Perception in Robotics course, Final project

Coupling of localization and depth data for mapping

Team 2: Evgeny Tsykunov, Valery Ilin, Stepan Perminov,
Aleksey Fedoseev, Elvira Zainulina

Skoltech, March 30 2020

Project team

Aleksey Fedoseev
MSc, 2nd year

Elvira Zainulina
MSc, 1st year

Evgeny Tsykunov
PhD, 4th year

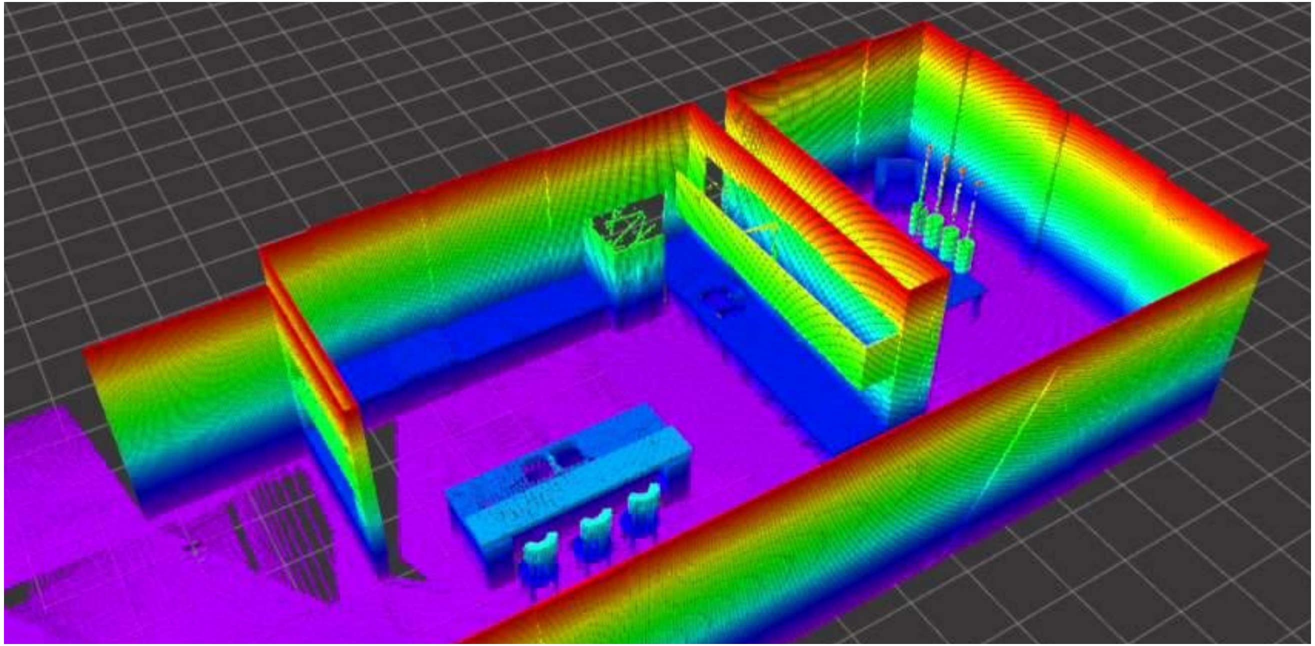
Stepan Perminov
MSc, 1st year

Valery Ilin
MSc, 1st year

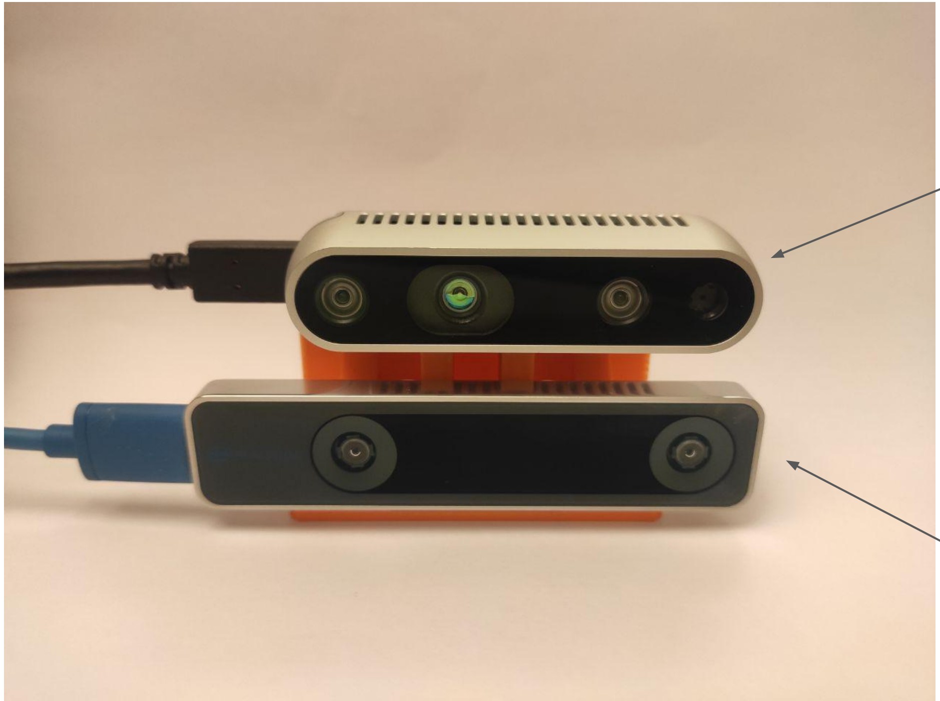
Global objective

Objective

How to **localize** and create a **3D map** combining of **tracking** and **depth** data?



