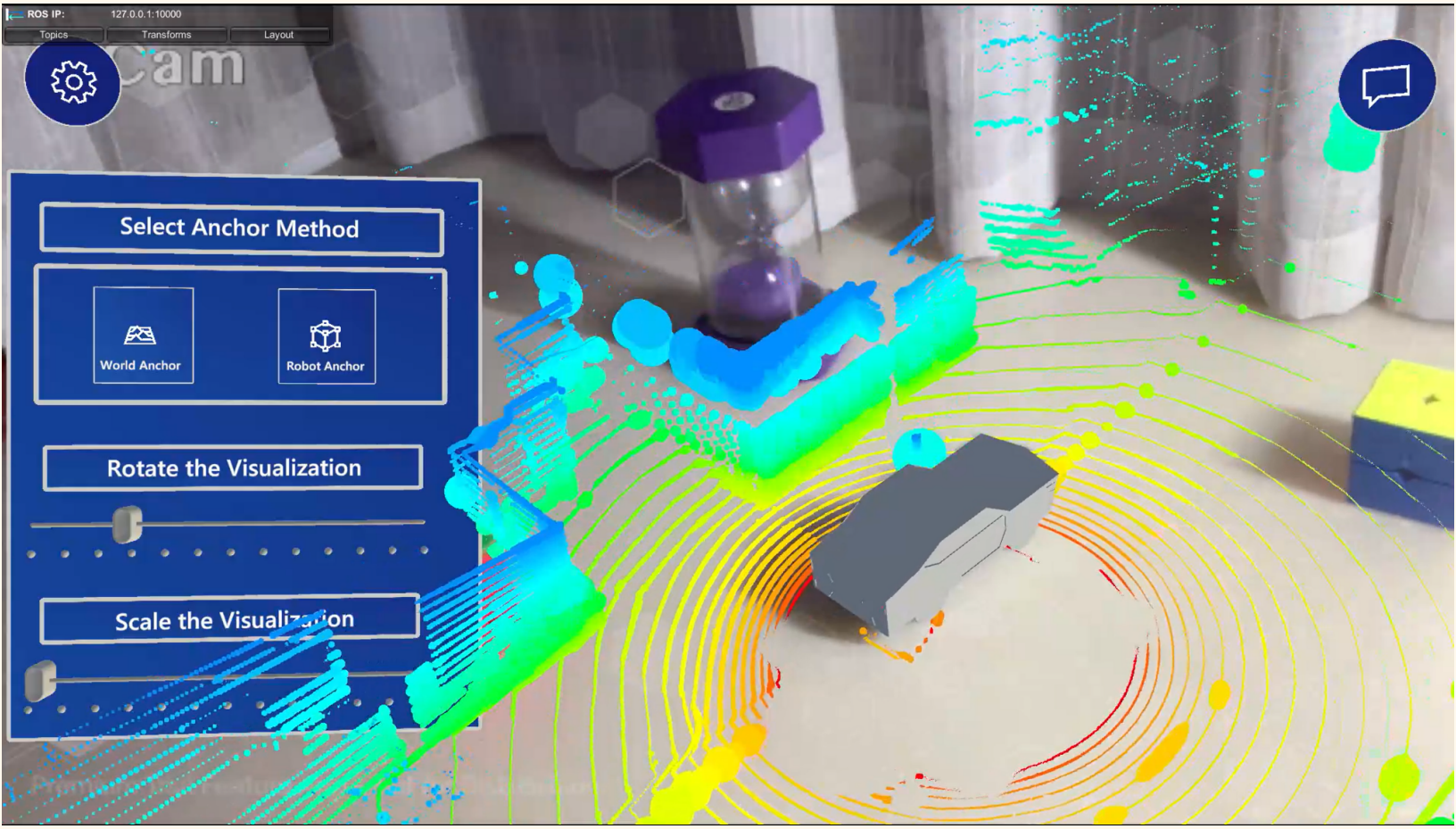
Data Visualization (COM-480 @ EPFL)

Mixed Reality Toolkit for ROS Visualization

Group Name: MRViz

Tommaso Gargiani, Linyida Zhang, Zijun Cui



Introduction

Nowadays, engineers developing robotics software have a serious constraint. They can visualize the robot's perception of the environment, but cannot have a straightforward impression of the difference between such perceived data and the real physical world. They often have to rely on uninformative visualizations of point clouds or heatmaps on a computer screen with no additional information about the robot's surroundings, making it rather difficult to find the flaws or debug the system.

