in order to see clinical use, clinical trials have to be done, along with approval from the U.S. Food and Drug Administration. Clinical trials provide a snapshot into how the system actually impacts the outcome of patients. This is the final critical validation step of developing a functional and successful product. As a primary goal of this project is to develop a computer-integrated surgical system that improves the outcome of patients and reduce operating time, clinical trials provide a snapshot to how the system can execute its goals and will be necessary to bring this system into use in the operating room.

8 Team Management

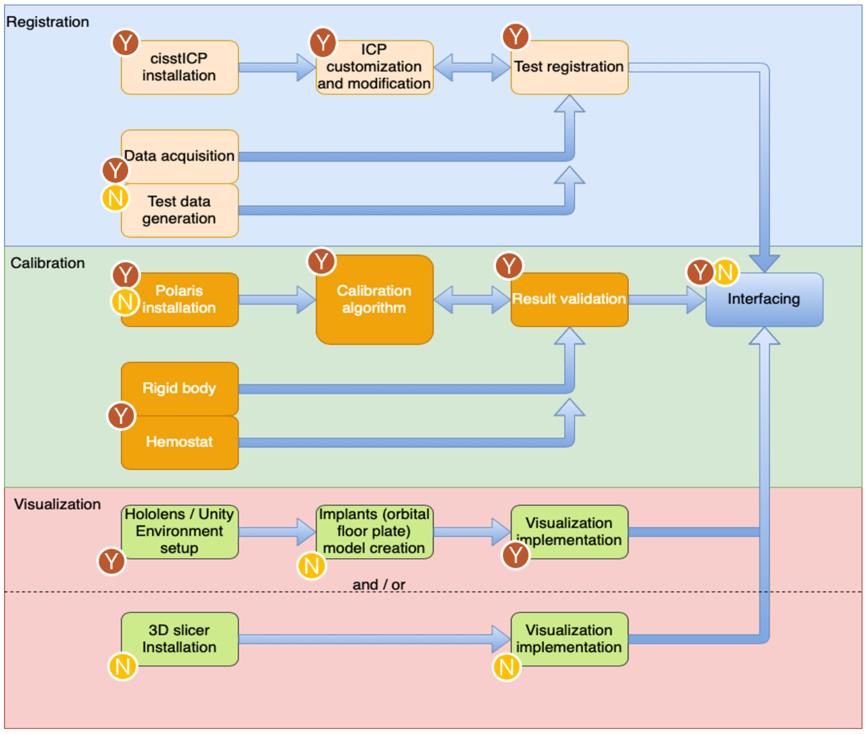


Figure 18: Breakdown of Portions of project done by each party, the yellow encircled letter N representing Nikhil Dave and brown encircled letter Y representing Yihao Liu

The figure above illustrates the portions of the project that were done by each member. The figure is broken down to reflect the amount done by each party for each of the three deliverable categories. Both parties worked in conjunction on the presentations and reports. Documentation of the system was also done in conjunction, depending on the party involved on that aspect of the project. Due to the coronavirus outbreak, the project had to be split up according to the resources available to each member. Fortunately, we were still able to achieve the declared deliverables.

9 What We Learned

Both members of the team learned a considerable amount from the implementation of this project. This was not just limited to the technical skills required to develop the project. In particular, the following lessons were learned below.

- It is incredibly important to know how to properly use the tools you are interested in using in order to be successful in developing a project. This includes software, computer languages, and hardware.

- Project planning is crucial ensuring that you are able meet your deliverables in the assigned amount of time. This includes developing an effective Gantt chart outlining your goals and the time you estimate will take to complete them. This gives you a visual tool to come back to and analyze if you are on track.

- The time spent validating and debugging your application can often take far longer then the time spend developing it.

- Proper and continual communication between team members as well as mentors is incredibly important in ensuring that issues are resolved efficiently and that all members understand wear the application stands in development.

- More often then not, through thorough investigation of possible solutions, one can find the source of their issue and resolve it without the need for help. It is important to investigate your problem thoroughly before asking for help.
- Interfacing with a tracking system is non-trivial. We spent a considerable amount of time developing a method to extract data from the NDI Polaris. Looking back, prioritizing this first would have given us more time to accomplish other goals.
- The importance of being able to pitch your application is critical. In order to illustrate the value of your project to others, it is important to have a concise pitch.
- Technical writing as well as presenting skills are hugely important in conveying what you accomplished and what your goals are.
- Documentation is critical in the development of a large project. Without proper documentation, it limits the repeatably of your achievements and can also force you to repeat development tasks/mistakes that you have already done due to forgetting your exact previous approach.

10 Acknowledgment

We would like to acknowledge our mentors Dr. Peter Kazanzides and Ehsan Azimi for continually meeting with us to answer our questions and giving us development suggestions. We would also like to acknowledge Anton Deguet for helping use the cisst libraries and also providing suggestions for the development of the project. We would like to thank our clinical mentors Dr. Cecil Qiu and Dr. Sashank Reddy, for allowing us to shadow a procedure as well as providing us with information regarding the nature of the procedure and the usability of our application. Our clinical mentors were also instrumental in providing us with the clinical material necessary to develop the project. Finally, we would like to thank Wenhao Gu for help in developing the Unity visualization application.

References

[1] E. Azimi, L. Qian, N. Navab, and P. Kazanzides. Alignment of the virtual scene to the tracking space of a mixed reality head-mounted display. *Retrieved from: https://arxiv.org/pdf/1703.05834.pdf*, 2019.

[2] Andras Lasso, Tamas Heffter, Adam Rankin, Csaba Pinter, Tamas Ungi, and Gabor Fichtinger. Plus: Open-source toolkit for ultrasound-guided intervention systems. *IEEE Transactions on Biomedical Engineering*, pages 2527–2537, Oct 2014.

[3] Y. Liu and T. Bernard. Programming assignment 4. *Johns Hopkins University Computer Integrated Surgery, https://github.com/bingogome/ICP_Matlab*, 10 2019.

[4] D. Langer M. Paul. Orbital floor fractures - eyewiki. *American Academy of Ophthalmology, Available: https://eyewiki.aao.org/Orbital_Floor_Fractures*, 2 2020.

[5] University College London Github repository. Unicodedecodeerror solution. *Retireved from: https://github.com/UCL/scikit-surgerynditracker/issues/17*, 2020.

[6] M. Yonkers S. Grob and J. Tao. Orbital fracture repair. *Seminars in Plastic Surgery, vol. 31, no. 1, pp. 31-39*, 2017.

[7] Ken; Lorensen Bill Schroeder, Will; Martin. The visualization toolkit (4th ed.). 2006.

[8] C. Bane Sullivan and Alexander Kaszynski. PyVista: 3d plotting and mesh analysis through a streamlined interface for the visualization toolkit (VTK). *Journal of Open Source Software*, 4(37):1450, may 2019.

[9] Junichi Tokuda, Gregory S. Fischer, Xenophon Papademetris, Ziv Yaniv, Luis Ibanez, Patrick

Cheng, Haiying Liu, Jack Blevins, Jumpei Arata, Alexandra J Golby, Tina Kapur, Steve Pieper, Everette C Burdette, Gabor Fichtinger, Clare M Tempany, and Nobuhiko Hata. Openigtlink: An open network protocol for image-guided therapy environment. int j med robot. *Int J Med Robot*, 5(4):423–34, 2009 Dec 2009.