Equipment and datasets



Intel RealSense D435i (Depth)

Image Sensor Technology:

Global Shutter, 3µm x 3µm pixel size

Depth Technology:

Active IR Stereo

Depth Field of View (FOV):

87°±3° x 58°±1° x 95°±3°

Sensor update data: 30Hz

Intel RealSense T265 (Pose)

Image Sensor:

Two Fisheye lenses with combined

163±5° FOV

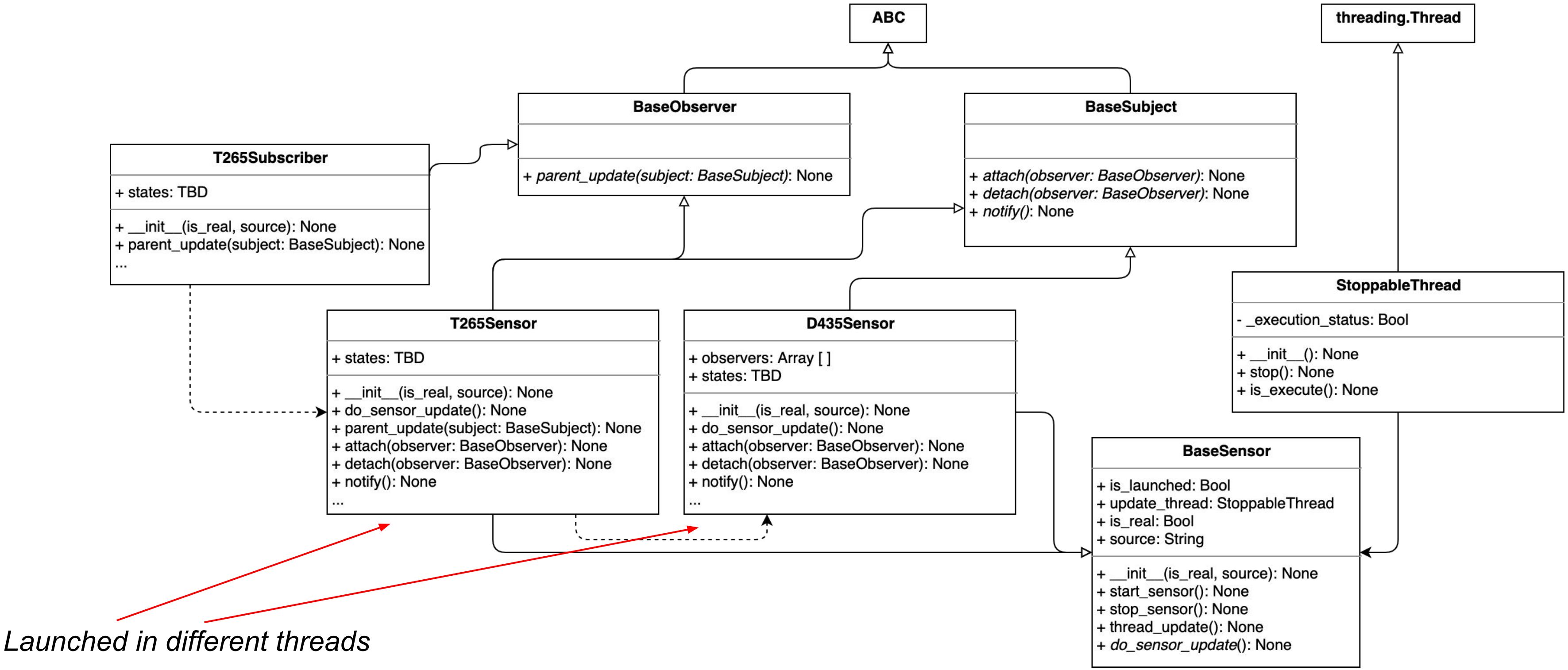
IMU: BMI055 IMU Sensor

SLAM: Intel Visual Inertial Odometry SLAM

Initial Data (depth)

Data synchronization

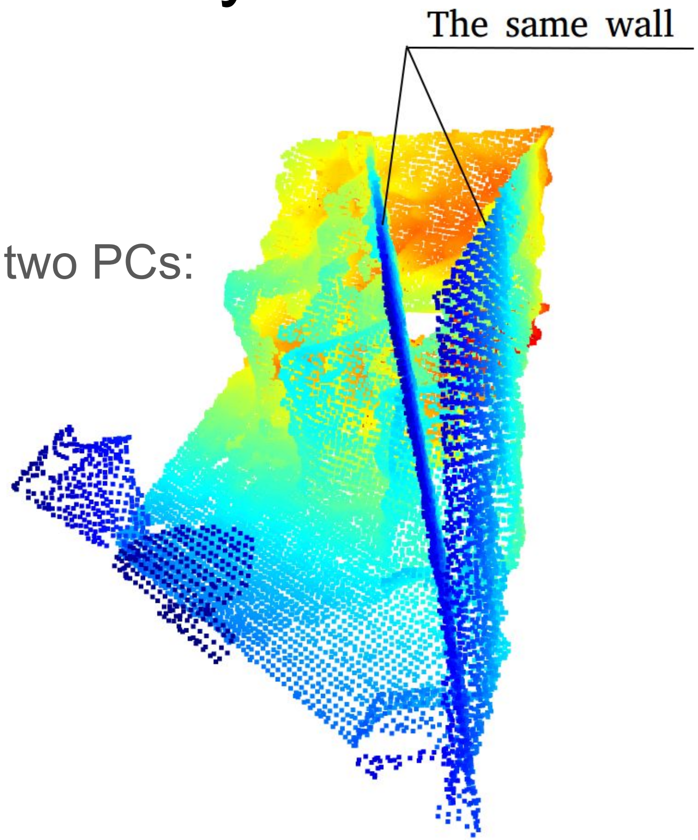
Depth stream - 30 Hz
Pose stream - 200 Hz



Class diagram for sensors data synchronization (reading and keeping)

Point cloud alignment. Why?

Misaligned two PCs:



Transformations

${}^{W}T_{T}$ → Transformation of **T265 sensor** w.r.t **world**

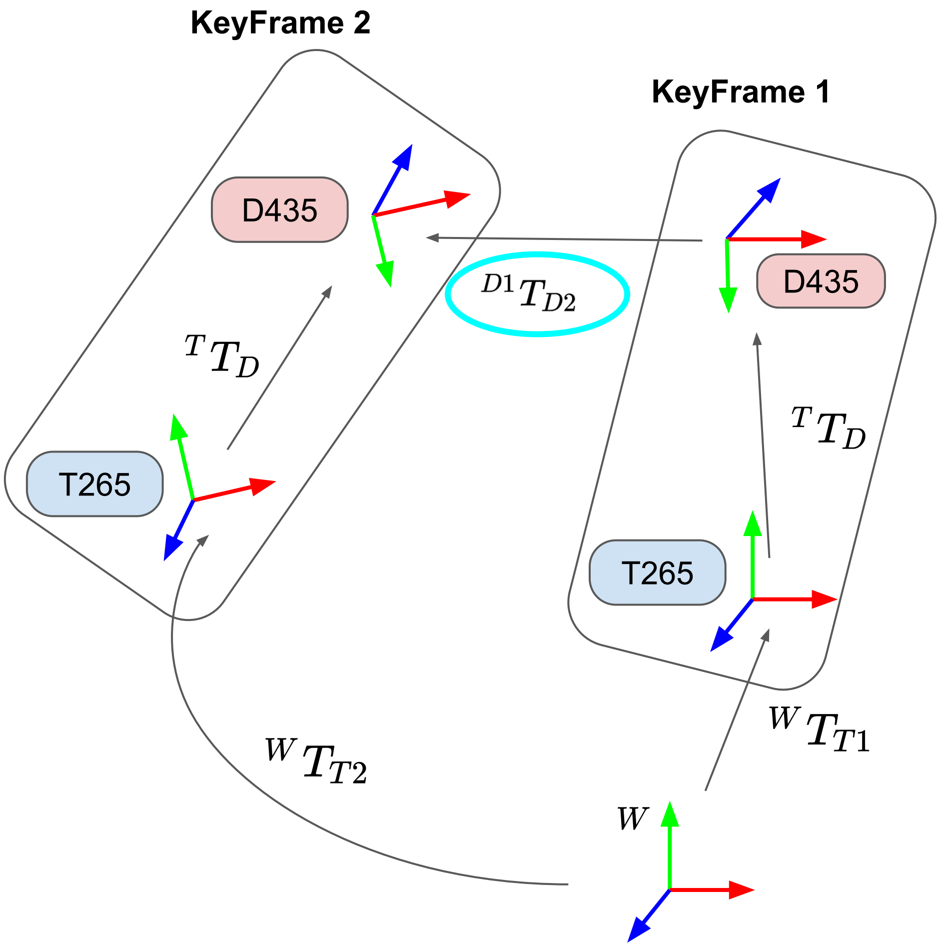
${}^{T}T_{D}$ → Transformation between sensors, **D435** w.r.t **T265**

