Tampere University

Wearable AR-based interaction interface for HRC

Developer version

trinity ENGAGE WITH AGILE MANUFACTURING

Preparation steps

- Hardware
- Preparing Software Environment
- Setting up Hololens/Kinect
- Installing Robot drivers, connecting robot to ROS

Hardware

- The following (or similar) hardware is needed in the setup
  - Workstation
    - Intel Core i7-6700HQ
    - 16GB RAM
    - NVIDIA GeForce GTX 970M
  - Depth Sensor
    - Microsoft KinectV2 (Other sensors are applicable, check compatibility with ROS environment)
  - Microsoft Hololens
  - Collaborative robot
    - UR5

Software Environment

- Installing linux
  - Recommended linux distribution is Ubuntu 18.04, since it is compatible with ROS melodic used in the setup. Installation instructions can be followed from here https://linuxtechlab.com/step-by-step-guide-to-install-ubuntu-18-04/
- Installing ROS
  - This implementation uses ROS Melodic distribution, Installation instructions can be followed from here http://wiki.ros.org/melodic/Installation/Ubuntu. Full-desktop installation is recommended
  - To check if ROS is installed correctly run **roscore** inside command line and see if it outputs any errors
- Installing Unity
  - The project uses Unity 3D editor 2017.4.1f1 version
  - Vuforia Engine v7.5.20 is also required
- Download implementations for the Module's ROS nodes from https://github.com/Herrandy/HRC-TUNI/. The nodes should be downloaded to the src folder of your ROS workspace
- Build Module's nodes using the following commands
  - Source your default and workspace ROS environments
  - cd <workspace_folder>
  - rosdep install -r --from-paths .
  - catkin_make -DCMAKE_BUILD_TYPE="Release"

Installing Depth Sensor

- Depth sensor is installed above the workspace
- The device should be installed perpendicular to the workspace area
- The device can be installed either to the ceiling or on a beam support

Kinect Drivers

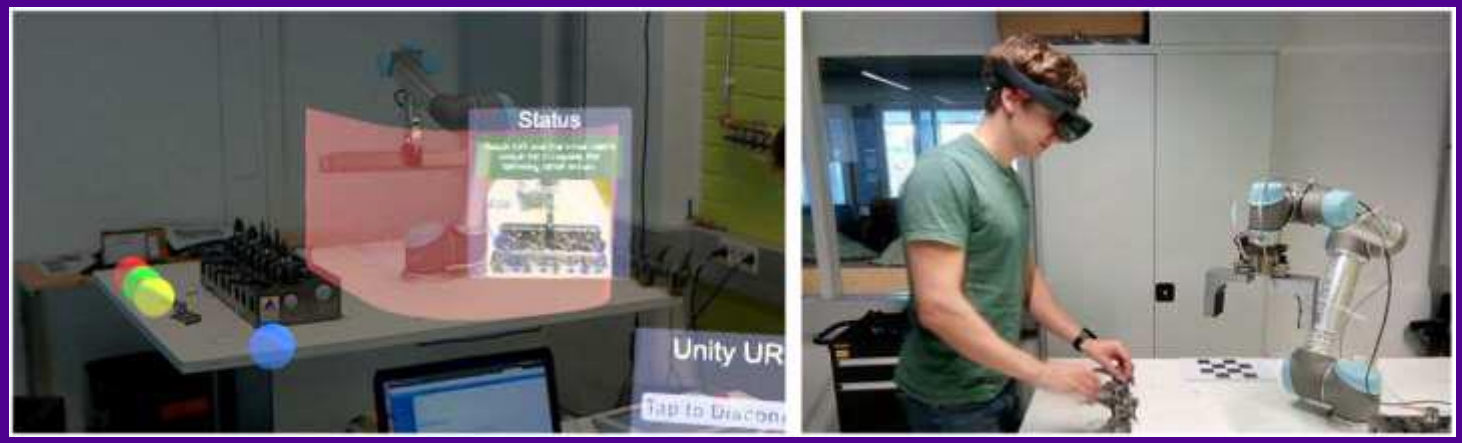
- To use Kinect, one should first install device drivers
  - Kinect V2 https://github.com/OpenKinect/libfreenect2
  - Azure Kinect https://docs.microsoft.com/en-us/azure/kinect-dk/sensor-sdk-download
  - Check if device is working correctly by running k4aviewer for azure, or provided test scripts for KinectV2
- Next, install ROS interface for Kinect
  - Kinect V2 https://github.com/code-iai/iai_kinect2
  - Azure Kinect https://github.com/microsoft/Azure_Kinect_ROS_Driver
  - The drivers provide test launch files to check whether the node publishes correct data
- For other depth sensors please check compatibility with ROS

