

Autonomous Reaming for Total Hip Replacement (ARTHUR)

Progress Review - 2

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

2 March, 2022



Previously



Validation of Test 1: Reading and displaying camera's serial number

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

$$s.t. \begin{bmatrix} x_{k+1} \\ x_{k+1} \\ y_{k+1} \\ z_{k+1} \\ d_{k+1} \end{bmatrix} = \begin{bmatrix} x_k + \dot{x}_k h \\ \dot{x}_k + u_{k,k} h + F_{x,k} h \\ \dot{y}_k + \dot{y}_k h \\ d_k + d_k h \\ d_k + u_{d,k} h + M_{d,k} h \end{bmatrix}$$

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

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

$$||x_k - x_d|| \le \varepsilon$$

$$||y_k - y_d|| \le \varepsilon$$

Preliminary controls formulation

Schedule

	Schedule		
Identifier	Capability Milestone(s)	Associated Tests	System Requirements
Progress Review 1 2/16/2022	- Test camera health and camera discovery via a ROS Node.	Test 1	M.F.1
	- Broadcast marker pose as a ROS transform &	Test 2	M.F.1
Progress Review 2	ROS topic	Test 3	M.F.3
31212022	- Validate the preliminary performance of the registration algorithm chosen	Test 4	M.N.1

H



Hardware: Robot Manipulator!



Figure: Workspace setup



Kinova Gen3

- Available until December 2022
- On-going discussions on porting code to Kinova Link-6 when APIs are made available



Progress Review #2 Tests

- ✓ Marker Pose Detection Test Perception and Sensing
- ✓ Marker Pose Visualization Test
- ✓ Preliminary Point Cloud Registration Test
- ✓ Documentation
 - Perception and Sensing

Further Updates

Registration Perception and Sensing
 Controls Planning and Controls
 Simulation Planning and Controls
 Hardware Hardware

Perception and Sensing

Perception and Sensing



Progress Review - 2 Tests

Goal: Read camera measurements from a ROS Node, identify the fiducial points with a pre-loaded geometry file and print the 6DOF marker pose.

Approach: Add functionality onto the previously developed camera_node to detect marker poses using the provided Atracsys SDK.

<u>.</u>	Test 2:
	Marker Pose Detection Test
	Objective
Test fiducial m	arker 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 can 2. Wait for the 3. Place marke	nera_node ROS Node and wait for the node to discover the camera ROS Node to load the geometry file. ers in front of the camera.
	Validation
- Camera's se - Geometry file - The marker p	rial number is printed. e is loaded. pose is printed.

SNo.	Approach	Pros	Cons
1	Develop custom functions for triangulation and marker pose detection.	 Deep understanding of codebase. Lighter implementation 	 Complex s/w engineering issues Writing low-latency code is difficult
2	Use marker pose detection APIs from Atracsys SDK	 Preprocessing of images not required. Robust, well-tested codebase. 	 Cannot resolve any latency blocks. Less documentation - customization will be time-consuming





Figure: Test 2 Setup

Figure: Marker (left) Usage of Marker on Registration Probe (right)



Results:

- ✓ Camera's serial number printed
- ✓ Geometry file loaded
- ✓ Marker pose printed in terminal

PROBLEMS 8 OUTPUT TERMINAL DEBUG CONSOLE	
Right Count: 3 Fiducials Count- 0 Found!	
geometry 9990 , rotation (-0.41 -0.91 0.05 -0.90 0.42 0.11 -0.12 -0.01 -0.99) , trans (157.47 105.72 677.90), error 0.185	
Publishing Raw Data Count- Left Count: 3 Right Count: 3 Fiducials Count- 0 Found!	
geometry 9990 , rotation (-0.41 -0.91 0.06 -0.90 0.42 0.11 -0.12 -0.01 -0.99) , trans (157.42 104.99 678.79), error 0.185	
Publishing Raw Data Count- Left Count: 3 Right Count: 3 Fiducials Count- 0 Found!	
geometry 9990 , rotation (-0.41 -0.91 0.06 -0.90 0.42 0.11 -0.12 -0.00 -0.99) , trans (157.38 104.41 679.62), error 0.185	
0	



Challenges:

- → The code was not reliable; marker would not be detected robustly.
 - Issue was resolved by marker recalibration
 - Re-calibration of the marker geometry was performed using the GUI
 - New geometry loaded onto the GUI to test the marker detection robustness
 - The results were as expected within the error tolerance values





Figure: Marker Re-calibration Results

Goal:

- Part 1. Publish 6-DoF marker
 pose on a ROS topic &
 broadcast the transform.
- **Part 2.** Visualize the marker frame on RViz at >50Hz

_	lest 3:
	Marker Pose Visualization Test
	Objective
Test the publis using RViz.	shing of marker poses onto a ROS topic and visualizing markers
Equipment	MRSD Desktop 2, Atracys SpryTrack 300 Camera, Markers
Elements	Perception Subsystem
Personnel	Gunjan Sethi
Location	NSH Basement
	Procedure
camera. 2. Wait for the 3. Place mark 4. Run rostopi topic in ROS. 5. Run RViz	ROS Node to load the geometry file. ers in front of the camera. c command-line tool to view messages on the marker-pose Validation
- Camera's se - Geometry file	rial number is printed. ∋ is loaded.