${}^{D1}T_{D2}$ → Transformation of **KeyFrame 2** w.r.t **KeyFrame 1**

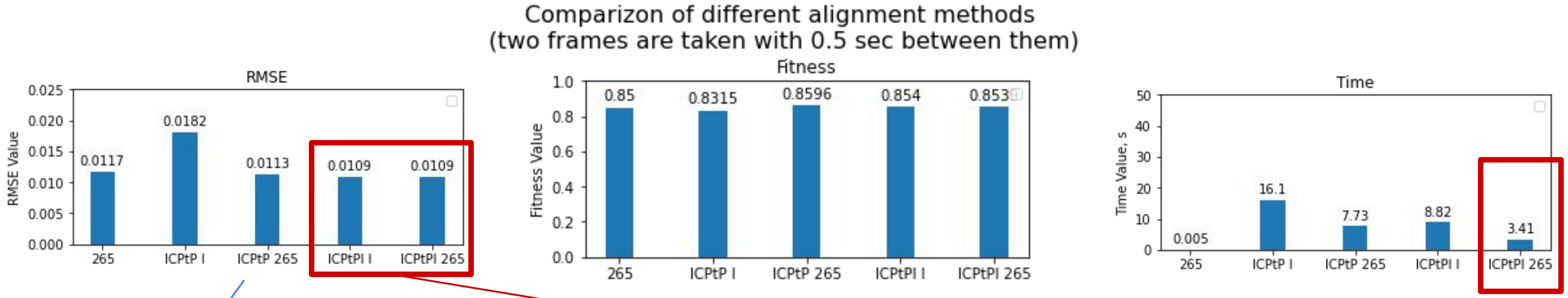
${}^{W}T_{D} = {}^{W}T_{T} \cdot {}^{T}T_{D}$ → Transformation of **D435 sensor** w.r.t **world**

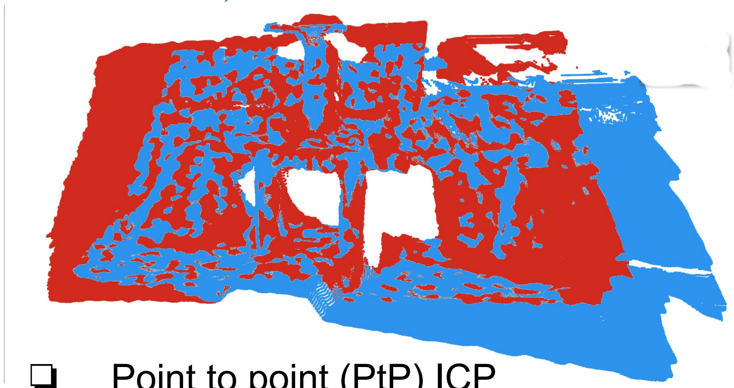
$${}^{W}T_{D2} = {}^{W}T_{D1} \cdot {}^{D1}T_{D2}$$

$$({}^{W}T_{D1})^{-1} \cdot {}^{W}T_{D2} = {}^{D1}T_{D2}$$

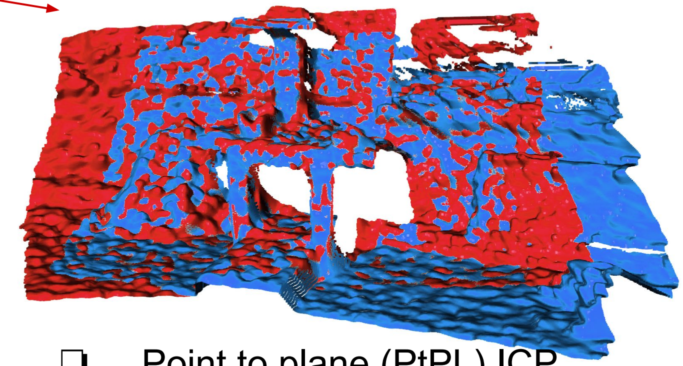


Point cloud alignment with Iterative Closest Point (ICP): Point to Point vs Point to Plane



