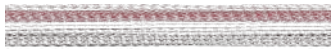
HIPSTER

# **A**utonomous **R**eaming for **T**otal **H**ip **R**eplacement **(ARTHuR)**

# Progress Review - 1

Team C: Kaushik Balasundar, Parker Hill, Anthony Kyu, Sundaram Seivur, Gunjan Sethi

16 February, 2022



# Schedule

| Schedule | | | |
|---|---|---|---|
| **Identifier** | **Capability Milestone(s)** | **Associated Tests** | **System Requirements** |
| **Progress Review 1**<br>**2/16/2022** | - Test camera health and camera discovery via a ROS Node. | Test 1 | M.F.1 |
| **Progress Review 2**<br>**3/2/2022** | - Markers can be read into ROS and create a point cloud comparable to the pelvis geometry<br>- Control method is capable of being used with robot manipulator virtually | Test 2<br>Test 3<br>Test 4<br>Test 13 | M.F.1<br>M.F.3<br>M.N.1 |
| **Progress Review 3**<br>**3/23/2022** | - Probe is able to be used to create a point cloud which can be visualized<br>- Waypoint and trajectory generation working in ROS<br>- Hardware verified for use in reamer assembly | Test 5<br>Test 11<br>Test 17 | M.F.1<br>M.F.2 |

# Progress Review #1 Tests

- ✓ Atracsys SDK Setup — Perception and Sensing
- ✓ Camera Setup Test — Perception and Sensing
- ✓ Marker Pose Detection — Perception and Sensing
- ✓ Documentation — Perception and Sensing

# Further Updates

- ✓ Registration — Perception and Sensing
- ✓ Controls — Planning and Controls
- ✓ Simulation — Planning and Controls
- ✓ Hardware — Hardware

# PR #1 Tests

# Motivation and Goal

**Motivation:** Cameras are generally shipped with ROS packages that allow the camera to be easily interfaced with ROS Nodes. With the Atracsys camera, we are only provided with an SDK that can be used to retrieve measurements.

**Goal:** Read camera measurements from a ROS Node. The outcome of this task is merely a proof-of-concept and further optimizations/improvements in the code will follow.

Figure: Atracsys SpryTrack 300

# **Atracsys SDK Setup:** Overview

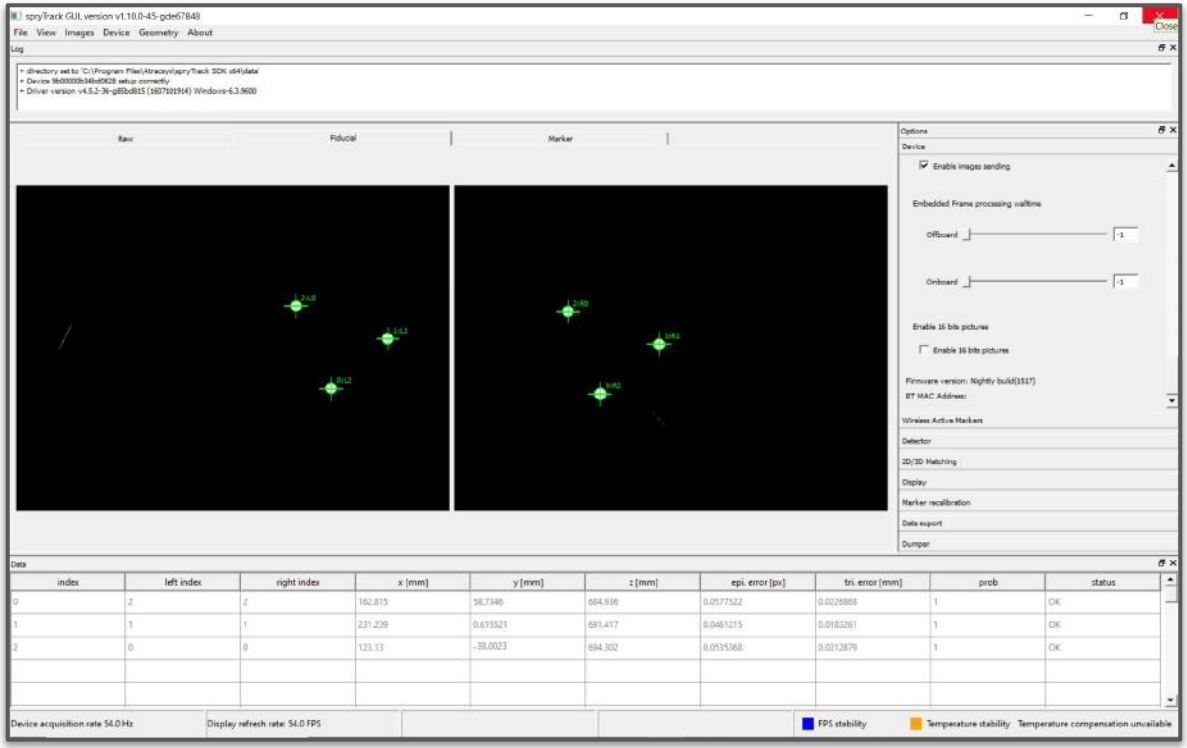



Figure: Atracsys Demo GUI

# **Camera Setup Test:** Overview

| Test 1: | |
|---|---|
| Camera Setup Test | |
| **Objective** | |
| Test camera health and camera discovery via a ROS Node. | |
| **Equipment** | MRSD Desktop 2, Atracys SpryTrack 300 Camera |
| **Elements** | Perception Subsystem |
| **Personnel** | Gunjan Sethi |
| **Location** | NSH Basement |