Approaches (Rotation Matrix to Quaternion)

Convert the incoming Marker frame to geometry_msgs/PoseStamped Message type.

SNo.	Approach	Pros	Cons
1	Use Eigen	Well tested codeLow latencyPopular choice	 Typecasting to Eigen message to ROS message required
2	Use tf libraries (rot_to_quat)	 Well tested code No typecasting required Popular choice 	 Deprecated functions; several dependency issues
3	Write a custom function	- Very simple implementation	 Need to perform robust testing and ensure FPS retention

Results

- ✓ Camera's serial number is printed.
- ✓ Geometry file is loaded.
- Pose published to a topic & appears on command-line (Part 1)
- ✓ Marker frame visualized on RViz (Part 2)

	372	
	PROBLEMS 8 OUTPUT TERMINAL DEE	UG CONSOLE
	nosition.	
	x 179 99940490722656	
	V: 103 06/68353271/8/	
	7. 012 731628/170688	
	orientation:	
	x0 3068847728014501	
	V: 0.0380/3516281230	
	7. 0 1599279350747184	
	W: 0 017864646773225443	
	position:	
	x: 179.0535430908203	
	v: 103.69467163085938	
	7: 912.8483276367188	
	orientation:	
	x: -0.29747654264783296	
	v: 0.9412360136353184	
	z: 0.1588864334747869	
	w: 0.01837320192802982	
	position:	
	x: 178.1487579345703	
	y: 103.60475158691406	
	z: 912.953857421875	
	orientation:	
	x: -0.28920635113952475	
	y: 0.9440958690333532	
	z: 0.15711438162071895	
	w: 0.018914201556084274	
3 🛆	▲ 0	