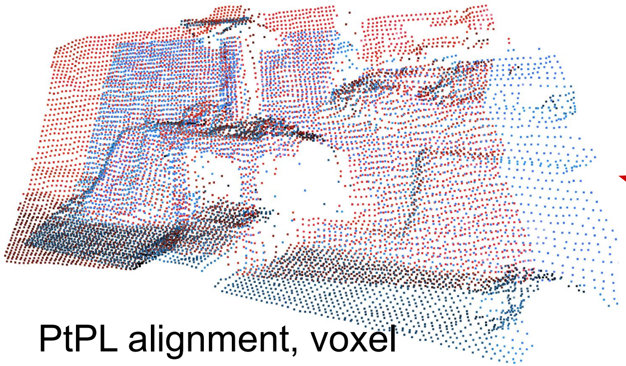


❑ Point to point (PtP) ICP

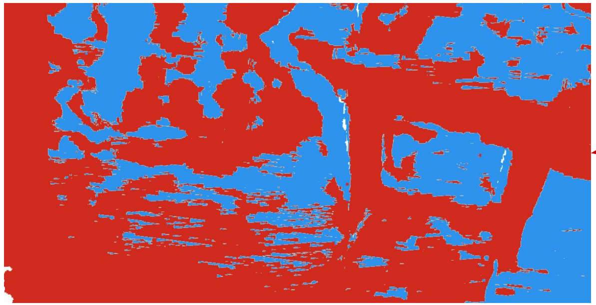


❑ Point to plane (PtPL) ICP

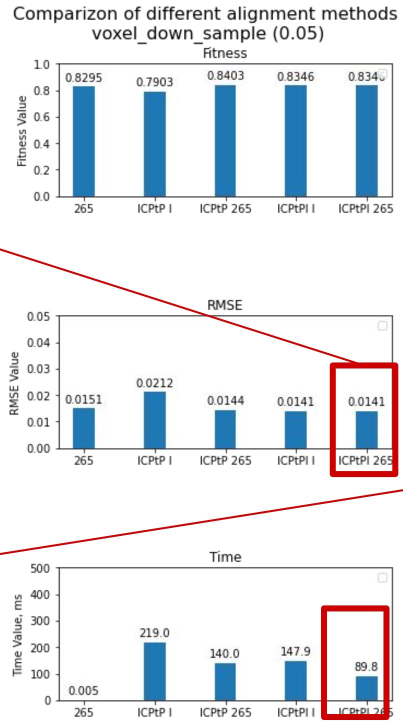
Point cloud alignment with Iterative Closest Point (ICP): Voxel sampling & Spatial filtering

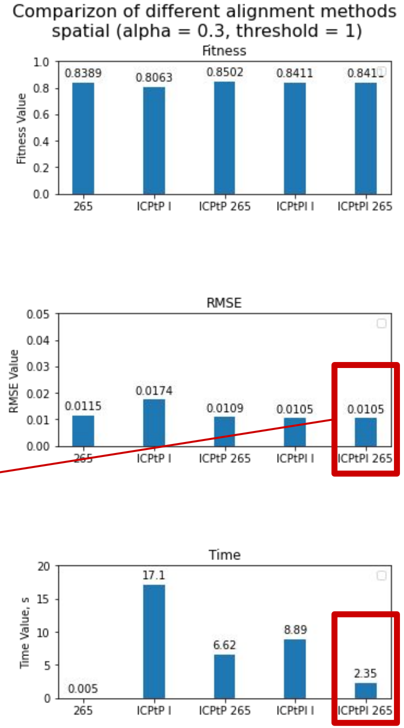


- PtPL alignment, voxel



- PtP alignment, spatial filtering [1]



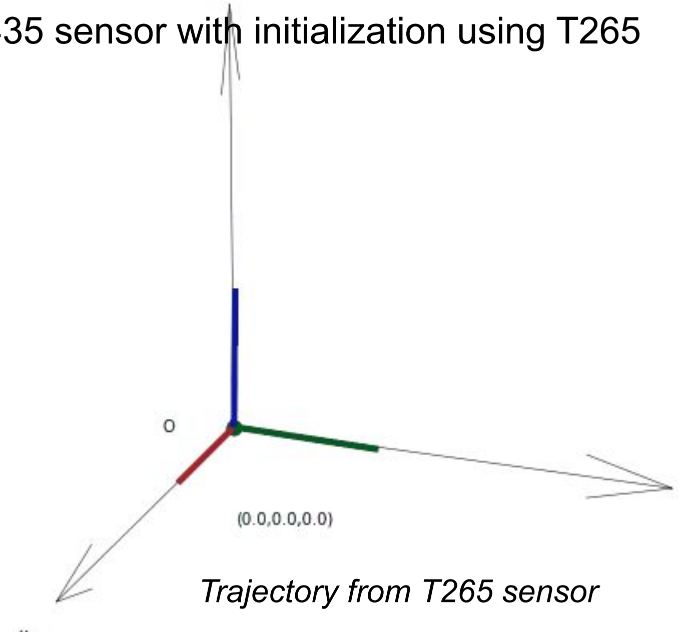


[1] Eduardo S. L. Gastal and Manuel M. Oliveira. 2011. Domain transform for edge-aware image and video processing. ACM Trans. Graph. 30, 4, Article 69 (July 2011)

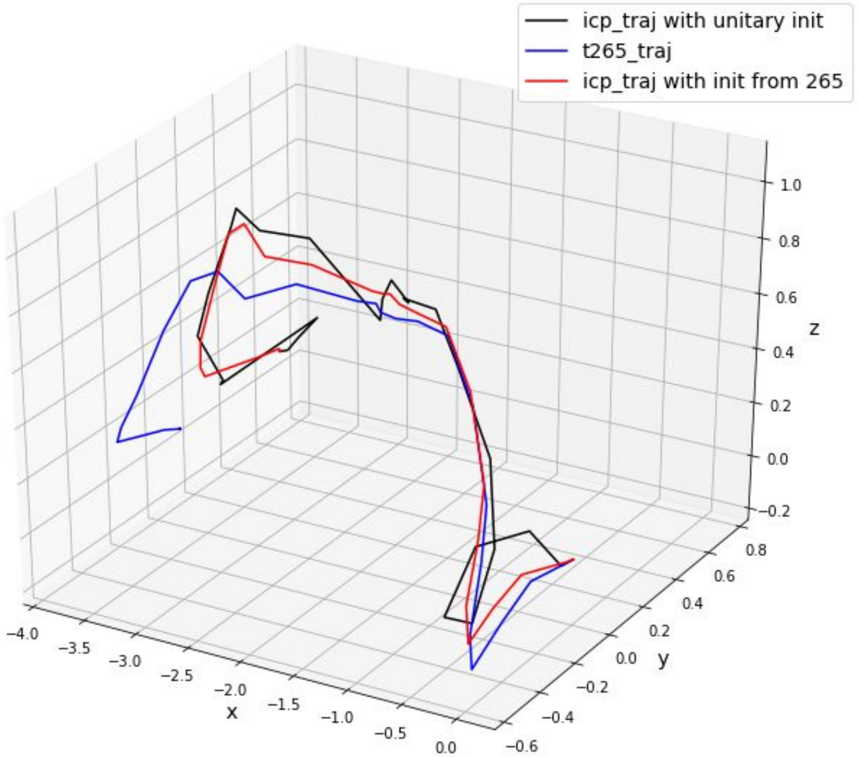
Trajectory estimation

Trajectory estimation using transformation matrices obtained from:

- T265 sensor
- T435 sensor with initialization using np.eye(4)
- T435 sensor with initialization using T265