Figure: Atracsys Camera Setup

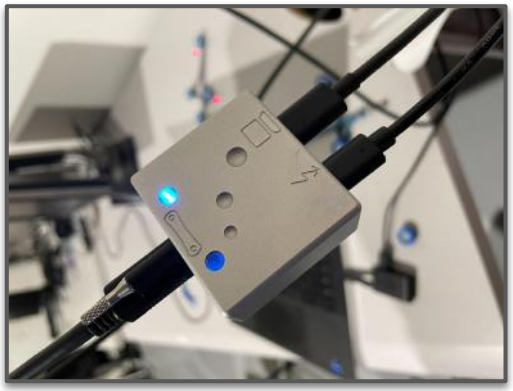


Figure: Power Injector for Camera

# **Camera Setup Test:** Approaches

| SNo. | Approach | Pros | Cons |
|---|---|---|---|
| 1 | Use ROS-IGTL-Bridge | - Well tested option | - Tested on an older ROS version.<br>- Little documentation on compatibility with Atracsys. |
| 2 | Run Atracsys SDK as a standalone application and communicate with ROS via Sockets | - Sufficient documentation available online | - New to socket programming, steep learning curve. |
| 3 | Link Atracsys Library Files with ROS Node using CMake | - Can be implemented with prior C++/CMake experience<br>- Simplifies the Perception subsystem architecture<br>- No need to integrate | - CMake is complicated and hard to get right<br>- If a new version of the SDK is released, static linking may require our codebase to be recompiled. |

# **Camera Setup Test:** Procedure and Setup

**Procedure**

1. Run the camera_node ROS Node and wait for the node to discover the camera.
2. Wait for the ROS Node to load the geometry file.

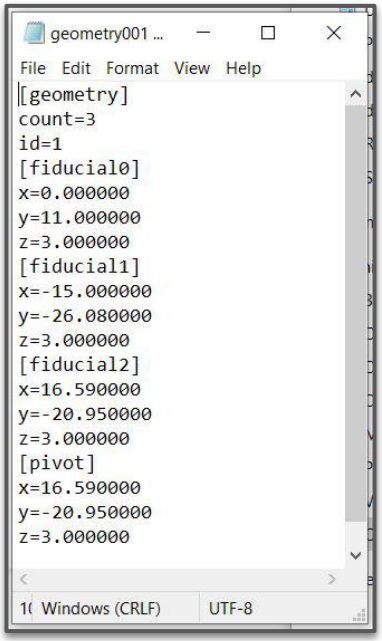



Figure: Geometry File

Figure: Marker with 3 Fiducials

# **Camera Setup Test:** Validation

**Validation**

- Camera's serial number is printed. Validation #1
- Geometry file is loaded. Validation #2

```
gsethi2409@gsethi2409:~/catkin_ws$ rosrun atracys_publisher camera_node
Running camera node
Detected one sTk 180 with serial number 0x9b00000b34bd0828
Enable onboard processing
Disable images sending
Loading geometry 1, composed of 3 fiducials
Loaded fiducial 0 (0, 11, 3)
Loaded fiducial 1 (-15, -26.08, 3)
Loaded fiducial 2 (16.59, -20.95, 3)
................................
```