Figure: Marker Detection Test Results



average rate: 54.889 min: 0.0185 max: 0.025s std dev: 0.00163s window: 54 average rate: 54.021 min: 0.0185 max: 0.025s std dev: 0.00102s window: 109 average rate: 54.035 average rate: 54.035 min: 0.0185 max: 0.025s std dev: 0.00092s window: 217 min: 0.0185 max: 0.025s std dev: 0.00092s window: 217 min: 0.0185 max: 0.025s std dev: 0.00092s window: 221 min: 0.0185 max: 0.025s std dev: 0.00092s window: 325 min: 0.0185 max: 0.025s std dev: 0.00097s window: 379 min: 0.0185 max: 0.025s std dev: 0.00097s window: 433 average rate: 54.035 min: 0.0185 max: 0.025s std dev: 0.00092s window: 487 min: 0.0185 max: 0.025s std dev: 0.00092s window: 487 min: 0.0185 max: 0.025s std dev: 0.00092s window: 541 average rate: 54.035 min: 0.0185 max: 0.025s std dev: 0.00095s window: 595 min: 0.0185 max: 0.027s std dev: 0.00095s window: 595	subscrit	bed to [/marker_p	oses]	10	1			
nin: 0.0106 max: 0.0205 std dev: 0.00053 window: 54 werage rate: 54.021 min: 0.0108 max: 0.025 std dev: 0.00126 window: 109 werage rate: 54.035 min: 0.0108 max: 0.025 std dev: 0.001025 window: 163 werage rate: 54.035 min: 0.0108 max: 0.025 std dev: 0.00055 window: 217 werage rate: 54.035 min: 0.0108 max: 0.0256 std dev: 0.000125 window: 271 werage rate: 54.035 min: 0.0108 max: 0.0256 std dev: 0.001025 window: 379 werage rate: 54.035 min: 0.0108 max: 0.0256 std dev: 0.000125 window: 379 werage rate: 54.035 min: 0.0186 max: 0.0256 std dev: 0.000975 window: 487 min: 0.0186 max: 0.0256 std dev: 0.000955 window: 541 werage rate: 54.032 min: 0.0186 max: 0.0255 std dev: 0.000855 window: 541 werage rate: 54.036 min: 0.0186 max: 0.0255 std dev: 0.000855 window: 541 werage rate: 54.036 min: 0.0186 max: 0.0255 std dev: 0.000855 window: 541 min: 0.0186 max: 0.0255 std dev: 0.000855 window: 541 min: 0.0186 max: 0.0255 std dev: 0.000855 window: 541 min: 0.0186 max: 0.0255 std dev: 0.000855 window: 649	average	rate: 54.669						
average rate: 54.021 min: 0.0108 max: 0.026s std dev: 0.00126 window: 109 average rate: 54.035 min: 0.0188 max: 0.026s std dev: 0.000025 window: 163 average rate: 54.026 min: 0.026 max: 0.026s std dev: 0.000025 window: 271 average rate: 54.025 min: 0.0186 max: 0.026s std dev: 0.000025 window: 379 average rate: 54.035 min: 0.0186 max: 0.026s std dev: 0.000075 window: 379 average rate: 54.035 min: 0.0186 max: 0.026s std dev: 0.000075 window: 433 min: 0.0186 max: 0.026s std dev: 0.000025 window: 487 min: 0.0186 max: 0.025s std dev: 0.000025 window: 541 average rate: 54.035 min: 0.0186 max: 0.025s std dev: 0.000025 window: 541 average rate: 54.036 min: 0.0186 max: 0.027s std dev: 0.000055 window: 595 min: 0.0186 max: 0.027s std dev: 0.000055 window: 595 min: 0.0186 max: 0.027s std dev: 0.000055 window: 595		min: 0.010s max:	0.0265	std	dev:	0.00163s	window:	.54
 min: 0.018; max: 0.0255 std dev: 0.00205 window: 109 average rate: 54.035 min: 0.018; max: 0.025 std dev: 0.00102; window: 163 average rate: 54.035 min: 0.018; max: 0.025 std dev: 0.0092; window: 217 average rate: 54.035 min: 0.018; max: 0.025 std dev: 0.0005; window: 271 average rate: 54.035 min: 0.018; max: 0.025 std dev: 0.0010; window: 325 average rate: 54.035 min: 0.018; max: 0.025 std dev: 0.0010; window: 379 average rate: 54.035 min: 0.018; max: 0.026; std dev: 0.0009; window: 433 average rate: 54.036 min: 0.018; max: 0.025; std dev: 0.0008; window: 541 min: 0.018; max: 0.025; ud dev: 0.0008; window: 595 average rate: 54.036 min: 0.018; max: 0.027; std dev: 0.0008; window: 595 average rate: 54.036 min: 0.018; max: 0.025; std dev: 0.0008; window: 649 	average	rate: 54.021						
average rate: 54.835 min:: 0.0188 max: 0.026s std dev: 0.00102s window: 163 average rate: 24.826 min:: 0.0188 max: 0.026s std dev: 0.00992s window: 217 average rate: 54.831 min:: 0.0186 max: 0.026s std dev: 0.00082s window: 271 average rate: 54.833 min:: 0.0186 max: 0.026s std dev: 0.00097s window: 325 average rate: 54.833 min:: 0.0186 max: 0.026s std dev: 0.00997s window: 433 average rate: 54.833 min:: 0.0186 max: 0.026s std dev: 0.00982s window: 487 average rate: 54.835 min:: 0.026s max: 0.026s std dev: 0.00082s window: 487 average rate: 54.835 min:: 0.0186 max: 0.026s std dev: 0.00085s window: 541 average rate: 54.836 min:: 0.0186 max: 0.027s std dev: 0.0095s window: 595 min:: 0.0186 max: 0.027s std dev: 0.0095s window: 595 min:: 0.0186 max: 0.027s std dev: 0.0095s window: 599		min: 0.0105 max:	0.0265	std	dev:	0.001205	window:	109