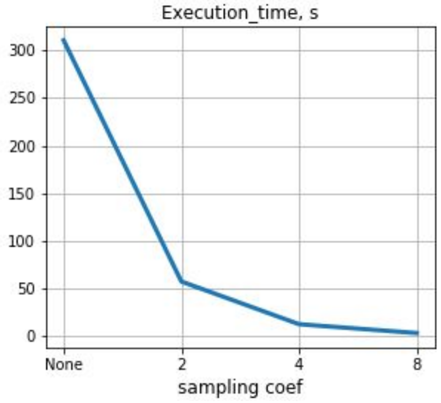
Trajectory from T265 sensor

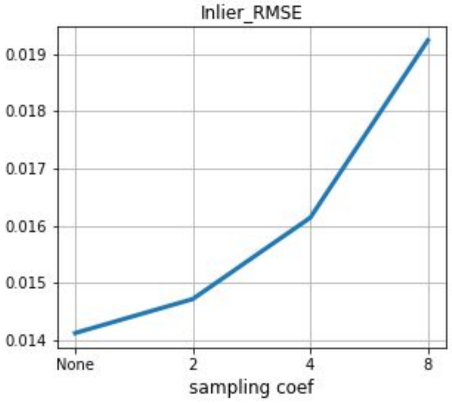


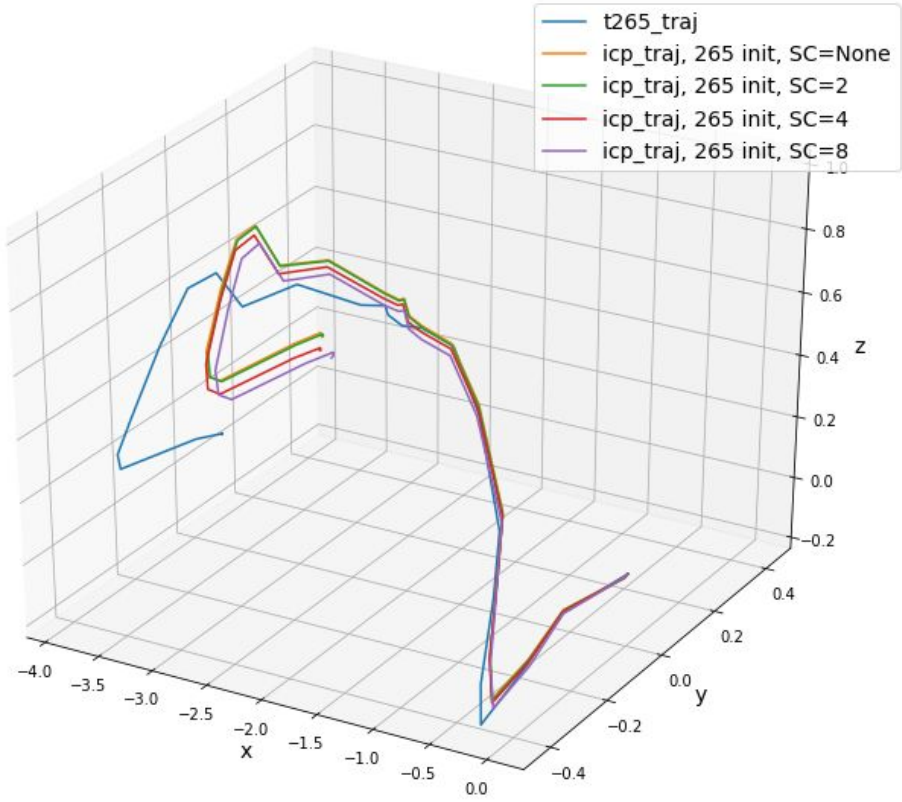
Trajectory comparison

Trajectory estimation from D435 sensor. Decimation

Comparison of the influence of **sampling coefficient** of the **decimation filter** on the trajectory estimation and ICP algorithm performance.

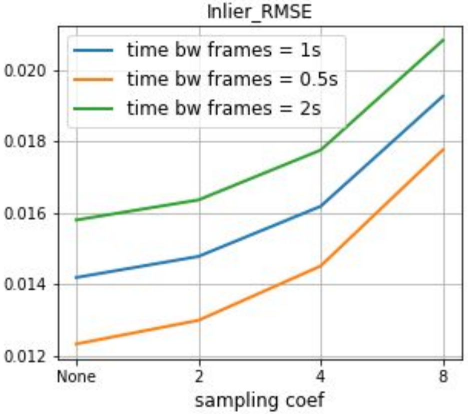


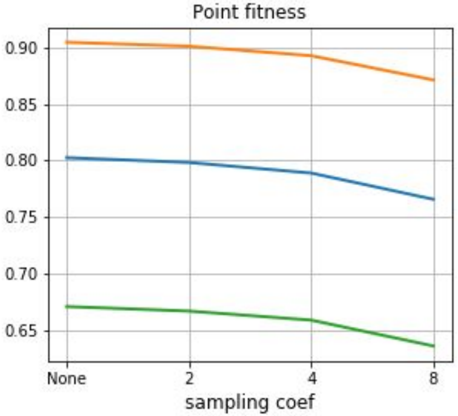


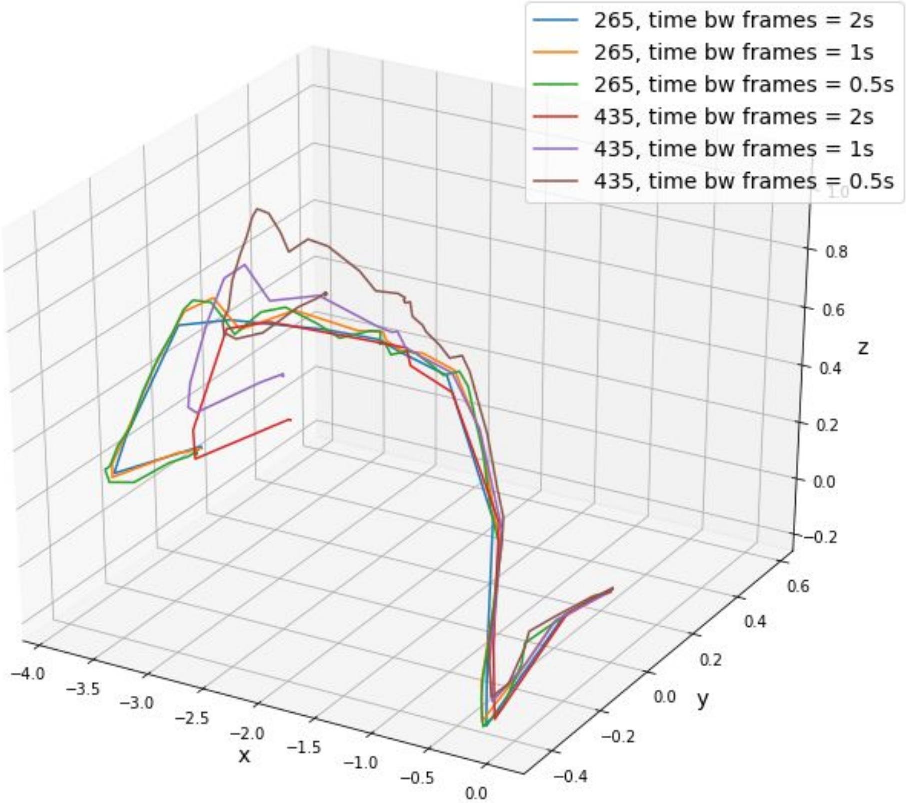


Trajectory estimation from D435 sensor
Time between frames

Influence of the time interval between frames on the trajectory estimation and the ICP algorithm execution.