Thanks to our toolkit, which overlays the robot's sensed data onto the real world, robotics engineers will benefit from seeing directly what their robot perceives by exploring the environment around them with HoloLens. Such features will simplify the development of ROS (Robot Operating System) software, allowing engineers to fully leverage sensor data by visualizing them in the mixed reality.

Path

Background

In the beginning of the class, we reached out to various labs and companies to understand their needs with a focus in data visualization. And we ended up accepting a project with Microsoft Mixed reality lab.

Dataset

ROS is a framework of nodes, i.e. processes that perform computation. Nodes communicate with each other by publishing messages to named buses called topics.

After identifying our project, we acquired the required dataset - ROS sensor data, which is the data source of topics in our final data visualization. In this project we used ROS bags, recordings of real-time message data flowing from the robot, as our source of data.

Thanks to the data logging capabilities of bags, we were able to simulate a robot's functionality even without operating one and to playback key moments.

General framework

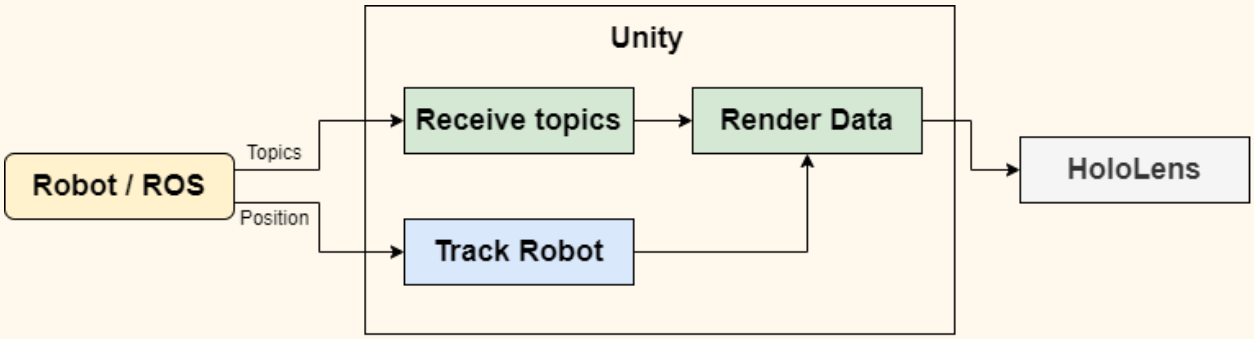


Fig.1 General framework

The components of our project are shown above. The input end (Robot/ROS) and output end (HoloLens) need to be configured first.

Robot/ROS: We set up the docker containers to run ros bag recording and set up the ros bridge to connect them to Unity. Because we were using a heavy dataset, the default example container was not flexible and many customization had to be done at that level.

HoloLens: We successfully deployed the HoloLens Emulator in VS2019 and ran it in Unity. But the difficulty here was that since we didn't have a real HoloLens device to test with, the simulation was very slow and didn't have environment inputs.

Unity

Unity is the main playground where we handle all the data, visualize different topics, and overlay the visualization to the robot accordingly.

ROS Connection: A robust connection between our existing project and ROS;

Receive Topics and **Render Data**: We set up the Unity ROS visualization package and MRTK package. MRTK is the Mixed Reality Toolkit, which can be used in cross-platform MR app development in Unity. (To get the version of all packages compatible within each other, it was even harder dealing with cross platform compatibility.)

Track Robot: We integrated the Vuforia SDK to track the robot using a 3D object tracking method.

Calibrate visualization: To make the visualization actually 'follow' the robot's position and pose, we set up two tracking methods: the world anchor method and the robot anchor method. We also implemented rotation and scaling functions for visualized results to align orientation.