Validation #1

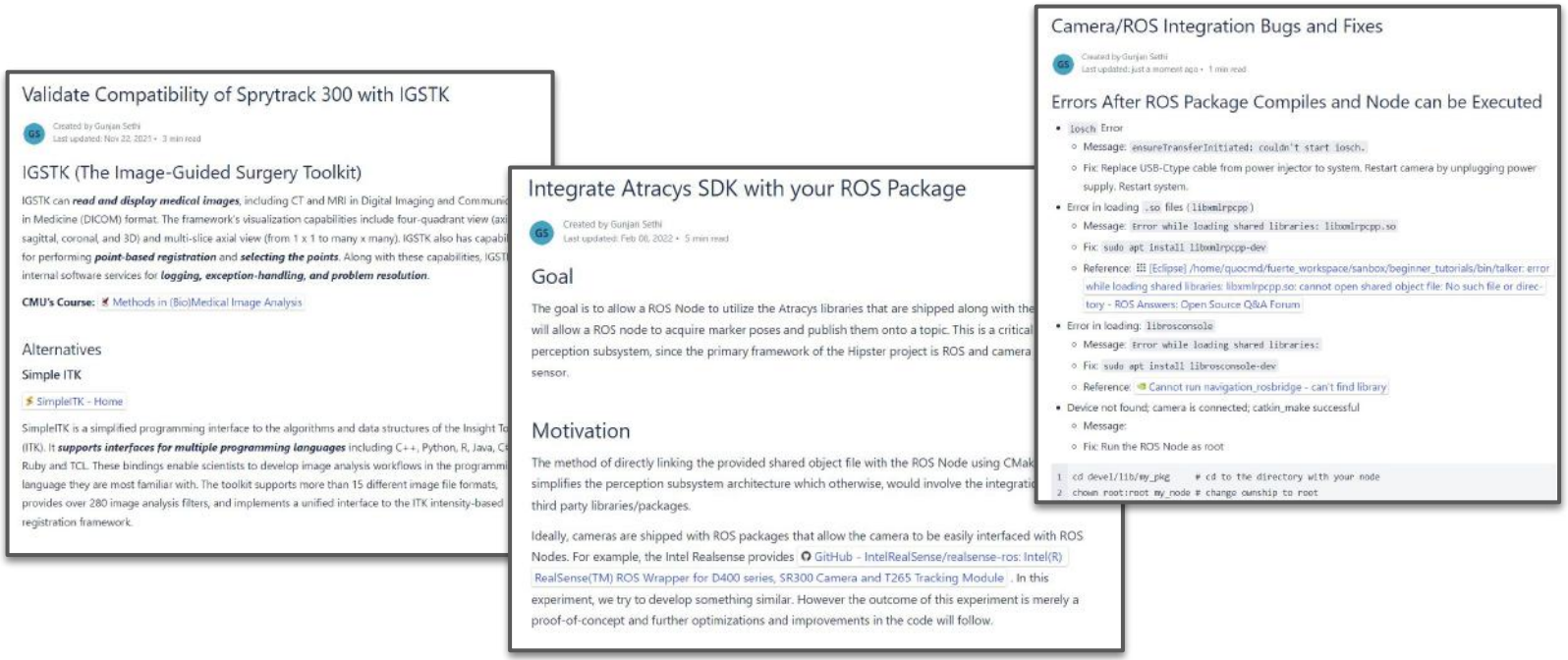
Validation #2

# **Camera Setup Test:** Challenges

- Challenge 1. Compilation Errors
  - ✓ Problem in compiling Atracsys SDK source files along with ROS Nodes. Switched to direct linking of library files.
- Challenge 2. Undefined References
  - ✓ Resolved by re-evaluating linking of files, correcting path errors, changing directory permissions.
- Challenge 3. Unclear of the differences between various library files in C++ (.a and .so)
  - ✓ Resolved by referring to tutorials and discussion forums online.

# **Camera Setup Test:** Documentation

3 major documents created.



Validate Compatibility of Sprytrack 300 with IGSTK

Created by Gunjan Sethi
Last updated: Nov 22, 2021 • 3 min read

IGSTK (The Image-Guided Surgery Toolkit)

IGSTK can ***read and display medical images***, including CT and MRI in Digital Imaging and Communic
in Medicine (DICOM) format. The framework's visualization capabilities include four-quadrant view (ax
sagittal, coronal, and 3D) and multi-slice axial view (from 1 x 1 to many x many). IGSTK also has capabil
for performing ***point-based registration*** and ***selecting the points***. Along with these capabilities, IGST
internal software services for ***logging, exception-handling, and problem resolution***.

**CMU's Course:** Methods in (Bio)Medical Image Analysis

Alternatives

Simple ITK

SimpleITK - Home

SimpleITK is a simplified programming interface to the algorithms and data structures of the Insight To
(ITK). It ***supports interfaces for multiple programming languages*** including C++, Python, R, Java, C#
Ruby and TCL. These bindings enable scientists to develop image analysis workflows in the programmi
language they are most familiar with. The toolkit supports more than 15 different image file formats,
provides over 280 image analysis filters, and implements a unified interface to the ITK intensity-based
registration framework.

Integrate Atracys SDK with your ROS Package

Created by Gunjan Sethi
Last updated: Feb 08, 2022 • 5 min read

Goal

The goal is to allow a ROS Node to utilize the Atracys libraries that are shipped along with the
will allow a ROS node to acquire marker poses and publish them onto a topic. This is a critical
perception subsystem, since the primary framework of the Hipster project is ROS and camera
sensor.

Motivation

The method of directly linking the provided shared object file with the ROS Node using CMak
simplifies the perception subsystem architecture which otherwise, would involve the integrati
third party libraries/packages.

Ideally, cameras are shipped with ROS packages that allow the camera to be easily interfaced with ROS
Nodes. For example, the Intel Realsense provides GitHub - IntelRealSense/realsense-ros: Intel(R)
RealSense(TM) ROS Wrapper for D400 series, SR300 Camera and T265 Tracking Module . In this
experiment, we try to develop something similar. However the outcome of this experiment is merely a
proof-of-concept and further optimizations and improvements in the code will follow.

Camera/ROS Integration Bugs and Fixes

Created by Gunjan Sethi
Last updated: just a moment ago • 1 min read

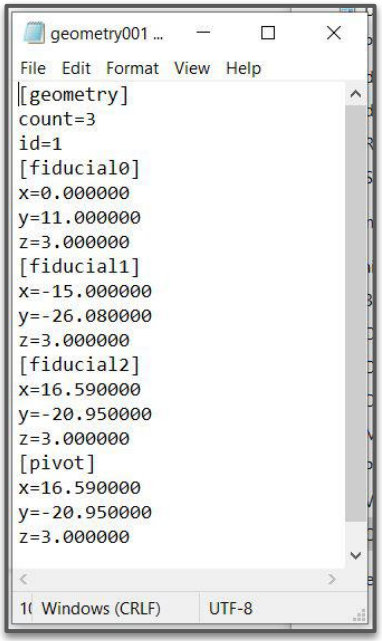
Errors After ROS Package Compiles and Node can be Executed

- `Iosch` Error
  - Message: `ensureTransferInitiated: couldn't start iosch.`
  - Fix: Replace USB-Ctype cable from power injector to system. Restart camera by unplugging power supply. Restart system.
- Error in loading `.so` files (`libxmlrpcpp`)
  - Message: `Error while loading shared libraries: libxmlrpcpp.so`
  - Fix: `sudo apt install libxmlrpcpp-dev`
  - Reference: [Eclipse] /home/quocmd/fuerte_workspace/sanbox/beginner_tutorials/bin/talker: error while loading shared libraries: libxmlrpcpp.so: cannot open shared object file: No such file or directory - ROS Answers: Open Source Q&A Forum
- Error in loading: `librosconsole`
  - Message: `Error while loading shared libraries:`
  - Fix: `sudo apt install librosconsole-dev`
  - Reference: Cannot run navigation_rosbridge - can't find library
- Device not found; camera is connected; catkin_make successful
  - Message:
  - Fix: Run the ROS Node as root