nin: 0.0185 max: 0.025s std dev: 0.00102s window: 163 average rate: 54.020 min: 0.0185 max: 0.025s std dev: 0.0092s window: 217 wirrage rate: 54.031 min: 0.0185 max: 0.025s std dev: 0.0005s window: 271 average rate: 54.035 min: 0.0186 max: 0.025s std dev: 0.00102s window: 325 average rate: 54.035 min: 0.0186 max: 0.025s std dev: 0.00097s window: 379 min: 0.0186 max: 0.025s std dev: 0.00097s window: 433 average rate: 54.035 min: 0.0185 max: 0.025s std dev: 0.00092s window: 487 min: 0.0185 max: 0.025s std dev: 0.00085s window: 541 average rate: 54.034 min: 0.0185 max: 0.027s std dev: 0.00085s window: 595 min: 0.0185 max: 0.027s std dev: 0.00085s window: 549	average	rate: 54.035						
average rate: 54.820 nin: 0.8185 max: 0.8265 std dev: 8.000925 window: 217 average rate: 54.831 nin: 0.8185 max: 0.8265 std dev: 8.000855 window: 271 average rate: 54.835 nin: 0.8185 max: 0.8265 std dev: 8.001825 window: 325 average rate: 54.839 nin: 0.8185 max: 0.8265 std dev: 8.000975 window: 433 average rate: 54.835 nin: 0.8185 max: 0.8265 std dev: 8.000925 window: 433 average rate: 54.832 nin: 0.8185 max: 0.8265 std dev: 8.000855 window: 487 average rate: 54.832 nin: 0.8185 max: 0.8255 std dev: 8.000855 window: 541 average rate: 54.834 nin: 0.8185 max: 0.8275 std dev: 8.000855 window: 541 average rate: 54.836 nin: 0.8185 max: 0.8275 std dev: 8.000925 window: 541 average rate: 54.836 nin: 0.8185 max: 0.8275 std dev: 8.000925 window: 545 average rate: 54.836 nin: 0.8185 max: 0.8275 std dev: 8.000925 window: 545 average rate: 54.836 nin: 0.8185 max: 0.8275 std dev: 8.000925 window: 545 average rate: 54.836 nin: 0.8185 max: 0.8275 std dev: 8.000925 window: 545 average rate: 54.836 nin: 0.8185 max: 0.8275 std dev: 8.000925 window: 545 average rate: 54.836 nin: 0.8185 max: 0.8275 std dev: 8.000925 window: 545 average rate: 54.836 nin: 0.8185 max: 0.8275 std dev: 8.000925 window: 549 average rate: 54.836 nin: 0.8185 max: 0.8275 std dev: 8.000925 window: 549 average rate: 54.836 nin: 0.8185 max: 0.8275 std dev: 8.000925 window: 549 average rate: 54.836 nin: 0.8185 max: 0.8275 std dev: 8.000925 window: 549 average rate: 54.836 nin: 0.8185 max: 0.8275 std dev: 8.000925 window: 549 average rate: 54.836 nin: 0.8185 max: 0.8275 std dev: 8.000925 window: 549 average rate: 54.836 nin: 0.8185 max: 0.8275 std dev: 8.000925 window: 549 average rate: 54.836 nin: 0.8185 max: 0.8275 std dev: 8.000925 window: 549 average rate: 54.836 nin: 0.8185 max: 0.8275 std dev: 8.000925 window: 549 average rate: 54.836 average rate: 54.836		min: 0.0105 max:	0.0265	std	dev:	0.001025	window:	
hin: 0.0109 max: 0.0205 std dev: 0.000925 window: 217 wirrage rate: 54.031 wirrage rate: 54.035 min: 0.0108 max: 0.0265 std dev: 0.00055 window: 271 wirrage rate: 54.035 min: 0.0108 max: 0.0265 std dev: 0.001025 window: 379 wirrage rate: 54.035 min: 0.0186 max: 0.0265 std dev: 0.000925 window: 433 wirrage rate: 54.035 min: 0.0186 max: 0.0265 std dev: 0.000925 window: 487 wirrage rate: 54.036 min: 0.0186 max: 0.0275 std dev: 0.000855 window: 595 min: 0.0186 max: 0.0275 std dev: 0.000955 window: 549	average	rate: 54.026						
average rate: 54.031 min: 0.0105 max: 0.0265 std dev: 0.00055 window: 271 average rate: 54.035 min: 0.0105 max: 0.0265 std dev: 0.00102s window: 325 average rate: 54.039 min: 0.0105 max: 0.0265 std dev: 0.00097s window: 379 average rate: 54.033 min: 0.0105 max: 0.0265 std dev: 0.00092s window: 433 average rate: 54.035 min: 0.0105 max: 0.0265 std dev: 0.00092s window: 487 average rate: 54.032 min: 0.0105 max: 0.0255 std dev: 0.00085s window: 541 average rate: 54.034 min: 0.0105 max: 0.0275 std dev: 0.00092s window: 595 average rate: 54.036 min: 0.0105 max: 0.0275 std dev: 0.00092s window: 541		min: 0.010s max:	0.0265	std	dev:	0.00092s	window:	
min: 0.0105 max: 0.0205 std dev: 0.00055 window: 271 average rate: 34.035 min: 0.0108 max: 0.025 std dev: 0.001025 window: 325 average rate: 54.035 min: 0.0186 max: 0.025 std dev: 0.00075 window: 379 average rate: 54.035 min: 0.0186 max: 0.026 std dev: 0.000925 window: 433 average rate: 54.035 min: 0.0186 max: 0.025 std dev: 0.00085 window: 541 average rate: 54.036 min: 0.0186 max: 0.0275 std dev: 0.00085 window: 549	average	rate: 54.031						