Setting up robot

- The module is designed to work with UR5 cobot
- The base of the robot is assumed to be stationary
- Install ROS drivers for the robot
  - ROS-industrial Universal Robot https://github.com/ros-industrial/universal_robot
  - UR modern driver https://github.com/ros-industrial/ur_modern_driver
- Check the IP address of your robot
- You can check if the robot connection is working by using **ping <robot_ip>** command
- To check driver, run **roslaunch ur_modern_driver ur5_bringup.launch robot_ip:=<robot_ip>.** Check if /joint_states topic publishes joint data

AR interface

- Purpose of the module
- Safety border
- Setting up UI
- Setting additional Parameters
- Testing Border
- Testing UI



Purpose of the module

- This module provides a visualization for the depth-based safety system and the UI using a Hololens+depth sensor setup
- The safety area is visualized as a virtual wall in hololens around the robot. The border is generated dynamically around the robot during its movement
- The module also visualizes UI elements that can provide instructions for the operator and can be interacted with touch controls

Safety border

- For visualization of the safety border, the ‘Depth-sensor safety model for HRC’ module should be deployed
- To install the module, please check the corresponding tutorial

Setting up UI

- It is assumed that the steps for safety module were performed, since some parameters are shared between modules
- The Unity scene for the demo should be imported
  - Create empty Unity Project
  - Open Unity scene: File → Open Scene → <path_to_package>\HoloRobo\Assets\Scenes\Scene.unity
- Vuforia provides tools for calibrating between the hololens and a pre-specified marker. After the calibration is done, measure the distance between the marker and the robot base and update the parameter in the code

Setting additional parameters

- Some parameters may need to be adjusted for module to work properly
- The parameters are stored inside **unity_msgs/configs/config.yaml** file
  - interaction_button_thres – the threshold in depth that defines whether the button was interacted with

Testing safety border visualization

- Run your test program for the robot
- Run the following commands in separate consoles
  - **roscore**
  - **roslaunch ur_modern_driver ur5_bringup.launch robot_ip:=<real robot 192.168.125.100 or simulated 127.0.0.1>**
  - **roslaunch kinect2_bridge kinect2_bridge.launch max_depth:=2.0 publish_tf:=true**
  - **roslaunch safety_model detect.launch safety_map_scale:=100 cluster_tolerance:=0.005 min_cluster_size:=200 anomalies_threshold:=20 cloud_diff_threshold:=0.02 viz:=false**
  - Build the Unity project, target build - hololens
  - Upload to Hololens
  - Start the application from the Hololens application list.
  - Establish the wireless communication channel between the computer and Hololens: rosrun ar tcp_server.py
- You should see red wall around the robot

Testing UI

- Run same commands as in previous step
- When putting your hand over ‘start’ and ‘stop’ buttons, the UI should change
- Putting your hands over ‘stop’ and ‘dead man switch’ buttons should stop the robot
- Putting your hands over ‘start’ and ‘dead man switch’ buttons should start robot again
- **dashboard_client.py** console should output what buttons are being interacted with.
- If buttons do not respond, try positioning your hand at higher locations over the button or changing **interaction_button_thres** to lower values

Config files

- List of important configuration files for the modules
  - **/calibration/launch/tf_broadcast.launch** –transformation from Kinect to robot
  - **/unity_msgs/configs/config.yaml** – parameters for the safety and UI modules
  - **/unity_msgs/configs/mobile_demo/projector_buttons.yaml** – locations of the UI elements

Maintenance and troubleshooting

- Here we give general recommendations for troubleshooting, for more specific problems please email to dmitrii.monakhov@tuni.fi, we will help
- General tips for troubleshooting
  - Always check if you sourced ROS environment on any new console tab before running commands. If you are using a single ROS environment, it may be easier to add source commands to **.bashrc** so it is done automatically
  - Check output of the scripts in the console, as it can give hints for errors
  - Remember that Time-of-flight depth sensors have a warm-up period during which temperature of the sensor and processed values can drift. This can cause very unpredictable and hard to catch errors, so we recommend running the sensor for 30-60 minutes before the test
  - Don’t forget to rebuild the modules after making changes in config/source file

Tampere University

Thank You

Dmitrii Monakhov
Dmitrii.Monakhov@tuni.fi

trinity ENGAGE WITH AGILE MANUFACTURING