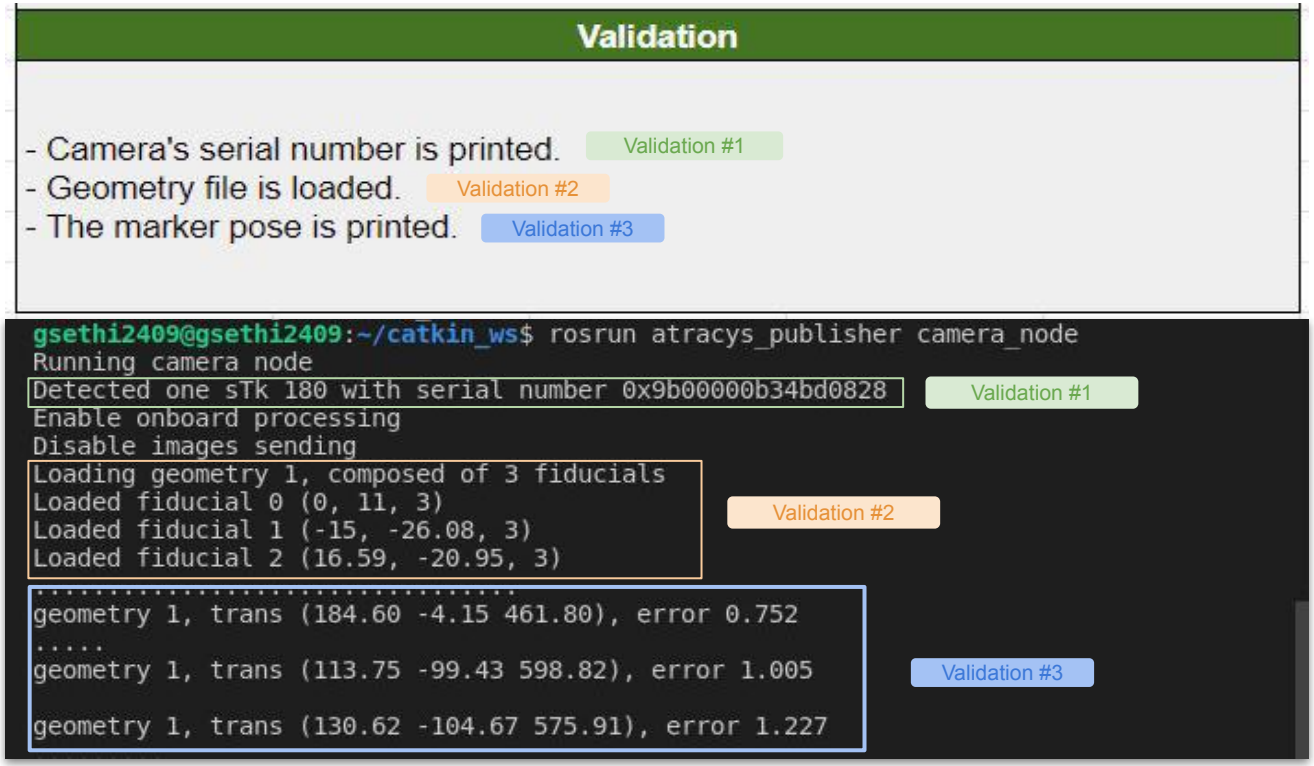
```
1 cd devel/lib/my_pkg     # cd to the directory with your node
2 chown root:root my_node # change ownship to root
```

# **Marker Pose Detection:** Overview

| Test 2: | |
|---|---|
| Marker Pose Detection Test | |
| **Objective** | |
| Test fiducial marker detection via a ROS Node. | |
| **Equipment** | MRSD Desktop 2, Atracys SpryTrack 300 Camera, Markers |
| **Elements** | Perception Subsystem |
| **Personnel** | Gunjan Sethi |
| **Location** | NSH Basement |
| **Procedure** | |
| 1. Run the camera_node ROS Node and wait for the node to discover the camera.<br>2. Wait for the ROS Node to load the geometry file.<br>3. Place markers in front of the camera. | |
| **Validation** | |
| - Camera's serial number is printed.<br>- Geometry file is loaded.<br>- The marker pose is printed. | |




# **Marker Pose Detection:** Validation



# Further Updates

# **Registration:** Overview

**Objective:**

$$\underset{R \in SO(3), t \in \mathbb{R}^3}{\arg\min} \|d(X, g(Y))\|_2^2$$

Find the rigid transformation parameters ***g*** (rotation matrix ***R ∈ SO(3)*** and translation vector ***t ∈ R3***) which best aligns the point cloud X to Y, such that the distance metric d is minimized.