average rate: 54.035 min: 0.0108 max: 0.026s std dev: 0.00102s window: 325 average rate: 54.039 min: 0.0108 max: 0.026s std dev: 0.00097s window: 379 average rate: 54.033 min: 0.0108 max: 0.026s std dev: 0.00092s window: 433 average rate: 54.035 min: 0.0108 max: 0.026s std dev: 0.00085s window: 487 average rate: 54.032 min: 0.0108 max: 0.025s std dev: 0.00085s window: 541 average rate: 54.034 min: 0.0108 max: 0.027s std dev: 0.00085s window: 595 average rate: 54.036 min: 0.0108 max: 0.027s std dev: 0.00095s window: 595 average rate: 54.036 min: 0.0108 max: 0.027s std dev: 0.00092s window: 549		min: 0.0105 max:	0.0265	std	dev:	0.00085s	window:	271
min: 0.0189 max: 0.0255 std dev: 0.001025 window: 325 werage rate: 54.029 min: 0.0186 max: 0.0255 std dev: 0.000975 window: 379 werage rate: 54.035 min: 0.0186 max: 0.0255 std dev: 0.000925 window: 433 werage rate: 54.035 min: 0.0186 max: 0.0255 std dev: 0.000855 window: 541 average rate: 54.036 min: 0.0186 max: 0.0275 std dev: 0.000855 window: 595 min: 0.0186 max: 0.0275 std dev: 0.000955 window: 595 min: 0.0186 max: 0.0275 std dev: 0.000955 window: 595	average	rate: 54.035						
average rate: 54.829 nin: 0.618% nax: 0.626% std dev: 0.00097% window: 379 average rate: 54.033 nin: 0.618% nax: 0.026% std dev: 0.00092% window: 433 average rate: 54.035 nin: 0.618% nax: 0.026% std dev: 0.00085% window: 541 average rate: 54.034 nin: 0.618% nax: 0.027% std dev: 0.00085% window: 555 average rate: 54.036 nin: 0.618% nax: 0.027% std dev: 0.00095% window: 549		min: 0.0105 max:	0.0265	std	dev:	0.001025	window:	
min: 0.018* max: 0.020* std dev: 0.00097* window: 379 werage rate: 54.033 min: 0.018* max: 0.025* std dev: 0.00092* window: 433 average rate: 54.035 min: 0.018* max: 0.025* std dev: 0.00085* window: 487 average rate: 54.032 average rate: 54.032 min: 0.018* max: 0.027* std dev: 0.00085* window: 595 mir: 0.018* max: 0.027* std dev: 0.00095* window: 595 mir: 0.018* max: 0.027* std dev: 0.00095* window: 595	average	rate: 54.029						
average rate: 54,033 min: 0.018% max: 0.026% std dev: 0.00092% window: 433 average rate: 54,035 min: 0.018% max: 0.026% std dev: 0.00085% window: 541 average rate: 54,034 min: 0.018% max: 0.027% std dev: 0.00085% window: 595 average rate: 54,036 min: 0.018% max: 0.027% std dev: 0.00095% window: 595 min: 0.018% max: 0.027% std dev: 0.00092% window: 649		min: 0.010s max:	0.0265	std	dev:	0.00097s	window:	
min: 0.0186 max: 0.0205 std dev: 0.00092s window: 433 werage rate: 54.035 min: 0.0186 max: 0.026s std dev: 0.00082s window: 487 average rate: 54.032 min: 0.0186 max: 0.0205 std dev: 0.00085s window: 541 average min: 0.0186 max: 0.027s std dev: 0.00095s window: 595 min: 0.0186 max: 0.027s std dev: 0.00095s window: 595 min: 0.0186 max: 0.027s std dev: 0.00095s window: 595	average	rate: 54.033						
average rate: 54.035 min: 0.0105 max: 0.0205 std dev: 0.000855 window: 487 average rate: 54.032 min: 0.0105 max: 0.0255 std dev: 0.000855 window: 541 average rate: 54.034 min: 0.0105 max: 0.0275 std dev: 0.000955 window: 595 average rate: 54.036 min: 0.0105 max: 0.0275 std dev: 0.000925 window: 649		min: 0.010s max:	0.0265	std		0.00092s	window:	433
min: 0.0186 max: 0.0256 std dev: 0.000855 window: 487 average rate: 54.032 min: 0.0186 max: 0.0256 std dev: 0.000855 window: 541 average rate: 54.034 min: 0.0184 max: 0.0275 std dev: 0.000955 window: 595 average rate: 54.036 min: 0.0185 max: 0.0275 std dev: 0.009925 window: 649	average	rate: 54.035						
average rate: 54.822 min: 0.0185 max: 0.0205 std dev: 0.000855 window: 541 average rate: 54.834 min: 0.0185 max: 0.0275 std dev: 0.00955 window: 595 average rate: 54.836 min: 0.0185 max: 0.0275 std dev: 0.009025 window: 649		min: 0.010s max:	0.0265	std	dev:	8.00088s	window:	
min: 0.019s max: 0.025 std dry: 0.00085s window: 541 average rate: 54.034 min: 0.019s max: 0.027s std dry: 0.00095s window: 595 average rate: 54.036 min: 0.010s max: 0.027s std dry: 0.00092s window: 649	average	rate: 54.032						
average rate: 54.034 min: 0.010s max: 0.027s std dev: 0.00095s window: 595 average rate: 54.036 min: 0.010s max: 0.027s std dev: 0.00092s window: 649		min: 0.010s max:	0.0265	std		0.000855	window:	541
min: 0.010s max: 0.027s std dev: 0.00095s window: 595 average rate: 54.036 min: 0.010s max: 0.027s std dev: 0.00092s window: 649	average	rate: 54.034						
average rate: 54.036 min: 0.010s max: 0.027s std dev: 0.00092s window: 649		min: 0.0105 max:	0.0275	std	dev:	0.00095s	window:	595
min: 0.010s max: 0.027s std dev: 0.00092s window: 649	average	rate: 54.036						
	-	min: 0.0105 max:	0.0275	std	dev:	0.00092s	window:	649