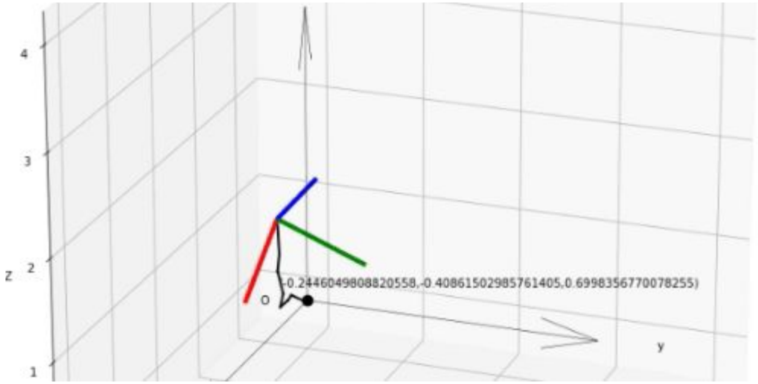


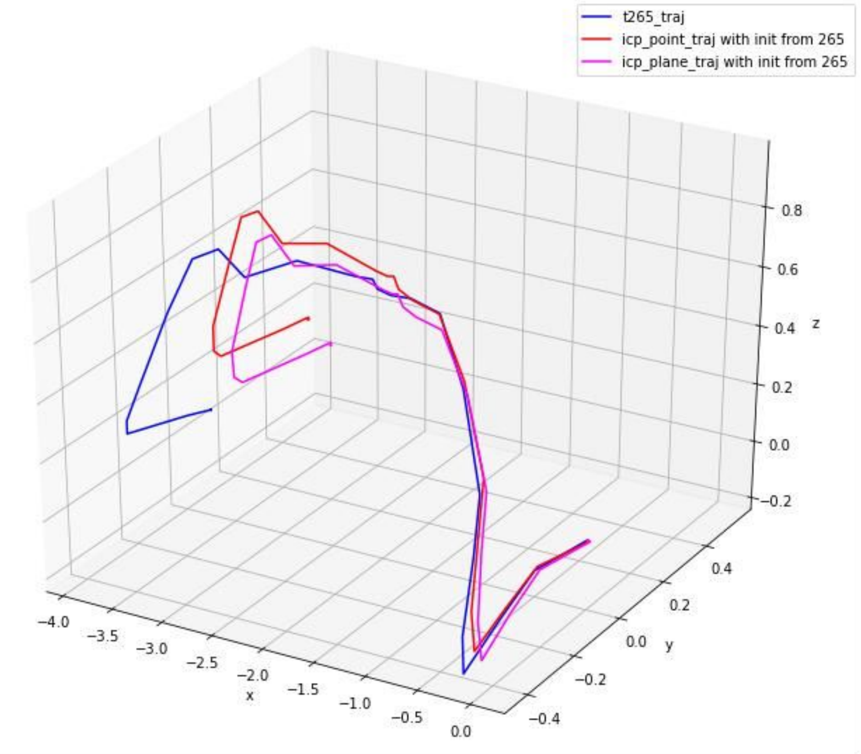


Trajectory estimation: Point vs Point and Point to Plane approaches

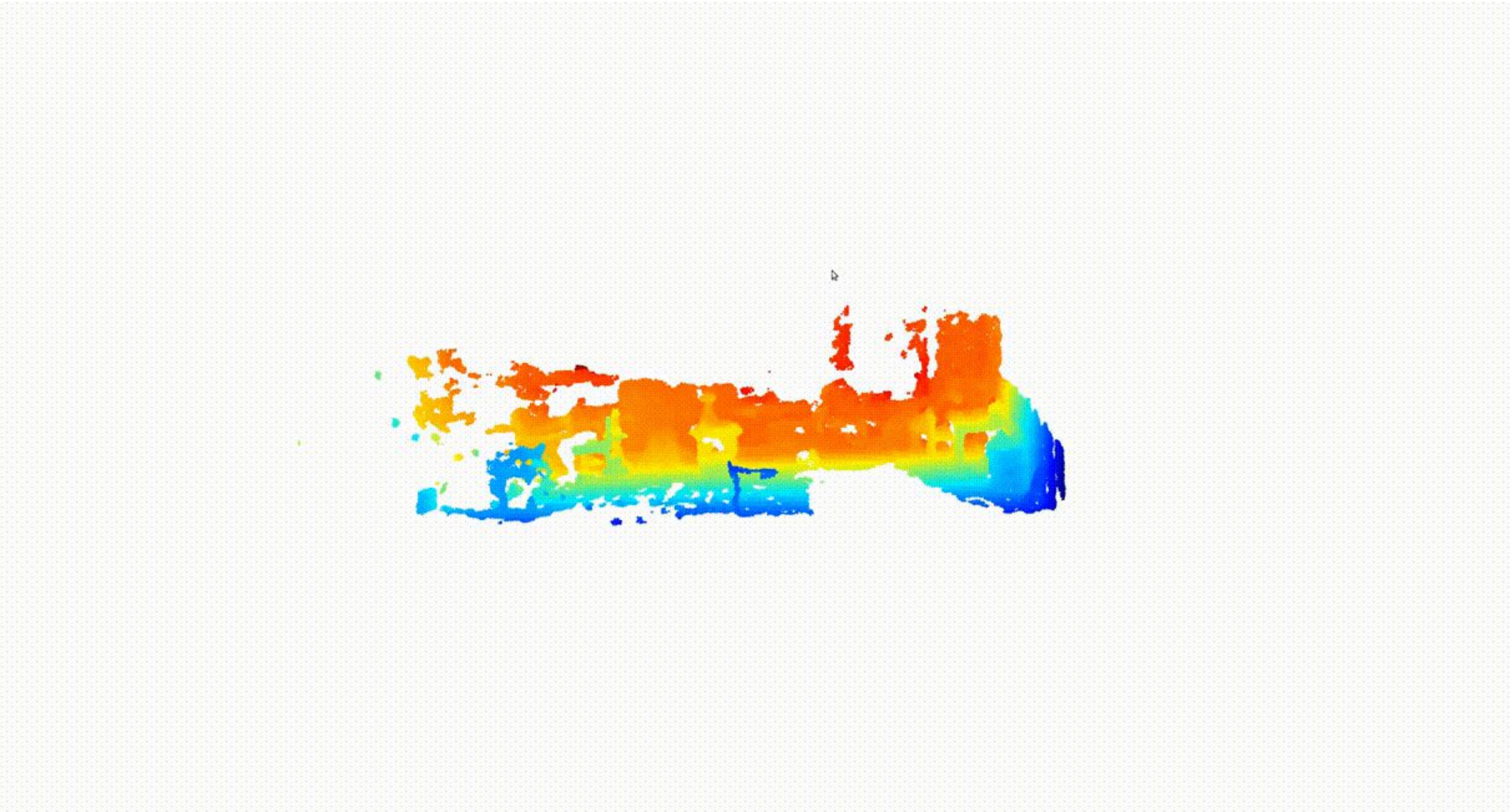
Trajectory estimation using transformation matrices obtained from:

- ❏ T265 sensor;
- ❏ T435 sensor, point to point ICP approach;
- ❏ T435 sensor, point to plane ICP approach;

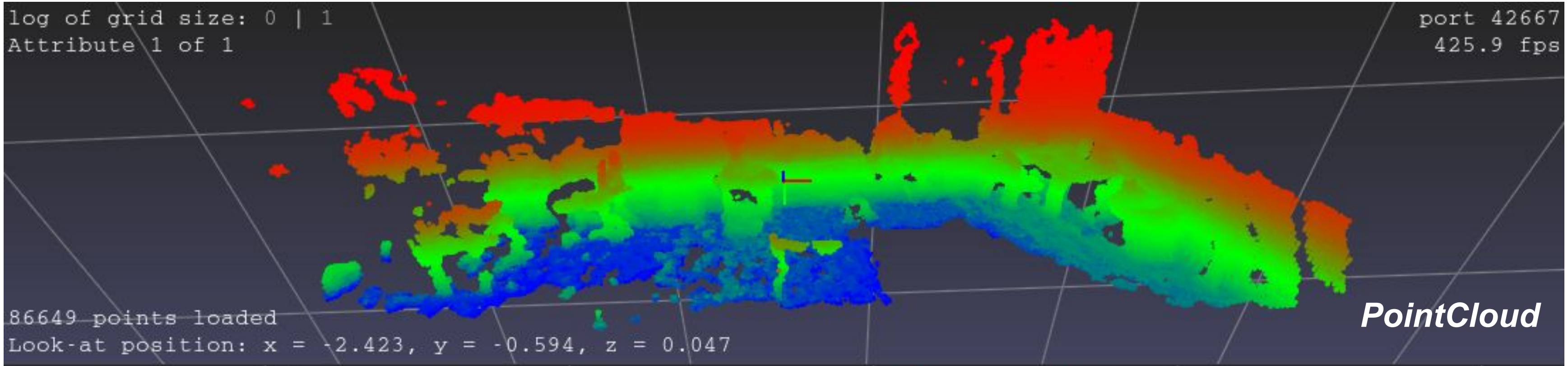


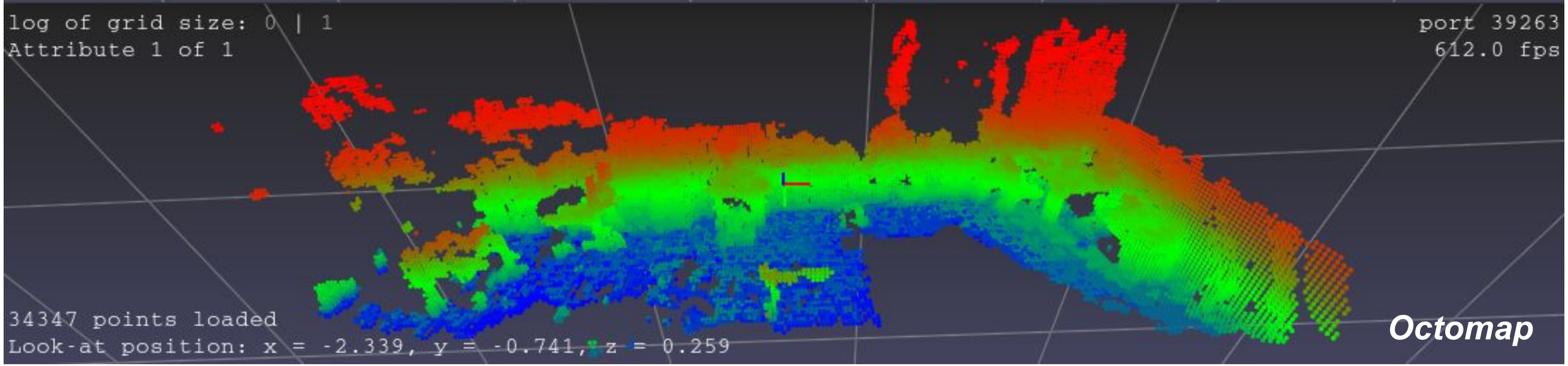


Mapping: Combined Resultant PointCloud

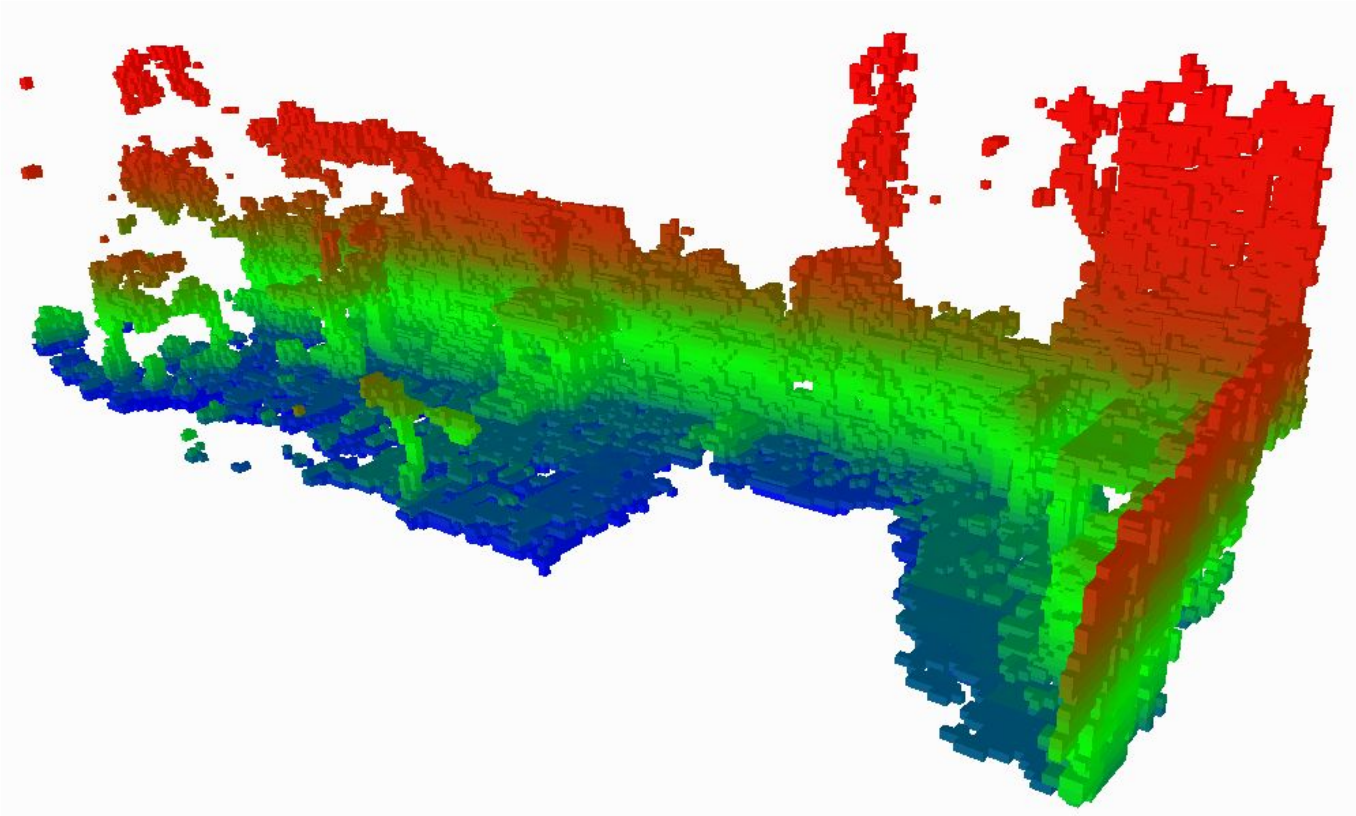


Mapping: PointCloud vs Octomap





Mapping: Octomap Voxel Representation



What we also tried to do:

1) To develop an refreshable mapping framework (visualization with voxels)
2) To launch computation in real-time

What we also wanted to do:

1) To capture ground truth with Vicon cameras (sensor in ISR Laboratory)
2) To deploy project on Jetson Nano computer (problems with some libs?)
3) To work together in Campus

Conclusion

1) Data sets from D435(depth) and T265(tracking) sensors were collected
2) Different methods for PointCloud alignment were implemented and compared
3) Some approaches for building camera trajectory were implemented and estimated
4) Different methods of data representation were included in the project
5) Combined resultant PointCloud was built
6) Resultant colored map on a base of Octomap module was created

References

1. Bayer, Jan, and Jan Faigl. "On Autonomous Spatial Exploration with Small Hexapod Walking Robot using Tracking Camera Intel RealSense T265." 2019 European Conference on Mobile Robots (ECMR). IEEE, 2019.