**Source Type:** Cross-source

1. **Source 1:** 3D scan of Pelvis from Kinect Sensor
2. **Source 2:** Points obtained using Registration Marker

**Tools Used:** Open3D/Python

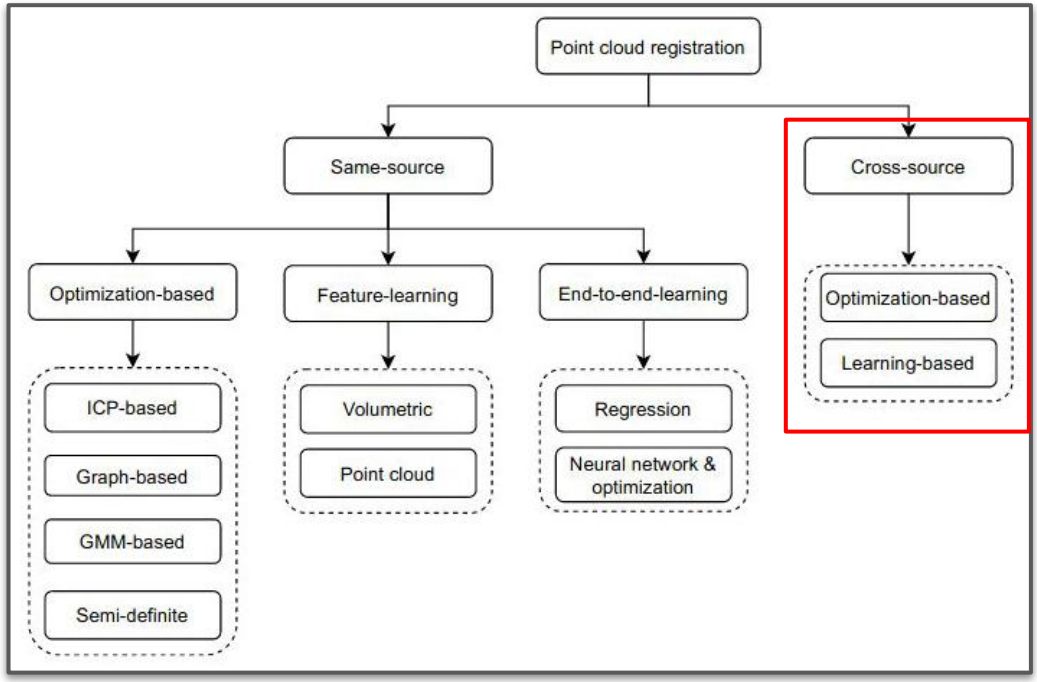



Figure: Registration Methods Overview

# Optimization Based Registration Methods

- Iterative Closest Point (ICP)
  - Widely used; extensive support available with Open3D
  - Correspondence and Transformation Estimation
  - Significant post processing for handling cross-source data
  - RANSAC used for refinement
  - Local & Global Registration
    - *Local:* Approximate Initial Transformation Provided
    - *Global:* Transformation Initialized with Identity Matrix
  - Variation based on distance metric used:
    - Minimize Point-point distance
    - Minimize Point-plane distance

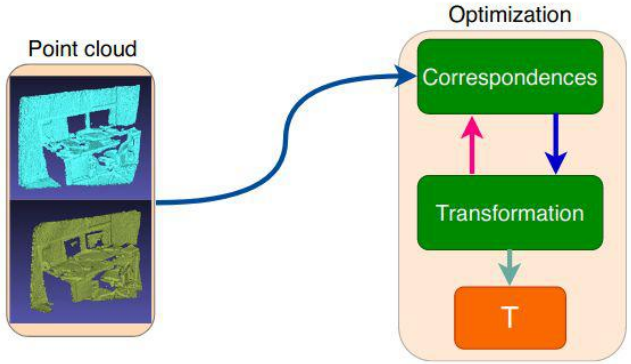



Figure: ICP Registration Overview

# Preliminary Registration Results with ICP

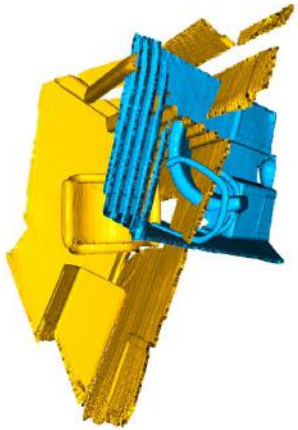


Figure: Two Pointclouds Initialized

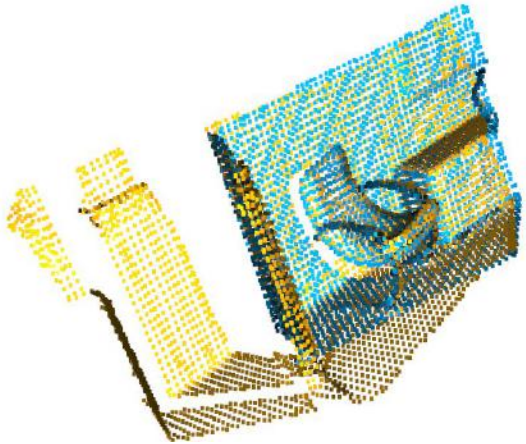


Figure: ICP Registration after Downsampling

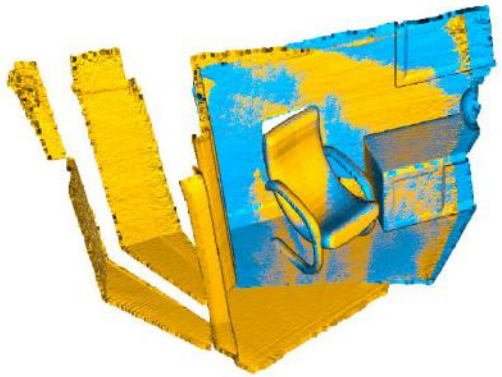


Figure: Result after RANSAC and upsampling

## Cross - Source Challenges

- **Noise and outliers:** Due to different sensor types, acquisition environments
- **Partial Overlap:** Only possible to retrieve the surface of the acetabulum
- **Density Difference:** Due to different imaging resolutions