Figure: Marker TF Broadcasting on RViz



Challenges:

- → TF functions not supported on TF2
 - Lack of functionality to convert rotation matrices to quaternion in TF2
 - TF2 required quaternion for rotation did not support rotation matrices
- → Eigen to ROS TF message conversion
 - Eigen outputs needed to be extracted and type-casted to ROS compatible dependencies
- → Debugging Missing Dependencies
 - CMake Errors were always not indicative of the root issue debugging this took time
- → Marker TF not visible on RViz
 - Units conversions and flipping of coordinate axes



Overview:

Goal: Validate the ability of the chosen algorithm to register the **simulated** acetabulum point cloud with the 3D scanned point cloud of the pelvis

Approach: Improve upon the ICP registration package offered by Open3D

	Test 4:			
	Preliminary Point Cloud Registration Test			
	Objective			
Validate the se ability of the re pelvis point cle	election of the registration algorithm for the use-case and test the egistration algorithm to register the simulated fiducial points onto the oud.			
Equipment	MRSD Desktop 2, Atracys SpryTrack 300 Camera, Markers			
Elements	Perception Subsystem			
Personnel	Kaushik Balasundar			
Location	NSH Basement			
	Procedure			
1. Load two po downsampled 2. Run the sel from pointclou	ointsets, one of the complete pelvis model and the other of the point cloud from the surface of the acetabulum. ected registration algorithm and visually validate the transformation d A to pointcloud B.			
	validation			



Approaches: Algorithms & Tools

- Tools: Open3D / ITK / PCL
 - Python-friendly
 - Prior experience
- Algorithms: Iterative Closest Point / Learning-based Methods
 - Native Open3D implementations & support
 - Industry standard for several years
 - Verified as a reliable method in other medical robotics applications



Figure: Registration Methods Overview



Approaches: Acquiring Test Model for Registration

- Off-the-shelf 3D model
- 3D scan model using Laser scanner (Konica Minolta Vivid 9i)
- 3D scan model using Camera / LiDAR setup (iPAD Pro / Kinect)







Figure: Off-the-shelf 3D model

Figure: Konica Minolta Vivid 9i

Figure: iPad Pro



Preliminary Experimentation



Figure: Two Pointclouds Initialized Figure: ICP Registration after Downsampling

Figure: Result after RANSAC and upsampling



Figure: Custom Pointclouds



Validation

Validation

- Pointcloud B needs to be roughly overlapping with pointcloud A's acetabulum region to indicate that the registration has taken place.