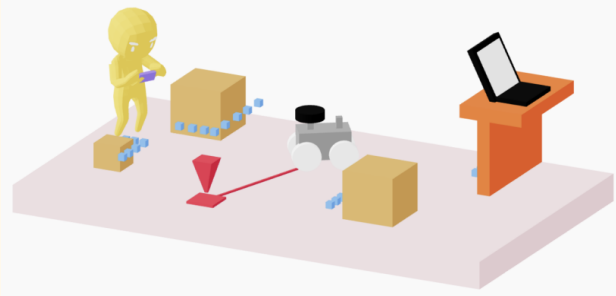


***Fig.2 World Anchor Method**: In the easiest scenario, the running odometry estimators running on the robot provide usable transform data. As the robot runs, the position of the sensors in the world frame is known. The position and orientation for every visualization is handled automatically. In this case users can manually position and orient a static anchor, and place it to the one that ROS transforms are mapping to.*

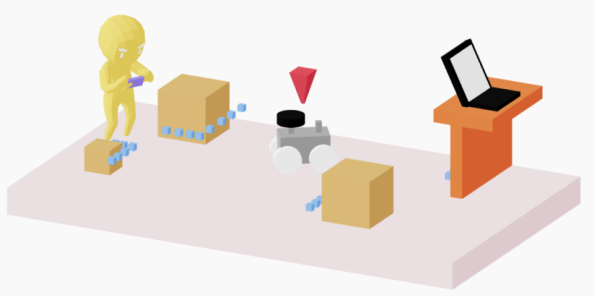


***Fig.3 Robot Anchor Method**: If the robot is not aware of its position, it would be up to us to get an estimation of the robot odometry to map it in the augmented reality scene. In this case, we track the robot position by exploiting undistorted images or 3D object tracking methods. The sensor visualisions are then transformed to the moving anchor that follows the robot*

UI design

We first sketched the UI design to group all features developed so far (see APP UI lifetime below) and optimized design choices during the process. The core UI elements can be broken into units:

Fig.4 UI design

- A scrollable topic list that lets users select topics to visualize;
- Two buttons that control the anchor methods (world anchor method and robot anchor method);
- Sliders that control the rotating and scaling of the drawn topics. The range of rotation slider is [-180°,180°], and the range of scaling slider is (0, 1];
- Multiple panels that hold all UI elements in a compact and unified way;
- As the UI is organized in 3D space, we use multiple cameras to render the UI to the front of the view so it will not be occluded by other GameObjects.

Other platforms

Android: We built and ran the app on an Android device.

Web: We designed the webpage to show our final visualization result and set up the demo-website using Next.js.