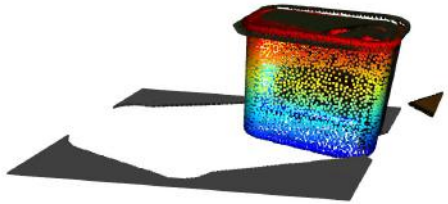
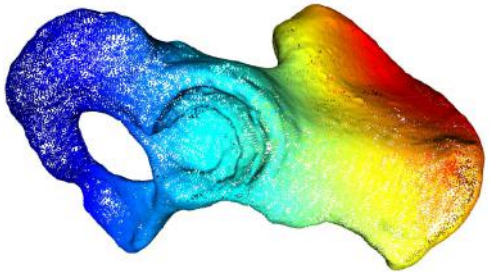


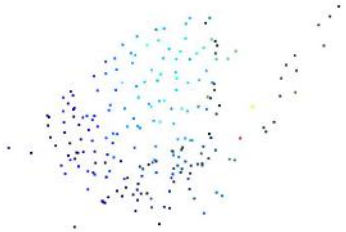
Figure: Cross-Registration ICP

# Open3D - ROS Integration

Convert Pelvis Mesh File Model to PointCloud & Downsample



Sample Collection using Registration Probe



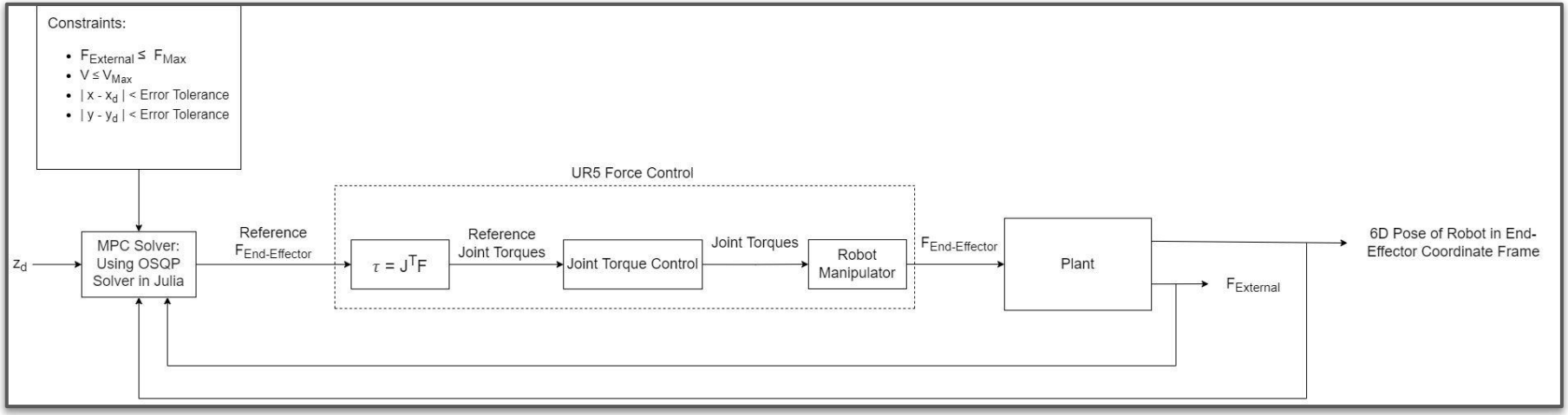
Global Registration

Transform Refinement with RANSAC

Suggestions on Registration Algorithms? Some options we've tried:

- Local & Global Registration with Iterative Closest Point (ICP) + Refinement with RANSAC

# **Controls:** High-Level Control Diagram


- MPC- Model Predictive Control; constraints - Limit force applied by end-effector, limit velocity, make sure xy positions are within tolerance

# **Controls:** Optimal Control Problem Setup

Initial draft of optimal control problem that MPC will solve; using OSQP to solve the optimal control problem

$$\min_{u_k, k\epsilon[1,H]} \quad \frac{1}{2}(z_H - zd)^T Q(z_H - zd) + \sum_{k=1}^{H-1} \frac{1}{2}(z_k - zd)^T Q(z_k - zd) + \frac{1}{2}u_k^T R u_k$$

$$\text{s.t.} \quad \begin{bmatrix} x_{k+1} \\ \dot{x_{k+1}} \\ y_{k+1} \\ \dot{y_{k+1}} \\ z_{k+1} \\ \dot{z_{k+1}} \\ \phi_{k+1} \\ \dot{\phi_{k+1}} \\ \theta_{k+1} \\ \dot{\theta_{k+1}} \\ \psi_{k+1} \\ \dot{\psi_{k+1}} \end{bmatrix} = \begin{bmatrix} x_k + \dot{x_k}h \\ \dot{x_k} + u_{x,k}h + F_{x,k}h \\ y_k + \dot{y_k}h \\ \dot{y_k} + u_{y,k}h + F_{y,k}h \\ z_k + \dot{z_k}h \\ \dot{z_k} + u_{z,k}h + F_{z,k}h \\ \phi_k + \dot{\phi_k}h \\ \dot{\phi_k} + u_{\phi,k}h + M_{\phi,k}h \\ \theta_k + \dot{\theta_k}h \\ \dot{\theta_k} + u_{\theta,k}h + M_{\theta,k}h \\ \psi_k + \dot{\psi_k}h \\ \dot{\psi_k} + u_{\psi,k}h + M_{\psi,k}h \end{bmatrix}$$