Figure: Source and target pointclouds

Figure: Downsampled pointclouds after initial registration

Figure: Point-to-point distance cost function results after RANSAC refinement



Challenges

- Post-processing 3D model from the scanner
 - Hole-filling using Autodesk MeshMixer
- Point cloud density differences
 - Source: 3D scanned pelvis (56704 points)
 - Target: Acetabular Surface (373 points)
- Hyperparameter tuning for registration and refinement
- RANSAC for fine-tuning post registration



Figure: Density disparity between source and target pointclouds



Further Updates



Simulation Update



Figure: Simulation Environment on Gazebo & RViz

Controls Update: Tasks

- Updated Optimal Control Problem w/ Guidance from Professor Zachary Manchester
- Create Model and Dynamics in Julia using packages from the CMU Robot **Exploration Lab**

ad

Write Constraints and Objective function in Julia Constraints:

$$\begin{aligned} \min_{u_k,k \in [1,H]} \quad & \frac{1}{2} (s_H - sd)^T \mathcal{Q}_H (s_H - sd) + \sum_{k=1}^{H-1} \frac{1}{2} (s_k - sd)^T \mathcal{Q} (s_k - sd) + \frac{1}{2} u_k^T R u_k \\ \text{s.t.} \quad & \begin{bmatrix} \dot{q} \\ \ddot{q} \end{bmatrix} = \begin{bmatrix} \dot{q} \\ M^{-1} (\tau - \tau_{External} - C\dot{q} - G) \end{bmatrix} \qquad \text{State} = \begin{bmatrix} q \\ \dot{q} \end{bmatrix} \\ u_k = \tau \le \tau_{Max} \\ ||F_{External}|| = ||B_{External} J \dot{q}_k|| \le F_{Max} \\ ||\dot{X}|| = ||J \dot{q}_k|| \le \dot{X}_{Max} \\ q_k \epsilon q_{Limits} \\ \dot{q}_k \epsilon \dot{q}_{Limits} \end{aligned}$$



Sk

Figure: Initial Optimal Control Formulation

Figure: Control Architecture

Controls Update: Tasks

- Updated Optimal Control Problem w/ Guidance from Professor Zachary Manchester
- Create Model and Dynamics in Julia using packages from the CMU Robot **Exploration Lab**

ad

Write Constraints and Objective function in Julia Constraints:

$$\begin{aligned} \min_{u_k,k \in [1,H]} \quad & \frac{1}{2} (s_H - sd)^T \mathcal{Q}_H (s_H - sd) + \sum_{k=1}^{H-1} \frac{1}{2} (s_k - sd)^T \mathcal{Q} (s_k - sd) + \frac{1}{2} u_k^T R u_k \\ \text{s.t.} \quad & \begin{bmatrix} \dot{q} \\ \ddot{q} \end{bmatrix} = \begin{bmatrix} \dot{q} \\ M^{-1} (\tau - \tau_{External} - C\dot{q} - G) \end{bmatrix} \qquad \text{State} = \begin{bmatrix} q \\ \dot{q} \end{bmatrix} \\ u_k = \tau \le \tau_{Max} \\ ||F_{External}|| = ||B_{External} J \dot{q}_k|| \le F_{Max} \\ ||\dot{X}|| = ||J \dot{q}_k|| \le \dot{X}_{Max} \\ q_k \epsilon q_{Limits} \\ \dot{q}_k \epsilon \dot{q}_{Limits} \end{aligned}$$



Sk

Figure: Initial Optimal Control Formulation

Figure: Control Architecture

Controls Update: Challenges

- Documentation is a little scattered:
 - Some libraries have dependencies on older versions of other libraries
- Even though packages have dynamics calculations, it doesn't provide an easy way to add external forces into consideration
- The way constraints are implemented in the packages makes it so that we have to keep track of extra variables within our state
 - Constraints must be directly from the state or control vectors
 - Those extra variables' dynamics also need to be calculated

Calculate dynamics

```
RigidBodyDynamics.dynamics!(dynamicsResult, mechanismState, u)
# Add the effects of external forces/torques into dynamics
qdd = M\((M * dynamicsResult.v) - τ_ext)
```

$$\begin{aligned} \min_{\substack{k,u_k,k\in\{1,H\}}} & \frac{1}{2}(s_H - sd)^T Q_H(s_H - sd) + \sum_{k=1}^{H-1} \frac{1}{2}(s_k - sd)^T Q(s_k - sd) + \frac{1}{2}u_k^T R u_k \\ \text{s.t.} & \begin{bmatrix} \dot{q} \\ \ddot{q} \\ \ddot{x} \\ \dot{F}_{External} \end{bmatrix} = \begin{bmatrix} M^{-1}(\tau - \tau_{External} - C\dot{q} - G) \\ \dot{J}\dot{q} + J\ddot{q} \\ B_{External}(\dot{J}\dot{q} + J\ddot{q}) \end{bmatrix} \quad \text{State} = \begin{bmatrix} q \\ \dot{q} \\ \dot{x} \\ F_{External} \end{bmatrix} \\ u_k = \tau \le \tau_{Max} \\ ||F_{External}|| = ||B_{External}J\dot{q}_k|| \le F_{Max} \\ ||\dot{X}|| = ||J\dot{q}_k|| \le \dot{X}_{Max} \\ q_k \in q_{Limits} \\ \dot{q}_k \in \dot{q}_{Limits} \end{aligned}$$