2. Point-to, Local Area Network Using. "Autonet: a high-speed, self-configuring local area network using point-to-point links." IEEE Journal on Selected Areas in Communications 9 (1991): 8.
3. Low, Kok-Lim. "Linear least-squares optimization for point-to-plane icp surface registration." Chapel Hill, University of North Carolina 4.10 (2004): 1-3.
4. Hornung, Armin, et al. "OctoMap: An efficient probabilistic 3D mapping framework based on octrees." Autonomous robots 34.3 (2013): 189-206.

All code from our project is presented in the repository on GitHub

Acknowledgements

Thanks to Professor and TAs Anastasia and Marsel for quick and useful responses on weekends!

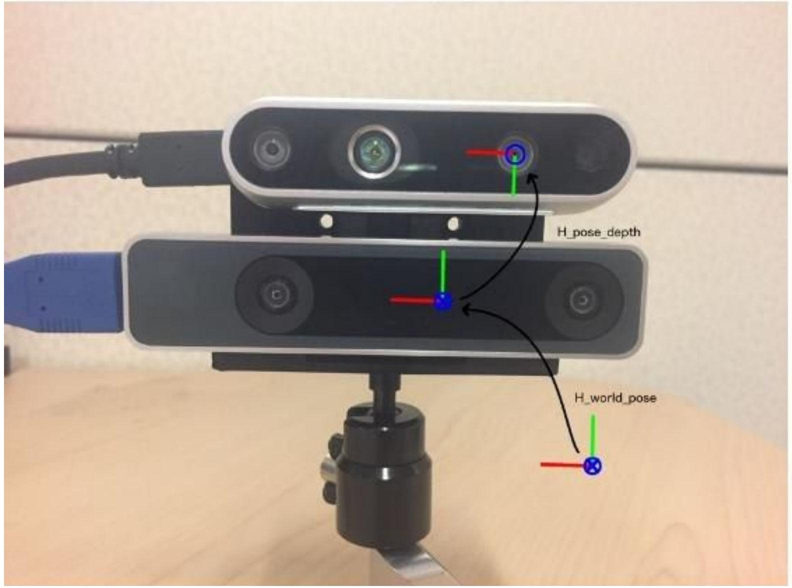
Software Python libs

1. Open3D
2. Pyrealsense
3. threading
4. NumPy
5. Matplotlib
6. OpenCV - images visualisation
7. PPTK - Point Cloud visualisation
8. Octomap
9. Pyglet, trimesh, glooey - Voxel Representation

We tried to avoid ROS installation, that's why we got many problems with packages installation

Thank you
for your attention

Camera frames



$$
{}^{T}T_{D} = \begin{pmatrix} 0.999968402 & -0.006753626 & -0.004188075 & -0.015890727 \\ -0.006685408 & -0.999848172 & 0.016093893 & 0.028273059 \\ -0.004296131 & -0.016065384 & -0.999861654 & -0.009375589 \\ 0 & 0 & 0 & 1 \end{pmatrix}
$$