$$||F_{external}|| \leq F_{Max}$$

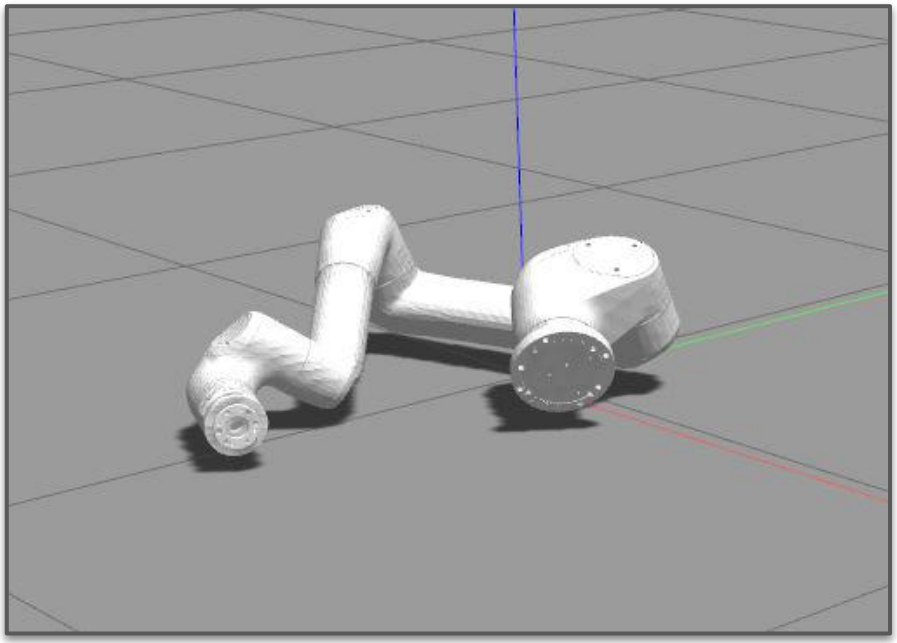
$$||V_k|| \leq V_{Max}$$

$$||x_k - x_d|| \leq \varepsilon$$

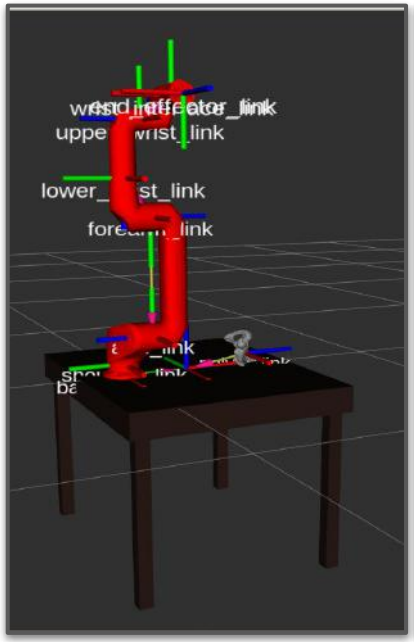
$$||y_k - y_d|| \leq \varepsilon$$

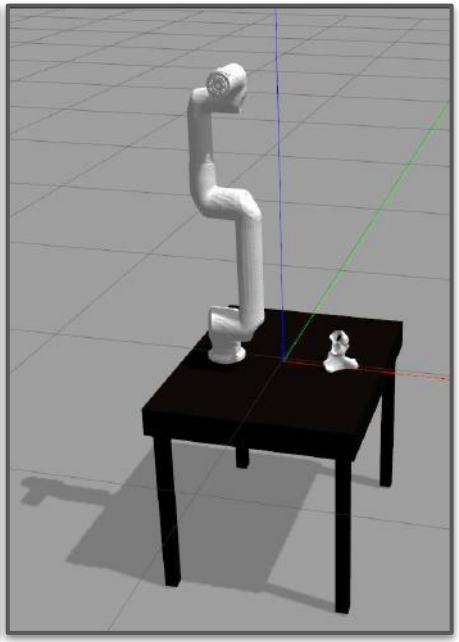
# **URDF:** Refinement & Simulation Setup