Figure: Updated Optimal Control Formulation

Controls Update: Code

function RobotDynamics.dynamics(model::Arthur, x, u)

Create a state of the mechanism model and a result struct for the dynamics dynamicsResult = RigidBodyDynamics.DynamicsResult(model.mechanism) mechanismState = RigidBodyDynamics.MechanismState(model.mechanism)

Get states and constants of system not dependent on model state

```
M = RigidBodyDynamics.mass_matrix(mechanismState)
num_q = RigidBodyDynamics.num_positions(model.mechanism)
q = x[1:num_q]
qd = x[num_q+1:2*num_q]
xd = x[2*num_q + 1:2*num_q + 6]
F = x[2*num_q + 7:2*num_q + 12]
Be = zeros(6, 6)
if (norm(xd) > 1e-5)
    for k = 1:3
        Be[k,k] = norm(F) / norm(xd)
end
end
```

Set mechanism state to current state
RigidBodyDynamics.set_configuration!(mechanismState, q)
RigidBodyDynamics.set velocity!(mechanismState, qd)

```
# Get variables dependent on state
J = getJacobian(model, q, qd)
τ ext = transpose(J)*Be*xd
```

end

Calculate dynamics
RigidBodyDynamics.dynamics!(dynamicsResult, mechanismState, u)
Add the effects of external forces/torques into dynamics
qdd = M\((M * dynamicsResult.v) - t ext)
x = getj(model, J, qd, q)*qd + J*qdd
F = Be*x
return [qd; qdd; x; F; 0; 0; 0; 0; 0; 0]

Create Empty ConstraintList
conSet = ConstraintList(n,m,N)

Control Bounds based on Robot Specs (Joint torque limits)
u bnd = [39.0, 39.0, 39.0, 39.0, 90.0, 90.0, 90.0]
control_bnd = BoundConstraint(n,m, u_min=-u_bnd, u_max=u_bnd)
add constraint!(conSet, control_bnd, 1:N-1)

Cartesian Velocity Bound

x max = 0.0005 # m/s vel_bnd = NormConstraint(n, m, x max, Inequality(), 15:20) add constraint!(conset, vel bnd, 1:N)

Force Bound
F_max = 20 # Newtons
F_bnd = NormConstraint(n, m, F_max, Inequality(), 21:26)
add_constraint!(conSet, F_bnd, 1:N)

Goal Constraint
goal = GoalConstraint(xf)
add_constraint!(conSet, goal, N)

Figure: Problem Constraints Code

Figure: Dynamics Implementation Code



Hardware Update: Vention Table

Update

• Set up a Vention Table stand for our robot arm!

Challenges:

- Had to hand tap M8 holes into some of the Vention bars for connections to be properly made
- Spent large amount of time attaching the Vention together

Future Work:

- Get wooden base created for the bottom of the table for electrical components and storage
- Mount e-stop



Figure: Vention Table Assembled



Hardware Update: Kinova Gen3 Arm

Update

• Set up the Kinova Gen3 Arm!

Challenges:

- Had to exchange the other robot arm we previously had
- Needed to move around some of the Vention bars to fit the base, and it could still only fit sideways.

Future Work:

• Begin working on controlling the arm with ROS





Hardware Update: PCB Designed for End-Effector

Update

 Created a motor control PCB schematic for controlling a brushed DC motor for the acetabular reamer assembly

Challenges:

- Realized a power distribution system was less needed for our system and thus changed our PCB to be more of a motor control PCB
- Had some issues with creating libraries of custom parts

Future Work:

- Finalize parts and board layout
- Order parts for the end-effector



H

Hardware Update: End-Effector Redesign

Update

- Redesigned our end-effector based on feedback from sponsor
- Now going for a clamping design as seen on the right

Challenges:

• 3D printed many similar designs with dimensional differences to find best fit

Future Work:

- Prototype final clamping design and look into rubbers that could be used with clamp
- Finalize end-effector design





Project Management Update: Updated Jira Roadmap



- Continue to work in 2-week sprints
- Hackathons on Fridays!



Plans : Progress Review 3



Progress Review #3 Tests

- Landmark Capture Perception and Sensing
- Waypoint/Trajectory Generation Planning and Controls
- Position and Force Control in Simulation
- Reamer Motor Speed and Torque Hard

Planning and Controls

Thank you! Note: Second Sec