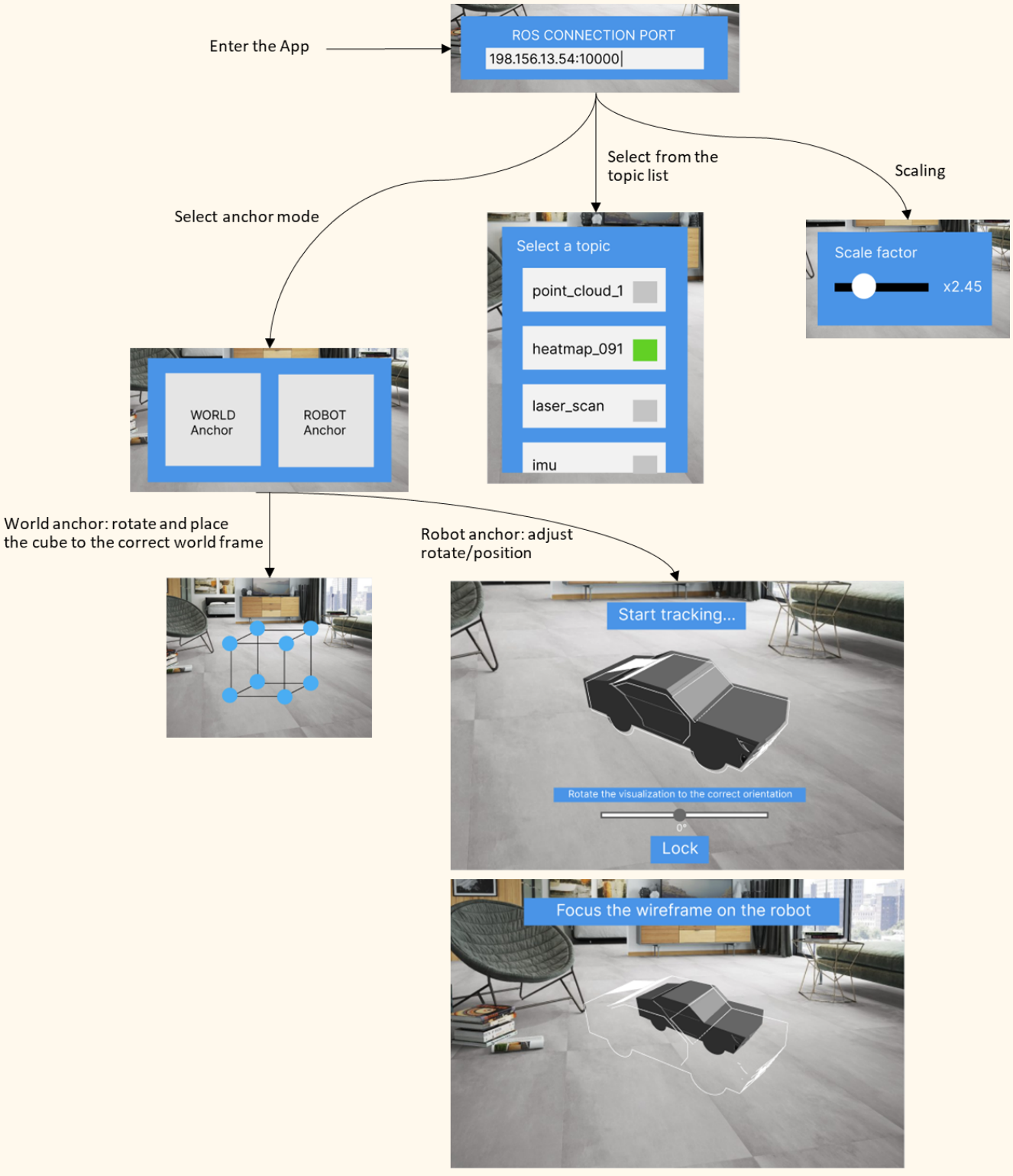


Fig.5 APP UI lifetime

Challenges and design decisions

Match robot coordinates & Scale the visualization

When we adopt the 3D object tracking method to obtain the position and pose of the robot, the captured model and the actual robot may have the same ‘up’ directions, but have different ‘forward’ directions. So we add a rotation function to allow users to adjust the visualized data to a reasonable direction once we have solved the robot's position and pose.

Another issue is that the visualization can be extremely wide because of the robot's surroundings. For example, the sensors can obtain point cloud data with a radius of ten meters. To observe the full picture and details of the data at different scales, a scaling slider is necessary.

Display interactive UI in HoloLens

The existing UI (the hud on top left of the screen) of Unity Visualizations Package (UVP) works only in Unity, but doesn't work on other platforms such as HoloLens or Android.

To solve this crucial issue, we firstly familiarized ourselves with MRTK and tried to create some simple UI such as a pressable button that can be compatible with HoloLens/Android. After understanding how the Unity Visualizations Package works, we managed to get a list of available topics with the UI we designed, and created a proper visualization of the selected topic.

Then we created a robust connection between our existing project (with MRTK UI), ROS and UVP. Based on this, we finally implemented basic functionality such as topic visualization, and expanded the UI with anchor method selection, visualization rotation and scaling.

Changes

Robot data acquisition method

In the world anchor method, the most crucial part is to locate the starting position and pose of the robot. In order to reuse functions and make the locating process more convenient, we still used 3D object tracking to get the initial position and pose of the robot, rather than letting users tap-and-place an agent and then slightly adjust it.

Platform extension

In the early stage of this project, Android was not included as a platform to run the data and show our final visualization results. But during the exploration, we managed to make packages compatible for the android system and make the UI work with an Android device. In this way, the ROS data and the visualization results can be extended to more common devices like Android phones, providing more platform choices and also making it easier to apply our toolkit to real life scenarios.

Contributions

All members worked well as a whole. During the development, we made all major decisions after discussions and helped each other based on what everyone specialized at. We specially thank Etienne Salimbeni for the interesting idea and his deep involvement.

- Tommaso Gargiani: HoloLens UI design and connection establishment between MRTK and ROS;
- Linyida Zhang: World/Robot anchor method, rotation and scaling of visualization, UI optimization;
- Zijun Cui: Website UI design and report writing.

Conclusion

In this project we developed visualization for a wide range of sensor information: heatmap/costmap, point clouds, odometry tracks, 2D laser scans, transform positions, etc. For each type of visualization we really highlighted the mapping fusion of the data to the real world, which can provide a practical solution when engineers need to see the sensed data from robots in the physical world.