Attach transmission + hardware interfaces + controllers at each joint



**Before**



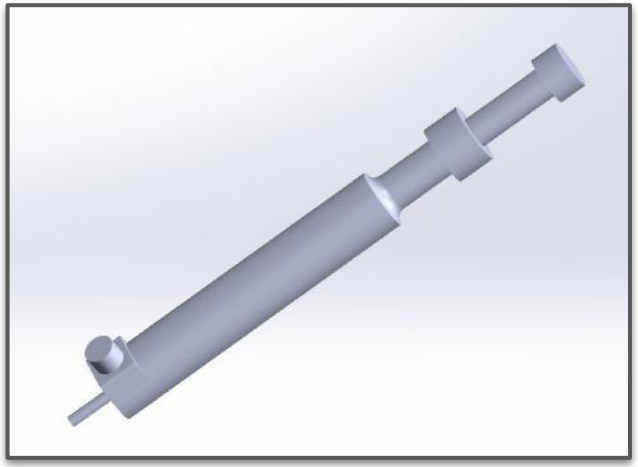
**After**

# **Controls:** Link-6 Joint Trajectory Control in Simulation

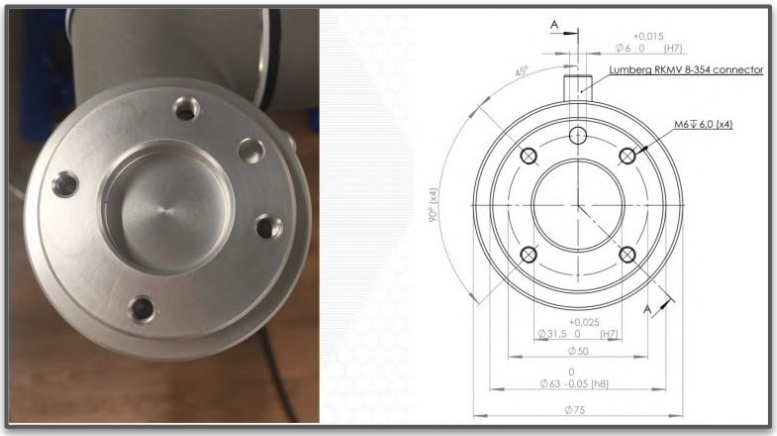
arthur_control > config > ! arthur_JTC.yaml

```yaml
arthur_controller:
  type: "position_controllers/JointTrajectoryController"
  joints:
    - joint_1
    - joint_2
    - joint_3
    - joint_4
    - joint_5
    - joint_6
  constraints:
          goal_time: 0.6
          stopped_velocity_tolerance: 0.05
          joint1: {trajectory: 0.1, goal: 0.1}
          joint2: {trajectory: 0.1, goal: 0.1}
          joint3: {trajectory: 0.1, goal: 0.1}
          joint4: {trajectory: 0.1, goal: 0.1}
          joint5: {trajectory: 0.1, goal: 0.1}
          joint6: {trajectory: 0.1, goal: 0.1}
  stop_trajectory_duration: 0.5
  state_publish_rate:  25
  action_monitor_rate: 10
```

# **Hardware:** Reaming Handle



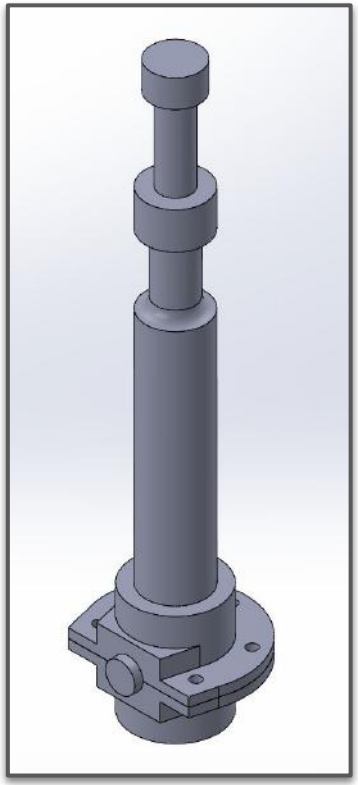
# **Hardware:** End-Effector Reaming Mount

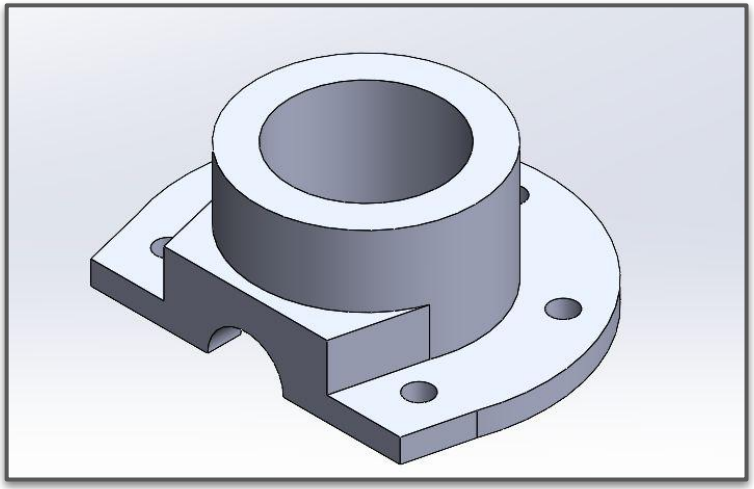



Ideas:

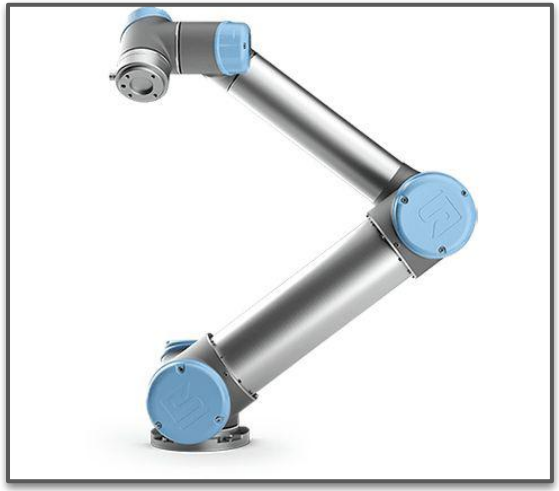
- Slot and Spring
  - Similar to marker mount on reaming handle
  - Marker pin would be under stress
- Bolted around marker pin
  - Two halves around the pin bolted together
- Clamp around the circular shaft
  - Use screws to tighten the clamp
- Set screws
  - Set screw onto circular shaft tightly

# **Hardware:** End-Effector Reaming Mount





# **Hardware:** Robot Manipulator



UR5

- Available from Professor Kroemer for “majority” of project


Kinova Gen3

- Availability from Sponsor TBD

# Future Work

| Schedule | | | |
|---|---|---|---|
| **Identifier** | **Capability Milestone(s)** | **Associated Tests** | **System Requirements** |
| **Progress Review 1 2/16/2022** | - Integration of the camera with ROS and the ability to detect marker poses in Atrascys SDK | Test 1 | M.F.1 |
| **Progress Review 2 3/2/2022** | - Markers can be read into ROS and create a point cloud comparable to the pelvis geometry<br>- Control method is capable of being used with robot manipulator virtually | Test 2<br>Test 3<br>Test 4<br>Test 13 | M.F.1<br>M.F.3<br>M.N.1 |

# Thank you!